Plotly: Data Visualization

Presented by: Christina DePerro

What is Plotly?

- Data visualization library in Python founded in Canada 2012 and launched as open-source library in 2015
- Acts as a JavaScript-based library creating a JavaScript code but binds with other languages, where it can convert the code to other languages, like Python (commonly used)
- Optimal for:
  - Creating **interactive** charts, allowing users to engage with the visualizations
  - Supporting a wide range of charts and graphs (40+) such as line graphs, bar charts, stacked charts, box plots, scatter plots, heat maps, 3D models, etc.
  - Integrating with web applications and other tools including Pandas, NumPy, SciPy, and Matplotlib

Why would a data scientist use Plotly?

- Data Exploration & Dashboard Development
  - Visualize the relationships between the dataset variables, identify any outliers
  - View and analyze time series, geospatial and multidimensional models
  - Share visualizations with others using URLs or integrate them with web-based applications such as:
    - HTML, JavaScript, CSS, **Jupyter Notebook**
  - Use high-level interfaces for quick and easy visualizations or for more extensive plots:

➡ 1. **Plotly Express: High level API – used to create simpler visuals, minimal code required**
2. Plotly Graph Objects: Low-level API – used for customization purposes
3. Dash: Python web framework, minimal code required

➡ 4. **Jupyter Notebook: Interactive environment to conduct Plotly data analysis**
5. JavaScript API: Web application for embedding Plotly visuals
6. Plotly for R: Create interactive visuals in R

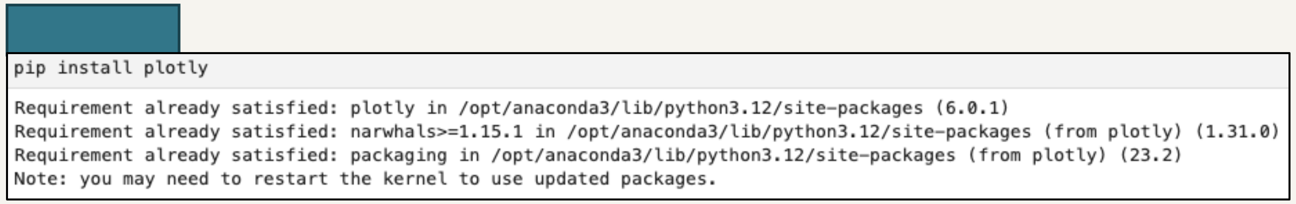
➡ 7. **Plotly for Pandas: Use Pandas DataFrames when creating plots**

Getting started in Jupyter Notebook

1

Launch Jupyter Notebook and install Plotly

Ensure Jupyter is upgraded to latest version, then install plotly. (Access files from GitHub:
and refer to "*Plotly Installation Instructions*" file for details)



```
pip install plotly
```

```
Requirement already satisfied: plotly in /opt/anaconda3/lib/python3.12/site-packages (6.0.1)
Requirement already satisfied: narwhals>=1.15.1 in /opt/anaconda3/lib/python3.12/site-packages (from plotly) (1.31.0)
Requirement already satisfied: packaging in /opt/anaconda3/lib/python3.12/site-packages (from plotly) (23.2)
Note: you may need to restart the kernel to use updated packages.
```

