MCDA 5511 TUTORIAL

Python Data Visualization

Matplotlib, Seaborn & Plotly

Somto Muotoe

January 2026

What You Will Learn

Matplotlib

The foundation - line, scatter, bar, histogram

Seaborn

Statistical plots - box, violin, heatmap, pairplot

Plotly

Interactive - hover, zoom, animations

By the end, you will know which library to use for any visualization task.

Why Visualize Data?

See patterns. Spot outliers. Tell stories.

Numbers Can Deceive

Anscombe's Quartet (1973): Four datasets with identical summary statistics but wildly different distributions.

Dataset I

x	y
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68

Dataset II

x	y
10	9.14
8	8.14
13	8.74
9	8.77
11	9.26
14	8.10
6	6.13
4	3.10
12	9.13
7	7.26
5	4.74

Dataset III

x	y
10	7.46
8	6.77
13	12.74
9	7.11
11	7.81
14	8.84
6	6.08
4	5.39
12	8.15
7	6.42
5	5.73

Dataset IV

x	y
8	6.58
8	5.76
8	7.71
8	8.84
8	8.47
8	7.04
8	5.25
19	12.50
8	5.56
8	7.91
8	6.89

All Four Datasets Share:

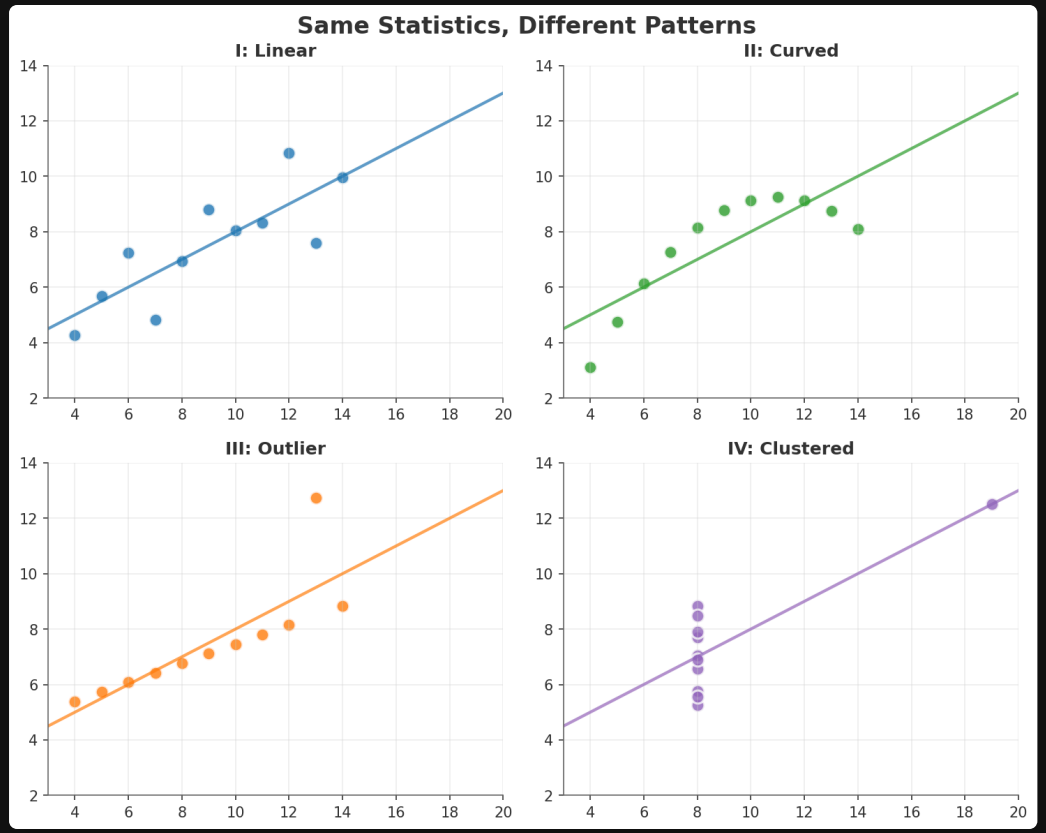
Mean of X:	9.0	Mean of Y:	7.5
Correlation:	0.816	Regression:	y = 0.5x + 3

?

What do these datasets look like when plotted?

Same Statistics, Different Patterns

Four datasets that share identical summary statistics but tell completely different stories.



I: Linear
A true linear relationship where the regression line fits well. This is what the statistics suggest all datasets should look like.

II: Curved
The data follows a clear curve, but linear regression misses it entirely. Lesson: always check for nonlinear patterns.

III: Outlier
One extreme point (y=12.74) dramatically influences the regression line. Without visualization, you would never know.

IV: Clustered
All points cluster at x=8 except one at x=19. The regression is entirely determined by that single outlier.

Always visualize your data before analysis. Summary statistics can completely miss patterns, outliers, and data quality issues.

The Python Visualization Stack

Matplotlib

Full control for publications

- Foundation of Python plotting
- Pixel-perfect customization
- PDF, PNG, SVG export

Seaborn

Beautiful statistical plots fast

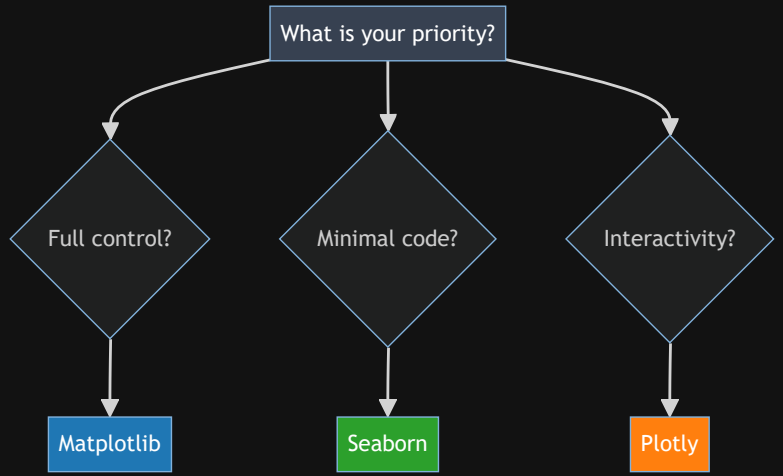
- Built on Matplotlib
- One-line statistical charts
- Works with DataFrames

Plotly

Interactive web visualizations

- Hover, zoom, pan
- Animations
- HTML export

When to Use What



Matplotlib
Publication-quality figures with pixel-perfect control over every element

Seaborn
Quick exploration with beautiful defaults and minimal code

Plotly
Dashboards and web apps with interactive hover, zoom, and pan

Note: All three can create statistical plots. Choose based on workflow priorities.

Built-in Datasets

We will use datasets that come with the libraries - no downloads needed.

