Analysis - Tristan da Cunha vessel traffic in ATBA

Cian Luck

09 Aug 2021

Table of Contents

Background

On March 16, 2011 the bulk carrier MS Oliva, en route from Brazil to Singapore, ran aground at Spinners Point on Nightingale Island. Soon afterwards, the vessel broke apart, spilling more than 300,000 gallons of oil and contaminating thousands of northern rockhopper penguins. The remoteness of Tristan da Cunha made mounting a rapid, large-scale containment logistically impossible. This event highlighted the elevated risk of environmental incidents due to the remoteness of Tristan da Cunha and its proximity to major shipping channels across the South Atlantic. To reduce the risk of another ship running aground, in April 2020 the government implemented a recommended 20 nautical mile exclusion zone around each island within the archipelago, and designated these as areas to be avoided (ATBA) by transiting vessels greater than 400 gross tonnage (gt).

The Blue Belt program is an initiative by the UK government to support the UK Overseas Territories with the protection and sustainable management of their marine environments. As part of this program, the UK Marine Management Organisation (MMO) has been providing support to Tristan da Cunha by detecting transits through the ATBA and coordinating efforts to communicate the presence of the ATBA to relevant flag states and vessel operators.

Through consultation with the MMO, we identified a number of complementary analyses that could add value to the ongoing monitoring of the ATBA. To that end, GFW analysed a global dataset of automatic identification system (AIS) transmissions to:

1. Quantify transits through the ATBA
2. Provide a full port-to-port track analysis of transiting vessels to identify which port authorities to prioritize in communicating the ATBA
3. Identify gaps in AIS transmission by fishing vessels potentially related to deliberate disabling of AIS devices

Setup

Load packages:

```
library(tidyverse)     # data manipulation and plotting
library(bigrquery)     # querying data through BigQuery
library(DBI)           # database interface
library(fishwatchr)    # internal R package developed by Global Fishing Watch for
common in-house analyses and functions
library(glue)          # used to format SQL queries in R
library(lubridate)     # format date time objects
library(here)          # useful package for specifying file locations
library(sf)            # simple features - used for spatial analysis
library(extrafont)     # load extra fonts for plotting
library(ggrepel)       # useful package for adding labels to ggplot objects
```

Establish connection to Big Query project:

```
con <- DBI::dbConnect(drv = bigrquery::bigquery(),
                      project = "world-fishing-827",
                      use_legacy_sql = FALSE)
```

For this analysis we queried data from the table at `world-fishing-827:scratch_cian.tdc_vessel_traffic_atba_2019_2021` which was created using the following query. Note that this is a big table with more than 49 million rows. Therefore throughout this analysis we used queries to pull manageable subsets of these data.

Load shapefiles for mapping:

```
# Areas to be Avoided (ATBA)
atba_sf <- sf::st_read("geodata/tdc_atba/ATBA_consolidate_25nm_buffer_wgs84.shp")
st_crs(atba_sf) <- 4326 # set coordinate reference system to WGS84

# Shapefiles of Tristan da Cunha and Gough Island - sourced from OpenStreetMap
tdc_sf <- sf::st_read("geodata/tdc_osm/tristan_da_cunha_archipelago_osm.shp")
st_crs(tdc_sf) <- 4326

# Shapefile of Tristan EEZ only - sourced from Marine Regions
eez_tdc <- fishwatchr::eez_sf %>% filter(MRGID_EEZ1 == 8382)
```

Map of vessel activity around Tristan da Cunha

Produce a map of total vessel traffic around Tristan da Cunha between 01 Jan 2019 and 30 Jun 2021.

Load the query to create a grid of total vessel activity (hours) per grid cell (0.1° x 0.1°):

```
query_1 <- readr::read_file(str_c("queries",
"q_tdc_atba_gridded_vessel_activity.sql", sep="/"))
```

Run the query:

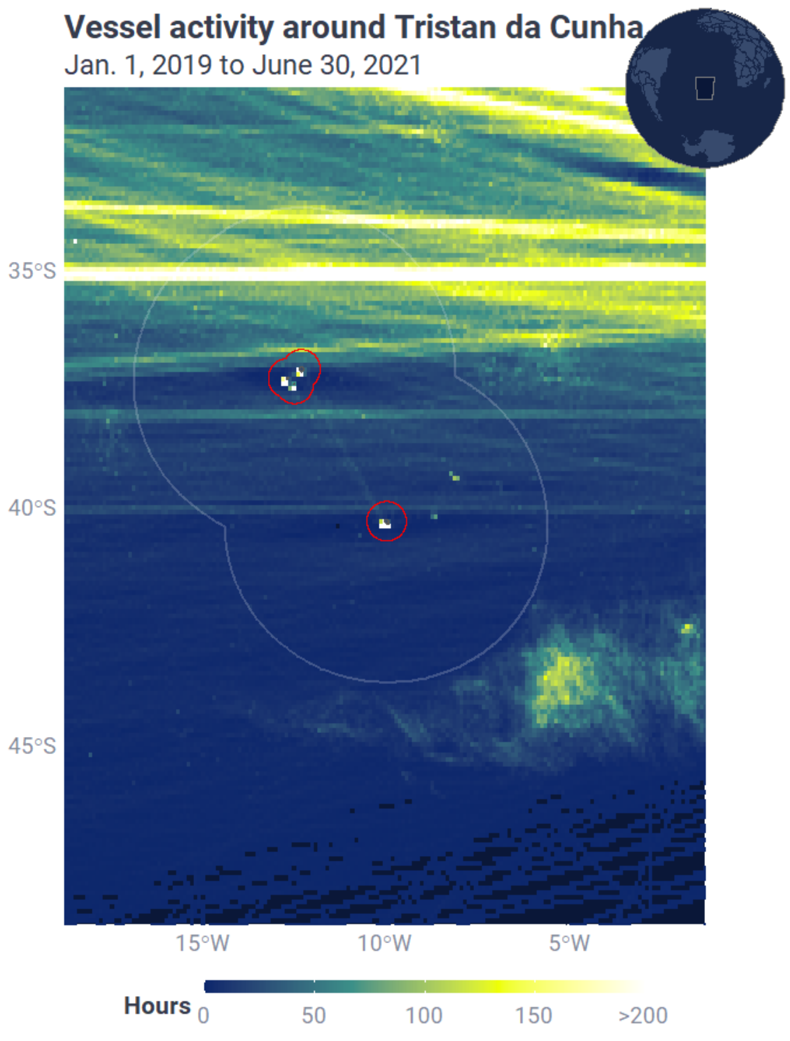
```
vt_gridded <- fishwatchr::gfw_query(query = query_1,
                                    run_query = TRUE,
                                    con = con)$data
```

Alternatively, the queried data can be loaded locally here:

```
vt_gridded <- readr::read_rds("data_production/data/gridded-vessel-activity-2019-
2021.rds")
```

Map of total vessel activity:

```
# set bounding area
bounding_1 <- fishwatchr::transform_box(xlim = c(-18, -2),
                                        ylim = c(-48, -32),
                                        output_crs = "+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs")

# gridded map of vessel activity
m_grid <- vt_gridded %>%
ggplot() +
  geom_raster(aes(x = lon_bin, # this plots the gridded effort data as a raster
                  y = lat_bin,
                  fill = hours)) +
  fishwatchr::geom_gfw_eez(lwd = 1) + # eez boundaries
  geom_sf(data = atba_sf, colour = "red", linetype = 1, fill = NA) + # atba
boundaries
  geom_sf(data = tdc_sf, fill = gfw_palette("map_country_dark")[1]) + # tristan da
cunha archipelago
  labs(title = "Vessel activity around Tristan da Cunha",
       subtitle = "Jan. 1, 2019 to June 30, 2021") +
  scale_fill_gradientn(colours = gfw_palette("map_effort_dark"),
                       limits = c(0,200),
                       oob = scales::squish,
                       na.value = NA,
                       name = "Hours",
                       labels = c("0", "50", "100", "150", ">200")) +
  theme_gfw_map_cian() +
  theme(plot.title = element_text(size = 21),
        plot.subtitle = element_text(size = 18),
        axis.text = element_text(size = 14),
        legend.title = element_text(size = 16),
        legend.text = element_text(size = 14)) +
  coord_sf(xlim = c(bounding_1$box_out[['xmin']], bounding_1$box_out[['xmax']]),
           ylim = c(bounding_1$box_out[['ymin']], bounding_1$box_out[['ymax']]),
           crs = bounding_1$out_crs)

# add a small globe showing the location of the map area
m_grid <- fishwatchr::add_little_globe(main_map = m_grid,
                                       main_box = bounding_1,
                                       globe_rel_size = 0.3,
                                       globe_just = 'center',
                                       globe_position = 'upperright')

m_grid
```


Vessel activity around Tristan da Cunha
Jan. 1, 2019 to June 30, 2021
35°S
40°S
45°S
15°W
10°W
5°W
Hours
0
50
100
150
>200

Transits through the ATBA

Vessel traffic through the ATBA and EEZ

Firstly, we analysed the total traffic, in terms of vessel hours, by vessels greater than 400 gross tons between 1 Jan. 2019 and 30 June 2021.

