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

Annex 4.2. Data Explore: MergedActivity

Author: Oscar Anton
Date: 2024

Version: 0.9

0. Loadings

Libraries

In [1]:
```
# Data general management
import numpy as np
import pandas as pd

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px

from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Automatic Exploratory Data Analysis (EDA) report
from ydata_profiling import ProfileReport

# Ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

General variables

In [2]:
```
# Main data folder
merged_activity_folder = '../3.DataPreprocess/DataMergedActivity'

# Toggle for export data to external file
file_export_enabled = False
```

Load base dataframe

In [3]:
```
# Load dataframe from external file
merged_activity = pd.read_feather(merged_activity_folder + '/' + 'merged_activity.feather')

# Check dataframe structure
print(f'Merged activity {merged_activity.shape} loaded')
merged_activity.head()
```

```
Merged activity (68000, 29) loaded
```

Out[3]:

	index	activity_id	date	hour	region	latitude	longitude	watertype	event_type	damage_status	...	classification_society	solas_desc	air_temp
0	0	1475897	2002-01-01	03:45	Mississippi	37.017330	-88.274720	river	Grounding	Undamaged	...	UNSPECIFIED	Non SOLAS	-38.50
1	1	1475897	2002-01-01	03:45	Mississippi	37.017330	-88.274720	river	Grounding	Undamaged	...	UNSPECIFIED	Non SOLAS	-38.50
2	2	1477008	2002-01-01	13:53	East Coast	39.322020	-76.363650	ocean	Damage to the Environment	Undamaged	...	UNSPECIFIED	Non SOLAS	-17.09
3	3	1477373	2002-01-01	18:10	Mississippi	31.525833	-87.971667	river	Material Failure (Vessels)	Damaged	...	UNSPECIFIED	Non SOLAS	11.00
4	4	1477402	2002-01-01	10:00	Gulf of Mexico	30.641667	-88.034167	ocean	Grounding	Undamaged	...	UNSPECIFIED	Non SOLAS	41.25

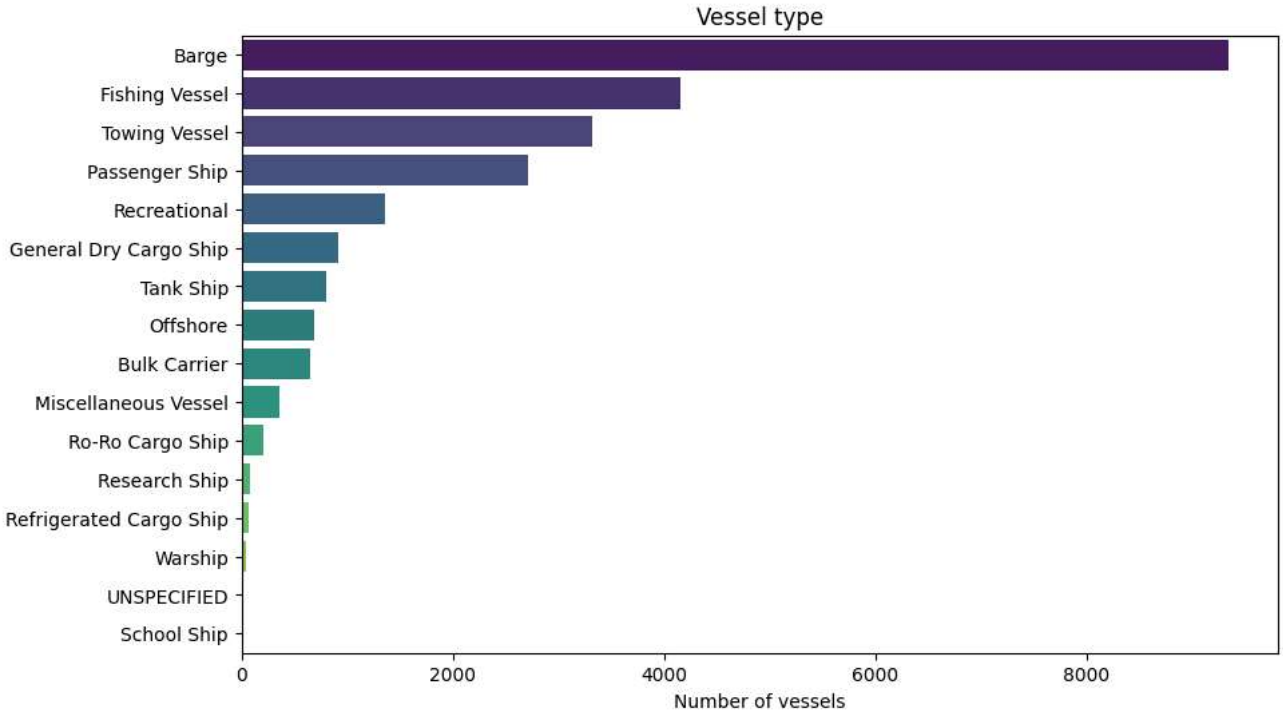
5 rows × 29 columns

1. Vessel features

1.1. vessel_class

Frequency

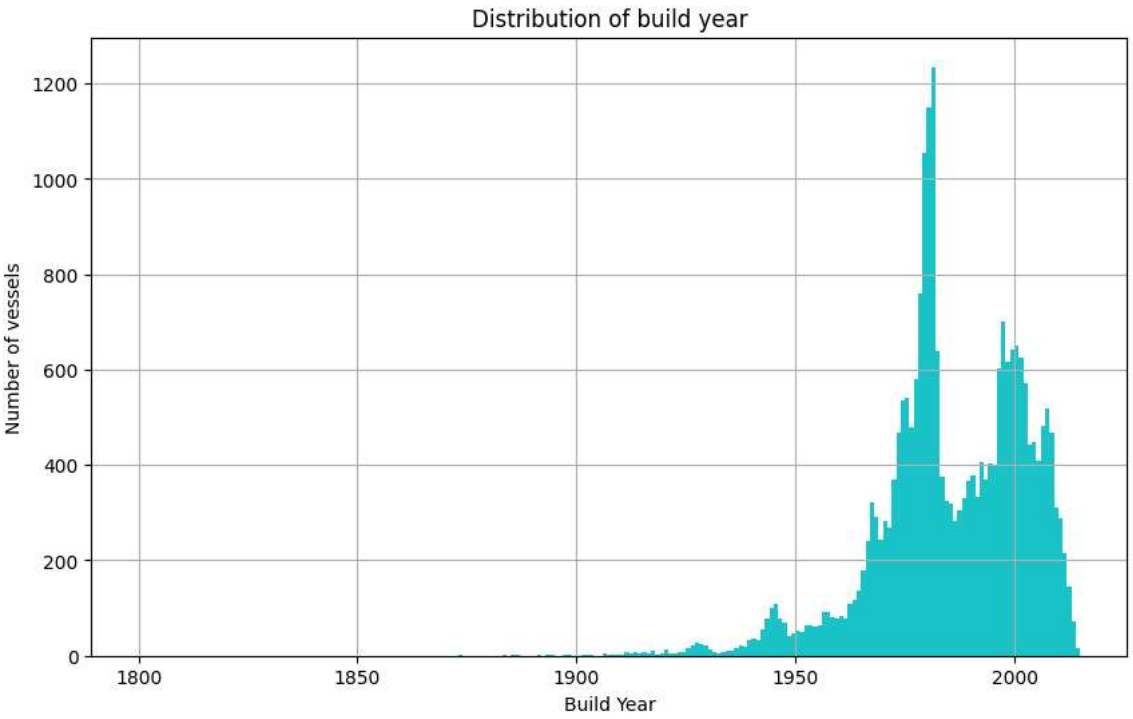
In [4]:
```
# Filter data: group by vessel class
filtered_df = (merged_activity
               .drop_duplicates(subset='vessel_id', keep='first')
               .groupby('vessel_class').size().reset_index(name='frequency')
               .sort_values(by='frequency', ascending=False))

# Plot barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='frequency', y='vessel_class', data=filtered_df,
            palette='viridis')

# Customize plot
plt.title('Vessel type')
plt.xlabel('Number of vessels')
plt.ylabel(None)
plt.show()
```



1.2. build_year

Frequency

In [5]:
```
# Filter data: build_year between 1800 and 2015
filtered_df = (merged_activity
               [(merged_activity['build_year'] >= 1800) &
                (merged_activity['build_year'] <= 2015)]
                .drop_duplicates(subset='vessel_id', keep='first'))

# Plot histogram
plt.figure(figsize=(10, 6))
sns.histplot(filtered_df['build_year'], bins=range(1800, 2016),
             edgecolor='None', color='#00bfc4', alpha=0.9)

# Customize plot
plt.title('Distribution of build year')
plt.xlabel('Build Year')
plt.ylabel('Number of vessels')
plt.grid(True)
plt.show()
```



Age at incident

In [6]:
```
# Filter data: age between 0 - 120
filtered_df = (merged_activity
               [(merged_activity['age'] > 0) &
                (merged_activity['age'] < 120)]
                .drop_duplicates(subset='vessel_id', keep='first'))

# Plot histogram
plt.figure(figsize=(10, 6))
sns.histplot(filtered_df['age'], bins=range(0, 120),
             edgecolor='None', color='#00bfc4', alpha=0.9)

# Customize plot
plt.title('Age at incident')
plt.xlabel('Age')
plt.ylabel('Number of vessels')
plt.grid(True)
plt.show()
```

Unlikely Age values

In [7]:
```
# Filter data: build_year between 1800 and 2015, age < 0
filtered_df = (merged_activity
               [(merged_activity['build_year'] >= 1800) &
                (merged_activity['build_year'] <= 2015)]
                .drop_duplicates(subset='vessel_id', keep='first')
```

```
                .sort_values(by='age', ascending=True)
                .head(10)
                [['vessel_id', 'vessel_name', 'imo_number', 'event_type', 'date', 'build_year', 'age']])

# Print these values
print(filtered_df)

# Fix some wrong values, according to public data (vesselfinder.com)
merged_activity.loc[merged_activity['vessel_id'] == 370425, 'build_year'] = 1992
merged_activity.loc[merged_activity['vessel_id'] == 813316, 'build_year'] = 2001
merged_activity.loc[merged_activity['vessel_id'] == 567313, 'build_year'] = 2005
merged_activity.loc[merged_activity['vessel_id'] == 1229111, 'build_year'] = 2005

# Recalculate age variable
merged_activity['age'] = pd.to_datetime(merged_activity['date']).dt.year - merged_activity['build_year']

# Check data
filtered_df = (merged_activity
               [(merged_activity['build_year'] >= 1800) &
                (merged_activity['build_year'] <= 2015)]
                .drop_duplicates(subset='vessel_id', keep='first')
                .sort_values(by='age', ascending=True)
                .head(10)
                [['vessel_id', 'vessel_name', 'imo_number', 'event_type', 'date', 'build_year', 'age']])

print(filtered_df)
```

```
      vessel_id       vessel_name imo_number                 event_type  \
32181     567313   DOLPHIN SEAFARI            Material Failure (Vessels)
571       370425  BENNO C. SCHMIDT    9040546                   Flooding
2443      813316         PEAPICKER             Damage to the Environment
61142    1229111    PACIFIC SPIRIT             Damage to the Environment
2923      568186      MIDNIGHT SUN    9232278   Damage to the Environment
59761    1110524              FSV6    9664988                 Grounding
11394     722960  ALASKAN EXPLORER    9244661   Damage to the Environment
42425    1052777           OPTI-EX             Damage to the Environment
58920    1167916        NOR EASTER            Material Failure (Vessels)
3149      571224           DCH 501                           Grounding

            date  build_year  age
32181 2008-03-25        2015   -7
571   2002-02-20        2009   -7
2443  2002-07-28        2006   -4
61142 2013-02-12        2015   -2
2923  2002-09-08        2003   -1
59761 2012-11-04        2013   -1
11394 2004-10-04        2005   -1
42425 2010-01-15        2011   -1
58920 2012-09-12        2012    0
3149  2002-09-30        2002    0
       vessel_id       vessel_name imo_number                 event_type  \
59761    1110524              FSV6    9664988                 Grounding
11394     722960  ALASKAN EXPLORER    9244661   Damage to the Environment
42425    1052777           OPTI-EX             Damage to the Environment
2923      568186      MIDNIGHT SUN    9232278   Damage to the Environment
27335     858429       PLUM ISLAND                Vessel Maneuverability
22917     847765           CBY 418            Material Failure (Vessels)
46622    1049360          CBX 2098                           Grounding
45045    1052395         ACL 10110                           Grounding
29795     804674   DOUBLE SKIN 143            Material Failure (Vessels)
41175     997864       PAULA RUBLE            Material Failure (Vessels)

