A comparison of statistical models used to characterize species-habitat associations with movement data: Appendix 3

Katie Florko

31/07/2023

Contents

Overview

This document contains a coding tutorial that demonstrates how to perform the analyses associated with the case study in "*A comparison of statistical models used to characterize species-habitat associations with movement data*". All analyses link the movement data of a ringed seal in eastern Hudson Bay, Canada, to modeled prey diversity.

Set-up

Load packages

```
# data wrangling
library(here)
library(tidyverse)
```

```r
library(lubridate)

# mapping
library(rnaturalearth)
library(rgdal)
library(terra)
library(sf)
library(viridis)

# fitting models
library(amt) # for selection functions
library(momentuHMM) # for hidden Markov model

# plotting
library(ggplot2)
```

Fish data

Next we load in the fish data, which contains the spatial distribution of prey diversity in 2012. This is a subset of the data from Florko et al. (2021a, 2021b).

```r
# load fish data
fish <- read.csv(here("data/preydiv.csv"))
```

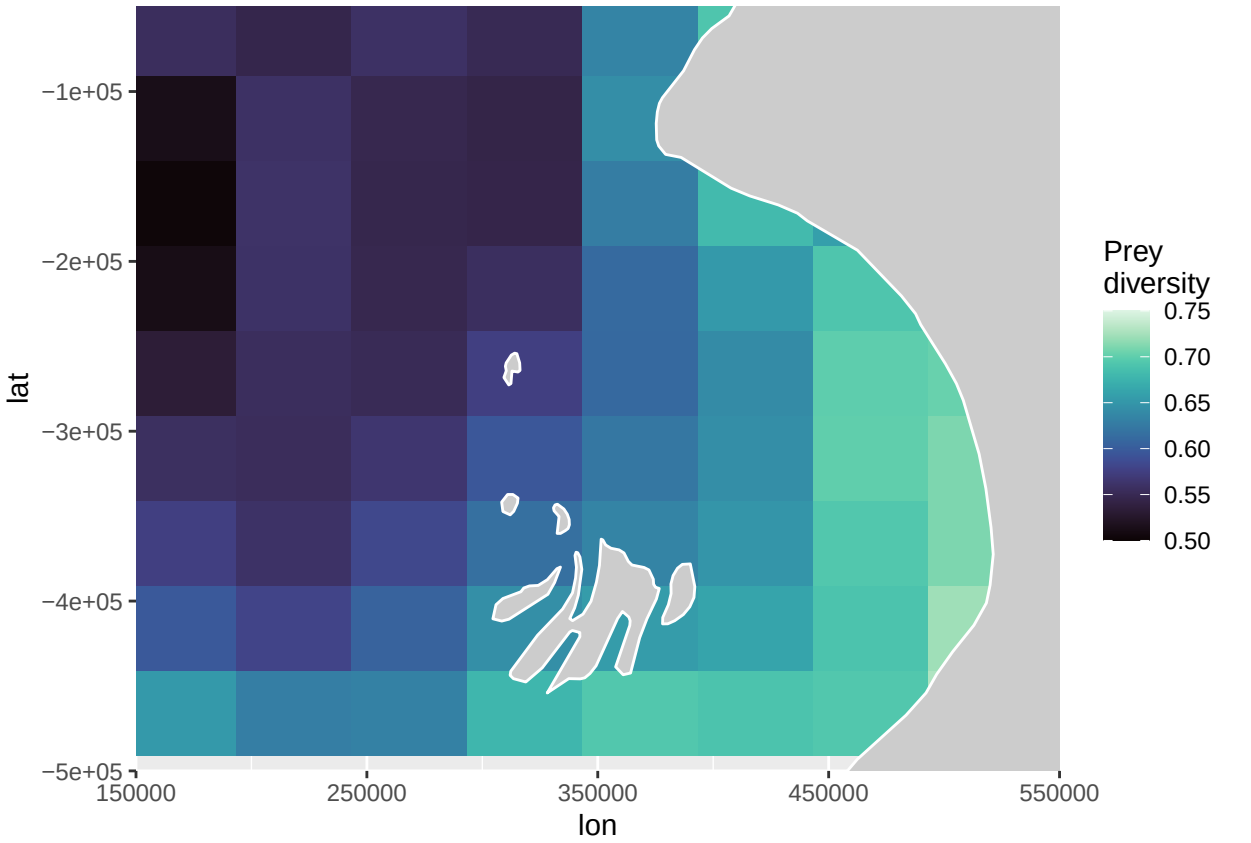
Using Visualize the fish data. *MAM: I think here you want to use sf rather than sp, sp is pretty much getting removed. The sf function for spTransform is st_transform I think. I think also that now instead of using CRS, all R functions use another system. That's why you are getting warnings. R now uses EPSG, I think the number you want is 3573 or something like that: https://epsg.io/3573 But please check.*

```r
# prepare world data for mapping
natearth <- ne_countries(scale = "medium",returnclass = "sp")
natearth <- natearth[which(natearth$continent!="Antarctica"),]
# projection: Lambert azimuthal equal area - Hudson Bay
nat_trans <- spTransform(natearth,CRS("+proj=laea +lat_0=60 +lon_0=-85 +x_0=0 +y_0=0 +datum=WGS84 +units
```

```
## Warning in spTransform(xSP, CRSobj, ...): NULL source CRS comment, falling back
## to PROJ string
```

```
## Warning in wkt(obj): CRS object has no comment
```

```r
# plot fish map
fishmap <- ggplot() +
  geom_tile(data = fish, aes(x = lon,y = lat,fill = preydiv)) +
  scale_fill_viridis(option = "mako", limits = c(0.5, 0.75), name = "Prey\ndiversity") +
  geom_polygon(data = nat_trans, aes(x=long,y=lat,group=group), fill = "grey80", color = "white") +
  coord_cartesian(xlim = c(150000,550000), ylim = c(-500000,-50000), expand = F)
fishmap
```



Seal data

Next we load in one movement track from a ringed seal equipped with an ARGOS satellite telemetry transmitter over the course of over four months in the winter of 2012-2013. This is a subset of the data from Florko et al. (2023a).

```
# load seal data
seal <- read.csv(here("data/seal_track_m.csv"))
head(seal)
```

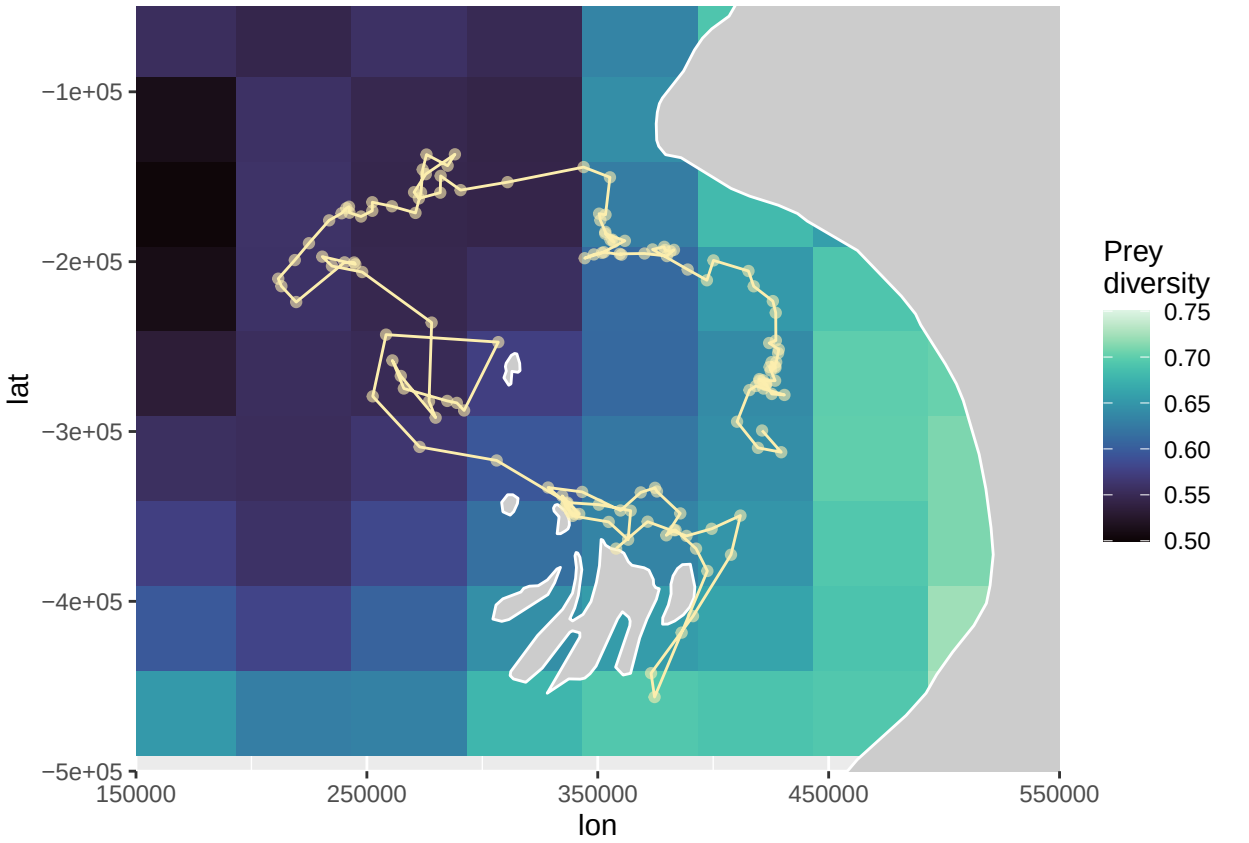
```
##        lon       lat             date       id
## 1 357940.9 -368949.6 2012-10-29 14:00 116484_1
## 2 371594.3 -353092.9 2012-10-30 14:00 116484_1
## 3 388429.2 -361478.4 2012-10-31 14:00 116484_1
## 4 399326.2 -357288.1 2012-11-01 14:00 116484_1
## 5 411788.2 -349500.7 2012-11-02 14:00 116484_1
## 6 407708.1 -372564.6 2012-11-03 14:00 116484_1
```

```
# ensure the data is in the correct format
seal <- seal %>%
  mutate(id = as.character(id),
         date = as.Date(date))
```

Visualize the seal data on top of the fish data.

```
# plot seal and fish data together
sealfishmap <- fishmap +
  geom_point(data=seal, aes(x=lon, y=lat), alpha = 0.6, color = "#FCEEAE") +
```

```
  geom_path(data=seal, aes(x=lon, y=lat), color = "#FCEEAE")

sealfishmap
```



Fit models

We will fit four models: 1) a resource selection function (RSF), 2) a step selection function (SSF) without habitat covariates in the movement kernel, 3) a SSF with a habitat covariate modifying the movement kernel, and 4) a hidden Markov model (HMM). All four of these models will include prey diversity as a covariate.

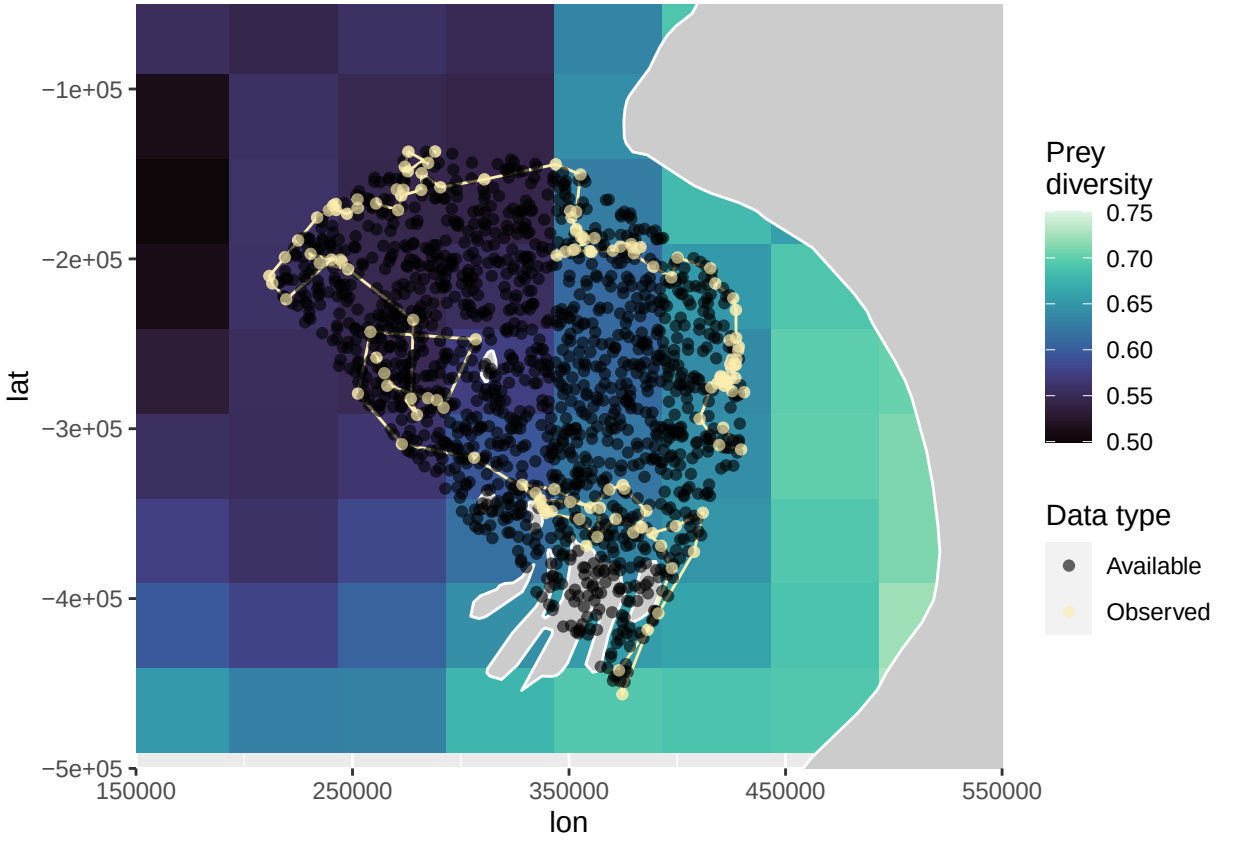
All of the step selection functions (both the RSF and two SSFs) will be fit using the `amt` package (Signer et al. 2019), while the HMM will use the functions from the `momentuHMM` package (McClintok and Michelot 2018).

RSF

The RSF is the simplest of the four models. Fitting an RSF to movement data first requires us to generate a sample of availability points and extract covariates for the used and available locations.

```
# prep data and generate availability sample
set.seed(2023)
data_rsf <- seal %>%
  make_track(lon, lat, date) %>% # convert data to track format
  random_points() # generate availability sample; default is ten times as many available points as obse

# plot used vs available locations on-top of prey diversity
```

```
data_rsf_map <- sealfishmap +
  geom_point(data=data_rsf, aes(x=x_, y=y_, color = case_), alpha = 0.6) +
  scale_color_manual(values = c("black", "#FCEEAE"),
                     label = c("Available", "Observed"), name = "Data type")
data_rsf_map
```



We can see that the availability sample is generated within the minimum convex polygon of the used samples. *MAM: if you are using the fish as a rater in all analyses, I would move that up in the set-up section. Also, why are you using WSG$ here, but LAEA above? Is there a mismatch in your projections?*

We now extract covariates for the used and available locations.

```
# rasterize and extract prey diversity covariate
fish_raster <- terra::rast(fish, crs = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")

data_rsf <- data_rsf %>%
  extract_covariates(fish_raster)
```

Next, we will fit the model. The response, `case_`, is used or available location, and `preydiv` is the covariate for prey diversity.

```
# fit rsf (a binomial logistic regression)
rsf1 <- data_rsf %>%
  amt::fit_rsf(case_ ~ preydiv, model = TRUE)
```

View the summary.

```r
# see model summary
summary(rsf1)
```

```
##
## Call:
## stats::glm(formula = formula, family = stats::binomial(link = "logit"),
##     data = data, model = TRUE)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5753  -0.4729  -0.4294  -0.3754   2.3363
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -6.118      1.400  -4.369 1.25e-05 ***
## preydiv        6.353      2.312   2.748    0.006 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 938.28  on 1539  degrees of freedom
## Residual deviance: 930.60  on 1538  degrees of freedom
## AIC: 934.6
##
## Number of Fisher Scoring iterations: 5
```

We see that prey diversity is a significant positive covariate. We do not interpret the intercept, since it is not ecologically meaningful in a RSF. See Fieberg et al. (2021) for a detailed discussion on how to interpret parameters.

