10.1071/WR17151_AC

Supplementary Material: *Wildlife Research*, 2019, 46(1), 76-88.

Incorporating movement patterns to discern habitat selection: black bears as a case study

Dana L. Karelus[A,E,*], *J. Walter McCown*[B], *Brian K. Scheick*[B], *Madelon van de Kerk*[C], *Benjamin M. Bolker*[D] *and Madan K. Oli*[A]

[A]Department of Wildlife Ecology and Conservation, and School of Natural Resources and Environment, 110 Newins-Ziegler Hall, University of Florida, Gainesville, FL 32611, USA.

[B]Florida Fish and Wildlife Conservation Commission, 1105 SW Williston Road, Gainesville, FL 32601, USA.

[C]School of Environmental and Forest Sciences, 114 Winkenwerder, University of Washington, Seattle, WA 98195, USA.

[D]Departments of Mathematics & Statistics and Biology, McMaster University, 314 Hamilton Hall, Hamilton, Ontario L8S 4K1, Canada.

[E]Corresponding author. Email: dana.karelus@sulross.edu

*Dana L. Karelus is now affiliated with Borderlands Research Institute, Sul Ross State University, C-21, Alpine, TX 79832, USA.

A mistake in the supplementary code was brought to the authors' attention whereby the turning angle for creating available/unused steps was incorrectly calculated based on the current used step instead of the previous used step. This resulted in minor changes to the results from the mixed conditional logistic regression models; however, the overall conclusions of the study remain unchanged. Here, we have corrected and replaced the supplementary code file.

