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](https://github.com/GlobalFishingWatch/pew_gfw_tristan_da_cunha/blob/main/queries/q_tdc_vessel_traffic_atba_voyages.sql). 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")

## Reading layer `ATBA\_consolidate\_25nm\_buffer\_wgs84' from data source   
## `C:\Users\cianl\My Drive\projects\pew\_gfw\_tristan\_da\_cunha\geodata\tdc\_atba\ATBA\_consolidate\_25nm\_buffer\_wgs84.shp'   
## using driver `ESRI Shapefile'  
## Simple feature collection with 1 feature and 13 fields  
## Geometry type: MULTIPOLYGON  
## Dimension: XY  
## Bounding box: xmin: -13.19894 ymin: -40.72459 xmax: -9.404225 ymax: -36.69803  
## Geodetic CRS: GCS\_unknown

st\_crs(atba\_sf) <- 4326  
  
# 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")

## Reading layer `tristan\_da\_cunha\_archipelago\_osm' from data source   
## `C:\Users\cianl\My Drive\projects\pew\_gfw\_tristan\_da\_cunha\geodata\tdc\_osm\tristan\_da\_cunha\_archipelago\_osm.shp'   
## using driver `ESRI Shapefile'  
## Simple feature collection with 49 features and 1 field  
## Geometry type: POLYGON  
## Dimension: XY  
## Bounding box: xmin: -12.70632 ymin: -40.37117 xmax: -9.874093 ymax: -37.06207  
## Geodetic CRS: WGS 84

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:

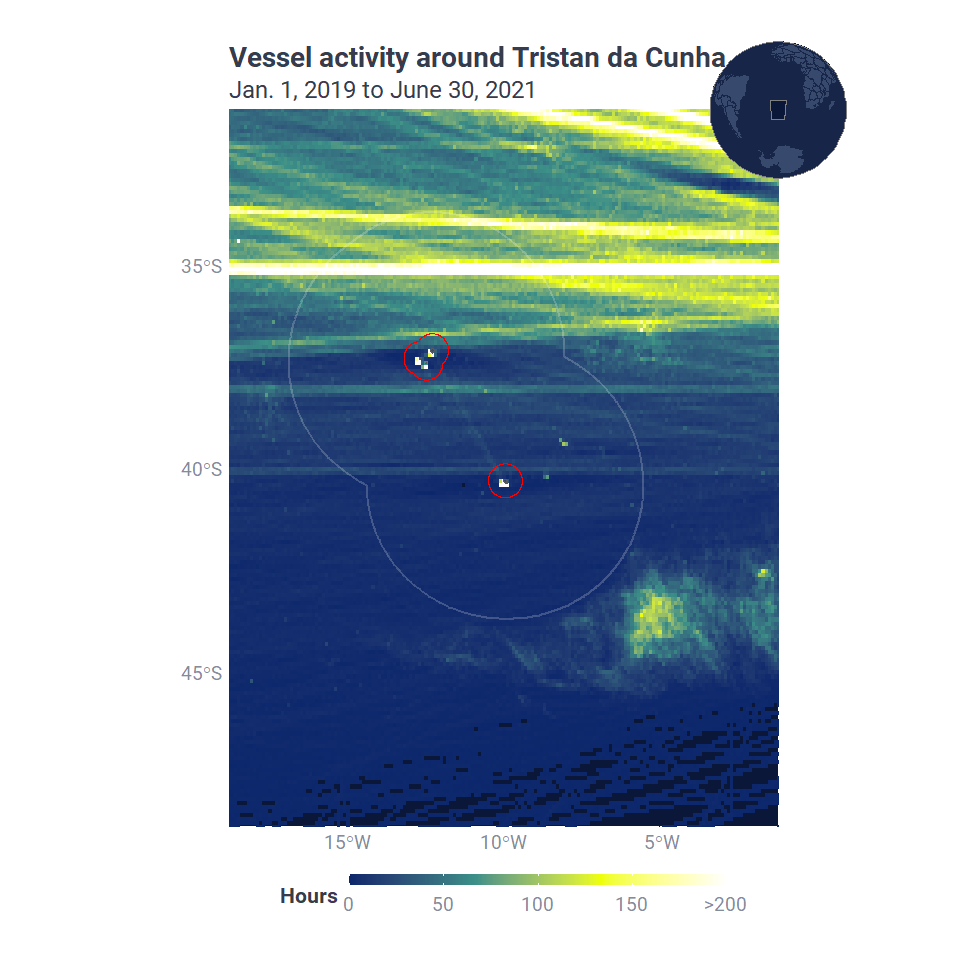
vt\_gridded <- readr::read\_rds("data\_production/data/gridded-vessel-activity-2019-2021.rds")

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)

