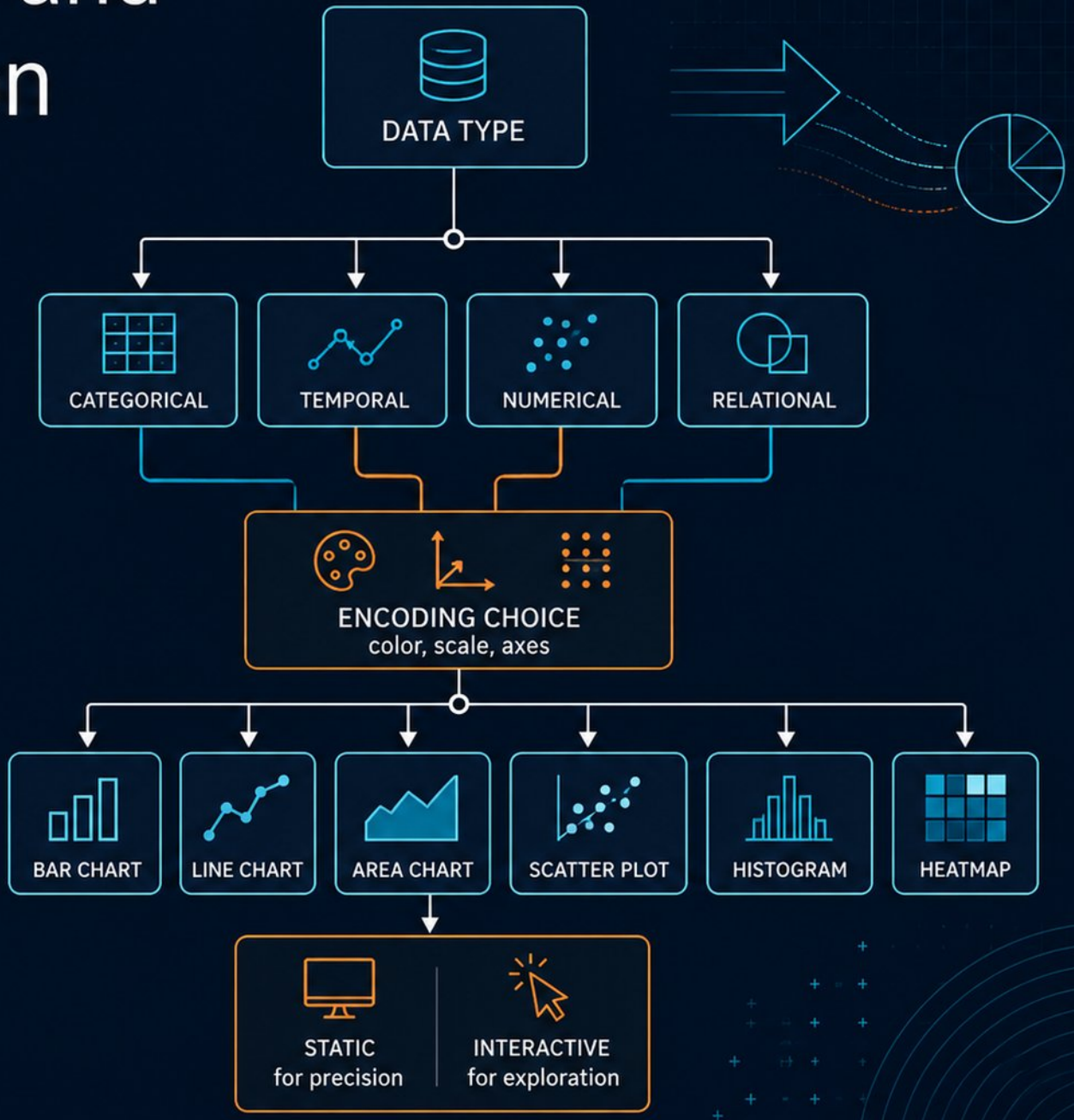
# Data Visualization Tools in Python

- Core skill for analysts and researchers to explore and communicate data
- Mature ecosystem: Matplotlib 3.10, Seaborn 0.13, Plotly 6, Altair
- Covers Matplotlib, Seaborn, Pandas, Plotly, Dash, Altair, and specialized tools
- Hands-on workflow: from rapid exploration to polished, publication-ready output