Supplementary Material – R code

```
#Supplementary Material - R code
# Karelus, D. L., McCown, J. W., Scheick, B. K., van de Kerk, M.,
# Bolker, B. M., and Oli, M. K. (2019). Incorporating movement
# patterns to discern habitat selection: black bears as a case study. Wildlife Research


library(moveHMM)
library(CircStats)

#Input data should include the following variables:
# ID = unique individual ID
# x = x UTM coordinate of animal location
# y = y UTM coordinate of animal location
# DateTime = date and time of animal location, in POSIXct format
# hour.x = Hour of location calculated with sin(2*pi*Hour/24)
# hour.y = Hour of location calculated with cos(2*pi*Hour/24)

#Optional variables:
# Sex
# Season


input_file <- "data/data.csv"
input_data <- read.csv(input_file)



## Prepare the data; i.e. compute the steps and angles from the locations
prep_data <- prepData(input_data,type="UTM")

#Set any instances of 0 step length to a very small number to avoid computational problems
prep_data$step[prep_data$step == 0] = 0.1



#Set starting parameters for step length (using Gamma distribution)
#Values for starting parameters need to be chosen based on your data.
#For more information on selecting starting values and choosing distributions,
#see https://cran.r-project.org/package=moveHMM/vignettes/moveHMM-guide.pdf

mu3state <- c(9, 17, 404) # step mean (3 parameters: one for each state)
sigma3state <- c(5, 12, 400) # step standard deviation
stepPar3state <- c(mu3state, sigma3state)

#Set starting parameters for turning angle (using Wrapped Cauchy distribution)
```

```
#Values for starting parameters need to be chosen based on your data

angleMean3state <- c(pi, pi, 0) # angle mean (3 parameters: one for each state)
kappa3state <- c(0.7, 0.9,  0.4) # angle concentration
anglePar3state <- c(angleMean3state,kappa3state)



#Run 3 state HMM with covariate of hour of day,
#Gamma distribution for step-length (default), and wrapped Cauchy for turning angle
#Distributions should be chosen based on your data.
# See https://cran.r-project.org/package=moveHMM/vignettes/moveHMM-guide.pdf

HMM_3state_hour = fitHMM(data=prep_data, nbStates=3,
            stepPar0=stepPar3state,
            anglePar0=anglePar3state, angleDist = "wrpcauchy",
            formula=~hour.x+hour.y)



#Add states and parameters to data using the Viterbi algorithm
HMMstates <- viterbi(HMM_3state_hour)


# The above code was adapted from the
# Guide to using moveHMM - The R Project for Statistical Computing
# see: https://cran.r-project.org/package=moveHMM/vignettes/moveHMM-guide.pdf
################################################################################


#Add state to the data
prep_data$state = HMMstates


#Add step mean, SD, angle mean, and angle conc
# for each state to the prep_data dataframe
prep_data$StepMean = NA
prep_data$StepMean[prep_data$state == 1] = HMM_3state_hour$mle$stepPar[1]
prep_data$StepMean[prep_data$state == 2] = HMM_3state_hour$mle$stepPar[3]
prep_data$StepMean[prep_data$state == 3] = HMM_3state_hour$mle$stepPar[5]

prep_data$StepSD = NA
prep_data$StepSD[prep_data$state == 1] = HMM_3state_hour$mle$stepPar[2]
prep_data$StepSD[prep_data$state == 2] = HMM_3state_hour$mle$stepPar[4]
prep_data$StepSD[prep_data$state == 3] = HMM_3state_hour$mle$stepPar[6]

prep_data$AngMean = NA
prep_data$AngMean[prep_data$state == 1] = HMM_3state_hour$mle$anglePar[1]
```

```
prep_data$AngMean[prep_data$state == 2] = HMM_3state_hour$mle$anglePar[3]
prep_data$AngMean[prep_data$state == 3] = HMM_3state_hour$mle$anglePar[5]

prep_data$AngConc = NA
prep_data$AngConc[prep_data$state == 1] = HMM_3state_hour$mle$anglePar[2]
prep_data$AngConc[prep_data$state == 2] = HMM_3state_hour$mle$anglePar[4]
prep_data$AngConc[prep_data$state == 3] = HMM_3state_hour$mle$anglePar[6]



#Add columns for "previous" step-length and turning angle (defined "current" step,
# step length and turning angle as calculated by moveHMM correspond to the "next" step
# therefore, calculated step-lengths, turning angles, and states are "forward-looking"
# but we need to know the parameters of the step that got the animal to
# the current location for step-selection "backward-looking")
prep_data$prevStep = NA
prep_data$prevAngle = NA


#Add columns for "previous" absolute angle (angle that made the current step)
prep_data$prevAA = NA


#Add column for previous GPS location
prep_data$prevX = NA
prep_data$prevY = NA


#Add columns for previous state (see note above for previous step and previous angle)
# and state parameters
prep_data$prevState = NA

prep_data$prevAngMean = NA
prep_data$prevAngConc = NA

prep_data$prevStepMean = NA
prep_data$prevStepSD = NA


#Make a vector of the unique individuals
unique_ID = unique(prep_data$ID)


#Loop through individuals to get the previous angles, states, absolute angles etc
for (i in 1:length(unique_ID)) {
  u.i = unique_ID[i]
  b.i = prep_data[prep_data$ID == u.i,]
```

```
  ############################################################
  #The following commented-out code was used previously but was in error,
  # This calculated the absolute angle of the "current" step rather than the angle of the previous step
  # for(j in 2:nrow(b.i)){
  #   b.i$prevX[j] = b.i$x[j-1]
  #   b.i$prevY[j] = b.i$y[j-1]
  #
  #   #Calculate absolute angle
  #   dist <- sqrt((b.i$x[j] - b.i$prevX[j])^2 + (b.i$y[j] - b.i$prevY[j])^2)
  #   dx <- b.i$x[j] - b.i$prevX[j]
  #   dy <- b.i$y[j] - b.i$prevY[j]
  #   b.i$prevAA[j] <- ifelse(dist < 1e-07, NA, atan2(dy, dx))
  ############################################################

  #This is the corrected code for the above commented-out section
  for(j in 3:nrow(b.i)){
    b.i$prevX[j] = b.i$x[j-1]
    b.i$prevY[j] = b.i$y[j-1]

    #Calculate absolute angle (adapted from adehabitatLT)
    dist <- sqrt((b.i$x[j-1] - b.i$x[j-2])^2 + (b.i$y[j-1] - b.i$y[j-2])^2)
    dx <- b.i$x[j-1] - b.i$x[j-2]