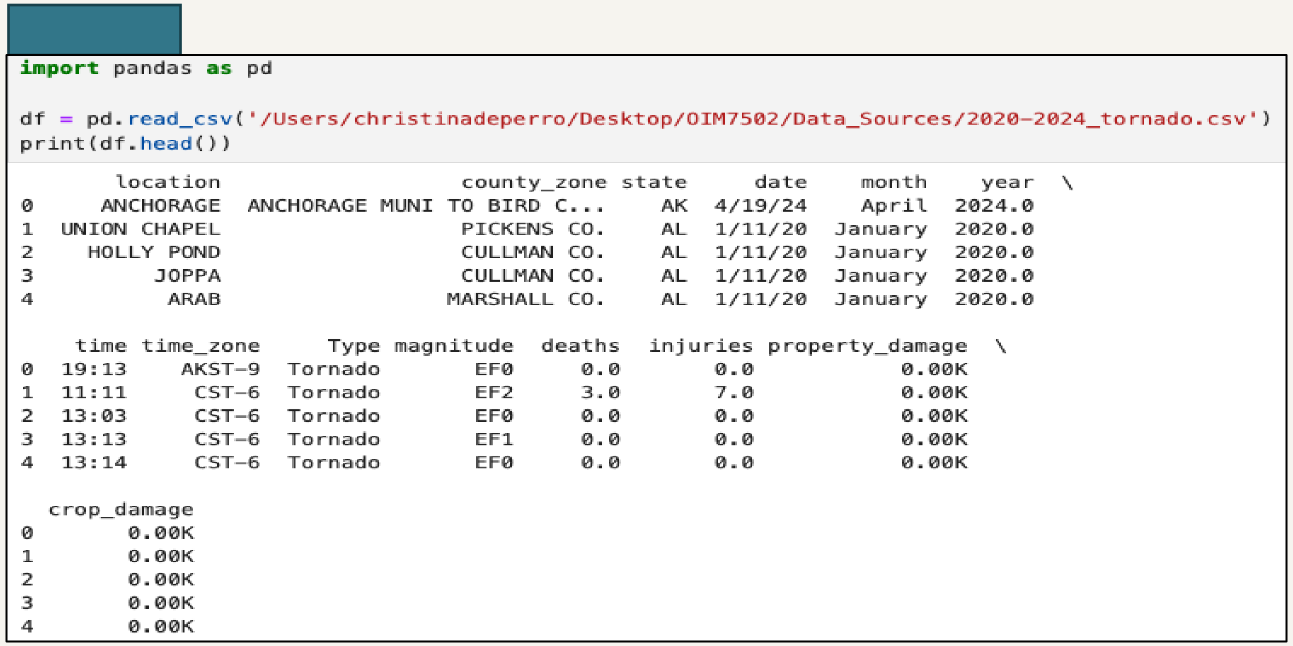
2

Import data using Pandas

Today's Goal: Analyze national tornado trends from US Department of Commerce's NOAA dataset.

Dataset includes:

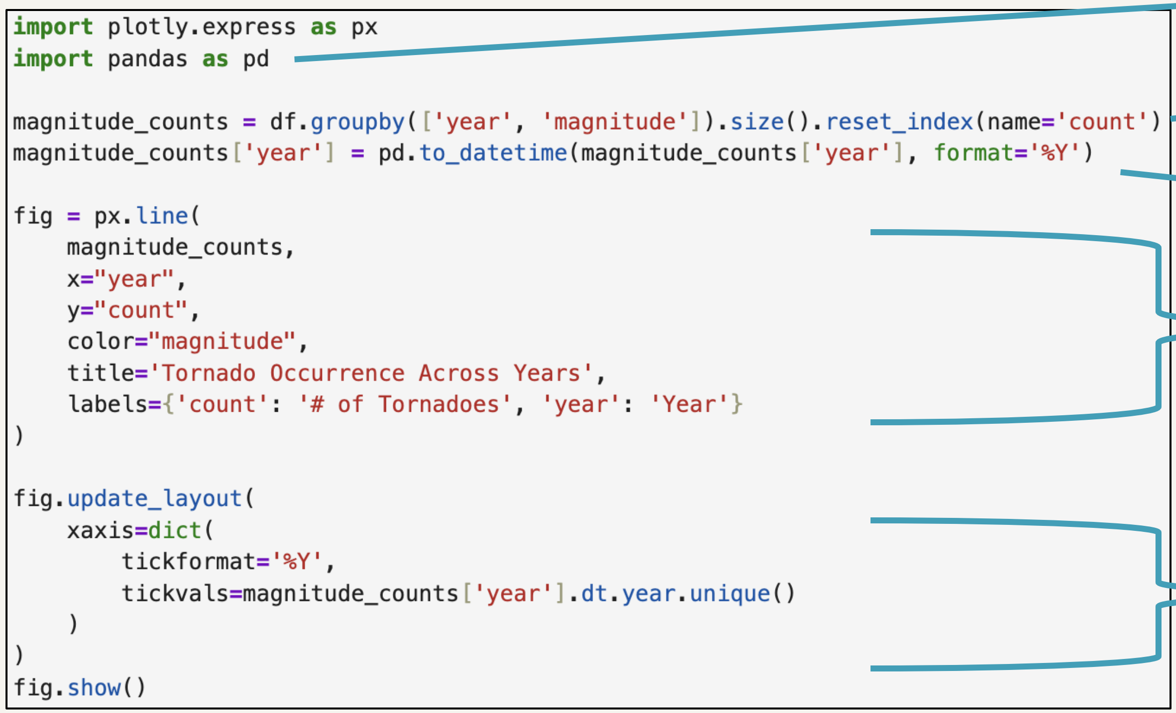
- 2020-2024 reporting of tornadoes in the US
- Tornado magnitude levels follow EF Scale, rating their intensities based on wind, speed and damage
  - **EFU:** Weakest, unknown speed
  - **EF0:** Weak: 65-85mph
  - **EF1:** Moderate: 86-110mph
  - **EF2:** Strong; 111-135mph
  - **EF3:** Severe; 136-165mph
  - **EF4:** Devastating; 166-200mph
  - **EF5:** Catastrophic; >200mph



```
import pandas as pd

df = pd.read_csv('/Users/christinadeperro/Desktop/OIM7502/Data_Sources/2020-2024_tornado.csv')
print(df.head())
```

```
        location                    county_zone state       date    month    year  \
0      ANCHORAGE  ANCHORAGE MUNI TO BIRD C...    AK  4/19/24    April  2024.0
1   UNION CHAPEL                  PICKENS CO.    AL  1/11/20  January  2020.0
2     HOLLY POND                  CULLMAN CO.    AL  1/11/20  January  2020.0
3          JOPPA                  CULLMAN CO.    AL  1/11/20  January  2020.0
4           ARAB                 MARSHALL CO.    AL  1/11/20  January  2020.0

    time time_zone     Type magnitude  deaths  injuries property_damage  \
0  19:13    AKST-9  Tornado       EF0     0.0       0.0           0.00K
1  11:11     CST-6  Tornado       EF2     3.0       7.0           0.00K
2  13:03     CST-6  Tornado       EF0     0.0       0.0           0.00K
3  13:13     CST-6  Tornado       EF1     0.0       0.0           0.00K
4  13:14     CST-6  Tornado       EF0     0.0       0.0           0.00K

  crop_damage
0       0.00K
1       0.00K
2       0.00K
3       0.00K
4       0.00K
```

Creating a Basic Line Graph



```python
import plotly.express as px
import pandas as pd

magnitude_counts = df.groupby(['year', 'magnitude']).size().reset_index(name='count')
magnitude_counts['year'] = pd.to_datetime(magnitude_counts['year'], format='%Y')

fig = px.line(
    magnitude_counts,
    x="year",
    y="count",
    color="magnitude",
    title='Tornado Occurrence Across Years',
    labels={'count': '# of Tornadoes', 'year': 'Year'}
)

fig.update_layout(
    xaxis=dict(
        tickformat='%Y',
        tickvals=magnitude_counts['year'].dt.year.unique()
    )
)
fig.show()
```

Provides flexibility to manipulate our DataFrame: **“2020-2024 Tornado”** csv

Group data by ‘year’ and ‘magnitude’ (df.groupby) and count rows in each group (.size). Add new column called ‘count’ (.reset_index). Outputs are stored in magnitude_counts.

