

```

# R script for analysing fine-scale data of white shark behaviour

# Load required libraries
library(dplyr)
library(trajr)
library(geosphere)
library(rstanarm)
library(shinystan)
library(PerformanceAnalytics)
library(mice)
library(sp)
library(lubridate)

# working directory
setwd("~/Desktop/R analysis/")

# read in 1st file and check for NA values
GPSdat <- read.csv("GPS_WSHK.csv")
md.pattern(GPSdat)

# change the date.time to numeric
GPSdat$time <- strptime(GPSdat$date.time, "%Y-%m-%d %H:%M:%S")
GPSdat$time <- as.numeric(GPSdat$time)

# create spatial points dataframe and change projection to UTM
pts <- SpatialPoints(GPSdat[, c("long", "lat")],
  proj4string = CRS("+proj=longlat +datum=WGS84"))
pts <- sp::spTransform(pts, CRS("+proj=utm +zone=56 +south +datum=WGS84"))

# Append UTM coordinates to original data
GPSdat$x <- coordinates(pts)[, 1]
GPSdat$y <- coordinates(pts)[, 2]

# Create ammended dataframe with UTM coordinates
coords <- GPSdat[, c("shark.ID", "shark.track", "time", "x", "y")]

# Create separte csv files for the different track segments for the different
# sharks (shark.track)
track_list <- split(coords, f = coords$shark.track)
sapply(names(track_list),
  function (x) write.csv(track_list[[x]],
    file = paste(x, "csv", sep = "."),
    row.names = FALSE))

# Building trajectory analysis framework
data.frame(levels(coords$shark.track))

# for building trajectory analysis framework, track segments less than 1 minute
# long were excluded.
track_files <- as.data.frame(rbind(
  # c("WSHK001.1a.csv", "WSHK001"), # 1 line
  # c("WSHK001.1b.csv", "WSHK001"), # 1 line
  # c("WSHK001.1c.csv", "WSHK001"), # 1 line

```

```

c("WSHK002.1a.csv", "WSHK002"),
# c("WSHK002.1b.csv", "WSHK002"),# 27 lines
c("WSHK003.1a.csv", "WSHK003"),
c("WSHK003.2a.csv", "WSHK003"),
c("WSHK004.1a.csv", "WSHK004"),
c("WSHK005.1a.csv", "WSHK005"),
c("WSHK006.1a.csv", "WSHK006"),
c("WSHK007.1a.csv", "WSHK007"),
c("WSHK008.1a.csv", "WSHK008"),
c("WSHK009.1a.csv", "WSHK009"),
c("WSHK010.1a.csv", "WSHK010"),
c("WSHK010.1b.csv", "WSHK010"),
c("WSHK010.1c.csv", "WSHK010"),
# c("WSHK010.1d.csv", "WSHK010"),# 28 lines
c("WSHK011.1a.csv", "WSHK011"),
c("WSHK012.1a.csv", "WSHK012"),
c("WSHK013.1a.csv", "WSHK013"),
c("WSHK013.2a.csv", "WSHK013"),
c("WSHK014.1a.csv", "WSHK014"),
c("WSHK014.2a.csv", "WSHK014"),
c("WSHK015.1a.csv", "WSHK015"),
c("WSHK016.1a.csv", "WSHK016"),
c("WSHK017.1a.csv", "WSHK017"),
c("WSHK018.1a.csv", "WSHK018"),
c("WSHK019.1a.csv", "WSHK019"),
c("WSHK019.2a.csv", "WSHK019"),
c("WSHK020.1a.csv", "WSHK020"),
c("WSHK021.1a.csv", "WSHK021"),
c("WSHK022.1a.csv", "WSHK022"),
c("WSHK022.2a.csv", "WSHK022"),
c("WSHK023.1a.csv", "WSHK023"),
c("WSHK024.1a.csv", "WSHK024"),
# c("WSHK025.1a.csv", "WSHK025"),# 1 line
c("WSHK025.2a.csv", "WSHK025"),
c("WSHK025.3a.csv", "WSHK025"),
c("WSHK026.1a.csv", "WSHK026"),
c("WSHK027.1a.csv", "WSHK027"),
c("WSHK028.1a.csv", "WSHK028"),
c("WSHK029.1a.csv", "WSHK029"),
c("WSHK030.1a.csv", "WSHK030"),
c("WSHK031.1a.csv", "WSHK031"),
c("WSHK032.1a.csv", "WSHK032"),
c("WSHK033.1a.csv", "WSHK033"),
c("WSHK034.1a.csv", "WSHK034"),
c("WSHK035.1a.csv", "WSHK035"),
#c("WSHK035.1b.csv", "WSHK035"),
c("WSHK035.1c.csv", "WSHK035"),
c("WSHK036.1a.csv", "WSHK036"),
c("WSHK036.1b.csv", "WSHK036"),
c("WSHK037.1a.csv", "WSHK037"),
c("WSHK038.1a.csv", "WSHK038"),
c("WSHK039.1a.csv", "WSHK039"),
c("WSHK040.1a.csv", "WSHK040"),
c("WSHK041.1a.csv", "WSHK041"),

