• Earthquake data preparation • Earthquake data visualization **Location Analytics - Geospatial Visualization** The ability to gain insights from the location or geographic component of the data. • The important component here is the location (coordinate values). • GIS - Geographic Information System **Credits** - Image from Internet **Real-time Geographic Data Analytics** Getting insights from the data that comes into the system and relating that to a particluar location is called Real-time Geographic data analytics. • Getting route navigation in Google Maps • Courier and postal services Military serives Getting the exact location of the enemy movements on the map to get informed **Credits** - Image from Internet **Historic Geographic Data Analytics** Getting insights from the historic data so as to predict the furture occurrence or disaster (purely by chance) completely based on the geographic location. • Predict and prevent (taking necessary steps) the occurrence based on the past data Disaster prevention efforts Floods Volcanoes Earthquakes **Credits** - Image from Internet **Python Packages for Geospatial Visualization** • GeoViews → explore & visualize geographical, meteorological, and oceanographic datasets • Folium → widely used geospatial data visualization library built on Leaflet.js framework • Plotly → Interactive map visualization - effectively uses Mapbox api • KeplerGL → built exclusively for jupyter notebook to visualize big geospatial data GeoPandas → workhorse for working with geo-data • geonamescache → used to retrieve location datasets in the form of python dictionaries import packages import pandas as pd import plotly.graph objects as go If you do not have the above packages, you can install by typing these commands on Command Prompt (CMD) py -m pip install pandas --user py -m pip install plotly --user **Dataset Description** Earthquake data (from Yesterday) - The data is obtained from USGS datasources. The data is updated every 1 minute. In this example we don't deal with streaming data. time • latitude longitude mag (magnitude) place **Data Source** → https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all\_day.csv data source = 'https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all day.csv' # eqdf - read csv() eqdf = pd.read\_csv(data\_source) # shape eqdf.shape (313, 22)In [4]: # head eqdf.head() Out[4]: time latitude longitude depth mag magType dmin updated place type gap 8km NE of 2021-06-17.0 108.0 0.041460 0.03 ... 16T15:39:54.984Z 2021-06earthquake 37.414166 -121.763168 5.26 1.31 Alum Rock, md 16T15:36:51.810Z CA 14km ENE 2021-06-95.0 0.085920 0.15 ... 16T15:38:06.758Z 2021-06of 35.655167 -117.525833 9.24 1.88 ml 24.0 earthquake 16T15:34:16.780Z Ridgecrest, CA 50 km SSE of Denali 2021-06-2021-06-NaN 0.75 ... 16T15:38:48.573Z 63.120000 -151.337600 ml NaN NaN National earthquake 16T15:33:14.418Z Park, Alaska 9km WNW 2021-06-60.0 0.005624 0.02 ... 16T15:37:11.276Z 2021-06of The earthquake 38.820167 -122.841164 2.03 0.86 md 19.0 16T15:25:32.320Z Geysers, CA 21 km W 2021-06-NaN 0.21 ... 16T15:21:56.275Z 64.842800 -148.457200 15.80 0.60 ml NaN NaN of Ester, earthquake 16T15:17:02.489Z Alaska 5 rows × 22 columns # columns print(list(eqdf.columns)) ['time', 'latitude', 'longitude', 'depth', 'mag', 'magType', 'nst', 'gap', 'dmin', 'rms', 'net', 'id', 'upda ted', 'place', 'type', 'horizontalError', 'depthError', 'magError', 'magNst', 'status', 'locationSource', 'm agSource'] In [6]: # take subset data (time, latitude, longitude, mag, place) eqdf = eqdf[['time', 'latitude', 'longitude', 'mag', 'place']] eqdf.head() Out[7]: latitude longitude mag place **0** 2021-06-16T15:36:51.810Z 37.414166 -121.763168 1.31 8km NE of Alum Rock, CA **1** 2021-06-16T15:34:16.780Z 35.655167 -117.525833 14km ENE of Ridgecrest, CA **2** 2021-06-16T15:33:14.418Z 63.120000 -151.337600 1.80 50 km SSE of Denali National Park, Alaska **3** 2021-06-16T15:25:32.320Z 38.820167 -122.841164 9km WNW of The Geysers, CA **4** 2021-06-16T15:17:02.489Z 64.842800 -148.457200 21 km W of Ester, Alaska # shape eqdf.shape Out[8]: (313, 5) # take "place" separately and display head eqdf['place'].head() 8km NE of Alum Rock, CA Out[9]: 0 14km ENE of Ridgecrest, CA 50 km SSE of Denali National Park, Alaska 9km WNW of The Geysers, CA 21 km W of Ester, Alaska Name: place, dtype: object # example dp = '33 km Hyderabad, India' print(type(dp)) print('----') gp = dp.split(', ') print(gp) print('----') print("Sub Area →", gp[0]) print("Sub Area →", gp[-2]) print('----') print("Area  $\rightarrow$ ", gp[-1]) print("Area →", gp[1]) <class 'str'> ['33 km Hyderabad', 'India'] Sub Area  $\rightarrow$  33 km Hyderabad Sub Area  $\rightarrow$  33 km Hyderabad Area → India Area → India **Data Preperation** Separate place by → **comma** and **space** # place list → split place\_list = eqdf['place'].str.split(', ') Print