HR Performance Calculations

Odd Jacobson

2022-12-13

This document contains the calculations for all the variables necessary for the statistical analysis. This includes our measure of home range performance, which is the proportion of locations from the complete segments that fall within the 95% UD estimates of the emulated regimes. We also gather the number of unique weeks, the number of recorded locations, and the home range crossing times and put everything into a master dataframe.

Note: Some of the code here is unnecessarily complicated and long. I apologize, I was not (and still am not) a great coder yet.

Load Packages

```
library(tidyverse)
library(ctmm)
library(sf)
library(ctmmweb)
library(lubridate)
library(sp)
```

Read in data

```
# read in data, fits, and akdes
AA <- readRDS("../Intermediate/DATA_aa.rds")
CE <- readRDS("../Intermediate/DATA_ce.rds")
RR <- readRDS("../Intermediate/DATA_rr.rds")
AA2 <- readRDS("../Intermediate/DATA_aa2.rds")
SP <- readRDS("../Intermediate/DATA_sp.rds")
FL <- readRDS("../Intermediate/DATA_fl.rds")
FITSa <-readRDS("../Intermediate/FITS_aa.rds")
FITSc <- readRDS("../Intermediate/FITS_ce.rds")
FITSr <- readRDS("../Intermediate/FITS_rr.rds")
FITSa2 <- readRDS("../Intermediate/FITS_aa2.rds")
FITSs <- readRDS("../Intermediate/FITS_sp.rds")
FITSf <- readRDS("../Intermediate/FITS_fl.rds")
AKDEa <- readRDS("../Intermediate/AKDE_aa.rds")
AKDEc <- readRDS("../Intermediate/AKDE_ce.rds")
AKDEr <- readRDS("../Intermediate/AKDE_rr.rds")
AKDEa2 <- readRDS("../Intermediate/AKDE_aa2.rds")
AKDEs <- readRDS("../Intermediate/AKDE_sp.rds")
AKDEf <- readRDS("../Intermediate/AKDE_fl.rds")
```

Functions to get ctmm calculations from FITS and AKDEs

```
# FUNCTIONS TO GET HR area and BI Overlap FOR DATAFRAME

## function to get HR area, CIs, and # HR xings from AKDEs
summarize_akde <- function(akde){

  summary <- summary(akde, units = FALSE)

  tibble(id = akde@info$identity,
         HR_low = (summary$CI[1])/1000000, # convert m2 to km2
         HR_area = (summary$CI[2])/1000000,
         HR_high = (summary$CI[3])/1000000,
         DOF = (summary$DOF[1]))

}

# wrapper to stack area info into data frame
make_df <- function(id){
  map_dfr(id, summarize_akde)
}

## function to get HRxing time from FITS
summarize_fits <- function(fit){

  summary <- summary(fit, units = FALSE)

  tibble(id = fit@info$identity,
         tau_low = (summary$CI[2,1]/3600), # convert seconds to hours
         tau = (summary$CI[2,2]/3600),
         tau_high = (summary$CI[2,3]/3600))

}

# wrapper to stack area info into data frame
add_tau <- function(id){
  map_dfr(id, summarize_fits)
}
```

