# Route Selection Analyses

Hester Brønnvik

7/5/2021

---

```
#set a working directory and load required packages
setwd("C:/Users/Tess Bronnvik/Desktop/Br-nnvik_honey_buzzard_ssf")
ssf_packs <- c("lubridate", "amt", "purrr","move", "mapview", "ggpubr", "survival", "sjPlot", "MASS", "
lapply(ssf_packs, require, character.only = TRUE)
```

---

---

```
#import required functions
NCEP.loxodrome.na <- function (lat1, lat2, lon1, lon2) {
  deg2rad <- pi/180
  acot <- function(x) {
    return(atan(1/x))
  }
  lat1 <- deg2rad * lat1
  lat2 <- deg2rad * lat2
  lon1 <- deg2rad * lon1
  lon2 <- deg2rad * lon2
  deltaLon <- lon2 - lon1
  pi4 <- pi/4
  Sig1 <- log(tan(pi4 + lat1/2))
  Sig2 <- log(tan(pi4 + lat2/2))
  deltaSig <- Sig2 - Sig1
  if (deltaLon == 0 && deltaSig > 0) {
    head <- 0
  }
  else if (deltaLon == 0 && deltaSig < 0) {
    head <- 180
  }
  else if (deltaSig == 0 && deltaLon > 0) {
    head <- 90
  }
  else if (deltaSig == 0 && deltaLon < 0) {
    head <- 270
  }
  else if (deltaSig < 0 && deltaLon < 0) {
    head <- acot(deltaSig/deltaLon) * 180/pi + 180
  }
```

```
  else if (deltaSig < 0 && deltaLon > 0) {
    head <- acot(deltaSig/deltaLon) * 180/pi + 180
  }
  else if (deltaSig > 0 && deltaLon > 0) {
    head <- acot(deltaSig/deltaLon) * 180/pi
  }
  else if (deltaSig > 0 && deltaLon < 0) {
    head <- acot(deltaSig/deltaLon) * 180/pi + 360
  }
  else {
    head <-NA}
  return(head)
}
source("wind_support_Kami.R")
```

---

---

```
# set criteria for tracks
stepNumber <- 150 # random steps
toleranceLength <- 15 # tolerance in minutes
wgs <- CRS("+proj=longlat +datum=WGS84 +no_defs") # map projection
```

---

**Prepare data for Movebank**

---

```
# retrieve the full data set
full_data <- read.csv("./original_data/full_buzz_set.csv",stringsAsFactors = F, header = T)
```

---

---

```
# reformat the timestamps
full_data$timestamp <- as.POSIXct(strptime(full_data$dt, format = "%Y-%m-%d %H:%M:%S"),tz = "UTC")
```

---

---

```
# select the autumn migrations
all_autumns <- full_data[grep("autumn", full_data$phase),]
```

---

---

```r
# find and remove duplicate observations
doubles <- all_autumns %>% dplyr::select(long, lat, name, timestamp) %>%
  duplicated
sum(doubles) # 59 duplicates
```

```
## [1] 59
```

```r
all_autumns <- all_autumns[doubles != TRUE,]
```

---

---

```r
# get the 204 reads when the birds did not move and remove them from the data
resting_reads <- read.csv("resting.csv", stringsAsFactors = FALSE)
all_autumns <- all_autumns %>% mutate(id_ts = paste(name,timestamp, sep="_")) %>%
  filter(!id_ts %in% resting_reads$id_ts)
```

---

---

```r
# order all the data by timestamp
all_autumns <- all_autumns %>%
  arrange(timestamp)
```

---

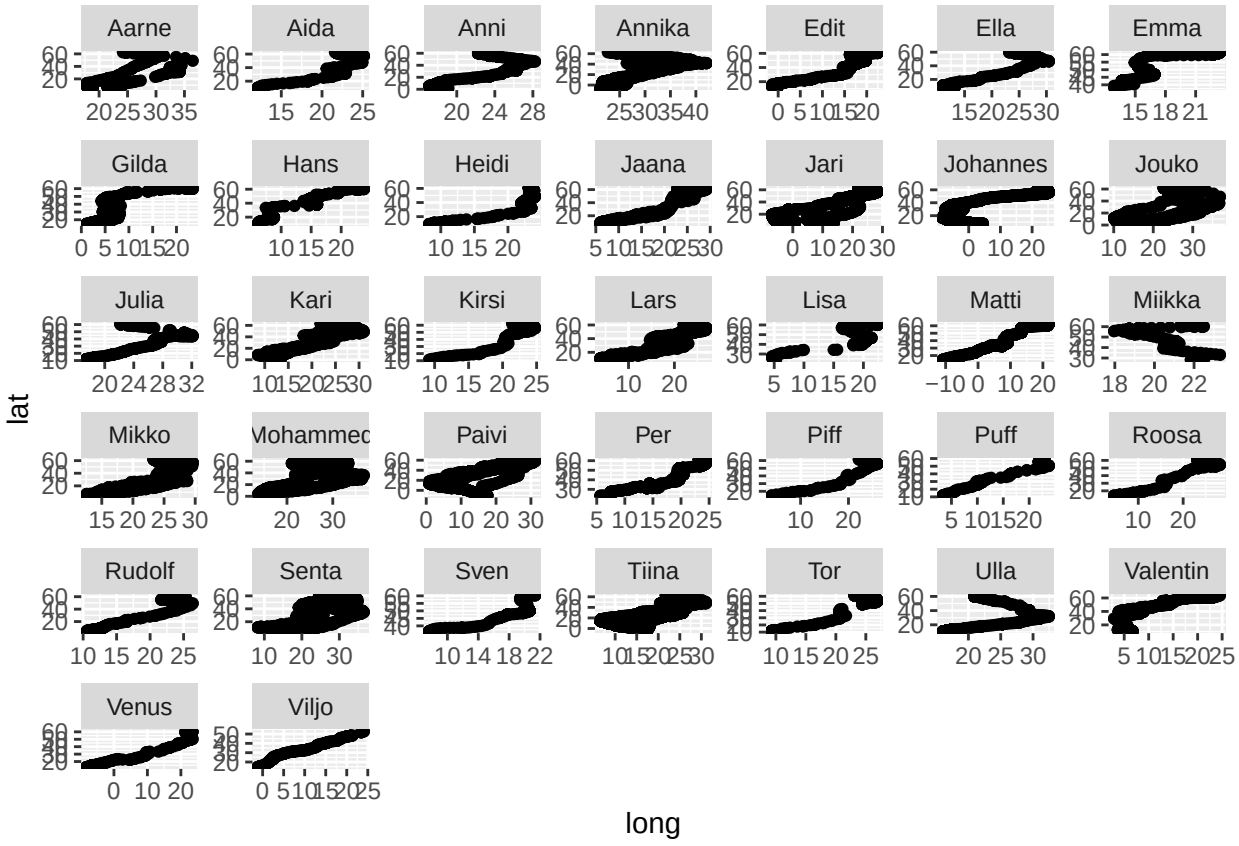
---

```r
# create a unique identifier for each migratory journey
all_autumns$id_year <- paste(all_autumns$name, all_autumns$yr, sep="_")
```

