Lesson 18: Seaborn Statistical Plots

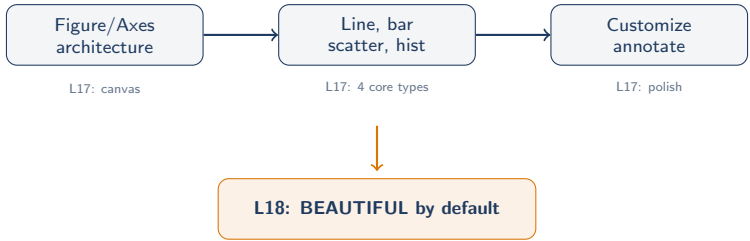
Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

Previously on Data Science...



L17 taught you: how to build any chart from scratch. **L18 asks:** "What if charts came with *statistical intelligence* built in?"

Seaborn builds on matplotlib – everything you learned in L17 still applies

Learning Objectives

After this lesson, you will be able to:

1. Explain when to use seaborn vs raw matplotlib
2. Create distribution plots (histogram, KDE, violin, box)
3. Build regression plots with automatic confidence bands
4. Create and interpret correlation heatmaps
5. Apply seaborn themes to financial visualizations

Seaborn = matplotlib + statistical smarts + beautiful defaults

Matplotlib Is Powerful but Ugly by Default

Matplotlib gives you total control. Seaborn gives you good taste.

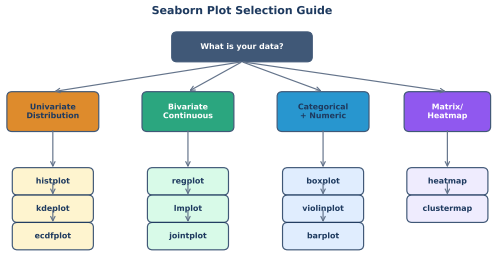
Think of it as Instagram filters for data:

- Same photo (data), but presented *better* with less effort
- Seaborn adds **statistical intelligence**: KDE, CI bands, regression
- Works directly with DataFrames – no manual data wrangling

When to use which:
Seaborn for statistical plots. Matplotlib for full customization.
In practice: use both together.

Seaborn is built on matplotlib – use sns for the chart, plt for fine-tuning

Seaborn Plot Types: The Map

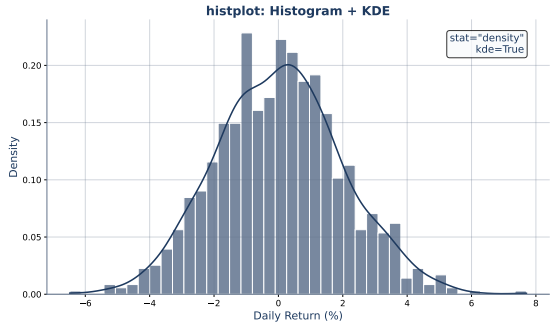


Choose based on: data type, variables, and question to answer

- **Distribution**: histplot, kdeplot, boxplot, violinplot
- **Relational**: scatterplot, lineplot, regplot
- **Matrix**: heatmap, clustermap

Know the map before you pick the plot

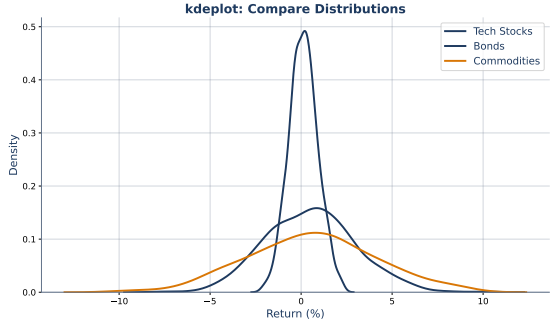
Histogram + KDE: Distribution at a Glance




```
sns.histplot(data, kde=True, stat='density')
```

One line of code gives you histogram + smooth density curve

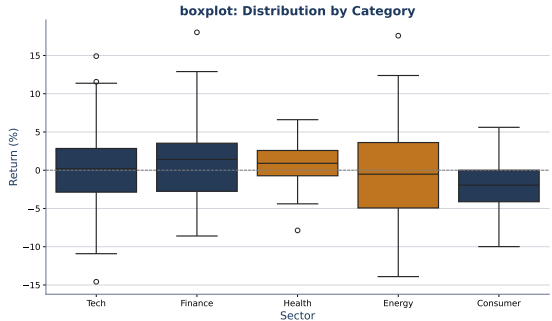
KDE: Comparing Distributions




```
sns.kdeplot(data=df, x='returns', hue='asset')
```