```

```

c("WSHK042.1a.csv", "WSHK042"),
c("WSHK043.1a.csv", "WSHK043"),
c("WSHK044.1a.csv", "WSHK044"),
c("WSHK045.1a.csv", "WSHK045"),
c("WSHK046.1a.csv", "WSHK046"),
c("WSHK047.1a.csv", "WSHK047"),
c("WSHK048.1a.csv", "WSHK048"),
c("WSHK050.1a.csv", "WSHK050"),
c("WSHK051.1a.csv", "WSHK051"),
c("WSHK052.1a.csv", "WSHK052"),
c("WSHK053.1a.csv", "WSHK053"),
c("WSHK054.1a.csv", "WSHK054"),
c("WSHK055.1a.csv", "WSHK055"),
# c("WSHK055.1b.csv", "WSHK055"), # 22 lines
c("WSHK055.2a.csv", "WSHK055"),
c("WSHK055.3a.csv", "WSHK055"),
# c("WSHK055.3b.csv", "WSHK055"),# 33 lines
c("WSHK056.1a.csv", "WSHK056"),
c("WSHK056.1b.csv", "WSHK056"),
# c("WSHK057.1a.csv", "WSHK057"), # 28 lines
c("WSHK058.1a.csv", "WSHK058"),
c("WSHK059.1a.csv", "WSHK059"),
c("WSHK060.1a.csv", "WSHK060"),
c("WSHK060.2a.csv", "WSHK060"),
c("WSHK061.1a.csv", "WSHK061"),
c("WSHK062.1a.csv", "WSHK062"),
c("WSHK062.2a.csv", "WSHK062"),
c("WSHK063.1a.csv", "WSHK063"),
c("WSHK063.2a.csv", "WSHK063"),
c("WSHK064.1a.csv", "WSHK064"),
c("WSHK064.2a.csv", "WSHK064"),
c("WSHK064.3a.csv", "WSHK064"),
c("WSHK064.4a.csv", "WSHK064"),
c("WSHK065.1a.csv", "WSHK065"),
c("WSHK066.1a.csv", "WSHK066"),
c("WSHK067.1a.csv", "WSHK067"),
c("WSHK068.1a.csv", "WSHK068"),
c("WSHK069.1a.csv", "WSHK069"),
c("WSHK069.2a.csv", "WSHK069"),
c("WSHK069.3a.csv", "WSHK069"),
c("WSHK070.1a.csv", "WSHK070"),
c("WSHK070.2a.csv", "WSHK070"),
c("WSHK071.1a.csv", "WSHK071"),
c("WSHK072.1a.csv", "WSHK072"),
c("WSHK072.2a.csv", "WSHK072"),
c("WSHK072.2b.csv", "WSHK072"),
c("WSHK073.1a.csv", "WSHK073"),
c("WSHK074.1a.csv", "WSHK074"),
c("WSHK074.2a.csv", "WSHK074"),
c("WSHK075.1a.csv", "WSHK075"),
c("WSHK076.1a.csv", "WSHK076"),
c("WSHK077.1a.csv", "WSHK077"),
c("WSHK077.2a.csv", "WSHK077"),
c("WSHK077.3a.csv", "WSHK077"),

