Encounter rate simulator

This vignette describes our analysis in a step-by-step guide.

Setup

1. **Setup this directory** on your local computer by either cloning this repo via `git` or downloading it as a zipped folder.
2. **Start a new script** in RStudio and save it within this directory's R folder.
3. **Load libraries** at the top of your script. Install the packages you don't yet have.

```
library(readr)
library(dplyr)
library(truncnorm)
library(solartime)
library(bear)
library(DescTools)
library(sf)
library(mattlib)
library(manipulateWidget)
```

4. **Source functions** at the top of your script:

```
source("function-sim.R")
source("function-encounter.R")
source("function-impacts.R")
```

Marine traffic data

Formatting AIS

Subsequent steps will require AIS data that meet the following criteria:

- Each row contains details for a single vessel.
- There is a column named `id`, containing a unique identifier for each vessel (e.g., MMSI).
- There is a column named `type`, containing the category or class of vessel.
- There is a column named `sog`, containing speed-over-ground in meters per second.
- There is a column named `length`, containing vessel length in meters.
- There are columns named `year` (e.g., 2018), `month` (e.g., 7 for July), and `doy` for day-of-year (e.g., 182 for July 1st).
- There is a column named `hour` for the local hour of day in which this record occurred.
- There are columns named `x` and `y` for longitude and latitude, respectively, in decimal degrees.
- Records for each unique vessel are filtered to only one per hour. This is to ensure that frequently-reporting vessels are not overrepresented but that vessels that occur often or linger in the area are not underrepresented.

Example script

Here is the code we used to format the raw AIS data (provided to us by the Canadian Coast Guard). You may use this dataset to familiarize yourself with the simulator.

First, read in the raw data:

```
ais <- read_csv("../data/ais/ais-2018.csv")
```

Check out the AIS data in its raw form:

```
nrow(ais)
#> [1] [illegible]

head(ais)
[illegible]
```

Now save only certain fields to a new dataframe, modifying column names and in some case changing units:

```
ais <- data.frame(id=ais$ID,
                  type=ais$Type,
                  sog=ais$SOG*0.51444,
                  length=ais$Length,
                  year=ais$Year,
                  month=ais$Month,
                  hour=as.numeric(gsub(":","",substr(ais$Time,1,2))),
                  doy=as.numeric(strftime(ais$Local.Time,format="%j")),
                  time=as.POSIXct(ais$Local.Time),
                  x=ais$Longitude,
                  y=ais$Latitude)
```

Now we filter the record so that each vessel is included only once per hour at most.

```
ais$id_hour <- paste0(ais$id,'-',ais$hour)
ais <- ais[! duplicated(ais$id_hour),]
ais$id_hour <- NULL
nrow(ais)
#> [1] [illegible]
```

Optionally, add columns for the elevation of the sun during each record. This would prove useful if you want to conduct strip-series assessments in a daily-explicit framework (daytime vs. nighttime risks and impacts).

```
ais$sol <- solartime::computeSunPositionDoyHour(doy=ais$doy,
                                                hour=ais$hour,
                                                latDeg=ais$y,
                                                longDeg=ais$x,
                                                timeZone=-7)[,1] * (180/pi)
```

You can then use solar elevation to determine which events occur during day (`sol > 0`) and which during night (`sol < 0`). Note that some studies have used civil twilight or other metrics to draw this distinction. Here we are just keeping things simple.

```
ais$night <- 0
ais$night[ais$sol < 0] <- 1
```

This is the resulting dataframe that will get passed to subsequent steps:

```
# Check out result
head(ais)
[illegible]
```

Prepare parameters for simulator

Marine traffic

First, filter the AIS data to the vessel class of interest (column `type` in the `ais` data frame). You can supply multiple classes if you wish, or skip this step to keep all classes.

```
traffic <- ais

# Define vessel types of interest
type_ops <- c("large ship","tanker")

# Filter to vessel type
matches <- which(tolower(as.character(ais$type)) %in% type_ops)
traffic <- traffic[matches,]
nrow(traffic)
#> [1] 266

# Filter to valid entries
traffic <- traffic %>% dplyr::filter(length > 5,
                                     sog > 2)

# Filter to study area
traffic <- traffic %>% dplyr::filter(x > -129.6,
                                     x < -129.3,
                                     y > 52.8,
                                     y < 53.35)

nrow(traffic)
#> [1] 8
```

Now simplify these data to only the essential parameters and add an approximation of vessel width:

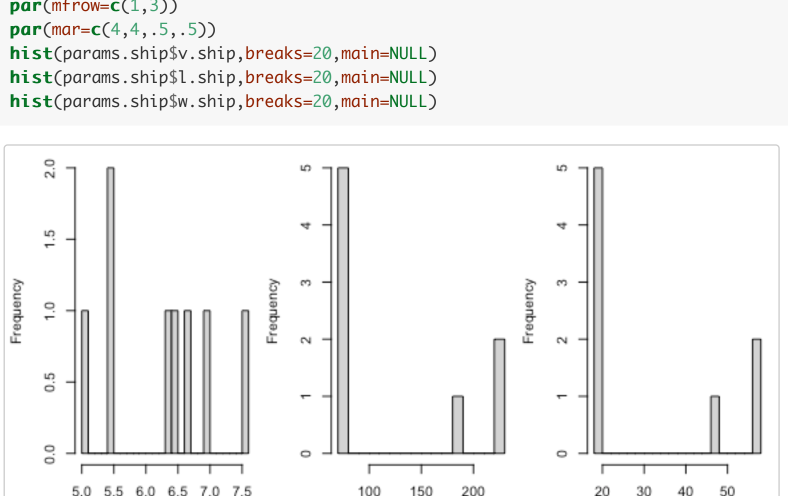
```
params.ship <- traffic %>%
  dplyr::select(v.ship = sog,
                l.ship = length) %>%
  dplyr::mutate(w.ship = 0.25*l.ship)
```

Check out the finalized traffic parameter set:

```
head(params.ship)
#>    v.ship l.ship w.ship
#> 1 5.45264     74  18.50
#> 2 5.40120     74  18.50
#> 3 5.04112     72  18.00
#> 4 6.32712     79  19.75
#> 5 7.51024     74  18.50
#> 6 6.63576    225  56.25
```

Visualize the distributions of these parameters (there are few values, since traffic in this area is currently quite rare):

```
par(mfrow=c(1,3))
par(mar=c(4,4,.5,.5))
hist(params.ship$v.ship,breaks=20,main=NULL)
hist(params.ship$l.ship,breaks=20,main=NULL)
hist(params.ship$w.ship,breaks=20,main=NULL)
```



To **simulate additional traffic** on top of the traffic already present, you can add rows to `params.ship`. This code emulates the traffic increase expected in 2023 in Gitga'at waters:

```
new_transits <- 750
v.ship <- rep(5.144,times=new_transits)# 10 knots or 0.5144 m/s
l.ship <- rep(300,times=new_transits)
w.ship <- l.ship*0.25
projected.traffic <- data.frame(v.ship,l.ship,w.ship)

params.ship <- rbind(params.ship, projected.traffic)
```

Whales

The parameters that characterize a whale in the encounter simulator can be defined as either a single value (e.g., 20 meters), or as a distribution of values from which to draw random values. In this example, we will do the latter.

(*Note:* In the analysis, this code is implemented in `00_whale.R`).

Define the size of the distributions you will use:

```
n <- 1000
```

These distributions will be truncated-normal distributions, to ensure that no value is unrealistically small or large.

Whale velocity

Velocity should be provided as meters per second. We draw values from the acoustic tracks of fin whales from the same study area in Hendricks et al. (2021).

```
v.whale = truncnorm::rtruncnorm(n,
                                a=0,
                                b=2.63,
                                mean=1.3611,
                                sd=.5)
```

Whale directivity

Whale directivity, which we define in our paper as the standard deviation of change in course heading from one minute to the next, is also drawn from Hendricks et al. (2021). In that paper, 62 acoustic tracks of at least 30 minutes duration were analyzed. Refer to `prep_delta_sd.R` for details on how this was determined. Note that, in this analysis, the standard deviation of course changes were log-transformed so that they followed a Gaussian distribution.

```
# Produce distribution of log-transformed values
delta.sd = rnorm(n,
                 mean = 1.0424,
                 sd = 0.82477)

# Revert from log-transformed to actual SD of course change
delta.sd <- exp(delta.sd)
```

Whale dimensions

Whale length is drawn from UAV-based photogrammetry measurements for fin whales in our study area, published in Keen et al. (2021).

