Supplementary Material 4

Estimating fishing effort from highly resolved geospatial data: focusing on passive gears

15 June 2023

The code presented here is a supplement to "Estimating fishing effort from highly resolved geospatial data: focusing on passive gears" by Tania Mendo, Gildas Glemarec, Jaime Mendo, Einar Hjorleifsson, Sophie Smout, Simon Northridge, Julien Rodriguez, Anna Mujal-Colilles and Mark James.

In this document we present a method for estimating gear soak time applied to a gillnet fishery in Denmark. This method is described in more detail in the paper.

We provide a sample data set of vessel tracking data collected every 1 minute from 4 trips conducted by 1 vessel. The sample data can be found here, DOI: XXXXXXXXXXXX

First, call the libraries into R.

Libraries needed

```
library(tidyverse)
library(sf)
library(sfheaders)
library(data.table)
```

Data needed

This script estimates the soak duration of passive fishing gears using spatial data at regular time intervals. Input data must provide the following variables:

Input data: The columns needed to run the code below are: trip_id: a unique trip identifier timestamp: date and time x: longitude in UTM y: latitude in UTM dist: distance between consecutive observations rf_prediction: inferred vessel activity: hauling gear (fishing) or not hauling gear (not_fishing) activity_id: unique identifier for each hauling or deployment event.

#User input parameters

The following parameters need to be decided by the user, based on knowledge on each fishery.

```
## Define a buffer area that will be used to match a setting event and a hauling event
width_buffer <- 75 # in metres

## Maximum number of days that a hauling event can occur for a matching deployment event
time_frame_fishing <- 30

## Specify Coordinate Reference System
crs_utm <- 32630

## Define threshold for length of the intersection, in this case 70% of the
```

```
#length of the distance covered during the hauling event.
overlap_threshold <- 0.7

## Indicate the time interval for your (regularised/interpolated) dataset
time_interval <- 60 # in seconds
```

