Lisbon School of Economics & Management
Universidade de Lisboa

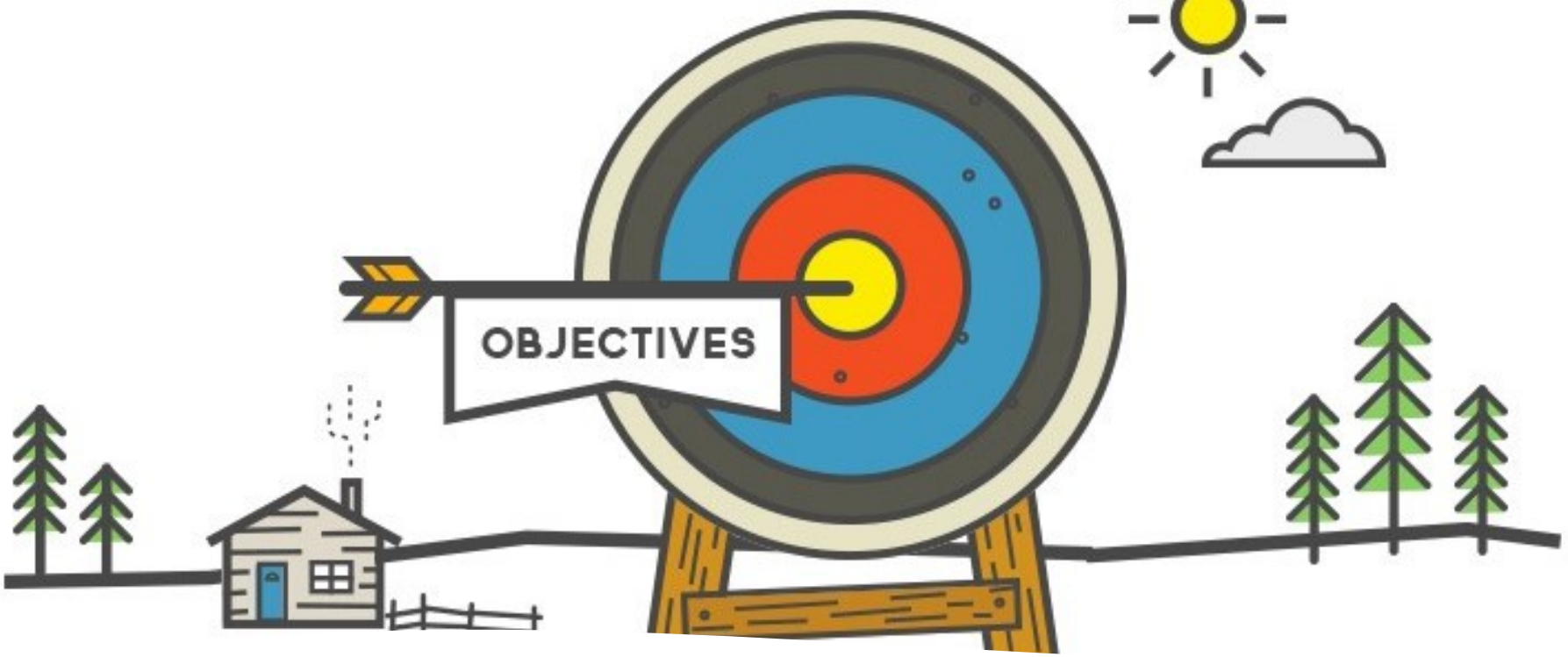
LISBOA
UNIVERSIDADE DE LISBOA

Visualization with Python

Matplotlib, Seaborn, Plotly

Prof. Carlos J. Costa, PhD

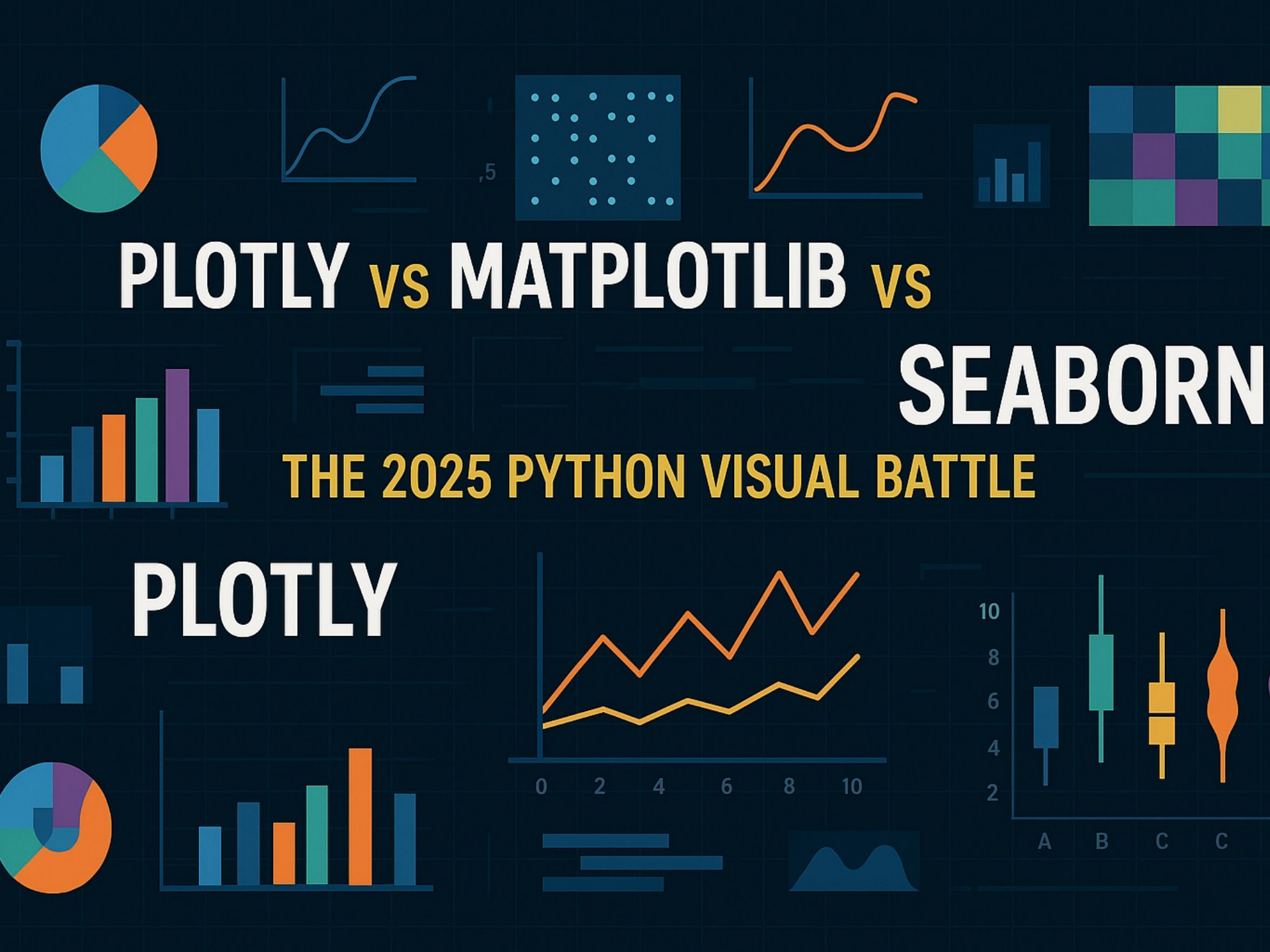
Saeed Angorani, DBA



Learning Goals

By the end of this session, you will:

- Create clear, professional visualizations
- Choose the right tool: static vs interactive
- Prepare messy data for visualization
- Build interactive dashboards
- Analyze real-world datasets


PLOTLY vs MATPLOTLIB vs
SEABORN
THE 2025 PYTHON VISUAL BATTLE
PLOTLY
0 2 4 6 8 10
10 8 6 4 2
A B C C

Python Visualization Stack

- **Matplotlib** - low-level control
- **Seaborn** - statistical and beautiful
- **Plotly** - interactive dashboards

When to Use What

Feature	Matplotlib	Seaborn	Plotly
Default styling	⭐	⭐⭐⭐	⭐⭐⭐⭐
Ease of use	⭐⭐	⭐⭐⭐⭐	⭐⭐⭐
Interactivity	⭐	⭐	⭐⭐⭐⭐⭐
Customization	⭐⭐⭐⭐	⭐⭐	⭐⭐⭐
Dashboard-ready	❌	❌	✅

Tool	Best For
Matplotlib	Full customization
Seaborn	Quick statistical plots
Plotly	Interactive dashboards


matplotlib

Basic Plot

```
import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [10,20,25,30]

plt.plot(x, y)
plt.title("Basic Line Plot")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

Customization

plt.plot(x, y, color='red', linestyle='--', marker='o’)

- Colors
- Markers
- Line styles

Multiple Plots