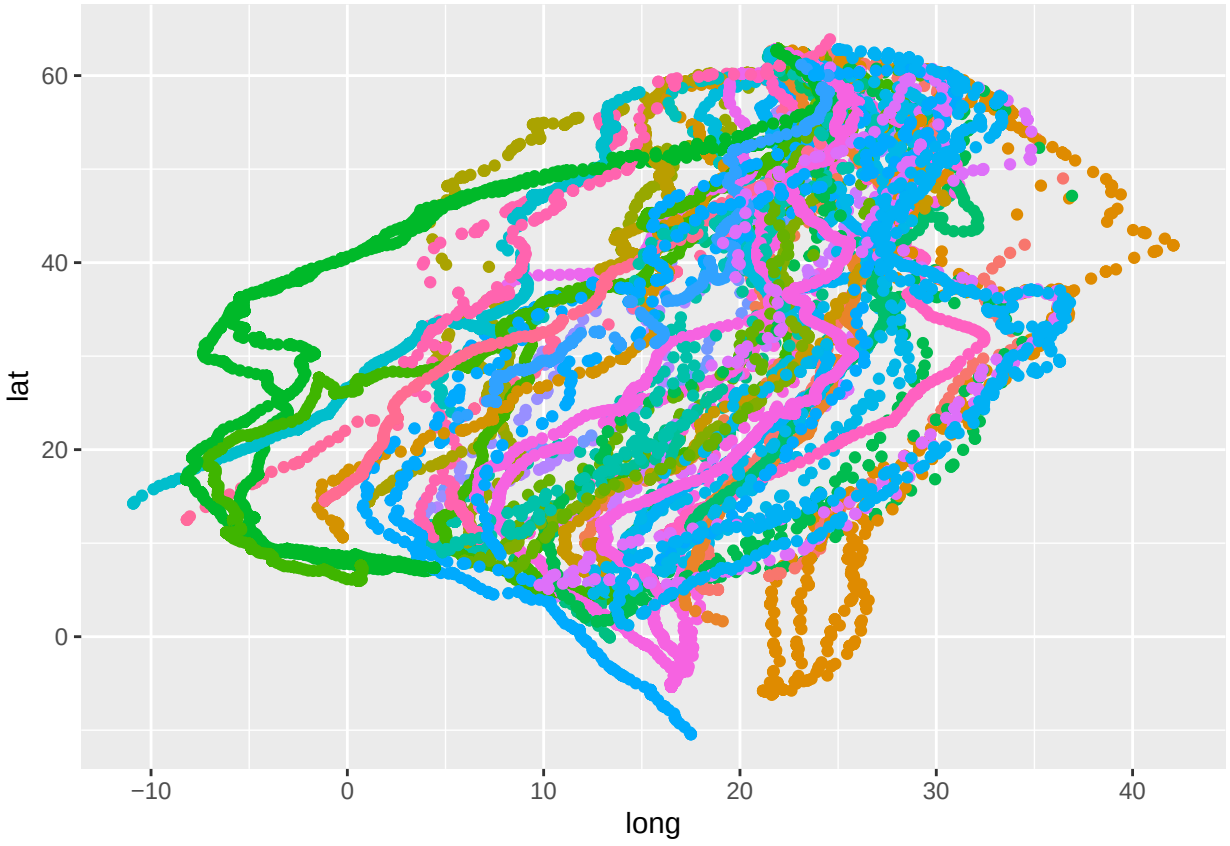
---

---

```r
# make simple plots of lat/long to check for outliers
ggplot(all_autumns, aes(x=long, y=lat)) + geom_point()+ facet_wrap(~name, scales="free")
```