            date  build_year  age
59761 2012-11-04        2013   -1
11394 2004-10-04        2005   -1
42425 2010-01-15        2011   -1
2923  2002-09-08        2003   -1
27335 2007-07-02        2007    0
22917 2006-11-06        2006    0
46622 2010-09-26        2010    0
45045 2010-07-03        2010    0
29795 2007-11-26        2007    0
41175 2009-10-10        2009    0
```

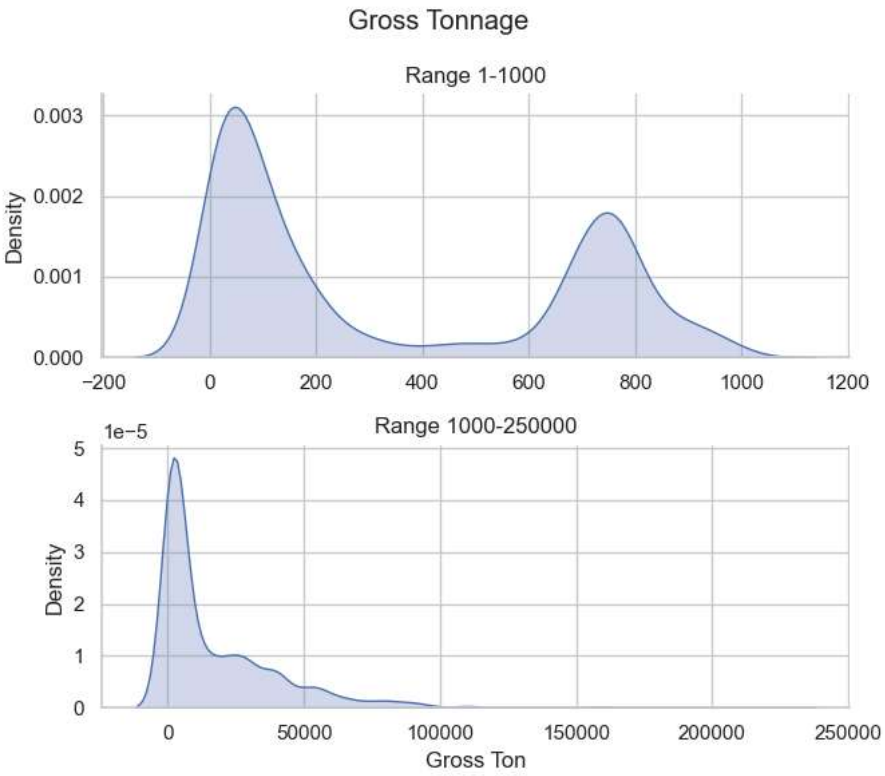
1.3. gross_tonnage

Density

In [8]:

```
# Filter data
filtered_df = (merged_activity
               [(merged_activity['gross_ton'] >= 1) &
                (merged_activity['gross_ton'] <= 250000)]
                .drop_duplicates(subset='vessel_id', keep='first'))

# Labels for facet wrap
labels = ["1-1000", "1000-250000"]
filtered_df['gross_ton_range'] = pd.cut(filtered_df['gross_ton'], bins=[0, 1000, 250000], labels=labels)

# Plot
sns.set(style="whitegrid")
g = sns.FacetGrid(filtered_df, col="gross_ton_range", col_wrap=1, height=3, aspect=7/3, sharey=False, sharex=False)
g.map(sns.kdeplot, "gross_ton", shade=True)
g.set_axis_labels("Gross Ton", "Density")
```

```
g.set_titles("Range {col_name}")

# Customize plot
plt.suptitle('Gross Tonnage')
plt.tight_layout()
plt.show()
```



Ranking

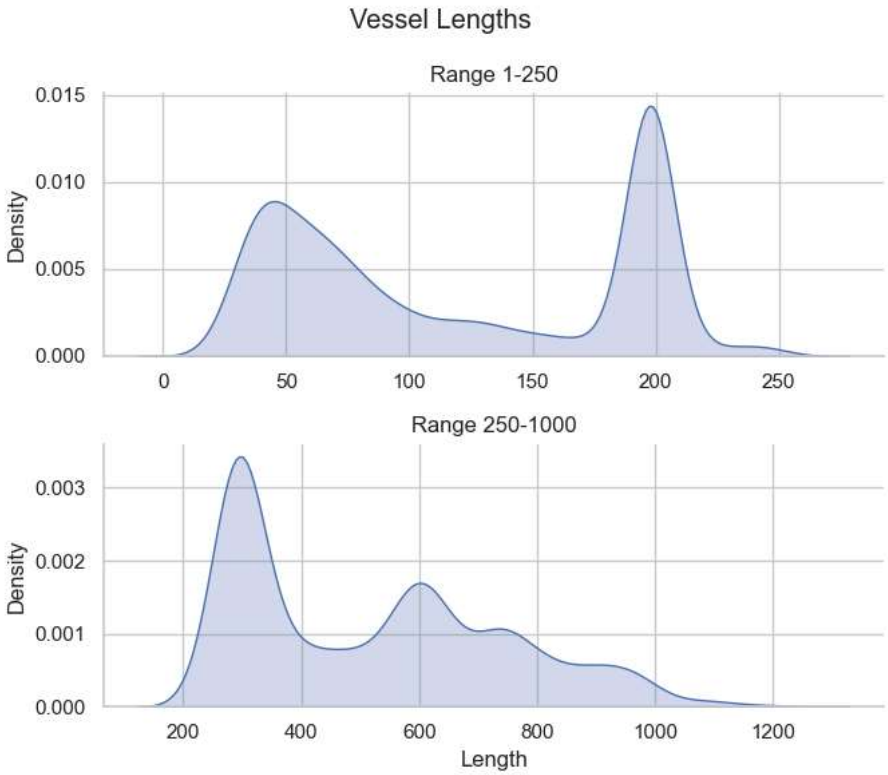
In [9]:
```
# Filter data
filtered_df = (merged_activity
               .drop_duplicates(subset='vessel_id', keep='first')
               .sort_values(by='gross_ton', ascending=False)
               .head(10)
               [['vessel_id', 'vessel_name', 'imo_number', 'build_year', 'gross_ton', 'length']])

# Print these values
print(filtered_df)
```

```
       vessel_id         vessel_name imo_number  build_year  gross_ton  length
55140     933484  ALLURE OF THE SEAS    9383948        2010     225282  1181.0
44147     933483   OASIS OF THE SEAS    9383936        2009     225282  1187.0
843       228358       BERGE PIONEER    7708314        1980     188728  1071.7
33832     437660             RAMLAH    9102239        1996     163882  1115.5
58153     617142           ENERGY R    9241114        2003     161306  1092.4
59075     586555      OVERSEAS MULAN    9230880        2002     161233  1092.0
50989     938329             SPYROS    9315367        2007     161175  1092.4
49195    1039985              DORRA    9386964        2009     160782  1092.6
5708      606521             ABQAIQ    9247182        2002     159990  1093.4
52693     881630     MAERSK NAUTILUS    9312494        2006     159911  1091.9
```

1.4. Length

Density

In [10]:
```
# Filter data
filtered_df = (merged_activity
               [(merged_activity['length'] >= 1) &
                (merged_activity['length'] <= 1250)]
                .drop_duplicates(subset='vessel_id', keep='first'))

# Labels for facet wrap
labels = ["1-250", "250-1000"]
filtered_df['length_range'] = pd.cut(filtered_df['length'], bins=[0, 250, 1250], labels=labels)

# Plot
sns.set(style="whitegrid")
g = sns.FacetGrid(filtered_df, col="length_range", col_wrap=1, height=3, aspect=7/3, sharey=False, sharex=False)
g.map(sns.kdeplot, "length", shade=True)
g.set_axis_labels("Length", "Density")
g.set_titles("Range {col_name}")

# Customize plot
plt.suptitle('Vessel Lengths')
plt.tight_layout()
plt.show()
```



Ranking

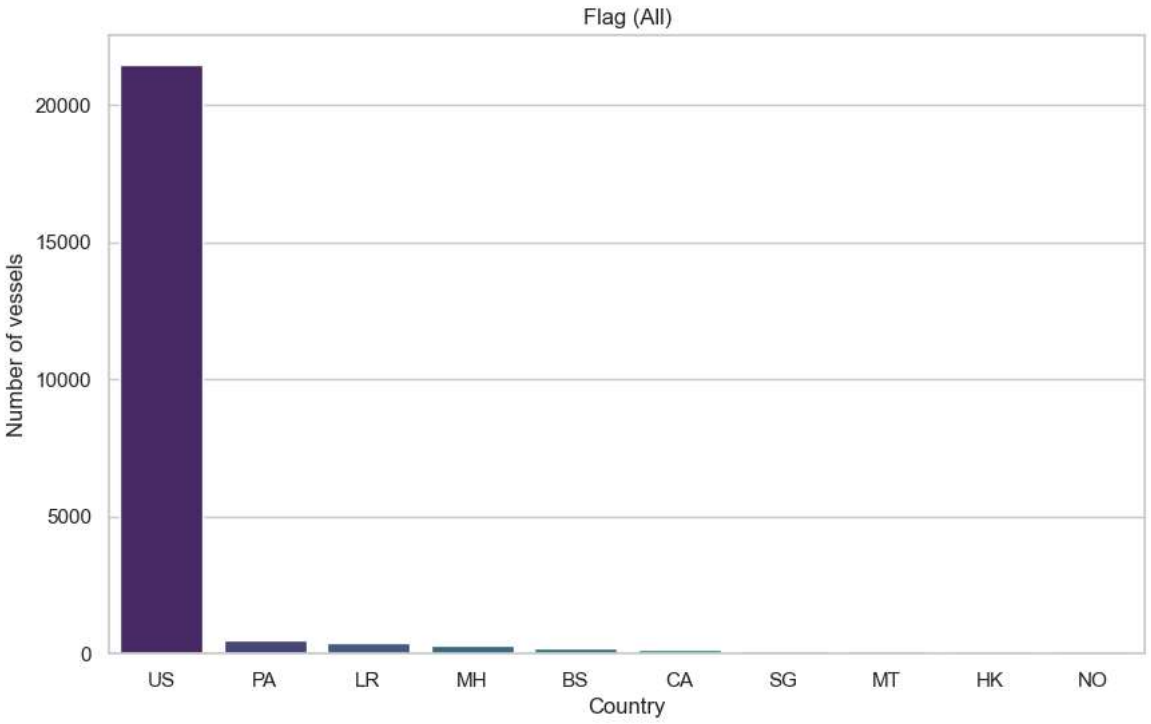
In [11]:
```
# Filter data
filtered_df = (merged_activity
               .drop_duplicates(subset='vessel_id', keep='first')
               .sort_values(by='length', ascending=False)
               .head(10)
               [['vessel_id', 'vessel_name', 'imo_number', 'build_year', 'gross_ton', 'length']])

# Print these values
print(filtered_df)
```

```
       vessel_id         vessel_name imo_number  build_year  gross_ton  length
38605    1001188        GRETE MAERSK    9302889        2005      97933  1203.8
58429     998455       GUDRUN MAERSK    9302877        2005      97933  1203.8
43125    1028411     MATHILDE MAERSK    9359052        2009      98268  1203.7
40660    1008200        MARIT MAERSK    9359040        2009      98268  1203.7
47295     999387      MARCHEN MAERSK    9359014        2007      98268  1203.7
44147     933483   OASIS OF THE SEAS    9383936        2009     225282  1187.0
55140     933484  ALLURE OF THE SEAS    9383948        2010     225282  1181.0
55772    1026695     CMA CGM IVANHOE    9365805        2008     111249  1148.0
56127     500915       SKAGEN MAERSK    9166792        1999      91560  1138.3
16571     489733        SVEND MAERSK    9166778        1999      91560  1138.3
```

1.5. flag

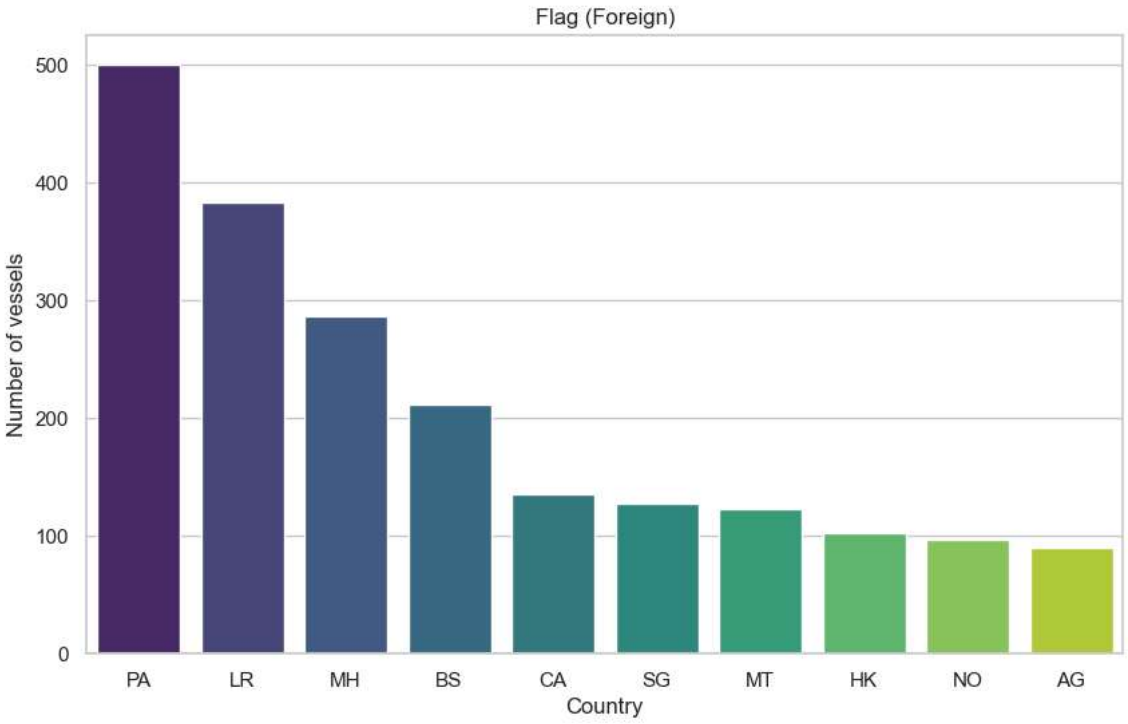
Frequency (All)

In [12]:
```
# Filter data: group by vessel class
filtered_df = (merged_activity
               .drop_duplicates(subset='vessel_id', keep='first')
               .groupby('flag_abbr').size().reset_index(name='frequency')
               .sort_values(by='frequency', ascending=False)
               .head(10))