KDE smooths the histogram – easier to compare multiple groups

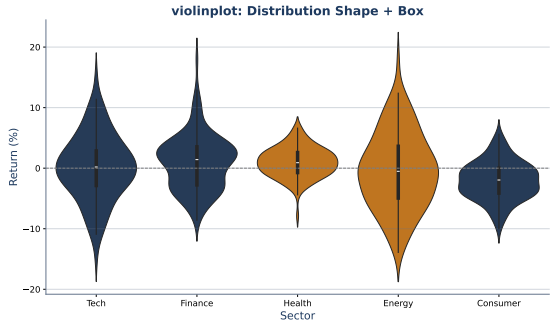
Boxplot: The Five-Number Summary



- Box: Q1 to Q3 (middle 50%)
- Line inside: median
- Whiskers: 1.5×IQR; dots beyond: outliers

Boxplots compress an entire distribution into one compact shape

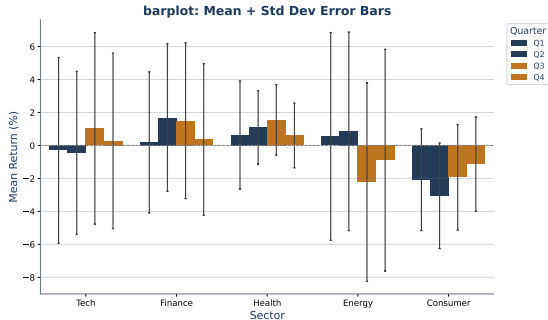
Violin Plot: Boxplot + KDE



Shows the *shape* of the distribution, not just summary statistics.

Use violin when the distribution shape matters – bimodal, skewed, or heavy-tailed

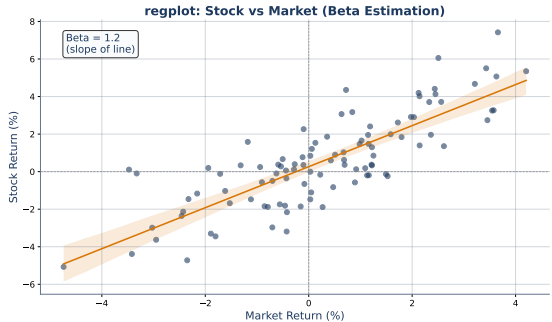
Bar Plot with Confidence Intervals



`sns.barplot(data=df, x='sector', y='return', ci=95)`
Seaborn automatically computes mean and confidence interval.

Error bars tell you whether differences are real or just noise

Regplot: Scatter + Regression in One



`sns.regplot(data=df, x='risk', y='return')`
Fits OLS line with 95% confidence band automatically.

The shaded band shows uncertainty – wider means less confident

Checkpoint: Boxplot or Violin?

Quick Check

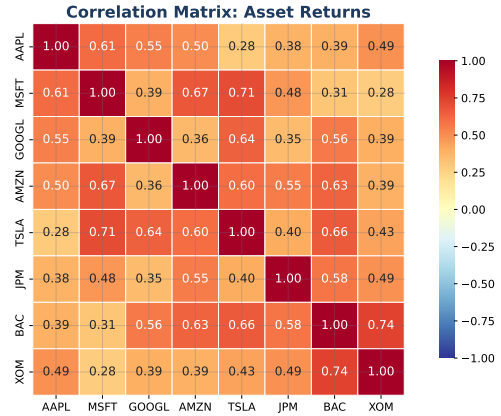
When does a violin plot beat a boxplot?

A) When you have few categories
B) When the distribution might be bimodal
C) When you need exact quartile values
D) When your data is normally distributed

