Lesson 19: Multi-Panel Figures

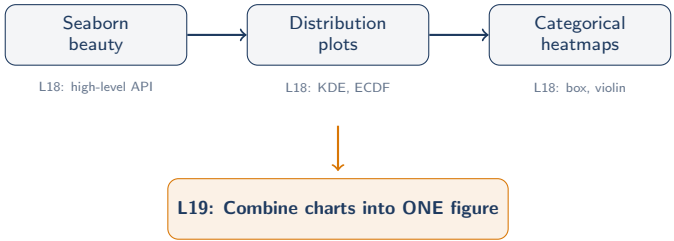
Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

Previously on Data Science...



L18 made: individual beautiful plots **L19 asks:** how do we assemble them into a *dashboard*?

A cockpit has many gauges – each shows one thing, together they fly the plane

Learning Objectives

After this lesson, you will be able to:

1. Explain subplot indexing and gridspec layouts
2. Create multi-panel figures with `plt.subplots()`
3. Apply shared axes for fair visual comparison
4. Build finance dashboards combining multiple chart types
5. Control figure sizing and aspect ratios for publication

Multi-panel figures are the bridge from exploration to communication

Why Multi-Panel Figures?

One chart shows one perspective. A dashboard shows the WHOLE picture.

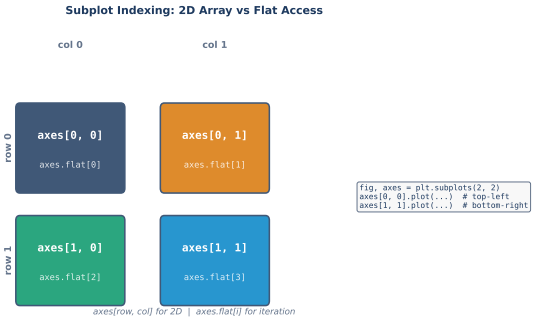
- A price chart alone misses volume context
- A return histogram alone misses timing
- A risk metric alone misses the story behind it

Think of it like a cockpit:

- Altimeter, speedometer, fuel gauge – each panel serves one purpose
- Together, the pilot sees everything at a glance

The goal: compose multiple views into one coherent visual narrative

Subplot Grid Indexing

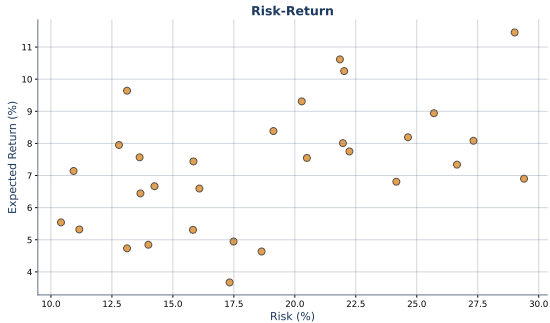


fig, axes = plt.subplots(rows, cols) – axes[row, col] selects each panel

Subplots: Price Line Chart

Line chart for time series – the first panel of a financial dashboard

Subplots: Risk-Return Scatter



Scatter plot compares risk vs return across assets – second panel

Shared Axes: Why They Matter

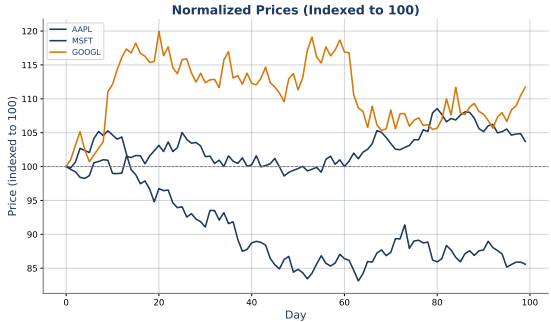
Without shared axes: eye jumps between scales, comparison impossible

With shared axes: visual alignment enables instant comparison

---

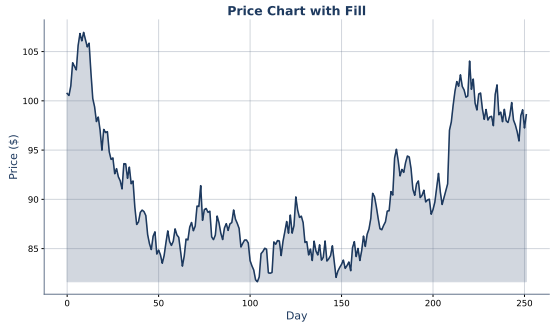
Use sharex=True / sharey=True when panels show the same variable

Normalized Comparison (Shared Y-Axis)



Normalize to 100 at start date for fair comparison across different price levels

Mixed Layouts: Unequal Panel Sizes

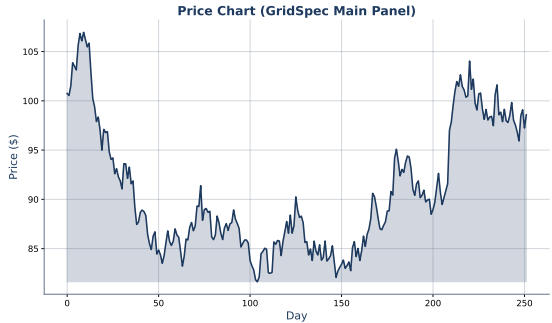


Key idea: not every panel deserves equal space

Main chart gets 2/3 height; supporting charts share the rest

Use gridspec_kw or GridSpec for unequal row/column ratios

GridSpec: Full Control Over Layout



GridSpec advantages: panels span rows/columns, independent sizing, mix chart types

---

fig.add_gridspec(nrows, ncols, height_ratios=[...]) for precise control

Checkpoint: Which Layout?