Tips

Restaurant tipping data

```
import seaborn as sns
tips = sns.load_dataset("tips")
```

244 rows: bill, tip, day, time, size

Penguins

Palmer penguins measurements

```
import seaborn as sns
df = sns.load_dataset(
    "penguins")
```

344 rows: species, bill, flipper, mass

Gapminder

World development data

```
import plotly.express as px
df = px.data.gapminder()
```

1704 rows: country, year, GDP, life expectancy

Matplotlib

The Foundation of Python Visualization

The Basic Pattern

Every Matplotlib plot follows the same pattern:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()     # 1. Create figure and axes
ax.plot(x, y)                 # 2. Plot your data
ax.set_xlabel("X Label")      # 3. Add labels
ax.set_ylabel("Y Label")
ax.set_title("Title")
plt.savefig("plot.png")       # 4. Save or show
```

Key insight: `fig` is the canvas, `ax` is where you draw. Always use `fig, ax = plt.subplots()` to start.

Line Plots

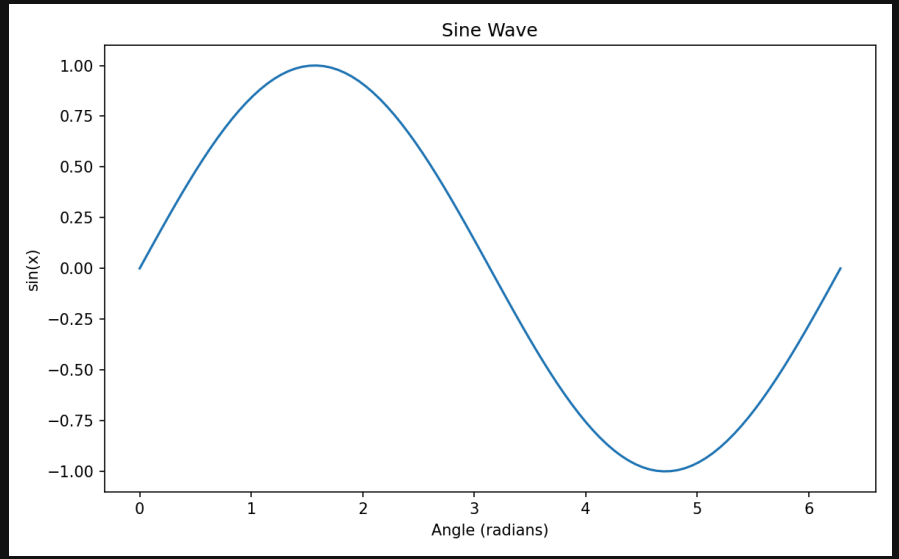
Connect data points with lines.

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

x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x)

fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title("Sine Wave")
```

- Great for continuous data
- Shows trends over time
- Use markers with `marker="o"`



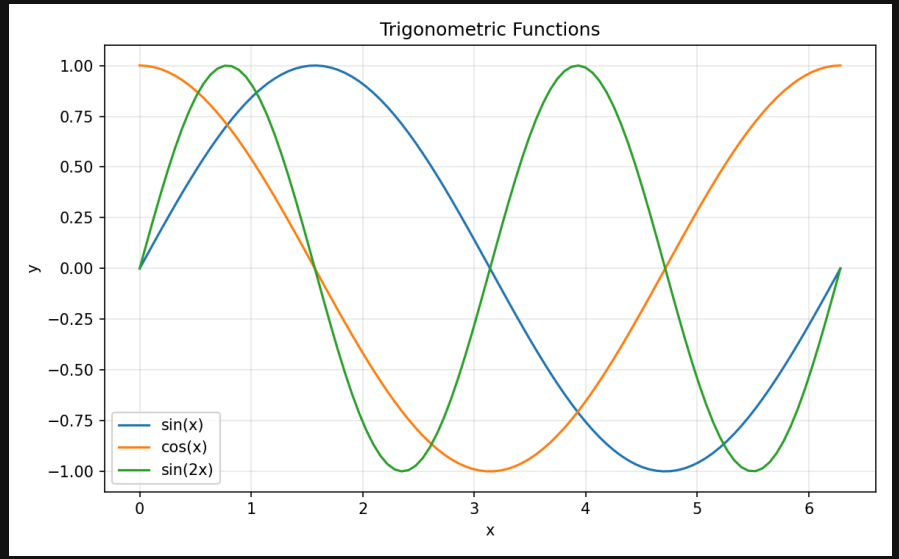
Multiple Lines

Compare series with a legend.

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

ax.plot(x, np.sin(x), label="sin")
ax.plot(x, np.cos(x), label="cos")
ax.legend()
```

- Use `label=` for each line
- Call `ax.legend()` to show
- Colors auto-assigned



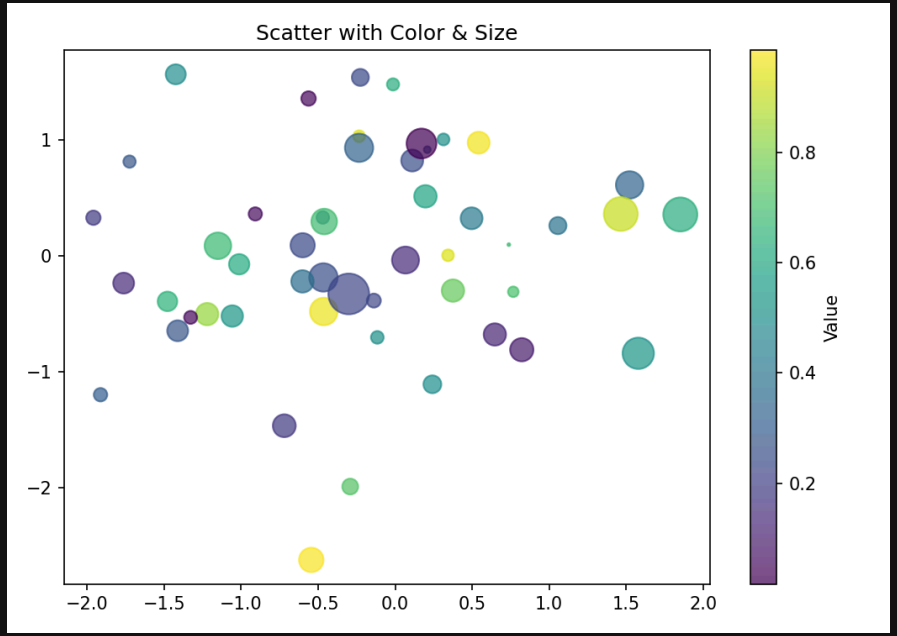
Scatter Plots

Show relationships between variables.

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

scatter = ax.scatter(
    x, y,
    c=colors,       # Color by value
    s=sizes,        # Size by value
    cmap="viridis"
)
fig.colorbar(scatter)
```