```
plt.plot(x, y)
plt.plot(x, [15,18,22,28])
plt.legend(["A", "B"])
```

Subplots

```
fig, ax = plt.subplots(1,2)

ax[0].plot(x,y)
ax[1].bar(x,y)
```

Exercise (5 min)

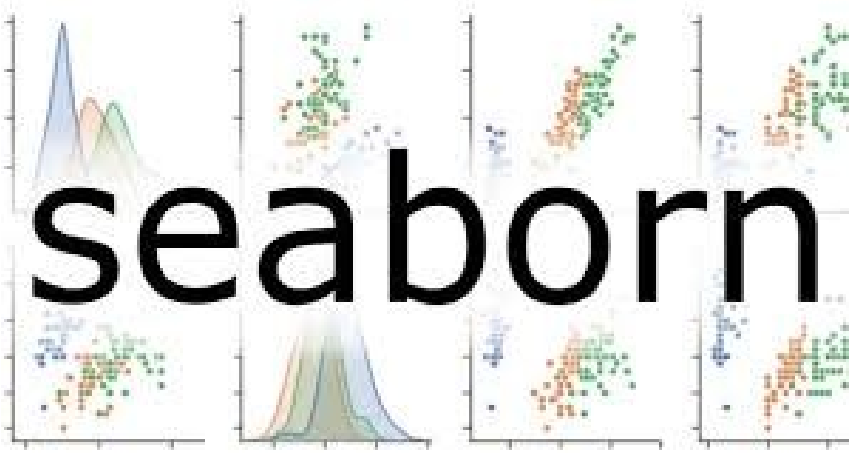
Create:

- Line chart
- Bar chart
- Add title and labels


seaborn

Why Seaborn

- Built on Matplotlib
- Better defaults
- Designed for data analysis

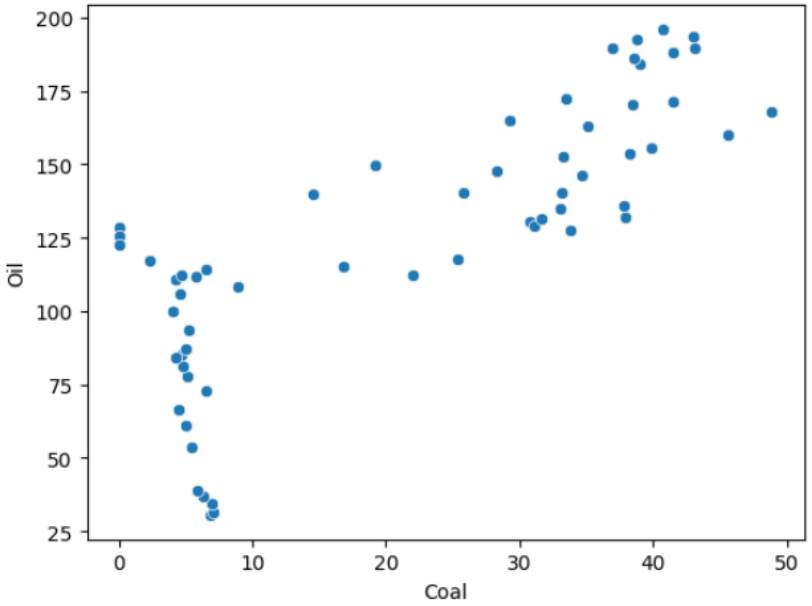


Load Dataset

