# Predictive analysis of naval incidents in the USA, 2002 - 2015:

## Annex 3.3. Preprocess Weather River

> Author: Oscar Anton
>
> Date: 2024
>
> License: CC BY-NC-ND 4.0 DEED
>
> Version: 0.9

# 0. Loadings

### Libraries

In [1]:

```
# General data management
import numpy as np
import pandas as pd

# File management
import os
import gzip

# Visualization
import plotly.graph_objects as go
import plotly.express as px
```

### General Variables

In [2]:

```
# Main data folders
import_data_folder = 'RawDataWeatherRiver'
export_data_folder = 'DataWeatherRiver'

# Toggle for export data to external file
file_export_enabled = False
# Toggle for calculations that takes a long time
protracted_calculation_enabled = False
```

# 1. Data Acquisition

## 1.1. Decompress and data concatenation

In [3]:

```
if protracted_calculation_enabled :
    # Get the list of files in the folder
    files = [file for file in os.listdir(import_data_folder) if file.endswith('.csv.gz')

    # Initialize an empty DataFrame to be filled with the data from the files
    land_stations_comb_1 = pd.DataFrame()
    # Iterate over the files and process each one
    for file in files:
        file_path = os.path.join(import_data_folder, file)

        # Read the compressed CSV file
        with gzip.open(file_path, 'rt') as file:
            df_temp = pd.read_csv(file)

            # Select and rename the specific columns
            df_temp = df_temp.iloc[:, :4]  # Select the first 4 columns
            df_temp.columns = ['STATION', 'DATE', 'ELEMENT', 'DATAVALUE']

            # Filter the DataFrame to include only desired elements
            df_temp = df_temp[df_temp['ELEMENT'].isin(['PRCP', 'TMAX', 'TMIN', 'AWND'])]

            # Convert the 'DATE' column to a datetime format
            df_temp['DATE'] = pd.to_datetime(df_temp['DATE'], format='%Y%m%d')

            # Pivot the DataFrame to convert it from long to wide format
            df_temp = df_temp.pivot(index=['STATION', 'DATE'], columns='ELEMENT', values

            # Concatenate with the final DataFrame
            land_stations_comb_1 = pd.concat([land_stations_comb_1, df_temp], ignore_ind

    # Column names to lowercase
    land_stations_comb_1.columns = land_stations_comb_1.columns.str.lower()
    print(f'land_stations_comb_1 {land_stations_comb_1.shape} created')
else:
    land_stations_comb_1 = pd.read_feather(export_data_folder + '/' + 'land_stations_com
    print(f'land_stations_comb_1 {land_stations_comb_1.shape} imported from {export_data
```

```
land_stations_comb_1 (154188121, 6) imported from DataWeatherRiver
```

## 1.2. Export dataframe

In [4]:

```
# Load or export to external file
if file_export_enabled :
    land_stations_comb_1.to_feather(export_data_folder + '/' + 'land_stations_comb_1.fea
    print(f'land_stations_comb_1 {land_stations_comb_1.shape} exported to {export_data_f
else:
    land_stations_comb_1 = pd.read_feather(export_data_folder + '/' + 'land_stations_com
    print(f'land_stations_comb_1 {land_stations_comb_1.shape} imported to {export_data_f
```

```
land_stations_comb_1 (154188121, 6) imported to DataWeatherRiver
```

# 2. Coordinates

## 2.1. Load Station coords

In [5]:

```
# Load data from txt file
ghcnd_stations = pd.read_fwf(import_data_folder + '/' + 'ghcnd_stations.txt',
                             widths=[11, 9, 10],
                             header=None,
                             names=["STATION", "LATITUDE", "LONGITUDE"])

# Column names to lowercase
ghcnd_stations.columns = ghcnd_stations.columns.str.lower()
```

```
# Data check
print(f'ghcnd_stations {ghcnd_stations.shape} loaded')
```

```
ghcnd_stations (124954, 3) loaded
```

# 2.2. Coords to Stations

## Data boundaries

In [6]:

```
# Join Coords
land_stations_comb_2 = land_stations_comb_1.merge(ghcnd_stations, how='right', left_on='

# Only observation with relevant data: No NA in weather variables
land_stations_comb_2 = land_stations_comb_2.dropna(subset=['tmax', 'tmin', 'prcp'], thre

# Only Mississippi area
land_stations_comb_2 = land_stations_comb_2[(land_stations_comb_2['longitude'] >= -100)
                                            (land_stations_comb_2['longitude'] <= -81.5)
                                            (land_stations_comb_2['latitude'] >= 31) &
                                            (land_stations_comb_2['latitude'] <= 49)]

# Save to external file
if file_export_enabled :
    land_stations_comb_2.reset_index().to_feather(export_data_folder + '/' + 'land_stati
    print(f'land_stations_comb_2 {land_stations_comb_2.shape} exported to {export_data_f
else:
    land_stations_comb_2 = pd.read_feather(export_data_folder + '/' + 'land_stations_com
    print(f'land_stations_comb_2 {land_stations_comb_2.shape} imported from {export_data
```

```
land_stations_comb_2 (30863751, 9) imported from DataWeatherRiver
```