# Plot barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='flag_abbr', y='frequency', data=filtered_df,
            palette='viridis')

# Customize plot
plt.title('Flag (All)')
plt.xlabel('Country')
plt.ylabel('Number of vessels')
plt.show()
```



Frequency (Foreign)

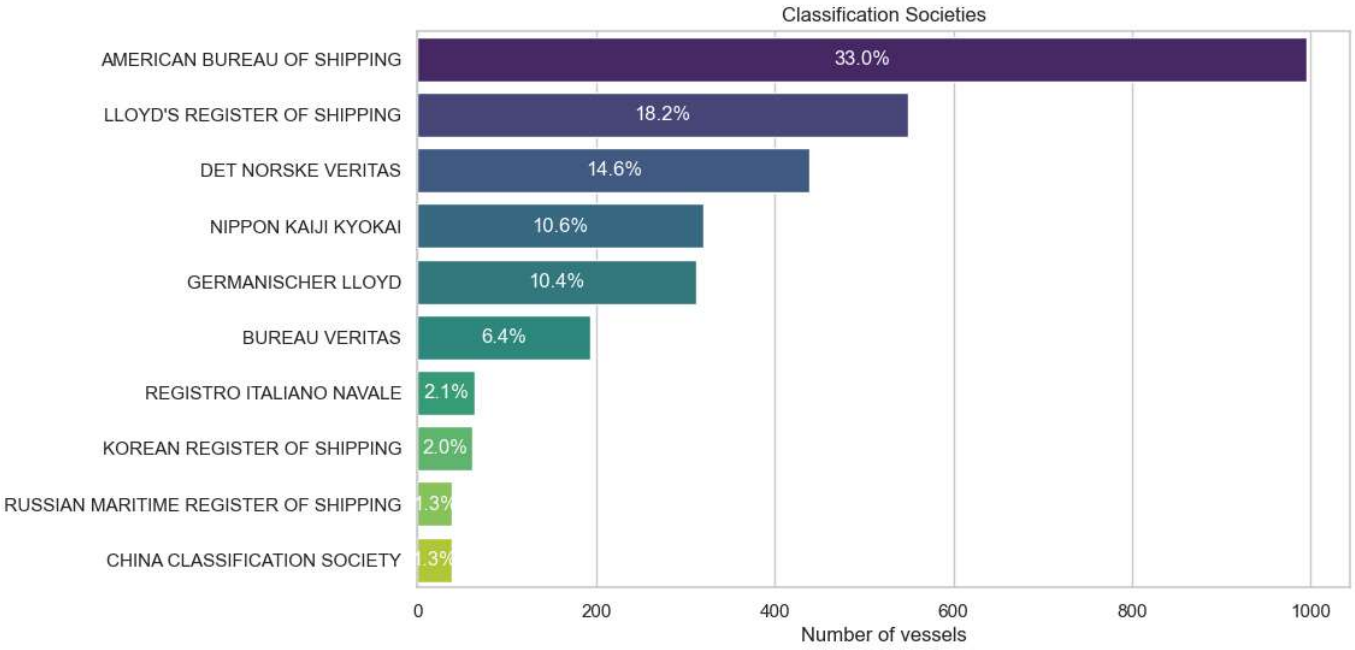
In [13]:
```
# Filter data: drop US flag, group by vessel class
filtered_df = (merged_activity
               [merged_activity['flag_abbr'] != "US"]
               .drop_duplicates(subset='vessel_id', keep='first')
               .groupby('flag_abbr').size().reset_index(name='frequency')
               .sort_values(by='frequency', ascending=False)
               .head(10))

# Plot barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='flag_abbr', y='frequency', data=filtered_df,
            palette='viridis')

# Customize plot
plt.title('Flag (Foreign)')
plt.xlabel('Country')
plt.ylabel('Number of vessels')
plt.show()
```



1.6. Classification Societies

In [14]:
```
# Filter data: group by classification_society
filtered_df = (merged_activity
               [merged_activity['classification_society'] != "UNSPECIFIED"]
```

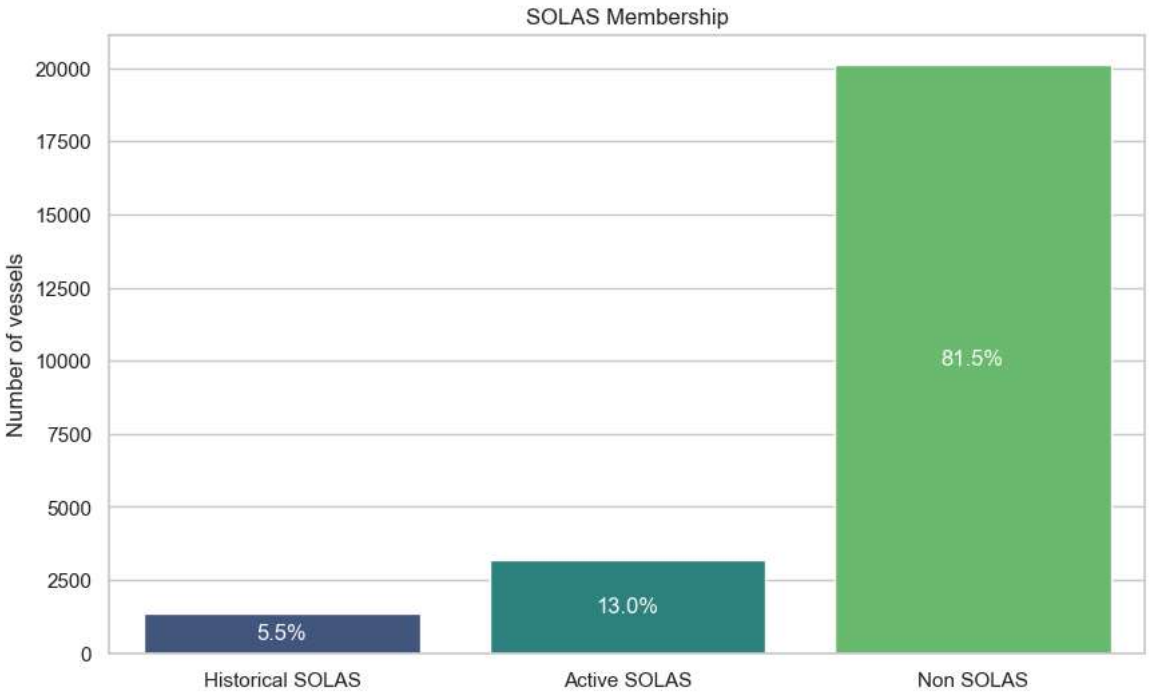
```
                .drop_duplicates(subset='vessel_id', keep='first')
                .groupby('classification_society').size().reset_index(name='frequency')
                .sort_values(by='frequency', ascending=False)
                .head(10))

# Plot barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='frequency', y='classification_society', data=filtered_df,
            palette='viridis')

# Percentage
filtered_df['percentage'] = filtered_df['frequency'] / filtered_df['frequency'].sum() * 100
for i, (value, percentage) in enumerate(zip(filtered_df['frequency'], filtered_df['percentage'])):
    plt.text(value / 2, i, f'{percentage:.1f}%', va='center', ha='center', color='white')

# Customize plot
plt.title('Classification Societies')
plt.xlabel('Number of vessels')
plt.ylabel(None)
plt.show()
```



1.7. SOLAS Membership

In [15]:

```
# Filter data: group by SOLAS
filtered_df = (merged_activity
                .drop_duplicates(subset='vessel_id', keep='first')
                .groupby('solas_desc').size().reset_index(name='frequency')
                .sort_values(by='frequency', ascending=True))

# Plot barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='solas_desc', y='frequency', data=filtered_df,
            palette='viridis')

# Percentage
filtered_df['percentage'] = filtered_df['frequency'] / filtered_df['frequency'].sum() * 100
for i, (value, percentage) in enumerate(zip(filtered_df['frequency'], filtered_df['percentage'])):
    plt.text(i, value / 2, f'{percentage:.1f}%', va='center', ha='center', color='white')

