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

Anexo 3.3. Preprocesado Weather River

Oscar Antón

diciembre de 2023

Carga de librerías

```
# Librería                          # Propósito
library(tictoc)                     # Monitorización de tiempo de cómputo
library(progress)                   # Monitorización de tiempo de cómputo

library(leaflet)                    # Visualización geográfica

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

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 la NOAA

National Oceanic and Atmospheric Administration de Estados Unidos

```r
if (save_switch  == 1) {
  # URLbase donde están los archivos
  base_url <- "https://www.ncei.noaa.gov/pub/data/ghcn/daily/by_year"
  # Carpeta de destino
  carpeta_local <- "RawDataWeatherRiver"

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

  # Crea una secuencia de años y meses desde 2002 hasta 2015
  anos <- 2002:2015

  # Función para descargar un archivo
  descargar_archivo <- function(ano, base_url, carpeta_local) {
    # Formatea el nombre del archivo
    nombre_archivo <- sprintf("%d.csv.gz", ano)

    # Combina la URL base con el nombre del archivo
    url_descarga <- paste0(base_url, "/", nombre_archivo)

    # Combina la ruta local con el nombre del archivo
    ruta_local <- file.path(carpeta_local, nombre_archivo)

    # Descarga el archivo
    download.file(url_descarga, destfile = ruta_local)

    # Retorna el nombre del archivo descargado
    return(nombre_archivo)
  }

  # Descarga los archivos utilizando mapply
  resultado_descargas <- mapply(descargar_archivo, anos, MoreArgs = list(base_url, carpeta
_local))
}
```

1.2. Descompresión y selección de datos anuales

Al guardar los data.tables

· Selección de variables: STATION, DATE, ELEMENT, DATAVALUE

· Selección de datos meteorológicos: PRCP, TMAX, TMIN

· Formato ancho: ELEMENT -> DATAVALUE

```
carpeta_origen <- "RawDataWeatherRiver"

carpeta_destino <- "DataWeatherRiver"

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

if (save_switch  == 1) {
  # Definimos una función para cargar y seleccionar variables de un archivo
  carga_seleccion <- function(file_path) {
    tic()
    # Cargamos el archivo CSV comprimido
    dt <- fread(cmd = paste("gzip -dc", file_path), select = 1:4, header = FALSE)

    # Establecemos nombres de las variables
    setnames(dt, c("STATION", "DATE", "ELEMENT", "DATAVALUE"))

    # Convertimos a formato ancho
    dt <- dt %>%
          filter(ELEMENT %in% c("PRCP", "TMAX", "TMIN", "AWND")) %>%
          pivot_wider(names_from = ELEMENT, values_from = DATAVALUE) %>%
          mutate(DATE = as.Date(as.character(DATE), format = "%Y%m%d"))

    return(dt)
  }

  # Obtenemos la lista de archivos tar.gz en el directorio
  archivos <- list.files(path = carpeta_origen, pattern = "\\.csv\\.gz$", full.names = TRU
E)

  # Creación de una lista de data.tables cargadas y seleccionadas
  data_list <- lapply(archivos, carga_seleccion)
}
```

---

2. Agregación de datos en un data.frame conjunto

2.1. Conjunto de datos único

```
if (save_switch  == 1) {
  tic()
  # Combina todas las data.tables en una única data.table
  estaciones_terrestres_combinado_1 <- rbindlist(data_list, use.names = TRUE)

  # Exportación y monitorización de datos
  loggedsave(estaciones_terrestres_combinado_1, carpeta_destino)
}
```

2.2. Coordenadas

```
estaciones_terrestres_combinado_1 <- readRDS("DataWeatherRiver/estaciones_terrestres_combinado_1.rds")
```

Adquisición de datos de estaciones

```
# Combina la ruta local con el nombre del archivo
ruta_ghcnd_stations <- file.path("RawDataWeatherRiver", "ghcnd_stations.txt")

if (save_switch  == 1) {
  # URLbase donde están los archivos
  url_descarga <- "https://www.ncei.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt"

  # Descarga el archivo
  download.file(url_descarga, destfile = ruta_ghcnd_stations)
}
# Lectura del archivo de ancho fijo
ghcnd_stations <- read.fwf(ruta_ghcnd_stations, widths = c(11, 9, 10), col.names = c("STATION", "LATITUDE", "LONGITUDE"))
```

Fusión de coordenadas geográficas con datos meteorológicos (filtrado geográficamente)