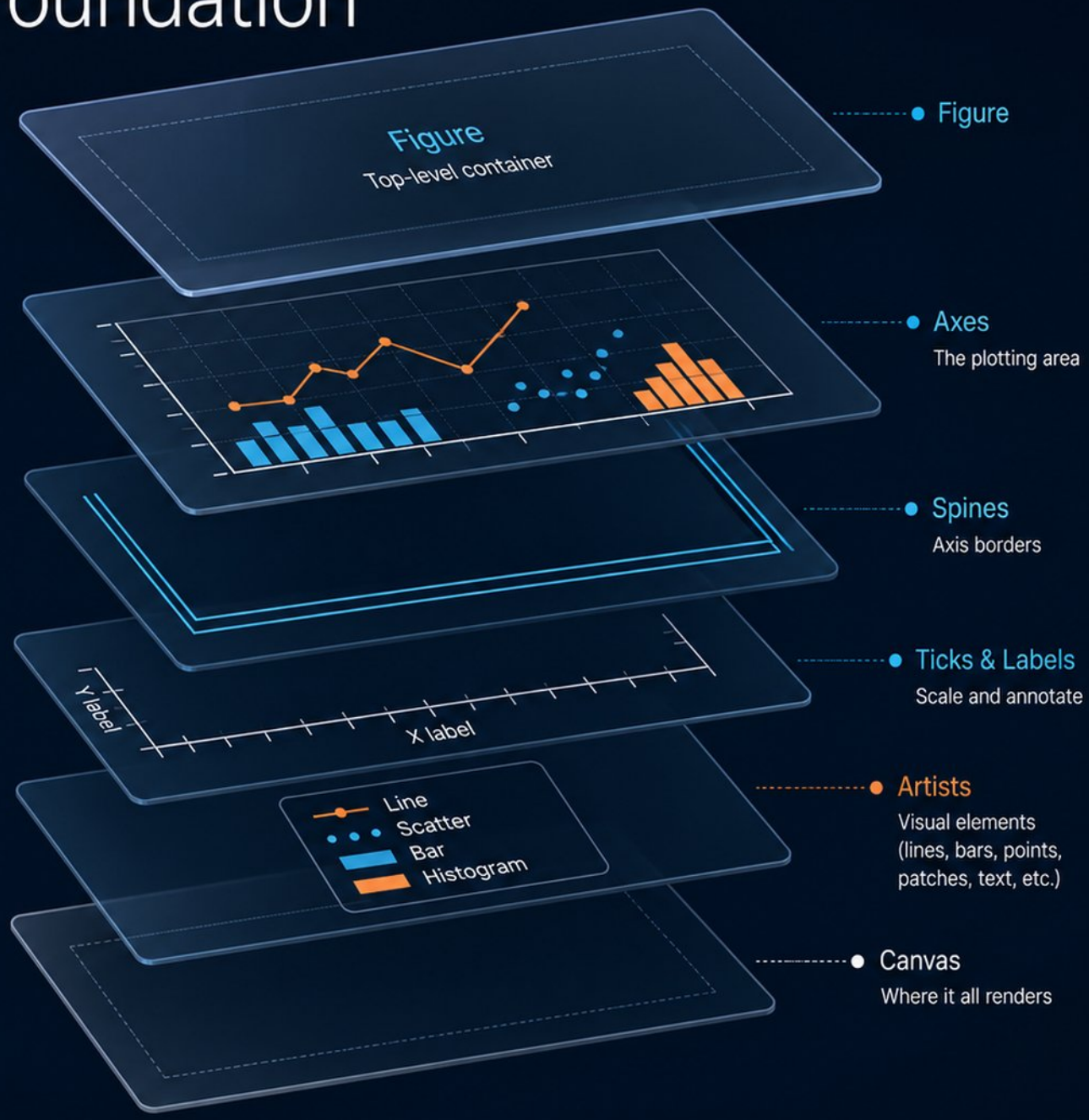
# Core Concepts and Chart Selection

- Match chart type to the analytical question
- Encode data with purpose: color, scale, axes
- Static for precision, interactive for exploration
- Design for clarity and accessibility