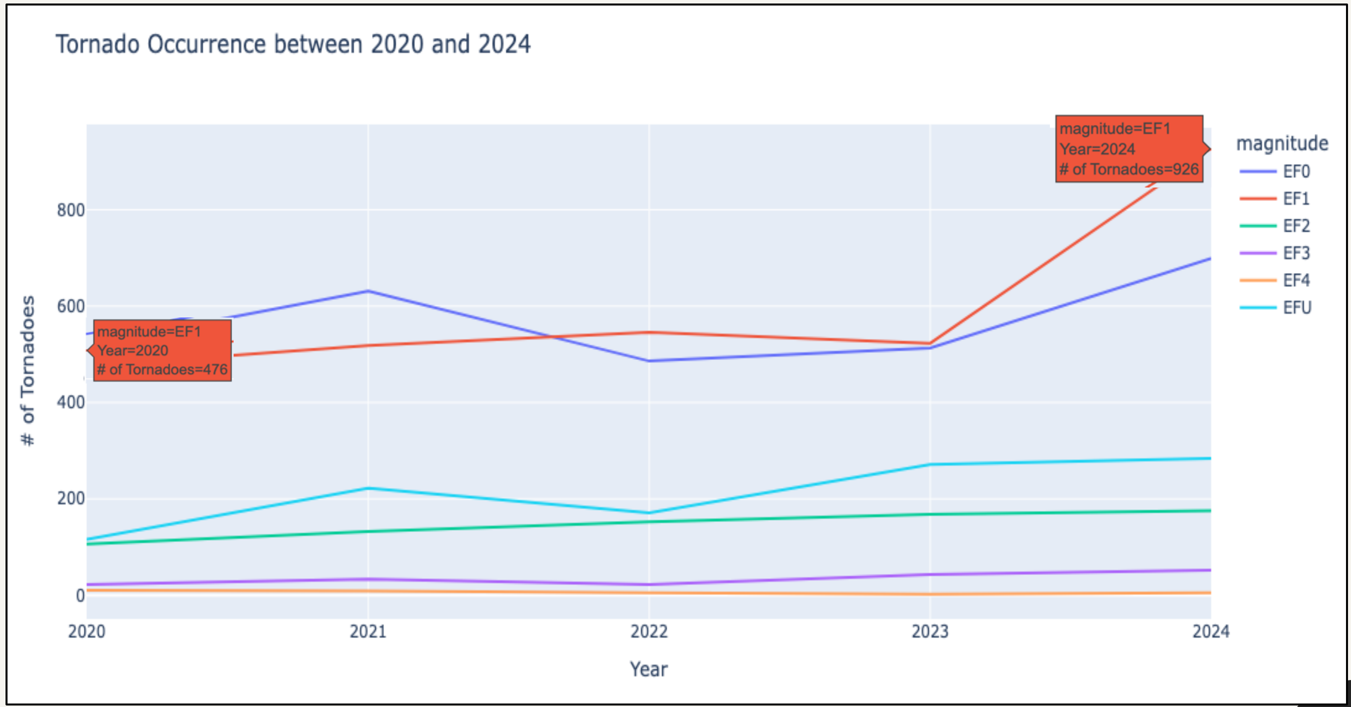
Convert values in 'year' column to datetime object using format=‘%Y’

Create line graph using plotly express (px.line). Pull in data from new DataFrame magnitude_counts. Identify x-value and y-values. The line graph will color code the tornados by their respective magnitude levels. Provide a title and customize the axis labels.

