Introduction to Big Data and Data Science (CSCE 5300 Section 005)*

Yunhe Feng

Assistant Professor, Department of Computer Science and Engineering

5th September, 2024

*The teaching materials are reorganized and reformed based on Prof. Ravi Vadapalli's slides (Ravi.Vadapalli@unt.edu, UNT & University of Miami)

Quiz 1

- Closed-book in-person Quiz
- 5 Questions: 1 point for each question
- Quiz time: 2:40 am - 3:05 am, Sept. 5, 2024

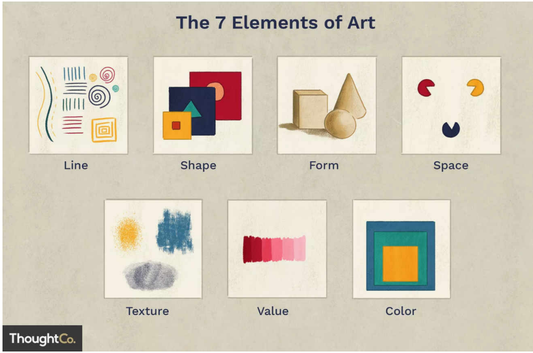
1. Concept of Data Visualization
2. Data Visualization in Python
3. Assignment

What is Data Visualization

- Data visualization is the graphical representation of information and data. - Tableau
- Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from. - TechTarget
- Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. - IBM

Seven Visual Elements[1]

- Line
- Shape
- Color
- Value: vary the value of the color to create depth and contrast
- Form: defined as a three-dimensional object
- Texture
- Space: whitespace or negative space



---

[1] https://uxplanet.org/7-visual-elements-of-design-bbd56eb063e9

Types of Data Visualization: Charts, Graphs, and Maps

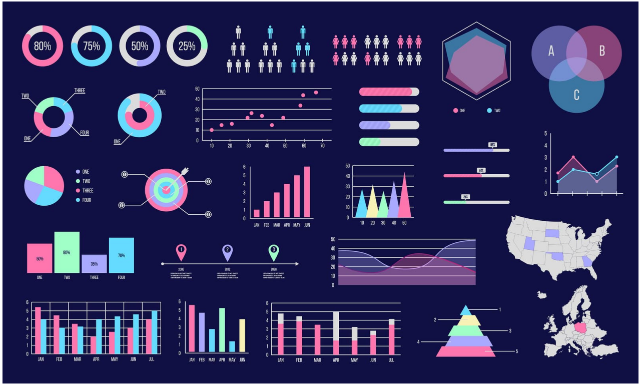


Figure 1: The Top 10 Types of Data Visualization Made Simple[2]

---

[2] https://boostlabs.com/blog/10-types-of-data-visualization-tools/

Why Data Visualization

- Human brain processes images 60,000 times faster than text[3].
- 90% of the information transmitted to the brain is visual[4].
- Our culture is visual.
- It helps people see, interact with, and better understand data.

---

[3] `https://ifvp.org/`

[4] `https://ifvp.org/`

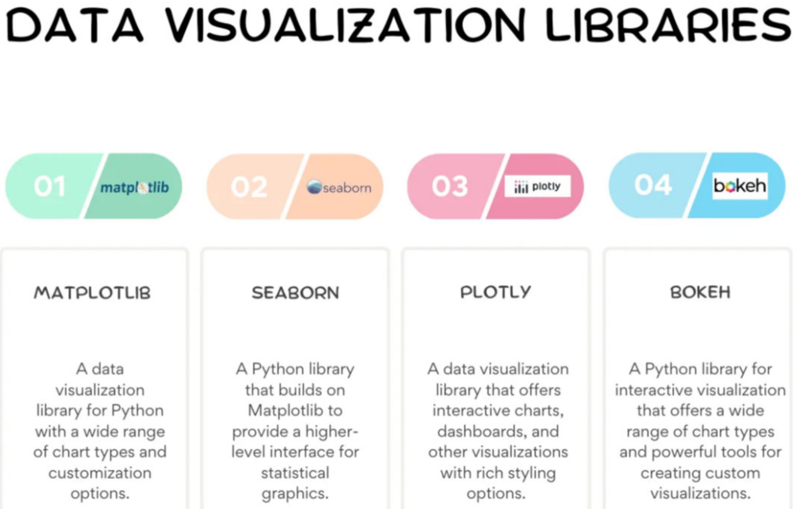
1 Concept of Data Visualization

2 Data Visualization in Python
    2D Visualization in Python
    3D Visualization in Python
    4D Visualization in Python

