Blue Whales and Lagrangian Features

Methods and Analysis

James A. Fahlbusch Max F. Czapanskiy John Calambokidis David E. Cade
Briana Abrahms Elliott L. Hazen Jeremy A. Goldbogen

18 July, 2022

Contents

Load Packages

```
pkgTest <- function(x)
{
  if (!require(x,character.only = TRUE))
  {
    install.packages(x,dep=TRUE)
    if(!require(x,character.only = TRUE)) stop("Package not found")
  }
}
`%notin%` <- Negate(`%in%`)
# Plotting and Data Wrangling
pkgTest("tidyverse")
pkgTest("RColorBrewer")
pkgTest("ggpubr")
pkgTest("lubridate")
pkgTest("rnaturalearth")
pkgTest("maptools")
pkgTest("ggmap")
pkgTest("metR")
pkgTest("kableExtra")
pkgTest("rnaturalearthdata")
pkgTest("sf")
pkgTest("sp")
pkgTest("moveHMM")
pkgTest("nlme")
pkgTest("stats")
pkgTest("AICcmodavg")
pkgTest('MetricsWeighted')
pkgTest("MASS")
pkgTest("ggspatial")

# Load a custom function for extracting the Confidence intervals
# for stationary probabilities
source('./functions/moveHMMstationaryCI.R')

# Function to find the Pooled Standard Deviation of a Sample of Samples
# i.e. ILI by dive has a stdev, but a group of dives will have a pooled stdev
# Input:
#   stdevs = standard deviations of sample
#   numSamples = sample sizes for each observation
pooledstd <- function(stdevs,numSamples) {
  psd <- sqrt( sum((numSamples-1)*(stdevs^2))/sum(numSamples-1) )
  return(psd)
}

# Function to convert between Logit and Probability
logit2prob <- function(logit){
  odds <- exp(logit)
  prob <- odds / (1 + odds)
  return(prob)
}
```

```r
# Function to Convert Probability to Logit
prob_logit <- function(x) {
  out <- log(x / (1 - x))
  return(out)
}
# Function to perform a KS Test and returns a dataframe with the result
ksResults <- function(data1, data2, alt, Distribution, ID) {
  resKS <- ks.test(x=data1,y=data2,alternative=alt)
  res <- data.frame(ID = ID,
                    Distribution = Distribution,
                    test = resKS$alternative,
                    pvalue = as.numeric(resKS$p.value),
                    dvalue = as.numeric(resKS$statistic),
                    sampleSizeX = as.numeric(length(data1)),
                    meanX = mean(data1,na.rm=TRUE),
                    sdX= sd(data1,na.rm = TRUE),
                    sampleSizeY = as.numeric(length(data2)),
                    meanY = mean(data2,na.rm=TRUE),
                    sdY= sd(data2,na.rm = TRUE))
  return(res)
}
```

Load and Process Data

```r
## Load Pre-processed Data
load("./dataProcessed/diveDataset.RData") # Load the processed dive dataframe
load("./dataProcessed/datasetFTLE.RData") # Load the processed dive+FTLE dataframe
load("./dataProcessed/datasetSimsFTLE.RData") # Load the Null Model data
load("./dataProcessed/datasetBackgroundFTLE.RData") # Load the Background Points data
load("./rmdFiles/world.RData") # load world SF basemap
load("./rmdFiles/bf.RData") # load Bathymetric Features
load("./rmdFiles/hfRadarStations.RData")
load("./rmdFiles/HFRadarCoverageMCP.RData")
# HF Radar Coverage Area and Station Locations
hfRadarStns$name <- "Radar Station"

## Load CAR1 models
# Real Data Model
if (file.exists("./Models/feed_glmmPQL_FTLE_CAR1_Hrs.rds")) {
  feed_glmmPQL_FTLE_CAR1_Hrs <- readRDS("./Models/feed_glmmPQL_FTLE_CAR1_Hrs.rds")
} else {
  cat(paste0("Could not find ./Models/feed_glmmPQL_FTLE_CAR1_Hrs.rds, will need to recreate below."))
}

# Time-Shifted Models
if (file.exists("./Models/feed_glmmPQL_p24_CAR1_Hrs.rds")) {
  feed_glmmPQL_p24_CAR1_Hrs <- readRDS("./Models/feed_glmmPQL_p24_CAR1_Hrs.rds")
} else {
  cat(paste0("Could not find ./Models/feed_glmmPQL_p24_CAR1_Hrs.rds, will need to recreate below."))
}
if (file.exists("./Models/feed_glmmPQL_p48_CAR1_Hrs.rds")) {
  feed_glmmPQL_p48_CAR1_Hrs <- readRDS("./Models/feed_glmmPQL_p48_CAR1_Hrs.rds")
} else {
  cat(paste0("Could not find ./Models/feed_glmmPQL_p48_CAR1_Hrs.rds, will need to recreate below."))
```

```
}
if (file.exists("./Models/feed_glmmPQL_p96_CAR1_Hrs.rds")) {
  feed_glmmPQL_p96_CAR1_Hrs <- readRDS("./Models/feed_glmmPQL_p96_CAR1_Hrs.rds")
} else {
  cat(paste0("Could not find ./Models/feed_glmmPQL_p96_CAR1_Hrs.rds, will need to recreate below."))
}
if (file.exists("./Models/feed_glmmPQL_p192_CAR1_Hrs.rds")) {
  feed_glmmPQL_p192_CAR1_Hrs <- readRDS("./Models/feed_glmmPQL_p192_CAR1_Hrs.rds")
} else {
  cat(paste0("Could not find ./Models/feed_glmmPQL_p192_CAR1_Hrs.rds, will need to recreate below."))
}

# Simulated Tracks Model
if (file.exists("./Models/feed_glmmPQL_CRW_CAR1_Hrs.rds")) {
  feed_glmmPQL_CRW_CAR1_Hrs <- readRDS("./Models/feed_glmmPQL_CRW_CAR1_Hrs.rds")
} else {
  cat(paste0("Could not find ./Models/feed_glmmPQL_CRW_CAR1_Hrs.rds, will need to recreate below."))
}

# Create a dataframe with no missing FTLE vales and columns for grouping and plotting
data_Gamm_ALL <- datasetFTLE %>%
  # Only include Dives with FTLE data
  dplyr::filter(is.nan(ftle48)==0, is.na(ftle48)==0) %>%
  group_by(depid) %>%
  # Add a time Variable for the AR1 correlation structure
  mutate(time = startI-min(startI),
         timeHrs = time/3600,
  # Add a binary Foraging field for Logistic Regression
         Foraging = if_else(LungeCount > 0,1,0),
  # Add a hourly feeding rate for use in HMM
         total_Duration = dive_Duration + surf_Duration,
         feedingRateHR = LungeCount/total_Duration*60) %>%
  ungroup()

# Create a feeding and non-feeding DF for plotting
bwFeeding <- data_Gamm_ALL %>% dplyr::filter(LungeCount>0) %>%
  mutate(Distribution = "Feeding") %>%
   group_by(depid) %>%
  mutate(weight = 1/(n()*length(unique(data_Gamm_ALL$depid)))) %>%
  ungroup()
bwNoFeeding <- data_Gamm_ALL %>%
  dplyr::filter(LungeCount==0) %>%
  mutate(Distribution = "Not Feeding")
# Create an overall feeding dataframe for calculating feeding rates and
# feeding rate percentiles
bwFeedingOverall <- diveDataset %>%
  dplyr::filter(LungeCount >0) %>%
  mutate(Distribution = "Feeding") %>%
  group_by(depid) %>%
  mutate(weight = 1/(n()*length(unique(data_Gamm_ALL$depid))),
         # Add a hourly feeding rate for use in HMM
         total_Duration = dive_Duration + surf_Duration,
         feedingRateHR = LungeCount/total_Duration*60) %>%
```

```
  ungroup()

## Process the background FTLE data
data_GammBackground <- datasetBackgroundFTLE %>%
  dplyr::filter(depid %in% unique(data_Gamm_ALL$depid),
                Integration == "ftle48") %>%
  rename(ftle48 = FTLE) %>%
  left_join(dplyr::select(data_Gamm_ALL, depid,
                          DiveNum,Foraging,LungeCount,dttzStart)) %>%
  dplyr::filter(!is.na(dttzStart)) %>%  # only include records in data_Gamm
  mutate(method = Distribution) %>%
  arrange(depid,dttzStart) %>%
  group_by(depid) %>%
  mutate(time = as.numeric(dttzStart)-min(as.numeric(dttzStart)),
         timeHrs = time/3600) %>%
  ungroup() %>%
  group_by(time) %>%
  mutate(FTLE_ID = paste0(depid, seq(1,n()),sep="-B")) %>%
  ungroup()

## Process the Simulated Track Data
data_GammSim <- datasetSimsFTLE %>%
  # dplyr::filter(FTLE_ID %in% unique(data_Gamm_ALL$depid)) %>%
  mutate(method = if_else(method=="BrownianBridge","CRW",method),
         Foraging = if_else(LungeCount > 0,1,0)) %>%
  dplyr::filter(is.nan(ftle48)==0, is.na(ftle48)==0) %>%
  arrange(method,depid,dttzStart) %>%
  group_by(depid) %>%
  # Add a time Variable for the AR1 correlation structure
  mutate(time = as.numeric(dttzStart)-min(as.numeric(dttzStart)),
         timeHrs = time/3600) %>%
  ungroup()

data_GammCRW <- data_GammSim %>%
  dplyr::filter(method=="CRW") %>%
  group_by(depid) %>%
  mutate(SimNum = as.numeric(unlist(str_split(depid,
                                              pattern = "_SimmCRW-"))[2])) %>%
  ungroup()

data_GammPlus24 <- data_GammSim %>%
  dplyr::filter(method=="_plus24hrs") %>%
  mutate(method = "plus24hrs") %>%
  ungroup()
data_GammPlus48 <- data_GammSim %>%
  dplyr::filter(method=="_plus48hrs") %>%
  mutate(method = "plus48hrs") %>%
  ungroup()
data_GammPlus96 <- data_GammSim %>%
  dplyr::filter(method=="_plus96hrs") %>%
  mutate(method = "plus96hrs") %>%
  ungroup()
data_GammPlus192 <- data_GammSim %>%
```

```
  dplyr::filter(method=="_plus192hrs") %>%
  mutate(method = "plus192hrs") %>%
  ungroup()

# Combine all data for Hypothesis testing
data_H_Tests <- data_Gamm_ALL %>%
  dplyr::select(depid,Foraging,LungeCount,ftle48,time,
                timeHrs,dttzStart,Lat,Long) %>%
  mutate(method = "Real Data",
         deployment = depid) %>%
# add CRW
rbind(.,dplyr::select(data_GammCRW,depid,Foraging,LungeCount,ftle48,time,
                      timeHrs,dttzStart,Lat,Long,method,deployment=FTLE_ID)) %>%
# add plus24
rbind(.,dplyr::select(data_GammPlus24,depid,Foraging,LungeCount,ftle48,time,
                      timeHrs,dttzStart,Lat,Long,method,deployment=FTLE_ID)) %>%
  # add plus48
rbind(.,dplyr::select(data_GammPlus48,depid,Foraging,LungeCount,ftle48,time,
                      timeHrs,dttzStart,Lat,Long,method,deployment=FTLE_ID)) %>%
  # add plus96
rbind(.,dplyr::select(data_GammPlus96,depid,Foraging,LungeCount,ftle48,time,
                      timeHrs,dttzStart,Lat,Long,method,deployment=FTLE_ID)) %>%
  # add plus192
rbind(.,dplyr::select(data_GammPlus192,depid,Foraging,LungeCount,ftle48,time,
                      timeHrs,dttzStart,Lat,Long,method,deployment=FTLE_ID)) %>%
# add Background Points
rbind(.,dplyr::select(data_GammBackground,depid,Foraging,LungeCount,ftle48,time,
                      timeHrs,dttzStart,Lat,Long,method,deployment=FTLE_ID))
# saveRDS(data_H_Tests, file = "./rmdFiles/data_H_Tests.rds")
```

Overview

Marine predators face the challenge of reliably finding prey that is patchily distributed in space and time. Predators make decisions at multiple hierarchically organized spatial and temporal scales, yet we have limited understanding of how habitat selection at multiple scales translates into foraging performance. There is mounting evidence that submesoscale (i.e., <100 km) ocean processes drive the formation of dense prey patches that should hypothetically provide feeding hot spots and increase predator foraging success. Here we integrate environmental remote-sensing with high-resolution ($\geq$ 32 Hz) animal-borne biologging data to evaluate submesoscale surface current features in relation to the dive-scale foraging performance of blue whales in the California Current System.

Methods

Blue Whale Data

Between 2016 and 2020, we deployed 23 high-resolution digital tags on blue whales in the California Current System (CCS). To assess the relationship between blue whale feeding and sub-mesoscale environmental features, we selected a subset of deployments that sampled data for > 24hours, collected GPS positions for > 66% of dives, and overlapped with the sampling footprint of the coastal HF Radar network. Several individuals initiated a southbound migration during the deployment and due to the shifts in behavior documented in Oestreich et al. 2020 were excluded from this study.

The resulting dataset contained:

```
d0 <- data.frame('Summary of Deployments' =
                   c("Number of Individuals",
                     "Number of Years Represented",
                     "Total Number of Dives",
                     "Total Number of Dives with GPS Locations",
                     "Total Number of Dives with Corresponding HF Radar Data"),
                 Total = c(length(unique(data_Gamm_ALL$depid)),
                           length(unique(lubridate::year(data_Gamm_ALL$dttzStart))),
                           length(diveDataset$depid),
                           length(datasetFTLE$depid[!is.na(datasetFTLE$Lat)]),
                           length(data_Gamm_ALL$depid) ))
x<-kable(d0,format='pandoc', align = 'lr', row.names = FALSE,
         col.names = gsub("[.]", " ", names(d0))) %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
 cat(x[3:7], sep="\n")
```

```
## Number of Individuals                                  10
## Number of Years Represented                             5
## Total Number of Dives                                9605
## Total Number of Dives with GPS Locations             8162
## Total Number of Dives with Corresponding HF Radar Data  7791
```

Deployment Overview

```
# Note: We use a pooled Standard Deviation for the overall summary across individuals.
ftleSummary <- datasetFTLE %>%
  group_by(depid) %>%
  summarize(Region=first(region),
            'Animal ID' = first(animalID),
            divesWithGPS = n(),
            divesWithFTLE = sum(!is.na(ftle48)))

diveFeedingRates <- diveDataset %>%
  dplyr::filter(LungeCount >0) %>%
  group_by(depid) %>%
  mutate(# Add a hourly feeding rate for use in HMM
         total_Duration = dive_Duration + surf_Duration,
         feedingRateHR = LungeCount/total_Duration*60) %>%
  summarize('Mean\nDive-By-Dive\nFeeding Rate' = round(mean(feedingRateHR),1),
            'sd\nDive-By-Dive\nFeeding Rate'= round(sd(feedingRateHR),1))

overallSummary <- diveDataset %>%
  mutate(Year = lubridate::year(dttzStart),
         yday= yday(dttzStart),
         hr = hour(dttzStart),
         Foraging = if_else(LungeCount > 0,1,0)) %>%
  group_by(depid) %>%
  mutate(totalDives = n(),
         DeploymentLength = round(difftime(last(dtStart),
                                           first(dtStart),units = "days"),1),
         percentFeeding = sum(Foraging)/n()) %>%
  ungroup() %>%
  group_by(depid,yday) %>%
```

```
  summarize(DeploymentLength = first(DeploymentLength),
            totalDives = first(totalDives),
            percentFeeding = first(percentFeeding),
            dailyLunges = sum(LungeCount),
            firstHour = first(hr),
            lastHour = last(hr),
            Year = first(Year)) %>%
  slice(-1,-n()) %>%
  group_by(depid) %>%
  summarize(Year = first(Year),
            DeploymentLength = first(DeploymentLength),
            totalDives = first(totalDives),
            percentFeeding = round(first(percentFeeding),2),
            'Mean\nDaily\nLunges' = round(mean(dailyLunges),0),
            'sd\nDaily\nLunges' = round(sd(dailyLunges),0),
            'Mean\nOverall Daily\nFeeding Rate' =  round(mean(dailyLunges/24),1),
            'sd\nOverall Daily\nFeeding Rate' = round(sd(dailyLunges/24),1)) %>%
  left_join(ftleSummary,by="depid") %>%
  left_join(diveFeedingRates,by="depid") %>%
  dplyr::select("Deployment\nID"=depid, Region, `Animal ID`, Year,
                'Length\n(days)' = DeploymentLength,
                'Total #\nDives' = totalDives,'Total #\nDives\nwith GPS'=divesWithGPS,
                'Total #\nDives\nwith FTLE'=divesWithFTLE,
                '%\nDives\nFeeding'= percentFeeding,everything())

# Summary dataframe for the calculation of the pooled std dev of
# dive feeding rates
frSummary <- bwFeedingOverall %>%
  group_by(depid) %>%
  summarize(meanFR = mean(feedingRateHR),
            sdFR = sd(feedingRateHR),
            sampleSize = n())
# Summary dataframe for the calculation of the pooled std dev of
# daily feeding rates
dailyFRSummary <- diveDataset %>%
  mutate(yday= yday(dttzStart)) %>%
  group_by(depid,yday) %>%
  summarize(dailyLunges = sum(LungeCount)) %>%
  slice(-1,-n()) %>%
  group_by(depid) %>%
  summarize(meanDailyLunges = round(mean(dailyLunges),0),
            sdDailyLunges = round(sd(dailyLunges),0),
            meanOverallDailyFeedingRate =  round(mean(dailyLunges/24),1),
            sdOverallDailyFeedingRate = round(sd(dailyLunges/24),2) ,
            numDays = n())

totals <- overallSummary %>%
  summarize("Total",
            "",# Region
            "  ",# Animal ID
            length(unique(Year)),
            sum(`Length\n(days)`),
            sum(`Total #\nDives`),
```