# Customize plot
plt.title('SOLAS Membership')
plt.xlabel(None)
plt.ylabel('Number of vessels')
plt.show()
```

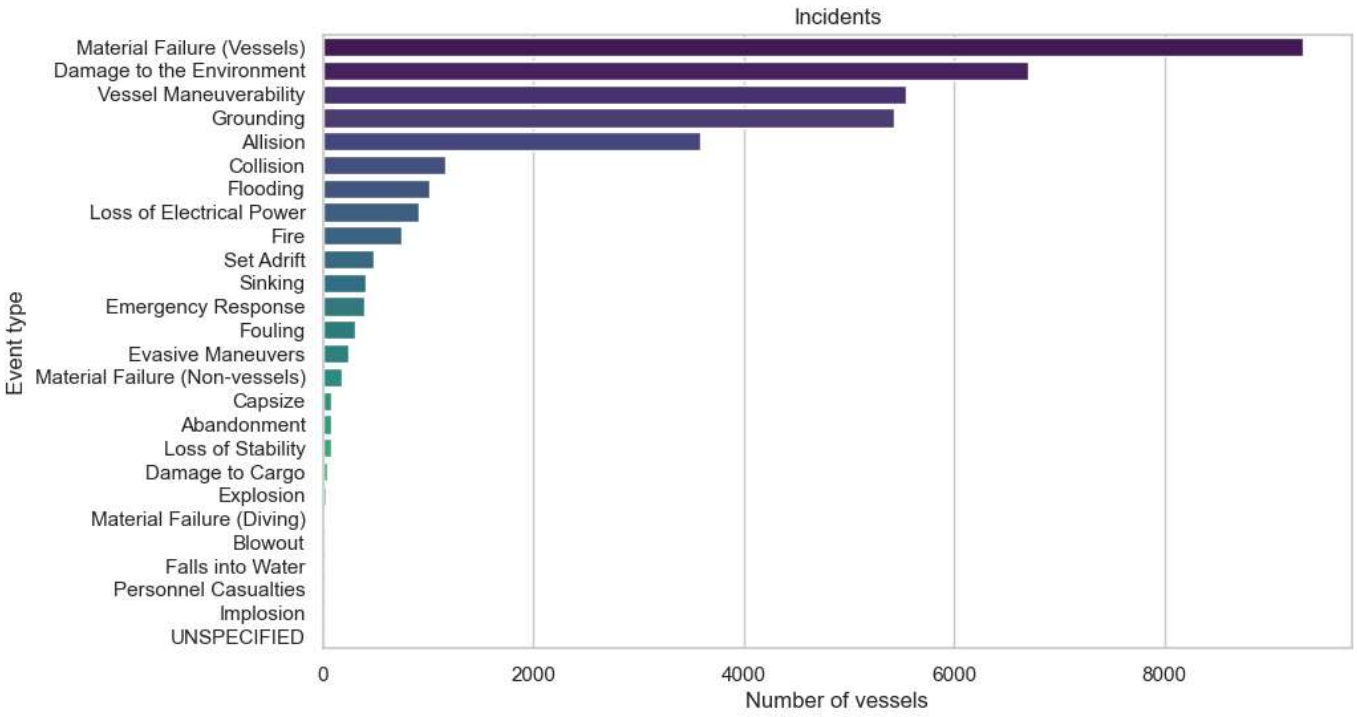


2. Incidents

2.1. event_type

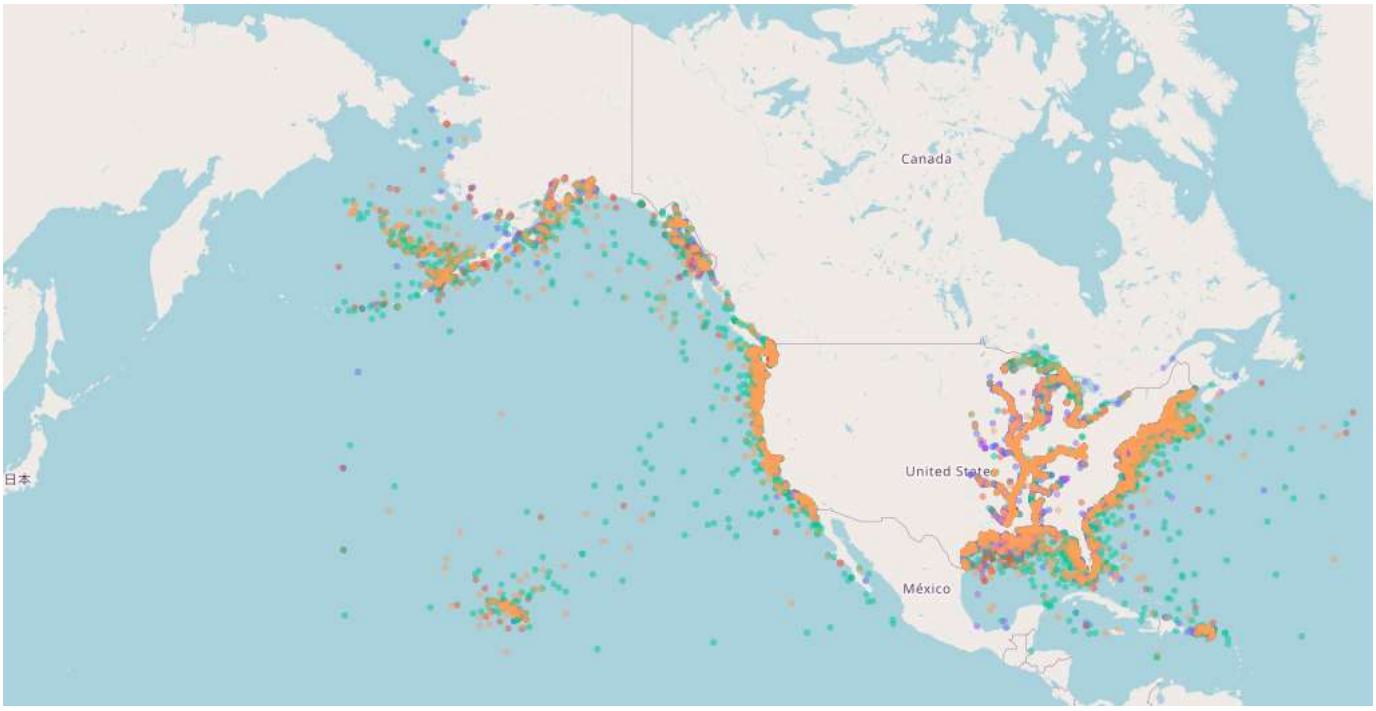
Frequency

In [16]:
```
# Filter data: group by event_type
filtered_df = (merged_activity
               [merged_activity['event_type'] != "No event"]
               .drop_duplicates(subset='activity_id', keep='first')
               .groupby('event_type').size().reset_index(name='frequency')
               .sort_values(by='frequency', ascending=False))

# Plot barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='frequency', y='event_type', data=filtered_df,
            palette='viridis')

# Customize plot
plt.title('Incidents')
plt.xlabel('Number of vessels')
plt.ylabel('Event type')
plt.show()
```



Location

In [17]:
```
# Filter data, top 5 event_type:
top_5_event_types = (merged_activity['event_type']
                     .value_counts()
                     .head(5)
                     .index)

filtered_df = (merged_activity
               [merged_activity['event_type']
                .isin(top_5_event_types)]
                .drop_duplicates(subset='activity_id', keep='first'))

# Create scatter mapbox
fig = px.scatter_mapbox(filtered_df,
                        lat="latitude", lon="longitude",
                        color="event_type",
                        custom_data=["activity_id", "vessel_id", "date"])

# Add popup data
fig.update_traces(opacity=0.5,
                  hovertemplate="<b>activity_id:</b> %{customdata[0]} \
                            <br><b>vessel_id:</b> %{customdata[1]} \
                            <br><b>date:</b> %{customdata[2]} \
                            <extra></extra>")

# Customize plot
fig.update_layout(
    legend = dict(yanchor="top", y=0.99, xanchor="right", x=0.99, bgcolor='rgba(255, 255, 255, 0.5)'),
    margin = {'l':0,'t':0,'b':0,'r':0},
    mapbox = {'style': "open-street-map",
              'center': {'lon': -112, 'lat': 48},
              'zoom': 2})

fig.show()
```

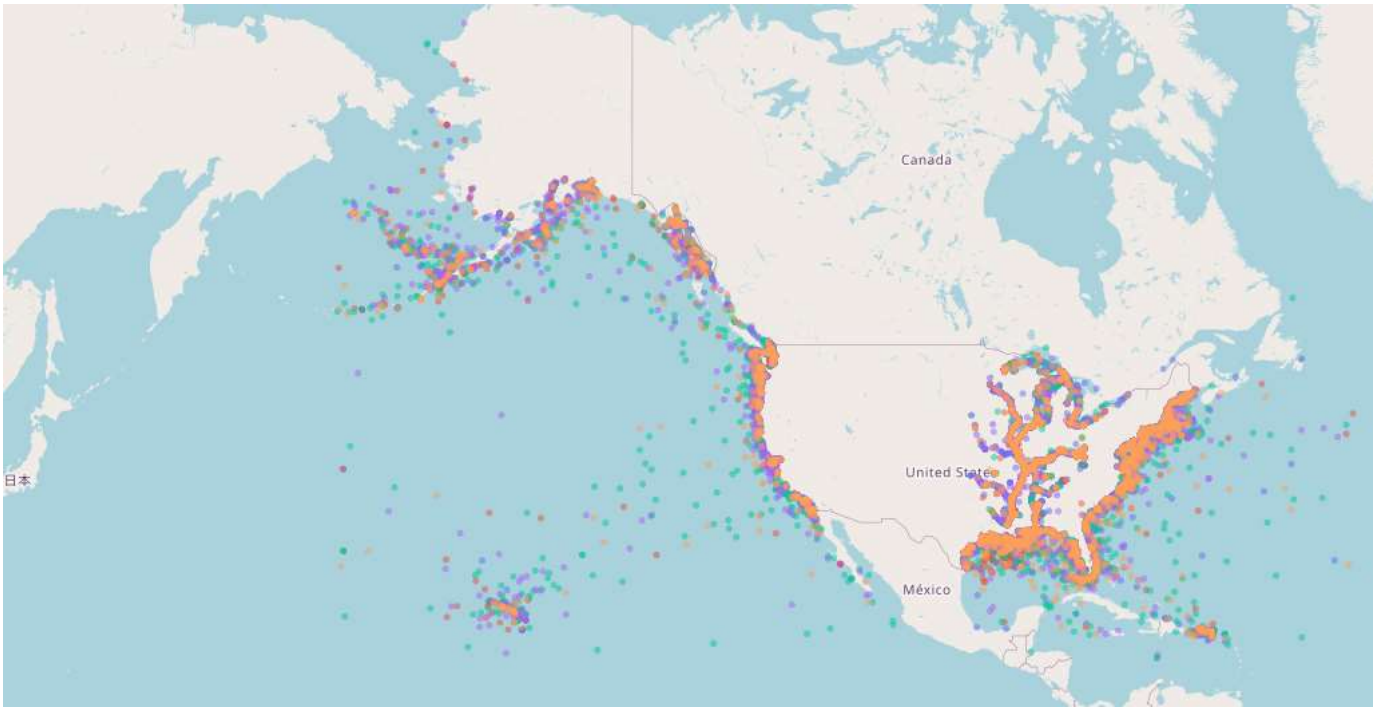


2.3. event_class

Location

In [18]:
```
# Filter data
filtered_df = (merged_activity
               .drop_duplicates(subset='activity_id', keep='first')
               .copy())

# Create scatter mapbox
fig = px.scatter_mapbox(filtered_df,
                        lat="latitude", lon="longitude",
                        color="event_class",
                        custom_data=["activity_id", "vessel_id", "date"])

# Add popup data
fig.update_traces(opacity=0.5,
                  hovertemplate="<b>activity_id:</b> %{customdata[0]} \
                            <br><b>vessel_id:</b> %{customdata[1]} \
                            <br><b>date:</b> %{customdata[2]} \
                            <extra></extra>")
```

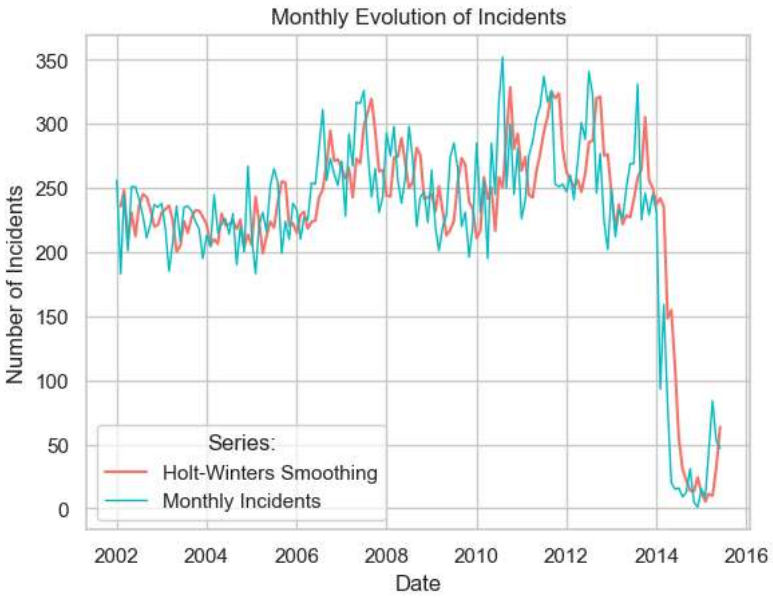
```
# Customize plot
fig.update_layout(
    legend = dict(yanchor="top", y=0.99, xanchor="right", x=0.99, bgcolor='rgba(255, 255, 255, 0.5)'),
    margin = {'l':0,'t':0,'b':0,'r':0},
    mapbox = {'style': "open-street-map",
              'center': {'lon': -112, 'lat': 48},
              'zoom': 2})

fig.show()
```



2.4. event_type

Time Series

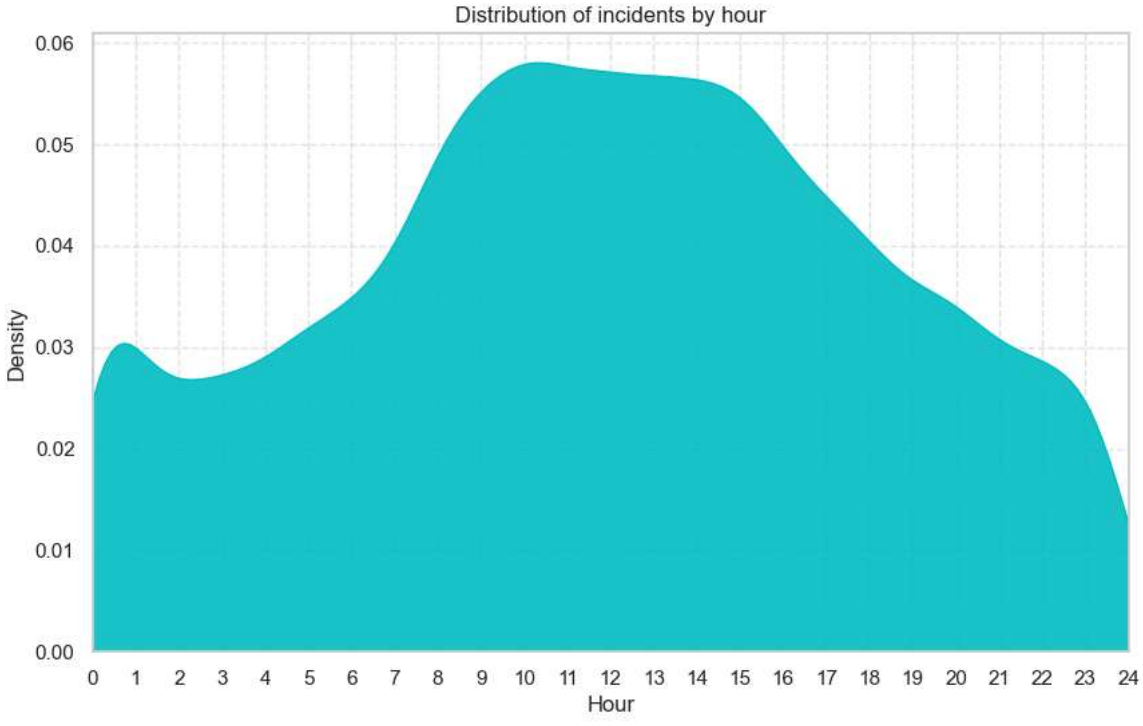
In [19]:
```
# Data filter
filtered_df = (merged_activity
               .drop_duplicates(subset='activity_id', keep='first')
               .copy())