3 Assignment

1. Concept of Data Visualization
2. Data Visualization in Python
    - 2D Visualization in Python
    - 3D Visualization in Python
    - 4D Visualization in Python
3. Assignment

Data Visualization Libraries



DATA VISUALIZATION LIBRARIES

01 matplotlib

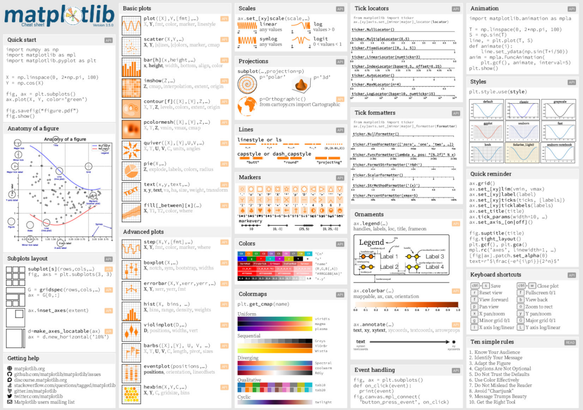
02 seaborn

03 plotly

04 bokeh

MATPLOTLIB	SEABORN	PLOTLY	BOKEH
A data visualization library for Python with a wide range of chart types and customization options.	A Python library that builds on Matplotlib to provide a higher-level interface for statistical graphics.	A data visualization library that offers interactive charts, dashboards, and other visualizations with rich styling options.	A Python library for interactive visualization that offers a wide range of chart types and powerful tools for creating custom visualizations.

Matplotlib - a Comprehensive Library for Data Visualizations in Python



matplotlib
Cheat sheet
Version 3.5.0

Quick start

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt


X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X, Y, color='green')

fig.savefig("figure.pdf")
fig.show()
```

Anatomy of a figure

Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Grid
Legend
Line (line plot)
Y axis label
Markers (scatter plot)
Figure
Axes
Line (line plot)
Spines
Minor tick label
X axis label

Subplots layout

```
subplot[s](rows,cols,…)
fig, axs = plt.subplots(3, 3)

G = gridspec(rows,cols,…)
ax = G[0,:]

ax.inset_axes(extent)

d=make_axes_locatable(ax)
ax = d.new_horizontal('10%')
```

Getting help

matplotlib.org
github.com/matplotlib/matplotlib/issues
discourse.matplotlib.org
stackoverflow.com/questions/tagged/matplotlib
gitter.im/matplotlib
twitter.com/matplotlib
Matplotlib users mailing list

Basic plots

`plot([X],Y,[fmt],…)`
X, Y, fmt, color, marker, linestyle

`scatter(X,Y,…)`
X, Y, [s]izes, [c]olors, marker, cmap

`bar[h](x,height,…)`
x, height, width, bottom, align, color

`imshow(Z,…)`
Z, cmap, interpolation, extent, origin

`contour[f]([X],[Y],Z,…)`
X, Y, Z, levels, colors, extent, origin

`pcolormesh([X],[Y],Z,…)`
X, Y, Z, vmin, vmax, cmap

`quiver([X],[Y],U,V,…)`
X, Y, U, V, C, units, angles

`pie(X,…)`
Z, explode, labels, colors, radius

`text(x,y,text,…)`
x, y, text, va, ha, size, weight, transform

`fill[_between][x](…)`
X, Y1, Y2, color, where

Advanced plots

`step(X,Y,[fmt],…)`
X, Y, fmt, color, marker, where

`boxplot(X,…)`
X, notch, sym, bootstrap, widths

`errorbar(X,Y,xerr,yerr,…)`
X, Y, xerr, yerr, fmt

`hist(X, bins, …)`
X, bins, range, density, weights

`violinplot(D,…)`
D, positions, widths, vert

`barbs([X],[Y], U, V, …)`
X, Y, U, V, C, length, pivot, sizes

`eventplot(positions,…)`
positions, orientation, lineoffsets

`hexbin(X,Y,C,…)`
X, Y, C, gridsize, bins

Scales

`ax.set_[xy]scale(scale,…)`
linear — any values
log — values > 0
symlog — any values
logit — 0 < values < 1

Projections

`subplot(…,projection=p)`
p='polar'
p='3d'
p=Orthographic()
from cartopy.crs import Cartographic

Lines

linestyle or ls
"-" ":" "--" "-." (0,(0.01,2))
capstyle or dash_capstyle
"butt" "round" "projecting"

Markers

'.' 'o' 's' 'P' 'X' '*' 'p' 'D' '<' '>' '^' 'v'
'1' '2' '3' '4' '+' 'x' '|' '_' 4 5 6 7
markevery
10 [0, -1] (25, 5) [0, 25, -1]

Colors

'Cn'
'x'
'name'
(R,G,B[,A])
'#RRGGBB[AA]'
'x.y'

Colormaps

`plt.get_cmap(name)`

Uniform: viridis, magma, plasma
Sequential: Greys, YlOrBr, Wistia
Diverging: Spectral, coolwarm, RdGy
Qualitative: tab10, tab20
Cyclic: twilight

Tick locators