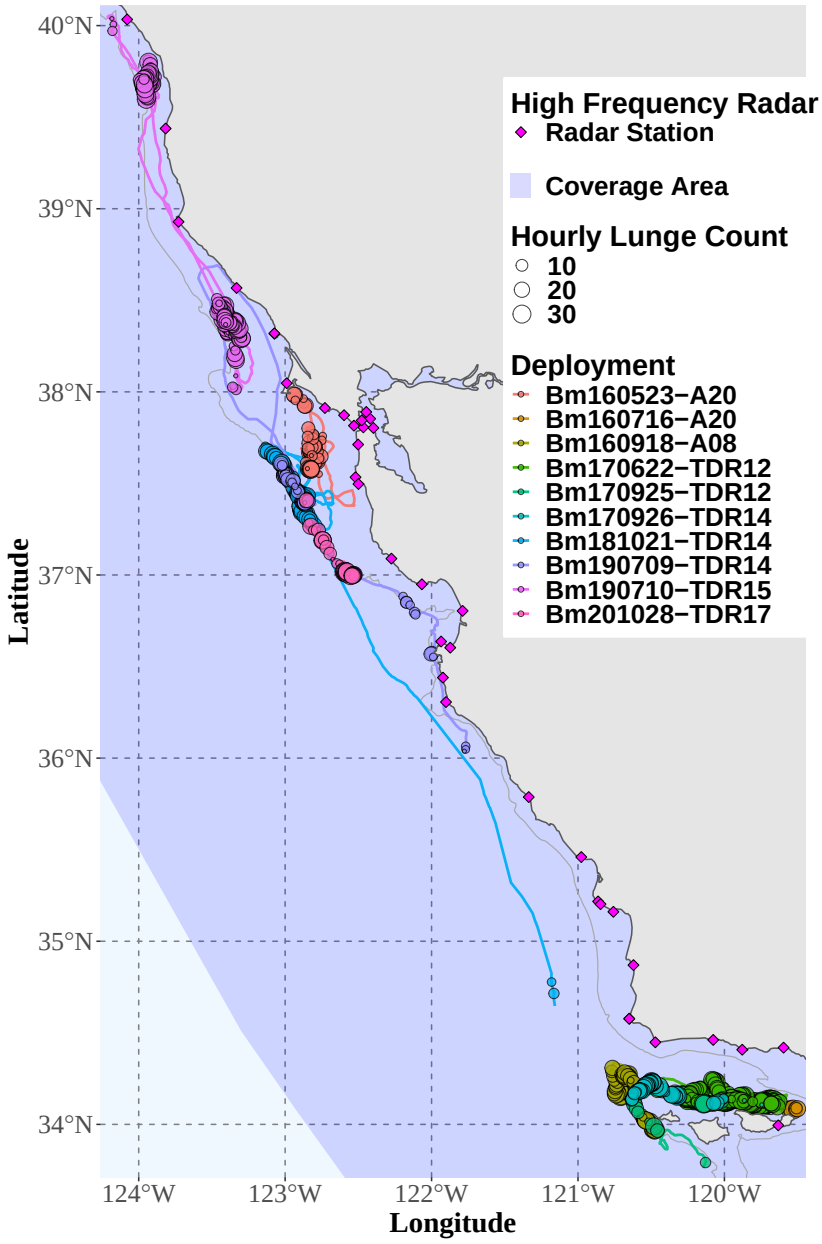
```
            sum(`Total #\nDives\nwith GPS`),
            sum(`Total #\nDives\nwith FTLE`),
            round(mean(`%\nDives\nFeeding`),2),
            round(mean(`Mean\nDaily\nLunges`),0),
            # Daily Lunges Pooled standard dev
            sdDLunges=round(pooledstd(stdevs = dailyFRSummary$sdDailyLunges,
                                      numSamples =  dailyFRSummary$numDays),0),
            round(mean(`Mean\nOverall Daily\nFeeding Rate`),1),
           # Daily Feeding Rate Pooled standard dev
            sdDFR=round(pooledstd(stdevs = dailyFRSummary$sdOverallDailyFeedingRate,
                                  numSamples = dailyFRSummary$numDays),1),
            round(mean(`Mean\nDive-By-Dive\nFeeding Rate`),3),
            # Dive by Dive Feeding Rate Pooled standard dev
            sdDbDFR=round(pooledstd(stdevs = frSummary$sdFR,
                                    numSamples = frSummary$sampleSize),1)
            )
colnames(totals) <- colnames(overallSummary)
d2<-rbind(overallSummary,totals) %>%
  dplyr::select(-`Mean\nDaily\nLunges`,-`sd\nDaily\nLunges`,
                -`Animal ID`)
kable(d2, align = 'lccccccccccc', row.names = FALSE,format="latex") %>%
  column_spec(1,width = "1.14in") %>%
  column_spec(2,width = "0.475in") %>%
  column_spec(3,width = "0.2in") %>%
  column_spec(4,width = "0.32in") %>%
  column_spec(5,width = "0.25in")  %>%
  column_spec(6,width = "0.25in")  %>%
  column_spec(7,width = "0.3in")  %>%
  column_spec(8,width = "0.3in")  %>%
  column_spec(9,width = "0.3in")  %>%
  column_spec(10,width = "0.3in")  %>%
  column_spec(11,width = "0.3in")  %>%
  column_spec(12,width = "0.3in")
```

Deployment ID	Region	Year	Length (days)	Total # Dives	Total # Dives with GPS	Total # Dives with FTLE	% Dives Feeding	Mean Overall Daily Feeding Rate	sd Overall Daily Feeding Rate	Mean Dive-By-Dive Feeding Rate	sd Dive-By-Dive Feeding Rate
Bm160523-A20	Central CA	2016	4.1	853	770	770	0.44	5.8	1.9	24.10	8.5
Bm160716-A20	Southern CA	2016	3.9	462	335	335	0.58	15.2	2.1	29.20	7.5
Bm160918-A08	Southern CA	2016	4.1	907	791	670	0.65	16.9	2.7	25.50	9.1
Bm170622-TDR12	Southern CA	2017	17.7	2728	2588	2569	0.53	14.4	2.8	23.20	5.6
Bm170925-TDR12	Southern CA	2017	4.1	505	456	255	0.18	3.5	2.8	22.00	9.2
Bm170926-TDR14	Southern CA	2017	4.3	373	259	259	0.60	13.8	0.6	25.90	7.7
Bm181021-TDR14	Central CA	2018	9.1	1152	908	908	0.32	8.0	3.4	21.00	5.4
Bm190709-TDR14	Central CA	2019	8.0	731	635	633	0.40	4.4	4.2	17.60	6.7
Bm190710-TDR15	Northern CA	2019	13.5	1502	1062	1039	0.50	10.7	5.7	24.40	8.8
Bm201028-TDR17	Central CA	2020	2.6	392	358	353	0.39	13.8	7.9	24.90	8.4
Total		5	71.4	9605	8162	7791	0.46	10.7	3.9	23.78	7.4

Map of Deployments

```
hourlySum <- datasetFTLE %>%
  mutate(yday = yday(dttzStart),
         hr = hour(dttzStart)) %>%
  group_by(depid,yday, hr) %>% #View()
  summarize(LungeCount = sum(LungeCount),
            Lat = mean(Lat,na.rm=TRUE),
            Long = mean(Long,na.rm=TRUE))
p<-ggplot() +
  geom_sf(data = world,color=NA,fill=NA) +
  geom_polygon(data=fortify(HFRadarCoverageMCP),
               aes(x=long,y=lat, alpha = name),
               color = NA, fill = "blue") +
  # add 200m contour
  geom_contour(data = bf,
               aes(x=x, y=y, z=z),
               breaks=c(-200),
               size=c(0.4),
               colour="darkgrey", show.legend = FALSE) +
  geom_text_contour(data = bf, aes(x=x, y=y,z = z),breaks=c(-200),
                    show.legend = FALSE, size = 2.2, alpha = .6, nudge_y = -.002) +
  # Add whale Locations
  geom_path(data=data_Gamm_ALL, aes(x=Long,y=Lat,group=depid,color=depid),
```

```
            alpha = 0.95,size = .9, show.legend = TRUE) +
  geom_point(data=hourlySum[hourlySum$LungeCount>0,],
             aes(x=Long,y=Lat,group = depid,
                 size = LungeCount, fill = depid),
             alpha = 0.75, shape=21,show.legend = TRUE) +
  geom_sf(data = world) +
  geom_point(data=hfRadarStns,
             aes(st_coordinates(hfRadarStns$geometry)[,1],
                 st_coordinates(hfRadarStns$geometry)[,2],
                 shape = name),size=2.5, fill = 'magenta') +
  scale_alpha_manual(values=c("Coverage Area"=.15))+
  scale_shape_manual(values = c("Radar Station" = 23))+
  coord_sf(xlim = c(min(data_Gamm_ALL$Long,na.rm = TRUE)-.05,
                    max(data_Gamm_ALL$Long,na.rm = TRUE)+.05),
           ylim = c(min(data_Gamm_ALL$Lat,na.rm = TRUE)-.05,
                    max(data_Gamm_ALL$Lat,na.rm = TRUE)+.05),
           expand = FALSE) +
  guides(size = guide_legend(order=4,
                             override.aes = list(linetype = 0)),
         alpha = guide_legend(order=2,
                              override.aes = list(linetype = 0,
                                                  size=0,color=NA)),
         shape = guide_legend(order=1,
                              override.aes = list(linetype = 0)))+
  labs(size = "Hourly Lunge Count",alpha=NULL,
       shape="High Frequency Radar", color="Deployment",fill="Deployment") +
  xlab("Longitude") +
  ylab("Latitude") +
  theme(axis.title = element_text(family="Times",face="bold", size=20),
        axis.text = element_text(family="Times", face="bold", size=18),
        panel.grid.major = element_line(color = gray(.5),
                                        linetype = "dashed", size = 0.5),
        panel.background = element_rect(fill = "aliceblue"),
        legend.text = element_text(face="bold", size=18),
        legend.title = element_text(face="bold", size=20),
        plot.title = element_text(face="bold", size=24,hjust = 0.5),
        legend.position= c(.8,.7),
        legend.key.size = unit(.5, "cm"),
        legend.box.background = element_rect(color="white",
                                             size=1,fill="white"),
        legend.box = "vertical", legend.key = element_blank(),
        legend.margin = margin(0.25,0.15,0.25,0.15, unit="cm"),
        legend.spacing.y = unit(.1, "cm"))
p
```



Whale Data Processing

Tags used in this study were Wildlife Computers TDR10-F (n=7) and Acousonde acoustic tags (n=3). All tags sampled depth, accelerometry (>= 32Hz) and GPS. Tags were deployed from a 6-7m Rigid Hull Inflatable Boat using a 4m carbon fiber pole. Tags were attached to the animal with 3 or 4 stainless steel darts that were 6cm long with 1-2 rows of petals (Szesciorka et al. 2016, Calambokidis et al. 2019). Tags were recovered and the data downloaded after detaching from the animal.

Raw tag data was pre-processed in Matlab using custom scripts (following Cade et al 2016) to calculate the animal's pitch, roll, and speed from the accelerometer sensor data. Individual feeding events (i.e. Lunges) were identified manually using stereotypical signatures in kinematic data. Raw GPS location data was filtered for unrealistic whale speeds (e.g. $>$ 6 meters/second) using the ArgosFilter package (Version 0.62) in R (Version 3.5.1). All processed data was down-sampled to 1Hz for analysis.

Dives were identified as excursions to a depth of greater than 10 meters. Locations were interpolated for

dives with a start time between 2 known positions less than 15 min from the dive's start time. Only dives with a corresponding GPS location were included in this analysis.

Environmental Data

HF Radar Data

For each deployment, surface current data were downloaded from the Southern California Coastal Ocean Observing System (SCCOOS, http://www.sccoos.org/data/hfrnet/) at hourly resolution (cells 6 km on a side) for the deployment period ± 7 days with a bounding box of ± 1 degree around the deployment locations.

Data gaps in the raw HF radar surface current measurements were restored using the algorithms described in Ameli and Shadden (2019), selecting a concave hull (alpha shape radius of 10 km) with the detection and exclusion of land.

FTLE Calculation

Using the restored surface current data, we calculated the backward-in-time FTLE using TRACE (http://transport.me.berkeley.edu/trace/), a Lagrangian analysis tool that follows the methodology described in Shadden et al. (2005; 2009). We used an evenly-spaced grid of tracers having 10 times the spatial resolution of the HF radar data (sensu Shadden et al., 2009), and applied a free-slip boundary condition to tracers near land. Tracer advection used a bilinear spatial interpolation and an adaptive 4th order Runge-Kutta-Fehlberg integration method. At every hourly time-step, the trajectories of the evenly spaced grid of tracers were integrated for the preceding 48 hour period, and FTLE was calculated from the time-dependent movement of tracer trajectories.

*TRACE Parameters +Initial Time: 1 day prior to Tag On Animal +Number of Initial Times: Number of hours in the deployment + 48 hours (this ensures the data will envelop the deployment by ± 1 day) +Initial Times Increment: 1 Hour +Integration Duration: 48 Hours +Reverse Direction of Time +Number of Tracers: FTLE is calculated with a grid of tracers having 10x the resolution of the input data. - e.g. the 6km HF radar data for a deployment may have a grid 39 X 49 cells (Long X Lat). The number of tracers for this deployment would be 390 X 490.

*TRACE Output
+The final resolution varies slightly by deployment, but is roughly 590m (i.e., 6km HF radar resolution/10).

Overlap with Environmental Data

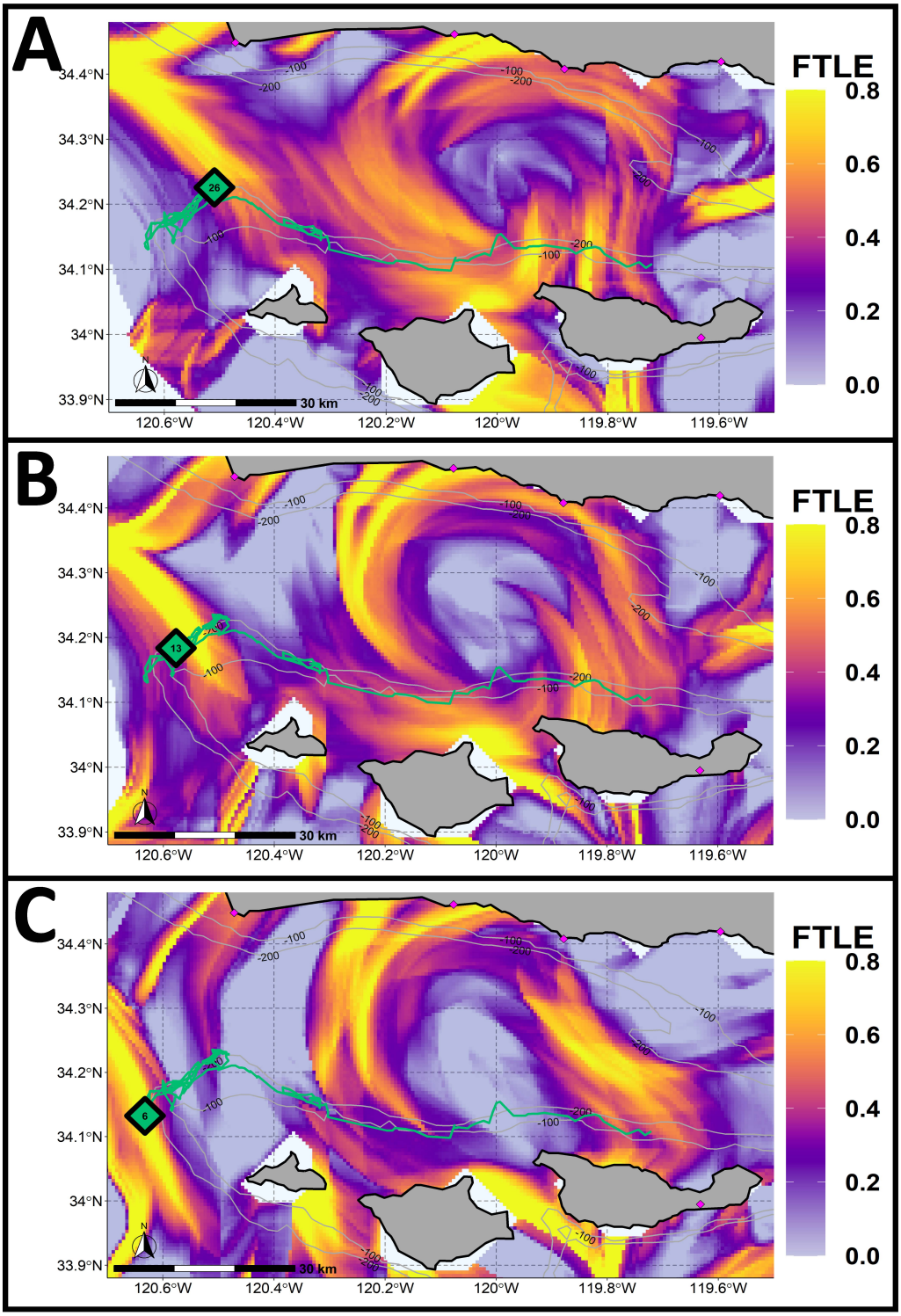
The sampling unit for this analysis is Blue Whale Dives and are classified as either feeding or non-feeding. Dives also have an associated Feeding Rate, which is the # of lunges per dive minute + post-dive surface period.

FTLE is sampled at hourly intervals, with each FTLE value representing ± 30 minutes from the sample timestamp. For each blue whale dive location, we extract the FTLE value from the corresponding FTLE layer.

The pattern we explore quantitatively is shown in the animation below.

Statistical Analysis

The movements of free-ranging predators can be described in terms of habitat utilization in which the predator uses a subset of the habitat available. The utilization is considered selective when a predator targets a certain set of features disproportionately to their availability in the environment (Johnson 1980). Within the hierarchical framework described by Johnson 1980, there have been many works describing the lower order selection types of blue whales using satellite tags (e.g. Abrahms et al 2018), which relates to the selection of a home range. Additionally, recent work by Cade et al 2021 has shed light on some of the highest order selection types, such as the selction for the densest prey within a patch (e.g. 4th order).



Panels A, B and C show 3 consecutive days at 14:00 (Local Time) of FTLE calculated from simulated particle trajectories after a 48-hour integration with the track from a 4-day blue whale deployment shown in green (100 and 200m isobaths shown in grey). Black diamonds represent the mean hourly location and number of lunges for the specific hour of FTLE data shown.