## Spherical geometry (s2) switched off

# 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



# 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:

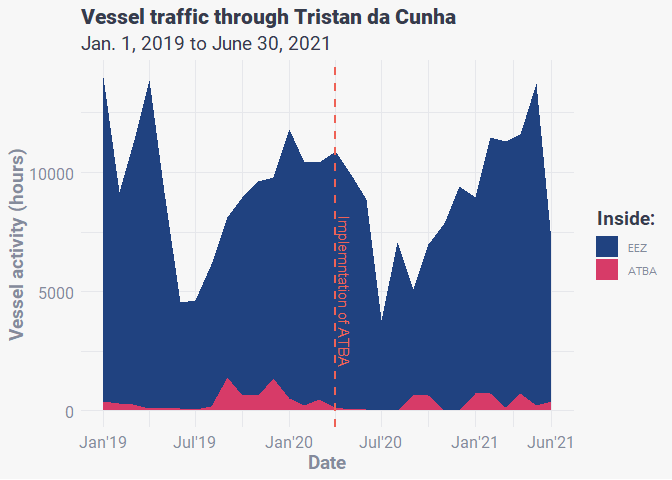
vt\_eez\_summary <- read\_rds("data\_production/data/summary-vessel-traffic-eez-2019-2021.rds")

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:
   1. IF inside atba/eez AND previous position outside: transit\_no <- t
   2. IF inside AND previous position inside AND previous position within 7 days: transit\_no <- t
   3. IF inside AND previous position inside AND previous position outside 7 days: transit\_no <- t+1
   4. 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))  
   
 # 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](https://www.vendeeglobe.org/) 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](https://www.science.org/doi/abs/10.1126/science.aao5646)), 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))

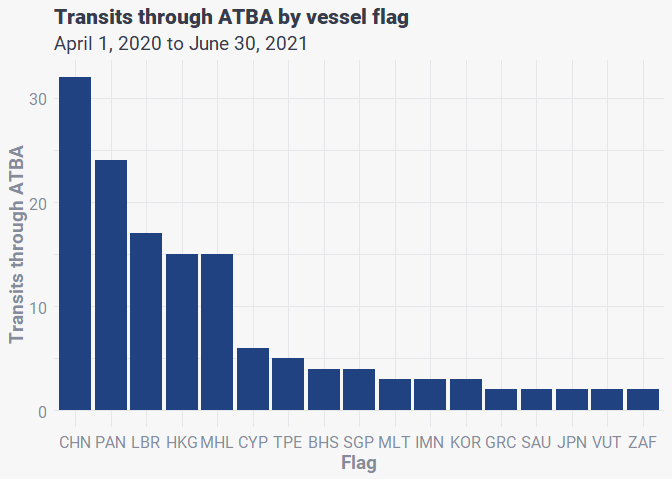
## # A tibble: 2 x 3  
## after\_atba n\_transits n\_vessels  
## <lgl> <int> <int>  
## 1 FALSE 454 416  
## 2 TRUE 125 123

# 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))



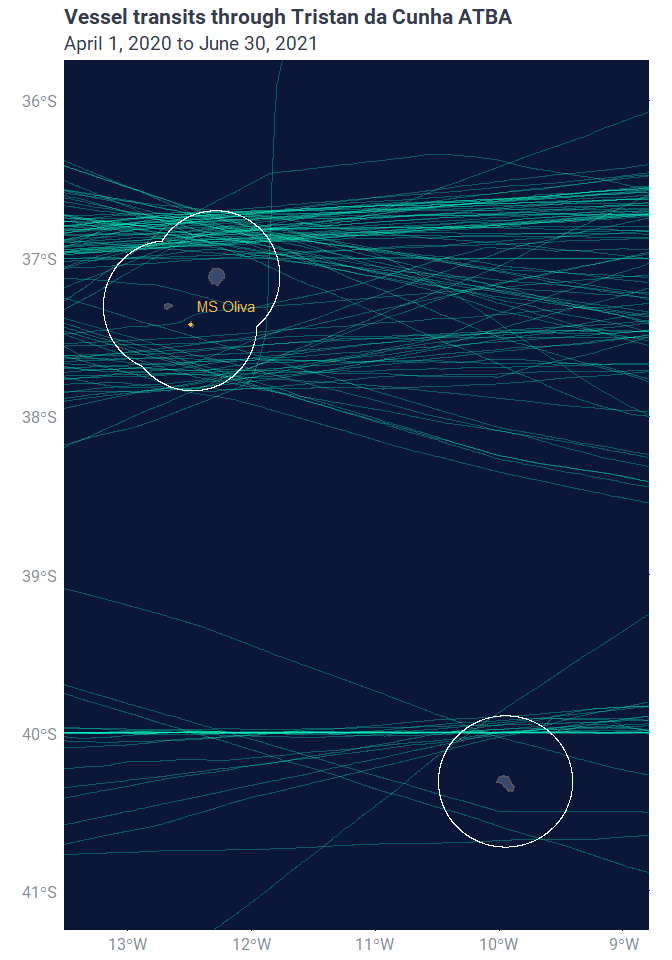
Number of transits by vessel class:

# 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)



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.