```
if (save_switch  == 1) {
  # Definimos el área de interés (mississippi)
  mississippi <- list(x1 = -100, y1 = 31, x2 = -81.5, y2 = 49)

  tic()
  # Búsqueda y unión de los valores de coordenadas según STATION
  # Se eliminan observaciones sin información meteorológica relevante
  # Se aplicará un filtrado geográfico para quedarnos con el área de interés
   estaciones_terrestres_coord_2 <- estaciones_terrestres_combinado_1[ghcnd_stations, on = "STATION", nomatch = NA][!is.na(TMAX) | !is.na(TMIN) | !is.na(PRCP)][(LONGITUDE >= mississippi$x1 & LONGITUDE <= mississippi$x2 & LATITUDE >= mississippi$y1 & LATITUDE <= mississippi$y2)]

  # Exportación y monitorización de datos
  loggedsave(estaciones_terrestres_coord_2, "DataWeatherRiver")
}
```

Cribado de datos

```
if (save_switch  == 1) {
  # Nos quedaremos solo con el 33% de observaciones con menos valores NA
  tic()
  estaciones_terrestres_crib_3 <- estaciones_terrestres_coord_2 %>%
  arrange(rowSums(is.na(.))) %>%  # Ordenar por cantidad de NA
  slice(1:round(0.33 * n()))  # Seleccionar el  primer 33%

  # Exportación y monitorización de datos
  loggedsave(estaciones_terrestres_crib_3, "DataWeatherRiver")
}
```

---

3. Combinación con Activity_id

```
# Cargar el dataframe de estaciones marítimas
estaciones_terrestres_crib_3 <- as.data.table(readRDS("DataWeatherRiver/estaciones_terrest
res_crib_3.rds"))

# Cargar el dataframe de EventsRiver (Acotado solo a ríos. Registros únicos)
EventsRiver <- as.data.table(readRDS("DataCasualtyAndPollution/Events.rds")) %>%
  filter(watertype == "river") %>%
  select(activity_id, date, longitude, latitude) %>%
  unique() %>%
  rename_all(toupper)

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

```
if (save_switch  == 1) {
  # Utilizar un enfoque de paralelización
  library(future)
  plan("multicore")

  tic()
  # Barra de progreso
  pb <- progress_bar$new(total = nrow(EventsRiver), format = "[:bar] :percent :eta")

  observacion_cercana <- function(incidente) {
    pb$tick()
    # Seleccionar datos correspondientes al Activity_id
    coord_incident <- EventsRiver[ACTIVITY_ID == incidente][1]
    # Seleccionar las observaciones meteorológicas cercanas (+/-2º)
    coord_station <- estaciones_terrestres_crib_3[
      DATE == coord_incident$DATE &
        between(LATITUDE, coord_incident$LATITUDE - 2, coord_incident$LATITUDE + 2) &
        between(LONGITUDE, coord_incident$LONGITUDE - 2, coord_incident$LONGITUDE + 2)
    ]

    # Calcular distancias usando una aproximación
    coord_station[, station_dist := sqrt((LATITUDE - coord_incident$LATITUDE)^2 + (LONGITU
DE - coord_incident$LONGITUDE)^2)]
    # Devolver la observación con la minima distancia
    return(coord_station[order(station_dist)][1][, ACTIVITY_ID := incidente])
  }

  # Aplicar la función a todos los incidentes
  WeatherRiver <- EventsRiver[, purrr::map_df(ACTIVITY_ID, observacion_cercana)]

  # Define carpeta de destino
  carpeta_destino <- "DataWeatherRiver"

  # Exportación y monitorización de datos
  loggedsave(WeatherRiver, carpeta_destino)
}
```