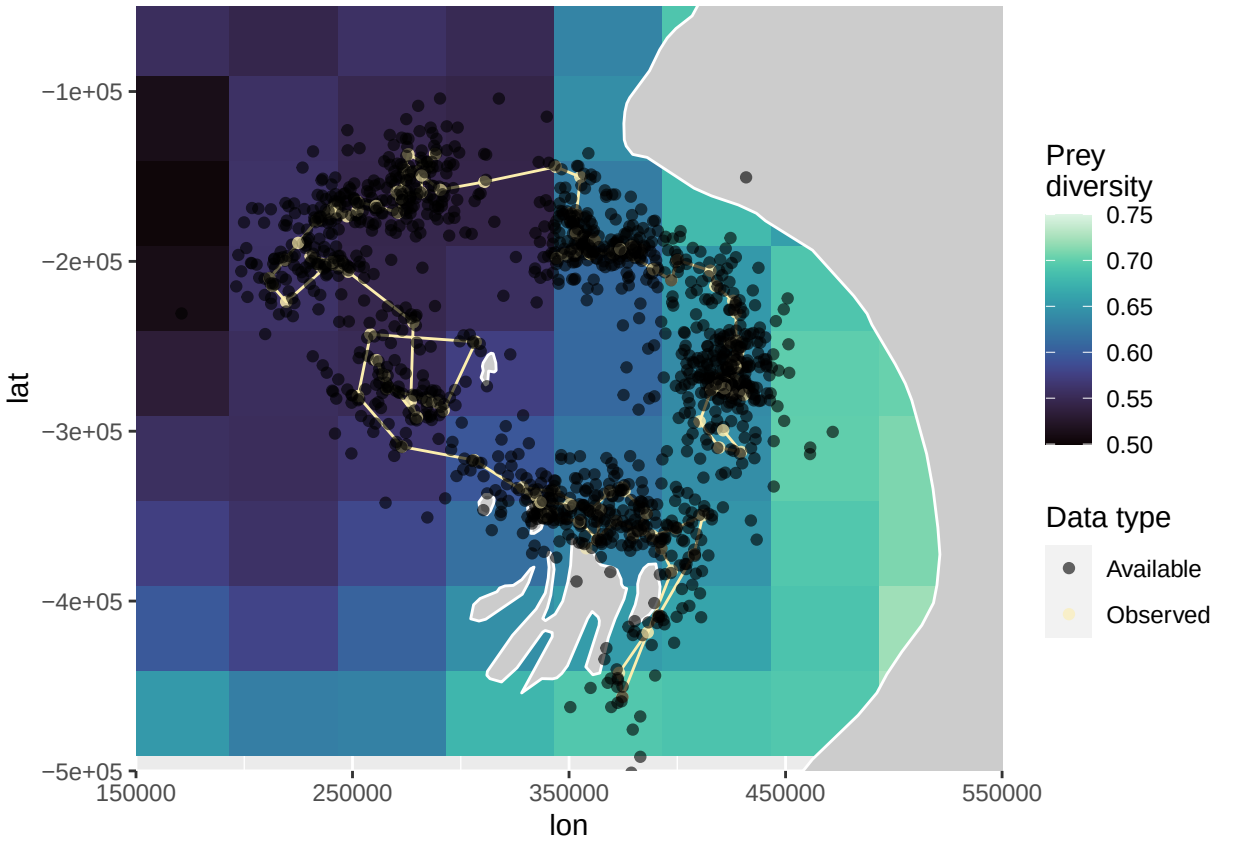
SSF

Next we will fit our two SSFs. The workflow is similar to that of the RSF, however, the availability sample is generated differently. We transform the seal locations into a track format, then convert the track data into step format (i.e., with a start and an end), and then generate the availability sample.

```r
# prep data and generate availability sample
set.seed(2023)
data_ssf <- seal %>%
  make_track(lon, lat, date) %>% #  convert data to track format
  steps() %>% # convert track data to step format (i.e., with a start and an end)
  random_steps()  # generate availability sample
```

It always helps to visualized the sample created.

```r
# plot used vs available locations on-top of prey diversity
data_ssf_map <- sealfishmap +
  geom_point(data=data_ssf, aes(x=x2_, y=y2_, color = case_), alpha = 0.6) +
  scale_color_manual(values = c("black", "#FCEEAE"),
                     label = c("Available", "Observed"), name = "Data type")
data_ssf_map
```



We can see that the availability sample is generated at each step and is not restricted to the minimum convex polygon.

We now extract prey diversity values for the used and available locations.

```
# extract prey diversity covariate
data_ssf <- data_ssf %>%
  extract_covariates(fish_raster, where = "end") #sample at end of step
```

We will transform both the step length and turning angle using log and cosine transformations, respectively.

```
# transform movement covariates
data_ssf <- data_ssf %>%
  mutate(cos_ta = cos(ta_),
         log_sl = log(sl_))
```

Next, we will fit the models. The response, `case_`, is used or available location, and `preydiv` is the covariate for prey diversity. `log_sl` is the log transformation of step length, and `cos_ta` is the cosine transformation of turning angle. `strata(step_id_)` specifies that the this is a *conditional* logistic regression that groups data by step identification number.

We are fitting two SSFs, one without a movement-related covariate (called ssf1), and one with a movement-related covariate (called ssf2).

```
# fit ssfs
## ssf1: ssf without covariate affecting movement kernel
ssf1 <- data_ssf %>%
  fit_clogit(case_ ~ preydiv + log_sl + cos_ta + strata(step_id_), model = TRUE)
```

```
## ssf2: ssf with covariate affecting movement kernel
ssf2 <- data_ssf %>%
  fit_clogit(case_ ~ preydiv*log_sl + cos_ta + strata(step_id_), model = TRUE)
```

First we will interpret ssf1.

```
# see model summary
summary(ssf1)
```

```
## Call:
## coxph(formula = Surv(rep(1, 1518L), case_) ~ preydiv + log_sl +
##     cos_ta + strata(step_id_), data = data, model = TRUE, method = "exact")
##
##   n= 1516, number of events= 138
##    (2 observations deleted due to missingness)
##
##              coef exp(coef)  se(coef)      z Pr(>|z|)
## preydiv  0.957799  2.605953  6.772711  0.141    0.888
## log_sl  -0.008404  0.991631  0.083328 -0.101    0.920
## cos_ta   0.031707  1.032215  0.130643  0.243    0.808
##
##         exp(coef) exp(-coef) lower .95 upper .95
## preydiv    2.6060     0.3837 4.477e-06 1.517e+06
## log_sl     0.9916     1.0084 8.422e-01 1.168e+00
## cos_ta     1.0322     0.9688 7.990e-01 1.333e+00
##
## Concordance= 0.494  (se = 0.029 )
## Likelihood ratio test= 0.09  on 3 df,   p=1
## Wald test            = 0.09  on 3 df,   p=1
## Score (logrank) test = 0.09  on 3 df,   p=1
```

This model estimates the coefficients (coef) as well as the exponent of the coefficient (exp(coef)). The coefficients are as in our regular logistic regression (the RSF), and the exp(coef) quantifies the relative intensity of use of two locations that differ by 1 unit of prey diversity but are otherwise the same. The model suggests our seal would be 2.6 times more likely to choose a location with 1 unit higher prey diversity (see Fieberg et al. 2021). However, this is not significant ($p = 0.888$), and we see that the scale of prey diversity is much smaller (range = 0.5-0.75), and thus an increase in of 2.6 times per 1 unit prey diversity is not a meaningful increase.

We don't interpret the values of log_sl or cos_ta, since we don't expect those to affect occurrence since the availability sample is generated using the step length and turning angle from the observed track.

We also see that we get a warning "2 observations deleted due to missingness", due to two of our available locations being found on land, where we do not have fish values. This could be mitigated by generating more than 10 (i.e., 15) available locations per used location, omitting the samples without prey diversity values, and then randomly selecting 10 available locations per used location of those that have values for prey diversity.

Next we will interpret ssf2.

```
# see model summary
summary(ssf2)
```

```
## Call:
## coxph(formula = Surv(rep(1, 1518L), case_) ~ preydiv * log_sl +
##     cos_ta + strata(step_id_), data = data, model = TRUE, method = "exact")
##
```

```
##   n= 1516, number of events= 138
##    (2 observations deleted due to missingness)
##
##                       coef  exp(coef)   se(coef)      z Pr(>|z|)
## preydiv         2.791e+01  1.318e+12  2.177e+01  1.282    0.200
## log_sl          1.663e+00  5.273e+00  1.286e+00  1.293    0.196
## cos_ta          2.338e-02  1.024e+00  1.311e-01  0.178    0.858
## preydiv:log_sl -2.744e+00  6.433e-02  2.101e+00 -1.306    0.192
##
##                 exp(coef) exp(-coef) lower .95 upper .95
## preydiv         1.318e+12  7.589e-13 3.919e-07 4.430e+30
## log_sl          5.273e+00  1.897e-01 4.241e-01 6.556e+01
## cos_ta          1.024e+00  9.769e-01 7.917e-01 1.324e+00
## preydiv:log_sl 6.433e-02  1.555e+01 1.046e-03 3.955e+00
##
## Concordance= 0.555  (se = 0.029 )
## Likelihood ratio test= 1.81  on 4 df,   p=0.8
## Wald test            = 1.81  on 4 df,   p=0.8
## Score (logrank) test = 1.81  on 4 df,   p=0.8
```

Similar to ssf1, we don't see any significant relationships. Note that we also included a term for preydiv*log_sl. Since we have this interaction, preydiv and log_sl can not be interpreted indepently, thus we interpret the interaction. The interaction is not signficant, so this model does not suggest that prey diversity affects the movement speed of our ringed seal.

HMM

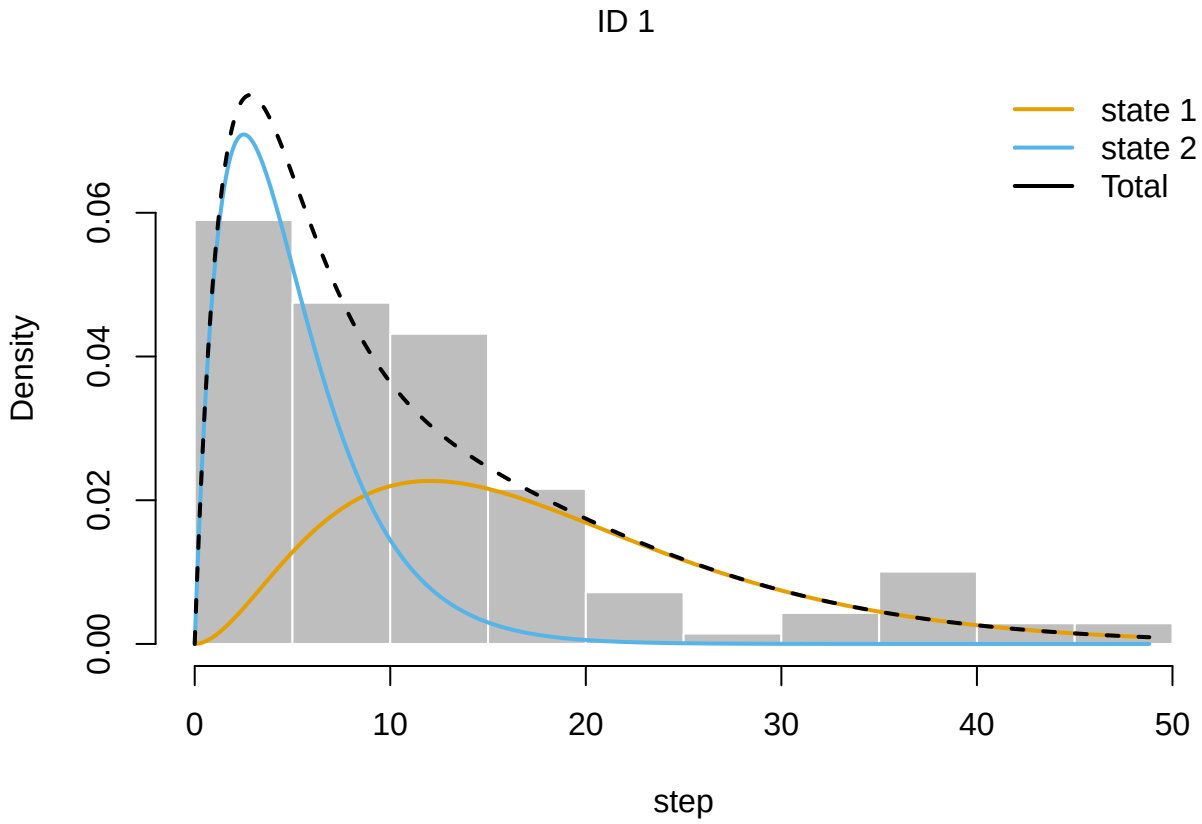
Finally, we will fit the HMM using momentuHMM (McClintock and Michelot, 2018). In preparation, we define initial parameters, and then update the parameters using our fit model, to ultimately fit a more refined model.

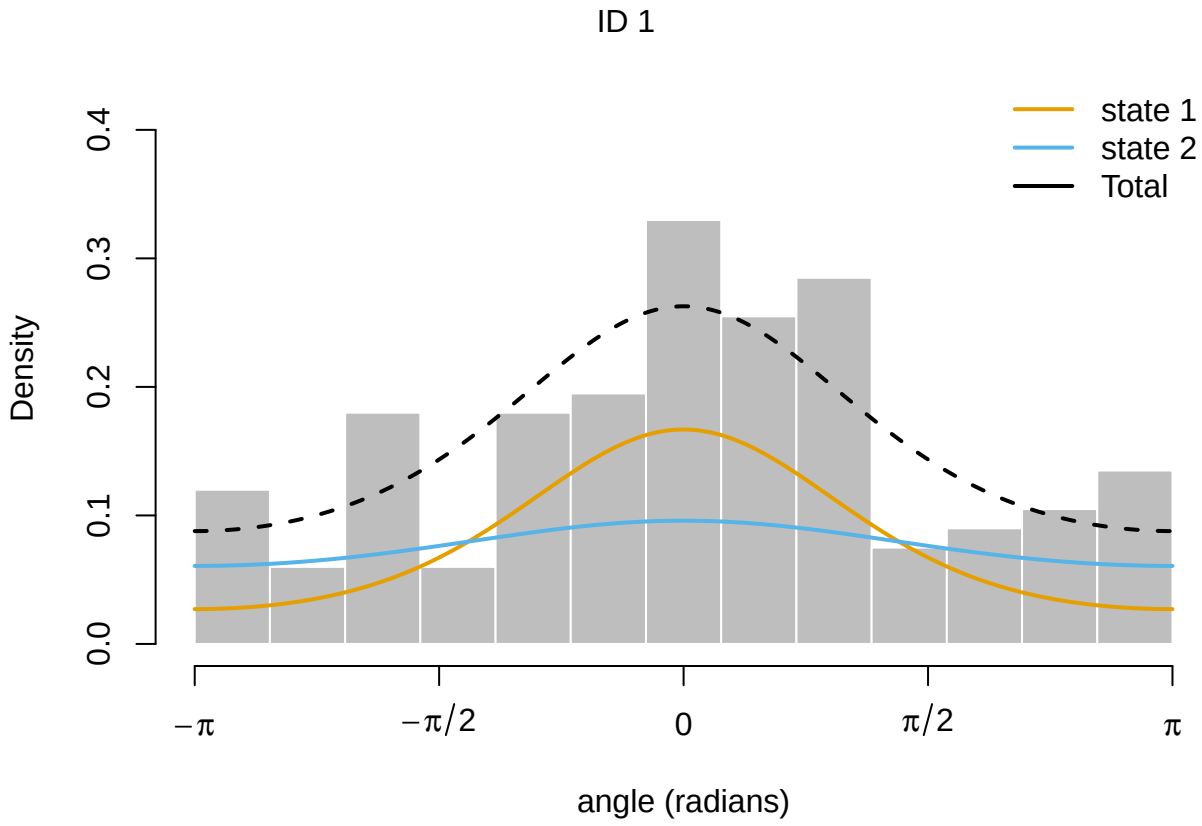
MAM: you need to break this down and explain the steps. I would include the two-state HMM that you fitted and show how you decided that it should be 3 states. Also, really you'd want to have it in km in the first time. The HMM is mostly dividing the state based on speed, so km would make more sense. This also means that the SSF would make more sense in KM. I would do the KM transformation above (I thought that's what you were doing with you projection, and do all analysis in km).

