report

June 28, 2023

1 Report on aviation incidents and accidents and their geographic distribution

This report uses open data from the Bundesamt für Flugunfalluntersuchung via mobilithek and the NTSB to look for some interesting corelations and distributions of aircraft accidents and incidents. It should not be intrepreted as a source of safety comparison or as a safety raiting or anything in that direction.

The Questions that intrests us: - How many, how serious, and where are the aviation incidents and accidents? - How does this change over the years? - Does this differ in different countries?

before running this notebook, install the python requirenments and the requrienments for mdbtools. Those can vary with your installation but should be `mdbtools mdbtools-dev unixodbc-dev` then run `make` in the data directory.

Because the german data only contains location names and not coordinates, a database is generated to fill in those coordinates, but sicnce the api used for the name resolution has a max request of 1/s, this is not generated every time. You have to run the `getLatLongForLocations.py` script manually to avoid unneccesary querys.

Since github does not show plotly graphs, either chek this file out or see the .pdf export

1.0.1 Data loading

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.io as pio
import plotly.express as px
import numpy as np

pio.renderers.default = "pdf"


genAvDE = pd.read_sql_table("generalAviation", "sqlite:///../data/de.sqlite")
comAvDE = pd.read_sql_table("commercialAviation", "sqlite:///../data/de.sqlite")
dataDE = pd.read_sql_table("all", "sqlite:///../data/de.sqlite")

us_ac_columns = ["ev_id", "Aircraft_Key", "damage", "acft_make", "acft_model",␣
 ↪"acft_series",
```

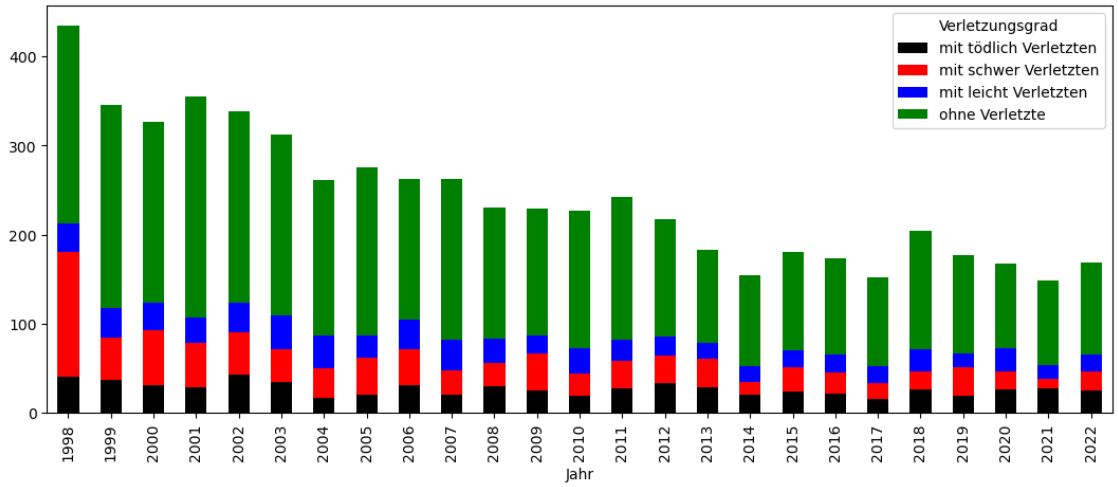
```
                "acft_category", "acft_reg_cls", "elt_manufacturer",␣
 ↪"elt_model"]
us_ij_columns = ["ev_id", "Aircraft_Key", "inj_person_category",␣
 ↪"injury_level", "inj_person_count"]


eventsUS = pd.read_sql_table("events_clean", "sqlite:///../data/us.sqlite")
aircraftUS =  pd.read_sql_table("aircraft", "sqlite:///../data/us.sqlite")
injuryUS =  pd.read_sql_table("injury", "sqlite:///../data/us.sqlite")


aircraftUS = aircraftUS[us_ac_columns]
injuryUS = injuryUS[us_ij_columns]


partUS = pd.merge(eventsUS, aircraftUS, how="left", on=["ev_id"], validate="1:
 ↪m")
dataUS = pd.merge(partUS, injuryUS, how="left", on=["ev_id","Aircraft_Key"],␣
 ↪validate="1:m")

eventsUS =eventsUS.astype(dtype={"ev_year":int})