This next query pulls vessel hours by over 400 gross ton vessels inside the **ATBA** between 01 Jan 2019 and 30 Jun 2021:

```
query_2 <- readr::read_file(str_c("queries", "q_tdc_atba_vessel_hours_atba.sql",
sep="/"))
```

Run the query:

```
vt_atba_summary <- fishwatchr::gfw_query(query = query_2,
                                         run_query = TRUE,
                                         con = con)$data
```

Alternatively, the queried data can be loaded locally here:

```
vt_atba_summary <- readr::read_rds("data_production/data/summary-vessel-traffic-atba-
2019-2021.rds")
```

This next query pulls vessel hours by over 400 gross ton vessels inside the **EEZ** between 01 Jan 2019 and 30 Jun 2021:

```
query_3 <- readr::read_file(str_c("queries", "q_tdc_atba_vessel_hours_eez.sql",
sep="/"))
```

Run the query:

```
vt_eez_summary <- fishwatchr::gfw_query(query = query_3,
                                        run_query = TRUE,
                                        con = con)$data
```

Alternatively, the queried data can be loaded locally here:

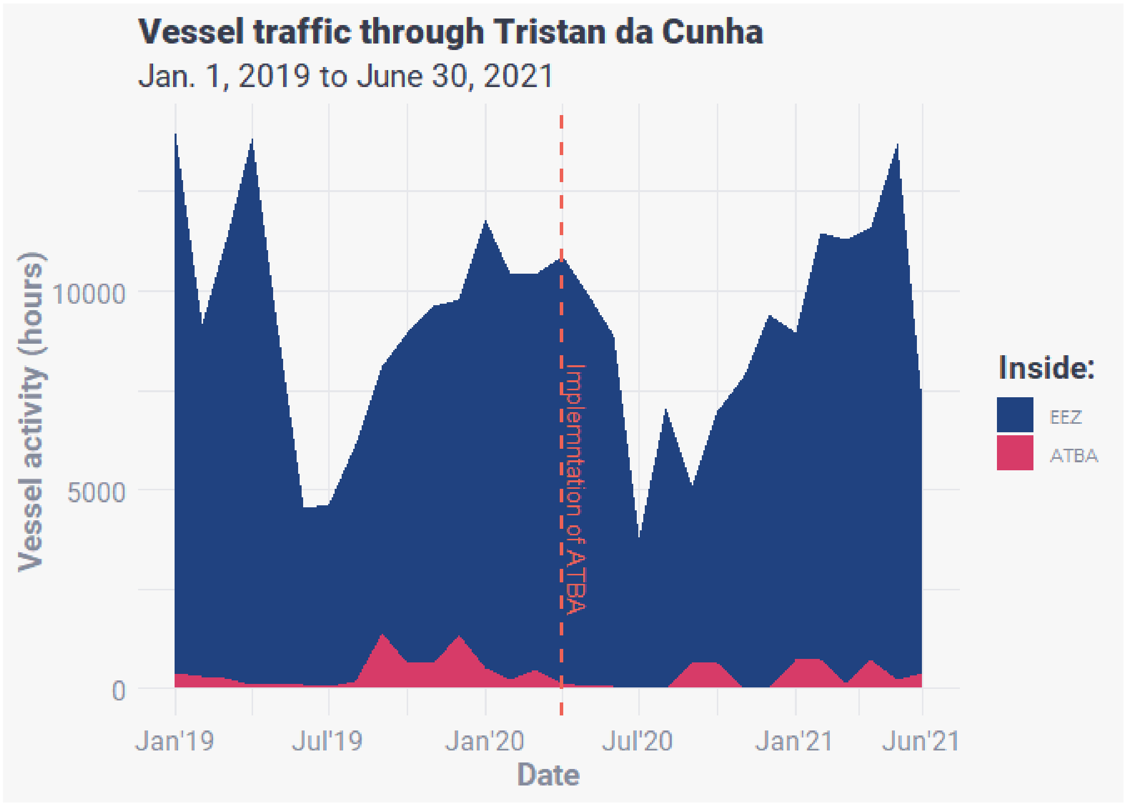
```
vt_eez_summary <- read_rds("data_production/data/summary-vessel-traffic-eez-2019-
2021.rds")
```

Generate monthly summaries of vessel activity per month within the ATBA and EEZ, and combine the two datasets into one:

```
# Summary of vessel hours per month within the ATBA
vt_atba_by_month <- vt_atba_summary %>%
  # this code rounds every date down to the beginning of the month
  mutate(month_bin = floor_date(date, unit = "month")) %>%
  group_by(month_bin) %>%
  summarise(atba = sum(total_hours))

# Summary of vessel hours per month within the EEZ
vt_eez_by_month <- vt_eez_summary %>%
  mutate(month_bin = floor_date(date, unit = "month")) %>%
  group_by(month_bin) %>%
  summarise(eez = sum(total_hours))

# Merge both summaries by month
vt_by_month <- merge(vt_atba_by_month, vt_eez_by_month, by = "month_bin") %>%
  # restructure the data so that we have one column for inside (atba/eez) and another
for total_hours
  pivot_longer(cols = c(atba, eez),
               names_to = "inside",
               values_to = "total_hours")
```

Plot vessel activity per month inside the EEZ and ATBA:

```
vt_by_month %>%
ggplot() +
  geom_area(aes(x = month_bin, y = total_hours,
                # reorder inside so that EEZ is on top
                fill = fct_reorder(inside, total_hours, .desc = TRUE)),
            position = "identity", alpha = 1) +
  scale_fill_manual(name = "Inside:",
                    labels = c("EEZ", "ATBA"),
                    values = gfw_palette("chart")[c(1,4)]) +
  # add and annotate a vertical line at the date the ATBA was implemented
  geom_vline(xintercept = lubridate::date("2020-04-01"), colour =
gfw_palette("chart")[5], size = 1, linetype = 2) +
  annotate("text", x = ymd("2020-04-20"), y = 5000, colour = gfw_palette("chart")[5],
           label = "Implemntation of ATBA", angle = -90, size = 4) +
  labs(title = "Vessel traffic through Tristan da Cunha",
       subtitle = "Jan. 1, 2019 to June 30, 2021",
       x = "Date",
       y = "Vessel activity (hours)") +
  scale_x_date(date_labels = "%b'%y",
               breaks = ymd("2019-01-01", "2019-07-01",
                            "2020-01-01", "2020-07-01",
                            "2021-01-01", "2021-06-01")) +
  theme_gfw_cian() +
  theme(plot.title = element_text(size = 16),
        plot.subtitle = element_text(size = 14),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.title = element_text(size = 14),
        legend.text = element_text(12))
```



Identifying transits

To identify transits through the ATBA we first selected all of the locations by vessels over 400 gross tons within the ATBA. We then identified the `trip_id` associated with each of these positions, and selected every location associated with each `trip_id`.

A `trip_id` is associated with all of the vessel locations between two **port visits**. For the purposes of this analysis, we considered that a port visit had occurred when a vessel:

1. Entered port (vessel within 3km of port)
2. Was stationary (speed less than 0.2 knots) or stopped transmitting its location for more than 4 hours
3. Exited port (vessel more than 4km from port)

This query pulls all positions from port-to-port trips that passed within the ATBA:

```
query_4 <- readr::read_file(str_c("queries",
"q_tdc_atba_trips_through_atba_all_positions.sql", sep="/"))
```

Run the query:

```
vt_atba_tracks <- fishwatchr::gfw_query(query = query_4,
                                        run_query = TRUE,
                                        con = con)$data
```

Alternatively, the queried data can be loaded locally here:

```
vt_atba_tracks <- readr::read_rds("data_production/data/vessl-tracks-over-400t-atba-
only.rds")
```

Reformat the data so that `inside_eez`, `inside_atba`, are formatted correctly and extract `date` from `timestamp`:

```
vt_atba_tracks <- vt_atba_tracks %>%
  mutate(inside_eez = inside_eez %>% as.logical(), # format TRUE/FALSE
         inside_atba = inside_atba %>% as.logical(), # format as TRUE/FALSE
         date = lubridate::date(timestamp)) # extract date
```

Because some vessels may have transited through the ATBA more than once, we needed to create a `transit_id` to differentiate between individual vessel transits. To do this, we wrote a custom function called `create_transit_id()` to create a `transit_id` for each transit, and used `purrr::map()` to iterate through a list of unique vessel ids (`ssvid`). We applied this function to create two `transit_ids`; one for each transit through the EEZ and another for each transit through the ATBA. The steps of the function are:

1. filter the dataframe to the `ssvid` in question
2. arrange by `timestamp`
3. for the first row - assign a `transit_no` as `t` or `NA` depending on whether the point is inside the EEZ/ATBA
4. loop through every subsequent row and:
    a. IF inside atba/eez AND previous position outside: `transit_no <- t`
    b. IF inside AND previous position inside AND previous position within 7 days: `transit_no <- t`
    c. IF inside AND previous position inside AND previous position outside 7 days: `transit_no <- t+1`
    d. IF outside AND previous position inside: `t <- t+1`

The function was defined as:

```
# ssvid: unique vessel identifier
# df: dataframe including vessel positions
# col_name: name of new column to filter by
# output: output the original dataframe with the new column included, or output the
new column as a vector
# time_dif: to avoid assigning multiple transit_ids to a vessel that skirted the edge
of the EEZ/ATBA, we included a minimum time period that had to pass before a vessel
was assigned a new transit_id (default 7 days)

create_transit_id <- function(ssvid, df, col_name = "inside_atba", output = "df",
time_dif = 7){

  # 1. filter df by ssvid and arrange by timestamp
  df_temp <- df %>%
    filter(ssvid == {{ssvid}}) %>%
    arrange(timestamp) %>%
    # create a numericcolumn called transit_no populated with NA
    mutate(transit_no = as.numeric(NA))
```

```r
# 2. set an initial transit_no value
t <- 1

# 3. for the first row, if inside atba set transit no to t
  if(df_temp[[1, as.character(col_name)]] == TRUE){
  df_temp[[1, "transit_no"]] <- t
} else {
  # else do nothing
}

# 4. loop through each row of df_temp, starting with the second row
for(i in 2:nrow(df_temp)){

  # if outside atba and previous position was inside
  # increase t by + 1
  if(df_temp[[i, as.character(col_name)]] == FALSE &
            df_temp[[i - 1, as.character(col_name)]] == TRUE){

       t <- t + 1

  # if inside atba and previous position outside
  # assign transit_no t
  } else if(df_temp[[i, as.character(col_name)]] == TRUE &
            df_temp[[i - 1, as.character(col_name)]] == FALSE){

     df_temp[[i,"transit_no"]] <- t

  # if inside atba and previous position was also inside atba
  # and within 1 week
  # then assign transit_no t
  } else if(df_temp[[i, as.character(col_name)]] == TRUE &
     df_temp[[i - 1, as.character(col_name)]] == TRUE &
     as.numeric(df_temp[[i, "date"]] - df_temp[[i - 1, "date"]]) <= time_dif){

      df_temp[[i,"transit_no"]] <- t

  # if inside atba and previous position was also inside atba
  # but more than 1 week later
  # then increase t + 1, and assign transit_no t
  } else if(df_temp[[i, as.character(col_name)]] == TRUE &
     df_temp[[i - 1, as.character(col_name)]] == TRUE &
     as.numeric(df_temp[[i, "date"]] - df_temp[[i - 1, "date"]]) > time_dif){

      t <- t + 1
      df_temp[[i,"transit_no"]] <- t

  } else {
     # do nothing
  }

}

# append ssvid and transit_no to create transit_id
df_temp <- df_temp %>%
  mutate(transit_id = if_else(!is.na(transit_no), str_c(ssvid,
as.character(transit_no), sep = "_"), as.character(NA)))

  # return full df or vector depending on specified output
  if(output == "df"){
    return(df_temp)
  } else if(output == "vector"){
    return(df_temp$transit_id)
  } else {
    stop("Invalid output. Must be 'df' or 'vector'")
  }

}
```