first 5 from place\_list # head place list.head() Out[12]: 0 [8km NE of Alum Rock, CA] [14km ENE of Ridgecrest, CA] [50 km SSE of Denali National Park, Alaska] [9km WNW of The Geysers, CA] 3 [21 km W of Ester, Alaska] Name: place, dtype: object Make two columns area and sub area from area\_list # apply def subarea\_extractor(x): return x[0] def area\_extractor(x): return x[-1] sub\_area = place\_list.apply(subarea\_extractor) area = place\_list.apply(area\_extractor) Print first 5 from area In [14]: # head area.head() Out[14]: 0 CA Alaska CA Alaska Name: place, dtype: object Print first 5 from sub\_area # head sub area.head() Out[15]: 0 8km NE of Alum Rock 14km ENE of Ridgecrest 50 km SSE of Denali National Park 9km WNW of The Geysers 3 21 km W of Ester Name: place, dtype: object Add area and sub\_area as columns in eqdf In [16]: # add columns eqdf['sub\_area'] = sub\_area eqdf['area'] = area # head eqdf.head() sub\_area latitude longitude mag place time area 8km NE of Alum Rock **0** 2021-06-16T15:36:51.810Z 37.414166 -121.763168 8km NE of Alum Rock, CA CA **1** 2021-06-16T15:34:16.780Z 35.655167 -117.525833 14km ENE of Ridgecrest, CA 14km ENE of Ridgecrest CA **2** 2021-06-16T15:33:14.418Z 63.120000 -151.337600 1.80 50 km SSE of Denali National Park, Alaska 50 km SSE of Denali National Park Alaska **3** 2021-06-16T15:25:32.320Z 38.820167 -122.841164 9km WNW of The Geysers, CA 9km WNW of The Geysers CA **4** 2021-06-16T15:17:02.489Z 64.842800 -148.457200 0.60 21 km W of Ester, Alaska 21 km W of Ester Alaska Remove the column place from eqdf In [18]: prep\_df = eqdf.drop(columns=['place'], axis=1) In [19]: # head prep\_df.head() Out[19]: latitude longitude mag time sub\_area area **0** 2021-06-16T15:36:51.810Z 37.414166 -121.763168 8km NE of Alum Rock CA **1** 2021-06-16T15:34:16.780Z 35.655167 -117.525833 14km ENE of Ridgecrest CA **2** 2021-06-16T15:33:14.418Z 63.120000 -151.337600 1.80 50 km SSE of Denali National Park Alaska 2021-06-16T15:25:32.320Z 38.820167 9km WNW of The Geysers -122.841164 CA 2021-06-16T15:17:02.489Z 64.842800 21 km W of Ester Alaska -148.457200 # shape prep\_df.shape Out[20]: (313, 6) Consider the data where magnitude is greater than or equal to 2 # prep\_df prep\_df = prep\_df[prep\_df['mag'] >= 2] # head prep\_df.head() Out[22]: time latitude longitude sub\_area mag area **6** 2021-06-16T14:56:25.687Z 59.004200 3.20 36 km WSW of Mud Bay -135.929700 Alaska **11** 2021-06-16T13:54:16.680Z 19.205667 -155.458160 2.17 2 km E of Pāhala Hawaii 2021-06-16T13:00:19.490Z 19.576500 137 km N of Charlotte Amalie U.S. Virgin Islands -64.801000 3.87 2021-06-16T12:56:03.070Z 19.810167 -155.040161 2.63 6 km ENE of Pāpa'ikou Hawaii 2021-06-16T12:53:18.408Z 60.002900 43 km ENE of Pedro Bay -153.463800 2.00 Alaska # shape prep\_df.shape Out[23]: **Pie Chart** Check the frequency of occurrence by area In [24]: # occ freq → value counts().to frame() occ\_freq = prep\_df['area'].value\_counts().to\_frame() occ\_freq.head() Out[25]: area Hawaii 23 Alaska 15 **Puerto Rico** CA 4 Idaho The default plot will not be so interactive # occ\_freq.plot(kind='pie', subplots=True) Pie chart with Plotly • labels → index values → area • Irtb → 0 refer → https://plotly.com/python/pie-charts/ # pie chart trace = go.Pie( labels=occ\_freq.index, values=occ\_freq.area layout = go.Layout( height=400, width=600, margin=dict(l=0, r=0, t=0, b=0)fig = go.Figure(data=[trace], layout=layout) fig.show() Hawaii Alaska Puerto Rico 26.7% CA Idaho Nevada 16.3% 1.16% El Salvador <del>-</del>1.16% Oklahoma 1.16% U.S. Virgin Islands 1.16% Ecuador **L**1.16% 1.16% Japan region 3.49% <del>-</del>1.16% Mid-Indian Didao 3.49% 1.16% 2.33% 1.16% 2.33%  $\cdot 1.16\%$ 1.16% -1.16% -1.16% <sup>L</sup>1.16% 1.16% -1.16% <sup>L</sup>2.33% **Bar Chart** Bar chart of magnitude based on area with Plotly x → sub\_area • y → mag • Irtb  $\rightarrow$  0 refer → https://plotly.com/python/bar-charts/ region = 'Alaska' # rdf from prep\_df rdf = prep\_df[prep\_df['area'] == region] In [29]: # head rdf.head() time latitude longitude mag sub\_area area **6** 2021-06-16T14:56:25.687Z 59.0042 -135.9297 3.2 36 km WSW of Mud Bay Alaska **16** 2021-06-16T12:53:18.408Z 60.0029 -153.4638 2.0 43 km ENE of Pedro Bay Alaska 51 km ENE of Ouzinkie Alaska **30** 2021-06-16T12:00:22.009Z 58.0366 -151.6549 3.0 2021-06-16T10:44:16.183Z 56.8456 97 km SE of Ugashik Alaska -156.3472 2.1 2021-06-16T09:07:25.732Z 51.5124 -175.0001 93 km SW of Atka Alaska 2.1 # shape rdf.shape Out[30]: (15, 6) # bar chart trace = go.Bar( x=rdf['sub area'], y=rdf['mag'] layout = go.Layout( height=400, width=600, margin=dict(l=0, r=0, t=0, b=0)fig = go.Figure(data=[trace], layout=layout) fig.show() 4 3.5 3 2.5 2 1.5 1 0.5 97 km SE OF UGASHIK 43 KM ENE OF PEDFO UUZIN 51 KM ENE OF OUZINKIE 36 km WSW OF MUD BAY 8 km 22 52 48 km km km ss km km ss km ss sw of ss of Nikiski Tyonek sinkie