Figure 1: Example animation of Blue Whale and FTLE

In this manusript, we are contributing to a knowledge gap of intermediate order selection processes. This includes the 3rd order, which pertains to the usage of habitat within a home range.

Distribution Analyses

We first used nonparametric two-sample Kolmogorov-Smirnov tests to determine whether blue whales select areas of high FTLE when feeding. We performed the test for all blue whales in our study collectively and individually.

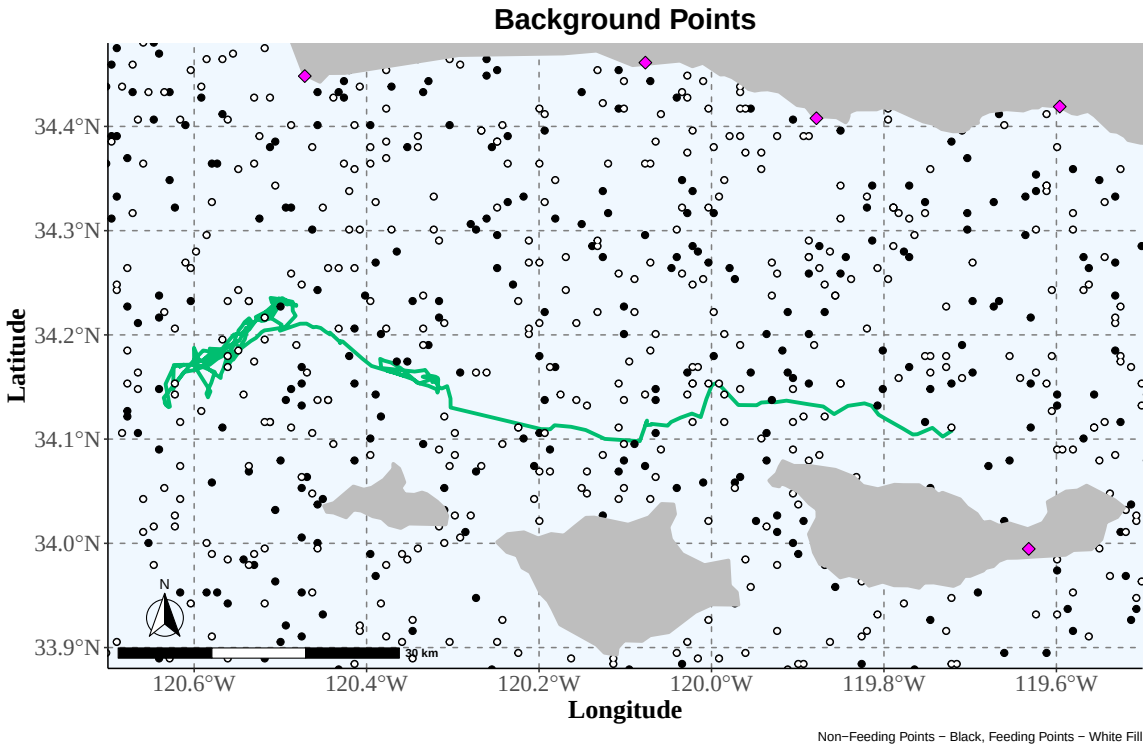
These distribution tests are important because they illustrate habitat selection within the foraging grounds, which aligns with the 3rd order process (i.e. selection of habitat areas within the home or subpopulation range) from Johnson 1980.

Background Sampling

To simulate a random distribution of FTLE values that an individual could have encountered in its surrounding environment (i.e. background), random locations were drawn from the spatial domain for each individual. Locations were generated at a 10:1 ratio of background locations to real locations, and we extracted the FTLE value of the background locations for the time-period of their corresponding animal location.

```
pBackground <- ggplot() +
  geom_sf(data = world,color=NA,fill=NA) +
  # add contours
  ggplot2::geom_contour(data = bf,
                        aes(x=x, y=y, z=z),
                        breaks=intervals,
                        size=c(0.4),
                        colour="darkgray", show.legend = FALSE) +
  geom_path(data=data_Gamm_ALL %>% dplyr::filter(depid == "Bm170926-TDR14"),
            aes(Long,Lat),size=1.25,alpha=1, color="#00BE70")+
  geom_point(data=data_GammBackground %>%
               dplyr::filter(depid == "Bm170926-TDR14"),
             aes(Long,Lat),size=2,alpha=1, color="black")+
  geom_point(data=data_GammBackground %>%
               dplyr::filter(depid == "Bm170926-TDR14",LungeCount>0),
             aes(Long,Lat),size=1,alpha=1, color="white")+
  geom_sf(data = world, size=1.25,fill="grey",color="grey") +
  geom_point(data=hfRadarStns,
             aes(st_coordinates(hfRadarStns$geometry)[,1],
                 st_coordinates(hfRadarStns$geometry)[,2],
                 size = name), fill = 'magenta', shape=23,show.legend = FALSE) +
  scale_size_manual(values = c("Radar Station" = 3), name="High Frequency Radar") +
  coord_sf(xlim = c(-120.7, -119.5),
           ylim = c(33.88, 34.48), expand = FALSE) +
  theme_classic() +
  annotation_scale(location = "bl", width_hint = 0.3,
                   text_face = "bold", text_col = "black") +
  annotation_north_arrow(location = "bl",
                         which_north = "true",
                         pad_x = unit(0.25, "in"),
                         pad_y = unit(0.25, "in"),
                         style = north_arrow_fancy_orienteering) +
  xlab("Longitude") +
  ylab("Latitude") +
  guides(color = guide_legend(order=1,direction = 'vertical', spacing.y=unit(.1, "cm")),
```

```
         shape = guide_legend(order=1,direction = 'vertical', spacing.y=unit(.1, "cm")),
         stroke = guide_legend(order=1,direction = 'vertical', spacing.y=unit(.1, "cm")),
         size = guide_legend(order=3,direction = 'vertical', spacing.y=unit(.1, "cm"),
                             override.aes = list(linetype = 1,shape=23)))+
  ggtitle("Background Points") +
  labs(caption = "Non-Feeding Points - Black, Feeding Points - White Fill")+
  theme(panel.grid.major = element_line(color = gray(.5),
                                        linetype = "dashed", size = 0.5),
        panel.background = element_rect(fill = "aliceblue"),
        axis.title = element_text(family="Times",face="bold", size=18),
        axis.text = element_text(family="Times",face="bold", size=16),
        legend.text = element_text(face="bold", size=18),
        legend.position = "bottom", legend.box = 'horizontal',
        legend.key.size = unit(.75, "cm"),
        legend.title = element_text(face="bold", size=20),
        legend.box.background = element_rect(color="white", size=1,fill="white"),
        legend.spacing.y = unit(.1, "cm"),legend.spacing.x = unit(1, "cm"),
        plot.margin=unit(c(0,2.5,0,0.25), "cm"),
        plot.title = element_text(face="bold", size=20, hjust = 0.5),
        plot.subtitle = element_text(face="bold", size=20, hjust = 0.5))
pBackground
```

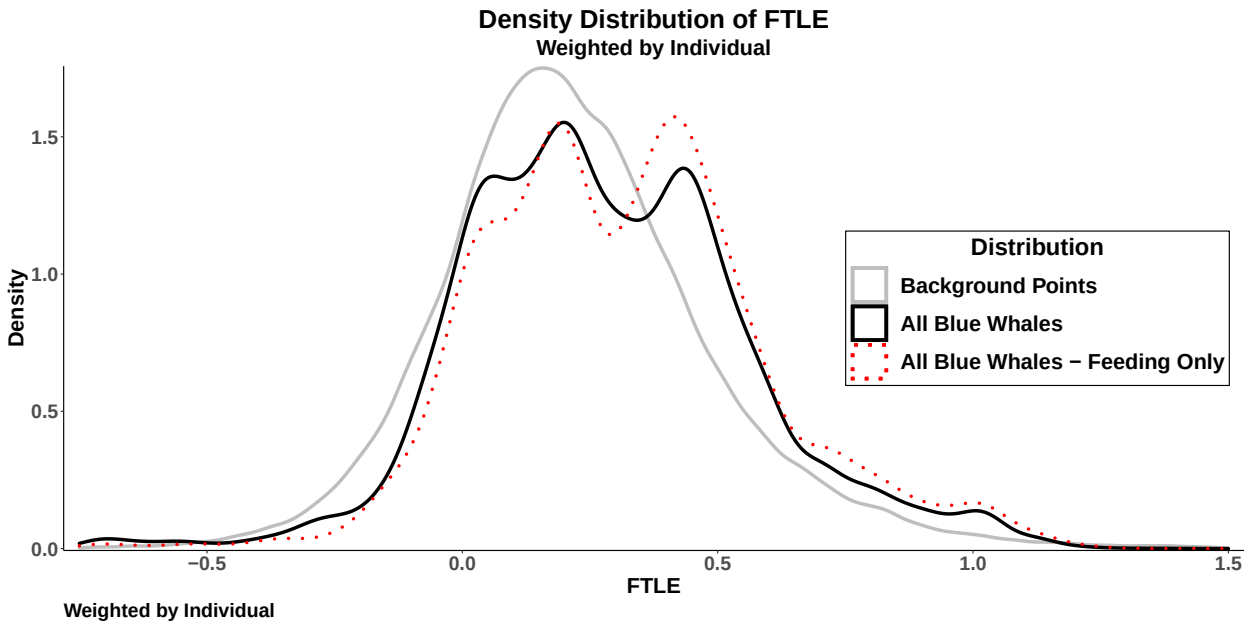


Density Plot

To account for varying deployment lengths, the density plot is weighted by individual.

```r
# Create a weight column to account for deployment length
bwAll <- data_Gamm_ALL %>%
  group_by(depid) %>%
  mutate(weight = 1/(n()*length(unique(data_Gamm_ALL$depid))),
         Distribution = "All") %>%
  ungroup()
bwFeeding<-bwFeeding %>%
  group_by(depid) %>%
  mutate(weight = 1/(n()*length(unique(bwFeeding$depid))),
         Distribution = "Feeding") %>%
  ungroup()
# Study area (Background) distribution (weighted by individual)
data_GammBackground <- data_GammBackground %>%
  group_by(depid) %>%
  mutate(weight = 1/(n()*length(unique(data_GammBackground$depid))),
         Distribution = "Background") %>%
  ungroup()
# Plot
pW<-ggplot() +
  geom_density(data=data_GammBackground,
               aes(x=ftle48,weight=weight,
                   linetype = "Background Points",
                   color= "Background Points"),
               size=1.5) +
  geom_density(data=bwAll,
                aes(x=ftle48,weight=weight,
                    linetype = paste0(Distribution, " Blue Whales"),
                    color=paste0(Distribution, " Blue Whales")),
                size=1.5) +
  geom_density(data=bwFeeding,
                aes(x=ftle48,weight=weight,
                    linetype = paste0("All Blue Whales - ", Distribution, " Only"),
                    color=paste0("All Blue Whales - ", Distribution, " Only")),
                size=1.5) +
  scale_linetype_manual(name = "Distribution",
                        values=c( "Background Points" = 1,
                                  "All Blue Whales" = 1,
                                  "All Blue Whales - Feeding Only" = 3))+
  scale_color_manual(name = "Distribution",
                     values=c( "Background Points" = "grey",
                               "All Blue Whales" = "black",
                               "All Blue Whales - Feeding Only" = "red"))+
  ylab("Density")+
  scale_x_continuous(name="FTLE",limits = c(-.75,1.5),
                     expand = c(0.0025,0.025), breaks = c(-0.5,0,0.5,1,1.5))+
  scale_y_continuous(name= "Density", expand=c(0.0025,0.0025) )+
  ggtitle("Density Distribution of FTLE", subtitle = "Weighted by Individual") +
  labs(caption = "Weighted by Individual")+
  theme_classic()+
  theme(axis.title = element_text(face="bold", size=20),
        axis.text = element_text( face="bold", size=18),
        plot.title = element_text(face="bold", size=24,hjust = 0.5),
        plot.subtitle = element_text( face="bold", size=20,hjust = 0.5),
```

```
        plot.caption = element_text( face="bold", size=18,hjust = 0),
        legend.text = element_text(face="bold", size=20),
        legend.title = element_text(face="bold", size=22,hjust = 0.5),
        legend.margin = margin(0.15,0.15,0.15,0.15, unit="cm"),
        legend.background = element_rect(color='black'),
        legend.key.size = unit(1.25,"cm"),
        legend.spacing.y = unit(.25, "cm"),
        legend.position = c(.825,.5))
pW
```



K-S Tests

```
#### KS-Test for All Together ####
ks_ftle<-ksResults(data1=bwAll$ftle48,data2=data_GammBackground$ftle48,
                   alt="less",Distribution='All',ID="All")
cat(paste0("Weighted Mean of FTLE (Blue Whales): ",
           round(MetricsWeighted::weighted_mean(x=bwAll$ftle48,
                                                w=bwAll$weight ),3)))
```

```
## Weighted Mean of FTLE (Blue Whales): 0.279
```

```
cat(paste0("Weighted Mean of FTLE (Background Points): ",
           round(MetricsWeighted::weighted_mean(x=data_GammBackground$ftle48,
                                                w=data_GammBackground$weight ),3)))
```

```
## Weighted Mean of FTLE (Background Points): 0.221
```

```
cat(paste0("K-S Test Result for All Blue Whales: \nD-Statistic ",
           round(ks_ftle$dvalue,4),
           "\np-Value ", round(ks_ftle$pvalue,5)))
```

```
## K-S Test Result for All Blue Whales:
## D-Statistic 0.0617
## p-Value 0
```

```
cat(paste0("# Feeding Dives with Locations: ",length(bwFeeding$depid)))
```

```
## # Feeding Dives with Locations: 3875
```

For each individual, we used a Kolmogorov-Smirnov test to compare the distributions of FTLE at the animal's location to the distribution of FTLE for randomly selected backkground points in the study area for that particular individual.

```
## Process Individual Deployments ##
ksOutput <- data.frame()
for(i in 1:length(unique(data_Gamm_ALL$depid))){
  # Get the depid and time zone for this deployment
  depid <- unique(data_Gamm_ALL$depid)[i]
  # filter by ID and remove dives without location data
  data_Sub <- data_Gamm_ALL %>%
    dplyr::filter(depid == unique(data_Gamm_ALL$depid)[i])
  # Subset the BackgroundData
  backgroundSub <- data_GammBackground %>%
    dplyr::filter(depid == unique(data_Gamm_ALL$depid)[i])
  ks_ftle_All<-ksResults(data1=data_Sub$ftle48,
                         data2=backgroundSub$ftle48,alt="less",
                         Distribution = 'ALLvsBkgd',
                         ID=unique(data_Gamm_ALL$depid)[i])
  # Save To OutputDF
  ksOutput <- rbind(ksOutput,ks_ftle_All)
  rm(data_Sub,backgroundSub,depid)
}
rm(i)
ksOutput %>% dplyr::select("Deployment ID"=ID,dvalue,pvalue)
```

Deployment ID	dvalue	pvalue
Bm160523-A20	0.0684416	0.0014188
Bm160716-A20	0.3053731	0.0000000
Bm160918-A08	0.1480597	0.0000000
Bm170622-TDR12	0.1081355	0.0000000
Bm170925-TDR12	0.2603922	0.0000000
Bm170926-TDR14	0.4223938	0.0000000
Bm181021-TDR14	0.2335903	0.0000000
Bm190709-TDR14	0.1156398	0.0000002
Bm190710-TDR15	0.1614052	0.0000000
Bm201028-TDR17	0.2963173	0.0000000

```
table1 <- d2 %>%
  left_join(dplyr::select(ksOutput, ID,"KS test D-value" = dvalue,
                          "KS test p-value" = pvalue),
            by= c("Deployment\nID"="ID"))
```

Probability of Feeding

To further explore the relationship between blue whale feeding behavior and FTLE, we compared the probability of feeding for the real whale data to that of several null models that control for spatial and temporal variability. To account for individual variation, we used a generalized mixed-effects model. The models incorporate a Continuous AR1 structure (corCAR1), which is a continuous-time first-order autocorrelation model to account for serial autocorrelation in the tag data. We fit a Generalized Linear Mixed Model via Penalized Quasi-Likelihood using the glmmPQL function of the MASS package (version 7.3-53).To facilitate

hypothesis testing in assessing feeding site selection, we generated null model datasets that control for either spatial or temporal variation. All null model analyses followed the same procedures as the blue whale data.

Random Site Selection Model

To determine whether blue whale feeding locations are significantly different from those of non-selecting individuals (Random Site Selection Model), we used the adehabitatLT package (v0.4.19) to create 10 randomized, simulated tracks for each individual. The simulated tracks were implemented as a correlated random walk (CRW) that use the same timestamps of feeding/non-feeding locations (i.e. retaining the ACF of the original data), but randomized the locations. We used the adehabitatHR package to calculate the scaling parameter (h) and the concentration parameter of turning angles (r) for each individual. We used the simm.crw function to simulate a correlated random walk for 10 tracks per individual, parameterized by the individual h and r values and using 10 random starting points from within the minimum convex polygon of each deployment. None of the simulations have points on land, and all simulated tracks use the same timestamps and feeding/non-feeding designation as the original data.