- `c=` maps color to data
- `s=` maps size to data
- Add colorbar for reference



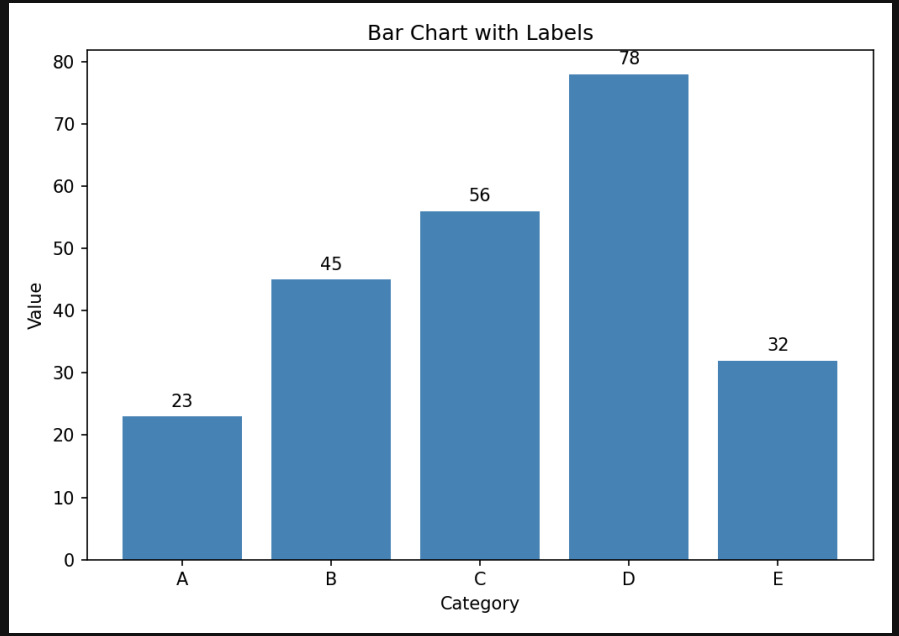
Bar Charts

Compare categories.

```
categories = ["A", "B", "C", "D"]
values = [23, 45, 56, 32]

fig, ax = plt.subplots()
ax.bar(categories, values)
```

- Use `ax.bar()` for vertical
- Use `ax.barh()` for horizontal
- Add value labels on top with `ax.text()`



Histograms

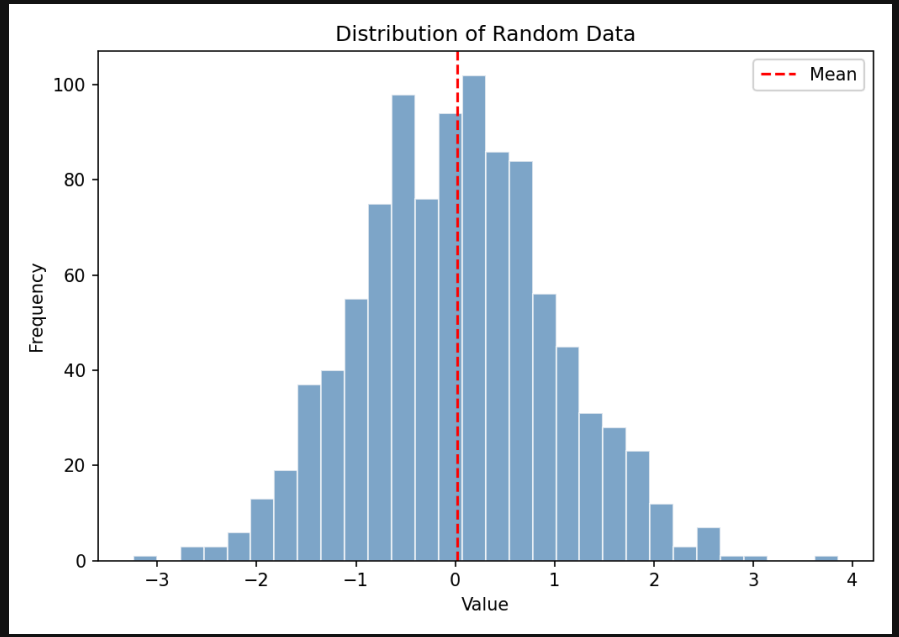
Show data distributions.

```
import numpy as np

data = np.random.randn(1000)

fig, ax = plt.subplots()
ax.hist(data, bins=30)
ax.axvline(data.mean(),
           color="red",
           linestyle="--")
```

- `bins=` controls granularity
- Show mean/median with `axvline()`
- Use `alpha=0.7` for overlapping



Subplots

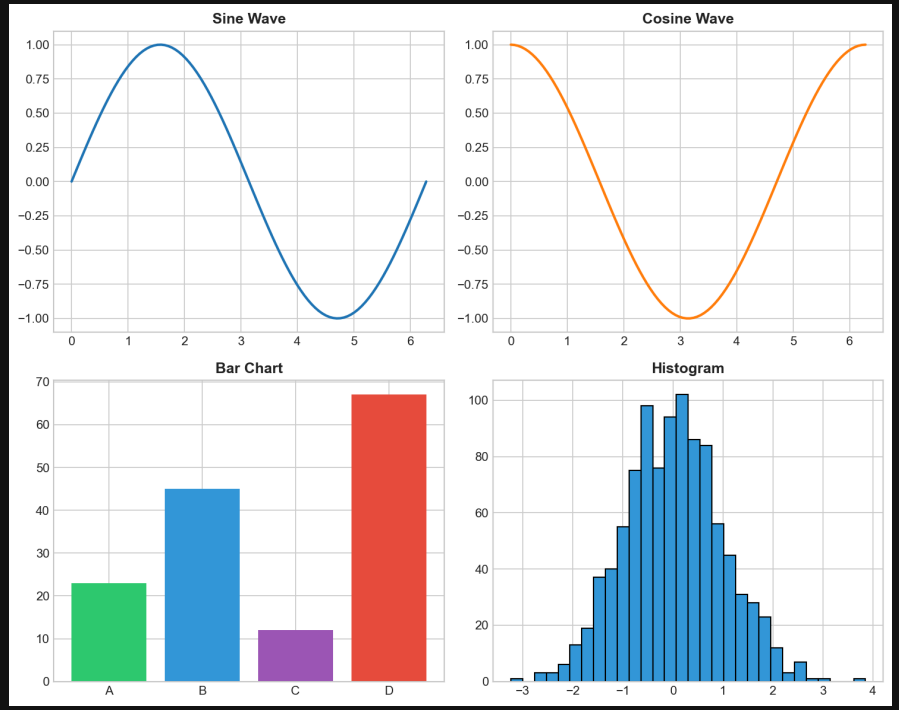
Multiple plots in one figure.

```
fig, axes = plt.subplots(2, 2)

axes[0, 0].plot(x, np.sin(x))
axes[0, 1].plot(x, np.cos(x))
axes[1, 0].bar(cats, vals)
axes[1, 1].hist(data)

plt.tight_layout()
```

- Access with `axes[row, col]`
- `tight_layout()` fixes spacing



Saving Figures

Export Code

```
fig.savefig("plot.png", dpi=300)
fig.savefig("plot.pdf")
fig.savefig("plot.svg")
```

> Always use `dpi=300` for print quality.

Common Options

```
fig.savefig("plot.png",
    dpi=300,              # Resolution
    bbox_inches="tight",  # Remove whitespace
    transparent=True,     # Clear background
    facecolor="white"     # Or set bg color
)
```

Format Guide

Format	Best For	Notes
PNG	Web, presentations	Raster, good compression
PDF	Publications, print	Vector, scales perfectly
SVG	Web, editing later	Vector, editable in Illustrator

Pro tip: Use `bbox_inches="tight"` to automatically crop whitespace around your figure.