Answer: B – violins reveal shape; boxplots hide bimodality

Boxplots summarize; violins reveal – choose based on what matters

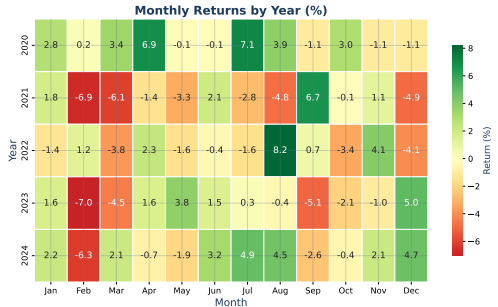
Heatmap: Correlation Matrix




```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

Heatmaps turn a matrix of numbers into a visual pattern you can scan in seconds

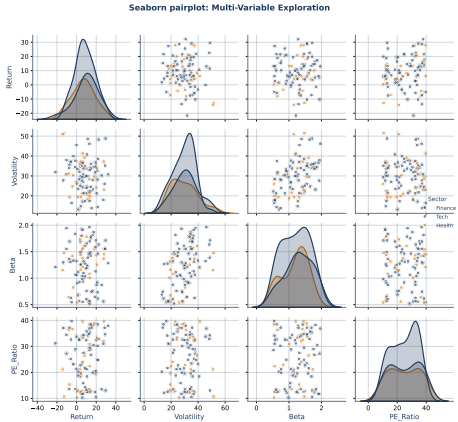
Heatmap: Monthly Returns Calendar



Pivot table + heatmap reveals seasonal patterns in returns.

Diverging colormaps make gains and losses instantly visible

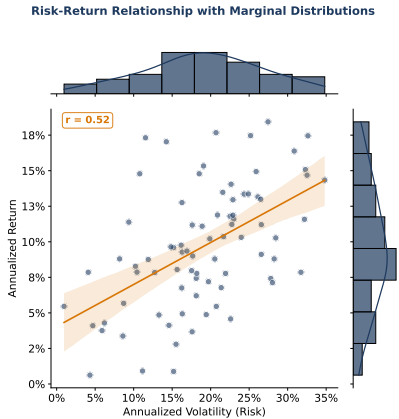
Pairplot: All Relationships at Once



`sns.pairplot(df, hue='category')` – one line, n^2 plots.

Pairplot is the fastest way to explore a new dataset

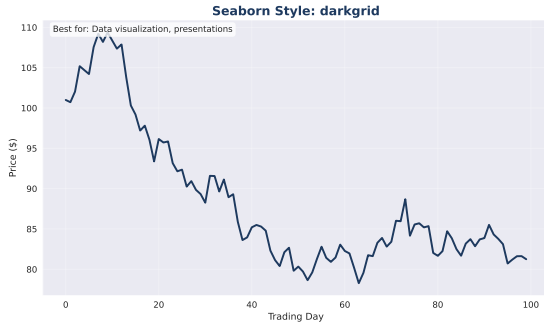
Jointplot: Marginal Distributions



Scatter + marginal histograms show relationship and distributions together.

Jointplot shows the bivariate relationship AND each variable's distribution

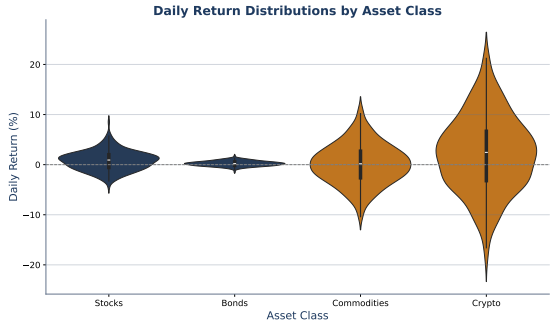
Themes: Change the Mood



`sns.set_style("darkgrid")` – or whitegrid, white, ticks.
`sns.set_context("talk")` – scale fonts for presentations.

Style sets the background; context sets the font scale

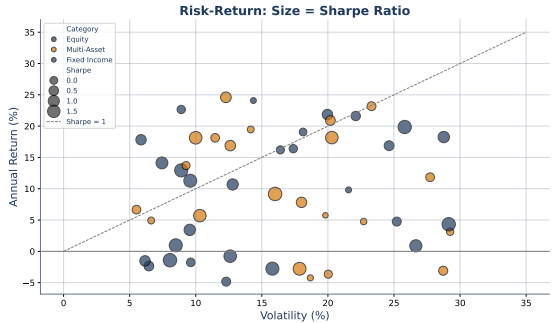
Finance: Asset Return Distributions



Compare return distributions across asset classes with one `kdeplot` call.

Fat tails in finance: real asset returns are not normally distributed

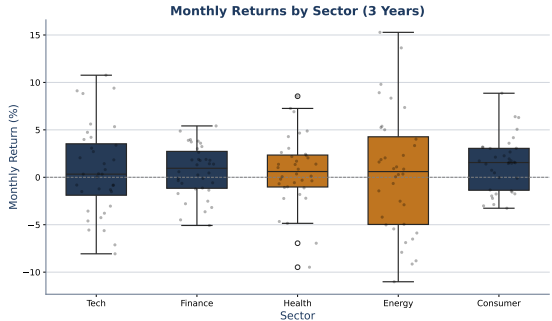
Finance: Risk-Return Scatter



`sns.regplot()` fits the risk-return relationship with confidence band.

Higher risk should mean higher return – regplot shows if the data agrees

Finance: Sector Return Comparison



One boxplot per sector reveals median, spread, and outliers.