Of Ugashik

North

North Map Visualization app link - http://earthquake-tracking-system.herokuapp.com/ lat → latitudes lon → longitudes mode → markers ■ size → 10 color → red text → sub\_area refer → https://plotly.com/python/scattermapbox/ region = 'Alaska' # filter it rdf = prep\_df[prep\_df['area'] == region] # lats, lons, text (sub area) region\_lats = rdf['latitude'].to\_list() region\_lons = rdf['longitude'].to\_list() region\_texts = rdf['sub\_area'].to\_list() In [34]: trace = go.Scattermapbox( lat=region\_lats, lon=region lons, mode='markers', marker=dict( size=10, color='red' text=region\_texts, hoverinfo='text' layout = go.Layout( height=400, width=600, margin=dict(l=0, r=0, t=0, b=0),mapbox\_style='stamen-terrain', mapbox=dict( center=dict( lat=region\_lats[0], lon=region\_lons[0] zoom=2 ) ) fig = go.Figure(data=[trace], layout=layout) fig.show()

What did we learn?

• Earthquake data - using pandas

Data preparationData filtering

• Earthquake data visualization

• Where it is used and how it is important for the business or the govt agency

• Python modules that support geospatial visualization

Location Analytics

Today's agenda

Location analytics

Real-time geographic DAHistorical geographic DA

• Python pacakges for geospatial visualization