data_processing

Marina Chaji

3/8/2022

Project setup

here(), load libraries, and set a data vintage.

```
# Set Path
here::i_am("data_wrangle/data_processing.Rmd")
```

```
## here() starts at /net/home2/mlee/Effort-Displacement---Scallop
```

```
# Please ensure you have the proper packages installed with (install.packages()) or a request to ITD if


library("here")
library("leaflet")
#library("tidyverse")
tidyverse_short<-c("broom","cli","crayon","dbplyr","dplyr","dtplyr","forcats","ggplot2","googledrive","
lapply(tidyverse_short, require, character.only = TRUE)
```

```
## Loading required package: broom

## Loading required package: cli

## Loading required package: crayon

##
## Attaching package: 'crayon'

## The following object is masked from 'package:cli':
##
##     num_ansi_colors

## Loading required package: dbplyr

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:dbplyr':
##
##     ident, sql

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
```

```
##     intersect, setdiff, setequal, union

## Loading required package: dtplyr

## Loading required package: forcats

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:crayon':
##
##     %+%

## Loading required package: googledrive

## Loading required package: googlesheets4

##
## Attaching package: 'googlesheets4'

## The following objects are masked from 'package:googledrive':
##
##     request_generate, request_make

## Loading required package: hms

## Loading required package: httr

## Loading required package: jsonlite

## Loading required package: lubridate

##
## Attaching package: 'lubridate'

## The following object is masked from 'package:hms':
##
##     hms

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## Loading required package: magrittr

## Loading required package: modelr

##
## Attaching package: 'modelr'

## The following object is masked from 'package:broom':
##
##     bootstrap

## Loading required package: pillar

##
## Attaching package: 'pillar'

## The following object is masked from 'package:dplyr':
##
##     dim_desc
## The following object is masked from 'package:cli':
##
##     style_bold

## Loading required package: purrr

##
## Attaching package: 'purrr'

## The following object is masked from 'package:magrittr':
##
##     set_names

## The following object is masked from 'package:jsonlite':
##
##     flatten

## Loading required package: readr

## Loading required package: readxl

## Loading required package: reprex

## Loading required package: rlang

##
## Attaching package: 'rlang'

## The following objects are masked from 'package:purrr':
##
##     %@%, as_function, flatten, flatten_chr, flatten_dbl, flatten_int,
##     flatten_lgl, flatten_raw, invoke, splice

## The following object is masked from 'package:magrittr':
##
##     set_names

## The following objects are masked from 'package:jsonlite':
##
##     flatten, unbox

## The following object is masked from 'package:crayon':
##
##     chr

## Loading required package: rstudioapi

## Loading required package: rvest

##
## Attaching package: 'rvest'

## The following object is masked from 'package:readr':
##
##     guess_encoding

## Loading required package: stringr

## Loading required package: tibble

## Loading required package: tidyr

##
## Attaching package: 'tidyr'
## The following object is masked from 'package:magrittr':
##
##     extract
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'xml2'
```

```
## The following object is masked from 'package:rlang':
##
##     as_list
```

```
library("sf")
```

```
## Linking to GEOS 3.7.2, GDAL 3.1.3, PROJ 6.3.2
```

```
library("raster")
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library("rgdal")
```

```
## Please note that rgdal will be retired by the end of 2023,
## plan transition to sf/stars/terra functions using GDAL and PROJ
## at your earliest convenience.
##
## rgdal: version: 1.5-27, (SVN revision 1148)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.1.3, released 2020/09/01
## Path to GDAL shared files: /usr/local/share/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 6.3.2, May 1st, 2020, [PJ_VERSION: 632]
## Path to PROJ shared files: /usr/share/proj
## Linking to sp version:1.4-6
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading sp or rgdal.
```

```
library("data.table")
```

```
##
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:raster':
##
##     shift
```

```
## The following object is masked from 'package:rlang':
##
##     :=
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library("tmaptools")
library("tmap")
library("RODBC")
#library("RODM")
library("epiDisplay")
```

```
## Loading required package: foreign

## Loading required package: survival

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following objects are masked from 'package:raster':
##
##     area, select

## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: nnet

##
## Attaching package: 'epiDisplay'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library("fredr")
vintage_string<-Sys.Date()
vintage_string<-gsub("-","_",vintage_string)
```

```
#This code looks into data_intermediate and sets the vintage_string according to the most recent data
datasets_list<-list.files(path=here("data","intermediate"), pattern="RESULT_COMPILED_")
datasets_list<-gsub("RESULT_COMPILED_","",datasets_list )
datasets_list<-gsub(".Rds","",datasets_list)
datasets_list<-gsub(".csv","",datasets_list)
vintage_string<-max(datasets_list)
rm(datasets_list)
```

We will:

1. Try to avoid copying data; when we rely on data from other people, we will read it directly into memory from the network location or Oracle.
2. Sometimes this is unnecessary, so we will copy external data into the "data/external" folder. We will have a separate subfolder for shapefiles.
3. Store an intermediate data product in "data/intermediate".
4. Store final data products in "data/main."

5. Use a vintage "suffix" to denote when we have extracted data.

Organization

We will:

1. Try to avoid copying data; when we rely on data from other people, we will read it directly into memory from the network location or Oracle.
2. Sometimes this is unnecessary, so we will copy external data into the "data/external" folder. We will have a separate subfolder for shapefiles.
3. Store an intermediate data product in "data/intermediate".
4. Store final data products in "data/main."
5. Use a vintage "suffix" to denote when we have extracted data.

Read in oracle passwords and set network directory

This is a block of code where we set up the oracle passwords and make R aware of folders on the network.

```
source(here("data_wrangle","credentials.R"))

# Set the network_location_desktop and network_location_remote variables somewhere OUTSIDE of this code

#Comment one of these out, depending on whether you are running this code on a server or locally (with
net<-network_location_desktop
net<-network_location_remote

# These are not part of the project path
offshoreWind_directory<-file.path(net,"home5", "dcorvi","OffshoreWind","offshoreWind4","data")
spacepanels_directory<-file.path(net,"home2", "mlee","dropoff","wind")
cost_directory<-file.path(net,"work5","socialsci","Trip_Costs","2007-2020")

# Set up paths.
East_Cst_crop_2020_path<- here("data","external","shapefiles","East_Cst_crop_2020_extended")
TMSQ_path<-here("data","external","shapefiles","Ten Minute Squares Cut North and Greater Atlantic")
All_Lease_Areas_Shapefile_path<-here("data","external","shapefiles","All_Lease_Areas_Shapefile")
```

```
#Read in RDS

Scallop_Linkingorg <- readRDS(here("data","intermediate",paste0("Scallop_Linkingorg_",vintage_string,".
RESULT_COMPILED <- readRDS(here("data","intermediate",paste0("RESULT_COMPILED_",vintage_string,".Rds"))
APSD_DMIS_2 <- readRDS(here("data","intermediate",paste0("APSD_DMIS_2_",vintage_string,".Rds")))
all_yrs_costs <- readRDS(here("data","intermediate",paste0("all_yrs_costs_",vintage_string,".Rds")))
```

Introduction

The main idea of the model is that the fishermen/decision-makers choose from a number of alternatives, where the choice occasion is a fishing trip and selects the one that yields the highest expected utility level on any given choice occasion. By observing and modeling how decision-makers change their preferred site option in response to the changes in the levels of the site attributes, it is possible to determine how decision-makers tradeoff between the different fishing ground characteristics.