locations_de = pd.read_sql_table("locations-new", "sqlite:///../data/loc.de.
 ↪sqlite")
locations_de.columns = ["Unfallort", "latitude", "longitude"]
dataDE_withLoc = pd.merge(dataDE, locations_de, how="left", on="Unfallort")
```

2 Statistics

How much accidents happen and how hard are persons injured every year?

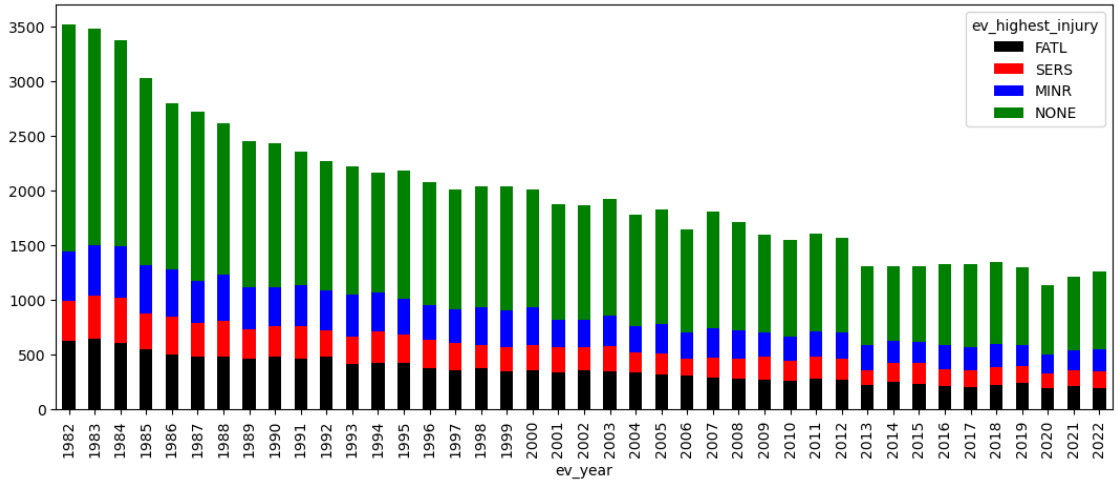
[ ]:
```
df = dataDE.groupby(["Jahr", "Verletzungsgrad"])["Aktenzeichen"].count()
df = df.reset_index()
df = pd.pivot(df,index="Jahr",columns="Verletzungsgrad")
df.columns =df.columns.get_level_values(1)
df = df.drop(columns="Verletzungen unbekannt")
df.plot.bar(stacked=True, figsize=(13,5), color={'mit leicht Verletzten':
 ↪"blue", 'mit schwer Verletzten': "red", "mit tödlich Verletzten":␣
 ↪"black",'ohne Verletzte': "green"},
            y=["mit tödlich Verletzten", 'mit schwer Verletzten','mit leicht␣
 ↪Verletzten', 'ohne Verletzte'])
```

[ ]:
```
<Axes: xlabel='Jahr'>
```



[ ]:
```
evUS = eventsUS
evUS = evUS[evUS["ev_year"] > 1981]
evUS = evUS[evUS["ev_year"] < 2023]

df = evUS.groupby(["ev_year","ev_highest_injury"])["ev_id"].count()
df = df.reset_index()
df = pd.pivot(df,index="ev_year",columns="ev_highest_injury")
df.columns =df.columns.get_level_values(1)
df.plot.bar(stacked=True, figsize=(13,5),
            y=["FATL","SERS","MINR","NONE"],
            color={"FATL":"black","SERS":"red","MINR":"blue","NONE":"green"}
            )
```

[ ]:
```
<Axes: xlabel='ev_year'>
```



The number of incidents seems to be decreasing over time and most incidents / accidents don't even have someone injured.

This seems to hold for both, but the dip from the flight restrictions during the cov-19 pandemic is more noticable in the US data.

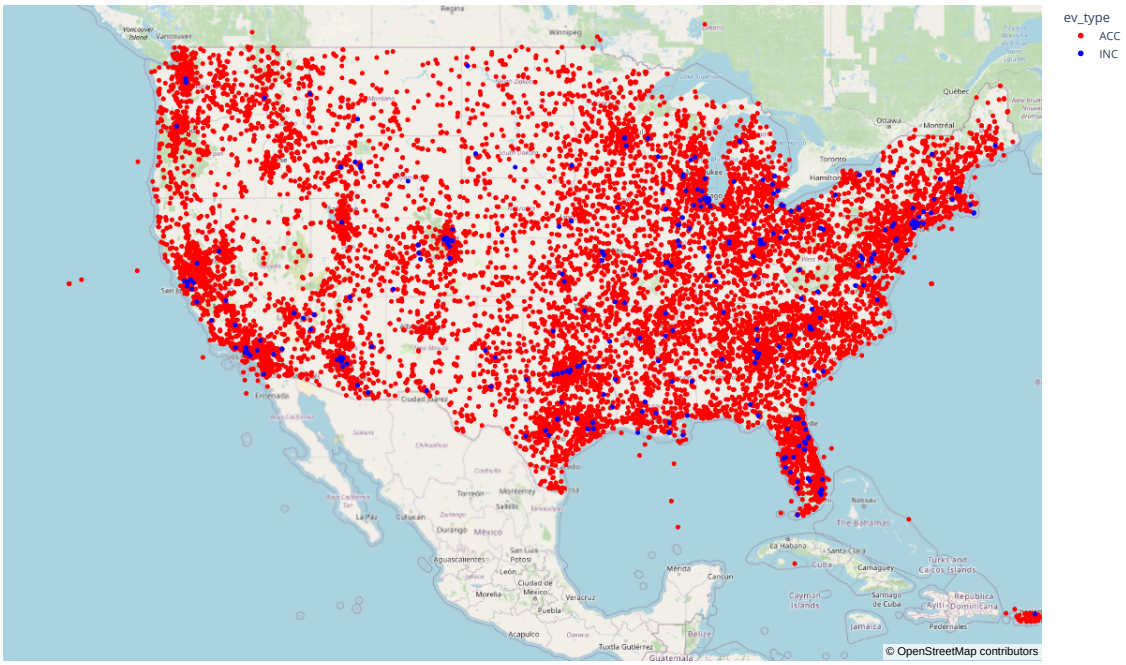
3 Graphs

3.1 Where and how Serious are the accidents (ACC) and incidents (INC) in the us?

To answer this question, we use plotly to draw a scatterplot of all accidents and incidents in the dataset for which we have coordinates, overlaying it on a map from OpenStreetMap. Then we color the points diferently to gain some insight into where the hotspots may lie, and change the size to visualize how many persons where injured.

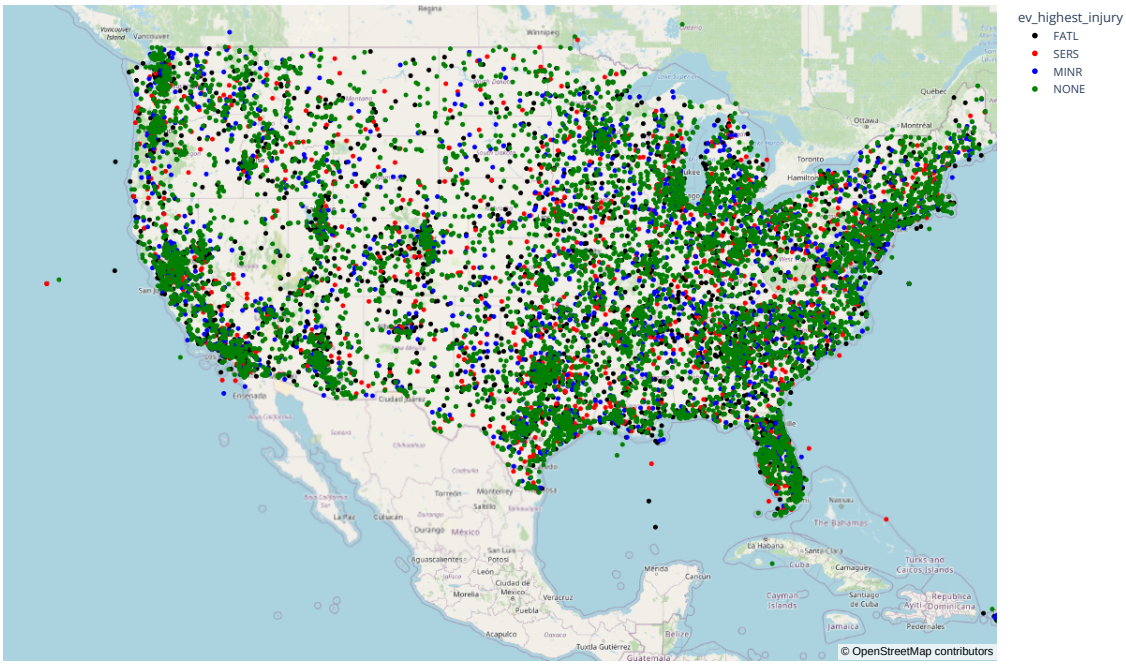
[ ]:
```
df = eventsUS
df = df.replace(r'^\s*$', np.nan, regex=True)
df.dropna(inplace=True)
df = df.astype({"dec_latitude": float, "dec_longitude": float, "inj_tot_t":␣
↪int})
```

[ ]:
```
fig1 = px.scatter_mapbox(df,
                         lat="dec_latitude",
                         lon="dec_longitude",
                         color="ev_type",
                         color_discrete_map={"ACC":"red","INC":"blue"},
                         category_orders={"ev_type":["ACC","INC"]},
                         hover_name="ev_id",
                         hover_data=["ev_id", "ev_year", "ev_highest_injury"],
                         center={"lat":35, "lon":-100},
                         zoom=3.5,
                         height=700,
                         width=1200)