Matplotlib Essentials

Common Customizations

```
ax.set_xlim(0, 10)        # Axis limits
ax.set_ylim(-1, 1)
ax.grid(True, alpha=0.3)  # Grid lines
ax.set_aspect("equal")    # Square plot
```

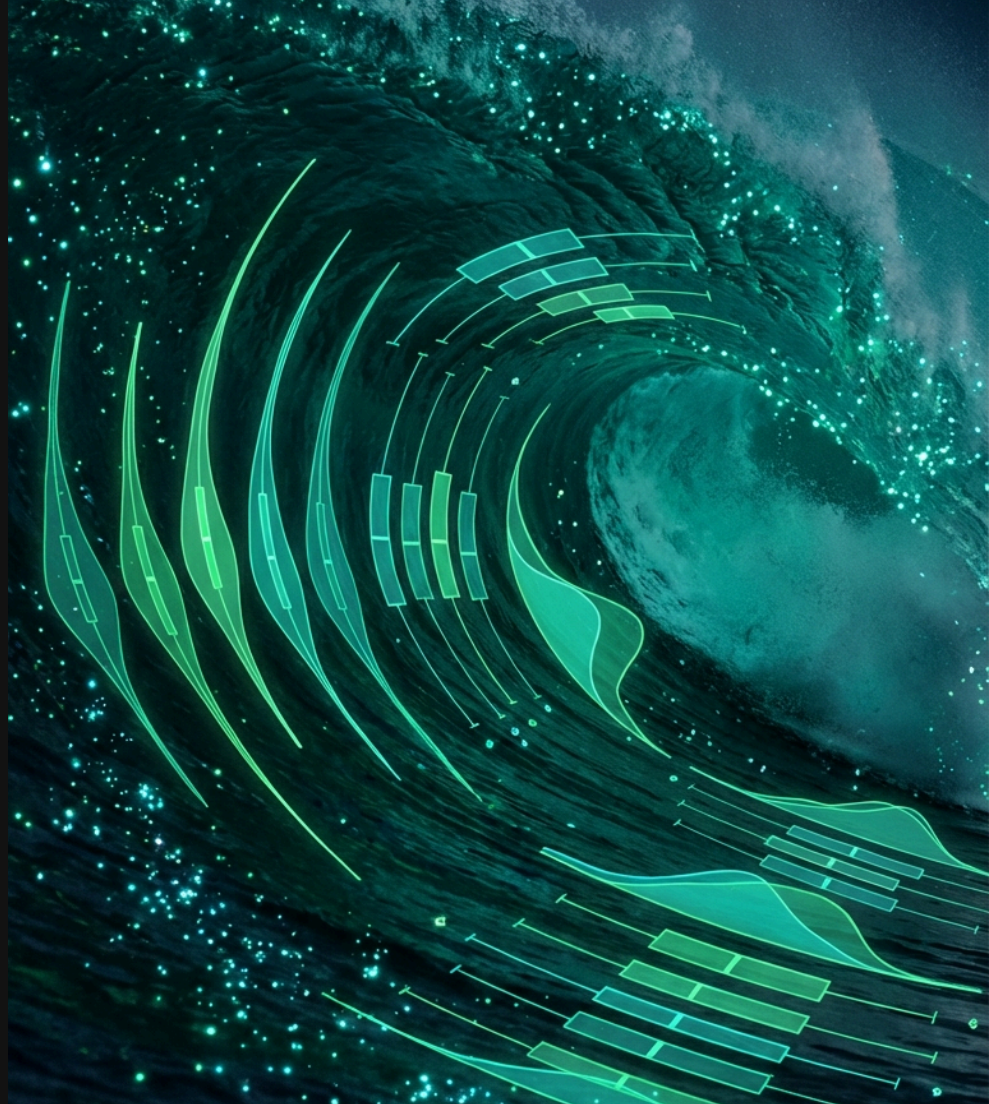
Line Styles

```
ax.plot(x, y, "r--")      # Red dashed
ax.plot(x, y, "b•–")      # Blue dot-dash
ax.plot(x, y, "go")       # Green circles
```

Remember: Matplotlib gives you full control. If something can be customized, Matplotlib can do it.

Seaborn

Statistical Visualization Made Easy



Seaborn in One Slide

What It Does

- Beautiful defaults out of the box
- Works directly with DataFrames
- Automatic statistical calculations
- Built on top of Matplotlib

The Pattern

```python
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset("tips")

fig, ax = plt.subplots()
sns.scatterplot(data=tips,
                x="total_bill",
                y="tip",
                ax=ax)
```

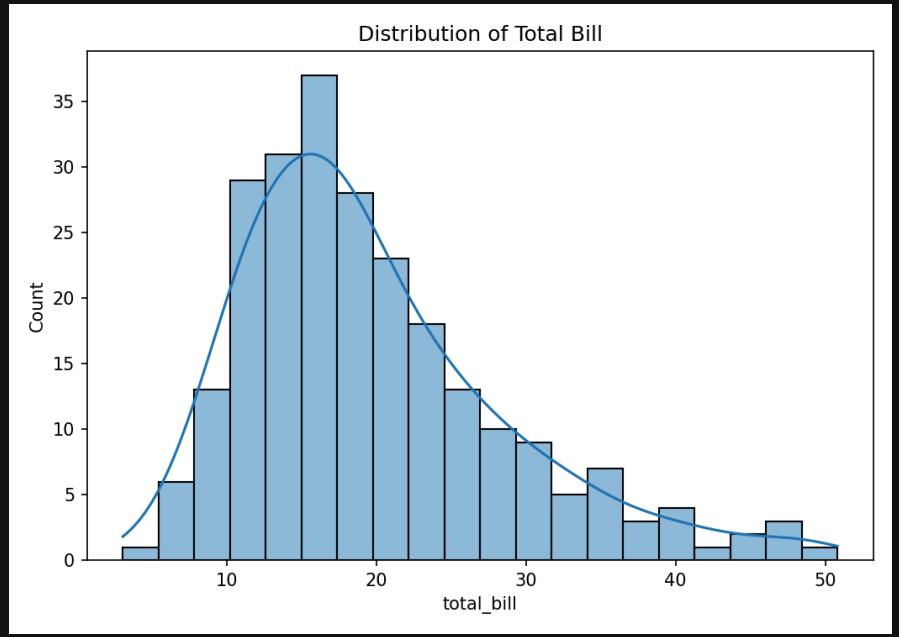
Key insight: Pass `data=` with a DataFrame, then use column names for `x=`, `y=`, `hue=`, etc.