Long Term Objectives

The project objective is to develop a site-choice model primarily, improve, maintain, and disseminate a standardized fisheries dependent data set and analytical summaries that provide a more precise, accurate, comprehensive, and timely evaluation of area-specific socioeconomic impacts associated with ecosystem fishery management initiatives, offshore energy development, and offshore aquaculture development. The site-choice model and underlying data set will help support fishery and ecosystem management decisions to achieve optimum yield in each fishery and the nation's most significant benefit.

Understanding the effects of wind energy areas that are early in the process may be more impactful from a policy perspective. So, not necessarily the current wind areas, but the next block that may be coming over the next 10-30 years. Also, cumulative effects may be important.

Empirical Setting

Scallop Fishery

We are modeling the location choices of fishing vessels in the Limited Access Days-at-Sea scallop fishery. There are approximately 300-330 of these fishing vessels. They are allocated "Open Area Days-at-Sea" and a quantity of trips and/or pounds into the "Access Areas." They catch approximately 95% of the scallops. The Limited Access DAS fleet can be further subdivided into Full-Time, Part-Time, and Occasional Fleets. Vessels primarily use the New Bedford scallop dredge, but a few use a smaller dredge or a bottom trawl. Over the 13 years in our dataset, there are approximately 40,000 trips taken by this fleet, split roughly evenly into "Open areas" and "Access Areas."

For Fishing Year 2016 and earlier, the fishing year Ran from March 1 to Feb 28/29. For fishing year 2017, the year ran from March 1 to March 31. For 2018 and later, the fishing year runs from April 1 to March 31.

Wind Energy

Here is a short description of the wind energy areas and how they will close (or not close) area to fishing. 18 wind areas currently under dev. But many more are likely.

How close will fishing be able to occur within Wind Lease Areas / Turbines?

The wind energy areas do not match the ten minute squares; we are currently planning on simulating the effects of closing a wind energy area by closing an entire ten minute square that is inside or touching a WEA.

The buried cable route from a WEA to shore is likely to be closed as well. Cable buried at shallow depths and marked with concrete.

We classify the vessels as FullTime, PartTime based on these PLAN_CAT variables that are in DMIS. We also generate categorical variables corresponding to LA and GC columns. Note that a vessel can hold both an LA and a GC permit at the same time. The summary tables below will have lots of observations corresponding to Scallop_Linkingorg[ftpt]=0, LA=0, and GC=0. This is expected, because it has everything from DMIS.

It's not entirely clear how the the PLAN_CAT variables are constructed in DMIS. They are derived from permit data, which is generated when the vessel owner renews a permit at the beginning of the fishing year, when an owner transfers a permit, or when a permit is added. The freqency of the underlyingy data is irregular, but approximately annual. The DMIS algorithm may match to the permit holdings on the day of the trip, or it may match to the permit held at the beginning of the year. When there is a conflict between these variables (PLAN_CAT, ftpt, GC, LA) and the variables based on ACTIVITY_CODE (Plan Code and Program Code), we should prefer the variables based on ACTIVITY_CODE.

```
# Bin the LA vessels into full time or part time.
Scallop_Linkingorg$ftpt<-"None"
Scallop_Linkingorg$ftpt[Scallop_Linkingorg$SC_2=="TRUE"]<-"FullTime"
Scallop_Linkingorg$ftpt[Scallop_Linkingorg$SC_5=="TRUE"]<-"FullTime"
Scallop_Linkingorg$ftpt[Scallop_Linkingorg$SC_7=="TRUE"]<-"FullTime"

Scallop_Linkingorg$ftpt[Scallop_Linkingorg$SC_3=="TRUE"]<-"PartTime"
Scallop_Linkingorg$ftpt[Scallop_Linkingorg$SC_6=="TRUE"]<-"PartTime"
Scallop_Linkingorg$ftpt[Scallop_Linkingorg$SC_8=="TRUE"]<-"PartTime"

Scallop_Linkingorg$ftpt[Scallop_Linkingorg$SC_4=="TRUE"]<-"Occasional"
Scallop_Linkingorg$ftpt[Scallop_Linkingorg$SC_9=="TRUE"]<-"Occasional"


# Construct a logical variable for GC
Scallop_Linkingorg$GC<-(Scallop_Linkingorg$LGC_A=="TRUE" | Scallop_Linkingorg$LGC_B=="TRUE" | Scallop_L

# Construct a logical variable for LA
Scallop_Linkingorg$LA<-(Scallop_Linkingorg$ftpt=="PartTime" | Scallop_Linkingorg$ftpt=="FullTime")


#Make some tables
table(Scallop_Linkingorg$ftpt)
```

```
##
## FullTime     None PartTime
##    87520  5215921    17576
```

```
table(Scallop_Linkingorg$GC)
```

```
##
##   FALSE    TRUE
## 4757157  562122
```

```
table(Scallop_Linkingorg$LA,Scallop_Linkingorg$GC)
```

```
##
##           FALSE    TRUE
##   FALSE 4706607  507576
##   TRUE    50550   54546
```

```
#Select certain columns
Scallop_Linkingorg_bak<-Scallop_Linkingorg
Scallop_Linkingorg<-dplyr::select(Scallop_Linkingorg, c(TRIP_ID,DOCID, ACTIVITY_CODE, ftpt, GC,LA))

is.logical(Scallop_Linkingorg$GC)
```

```
## [1] TRUE
```

```
is.logical(Scallop_Linkingorg$LA)
```

```
## [1] TRUE
```

We don't want to create a single plan column, because a vessel could have multiple kinds of scallop permits. Instead, if we want just the Fulltime LA vessels, we can do something like:

```
Limited_Access <-Scallop_Linkingorg %>%
    filter(LA=="TRUE")
```

```
Limited_Access_ft<-Limited_Access %>%
    filter(ftpt=="FullTime")
```

Data Cleaning

1. Construct total revenue at the subtrip level.
2. Filter down to only Scallop Species
3. Seperate Dates & Times and Delete Old Dates Column
4. Delete Columns that are not need
5. NESPP3 & SOURCE Values do not vary across the observations, so these two columns can be deleted

```
APSD_DMIS_2<-APSD_DMIS_2 %>%
  group_by(IMGID) %>%
  mutate(DOLLAR_ALL_SP=sum(DOLLAR)) %>%
  relocate(DOLLAR_ALL_SP, .after=DOLLAR)
```

```
Scallops <- APSD_DMIS_2 %>%
  filter (SPPNAME == "SCALLOPS/BUSHEL")

#Separate Dates & Times
Scallops$Date <- as.Date(Scallops$DATE_TRIP)
Scallops$Time <- format(Scallops$DATE_TRIP,"%H:%M:%S")

#Drop columns that are not needed
Scallops$NESPP3<- NULL
Scallops$SOURCE<- NULL
```

