Example of our problem

Luke Wilde

7/13/2021

set up

```
#clean environment
rm(list=ls(all=TRUE))

#load constants and functions
source("C:/Users/14064/Desktop/Local Loc/nestAutoEval/Functions/standard_functions.R")
source("C:/Users/14064/Desktop/Local Loc/nestAutoEval/Functions/format_functions.R")
source("C:/Users/14064/Desktop/Local Loc/nestAutoEval/Constants/file_locations.R")

#load in packages
setUp(c("randomForest",
        "m2b",
        "moveHMM",
        "momentuHMM",
        "dplyr",
        "tidyverse",
        "caret",
        "mlbench",
        "nestR",
        "coda",
        "jagsUI",
        "R2jags",
        "runjags",
        "rjags",
        "tidymodels"))

#RF generated predictions
#predictions <- load("./predictions/RF.Rda")

#on my local machine
load("C:/Users/14064/Desktop/Local Loc/nestAutoEval/predictions/RF.Rda")
names(predictions)[3] <- "b"

predictions$b <- as.numeric(predictions$b)

#### 8. run the function to create matrices ####
build_matrices(RF_prediction = predictions, season.begin = "03-25", season.end = "08-20", period_length = 24, behavior_signa
l= "1")

matrices$mat_beh
```

```
#subset to those with complete incubation cycles
mat_keep_rows <- c("2015-2014", "2016-2013", "2018-2014", "2002-2014", "2002-2015")

matrices$mat_beh_full <-  matrices$mat_beh[rownames(matrices$mat_beh) %in% mat_keep_rows, ]
matrices$mat_fix_full <-  matrices$mat_fix[rownames(matrices$mat_fix) %in% mat_keep_rows, ]
```

here are the matrices

```
matrices$mat_beh_full
```

```
##           [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## 2002-2014   NA    1    1    1    1    1    1    1    1     1     1     1     1
## 2002-2015   NA    1    1    1    1    1    1    1    1     1     1     1     1
## 2015-2014   NA    1    1    1    1    1    1    1    1     1     1     1     1
## 2016-2013   NA    1    1    1    1    1    1    1    1     1     1     1     1
## 2018-2014   NA    1    1    1    1    1    1    1    1     1     1     1     1
##           [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## 2002-2014     1     1     1     1     1     1     1     1     1     1     1
## 2002-2015     1    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
## 2015-2014     1     1     1     1     1     1     1     1     1     1     1
## 2016-2013     1     1     1     1     1     1     1     1     1     1     1
## 2018-2014     1     1     1     1     1     1     1     1     1     1     1
##           [,25]
## 2002-2014     1
## 2002-2015    NA
## 2015-2014     1
## 2016-2013     1
## 2018-2014    NA
```

```
matrices$mat_fix_full
```

```
##           [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## 2002-2014   36   75   89   92   89   67   52   48   48    46    47    47    43
## 2002-2015   25   19   28   10    0   25   23   23   23     0    15    28    30
## 2015-2014   60   62   63   58   61   64   62   61   61    67    65    72    58
## 2016-2013   63   95  210  291  284  278  161  193  195   190   196   177   175
## 2018-2014   10   14   18   18   18   17   17   17   18    17    18    15    16
##           [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## 2002-2014    47    47    47    45    69    76   179    12    23    40    24
## 2002-2015    37     0     0     0     0     0     0     0     0     0     0
## 2015-2014    70    63    61    69    64    67    67    52    62    63    64
## 2016-2013   188     0   110   181    14     0     0     0     0     0     0
## 2018-2014    18    18    16    18    14    18    18    17    15    17    18
##           [,25]
## 2002-2014    38
## 2002-2015     0
## 2015-2014    66
## 2016-2013    19
## 2018-2014    15
```

and using your estimate_outcomes() function for an example while our custom function is in the works:

```
btgo_outcomes <- nestR::estimate_outcomes(matrices$mat_fix, matrices$mat_beh, model = "phi_time_p_time", mcmc_params = list
(burn_in = 1000, n_chain = 3, thin = 5, n_adapt = 1000, n_iter = 5000))
```

```
## Warning in max(which(ch[i, ] > 0)): no non-missing arguments to max; returning
## -Inf
```

```
## Error in n1:n2: result would be too long a vector
```

```
#view surv prob
btgo_outcomes$z
```

```
## Error in eval(expr, envir, enclos): object 'btgo_outcomes' not found
```