Calculations

Getting frequency and proportion of points that fell within the range estimates using %over% from the sf package, also HR xing time, etc.

```
## CALCULATING PROPORTION POINTS WITHIN HR FROM TOTAL SAMPLE/ CIs/ HRxing time

# creates list of dataframes with overlap of total points in each HR with CIs then
↪  extracts proportions
# AA
AAt <- AAp <-  AAp_low <- AAp_high <- AAf <-  AAf_low <- AAf_high <- AAx <- AAx_low <-
↪  AAx_high <- list()
for(i in 1:length(AKDEa)){
```

```
  AAt[[i]] <- SpatialPoints.telemetry(AA[1]) %over%
↪  SpatialPolygonsDataFrame.UD(AKDEa[[i]] ,level.UD=0.95) %>%
    table(useNA = "always") %>% # keep number of points that fell outside upper CI
    data.frame()  %>%
    mutate(Prop = round(Freq/sum(Freq), digits = 3), # calculates proportion from
     ↪  frequency
           Freq = Freq);
  AAf[[i]] <- AAt[[i]]$Freq[1] + AAt[[i]]$Freq[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  AAf_low[[i]] <- AAt[[i]]$Freq[3]; # proportion only within lower bound
  AAf_high[[i]] <- AAt[[i]]$Freq[1] + AAt[[i]]$Freq[2] + AAt[[i]]$Freq[3]; # all combined
↪  to get upper bound
  AAp[[i]] <- AAt[[i]]$Prop[1] + AAt[[i]]$Prop[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  AAp_low[[i]] <- AAt[[i]]$Prop[3]; # proportion only within lower bound
  AAp_high[[i]] <- AAt[[i]]$Prop[1] + AAt[[i]]$Prop[2] + AAt[[i]]$Prop[3]; # all combined
↪  to get upper bound
  AAx[[i]] <- as.numeric((FITSa[[i]][names(FITSa[[i]])=='tau']$tau[1])/3600) # get HR
↪  xing time
}
names(AAt) <- names(AAp_low) <- names(AAp_high) <- names(AAp) <- names(AAf_low) <-
↪  names(AAf_high) <- names(AAf) <- names(AAx) <- names(AA)

# CE
CEt <- CEp <- CEp_low <- CEp_high <- CEf <- CEf_low <- CEf_high <- CEx <- list()
for(i in 1:length(AKDEc)){
  CEt[[i]] <- SpatialPoints.telemetry(CE[1]) %over%
↪  SpatialPolygonsDataFrame.UD(AKDEc[[i]] ,level.UD=0.95) %>%
    table(useNA = "always") %>% # keep number of points that fell outside upper CI
    data.frame()  %>%
    mutate(Prop = round(Freq/sum(Freq), digits = 3), # calculates proportion from
     ↪  frequency
           Freq = Freq);
  CEf[[i]] <- CEt[[i]]$Freq[1] + CEt[[i]]$Freq[3]; # takes the frequency that fell within
↪  lower CI plus proportion that fell between mean and lower
  CEf_low[[i]] <- CEt[[i]]$Freq[3];
  CEf_high[[i]] <- CEt[[i]]$Freq[1] + CEt[[i]]$Freq[2] + CEt[[i]]$Freq[3];
  CEp[[i]] <- CEt[[i]]$Prop[1] + CEt[[i]]$Prop[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  CEp_low[[i]] <- CEt[[i]]$Prop[3];
  CEp_high[[i]] <- CEt[[i]]$Prop[1] + CEt[[i]]$Prop[2] + CEt[[i]]$Prop[3];
  CEx[[i]] <- as.numeric((FITSc[[i]][names(FITSc[[i]])=='tau']$tau[1])/3600) # get HR
↪  xing time

}
names(CEt) <- names(CEp_low) <- names(CEp_high) <- names(CEp) <- names(CEf_low) <-
↪  names(CEf_high) <- names(CEf) <- names(CEx) <-  names(CE)

# RR
RRt <- RRp <- RRp_low <- RRp_high <- RRf <- RRf_low <- RRf_high <- RRx <-  list()
for(i in 1:length(AKDEr)){
  RRt[[i]] <- SpatialPoints.telemetry(RR[1]) %over%
↪  SpatialPolygonsDataFrame.UD(AKDEr[[i]] ,level.UD=0.95) %>%
```

```
    table(useNA = "always") %>% # keep number of points that fell outside upper CI
    data.frame()  %>%
    mutate(Prop = round(Freq/sum(Freq), digits = 3), # calculates proportion from
     ↪  frequency
           Freq = Freq);
  RRf[[i]] <- RRt[[i]]$Freq[1] + RRt[[i]]$Freq[3]; # takes frequency that fell within
↪  lower CI plus proportion that fell between mean and lower
  RRf_low[[i]] <- RRt[[i]]$Freq[3];
  RRf_high[[i]] <- RRt[[i]]$Freq[1] + RRt[[i]]$Freq[2] + RRt[[i]]$Freq[3];
  RRp[[i]] <- RRt[[i]]$Prop[1] + RRt[[i]]$Prop[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  RRp_low[[i]] <- RRt[[i]]$Prop[3];
  RRp_high[[i]] <- RRt[[i]]$Prop[1] + RRt[[i]]$Prop[2] + RRt[[i]]$Prop[3];
  RRx[[i]] <- as.numeric((FITSr[[i]][names(FITSr[[i]])=='tau']$tau[1])/3600) # get HR
↪  xing time

}
names(RRt) <- names(RRp_low) <- names(RRp_high) <- names(RRp)  <- names(RRf_low) <-
