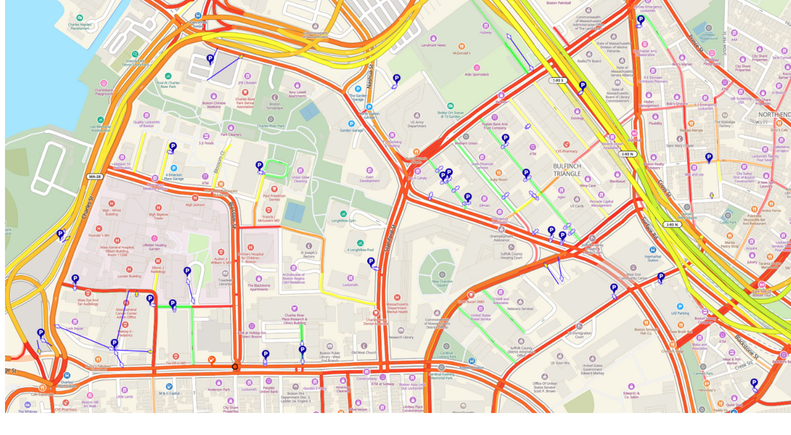
Today's agenda

- Location analytics
  - Real-time geospatial GA
  - Historical geospatial GA
- Python packages for geospatial visualization
- 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

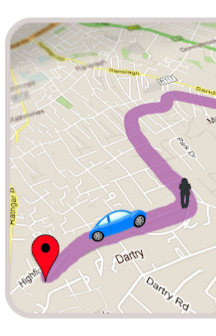


Credits - Image from internet

Real-time Geographic Data Analytics

Getting insights from the data that comes into the system and tracking that to a particular location is called Real-time Geographic Data Analytics.

- Getting route navigation in Google Maps
- Courier and postal services
- Military entries
  - 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 future 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-prone/affected areas
  - Floods
  - Volcanoes
  - Earthquakes

Credits - Image from internet

Python Packages for Geospatial Visualization

- GeoViews
- Folium
- Plotly
- Geoplotlib
- GeoPandas
- geonamescache

import packages

```
import pandas as pd
import plotly.graph_objects as go
```

If you do not have the above packages, you can install it by typing these commands on `Command Prompt` (CMD).

- `pip install pandas --user`
- `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 are to 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"
# read - csv
eqdf = pd.read_csv(data_source)
```

```
# shape
eqdf.shape
```

(313, 22)

```
# head
eqdf.head()
```

[illegible]

5 rows × 22 columns

```
# columns
eqdf = eqdf[['time', 'latitude', 'longitude', 'mag', 'place']]
```

```
# head
eqdf.head()
```

	time	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	1.88	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	0.86	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

```
# shape
eqdf.shape
```

(313, 5)

In [9]:
```
# take "place" separately and display head
eqdf['place'].head()
```

Out[9]:
```
0                      8km NE of Alum Rock, CA
1                   14km ENE of Ridgecrest, CA
2    50 km SSE of Denali National Park, Alaska
3                   9km WNW of The Geysers, CA
4                     21 km W of Ester, Alaska
Name: place, dtype: object
```

In [10]:
```
# 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 →", gp[-1])
# or
print("Area →", gp[1])
```

```
<class 'str'>
------------
['33 km Hyderabad', 'India']
------------
Sub Area → 33 km Hyderabad
Sub Area → 33 km Hyderabad
------------
Area → India
Area → India
```

Data Preperation

Separate `place` by → **comma** and **space**

In [11]:
```
# place_list → split
place_list = eqdf['place'].str.split(', ')
```

Print first `5` from `place_list`

In [12]:
```
# head
place_list.head()
```

Out[12]:
```
0                      [8km NE of Alum Rock, CA]
1                   [14km ENE of Ridgecrest, CA]
2    [50 km SSE of Denali National Park, Alaska]
3                   [9km WNW of The Geysers, CA]
4                     [21 km W of Ester, Alaska]
Name: place, dtype: object
```

Make two columns `area` and `sub_area` from area_list

In [13]:
```
# 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
1        CA
2    Alaska
3        CA
4    Alaska
Name: place, dtype: object
```

Print first `5` from `sub_area`

In [15]:
```
# head
sub_area.head()
```

Out[15]:
```
0                 8km NE of Alum Rock
1              14km ENE of Ridgecrest
2    50 km SSE of Denali National Park
3              9km WNW of The Geysers
4                    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
```

In [17]:
```
# head
eqdf.head()
```

Out[17]:

	time	latitude	longitude	mag	place	sub_area	area
0	2021-06-16T15:36:51.810Z	37.414166	-121.763168	1.31	8km NE of Alum Rock, CA	8km NE of Alum Rock	CA
1	2021-06-16T15:34:16.780Z	35.655167	-117.525833	1.88	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	0.86	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
prep_df = eqdf.drop(columns=['place'], axis=1)
```