You need: price chart + volume bars + drawdown

Which layout do you choose?

A: 1×3 grid equal panels	B: GridSpec 3 rows, 1 col	C: Separate 3 figures

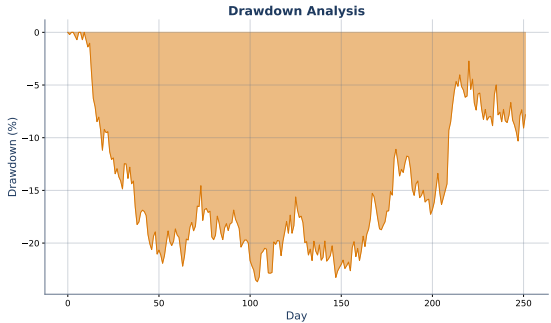
Think: do all panels share the same x-axis (time)?

Hint: price is the main story, volume and drawdown support it

---

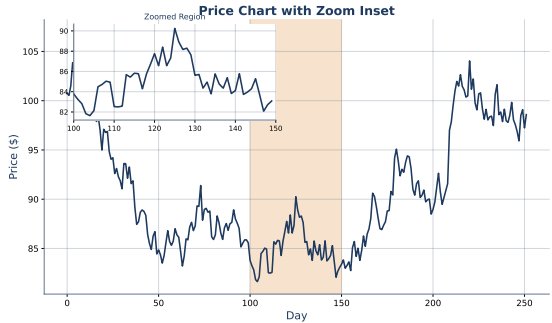
Answer: B – stacked rows with shared x-axis, price panel largest

GridSpec: Drawdown Panel



Drawdown = (price - peak) / peak – always negative, fill_between shows depth

Nested Plots: Zoom Insets



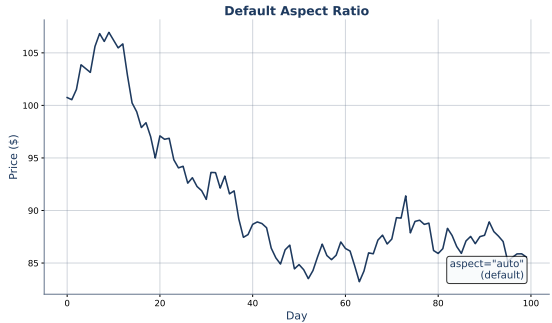
Use cases:

- Zoom into a crash period within a full time series
- Show distribution inset alongside main chart

---

inset_axes from mpl_toolkits.axes_grid1 creates plot-within-a-plot

Figure Sizing for Publication

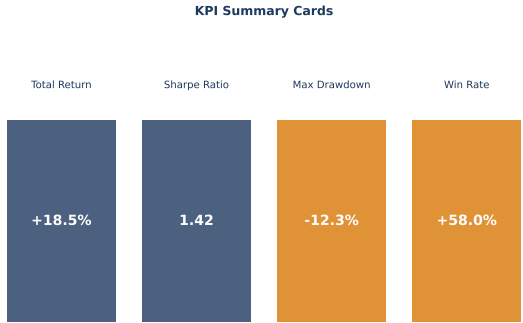


Rules: Presentation: (10,6) Paper: (8,5) Dashboard: (14,10)

---

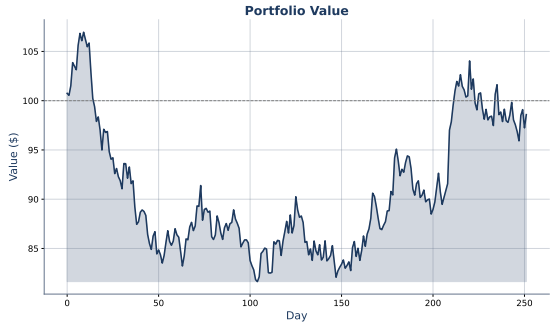
figsize=(width, height) in inches – set BEFORE adding subplots

Dashboard Element: KPI Cards



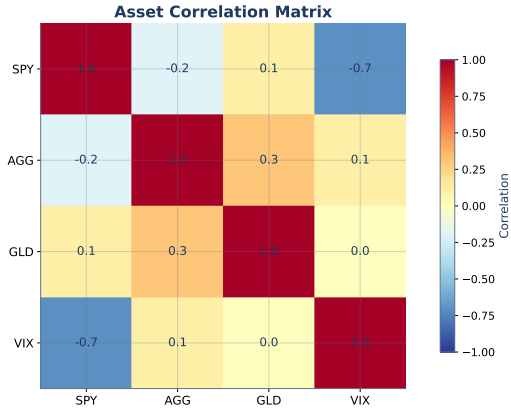
Text-based KPI metrics at the top of a dashboard – the headline numbers