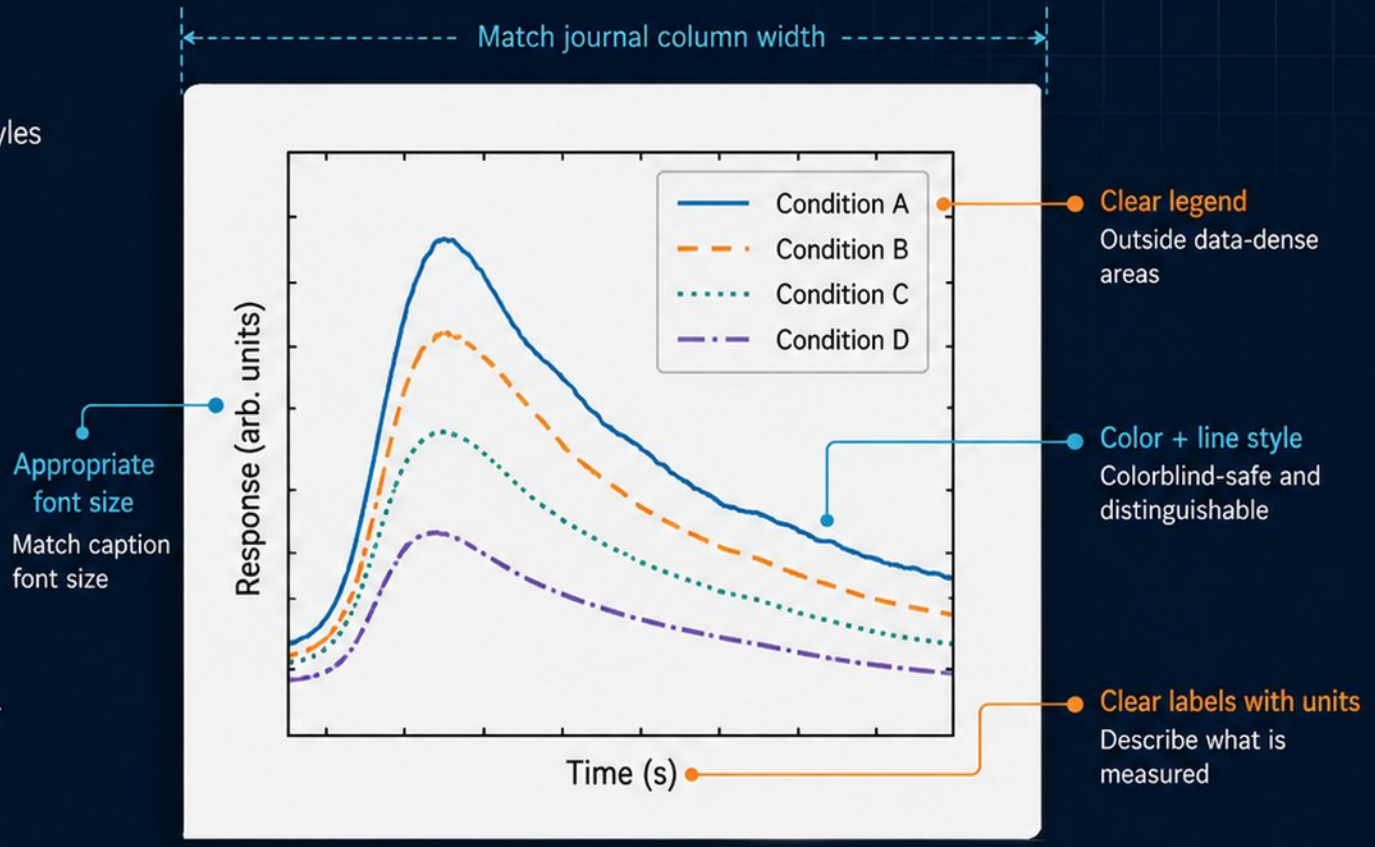
# The Matplotlib Foundation

- Figure and Axes object model core
- Line, bar, scatter, and histogram plots
- Full control over ticks, labels, legends, styles
- Ubiquitous foundation: Seaborn and pandas build on it
- Vital for pixel-perfect static and publication figures



# Publication-Ready Matplotlib

- Set global rcParams for consistent fonts, sizes, and styles