Next define the list of ssvids to iterate through:

```
ssvid_list <- vt_atba_tracks %>% distinct(ssvid) %>% .$ssvid
names(ssvid_list) <- ssvid_list
```

Then iterate `create_transit_id()` over `ssvid_list` using `purrr::map()`.

```
# first, create a transit_id for each transit through the EEZ
# iterate the function create_transit_id over full ssvid_list using purrr::map()
vt_transits_eez <- purrr::map(.x = ssvid_list,
                        .f = create_transit_id,
                        df = vt_atba_tracks,
                        col_name = "inside_eez",
                        output = "df",
                        time_dif = 7) %>%
  # bind the resulting list of dataframes together
  dplyr::bind_rows() %>%
  # next rename these new columns to include the suffix _eez
  rename(transit_id_eez = transit_id,
         transit_no_eez = transit_no)

# then create a transit_id for each transit through the ATBA
# iterate the function create_transit_id over full ssvid_list using purrr::map()
vt_transits_atba <- purrr::map(.x = ssvid_list,
                        .f = create_transit_id,
                        df = vt_transits,
                        col_name = "inside_atba",
                        output = "df") %>%
  dplyr::bind_rows() %>%
# rename new columns to include the suffix _atba
  rename(transit_id_atba = transit_id,
         transit_no_atba = transit_no)

# cbind transit_no_atba and transit_id_atba to vt_transits_eez
vt_transits <- cbind(vt_transits_eez,
                     vt_transits_atba %>% dplyr::select(transit_no_atba,
transit_id_atba))
```

Alternatively, the processed data can be loaded locally here:

```
vt_transits <- readr::read_rds("data_production/data/vessel-traffic-eez-only-
transit_id_fixed_v4.rds")
```

For each `transit_id_eez` we then determined if the vessel:

1. passed through the ATBA
2. slowed below 0.2 knots while inside the ATBA

```
# which transits passed through the atba
transit_info <- vt_transits %>%
  group_by(transit_id_eez) %>%
  summarise(through_atba = sum(as.numeric(inside_atba)) > 0)

vt_transits <- vt_transits %>%
  merge(transit_info, by = "transit_id_eez", all.x = TRUE)

# which transits slowed while in the atba
transit_speed <- vt_transits %>%
  filter(inside_atba == TRUE) %>%
  group_by(transit_id_eez) %>%
  summarise(slowed_in_atba = min(speed_knots) <= 0.2)

vt_transits <- vt_transits %>%
  merge(transit_speed, by = "transit_id_eez", all.x = TRUE)
```

Which vessels made the most transits? First we summarised the number of transits made through the ATBA, per vessel, per year. Including year allows us to merge these data with the identity information associated with that vessel id that year.

```
vessels_by_n_transits <- vt_transits %>%
  filter(date >= ymd("2020-04-01"), # only inlucde transits after the ATBA was
implemented
         slowed_in_atba == FALSE, # that did not slow below 0.2 knots inside the ATBA
         through_atba == TRUE) %>% # from transits that passed through the atba
  group_by(ssvid, year) %>%
  summarise(n_transits = n_distinct(transit_id_atba, na.rm = TRUE),
            .groups = "keep") # count the number of transit_id_atba values

vessels_by_n_transits

## # A tibble: 133 x 3
## # Groups:   ssvid, year [133]
##    ssvid      year n_transits
##    <chr>     <int>      <int>
##  1 111111234  2021          1
##  2 200006639  2021          1
##  3 209447000  2020          1
##  4 209682000  2020          1
##  5 209854000  2020          1
##  6 210043000  2020          1
##  7 210655000  2020          1
##  8 212460000  2021          1
##  9 227498430  2021          1
## 10 227837860  2020          1
## # ... with 123 more rows
```

These data were uploaded to BigQuery as `world-fishing-827.scratch_cian.atba_ssvids`. The following query merges the data in `vessels_by_n_transits` with the best information available on vessel identity including the most commonly transmitted vessel name, flag, vessel class, and gross tonnage:

```
query_5 <- readr::read_file(str_c("queries", "q_tdc_atba_transits_vessel_info.sql",
sep="/"))
```

Run query:

```
vessels_by_n_transits <- fishwatchr::gfw_query(query = query_5,
                                                run_query = TRUE,
                                                con = con)$data
```

Alternatively, the queried data can be loaded locally here:

```
vessels_by_n_transits <-
readr::read_rds("data_production/data/vessels_by_n_transits_v2.rds")
```

After inspecting the data, there appeared to be a number of sailing vessels that passed through the ATBA as part of the Vendée Globe round-the-world race. The algorithm used to predict gross tonnage of vessels generally performs well, with a predictive score of 0.77 (see Kroodsma et al. 2018), but the training data didn't include long-haul sailing vessels such as these, and subsequently the algorithm may not perform as well for these vessels. Additionally, the risk of these vessels running aground and causing a serious environmental incident, such as that caused by the MS Oliva, is low. Therefore we excluded these vessels from further analysis:

First we created a list of `ssvids` which most commonly self-reported as `Sailing`, and then filtered out these `ssvids` from the transit data:

```
sailing_vessels <- vessels_by_n_transits %>%
  unnest(shiptype) %>%
  group_by(ssvid) %>%
  top_n(1, wt = count) %>%
  filter(value == "Sailing")
```

Looking at these vessels we could see that all of these vessels, with the exception of the `LU HUANG YUAN YU 107` was flagged to FRA. The `LU HUANG YUAN YU 107` is definitely not a sailing vessel, so the self-reported value in this case was not accurate.

If we instead filtered for only French-flagged vessels we got a more conservative list of sailing vessels, and so this was the exclusion criteria we used:

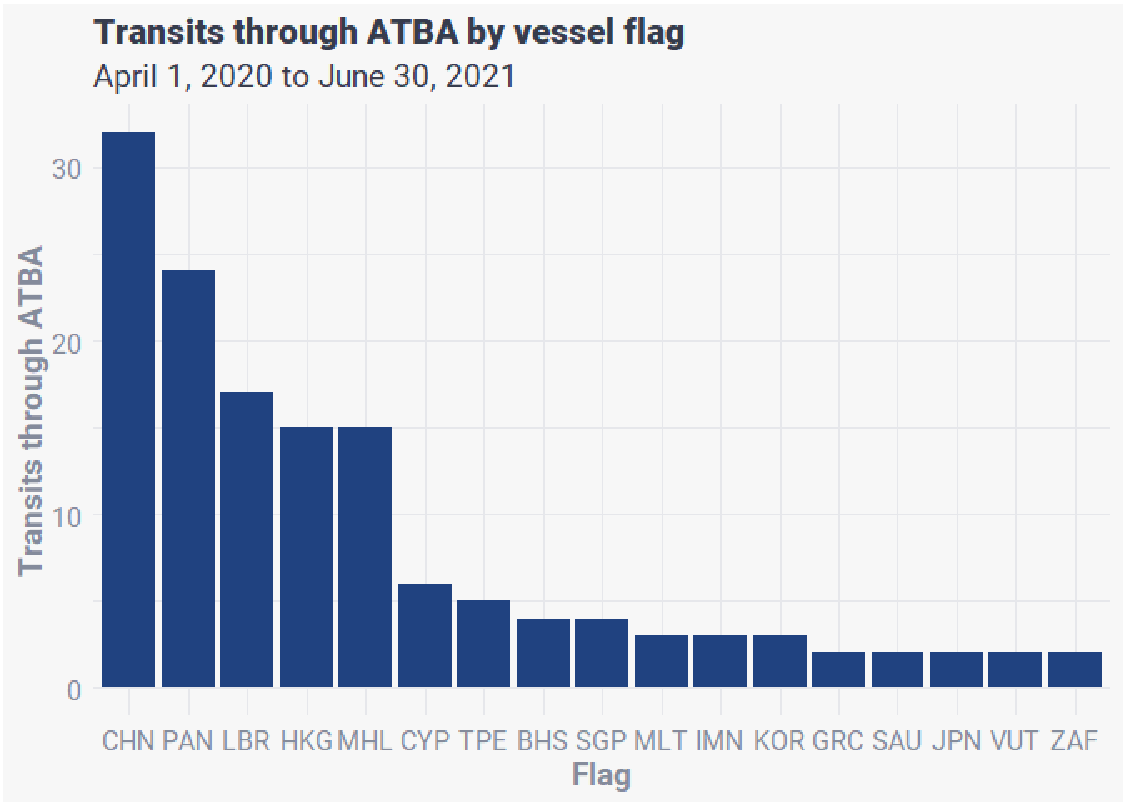
```
sailing_vessels <- vessels_by_n_transits %>%
  unnest(shiptype) %>%
  group_by(ssvid) %>%
  top_n(1, wt = count) %>%
  filter(best_flag == "FRA")
```