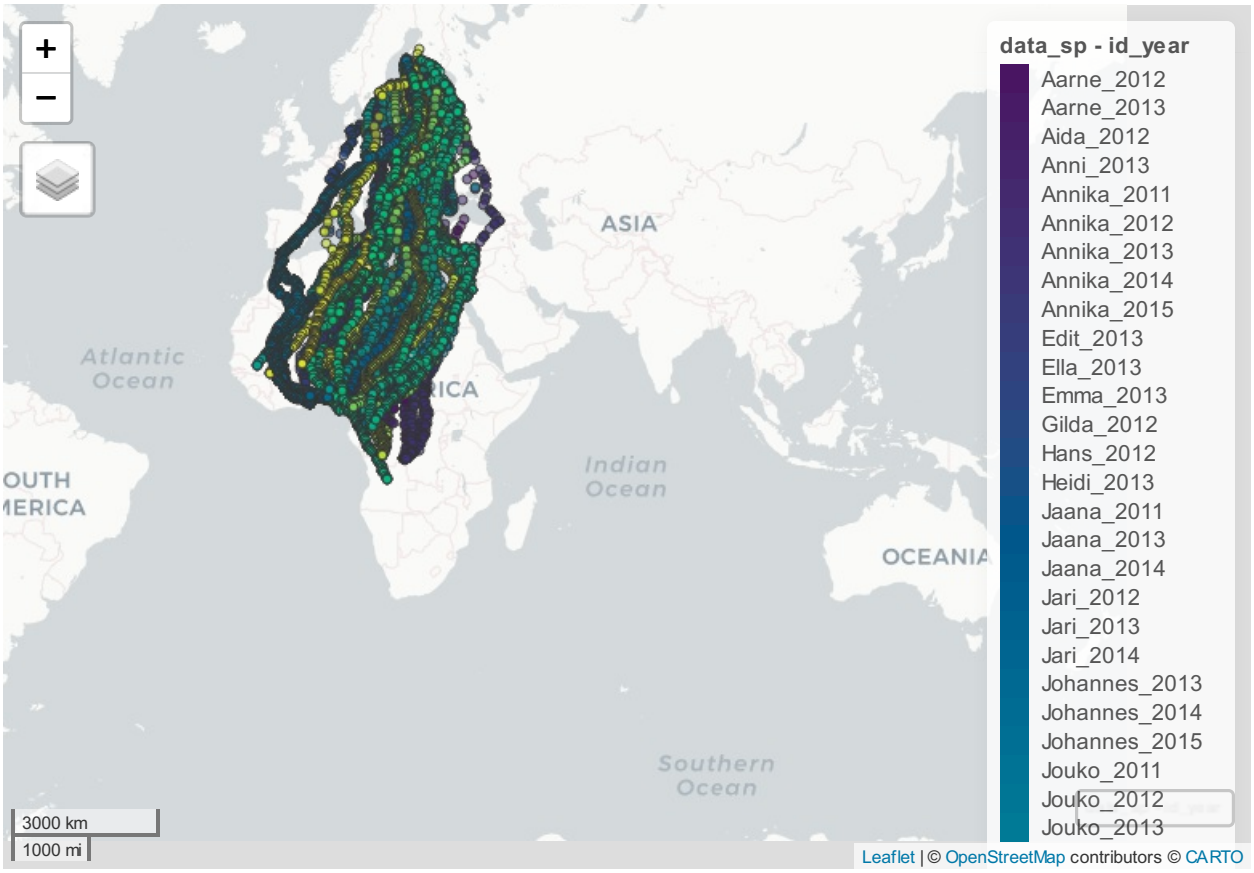



```
ggplot(all_autumns, aes(x=long, y=lat, color=as.factor(name))) + geom_point() + theme(legend.position =
```




---

```
# make a map of individual tracks
data_sp <- all_autumns # store the data, then create a spatial object for plotting
coordinates(data_sp) <- ~ long + lat # set coordinates
proj4string(data_sp) <- wgs # set projection
mapView(data_sp, zcol = "id_year", burst = F, cex = 3, color = rainbow) # plot on a map
```




---

```
# the data need to be re-sampled at different rates.
# separate them into different objects so that each can be processed individually.

# the individuals with 1 hour sampling rates
autumns_one <- all_autumns %>% filter(name == "Anni" | name == "Per" |
                                      name == "Sven"| name == "Viljo" |
                                      name == "Jari" |
                                      name == "Johannes")

# the individuals with 2 hour sampling rates
autumns_two <- all_autumns %>% filter(name == "Edit" | name == "Emma" | name == "Julia" |
                                      name == "Matti"| name == "Miikka"| name == "Ulla" |
                                      name == "Aida" |name == "Ella" | name == "Heidi" |
                                      name == "Kirsi" | name == "Gilda" | name == "Lisa"|
                                      name == "Aarne" | name == "Valentin")

# the individuals with 3 hour sampling rates
autumns_three <- all_autumns %>% filter(name == "Mohammed")

# the individuals with 4 hour sampling rates
autumns_four <- all_autumns %>% filter(name == "Annika" | name == "Jaana"| name == "Kari" |
                                       name == "Lars"| name == "Piff"|
                                       name == "Puff" | name == "Roosa"| name == "Senta"|
                                       name == "Tor" | name == "Tiina" |name == "Jouko" |
```

```
                                        name == "Mikko" |name == "Paivi" | name == "Hans" |
                                        name == "Venus" | name == "Rudolf")
```