- Match figure size to journal column width and font size to captions
- Use colorblind-safe palettes with color + line style distinction
- Export PDF/SVG (vector) or high-DPI PNG (300+ dpi), never JPEG
- Avoid pixelated text, vague labels, and missing units




PDF SVG

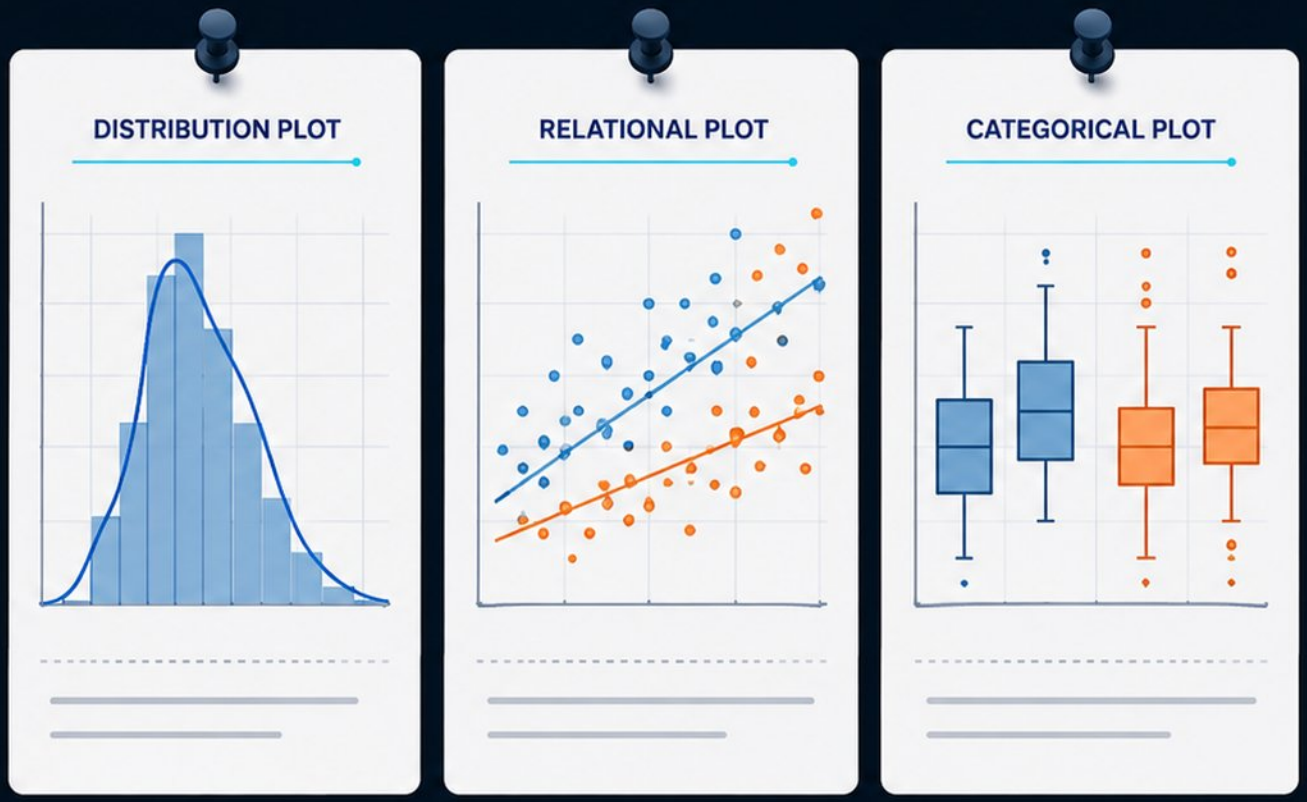
Preferred: PDF/SVG (vector)
Scalable, crisp, and print-ready