#Construct scallop fishing year variable

```
Scallops$scallop_fishing_year<-"None"
Scallops$Date <- as.character.Date(Scallops$Date)

# The Scallop fishing years from the year 2016 and all prior are from March 1st to February 28th or 29t
Scallops$scallop_fishing_year[Scallops$Date >= "2007-03-01" & Scallops$Date <= "2008-02-29"] <- "2007"
Scallops$scallop_fishing_year[Scallops$Date >= "2008-03-01" & Scallops$Date <= "2009-02-29"] <- "2008"
Scallops$scallop_fishing_year[Scallops$Date >= "2009-03-01" & Scallops$Date <= "2010-02-29"] <- "2009"
Scallops$scallop_fishing_year[Scallops$Date >= "2010-03-01" & Scallops$Date <= "2011-02-29"] <- "2010"
Scallops$scallop_fishing_year[Scallops$Date >= "2011-03-01" & Scallops$Date <= "2012-02-29"] <- "2011"
Scallops$scallop_fishing_year[Scallops$Date >= "2012-03-01" & Scallops$Date <= "2013-02-29"] <- "2012"
Scallops$scallop_fishing_year[Scallops$Date >= "2013-03-01" & Scallops$Date <= "2014-02-29"] <- "2013"
Scallops$scallop_fishing_year[Scallops$Date >= "2014-03-01" & Scallops$Date <= "2015-02-29"] <- "2014"
Scallops$scallop_fishing_year[Scallops$Date >= "2015-03-01" & Scallops$Date <= "2016-02-29"] <- "2015"
Scallops$scallop_fishing_year[Scallops$Date >= "2016-03-01" & Scallops$Date <= "2017-02-29"] <- "2016"

#The scallop fishing year in 2017 was from March 1st to March 31st
Scallops$scallop_fishing_year[Scallops$Date >= "2017-03-01" & Scallops$Date <= "2018-03-31"] <- "2017"

#The scallop fishing years from 2018 onward are from April 1st to March 31st
Scallops$scallop_fishing_year[Scallops$Date >= "2018-04-01" & Scallops$Date <= "2019-03-31"] <- "2018"
Scallops$scallop_fishing_year[Scallops$Date >= "2019-04-01" & Scallops$Date <= "2020-03-31"] <- "2019"
Scallops$scallop_fishing_year[Scallops$Date >= "2020-04-01" & Scallops$Date <= "2021-03-31"] <- "2020"
Scallops$scallop_fishing_year[Scallops$Date >= "2021-04-01" & Scallops$Date <= "2022-03-31"] <- "2021"
Scallops$scallop_fishing_year[Scallops$Date >= "2022-04-01" & Scallops$Date <= "2023-03-31"] <- "2022"
```

Merging

1. Merge Scallops & VTR Data Sets (RESULT.COMPILED). We keep all columns from both the APSD_DMIS_2 and RESULT.COMPILED datasets. We also:
   1. Filter out 2020 values
   2. Delete Extra PERMIT Column because there were a few missing values.
   3. Delete all TRIPCATG that are not 1. This isolates all commercial trips
   4. Drop rows corresponding to a “Not Fished” VTR.
2. Join the output of (1) with Activity Codes
3. Verify that we get what we think we should get.

```
##1. Merge Scallops & VTR Data Sets (RESULT.COMPILED). We keep all columns from both the APSD_DMIS_2 and

# all.x = TRUE & all.y = FALSE means I am keeping data with no match from DMIS table but dropping data w

# DOCID is used because of the following found in the data dictionary "VESLOG Trip record identifier, wh
VTR_DMIS_merge <- merge(RESULT_COMPILED,Scallops, by.x = "TRIPID", by.y = "DOCID", all.x = FALSE, all.y

## Filter out 2020 values
VTR_DMIS_merge <- VTR_DMIS_merge %>%
  filter(YEAR <= "2019")

# Delete Extra PERMIT Column
## Note: X was deleted because PERMIT.y had zero NAs and PERMIT.x had 25
VTR_DMIS_merge$PERMIT.x <- NULL

# Delete all TRIPCATG that are not 1. This isolates all commercial trips
## Type of trip: 1=Commercial; 2=Party; 3=Charter; 4=RSA/EFP. Note: RSA/EFP landings represent a small a

VTR_DMIS_merge <- VTR_DMIS_merge %>%
  filter(TRIPCATG == "1")
VTR_DMIS_merge$TRIPCATG <- NULL

# Delete all NOT_FISHED that are not 0. This indicates whether the 'Did not fish' box was checked on the
VTR_DMIS_merge <- VTR_DMIS_merge %>%
  filter(NOT_FISHED == "0")
VTR_DMIS_merge$NOT_FISHED <- NULL


## 2.
###Join VTR & DMIS Data with Activity Codes

# Delete duplicate rows; These are rows that share the same TRIPID, DOLLAR,LANDED, & TRIP_LENGTH
## Note: VTRs are self-reported and there is a potential for records to be submitted to regional office
VTR_DMIS_AC <- merge(VTR_DMIS_merge,Scallop_Linkingorg, by.x = "TRIPID", by.y = "DOCID", all.x = TRUE, a
VTR_DMIS_AC <- VTR_DMIS_AC %>%
  distinct(TRIPID,DOLLAR,TRIP_LENGTH,LANDED, .keep_all = TRUE)

## Created two sets of cost joins.
### 1. Before LA Estimation
### 2. After LA Estimation
VTR_DMIS_AC <- merge(VTR_DMIS_AC,all_yrs_costs, by.x = "TRIPID", by.y = "VTR_TRIPID", all.x = TRUE,all.y

#Split Activity codes to allow for easier data management. VMS Declaration code book is broken down by a
```

```
VTR_DMIS_AC$ACTIVITY_CODE <- as.character(VTR_DMIS_AC$ACTIVITY_CODE)
VTR_DMIS_AC <- VTR_DMIS_AC %>% separate(ACTIVITY_CODE, into = c('Plan Code','Program Code','Area Identi
```

```
## Warning: Expected 4 pieces. Missing pieces filled with `NA` in 158536 rows [1,
## 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
VTR_DMIS_AC$`-` <- NULL




## 3.
### Testing Reported NAs in new data set (that they are relatively even across all years)
#### Note: The variable used in this command can be substituted for whatever needs to be tested. In thi

testing <- VTR_DMIS_AC %>%
group_by("YEAR") %>%
  filter(is.na(OPERNUM))
```