fig1.update_layout(mapbox_style="open-street-map")
fig1.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig1.show()
```



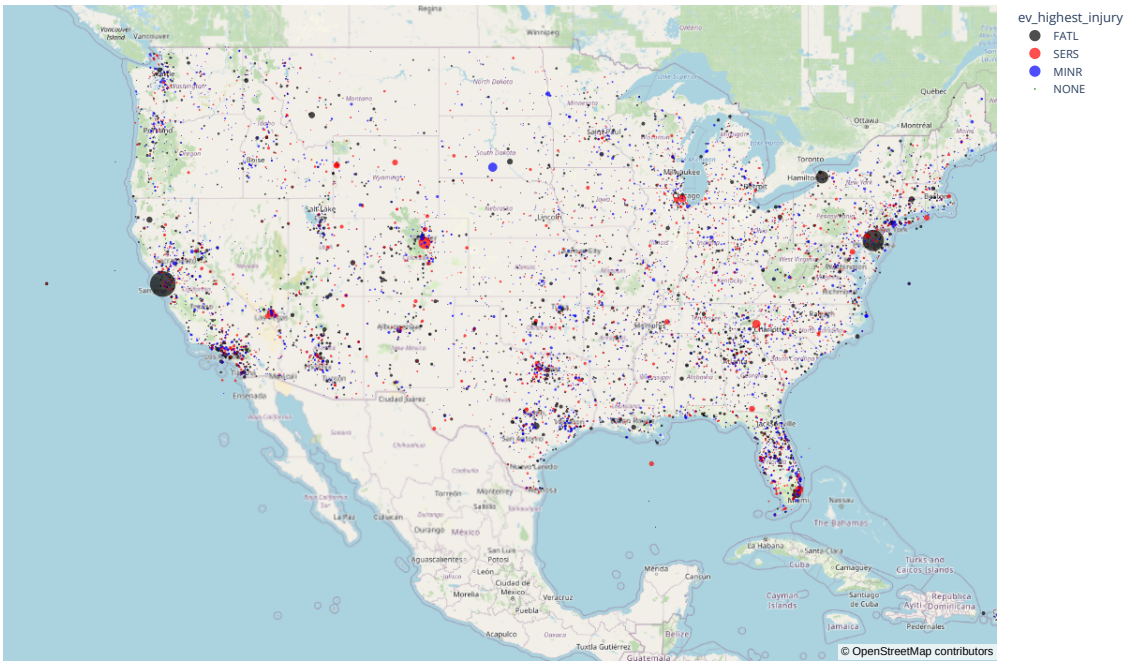
Location of accidents and incidents in the us

[ ]:
```
fig2 = px.scatter_mapbox(df,
                         lat="dec_latitude",
                         lon="dec_longitude",
                         color="ev_highest_injury",
                         color_discrete_map={"FATL":"black","SERS":"red","MINR":
 ↪"blue","NONE":"green"},
                         category_orders={"ev_highest_injury":
 ↪["FATL","SERS","MINR", "NONE"]},
                         hover_name="ev_id",
                         hover_data=["ev_id", "ev_year", "ev_highest_injury"],
                         center={"lat":35, "lon":-100},
                         zoom=3.5,
                         height=700,
                         width=1200)

fig2.update_layout(mapbox_style="open-street-map")
fig2.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig2.show()
```



Locations of Accidents and incidents collored according to the maximum injury level

[ ]:
```
fig = px.scatter_mapbox(df,
                        lat="dec_latitude",
                        lon="dec_longitude",
                        color="ev_highest_injury",
                        color_discrete_map={"FATL":"black","SERS":"red","MINR":
 ↪"blue","NONE":"green"},
                        category_orders={"ev_highest_injury":
 ↪["FATL","SERS","MINR", "NONE"]},
                        size="inj_tot_t",
                        hover_name="ev_id",
                        hover_data=["ev_id", "ev_year", "ev_highest_injury"],
                        center={"lat":35, "lon":-100},
                        zoom=3.5,
                        height=700,
                        width=1200)

fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



Locations of accidents and incidents collored according to the maximum injury level, with the size depending on the persons injured.