Alternative: High-DPI PNG (300+ dpi), never JPEG

Colorblind-safe palette (example)

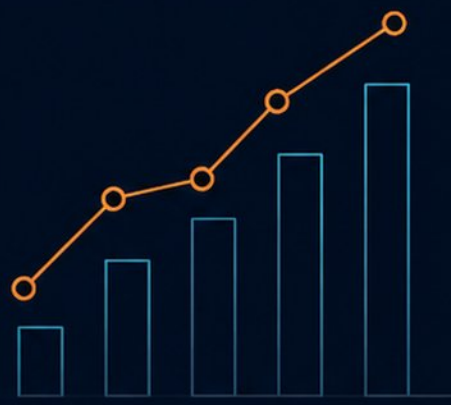
# Seaborn for Statistical Graphics

- High-level statistical plots, beautiful defaults
- Distribution, relational, and categorical plots
- Faceting for comparison and grouped analysis
- One-line plots over raw Matplotlib code
- Renders static, publication-quality figures



# Pandas Built-in Plotting


- Plot directly from DataFrames with .plot()
- Fast exploration: line, bar, histogram, scatter
- Best for initial data checks and quick EDA
- Limited customization versus Matplotlib and Seaborn
- Smooth integration with data wrangling workflows



# Interactive Visualization with Plotly

- **Plotly Express:** high-level API, one-line charts from DataFrames

- **Graph Objects:** low-level control for custom traces and layouts

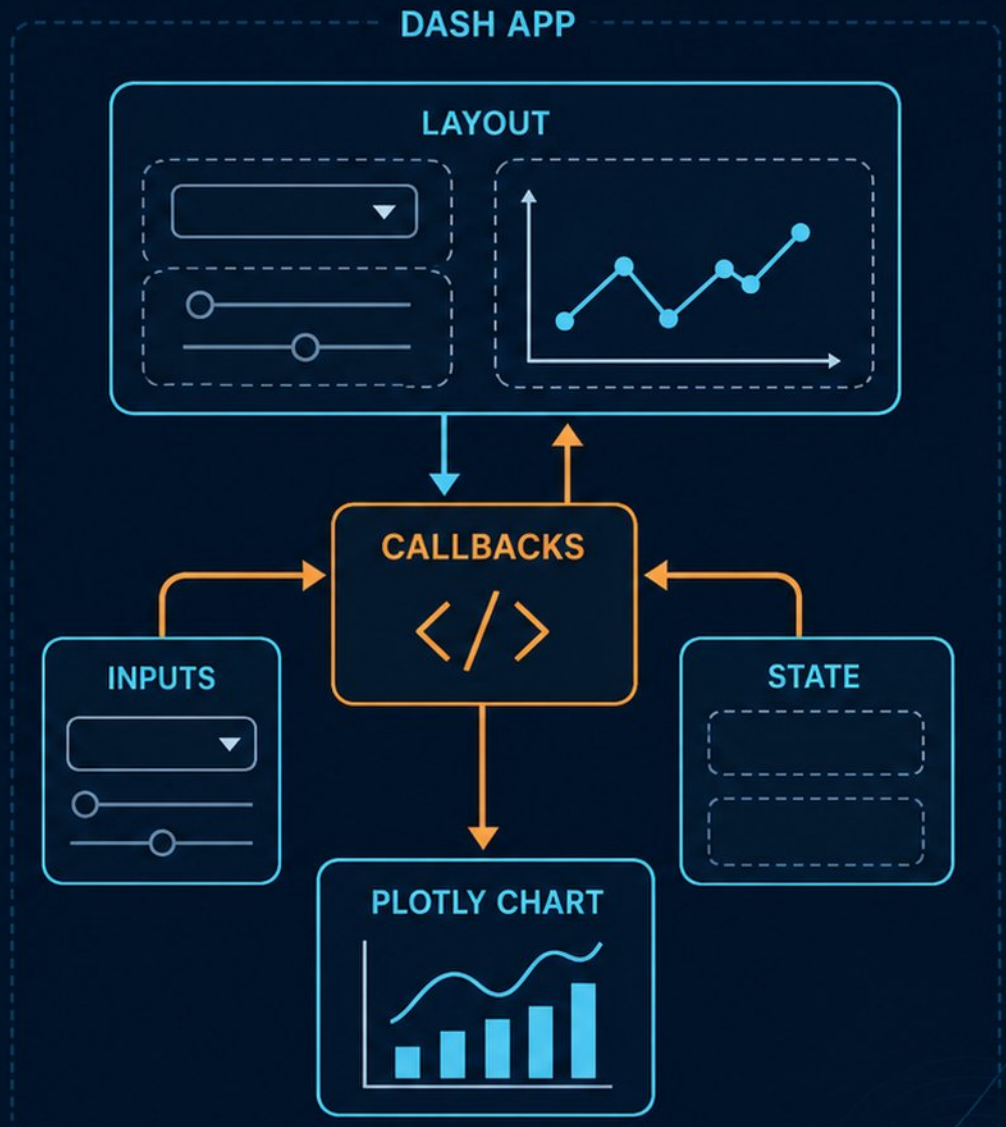
- **Built-in interactivity:** hover, zoom, pan, and clickable legends

- `fig.write_html()` exports a standalone, shareable file

- **Ideal for dashboards and web pages**, not static reports

# Dashboards and Web Embedding with Dash

- Dash = Python-only analytical web apps; no JavaScript required
- Core structure: layout, callbacks, and state inputs
- Callbacks wire dropdowns and filters to Plotly charts
- Production deployment via Gunicorn and managed platforms
- Combine pandas aggregation and Plotly for interactive analytics