---

---

```
# then build tracks at each rate

#create one hourly tracks
autumn_tracks_1 <- lapply(split(autumns_one, autumns_one$id_year), function(x){
  # make a track object
  trk <- mk_track(x,.x=long, .y=lat, .t=timestamp, id = name, crs = wgs)

  # resample to a consistent time between steps
  trk <- track_resample(trk, rate = minutes(60), tolerance = minutes(toleranceLength))

  # remove bursts with fewer than 3 steps to allow pythagorean headings
  trk <- filter_min_n_burst(trk, 3)

  # burst steps
  burst <- steps_by_burst(trk, keep_cols = "start")

  # create random steps using fitted gamma and von mises distributions and append
  rnd_stps <- burst %>% random_steps(n_control = stepNumber)

}) %>% reduce(rbind)

# create two hourly tracks
autumn_tracks_2 <- lapply(split(autumns_two, autumns_two$id_year), function(x){
  # make a track object
  trk <- mk_track(x,.x=long, .y=lat, .t=timestamp, id = name, crs = wgs)

  # resample to a consistent time between steps
  trk <- track_resample(trk, rate = minutes(120), tolerance = minutes(toleranceLength))

  # remove bursts with fewer than 3 steps to allow pythagorean headings
  trk <- filter_min_n_burst(trk, 3)

  # burst steps
  burst <- steps_by_burst(trk, keep_cols = "start")

  # create random steps using fitted gamma and von mises distributions and append
  rnd_stps <- burst %>% random_steps(n_control = stepNumber)

}) %>% reduce(rbind)

# create three hourly tracks
autumn_tracks_3 <- lapply(split(autumns_three, autumns_three$id_year), function(x){
  # make a track object
  trk <- mk_track(x,.x=long, .y=lat, .t=timestamp, id = name, crs = wgs)

  # resample to a consistent time between steps
```

```
  trk <- track_resample(trk, rate = minutes(180), tolerance = minutes(toleranceLength))

  # remove bursts with fewer than 3 steps to allow pythagorean headings
  trk <- filter_min_n_burst(trk, 3)

  # burst steps
  burst <- steps_by_burst(trk, keep_cols = "start")

  # create random steps using fitted gamma and von mises distributions and append
  rnd_stps <- burst %>% random_steps(n_control = stepNumber)

}) %>% reduce(rbind)

# create four hourly tracks
autumn_tracks_4 <- lapply(split(autumns_four, autumns_four$id_year), function(x){
  # make a track object
  trk <- mk_track(x,.x=long, .y=lat, .t=timestamp, id = name, crs = wgs)

  # resample to a consistent time between steps
  trk <- track_resample(trk, rate = minutes(240), tolerance = minutes(toleranceLength))

  # remove bursts with fewer than 3 steps to allow pythagorean headings
  trk <- filter_min_n_burst(trk, 3)

  # burst steps
  burst <- steps_by_burst(trk, keep_cols = "start")

  # create random steps using fitted gamma and von mises distributions and append
  rnd_stps <- burst %>% random_steps(n_control = stepNumber)

}) %>% reduce(rbind)
```

---

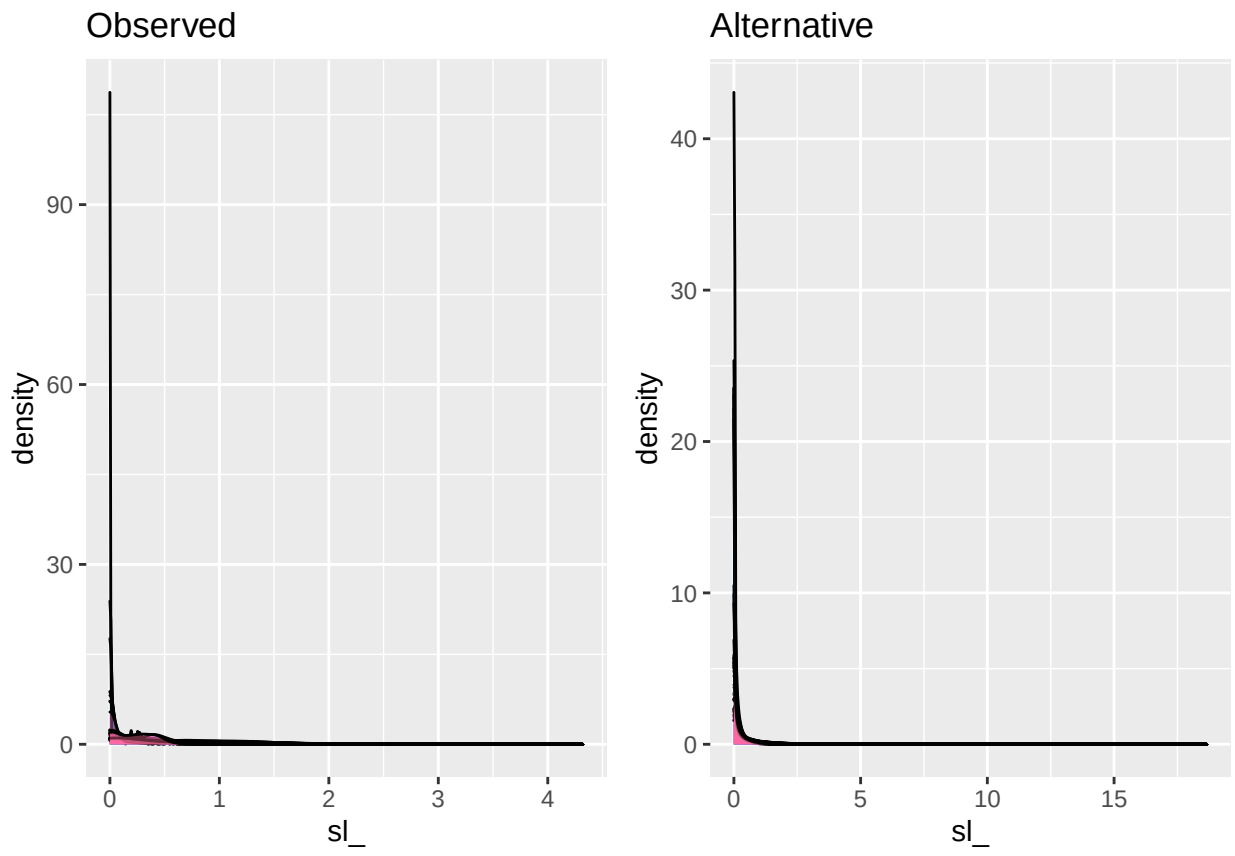
---

```
# combine those back into one data set
autumn_track <- rbind(autumn_tracks_1,autumn_tracks_2,autumn_tracks_3,autumn_tracks_4)
```