```

c("WSHK078.1a.csv", "WSHK078"),
c("WSHK078.2a.csv", "WSHK078"),
c("WSHK079.1a.csv", "WSHK079"),
c("WSHK079.2a.csv", "WSHK079"),
c("WSHK081.1a.csv", "WSHK081"),
c("WSHK082.1a.csv", "WSHK082"),
c("WSHK083.1a.csv", "WSHK083"),
c("WSHK084.1a.csv", "WSHK084"),
c("WSHK085.1a.csv", "WSHK085"),
c("WSHK085.2a.csv", "WSHK085"),
c("WSHK085.3a.csv", "WSHK085"),
c("WSHK086.1a.csv", "WSHK086"),
c("WSHK086.2a.csv", "WSHK086"),
c("WSHK086.3a.csv", "WSHK086"),
c("WSHK086.4a.csv", "WSHK086"),
c("WSHK086.5a.csv", "WSHK086"),
c("WSHK086.6a.csv", "WSHK086"),
c("WSHK087.1a.csv", "WSHK087"),
c("WSHK088.1a.csv", "WSHK088"),
c("WSHK089.1a.csv", "WSHK089"),
c("WSHK089.2a.csv", "WSHK089"),
c("WSHK090.1a.csv", "WSHK090"),
c("WSHK090.2a.csv", "WSHK090"),
c("WSHK090.3a.csv", "WSHK090"),
c("WSHK091.1a.csv", "WSHK091"),
c("WSHK092.1a.csv", "WSHK092"),
c("WSHK092.2a.csv", "WSHK092"),
c("WSHK093.1a.csv", "WSHK093"),
c("WSHK093.2a.csv", "WSHK093"),
c("WSHK093.2b.csv", "WSHK093"),
c("WSHK093.3a.csv", "WSHK093"),
c("WSHK093.4a.csv", "WSHK093"),
c("WSHK094.1a.csv", "WSHK094"),
c("WSHK094.2a.csv", "WSHK094"),
c("WSHK094.3a.csv", "WSHK094"),
c("WSHK094.4a.csv", "WSHK094"),
c("WSHK094.5a.csv", "WSHK094"),
c("WSHK095.1a.csv", "WSHK095"),
c("WSHK095.2a.csv", "WSHK095"),
c("WSHK096.1a.csv", "WSHK096"),
c("WSHK097.1a.csv", "WSHK097"),
c("WSHK097.2a.csv", "WSHK097"),
c("WSHK098.1a.csv", "WSHK098"),
c("WSHK098.2a.csv", "WSHK098"),
c("WSHK098.3a.csv", "WSHK098"),
c("WSHK099.1a.csv", "WSHK099"),
c("WSHK099.2a.csv", "WSHK099"),
c("WSHK100.1a.csv", "WSHK100"),
c("WSHK101.1a.csv", "WSHK101"),
c("WSHK102.1a.csv", "WSHK102"),
c("WSHK103.1a.csv", "WSHK103"),
c("WSHK104.1a.csv", "WSHK104"),
c("WSHK105.1a.csv", "WSHK105"),
c("WSHK106.1a.csv", "WSHK106"),

```
c("WSHK107.1a.csv", "WSHK107"),
c("WSHK108.1a.csv", "WSHK108"),
c("WSHK109.1a.csv", "WSHK109"),
c("WSHK110.1a.csv", "WSHK110"),
c("WSHK110.2a.csv", "WSHK110"),
c("WSHK111.1a.csv", "WSHK111")), stringsAsFactors = FALSE)
```

```
# create column names and add structure
colnames(track_files) <- c("track.seg", "shark.ID")
csvStruct <- list(x = "x", y = "y", time = "time")
```

```
# build framework for multiple trajectories.
trjs <- TrajsBuild(track_files$track.seg, spatialUnits = "m",
  csvStruct = csvStruct)
```

```
# extracting indices
trjIndices <- function(trj){
```

```
  # Speed along trajectory over time
  derivs <- TrajDerivatives(trj)
  meanSpeed <- mean(derivs$speed)
  speedSD <- sd(derivs$speed)
```

```
  # Distance between start and end of a trajectory
  TrajDist <- TrajDistance(trj, startIndex = 1, endIndex = nrow(trj))
```

```
  # Duration time taken to get from A to B
  Duration <- TrajDuration(trj, startIndex = 1, endIndex = nrow(trj))
```

```
  # GroundCover or CrowSpeed or Velocity = Distance/Duration
  Velocity <- TrajDist/Duration