# ltair and
# eclarative Visualization

Declarative grammar of graphics: describe what, not how

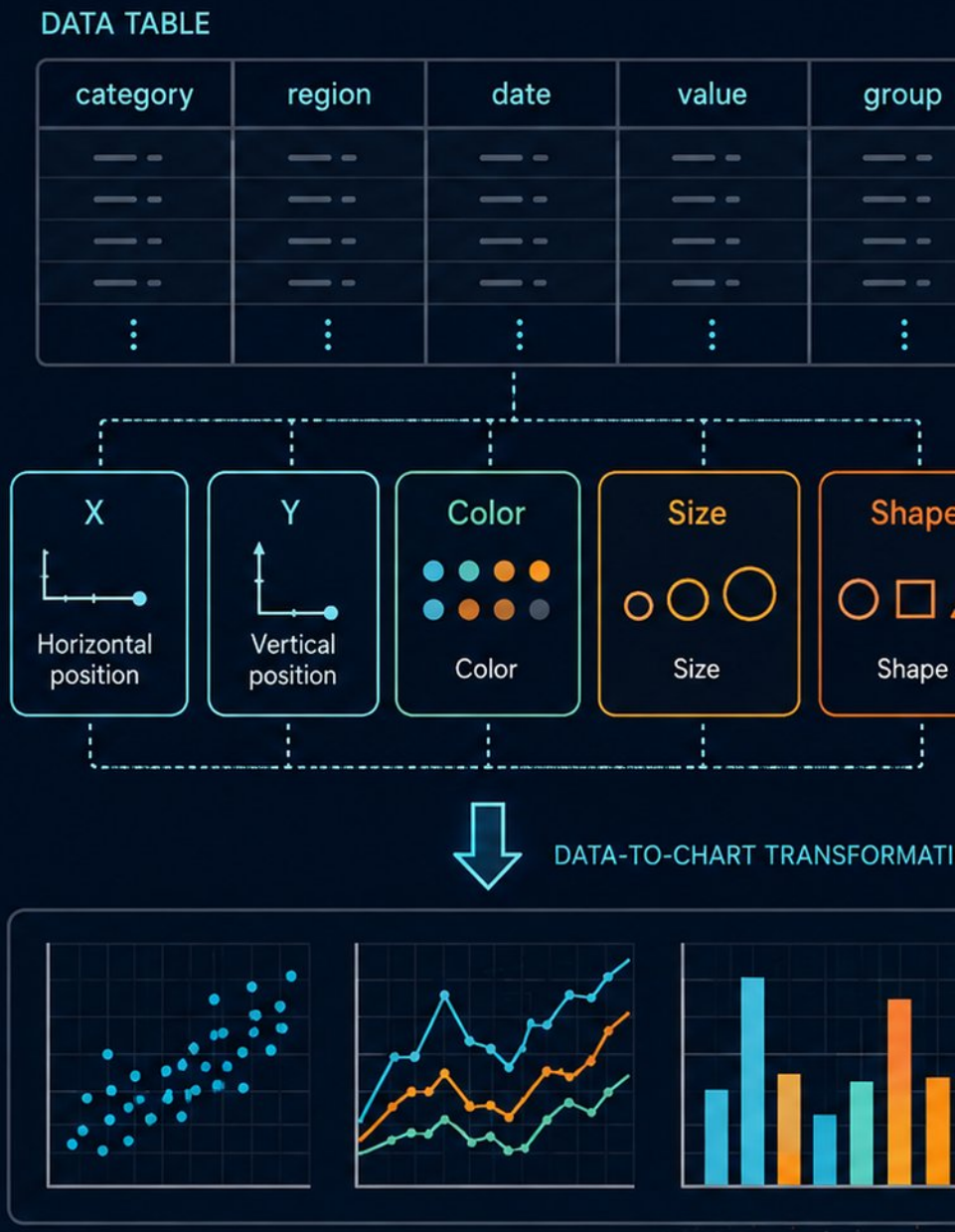
Encoding channels map data columns to visual properties