here is the function we use to create the obs and surv matrices. FYI, we coded nesting as ‘1’ and chick-tending as ‘4’ in our tracks

```
#### Function to construct the survival and observation matrices ####
build_matrices <- function(RF_prediction, season.begin = "01-01", season.end = "12-31", period_length = 27, behavior_signal=
"1"){

  # check input data
  # Check that all the fields are there
  if (any(!c(exists("id", where = RF_prediction),
             exists("t", where = RF_prediction),
             exists("b", where = RF_prediction)))) {

    stop("Either data does not include required fields or column names are different.
         Check that gps_data includes id, t, and b.")   }

  #check classes
  if (!(class(RF_prediction$id) == "character")) stop("id column needs to be character")
  if (!(class(RF_prediction$b) == "numeric")) stop("b column needs to be numeric")
  if (!(inherits(RF_prediction$t, "POSIXct"))) stop("t needs to be 'POSIXct' format")
  if (sum(is.na(RF_prediction$b)) > 0) stop("please exclude rows where date is NA")



  #extract Julian day
  Julian <- as.numeric(format(RF_prediction$t, "%j"))
  RF_prediction <- cbind(RF_prediction, Julian)

  #fate
  beh <- RF_prediction %>% group_by(id, Julian) %>% count(b)

  #define fate
  beh <- beh[beh$b==as.numeric(behavior_signal),]

  #create empty fate matrix
  mat_beh <- matrix(NA, nrow = length(unique(beh$id)), ncol = 365)

  #loop through individuals
  for(i in 1:length(beh$id)){
    mat_beh[match(beh[i,1], pull(unique(beh[,1]))),
            as.numeric(beh[i, 2])] <- as.numeric(beh[i,4])
  }
```

```
#fate matrix
mat_beh[is.na(mat_beh)] <- 0

for(i in 1:nrow(mat_beh)){ tmp <-mat_beh[i,max(which(mat_beh[i,] > 2))]}

apply(mat_beh>0,2,which.max)

max.col(t(mat_beh >0), "last")

#GPS fixes

fixes <- RF_prediction %>% group_by(id) %>% count(Julian)

#create blank matrix to fill
mat_fix <- matrix(NA, nrow = length(unique(fixes$id)), ncol = 365)


for(i in 1:length(fixes$id)){
  mat_fix[match(fixes[i,1], pull(unique(fixes[,1]))),
          as.numeric(fixes[i, 2])] <- as.numeric(fixes[i,3])
}

#GPS_fix_matrix
mat_fix[is.na(mat_fix)] <- 0


#


colnames(mat_fix) <- colnames(mat_beh) <- NULL

rownames(mat_fix) <- rownames(mat_beh) <- c(unique(beh$id))


#convert to POSIXct
season.begin_fmt <- as.POSIXct(season.begin, format = "%m-%d")
season.end_fmt <- as.POSIXct(season.end, format = "%m-%d")


## subset matrices to season lenght ##
```

```
mat_beh <- mat_beh[,as.numeric(format(season.begin_fmt, "%j")):as.numeric(format(season.end_fmt, "%j"))]

mat_fix <- mat_fix[,as.numeric(format(season.begin_fmt, "%j")):as.numeric(format(season.end_fmt, "%j"))]


#identify first non-zero value in each row
tmp_start <- as.vector(apply(mat_beh, 1, function(x) which(x!=0, arr.ind=T)))

#convert to list, then vector
lst <- list(NA,nrow(mat_beh))

for(i in 1:nrow(mat_beh)){
  lst[i] <- min(tmp_start[[i]])
}

tmp_start <- as.vector(do.call(rbind, lst))

tmp_end <- tmp_start + period_length

lst <- list(NA,nrow(mat_beh))
#use each to subset the matrices
for(i in 1:nrow(mat_beh)){
  lst[[i]] <- mat_beh[i,c(tmp_start[i]:tmp_end[i])]
}

mat_beh_final <- as.matrix(do.call(rbind, lst))

lst <- list(NA,nrow(mat_beh))
for(i in 1:nrow(mat_beh)){
  lst[[i]] <- mat_fix[i,c(tmp_start[i]:tmp_end[i])]
}

mat_fix_final <- as.matrix(do.call(rbind, lst))

#set rownames back
rownames(mat_fix_final) <- rownames(mat_beh_final) <- rownames(mat_beh)

#rename for ease
mat_fix <- mat_fix_final
mat_beh <- mat_beh_final
```

```
  #get things ready for Bayesian! State matrices are always kept to 1 or 0, dead or alive




  for(i in 1:nrow(mat_beh_final)){
    # The earliest "sighting" will always be the first day of the attempt
    #the last sighting will also be one
    n1 <- 1

    #identify last sighting
    n2 <- max(which(mat_beh_final[i,]>0))

    #mat_beh_final[mat_beh_final > 0] <- 1

    # ATTN: ---- THIS IS WHERE WE WOULD WANT A SENSITIVITY ANALYSIS - TO SEE A MINIMUM CUT OFF FOR THE NUMBER OF TIMES A BEH
AVIOR IS REGISTERED BEFORE WE CALL IT A '1' ----

    #set all between first and last to 1
    mat_beh_final[i, n1:n2] <- 1

    #reset the first to NA (this is by definition, we always assume first day we see them with certainty)
    mat_beh_final[i, n1] <- NA


    # Now set any states remaining as 0 to NA so that JAGS will estimate them
    mat_beh_final[mat_beh_final == 0] <- NA

  }




  matrices <<- list(mat_fix_final, mat_beh_final)
  names(matrices) <<- c("mat_fix", "mat_beh")
```

```
}

initialize_z <- function(matrices) {
  # Initialize state using the "capture history" (in CMR parlance)
  state <- ch

  # Loop through each nest
  for (i in 1:nrow(ch)) {
    # The earliest "sighting" will always be the first day of the attempt
    n1 <- 1

    # The last sighting is the last time the animal was observed at the nest
    n2 <- max(which(ch[i,] > 0))

    # Set all states between first and last to 1
    state[i, n1:n2] <- 1

    # Reset first to NA (because always see them on first day by definition)
    state[i, n1] <- NA
  }

  # Now set any states remaining as 0 to NA so that JAGS will estimate them
  state[state == 0] <- NA

  # Return
  return(state)
}
```