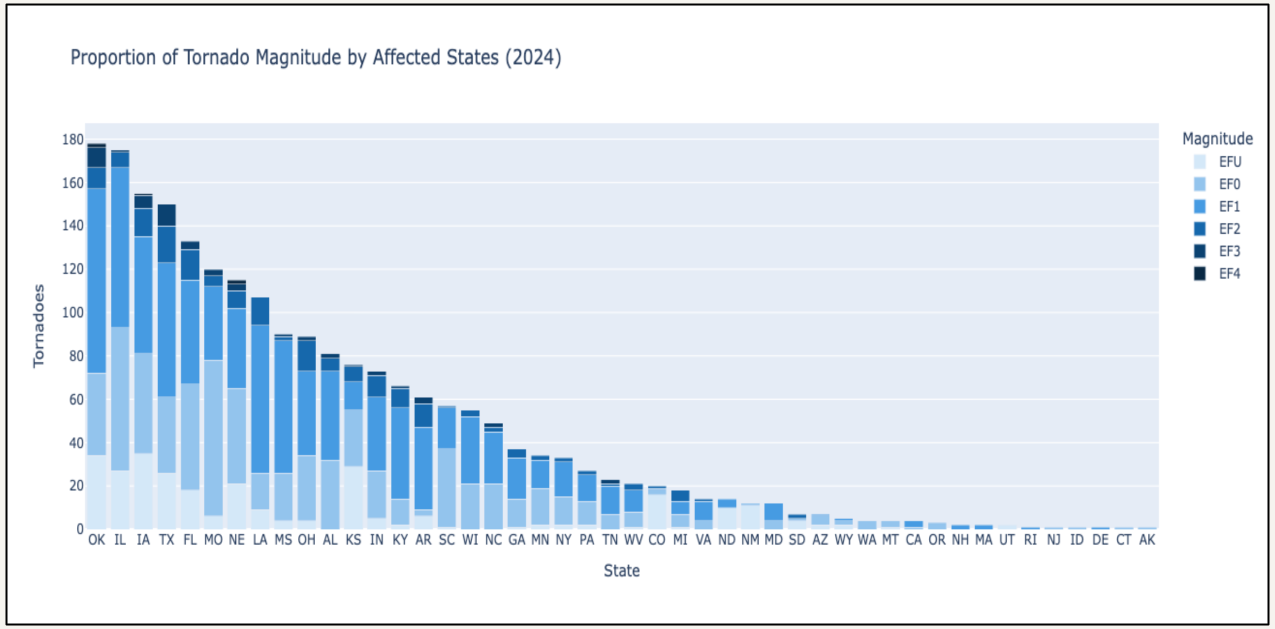
Update the layout for the x-axis using (dict) to modify the ticks. ‘%Y’ must display the year in a 4-digital format. Unique values from magnitude_counts will be extracted to appear as ticks on x-axis.