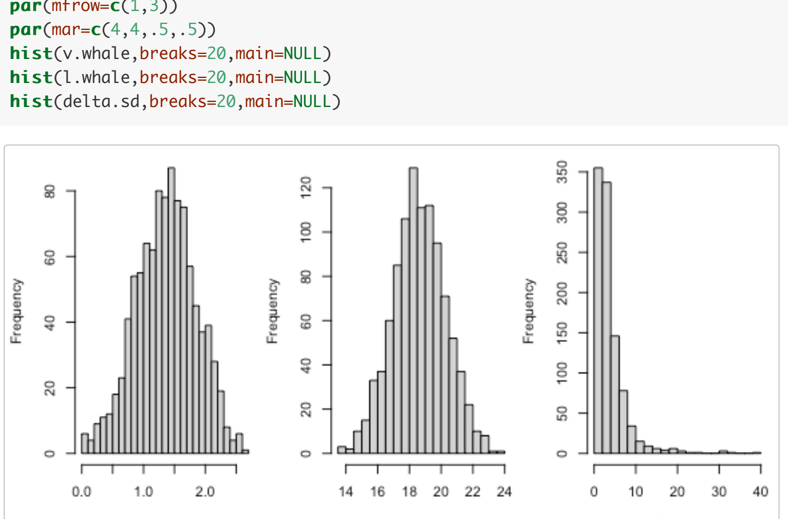
```
l.whale = truncnorm::rtruncnorm(n,0,40,18.60375,1.649138)
```

To estimate whale widths, we use the ratio of fluke width to body length from Keen et al. (2021).

```
w.whale = .2074*l.whale
```

Visualize the distributions of these parameters:

```
par(mfrow=c(1,3))
par(mar=c(4,4,.5,.5))
hist(v.whale,breaks=20,main=NULL)
hist(l.whale,breaks=20,main=NULL)
hist(delta.sd,breaks=20,main=NULL)
```



Run simulator

We are now ready to run our encounter rate simulation.

Define the filename of the `R` data object that will store your results.

```
results_filename <- 'demo_results.rds'
```

Define the number of iterations you want. We suggest no fewer than 100.

```
iterations <- 100
```

Stage empty objects into which results will be placed during each iteration.

```
encounter_tally <- c() # simple tally of imminent encounters
summaries <- data.frame() # summaries of each iteration
records <- list() # list of detailed info for each imminent encounter
```

```
# Setup a multi-pane plot to watch results
par(mfrow=c(3,3))

# Loop through iterations
for(b in 1:iterations){

  # Run simulator
  sim_b <- encounter_simulator(params.ship=params.ship,
                               v.whale=v.whale,
                               l.whale=l.whale,
                               w.whale=w.whale,
                               delta.sd=delta.sd,
                               B=100,
                               toplot=FALSE)

  # Summary of each iteration
  summary_b <- sim_b$summary
  summary_b$iteration <- b
  summary_b

  # Add to summary df for all iterations
  summaries <- rbind(summaries, summary_b)

  # Note number of imminent encounters that occurred in this iteration
  encounters <- which(summary_b$encounter==1)
  tot_encounters <- length(encounters)
  encounter_tally <- c(encounter_tally,tot_encounters)

  # Details for each iteration
  records_b <- sim_b$records

  # Get records for runs that results in an encounter
  if(tot_encounters > 0){
    encounter_records <- records_b[encounters]
    length(encounter_records)
    records <- c(records, encounter_records)
  }

  # Save results to RDS in each iteration to ensure work is never lost
  results_list <- list(encounter_tally = encounter_tally,
                       summaries = summaries,
                       records = records)
  saveRDS(results_list,
          file=results_filename)

  # Print status report
  print(paste0(Sys.time()," | Run ",b," | ",
               tot_encounters," imminent encounter(s) ..."))

}
```

Visualize results

Read in your results object, if you need to:

```
results <- readRDS(results_filename)
```

Determine mean expected encounter rate:

```
DescTools::BootCI(results$encounter_tally, FUN=mean)
#>      mean    lwr.ci    upr.ci
#> 10.420000  9.879736 10.976180

sd(results$encounter_tally)
#> [1] 2.9135
```

```
par(mfrow=c(2,2))
par(mar=c(4.2,4.2,2,1))
hist(results$encounter_tally,
     xlab=NULL, main='Encounter rate',
     breaks=seq(0,1.2*max(results$encounter_tally),length=20))

hist(results$summaries$proximity_m,
     breaks=20, xlab=NULL,
     main="Nearest proximity")

hist(results$summaries$whale_hdg,
     breaks=20, xlab=NULL,
     main="Whale heading at proximity")

hist(results$summaries$ship_hdg,
     breaks=20, xlab=NULL,
     main="Vessel heading at proximity")
```