Dashboard Element: Portfolio Value



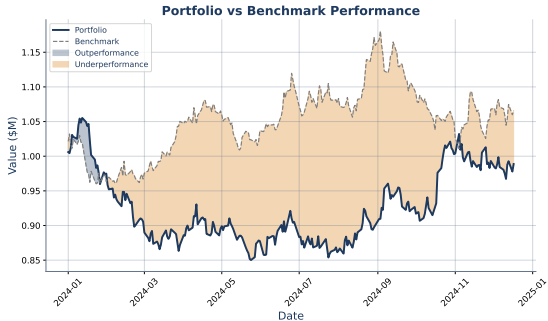
Cumulative portfolio value over time – the main chart of any finance dashboard

Dashboard Element: Correlation Heatmap



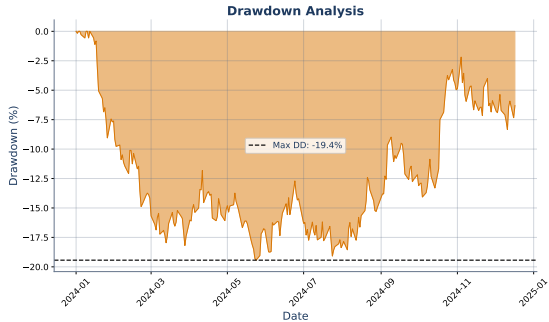
Asset correlation structure – use annot=True and a diverging colormap

Finance Dashboard: Portfolio vs Benchmark



Compare portfolio to benchmark index – the first question every investor asks

Finance Dashboard: Drawdown Analysis



Maximum drawdown and recovery periods – essential risk monitoring

Hands-On: Build a 3-Panel Dashboard

Task: Create a finance dashboard with three stacked panels.

Panel 1 (top, 50% height): Cumulative returns – your portfolio vs S&P 500

Panel 2 (middle, 25%): Daily returns bar chart (green/red)

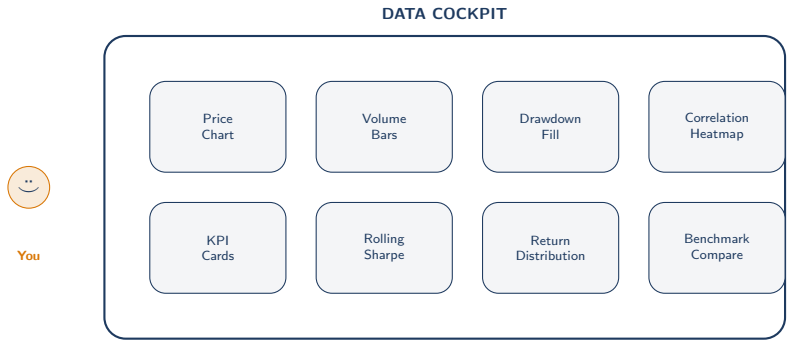
Panel 3 (bottom, 25%): Rolling 60-day volatility

Requirements:

- Shared x-axis across all three panels
- Consistent color scheme (navy for portfolio, amber for benchmark)
- `fig.suptitle()` with your portfolio name

Use GridSpec with height_ratios=[2, 1, 1] for the 50/25/25 split

Your Data, One Glance



Every gauge answers one question – together, they tell the whole story

Key Takeaways

What you learned today:

1. **Subplots:** `plt.subplots(rows, cols)` creates a grid of axes
2. **Shared axes:** `sharex/sharey` enable visual comparison
3. **GridSpec:** unequal panel sizes for main + supporting views
4. **Insets:** zoom or overlay details within a chart
5. **Dashboards:** KPI cards + charts + heatmaps = complete picture

Rule of thumb: 4–6 panels is ideal; more than 7 creates clutter

---

A dashboard is a visual argument – every panel should earn its space

Next: Data Storytelling (L20)

You can now build multi-panel figures.
But can you make them persuade?

L20 will cover:

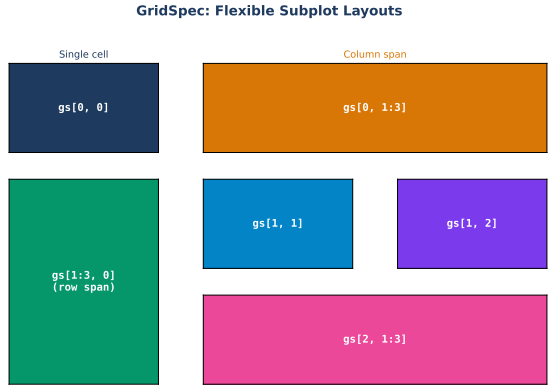
- Narrative arc – structuring a visual story
- Chart selection – picking the right chart for the message
- Color strategy – guiding attention with color
- Before/after transformations – turning bad charts into good ones
- Executive summaries – presenting to stakeholders

