Earthquakes distribution analysis.

Roberto Cideos

Abstract

In the following project, I'll be using the **Significant Earthquake Database** provided by the NOAA in order to find out if there is a location-distribution pattern in the earthquakes occurrences. Then, I'll use a Multiple Linear Regression to test and train a model, based on the given dataset.

Motivation

I want to find out if there is an intrinsically correlation between the location of a given earthquake and its occurrence. We know theoretically that earthquakes happen more frequently in some specific places around the world, but to verify this I will plot each earthquake and its given location to find out patterns of its distribution.

Dataset(s)

The dataset is made out of 57 columns measured and investigated by the NOAA. We are going to clean the dataset and use only 6 columns, which are: Longitude, Latitude, Year, Month, Day and EQ_Primary (which is the first wave registered on every earthquake event).

Dataset can be downloaded from here: NOAA_Earthquakes Dataset

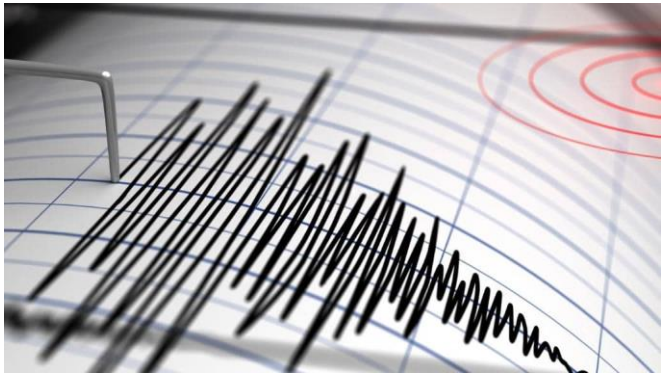
Data Preparation and Cleaning

We need to get rid of the “NAN” values presented on the examined columns, which were given in the previous slide. To remember, we are going to be using only 6 of the 57 columns.

	LONGITUDE	LATITUDE	DAY	MONTH	YEAR	EQ_PRIMARY
1279	121.058	24.223	17	5	2000	5.4
1816	120.600	23.200	22	12	1930	5.5
1275	120.736	23.407	17	7	1998	5.7
1819	120.500	24.200	7	6	1935	5.8
1278	120.506	23.445	22	10	1999	5.9

Research Question(s)

Is there a correlation between the location of a registered earthquake and its occurrence?



Methods

I analysed the data using visual resources in order to find out where do the most amount of earthquakes occur. Also, I used a multivariate linear regression analysis to created a model which can be tested and trained to predict earthquakes.

Linear Assumption:

$$\hat{T} = T_0 + \frac{\delta T}{\delta x}\Delta x + \frac{\delta T}{\delta y}\Delta y + \frac{\delta T}{\delta z}\Delta z \qquad (1)$$

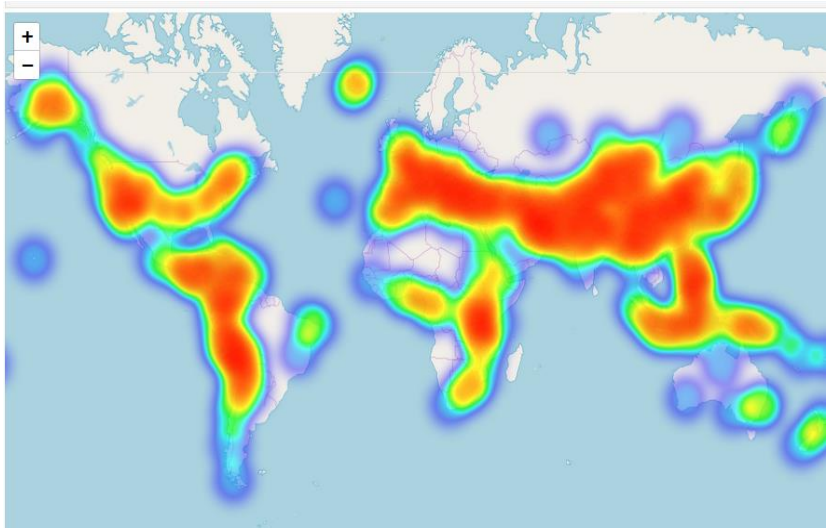
Multivariate Linear Regression Solution:

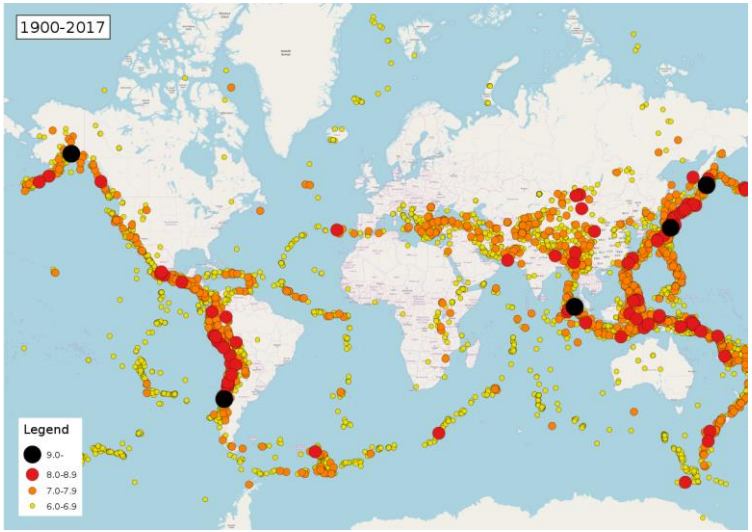
$$\begin{bmatrix} n & \sum x & \sum y & \sum z \\ \sum x & \sum x^2 & \sum xy & \sum xz \\ \sum y & \sum xy & \sum y^2 & \sum yz \\ \sum z & \sum xz & \sum yz & \sum z^2 \end{bmatrix} \begin{bmatrix} T_0 \\ \frac{\delta T}{\delta x} \\ \frac{\delta T}{\delta y} \\ \frac{\delta T}{\delta Z} \end{bmatrix} = \begin{bmatrix} \sum T \\ \sum Tx \\ \sum Ty \\ \sum Tz \end{bmatrix} \qquad (2)$$

Findings

In the following picture, we can appreciate that the three regions were the most amount of earthquakes have occurred since 1900 are (approximately): China, Lowest part of Russia & Central America.

Findings





- HeatMap created to test the distribution of the earthquakes from 1900-2016 with the dataset
- HeatMap provided by Wikipedia to test the results of our dataset.

Limitations

1. The dataset is compounded only by the most signicant earthquakes measured from 1900-2016.
2. The measurement tools have radically changed from 1900-2016. Therefore, the precission of the instruments has been different accross these years which can be a limitation to the liability of the dataset.

Conclusions

1. Earthquakes have a strong dependency on where have they occurred before, no we can verify the theory that assures this fact.

2. We can test and train models to predict where and when does the next earthquakes are going to occur, since those variables hugely depend on location of previous events.

Acknowledgements

I only can thank the Edx staff for teaching with absolute commitment and clearness to us. Also, I must thank to the NOAA for giving me the opportunity to work with its dataset.

References

1. I used this Kaggle repository to learn a little bit more about how to use Heatmap: Link

2. Folium, Matplotlib, Pandas, NumPy

Code

[66]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
data = pd.read_csv('C:\\Users\\rcide\\Desktop\\SanDiegoX\\Final Project\\Airplane_Crashes_and_Fatalities_Since_1908.csv')
data.shape
```

[66]:
```
(5268, 13)
```

[36]:
```
missing_values = ["n/a", "na", "--","NaN","-"]
df = pd.read_csv('C:\\Users\\rcide\\Desktop\\SanDiegoX\\Final Project\\Airplane_Crashes_and_Fatalities_Since_1908.csv', na_values
```

[16]:
```
df2=df.dropna()
```

[19]:
```
df.shape
```

[19]:
```
(5268, 13)
```

[20]:
```
df2.shape
```

[20]:
```
(897, 13)
```

[30]:
```
sum(mask1)
```

[30]:
```
0
```

[508]:
```
missing_values = ["n/a", "na", "--","NaN","-"," "]
df3 = pd.read_csv('C:\\Users\\rcide\\Desktop\\SanDiegoX\\Final Project\\earthquakes.csv', sep=";", na_values = missing_values)
```

Code

[510]:
```
earthquakes = df3.dropna(axis = 0, subset=["LATITUDE", "LONGITUDE", "YEAR", "EQ_PRIMARY", "COUNTRY"])
```

[ ]:
```
map_1
```

[512]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
missing_values = ["n/a", "na", "--","NaN","-"," "]
df4 = pd.read_csv('C:\\Users\\rcide\\Desktop\\SanDiegoX\\Final Project\\earthquake_data.csv', sep=";", na_values = missing_valu
```

[514]:
```
earthquakes = df4.dropna(axis = 0, subset=["LATITUDE", "LONGITUDE", "Date", "EQ_PRIMARY", "REGION"])
```