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.

query\_6 <- readr::read\_file(str\_c("queries", "q\_tdc\_atba\_squid\_jigger\_transits\_fishing.sql", sep="/"))

Run the query:

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:

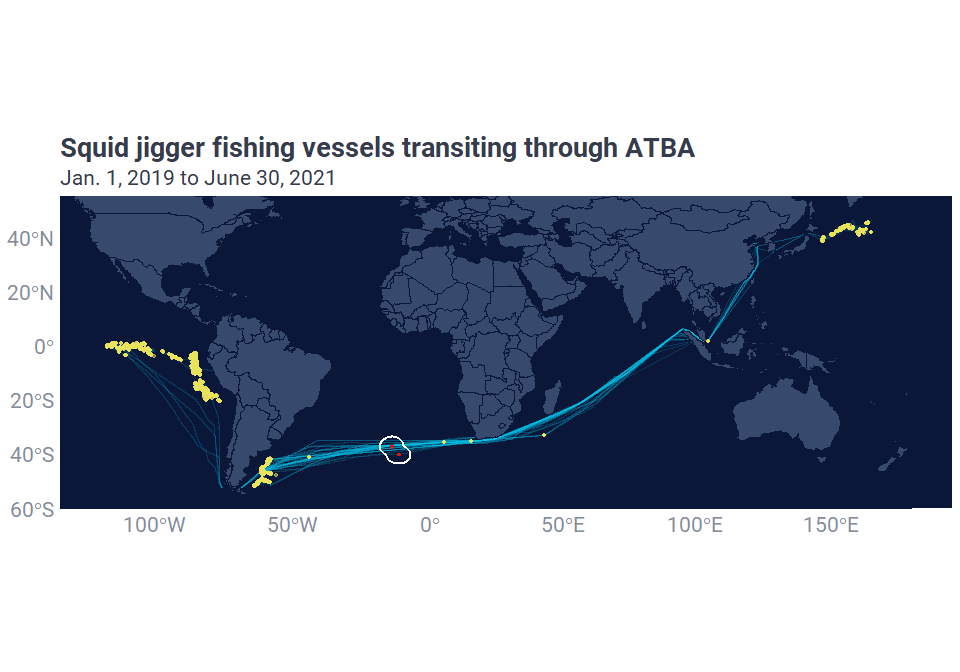
squid\_tracks\_2 <- squid\_tracks\_2 %>%   
 group\_by(trip\_id) %>%   
 summarise(through\_atba = sum(inside\_atba) > 0) %>%   
 merge(squid\_tracks\_2, by = "trip\_id", all.y = TRUE) %>%   
 filter(through\_atba == TRUE)

Alternatively, the queried data can be loaded locally here:

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:

# 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)



## 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:

# 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.

vt\_atba\_sf$dist\_to\_shore\_m <- st\_distance(vt\_atba\_sf, tdc\_sf) %>%   
 apply(1, min)

Merged back with vt\_transits:

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:

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:

# 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:

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)

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:

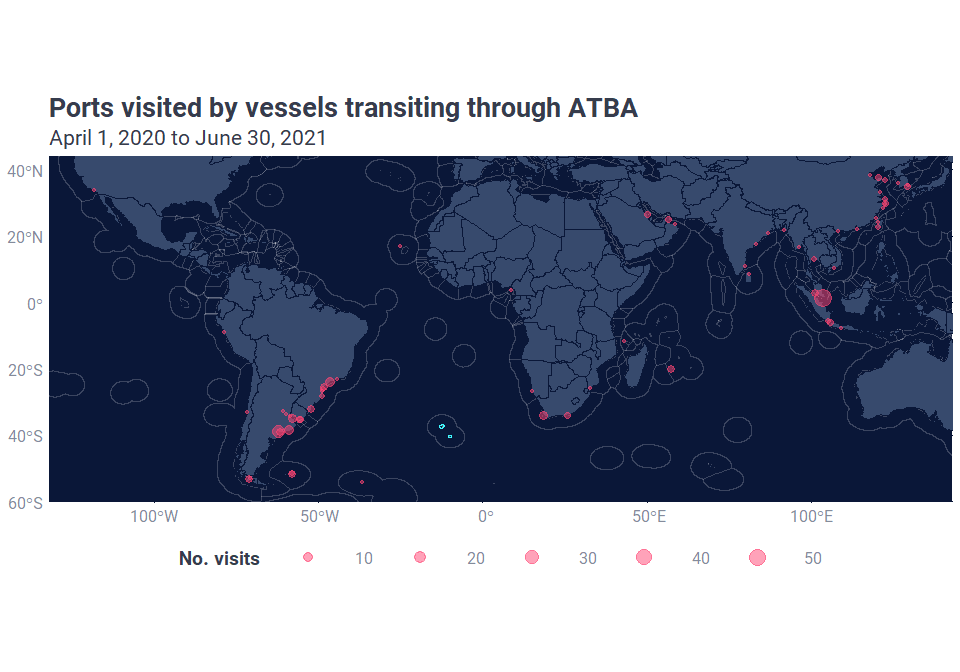
transit\_ports <- readr::read\_rds("data\_production/data/transit\_ports\_c4.rds")