# Calculating year and month
filtered_df['year'] = filtered_df['date'].dt.year
filtered_df['month'] = filtered_df['date'].dt.month

# Counting incidents per month
incidents_month = filtered_df.groupby(['year', 'month']).size().reset_index(name='incidents_month')

# Creating 'date' column
incidents_month['date'] = pd.to_datetime(incidents_month[['year', 'month']].assign(day=1))

# Smoothing using Holt-Winters
model = ExponentialSmoothing(incidents_month['incidents_month'], trend=None, seasonal=None)
fit = model.fit()
incidents_month['incidents_smooth'] = fit.fittedvalues.shift(1)

# Sorting and visualization
incidents_month = incidents_month.sort_values(['year', 'month']).reset_index(drop=True)

# Visualization
plt.plot(incidents_month['date'], incidents_month['incidents_smooth'],
         color='#f8766d', label='Holt-Winters Smoothing')
plt.plot(incidents_month['date'], incidents_month['incidents_month'],
         color='#00bfc4', label='Monthly Incidents', linewidth=1)
plt.xlabel('Date')
plt.ylabel('Number of Incidents')
plt.title('Monthly Evolution of Incidents')
plt.legend(title='Series:')
plt.grid(True)
plt.show()
```



2.5. Incident hour

Frequency

In [20]:
```
# Data filter
filtered_df = (merged_activity
               .drop_duplicates(subset='activity_id', keep='first')
               .copy())

# Calculate hour in decimal format
filtered_df['hour_dec'] = round(pd.to_numeric(filtered_df['hour'].str.split(':').str[0]) +
                                (pd.to_numeric(filtered_df['hour'].str.split(':').str[1]) / 60), 2)

# Density plot
plt.figure(figsize=(10, 6))
sns.kdeplot(data=filtered_df, x='hour_dec', fill=True, color="#00bfc4", alpha=0.9)

# Customize plot
plt.xlim(0, 24)
plt.xticks(range(25))
plt.title("Distribution of incidents by hour")
plt.xlabel("Hour")
plt.ylabel("Density")
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```



2.6. damage_assesment

Ranking

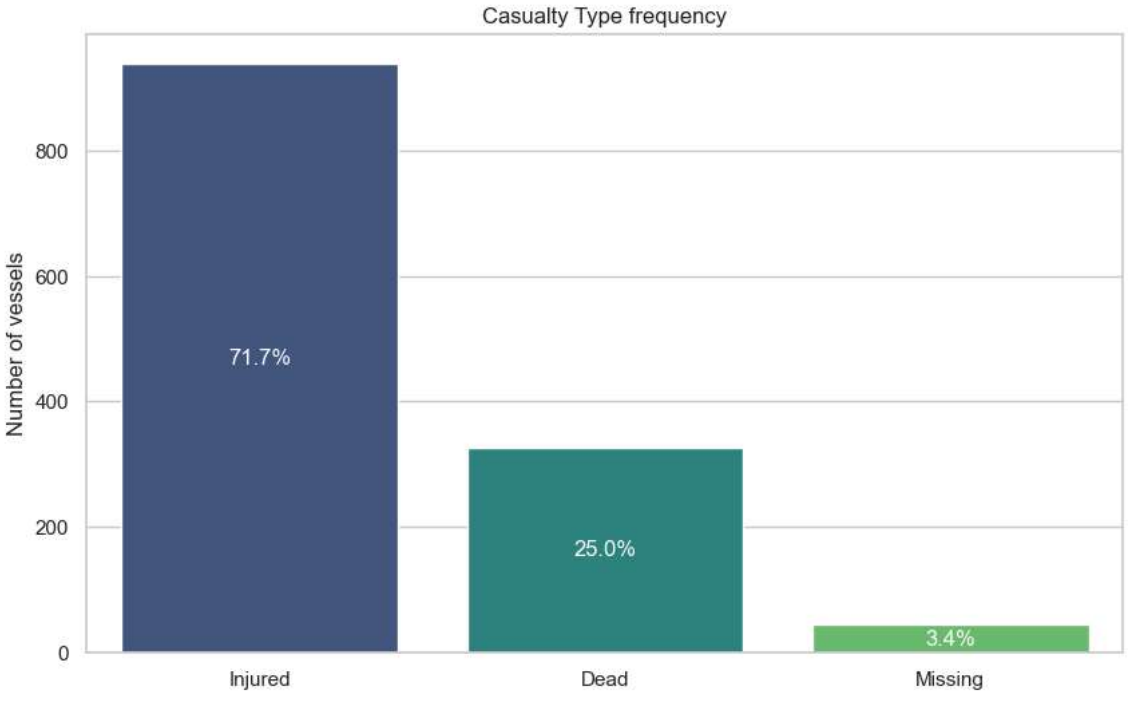
In [21]:
```
# Filter data
filtered_df = (merged_activity
               .drop_duplicates(subset='activity_id', keep='first')
               .sort_values(by='damage_assessment', ascending=False)
               .head(10)
               [['activity_id', 'vessel_name', 'event_type', 'damage_assessment']])

# Print these values
print(filtered_df)
```

```
       activity_id      vessel_name                 event_type  \
49392      3964637            C 533                   Allision
49395      3965662        SUNSET I                   Grounding
49347      3960377   RIVER ELEGANCE           Evasive Maneuvers
7191       1966333  RICHARD A BAKER                   Grounding
6682       1902882   GILBERT TAYLOR  Material Failure (Vessels)
6680       1900683       RUBY RIVER                   Allision
6679       1896064         JAY LUHR  Material Failure (Vessels)
24507      2865301      KIRBY 28037                   Allision
24508      2865600     KIRBY 30026B  Material Failure (Vessels)
24563      2870902   HARRY J. BROCK                   Grounding

       damage_assessment
49392        350000000.0
49395        350000000.0
49347        350000000.0
7191          98000000.0
6682          85000000.0
6680          85000000.0
6679          85000000.0
24507         60000000.0
24508         60000000.0
24563         60000000.0
```

2.7. casualty

Frequency

In [22]:
```
# Filter data: group by casualty
filtered_df = (merged_activity
               .drop_duplicates(subset='activity_id', keep='first')
               .drop(merged_activity[merged_activity['casualty'] == "UNSPECIFIED"].index)
               .groupby('casualty').size().reset_index(name='frequency')
               .sort_values(by='frequency', ascending=False))

# Plot barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='casualty', y='frequency', data=filtered_df, palette='viridis')

# Percentage
filtered_df['percentage'] = filtered_df['frequency'] / filtered_df['frequency'].sum() * 100
for i, (value, percentage) in enumerate(zip(filtered_df['frequency'], filtered_df['percentage'])):
    plt.text(i, value / 2, f'{percentage:.1f}%', va='center', ha='center', color='white')

# Customize plot
plt.title('Casualty Type frequency')
plt.xlabel(None)
plt.ylabel('Number of vessels')
plt.show()
```