    dy <- b.i$y[j-1] - b.i$y[j-2]
    b.i$prevAA[j] <- ifelse(dist < 1e-07, NA, atan2(dy, dx))
    #End of corrections for this section

    b.i$prevStep[j]  = b.i$step[j-1]
    b.i$prevAngle[j] = b.i$angle[j-1]

    b.i$prevState[j]  = b.i$state[j-1]
    b.i$prevAngMean[j]  = b.i$AngMean[j-1]
    b.i$prevAngConc[j]  = b.i$AngConc[j-1]
    b.i$prevStepMean[j] = b.i$StepMean[j-1]
    b.i$prevStepSD[j] = b.i$StepSD[j-1]

  }

  prep_data[prep_data$ID == u.i,] = b.i
}


#Remove rows with NAs for current or previous location
# and/or previous abs angle due to missed fixes
completeFun <- function(data, desiredCols) {
  completeVec <- complete.cases(data[, desiredCols])
  return(data[completeVec, ])
}
```

```
prepd2 = completeFun(prep_data, c("x", "prevX", "prevAA"))



#Prepare to create available/unused locations

#Add column for step ID (same as row ID)
prepd2$StepID = row.names(prepd2)


#Add "used" column, set equal to 1 for used locations
prepd2$Used = 1


#set the number of randomly available points for each used location
n = 6


set.seed(3)

#Create a blank dataframe
prepd3 = data.frame(matrix(ncol = 19, nrow=0))

endrows = nrow(prepd2)+n*nrow(prepd2)

my.list <- vector("list", endrows)


#Loop through locations to create the available/unused locations
for(i in 1:nrow(prepd2)){

  #Get parameters
  A.mean.i = prepd2$prevAngMean[i]
  A.conc.i = prepd2$prevAngConc[i]

  S.mean.i = prepd2$prevStepMean[i]
  S.SD.i   = prepd2$prevStepSD[i]

  #Get new headings from von mises distribution and params
  angs.i = rwrpcauchy(n, location = A.mean.i, rho = A.conc.i)

  #Add current heading to each of the new angles to get new headings
  abs.ang.i = prepd2$prevAA[i]

  headings.i = angs.i + abs.ang.i
```

```
#Now get distances
#calculate shape and rate for gamma dist
shape.i = S.mean.i^2/S.SD.i^2

rate.i  = S.mean.i/S.SD.i^2

#get distances from gamma dist
dists.i = rgamma(n, shape = shape.i, rate = rate.i)

#Get current x and y to calculate available x and y
locx.i = prepd2$prevX[i]
locy.i = prepd2$prevY[i]

#cos(theta)=adjacent/hypotenuse
availx.i = locx.i  + dists.i*(cos(headings.i))

#sin(theta)=opposite/hypotenuse
availy.i = locy.i  + dists.i*(sin(headings.i))


#Put all together

############################################################
#Previously, the line below added the incorrect angle to the dataframe, although this had no
#consequence on further code
# avails.i = cbind.data.frame(headings.i, dists.i, availx.i, availy.i)
############################################################

#This is the corrected line
avails.i = cbind.data.frame(angs.i, dists.i, availx.i, availy.i)

colnames(avails.i) <- c("prevAngle", "prevStep", "x", "y")

#Add used column, set equal to 0 for available/unused locations
avails.i$Used = 0

#Add datetime
avails.i$DateTime = rep(prepd2$DateTime[i], n)

#Add animal ID and sex and season (if relevant for your analyses)
avails.i$ID = rep(prepd2$ID[i], n)
avails.i$Sex = rep(prepd2$Sex[i], n)  #If you are including sex as a variable
avails.i$Season = rep(prepd2$Season[i], n) #If you are including season as a variable

avails.i$prevAA = rep(prepd2$prevAA[i], n)
avails.i$prevX = rep(prepd2$prevX[i], n)
avails.i$prevY = rep(prepd2$prevY[i], n)
```

```
  avails.i$prevState = rep(prepd2$prevState[i], n)
  avails.i$prevAngMean = rep(prepd2$prevAngMean[i], n)
  avails.i$prevAngConc = rep(prepd2$prevAngConc[i], n)
  avails.i$prevStepMean = rep(prepd2$prevStepMean[i], n)
  avails.i$prevStepSD = rep(prepd2$prevStepSD[i], n)


  #Add StepID
  avails.i$StepID = rep(prepd2$StepID[i], n)

  #Get used data
  prepd3.i = prepd2[i,]

  #Put them together
  prepd32.i = prepd3.i[, which(names(prepd3.i) %in% (names(avails.i)))]


  prepd3.i = rbind(prepd32.i, avails.i)


  my.list[[i]] <- prepd3.i

}

colnames(prepd3) = colnames(prepd3.i)
prepd3 <- rbind(prepd3, do.call(rbind, my.list))

prepd3 = data.frame(prepd3)


#At this point, you need to obtain environmental variables for all used/unused locations that you wish to
#test
#Our data including environmental variables is available for download at:
# http://ufdc.ufl.edu/IR00010695/00001


#In our case, we tested
#land cover types, distance to major roads, distance to minor roads, and distance to creeks
# hab + rds1to2km + rds3to4km + creekkm
#Once you have added your environmental variables, we are ready to run Two Step Conditional Logisitic
#regression


#prepd3_withCovariates = prepd3 with environmental covariates

library(TwoStepCLogit)
```

```
#Model using overall data
tscl1 <- Ts.estim(formula = Used ~ hab + rds1to2km + rds3to4km + creekkm +
            strata(StepID) + cluster(ID),
          data = prepd3_withCovariates,
          random = ~hab + rds1to2km + rds3to4km + creekkm,
          all.m.1 = TRUE,
          D = "UN")


#By season, if desired (seasons are factors, set equal to 1, 2, or 3)
winter = prepd3_withCovariates[prepd3_withCovariates$Season == 1,]
summer = prepd3_withCovariates[prepd3_withCovariates$Season == 2,]
fall   = prepd3_withCovariates[prepd3_withCovariates$Season == 3,]


#By state
AllState1 = prepd3_withCovariates[prepd3_withCovariates$prevState == 1,]
AllState2 = prepd3_withCovariates[prepd3_withCovariates$prevState == 2,]
AllState3 = prepd3_withCovariates[prepd3_withCovariates$prevState == 3,]

#Run models for each set of seasonal data
#Example, summer
tscl.summer <- Ts.estim(formula = Used ~ hab + rds1to2km + rds3to4km + creekkm +
            strata(StepID) + cluster(ID),
          data = summer,
          random = ~hab + rds1to2km + rds3to4km + creekkm,
          all.m.1 = TRUE,
          D = "UN")

tscl.state2 <- Ts.estim(formula = Used ~ hab + rds1to2km + rds3to4km + creekkm +
                  strata(StepID) + cluster(ID),
                data = AllState2,
                random = ~hab + rds1to2km + rds3to4km + creekkm,
                all.m.1 = TRUE,
                D = "UN")
```