```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_locator(locator)
ticker.NullLocator()
ticker.MultipleLocator(0.5)
ticker.FixedLocator([0, 1, 5])
ticker.LinearLocator(numticks=3)
ticker.IndexLocator(base=0.5, offset=0.25)
ticker.AutoLocator()
ticker.MaxNLocator(n=4)
ticker.LogLocator(base=10, numticks=15)
```

Tick formatters

```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_formatter(formatter)
ticker.NullFormatter()
ticker.FixedFormatter(['zero', 'one', 'two', …])
ticker.FuncFormatter(lambda x, pos: "[%.2f]" % x)
ticker.FormatStrFormatter('>%d<')
ticker.ScalarFormatter()
ticker.StrMethodFormatter('{x}')
ticker.PercentFormatter(xmax=5)
```

Ornaments

`ax.legend(…)`
handles, labels, loc, title, frameon

`ax.colorbar(…)`
mappable, ax, cax, orientation

`ax.annotate(…)`
text, xy, xytext, xycoords, textcoords, arrowprops

Event handling

```
fig, ax = plt.subplots()
def on_click(event):
  print(event)
fig.canvas.mpl_connect(
  'button_press_event', on_click)
```

Animation

```
import matplotlib.animation as mpla

T = np.linspace(0, 2*np.pi, 100)
S = np.sin(T)
line, = plt.plot(T, S)
def animate(i):
    line.set_ydata(np.sin(T+i/50))
anim = mpla.FuncAnimation(
    plt.gcf(), animate, interval=5)
plt.show()
```

Styles

`plt.style.use(style)`

Quick reminder

```
ax.grid()
ax.set_[xy]lim(vmin, vmax)
ax.set_[xy]label(label)
ax.set_[xy]ticks(ticks, [labels])
ax.set_[xy]ticklabels(labels)
ax.set_title(title)
ax.tick_params(width=10, …)
ax.set_axis_[on|off]()

