TFM: Análisis predictivo de incidentes navales en EEUU, 2002 - 2015

Anexo 3.1. Preprocesado All Casualty & Pollution

Oscar Antón

diciembre de 2023

Carga de librerías y funciones personalizadas

```
# Librería                          # Propósito
library(tictoc)                     # Monitorización de tiempo de cómputo
library(data.table)                 # Manejo eficiente de conjuntos de datos
library(tidyverse)                  # Sintaxis para el manejo de datos. Incluye dplyr, ggplo
t2, etc.

library(leaflet)                    # Representación geográfica
library(sf)                         # Simple Features. Manejo de coordenadas geoespaciales
library(rnaturalearth)              # Obtención de datos geográficos

source("../4.Functions/myCustomFunctions.R")
```

Switches

```
# Guardar datos o no
save_switch <- 0
# Utilizado para no tener que procesar todos los datos en cada renderizado
```

---

1. Adquisición de datos

1.1. Descarga de los datos anuales de USCG

```
if (save_switch  == 1) {
  # url donde están los datos
  url_descarga <- "https://www.dco.uscg.mil/Portals/9/DCO%20Documents/5p/CG-5PC/INV/docs/MISLE%20DATA.zip?ver=2017-10-30-120914-487"

  # Descarga el archivo
  download.file(url_descarga, destfile = "RawDataAllCasualtyAndPollution")
}
```

1.2. Descompresión de datos

```
if (save_switch  == 1) {
  # Nombres de las variables
  VariableNames <- read.delim("RawDataAllCasualtyAndPollution/VariableNames.txt", sep = "\t", quote = "", stringsAsFactors = FALSE, fileEncoding = "ISO-8859-1")

  # Define carpeta de origen
  carpeta_origen <- "RawDataAllCasualtyAndPollution"

  # Verifica si la carpeta local existe, y si no, la crea
  if (!dir.exists(carpeta_origen)) {
    dir.create(carpeta_origen, recursive = TRUE)
  }

  # Obtener la lista de archivos que empiezan con "Misle" en la carpeta
  file_list <- list.files(carpeta_origen, pattern = "^Misle.*\\.txt$", full.names = TRUE)

  # Guardamos la salida para documentar el proceso
  sink(paste0(carpeta_origen, "/", "descripciondatatables.txt"), append = TRUE)

  # Cargar los archivos como dataframes en una lista
  dataframes_list <- lapply(file_list, function(file) {
    # Nombre de archivo
    df_name <- gsub(".*/(.*?)\\.txt.*", "\\1", file)
    tic()
    # Lectura del txt
    df <- read.delim(file, header = TRUE, sep = "\t", quote = "", stringsAsFactors = FALSE, fileEncoding = "ISO-8859-1",
                     col.names = VariableNames$Name[VariableNames$Table == df_name])

    # Exportación y monitorización de datos
    loggedsave(df_name, carpeta_origen)

    })
}
```

2. Acotación de datos

```
# Define carpeta de destino
carpeta_destino <- "DataCasualtyAndPollution"

# Verifica si la carpeta local existe, y si no, se crea
if (!dir.exists(carpeta_destino)) {
  dir.create(carpeta_destino, recursive = TRUE)
}
```

· Se establecerá un cuadro delimitador (bounding box) norteamericano: latitud 15º, 70º y longitud -45º, -180º.

· Se descartarán variables sin interés para el análisis.

· Se dejarán registros únicos en función de las variables identificativas.

2.1. VslEvents

```
tic()

VslEvents <- readRDS("RawDataAllCasualtyAndPollution/MisleVslEvents.rds") %>%

  mutate(date = as.Date(timeline_dt), hour = substr(timeline_dt, 12, 16)) %>%

  # Supresión de variables sin relevancia para el estudio
  select(-timeline_dt, -case_id, -fk_d_vessel,
         -vessel_service, -vessel_type, -vessel_subtype,
         -event_class, -event_subclass
         ) %>%
  # Acotación geográfica y por fecha
  filter(between(latitude, 15, 70),
         between(longitude, -180, -45),
         between(date, as.Date("2002-01-01"), as.Date("2015-12-31"))
         ) %>%
  # Filtro de duplicidades, solo por variables identificativas
  distinct(activity_id, vessel_id, event_type, .keep_all = TRUE)

# Exportación y monitorización de datos
if (save_switch  == 1) {
  loggedsave(VslEvents, carpeta_destino)
}
```