Converting Revenues to Real Dollars

In order to construct operating profits later on we need to ensure that DOLLAR and TRIP_COST_WINSOR_2020_DOL match. The original data set has DOLLAR in nominal dollars and TRIP_COST_WINSOR_2020_DOL is in 2020 Real Dollars. To make the conversion we will use GDP Implicit Price Deflators collected from the Federal Reserve Bank and adjust the index to Q2 2020 to match calculation methods of the TRIP_COST_WINSOR_2020_DOL variable.

```
# Instructions for using the fredr package, a R client for the 'FRED' API, to import Federal Reserve de

## In order to obtain your API key you must create an account and submit a description of this applicat

## Alternatively, you can set a once per session key by using the following command
### fredr_set_key("Your_FRED_API_key_here")



# Test it out
stopifnot(fredr_has_key()==TRUE)
```

```
# In order to convert revenues to match costs which are reflected in Q2 2020 Dollars deflator, values a



deflators <- fredr(
  series_id = "GDPDEF",
  observation_start = as.Date("2007-01-01"),
  observation_end = as.Date("2020-06-01"),
  realtime_start =NULL,
  realtime_end =NULL,
  frequency = "q")

# Assign Quarters

deflators <- deflators %>%
          dplyr::select(date,value) %>%
          mutate(date = lubridate::quarter(date,
                         type = "quarter",
                         fiscal_start = 1,
                         with_year = TRUE))
```

```
# The default index year for the GDP Implicit Price Deflator in United States data is 2015 which needs
## Index quarter/year can be updated by changing "2020.2" to desired quarter and year combination

assign("base_year_index",deflators[deflators$date == "2020.2","value"])
base_year_index <- as.numeric(base_year_index)

deflators <- deflators %>%
            mutate(value = value/(base_year_index))

# Break down current dates into years and quarters
VTR_DMIS_AC <- VTR_DMIS_AC %>%
                  mutate(qdate = lubridate::quarter(Date,
                                  type = "quarter",
                                  fiscal_start = 1,
                                  with_year = TRUE)) %>%
                                  left_join(deflators, by =c("qdate"="date")) %>%
               mutate(DOLLAR_ALL_SP_2020=DOLLAR_ALL_SP/value)


# Delete unnecessary data frame and variable
deflators <- NULL
VTR_DMIS_AC$qdate <- NULL
```

Data Aggregating Trip Revenues & Delete duplicate TRIPIDs

Subtrips are generated when a vessel switches gear or statistical areas. Subtrips have identical TRIPID/DOCID. A trip may have many (8+) subtrips, but the majority of trips observed only have one subtrip (95.7% using original VTR & DMIS merged data set). If a trip has just 1 subtrip, the trip took place in a single statistical area. If a trip crosses four different statistical areas, the NSUBTRIP is then equal to 4, and the landings, value, latitude, and longitude are reported separately for each area.

```
table(VTR_DMIS_AC$NSUBTRIP)
```

```
##
##      1      2      3      4      5      6      7      8
## 165486   2890    631    229     79      2      7      1
```

Since our goal is to estimate a choice model at the trip level, we need to construct trip level variables. We retained the subtrip attributes (GEARCODE, DDLAT, DDLON) corresponding to the subtrip with the highest DOLLAR. We constructed trip-level values for revenue, pounds, and landed (DOLLAR, POUNDS, LANDED). The trip level variables are prefixed with "Agg_".

1. Aggregate DOLLAR, POUNDS, LANDED
2. Add back into original data set
3. Check / Test Maximum DOLLAR values by grouping by TRIPID
4. Drop duplicate TRIPIDs by keeping maximum DOLLAR values

This may not be realistic. There are anecdotes of vessels fishing in one spot on the way to another, further offshore spot. Subtrips may be a bigger issue when we extend to other fisheries.If we have the ability to model fishing choices at a finer scale than at a trip, this can be modified fairly easily.

DEPRECATED - Code to aggregate subtrip landings to subtrips.

We are now pulling subtrips along to the end instead of aggregating. If you want to contract multi-area or multi-gear trips down to a single observation, this is how you would do it.

```
### 1. Aggregate DOLLAR, POUNDS, LANDED
Agg_DOL_POUN_LAND <- VTR_DMIS_AC %>%
  group_by(TRIPID) %>%
 summarise(Agg_DOLLAR = sum(DOLLAR), Agg_POUNDS = sum(POUNDS), Agg_LANDED = sum(LANDED))

#### Testing to make sure there are no duplicates in TRIPID groups; this should equal 0
sum(duplicated(Agg_DOL_POUN_LAND$TRIPID))
stopifnot(sum(duplicated(Agg_DOL_POUN_LAND$TRIPID))==0)

###  2. Add back into original data set
#### all = FALSE is used to keep only rows that match from the data frames
VTR_DMIS_AC_Agg <- merge(VTR_DMIS_AC,Agg_DOL_POUN_LAND, by.x = "TRIPID", by.y = "TRIPID", all.x = TRUE,


### 3.Parse out Maximum Scallop Dollar amounts in order to drop lesser subtrips
VTR_DMIS_AC_Agg <- VTR_DMIS_AC_Agg %>%
  group_by(TRIPID) %>%
  filter(DOLLAR == max(DOLLAR))


### Another way to check this is by running the following code: VTR_DMIS_AC_Agg %>% group_by(TRIPID) %>

## Test out
sum(duplicated(VTR_DMIS_AC_Agg$TRIPID))

stopifnot(sum(duplicated(VTR_DMIS_AC_Agg$TRIPID))==0)
```