fig.suptitle(title)
fig.tight_layout()
plt.gcf(), plt.gca()
mpl.rc('axes', linewidth=1, …)
[fig|ax].patch.set_alpha(0)
text=r'$\frac{-e^{i\pi}}{2^n}$'
```

Keyboard shortcuts

ctrl + s Save
ctrl + w Close plot
r Reset view
f Fullscreen 0/1
f View forward
b View back
p Pan view
o Zoom to rect
x X pan/zoom
y Y pan/zoom
g Minor grid 0/1
G Major grid 0/1
l X axis log/linear
L Y axis log/linear

Ten simple rules

1. Know Your Audience
2. Identify Your Message
3. Adapt the Figure
4. Captions Are Not Optional
5. Do Not Trust the Defaults
6. Use Color Effectively
7. Do Not Mislead the Reader
8. Avoid "Chartjunk"
9. Message Trumps Beauty
10. Get the Right Tool

Figure 2: Matplotlib cheatsheets[5]

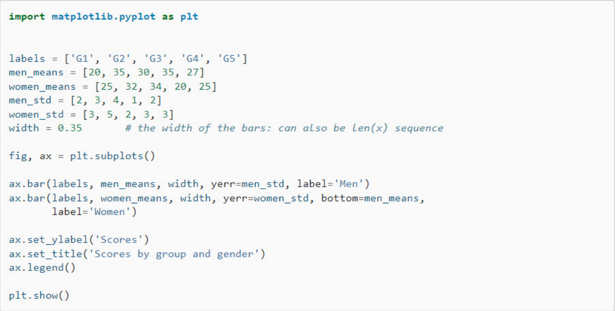
---

[5] https://matplotlib.org/cheatsheets/_images/cheatsheets-1.png

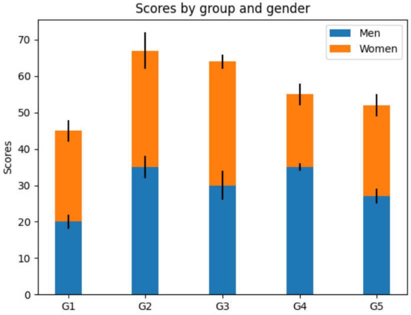
How to Learn Matplotlib

Follow a detailed Matplotlib tutorial:
`https://matplotlib.org/stable/tutorials/index.html`

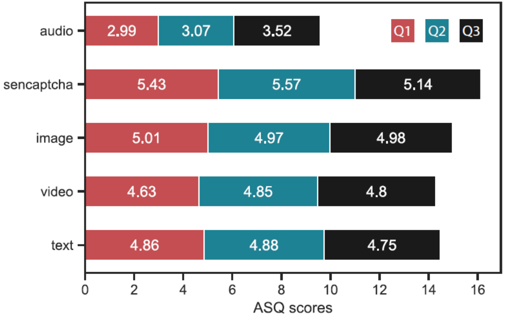
Stacked Bar Chart



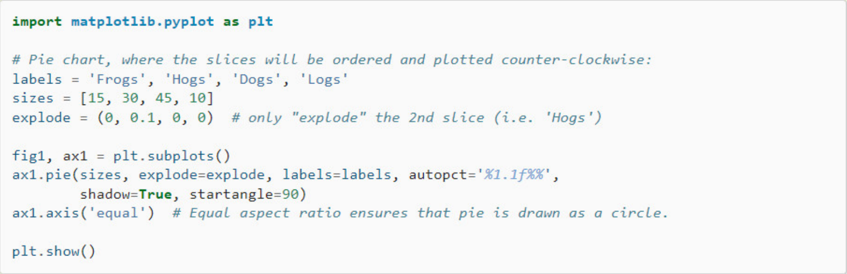
```
import matplotlib.pyplot as plt


labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 35, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]
men_std = [2, 3, 4, 1, 2]
women_std = [3, 5, 2, 3, 3]
width = 0.35       # the width of the bars: can also be len(x) sequence

fig, ax = plt.subplots()

ax.bar(labels, men_means, width, yerr=men_std, label='Men')
ax.bar(labels, women_means, width, yerr=women_std, bottom=men_means,
       label='Women')

ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.legend()

plt.show()
```



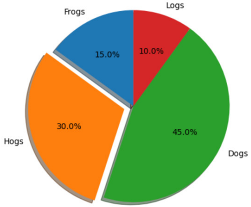
Stacked Bar Chart



Pie Chart

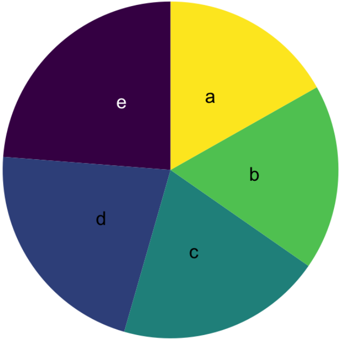


```python
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```



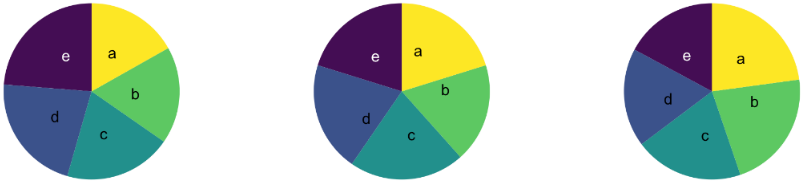
Bye Bye, Pie[6]

Humans are pretty bad at reading angles



Bye Bye, Pie[7]