↪  names(RRf_high) <- names(RRf) <- names(RRx) <- names(RR)

# AA2
# AA
AA2t <- AA2p <-  AA2p_low <- AA2p_high <- AA2f <-  AA2f_low <- AA2f_high <- AA2x <-
↪  list()
for(i in 1:length(AKDEa2)){
  AA2t[[i]] <- SpatialPoints.telemetry(AA2[1]) %over%
↪  SpatialPolygonsDataFrame.UD(AKDEa2[[i]] ,level.UD=0.95) %>%
    table(useNA = "always") %>% # keep number of points that fell outside upper CI
    data.frame()  %>%
    mutate(Prop = round(Freq/sum(Freq), digits = 3), # calculates proportion from
     ↪  frequency
           Freq = Freq);
  AA2f[[i]] <- AA2t[[i]]$Freq[1] + AA2t[[i]]$Freq[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  AA2f_low[[i]] <- AA2t[[i]]$Freq[3]; # proportion only within lower bound
  AA2f_high[[i]] <- AA2t[[i]]$Freq[1] + AA2t[[i]]$Freq[2] + AA2t[[i]]$Freq[3]; # all
↪  combined to get upper bound
  AA2p[[i]] <- AA2t[[i]]$Prop[1] + AA2t[[i]]$Prop[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  AA2p_low[[i]] <- AA2t[[i]]$Prop[3]; # proportion only within lower bound
  AA2p_high[[i]] <- AA2t[[i]]$Prop[1] + AA2t[[i]]$Prop[2] + AA2t[[i]]$Prop[3]; # all
↪  combined to get upper bound
  AA2x[[i]] <- as.numeric((FITSa2[[i]][names(FITSa2[[i]])=='tau']$tau[1])/3600) # get HR
↪  xing time
}
names(AA2t) <- names(AA2p_low) <- names(AA2p_high) <- names(AA2p) <- names(AA2f_low) <-
↪  names(AA2f_high) <- names(AA2f) <- names(AA2x) <- names(AA2)

# SP
SPt <- SPp <-  SPp_low <- SPp_high <- SPf <-  SPf_low <- SPf_high <- SPx <- list()
for(i in 1:length(AKDEs)){
  SPt[[i]] <- SpatialPoints.telemetry(SP[1]) %over%
↪  SpatialPolygonsDataFrame.UD(AKDEs[[i]] ,level.UD=0.95) %>%
```

```r
    table(useNA = "always") %>% # keep number of points that fell outside upper CI
    data.frame()  %>%
    mutate(Prop = round(Freq/sum(Freq), digits = 3), # calculates proportion from
     ↪  frequency
           Freq = Freq);
  SPf[[i]] <- SPt[[i]]$Freq[1] + SPt[[i]]$Freq[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  SPf_low[[i]] <- SPt[[i]]$Freq[3]; # proportion only within lower bound
  SPf_high[[i]] <- SPt[[i]]$Freq[1] + SPt[[i]]$Freq[2] + SPt[[i]]$Freq[3]; # all combined
↪  to get upper bound
  SPp[[i]] <- SPt[[i]]$Prop[1] + SPt[[i]]$Prop[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  SPp_low[[i]] <- SPt[[i]]$Prop[3]; # proportion only within lower bound
  SPp_high[[i]] <- SPt[[i]]$Prop[1] + SPt[[i]]$Prop[2] + SPt[[i]]$Prop[3]; # all combined
↪  to get upper bound
  SPx[[i]] <- as.numeric((FITSs[[i]][names(FITSs[[i]])=='tau']$tau[1])/3600) # get HR
↪  xing time
}
names(SPt) <- names(SPp_low) <- names(SPp_high) <- names(SPp) <- names(SPf_low) <-
↪  names(SPf_high) <- names(SPf) <- names(SPx) <- names(SP)

# CE
FLt <- FLp <-  FLp_low <- FLp_high <- FLf <-  FLf_low <- FLf_high <- FLx <- list()
for(i in 1:length(AKDEf)){
  FLt[[i]] <- SpatialPoints.telemetry(FL[1]) %over%
↪  SpatialPolygonsDataFrame.UD(AKDEf[[i]] ,level.UD=0.95) %>%
    table(useNA = "always") %>% # keep number of points that fell outside upper CI
    data.frame()  %>%
    mutate(Prop = round(Freq/sum(Freq), digits = 3), # calculates proportion from
     ↪  frequency
           Freq = Freq);
  FLf[[i]] <- FLt[[i]]$Freq[1] + FLt[[i]]$Freq[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  FLf_low[[i]] <- FLt[[i]]$Freq[3]; # proportion only within lower bound
  FLf_high[[i]] <- FLt[[i]]$Freq[1] + FLt[[i]]$Freq[2] + FLt[[i]]$Freq[3]; # all combined
↪  to get upper bound
  FLp[[i]] <- FLt[[i]]$Prop[1] + FLt[[i]]$Prop[3]; # takes the proportion that fell
↪  within lower CI plus proportion that fell between mean and lower
  FLp_low[[i]] <- FLt[[i]]$Prop[3]; # proportion only within lower bound
  FLp_high[[i]] <- FLt[[i]]$Prop[1] + FLt[[i]]$Prop[2] + FLt[[i]]$Prop[3]; # all combined
↪  to get upper bound
  FLx[[i]] <- as.numeric((FITSf[[i]][names(FITSf[[i]])=='tau']$tau[1])/3600) # get HR
↪  xing time
}
names(FLt) <- names(FLp_low) <- names(FLp_high) <- names(FLp) <- names(FLf_low) <-
↪  names(FLf_high) <- names(FLf) <- names(FLx) <- names(FL)
```