Histograms

Show distribution of a single variable.

```
sns.histplot(
    data=tips,
    x="total_bill",
    kde=True     # Add density curve
)
```

- `kde=True` adds smooth density
- `hue=` colors by category
- `bins=` controls detail level

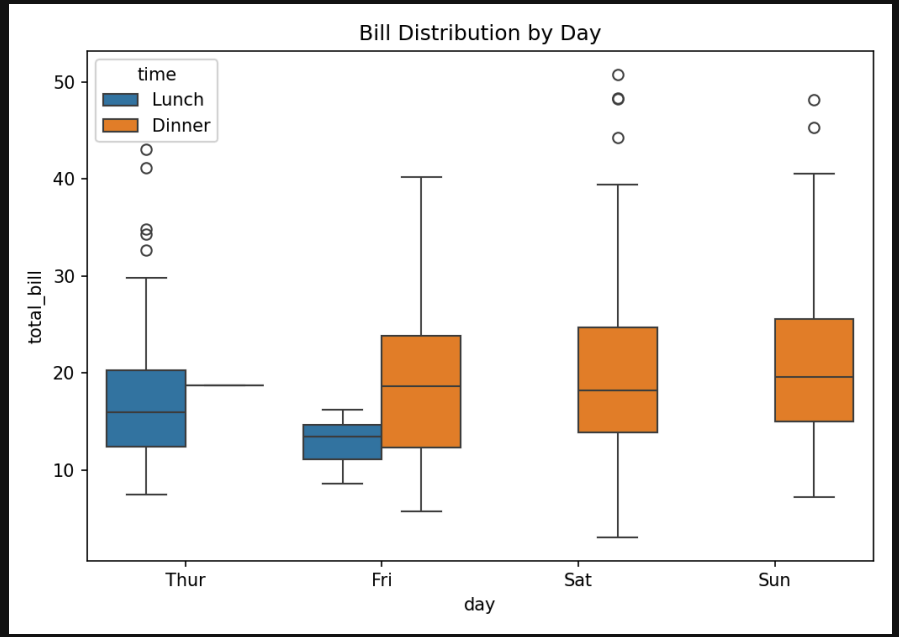


Box Plots

Compare distributions across categories.

```
sns.boxplot(
    data=tips,
    x="day",
    y="total_bill",
    hue="time"
)
```

- Shows median, quartiles, outliers
- Great for comparing groups
- Use `hue=` for sub-groups

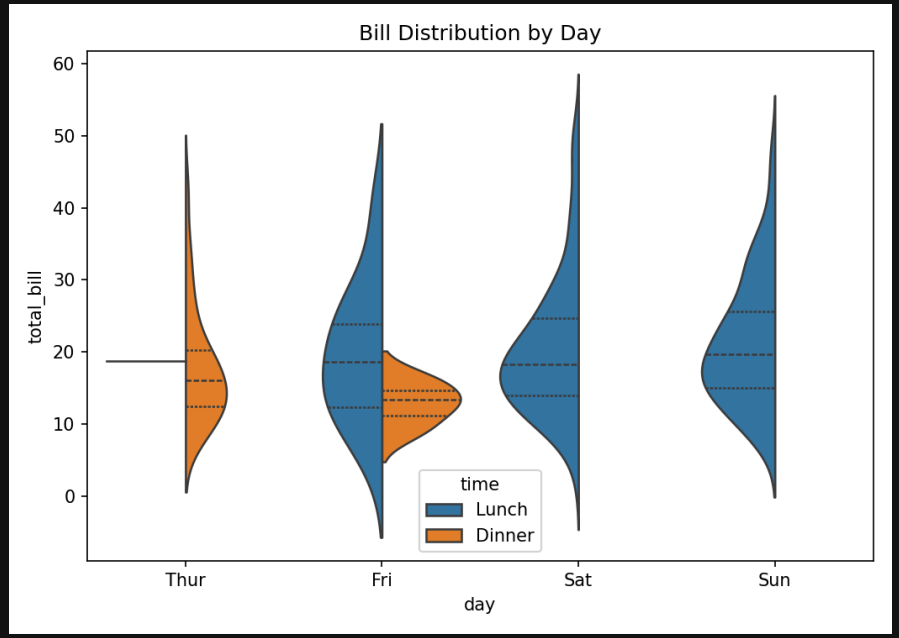


Violin Plots

See the full distribution shape.

```
sns.violinplot(
    data=tips,
    x="day",
    y="total_bill",
    hue="time",
    split=True
)
```

- Shows distribution shape via KDE
- `split=True` for side-by-side
- Better than box plots for bimodal data

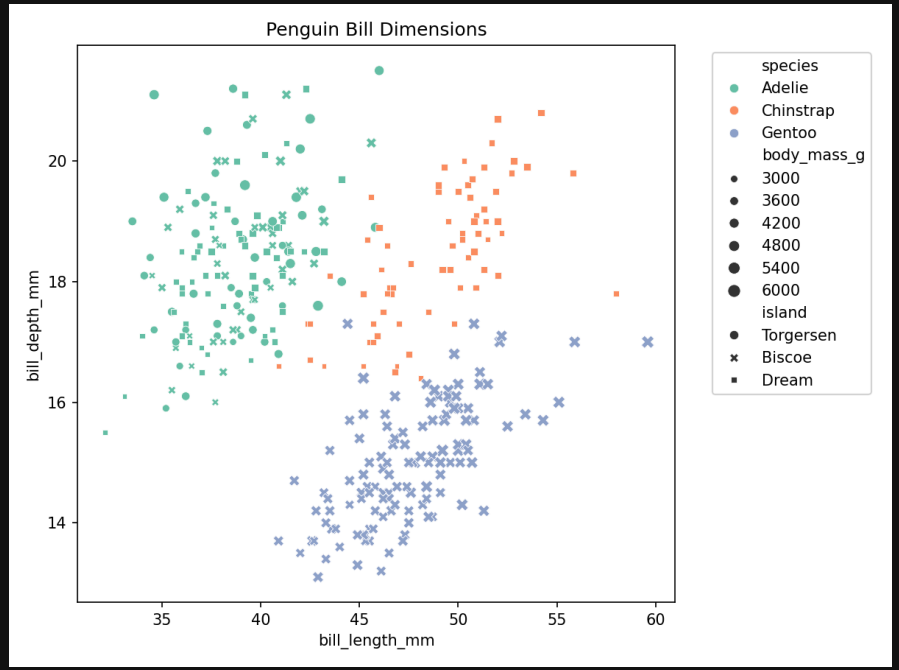


Scatter Plots with Encodings

Map multiple variables to visual properties.

```
sns.scatterplot(
    data=penguins,
    x="bill_length_mm",
    y="bill_depth_mm",
    hue="species",    # Color
    size="body_mass_g" # Size
)
```

- `hue=` colors by category
- `size=` scales by value
- `style=` changes marker shape

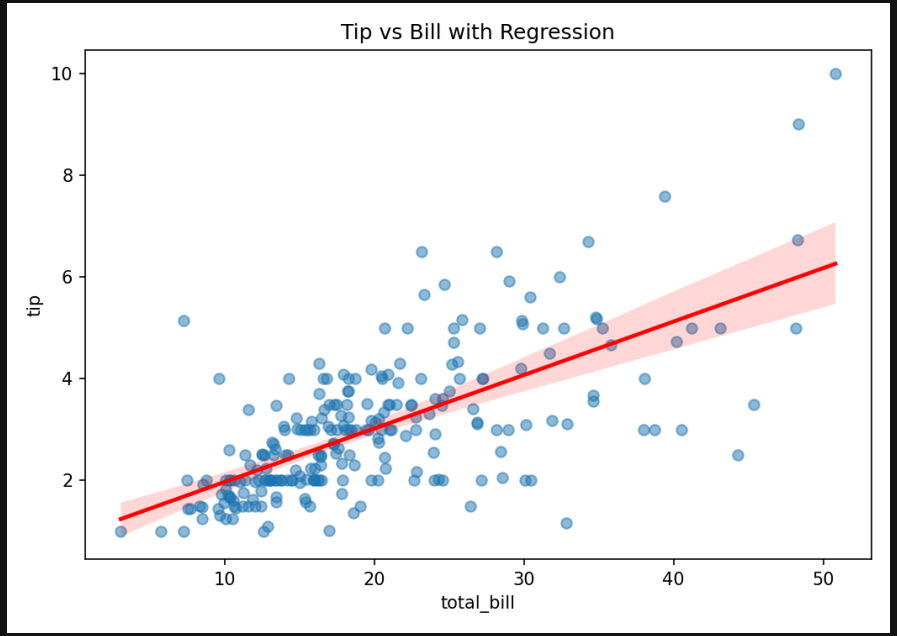


Regression Plots

Show relationships with trend lines.

```
sns.regplot(
    data=tips,
    x="total_bill",
    y="tip",
    ci=95
)
```