Module 3 finale: from raw numbers to compelling visual narratives

---

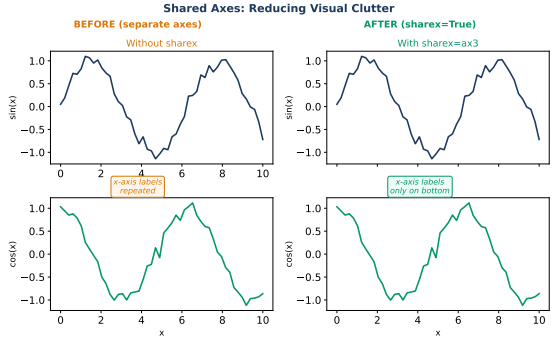
L20 is the capstone of the Statistics & Visualization module

Self-Study: GridSpec Anatomy



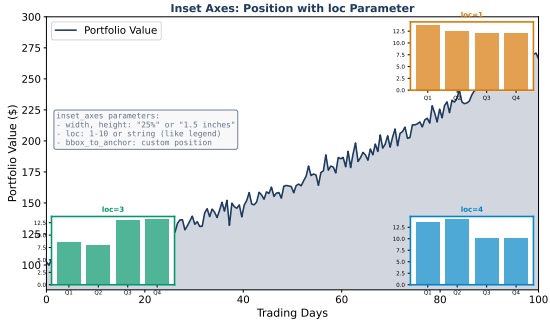
GridSpec divides the figure into a grid – subplots can span cells

Self-Study: Shared Axes Concept



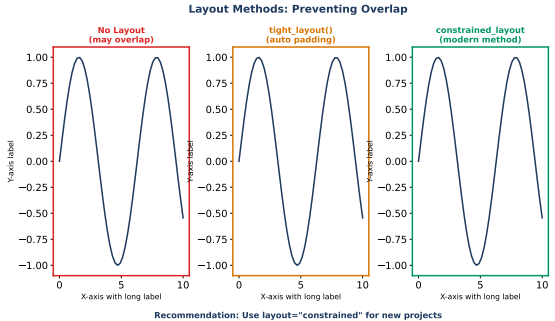
Shared axes align scales across panels for direct comparison

Self-Study: Inset Positioning



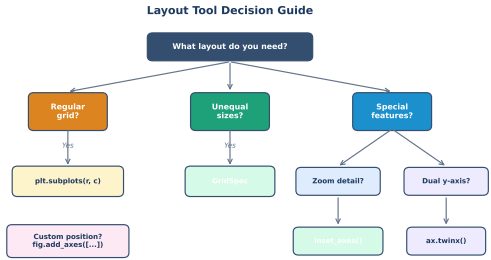
Position inset axes using [x, y, width, height] in figure coordinates

Self-Study: Layout Spacing



plt.subplots_adjust() and constrained_layout control spacing

Self-Study: Layout Decision Guide

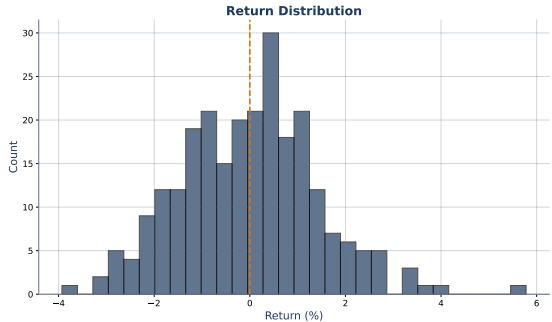


Simple grids: subplots() | Complex layouts: GridSpec | Special: inset_axes, twinx

---

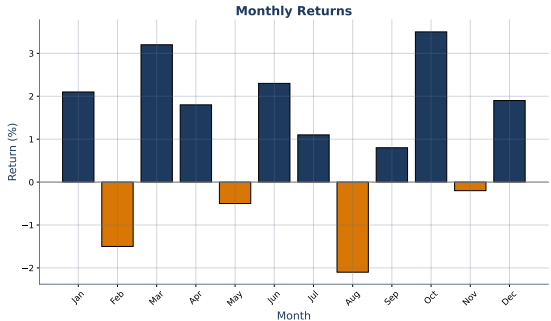
Choose subplots for equal grids, GridSpec for unequal, insets for overlays

Self-Study: Return Histogram



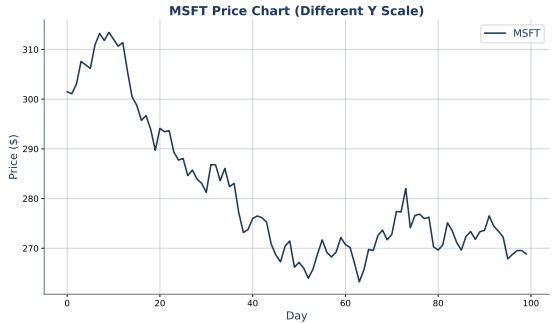
Histogram shows return distribution – pair with price chart for context