2.2. Vessel

```
tic()

Vessel <- readRDS("RawDataAllCasualtyAndPollution/MisleVessel.rds") %>%

  # Supresión de variables sin relevancia para el estudio
  select(-gk_d_vessel,
         -managing_owner_id, -managing_owner,
         -net_ton, -itc_breadth, -itc_depth, -itc_gross_ton, -itc_length, -itc_net_ton,
         -draft_design, -draft_design_units, -deadweighttonnage_units, -hailing_port, -hai
ling_port_state, -hailing_port_province,
         -route_type, -cargo_authorization_type, -documented_ind, -documented_status_type,
-inspected_ind, -inspected_desc, -state_vessel_ind, -state_vessel_desc,
         -lloyds_ind, -lloyds_desc, -solas_ind, -insp_subchapter_type,
         -vessel_type, -vessel_subtype, -vessel_service,
         -max_passengers_allowed, -max_crew, -self_propelled_ind,
         -call_sign, -official_number, -hull_number, -rbs_hull_number, -vessel_age,
         -hull_build_party_name, -completed_by_party_name,
         -filler
         ) %>%

  # Filtro de duplicidades, solo por variables identificativas
  distinct(vessel_id, vessel_name, .keep_all = TRUE)

# Exportación y monitorización de datos
if (save_switch  == 1) {
  loggedsave(Vessel, carpeta_destino)
}
```

2.3. Injury

```
tic()

Injury <- readRDS("RawDataAllCasualtyAndPollution/MisleInjury.rds") %>%
  # Se eliminan las variables sin relevancia para el estudio
  select(-fk_d_vessel,
         -vessel_service, -vessel_type, -vessel_subtype,
         -facility_id, -facility_name, -facility_type_desc, -facility_activity_role_desc
         ) %>%

  filter(between(latitude, 15, 70),
         between(longitude, -180, -45)) %>%

  # Filtro de duplicidades, solo por variables identificativas
  distinct(activity_id, vessel_id, .keep_all = TRUE)

# Exportación y monitorización de datos
if (save_switch  == 1) {
  loggedsave(Injury, carpeta_destino)
}
```

2.4. VslPoll

```
tic()
# Se eliminan las variables sin relevancia para el estudio
VslPoll <- readRDS("RawDataAllCasualtyAndPollution/MisleVslPoll.rds") %>%
  # Supresión de variables sin relevancia para el estudio
  select(-case_id, -fk_d_vessel,
         -vessel_service, -vessel_type, -vessel_subtype,
          -substance_name, -substance_class, -substance_subclass, -substance_type, -substan
ce_subtype,
          -discharge_amnt_water, -discharge_amnt_land, -discharge_amnt_air, -discharge_amnt
_enclosed,
          -potential_amnt_total, -potential_amnt_water, -potential_amnt_land, -potential_am
nt_air, -potential_amnt_enclosed, -contained_amnt,
         -discharge_potential_type, -discharge_situation_type,
          -discharge_estimated_land, -discharge_estimated_air, -discharge_estimated_water,
-discharge_estimated_encl,
         -potential_case, -potential_estimated, -contained_estimated, -unit_of_measure
         ) %>%

  filter(between(latitude, 15, 70),
         between(longitude, -180, -45)) %>%

  # Filtro de duplicidades, solo por variables identificativas
  distinct(activity_id, vessel_id, .keep_all = TRUE)

# Exportación y monitorización de datos
if (save_switch  == 1) {
  loggedsave(VslPoll, carpeta_destino)
}
```