So with that in mind, how many transits occurred before and after the implementation of the ATBA? To answer this we counted the number of distinct `transit_id` values:

```
# all transits between 01 Jan 2019 - 30 Jun 2020
vt_transits %>%
  filter(slowed_in_atba == FALSE,
         through_atba == TRUE,
         # exclude sailing vessels
         !ssvid %in% sailing_vessels$ssvid) %>%
  mutate(after_atba = date >= ymd("2020-04-01"),
         month = date %>% floor_date("months")) %>%
  group_by(after_atba) %>%
  summarise(n_transits = n_distinct(transit_id_atba, na.rm = TRUE),
            n_vessels = n_distinct(ssvid, na.rm = TRUE))

## # A tibble: 2 x 3
##   after_atba n_transits n_vessels
##   <lgl>           <int>     <int>
## 1 FALSE             454       416
## 2 TRUE              125       123

# all transits limited to 12 months before/after the implementation of the ATBA
# (01 Apr 2019 - 31 Mar 2020)
vt_transits %>%
  filter(slowed_in_atba == FALSE,
         through_atba == TRUE,
         date %>% between(ymd("2019-04-01"), ymd("2021-03-31")),
         # exclude sailing vessels
         !ssvid %in% sailing_vessels$ssvid) %>%
  mutate(after_atba = date >= ymd("2020-04-01"),
         month = date %>% floor_date("months")) %>%
  group_by(after_atba) %>%
  summarise(n_transits = n_distinct(transit_id_atba, na.rm = TRUE),
            n_vessels = n_distinct(ssvid, na.rm = TRUE))

## # A tibble: 2 x 3
##   after_atba n_transits n_vessels
##   <lgl>           <int>     <int>
## 1 FALSE             327       304
## 2 TRUE               96        96
```

Plot the number of transits per vessel flag:

```
# summarise the number of transits per flag
flag_transit_summary <- vt_transits %>%
  filter(date >= ymd("2020-04-01"),
         through_atba == TRUE,
         slowed_in_atba == FALSE,
         # exclude sailing vessels
         !ssvid %in% sailing_vessels$ssvid) %>%
  mutate(best_flag = best_flag %>% recode_factor("TWN" = "TPE")) %>%
  group_by(best_flag) %>%
  summarise(n_transits = n_distinct(transit_id_atba, na.rm = TRUE)) %>%
  # important for plotting - arrange in descending order by n_transits
  arrange(desc(n_transits))

# plot the number of transits per flag
vt_transits %>%
  filter(date >= ymd("2020-04-01"),
         through_atba == TRUE,
        slowed_in_atba == FALSE,
        !is.na(best_flag),
        # exclude sailing vessels
        !ssvid %in% sailing_vessels$ssvid) %>%
  group_by(best_flag) %>%
  summarise(n_transits = n_distinct(transit_id_atba)) %>%
  # reorder the factor levels of best_flag according to the number of transits per
flag
  # useful for making a pretty plot
  mutate(best_flag = best_flag %>%
           recode_factor("TWN" = "TPE") %>%
           factor(levels = flag_transit_summary$best_flag)) %>%
  ggplot() +
    geom_col(aes(x = best_flag, y = n_transits), fill = gfw_palette("chart")[1]) +
    labs(x = "Flag", y = "Transits through ATBA",
        title = "Transits through ATBA by vessel flag",
        subtitle = "April 1, 2020 to June 30, 2021") +
    theme_gfw_cian() +
    theme(plot.title = element_text(size = 16),
         plot.subtitle = element_text(size = 14),
         axis.text = element_text(size = 12),
         axis.title = element_text(size = 14),
         legend.title = element_text(size = 14),
         legend.text = element_text(size = 12))
```

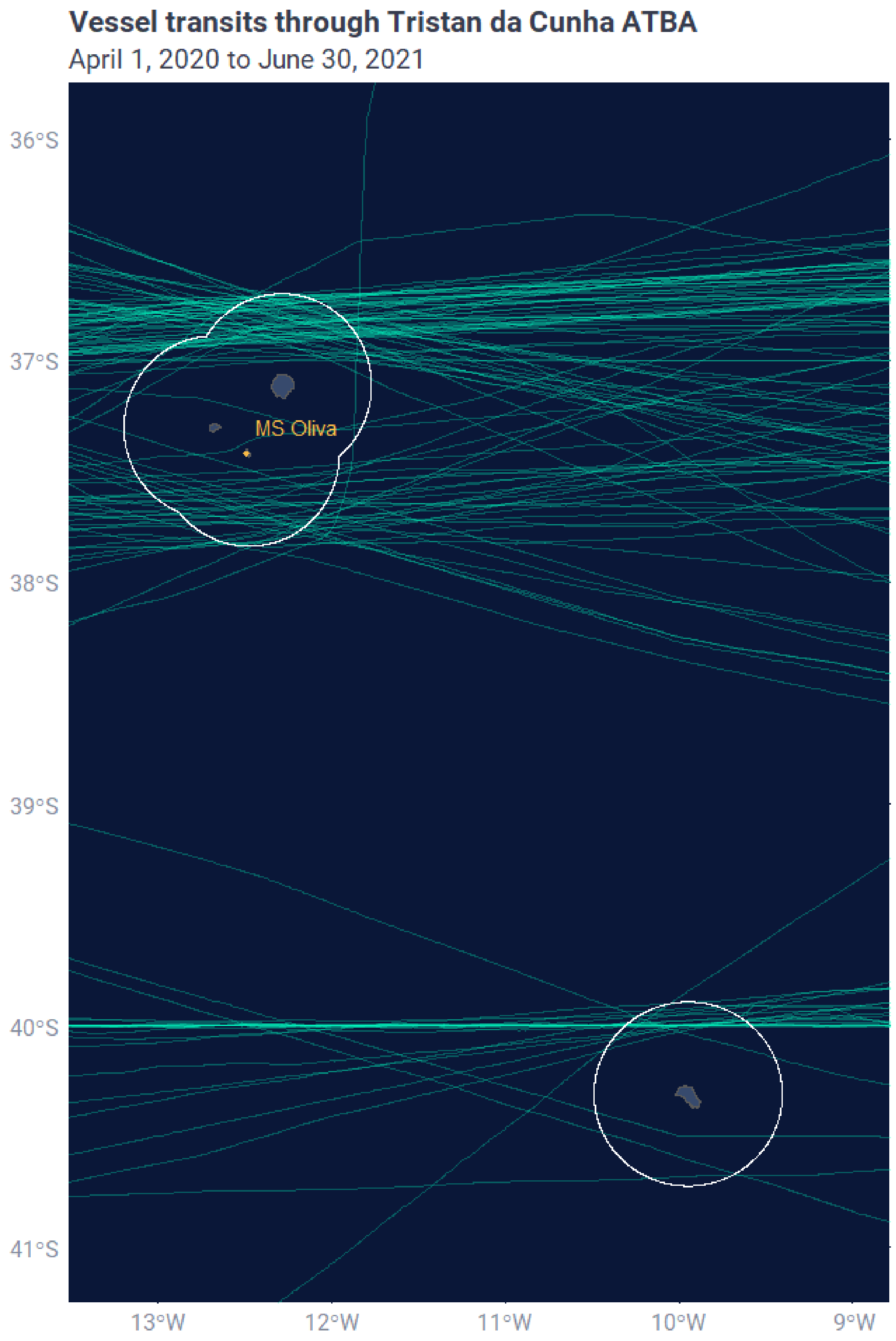


Number of transits by vessel class:

```
# summarise the number of transits per flag
vt_transits %>%
  filter(date >= ymd("2020-04-01"),
         through_atba == TRUE,
         slowed_in_atba == FALSE,
         # exclude sailing vessels
         !ssvid %in% sailing_vessels$ssvid) %>%
  group_by(best_vessel_class) %>%
  summarise(n_transits = n_distinct(transit_id_atba, na.rm = TRUE)) %>%
  arrange(desc(n_transits))

## # A tibble: 10 x 2
##    best_vessel_class  n_transits
##    <chr>                   <int>
##  1 cargo                      90
##  2 squid_jigger               20
##  3 specialized_reefer          4
##  4 tanker                      3
##  5 cargo_or_tanker             2
##  6 drifting_longlines          2
##  7 fishing                     1
##  8 gear                        1
##  9 non_fishing                 1
## 10 set_longlines               1
```

Map the transits through the ATBA:

```
# create a dataframe of the coordinates of where the MS Oliva ran aground
ms_oliva <- data.frame(name = "MV Oliva",
                      lon = -12.493,
                      lat = -37.419)

# This code sets a bounding box for the map area
bounding_atba <- fishwatchr::transform_box(xlim = c(-13.3, -9),
                                          ylim = c(-41, -36),
                                          output_crs = "+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs")

# use ggplot2 to create a map of transits
vt_transits %>%
  # filter data to only include transits after the ATBA was established
  filter(date >= ymd("2020-04-01"),
         through_atba == TRUE,
         slowed_in_atba == FALSE,
         # exclude sailing vessels
         !ssvid %in% sailing_vessels$ssvid) %>%
  group_by(as.factor(transit_id_eez)) %>%
  arrange(timestamp) %>%
  ggplot() +
  geom_path(aes(x = lon, y = lat, group = transit_id_eez), colour =
gfw_palette("map_presence_dark")[3], alpha = 0.3) +
  geom_sf(data = atba_sf, colour = "white", linetype = 1, fill = NA) +
  geom_sf(data = tdc_sf, fill = gfw_palette("map_country_dark")[1]) +
  geom_point(data = ms_oliva, aes(x = lon, y = lat), shape = 18, colour =
gfw_palette("sand")[1]) +
  annotate("text", x = -12.2, y = -37.3, colour = gfw_palette("sand")[1], label = "MS
Oliva") +
  labs(title = "Vessel transits through Tristan da Cunha ATBA",
       subtitle = "April 1, 2020 to June 30, 2021") +
  theme_gfw_map_cian() +
  theme(plot.title = element_text(size = 16),
        plot.subtitle = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.title = element_text(size = 14),
        legend.text = element_text(size = 12),
        legend.position = c(0.7, 0.95)) +
  coord_sf(xlim = c(bounding_atba$box_out[['xmin']],
bounding_atba$box_out[['xmax']]),
           ylim = c(bounding_atba$box_out[['ymin']],
bounding_atba$box_out[['ymax']]),
           crs = bounding_atba$out_crs)
```


Vessel transits through Tristan da Cunha ATBA
April 1, 2020 to June 30, 2021
MS Oliva
36°S
37°S
38°S
39°S
40°S
41°S
13°W
12°W
11°W
10°W
9°W