Caused by the relatively long timeframe of the records, these maps are a bit cluterded.

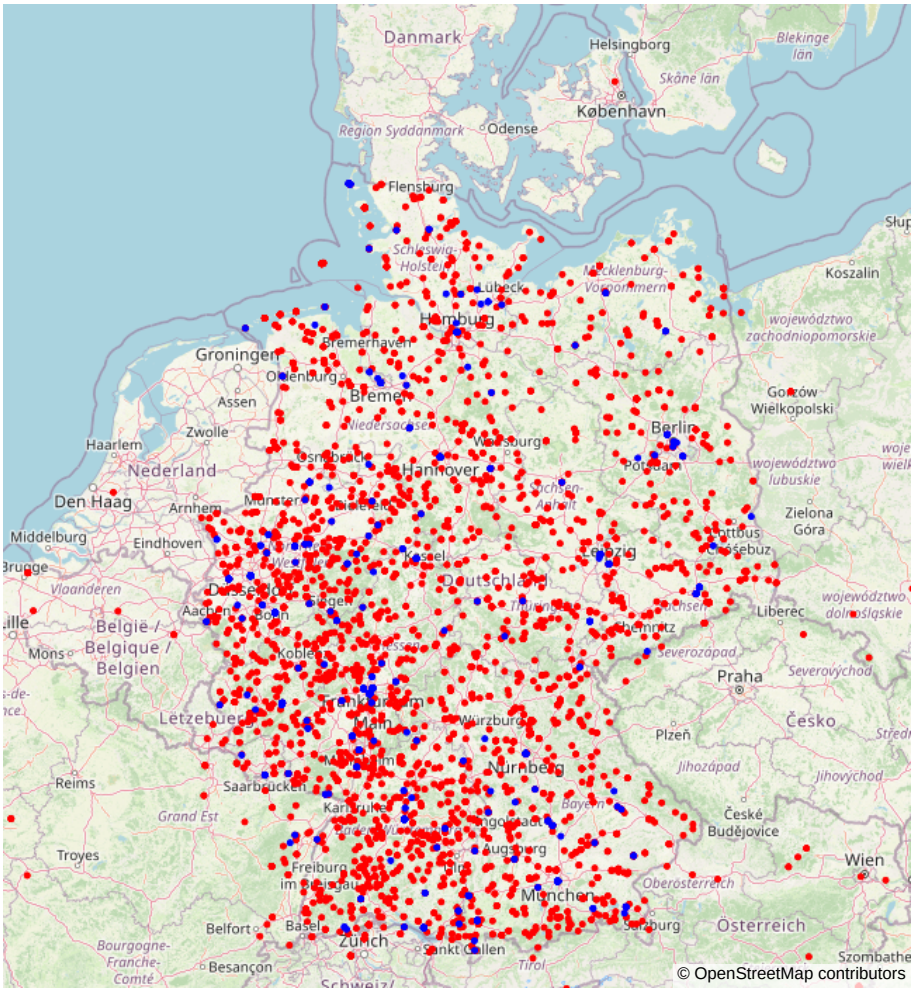
3.2 Where and how Serious are the accidents (Unfall) and incidents (Schwere Störung) in Germany?

To answer this question, we try to match the location names form the database to coordinates by using geopy / Nominatim. Then we use plotly to draw a scatterplot of all accidents and incidents in the dataset for which we have coordinates, overlaying it on a map from OpenStreetMap. In another chart we color the points diferently to gain some further insight.

[ ]:
```
dataDE_withLoc = dataDE_withLoc.drop(columns=["Luftfahrzeugart", "Betriebsart"])
dataDE_withLoc = dataDE_withLoc.dropna()

df =dataDE_withLoc
```