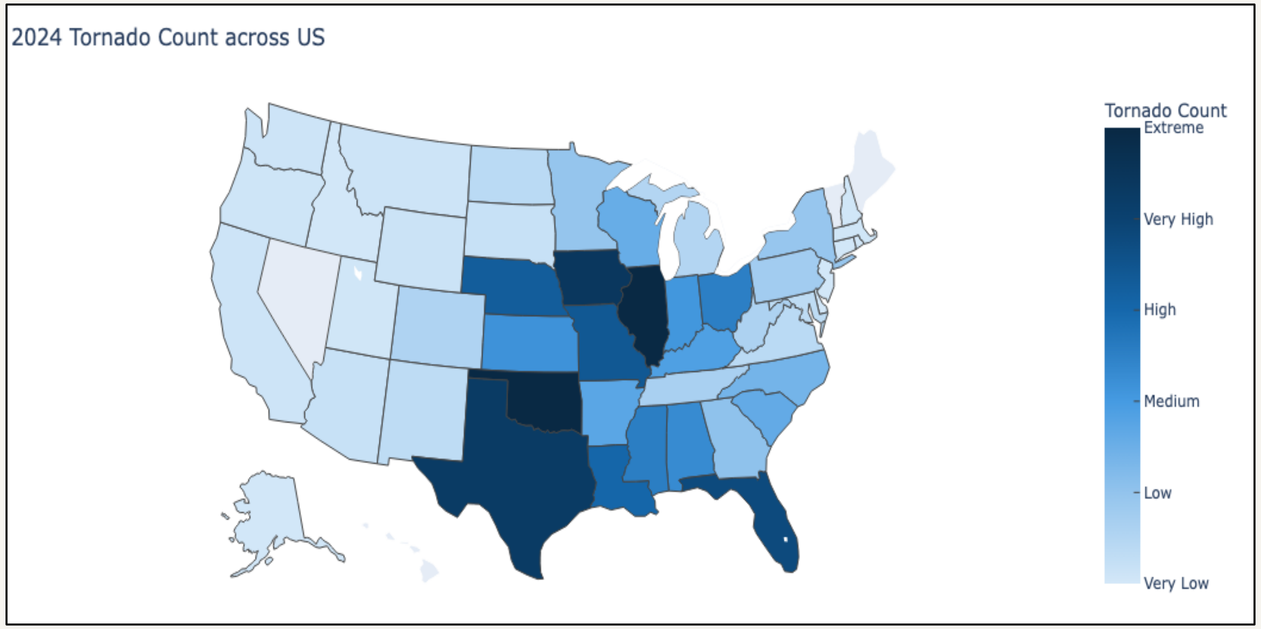
Visualizing the Line Graph

```python
import plotly.express as px
import pandas as pd

magnitude_counts = df.groupby(['year', 'magnitude']).size().reset_index(name='count')
magnitude_counts['year'] = pd.to_datetime(magnitude_counts['year'], format='%Y')

fig = px.line(
    magnitude_counts,
    x="year",
    y="count",
    color="magnitude",
    title='Tornado Occurrence Across Years',
    labels={'count': '# of Tornadoes', 'year': 'Year'}
)

fig.update_layout(
    xaxis=dict(
        tickformat='%Y',
        tickvals=magnitude_counts['year'].dt.year.unique()
    )
)
fig.show()
```



Analyzing Aggregated Data Results





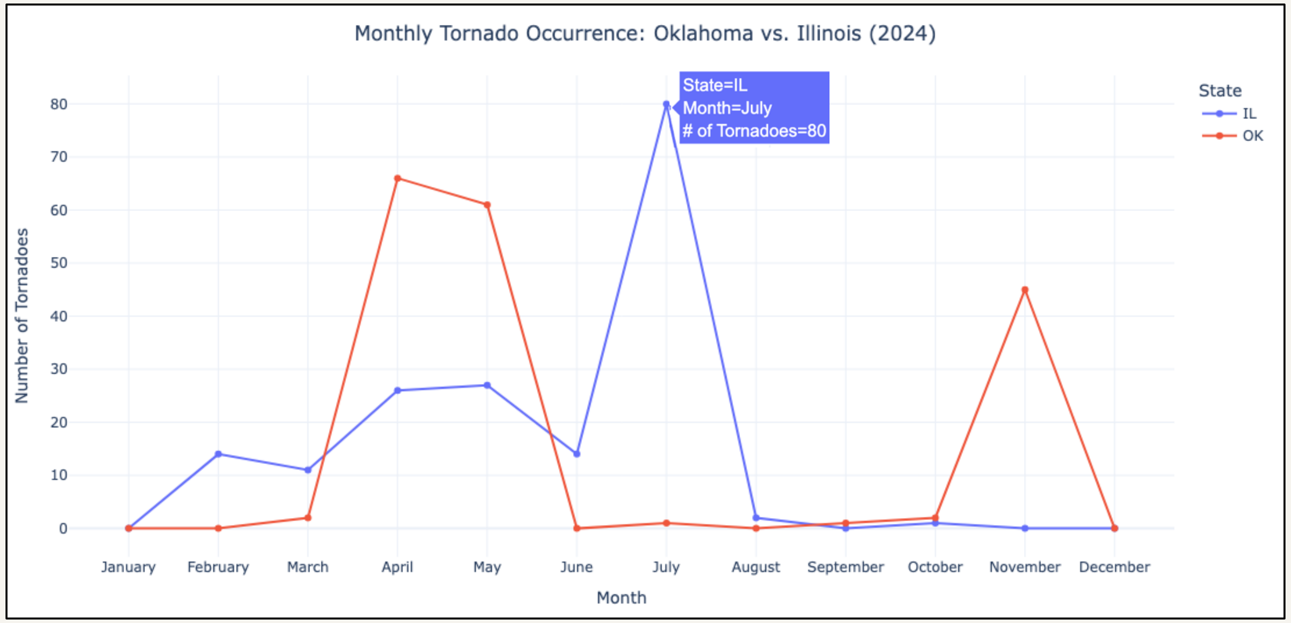
Stacked bar chart displaying the distribution of tornado magnitude levels for each affected state in 2024.