For additional context, we plotted the tracks of all squid jiggers that actually transited through the ATBA. First, we made a list of all squid jiggers that transited through the ATBA.

```r
ssvid_squid_2 <- vt_transits %>%
  filter(slowed_in_atba == FALSE,
         through_atba == TRUE,
         best_vessel_class == "squid_jigger") %>%
  distinct(ssvid) %>%
  pull(ssvid)
```

We then ran a modified version of the original query to select all of the port-to-port positions from the squid jigger vessels in `vt_transits`.

```r
query_6 <- readr::read_file(str_c("queries",
"q_tdc_atba_squid_jigger_transits_fishing.sql", sep="/"))
```

Run the query:

```r
squid_tracks_2 <- fishwatchr::gfw_query(query = query_6,
                                        run_query = TRUE,
                                        con = con)$data
```

Filter to only keep trips that transit through the ATBA:

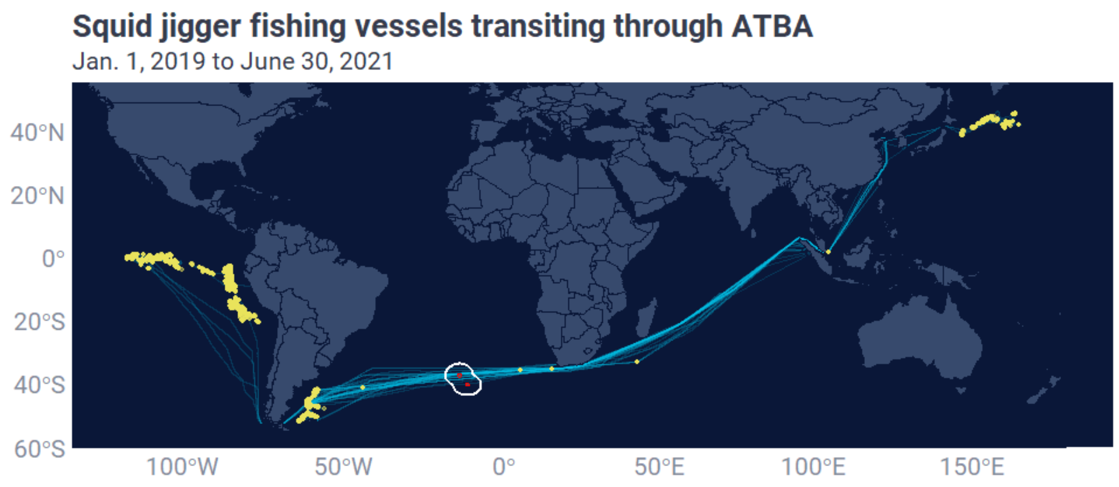
```r
squid_tracks_2 <- squid_tracks_2 %>%
  group_by(trip_id) %>%
  summarise(through_atba = sum(inside_atba) > 0) %>%
  merge(squid_tracks_2, by = "trip_id", all.y = TRUE) %>%
  filter(through_atba == TRUE)
```

Alternatively, the queried data can be loaded locally here:

```r
squid_tracks_2 <-
readr::read_rds("data_production/data/squid_jigger_tracks_full.rds")
```

Create a map of these tracks including locations of fishing activity:

```r
# filter out trips that cross 180 longitude
# these are difficult to map
trip_sum <- squid_tracks_2 %>%
  group_by(trip_id) %>%
  summarise(keep = min(lon) > -120) %>%
  filter(keep == TRUE)

# set the bounding area for the map
bounding_3 <- fishwatchr::transform_box(xlim = c(-120, 180),
                                        ylim = c(-55, 50),
                                        output_crs = "+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs")

# use ggplot to create the map
squid_tracks_2 %>%
  filter(trip_id %in% trip_sum$trip_id) %>%
  arrange(timestamp) %>%
  ggplot() +
    geom_path(aes(x = lon, y = lat, group = trip_id), colour =
gfw_palette("tracks")[1], alpha = 0.2) +
    geom_sf(data = eez_tdc, fill = NA, colour = "white", size = 1) +
    fishwatchr::geom_gfw_land() +
    geom_sf(data = atba_sf, colour = "red", linetype = 1, fill = NA) +
    geom_sf(data = tdc_sf, fill = gfw_palette("map_country_dark")[1]) +
    geom_point(data = squid_tracks_2 %>% filter(fishing_hours > 0),
               aes(x = lon, y = lat),
               colour = gfw_palette("yellow")[1], size = 1, alpha = 0.2) +
    labs(title = "Squid jigger fishing vessels transiting through ATBA",
         subtitle = "Jan. 1, 2019 to June 30, 2021") +
    theme_gfw_map_cian() +
    theme(plot.title = element_text(size = 20),
          plot.subtitle = element_text(size = 16),
          axis.text = element_text(size = 16),
          legend.title = element_text(size = 20),
          legend.text = element_text(size = 16)) +
    coord_sf(xlim = c(bounding_3$box_out[['xmin']], bounding_3$box_out[['xmax']]),
             ylim = c(bounding_3$box_out[['ymin']], bounding_3$box_out[['ymax']]),
             crs = bounding_3$out_crs)
```



Squid jigger fishing vessels transiting through ATBA
Jan. 1, 2019 to June 30, 2021
40°N
20°N
0°
20°S
40°S
60°S
100°W
50°W
0°
50°E
100°E
150°E

Distance to shore

Here we used the `sf` (simple features) package to work out the distance to shore for every position within the ATBA. To save computing time we excluded all points outside of the ATBA. We created a `row_id` in `vt_transits` to merge results.

Create an `sf` object of points within the ATBA, and only include necessary columns to save computing time:

```r
# create a row_id col in vt_transits for merging
vt_transits <- vt_transits %>%
  mutate(row_id = seq(1, nrow(vt_transits), length.out = nrow(vt_transits)))

# filter to just positions inside the atba
vt_atba_sf <- vt_transits %>%
  filter(inside_atba == TRUE) %>%
  dplyr::select(row_id, lat, lon) %>%
  st_as_sf(coords = c('lon', 'lat'), crs = 4326)
```

`sf::st_ditance(vt_atba_sf, tdc_sf)` calculates the Euclidean distance (in metres) between each point of `vt_atba_sf` and each feature of `tdc_sf` (shapefile of the Tristan da Cunha arhcipelago), returning a distance matrix. `apply(1, min)` then iterates through each row of this matrix and selects the minimum distance for each row - i.e. the minimum distance between each point of `vt_atba_sf` and the closest feature of `tdc_sf`.

```r
vt_atba_sf$dist_to_shore_m <- st_distance(vt_atba_sf, tdc_sf) %>%
  apply(1, min)
```

Merged back with `vt_transits`:

```r
vt_transits <- vt_transits %>%
  merge(vt_atba_sf %>% st_drop_geometry(),
        by = "row_id", all.x = TRUE)
```

Converted distance from metres to kilometres and round to 2 decimal places:

```r
vt_transits <- vt_transits %>%
  mutate(dist_to_shore_km = round(dist_to_shore_m/1000, 2))
```

Updated the list of vessels to include min distance to shore. First we calculated the min distance to shore for each transit, and then merged the results with `vt_transits` using `transit_id_atba`:

```r
# minimum distance to shore for every transit passing through the atba
transit_dist <- vt_transits %>%
  # slowed_in_atba == FALSE key to eliminate non transits
  filter(date >= ymd("2020-04-01"),
         !is.na(transit_id_atba),
         inside_atba == TRUE,
         slowed_in_atba == FALSE) %>%
  group_by(transit_id_atba) %>%
  summarise(min_dist_to_shore_km = min(dist_to_shore_km, na.rm = TRUE))

# merge distances with vt_transits
vt_transits <- vt_transits %>%
  merge(transit_dist, by = "transit_id_atba", all.x = TRUE)
```

Updated the list of transiting vessels:

```r
vessels_by_dist <- vt_transits %>%
  filter(date >= ymd("2020-04-01"),
         !is.na(transit_id_atba),
         through_atba == TRUE,
         slowed_in_atba == FALSE) %>%
  group_by(ssvid) %>%
  summarise(min_dist_to_shore_km = min(min_dist_to_shore_km, na.rm = TRUE)) %>%
  merge(vessels_by_n_transits) %>%
  arrange(min_dist_to_shore_km, best_flag)

# create a version that excludes sailing vessels
vessels_by_dist_no_sailing <- vessels_by_dist %>% filter(!ssvid %in%
sailing_vessels$ssvid)
```

Are vessels avoiding the ATBA?

What proportion of transits that passed through the EEZ also passed through the ATBA? Does this proportion change during the 12 months before and after the implementation of the ATBA?

```
# before ATBA
vt_transits %>%
  filter(date %>% between(ymd("2019-04-01"), ymd("2020-03-31")),
         # exclude sailing vessels
        !ssvid %in% sailing_vessels$ssvid) %>%
  group_by(transit_id_eez) %>%
  summarise(n_transits = n_distinct(transit_id_eez, na.rm = TRUE),
            through_atba = n_distinct(transit_id_atba, na.rm = TRUE) > 0) %>%
  summarise(n_transits_eez = sum(n_transits),
            n_transits_atba = sum(through_atba),
            prop_through_atba = n_transits_atba/n_transits_eez)

## # A tibble: 1 x 3
##   n_transits_eez n_transits_atba prop_through_atba
##            <int>           <int>             <dbl>
## 1            344             340             0.988

# after ATBA
vt_transits %>%
  filter(date %>% between(ymd("2020-04-01"), ymd("2021-03-31")),
         # exclude sailing vessels
        !ssvid %in% sailing_vessels$ssvid) %>%
  group_by(transit_id_eez) %>%
  summarise(n_transits = n_distinct(transit_id_eez, na.rm = TRUE),
            through_atba = n_distinct(transit_id_atba, na.rm = TRUE) > 0) %>%
  summarise(n_transits_eez = sum(n_transits),
            n_transits_atba = sum(through_atba),
            prop_through_atba = n_transits_atba/n_transits_eez)

## # A tibble: 1 x 3
##   n_transits_eez n_transits_atba prop_through_atba
##            <int>           <int>             <dbl>
## 1            112             103             0.920
```

Did vessel traffic (hours of vessel activity) through the EEZ and ATBA change after the ATBA was implemented?

```
# eez
vt_transits %>%
  filter(inside_eez == TRUE,
         !ssvid %in% sailing_vessels$ssvid,
         date %>% between(ymd("2019-04-01"), ymd("2021-03-31"))) %>%
  mutate(after_atba = date >= ymd("2020-04-01")) %>%
  group_by(after_atba) %>%
  summarise(total_vessel_hours = sum(hours, na.rm = TRUE))

## # A tibble: 2 x 2
##   after_atba total_vessel_hours
##   <lgl>                   <dbl>
## 1 FALSE                  17924.
## 2 TRUE                    7205.

# atba
vt_transits %>%
  filter(inside_atba == TRUE,
         !ssvid %in% sailing_vessels$ssvid,
         date %>% between(ymd("2019-04-01"), ymd("2021-03-31"))) %>%
  mutate(after_atba = date >= ymd("2020-04-01")) %>%
  group_by(after_atba) %>%
  summarise(total_vessel_hours = sum(hours, na.rm = TRUE))

## # A tibble: 2 x 2
##   after_atba total_vessel_hours
##   <lgl>                   <dbl>
## 1 FALSE                   5754.
## 2 TRUE                    3107.
```

Port to port tracking

Using the same methodology, as described above, to identify likely port visits, we wanted to identify the port that each vessel was travelling from and to when it transited through the ATBA. To do this, we used the `trip_id` associated with every trip that included a transit through the ATBA, to look up the name of the ports visited before and and after each transit.