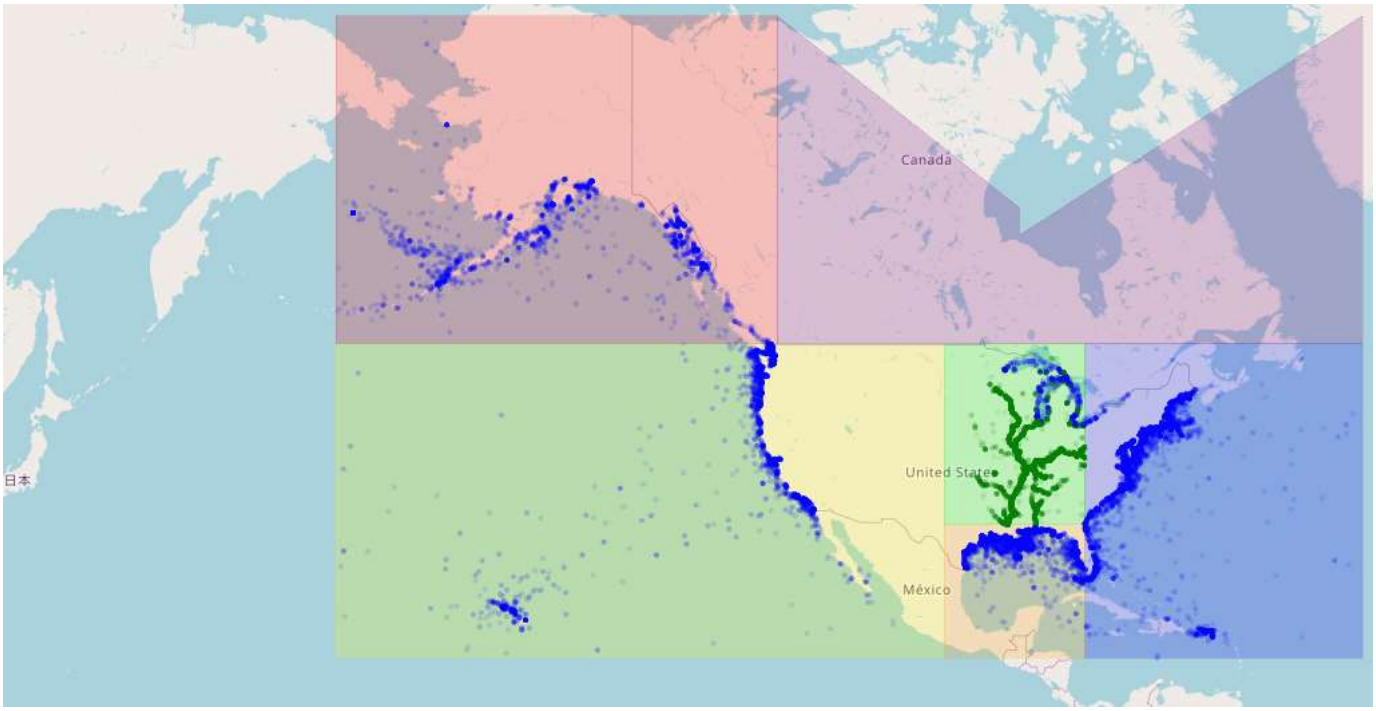


3. Geography

3.1. Regions

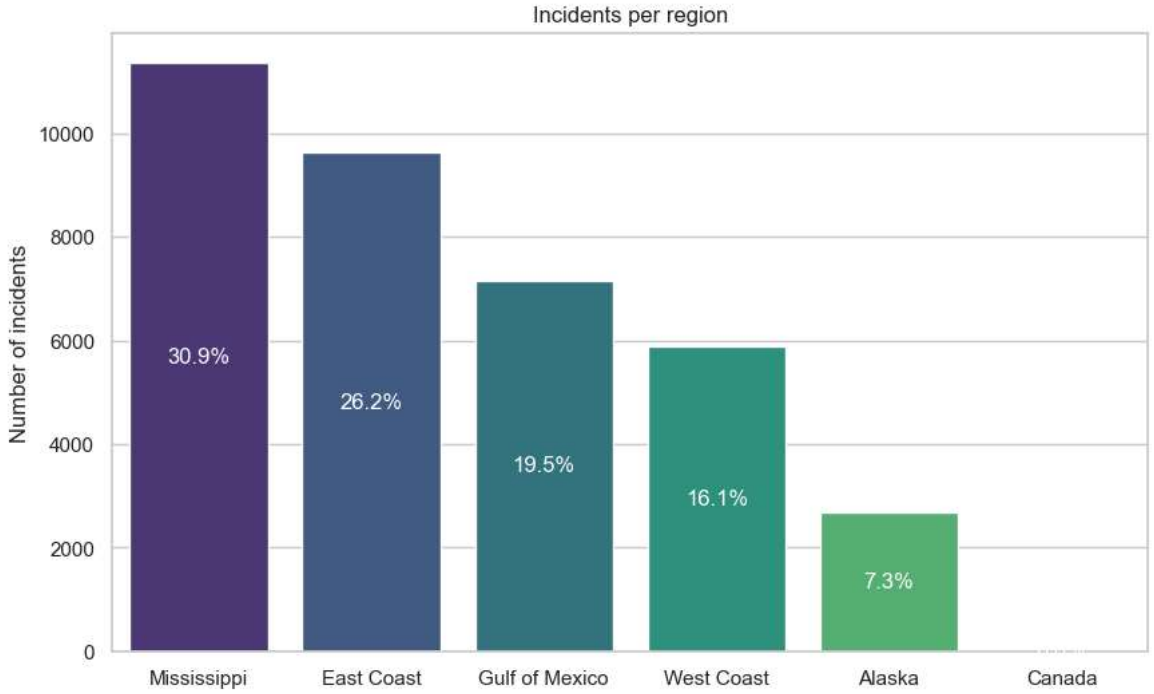
Map

In [23]:
```
# Create and configure map layout
fig = go.Figure(
    layout=dict(
        legend=dict(yanchor="top", y=0.99, xanchor="right", x=0.99, bgcolor='rgba(255, 255, 255, 0.5)'),
        margin={'l': 0, 't': 0, 'b': 0, 'r': 0},
        mapbox={
            'style': "open-street-map",
            'center': {'lon': -112, 'lat': 48},
            'zoom': 2
        }
    )
)

# Add region rectangles
fig.add_trace(go.Scattermapbox(
    name="Alaska",
    mode = "lines", fill = "toself", line=dict(color='rgba(0,0,0,0)'),fillcolor='rgba(255, 0, 0, 0.2)',
    lon = [-180, -180, -122, -122], lat = [70, 49, 49, 70, 70]))

fig.add_trace(go.Scattermapbox(
    name="Canada",
    mode = "lines", fill = "toself", line=dict(color='rgba(0,0,0,0)'),fillcolor='rgba(128, 0, 128, 0.2)',
    lon = [-122, -122, -45, -45], lat = [70, 49, 49, 70, 70]))

fig.add_trace(go.Scattermapbox(
    name="East Coast",
    mode = "lines", fill = "toself", line=dict(color='rgba(0,0,0,0)'),fillcolor='rgba(0, 0, 255, 0.2)',
    lon = [-81.5, -81.5, -45, -45], lat = [49, 15, 15, 49, 70]))

fig.add_trace(go.Scattermapbox(
    name="West Coast",
    mode = "lines", fill = "toself", line=dict(color='rgba(0,0,0,0)'),fillcolor='rgba(255, 255, 0, 0.2)',
    lon = [-180, -180, -100, -100], lat = [49, 15, 15, 49, 49]))

fig.add_trace(go.Scattermapbox(
    name="Gulf of Mexico",
    mode = "lines", fill = "toself", line=dict(color='rgba(0,0,0,0)'),fillcolor='rgba(255, 165, 0, 0.2)',
    lon = [-100, -100, -81.5, -81.5], lat = [31, 15, 15, 31, 31]))

fig.add_trace(go.Scattermapbox(
    name="Mississippi",
    mode = "lines", fill = "toself", line=dict(color='rgba(0,0,0,0)'),fillcolor='rgba(0, 255, 0, 0.2)',
    lon = [-100, -100, -81.5, -81.5], lat = [49, 31, 31, 49, 49]))

# Add watertype Makers
marker_text = merged_activity.apply(lambda row:f"activity_id:{row['activity_id']}<br>date: {row['date']}", axis=1)
marker_color = merged_activity['watertype'].map({'river': 'green', 'ocean': 'blue'})

fig.add_trace(go.Scattermapbox(
    name="watertype",
    mode='markers',
    marker=dict(size=5, color=marker_color, opacity=0.1),
    text=marker_text,
    lat=merged_activity['latitude'], lon=merged_activity['longitude']
))

# Show map
fig.show()
```



3.2. Incidents per Region

In [24]:
```
# Filter data: group by region
filtered_df = (merged_activity
               .drop_duplicates(subset='activity_id', keep='first')
               .groupby('region').size().reset_index(name='frequency')
               .sort_values(by='frequency', ascending=False))

# Plot barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='region', y='frequency', data=filtered_df, palette='viridis')

# Percentage
filtered_df['percentage'] = filtered_df['frequency'] / filtered_df['frequency'].sum() * 100
for i, (value, percentage) in enumerate(zip(filtered_df['frequency'], filtered_df['percentage'])):
    plt.text(i, value / 2, f'{percentage:.1f}%', va='center', ha='center', color='white')

# Customize plot
plt.title('Incidents per region')
plt.xlabel(None)
plt.ylabel('Number of incidents')
plt.show()
```



3.3. Most frequent Incident per Region

In [25]:
```
# Drop duplicate rows keeping the first occurrence of each activity_id
filtered_df = (merged_activity
```

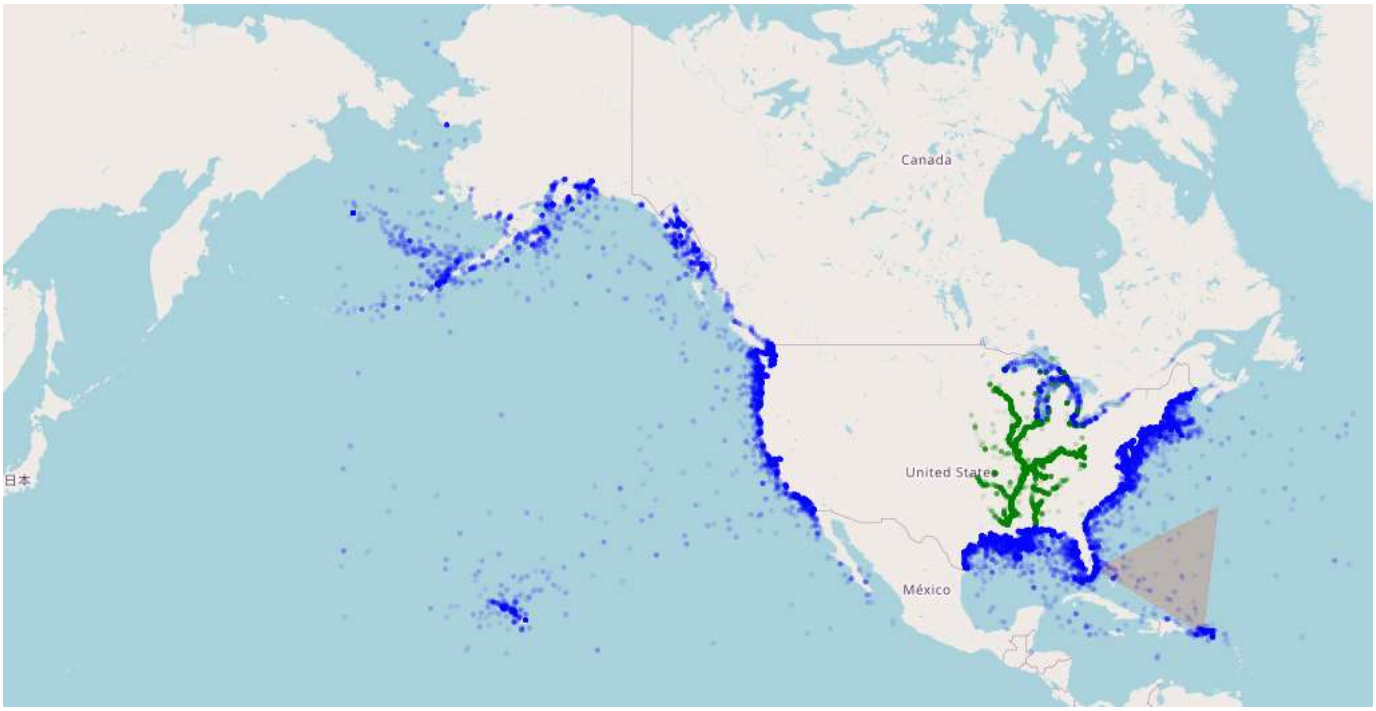
```
                .drop_duplicates(subset='activity_id', keep='first')
                .copy())

# Calculate the number of events per region
filtered_df['num_events_per_region'] = (filtered_df
                                        .groupby('region')['region']
                                        .transform('count'))

# Find the most frequent event per region
filtered_df['most_frequent_event'] = (filtered_df
                                      .groupby('region')['event_type']
                                      .transform(lambda x: x.value_counts().index[0]))

# Select required columns, drop duplicate rows, sort by the number of events
filtered_df = (filtered_df
               [['region', 'most_frequent_event', 'num_events_per_region']]
               .drop_duplicates()
               .sort_values(by='num_events_per_region', ascending=False))

# Display the resulting DataFrame
print(filtered_df)
```

```
              region         most_frequent_event  num_events_per_region
0        Mississippi                   Grounding                  11374
2          East Coast  Material Failure (Vessels)                   9630
4      Gulf of Mexico   Damage to the Environment                   7163
24         West Coast  Material Failure (Vessels)                   5900
26             Alaska   Damage to the Environment                   2692
45677          Canada  Material Failure (Vessels)                      1
```

3.4. Bonus: Bermuda Triangle

In [26]:

```
# Create and configure map layout
fig = go.Figure(
    layout=dict(
        legend=dict(yanchor="top", y=0.99, xanchor="right", x=0.99, bgcolor='rgba(255, 255, 255, 0.5)'),
        margin={'l': 0, 't': 0, 'b': 0, 'r': 0},
        mapbox={
            'style': "open-street-map",
            'center': {'lon': -112, 'lat': 48},
            'zoom': 2
        }
    )
)

# Add region triangle
fig.add_trace(go.Scattermapbox(
    name="Bermuda Triangle",
    mode = "lines", fill = "toself", line=dict(color='rgba(0,0,0,0)'),fillcolor='rgba(255, 69, 0, 0.2)',
    lon = [-64, -80, -66], lat = [33, 26, 18]))

# Add watertype Makers
marker_text = (merged_activity
               .apply(lambda row:f"activity_id:{row['activity_id']}<br> date: {row['date']}<br> watertype: {row['watertype']}",axis=1))
marker_color = (merged_activity['watertype']
                .map({'river': 'green', 'ocean': 'blue'}))

fig.add_trace(go.Scattermapbox(
    name="watertype",
    mode='markers',
    marker=dict(size=5, color=marker_color, opacity=0.1),
    text=marker_text,
    lat=merged_activity['latitude'], lon=merged_activity['longitude']
))

# Show map
fig.show()
```