For dashboard purposes, a data scientist might add a filter by region to aggregate the data and provide a cleaner visualization.

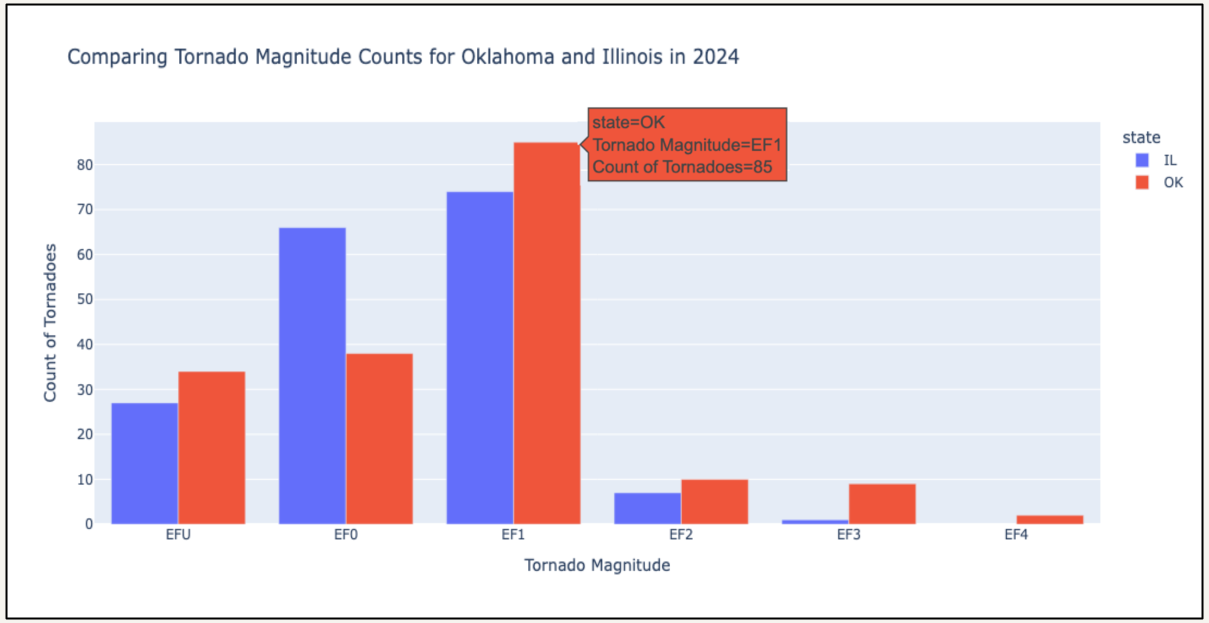
Choropleth map showing the frequency levels of tornado occurrence across each state in 2024.

Increase granularity and use coordinate data to create a heat map on the county or zip code level.

Analyzing Granular Data Results



Time series line graph comparing the top two states with the highest tornado occurrence in 2024.