Trips reported on land will be dropped from observations

```
#########################################################################################
# change these variables to read in the veslogDMISmerge and what the network path to the shared drive i
coordinate_table_input <-VTR_DMIS_AC
lat_column = "DDLAT"
lon_column = "DDLON"
shapefile_path<-East_Cst_crop_2020_path
#########################################################################################
shapefile_path_to_spatialpolygons <- function(shapefile_path,
                                              projection = CRS("+proj=longlat +datum=NAD83 +no_defs +el

  # shapefile_path = "C:/Users/dennis.corvi/Documents/R/Projects/OffshoreWindDev/offshoreWind/areas_min
  # projection = CRS("+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0")
  layer_name = unique(gsub(pattern="(.+)(.shp$)","\\1", ignore.case = TRUE , list.files(path=shapefile_p
  if (length(layer_name)==0) {
    stop("Shapefile path does not contain a shapefile")
  }
  if (length(layer_name) > 1) {
    file_list <- list.files(shapefile_path, pattern = "*shp$", full.names = TRUE)
    shapefile_list <- lapply(file_list, sf::read_sf)
    all_shapes <- sf::st_as_sf(data.table::rbindlist(shapefile_list))
    all_shapes <- all_shapes[,(names(all_shapes) %in% c("Name"))]
    all_shapes <- sf::as_Spatial(all_shapes, cast = TRUE, IDs = paste0("ID", seq_along(from)))
    all_shapes@data$NAME <- all_shapes@data$Name
    all_shapes@data$Name <- NULL
  } else { # if only one shape
    all_shapes <- rgdal::readOGR(dsn=shapefile_path, layer=layer_name, verbose=F)
  }
  all_shapes <- spTransform(all_shapes, CRS=projection)
  return(all_shapes)
}
##################################################################################

crs = CRS("+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0")
shapefile_area <- SpatialPolygonsDataFrame(aggregate(shapefile_path_to_spatialpolygons(shapefile_path, 
```

```
## Loading required namespace: rgeos
```

```
coordinate_table <- as_tibble(coordinate_table_input %>%
                                rename("LAT" = .data[[lat_column]], "LON" = .data[[lon_column]]) %>%
                                drop_na(LON, LAT) %>%
                                mutate(LON = if_else(LON>1, LON*-1, LON )) %>%
                                relocate(LON, LAT)) # drop LAT LON NAs, correct LON, change column orde

xy <- coordinate_table[,c(1,2)]
coordinate_table <- SpatialPointsDataFrame(coords = xy, data = coordinate_table, proj4string = crs)
coordinate_table <- spTransform(coordinate_table, CRSobj = crs)

vtridx <- over(coordinate_table, shapefile_area)

colnames(vtridx)[1] <- "NAME"

coordinate_table$Area <- vtridx$NAME
coordinate_table <- coordinate_table@data

VTR_DMIS_AC <- coordinate_table %>%
  mutate_if(is.factor, as.character) %>%
  mutate(Area = if_else(is.na(Area), "Non-land", Area)) %>% # change NAs to read "Non-land"
  rename("{lat_column}" := LAT, "{lon_column}" := LON) %>%  # change lat lon columns back to original n
  filter(Area == "Non-land")
#Delete Area Variable; Served its purpose as a filter
VTR_DMIS_AC$Area <- NULL
```

```
# Spatial join with ten minute squares

## Read in your shapefile
### Note: Viewing the table after this is done is helpful to ensure that the shapefile looks how you ex

## Import the data set you want to combine with your imported shape file


TMSQ_sp <- st_read(TMSQ_path)
```

```
## Reading layer `Ten_Minute_Squares_Clip6' from data source
##   '/net/home2/mlee/Effort-Displacement---Scallop/data/external/shapefiles/Ten Minute Squares Cut Nor
##   using driver 'ESRI Shapefile'
## Simple feature collection with 2410 features and 15 fields
## Geometry type: POLYGON
## Dimension:     XYZ
## Bounding box:  xmin: -77.33333 ymin: 35.33333 xmax: -65 ymax: 45.33333
## z_range:       zmin: 0 zmax: 0
## Geodetic CRS:  NAD83
```

```
## Run the below chunk to see your shapefile plotted out
#qtm(TMSQ_sp) + tm_legend(show = FALSE)

# Preserve the DDLAT and DDLON fields
VTR_DMIS_AC$DDLAT_bak<-VTR_DMIS_AC$DDLAT
VTR_DMIS_AC$DDLON_bak<-VTR_DMIS_AC$DDLON



point_geo <- st_as_sf(VTR_DMIS_AC,
                      coords = c(x  = "DDLON", y = "DDLAT"), crs = crs )

final_product <- st_join(point_geo, TMSQ_sp, join = st_within)
```

```
## st_as_s2(): dropping Z and/or M coordinate
```

```
#This chunk uses a "within" join, but other options are available using the sf package 1.0-6.
## st_intersects,st_disjoint,st_touches,st_crosses,st_within,st_contains,st_contains_properly,st_overla

#Delete unnecessary variables from join: keep the geometry, MN30SQID, and MN10SQID columns
final_product[,c('MN10SQROW','MN10SQCOL','POINT_Y','POINT_X','XTXT','YTXT','DG1SQLAT','DG1SQLON','DG1SQ

#final_product$geometry_old<-final_product$geometry
#Lease Area Joins

lease_sp <- st_read(All_Lease_Areas_Shapefile_path)
```

```
## Reading layer 'Github_Lease_Areas_from_BOEM_08_17_2022_Modified' from data source '/net/home2/mlee/E
##   using driver 'ESRI Shapefile'
## Simple feature collection with 26 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -75.90347 ymin: 36.14112 xmax: -70.02155 ymax: 41.2988
## Geodetic CRS:  NAD83
```

```