[515]:
```
import folium
from folium.plugins import MarkerCluster

#Making plot less busy - clustering by regions. Choosing other markers.
map_2 = folium.Map(location=[5, 5], zoom_start=1.5)

#We make a dictionary giving each region its own cluster
cluster_dic = {"region_{}".format(region): folium.plugins.MarkerCluster().add_to(map_2) for region in pd.unique(earthquakes["RE
for country in pd.unique(earthquakes["NAME"]):
    lon_lat_coun = earthquakes.loc[earthquakes["NAME"] == country,["LATITUDE", "LONGITUDE", "Date", "EQ_PRIMARY", "REGION"]].va
    for x, y, date, magnitude, region in lon_lat_coun:
        folium.Marker(
            location=[x,y],
            popup='{} - Earthquake of {} Mg.'.format(date, magnitude),
            icon=folium.Icon(color='red', icon='asterisk'),
            ).add_to(cluster_dic["region_{}".format(region)])
#Uncomment only if you want to save file map
#map_2.save('second_attempt.html')

plot
```

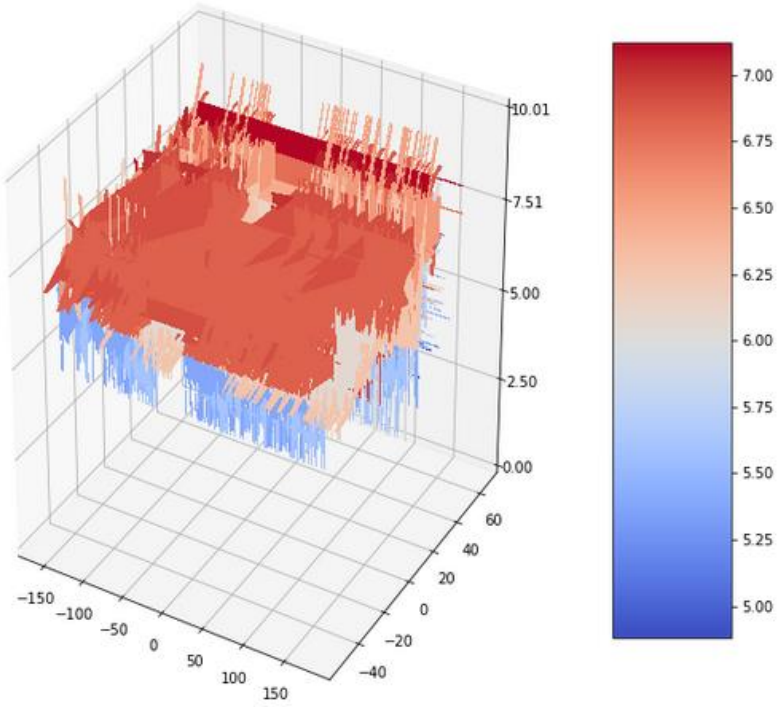
Code


+
−
6
21
12
7
222
75
15
30
355
58
557
4
33
3
235
2
37
95
23
4
136
11
9
33
Leaflet

Code

```
# This import registers the 3D projection, but is otherwise unused.
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import

import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter


fig = plt.figure()
ax = fig.gca(projection='3d')

# Make data.
X = earthquakes.loc[: , ["LONGITUDE"]]
Y = earthquakes.loc[: , ["LATITUDE"]]
X, Y = np.meshgrid(X, Y)
Z = earthquakes.loc[: , ["EQ_PRIMARY"]]

# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(0, 10.01)
ax.zaxis.set_major_locator(LinearLocator(5))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=1, aspect=5)

plt.show()
plt.rcParams["figure.figsize"] = (10,10)
```