- Automatically fits regression
- Shaded area shows 95% CI
- Use `lmplot()` for faceting



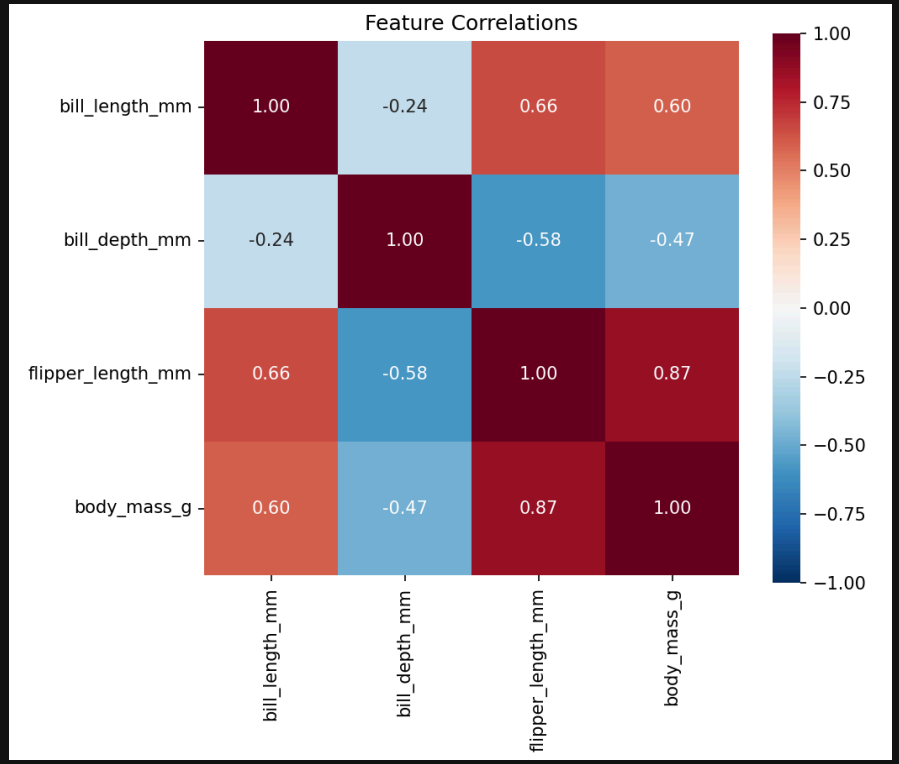
Heatmaps

Visualize correlations or matrices.

```
corr = penguins.select_dtypes(
    "number"
).corr()

sns.heatmap(
    corr,
    annot=True,
    cmap="RdBu_r",
    center=0
)
```

- `annot=True` shows values
- `center=0` for diverging data
- Perfect for correlation matrices

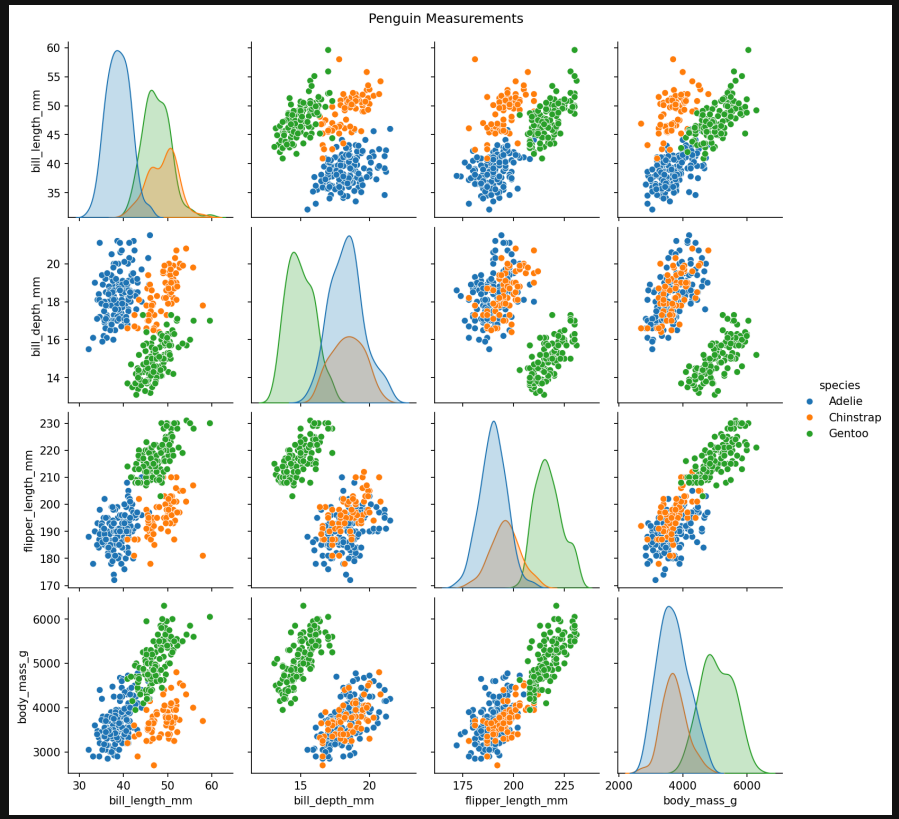


Pair Plots

See all relationships at once.

```
sns.pairplot(
    penguins,
    hue="species",
    diag_kind="kde"
)
```

- Scatter plots for all pairs
- Distributions on diagonal
- Great for exploratory analysis



Seaborn Essentials

Set a Theme

```
sns.set_theme(style="whitegrid")
# Options: darkgrid, whitegrid,
#          dark, white, ticks
```

Custom Palette

```
sns.set_palette("colorblind")
# Or: Set1, Set2, husl, viridis
```

Remember: Seaborn is built on Matplotlib. You can always customize further with `ax.set_title()`, `ax.set_xlabel()`, etc.