```
import pandas as pd
import seaborn as sns
file="https://raw.githubusercontent.com/masterfloss/energyCountry/refs/heads/main/energy-consumption-by-source-and-country.csv"
df=pd.read_csv(file)
df.head()
```

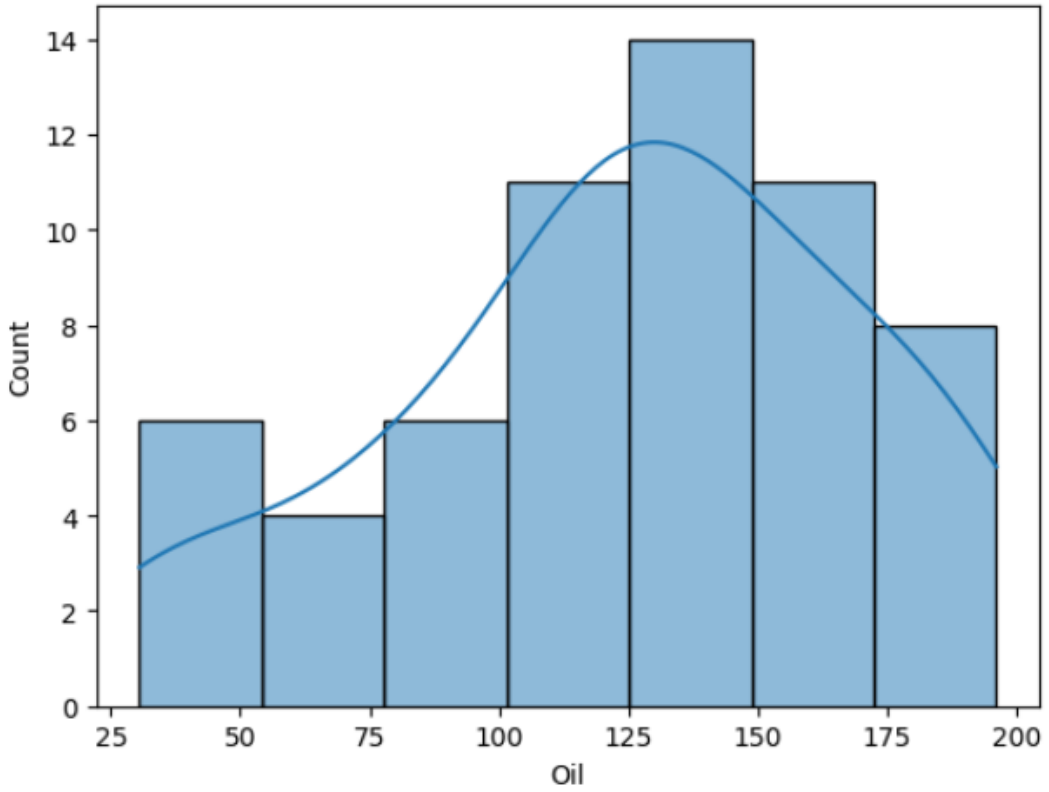
Scatter Plot

```
#Scatter Plot
dfP=df[df.Entity=="Portugal"]
sns.scatterplot(data=dfP, x="Coal", y="Oil")
```



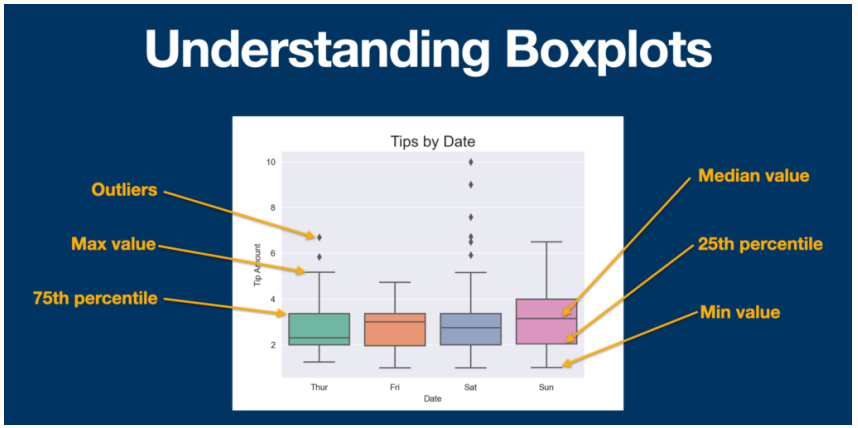
Distribution Plot