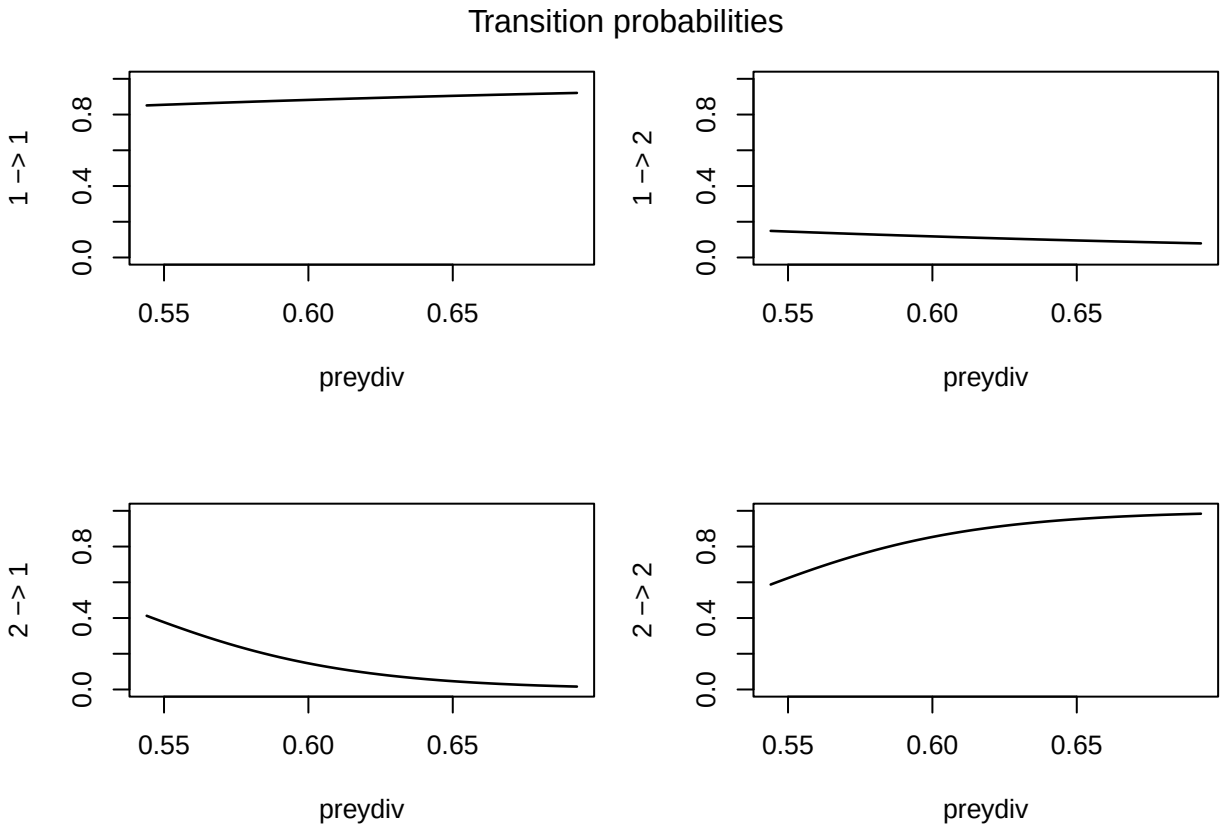
```
## prep data in lat/lon and refit model (so step length is in kilometers)
data_hmm <- seal %>%
  make_track(lon, lat, date) %>%
  extract_covariates(fish_raster) %>%
    mutate(ID = 1,
          x = x_,
          y = y_,
          date = t_) %>%
  dplyr::select(ID, x, y, date, preydiv) %>%
  as.data.frame() %>%
  st_as_sf(coords = c("x", "y")) %>%
  st_set_crs("+proj=laea +lat_0=60 +lon_0=-85 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_defs +ellps=WGS84
  st_transform("+proj=longlat +datum=WGS84") %>%
  mutate(long = unlist(map(geometry,1)),
          lat = unlist(map(geometry,2))) %>%
  st_drop_geometry()

# do momentuhmm which calculates step length in KM
```

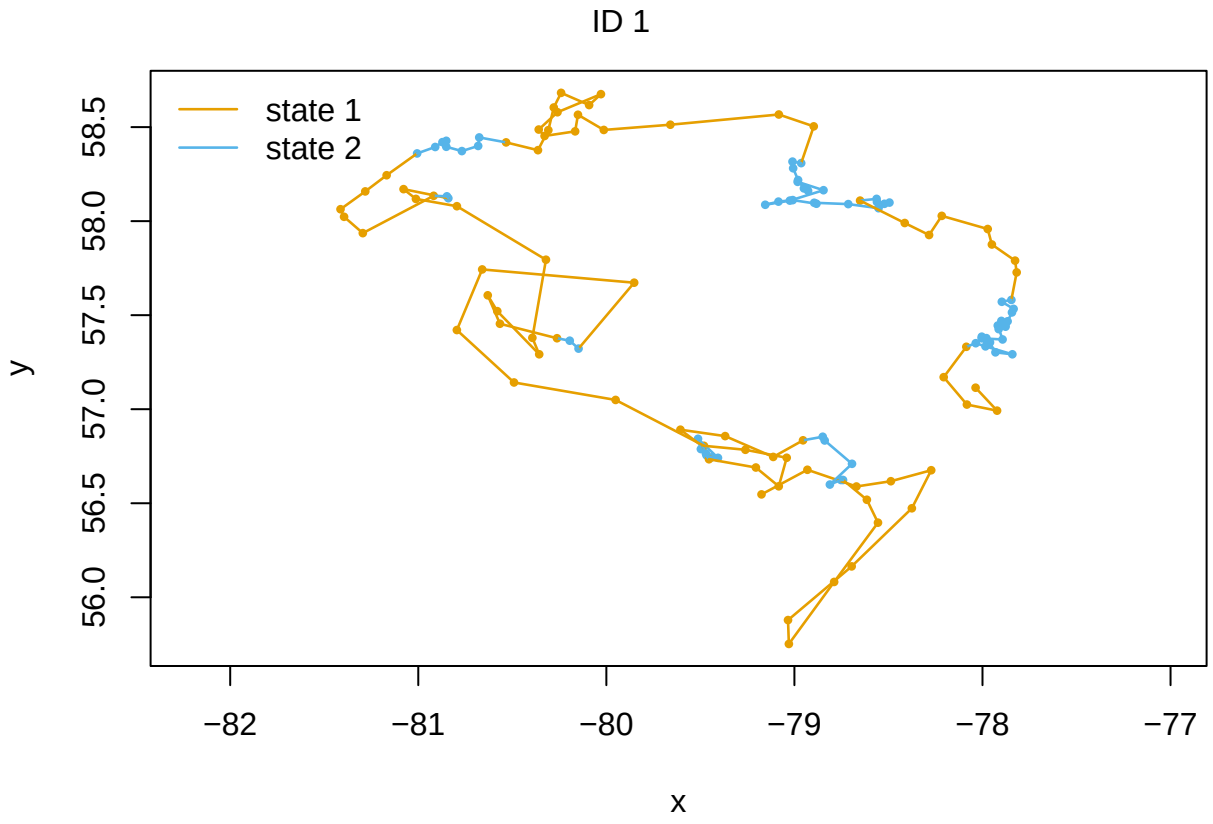
```r
data_hmm <- momentuHMM::prepData(data_hmm, coordNames = c("long", "lat"), type = "LL", covNames = c("pr

# first let us try fitting a two-state HMM
# define parameters
nbStates <- 2 # number of states
stepDist <- "gamma" # step distribution
angleDist <- "vm" # turning angle distribution

mu0 <- c(5, 38) # mean step length for each state
sigma0 <- c(3, 8) # sd step length for each state
stepPar0 <- c(mu0, sigma0)
kappa0 <- c(0.35, 0.5)  # turning angle for each state

formula = ~ preydiv # identify covariates

# fit HMM (with step in km)
set.seed(2023)
hmm1_km <- momentuHMM::fitHMM(data=data_hmm, nbStates=nbStates,
            dist=list(step=stepDist,angle=angleDist),
            Par0=list(step=stepPar0,angle=kappa0))

# retrieve parameters to refine model
Par0_hmm1_km <- momentuHMM::getPar0(hmm1_km, formula=formula)

# fit a refined HMM with parameters from hmm1_km
set.seed(2023)
hmm2_km <- momentuHMM::fitHMM(data=data_hmm, nbStates=2,
            dist=list(step=stepDist,angle=angleDist),
            Par0=Par0_hmm1_km$Par,
            delta0 = Par0_hmm1_km$delta,
            beta0 = Par0_hmm1_km$beta,
            formula=formula)
plot(hmm2_km)
```

```
## Decoding state sequence... DONE
```


ID 1
state 1
state 2
Total
Density
0.00
0.02
0.04
0.06
0
10
20
30
40
50
step


ID 1
state 1
state 2
Total
Density
0.0
0.1
0.2
0.3
0.4
−π
−π/2
0
π/2
π
angle (radians)

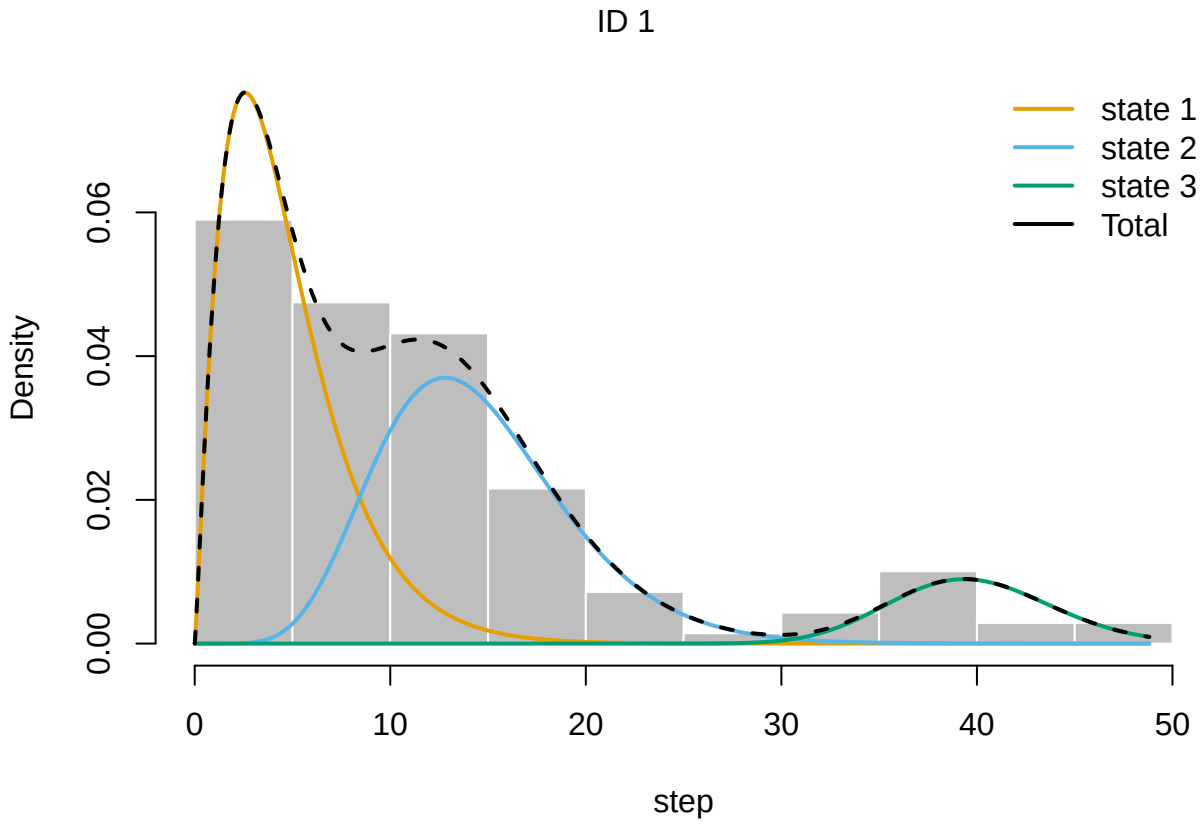

Transition probabilities
1 -> 1
2 -> 1
1 -> 2
2 -> 2
0.0
0.4
0.8
0.55
0.60
0.65
preydiv