Self-Study: Monthly Bar Chart



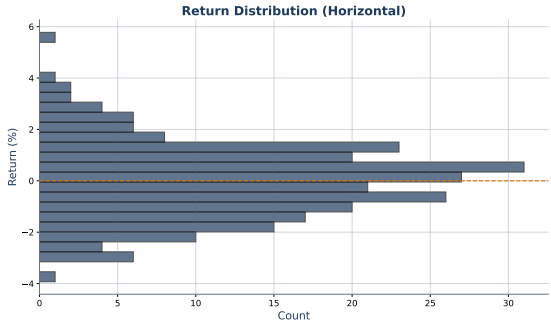
Bar chart for discrete time periods – green/red for gain/loss

Self-Study: Different Scale Prices



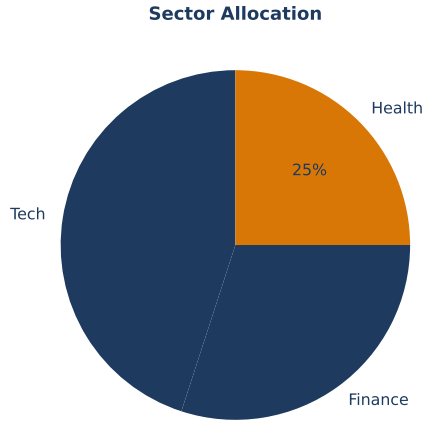
Multiple assets at different price levels need normalization or dual axes

Self-Study: Return Distribution Horizontal



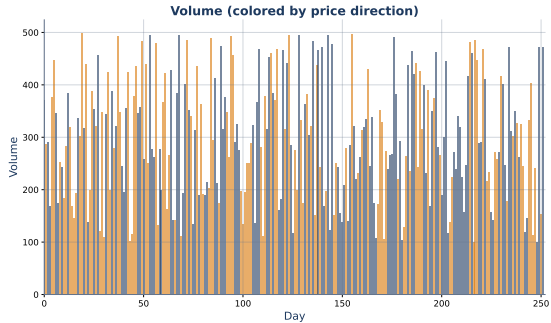
Horizontal histogram pairs well as a side panel to a time series chart

Self-Study: Sector Allocation Pie



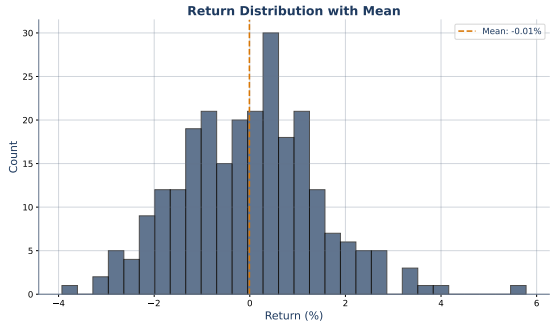
Pie chart shows portfolio composition – use reduced width for square charts

Self-Study: Volume Bar



Volume bars typically sit below a price chart with shared x-axis

Self-Study: Return Histogram with Mean

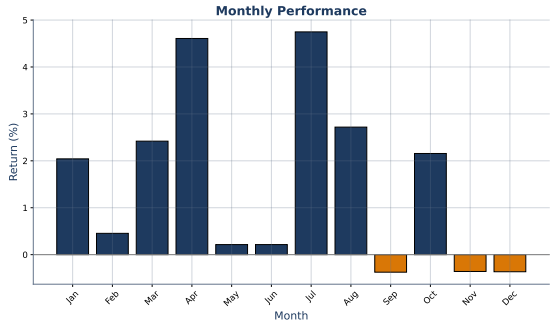


Histogram with mean line annotation highlights central tendency

Self-Study: Rolling Volatility

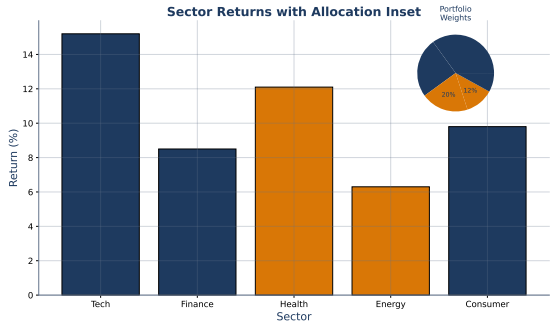
Rolling standard deviation shows time-varying risk

Self-Study: Monthly Returns Bar



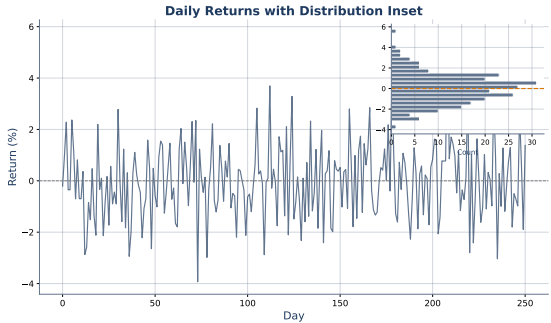
Monthly bar chart with positive/negative colors – a dashboard staple