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)

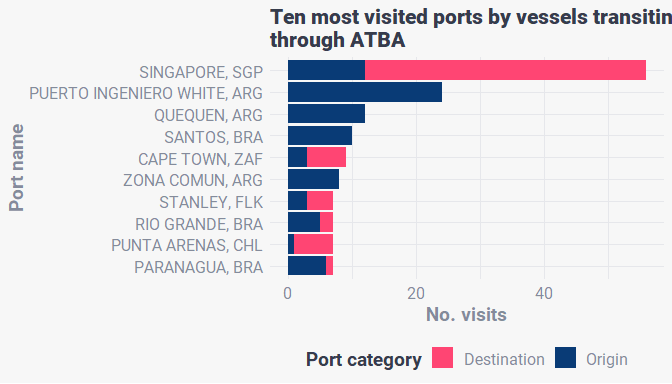
Here we created a map with points showing the locations of each port and the points sized in proportion to the number of transits beginning or ending there:

# 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)



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.

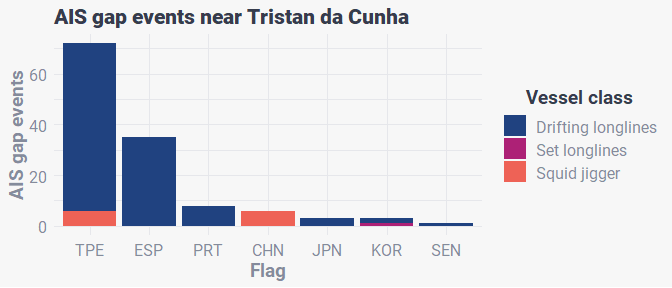
gap\_events\_long <- gap\_events %>%   
 mutate(off = str\_c(off\_lon, off\_lat, sep = ","),  
 on = str\_c(on\_lon, on\_lat, sep = ",")) %>%   
 dplyr::select(gap\_id, ssvid, gap\_hours, gap\_start, gap\_end, off, on, vessel\_tonnage\_gt, vessel\_class) %>%   
 pivot\_longer(off:on,  
 names\_to = "gap\_event",  
 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  
)

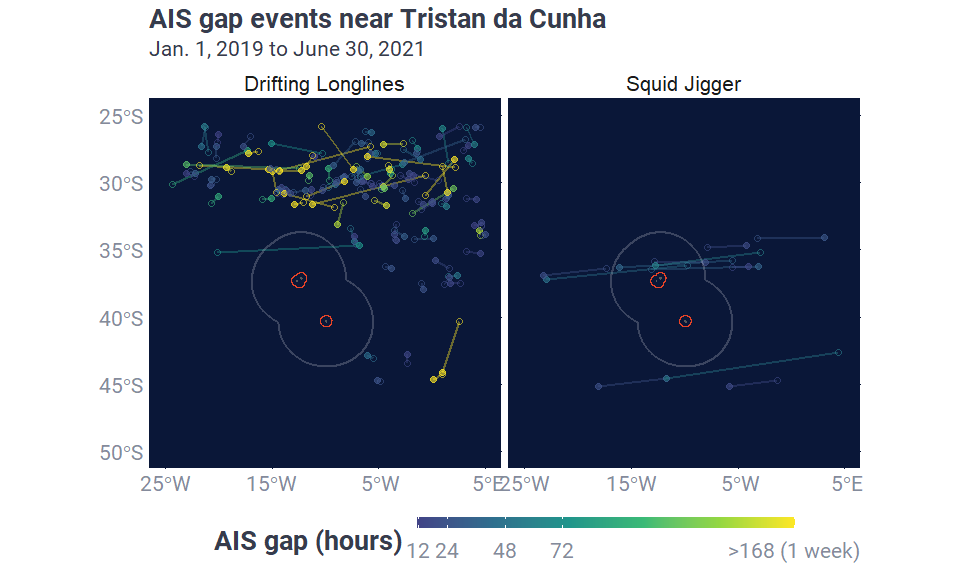
How many gap events were detected by vessels belonging to each vessel class and vessel flag?

# 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",  
 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),   
 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)