First, we created a list of the `trip_ids` for transits passing through the ATBA, post-implementation:

```
transits_post_atba <- vt_transits %>%
  filter(slowed_in_atba == FALSE,
         through_atba == TRUE,
         date >= ymd("2020-04-01"),
         !ssvid %in% sailing_vessels$ssvid) %>%
  distinct(trip_id) %>%
  pull(trip_id)

transits_post_atba[1:10]

##  [1] "111111234-017452bc9608" "200006639-01741ea66b28" "209447000-0175b77f1f38"
##  [4] "209682000-017145ed8fe8" "209854000-0174e92587a8" "210043000-017385d2d670"
##  [7] "210655000-0175ee56e2e8" "212460000-01776ca0a7a0" "229609000-0178229fcf90"
## [10] "229623000-0171b32245e0"
```

Then we defined a query to identify the names and locations of ports visited before and after transiting through the ATBA:

```
query_7 <- readr::read_file(str_c("queries", "q_tdc_atba_ports_visited.sql",
sep="/"))
```

Run query:

```
transit_ports <- fishwatchr::gfw_query(query = query_7,
                                       run_query = TRUE,
                                       con = con)$data
```

Alternatively, the queried data can be loaded locally here:

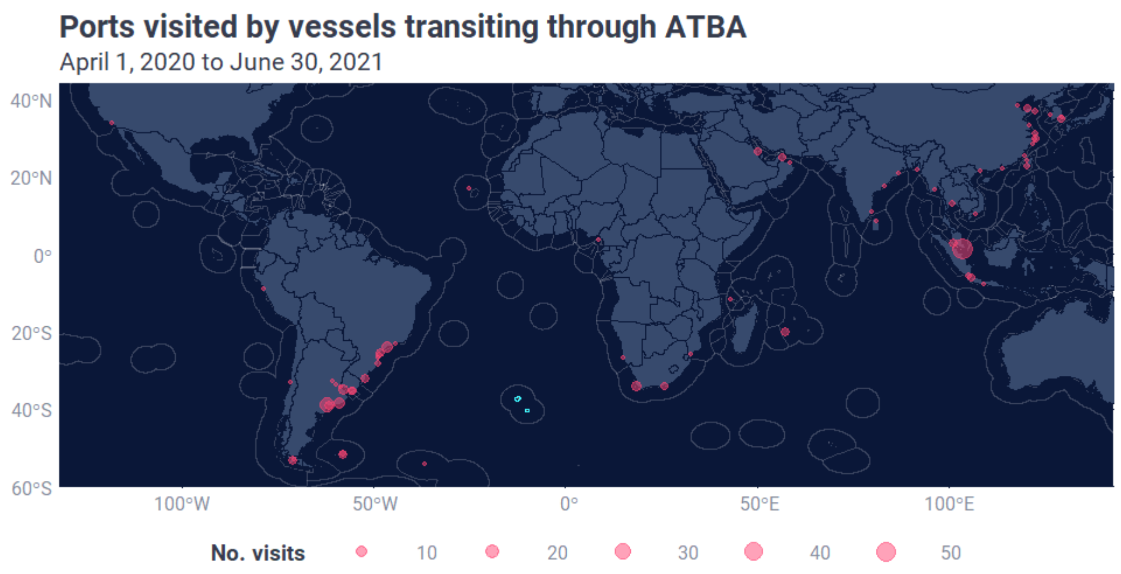
```
transit_ports <- readr::read_rds("data_production/data/transit_ports_c4.rds")
```

To make a map of origin and destination ports we needed to reformat the `transit_ports` dataframe into a long version, with port label, lon, lat, and number of voyages starting and ending there. Some ports had multiple anchorage points associated with them. In these cases we grouped the anchorages by `port_label` and averaged the lon and lat coordinates.

```
# create long version of dataframe
# filter to only include ports of origin
# create a col called cat (short for category) and populate with "origin"
ports_origin <- transit_ports %>%
  filter(!is.na(trip_start_anchorage_label)) %>%
  mutate(port_label = str_c(trip_start_anchorage_label, trip_start_anchorage_country,
sep = ", ")) %>%
  group_by(port_label) %>%
  summarise(n_voyages = n(),
            lon = mean(trip_start_anchorage_lon, na.rm = TRUE),
            lat = mean(trip_start_anchorage_lat, na.rm = TRUE)) %>%
  mutate(cat = "origin")

# note there are 6 fewer destinations as 6 trips were still active at the end of our
time range
# and hadn't yet reached a port

# filter to only include ports of origin
# create a col called cat (short for category) and populate with "origin"
ports_destination <- transit_ports %>%
  filter(!is.na(trip_end_anchorage_label)) %>%
  mutate(port_label = str_c(trip_end_anchorage_label, trip_end_anchorage_country, sep
= ", ")) %>%
  group_by(port_label) %>%
  summarise(n_voyages = n(),
            lon = mean(trip_end_anchorage_lon, na.rm = TRUE),
            lat = mean(trip_end_anchorage_lat, na.rm = TRUE)) %>%
  mutate(cat = "destination")

# bind the two dataframes together
ports_long <- rbind(ports_origin, ports_destination)
```

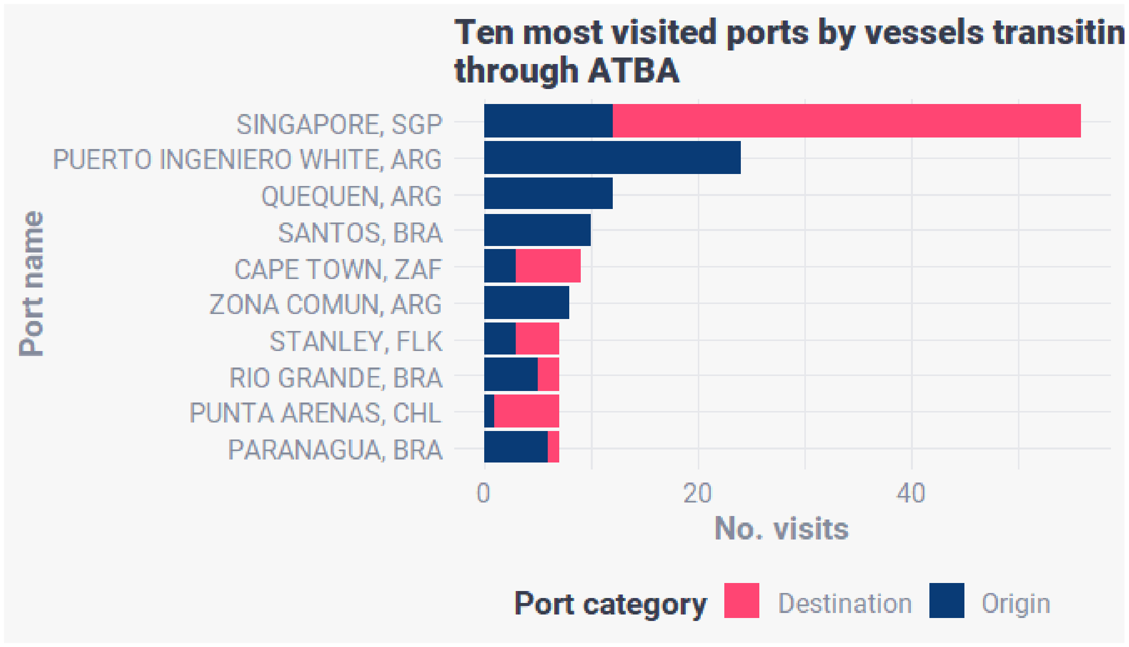
Here we created a map with points showing the locations of each port and the points sized in proportion to the number of transits beginning or ending there:

```
# set the bounding are for the map
bounding_ports <- fishwatchr::transform_box(xlim = c(min(ports_long$lon) - 1,
                                                      max(ports_long$lon) + 1),
                                            ylim = c(min(ports_long$lat) - 1,
                                                      max(ports_long$lat) + 1),
                                            output_crs = "+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs")

# create the map using ggplot2
ports_long %>%
  # here we wanted to summarise the number of pots starting OR ending in each port
  # so we used group_by(port_label), excluding the origin/destination category
  group_by(port_label) %>%
  summarise(n_voyages = sum(n_voyages), .groups = "keep",
            lon = mean(lon),
            lat = mean(lat)) %>%
    ggplot() +
  fishwatchr::geom_gfw_eez(theme = 'dark') +
  fishwatchr::geom_gfw_land(theme = 'dark') +
  geom_point(aes(x = lon, y = lat, size = n_voyages), alpha = 0.5, colour =
gfw_palette("map_reception_light")[4]) +
  scale_size_continuous(name = "No. visits") +
  geom_sf(data = atba_sf, colour = gfw_palette("diverging")[1], linetype = 1, fill =
NA) +
  geom_sf(data = tdc_sf, fill = gfw_palette("map_country_dark")[1]) +
  labs(title = "Ports visited by vessels transiting through ATBA",
       subtitle = "April 1, 2020 to June 30, 2021") +
  theme_gfw_map_cian(theme = 'dark') +
  theme(legend.position = "bottom",
        plot.title = element_text(size = 20),
        plot.subtitle = element_text(size = 16),
        axis.text = element_text(size = 12),
        legend.title = element_text(size = 14),
        legend.text = element_text(size = 12)) +
  coord_sf(xlim = c(bounding_ports$box_out[['xmin']],
bounding_ports$box_out[['xmax']]),
           ylim = c(bounding_ports$box_out[['ymin']],
bounding_ports$box_out[['ymax']]),
           crs = bounding_ports$out_crs)
```