Self-Study: Bar with Pie Inset



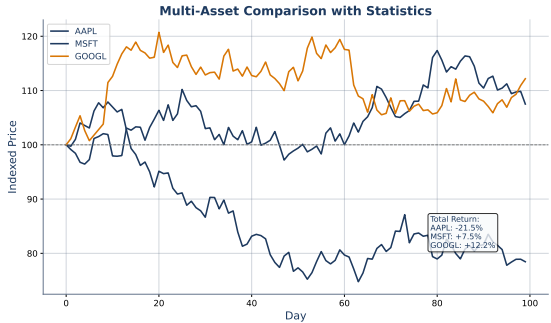
Combine chart types with inset axes for compact visual storytelling

Self-Study: Returns with Histogram Inset



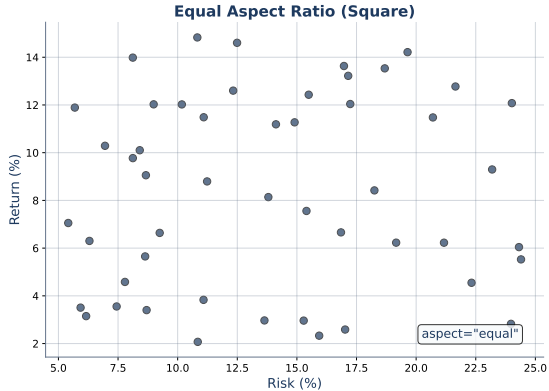
Distribution inset alongside time series shows both timing and shape

Self-Study: Multi-Asset with Stats Inset



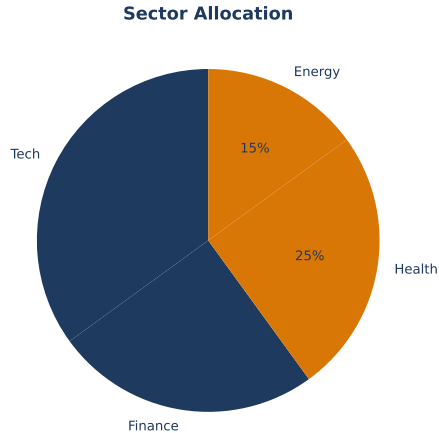
Statistics table as inset element – useful for summary panels

Self-Study: Equal Aspect Scatter



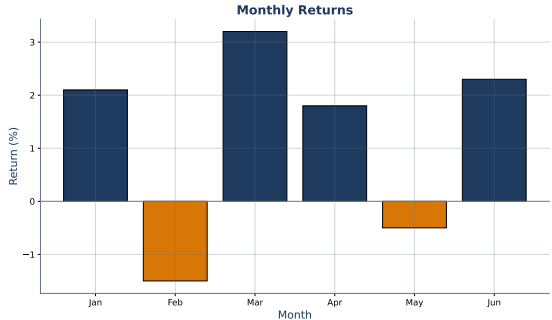
ax.set_aspect('equal') ensures 1:1 ratio for risk-return plots

Self-Study: Allocation Pie



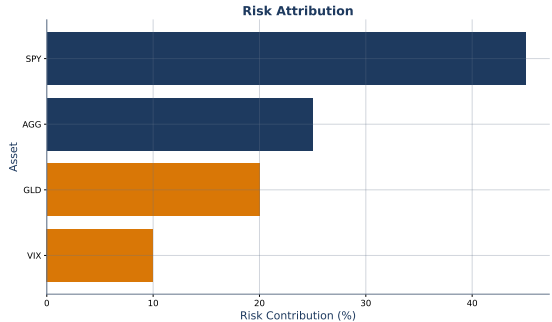
Current portfolio allocation breakdown – a dashboard essential

Self-Study: Monthly Returns Bar (Dashboard)



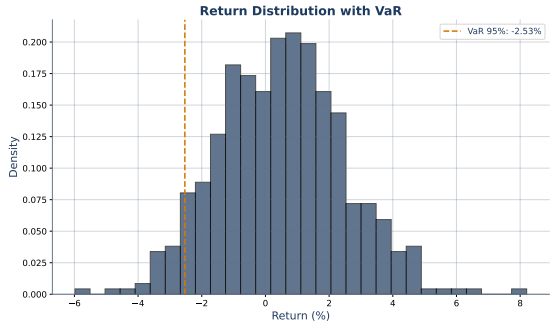
Monthly performance bar chart for dashboard panels

Self-Study: Risk Attribution



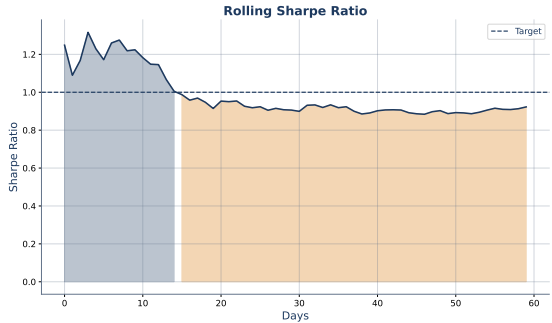
Risk contribution by asset or factor – horizontal bar works well