Plotly

Interactive Visualizations for the Web

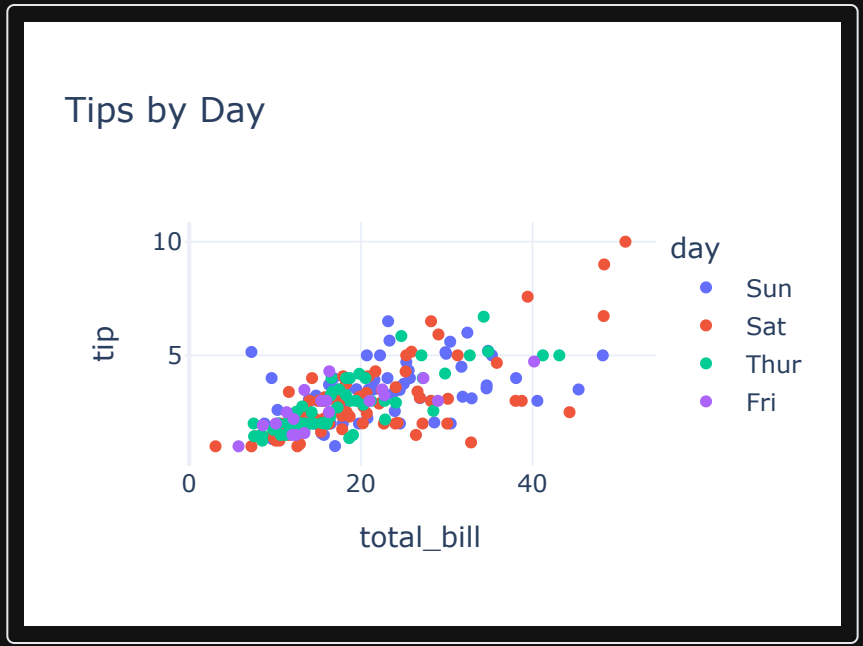
Why Interactive?

```python
import plotly.express as px

tips = px.data.tips()

fig = px.scatter(
    tips,
    x="total_bill",
    y="tip",
    color="day",
    hover_data=["size"]
)
fig.show()
```

Try it: hover for details, zoom by dragging, double-click to reset



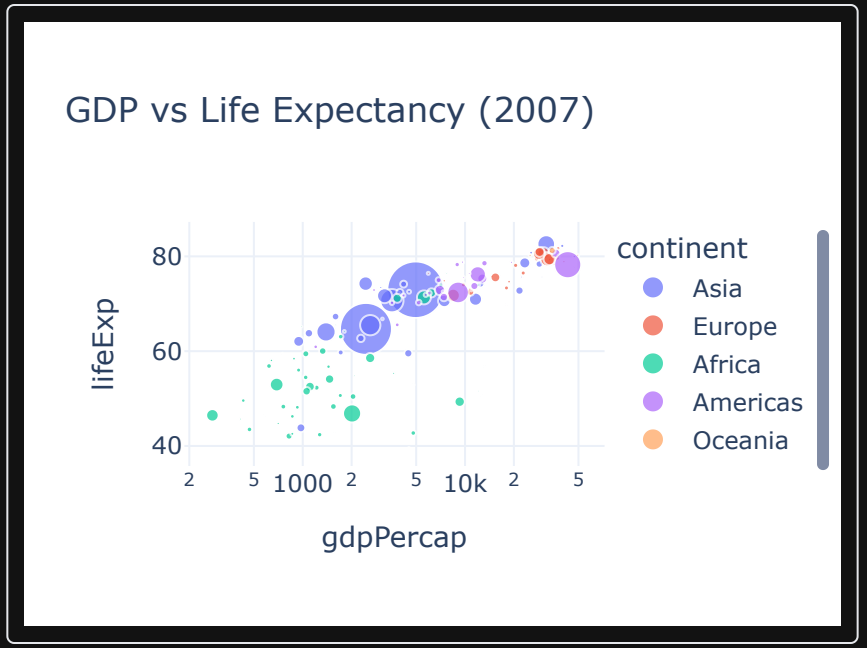
Interactive Scatter Plot

```python
import plotly.express as px

df = px.data.gapminder()
df_2007 = df[df["year"] == 2007]

fig = px.scatter(
    df_2007,
    x="gdpPercap",
    y="lifeExp",
    size="pop",
    color="continent",
    hover_name="country",
    log_x=True
)
```

- Hover to see country name
- Click legend to toggle continents
- Use toolbar to zoom/download

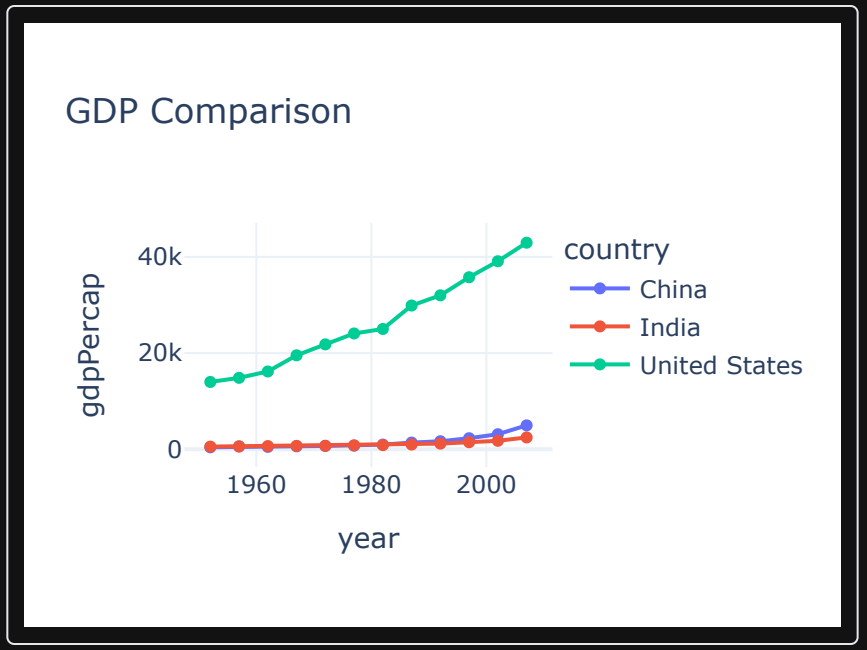


Line Charts

```python
countries = ["United States", "China", "India"]
df_subset = df[df["country"].isin(countries)]

fig = px.line(
    df_subset,
    x="year",
    y="gdpPercap",
    color="country",
    markers=True
)
```

- Hover shows exact values
- Click legend to hide/show lines
- Use `markers=True` for data points

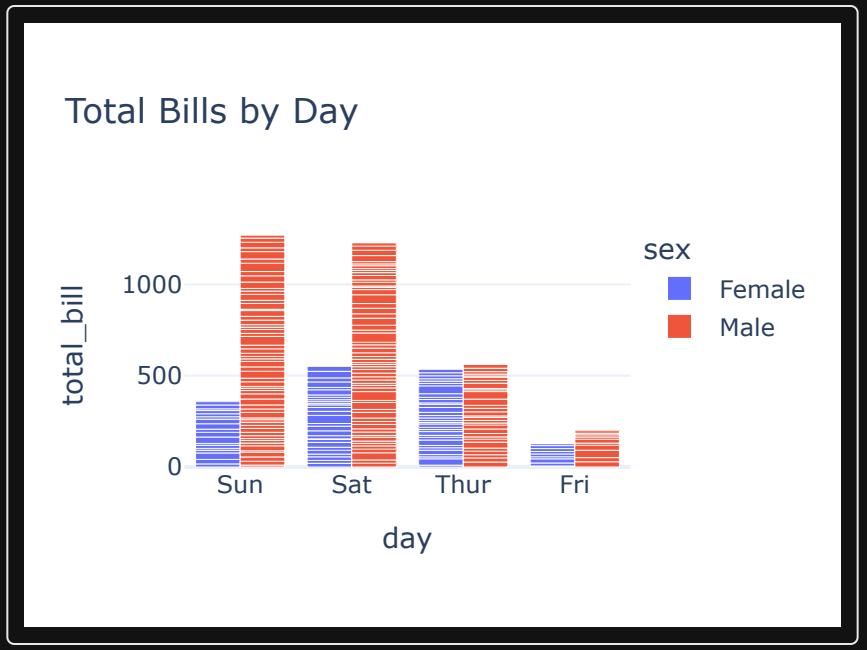


Bar Charts

```
tips = px.data.tips()

fig = px.bar(
    tips,
    x="day",
    y="total_bill",
    color="sex",
    barmode="group"  # or "stack"
)
```