```

```
  # Track straightness
  Straightness <- TrajStraightness(trj)
```

```
  # Trajectory length - cumulative length of a trajectory which is the total
  # distance travelled
  TrackLength <- TrajLength(trj, startIndex = 1, endIndex = nrow(trj))
```

```
# Return a list with all of the statistics for trajectory
```

```
  list(meanSpeed = meanSpeed,
    TrajDist = TrajDist,
    Duration = Duration,
    Velocity = Velocity,
    TrackLength = TrackLength,
    Straightness = Straightness)
}
```

```
# Calculate all stats for all trajectories and put into a data frame
```

```
trajStats <- TrajsMergeStats(trjs, trjIndices)
trajStats <- as.data.frame(trajStats)
trajStats$shark.ID <- track_files$shark.ID
colnames(trajStats)
trajStats <- trajStats[c("shark.ID", "meanSpeed", "TrajDist",
```

```
"Duration", "Velocity", "TrackLength", "Straightness")]
```

```
# Calculate weighted Means for the stats using 'trajStats$Duration' to
# determine the weights
trajStatsDur <- subset(trajStats, select=c("shark.ID", "Duration"))
trajSumDur <- aggregate(trajStatsDur["Duration"], trajStatsDur["shark.ID"], sum)
trajSumDur <- rename(trajSumDur, "Dur.sum" = "Duration")
trajmerge <- merge(trajStatsDur, trajSumDur, by = "shark.ID", all.x=TRUE)
trajStats$weight <- trajmerge$Duration / trajmerge$Dur.sum
```

```
# Apply weights to the columns
{
  trajStats$meanSpeed <- trajStats$meanSpeed * trajStats$weight
  trajStats$Emax <- trajStats$meanSpeed * trajStats$weight
  trajStats$TrajDist <- trajStats$TrajDist * trajStats$weight
  trajStats$Duration <- trajStats$Duration * trajStats$weight
  trajStats$Velocity <- trajStats$Velocity * trajStats$weight
  trajStats$TrackLength <- trajStats$TrackLength * trajStats$weight
  trajStats$Straightness <- trajStats$Straightness * trajStats$weight
}
```

```
# Populate back to one row for each shark.ID by the values on track segments.
# This will end in a dataframe with the weighted means of the stats from the
# trajectory analysis
trajDat <- aggregate(. ~ shark.ID, data = trajStats, sum)
trajDat <- trajDat[, c("shark.ID", "meanSpeed", "TrajDist",
  "Velocity", "TrackLength", "Straightness")]
```

```
# Read in flight data dataframe and merge with the trajstats trajectory data
moreDat <- read.csv("Flight_WSHK.csv")
dat <- merge(moreDat, trajDat, by = "shark.ID", row.names=FALSE, all = TRUE)
```

```
# Read in objects dataframe
objects <- read.csv("Object_WSHK.csv")
```

```
# Combine some variables
objects$object.all <- objects$sub.object + objects$ray + objects$wrack +
  objects$surfer
objects$interact <- objects$shark + objects$dolphin
objects$big.school <- objects$big.fish.school
objects$small.school <- objects$small.fish.school
objects$fish.school <- objects$big.school + objects$small.school
```

```