Self-Study: Return Distribution with VaR



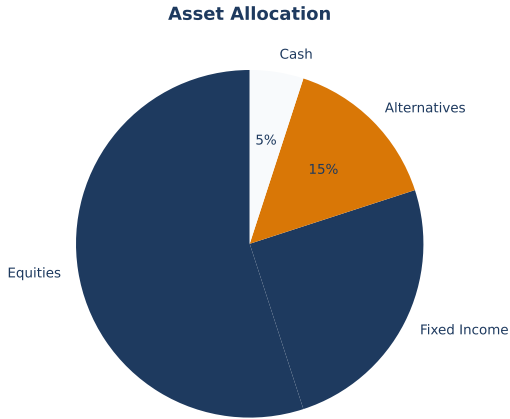
Distribution with VaR threshold marked – key risk metric visualization

Self-Study: Rolling Sharpe Ratio



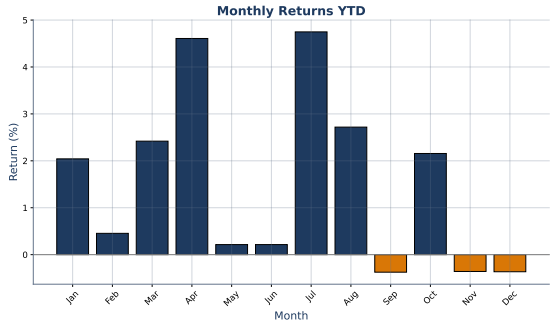
Time-varying risk-adjusted performance – is your strategy improving?

Self-Study: Asset Allocation



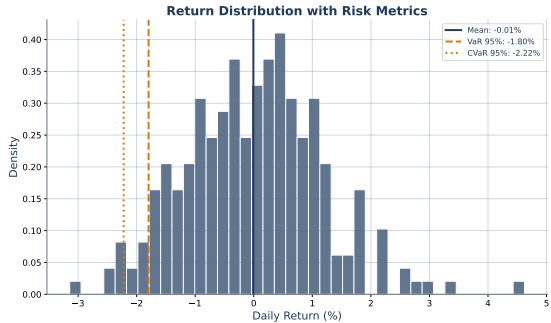
Current weights across asset classes in a finance dashboard

Self-Study: Monthly Returns YTD



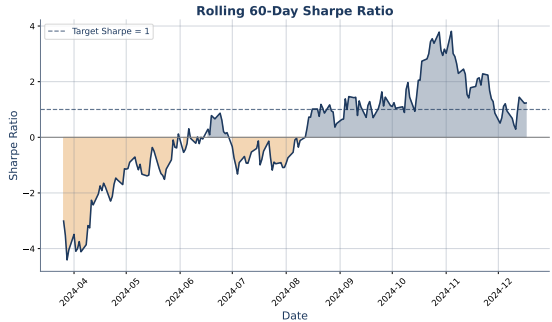
Year-to-date monthly performance – track calendar year progress

Self-Study: Return Distribution Full



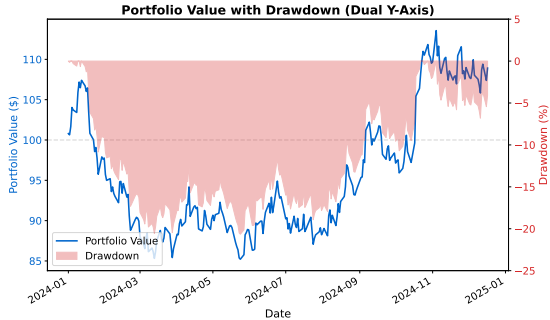
Full return distribution with statistics overlay

Self-Study: Rolling Sharpe 60-Day



60-day rolling Sharpe ratio for short-term performance monitoring

Self-Study: Dual Axis (twinx)



ax.twinx() overlays two y-scales on one x-axis – use sparingly

Self-Study: Shared Legend

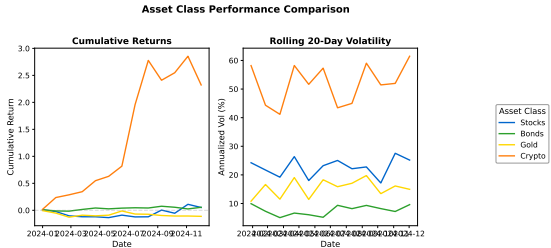
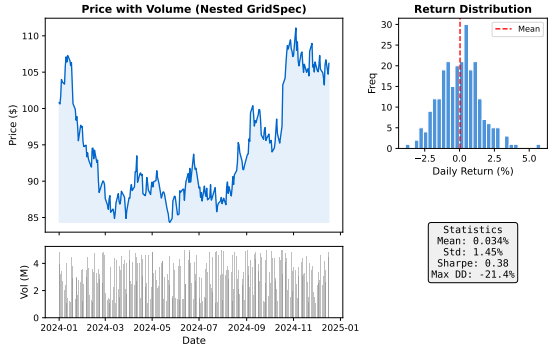


