Lesson 19 Summary: Multi-Panel Figures

Data Science with Python – Key Concepts

Data Science Program

Multi-Panel Overview

Multi-Panel Figures

Layout Tips:

`tight_layout()` | `constrained_layout` | `sharex/sharey`

Combine views to tell a complete data story

Multi-panel figures combine views for comprehensive analysis

plt.subplots()

Create grid of subplots:

- **Basic:** `fig, axs = plt.subplots(2, 2)`
- **Size:** `figsize=(12, 8)`
- **Shared axes:** `sharex=True, sharey=True`

Access subplots:

```
axs[0, 0].plot(...)  # Top-left
axs[1, 1].bar(...)  # Bottom-right
```

subplots() is the most common method for multi-panel

GridSpec

Custom grid layouts:

- Panels of different sizes
- Spanning multiple rows/columns
- Complex dashboard layouts

Example:

```
gs = fig.add_gridspec(3, 3)
ax1 = fig.add_subplot(gs[0, :])  # Top row
ax2 = fig.add_subplot(gs[1:, 0]) # Left column
```

GridSpec enables flexible, asymmetric layouts

Inset Axes

Add zoom or detail views:

- **Zoom:** Magnify interesting region
- **Summary:** Add statistics or KPIs
- **Distribution:** Show marginal histogram

Example:

```
inset = ax.inset_axes([0.6, 0.6, 0.35, 0.35])
inset.plot(...)
```

Insets add context without separate panels

Twin Axes

Two scales on same plot:

- Price and volume
- Temperature and precipitation
- Primary and secondary metrics

Example:

```
ax2 = ax1.twinx()
ax1.plot(dates, prices, color='blue')
ax2.bar(dates, volume, color='gray', alpha=0.3)
```

Use twin axes sparingly – can be confusing

Layout Management

Prevent overlap:

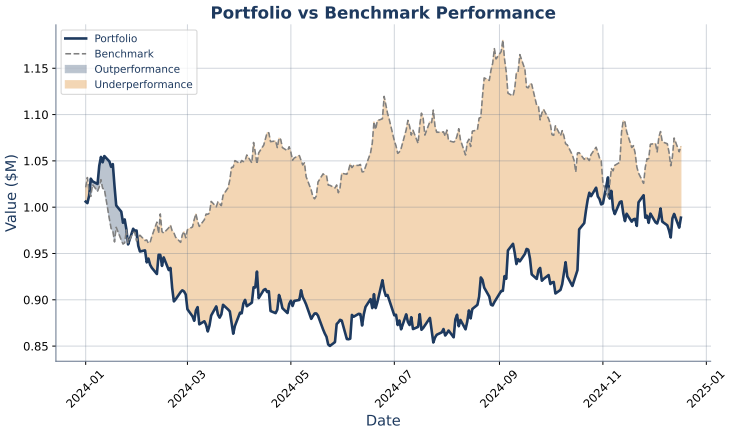
- **tight_layout():** Automatic spacing
- **constrained_layout:** More sophisticated
- **subplots_adjust():** Manual control

Best practice:

```
plt.tight_layout()
# Or at creation:
fig, axs = plt.subplots(..., constrained_layout=True)
```

Always check layout before saving

Dashboard Example



Combine multiple views for complete portfolio analysis

Design Principles

Effective multi-panel design:

- **Related:** Group logically connected views
- **Aligned:** Share axes where appropriate
- **Balanced:** Important data gets more space
- **Clear:** Label each panel clearly

Rule: Every panel should answer a specific question

Multi-panel figures tell a complete data story

Quick Reference

Essential Multi-Panel Commands:

Operation	Syntax
Grid subplots	`fig, axs = plt.subplots(2, 2)`
Shared x-axis	`plt.subplots(..., sharex=True)`
GridSpec	`gs = fig.add_gridspec(3, 3)`
Spanning	`fig.add_subplot(gs[0, :])`
Inset	`ax.inset_axes([x, y, w, h])`
Twin axis	`ax2 = ax1.twinx()`
Fix layout	`plt.tight_layout()`

Combine techniques for professional dashboards