# Remove the track segment with the acoustic receiver and boat interaction
# because I don't want this occurrence influencing the model
toBeRemoved<-which(objects$boat.receiver > 1)
toBeRemoved
objects <- objects[-toBeRemoved,]
```

```
#aggregate the track segments in the objects dataframe per shark ID
objects2 <- aggregate(. ~ shark.ID, data = objects, sum)
```

```
# weight according to time tracked
{
```

```

objects2$object.all <- objects2$object.all / objects2$time.tracked
objects2$interact <- objects2$interact / objects2$time.tracked
objects2$shark <- objects2$shark / objects2$time.tracked
objects2$dolphin <- objects2$dolphin / objects2$time.tracked
objects2$big.school <- objects2$big.school / objects2$time.tracked
objects2$small.school <- objects2$small.school / objects2$time.tracked
objects2$fish.school <- objects2$fish.school / objects2$time.tracked
}

# Append dataframe
objects2 <- objects2[, c("shark.ID", "object.all", "interact", "shark",
                        "dolphin", "big.school", "small.school", "fish.school")]
dat <- merge(dat, objects2, by = "shark.ID", row.names=FALSE, all = TRUE)

# Insert manually calculated velocity for SHK001 - 95 m track in 146 seconds
# Note that this was an opportunistic track from the NSW DPI drone trials,
# which is why it wasn't a clean track, but still usable for velocity
SHK001dist <- distm(c(153.43566730, -29.10875007), c(153.43571570, -29.10961127),
                  fun = distHaversine)
SHK001time <- 126
SHK001Crow <- SHK001dist/SHK001time
dat$Velocity[1] <- SHK001Crow
dat$TrackLength[1] <- 1

# Eliminating NAs. Note that 'velocity' will now have one less NA than the
# other track analytics
validRows = which(!is.na(dat[, "Velocity"]))
dat2 = dat[validRows, ]
table(dat2$wind.dir) # there are 15 cases of nil wind so will run the analysis
# using simplified wind direction of onshore vs offshore
# vs NIL, which was beach specific in its direction
# classification

# turn tidal stage into circular reference for analysis
dat2$tide.sin <- sin(2*pi*dat2$tide/6)
dat2$tide.cos <- cos(2*pi*dat2$tide/6)

# time as a covariate, similar to tide
# Convert time to a numeric and take the mid-point of the track
dat2$track.start <- as.character(dat2$track.start)
dat2$track.end <- as.character(dat2$track.end)
min <- '00:00:01'
max <- '24:00:00'
as.numeric(hms(min)) # min = 1
as.numeric(hms(max)) # max = 86400
dat2$track.start <- as.numeric(hms(dat2$track.start))
dat2$track.end <- as.numeric(hms(dat2$track.end))
dat2$time.mean <- dat2$track.start+(dat2$track.end-dat2$track.start)/2

# see if there was an even spread of samples throughout the day
plot(dat2$time.mean) # no pattern observed

# turn the mean time of the track to a circular reference scale on the 24hr time cycle
dat2$time.sin <- sin(2*pi*dat2$time.mean/86400)