Boxplots let portfolio managers compare sector risk profiles at a glance

Hands-On: Seaborn Financial Dashboard

Task: Recreate L17's dashboard with seaborn in half the code.

1. Create a DataFrame with daily returns for 3 assets
2. **Plot 1:** `sns.kdeplot()` comparing return distributions
3. **Plot 2:** `sns.boxplot()` comparing assets
4. **Plot 3:** `sns.heatmap()` of correlation matrix
5. **Plot 4:** `sns.regplot()` risk vs return

Compare: How many lines of code vs your L17 dashboard?

Hands-on: 10 minutes – seaborn should cut your code by 50%

Same Data, Better Story

"Seaborn doesn't replace matplotlib. It makes matplotlib dress up for the occasion."

Use seaborn for statistical charts, matplotlib for everything else – together they cover it all

Key Takeaways

What you now know:

1. **Seaborn vs matplotlib**: seaborn for statistics, matplotlib for full control
2. **Distribution plots**: histplot, kdeplot, boxplot, violinplot reveal data shape
3. **Regression plots**: regplot adds OLS line + confidence band automatically
4. **Heatmaps**: correlation matrices and time patterns in one glance
5. **Themes**: set_style() and set_context() for professional, presentation-ready output

---

You now have two complementary tools – matplotlib for control, seaborn for speed

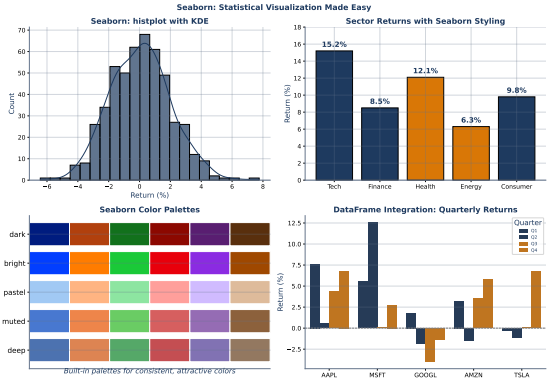
Coming Up: L19 – Multi-Panel Figures

One chart tells a fact. Multiple charts tell a story.

- Arrange subplots into coherent dashboards
- Shared axes, consistent styling, figure-level layout
- Professional multi-panel figures for reports and papers

L19 teaches you to compose individual charts into publication-quality figures

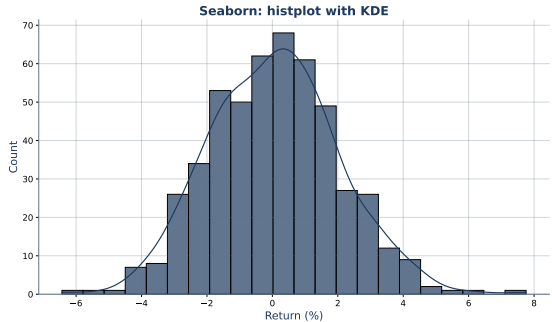