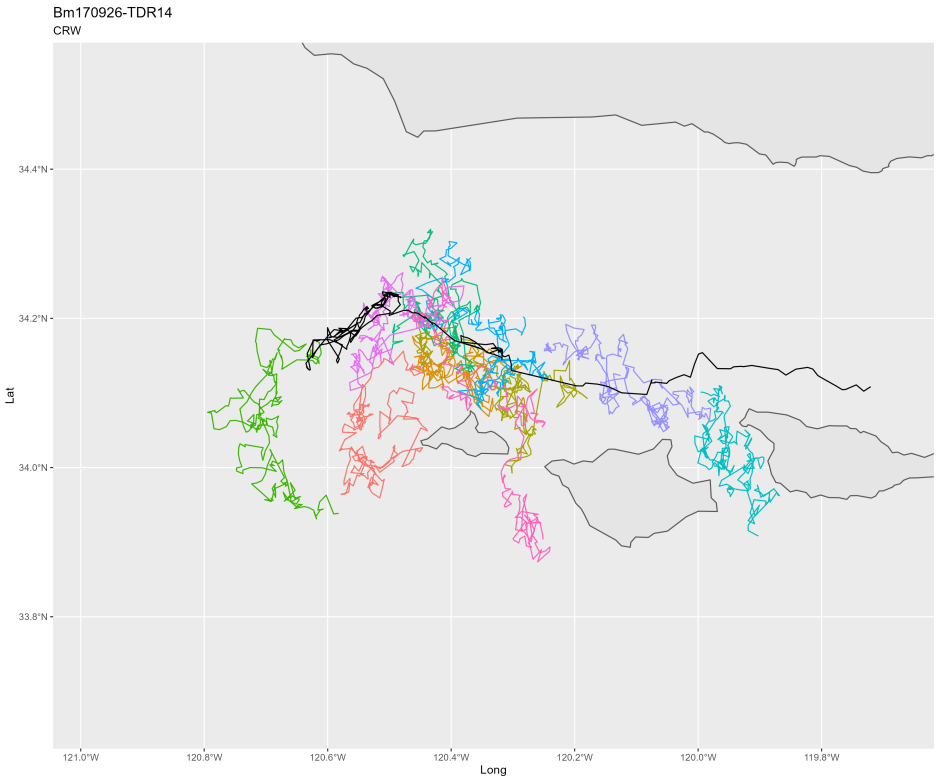


Figure 2: Example of track simulations for 1 individual

Temporal Null Model

To determine whether there is a temporal relationship with whale feeding locations, we shifted the time of the movement data for each individual forward in time by 24, 48, 96 and 192 hours , which is a way of controlling for location (*i.e. all of the feeding locations are identical to the real whale data*). This addresses the question of whether the FTLE features are tied to specific locations (null model), or if there is a temporal relationship. The time-shifted tracks for each individual retains the ACF of the original data.

Feeding GLMM

(Feeding ~ FTLE) for real whale data

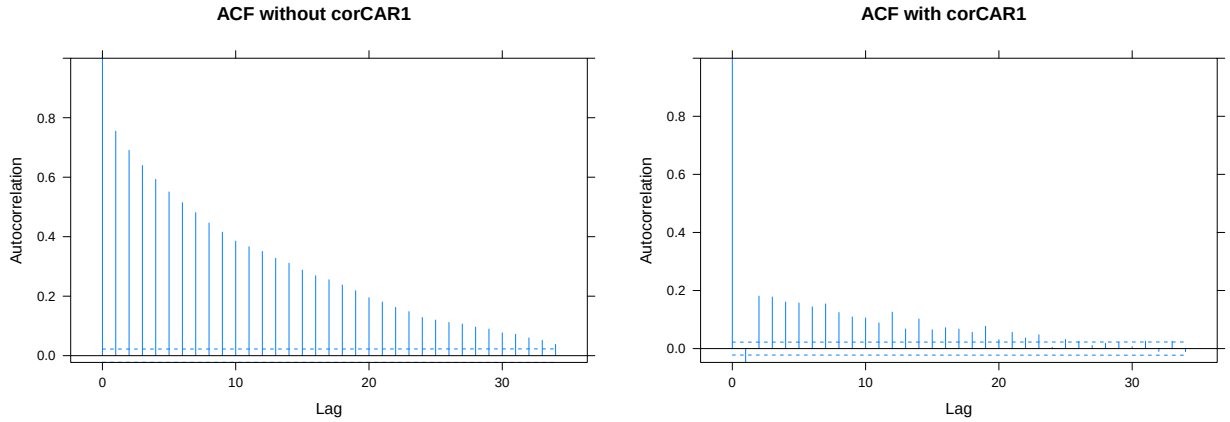
```
# Use previously processed model to save time
if(!exists("feed_glmmPQL_FTLE_CAR1_Hrs")){
# Use Continuous AR1 structure
feed_glmmPQL_FTLE_CAR1_Hrs <-
  glmmPQL(Foraging ~ ftle48 ,
          random = ~ 1 | depid,
          correlation=corCAR1(form=~ timeHrs|depid),
          family = binomial,
          data = data_Gamm_ALL)
# Save Model results to a file
saveRDS(feed_glmmPQL_FTLE_CAR1_Hrs,
        file = "./Models/feed_glmmPQL_FTLE_CAR1_Hrs.rds")
}
summary(feed_glmmPQL_FTLE_CAR1_Hrs)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: data_Gamm_ALL
##    AIC BIC logLik
##     NA  NA     NA
##
## Random effects:
##  Formula: ~1 | depid
##         (Intercept) Residual
## StdDev:   0.5057144 1.049903
##
## Correlation Structure: Continuous AR(1)
##  Formula: ~timeHrs | depid
##  Parameter estimate(s):
##       Phi
## 0.0323693
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: Foraging ~ ftle48
##                  Value Std.Error   DF    t-value p-value
## (Intercept) -0.1543181 0.1734333 7780 -0.8897837  0.3736
## ftle48       0.4481486 0.1479604 7780  3.0288423  0.0025
##  Correlation:
##        (Intr)
## ftle48 -0.224
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.5647991 -0.9174661 -0.6004437  0.7962540  1.5208139
##
## Number of Observations: 7791
## Number of Groups: 10
```

The autocorrelation of the data is reduced using the correlation structure.

```
# check model without the temporal autocorrelation term
feed_glmmPQL_FTLE <- update(feed_glmmPQL_FTLE_CAR1_Hrs,correlation = NULL)
```

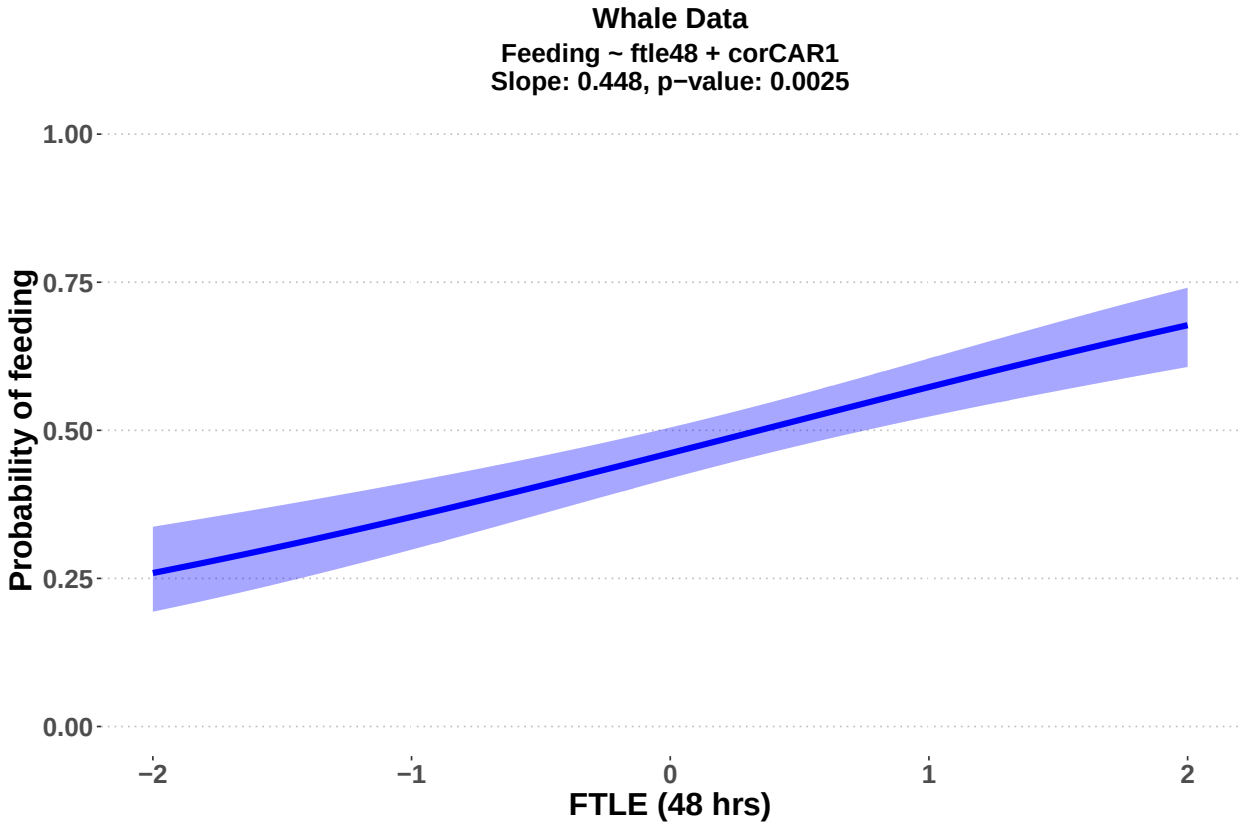
```
plot(ACF(feed_glmmPQL_FTLE,resType="normalized"),
     alpha=0.05, main="ACF without corCAR1")
plot(ACF(feed_glmmPQL_FTLE_CAR1_Hrs,resType="normalized"),
     alpha=0.05, main="ACF with corCAR1")
```



Plot the model

```
testSet <- data.frame(ftle48 = seq(-2, 2, length.out = 100))
# This function outputs the predicted values in probability space
testSet$fit <- predict(feed_glmmPQL_FTLE_CAR1_Hrs,testSet,
                       type = "response", level=0)
# This function outputs the predicted values in Logit space
SE <- predictSE(feed_glmmPQL_FTLE_CAR1_Hrs,testSet,
                type = "response", level = 0,se.fit = TRUE)
# Convert the Logit Standard errors into Probabilities
testSet$lwr <- logit2prob(SE$fit-SE$se.fit)
testSet$upr <- logit2prob(SE$fit+SE$se.fit)
testSet$model <- "Whale Data"
ggplot(data=testSet,aes(x=ftle48,y=fit)) +
  geom_ribbon(data=testSet, aes(x=ftle48,
                                ymin = lwr, ymax = upr, color = NULL),
              fill="blue", alpha = .35) +
  geom_line(data=testSet, aes(y = fit),color="blue", size = 2) +
  ggtitle("Whale Data",
          subtitle= paste0("Feeding ~ ftle48 + corCAR1\n","Slope: ",
                           round(as.data.frame(
                             coef(summary(
                               feed_glmmPQL_FTLE_CAR1_Hrs)))$Value[2],
                             3), ", p-value: ",
                           round(as.data.frame(
                             coef(summary(
                               feed_glmmPQL_FTLE_CAR1_Hrs)))$'p-value'[2],
                             4)))+
  coord_cartesian(ylim = c(0, 1)) +
  labs(x = "FTLE (48 hrs)", y = "Probability of feeding") +
  theme_pubclean()+
  theme(axis.title = element_text(face="bold", size=22),
        axis.text = element_text( face="bold", size=18),
        plot.title = element_text(face="bold", size=20,hjust = 0.5),
        plot.subtitle = element_text( face="bold", size=18,hjust = 0.5),
```

```
        plot.caption = element_text( face="bold", size=10,hjust = 0),
        legend.position="bottom",
        legend.text = element_text(face="bold", size=20),
        legend.title = element_text(face="bold", size=24),
        strip.text.x = element_text(size = 22, face="bold"))
```



Null Model GLMMs

To evaluate the Random Site Selection Model we used a mixed effects model to evaluate the significance of the relationship.

For the temporal persistence analysis, we hypothesized that if FTLE features are location-specific and persistent for multiple days, we would observe the same relationship across all time-shifted models and the real data.

Random Site Selection Model Analysis of the randomized simulated tracks (e.g. correlated random walks) showed a negative relationship with FTLE, though the slope was not significantly different from 0.

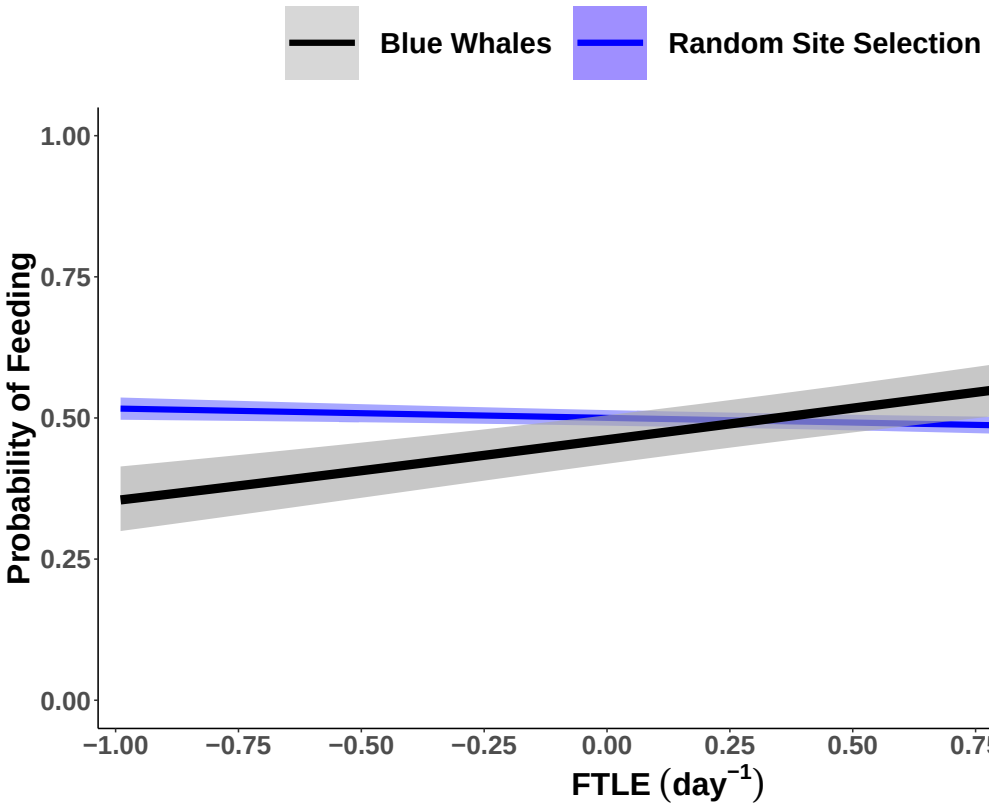
```
# Use previously processed model to save time
if(!exists("feed_glmmPQL_CRW_CAR1_Hrs")){
  feed_glmmPQL_CRW_CAR1_Hrs <-
    glmmPQL(Foraging ~ ftle48 ,
            random = ~ 1 | depid,
            correlation=corCAR1(form=~timeHrs|depid),
            family = binomial,
            data = data_GammCRW)
  saveRDS(feed_glmmPQL_CRW_CAR1_Hrs,
          file = "./Models/feed_glmmPQL_CRW_CAR1_Hrs.rds")
```

```
}
testSetSimmCRW <- data.frame(ftle48 = seq(-2, 2, length.out = 100))
# This function outputs the predicted values in probability space
testSetSimmCRW$fit <- predict(feed_glmmPQL_CRW_CAR1_Hrs,
                              testSetSimmCRW, type = "response", level=0)
# This function outputs the predicted values in Logit space
SE_CRW <- predictSE(feed_glmmPQL_CRW_CAR1_Hrs,testSetSimmCRW,
                    type = "response", level = 0,se.fit = TRUE)
# Convert the Logit Standard errors into Probabilities
testSetSimmCRW$lwr <- logit2prob(SE_CRW$fit-SE_CRW$se.fit)
testSetSimmCRW$upr <- logit2prob(SE_CRW$fit+SE_CRW$se.fit)
testSetSimmCRW$model <- "CRW Tracks"
summary(feed_glmmPQL_CRW_CAR1_Hrs)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: dataCRW
##    AIC BIC logLik
##     NA  NA     NA
##
## Random effects:
##  Formula: ~1 | depid
##         (Intercept) Residual
## StdDev:    0.509523 1.050497
##
## Correlation Structure: Continuous AR(1)
##  Formula: ~timeHrs | depid
##  Parameter estimate(s):
##        Phi
## 0.03237509
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: Foraging ~ ftle48
##                   Value  Std.Error    DF    t-value p-value
## (Intercept)  0.00063168 0.05524520 75530  0.0114342  0.9909
## ftle48      -0.06623883 0.04668666 75530 -1.4187955  0.1560
##  Correlation:
##        (Intr)
## ftle48 -0.205
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.4714497 -0.9253140 -0.6446639  0.7760937  1.4231437
##
## Number of Observations: 75631
## Number of Groups: 100
```

```
## Figure 3 A ####
# Random Site Selection Model
ggplot() +
  geom_ribbon(data=testSetSimmCRW,
              aes(x=ftle48,ymin=lwr, ymax=upr,
```

```
                fill="Random Site Selection Model"), alpha = .35) +
geom_line(data=testSetSimmCRW,
          aes(x=ftle48,y = fit,
              color = "Random Site Selection Model"),
          size = 2)+
geom_ribbon(data=testSet,
            aes(x=ftle48,ymin = lwr, ymax = upr,
                fill = "Blue Whales"), alpha = .65) +
geom_line(data=testSet,
          aes(x=ftle48,y = fit, color = "Blue Whales"),
          size = 3)+
scale_fill_manual(name= NULL,
                  values = c( "Blue Whales" = "dark grey",
                              "Random Site Selection Model" = "blue")) +
scale_color_manual(name= NULL,
                   values = c( "Blue Whales" = "black",
                               "Random Site Selection Model" = "blue")) +
guides(fill = guide_legend(direction="horizontal",
                           override.aes = list(fill=c('gray80','slateblue1'),
                                               size=2)),
       color = guide_legend(direction="horizontal"))+
coord_cartesian(ylim = c(0, 1)) +
scale_x_continuous(limits = c(-1,1.25),
                   expand = c(0.0025,0.03),
                   breaks = seq(-1.25,1.5,.25))+
xlab(expression( bold(FTLE~(day^bold({bold("-1")})))) )+
ylab("Probability of Feeding") +
theme_classic()+
theme(axis.title = element_text(face="bold", size=22),
      axis.text = element_text( face="bold", size=18),
      plot.title = element_text(face="bold", size=20,hjust = 0.5),
      plot.subtitle = element_text( face="bold", size=18,hjust = 0.5),
      plot.caption = element_text( face="bold", size=10,hjust = 0),
      legend.position= 'top',
      legend.key.size = unit(2,"cm"),
      legend.background = element_rect(color=NA),
      legend.key = element_rect(fill=NA,color=NA),
      legend.text = element_text(face="bold", size=20),
      legend.title = element_text(face="bold", size=24),
      strip.text.x = element_text(size = 22, face="bold"))
```