## Run the below chunk to see your shapefile plotted out
#qtm(lease_sp) + tm_legend(show = FALSE)

## This chunk uses the current data set, converts it into a sf geospatial object, and bins it into the


point_geo_lease <- st_as_sf(final_product,
                      coords = c(x  = "DDLON", y = "DDLAT"), crs = crs )


final_product_lease <- st_join(point_geo_lease, lease_sp, join = st_within)
```

```
#geometry carries over all the way from the initial read in.
identical(final_product_lease$geometry, point_geo$geometry)
```

```
## [1] TRUE
```

```
#This chunk uses a "within" join, but other options are available using the sf package 1.0-6.
## st_intersects,st_disjoint,st_touches,st_crosses,st_within,st_contains,st_contains_properly,st_overla


#Recover the DDLAT and DDLON fields.
colnames(final_product_lease)[colnames(final_product_lease) == "DDLON_bak"] <- "DDLON"
colnames(final_product_lease)[colnames(final_product_lease) == "DDLAT_bak"] <- "DDLAT"

stopifnot(is.numeric(final_product_lease$MN10SQID))

stopifnot(is.numeric(final_product_lease$MN30SQID))

#Delete unnecessary lease number variable and update lease area variable name to match prior analysis
final_product_lease$LEASE_NUMB <- NULL
colnames(final_product_lease)[colnames(final_product_lease) == "LEASE_NU_1"] <- "NAME"




# If you are planning on turning off the next chunk which integrates the FishSET method of zone assignm

#save to RDS
# rds_savename<-paste0("final_product_lease",vintage_string,".Rds")
# csv_savename<-paste0("final_product_lease",vintage_string,".csv")

# saveRDS(final_product_lease, file=here("data","main",rds_savename))
# write.csv(final_product_lease,file=here("data","main",csv_savename), row.names = FALSE)
```

Zone and Lease Assignments

Using the FishSET spatial join method the chunk below will create two new variables, ZoneID and lease_FS. ZoneID will reflect what zone an observation will be binned into after the spatial join. lease_FS contains the name of the wind lease area an observation takes place in using the same join method. Note if an observation is not within a wind lease area the lease_FS value will be NA.

```
#Contents used have come from the "Zone Assignment" Rmd created by Bryce McMagnus using FishSET zone as

## Read in your shapefile
### Note: Viewing the table after this is done is helpful to ensure that the shapefile looks how you ex

# Load in 10 minute squares
tenMinSqr_new <-
  here("data",
       "external",
       "shapefiles",
       "Ten Minute Squares Cut North and Greater Atlantic") %>%
  st_read() %>%
  st_zm() # remove Z/M dimensions from feature
```

```r
# Load in All Lease Areas
lease <-
  here("data",
       "external",
       "shapefiles",
       "All_Lease_Areas_Shapefile") %>%
  st_read() %>%
  st_zm()




## Zone assignment; This is the approach to assign observations to zones.
# create sf version of data, convert to WGS84
# 4326 is shorthand for WGS84 (https://epsg.io/4326)



## Lease Areas

crs <- 4326
final_product_lease_FS <-
  st_as_sf(x = final_product_lease, coords = c("DDLON", "DDLAT"),
           crs = crs)
# convert Squares to WGS84
lease <- st_transform(lease, crs = st_crs(final_product_lease_FS))
# same results as st_within
inter <- sf::st_intersects(final_product_lease_FS, lease)
inter_save <- inter

 if (any(lengths(inter) > 1)) { # if more than one zone intersects, assign to closest zone

  dub <- which(lengths(inter) > 1)
  inter[dub] <- st_nearest_feature(final_product_lease_FS[dub,], lease_FS)
 }
# Add ZoneID column to data
pts <- as.data.frame(as.numeric(inter))
colnames(pts) <- "col.id"
final_product_lease_FS$lease_FS <- lease$NAME[pts$col.id]
final_product_lease_FS$lease_FS <- lease$NAME[as.numeric(inter)]






# Zone Assignments
tenMinSqr_new <- st_transform(tenMinSqr_new, crs = st_crs(final_product_lease_FS))

# same results as st_within
inter <- sf::st_intersects(final_product_lease_FS, tenMinSqr_new)

inter_save <- inter
```

```
 if (any(lengths(inter) > 1)) { # if more than one zone intersects, assign to closest zone

  dub <- which(lengths(inter) > 1)
  inter[dub] <- st_nearest_feature(final_product_lease_FS[dub,], tenMinSqr_new)
 }

# Add ZoneID column to data
pts <- as.data.frame(as.numeric(inter))
colnames(pts) <- "col.id"
pts$ID <- tenMinSqr_new$MN10SQID[pts$col.id]
final_product_lease_FS$ZoneID <- pts$ID

#Delete unnecessary lease number variable and update lease area variable name to match prior analysis
final_product_lease_FS$LEASE_NUMB <- NULL
colnames(final_product_lease_FS)[colnames(final_product_lease_FS) == "LEASE_NU_1"] <- "NAME"


#Delete geometry column & duplicates
final_product_lease_FS$geometry <- NULL
final_product_lease_FS <- final_product_lease_FS %>% distinct(TRIPID,DOLLAR,TRIP_LENGTH,LANDED, .keep_al
final_product_lease <- final_product_lease_FS %>% distinct(TRIPID,DOLLAR,TRIP_LENGTH,LANDED, .keep_all =


#save to RDS
rds_savename<-paste0("final_product_lease_",vintage_string,".Rds")
csv_savename<-paste0("final_product_lease_",vintage_string,".csv")


saveRDS(final_product_lease, file=here("data","main",rds_savename))
write.csv(final_product_lease,file=here("data","main",csv_savename), row.names = FALSE)
```