In [19]:
```
# head
prep_df.head()
```

Out[19]:

	time	latitude	longitude	mag	sub_area	area
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	1.88	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	0.86	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

In [20]:
```
# shape
prep_df.shape
```

Out[20]: (313, 6)

Consider the data where magnitude is greater than or equal to `2`

In [21]:
```
# prep_df
prep_df = prep_df[prep_df['mag'] >= 2]
```

In [22]:
```
# head
prep_df.head()
```

Out[22]:

	time	latitude	longitude	mag	sub_area	area
6	2021-06-16T14:56:25.687Z	59.004200	-135.929700	3.20	36 km WSW of Mud Bay	Alaska
11	2021-06-16T13:54:16.680Z	19.205667	-155.458160	2.17	2 km E of Pāhala	Hawaii
14	2021-06-16T13:00:19.490Z	19.576500	-64.801000	3.87	137 km N of Charlotte Amalie	U.S. Virgin Islands
15	2021-06-16T12:56:03.070Z	19.810167	-155.040161	2.63	6 km ENE of Pāpā'ikou	Hawaii
16	2021-06-16T12:53:18.408Z	60.002900	-153.463800	2.00	43 km ENE of Pedro Bay	Alaska

In [23]:
```
# shape
prep_df.shape
```

Out[23]: (86, 6)

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()
```

In [25]:
```
occ_freq.head()
```

Out[25]:

	area
Hawaii	23
Alaska	15
Puerto Rico	14
CA	4
Idaho	3

The default plot will not be so interactive

In [26]:
```
# occ_freq.plot(kind='pie', subplots=True)
```

`Pie` chart with `Plotly`

- labels → index
- values → area
- lrtb → 0

refer → https://plotly.com/python/pie-charts/

In [27]:
```
# 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()
```

Bar Chart

`Bar` chart of magnitude based on area with `Plotly`

- x → sub_area
- y → mag
- lrtb → 0

refer → https://plotly.com/python/bar-charts/

In [28]:
```
region = 'Alaska'

# rdf from prep_df
rdf = prep_df[prep_df['area'] == region]
```

In [29]:
```
# head
rdf.head()
```

Out[29]:

	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
30	2021-06-16T12:00:22.009Z	58.0366	-151.6549	3.0	51 km ENE of Ouzinkie	Alaska
46	2021-06-16T10:44:16.183Z	56.8456	-156.3472	2.1	97 km SE of Ugashik	Alaska
66	2021-06-16T09:07:25.732Z	51.5124	-175.0001	2.1	93 km SW of Atka	Alaska

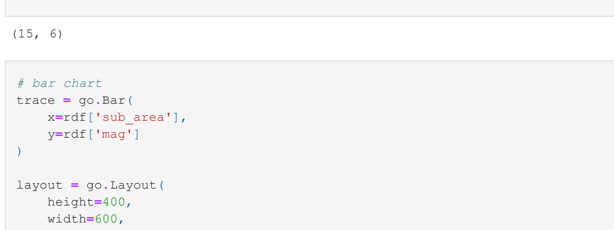
In [30]:
```
# shape
rdf.shape
```

Out[30]: (15, 6)

In [31]:
```
# 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()
```



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/

In [32]:
```
region = 'Alaska'

# filter it
rdf = prep_df[prep_df['area'] == region]
```

In [33]:
```
# 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?

- Location Analytics
- Where it is used and how it is important for the business or the govt agency
- Python modules that support geospatial visualization
- Earthquake data - using pandas
  - Data preparation
  - Data filtering
- Earthquake data visualization