Self-Study: Seaborn Quick Introduction



- `import seaborn as sns` – convention
- Built on matplotlib, integrates with pandas DataFrames

Seaborn wraps matplotlib to make statistical plots easier

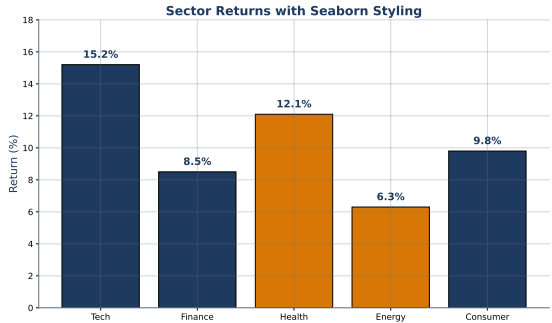
Self-Study: Histplot with KDE Overlay



- `sns.histplot(data, kde=True)` adds smooth density overlay
- Combines discrete bins with continuous estimate

KDE overlay helps see the underlying distribution shape

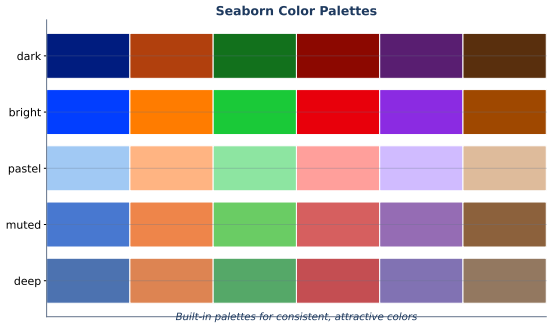
Self-Study: Sector Bar Chart



- `sns.barplot()` computes mean and adds error bars automatically
- Pass `hue=` for grouped comparisons

Seaborn barplot is statistical – it aggregates data for you

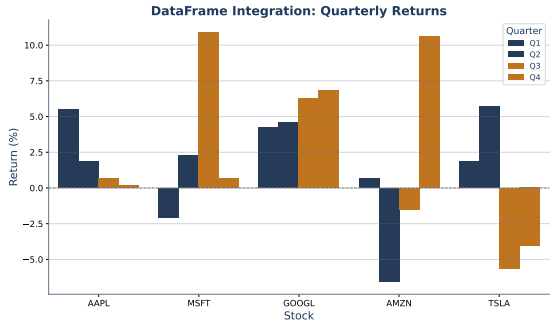
Self-Study: Color Palettes



- `sns.color_palette("Set2")` – qualitative for categories
- `sns.color_palette("coolwarm")` – diverging for +/-
- `sns.color_palette("Blues")` – sequential for magnitude

Match palette type to data type: categorical, diverging, or sequential

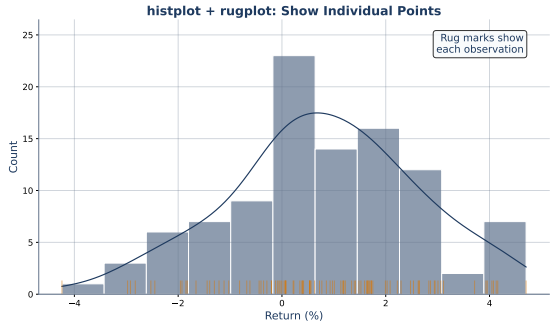
Self-Study: DataFrame Integration



- Pass `data=df, x='col_name', hue='group'` directly
- No manual array extraction needed

Seaborn speaks pandas natively – pass column names, not arrays

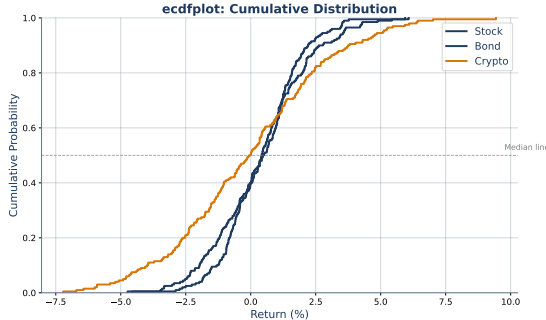
Self-Study: Histogram with Rugplot



- Rugplot shows individual observations as tick marks along the axis
- Useful for small samples where each data point matters

Rugplot reveals clustering and gaps that bins might hide

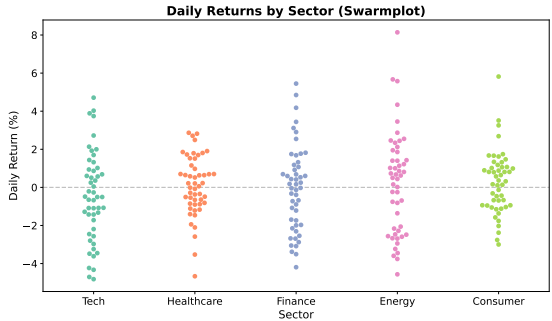
Self-Study: ECDF Plot



- Empirical CDF: no bins, no smoothing, just cumulative probability
- Read off percentiles directly from the y-axis

ECDF avoids all binning decisions – the most honest distribution plot