```

```

dat2$time.cos <- cos(2*pi*dat2$time.mean/86400)
dat2$time.sin2 <- sinpi(2*pi*dat2$time.mean/86400)
dat2$time.cos2 <- cospi(2*pi*dat2$time.mean/86400)
# Time.sin now goes from high in the morning (0.971) to low in the evening (-0.938)
# and crosses into negative during midday
# Time.cos now starts relatively high numbers (-2.38) and goes low (-1)
# at midday and comes back up again in the evening (-3.47)

```

```

# create analysis dataframe
data = dat2[, c("date", "shark.ID", "location", "loc.type", "time.tracked",
               "vis", "tide.sin", "tide.cos", "time.mean", "time.sin", "time.cos",
               "length", "wind.dir2", "wind.str", "temp", "cloud", "object.all",
               "interact", "shark", "dolphin", "big.school", "small.school",
               "fish.school", "swim.direction", "meanSpeed", "Velocity",
               "Straightness")]

```

```

# Specifying variables as factor or numeric
data[, "tide.sin"] = as.numeric(data[, "tide.sin"])
data[, "tide.cos"] = as.numeric(data[, "tide.cos"])
data[, "time.sin"] = as.numeric(data[, "time.sin"])
data[, "time.cos"] = as.numeric(data[, "time.cos"])
data[, "wind.dir2"] = as.factor(data[, "wind.dir2"])
data[, "wind.dir2"] = as.factor(data[, "wind.dir2"])
data[, "vis"] = as.numeric(data[, "vis"])
data[, "cloud"] = as.numeric(data[, "cloud"])

```

End of Part 1

```

# Creating the base code for distribution plots to assist in assessing model
# outputs
plotPosteriors <- function(stanOutput, numRows = 1, numCols = 1)
{
  fade <- function(colors,alpha)
  {
    rgbcols <- col2rgb(colors)
    rgb(rgbcols[1,],rgbcols[2,],rgbcols[3,],alpha,max=255)
  }
  paramNames <- names(stanOutput$coefficients)
  randomEffectNames <- startsWith(paramNames, "b[(")
  interceptNames <- startsWith(paramNames, "(Intercept)")
  relevantParamNames <- paramNames[!(randomEffectNames & !interceptNames)]
  relevantParamInd <- which(paramNames %in% relevantParamNames)
  par(mfrow = c(numRows, numCols))
  for(i in 1:length(relevantParamNames))
  {
    theseSamples <- c(stanOutput$stanfit@sim$samples[[1]][relevantParamInd[i]][1],
                      stanOutput$stanfit@sim$samples[[3]][relevantParamInd[i]][1],
                      stanOutput$stanfit@sim$samples[[2]][relevantParamInd[i]][1],
                      stanOutput$stanfit@sim$samples[[4]][relevantParamInd[i]][1])
    probbGreaterZero = round(length(which(theseSamples >= 0))/length(theseSamples), 3)
    probbLessThanZero = round(length(which(theseSamples < 0))/length(theseSamples), 3)
    d <- density(theseSamples)
    plot(x = range(d$x), y = c(0, max(d$y)), col = "white",

```



```

      xlab = relevantParamNames[i], ylab = "Density")
posInd = which(d$x > 0)
negInd = which(d$x < 0)
if(length(posInd) > 0)
{
  ind <- posInd
  polygon(x = c(d$x[ind], rev(d$x[ind]), d$x[ind][1]),
          y = c(rep(0, length(d$y[ind])), rev(d$y[ind]), 0),
          border = NA, col = fade("blue", 100))
}
if(length(negInd) > 0)
{
  ind <- negInd
  polygon(x = c(d$x[ind], rev(d$x[ind]), d$x[ind][1]),
          y = c(rep(0, length(d$y[ind])), rev(d$y[ind]), 0),
          border = NA, col = fade("red", 100))
}

#polygon(x = c(d$x, rev(d$x), d$x[1]), y = c(rep(0, length(d$y)), rev(d$y), 0),
# border = NA, col = fade("blue", 50))
lines(x = c(0, 0), y = c(0, 2*max(d$y)))
text(as.character(probLessThanZero), x = 0, y = 0.5*max(d$y), pos = 2)
text(as.character(probGreaterThanZero), x = 0, y = 0.5*max(d$y), pos = 4)
}
}