Ports visited by vessels transiting through ATBA
April 1, 2020 to June 30, 2021
40°N
20°N
0°
20°S
40°S
60°S
100°W
50°W
0°
50°E
100°E
No. visits
10
20
30
40
50

Next we plotted the number of transits that started or ended in the 10 ports most frequently visited by transiting vessels:

```
# create a summary table of the top 10 most visited ports
# useful for plotting
ports_sum_top10 <- ports_long %>%
  group_by(port_label) %>%
  summarise(n_voyages = sum(n_voyages)) %>%
  top_n(10, n_voyages) %>%
  arrange(n_voyages)

# create a horizontal barplot coloured by the number of transits that started or
ended in each port
ports_long %>%
  filter(port_label %in% ports_sum_top10$port_label) %>%
  ggplot(aes(y = port_label %>% factor(levels = ports_sum_top10 %>%
pull(port_label)),
             x = n_voyages, fill = cat)) +
  geom_col() +
  scale_fill_manual(values = gfw_palette('map_reception_light', type =
"discrete")[c(4,2)],
                    name = "Port category",
                    labels = c("Destination", "Origin")) +
  labs(x = "No. visits",
       y = "Port name",
       title = "Ten most visited ports by vessels transiting\nthrough ATBA") +
  theme_gfw_cian() +
  theme(axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        plot.title = element_text(size = 16),
        legend.title = element_text(size = 14),
        legend.text = element_text(size = 12),
        legend.position = "bottom")
```



We created a wide table of transits ending and beginning in each port to be included as an appendix in the final report.

```
ports_wide <- ports_long %>%
  pivot_wider(names_from = cat,
              values_from = n_voyages,
              values_fill = 0) %>%
  separate(port_label, c('port','country'), sep = ", ", remove = FALSE) %>%
  arrange(country) %>%
  dplyr::select(-port, -country)
```

Gaps in AIS transmission

To understand how gaps in AIS transmission may affect our ability to detect transiting vessels, we used a dataset of suspected AIS disabling events. The methodology is currently under peer review, but in short, this dataset applies a classification model to gap events longer than 12 hours to identify potential disabling of AIS devices. This model accounts for the quality of satellite reception, the location and duration of a gap event, and the rate of AIS transmission before and after the gap occurred.

This query pulls all AIS gap events longer than 12 hours near Tristan da Cunha (between -25° and 5° longitude, and -25° and -50° latitude) between Jan. 2019 and June 2021. The query includes only gap events longer than 12 hours as satellite reception can vary a lot in shorter time periods. We used a minimum distance to shore of 10 nautical miles to exclude vessels stationary vessels at Tristan. We only kept AIS gap events where the gap started or ended in a location from where 5 or more locations had been transmitted to exclude gaps occurring in low reception areas. Finally, based on previous analysis, we considered gap events where the vessel had transmitted 19 or more AIS positions in the 12 hours previous to a 12 hour gap to represent potential disabling of AIS devices.

```
query_8 <- readr::read_file(str_c("queries", "q_tdc_atba_gap_events.sql", sep="/"))
```

Run the query:

```
gap_events <- fishwatchr::gfw_query(query = query_8,
                                    run_query = TRUE,
                                    con = con)$data
```

Alternatively, the queried data can be loaded locally here:

```
gap_events <- readr::read_rds("data_production/data/ais_gaps_intentional.rds")
```

We reformatted the data into a structure including a column for:

- `gap_id`: the identity code assigned to each gap event
- `ssvid`: unique vessel identifier
- `gap_hours`: duration of the gap event in hours
- `gap_start`: date and timestamp of when the gap event begun (i.e. when the AIS device was turned off)
- `gap_end`: date and timestamp of when the gap event ended (i.e. when the AIS device was turned back on)
- `vessel_tonnage_gt`: gross tonnage of the vessel
- `vessel_class`: best known vessel class - in this dataset all vessels are fishing vessel classes
- `gap_event`: `off` (AIS turned off) or on (AIS device turned on)
- `lon`: longitude in decimal degrees
- `lat`: latitude in decimal degrees

Note, that with the exception of `gap_event`, `lon`, and `lat`, the data for each `gap_id` are duplicated, with one row for the on and one row for the `off` position. This is for mapping purposes, and to avoid double counting gaps, we use `n_distinct(gap_id)` to count the number of distinct values of `gap_id`.

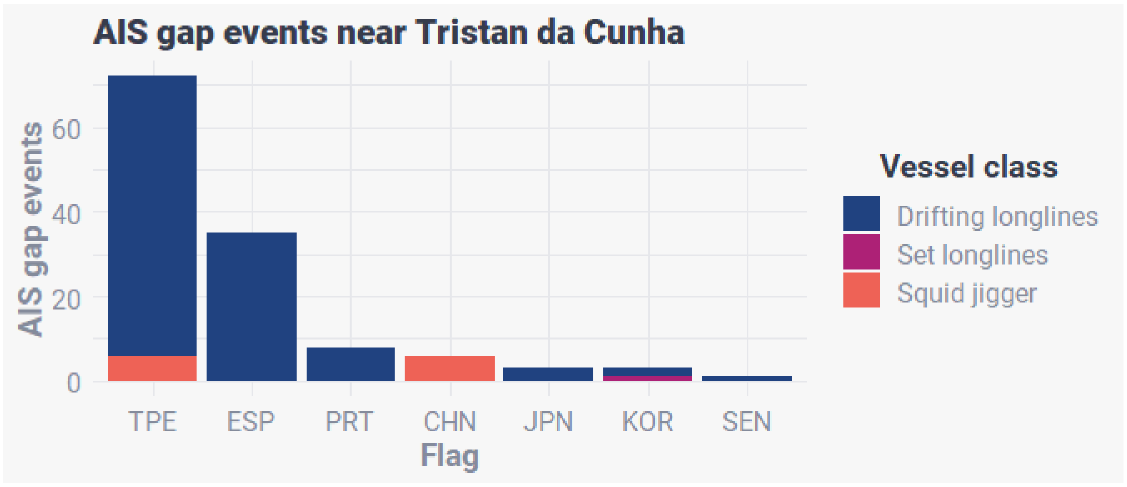
```
gap_events_long <- gap_events %>%
  mutate(off = str_c(off_lon, off_lat, sep = ","),
         on = str_c(on_lon, on_lat, sep = ",")) %>%
  dplyr::select(gap_id, ssvid, gap_hours, gap_start, gap_end, off, on,
vessel_tonnage_gt, vessel_class) %>%
  pivot_longer(off:on,
               names_to = "gap_event",
               values_to = "lonlat") %>%
  separate(lonlat,
           into = c("lon", "lat"),
           sep = ",") %>%
  mutate(lon = lon %>% as.numeric(),
         lat = lat %>% as.numeric())
```

Which gaps crossed the EEZ? Here we converted transits into spatial objects and use the spatial function `sf::st_intersects()` to return a TRUE/FALSE column for which transits intersected with the EEZ

```
# convert transits to a spatial linestring object
gaps_sf <- gap_events_long %>%
  st_as_sf(coords = c('lon', 'lat'), crs = 4326) %>%
  group_by(gap_id) %>%
  summarise(do_union = FALSE) %>%
  st_cast('LINESTRING')

# use sf::st_intersects to return TRUE/FALSE for each transit that crossed the EEZ
gaps_sf$crosses_eez <- st_intersects(gaps_sf, eez_tdc, sparse = FALSE) %>%
as.logical()

# merge this information to the gap_events_long dataframe using gap_id
gap_events_long <- merge(
  x = gap_events_long,
  y = gaps_sf %>% st_drop_geometry(),
  by = "gap_id",
  all.x = TRUE
)
```

How many gap events were detected by vessels belonging to each vessel class and vessel flag?

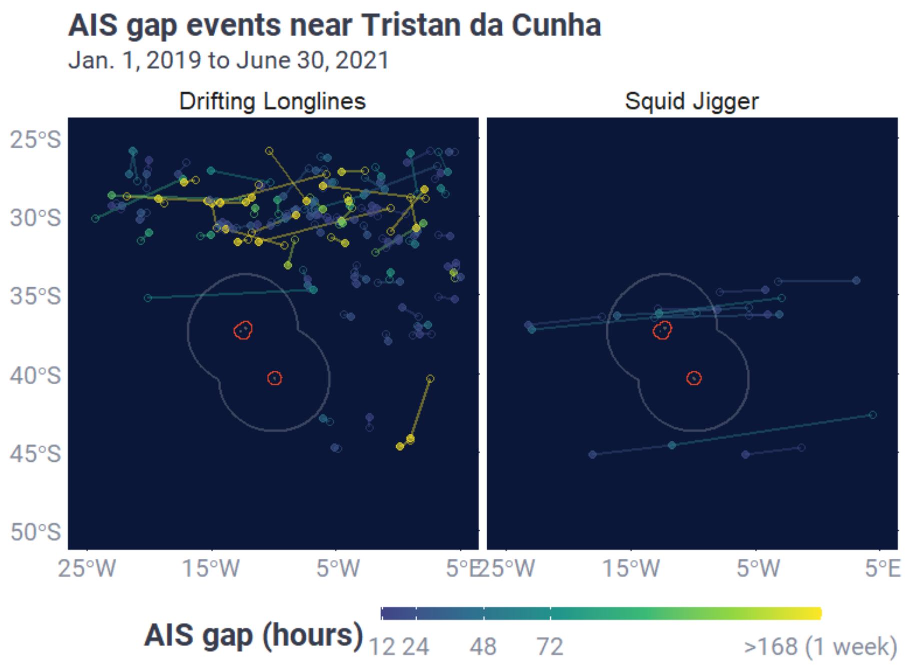
```
# summarise gap events per flag and vessel class
# arrange in descending order of number of gaps
# this allows us to arrange the vessel flags in the next plot by number of gap events
gap_summary <- gap_events %>%
  group_by(flag, vessel_class) %>%
  summarise(n_gaps = n_distinct(gap_id),
            gap_duration = mean(gap_hours)) %>%
  arrange(desc(n_gaps))

# plot the number of gap events per flag as a bar plot
# colour each barplot by vessel category
gap_summary %>%
  mutate(flag = flag %>%
           factor(levels = distinct(gap_summary, flag) %>%
                    pull(flag)) %>%
                    recode_factor("TWN" = "TPE")) %>%
ggplot() +
    geom_col(aes(x = flag, y = n_gaps, fill = vessel_class)) +
    scale_fill_manual(values = gfw_palette("chart")[c(1,3,5)],
                      name = "Vessel class",
                      labels = c("Drifting longlines", "Set longlines", "Squid
jigger")) +
    labs(x = "Flag",
         y = "AIS gap events",
         title = "AIS gap events near Tristan da Cunha") +
    theme_gfw_cian() +
    theme(plot.title = element_text(size = 16),
          plot.subtitle = element_text(size = 14),
          axis.title = element_text(size = 14),
          axis.text = element_text(size = 12),
          legend.title = element_text(size = 14),
          legend.text = element_text(size = 12))
```