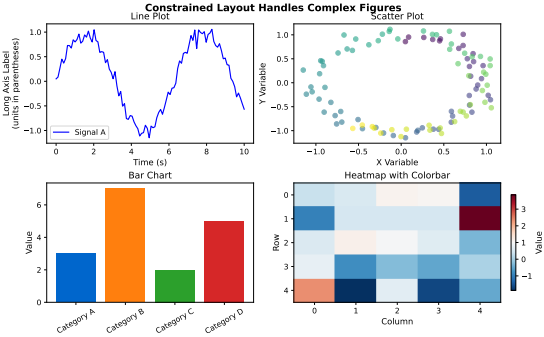
fig.legend() places one legend for multiple subplots

Self-Study: Nested GridSpec



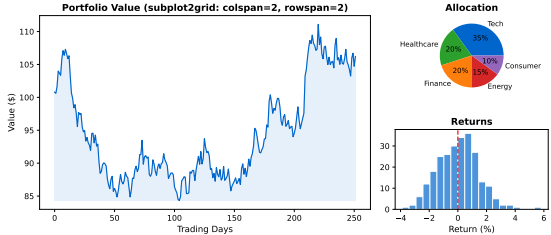
GridSpec within GridSpec for complex hierarchical layouts

Self-Study: Constrained vs Tight Layout



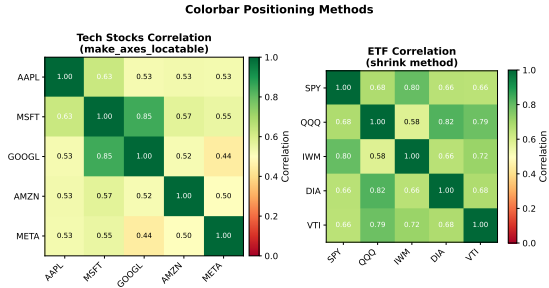
constrained_layout=True is more robust than tight_layout()

Self-Study: subplot2grid



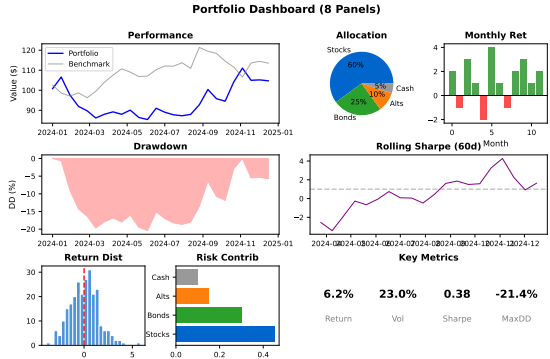
plt.subplot2grid() is an older API – prefer GridSpec for new code

Self-Study: Colorbar Layout



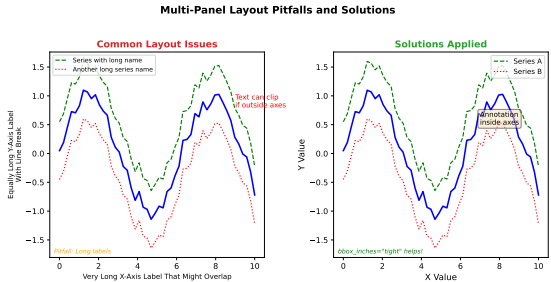
Place colorbars carefully to avoid distorting subplot alignment

Self-Study: Assembled Dashboard



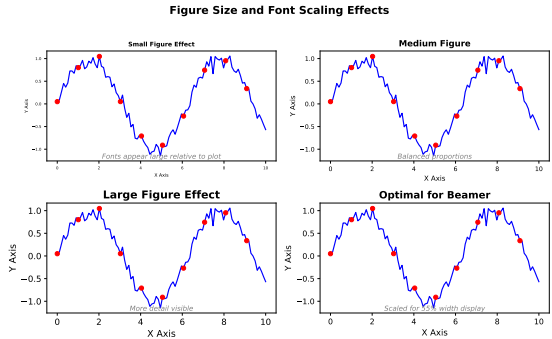
Complete assembled dashboard combining all techniques from this lesson

Self-Study: Common Pitfalls



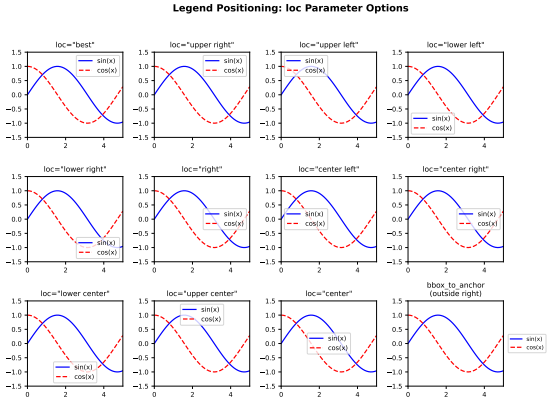
Overlapping labels, misaligned axes, inconsistent colors – avoid these

Self-Study: Figure Size Effects



Changing figsize affects font relative size – test at target display size

Self-Study: Legend Positioning



bbox_to_anchor places legend outside the axes to avoid overlap