Creating mini data frames of overlaps, taus, hr area, number of xings

```r
# creating dataframe with HR_area and xings
HR <- make_df(AKDEa) %>%
```

```
  rbind(make_df(AKDEc)) %>%
  rbind(make_df(AKDEr)) %>%
  rbind(make_df(AKDEa2)) %>%
  rbind(make_df(AKDEs)) %>%
  rbind(make_df(AKDEf)) %>%
  #filter(!grepl('all', id)) %>%
  dplyr::select(-id)

# creating datafram with taus
tau <- add_tau(FITSa) %>%
  rbind(add_tau(FITSc)) %>%
  rbind(add_tau(FITSr)) %>%
  rbind(add_tau(FITSa2)) %>%
  rbind(add_tau(FITSs)) %>%
  rbind(add_tau(FITSf)) %>%
  #filter(!grepl('all', id)) %>%
  dplyr::select(-id)
```

Master data frame

```
# add info with proportions, number of days and weeks, temporal length, absolute sample
↪  size from all three trials
# functions within create vectors from each timestamp column (1st column) within each
↪  element of the list

DF <- tibble(ID = names(AA),
             Group = "AA",
             Prop = as.numeric(unlist(AAp)),
             Prop_low = as.numeric(unlist(AAp_low)),
             Prop_high = as.numeric(unlist(AAp_high)),
             Freq = as.numeric(unlist(AAf)),
             Freq_low = as.numeric(unlist(AAf_low)),
             Freq_high = as.numeric(unlist(AAf_high)),
             total_points = length(SpatialPoints.telemetry(AA[1])),
             #Tau = as.numeric(unlist(AAx)),
             num_unidays = sapply( unname(sapply(AA, "[[", 1)), function(x)
             ↪  length(unique(date(x)))),
             num_uniweeks = sapply( unname(sapply(AA, "[[", 1)), function(x)
             ↪  length(unique(week(x)))),
             day_span = sapply( unname(sapply(AA, "[[", 1)), function(x) difftime(max(x),
             ↪  min(x), units = "days")),
             num_points = sapply( unname(sapply(AA, "[[", 1)), length)) %>%
  rbind(tibble(ID = names(CE),
               Group = "CE",
               Prop = as.numeric(unlist(CEp)),
               Prop_low = as.numeric(unlist(CEp_low)),
               Prop_high = as.numeric(unlist(CEp_high)),
               Freq = as.numeric(unlist(CEf)),
               Freq_low = as.numeric(unlist(CEf_low)),
               Freq_high = as.numeric(unlist(CEf_high)),
               total_points = length(SpatialPoints.telemetry(CE[1])),
```

```
              #Tau = as.numeric(unlist(CEx)),
              num_unidays = sapply( unname(sapply(CE, "[[", 1)), function(x)
              ↪  length(unique(date(x)))),
              num_uniweeks = sapply( unname(sapply(CE, "[[", 1)), function(x)
              ↪  length(unique(week(x)))),
              day_span = sapply( unname(sapply(CE, "[[", 1)), function(x)
              ↪  difftime(max(x), min(x), units = "days")),
              num_points = sapply( unname(sapply(CE, "[[", 1)), length))) %>%
rbind(tibble(ID = names(RR),
              Group = "RR",
              Prop = as.numeric(unlist(RRp)),
              Prop_low = as.numeric(unlist(RRp_low)),
              Prop_high = as.numeric(unlist(RRp_high)),
              Freq = as.numeric(unlist(RRf)),
              Freq_low = as.numeric(unlist(RRf_low)),
              Freq_high = as.numeric(unlist(RRf_high)),
              total_points = length(SpatialPoints.telemetry(RR[1])),
              #Tau = as.numeric(unlist(RRx)),
              num_unidays = sapply( unname(sapply(RR, "[[", 1)), function(x)
              ↪  length(unique(date(x)))),
              num_uniweeks = sapply( unname(sapply(RR, "[[", 1)), function(x)
              ↪  length(unique(week(x)))),
              day_span = sapply( unname(sapply(RR, "[[", 1)), function(x)
              ↪  difftime(max(x), min(x), units = "days")),
              num_points = sapply( unname(sapply(RR, "[[", 1)), length))) %>%
rbind(tibble(ID = names(AA2),
              Group = "AA2",
              Prop = as.numeric(unlist(AA2p)),
              Prop_low = as.numeric(unlist(AA2p_low)),