---

---

```
# plot the step lengths for observed and random steps
obs <- autumn_track[autumn_track$case_ == TRUE,] %>% dplyr::select(id, sl_) %>%
  ggplot(aes(sl_, fill = factor(id))) + ggtitle("Observed") + geom_density(alpha = 0.4)
rand <- autumn_track[autumn_track$case_ == FALSE,] %>% dplyr::select(id, sl_) %>%
  ggplot(aes(sl_, fill = factor(id))) + ggtitle("Alternative") + geom_density(alpha = 0.4)
ggarrange(obs, rand, ncol=2, nrow=1, legend = "none")
```




```
# calculate headings for each step
autumn_track <- autumn_track %>% rowwise() %>%
  mutate(heading = NCEP.loxodrome.na(y1_, y2_, x1_, x2_)) %>%
  ungroup() %>% add_row()
```

```
# make the columns meet Movebank's upload expectations
names(autumn_track)[c(2,4)] <-c("location-long", "location-lat") # select the end point for each step
autumn_track$timestamp<-autumn_track$t1_ # select the start time for each step
autumn_track$timestamp <- paste0(autumn_track$timestamp,".000" )
```

```
# split the data to conform to Movebank file size limits
half1 <- autumn_track[1:(0.5*nrow(autumn_track)),]
half2 <- autumn_track[(0.5*nrow(autumn_track)):nrow(autumn_track),]
```

```
# export each file for Movebank annotation
write.csv(half1, "HB1_date.csv", row.names = FALSE)
write.csv(half1, "HB2_date.csv", row.names = FALSE)
```

**Prepare data for models**

```
# read in the annotated data
half1 <- read.csv("half_track1_june_150s-4433398109146553161.csv",stringsAsFactors = F, header = T)
half2 <- read.csv("half_track2_june_150s-4573417134914709132.csv",stringsAsFactors = F, header = T)
ssfdata <- rbind(half1,half2) %>% # reformat columns
  mutate(timestamp = as.POSIXct(strptime(timestamp, format = "%Y-%m-%d %H:%M:%S"),tz = "UTC"),
         year = format(as.POSIXct(timestamp,format="%Y-%m-%d %H:%M:%S"),"%Y"),
         id_year = paste(id,year,sep="_"),
         case_ = as.numeric(case_))
```