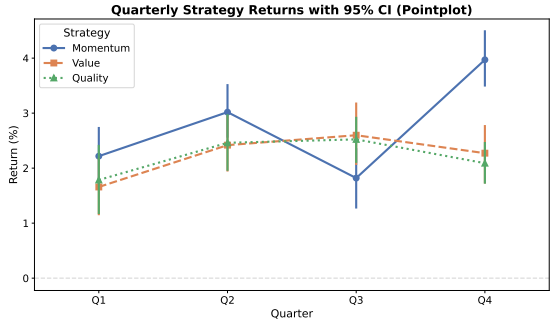
Self-Study: Swarm Plot



- Every point visible, no overlap
- Best for small-to-medium datasets (under 500 points)

Swarm plots show every data point – the most detailed categorical view

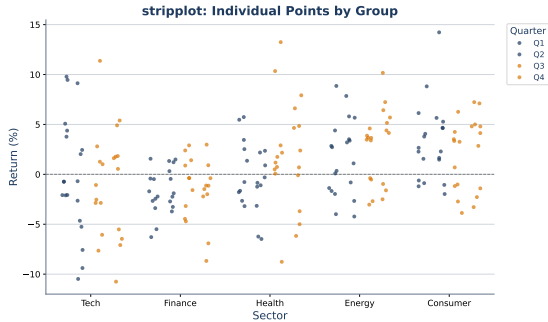
Self-Study: Point Plot



- Shows mean + CI as point and line
- Good for showing trends across ordered categories

Point plots emphasize the central tendency and its uncertainty

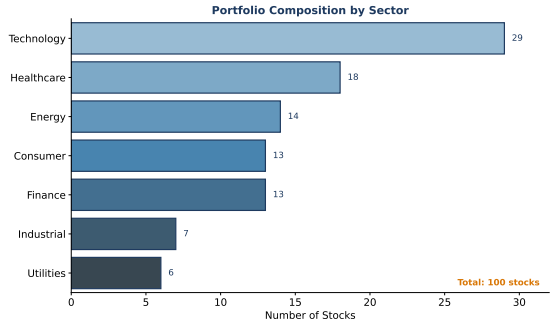
Self-Study: Strip Plot



- Similar to swarm but with random jitter instead of algorithm
- Faster for large datasets, may have overlap

Strip plots are the quick-and-dirty version of swarm plots

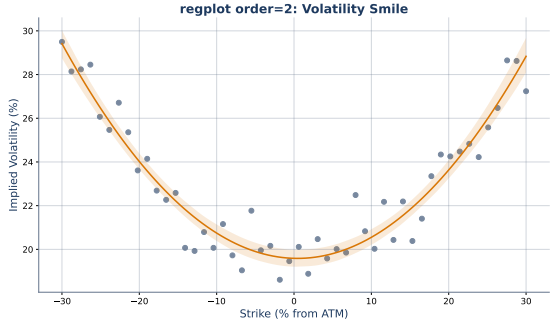
Self-Study: Count Plot



- `sns.countplot(data=df, x='category')` – counts observations per category
- Essentially a histogram for categorical data

Countplot is the categorical equivalent of a histogram

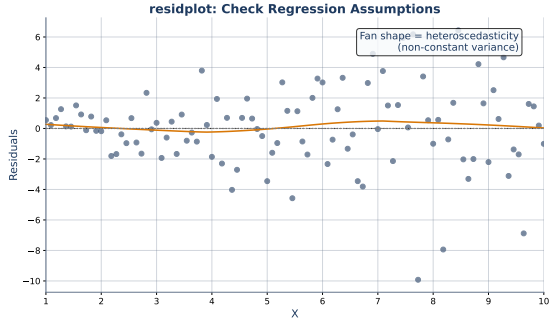
Self-Study: Polynomial Regression



- `sns.regplot(order=2)` fits a polynomial curve
- Use when the relationship is clearly non-linear

Higher-order fits capture curvature but risk overfitting

Self-Study: Residual Plot

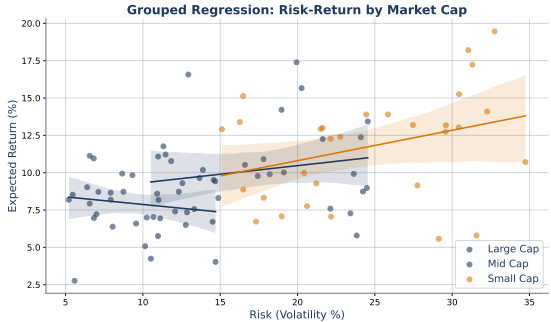


- `sns.residplot()` shows residuals from the regression fit
- Look for patterns: random = good fit, curved = model needs improvement

---

Residual plots diagnose whether your regression model is appropriate

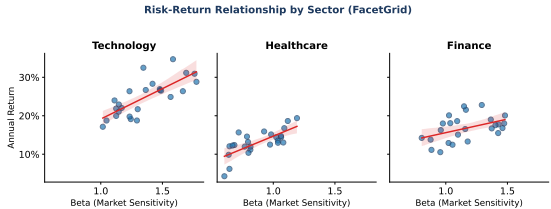
Self-Study: Grouped Regression



- `hue='group'` fits separate regression lines per group
- Reveals whether the relationship differs across categories

Grouped regression detects Simpson's paradox before it bites you