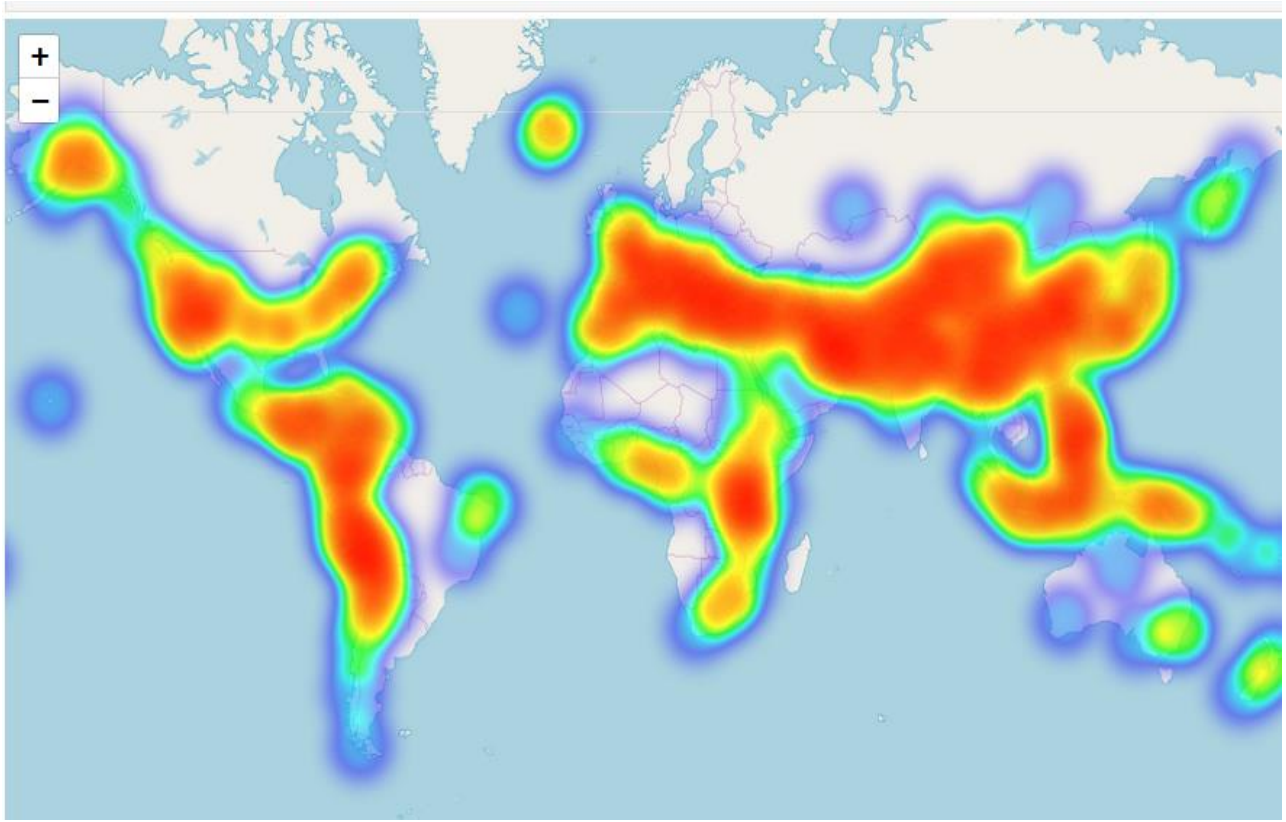
Code


10.01
7.51
5.00
2.50
0.00
−150
−100
−50
0
50
100
150
60
40
20
0
−20
−40
7.00
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00

Code

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import folium
from folium import plugins
from folium.plugins import HeatMap
missing_values = ["n/a", "na", "--","NaN","-"," "]
df4 = pd.read_csv('C:\\Users\\rcide\\Desktop\\SanDiegoX\\Final Project\\earthquake_data.csv', sep=";", na_values = missing_v
map_hooray = folium.Map(location=[13.801602, -89.268206],
                    zoom_start = 8)

# Ensure you're handing it floats
df4['LATITUDE'] = df4['LATITUDE'].astype(float)
df4['LONGITUDE'] = df4['LONGITUDE'].astype(float)

#
# List comprehension to make out list of lists
heat_data = [[row['LATITUDE'],row['LONGITUDE']] for index, row in df4.iterrows()]

# Plot it on the map
HeatMap(heat_data).add_to(map_hooray)

# Display the map
map_hooray
```

Code



Code

```
#### Encoding Categorical Data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

missing_values = ["n/a", "na", "--","NaN","-"," "]
df44 = pd.read_csv('C:\\Users\\rcide\\Desktop\\SanDiegoX\\Final Project\\earthquake_data.csv', sep=";", na_values = missing_values
df4=df44.dropna(subset=['LONGITUDE','LATITUDE','DAY','MONTH','YEAR','EQ_PRIMARY']).sort_values(by=['YEAR'])

LON = df4.loc[: , ["LONGITUDE"]]
LAT = df4.loc[: , ["LATITUDE"]]
D = df4.loc[: , ["DAY"]]
M = df4.loc[: , ["MONTH"]]
Y = df4.loc[: , ["YEAR"]]
XY=pd.DataFrame(df4,columns=['LONGITUDE','LATITUDE','YEAR','DAY','MONTH'])
Z = df4.loc[: , ["EQ_PRIMARY"]]
```

```
import sklearn
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(XY, Z, test_size = 0.2, random_state = 0)
```

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

```
y_pred = regressor.predict(X_test)
```

Code

```
plt.plot(y_pred, 'r--', y_test.reset_index(drop = True), 'b--')
plt.show()
```