```
sns.histplot(dfP.dropna()["Oil"], kde=True)
```



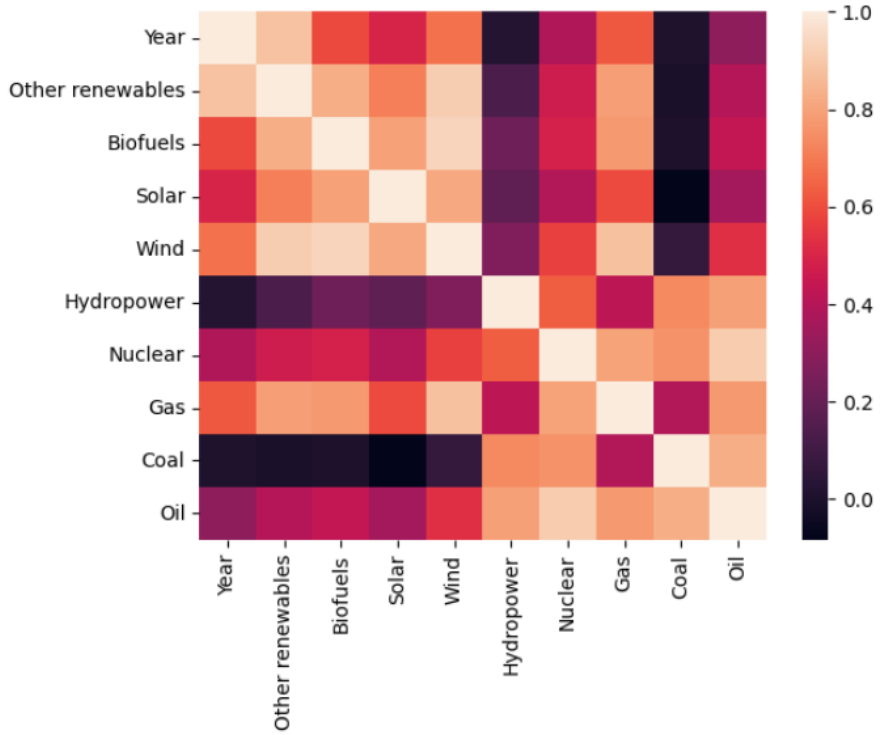
Boxplot

```
dfPS=df[(df.Entity=="Portugal") | (df.Entity=="Spain")]
sns.boxplot(data=dfPS, x="Entity", y="Oil")
```



Heatmap (Correlation)

```
Df_Nb=dfPS.select_dtypes(include='number’)
sns.heatmap(Df_Nb.corr())
```



Exercise (5 min)

- Using dataset:
- Create:
  - Histogram
  - Boxplot
  - Heatmap


plotly

Why Plotly

- Interactive charts
- Zoom, hover, filter
- Great for dashboards

Basic Plot

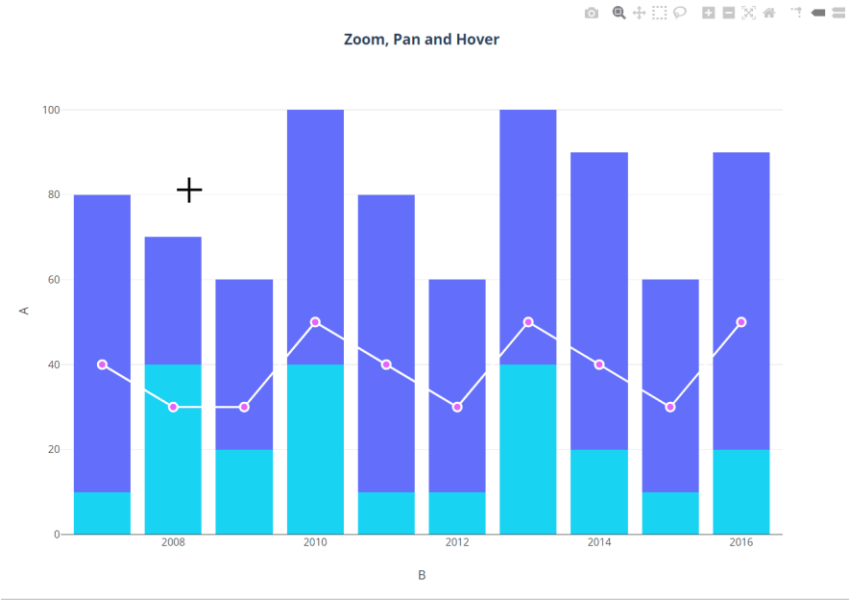
```
import plotly.express as px

fig = px.scatter(dfPS, x="Coal", y="Oil")
fig.show()
```

```
fig = px.scatter(dfPS, x="Coal", y="Oil", )
fig.show()
```

Interactive Features

- Hover info
- Zoom
- Dynamic filtering



Interactive Bar Chart

```
fig = px.bar(dfPS, x="Year", y="Solar", color="Entity")
fig.show()
```

Dashboard Style