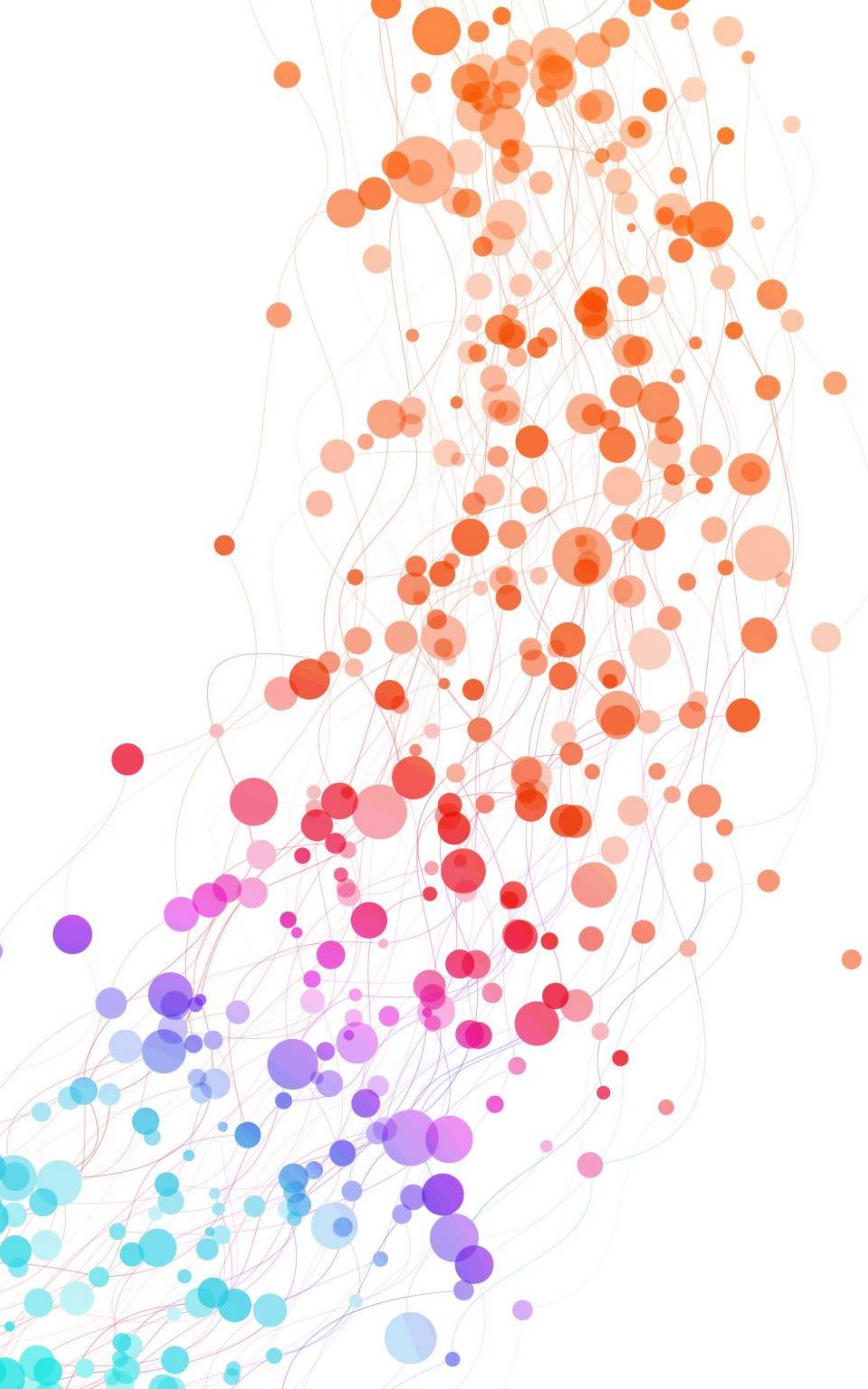
Grouped bar chart comparing tornado magnitude levels against the top two tornado-prone states in 2024.

Import weather data to compare and analyze storm conditions attributing to tornado frequency.

Class Tutorial

GitHub File Name Reference: "Tornado_Pie_Shell"

1. Create a **pie chart**
2. Filter data to **2024**
3. Group the data to view the **proportion of tornado magnitudes** across **all states**



Thank You