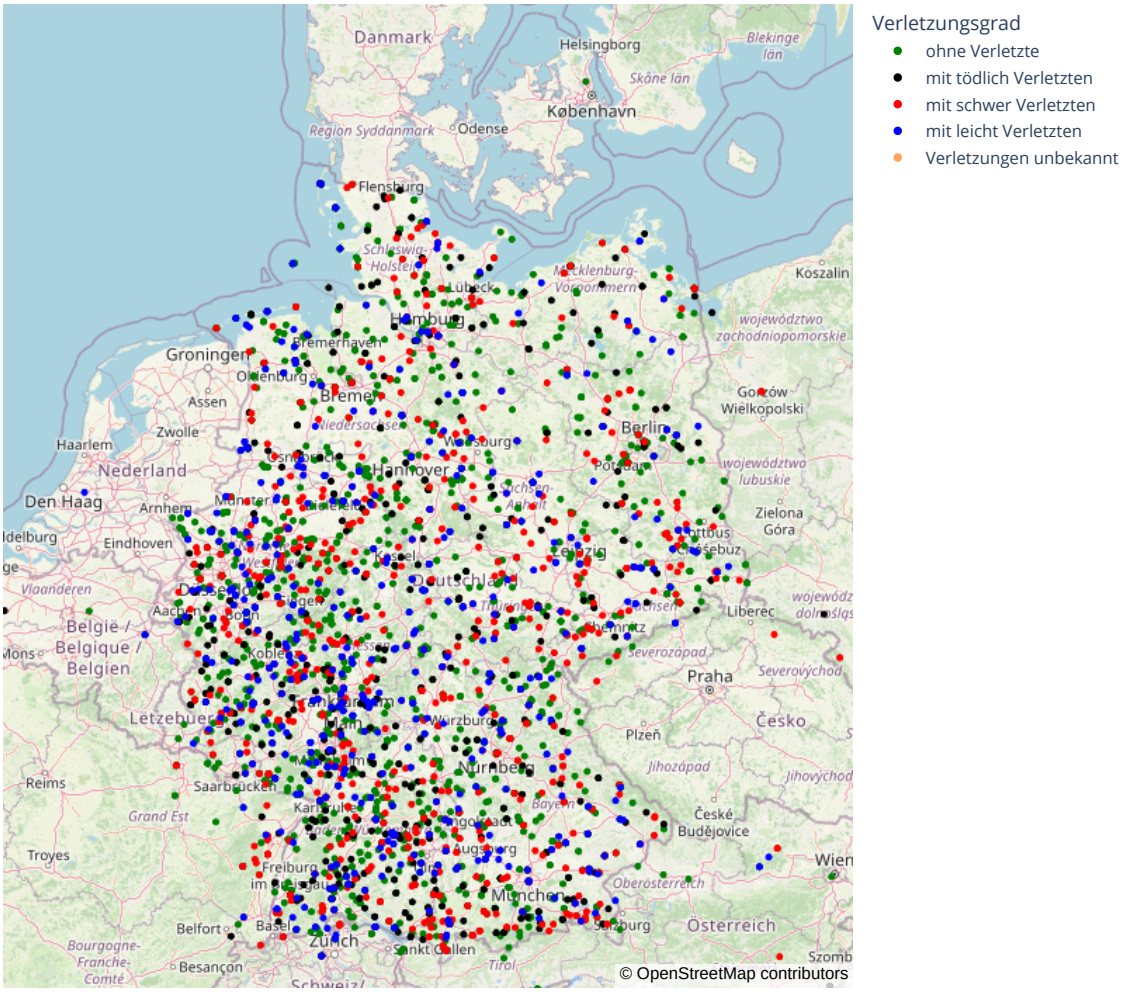
[ ]:
```
figd1 = px.scatter_mapbox(df,
                          lat="latitude",
                          lon="longitude",
                          color="Ereignisschwere",
                          color_discrete_map={"Schwere Störung":"blue","Unfall":
 ↪"red"},
                          hover_name="Ereignisschwere",
                          hover_data=["Jahr", "Ereignisschwere",␣
 ↪"Verletzungsgrad"],
```

```
                        center={"lat":52, "lon":10},
                        zoom=5,
                        height=700,
                        width=800)

figd1.update_layout(mapbox_style="open-street-map")
figd1.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
figd1.show()
```



[ ]:
```
figd1 = px.scatter_mapbox(df,
                        lat="latitude",
                        lon="longitude",
                        color="Verletzungsgrad",
                        color_discrete_map={"mit tödlich Verletzten":
 ↪"black","mit schwer Verletzten": "red",
```

```
                                                "mit leicht Verletzten": "blue",␣
↪"ohne Verletzte":"green"},
                        hover_name="Ereignisschwere",
                        hover_data=["Jahr", "Ereignisschwere",␣
↪"Verletzungsgrad", "Unfallort"],
                        center={"lat":52, "lon":10},
                        zoom=5,
                        height=700,
                        width=800)

figd1.update_layout(mapbox_style="open-street-map")
figd1.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
figd1.show()
```