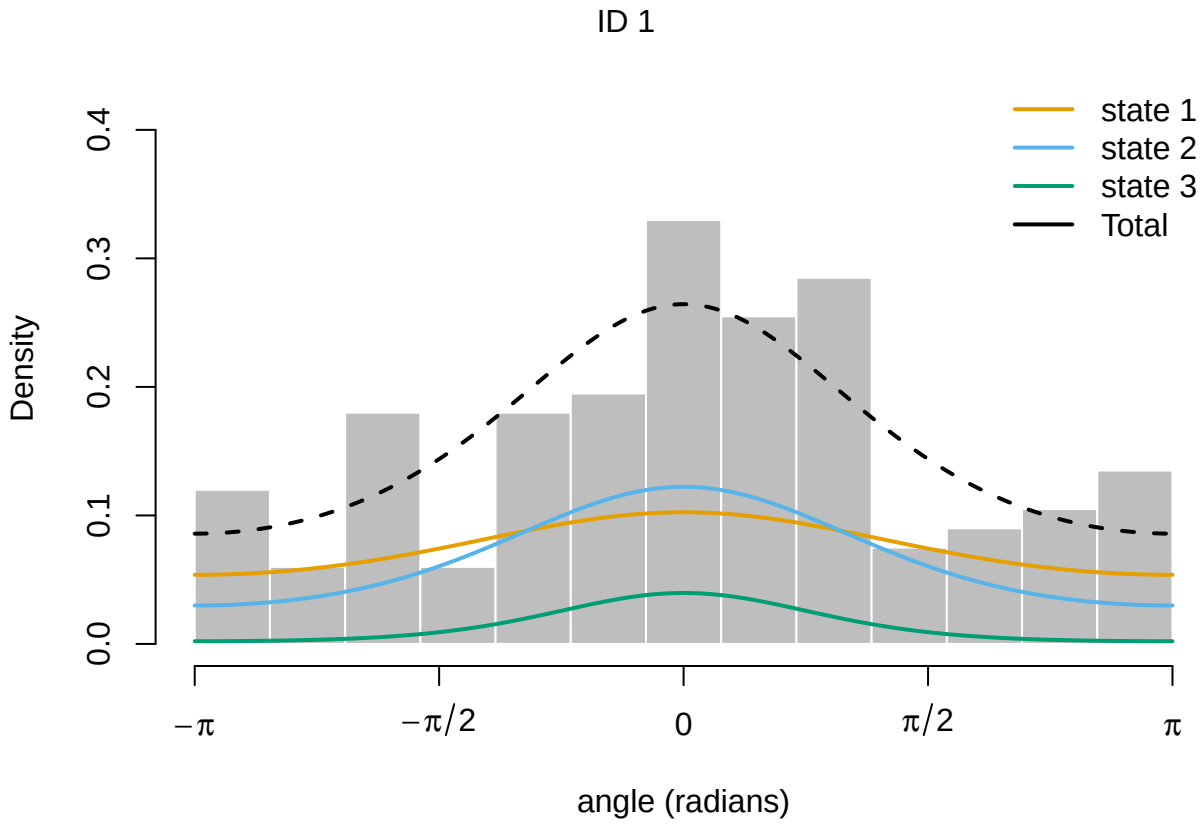


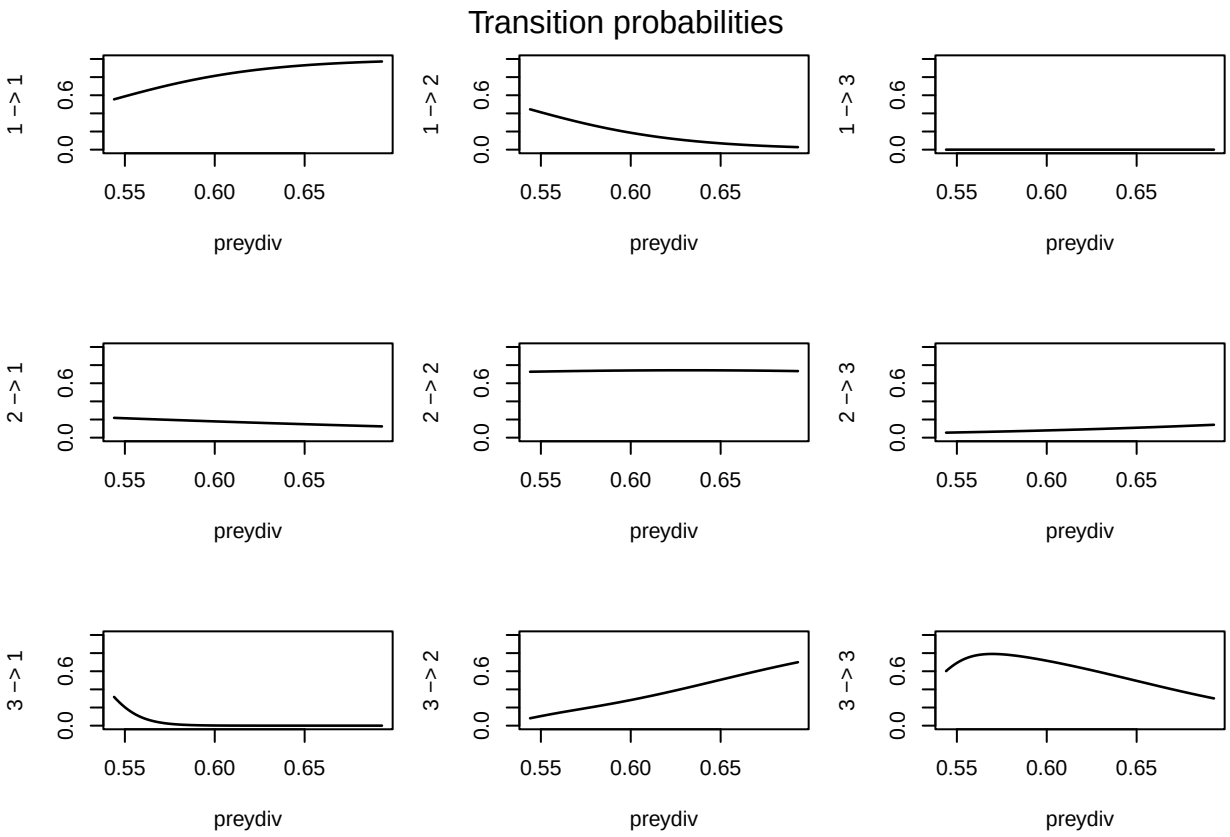

```
# we can see from the step-length histogram and the map that it doesn't look like this HMM is capturing


# let us try fitting a three-state HMM

# define parameters
nbStates <- 3 # number of states
stepDist <- "gamma" # step distribution
angleDist <- "vm" # turning angle distribution

mu0 <- c(5, 12, 38) # mean step length for each state
sigma0 <- c(3, 5, 8) # sd step length for each state
stepPar0 <- c(mu0, sigma0)
kappa0 <- c(0.35, 0.55, 0.5)  # turning angle for each state

formula = ~ preydiv # identify covariates

# fit HMM (with step in km)
set.seed(2023)
hmm3_km <- momentuHMM::fitHMM(data=data_hmm, nbStates=nbStates,
            dist=list(step=stepDist,angle=angleDist),
            Par0=list(step=stepPar0,angle=kappa0))

# retrieve parameters to refine model
Par0_hmm3_km <- momentuHMM::getPar0(hmm3_km, formula=formula)
```

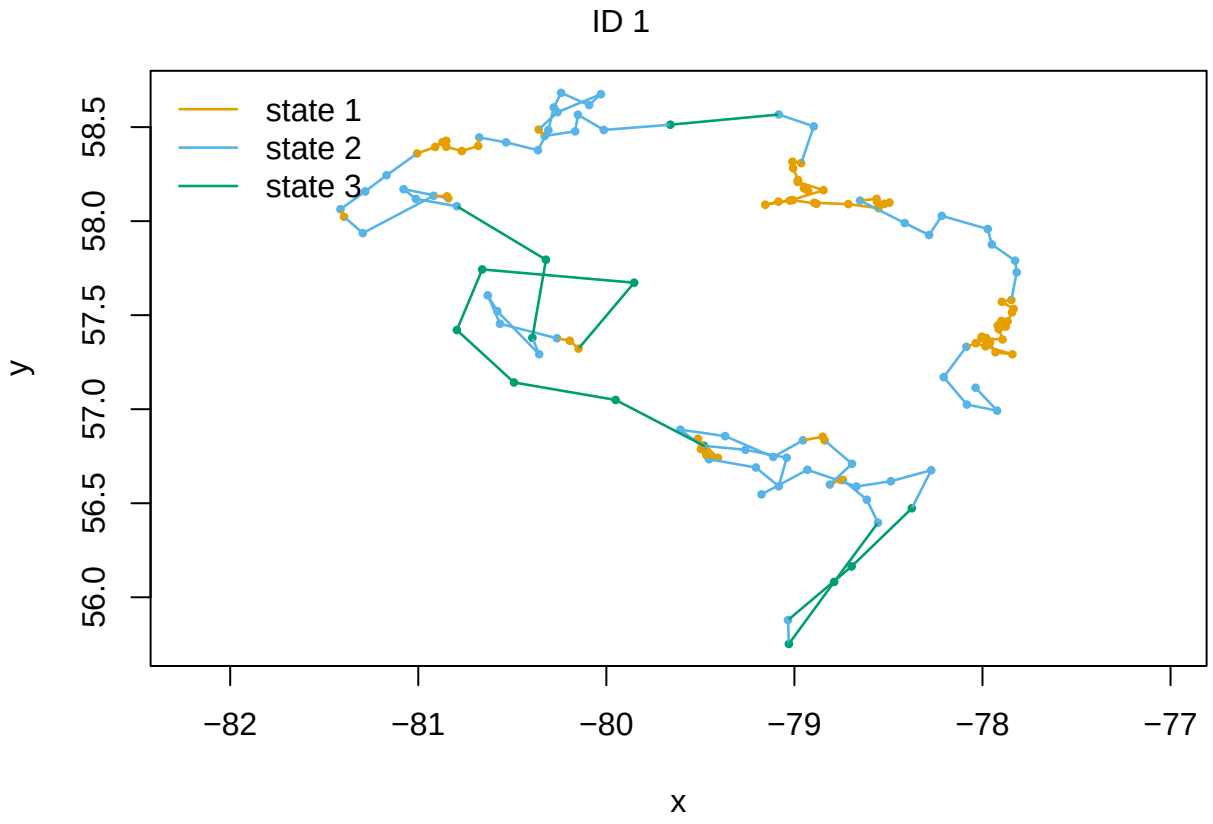
```r
# fit a refined HMM with parameters from hmm1_km
set.seed(2023)
hmm4_km <- momentuHMM::fitHMM(data=data_hmm, nbStates=3,
             dist=list(step=stepDist,angle=angleDist),
             Par0=Par0_hmm3_km$Par,
             delta0 = Par0_hmm3_km$delta,
             beta0 = Par0_hmm3_km$beta,
             formula=formula)

plot(hmm4_km)
```

```
## Decoding state sequence... DONE
```




ID 1
state 1
state 2
state 3
Total
Density
0.0
0.1
0.2
0.3
0.4
−π
−π/2
0
π/2
π
angle (radians)


Transition probabilities
1 -> 1
1 -> 2
1 -> 3
2 -> 1
2 -> 2
2 -> 3
3 -> 1
3 -> 2
3 -> 3
0.0
0.6
0.55
0.60
0.65
preydiv



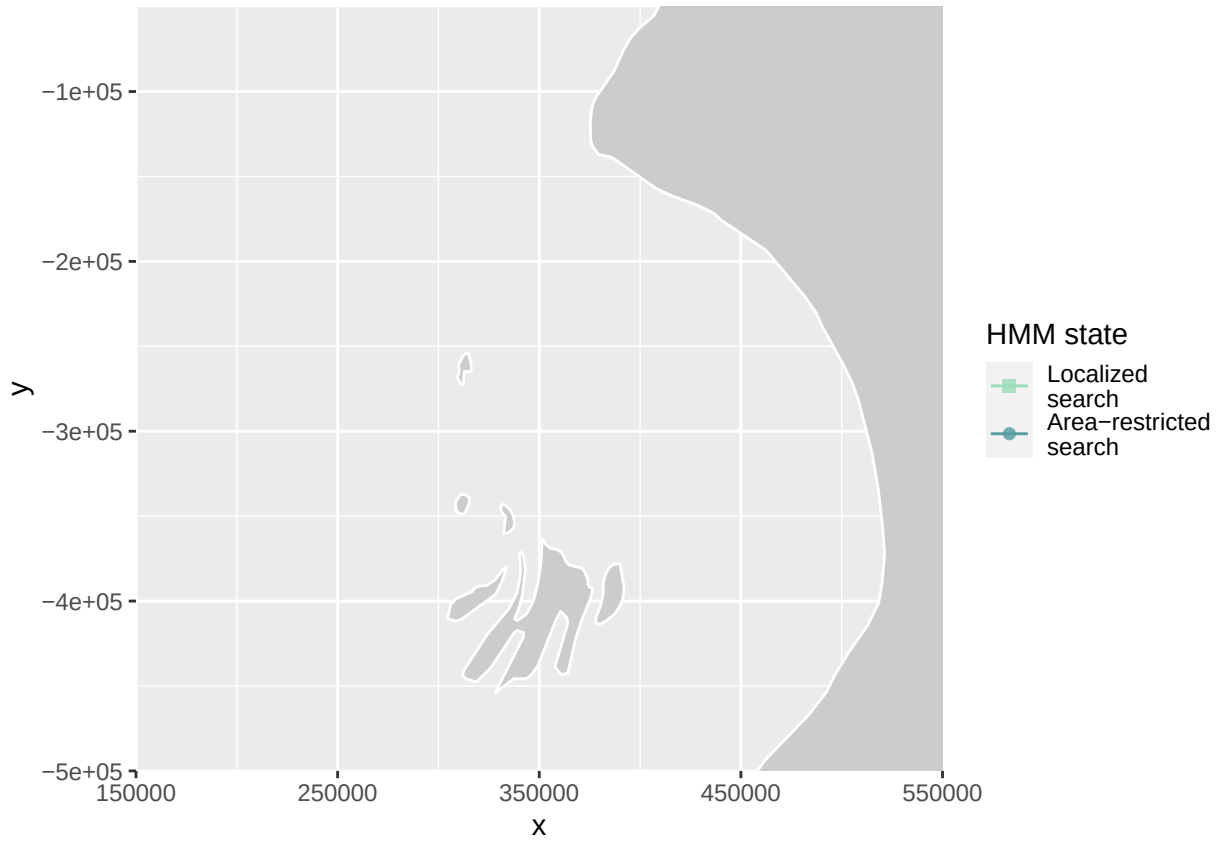

```
# Much better! We can see that those long step-lengths are captured in state 3, and the map of the deco

# add the state estimate from the HMM
data_hmm$state <- as.factor(momentuHMM::viterbi(hmm4_km))


# view the regression coeffients for the transition probabilities
print(hmm2_km)
```

```
## Value of the maximum log-likelihood: -700.7697
##
##
## step parameters:
## ----------------
##       state 1  state 2
## mean 18.29472 5.038059
## sd   10.67237 3.568961
##
## angle parameters:
## -----------------
##                  state 1   state 2
## mean          0.0000000 0.0000000
## concentration 0.9087117 0.2285965
##
## Regression coeffs for the transition probabilities:
## ---------------------------------------------------
```

```
##                 1 -> 2    2 -> 1
## (Intercept)  0.8567755  13.34018
## preydiv     -4.7800885 -25.17205
##
## Transition probability matrix (based on mean covariate values):
## ---------------------------------------------------------------
##           state 1   state 2
## state 1 0.8845774 0.1154226
## state 2 0.1305702 0.8694298
##
## Initial distribution:
## ---------------------
##      state 1      state 2
## 9.999998e-01 2.380845e-07
```

We plot the decoded states (estimated behaviours).

```
# plot decoded states
data_hmm$state <- as.factor(momentuHMM::viterbi(hmm2_km))
hmmstate_plot <- ggplot() +
  scale_fill_viridis(option = "mako", limits = c(0.5, 0.75), name = "Prey\ndiversity") +
  geom_path(data=data_hmm, aes(x=x, y=y, color = state, group =ID)) +
  geom_point(data=data_hmm, aes(x=x, y=y, color = state, shape = state), size=2, alpha = 0.8) +
  scale_color_manual(values = c("#99DDB6", "#539D9C", "#312C66"),
                     labels = c("Localized\nsearch", "Area-restricted\nsearch", "Travelling"),
                     name = "HMM state") +
  scale_shape_manual(values = c(15, 16, 17),
                     labels=c("Localized\nsearch", "Area-restricted\nsearch", "Travelling"),
                     name = "HMM state") +
  geom_polygon(data = nat_trans, aes(x=long,y=lat,group=group), fill = "grey80", color = "white") +
  coord_cartesian(xlim = c(150000,550000), ylim = c(-500000,-50000), expand = F)
hmmstate_plot
```



Plot a histogram of step lengths for each state. In order to do this, we need to transform the movement data to UTM and refit our HMM in order to get step length in a meaningful unit (i.e., kilometers).

MAM: see comment above re km. Also, this needs to be broken down and explained. You need to talk about what you are doing, the arguments of the main functions, and interpret the results. As for main text, please change the name of the 3 behaviours. Also, it may be more straightforward if for some of these figures you just use the functions from momentuHMM, rather than you own plot functions (less code). I don't think it matters if the colours don't match the paper, and you can refer the reader to another script with the details to recreate your exact figures.