Map of AIS gap events by fishing vessels, faceted by vessel class (excluding set_longlines):

```
# create vessel category labels for labelling map facets
vcat_labs <- c(squid_jigger = "Squid Jigger", drifting_longlines = "Drifting
Longlines")

# set bounding area
bounding_2 <- fishwatchr::transform_box(xlim = c(-25, 5),
                                        ylim = c(-50, -25),
                                        output_crs = "+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs")

# map of gaps
gap_events_long %>%
  # create a new column gap_hours_alt that associates a gap duration with the off
point
  # this allows us to map the colour fill of the point to the gap duration
  mutate(gap_hours_alt = if_else(gap_event == "off", gap_hours, as.numeric(NA))) %>%
  filter(vessel_class != "set_longlines") %>%
  ggplot() +
  geom_path(aes(x = lon, y = lat, colour = gap_hours, group = gap_id), size = 1,
alpha = 0.4) +
  geom_point(aes(x = lon, y = lat, colour = gap_hours, fill = gap_hours_alt), size =
2, shape = 21, alpha = 0.6) +
  fishwatchr::geom_gfw_eez(lwd = 1) +
  geom_sf(data = atba_sf, colour = gfw_palette("primary")[1], linetype = 1, fill =
NA) +
  geom_sf(data = tdc_sf, fill = gfw_palette("map_country_dark")[1]) +
  facet_grid(~ vessel_class,
             labeller = labeller(vessel_class = vcat_labs)) +
  labs(title = "AIS gap events near Tristan da Cunha",
```

```r
      subtitle = "Jan. 1, 2019 to June 30, 2021") +
  scale_colour_viridis_c(name = "AIS gap (hours)",
                         begin = 0.2,
                         end = 1.0,
                         option = "viridis",
                         limits = c(12, 168),
                         oob = scales::squish,
                         breaks = c(12, 24, 48, 72, 168),
                         labels = c("12", "24", "48", "72", ">168 (1 week)")) +
  scale_fill_viridis_c(name = "AIS gap (hours)",
                         begin = 0.2,
                         end = 1.0,
                         option = "viridis",
                         limits = c(12, 168),
                         oob = scales::squish,
                         breaks = c(12, 24, 48, 72, 168),
                         labels = c("12", "24", "48", "72", ">168 (1 week)"),
                       na.value = NA) +
  scale_alpha_manual(values = c(0, 1)) +
  scale_x_continuous(breaks = c(-25, -15, -5, 5)) +
  guides(fill = "none", alpha = "none") +
  theme_gfw_map_cian() +
  theme(plot.title = element_text(size = 20),
        plot.subtitle = element_text(size = 16),
        axis.text = element_text(size = 16),
        legend.title = element_text(size = 20),
        legend.text = element_text(size = 16),
        strip.text = element_text(size = 16)) +
  coord_sf(xlim = c(bounding_2$box_out[['xmin']], bounding_2$box_out[['xmax']]),
           ylim = c(bounding_2$box_out[['ymin']], bounding_2$box_out[['ymax']]),
           crs = bounding_2$out_crs)
```



Looking at this map we can see that the straight-line interpolations between off and on positions for a number of AIS gap events traversed Tristan da Cunha's EEZ. Most of these gap events were associated with squid jigger fishing vessels. Next, we wanted to look at the extended tracks of these vessels to assess whether these gap events were atypical of normal vessel activity.

```
# create a list of vessel ids associated with squid jigger fishing vessels
# with AIS gap events
ssvid_squid <- gap_events %>%
  filter(vessel_class == "squid_jigger") %>%
  distinct(ssvid) %>%
  pull(ssvid)
```

Tweak my original track query to pull the above tracks for 2019-2020 from the `pipe_production_YYYYMMDD_fishing` table

```
query_9 <- readr::read_file(str_c("queries",
"q_tdc_atba_squid_jigger_voyages_gaps.sql", sep="/"))
```

Run the query:

```
squid_tracks <- fishwatchr::gfw_query(query = query_9,
                                      run_query = TRUE,
                                      con = con)$data
```

Alternatively, the queried data can be loaded locally here:

```
squid_tracks <- readr::read_rds("data_production/data/squid_jigger_tracks_gaps.rds")
```

Query all the gap events along these tracks for these 10 vessels:

```
query_10 <- readr::read_file(str_c("queries",
"q_tdc_atba_squid_jigger_gap_events.sql", sep="/"))
```

Run the query:

```
squid_gaps <- fishwatchr::gfw_query(query = query_10,
                                    run_query = TRUE,
                                    con = con)$data
```

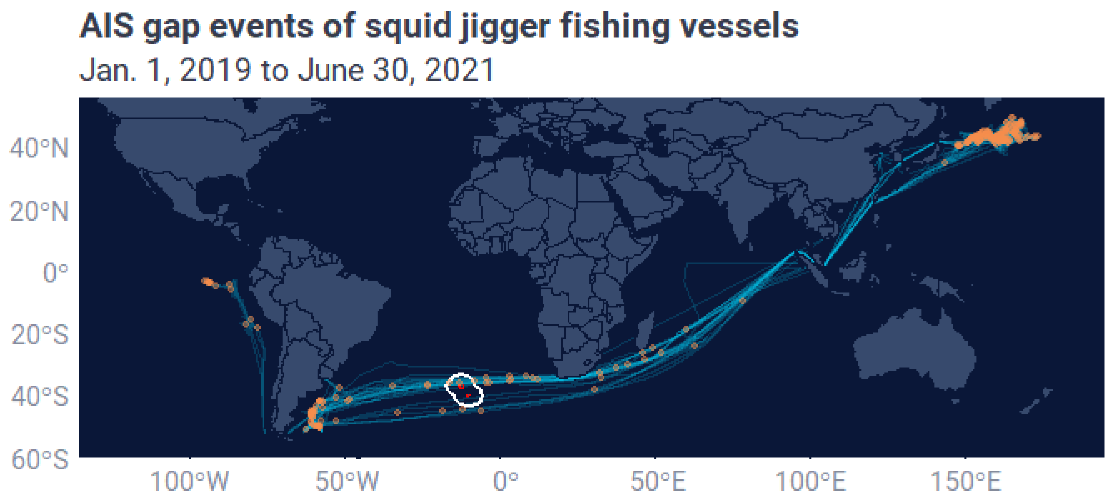
Alternatively, the queried data can be loaded locally here:

```
squid_gaps <- readr::read_rds("data_production/data/ais_gaps_squid.rds")
```

Create a map of these tracks with all associated gap events:

```
# filter out trips that cross 180° longitude
# these are complicated to map
trip_sum <- squid_tracks %>%
  group_by(trip_id) %>%
  summarise(keep = min(lon) > -120) %>%
  filter(keep == TRUE)

# set bounding area
bounding_3 <- fishwatchr::transform_box(xlim = c(-120, 180),
```

```r
                                                ylim = c(-55, 50),
                                                output_crs = "+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs")

# map tracks and gap events
squid_tracks %>%
  # keep only tracks that don't cross 180° longitude
  filter(trip_id %in% trip_sum$trip_id) %>%
  arrange(timestamp) %>%
  ggplot() +
    geom_path(aes(x = lon, y = lat, group = trip_id), colour =
gfw_palette("tracks")[1], alpha = 0.2) +
    geom_sf(data = eez_tdc, fill = NA, colour = "white", size = 1) +
    fishwatchr::geom_gfw_land() +
    geom_sf(data = atba_sf, colour = "red", linetype = 1, fill = NA) +
    geom_sf(data = tdc_sf, fill = gfw_palette("map_country_dark")[1]) +
  # add gap events as points
    geom_point(data = squid_gaps, aes(x = off_lon, y = off_lat), colour =
gfw_palette("orange")[1], size = 1, alpha = 0.4) +
    labs(title = "AIS gap events of squid jigger fishing vessels",
         subtitle = "Jan. 1, 2019 to June 30, 2021") +
    theme_gfw_map_cian() +
    theme(plot.title = element_text(size = 16),
          plot.subtitle = element_text(size = 14),
          axis.text = element_text(size = 12)) +
    coord_sf(xlim = c(bounding_3$box_out[['xmin']], bounding_3$box_out[['xmax']]),
             ylim = c(bounding_3$box_out[['ymin']], bounding_3$box_out[['ymax']]),
             crs = bounding_3$out_crs)
```



AIS gap events of squid jigger fishing vessels
Jan. 1, 2019 to June 30, 2021
40°N
20°N
0°
20°S
40°S
60°S
100°W
50°W
0°
50°E
100°E
150°E