```

```

# Check for NAs again
md.pattern(data) # no NAs in velocity but the other track metrics have one

```

```

# eliminate NAs using meanSpeed
validRows1 <- which(!is.na(data[, "meanSpeed"]))
datb <- data[validRows1, ]

```

```

# Create duplicate dataframe and rescale all numeric variables
datc <- datb
rescale <- function(x) (x-min(x))/(max(x) - min(x))
ind <- sapply(datc, is.numeric)
datc[ind] <- lapply(datc[ind], rescale)

```

```

#####

```

```

##### Swim direction analysis #####

```

```

# create duplicate dataframe from swim direction and simplify the dependant variable

```

```

# swim direction to indicate either 'north' or 'south' directions

```

```

datz <- datc

```

```

datz[, "swim.dir.simple"] = NA

```

```

for(i in 1:nrow(datz))

```

```

{
  if(datz[i, "swim.direction"] == "Circular" | datz[i, "swim.direction"] == "East" |
    datz[i, "swim.direction"] == "West")
  {

```

```

    datz[i, "swim.dir.simple"] = "NA"
  }
}

```

```

if(datz[i, "swim.direction"] == "North")

```

```

{
  datz[i, "swim.dir.simple"] = "1"
}
if(datz[i, "swim.direction"] == "South")
{
  datz[i, "swim.dir.simple"] = "0"
}
}

datz[, "swim.dir.simple"] <- as.numeric(datz[, "swim.dir.simple"])
validRows <- which(!is.na(datz[, "swim.dir.simple"]))
datz <- datz[validRows, ]
datz[, "swim.dir.simple"] <- as.numeric(datz[, "swim.dir.simple"])

#Swim direction full model using binomial family distribution
fitSwim1 <- stan_glmer (swim.dir.simple ~ length + vis + wind.str + wind.dir2 +
  tide.sin + tide.cos + time.sin + time.cos +
  (1|date) + (1|location), data = datz,
  family = binomial(link = "logit"))

# print summary and plot the posterior distributions
print(summary(fitSwim1), digits = 3)
plotPosteriors(fitSwim1, numRows = 4, numCols = 3)
table(data$location, data$swim.direction)

# Check model
launch_shinystan(fitSpd)

#####
##### Mean Speed #####
# Full model for Mean speed
fitSpd <- stan_lmer (meanSpeed ~ length + vis + wind.str + wind.dir2 + tide.sin +
  tide.cos + time.sin + time.cos + object.all + fish.school +
  interact + (1|date) + (1|location), data = date)

# print summary and plot posterior distributions
print(summary(fitSpd), digits = 3)
plotPosteriors(fitSpd, numRows = 4, numCols = 3)
posterior_interval(fitSpd, prob=0.95)

# Check model
launch_shinystan(fitSpd)

# Calculating effect sizes for mean speed
# length coeff (0.990 prob)
0.290*(1.2258-0.5441)/(3.980-1.915) # 0.0957 corrected coefficient
0.122*(1.2258-0.5441)/(3.980-1.915) # 0.0403 corrected sd

# fish.school ceoff (0.996 prob)
0.488*(1.2258-0.5441) # 0.3300 Coeff
0.180*(1.2258-0.5441) # 0.1227 sd

# interactions coeff (0.939 prob)