Some summary statistcs

```
# Here are a few summary statistics tables.  Nothing too fancy. This may be sufficient.

summary(final_product_lease)
```

```
##      TRIPID             OPERATOR            OPERNUM            NSUBTRIP
##  Min.   :2.679e+06   Length:165868      Min.   :  410392   Min.   :1.000
##  1st Qu.:3.157e+06   Class :character   1st Qu.:10002645   1st Qu.:1.000
##  Median :4.020e+06   Mode  :character   Median :10009375   Median :1.000
##  Mean   :6.238e+11                      Mean   :10008984   Mean   :1.031
##  3rd Qu.:4.891e+06                      3rd Qu.:10014818   3rd Qu.:1.000
##  Max.   :4.105e+13                      Max.   :10024074   Max.   :8.000
##                                         NA's   :4370
##       CREW          VTR_PORTNUM         IMGID               YEAR
##  Min.   : 1.000   Min.   : 71011   Min.   :2.468e+06   Length:165868
##  1st Qu.: 3.000   1st Qu.:240403   1st Qu.:2.874e+06   Class :character
##  Median : 3.000   Median :330127   Median :3.755e+06   Mode  :character
##  Mean   : 3.843   Mean   :299650   Mean   :6.228e+13
##  3rd Qu.: 5.000   3rd Qu.:330309   3rd Qu.:4.657e+06
##  Max.   :33.000   Max.   :499101   Max.   :4.105e+15
##  NA's   :138      NA's   :3        NA's   :4
##    DATE_TRIP                      VTR_PORT          VTR_STATE
##  Min.   :2007-05-01 01:00:00   Length:165868      Length:165868
##  1st Qu.:2009-05-27 16:45:00   Class :character   Class :character
##  Median :2012-08-17 04:57:00   Mode  :character   Mode  :character
##  Mean   :2012-12-30 16:53:21
##  3rd Qu.:2016-06-17 11:37:30
##  Max.   :2019-12-31 10:00:00
##
##   TRIP_LENGTH         PERMIT.y        DEALNUM              DOLLAR
##  Min.   : 0.0000   Min.   :110681   Length:165868      Min.   :      0.5
##  1st Qu.: 0.5938   1st Qu.:231428   Class :character   1st Qu.:   2141.0
##  Median : 0.9167   Median :310979   Mode  :character   Median :   3928.0
##  Mean   : 2.6138   Mean   :285436                      Mean   :  35550.8
##  3rd Qu.: 2.7083   3rd Qu.:330784                      3rd Qu.:   8697.7
##  Max.   :24.7500   Max.   :550026                      Max.   :1413380.0
##
##  DOLLAR_ALL_SP           POUNDS              LANDED            GEARCODE
##  Min.   :      1.6   Min.   :      0.5   Min.   :     0.29   Length:165868
##  1st Qu.:   2445.0   1st Qu.:   2069.9   1st Qu.:   250.00   Class :character
##  Median :   4809.0   Median :   3332.0   Median :   400.00   Mode  :character
##  Mean   :  37718.8   Mean   :  31911.1   Mean   :  3834.71
##  3rd Qu.:  11952.0   3rd Qu.:   6147.0   3rd Qu.:   750.00
##  Max.   :1413380.0   Max.   :1186125.0   Max.   :142392.00
##  NA's   :60
##  SECGEARFISH          SPPNAME              geoid             namelsad
##  Length:165868      Length:165868      Min.   :9.008e+08   Length:165868
##  Class :character   Class :character   1st Qu.:2.501e+09   Class :character
##  Mode  :character   Mode  :character   Median :3.401e+09   Mode  :character
##                                        Mean   :3.106e+09
##                                        3rd Qu.:3.403e+09
##                                        Max.   :5.170e+09
##                                        NA's   :1774
##    state_fips       port_lat        port_lon      previous_namelsad
##  Min.   : 7.00   Min.   :34.71   Min.   :-76.86   Length:165868
##  1st Qu.:24.00   1st Qu.:39.57   1st Qu.:-74.23   Class :character
##  Median :33.00   Median :40.87   Median :-72.52   Mode  :character
##  Mean   :29.92   Mean   :40.60   Mean   :-72.64
##  3rd Qu.:33.00   3rd Qu.:41.64   3rd Qu.:-70.93
##  Max.   :49.00   Max.   :44.95   Max.   :-66.98
##  NA's   :1643    NA's   :1774    NA's   :1774
##  previous_state_fips previous_geoid      previous_port_lat previous_port_lon
##  Min.   : 7.00       Min.   :9.008e+08   Min.   :34.71     Min.   :-76.86
##  1st Qu.:24.00       1st Qu.:2.501e+09   1st Qu.:39.57     1st Qu.:-74.23
##  Median :33.00       Median :3.401e+09   Median :40.87     Median :-72.52
##  Mean   :29.96       Mean   :3.110e+09   Mean   :40.59     Mean   :-72.65
##  3rd Qu.:33.00       3rd Qu.:3.403e+09   3rd Qu.:41.64     3rd Qu.:-70.93
##  Max.   :49.00       Max.   :5.181e+09   Max.   :44.95     Max.   :-66.98
##  NA's   :1749        NA's   :1882        NA's   :1882      NA's   :1882
##      Date               Time           scallop_fishing_year   TRIP_ID
##  Length:165868      Length:165868      Length:165868        Length:165868
##  Class :character   Class :character   Class :character     Class :character
##  Mode  :character   Mode  :character   Mode  :character     Mode  :character
```

```
##
##
##
##
##   Plan Code          Program Code       Area Identifier        ftpt
##  Length:165868      Length:165868      Length:165868      Length:165868
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##      GC              LA              hours          DB_LANDING_YEAR
##  Mode :logical   Mode :logical   Min.   :  0.0333   Min.   :2007
##  FALSE:34428     FALSE:117141    1st Qu.: 14.2500   1st Qu.:2009
##  TRUE :131381    TRUE :48727     Median : 22.0000   Median :2012
##  NA's :59                        Mean   : 62.7184   Mean   :2013
##                                  3rd Qu.: 65.0000   3rd Qu.:2016
##                                  Max.   :594.0000   Max.   :2019
##                                  NA's   :1          NA's   :1
##  TRIP_COST_2020_DOL TRIP_COST_WINSOR_2020_DOL OBSERVED_COST_DUMMY
##  Min.   :   16.78   Min.   :   29.47          Min.   :0.00000
##  1st Qu.:  637.13   1st Qu.:  637.13          1st Qu.:0.00000
##  Median : 1223.95   Median : 1223.95          Median :0.00000
##  Mean   : 4742.65   Mean   : 4699.89          Mean   :0.04871
##  3rd Qu.: 5377.22   3rd Qu.: 5376.89          3rd Qu.:0.00000
##  Max.   :52122.12   Max.   :30595.61          Max.   :1.00000
##  NA's   :1          NA's   :1                 NA's   :1
##      value        DOLLAR_ALL_SP_2020      DDLAT           DDLON
##  Min.   :0.8192   Min.   :      1.7   Min.   :36.00   Min.   :-75.92
##  1st Qu.:0.8414   1st Qu.:   2883.9   1st Qu.:39.38   1st Qu.:-73.53
##  Median :0.8881   Median :   5171.7   Median :40.20   Median :-72.83
##  Mean   :0.8947   Mean   :  41735.7   Mean   :40.34   Mean   :-71.95
##  3rd Qu.:0.9360   3rd Qu.:  13254.7   3rd Qu.:41.09   3rd Qu.:-70.17
##  Max.   :1.0011   Max.   :1617857.4   Max.   :44.77   Max.   :-66.03
##                   NA's   :60
##     MN30SQID        MN10SQID          NAME               ZoneID
##  Min.   :35734   Min.   :357311   Length:165868      Min.   :357311
##  1st Qu.:39731   1st Qu.:397331   Class :character   1st Qu.:397331
##  Median :40714   Median :407121   Mode  :character   Median :407121
##  Mean   :40560   Mean   :405618                      Mean   :405618
##  3rd Qu.:41691   3rd Qu.:416922                      3rd Qu.:416922
##  Max.   :44691   Max.   :446966                      Max.   :446966
##
```

```
table(final_product_lease$YEAR)
```

```
##
##  2007  2008  2009  2010  2011  2012  2013  2014  2015  2016  2017  2018  2019
## 15905 18396 16483 11308 12546 12082 10970 10022 10728 12750 10922 12152 11604
```

```
table(final_product_lease$GEARCODE)
```

```
##
##     DREDGE-CLAM    DREDGE-OTHER  DREDGE-SCALLOP   GILLNET-OTHER    GILLNET-SINK
##           7996              35          135051               1              98
##        HANDLINE LONGLINE-BOTTOM           OTHER       POT-OTHER     SEINE-OTHER
##              99               3               2              94               1
##    TRAWL-BOTTOM
##           22488
```

```
table(final_product_lease$ftpt)
```

```
##
## FullTime     None PartTime
##    42465   117141     6262
```

```
table(final_product_lease$VTR_STATE)
```

```
##
##    CT    DE    MA    MD    ME    NC    NH    NJ    NY    RI    VA
##  2371   196 58486  6162  5319   453  1831 63180 13779  7514  6536
```

```
table(final_product_lease$`Plan Code`)
```

```
##
##    DOF    HER    MID    MNK    NMS    SCO    SES    SMB
##   3459      2     12    431   5644   7280 139898    222
```

```
table(final_product_lease$`Program Code`)
```

```
##
##    BDP   CML   COM   DOF   HER   MMQ   MNK   MUL   NAC   NAF   NAS   NMA   OQU
##      6  1221   324  2190     2    54   161  1944     1     1    15     6  4462
##    PWD   REC   RSA   SAA   SAC   SAM   SAS   SCA   SCF   SCG   SCI   SEC   SFC
##    264     5   557 22496     1    13    67 18086     1 98498     4  2509  2764
##    SLM   SMA   SQI   SQL   SQM   SWE   TSP   TST   USC
##     25   327     4   134    59    11     3    35   698
```