Plot CRW + Blue Whale Models

```
ggsave("Output/Figure4a.png", width=12, height=8, units = "in", dpi=400)
```

Temporal Persistence When we explored the temporal persistence of FTLE features, we found that the relationship between blue whale feeding site selection and FTLE did not persist for temporal shifts greater than 24-hours.

```
# Use previously processed model to save time
if(!exists("feed_glmmPQL_p24_CAR1_Hrs")){
  feed_glmmPQL_p24_CAR1_Hrs <-
    glmmPQL(Foraging ~ ftle48 ,
            random = ~ 1 | depid,
            correlation=corCAR1(form=~timeHrs|depid),
            family = binomial,
            data = data_H_Tests[data_H_Tests$method == "plus24hrs",])
  saveRDS(feed_glmmPQL_p24_CAR1_Hrs,
          file = "./Models/feed_glmmPQL_p24_CAR1_Hrs.rds")
}
testSetp24 <- data.frame(ftle48 = seq(-2, 2, length.out = 100))
# This function outputs the predicted values in probability space
testSetp24$fit <- predict(feed_glmmPQL_p24_CAR1_Hrs,testSetp24,
                          type = "response", level=0)
# This function outputs the predicted values in Logit space
SE_p24 <- predictSE(feed_glmmPQL_p24_CAR1_Hrs,testSetp24,
                    type = "response", level = 0,se.fit = TRUE)
# Convert the Logit Standard errors into Probabilities
testSetp24$lwr <- logit2prob(SE_p24$fit-SE_p24$se.fit)
```

```
testSetp24$upr <- logit2prob(SE_p24$fit+SE_p24$se.fit)
testSetp24$model <- "Plus 24hrs"
summary(feed_glmmPQL_p24_CAR1_Hrs)
```

24 Hour Time-Shift

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: data_H_Tests[data_H_Tests$method == "plus24hrs", ]
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | depid
##         (Intercept) Residual
## StdDev:   0.4967084 1.049073
##
## Correlation Structure: Continuous AR(1)
##  Formula: ~timeHrs | depid
##  Parameter estimate(s):
##        Phi
## 0.03276111
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: Foraging ~ ftle48
##                  Value Std.Error   DF    t-value p-value
## (Intercept) -0.1495937 0.1704299 7780 -0.8777429  0.3801
## ftle48       0.4905349 0.1607848 7780  3.0508787  0.0023
##  Correlation:
##        (Intr)
## ftle48 -0.218
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.6149449 -0.9241184 -0.5708620  0.8111037  1.6208082
##
## Number of Observations: 7791
## Number of Groups: 10
```

```
# Use previously processed model to save time
if(!exists("feed_glmmPQL_p48_CAR1_Hrs")){
  feed_glmmPQL_p48_CAR1_Hrs <-
    glmmPQL(Foraging ~ ftle48 ,
            random = ~ 1 | depid,
            correlation=corCAR1(form=~timeHrs|depid),
            family = binomial,
            data = data_H_Tests[data_H_Tests$method == "plus48hrs",])
  saveRDS(feed_glmmPQL_p48_CAR1_Hrs,
          file = "./Models/feed_glmmPQL_p48_CAR1_Hrs.rds")
}
testSetp48 <- data.frame(ftle48 = seq(-2, 2, length.out = 100))
# This function outputs the predicted values in probability space
testSetp48$fit <- predict(feed_glmmPQL_p48_CAR1_Hrs,testSetp48,
```

```
                          type = "response", level=0)
# This function outputs the predicted values in Logit space
SE_p48 <- predictSE(feed_glmmPQL_p48_CAR1_Hrs,testSetp48,
                    type = "response", level = 0,se.fit = TRUE)
# Convert the Logit Standard errors into Probabilities
testSetp48$lwr <- logit2prob(SE_p48$fit-SE_p48$se.fit)
testSetp48$upr <- logit2prob(SE_p48$fit+SE_p48$se.fit)
testSetp48$model <- "Plus 48hrs"
summary(feed_glmmPQL_p48_CAR1_Hrs)
```

48 Hour Time-Shift

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: data_H_Tests[data_H_Tests$method == "plus48hrs", ]
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | depid
##         (Intercept) Residual
## StdDev:   0.5114645 1.050113
##
## Correlation Structure: Continuous AR(1)
##  Formula: ~timeHrs | depid
##  Parameter estimate(s):
##       Phi
## 0.03315746
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: Foraging ~ ftle48
##                  Value Std.Error   DF    t-value p-value
## (Intercept) -0.0876077 0.1748697 7780 -0.5009886  0.6164
## ftle48       0.2290243 0.1689419 7780  1.3556398  0.1753
##  Correlation:
##        (Intr)
## ftle48 -0.215
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.6398414 -0.9192960 -0.6087158  0.7805321  1.4669732
##
## Number of Observations: 7791
## Number of Groups: 10
```

```
# Use previously processed model to save time
if(!exists("feed_glmmPQL_p96_CAR1_Hrs")){
  feed_glmmPQL_p96_CAR1_Hrs <-
    glmmPQL(Foraging ~ ftle48 ,
            random = ~ 1 | depid,
            correlation=corCAR1(form=~timeHrs|depid),
            family = binomial,
            data = data_H_Tests[data_H_Tests$method == "plus96hrs",])
```

```
  saveRDS(feed_glmmPQL_p96_CAR1_Hrs,
          file = "./Models/feed_glmmPQL_p96_CAR1_Hrs.rds")
}
testSetp96 <- data.frame(ftle48 = seq(-2, 2, length.out = 1000))
# This function outputs the predicted values in probability space
testSetp96$fit <- predict(feed_glmmPQL_p96_CAR1_Hrs,testSetp96,
                          type = "response", level=0)
# This function outputs the predicted values in Logit space
SE_p96 <- predictSE(feed_glmmPQL_p96_CAR1_Hrs,testSetp96,
                    type = "response", level = 0,se.fit = TRUE)
# Convert the Logit Standard errors into Probabilities
testSetp96$lwr <- logit2prob(SE_p96$fit-SE_p96$se.fit)
testSetp96$upr <- logit2prob(SE_p96$fit+SE_p96$se.fit)
testSetp96$model <- "Plus 96hrs"
summary(feed_glmmPQL_p96_CAR1_Hrs)
```

96 Hour Time-Shift

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: data_H_Tests[data_H_Tests$method == "plus96hrs", ]
##    AIC BIC logLik
##     NA  NA     NA
##
## Random effects:
##  Formula: ~1 | depid
##         (Intercept) Residual
## StdDev:   0.5156173 1.050478
##
## Correlation Structure: Continuous AR(1)
##  Formula: ~timeHrs | depid
##  Parameter estimate(s):
##        Phi
## 0.03309799
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: Foraging ~ ftle48
##                   Value Std.Error   DF    t-value p-value
## (Intercept) -0.06868501 0.1771187 7725 -0.3877909  0.6982
## ftle48       0.15722935 0.1804362 7725  0.8713849  0.3836
##  Correlation:
##        (Intr)
## ftle48 -0.233
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.4612027 -0.9186903 -0.6344998  0.7646907  1.4177460
##
## Number of Observations: 7736
## Number of Groups: 10
```

```
# Use previously processed model to save time
if(!exists("feed_glmmPQL_p192_CAR1_Hrs")){
```

```
  feed_glmmPQL_p192_CAR1_Hrs <-
    glmmPQL(Foraging ~ ftle48 ,
            random = ~ 1 | depid,
            correlation=corCAR1(form=~timeHrs|depid),
            family = binomial,
            data = data_H_Tests[data_H_Tests$method == "plus192hrs",])
  saveRDS(feed_glmmPQL_p192_CAR1_Hrs,
          file = "./Models/feed_glmmPQL_p192_CAR1_Hrs.rds")
}
testSetp192 <- data.frame(ftle48 = seq(-2, 2, length.out = 1000))
# This function outputs the predicted values in probability space
testSetp192$fit <- predict(feed_glmmPQL_p192_CAR1_Hrs,testSetp192,
                           type = "response", level=0)
# This function outputs the predicted values in Logit space
SE_p192 <- predictSE(feed_glmmPQL_p192_CAR1_Hrs,testSetp192,
                     type = "response", level = 0,se.fit = TRUE)
# Convert the Logit Standard errors into Probabilities
testSetp192$lwr <- logit2prob(SE_p192$fit-SE_p192$se.fit)
testSetp192$upr <- logit2prob(SE_p192$fit+SE_p192$se.fit)
testSetp192$model <- "Plus 192hrs"
summary(feed_glmmPQL_p192_CAR1_Hrs)
```

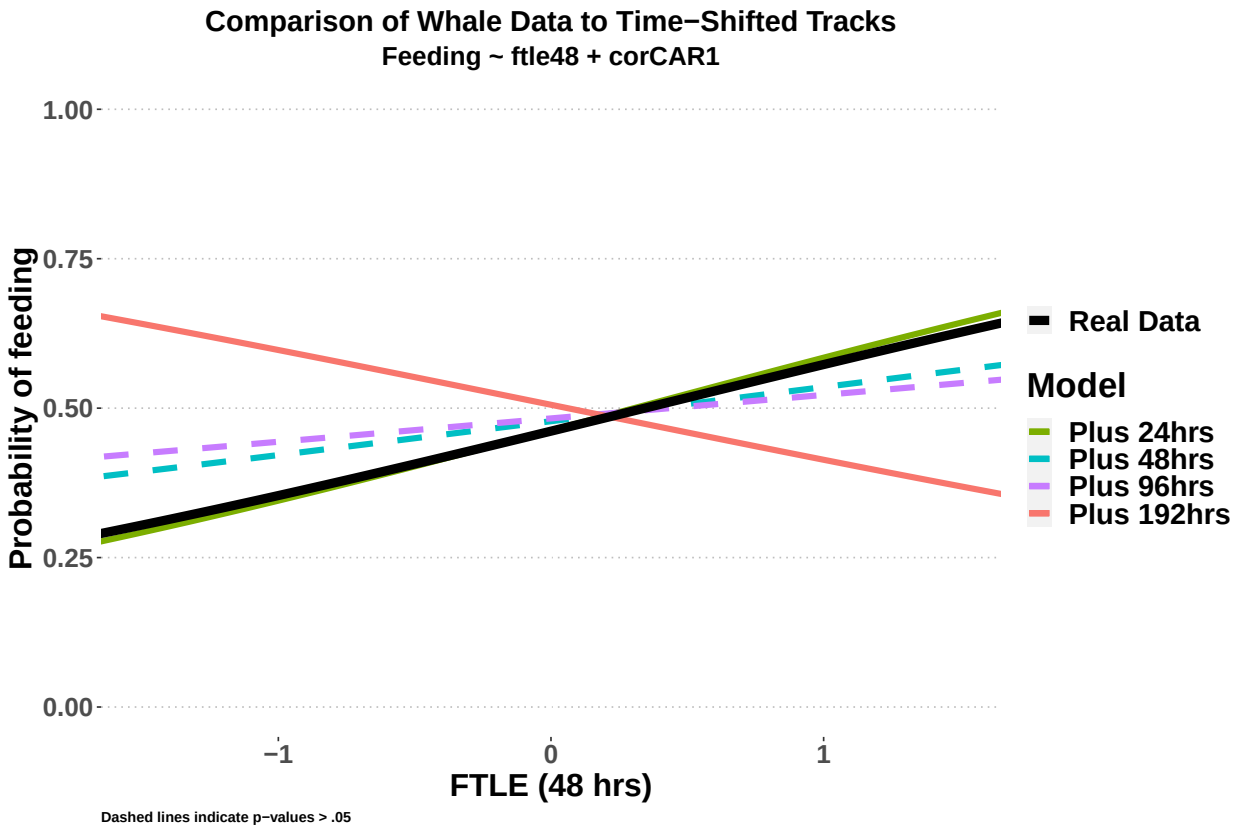
192 Hour Time-Shift

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: data_H_Tests[data_H_Tests$method == "plus192hrs", ]
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | depid
##         (Intercept) Residual
## StdDev:   0.4690788  1.05032
##
## Correlation Structure: Continuous AR(1)
##  Formula: ~timeHrs | depid
##  Parameter estimate(s):
##       Phi
## 0.03352094
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: Foraging ~ ftle48
##                 Value Std.Error   DF    t-value p-value
## (Intercept)  0.0225370 0.1634400 7511  0.1378918  0.8903
## ftle48      -0.3714908 0.1750273 7511 -2.1224729  0.0338
##  Correlation:
##        (Intr)
## ftle48 -0.224
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.5229758 -0.9255259 -0.6938895  0.8230433  1.4335623
##
```

```
## Number of Observations: 7522
## Number of Groups: 10
```

Plot Time-Shift + Blue Whale Models Varying the time-shifts shows a decrease is the relationship between FTLE and the probability of feeding at increasing time-shift values. *(The +24 hour and +192 hour Models were both significant)* Models with timeshifts > 24 hours showed a decreasing slope with increasing time-shift values.

```
ggplot() +
  geom_line(data=testSetp24,
            aes(x=ftle48,y = fit, color=model), size = 2)+
  geom_line(data=testSetp48,
            aes(x=ftle48,y = fit, color=model), size = 2, linetype=2)+
  geom_line(data=testSetp96,
            aes(x=ftle48,y = fit, color=model), size = 2, linetype=2)+
  geom_line(data=testSetp192,
            aes(x=ftle48,y = fit, color=model), size = 2, linetype=1)+
  geom_line(data=testSet,
            aes(x=ftle48,y = fit,alpha = "Real Data"), size = 3)+
  scale_color_discrete(name= "Model",
                       breaks=c("Plus 24hrs","Plus 48hrs",
                                "Plus 96hrs","Plus 192hrs")) +
  scale_alpha_manual(name=NULL,values=c("Real Data"=1)) +
  ggtitle("Comparison of Whale Data to Time-Shifted Tracks",
          subtitle= "Feeding ~ ftle48 + corCAR1")+
  coord_cartesian(ylim = c(0, 1), xlim = c(-1.5,1.5)) +
  labs(x = "FTLE (48 hrs)", y = "Probability of feeding",
       caption = "Dashed lines indicate p-values > .05") +
  theme_pubclean()+
  theme(axis.title = element_text(face="bold", size=22),
        axis.text = element_text( face="bold", size=18),
        plot.title = element_text(face="bold", size=20,hjust = 0.5),
        plot.subtitle = element_text( face="bold", size=18,hjust = 0.5),
        plot.caption = element_text( face="bold", size=10,hjust = 0),
        legend.position="right",
        legend.text = element_text(face="bold", size=20),
        legend.title = element_text(face="bold", size=24),
        strip.text.x = element_text(size = 22, face="bold"))
```



Null Model Results Supplemental Table 1 GLMM Model results for Null Models used in this study.

```
suppT1 <- as.data.frame(
  rbind(c("Blue Whales",
          as.data.frame(round(coef(summary(
            feed_glmmPQL_FTLE_CAR1_Hrs)),3) )$Value[2],
          as.data.frame(round(coef(summary(
            feed_glmmPQL_FTLE_CAR1_Hrs)),3) )$'p-value'[2]),
        c("Random Site Selection Model (CRW Tracks)",
          as.data.frame(round(coef(
            summary(feed_glmmPQL_CRW_CAR1_Hrs)),3) )$Value[2],
          as.data.frame(round(coef(
            summary(feed_glmmPQL_CRW_CAR1_Hrs)),3) )$'p-value'[2]),
        c("Temporal (+24 hours)",
          as.data.frame(round(coef(
            summary(feed_glmmPQL_p24_CAR1_Hrs)),3) )$Value[2],
          as.data.frame(round(coef(
            summary(feed_glmmPQL_p24_CAR1_Hrs)),3) )$'p-value'[2]),
        c("Temporal (+48 hours)",
          as.data.frame(round(coef(
            summary(feed_glmmPQL_p48_CAR1_Hrs)),3) )$Value[2],
          as.data.frame(round(coef(
            summary(feed_glmmPQL_p48_CAR1_Hrs)),3) )$'p-value'[2]),
        c("Temporal (+96 hours)",
          as.data.frame(round(coef(
            summary(feed_glmmPQL_p96_CAR1_Hrs)),3) )$Value[2],
          as.data.frame(round(coef(
```

Model	Slope	p-Value
Blue Whales	0.448	0.002
Random Site Selection Model (CRW Tracks)	-0.066	0.156
Temporal (+24 hours)	0.491	0.002
Temporal (+48 hours)	0.229	0.175
Temporal (+96 hours)	0.157	0.384
Temporal (+192 hours)	-0.371	0.034

```
            summary(feed_glmmPQL_p96_CAR1_Hrs)),3) )$'p-value'[2]),
        c("Temporal (+192 hours)",
          as.data.frame(round(coef(
            summary(feed_glmmPQL_p192_CAR1_Hrs)),3) )$Value[2],
          as.data.frame(round(coef(
            summary(feed_glmmPQL_p192_CAR1_Hrs)),3) )$'p-value'[2])
        )
  )
colnames(suppT1)<- c("Model","Slope","p-Value")
# write_csv(suppT1,file="./rmdFiles/SuppTable1.csv")
kable(suppT1, align = 'lcc', row.names = FALSE) %>%
  kable_styling(bootstrap_options = c("striped", "hover",
                                      "condensed", "responsive"))
```

Hidden Markhov Models

These models take advantage of the inherent serial autocorrelation of animal tag data to determine the probability of switching between discrete states. In our case, the states are known a priori. We are thus calculating the transition probabilities with more certainty than the typical usage of a HMM.