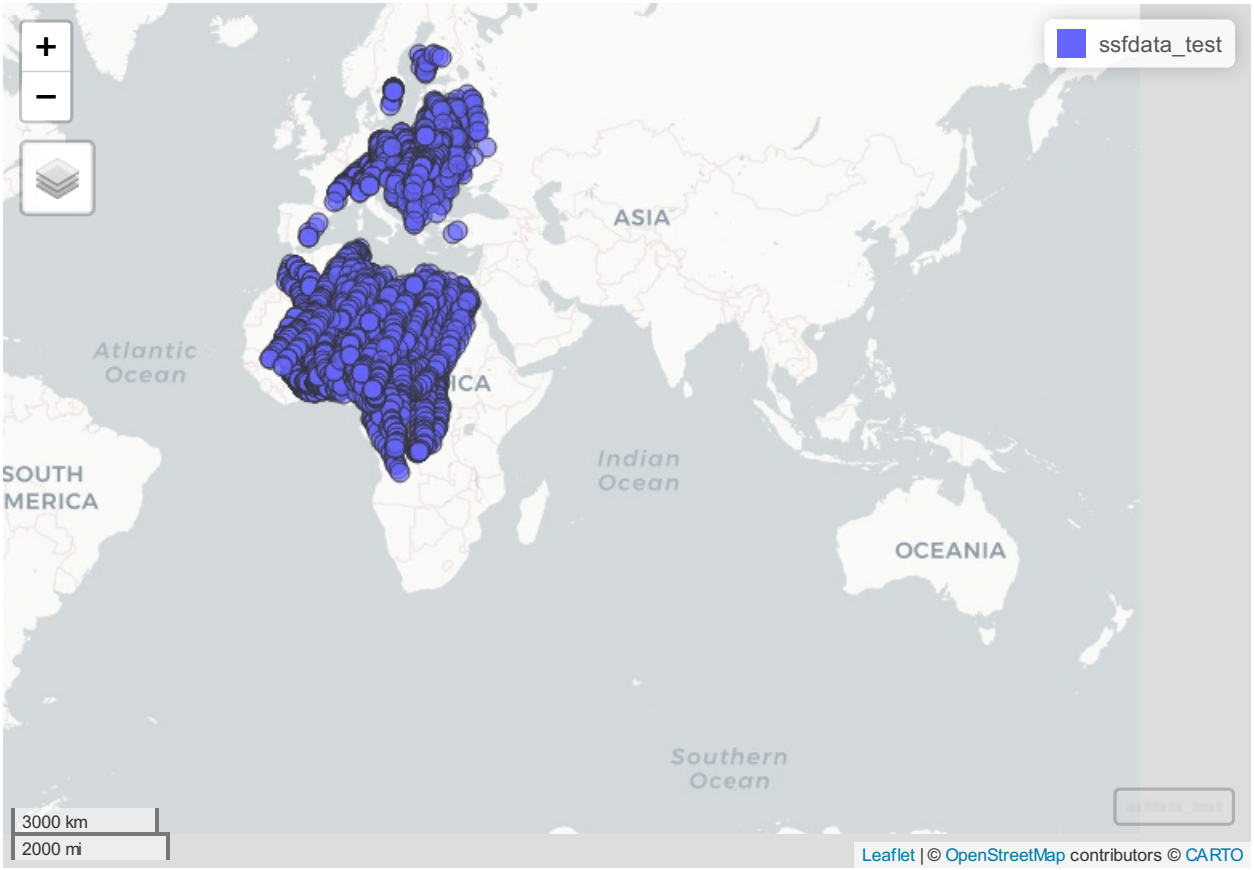
```
# check how many true steps survived filtering
print(ssfdata %>% group_by(id_year) %>% tally(case_ == 1), n= 75)
```

```
# derive the cross and tail wind components
colnames(ssfdata)[c(1,2,3,4,17,18,20,21)] <- c("lon1","lon2","lat1","lat2","u_wind","v_wind","thermal",

ssfdata$tail <- wind_support(u = ssfdata$u_wind,v = ssfdata$v_wind, heading = ssfdata$heading)
ssfdata$cross <- cross_wind(u = ssfdata$u_wind,v = ssfdata$v_wind, heading = ssfdata$heading)
```

```
# find and remove sea crossings
seacrossing_points <- !is.na(ssfdata$sea_surface) # get points over the
ssfdata$id_year_burst_step <- paste(ssfdata$id,ssfdata$year,
                                    ssfdata$burst_,ssfdata$step_id_,sep = "_") # create a new column of
seacrossings <- ssfdata[seacrossing_points == T,] # make an object of the points over seas
ssfdata <- ssfdata %>% # remove all steps containing seacrossings
  filter(!id_year_burst_step %in% seacrossings$id_year_burst_step)
```

```
# map the data removing seacrossings to double-check accuracy
ssfdata_test <- ssfdata %>% distinct(lon1, lat1, .keep_all = TRUE)
coordinates(ssfdata_test) <- ~lon1 + lat1
proj4string(ssfdata_test) <- wgs
mapView(ssfdata_test)
```




```
#drop rows with NA in thermal column and then correct for new step numbers

thermalNA <- is.na(ssfdata$thermal)
ssfdata <- ssfdata[thermalNA != T,]


# select 101 steps from each step_id_. Choosing the first 101 allows protection of True steps.

ssfdata <- ssfdata %>%
  group_by(id_year, burst_, step_id_) %>%
  slice(1:101) %>%
  ungroup()
```

```
#standardize the predictors
ssfdata <- ssfdata %>%
  mutate(scaled_cross = abs(as.numeric(scale(cross, center = T, scale = T))), # take the absoulte value
         scaled_tail = as.numeric(scale(tail, center = T, scale = T)),
         scaled_thermal = as.numeric(scale(thermal, center = T, scale = T)),)
```

**Build the model**

```
# write the model function
ssf_modelTCt <- function(df) {
  clogit(case_ ~ scaled_tail + scaled_cross + scaled_thermal + strata(step_id_), data = df)
}
```

```
#check how many true steps survived filtering
print(ssfdata %>% group_by(id_year) %>% tally(case_ == 1), n= 75)
```

```
# exclude certain data
ssf_modeling_data <- ssfdata %>% filter(id_year != "Johannes_2015" & # only three steps
                                          id_year != "Jouko_2015" & # only one step
                                          id_year != "Mohammed_2019") # incomplete record
```

```
# fit separate models for each journey, collect the coefficients, and calculate the AICs
SSF_results <- ssf_modeling_data %>%
  group_by(id_year) %>%
  nest() %>%
  mutate(ssf_modelTCt = purrr::map(data, ssf_modelTCt),
         ssf_coefsTCt = purrr::map(ssf_modelTCt, coef),
         AIC_TCt = map_dbl(ssf_modelTCt, ~AIC(.)))
```

**Extract the model coefficients for plotting**

---

```
# flatten the coefficient column
ssf_coefs <- unnest(SSF_results, ssf_coefsTCt)

#the coefficients for each variable are now in a single column, group them by their identities
tail_coefs <- do.call(rbind, lapply(unique(ssf_coefs$id_year), function(x){head(ssf_coefs[ssf_coefs$id_
cross_coefs <- do.call(rbind, lapply(unique(ssf_coefs$id_year), function(x){head(ssf_coefs[ssf_coefs$id
thermal_coefs <- do.call(rbind, lapply(unique(ssf_coefs$id_year), function(x){tail(ssf_coefs[ssf_coefs$

#create a common label for the variables
tail_coefs$var <- "Wind support"
cross_coefs$var <- "Crosswind"
thermal_coefs$var <- "Thermal uplift"


#bind them into a single object again with their identities appended
plot_data <- rbind(tail_coefs, cross_coefs,thermal_coefs)
```