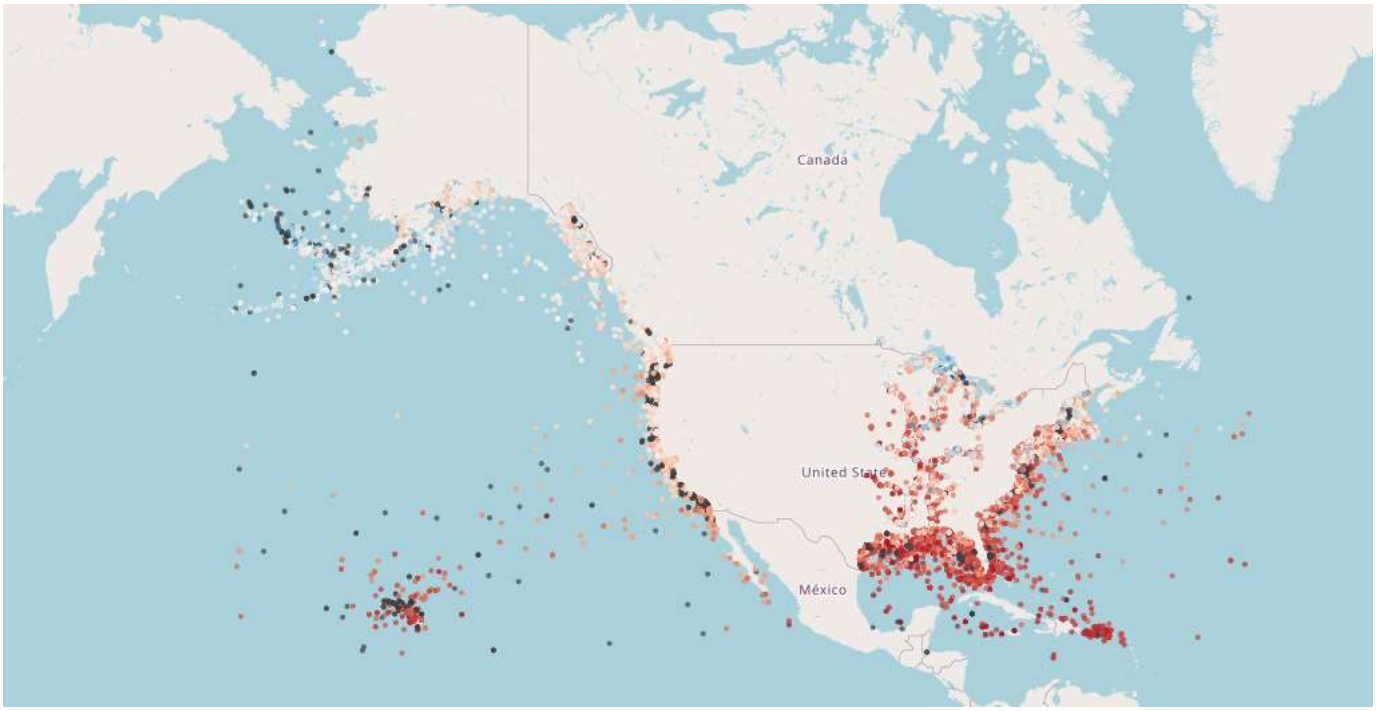


4. Meteorology

4.1. Temperature

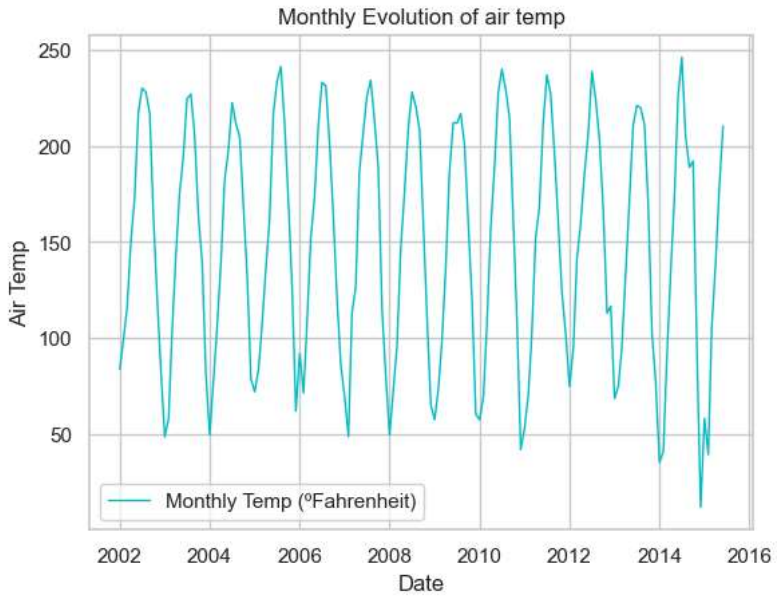
Map

In [27]:
```
# Create and configure map layout
fig = go.Figure(
    layout=dict(
        legend=dict(yanchor="top", y=0.99, xanchor="right", x=0.99),
        margin={'l': 0, 't': 0, 'b': 0, 'r': 0},
        mapbox={
            'style': "open-street-map",
            'center': {'lon': -112, 'lat': 48},
            'zoom': 2
        }
    )
)

# Add watertype Markers
marker_text = (merged_activity
               .apply(lambda row:f"activity_id:{row['activity_id']}<br> date: {row['date']}<br> air_temp: {row['air_temp']}", axis=1))
marker_color = merged_activity['air_temp']

fig.add_trace(go.Scattermapbox(
    name="air_temp",
    mode='markers',
    marker=dict(size=5, color=marker_color, opacity=0.7, colorscale='RdBu_r', colorbar=dict(title="Air Temperature")),
    text=marker_text,
    lat=merged_activity['latitude'], lon=merged_activity['longitude']
))

# Show the figure
fig.show()
```



Time Series

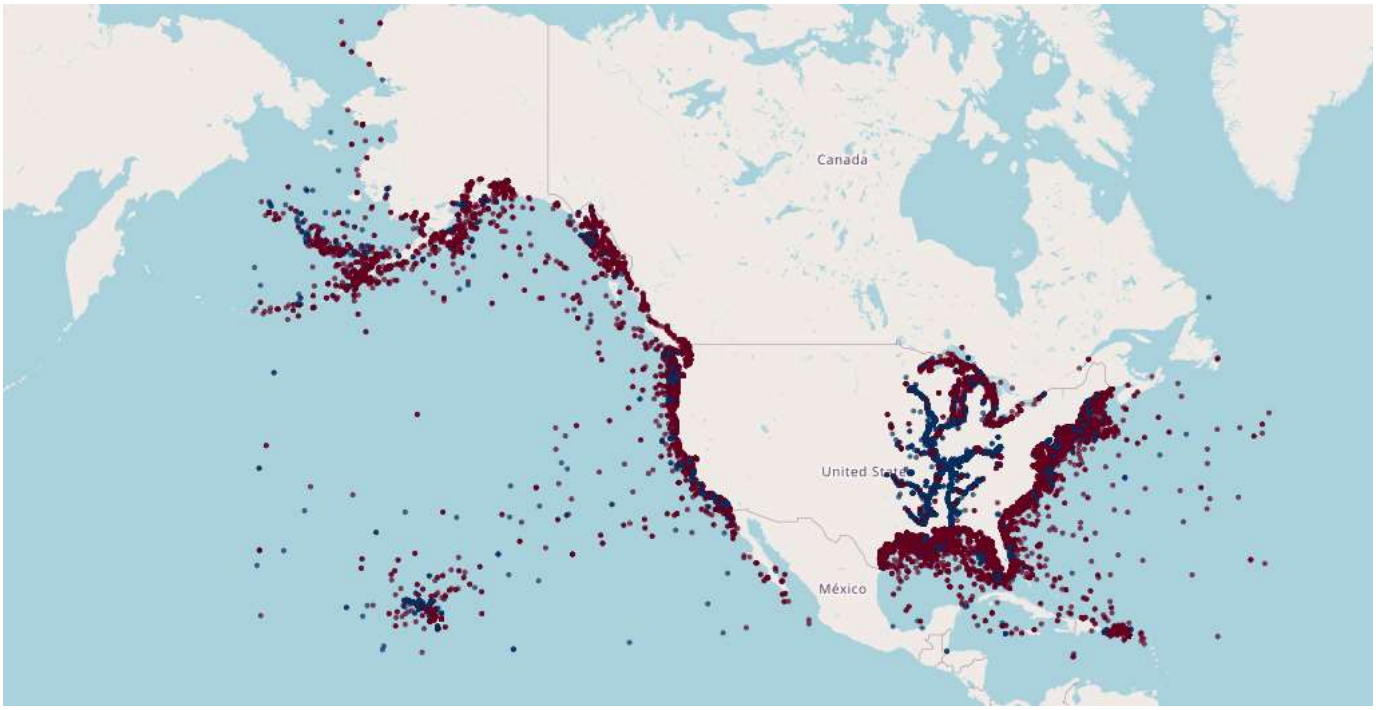
In [28]:
```
# Data filter
filtered_df = (merged_activity
               .drop_duplicates(subset='activity_id', keep='first')
               .copy())

# Calculating year and month
filtered_df['year'] = filtered_df['date'].dt.year
filtered_df['month'] = filtered_df['date'].dt.month

# Calculating monthly temp mean
month_mean_temp = filtered_df.groupby(['year', 'month'])['air_temp'].mean().reset_index()

# Creating a date column from year and month
month_mean_temp['date'] = pd.to_datetime(month_mean_temp[['year', 'month']].assign(day=1))

# Visualization
plt.plot(month_mean_temp['date'], month_mean_temp['air_temp'],
         color='#00bfc4', label='Monthly Temp (ºFahrenheit)', linewidth=1)
plt.xlabel('Date')
plt.ylabel('Air Temp')
plt.title('Monthly Evolution of air temp')
plt.legend()
plt.grid(True)
plt.show()
```



4.2. wind_speed

Map

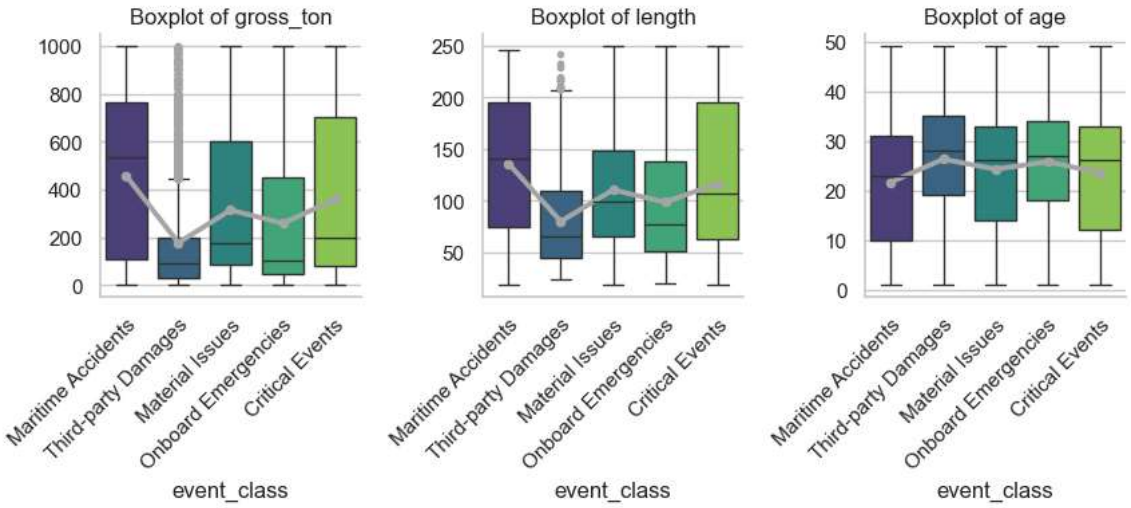
In [29]:
```
# Create and configure map layout
fig = go.Figure(
    layout=dict(
        legend=dict(yanchor="top", y=0.99, xanchor="right", x=0.99),
        margin={'l': 0, 't': 0, 'b': 0, 'r': 0},
        mapbox={
            'style': "open-street-map",
            'center': {'lon': -112, 'lat': 48},
            'zoom': 2
        }
    )
)

# Add watertype Markers
marker_text = (merged_activity
               .apply(lambda row:f"activity_id:{row['activity_id']}<br> date: {row['date']}<br> wind_speed: {row['wind_speed']}", axis=1))

merged_activity['wind_speed'] = pd.to_numeric(merged_activity['wind_speed'], errors='coerce')
marker_color = np.digitize(merged_activity['wind_speed'], np.percentile(merged_activity['wind_speed'], [33, 66]))

fig.add_trace(go.Scattermapbox(
    name="wind_speed",
    mode='markers',
    marker=dict(size=5, color=marker_color, opacity=0.7, colorscale='RdBu_r', colorbar=dict(title="wind_speed")),
    text=marker_text,
    lat=merged_activity['latitude'], lon=merged_activity['longitude']
))

# Show the figure
fig.show()
```



5. event_class VS Other Variables

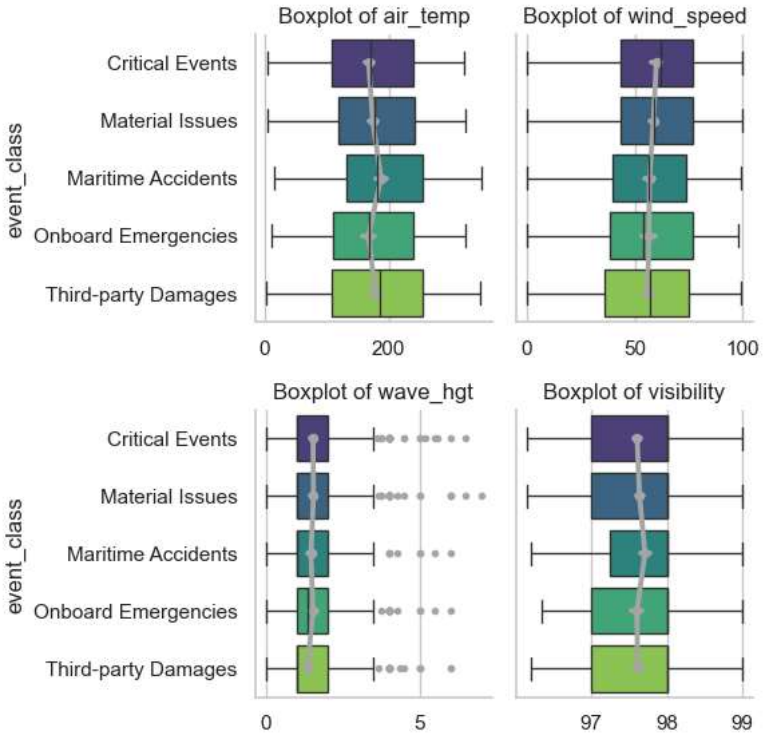
5.1. event_class VS Vessel Features

In [30]:
```
# Filter data
filtered_df = (merged_activity
               [['event_class', 'gross_ton', 'length', 'age']]
               [(merged_activity['gross_ton'] < 1000)
                & (merged_activity['length'] < 250)
                & (merged_activity['age'] > 0) & (merged_activity['age'] < 50)])

# Pivot data
filtered_df = pd.melt(filtered_df, id_vars='event_class', var_name='variable', value_name='value')

# Plot boxplot
g = sns.FacetGrid(filtered_df, col='variable', sharey=False, col_wrap=3)
g.map_dataframe(sns.boxplot, x='event_class', y='value', palette='viridis',
                flierprops=dict(markerfacecolor='darkgrey', markeredgecolor='none', markersize=4))
g.map_dataframe(sns.pointplot, x='event_class', y='value', color='darkgrey', markers='.')

# Customize plot
g.set_titles("Boxplot of {col_name}")
g.set_axis_labels("event_class", "")
g.set_xticklabels(rotation=45, rotation_mode='anchor', ha ='right')

plt.show()
```