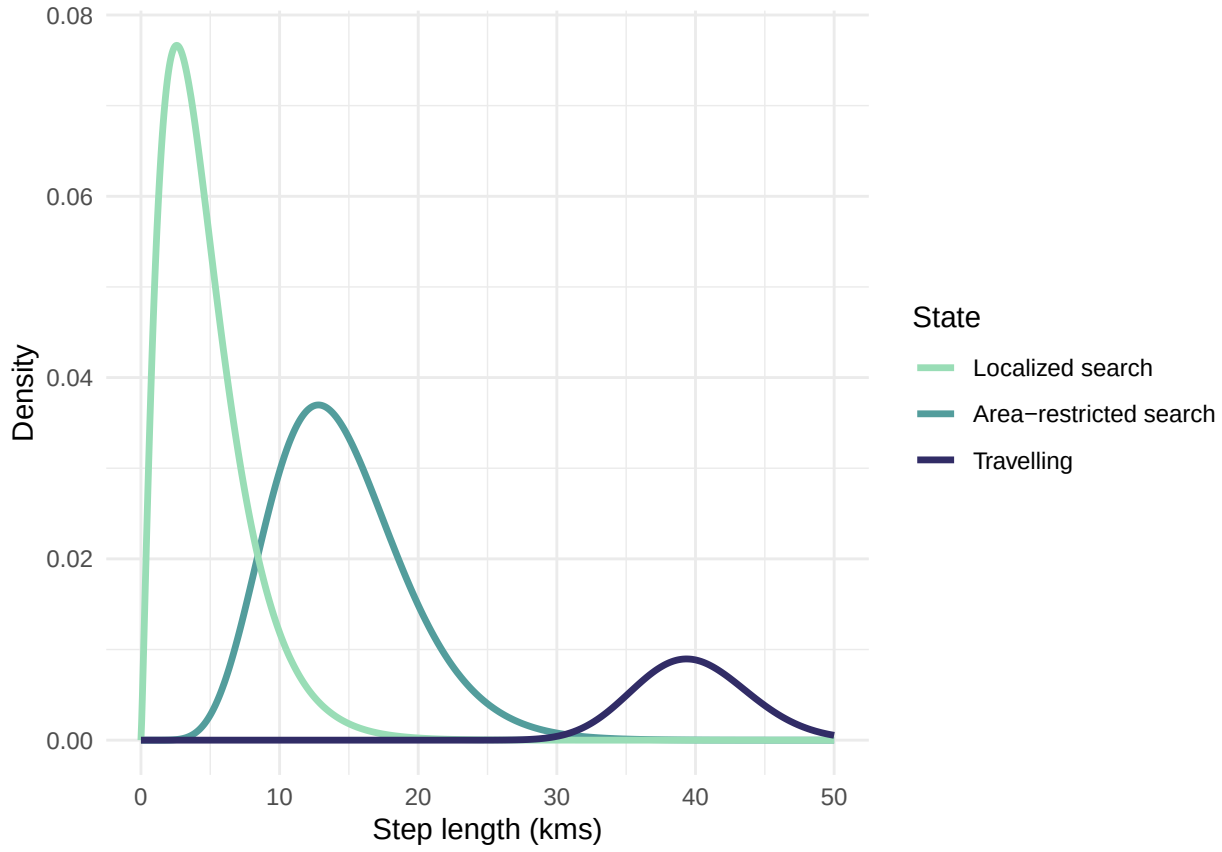
```
## prep data in lat/lon and refit model (so step length is in kilometers)
data_hmm <- seal %>%
  make_track(lon, lat, date) %>%
  extract_covariates(fish_raster) %>%
    mutate(ID = 1,
          x = x_,
          y = y_,
          date = t_) %>%
  dplyr::select(ID, x, y, date, preydiv) %>%
  as.data.frame() %>%
  st_as_sf(coords = c("x", "y")) %>%
  st_set_crs("+proj=laea +lat_0=60 +lon_0=-85 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_defs +ellps=WGS84
  st_transform("+proj=longlat +datum=WGS84") %>%
  mutate(long = unlist(map(geometry,1)),
           lat = unlist(map(geometry,2))) %>%
  st_drop_geometry()
```

```r
# do momentuhmm which calculates step length in KM
data_hmm <- momentuHMM::prepData(data_hmm, coordNames = c("long", "lat"), type = "LL", covNames = c("pr

# define parameters
nbStates <- 3
stepDist <- "gamma" # step distribution
angleDist <- "vm" # turning angle distribution

mu0 <- c(5, 12, 38)
sigma0 <- c(3, 5, 8)
stepPar0 <- c(mu0, sigma0)
kappa0 <- c(0.35, 0.55, 0.5)

# fit HMM (with step in km)
set.seed(2023)
hmm1_km <- momentuHMM::fitHMM(data=data_hmm, nbStates=nbStates,
            dist=list(step=stepDist,angle=angleDist),
            Par0=list(step=stepPar0,angle=kappa0))

# get parameters
Par0_hmm1_km <- momentuHMM::getPar0(hmm1_km, formula=formula)

# fit HMM with parameters from hmm1_km
set.seed(2023)
hmm2_km <- momentuHMM::fitHMM(data=data_hmm, nbStates=3,
            dist=list(step=stepDist,angle=angleDist),
            Par0=Par0_hmm1_km$Par,
            delta0 = Par0_hmm1_km$delta,
            beta0 = Par0_hmm1_km$beta,
            formula=formula)

# add the state estimate from the HMM
data_hmm$state <- as.factor(momentuHMM::viterbi(hmm2_km))

# calculate frequencies of states
v <- momentuHMM::viterbi(hmm2_km)
stateFreq <- table(v) / length(v)

# plot colours
colours.states <- c("#99DDB6", "#539D9C", "#312C66")

# generate sequence for x axis of density functions
x <- seq(0, 50, length=1000)

# get converged mean and sd for each state
meanARS <- hmm2_km$mle$step[1,1]
sdARS <- hmm2_km$mle$step[2,1]

meanCR <- hmm2_km$mle$step[1,2]
sdCR <- hmm2_km$mle$step[2,2]

meanTR <- hmm2_km$mle$step[1,3]
```

```r
sdTR <- hmm2_km$mle$step[2,3]

# calculate shape and scale of the gamma distributions from mean and sd
sh <- function(mean, sd) { return(mean^2 / sd^2)}
sc <- function(mean, sd) { return(sd^2 / mean)}

# get density functions of the distributions
y_ARS <- dgamma(x, shape=sh(meanARS,sdARS),  scale=sc(meanARS,sdARS)) * stateFreq[[1]]
y_CR <- dgamma(x, shape=sh(meanCR,sdCR),  scale=sc(meanCR,sdCR)) * stateFreq[[2]]
y_TR <- dgamma(x, shape=sh(meanTR,sdTR),  scale=sc(meanTR,sdTR)) * stateFreq[[3]]


# combine densities in a single dataframe for more convenient plotting
df.y_ARS <- data.frame(dens=y_ARS, State="Foraging", x=x)
df.y_CR <- data.frame(dens=y_CR,  State="ARS", x=x)
df.y_TR <- data.frame(dens=y_TR,  State="Travelling", x=x)
statedis <- rbind(df.y_ARS, df.y_CR, df.y_TR)

# plot distributions
hmm_stepdist_plot <- ggplot() +
  geom_line(data=statedis,aes(x=x,y=dens,colour=State,linetype=State), size=1.2) +
  scale_colour_manual(values=c(colours.states,"#000000"),
              breaks = c('Foraging', 'ARS', 'Travelling'),
              labels=c("Localized search", "Area-restricted search", "Travelling")) +
  scale_linetype_manual(values=c("solid","solid", "solid"),
              breaks = c('Foraging', 'ARS', 'Travelling'),
              labels=c("Localized search", "Area-restricted search", "Travelling")) +
  ylab("Density") +
  xlab("Step length (kms)") +
  theme_minimal()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.

## Warning: Please use ‘linewidth‘ instead.
```

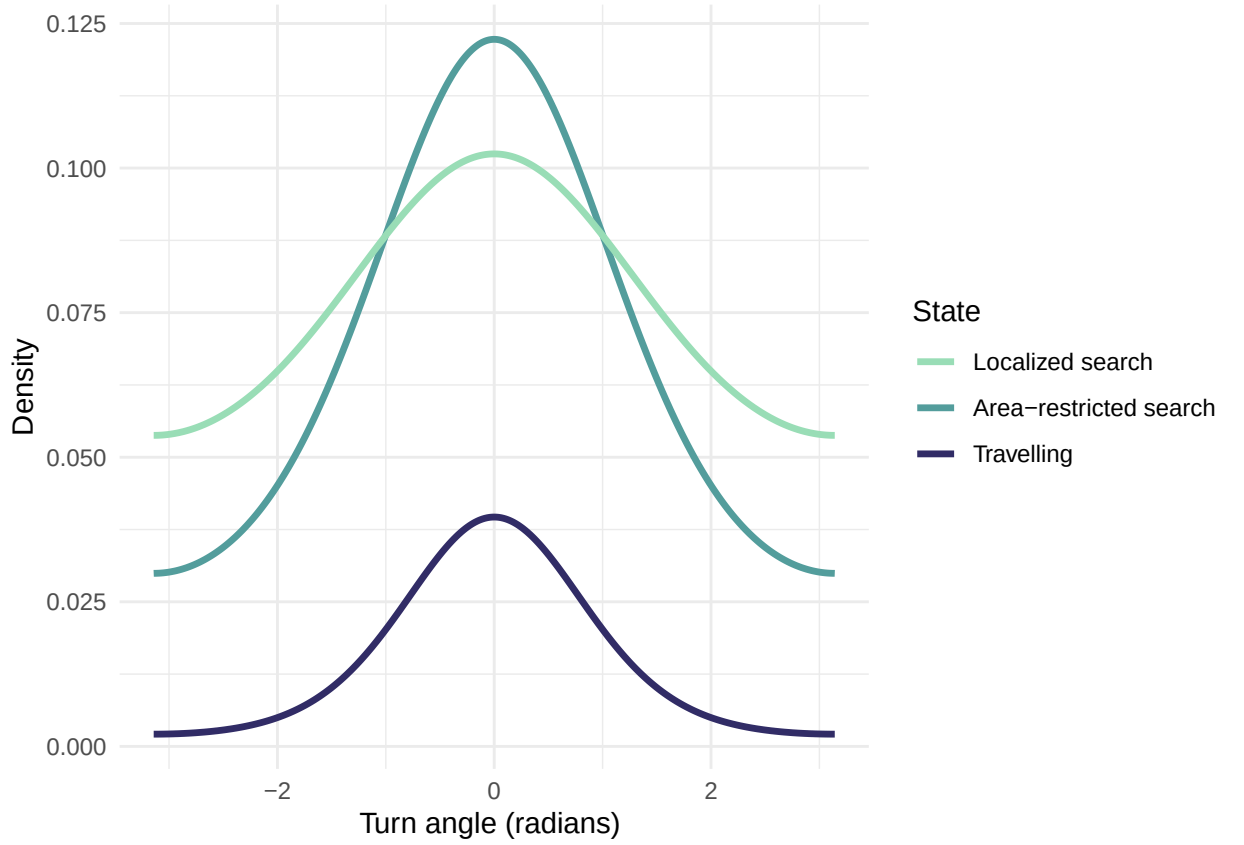
```r
hmm_stepdist_plot
```



Plot a histogram of turning angle for each state.

```
# generate sequence for x axis of density functions
x <- seq(-pi, pi,length=1000)

# get converged mean and concentration for each state
meanARS <- hmm2_km$mle$angle[1,1]
sdARS <- hmm2_km$mle$angle[2,1]

meanCR <- hmm2_km$mle$angle[1,2]
sdCR <- hmm2_km$mle$angle[2,2]

meanTR <- hmm2_km$mle$angle[1,3]
sdTR <- hmm2_km$mle$angle[2,3]

# get density functions of the distributions
y_ARS <- CircStats::dvm(x, mu=meanARS,  kappa=sdARS) * stateFreq[[1]]
y_CR <- CircStats::dvm(x, mu=meanCR,  kappa=sdCR) * stateFreq[[2]]
y_TR <- CircStats::dvm(x, mu=meanTR,  kappa=sdTR) * stateFreq[[3]]


# combine densities in a single dataframe for more convenient plotting
df.y_ARS <- data.frame(dens=y_ARS, State="Foraging",x=x)
df.y_CR <- data.frame(dens=y_CR,  State="ARS",x=x)
df.y_TR <- data.frame(dens=y_TR,  State="Travelling",x=x)

cmb <- rbind(df.y_TR, df.y_CR, df.y_ARS)
```

```
# plot distributions
hmm_angledist_plot <- ggplot() +
  geom_line(data=cmb,aes(x=x,y=dens,colour=State), size = 1.2) +
  scale_colour_manual(values=c(colours.states), breaks = c('Foraging', 'ARS', 'Travelling'), labels=c("
  scale_x_continuous(limits=c(-pi,pi))+
  ylab("Density") +
  xlab("Turn angle (radians) ") +
  theme_minimal()
hmm_angledist_plot
```



Plots of predicted relationships with the covariate

Here we are plotting a model's estimated relationship between the resource covariate and probability of selection can be useful for general ecological inference. We will calculate the log of the relative selection strength (log-RSS) for each selection function model. The log-RSS is a measure of how likely a location (for the RSF) or step (for SSFs) is to end in a proposed location (x1) to a single reference location (x2, the mean prey diversity), where zero indicates no preference, >1 indicates selection, and <1 indicates avoidance (Avgar et al. 2017, Fieberg et al. 2021).

First, we prepare a dataframe to predict on.

```
# prep the fish data
newfish <- fish_raster %>%
  terra::as.data.frame(xy = TRUE) %>%
  filter(x > 100000 & x < 600000 & y > -550000 & y < 0)
```

RSF

Since the RSF does not incorporate movement, we will calculate the log-RSS of the movement-free habitat selection kernel. This is easily done using `log_rss()` from `amt`.

First, we make a base dataframe to create x1 and x2 from. The values of `log_sl` and `cos_ta` do not matter, but we need populated columns in order for the `log_rss` function to work.

```
base <- newfish %>%
  mutate(log_sl = log(45),
         cos_ta = cos(1))

# x1 is our base dataframe
x1 <- base
```

Next, we modify the base data frame, where prey diversity is held at its mean

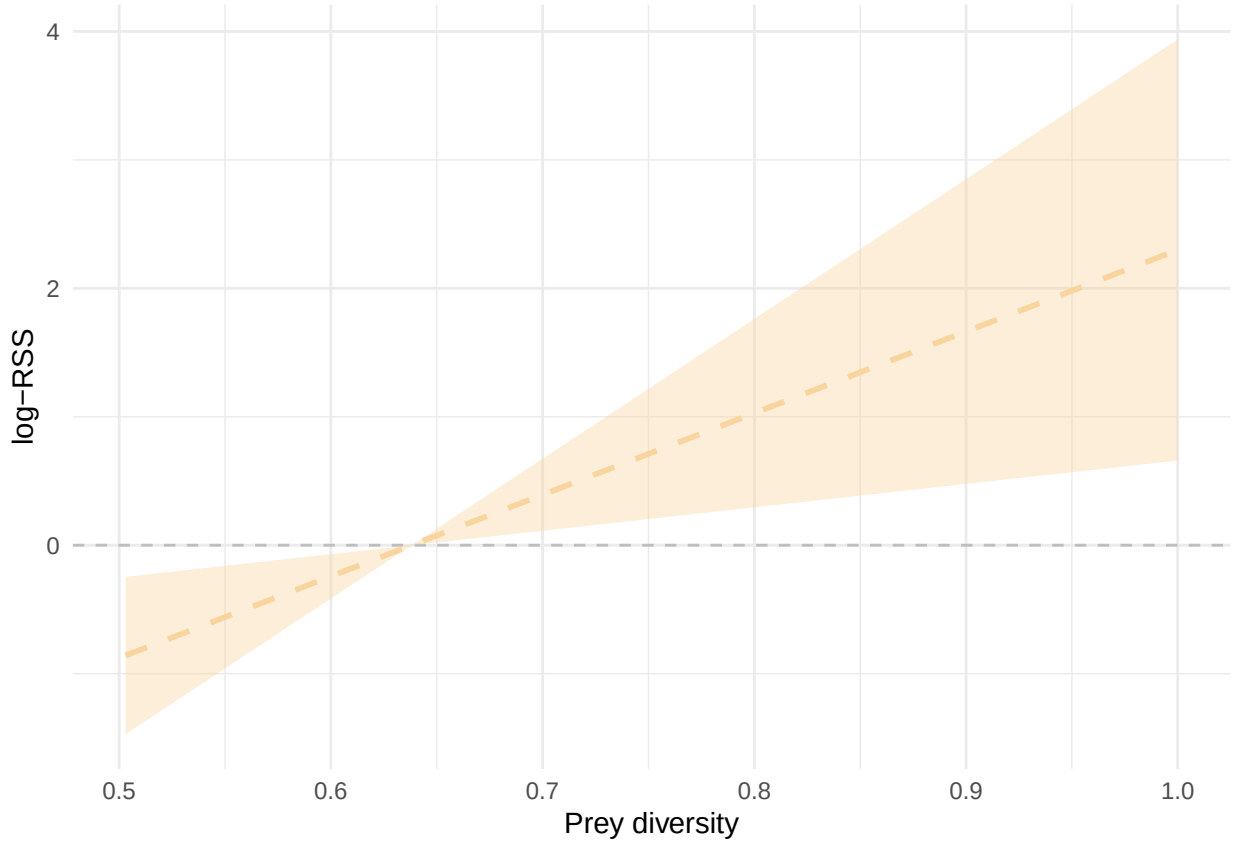
```
x2 <- base  %>%
  mutate(preydiv = mean(base$preydiv))
```

Now we will apply `log_rss()` to each row. Since `log_rss()` only assessed one location relative to a reference point, we will use lapply to iterate through all locations.

```
log_rss_list <- lapply(1:nrow(x1), function(i) {
  # Calculate log-RSS for that row
  xx <- log_rss(rsf1, x1[i,], x2[i,], ci = "se")
  # Return the element $df
  return(xx$df)
})

# combine rows
res1 <- dplyr::bind_rows(log_rss_list)
```

Visualize results.

```
# plot
line_rsf <- ggplot(res1, aes(x = preydiv_x1, y = (log_rss))) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
  geom_line(size = 1, color = "#F8D59F",linetype = 2) +
  geom_ribbon(aes(ymin=lwr, ymax=upr, x=preydiv_x1), alpha = 0.4, fill = "#F8D59F") +
  xlab("Prey diversity") +
  ylab("log-RSS") +
  theme_minimal()
line_rsf
```



We see a positive relationship between prey diversity and log-RSS, which suggests that our seal are more likely to be present in areas with higher prey diversity than areas with low prey diversity.