- Hover shows individual bar values
- Click legend to filter categories
- Use `barmode="group"` or `barmode="stack"`

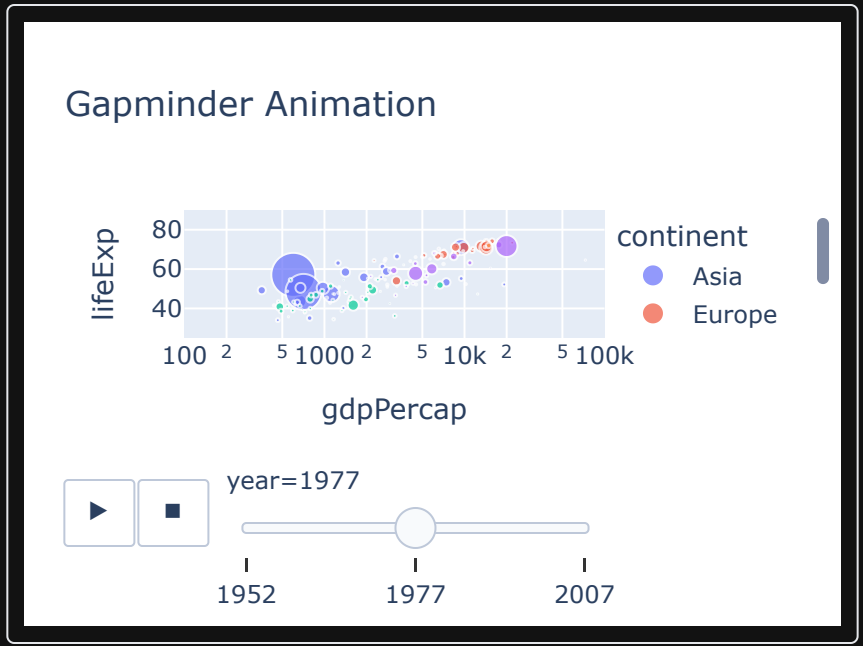


Animations

```python
df = px.data.gapminder()

fig = px.scatter(
    df,
    x="gdpPercap",
    y="lifeExp",
    size="pop",
    color="continent",
    hover_name="country",
    animation_frame="year",
    log_x=True,
    range_y=[25, 90]
)
```

- `animation_frame=` column to animate
- `range_x=` , `range_y=` fix axes
- Famous "Hans Rosling" style



Press play or drag the slider

3D Plots

```
df = px.data.iris()

fig = px.scatter_3d(
    df,
    x="sepal_length",
    y="sepal_width",
    z="petal_width",
    color="species"
)
```

- Drag to rotate the view
- Scroll to zoom in/out
- Great for 3+ dimensional data
- Use sparingly - 2D often clearer

Saving Plotly Figures

Interactive HTML

```
fig.write_html("plot.html")
```

- Opens in any browser
- Full interactivity preserved
- Can email or embed in websites

Static Images

```
# Requires: pip install kaleido
fig.write_image("plot.png")
fig.write_image("plot.pdf")
```

- Good for presentations
- Loses interactivity
- Same as Matplotlib output

Tip: Use HTML for sharing data exploration. Use PNG/PDF when interactivity is not needed.

Plotly Essentials

Themes

```
fig = px.scatter(df, x="x", y="y",
                 template="plotly_white")
# Options: plotly, plotly_white,
#          plotly_dark, ggplot2
```

Custom Layout

```
fig.update_layout(
    title="My Title",
    xaxis_title="X Label",
    yaxis_title="Y Label"
)
```

Remember: Plotly Express is for quick plots. For complex customization, Plotly also has `graph_objects` (not covered here).

Summary

Comparing the libraries and key takeaways

Quick Comparison

	Matplotlib	Seaborn	Plotly
Best For	Publications, full control	Quick stats, beautiful defaults	Web, exploration
Learning	Moderate	Easy	Easy
Interactivity	No	No	Yes
3D/Animation	Complex	No	Easy
Output	PNG, PDF, SVG	Via Matplotlib	HTML, PNG

Use Matplotlib when:

- You need pixel-perfect control
- Creating publication figures
- PDF/print output

Use Seaborn when:

- Exploring data quickly
- Need statistical plots
- Want beautiful defaults

Use Plotly when:

- Users need to interact
- Building dashboards
- Sharing via web

The Cheat Sheet

Matplotlib

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot(x, y)
ax.scatter(x, y)
ax.bar(cats, vals)
ax.hist(data)

fig.savefig("plot.png", dpi=300)
```

Seaborn

```
import seaborn as sns

sns.histplot(data=df, x="col")
sns.boxplot(data=df, x="a", y="b")
sns.scatterplot(data=df, x="a",
                y="b", hue="c")
sns.heatmap(corr, annot=True)
sns.pairplot(df, hue="cat")
```

Plotly

```
import plotly.express as px

px.scatter(df, x="a", y="b")
px.line(df, x="x", y="y")
px.bar(df, x="cat", y="val")
px.scatter(df, ...,
           animation_frame="year")

fig.write_html("plot.html")
```

Exercises

Hands-on practice with all three libraries

Practice Exercises

Apply what you learned to analyze real datasets.

Part 1: Sales Data

- Which region has highest revenue?
- Revenue trends over time
- Compare profit margins by product

Part 2: Weather Data

- Temperature distributions by city
- Humidity vs precipitation correlation
- Weather variable heatmap

Part 3: Experiment Data

- Dose-response curves with error bars
- Visualize statistical significance
- Compare treatments at highest dose

Part 4: Interactive (Plotly)

- Interactive sales explorer
- Animated weather comparison

Part 5: Challenge

- Tell a story with one figure

File: `exercises/visualization-exercises.ipynb` - Uses project datasets in `data/`

Resources

Matplotlib

- Gallery
- Cheatsheets
- Tutorials

Seaborn

- Gallery
- Tutorial
- API Reference

Plotly

- Examples
- Express Guide
- Dash (Dashboards)

Next step: Try recreating a visualization from one of the galleries using your own data.

Now, Go Build Something

- **Matplotlib** for full control
- **Seaborn** for quick statistics
- **Plotly** for interactivity

Before class: Try the exercises in the notebook.

somto.muotoe@smu.ca