              Prop_high = as.numeric(unlist(AA2p_high)),
              Freq = as.numeric(unlist(AA2f)),
              Freq_low = as.numeric(unlist(AA2f_low)),
              Freq_high = as.numeric(unlist(AA2f_high)),
              total_points = length(SpatialPoints.telemetry(AA2[1])),
              #Tau = as.numeric(unlist(AA2x)),
              num_unidays = sapply( unname(sapply(AA2, "[[", 1)), function(x)
              ↪  length(unique(date(x)))),
              num_uniweeks = sapply( unname(sapply(AA2, "[[", 1)), function(x)
              ↪  length(unique(week(x)))),
              day_span = sapply( unname(sapply(AA2, "[[", 1)), function(x)
              ↪  difftime(max(x), min(x), units = "days")),
              num_points = sapply( unname(sapply(AA2, "[[", 1)), length))) %>%
rbind(tibble(ID = names(SP),
              Group = "SP",
              Prop = as.numeric(unlist(SPp)),
              Prop_low = as.numeric(unlist(SPp_low)),
              Prop_high = as.numeric(unlist(SPp_high)),
              Freq = as.numeric(unlist(SPf)),
              Freq_low = as.numeric(unlist(SPf_low)),
              Freq_high = as.numeric(unlist(SPf_high)),
              total_points = length(SpatialPoints.telemetry(SP[1])),
              #Tau = as.numeric(unlist(SPx)),
              num_unidays = sapply( unname(sapply(SP, "[[", 1)), function(x)
              ↪  length(unique(date(x)))),
```

```
                num_uniweeks = sapply( unname(sapply(SP, "[[", 1)), function(x)
                ↪  length(unique(week(x)))),
                day_span = sapply( unname(sapply(SP, "[[", 1)), function(x)
                ↪  difftime(max(x), min(x), units = "days")),
                num_points = sapply( unname(sapply(SP, "[[", 1)), length))) %>%
  rbind(tibble(ID = names(FL),
                Group = "FL",
                Prop = as.numeric(unlist(FLp)),
                Prop_low = as.numeric(unlist(FLp_low)),
                Prop_high = as.numeric(unlist(FLp_high)),
                Freq = as.numeric(unlist(FLf)),
                Freq_low = as.numeric(unlist(FLf_low)),
                Freq_high = as.numeric(unlist(FLf_high)),
                total_points = length(SpatialPoints.telemetry(FL[1])),
                #Tau = as.numeric(unlist(FLx)),
                num_unidays = sapply( unname(sapply(FL, "[[", 1)), function(x)
                ↪  length(unique(date(x)))),
                num_uniweeks = sapply( unname(sapply(FL, "[[", 1)), function(x)
                ↪  length(unique(week(x)))),
                day_span = sapply( unname(sapply(FL, "[[", 1)), function(x)
                ↪  difftime(max(x), min(x), units = "days")),
                num_points = sapply( unname(sapply(FL, "[[", 1)), length))) %>%
  cbind(HR) %>% #combine HR dataframe
  cbind(tau) %>% #add taus
  mutate(xings_low = (day_span*24)/tau_high, # high tau gives lower CI of hr_xings and
  ↪  vice versa
        xings = (day_span*24)/tau, # this is higher than DOF of model because of
        ↪  irregular data (see ctmm group conversations)
        xings_high = (day_span*24)/tau_low) %>%
  mutate_if(is.numeric, round, digits = 3) %>%
  relocate(Group)

# filter out complete segments
emulated_df <- DF %>%
  filter(!grepl('all', ID)) %>% # removes columns using all points, only want emulated
  ↪  regimes
  mutate(clumped_or_random = str_sub(ID,1,nchar(ID)-1),
        ID = str_c(toupper(str_sub(ID,1,1)), str_sub(ID, -1, -1)),
        raw_locs = real_locs$num_points,
        locs_30sec = locs_30sec$num_points)

# filter out sampling regimes
true_df <- DF %>%
  filter(grepl('all', ID)) %>%
  dplyr::select(Group, HR_low, HR_area, HR_high, DOF, tau_low, tau, tau_high, xings_low,
↪  xings, xings_high)


# save dataframes
saveRDS(emulated_df, "Intermediate/DF_Performance.rds")
saveRDS(true_df, "Intermediate/DF_True_HRs.rds")
```