SSF

We can also calculate the log-RSS for our SSFs following the same workflow as for the RSF. Since log_rss() passes to predict(), it is important that the fit SSF includes `model = TRUE`.

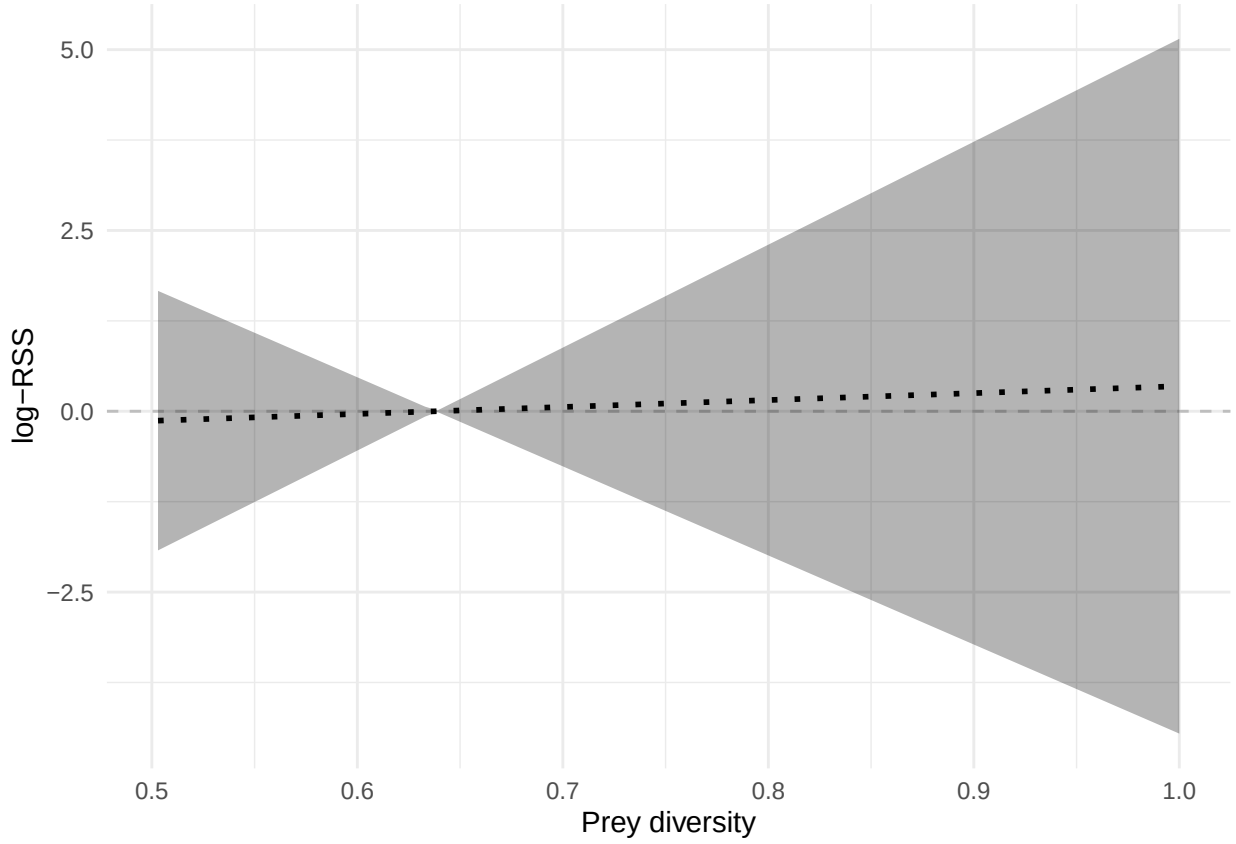
First, we will calculate log-RSS for ssf1. *MAM: you have to break this down and explain*

```
## log-RSS prediction for ssf1
# apply log_rss() to each row
log_rss_list <- lapply(1:nrow(x1), function(i) {
  # Calculate log-RSS for that row
  xx <- log_rss(ssf1, x1[i,], x2[i,], ci = "se")
  # Return the element $df
  return(xx$df)
})

# combine rows
res2 <- dplyr::bind_rows(log_rss_list) %>%
  mutate(Speed = "without int.")
```

Visualize results.

```
# plot
line_ssf1 <- ggplot() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "grey") +
```

```
  geom_line(data = res2, aes(x = preydiv_x1, y = log_rss, color = Speed, group = Speed), size = 1, linet
  geom_ribbon(data = res2, aes(ymin=lwr, ymax=upr, x=preydiv_x1, fill = Speed, group = Speed), alpha = 0
  xlab("Prey diversity") +
  ylab("log-RSS") +
  theme_minimal()
line_ssf1
```



We see no relationship between prey diversity and log-RSS, which suggests that our seal is similarily likely to be present in areas with higher prey diversity and areas with low prey diversity.

Now we will calculate log-RSS for ssf2. We will estimate the log-RSS for three different step-lengths (slow, moderate, fast). We set these speeds as the 25th, 50th, and 75th percentiles of step-length, then loop the log-RSS for each speed. First, identify what the percentiles are.

```
## log-RSS prediction for ssf2
# determine the 25th (slow), 50th (moderate), and 75th (fast) percentiles of step-length
nums <- seal %>%
  make_track(lon, lat, date) %>%
  steps %>%
  mutate(log_sl = log(sl_)) %>%
  summarize(quants = quantile(log_sl, c(0.25, 0.5, 0.75))) %>%
  pull()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was
## deprecated in dplyr 1.1.0.

## Warning: Please use `reframe()` instead.
```

```
## Warning: When switching from `summarise()` to `reframe()`, remember that
## `reframe()` always returns an ungrouped data frame and adjust accordingly.
```

```
nums
```

```
##      25%      50%      75%
## 8.394073 9.190435 9.615853
```

Now apply log-RSS to each row, for each speed (step length percentile).

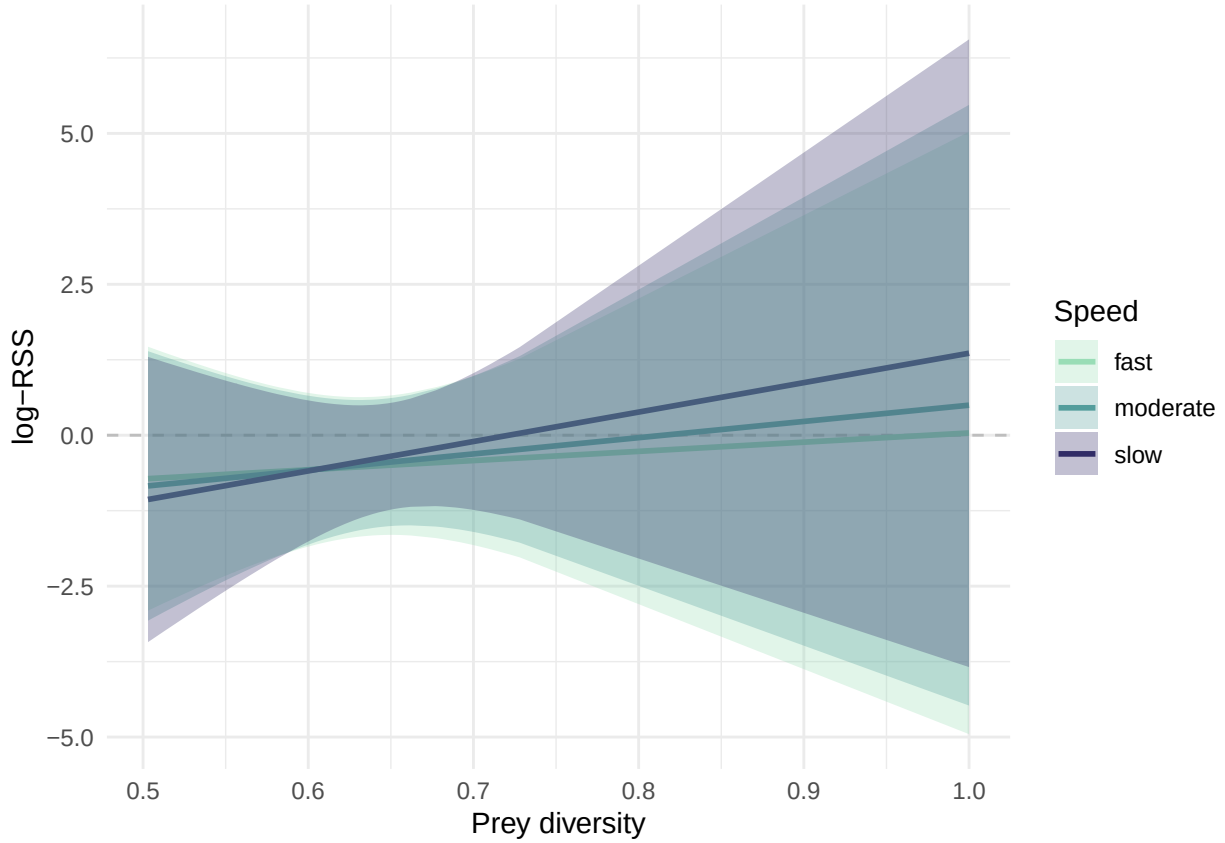
```
# set-up to run function for each speed
results_ssf2 <- lapply(nums, function(i) {
x1$log_sl <- i

# calculate log-RSS
  log_rss_list <- lapply(1:nrow(x1), function(i) {
    # Calculate log-RSS for that row
    xx <- log_rss(ssf2, x1[i,], x2[i,], ci = "se")
    # Return the element $df
    return(xx$df)
  })
  # bind rows within each speed's prediction
  res3 <- dplyr::bind_rows(log_rss_list)
} )
```

Bind the output together.

```
# bind rows of all speed's prediction
results_ssf2 <- dplyr::bind_rows(results_ssf2) %>%
  mutate(log_sl_x1 = as.factor(round(log_sl_x1,1)),
         Speed = dplyr::case_when(as.factor(log_sl_x1) == '8.4' ~ "slow",
                                  as.factor(log_sl_x1) == "9.2" ~ "moderate",
                                  as.factor(log_sl_x1) == "9.6" ~ "fast"))
```

Visualize results.

```
# plot
line_ssf2 <- ggplot(results_ssf2, aes(x = preydiv_x1, y = (log_rss))) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
  geom_line(size = 1, aes(color = Speed, group = Speed, linetype = Speed)) +
  geom_ribbon(aes(ymin=lwr, ymax=upr, x=preydiv_x1, fill = Speed, group = Speed), alpha = 0.3) +
  scale_colour_manual(values=c(colours.states,"#000000")) +
  scale_fill_manual(values=c(colours.states,"#000000")) +
  scale_linetype_manual(values = c("solid", "solid", "solid")) +
  xlab("Prey diversity") +
  ylab("log-RSS") +
  theme_minimal()
line_ssf2
```

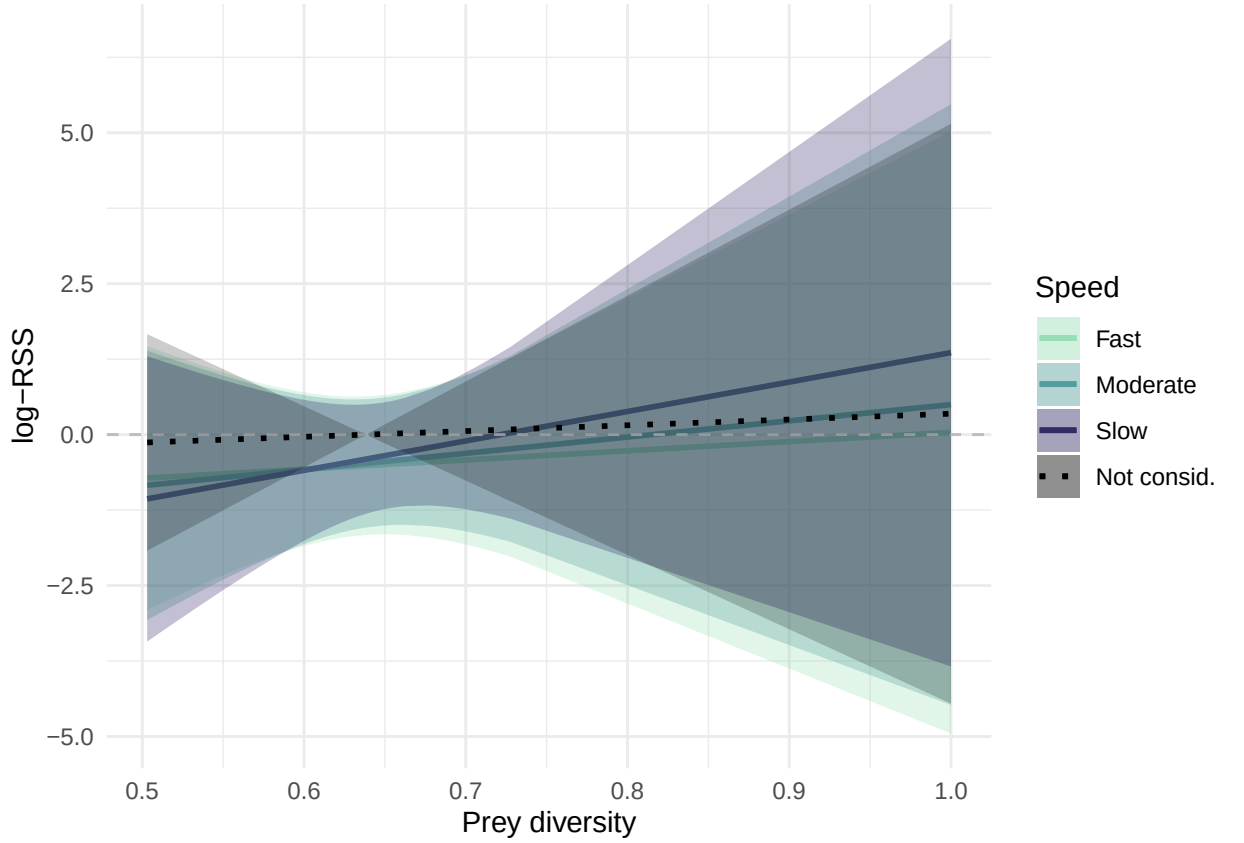


We can see a weak positive relationship between prey diversity and log-RSS, however, with confidence intervals (shading) covering zero, suggesting no significant relationship between prey diversity and log-RSS. We also see that the confidence intervals cover each other, suggesting that different speeds do not have different relationships between prey diversity and log-RSS.