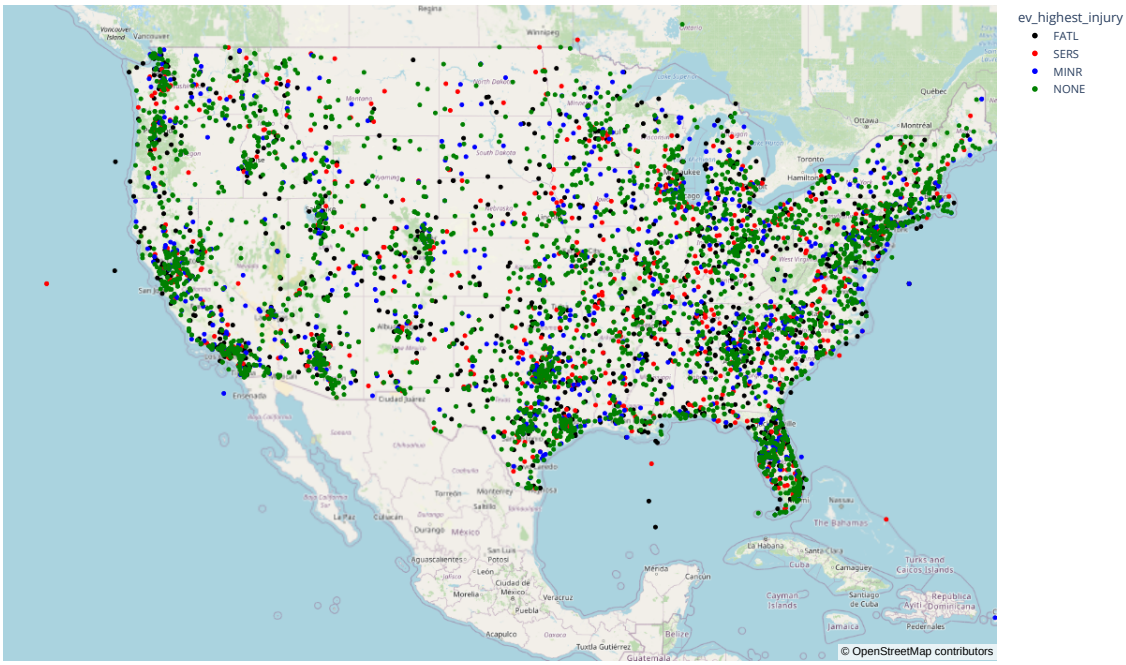


4 Graphs with data from recent years (2000-2022)

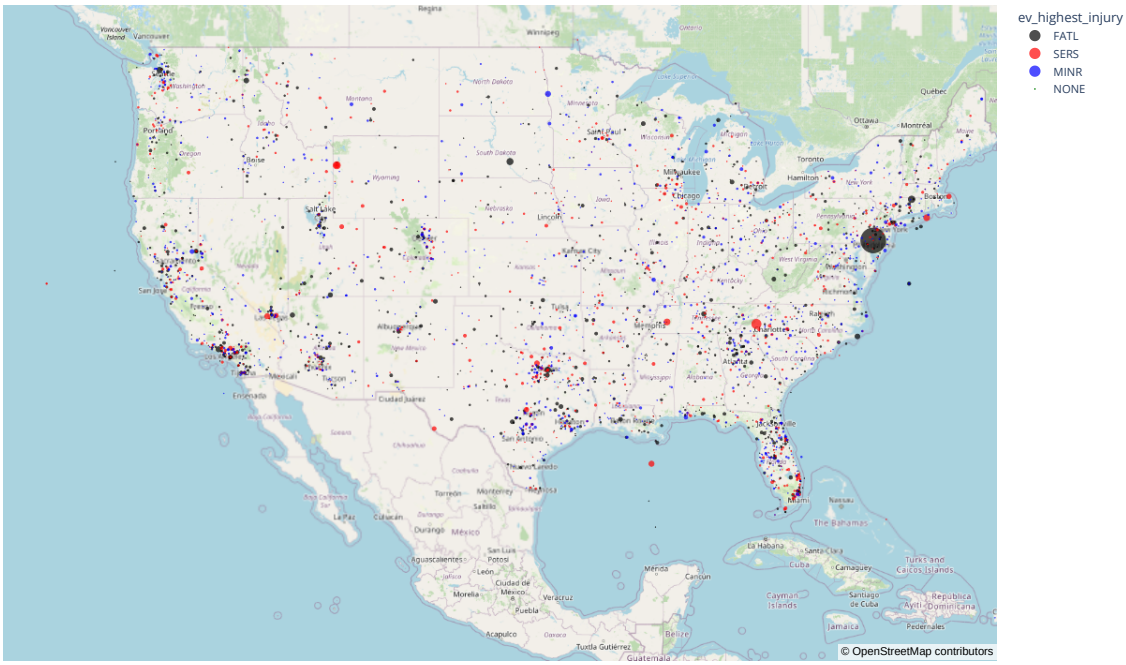
The Data was reduced to the timeframe from 20017 to 2022 so that the resulting range would be relatively recent. This was done to improve visual clutter. Anoter plot was added that visualises the different years the events happend.

[ ]:
```
dataUS_short = eventsUS.astype({"ev_year": int})
dataUS_short = dataUS_short[dataUS_short["ev_year"] > 2016]
dataUS_short = dataUS_short[dataUS_short["ev_year"] < 2023]
dataUS_short = dataUS_short.replace(r'^\s*$', np.nan, regex=True)

dataUS_short.dropna(inplace=True)
dataUS_short = dataUS_short.astype({"dec_latitude": float, "dec_longitude":␣
 ↪float, "inj_tot_t": int})
dataUS_short= dataUS_short[["ev_id", "ev_type", "ev_year", "dec_latitude",␣
 ↪"dec_longitude", "ev_highest_injury", "inj_tot_t"]]

dataDE_short = dataDE_withLoc
dataDE_short = dataDE_short[dataDE_short["Jahr"] > 2016]
dataDE_short = dataDE_short[dataDE_short["Jahr"] < 2023]
```