2.5. Activity

```
tic()
# Se eliminan las variables sin relevancia para el estudio
Activity <- readRDS("RawDataAllCasualtyAndPollution/MisleActivity.rds") %>%

  mutate(date = as.Date(incident_dt, format = "%m/%d/%Y")) %>%
  mutate(damage_assessment = vessel_property_damage + cargo_property_damage + facility_pro
perty_damage + other_property_damage) %>%
  # Supresión de variables sin relevancia para el estudio
  select(-case_id, -incident_dt,
         -dept_name, -activity_type, -activity_status, -activity_status_subtype,
          -vessel_property_damage, -cargo_property_damage, -facility_property_damage, -othe
r_property_damage,
         ) %>%

  filter(between(date, as.Date("2002-01-01"), as.Date("2015-12-31"))) %>%

  # Filtro de duplicidades, solo por variables identificativas
  distinct(activity_id, date, .keep_all = TRUE)

# Exportación y monitorización de datos
if (save_switch  == 1) {
  loggedsave(Activity, carpeta_destino)
}
```

---

3. Eventos: Clasificación geográfica

```
# Lectura de datos
Events <- as.data.table(readRDS("DataCasualtyAndPollution/VslEvents.rds"))
```

3.1. Region

```
# Establecimiento de la variable region en función de coordenadas
Events <- Events %>%
  mutate(region = case_when(
    between(latitude, 49, 70) & between(longitude, -180, -122) ~ "Alaska",
    between(latitude, 49, 70) & between(longitude, -122, -45) ~ "Canada",
    between(latitude, 15, 49) & between(longitude, -81.5, -45) ~ "East Coast",
    between(latitude, 15, 49) & between(longitude, -180, -100) ~ "West Coast",
    between(latitude, 15, 31) & between(longitude, -100, -81.5) ~ "Gulf of Mexico",
    between(latitude, 31, 49) & between(longitude, -100, -81.5) ~ "Mississippi",
    TRUE ~ "Otra Zona"
  ))
```

3.2. Watertype: Rio / Mar

```
# Obtener datos geoespaciales del área continental: Estados Unidos, México y Canada
us_states <- ne_states(country = "United States of America", returnclass = "sf")
canada <- ne_states(country = "Canada", returnclass = "sf")
mexico <- ne_states(country = "Mexico", returnclass = "sf")

continental <- rbind(us_states, mexico, canada)

# Convertir el dataframe a un objeto sf
datos_sf <- st_as_sf(Events[, c("activity_id", "longitude", "latitude")], coords = c("long
itude", "latitude"), crs = st_crs(us_states))

# Función st_within para calcular los puntos que estén dentro del conjunto comparativo
# (Asigna el número de estado o Integrer(0) = Fuera)
datos_sf$watertype <- st_within(datos_sf, continental)

# Renombrado de valores: Ocean para 0, River para lo demás
datos_sf$watertype <- if_else(is.na(as.numeric(datos_sf$watertype)), "ocean", "river")

# Se añade al dataframe VslEvents
Events$watertype <- datos_sf$watertype
```

3.3. Acotación por tipo de medio (Oceano / Rio)

Para acotar el análisis, se tomarán los datos de incidentes ocurridos en:

· Océanos norteamericanos

· Cuenca fluvial de Mississippi

```
# Se prescinde de los incidentes en río que no sean en la zona Mississipi
Events <- Events %>%
  filter(watertype != "river" | region == "Mississippi") %>%
  filter(!is.na(vessel_id))
```

3.4. Guardado de datos de eventos

```
# Define carpeta de destino
carpeta_destino <- "DataCasualtyAndPollution"

# Exportación y monitorización de datos
if (save_switch  == 1) {
  loggedsave(Events, carpeta_destino)
}
```

4. Lesiones

```
# Cargar el dataframe de Lesiones
Injuries <- as.data.table(readRDS("DataCasualtyAndPollution/Injury.rds"))
```

---

5. Contaminación

```
# Cargar el dataframe de Contaminación
Pollution <- as.data.table(readRDS("DataCasualtyAndPollution/VslPoll.rds"))
```

---

6. Valorización

```
Activity <- as.data.table(readRDS("RawDataAllCasualtyAndPollution/MisleActivity.rds")) %>%
  mutate(total_amount = vessel_property_damage + cargo_property_damage + facility_property
_damage + other_property_damage)
```