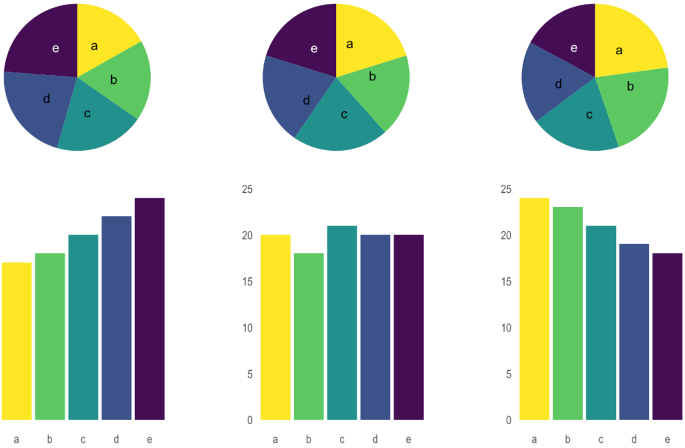
Humans are pretty bad at reading angles



---

[7] `https://www.data-to-viz.com/caveat/pie.html`

Bye Bye, Pie[8]

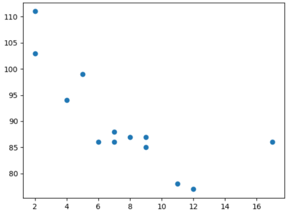


---

[8] `https://www.data-to-viz.com/caveat/pie.html`

Scatter Plot[9]

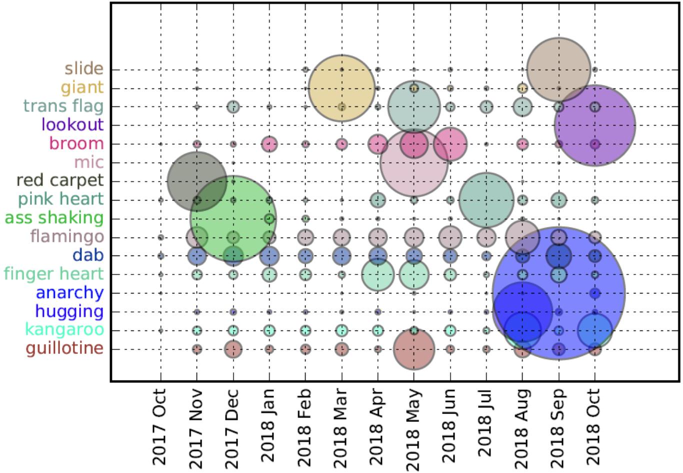
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```



---

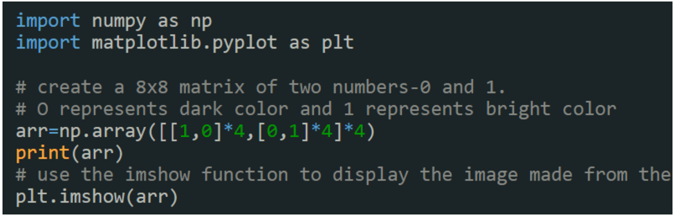
[9] https://www.w3schools.com/python/matplotlib_scatter.asp

Scatter Plot

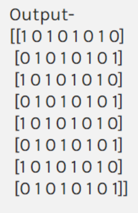


*The teaching materials are reorganized and reformed based on Prof. Ravi Vadapalli's slides (Ravi.Vadapalli@unt.edu, UNT & University of Miami)

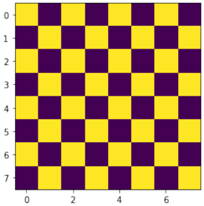
Image Show - *imshow()*[10]



```
import numpy as np
import matplotlib.pyplot as plt

# create a 8x8 matrix of two numbers-0 and 1.
# O represents dark color and 1 represents bright color
arr=np.array([[1,0]*4,[0,1]*4]*4)
print(arr)
# use the imshow function to display the image made from the
plt.imshow(arr)
```



```
Output-
[[1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]]
```