2 State HMM

We start with a 2 state HMM (Feeding vs Not Feeding) to validate the results from the GLMM in the previous analysis.

```
ftleGrid <- seq(-1.25,1.5, length = 1000)
# Design matrix for prediction of stationary probabilities
covs <- cbind("intercept" = 1,
              "ftle" = ftleGrid)
divesHMM2state <- data_Gamm_ALL %>%
  mutate(yday = yday(dttzStart)) %>%
  # Make sure there is environmental data for every position
  dplyr::filter(is.nan(ftle48)==0, is.na(ftle48)==0) %>%
  # only include columns in analysis
  dplyr::select(depid,DiveNum,dttzStart,maxDepth,
                yday,Lat,Long,LungeCount,
                ftle48, dive_Duration, surf_Duration) %>%
  mutate(ID = depid,
         ftle = ftle48,
         lunges = LungeCount,
         lat = Lat,
         lon = Long) %>%
  as.data.frame() %>%
  # 2-State (Feeding Vs Not Feeding)
```

```
  mutate(stateFeeding = case_when(LungeCount == 0 ~ 1,
                                  LungeCount > 0 ~ 2 ))
# Prepare data
dives_prepped2state <- prepData(divesHMM2state, type = "LL",
                                coordNames = c("lon", "lat"))
```

Prepare the data:

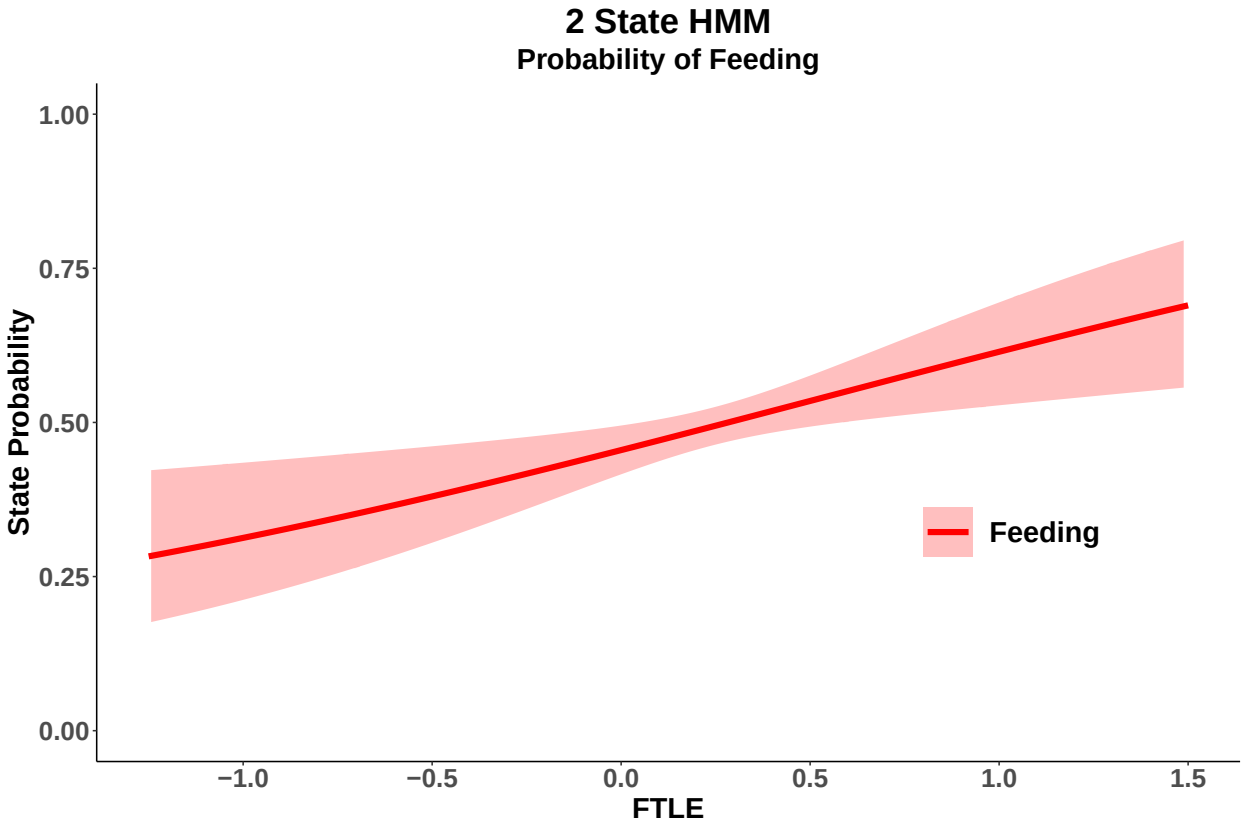
```
# Fit HMM
# Feeding vs Not Feeding
# Initial parameters (known states are based on # lunges)
step_parsFQ <- dives_prepped2state %>%
  group_by(stateFeeding) %>%
  summarize(mu0 = mean(step, na.rm = TRUE),
            sigma0 = sd(step, na.rm = TRUE))

step_par0FQ <- c(step_parsFQ$mu0, step_parsFQ$mu0)
angle_par0FQ <- c(c(0, 0), c(1, 1))
## Feeding ~ FTLE
dive_hmm_feeding_ftleQ <- fitHMM(dives_prepped2state,
                         nbStates = 2,
                         stepPar0 = step_par0FQ,
                         anglePar0 = angle_par0FQ,
                         formula = ~ ftle,
                         knownStates = dives_prepped2state$stateFeeding)
betaF <- dive_hmm_feeding_ftleQ$mle$beta
```

Fit the model:

Stationary state probabilities These probabilities reflect the long-term proportion of time the model spends in each state as a function of FTLE. They can be interpreted as the long-term probabilities of being in each state at different values of FTLE.

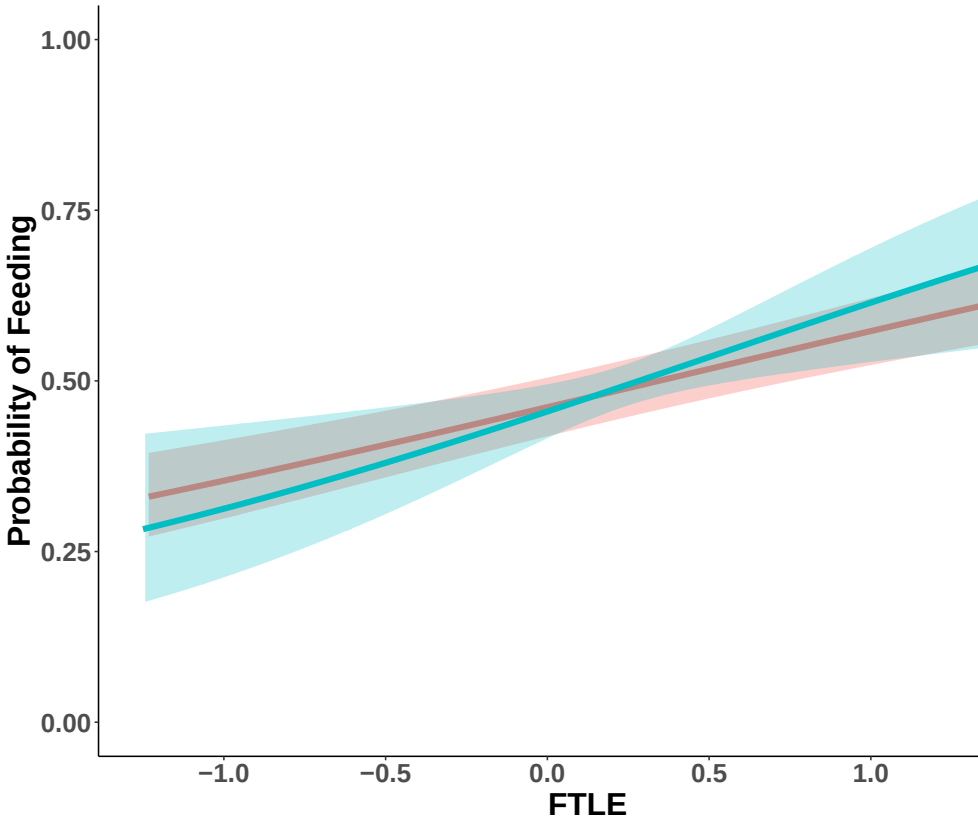
```
# Compute stationary distribution for each row
stat <- stationary(m =dive_hmm_feeding_ftleQ,covs=covs)
stat_DF <- as.data.frame(stat) %>%
  rename("Not Feeding"="V1","Feeding"="V2") %>%
  gather(key="State",value="Probability") %>%
  group_by(State) %>%
  mutate(FTLE = ftleGrid) %>%
  ungroup()

# Calculate the HMM confidence intervals
ci2state<-stationaryCI(dive_hmm_feeding_ftleQ)
ci2stateDFlwr <- data.frame(ftle = ci2state$xValues) %>%
  cbind(ci2state$lowerCI) %>%
  rename("Not Feeding"="1","Feeding"="2") %>%
  gather(key="State",value="ciLwr",-ftle)
ci2stateDFupr <- data.frame(ftle = ci2state$xValues) %>%
  cbind(ci2state$upperCI) %>%
  rename("Not Feeding"="1","Feeding"="2") %>%
  gather(key="State",value="ciUpr",-ftle)
ci2stateDF <- left_join(ci2stateDFlwr,ci2stateDFupr,
```

```
                        by=c("ftle","State"))

pHMM2 <- ggplot()+
  geom_ribbon(data=ci2stateDF%>% filter(State == "Feeding"),
              aes(x=ftle,ymin = ciLwr, ymax = ciUpr,
                  fill=State,group=State), alpha = .25) +
  geom_line(data=stat_DF %>% filter(State == "Feeding"),
            aes(x=FTLE,y=Probability,group=State,color=State),
            size=2)+
  scale_color_manual(name=NULL,values = c("Feeding"="red")) +
  scale_fill_manual(name=NULL,values = c("Feeding"="red")) +
  ylab("State Probability") + xlab("FTLE")+
  ylim(0,1)+
  ggtitle("2 State HMM",subtitle = "Probability of Feeding")+
  scale_x_continuous(breaks=seq(-2,2,.5),limits=c(-1.25,1.5))+
  theme_classic()+
  theme(axis.title = element_text(face="bold", size=20),
        axis.text = element_text( face="bold", size=18),
        plot.title = element_text(face="bold", size=24,hjust = 0.5),
        plot.subtitle = element_text( face="bold", size=20,hjust = 0.5),
        plot.caption = element_text( face="bold", size=16,hjust = 0),
        legend.text = element_text(face="bold", size=20),
        legend.title = element_text(face="bold", size=22,hjust = 0.5),
        legend.margin = margin(0.15,0.15,0.15,0.15, unit="cm"),
        legend.key.size = unit(1.25,"cm"),
        legend.position = c(.8,.35))
pHMM2
```




```
ggplot() +
  geom_ribbon(data=testSet,
              aes(x=ftle48,ymin = lwr, ymax = upr,
                  color = NULL,fill="GLMM"), alpha = .35) +
  geom_line(data=testSet,
            aes(x=ftle48, y = fit, color = "GLMM"), size = 2) +
  geom_ribbon(data=ci2stateDF%>% filter(State == "Feeding"),
              aes(x=ftle,ymin = ciLwr, ymax = ciUpr, color = NULL,fill="HMM"),
              alpha = .25) +
  geom_line(data=stat_DF %>% filter(State == "Feeding"),
            aes(x=FTLE, y=Probability, group=State,color="HMM"),size=2)+
  scale_color_discrete(name= "Model")+
  scale_fill_discrete(name= "Model")+
  scale_x_continuous(breaks=seq(-2,2,.5),limits=c(-1.25,1.5))+
  coord_cartesian(ylim = c(0, 1)) +
  labs(x = "FTLE", y = "Probability of Feeding") +
  theme_classic()+
  theme(axis.title = element_text(face="bold", size=22),
        axis.text = element_text( face="bold", size=18),
        plot.title = element_text(face="bold", size=20,hjust = 0.5),
        plot.subtitle = element_text( face="bold", size=18,hjust = 0.5),
        plot.caption = element_text( face="bold", size=10,hjust = 0),
        legend.position="right",
        legend.text = element_text(face="bold", size=20),
        legend.title = element_text(face="bold", size=24, hjust = 0.5),
```

```
        strip.text.x = element_text(size = 22, face="bold"))
```



Comparison of GLMM and HMM

4-State Feeding Rate HMM

Given the positive relationship observed in feeding behavior in relation to FTLE, we wanted to explore whether FTLE influenced Feeding Performance. Here we define Feeding rate as:

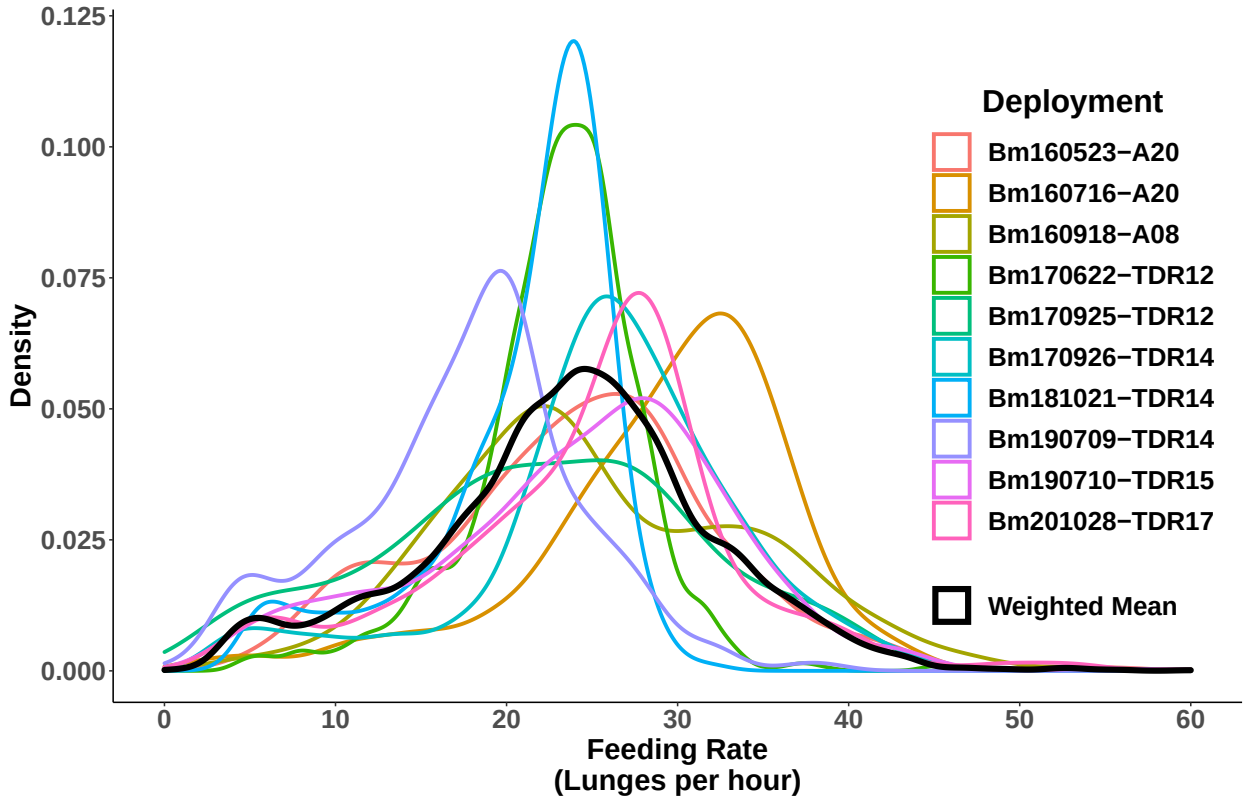
$$FeedingRate = \frac{\#Lunges_{perdive}}{DiveDuration + PostDiveSurfaceDuration}$$

and it is represented in Lunges per hour.

Individual Feeding Rate Distributions To account for differences in feeding rates between individuals, we normalize the feeding rate by percentile rank for each individual. For this analysis, we split the normalized distribution of feeding rates into groups that will make up the states in a HMM. The states are split at the 25th and 75th percentiles, separating the tails (Light and Heavy) from the bulk of the feeding (i.e. middle 50% of feeding rates).

```
p1 <- ggplot() +
  geom_density(data=bwFeedingOverall,
               aes(x=feedingRateHR,group=depid,color=depid),size=1.25) +
  geom_density(data=bwFeedingOverall,
               aes(x=feedingRateHR,weight=weight, alpha="Weighted Mean"),
               size=2)+
  scale_color_discrete(name = "Deployment")+
  scale_alpha_manual(name=NULL,values=c(1))+
  scale_x_continuous(breaks=seq(0,60,10),limits = c(0,60))+
  xlab("Feeding Rate\n(Lunges per hour)") +
```

```
  ylab("Density") +
  labs(caption = "Mean Line (black) is Weighted by Individual")+
  guides(color = guide_legend(order=1))+
  theme_classic()+
  theme(axis.title = element_text(face="bold", size=20),
        axis.text = element_text( face="bold", size=18),
        plot.title = element_text(face="bold", size=24,hjust = 0.5),
        plot.subtitle = element_text( face="bold", size=20,hjust = 0.5),
        plot.caption = element_text( face="bold", size=16,hjust = 0),
        legend.text = element_text(face="bold", size=18),
        legend.title = element_text(face="bold", size=22,hjust = 0.5),
        legend.margin = margin(0.15,0.15,0.15,0.15, unit="cm"),
        legend.background = element_rect(color=NA),
        legend.key = element_rect(fill=NA,color=NA),
        legend.key.size = unit(1,"cm"),
        legend.position = c(0.85,0.5))
p1
```