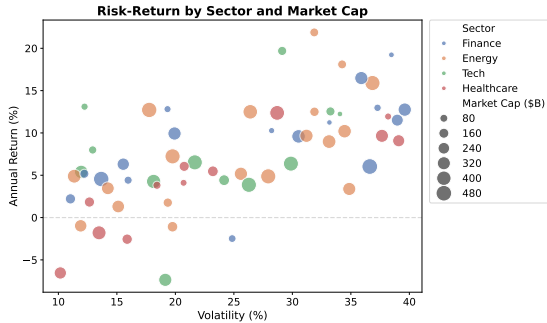
Self-Study: Faceted Regression (lmplot)



- `sns.lmplot(col='sector')` creates separate panels per group
- Figure-level function with built-in faceting

Implot combines regression with faceting for multi-group analysis

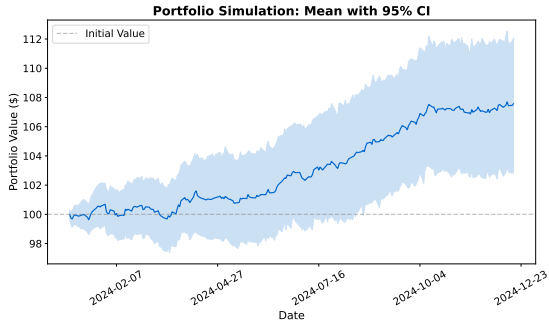
Self-Study: Scatterplot with Size and Style



- `sns.scatterplot(size='mcap', hue='sector', style='type')`
- Encode up to 4 variables in one scatter plot

Seaborn scatterplot handles multi-dimensional encoding automatically

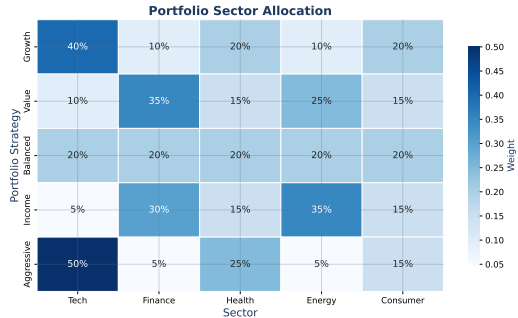
Self-Study: Lineplot for Time Series



- `sns.lineplot()` with confidence band from multiple observations
- Automatically aggregates and shows uncertainty

Seaborn lineplot adds confidence bands that matplotlib line plots lack

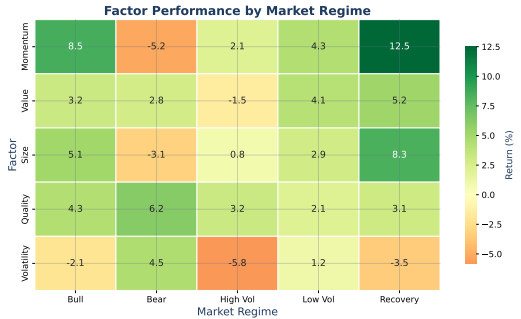
Self-Study: Sector Allocation Heatmap



- Heatmap of portfolio weights by sector and time period
- Track allocation drift visually

Portfolio managers use allocation heatmaps to monitor drift

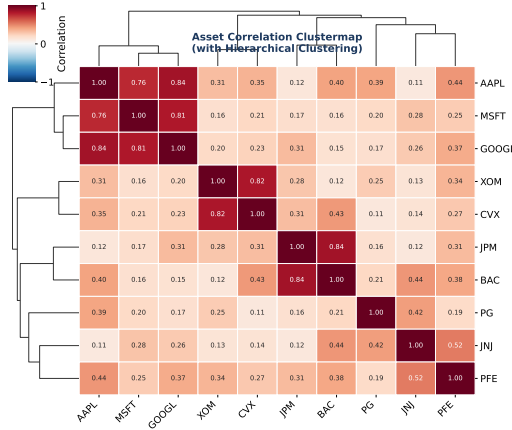
Self-Study: Factor Performance Heatmap



- Factor exposure across time or across assets
- Diverging colormap highlights over/under-weight

Factor heatmaps are standard in quantitative portfolio analysis

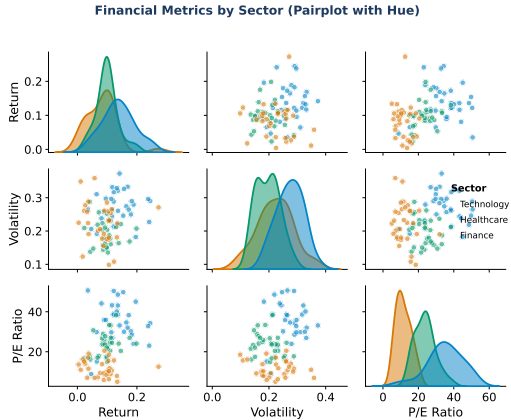
Self-Study: Clustermap



- `sns.clustermap()` reorders rows/columns by similarity
- Dendrograms show hierarchical clustering structure

Clustermap reveals hidden groupings in correlation or feature matrices

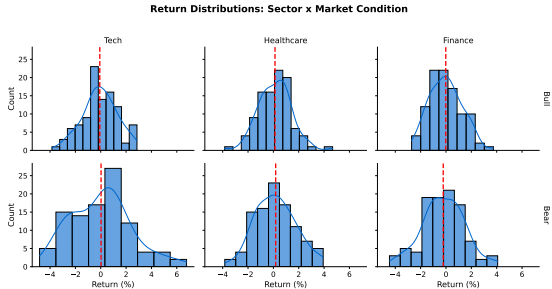
Self-Study: Pairplot with Hue



- `hue='group'` colors points by category across all panels
- Reveals whether groups cluster differently across variables