---

add the life stage and migration number

---

#separate the id_year column so that stage is identified by individual

```
_____
r
plot_data
<-
separate(plot_data,
col =
"id_year",
c("id",
"year"),
remove
= F)
plot_data$stage
<- NA
#
create
an
empty
column
to
store
values
plot_data$stage[which(plot_data$id
==
"Aarne"
|
plot_data$id
==
"Annika"
|
plot_data$id
==
"Jari"
|
plot_data$id
==
"Johannes"
|
plot_data$id
==
"Jouko"
|
plot_data$id
==
"Mikko"
|
plot_data$id
==
"Paivi"
|
plot_data$id
==
"Tiina"
|
plot_data$id
==
"Kari")]
<-
"adult"
#
label
individuals
```

```
#add the number of each migration
plot_data$Migration <- NA
plot_data$Migration[which(plot_data$id_year == "Jaana_2013" | # add year 2 info
                            plot_data$id_year == "Lars_2013" |
                            plot_data$id_year == "Mohammed_2018" |
                            plot_data$id_year == "Senta_2015" |
                            plot_data$id_year == "Valentin_2015" |
                            plot_data$id_year == "Aarne_2013" |
                            plot_data$id_year == "Annika_2012" |
                            plot_data$id_year == "Jari_2013" |
                            plot_data$id_year == "Johannes_2014" |
                            plot_data$id_year == "Jouko_2012" |
                            plot_data$id_year == "Kari_2012" |
                            plot_data$id_year == "Paivi_2014" |
                            plot_data$id_year == "Mikko_2012" |
                            plot_data$id_year == "Tiina_2012")] <- "2"

plot_data$Migration[which(plot_data$id_year == "Jaana_2014" | # add year 3 info
                            plot_data$id_year == "Lars_2014" |
                            plot_data$id_year == "Senta_2016" |
                            plot_data$id_year == "Annika_2013" |
                            plot_data$id_year == "Jari_2014" |
                            plot_data$id_year == "Johannes_2015" |
                            plot_data$id_year == "Jouko_2013" |
                            plot_data$id_year == "Mikko_2013" | # missing
                            plot_data$id_year == "Paivi_2015" |
                            plot_data$id_year == "Mohammed_2019" |
                            plot_data$id_year == "Tiina_2013")] <- "3"

plot_data$Migration[which(plot_data$id_year == "Lars_2015" | # add year 4 info
                            plot_data$id_year == "Senta_2017" |
                            plot_data$id_year == "Annika_2014" |
                            plot_data$id_year == "Jouko_2014" |
                            plot_data$id_year == "Mikko_2014" |
                            plot_data$id_year == "Paivi_2016" |
                            plot_data$id_year == "Tiina_2014")] <- "4"

plot_data$Migration[which(plot_data$id_year == "Annika_2015" | # add year 5 info
                            plot_data$id_year == "Jouko_2015" |
                            plot_data$id_year == "Mikko_2015" |
                            plot_data$id_year == "Paivi_2017" |
                            plot_data$id_year == "Tiina_2015")] <- "5"
plot_data$Migration[which(is.na(plot_data$Migration))] <- "1" # everything else is year 1
```