---

[10] https://www.pythonpool.com/matplotlib-imshow/

Image Show - *imshow()*

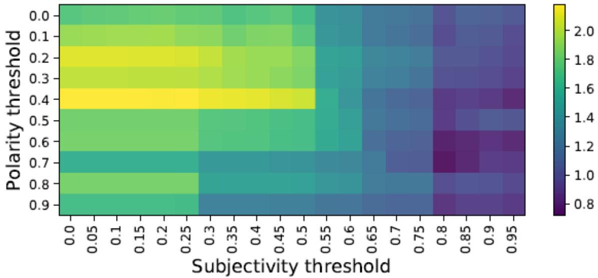
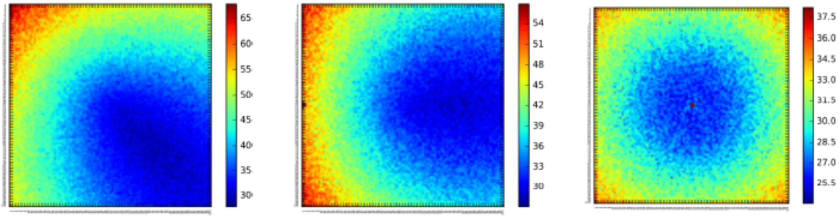


Figure 12: The ratio of # of positive tweets over # of negative tweets with different polarity and subjectivity thresholds. Positive emotions dominate when the ratio is greater than one. Otherwise, negative emotions are more popular.

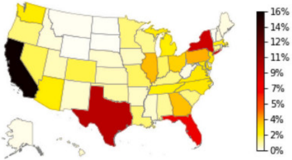
Image Show - *imshow()*



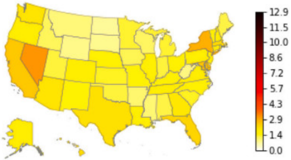
(a) Corner (b) Mid edge (c) Center

Fig. 5. Values of $\mu - 2.5\sigma$ for random guessing attacks

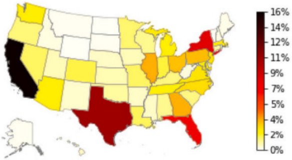
Map Plot - Basemap Matplotlib Toolkit



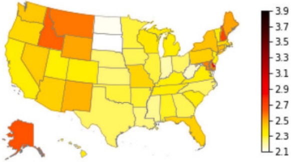
(a) Tweet distribution by state



(b) # of geo-tagged tweets per 1000 residents

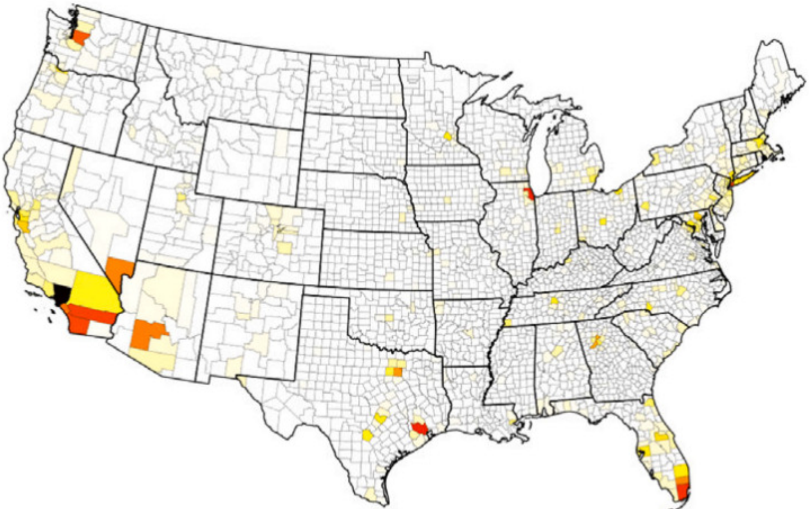


(c) Twitter user distribution by state



(d) # of geo-tagged tweets per Twitter user

Map Plot - Basemap Matplotlib Toolkit



Map Plot - Basemap Matplotlib Toolkit

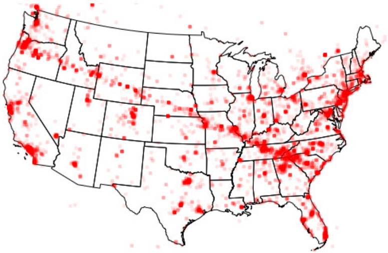


Fig. 6. Distribution of tweets with exact latitude and longitude coordinates.

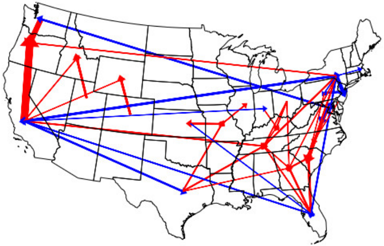


Fig. 7. The 50 most popular interstate trips during the event.