Typically these models would be interpreted independently. But it's worth noting that while the effects are minimal and the confidence intervals overlap, when comparing ssf1 and ssf2, ssf2 provides more information about the animal's relationship with prey diversity.

```
# Plot
ggplot(results_ssf2, aes(x = preydiv_x1, y = (log_rss))) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
  geom_line(size = 1, aes(color = Speed, group = Speed, linetype = Speed)) +
  geom_ribbon(aes(ymin=lwr, ymax=upr, x=preydiv_x1, fill = Speed, group = Speed), alpha = 0.3) +
  xlab("Prey diversity") +
  ylab("log-RSS") +
  theme_minimal() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
  geom_line(data = res2, aes(x = preydiv_x1, y = log_rss, color = Speed, group = Speed, linetype = Speed
  geom_ribbon(data = res2, aes(ymin=lwr, ymax=upr, x=preydiv_x1, fill = Speed, group = Speed), alpha =
  scale_color_manual(values = c("#99DDB6", "#539D9C", "#312C66", "black"),
                     labels = c("Fast", "Moderate", "Slow", "Not consid."),
                     name = "Speed") +
  scale_fill_manual(values = c("#99DDB6", "#539D9C", "#312C66", "black"),
                     labels = c("Fast", "Moderate", "Slow", "Not consid."),
                     name = "Speed") +
  scale_linetype_manual(values = c("solid", "solid", "solid", "dotted"),
```

```
                    labels = c("Fast", "Moderate", "Slow", "Not consid."),
                    name = "Speed")
```



Here, we see that adding the interaction between prey diversity and step length did not better explain our data or provide additional ecological insight about our seal.

HMM

Exploring the insight on the relationship between our seal and prey diversity is fundamentally different for the HMM. Here, we will grab and plot the stationary state probabilities. This is easily done using `plotStationary()` from `momentuHMM`.

```
# grab stationary probabilities
ps <- momentuHMM::plotStationary(hmm4_km, plotCI= TRUE, return = TRUE)
```

```
# grab values for data frame
state1 <- ps$preydiv$'state 1' %>% mutate(state = 1)
state2 <- ps$preydiv$'state 2' %>% mutate(state = 2)
state3 <- ps$preydiv$'state 3' %>% mutate(state = 3)

# bind to one data frame
pdat <- rbind(state1, state2, state3) %>%
  mutate(state = as.character(state))
```

Visualize results.

```
# plot
line_hmm <- ggplot() +
```

```
  geom_line(data = pdat, aes(x = cov, y = est, color = state)) +
  geom_ribbon(data = pdat, aes(x=cov, y=est, ymax=est+se, ymin=est-se, fill = state),
              alpha = 0.4, show.legend = TRUE) +
  ylab("Stationary state probabilties") +
  xlab("Prey diversity") +
  scale_color_manual(values = c("#99DDB6", "#539D9C", "#312C66"), name = "HMM state",
                     labels=c("Slow movement", "Moderate movement", "Fast movement")) +
    scale_fill_manual(values = c("#99DDB6", "#539D9C", "#312C66"), name = "HMM state",
                     labels=c("Slow movement", "Moderate movement", "Fast movement")) +
  theme_minimal()

line_hmm
```

Here we can see that each state has a different relationship between the stationary state probability and prey diversity. The slow movement state has a positive relationship between stationary state probability and prey diversity, the moderate movement state has a negative relationship with prey diversity, and the fast movement state does not appear to have a directional relationship with prey diversity.

Prediction maps

Now we will estimate the utilization distributions from each model to demonstrate how differences in the relationships with a covariate can results in vastly different spatial patterns. The utilization distribution is defined as the two-dimensional relative frequency distribution of space use of an animal (Van Winkle 1975). This is a simple calculation for the RSF, where we multiply the model coefficient with the resource (prey diversity), exponentiate (since it is a logistic regression), and normalize the estimate. The calculations are more complex for the SSFs since they are conditional models that integrate the movement process. Thus, for the SSFs we calculate the steady-state utilization distribution (SSUD), which is the long-term expectation of the space-use distribution across the landscape (Signer et al. 2017). `amt` has functions to estimate the SSUD.

RSF

We can predict the estimated probability of use from the RSF by hand. First, grab the model coefficients and predict for each cell.

```
# grab model coefficients
modcoef <- summary(rsf1)$coef

# prediction for each cell
x <- exp(modcoef[2] * newfish$preydiv)
```

We will normalize the results next.

```
# range fn
range01 <- function(x){(x-min(x))/(max(x)-min(x))}

# set the range from zero to one
newfish$rsf_prediction <- range01(x)
```

Visualize results.

```
# plot
map_rsf <- ggplot() +
  geom_tile(data = newfish, aes(x = x,y = y, fill = rsf_prediction)) +
  scale_fill_viridis(option = "mako", name = "RSF prediction",limits = c(0,0.14)) +
  geom_polygon(data = nat_trans, aes(x=long,y=lat,group=group), fill = "grey80", color = "white") +
```

```
  coord_cartesian(xlim = c(150000,550000), ylim = c(-500000,-50000), expand = F) +
  theme_void()
```

```
## Regions defined for each Polygons
```

```
map_rsf
```

SSF

We can use the `amt` functions to estimate the SSUDs from the simple SSF that does not allow prey diversity to affect the movement kernel.

MAM: you need to explain why you are fitting it again

```
# generate availability sample
set.seed(2023)
data_ssf <- seal %>%
  mutate(date = as.POSIXct(date)) %>%
  make_track(lon, lat, date) %>%
  steps() %>%
  random_steps() %>%
  arrange(case_) %>%
  amt::extract_covariates(fish_raster, where = "both") # %>%
  #na.omit()

# fit SSF1 model
m1 <- data_ssf |>
```

```
  fit_clogit(case_ ~ preydiv_end +  cos(ta_) + log(sl_) +
               strata(step_id_))
```

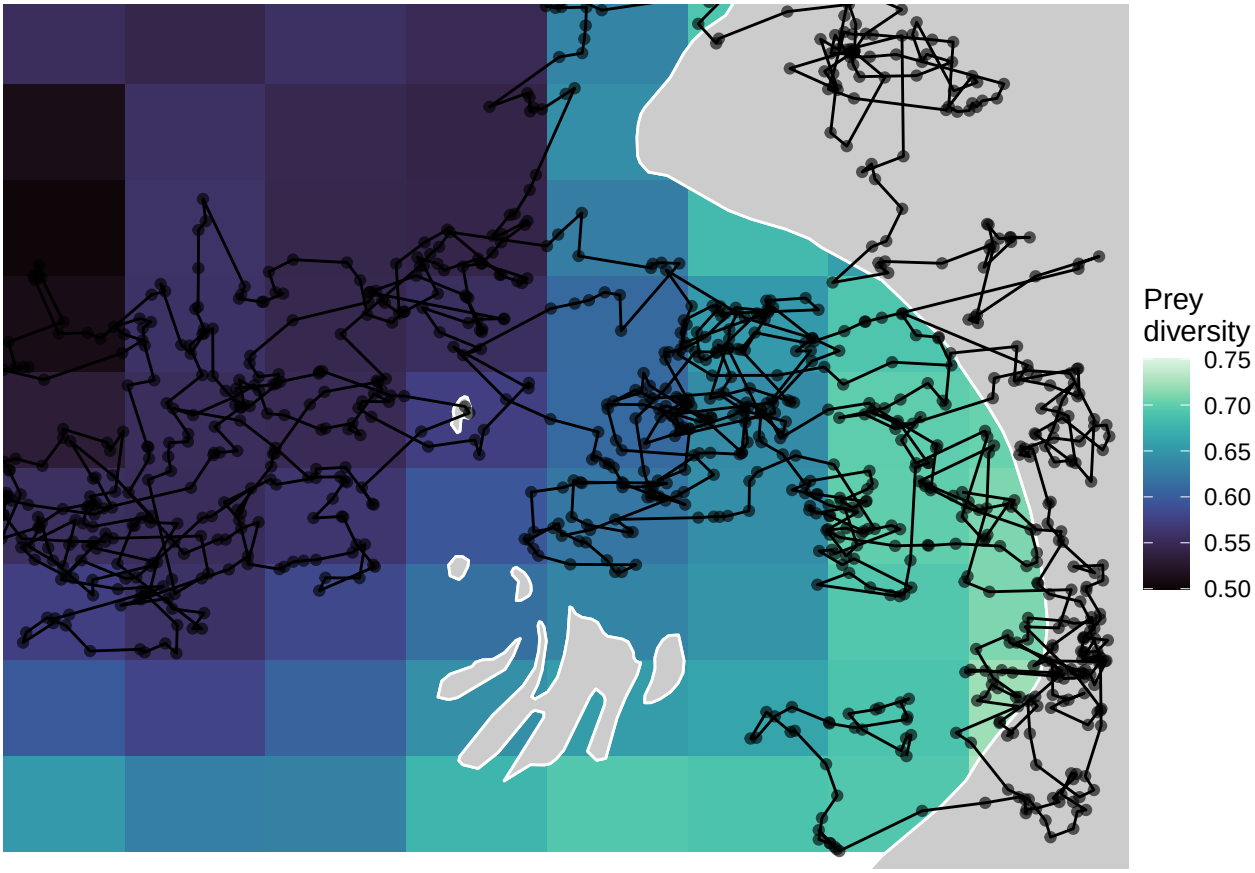
We will now simulate a track and visually observe it.

MAM: break it down and explain, e.g. why burn in, etc

```
# set starting position for the simulation
start <- make_start((seal %>%
                       mutate(date = as.POSIXct(date)) %>%
                       make_track(lon, lat, date))[1,])

# Set constants
n_steps = 1e3 # number of steps
n_steps1 = n_steps + 1 # number of steps +1
burnin <- n_steps/50 # number of locations to remove for burn-in

# generate redistribution kernel
k1 <- redistribution_kernel(m1, map = fish_raster, start = start,
                            stochastic = TRUE,
                            tolerance.outside = 1,
                            n.control = 1e3)

# simulate path
set.seed(2023)
p1 <- amt::simulate_path(x = k1, n = n_steps, start = start, verbose = TRUE)

# burn-in
p1_burnt <- p1 %>% slice(-c(1:burnin))

# plot simulated track
ssf_track_1 <- fishmap +
  geom_polygon(data = nat_trans, aes(x=long,y=lat,group=group), fill = "grey80", color = "white") +
  geom_point(data = p1_burnt, aes(x = x_,y = y_), alpha = 0.61) +
  geom_path(data = p1_burnt, aes(x = x_,y = y_)) +
  theme_void()
```

```
## Regions defined for each Polygons
```

```
ssf_track_1
```



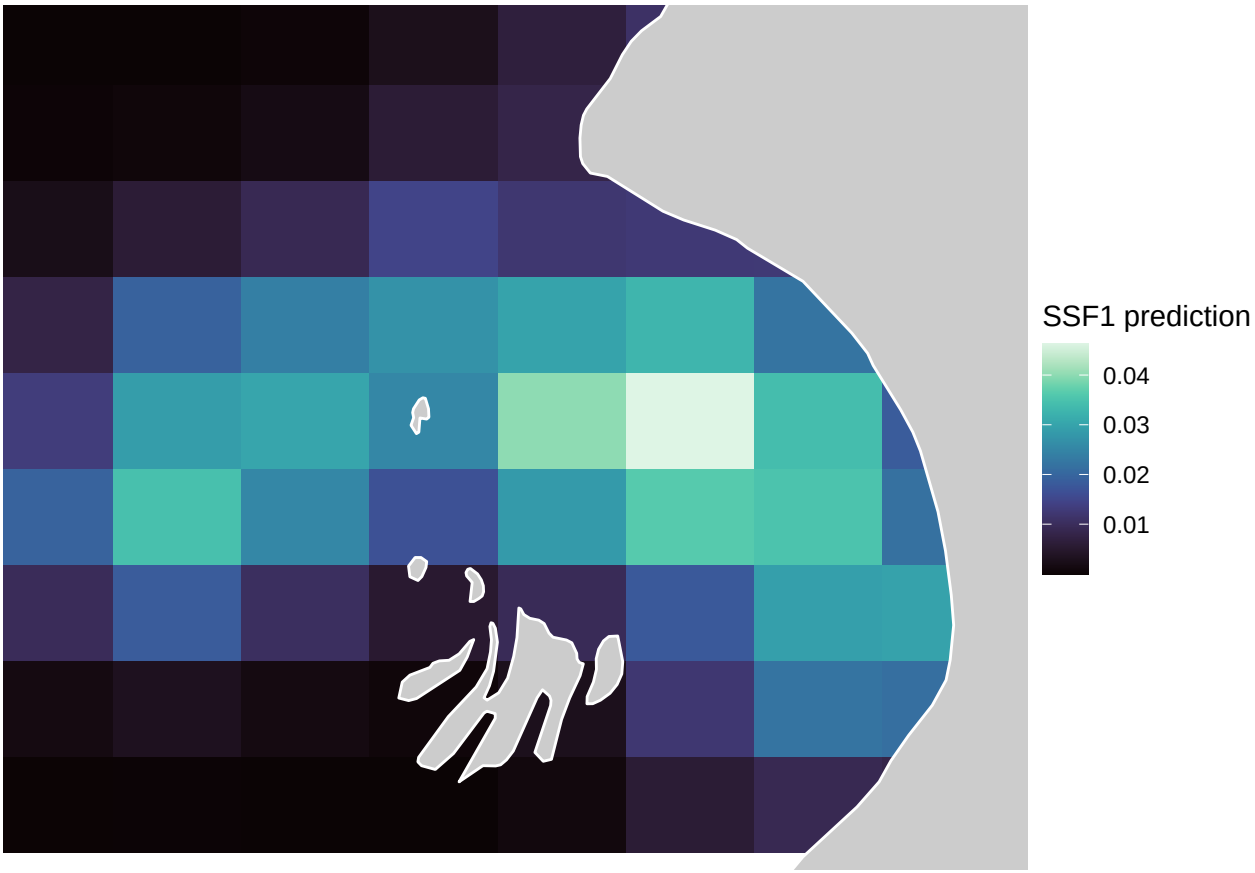
We can see that it mostly stays within the study area. We will use this track to estimate the SSUD, and visualize the results.

MAM: You also need to state above that you are not using the same number of simulations for the tutorial, because it would take too long to run. Add a warning for anything that takes more than 1-2 min. (e.g. this will likely take time to run).

```
# estimate SSUD
uds_ssf1 <- tibble(rep = 1:n_steps1,
    x_ = p1$x_, y_ = p1$y_,
    t_ = p1$t_, dt = p1$dt) |>
  filter(!is.na(x_)) |>
  make_track(x_, y_) |>
  hr_kde(trast = fish_raster, which_min = "global") %>%
  hr_ud() %>%
  terra::as.data.frame(xy = TRUE)

# plot SSUD
map_ssf1 <- ggplot() +
  geom_tile(data = uds_ssf1, aes(x = x,y = y, fill = preydiv)) +
  scale_fill_viridis(option = "mako", name = "SSF1 prediction") +
  geom_polygon(data = nat_trans, aes(x=long,y=lat,group=group), fill = "grey80", color = "white") +
  coord_cartesian(xlim = c(150000,550000), ylim = c(-500000,-50000), expand = F) +
  theme_void()
```

```
## Regions defined for each Polygons
```

```
map_ssf1
```



We will follow the same steps to generate a SSUD for the SSF that allows prey diversity to affect the movement kernel.