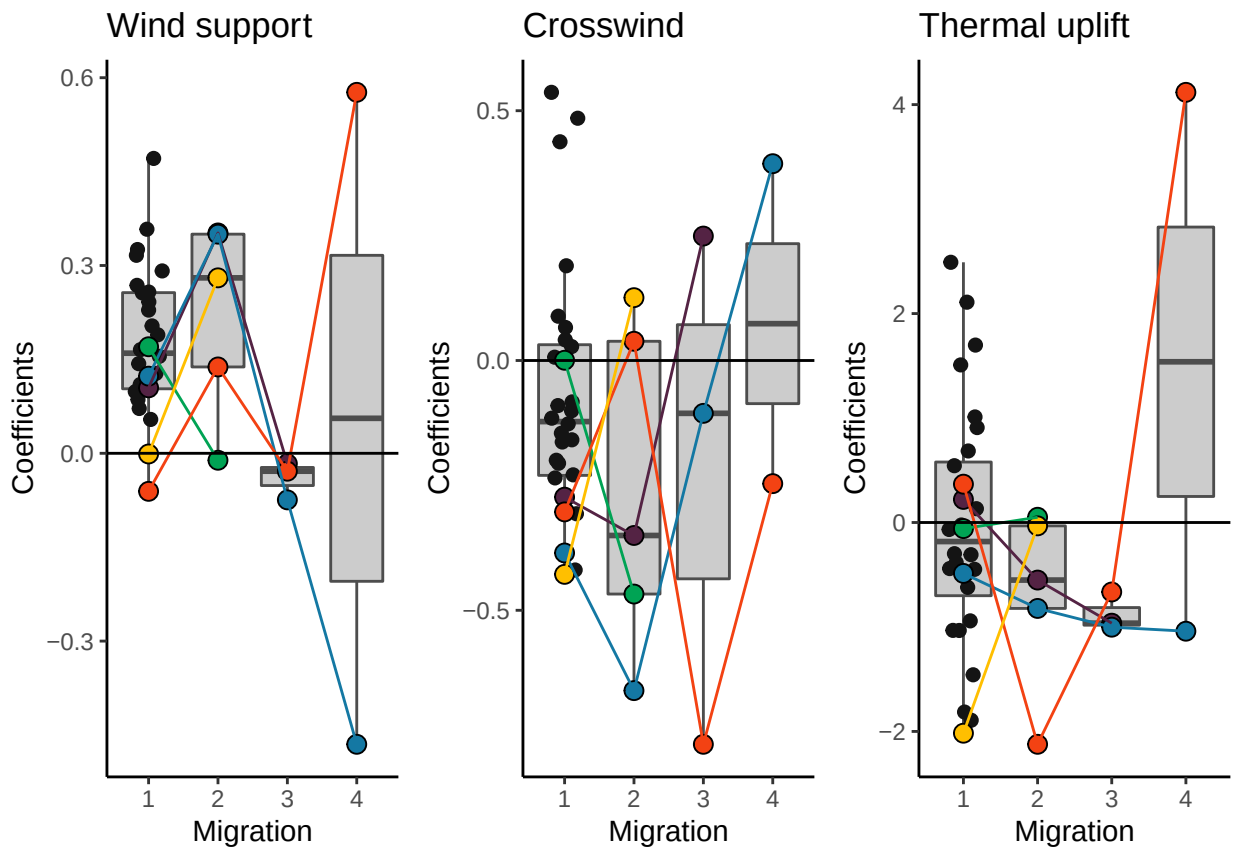
**Plot the coefficients**

```r
# select colors that are distinct
id_colors <- c("#542344","#1478A3","#00A354","#F34213","#FFBF00")


#first, select the data from the stage and variable of interest, then select the data from individuals

tail_plot_data <- plot_data %>% filter(var == "Wind support" & stage == "juvenile")
tail_plot_highlight <- tail_plot_data %>% filter(id == "Jaana" | id == "Senta" | id == "Valentin" | id =
tail_plot_data_points <- tail_plot_data %>% filter(id != "Jaana" & id != "Senta" & id != "Valentin" & id
tail_plot <- ggplot(tail_plot_data, aes(x=Migration, y=ssf_coefsTCt)) +
  geom_boxplot(outlier.shape = NA, color = "grey30", fill = "grey80") +
  geom_jitter(data = tail_plot_data_points, aes(), size=2, width=0.2, color = "grey8") +
  geom_point(data = tail_plot_highlight, aes(x=Migration, y=ssf_coefsTCt, fill = id), shape = 21, size =
  geom_line(data = tail_plot_highlight, aes(group = id, color =id)) +
  scale_fill_manual(values = id_colors) +
  scale_color_manual(values = id_colors) +
  labs(title = "Wind support", x="Migration", y = "Coefficients" ) +
  geom_hline(yintercept = 0) +
  theme_classic() +
  theme(legend.position = "none")
cross_plot_data <- plot_data %>% filter(var == "Crosswind" & stage == "juvenile")
cross_plot_highlight <- cross_plot_data %>% filter(id == "Jaana" | id == "Senta" | id == "Valentin" | id
cross_plot_data_points <- cross_plot_data %>% filter(id != "Jaana" & id != "Senta" & id != "Valentin" &
cross_plot <- ggplot(cross_plot_data, aes(x=Migration, y=ssf_coefsTCt)) +
  geom_boxplot(outlier.shape = NA, color = "grey30", fill = "grey80") +
  geom_jitter(data = cross_plot_data_points, aes(), size=2, width=0.2, color = "grey8") +
  geom_point(data = cross_plot_highlight, aes(x=Migration, y=ssf_coefsTCt, fill = id), shape = 21, size
  geom_line(data = cross_plot_highlight, aes(group = id, color =id)) +
  scale_fill_manual(values = id_colors) +
  scale_color_manual(values = id_colors) +
  labs(title = "Crosswind", x="Migration", y = "Coefficients" ) +
  geom_hline(yintercept = 0) +
  theme_classic() +
  theme(legend.position = "none")
heat_plot_data <- plot_data %>% filter(var == "Thermal uplift" & stage == "juvenile")
heat_plot_highlight <- heat_plot_data %>% filter(id == "Jaana" | id == "Senta" | id == "Valentin" | id =
heat_plot_data_points <- heat_plot_data %>% filter(id != "Jaana" & id != "Senta" & id != "Valentin" & id
heat_plot <- ggplot(heat_plot_data, aes(x=Migration, y=ssf_coefsTCt)) +
  geom_boxplot(outlier.shape = NA, color = "grey30", fill = "grey80") +
  geom_jitter(data = heat_plot_data_points, aes(), size=2, width=0.2, color = "grey8") +
  geom_point(data = heat_plot_highlight, aes(x=Migration, y=ssf_coefsTCt, fill = id), shape = 21, size =
  geom_line(data = heat_plot_highlight, aes(group = id, color =id)) +
  scale_fill_manual(values = id_colors) +
  scale_color_manual(values = id_colors) +
  labs(title = "Thermal uplift", x="Migration", y = "Coefficients" ) +
  geom_hline(yintercept = 0) +
  theme_classic() +
  theme(legend.position = "none")

ggarrange(tail_plot, cross_plot, heat_plot, ncol=3, nrow=1, legend = "none")
```


Wind support
Crosswind
Thermal uplift
Coefficients
Migration
0.6
0.3
0.0
−0.3
0.5
−0.5
4
2
−2
1
2
3
4