## Screening: 33% min NAs

In [7]:

```
# Sort the DataFrame by the sum of null values in each row
land_stations_comb_3 = land_stations_comb_2.loc[
    land_stations_comb_2.isnull().sum(axis=1).sort_values().index]

# Select the first rows up to 33% of the total rows
percentage_rows = round(0.33 * len(land_stations_comb_3))
land_stations_comb_3 = land_stations_comb_3.iloc[:percentage_rows]

# Save to external file
if file_export_enabled :
    land_stations_comb_3.reset_index().to_feather(export_data_folder + '/' + 'land_stati
    print(f'land_stations_comb_3 {land_stations_comb_3.shape} exported to {export_data_f
else:
    land_stations_comb_3 = pd.read_feather(export_data_folder + '/' + 'land_stations_com
    print(f'land_stations_comb_3 {land_stations_comb_3.shape} imported from {export_data
```

```
land_stations_comb_3 (10185038, 9) imported from DataWeatherRiver
```

## Load Weather river data

In [8]:

```
# Load dataframe
land_stations_comb_3 = pd.read_feather(export_data_folder + '/' + 'land_stations_comb_3.

# Extract only date, leaving hour
land_stations_comb_3['date'] = pd.to_datetime(land_stations_comb_3['date']).dt.date
```

```
# Variable check
land_stations_comb_3['date'].head()
```

Out[8]:

```
0    2015-01-18
1    2012-04-11
2    2012-04-12
3    2012-04-13
4    2012-04-14
Name: date, dtype: object
```

# 3. Join activity_id

## 3.1. Load Incidents in Rivers

In [9]:

```
# Load dataframe
Events = pd.read_feather('DataCasualtyAndPollution' + '/' + 'Events.feather')

# Variable selection
EventsRiver = Events[(Events.watertype == 'river')][['activity_id', 'date', 'longitude',

# Extract only date, leaving hour
EventsRiver['date'] = pd.to_datetime(EventsRiver['date']).dt.date

# Drop duplicates
EventsRiver = EventsRiver.drop_duplicates()

# Data shape check
print(f'EventsRiver {EventsRiver.shape} created')
```

```
EventsRiver (11274, 4) created
```

## 3.2. Nearest weather observation to each river incident

In [10]:

```
# Function to calculate nearest weather observation
def near_observation(incident):
    # Select data corresponding to this Activity_id
    coord_incident = EventsRiver[EventsRiver['activity_id'] == incident].iloc[0]

    # Select all weather observations for this day
    coord_station = land_stations_comb_3[(land_stations_comb_3['date'] == coord_incident

    # Approximate distances
    coord_station['station_dist'] = np.sqrt((coord_station['latitude'] - coord_incident[
                                            (coord_station['longitude'] - coord_incident

    # Return the recorded weather observation located at minimum distance
    min_distance_row = coord_station[coord_station['station_dist'] == coord_station['sta
    # Add activity_id to weather data
    min_distance_row['activity_id'] = incident

    #if coord_station.empty:
        #return pd.Series(dtype='float64')
    return min_distance_row.drop_duplicates(subset=['activity_id'], keep='first')

# Concatenate function returns to create a dataframe
if protracted_calculation_enabled :
    WeatherRiver = pd.concat([near_observation(incident) for incident in EventsRiver['ac
    print(f'WeatherRiver {WeatherRiver.shape} created')
else:
    WeatherRiver = pd.read_feather(export_data_folder + '/' + 'WeatherRiver.feather')
    print(f'WeatherRiver {WeatherRiver.shape} imported from {export_data_folder}')
```

```
WeatherRiver (11274, 12) imported from DataWeatherRiver
```

In [11]:
```
# Export to external file
if file_export_enabled :
    WeatherRiver.reset_index().to_feather(export_data_folder + '/' + 'WeatherRiver.feath
    print(f'WeatherRiver {WeatherRiver.shape} exported to {export_data_folder}')
else:
    WeatherRiver = pd.read_feather(export_data_folder + '/' + 'WeatherRiver.feather')
    print(f'WeatherRiver {WeatherRiver.shape} imported from {export_data_folder}')
```

```
WeatherRiver (11274, 12) imported from DataWeatherRiver
```

# 4. Data check: Map

## 4.1. Dataframe structure

In [12]:
```
# Print first observations
WeatherRiver.head()
```

Out[12]:

| | level_0 | index | station | date | awnd | prcp | tmax | tmin | latitude | longitude | sta |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 5150464 | 102573178 | USC00013160 | 2013-04-30 | NaN | 8.0 | 256.0 | 122.0 | 32.8347 | -88.1342 | |
| **1** | 9721137 | 118137437 | USC00236641 | 2013-05-31 | NaN | 391.0 | 294.0 | 178.0 | 37.7342 | -89.9200 | |
| **2** | 1875641 | 113155576 | USC00151227 | 2013-07-08 | NaN | 0.0 | 311.0 | 194.0 | 37.5319 | -87.2669 | |
| **3** | 9790853 | 118234262 | USC00237452 | 2013-07-02 | NaN | 69.0 | 267.0 | 161.0 | 38.6308 | -90.2708 | |
| **4** | 244965 | 150268301 | USW00014920 | 2013-04-09 | 44.0 | 465.0 | 61.0 | 17.0 | 43.8792 | -91.2531 | |