Estimate soak time

The following code will look at each trip, identify the hauling events, and then go sequentially back in time to try to allocate the most likely setting event that corresponds to each hauling event (activity_id).

```
df <- read.csv("df_test_data.txt")
df$timestamp <- as.POSIXct((df$timestamp),
                           format = "%Y-%m-%dT%H:%M:%OSZ")
df$date <- as.Date(df$date)

soak_time <- data.frame() ### create data frame to store soak time data

df_fishing <- df %>%
  dplyr::filter(rf_prediction == "fishing")#choose only hauling events

trips_list <- unique(df_fishing$trip_id)#2 trips with hauling events in this example

for (k in unique(trips_list)) {

  hauls <- subset(df_fishing, df_fishing$trip_id == k) # select a trip

  starting_date <- unique(hauls$date)
  end_date <- starting_date - time_frame_fishing # only consider 30 days before
  #hauling as potential deployment events

  ### select only predicted/inferred non-fishing locations and for the timeframe specified

  df_trip_not_fishing <- df %>%
    filter(rf_prediction == "not_fishing")%>%
    dplyr::filter(date >= end_date& date <= starting_date )

  ### Loop for dates
  dates_list <- sort(unique(df_trip_not_fishing$date),
                     decreasing = TRUE)

  for (j in unique(dates_list)) {
     #j = dates_list[1]

    points_not_fishing <- df_trip_not_fishing %>%
      dplyr::filter(date == j) %>%
      ## construct sf object
      st_as_sf(coords = c("x", "y"), crs = crs_utm)

    ### create sf line objects, specify grouping by hauling event
    lines_fishing <- hauls %>%
      # dplyr::filter(date == j) %>%
```

```
  st_as_sf(coords = c("x", "y"), crs = crs_utm) %>%
  dplyr::group_by(activity_id) %>%
  dplyr::summarise(do_union = FALSE) %>%
  st_cast("LINESTRING")

### Add a buffer around the hauling events to allow for potential distances
### between the net being hauled from a different position due to tide, or
### length of the rope connecting the net to the buoy
lines_fishing_buff <- st_buffer(lines_fishing,
                                dist = width_buffer)

### Intersect the polygons on hauling activities with the lines for non
### fishing activities
Inter <- st_intersection(points_not_fishing,
                         lines_fishing_buff)

if (nrow(Inter) == 0) {
  print(paste("j =", j, "No intersection"))
} else {

  Inter <- sf_to_df(Inter, fill = TRUE) %>%
    ### Rename activity variables
    dplyr::rename(hauling_id = activity_id.1,
                  setting_id = activity_id) %>%
    dplyr::mutate(intersection_id = paste(setting_id,
                                          hauling_id)) %>%
    group_by(intersection_id) %>%
    ### Estimate dt, the time between consecutive observations per group
    dplyr::mutate(dt = difftime(strptime(timestamp,
                                         "%Y-%m-%d %H:%M:%S"),
                                lag(strptime(timestamp,
                                             "%Y-%m-%d %H:%M:%S"))),
                  dt = as.numeric(dt, units = 'secs')) %>%
    dplyr::mutate(dt = if_else(is.na(dt),
                               time_interval,
                               dt)) %>%
    ### Update the variable intersection_id
    dplyr::mutate(counter = rleid(dt)) %>%
    dplyr::mutate(intersection_id = paste(intersection_id, counter)) %>%
    ### Distances in x and y
    dplyr::mutate(dx = c(0, abs(diff(x))),
                  dy = c(0, abs(diff(y)))) %>%
    dplyr::mutate(dist = sqrt(dx^2 + dy^2)) %>%
    dplyr::mutate(intersection_length = sum(dist, na.rm = T)) %>%
    ungroup() %>%
    group_by(hauling_id) %>%
    ### Select the rows with highest values of intersection_length per hauling_id
    dplyr::slice_max(intersection_length, n = 1)


  #Rename the intersection_id when non-fishing data point leaves the fishing buffer

  ### Choose the longest intersection length between hauls and sets for each haul
```

```
    haul_length <- hauls %>%
      dplyr::rename(hauling_id = activity_id) %>%
      dplyr::group_by(hauling_id) %>%
      dplyr::summarise(haul_length = sum(dist, na.rm = TRUE))

    Inter <- left_join(Inter,
                       haul_length,
                       by = "hauling_id")

    Inter <- Inter %>%
      dplyr::filter(intersection_length > haul_length * overlap_threshold)

    if (nrow(Inter) == 0) {
      print(paste("j =", j, "No valid intersection"))
    } else {

      ### Information about hauling events
      hauling_info <- hauls %>%
        dplyr::filter(activity_id %in% unique(Inter$hauling_id)) %>%
        dplyr::rename(hauling_id = activity_id) %>%
        group_by(hauling_id) %>%
        dplyr::summarise(mean_timestamp_hauling = mean(timestamp),
                         hauling_id = unique(hauling_id))

      ### Information about deployment events
      deployment_info <- Inter %>%
        dplyr::group_by(intersection_id,
                        hauling_id) %>%
        dplyr::summarise(mean_timestamp_setting = mean(timestamp))

      ## Estimate soak time by joining a setting event with the corresponding
      #hauling event
      soak <- left_join(hauling_info,
                        deployment_info,
                        by = "hauling_id") %>%
        ## In this specific fishery, a net may be hauled on the same day it was set.
        #However, the minimum soak duration is 2 hours
        dplyr::filter(mean_timestamp_hauling - mean_timestamp_setting >
                        lubridate::hours(2))

      if (nrow(soak) == 0) {

        print(paste("j =", j, "No intersection after thresholds applied"))
      } else {

        ## Update the soak time data with the soak time values from day j
        soak_time <- rbind(soak_time, soak)
        sel <- factor(soak$hauling_id)
        hauls <- hauls[!hauls$activity_id %in% sel,]

        ## Check if all hauls have been accounted for
        if (length(levels(as.integer(hauls$activity_id)) ==
                   levels(as.integer(soak$hauling_id))) == 0) {
```

```
            print(paste("j =", j, "All hauls accounted for"))

          } else {

          }
        }
      }
    }
  }
}

### Estimate soak time for each matching pair of set/haul
soak_time$est_soak_time <- difftime(soak_time$mean_timestamp_hauling,
                                    soak_time$mean_timestamp_setting,
                                    units = "hours")

write.csv(soak_time, "soak_time_final.csv", row.names = FALSE)
```