Map Plot - Basemap Matplotlib Toolkit

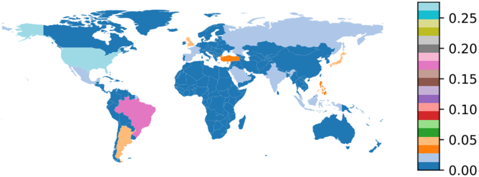


Figure 2: Country-level dist. of tweets

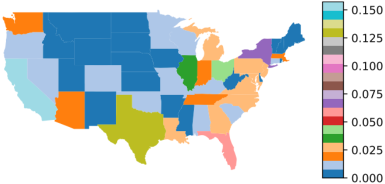


Figure 3: State-level dist. of U.S. tweets

Map Plot - Basemap Matplotlib Toolkit

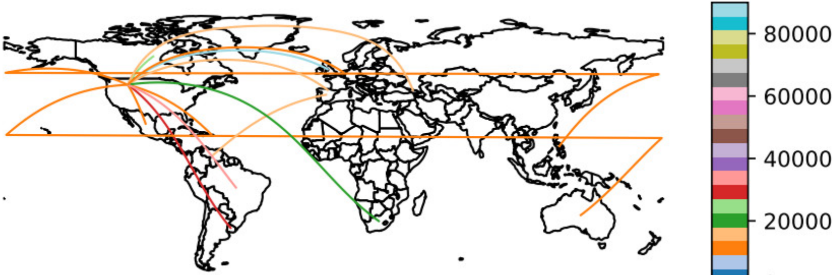


Figure 6: International travels in 2018

Others - Radar Chart

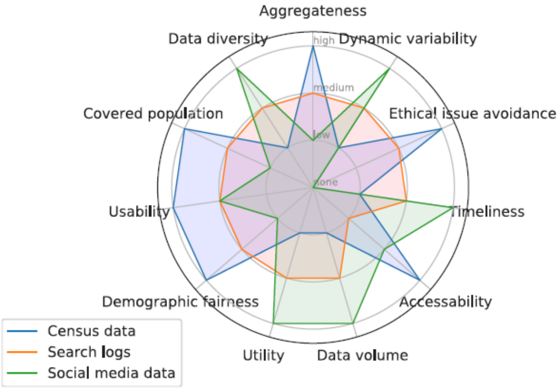


Fig. 1. Characteristics of census data, search logs data, and social media data.

Others - Word Cloud in Natural Language Processing

awesome
kind
amazing
incredible
happy
wonderful
welcome
impressive
perfectly
lol
great
win
loved
spectacular
nice
greatest
honestly
perfect
experienced
honest
lmao
liked
best
good
bright
loving
brilliant
excellent
breathtaking
beautiful
magnificent

Others - Emoji Cloud

pip install EmojiCloud

- Tutorial: `https://lnkd.in/gY_BMWP6`
- Source Code: `https://lnkd.in/gD-R9uai`
- Paper: `https://lnkd.in/g7DDgjE6`
- Slides: `https://lnkd.in/gs_sTTiG`
- Online Service: `https://emojicloud.org` (available soon)

Others - PCA and t-SNE for Dimensionality Reduction[11]

- PCA: Principal component analysis
  - Try to find a global structure
  - Low dimensional subspace
  - Can lead to local inconsistencies (far away point can become nearest neighbors)
- t-SNE: t-distributed Stochastic Neighbor Embedding
  - Try to preserve local structure
  - Low dimensional neighborhood should be the same as original neighborhood.

---

[11] https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec13_handout.pdf

Others - PCA and t-SNE for Dimensionality Reduction[12]