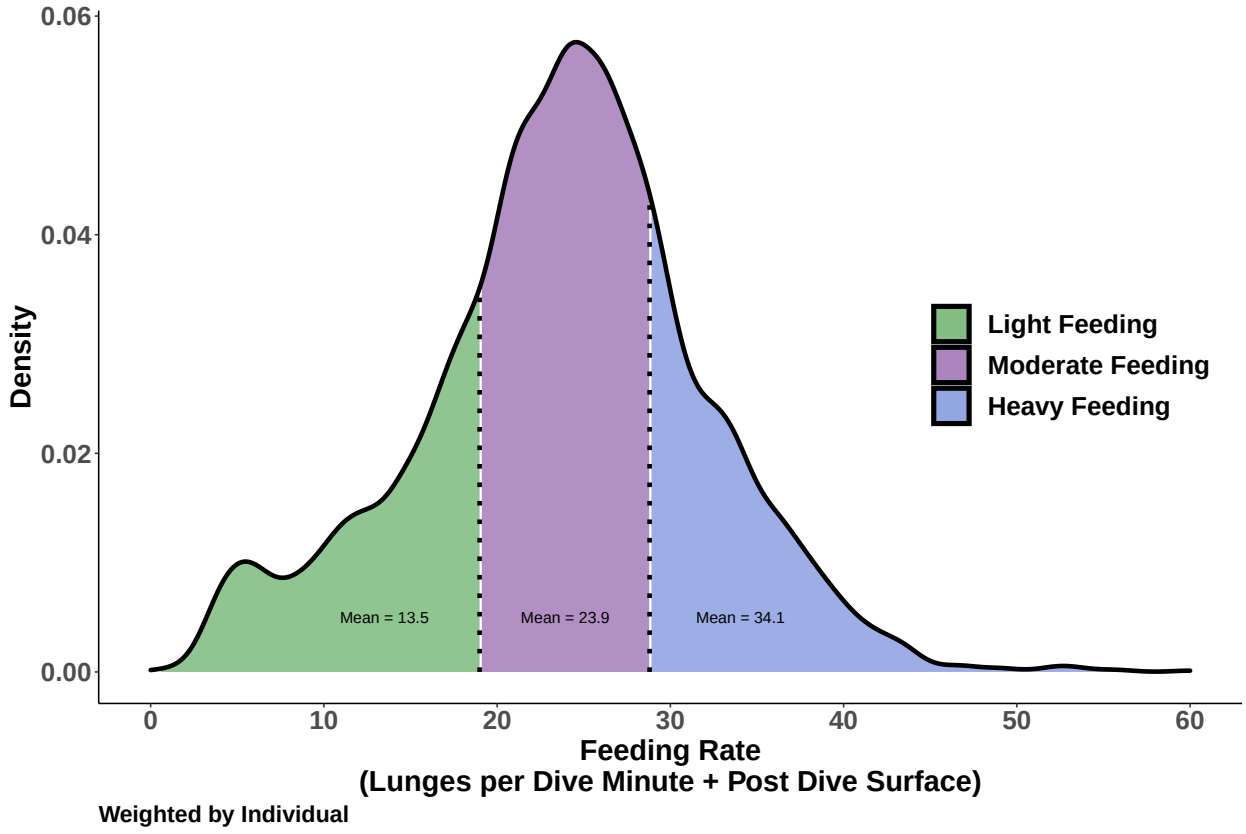
Mean Line (black) is Weighted by Individual

Mean Feeding Rate Distribution Here we show the states identified for the mean distribution, which will be used later in the analysis.

```
percentileFRlwr <-0.25
percentileFRupr <- 0.75

qLwrW <- weighted_quantile(x=bwFeedingOverall$feedingRateHR,
                           w = bwFeedingOverall$weight,
                           probs = percentileFRlwr,na.rm = TRUE)
```

```
qUprW <- weighted_quantile(x=bwFeedingOverall$feedingRateHR,
                           w = bwFeedingOverall$weight,
                           probs = percentileFRupr,na.rm = TRUE)

lightMean <- bwFeedingOverall %>%
  dplyr::filter(feedingRateHR < qLwrW) %>%
  summarize(mean(feedingRateHR)) %>% unlist()
moderateMean <- bwFeedingOverall %>%
  dplyr::filter(feedingRateHR >= qLwrW,
                feedingRateHR < qUprW) %>%
  summarize(mean(feedingRateHR)) %>% unlist()
heavyMean <- bwFeedingOverall %>%
  dplyr::filter(feedingRateHR >= qUprW) %>%
  summarize(mean(feedingRateHR)) %>% unlist()

statePal <- c("Not Feeding"="firebrick3",
              "Light Feeding"="forestgreen",
              "Moderate Feeding"="darkorchid4",
              "Heavy Feeding"="royalblue3")
p <- ggplot() +
  geom_density(data=bwFeedingOverall,
               aes(x=feedingRateHR,weight=weight))
# subset region and plot
d <- ggplot_build(p)$data[[1]]
pStateDensity <- p +
  geom_area(data = subset(d, x < qLwrW),
            aes(x=x, y=y, fill="Light Feeding", alpha = "Light Feeding")) +
  geom_area(data = subset(d, x >= qLwrW & x <= qUprW),
            aes(x=x, y=y, fill="Moderate Feeding",alpha="Moderate Feeding")) +
  geom_area(data = subset(d, x > qUprW),
            aes(x=x, y=y, fill="Heavy Feeding", alpha="Heavy Feeding")) +
  geom_segment(aes(x=qLwrW, xend=qLwrW,y=0,
                   yend=approx(x = d$x, y = d$y, xout = qLwrW)$y),
               color="black", size=1.5,linetype=3)+
  geom_segment(aes(x=qUprW, xend=qUprW,y=0,
                   yend=approx(x = d$x, y = d$y, xout = qUprW)$y),
               color="black", size=1.5,linetype=3)+
  annotate("text", x = lightMean, y = .005,
           label = paste0("Mean = ", round(lightMean,1))) +
  annotate("text", x = moderateMean, y = .005,
           label = paste0("Mean = ", round(moderateMean,1))) +
  annotate("text", x = heavyMean, y = .005,
           label = paste0("Mean = ", round(heavyMean,1))) +
  scale_alpha_manual(name=NULL,values=c(.5,.5,.5),
                     breaks=c("Light Feeding","Moderate Feeding","Heavy Feeding")) +
  scale_fill_manual(name=NULL,values=statePal,
                    breaks=c("Light Feeding","Moderate Feeding","Heavy Feeding")) +
  geom_density(data=bwFeedingOverall,
               aes(x=feedingRateHR,weight=weight),size = 1.5)+
  scale_x_continuous(breaks=seq(0,60,10),limits = c(0,60))+
  xlab("Feeding Rate\n(Lunges per Dive Minute + Post Dive Surface)") +
  ylab("Density") +
  labs(caption = "Weighted by Individual")+
```

```
  guides(fill = guide_legend(override.aes = list(alpha=.15)))+
  theme_classic()+
  theme(axis.title = element_text(face="bold", size=20),
        axis.text = element_text( face="bold", size=18),
        plot.title = element_text(face="bold", size=24,hjust = 0.5),
        plot.subtitle = element_text( face="bold", size=20,hjust = 0.5),
        plot.caption = element_text( face="bold", size=16,hjust = 0),
        legend.text = element_text(face="bold", size=18),
        legend.title = element_text(face="bold", size=22,hjust = 0.5),
        legend.margin = margin(0.15,0.15,0.15,0.15, unit="cm"),
        legend.background = element_rect(color=NA),
        legend.key = element_rect(fill=NA,color=NA),
        legend.key.size = unit(1,"cm"),
        legend.position = c(0.85,0.5))

pStateDensity
```




```
ftleGrid <- seq(-1.25,1.5, length = 1000)
# Design matrix for prediction of stationary probabilities
covs <- cbind("intercept" = 1,
              "ftle" = ftleGrid)
divesHMM4state <- data_Gamm_ALL %>%
  mutate(yday = yday(dttzStart)) %>%
  # Make sure there is environmental data for every position
  dplyr::filter(is.nan(ftle48)==0, is.na(ftle48)==0) %>%
```

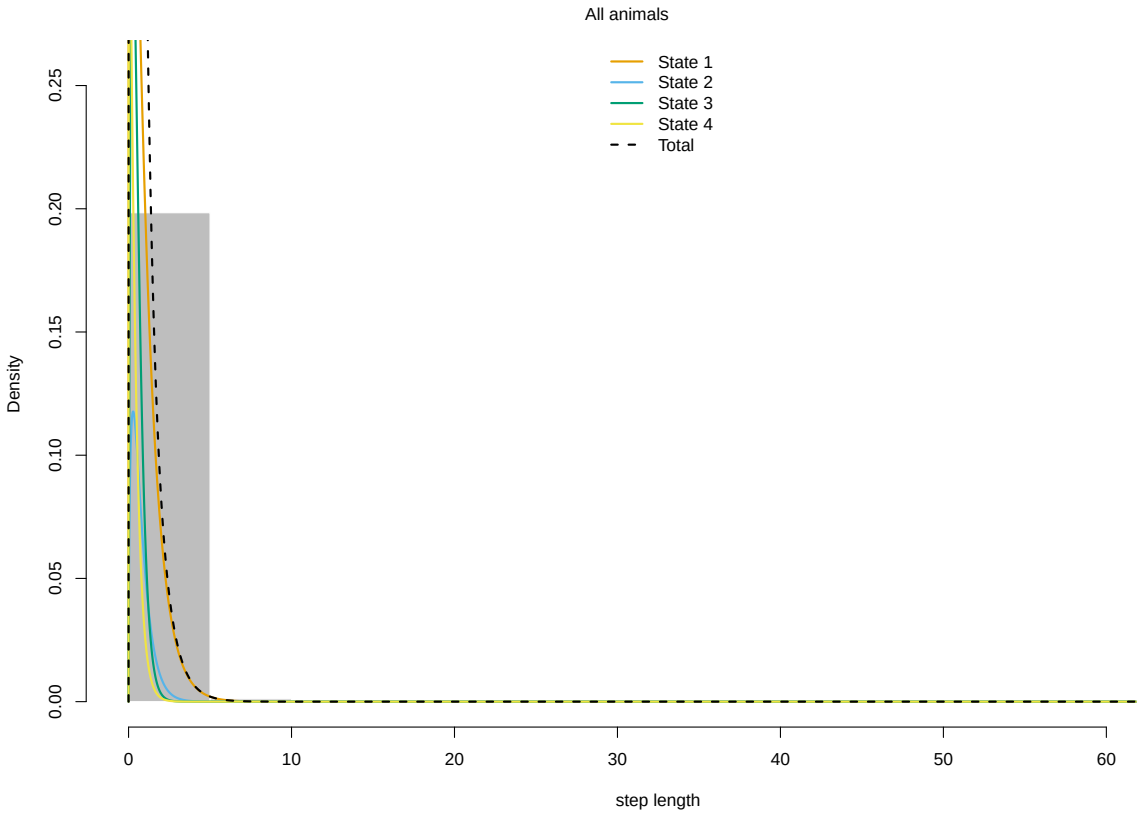
```
  # only include columns in analysis
  dplyr::select(depid,DiveNum,dttzStart,maxDepth,yday,
                Lat,Long,LungeCount,
                feedingRateHR,ftle48,
                dive_Duration, surf_Duration) %>%
  group_by(depid) %>%
  # Rank feeding rates by individual
    mutate(frPercentile =
             ecdf(
               bwFeedingOverall$feedingRateHR[
                 bwFeedingOverall$depid == .$depid])(feedingRateHR)) %>%
  ungroup() %>%
  mutate(ID = depid,
         ftle = ftle48,
         lunges = LungeCount,
         lat = Lat,
         lon = Long,
# 4-State Feeding Rate (Not Feeding, Light Feeding, Feeding, Heavy Feeding)
         stateFeedingRate = case_when(
           frPercentile == 0 ~ 1,
           (frPercentile > 0 & frPercentile <= percentileFRlwr) ~ 2,
           (frPercentile > percentileFRlwr & frPercentile <= percentileFRupr) ~ 3,
           frPercentile > percentileFRupr ~ 4) ) %>%
  as.data.frame()
# Prepare data
dives_prepped4state <- prepData(divesHMM4state, type = "LL",
                                coordNames = c("lon", "lat"))
```

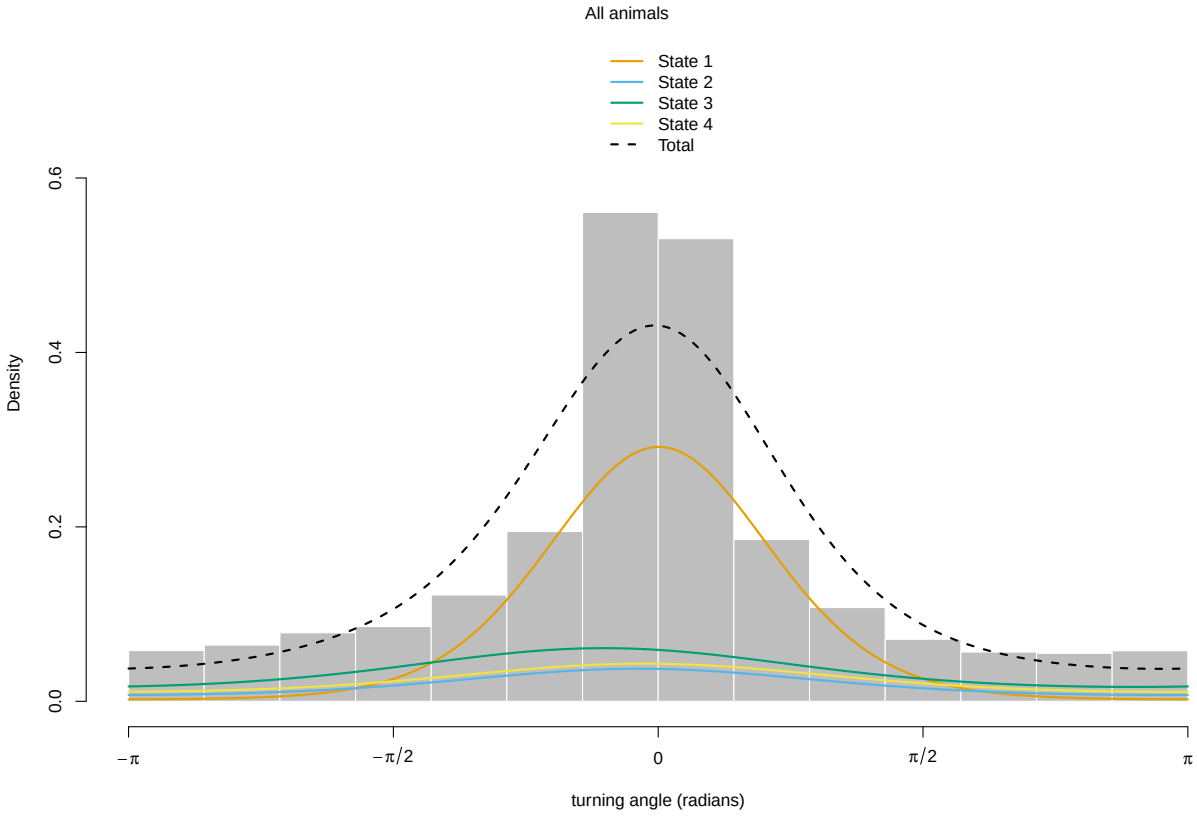
Prepare the data:

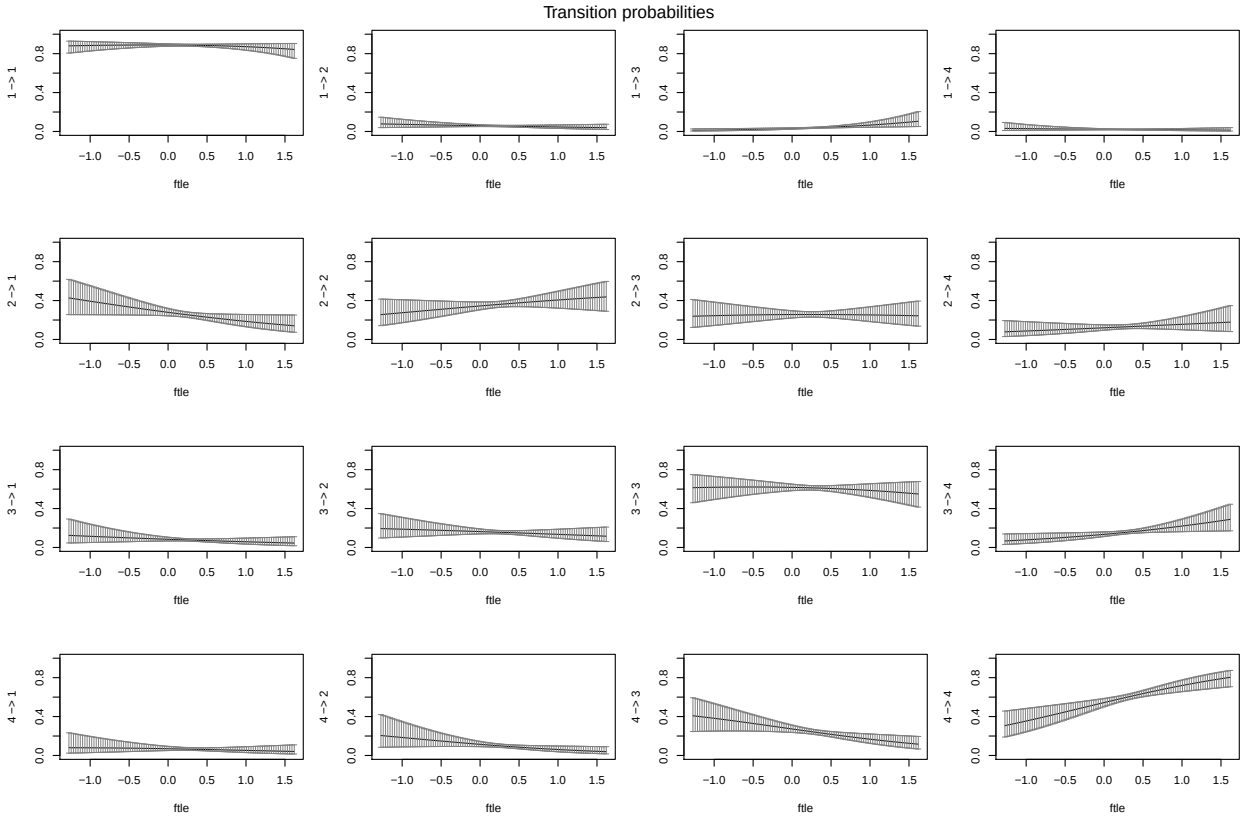
```
# Initial parameters (known states are based on # lunges)
step_parsFRQ2 <- dives_prepped4state %>%
  group_by(stateFeedingRate) %>%
  summarize(mu0 = mean(step, na.rm = TRUE),
            sigma0 = sd(step, na.rm = TRUE))

step_par0FRQ2<- c(step_parsFRQ2$mu0, step_parsFRQ2$mu0)
angle_par0FRQ2 <- c(c(0, 0, pi, pi), c(1, 1, 1, 1))
## Feeding Rate ~ FTLE
dive_hmm_FeedingRate2_ftleQ <-
  fitHMM(dives_prepped4state,
         nbStates = 4,
         stepPar0 = step_par0FRQ2,
         anglePar0 = angle_par0FRQ2,
         formula = ~ ftle,
         knownStates = dives_prepped4state$stateFeedingRate)

#Foraging Rate~ FTLE
plot(dive_hmm_FeedingRate2_ftleQ, ask = FALSE,
     plotTracks = FALSE,plotCI =TRUE)
```