```
fig = px.scatter(dfPS, x="Coal", y="Oil",
                 size="Solar", color="Entity")
fig.show()
```

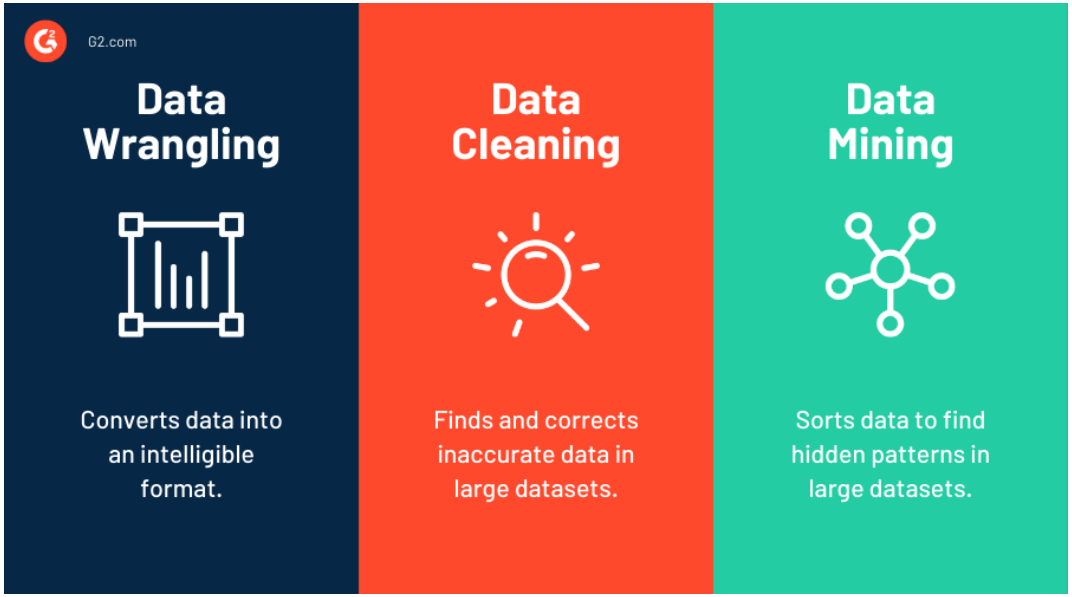
Exercise (5 min)

Create:

- Interactive scatter
- Add color + size dimensions

DATA WRANGLING

- **Why Data Prep Matters**
  - Bad data contribute to misleading charts



Common Tasks

- Handling missing values
- Filtering
- Grouping
- Aggregation

Example

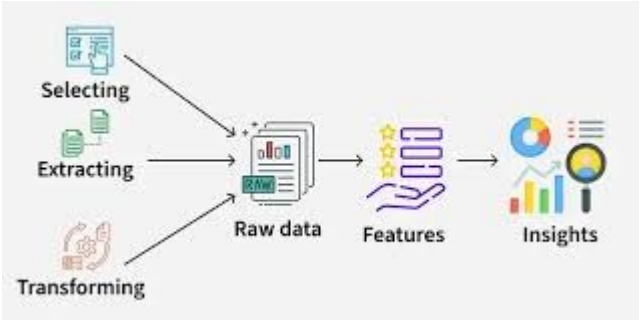
```
df.groupby("Entity")["Per capita energy consumption"].mean()
```

Cleaning Data

df.dropna()
df.fillna(0)

Feature Engineering

- Feature Selection
- Feature Transformation
- Feature Creation (Encoding, Binning)
- Feature Extraction (Automated in Deep Learning)



Feature Engineering

```
file1=https://raw.githubusercontent.com/masterfloss/energyCountry/refs/heads/main/energy-consumption-by-source-and-country.csv
file2=https://raw.githubusercontent.com/masterfloss/energyCountry/refs/heads/main/per-capita-energy-use.csv

df1 = pd.read_csv(file1)
df2 = pd.read_csv(file2)
df_merged = pd.merge(df1, df2, on=['Entity', 'Year'], how='inner')
df_merged.head()
```

Feature Engineering

```
listEnergies=['Other
renewables','Biofuels','Solar','Wind','Hydropower','Nuclear','Gas','Co
al','Oil']

total_energy = df_merged[listEnergies].sum(axis=1)


for i in listEnergies:
    df_merged[i+'_percentual'] = df_merged[i] / total_energy
```

Lab Task (20 min)

- Obtain Dataset
- Clean dataset
- Create:
  - Line plot
  - Boxplot
  - Heatmap
- Build:
  - Interactive Plotly chart

Extra

- Create a **mini dashboard**
- Combine:
  - Scatter
  - Bar
  - KPI metric

Key Takeaways

- Matplotlib - control
- Seaborn - simplicity and stats
- Plotly - interactivity
- Data prep is essential

Q&A / Discussion

"Any questions?"