Hue in pairplot turns exploration into pattern discovery

Self-Study: FacetGrid Custom



- `g = sns.FacetGrid(df, col='group')` then `g.map(sns.histplot, 'x')`
- Full control over panel layout and mapping

FacetGrid is the engine behind lmplot, catplot, and relplot

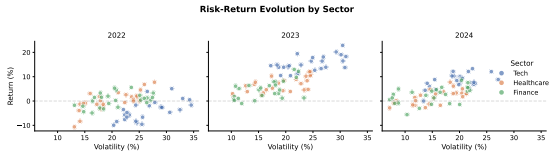
Self-Study: Catplot (Faceted Categorical)

- `sns.catplot(kind='box', col='year')` – faceted categorical plots
- Combines any categorical plot type with faceting

---

catplot is the figure-level version of boxplot, violinplot, etc.

Self-Study: Relplot (Faceted Relational)

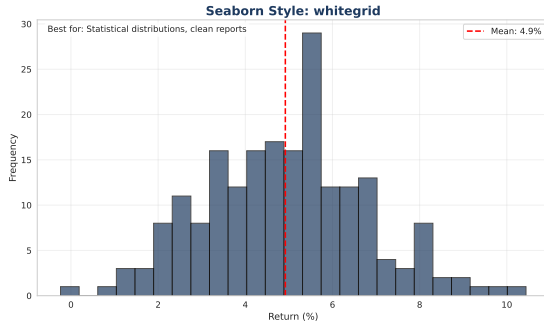


- `sns.relplot(kind='scatter', col='sector')` – faceted scatter/line
- Figure-level function with built-in faceting

---

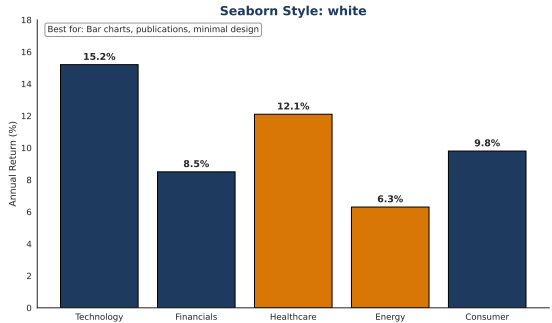
relplot combines scatterplot or lineplot with multi-panel layout

Self-Study: Whitegrid Style



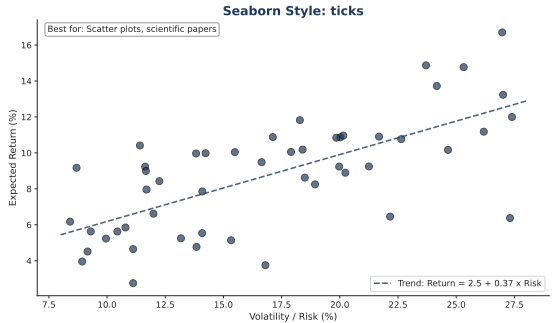
Whitegrid: clean background, grid lines help read values

Self-Study: White Style



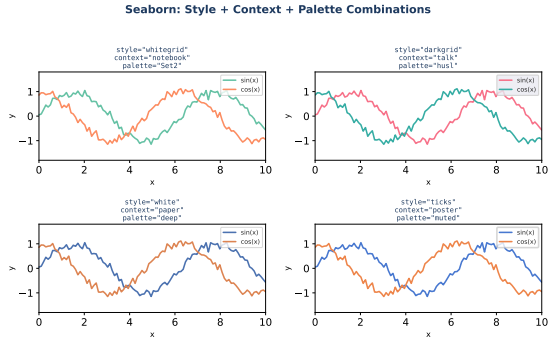
White: minimal – best for bar charts and publications

Self-Study: Ticks Style



Ticks: scientific look – combine with sns.despine() for clean axes

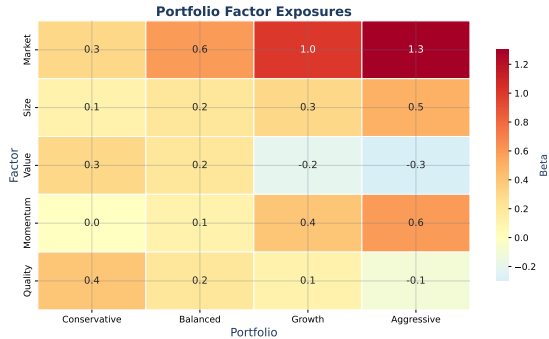
Self-Study: Style + Context + Palette



- `set_style()` = background and grid
- `set_context()` = font scale (paper, notebook, talk, poster)
- `set_palette()` = color scheme

Three knobs control seaborn's entire visual appearance

Self-Study: Factor Exposures



- Visualize factor loadings (market, size, value) for portfolio analysis
- Heatmap or grouped bar chart depending on dimensionality

Factor exposure analysis is the backbone of quantitative investing