---

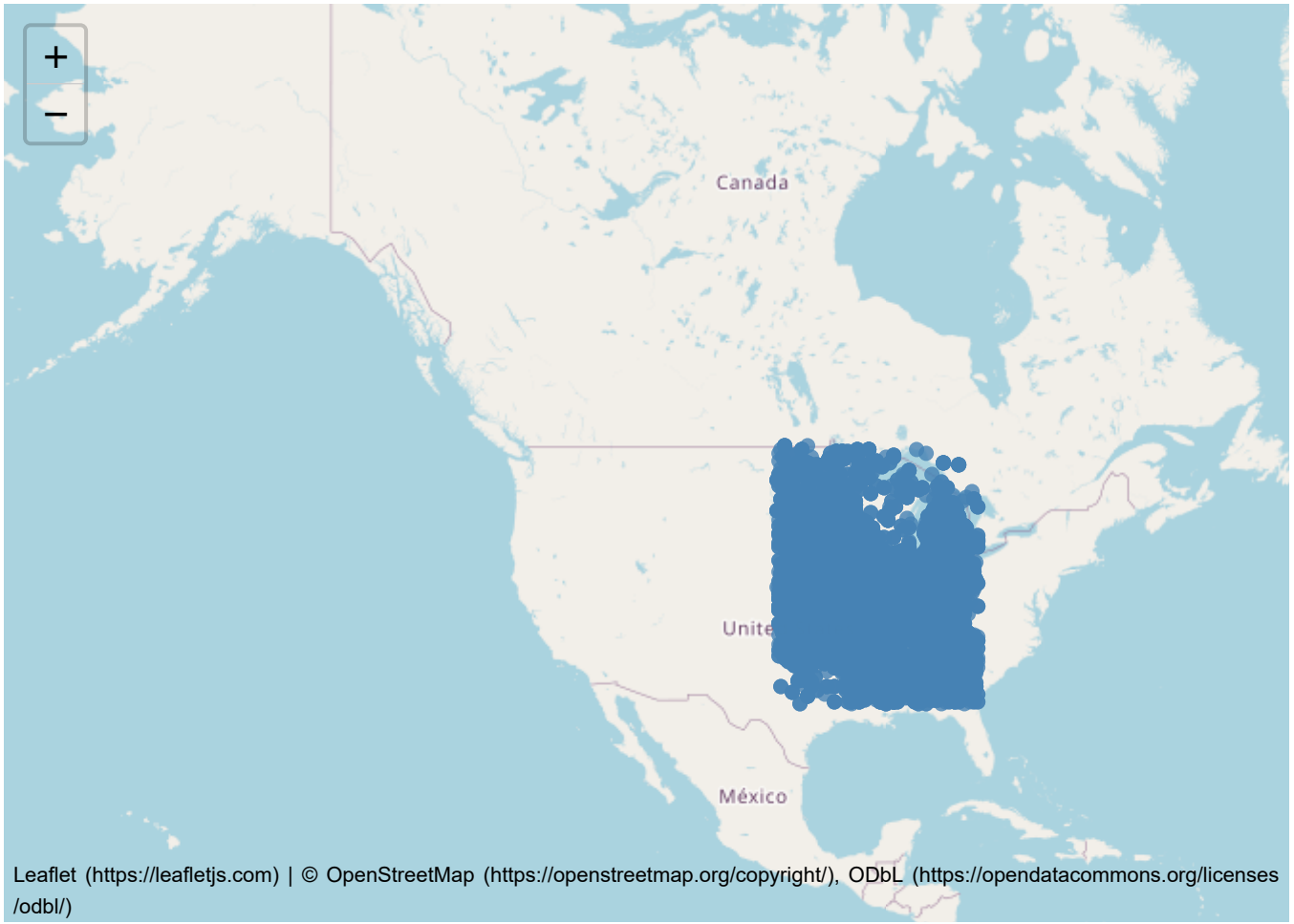
4. Verificación: Representación geográfica

4.1. Datos metereológicos

```
# Cargar el dataframe de estaciones marítimas
estaciones_terrestres_crib_3 <- as.data.table(readRDS("DataWeatherRiver/estaciones_terrest
res_crib_3.rds"))
```

```
# Crear el mapa con una pequeña muestra aleatoria para no saturar la salida
leaflet(estaciones_terrestres_crib_3 %>% sample_frac(0.0005)) %>%
  setView(lng = -112, lat = 48, zoom = 3) %>%
  addTiles() %>%
  addCircleMarkers(
    radius = 4,
    popup=~paste("Station:", STATION, "<br>",
                 "Date:", DATE, "<br>",
                 "longitude:", LONGITUDE, "<br>",
                 "latitude:", LATITUDE, "<br>",
                 "T Max:", TMAX, "<br>",
                 "T Min:", TMIN, "<br>",
                 "Precip:", PRCP, "<br>",
                 "Wind:", AWND, "<br>"
                 ),
    fillOpacity = 0.8,
    color = "steelblue",
    stroke = FALSE
  )
```

```
## Assuming "LONGITUDE" and "LATITUDE" are longitude and latitude, respectively
```



4.2. Datos metereológicos con activity_id

```
# Cargar el dataframe de estaciones terrestres
WeatherRiver <- as.data.table(readRDS("DataWeatherRiver/WeatherRiver.rds"))
```

```
# Crear el mapa con una muestra aleatoria para no saturar la salida
leaflet(WeatherRiver %>% sample_frac(0.01)) %>%
  setView(lng = -112, lat = 48, zoom = 3) %>%
  addTiles() %>%
  addCircleMarkers(
    radius = 4,
    popup=~paste("Station:", STATION, "<br>",
                 "Activity_id:", ACTIVITY_ID, "<br>",
                 "Date:", DATE, "<br>",
                 "longitude:", LONGITUDE, "<br>",
                 "latitude:", LATITUDE, "<br>",
                 "T Max:", TMAX, "<br>",
                 "T Min:", TMIN, "<br>",
                 "Precip:", PRCP, "<br>",
                 "Wind:", AWND, "<br>",
                 "Station dist:", station_dist, "<br>"
                 ),
    fillOpacity = 0.8,
    color = "orange",
    stroke = FALSE
  )
```

```
## Assuming "LONGITUDE" and "LATITUDE" are longitude and latitude, respectively
```