```

```

0.214*(1.2258-0.5441) # 0.149 coeff
0.140*(1.2258-0.5441) # 0.095 sd

# time sine (0.949 prob)
# transforming to time points that are of interest.
-0.142*(1.2258-0.5441)/(max(datb$time.sin)- min(datb$time.sin)) # -0.0507 coeff
0.087*(1.2258-0.5441)/(max(datb$time.sin)- min(datb$time.sin)) # 0.0311 sd
# time cosine (0.952 prob)
0.118*(1.2258-0.5441)/(max(datb$time.cos)- min(datb$time.cos)) # 0.129 coeff
0.084*(1.2258-0.5441)/(max(datb$time.cos)- min(datb$time.cos)) # 0.075 sd
# note that the transformed coefficients still refer to sine and cosine transformed versions of time
# This result indicates that sharks have very slightly higher mean swim speeds in the afternoon
# (or as the day progresses), as indicated by Sine. A stronger effect is that sharks swim faster in the
# morning and afternoon as opposed to around midday, as indicated by cosine.

```

```

# Time.sin now goes from high in the morning (0.971) to low in the evening (-0.938)
# and crosses into negative during midday
# Time.cos now starts relatively high numbers (-2.38) and goes low (-1)
# at midday and comes back up again in the evening (-3.47)

```

```

#####
##### Straightness #####
# Full model for straightness
fitStraight <- stan_lmer (Straightness ~ length + vis + wind.str + wind.dir2 +
                        tide.sin + tide.cos + time.sin + time.cos + object.all +
                        fish.school + interact + (1|date) + (1|location),
                        data = date)

```

```

# print summary and plot posterior distributions
print(summary(fitStraight), digits = 3)
plotPosteriors(fitStraight, numRows = 4, numCols = 3)
posterior_interval(fitStraight, prob=0.95)

```

```

# check model
launch_shinystan(fitStraight)

```

```

# Calculating effect sizes for straightness
# length coeff (prob 0.986)
0.294*(0.99759-0.02904)/(3.980-1.915) # 0.1390 corrected coefficient
0.133*(0.99759-0.02904)/(3.980-1.915) # 0.0624 corrected sd

```

```

# fish.school (prob 0.965)
-0.354*(0.99759-0.02904) # -0.3390 coeff
0.195*(0.99759-0.02904) # 0.1889 sd

```

```

# time cosine (0.905 prob)
0.118*(0.99759-0.02904)/(max(datb$time.cos)- min(datb$time.cos)) # 0.1504 coeff
0.089*(0.99759-0.02904)/(max(datb$time.cos)- min(datb$time.cos)) # 0.1134 sd
summary(data)

```

```

#####
#### rate-of-travel / Velocity #####

```

```

# redo dataframe to include the extra velocity calculation (line 337)
validRows2 <- which(!is.na(data[, "Velocity"]))
datd <- data[validRows2, ]

# and rescale the variables again
rescale <- function(x) (x-min(x))/(max(x) - min(x))
ind <- sapply(datd, is.numeric)
datd[ind] <- lapply(datd[ind], rescale)

# Velocity full model
fitVcity <- stan_lmer(Velocity ~ length + vis + wind.str + wind.dir2 + tide.sin +
  tide.cos + time.sin + time.cos + object.all + fish.school +
  interact + (1 | date) + (1|location), data=datd)

# print summary and plot posterior distribution
print(summary(fitVcity), digits = 3)
plotPosteriors(fitVcity, numRows = 4, numCols = 3)
posterior_interval(fitVcity, prob=0.95)

# check model
launch_shinystan(fitVcity)

# Calculating effect sizes for velocity
# length coeff (prob 1.00)
0.349*(1.1109-0.0158)/(3.980-1.915) # 0.1877 corrected coefficient
0.1044*(1.1109-0.0158)/(3.980-1.915) # 0.055 corrected sd

# time cosine (0.988 prob)
0.170*(1.1109-0.0158)/(max(datb$time.cos)- min(datb$time.cos)) # 0.2406 coeff
0.072*(1.1109-0.0158)/(max(datb$time.cos)- min(datb$time.cos)) # 0.1037 sd
# note that the transformed coefficients still refer to cosine transformed versions of time
# This result indicates that sharks have higher net velocities in the morning and afternoon
# as opposed to around midday, as indicated by cosine.

```