5.2. event_class VS Meteorology

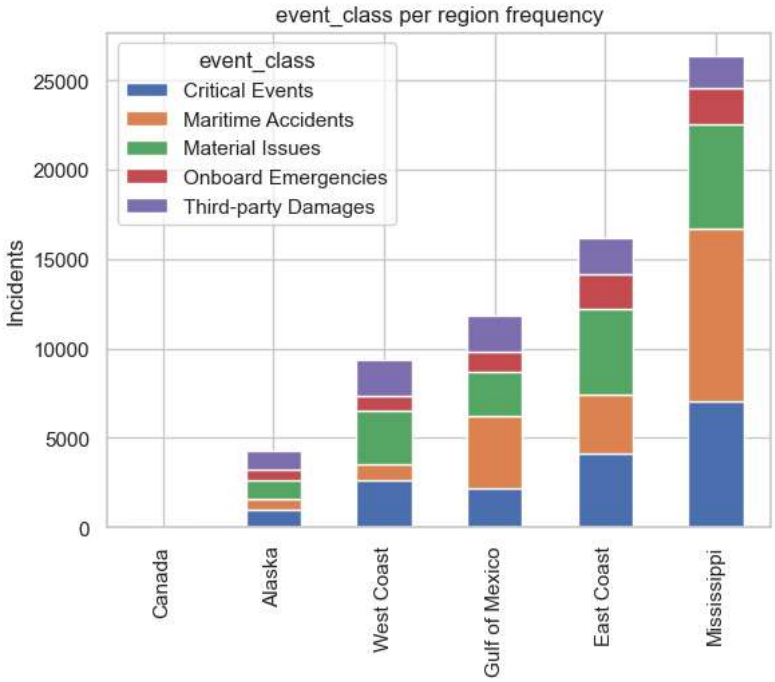
In [31]:
```
# Filter data
filtered_df = (merged_activity
               [['event_class', 'air_temp', 'wind_speed', 'wave_hgt', 'visibility']]
               [(merged_activity['air_temp'] > 0)
                & (merged_activity['wind_speed'] < 100)
                & (merged_activity['wave_hgt'] < 10)
                & (merged_activity['visibility'] > 96)])

# Pivot data
filtered_df = pd.melt(filtered_df, id_vars='event_class', var_name='variable', value_name='value')

# Plot boxplot
g = sns.FacetGrid(filtered_df, col='variable', sharex=False, col_wrap=2)
g.map_dataframe(sns.boxplot, x='value', y='event_class', palette='viridis',
                flierprops=dict(markerfacecolor='darkgrey', markeredgecolor='none', markersize=4))
g.map_dataframe(sns.pointplot, x='value', y='event_class', color='darkgrey', markers='.')

# Customize plot
g.set_titles("Boxplot of {col_name}")
g.set_axis_labels("", "event_class")

plt.show()
```



5.3. event_class per region

In [32]:
```
# Filter and pivot data:
filtered_df = (merged_activity
               .groupby(['region', 'event_class'])
               .size().reset_index(name='frequency')
               .pivot(index='region', columns='event_class', values='frequency'))

# Sorting per total
filtered_df['Total'] = filtered_df.sum(axis=1)
```

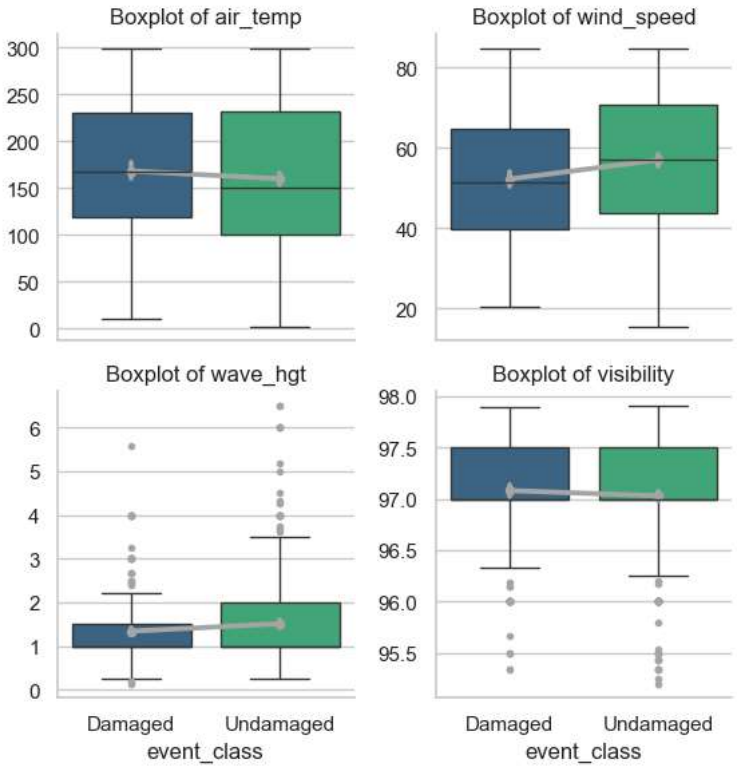
```
filtered_df = filtered_df.sort_values(by='Total', ascending=True)
filtered_df.drop('Total', axis=1, inplace=True)

# Plot
filtered_df.plot(kind='bar', stacked=True)

# Customize plot
plt.xlabel('')
plt.ylabel('Incidents')
plt.title('event_class per region frequency')
plt.legend(title='event_class')

plt.show()
```



5.X. damage_status VS Meteorology

In [33]:

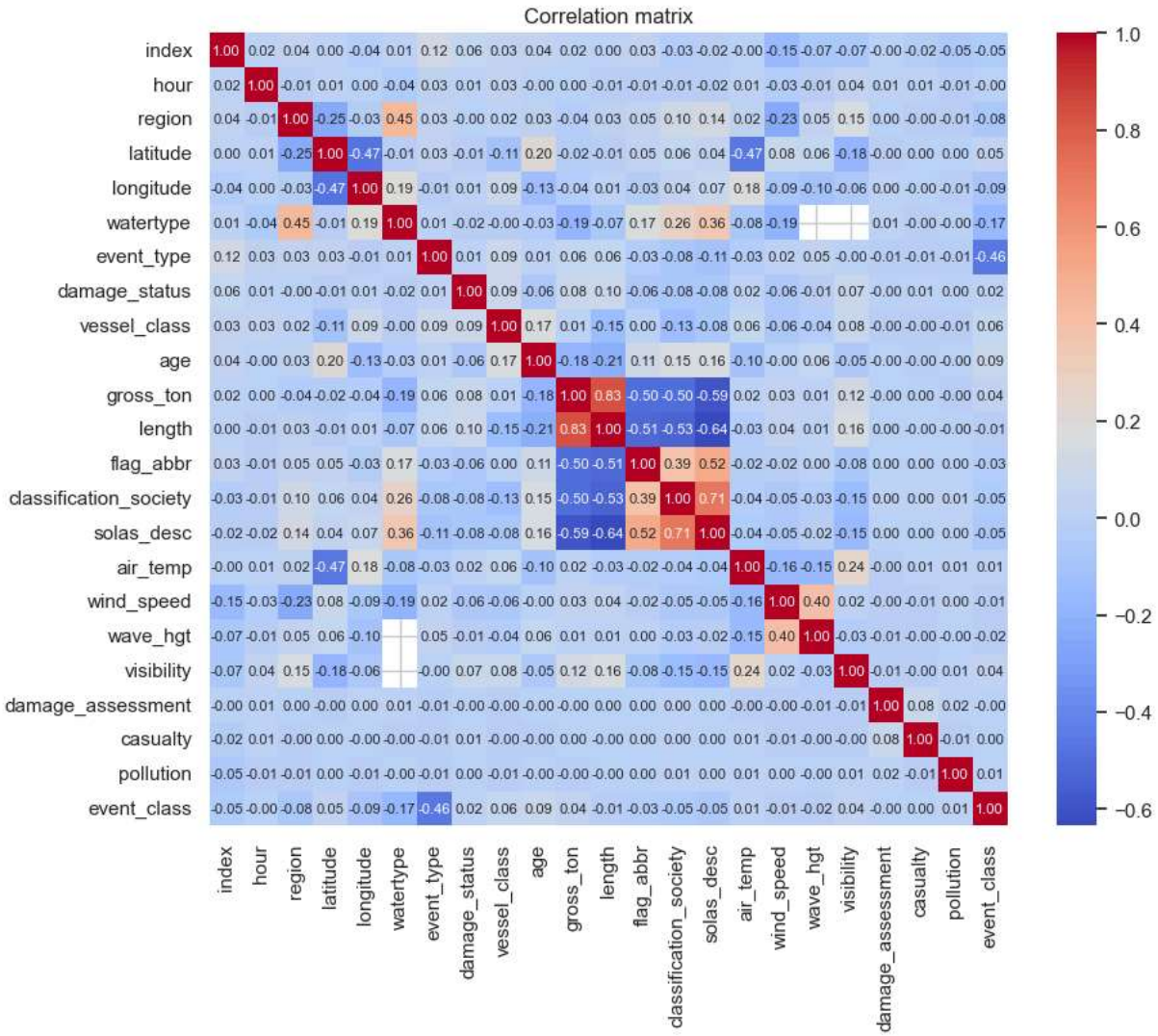
```
# Filter data
filtered_df = (merged_activity
               [['damage_status', 'air_temp', 'wind_speed', 'wave_hgt', 'visibility']]
               [((merged_activity['damage_status'] == "Damaged")
                 | (merged_activity['damage_status'] == "Undamaged"))
                 & (merged_activity['air_temp'] > 0) & (merged_activity['air_temp'] < 300)
                 & (merged_activity['wind_speed'] > 15) & (merged_activity['wind_speed'] < 85)
                 & (merged_activity['wave_hgt'] > 0) & (merged_activity['wave_hgt'] < 10)
                 & (merged_activity['visibility'] > 95) & (merged_activity['visibility'] < 98)])

# Pivot data
filtered_df = pd.melt(filtered_df, id_vars='damage_status', var_name='variable', value_name='value')

# Plot boxplot
g = sns.FacetGrid(filtered_df, col='variable', sharey=False, col_wrap=2)
g.map_dataframe(sns.boxplot, x='damage_status', y='value', palette='viridis',
                flierprops=dict(markerfacecolor='darkgrey', markeredgecolor='none', markersize=4))
g.map_dataframe(sns.pointplot, x='damage_status', y='value', color='darkgrey', markers='.')

# Customize plot
g.set_titles("Boxplot of {col_name}")
g.set_axis_labels("event_class", "")

plt.show()
```



6. Correlations

6.1. Correlation matrix

In [34]:
```
# Drop unvaluable variables
filtered_df= (merged_activity
              .drop(['activity_id', 'date', 'build_year', 'vessel_id', 'imo_number', 'vessel_name'],
                    axis=1))

# Convert hour to decimal format
filtered_df['hour'] = (filtered_df['hour'].str.split(":")
                       .apply(lambda x: int(x[0]) + int(x[1]) / 60))

# Convert to numerical using cat.codes
filtered_df['region'] = filtered_df['region'].astype('category').cat.codes
filtered_df['watertype'] = filtered_df['watertype'].astype('category').cat.codes
filtered_df['event_type'] = filtered_df['event_type'].astype('category').cat.codes
filtered_df['damage_status'] = filtered_df['damage_status'].astype('category').cat.codes
filtered_df['vessel_class'] = filtered_df['vessel_class'].astype('category').cat.codes
filtered_df['flag_abbr'] = filtered_df['flag_abbr'].astype('category').cat.codes
filtered_df['classification_society'] = filtered_df['classification_society'].astype('category').cat.codes
filtered_df['solas_desc'] = filtered_df['solas_desc'].astype('category').cat.codes
filtered_df['casualty'] = filtered_df['casualty'].astype('category').cat.codes
filtered_df['pollution'] = filtered_df['pollution'].astype('category').cat.codes
filtered_df['event_class'] = filtered_df['event_class'].astype('category').cat.codes

# Convert all to numeric
filtered_df = filtered_df.apply(lambda x: pd.to_numeric(x, errors='coerce'))

# Heatmap for correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(filtered_df.corr(), annot=True, cmap='coolwarm', annot_kws={"fontsize":8}, fmt=".2f")
plt.title('Correlation matrix')
plt.show()
```



7. Extra: Automatic EDA report

In [35]:
```
# Create ydata_profiling report
profile = ProfileReport(merged_activity, title='merged_activity: EDA')

# Export inform
if file_export_enabled :
    profile.to_file("Exported Reports/merged_activity_EDA.html")
else:
    print('EDA report already exported')
```

```
EDA report already exported
```