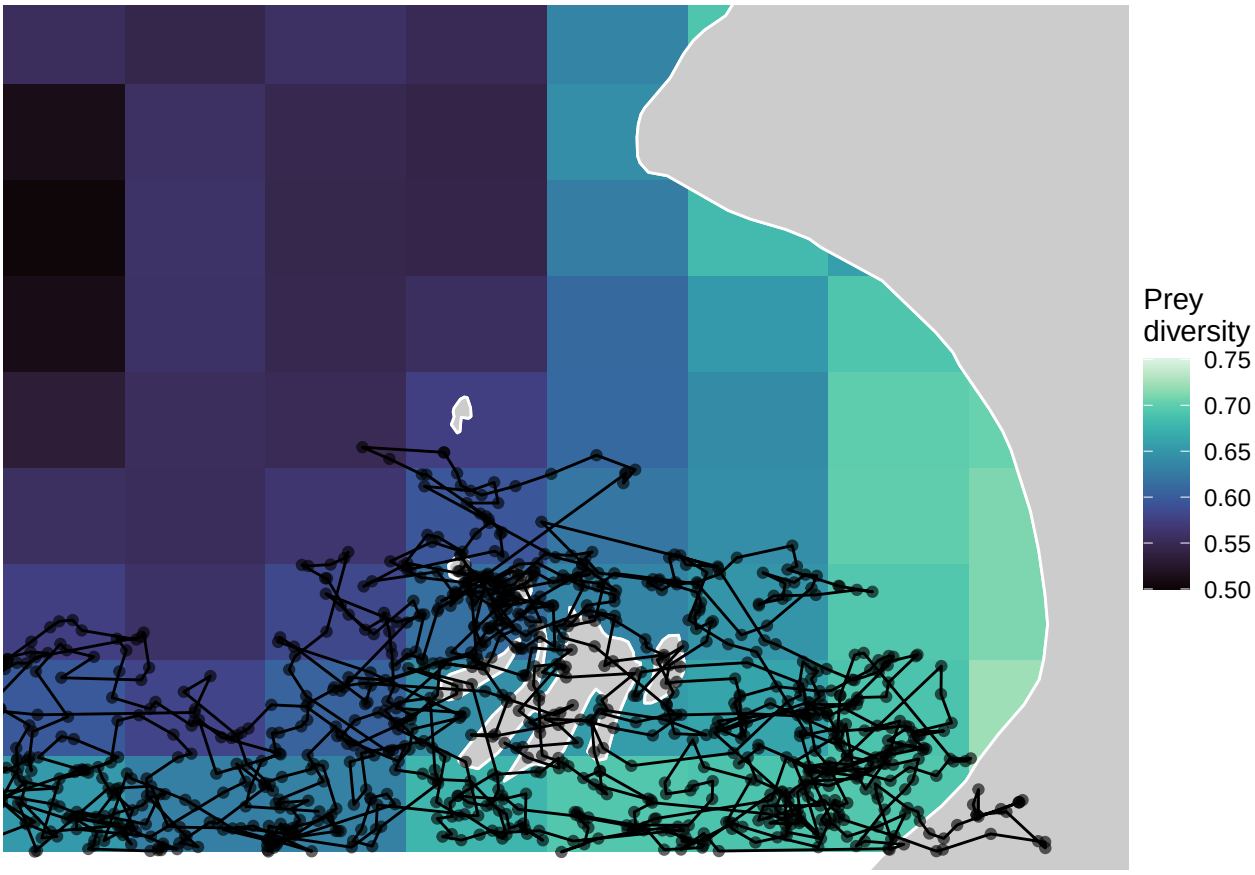
MAM: please ask J. Signer about the tolerance.outside = 1 in the discussion chain on Github. And then explain what it means here, because it's pretty weird. You need to break this down and explain.

```
# fit SSF2 model
m2 <- data_ssf |>
  fit_clogit(case_ ~ preydiv_end +  cos(ta_)+
               preydiv_end:log(sl_) +
               strata(step_id_))


# set starting position for the simulation
set.seed(2023)
start <- make_start((seal %>%
                       mutate(date = as.POSIXct(date)) %>%
                       make_track(lon, lat, date))[1,])

# generate redistribution kernel
k2 <- redistribution_kernel(m2, map = fish_raster, start = start,
                            stochastic = TRUE,
                            tolerance.outside = 1,
                            n.control = 1e3)
```

```
# Now simulate a path of length 1e3
n_steps = 1e3
n_steps1 = n_steps + 1
set.seed(2023)
p2 <- amt::simulate_path(x = k2, n = n_steps, start = start)

# plot simulated track
ssf_track_2 <- fishmap +
  geom_polygon(data = nat_trans, aes(x=long,y=lat,group=group), fill = "grey80", color = "white") +
  geom_point(data = p2, aes(x = x_,y = y_), alpha = 0.61) +
  geom_path(data = p2, aes(x = x_,y = y_)) +
  theme_void()
```

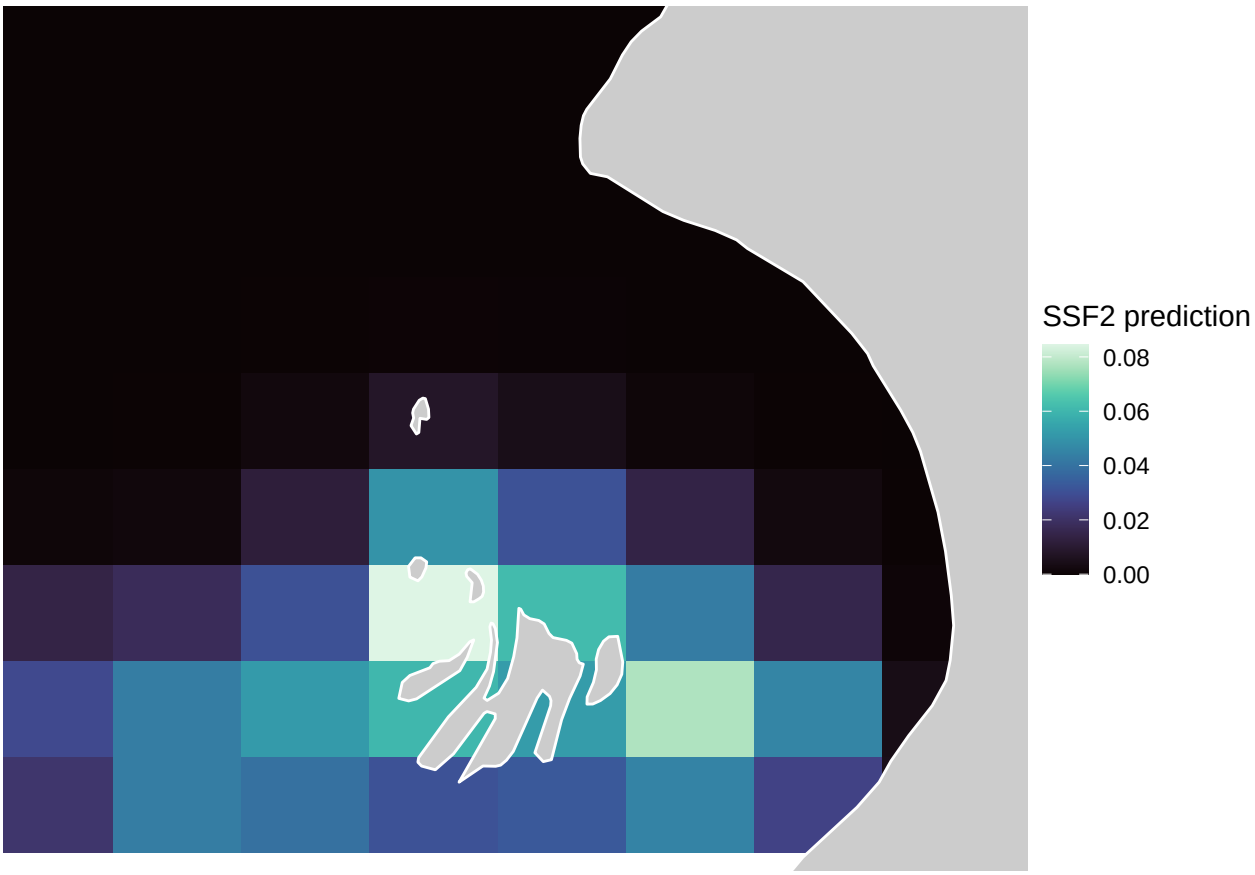
```
## Regions defined for each Polygons
```

```
ssf_track_2
```




```
# estimate SSUD
uds_ssf2 <- tibble(rep = 1:n_steps1,
    x_ = p2$x_, y_ = p2$y_,
    t_ = p2$t_, dt = p2$dt) |>
  filter(!is.na(x_)) |>
  make_track(x_, y_) |>
  hr_kde(trast = fish_raster, which_min = "local") %>%
  hr_ud() %>%
  terra::as.data.frame(xy = TRUE)
```

```
# plot SSUD
map_ss2 <- ggplot() +
  geom_tile(data = uds_ssf2, aes(x = x,y = y, fill = preydiv)) +
  scale_fill_viridis(option = "mako", name = "SSF2 prediction") +
  geom_polygon(data = nat_trans, aes(x=long,y=lat,group=group), fill = "grey80", color = "white") +
  coord_cartesian(xlim = c(150000,550000), ylim = c(-500000,-50000), expand = F) +
  theme_void()

## Regions defined for each Polygons
map_ss2
```



HMM

We will first estimate the stationary state probabilities of each state based on prey diversity. This is easily done using the `momentuHMM` function `stationary()`.

MAM: is some of this done above? Maybe you could combine? You need to break down and explain (e.g. what's the new fish data, etc. Note that the new fish data could be explained at the top somewhere since you are using it in all predictions.)

```
# grab estimated stationary state probabilities from our fitted HMM
x <- as.data.frame(momentuHMM::stationary(hmm4_km, data.frame(preydiv = newfish$preydiv)))
newfish$hmm_state1 <- x$state.1
newfish$hmm_state2 <- x$state.2
newfish$hmm_state3 <- x$state.3

# prepare data
```

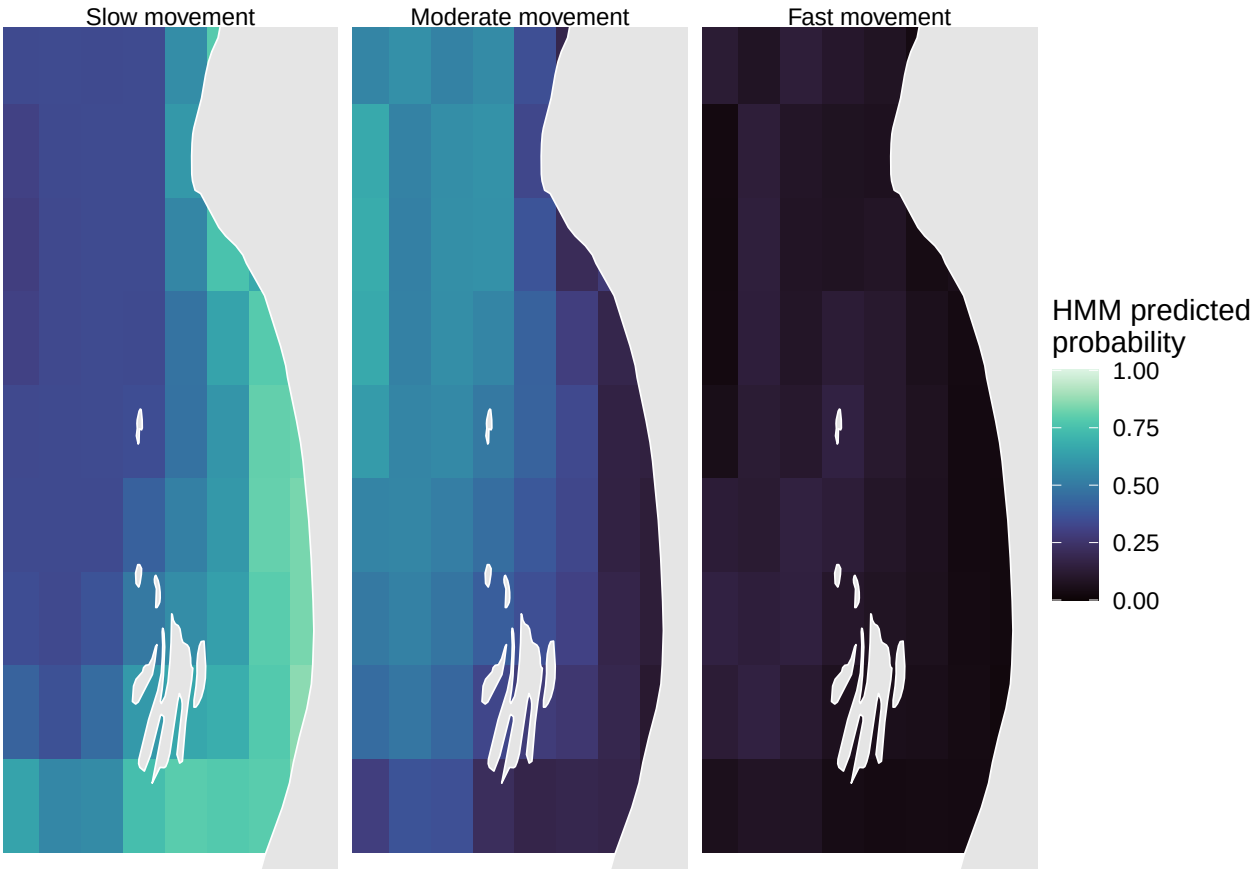
```
newfish_long <- newfish %>%
  tidyr::pivot_longer(cols = hmm_state1:hmm_state3,
               names_to = "model", values_to = "prediction") %>%
  mutate(dplyr::across(model, factor, levels=
               c("hmm_state1", "hmm_state2", "hmm_state3")))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `dplyr::across(...)`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```
# plot
map_hmm <- ggplot() +
  geom_tile(data = newfish_long, aes(x = x, y = y, fill = prediction)) +
  scale_fill_viridis(option = "mako", limits = c(0,1)) +
  labs(fill = 'HMM predicted\nprobability') +
  geom_polygon(data = nat_trans, aes(x=long,y=lat,group=group), colour = "white", fill = "grey90", size
  coord_cartesian(xlim = c(150000,550000), ylim = c(-500000,-50000), expand = F) +
  facet_wrap(~model, labeller = as_labeller(c('hmm_state1' = "Slow movement",
                                              'hmm_state2' = "Moderate movement",
                                              'hmm_state3' = "Fast movement"))) +
  theme_void()
```

```
## Regions defined for each Polygons
```

```
map_hmm
```



MAM: interpret the results.

References

Avgar, T., Lele, S. R., Keim, J. L., & Boyce, M. S. (2017). Relative selection strength: Quantifying effect size in habitat-and step-selection inference. Ecology and Evolution, 7(14). 5322-5330.https://doi.org/10.1002/ece3.3122

Fieberg, J., Signer, J., Smith, B., & Avgar, T. (2021). A 'How to'guide for interpreting parameters in habitat-selection analyses. Journal of Animal Ecology, 90(5), 1027-1043. https://doi.org/10.1111/1365-2656.13441

Florko, K.R.N., Tai, T.C., Cheung, W.W.L., Sumaila, U.R., Ferguson, S.H., Yurkowski, D.J., Auger-Méthé, M. (2021). Predicting how climate change threatens the prey base of Arctic marine predators. Ecology Letters, 24: 2563-2575. https://doi.org/10.1111/ele.13866

Florko, K.R.N., Tai, T.C., Cheung, W.W.L., Sumaila, U.R., Ferguson, S.H., Yurkowski, D.J., Auger-Méthé, M. (2021b), Predicting how climate change threatens the prey base of Arctic marine predators, Dryad, Dataset, https://doi.org/10.5061/dryad.x69p8czjs

Florko, K.R.N., Shuert, C.R., Cheung, W.W.L., Ferguson, S.H., Jonsen, I.D., Rosen, D.A.S., Sumaila, U.R., Tai, T.C., Yurkowski, D.J., Auger-Méthé, M. (2023). Linking movement and dive data to prey distribution models: new insights in foraging behavior and potential pitfalls of movement analyses. Movement Ecology, 11:17 https://doi.org/10.1186/s40462-023-00377-2

McClintock, B.T., Michelot, T. (2018). momentuHMM: R package for generalized hidden markov models of animal movement. Methods in Ecology and Evolution, 9, 1518-1530. https://doi.org/10.1111/2041-210X.12995

Signer, J., Fieberg, J., & Avgar, T. (2017). Estimating utilization distributions from fitted step-selection functions. Ecosphere, 8(4), e01771. https://doi.org/10.1002/ecs2.1771

Signer, J., J. Fieberg, and T. Avgar. (2019). Animal movement tools (amt): R package for managing tracking data and conducting habitat selection analyses. Ecology and Evolution, 9:880–890. https://doi.org/10.1002/ece3.4823

Van Winkle, W. (1975). Comparison of several probabilistic home-range models. The Journal of wildlife Management, 118-123. https://doi.org/10.2307/3800474