R Session Information

```
sessionInfo()
```

```
## R version 4.0.5 (2021-03-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Red Hat Enterprise Linux 8.6 (Ootpa)
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib64/libopenblasp-r0.3.15.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
```

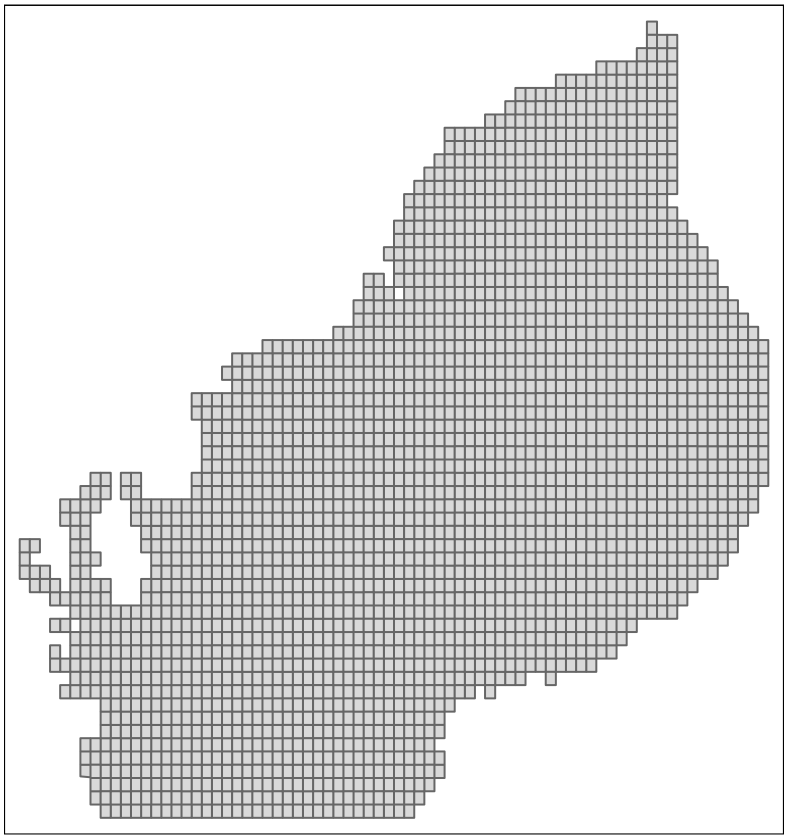

Figure 1: 10 minutes squares

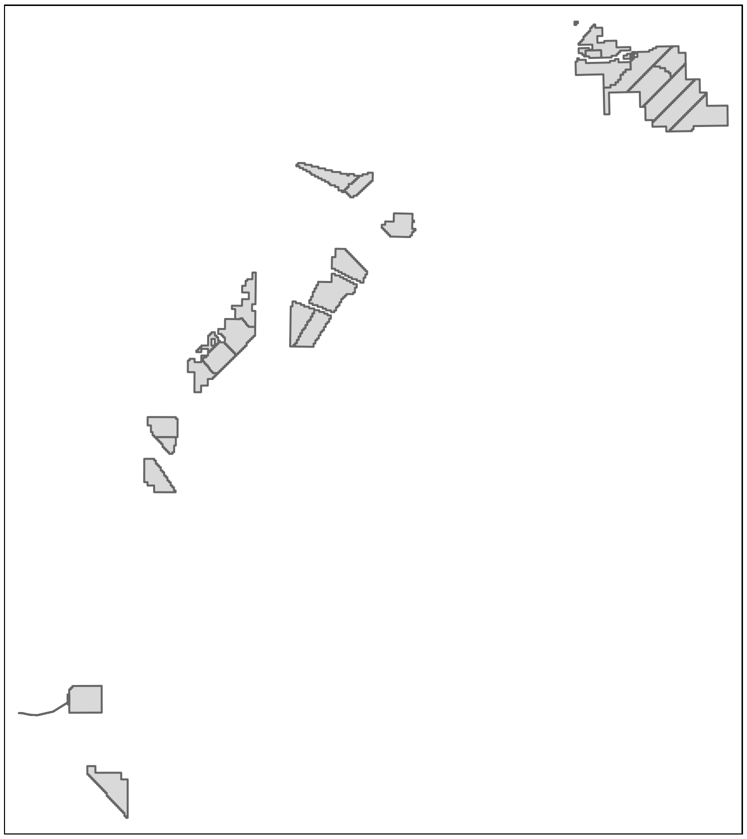

Figure 2: Wind Energy Areas

```
##
## other attached packages:
##  [1] fredr_2.1.0         epiDisplay_3.5.0.1  nnet_7.3-16
##  [4] MASS_7.3-54         survival_3.2-13     foreign_0.8-81
##  [7] RODBC_1.3-19        tmap_3.3-3          tmaptools_3.1-1
## [10] data.table_1.14.2   rgdal_1.5-27        raster_3.5-2
## [13] sp_1.4-6            sf_1.0-4            xml2_1.3.3
## [16] tidyr_1.1.4         tibble_3.1.6        stringr_1.4.0
## [19] rvest_1.0.2         rstudioapi_0.13     rlang_1.0.2
## [22] reprex_2.0.1        readxl_1.3.1        readr_2.1.1
## [25] purrr_0.3.4         pillar_1.7.0        modelr_0.1.8
## [28] magrittr_2.0.3      lubridate_1.8.0     jsonlite_1.7.2
## [31] httr_1.4.2          hms_1.1.1           googlesheets4_1.0.0
## [34] googledrive_2.0.0   ggplot2_3.3.5       forcats_0.5.1
## [37] dtplyr_1.2.1        dplyr_1.0.9         dbplyr_2.1.1
## [40] crayon_1.5.1        cli_3.3.0           broom_0.7.10
## [43] leaflet_2.0.4.1     here_1.0.1
##
## loaded via a namespace (and not attached):
##  [1] fs_1.5.2                RColorBrewer_1.1-3      rprojroot_2.0.2
##  [4] tools_4.0.5             backports_1.4.0         utf8_1.2.2
##  [7] R6_2.5.1                KernSmooth_2.23-20      rgeos_0.5-8
## [10] DBI_1.1.1               colorspace_2.0-3        withr_2.5.0
## [13] tidyselect_1.1.1        curl_4.3.2              compiler_4.0.5
## [16] leafem_0.1.6            scales_1.2.0            classInt_0.4-3
## [19] proxy_0.4-26            digest_0.6.29           rmarkdown_2.11
## [22] base64enc_0.1-3         dichromat_2.0-0         pkgconfig_2.0.3
## [25] htmltools_0.5.2         highr_0.9               fastmap_1.1.0
## [28] htmlwidgets_1.5.4       generics_0.1.2          crosstalk_1.2.0
## [31] s2_1.0.7                Matrix_1.3-4            Rcpp_1.0.7
## [34] munsell_0.5.0           fansi_1.0.3             abind_1.4-5
## [37] lifecycle_1.0.1         terra_1.4-22            stringi_1.7.6
## [40] leafsync_0.1.0          yaml_2.2.1              grid_4.0.5
## [43] parallel_4.0.5          lattice_0.20-45         splines_4.0.5
## [46] stars_0.5-4             knitr_1.36              codetools_0.2-18
## [49] wk_0.5.0                XML_3.99-0.8            glue_1.6.2
## [52] evaluate_0.14           leaflet.providers_1.9.0 vctrs_0.4.1
## [55] png_0.1-7               tzdb_0.2.0              cellranger_1.1.0
## [58] gtable_0.3.0            assertthat_0.2.1        xfun_0.31
## [61] lwgeom_0.2-8            e1071_1.7-9             class_7.3-19
## [64] viridisLite_0.4.0       gargle_1.2.0            units_0.7-2
## [67] ellipsis_0.3.2
```

```
Sys.Date()
```

```
## [1] "2022-08-22"
```

This may be useful for diagnosing and troubleshooting one day.

Here is some code that we are no longer using.

Code to filter on the Limited Access (LA) Fleet using landings and crew size

We considered filtering out the LADAS scallop fleet by using landings greater than or equal to 850 pounds and Crew less than or equal to 8. These are based on crew limits. We are using the activities codes instead. In summary:

FY2007-2014: No limit on crew (except for 7 in DMV starting in FY2014) FY2015-2019: 8

> Initially, vessels had the same crew limits in access areas as they did on DAS. However, Framework 18(fishing year 2006) eliminated the seven-person crew limit (five-person limit for small dredge category vessels) for scallop access area trips. The purpose of this was to eliminate inefficiencies caused by the crew limit for fishing activity that is limited by a possession limit. The crew limit was established to control vessels' shucking capacity when fishing under DAS.
>
> Eight years later, Framework 25 (fishing year 2014) imposed a crew limit of seven individuals (the same as DAS) per limited access vessel (five-person limit for small dredge category vessels) in DMV. The purpose of this was to protect small scallops and discourage vessels from highgrading.
>
> Framework 26 (fishing year 2015) implemented crew limits for all access areas. In an effort to protect small scallops and discourage vessels from high-grading. Framework 26 imposed a crew limit of eight individuals (one extra from DAS) per limited access vessel, including the captain, when fishing in any scallop access area. If a vessel is participating in the small dredge program, it may not have more than six people (one extra from DAS) on board, including the captain, on an access area trip.
>
> Finally, because the scallops in the NLS–S–D were expected to have lower yield than similar sized scallops in other areas, Framework 32 (fishing year 2020) allowed two additional crew members aboard both limited access full-time (10 in total) and limited access full-time small dredge vessels (8 in total). This allowed vessels to add additional crew members to increase the shucking capacity of the vessel and reach the possession limit in a time more consistent with other access areas. (Travis Ford @ GARFO - Nov 17,2021)

FY2007-2014: No limit (except for 7 in DMV starting in FY2014) FY2015-2019: 8

```r
LA_Estimate <- VTR_DMIS_AC_Agg %>% filter(Agg_LANDED >= 850 & CREW <= 8)
```