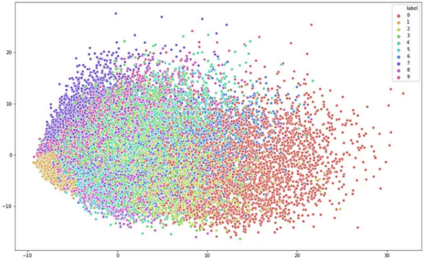


Figure 3: MNIST after PCA

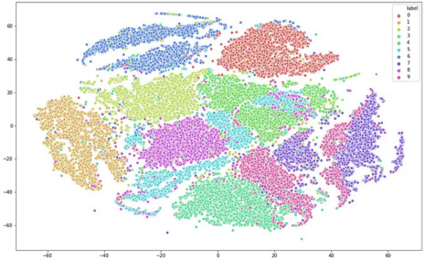
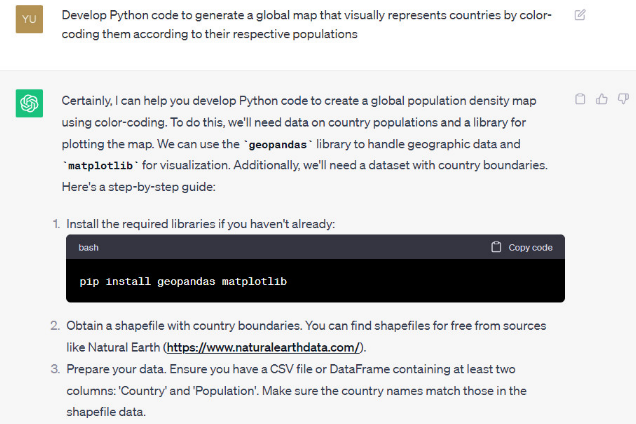


Figure 4: MNIST after t-SNE

---

[12] `shorturl.at/sAPR7`

ChatGPT for Data Visualization



Develop Python code to generate a global map that visually represents countries by color-coding them according to their respective populations

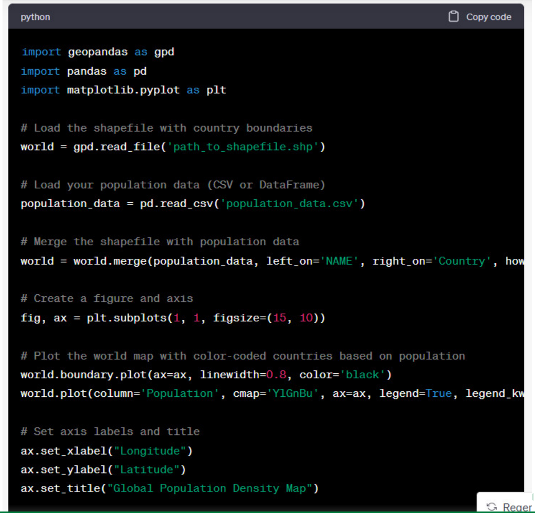
Certainly, I can help you develop Python code to create a global population density map using color-coding. To do this, we'll need data on country populations and a library for plotting the map. We can use the `` `geopandas` `` library to handle geographic data and `` `matplotlib` `` for visualization. Additionally, we'll need a dataset with country boundaries. Here's a step-by-step guide:

1. Install the required libraries if you haven't already:

```bash
pip install geopandas matplotlib
```

2. Obtain a shapefile with country boundaries. You can find shapefiles for free from sources like Natural Earth (https://www.naturalearthdata.com/).
3. Prepare your data. Ensure you have a CSV file or DataFrame containing at least two columns: 'Country' and 'Population'. Make sure the country names match those in the shapefile data.

ChatGPT for Data Visualization



```python
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt

# Load the shapefile with country boundaries
world = gpd.read_file('path_to_shapefile.shp')

# Load your population data (CSV or DataFrame)
population_data = pd.read_csv('population_data.csv')

# Merge the shapefile with population data
world = world.merge(population_data, left_on='NAME', right_on='Country', how

# Create a figure and axis
fig, ax = plt.subplots(1, 1, figsize=(15, 10))

# Plot the world map with color-coded countries based on population
world.boundary.plot(ax=ax, linewidth=0.8, color='black')
world.plot(column='Population', cmap='YlGnBu', ax=ax, legend=True, legend_kw

# Set axis labels and title
ax.set_xlabel("Longitude")
ax.set_ylabel("Latitude")
ax.set_title("Global Population Density Map")
```

1 Concept of Data Visualization

2 Data Visualization in Python
  2D Visualization in Python
  3D Visualization in Python
  4D Visualization in Python

3 Assignment

Three-Dimensional Plotting in Matplotlib

`https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html`

1 Concept of Data Visualization

2 Data Visualization in Python
  2D Visualization in Python
  3D Visualization in Python
  4D Visualization in Python

3 Assignment

4D Visualization

1. Concept of Data Visualization
2. Data Visualization in Python
3. **Assignment**

Assignment-2 (4.0 pt.)

- Plot a 2D stacked bar chart using Python (2 pts.)
- Plot a 3D figure using Python (2 pts.)