Fit the model:

```
## Decoding states sequence... DONE
```



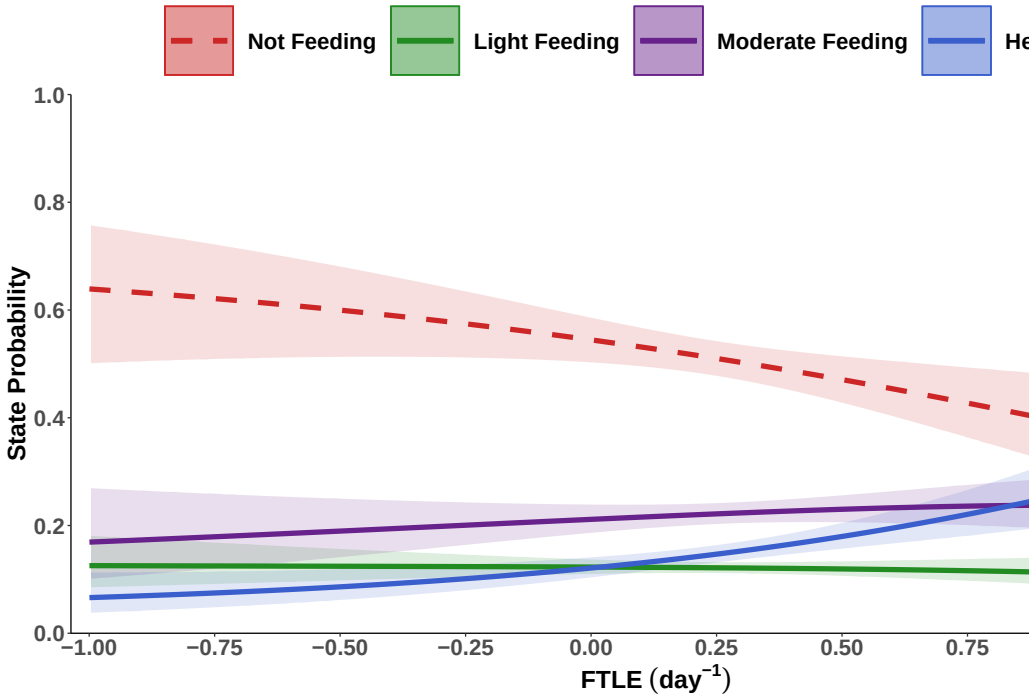





```
ci<-stationaryCI(dive_hmm_FeedingRate2_ftleQ)
ciDFlwr <- data.frame(ftle = ci$xValues) %>%
  cbind(ci$lowerCI) %>%
  rename("Not Feeding"="1","Light Feeding"="2",
         "Moderate Feeding"="3","Heavy Feeding"="4") %>%
  gather(key="State",value="ciLwr",-ftle)
ciDFupr <- data.frame(ftle = ci$xValues) %>%
  cbind(ci$upperCI) %>%
  rename("Not Feeding"="1","Light Feeding"="2",
         "Moderate Feeding"="3","Heavy Feeding"="4") %>%
  gather(key="State",value="ciUpr",-ftle)
ciDF <- left_join(ciDFlwr,ciDFupr,by=c("ftle","State"))

# Compute stationary distribution for each row
stat3 <- stationary(m =dive_hmm_FeedingRate2_ftleQ,covs=covs)
stat3_DF <- as.data.frame(stat3) %>%
  rename("Not Feeding"="V1","Light Feeding"="V2",
         "Moderate Feeding"="V3","Heavy Feeding"="V4") %>%
  gather(key="State",value="Probability") %>%
  group_by(State) %>%
  mutate(FTLE = ftleGrid) %>%
  ungroup()

pHMM4 <- ggplot()+
  geom_ribbon(data=ciDF %>% filter(State=="Not Feeding"),
```

```
            aes(x=ftle,ymin=ciLwr,ymax=ciUpr,color=NULL,
                group=State,fill=State), alpha = .15) +
geom_ribbon(data=ciDF %>% filter(State=="Light Feeding"),
            aes(x=ftle,ymin=ciLwr,ymax=ciUpr,color=NULL,
                group=State,fill=State), alpha = .15) +
geom_ribbon(data=ciDF %>% filter(State=="Moderate Feeding"),
            aes(x=ftle,ymin=ciLwr,ymax=ciUpr,color=NULL,
                group=State,fill=State), alpha = .15) +
geom_ribbon(data=ciDF %>% filter(State=="Heavy Feeding"),
            aes(x=ftle,ymin=ciLwr,ymax=ciUpr,color=NULL,
                group=State,fill=State), alpha = .15) +
geom_line(data=stat3_DF %>%
            filter(State=="Not Feeding"),
          aes(x=FTLE,y=Probability,group=State,
              color=State,linetype=State),size=2)+
geom_line(data=stat3_DF %>% filter(State=="Light Feeding"),
          aes(x=FTLE,y=Probability,group=State,color=State,
              linetype=State),size=2)+
geom_line(data=stat3_DF %>%
            filter(State=="Moderate Feeding"),
          aes(x=FTLE,y=Probability,group=State,color=State,
              linetype=State),size=2)+
geom_line(data=stat3_DF %>%
            filter(State=="Heavy Feeding"),
          aes(x=FTLE,y=Probability,group=State,
              color=State,linetype=State),size=2)+
scale_color_manual(name=NULL,
                   values = c("Heavy Feeding"="royalblue3",
                              "Light Feeding"="forestgreen",
                              "Moderate Feeding"="darkorchid4",
                              "Not Feeding"="firebrick3"),
                   breaks = c("Not Feeding",
                              "Light Feeding",
                              "Moderate Feeding",
                              "Heavy Feeding")) +
scale_fill_manual(name=NULL,
                   values = c("Heavy Feeding"="royalblue3",
                              "Light Feeding"="forestgreen",
                              "Moderate Feeding"="darkorchid4",
                              "Not Feeding"="firebrick3"),
                   breaks = c("Not Feeding",
                              "Light Feeding",
                              "Moderate Feeding",
                              "Heavy Feeding")) +
ylab("State Probability") +
xlab(expression( bold(FTLE~(day^bold({bold("-1")})))) )+
scale_x_continuous(breaks=seq(-1.25,1.5,.25),limits=c(-1,1.26),
                   expand = c(0.005, 0.025))+
  scale_y_continuous(breaks=seq(0,1,.2),limits=c(0,1),
                     expand = c(0, 0))+
scale_linetype_manual(name=NULL,
                      values = c("Not Feeding"="dashed",
                                 "Light Feeding"="solid",
```

```
                                  "Moderate Feeding"="solid",
                                  "Heavy Feeding"="solid"),
                       breaks = c("Not Feeding",
                                  "Light Feeding",
                                  "Moderate Feeding",
                                  "Heavy Feeding")) +

  theme_classic()+
  theme(axis.title = element_text(face="bold", size=20),
        axis.text = element_text( face="bold", size=18),
        axis.text.x = element_text(face="bold", size = 18),
        plot.title = element_text(face="bold", size=24,hjust = 0.5),
        plot.subtitle = element_text( face="bold", size=20,hjust = 0.5),
        plot.caption = element_text( face="bold", size=16,hjust = 0),
        legend.text = element_text(face="bold", size=18),
        legend.title = element_text(face="bold", size=22,hjust = 0.5),
        legend.margin = margin(0.15,0.15,0.15,0.15, unit="cm"),
        legend.background = element_rect(color=NA),
        legend.key = element_rect(fill=NA,color=NA),
        legend.key.size = unit(2,"cm"),
        legend.position = "top")
pHMM4
```



Stationary State Probabilities

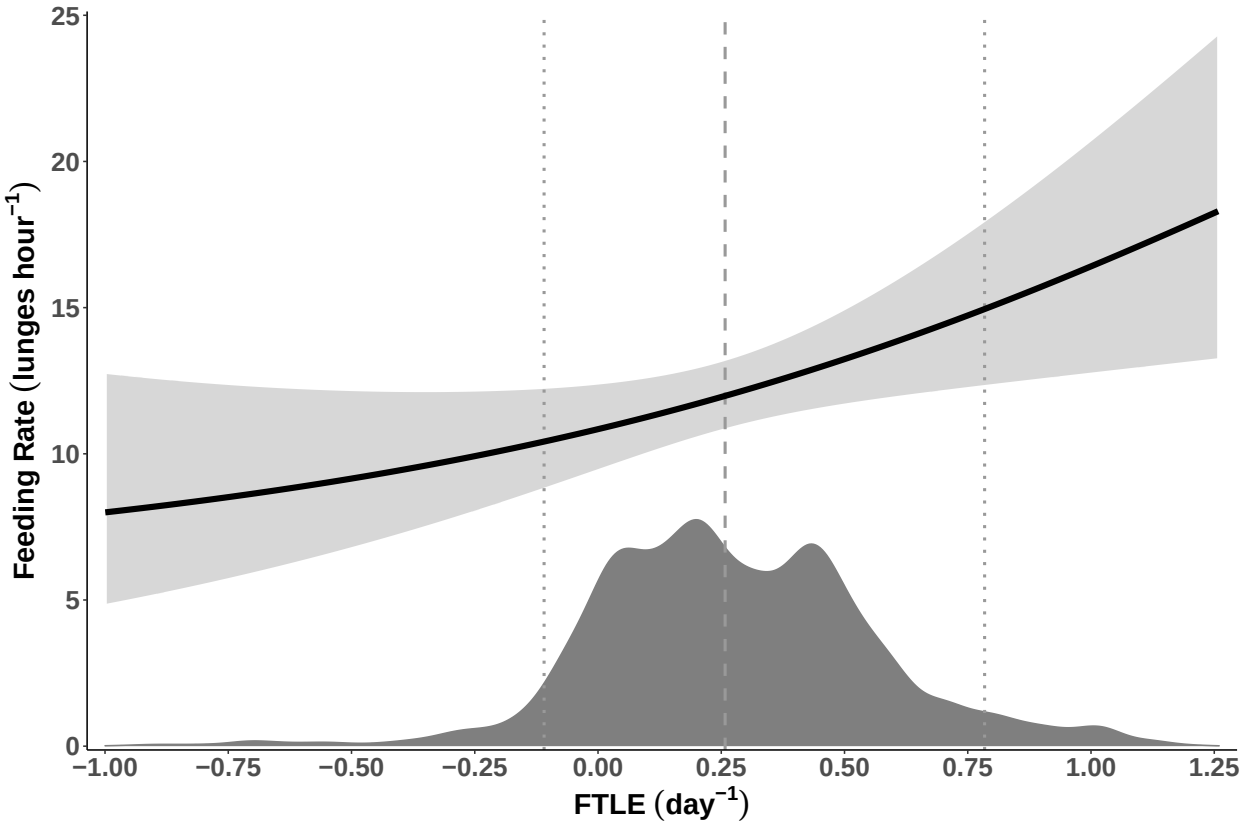
Estimated Feeding Rates Here we use the weighted distribution of feeding rates to estimate the overall feeding rate of an average blue whale in relation to FTLE. For each of the 4 states from the model above, we calculated the mean feeding rate for each state. Using the stationary state probabilites of the HMM, calculated the overall feeding rate across the range of FTLE values i = -1 to 1.25:

$$FeedingRate_{(FTLE_i)} = \sum_{state=1}^{4} StationaryProbability_{state_{FTLE_i}} * FeedingRate_{state}$$

We made the same calculations using the upper and lower Confidence Intervals of the model. In the plot below, we show these estimated feeding rates for an 'average whale' in our dataset, as well as the weighted density distribution of FTLE that the animals in our study encountered. Vertical lines are the 5th,50th and 95th perceniles of that distrubution.

```
feeding_rates <-
  data.frame(stateFeedingRate = c(1,2,3,4),
             feedingRateHr = c(0.0,lightMean,moderateMean,heavyMean),
             State = c("Not Feeding", "Light Feeding",
                       "Moderate Feeding", "Heavy Feeding"))

# join the estimated state feeding rate to the predicted probabilites
feeding_rate_by_ftle <- stat3_DF %>%
  left_join(feeding_rates, by = "State") %>%
  group_by(FTLE) %>%
  summarize(weightedFeedingHourly = sum(Probability * feedingRateHr),
            .groups = "drop")
feeding_rate_by_ftleCI <- ciDF %>%
  left_join(feeding_rates, by = "State") %>%
  group_by(ftle) %>%
  summarize(weightedFeedingHourlyLWR = sum(ciLwr * feedingRateHr),
            weightedFeedingHourlyUPR = sum(ciUpr * feedingRateHr),
            .groups = "drop")

(pFeedingRateHr <-
    ggplot() +
    geom_density(data=bwAll,
                 aes(x=ftle48,weight=weight,y= ..density.. * 5),
                 color = "grey50", fill = "grey50") +
    geom_ribbon(data=feeding_rate_by_ftleCI,
                aes(x=ftle,
                    ymin = weightedFeedingHourlyLWR,
                    ymax = weightedFeedingHourlyUPR),
                fill = 'lightgrey', alpha = .90) +
    geom_vline(xintercept = weighted_quantile(x=bwAll$ftle48,
                                              w = bwAll$weight,
                                              probs = c(0.05, 0.5, 0.95),
                                              na.rm = TRUE),
               color = "grey60",
               linetype = c("dotted", "dashed", "dotted"),
               size = 1) +
    geom_line(data=feeding_rate_by_ftle,
              aes(x=FTLE, y=weightedFeedingHourly), size = 2) +
    scale_x_continuous(breaks=seq(-1.25,1.5,.25),
                       limits=c(-1,1.26),
                       expand = c(0.005, 0.025))+
    scale_y_continuous(limits=c(0,25),
                       expand = c(0.005, 0.005)) +
    expand_limits(y = c(0, 20)) +
    ylab(expression( bold(Feeding~Rate~(lunges~hour^{bold("-1")})))) +
    xlab(expression( bold(FTLE~(day^bold({bold("-1")}))))  )+
    theme_classic() +
    theme(axis.title = element_text(face="bold", size=20),
          axis.text = element_text(face="bold", size=18),
```

```
          plot.title = element_text(face="bold", size=24,hjust = 0.5),
          plot.subtitle = element_text( face="bold", size=20,hjust = 0.5),
          plot.caption = element_text( face="bold", size=16,hjust = 0),
          legend.text = element_text(face="bold", size=20),
          legend.title = element_text(face="bold", size=22,hjust = 0.5),
          legend.margin = margin(0.15,0.15,0.15,0.15, unit="cm"),
          legend.background = element_rect(color="black"),
          legend.key = element_rect(fill=NA),
          legend.key.size = unit(1.25,"cm")))
```



Feeding Rate Results Table Key Metrics for Table

```
# Encountered FTLE
estimatedFTLE <- as.data.frame(t(round(weighted_quantile(x=bwAll$ftle48,
                  w = bwAll$weight,
                  probs = c(0.05, 0.5, 0.95),
                  na.rm = TRUE),3)) ) %>%
  mutate(Estimate = "Encountered FTLE") %>%
  dplyr::select(Estimate,everything())

# Probabilities
probs <- stat3_DF %>%
  group_by(State) %>%
  summarize(
    Fifth=round(first(Probability[FTLE >= weighted_quantile(
      x=bwAll$ftle48, w = bwAll$weight,probs = c(0.05),
      na.rm = TRUE)]),3),
```

```
    Fiftieth=round(first(Probability[FTLE >= weighted_quantile(
      x=bwAll$ftle48,w = bwAll$weight,probs = c(0.5),
      na.rm = TRUE)]),3),
    NinetyFifth=round(first(Probability[FTLE >= weighted_quantile(
      x=bwAll$ftle48,w = bwAll$weight,probs = c(0.95),
      na.rm = TRUE)]),3)) %>%
  mutate(State =factor(State,
                       levels=c("Not Feeding",
                                "Light Feeding",
                                "Moderate Feeding",
                                "Heavy Feeding"))) %>%
  arrange(State)
colnames(probs) <- colnames(estimatedFTLE)

estimatedFR <- feeding_rate_by_ftle %>%
    summarize(
      Fifth=round(first(weightedFeedingHourly[FTLE >= weighted_quantile(
        x=bwAll$ftle48,w = bwAll$weight,probs = c(0.05),
        na.rm = TRUE)]),2),
      Fiftieth=round(first(weightedFeedingHourly[FTLE >= weighted_quantile(
        x=bwAll$ftle48,w = bwAll$weight,probs = c(0.5),
        na.rm = TRUE)]),2),
      NinetyFifth=round(first(
        weightedFeedingHourly[FTLE >= weighted_quantile(
        x=bwAll$ftle48,w = bwAll$weight,probs = c(0.95),
        na.rm = TRUE)]),2))%>%
  mutate(Estimate = "Estimated Feeding Rate") %>%
  dplyr::select(Estimate,everything())
colnames(estimatedFR) <- colnames(estimatedFTLE)

table2 <- rbind(estimatedFTLE,probs,estimatedFR)
table2
```

Estimate	5%	50%	95%
Encountered FTLE	-0.109	0.258	0.784
Not Feeding	0.558	0.509	0.421
Light Feeding	0.123	0.121	0.116
Moderate Feeding	0.207	0.222	0.236
Heavy Feeding	0.112	0.148	0.227
Estimated Feeding Rate	10.430	11.980	14.950

Percent increases in estimated feeding rates reported in the manuscript:

```
# For 5th to 95th
overall_5th_to_95th <- (estimatedFR$`95%`-estimatedFR$`5%`)/estimatedFR$`5%`
# For 5th to 50th
lower_5th_to_50th <- (estimatedFR$`50%`-estimatedFR$`5%`)/estimatedFR$`5%`
# For 50th to 95th
upper_50th_to_95th <- (estimatedFR$`95%`-estimatedFR$`50%`)/estimatedFR$`50%`

cat("For 5th to 95th: ",round(overall_5th_to_95th,2),
    "\nFor 5th to 50th: ",round(lower_5th_to_50th,2),
    "\nFor 50th to 95th: ",round(upper_50th_to_95th,2))
```

```
## For 5th to 95th:  0.43
```

```
## For 5th to 50th:  0.15
## For 50th to 95th:  0.25
```