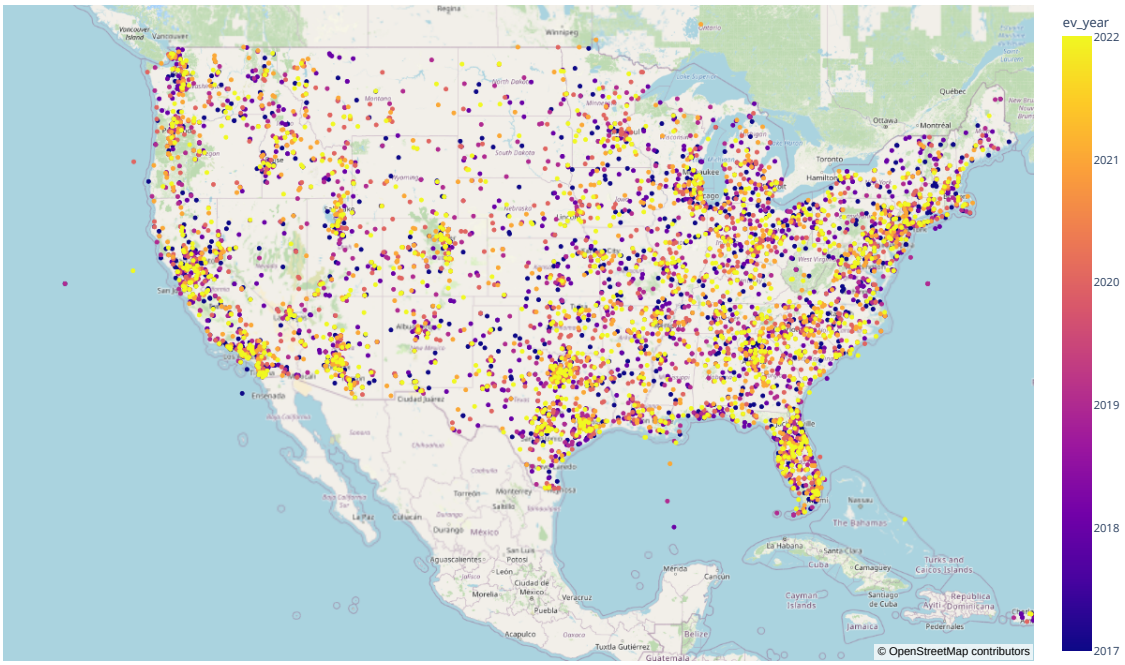
[ ]:
```
fig = px.scatter_mapbox(dataUS_short,
                        lat="dec_latitude",
                        lon="dec_longitude",
                        color="ev_highest_injury",
                        color_discrete_map={"FATL":"black","SERS":"red","MINR":
 ↪"blue","NONE":"green"},
                        category_orders={"ev_highest_injury":
 ↪["FATL","SERS","MINR", "NONE"]},
                        hover_name="ev_id",
                        hover_data=["ev_id", "ev_year", "ev_highest_injury"],
                        center={"lat":35, "lon":-100},
                        zoom=3.5,
                        height=700,
                        width=1200)

fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



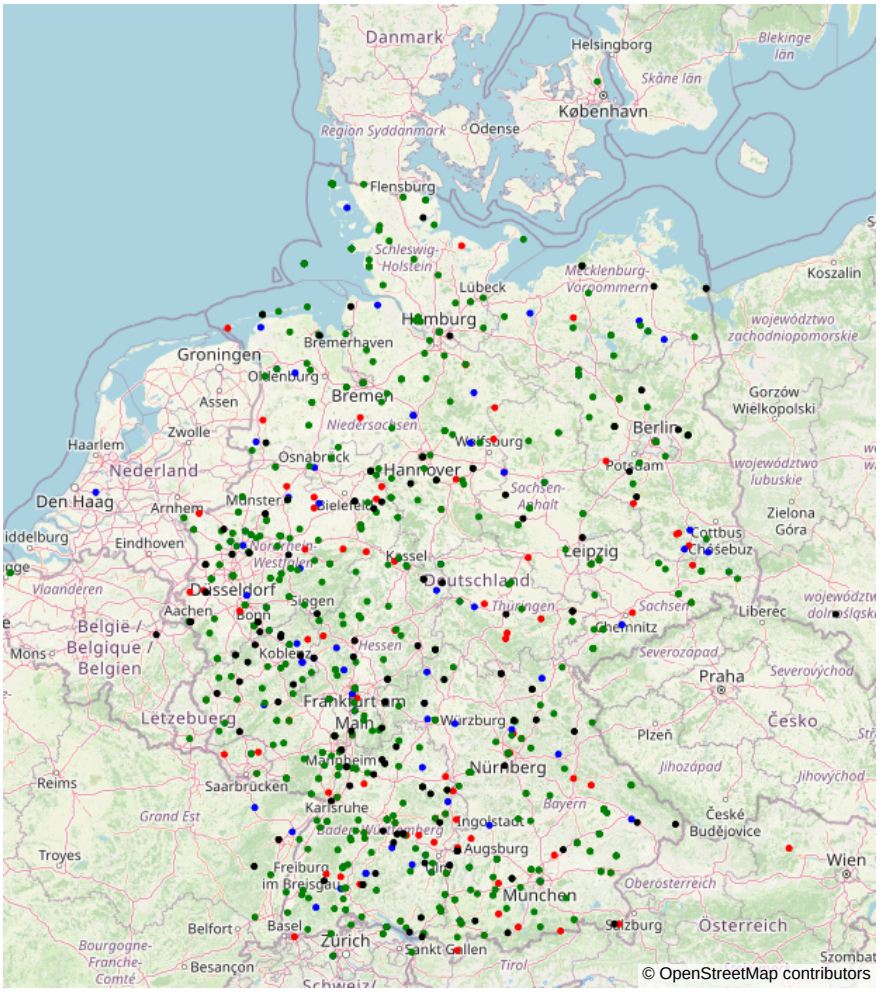
[ ]:
```
fig = px.scatter_mapbox(dataUS_short,
                        lat="dec_latitude",
                        lon="dec_longitude",
                        color="ev_highest_injury",
                        color_discrete_map={"FATL":"black","SERS":"red","MINR":
 ↪"blue","NONE":"green"},
                        category_orders={"ev_highest_injury":
 ↪["FATL","SERS","MINR", "NONE"]},
                        size="inj_tot_t",
                        hover_name="ev_id",
                        hover_data=["ev_id", "ev_year", "ev_highest_injury"],
                        center={"lat":35, "lon":-100},
                        zoom=3.5,
                        height=700,
                        width=1200)

fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



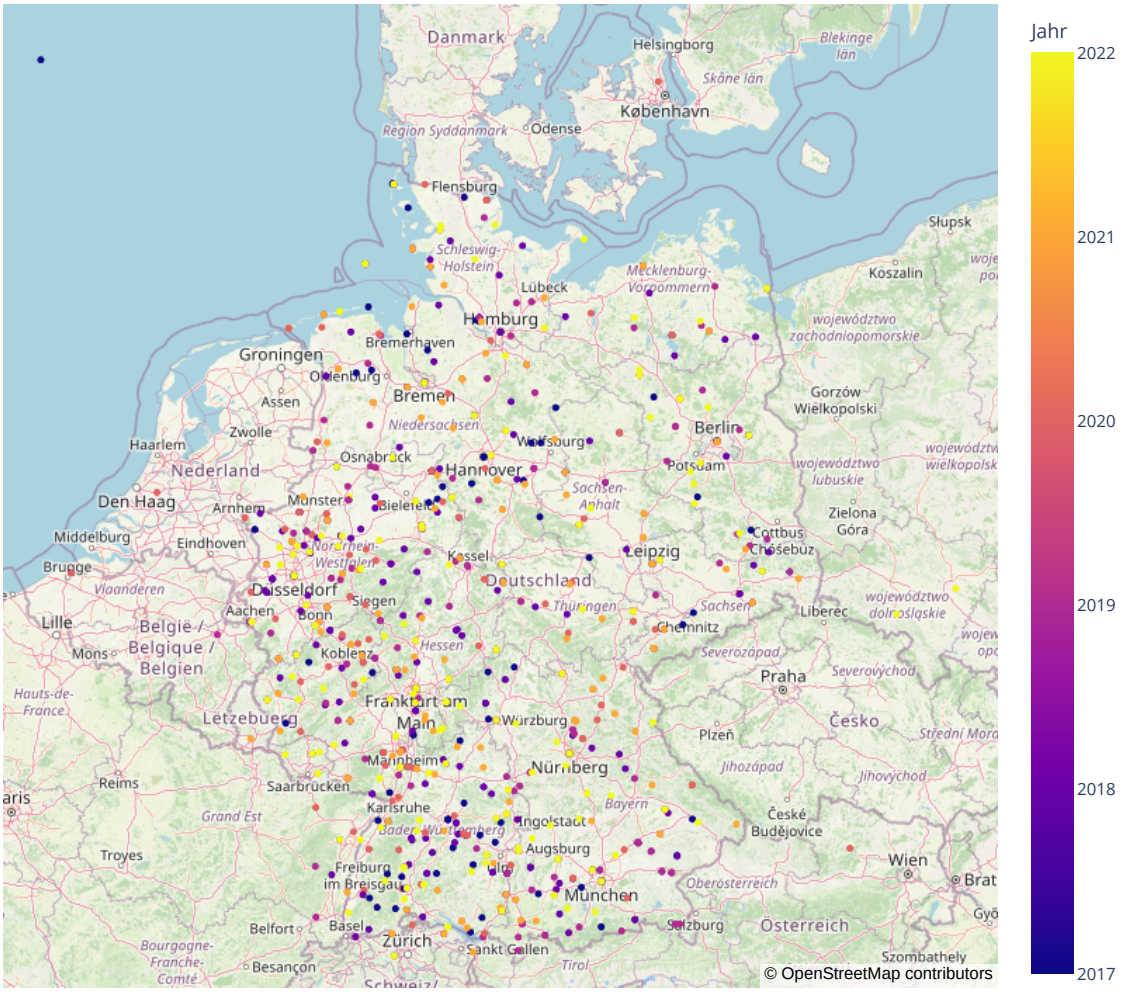
[ ]:
```
fig = px.scatter_mapbox(dataUS_short,
                        lat="dec_latitude",
                        lon="dec_longitude",
                        color="ev_year",
                        hover_name="ev_id",
                        hover_data=["ev_id", "ev_year", "ev_highest_injury"],
                        center={"lat":35, "lon":-100},
                        zoom=3.5,
                        height=700,
                        width=1200)

fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



[ ]:
```
figd1 = px.scatter_mapbox(dataDE_short,
                          lat="latitude",
                          lon="longitude",
                          color="Verletzungsgrad",
                          color_discrete_map={"mit tödlich Verletzten":
 ↪"black","mit schwer Verletzten": "red",
                                              "mit leicht Verletzten": "blue",␣
 ↪"ohne Verletzte":"green"},
                          hover_name="Ereignisschwere",
                          hover_data=["Jahr", "Ereignisschwere",␣
 ↪"Verletzungsgrad", "Unfallort"],
                          center={"lat":52, "lon":10},
                          zoom=5,
                          height=700,
                          width=800)

figd1.update_layout(mapbox_style="open-street-map")
figd1.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
figd1.show()
```



[ ]:
```
figd1 = px.scatter_mapbox(dataDE_short,
                          lat="latitude",
                          lon="longitude",
                          color="Jahr",
                          hover_name="Ereignisschwere",
                          hover_data=["Jahr", "Ereignisschwere",␣
 ↪"Verletzungsgrad", "Unfallort"],
                          center={"lat":52, "lon":10},
                          zoom=5,
                          height=700,
                          width=800)

figd1.update_layout(mapbox_style="open-street-map")
figd1.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
figd1.show()
```


Jahr
2022
2021
2020
2019
2018
2017
Danmark
Helsingborg
Blekinge län
Skåne län
København
Region Syddanmark
Odense
Flensburg
Słupsk
Schleswig-Holstein
Mecklenburg-Vorpommern
Lübeck
Hamburg
Koszalin
województwo zachodniopomorskie
Bremerhaven
Groningen
Oldenburg
Bremen
Gorzów Wielkopolski
Assen
Niedersachsen
Zwolle
Berlin
Haarlem
Wolfsburg
Osnabrück
Nederland
Hannover
Potsdam
województwo lubuskie
województwo wielkopolskie
Den Haag
Arnhem
Münster
Bielefeld
Sachsen-Anhalt
Zielona Góra
Middelburg
Eindhoven
Cottbus
Chóśebuz
Leipzig
Brugge
Kassel
Vlaanderen
Deutschland
Düsseldorf
Thüringen
Sachsen
Liberec
województwo dolnośląskie
Aachen
Bonn
Siegen
Lille
Chemnitz
België / Belgique / Belgien
Mons
Koblenz
Hessen
Severozápad
Praha
Severovýchod
Hauts-de-France
Frankfurt am Main
Würzburg
Česko
Plzeň
Střední Morava
Lëtzebuerg
Mannheim
Nürnberg
Jihozápad
Jihovýchod
Reims
Saarbrücken
Bayern
Karlsruhe
České Budějovice
Grand Est
Ingolstadt
Troyes
Augsburg
Wien
Freiburg im Breisgau
Oberösterreich
München
Belfort
Basel
Salzburg
Österreich
Zürich
Sankt Gallen
Bourgogne-Franche-Comté
Besançon
Tirol
Szombathely
© OpenStreetMap contributors