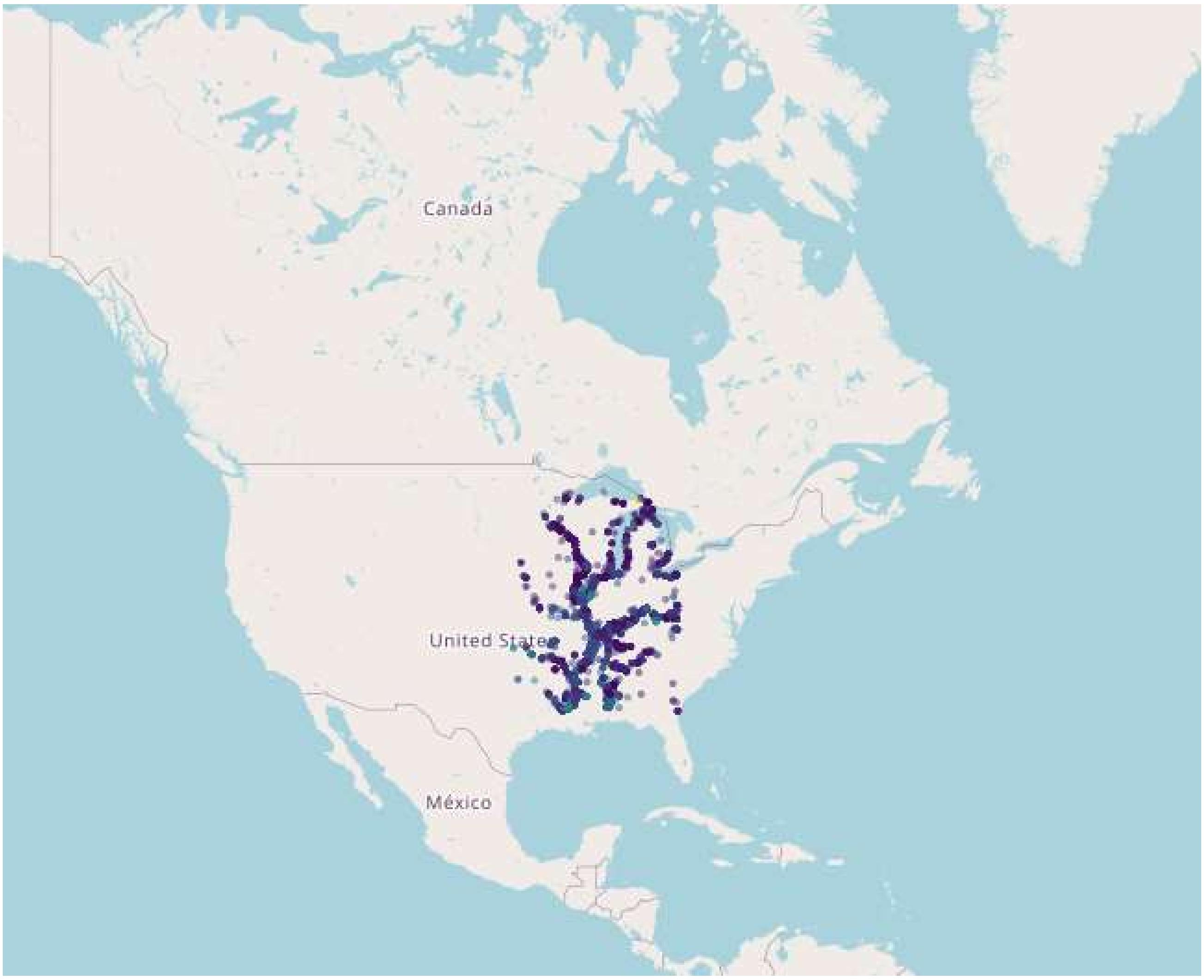
## 4.2. Map visualization

In [13]:
```
# Create figure object
fig = go.Figure()

# Aggregate WeatherRiver points
fig.add_trace(go.Scattermapbox(
    lat=WeatherRiver['latitude'],
    lon=WeatherRiver['longitude'],
    mode='markers',
    marker=dict(
        size=5,
        color=np.log1p(WeatherRiver['station_dist']),   # logarithmic scale
        colorscale=px.colors.sequential.Viridis,
        opacity=0.5,
    ),
    text=WeatherRiver.apply(lambda row:f"station:{row['station']}<br>station_dist: {row[
))
```

```
# Set up map design
fig.update_layout(
    margin ={'l':0,'t':0,'b':0,'r':0},
    mapbox = {
        'style': "open-street-map",
        'center': {'lon': -112, 'lat': 48},
        'zoom': 2})

# Show map
fig.show()
```


Canada
United States
México