Built-in layering, faceting, and interaction composition

5,000-row default limit requires VegaFusion for large data

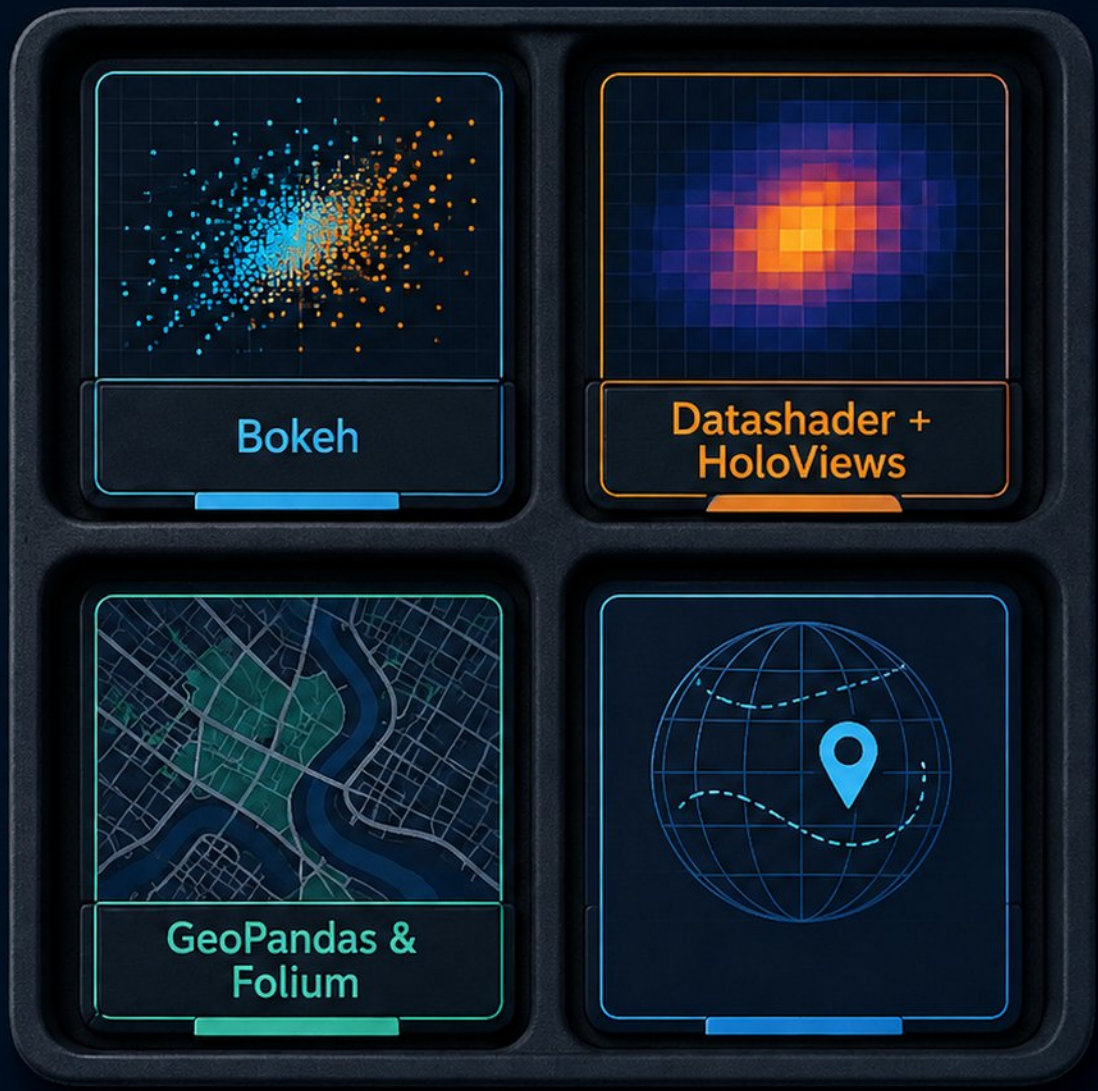
Altair: clean specs for complex facets; Plotly: richer chart types and ecosystem

Using Vega-Lite grammar, Altair excels at rapid exploratory analysis with concise, unambiguous code.



# Specialized Tools for Niche Needs

- Bokeh: large, streaming interactive datasets (200k+ points)
- Datashader + HoloViews: rasterize millions of points for density views
- GeoPandas & Folium: static and interactive geospatial mapping
- Match tool to data scale to avoid tool creep



# Practical Workflow and Style Guidelines

- Build reusable plotting functions with `fig, ax = plt.subplots()`
- Apply `.mplstyle` files for consistent fonts, colors, sizes across figures
- Separate display DPI from savefig DPI (300+ for publication)
- Use global rcParams: labelsize, layout='constrained', perceptually uniform colormaps
- Debug: check for blank figures, clipped labels, and duplicate style issues

# Exporting and Sharing Visual Outputs

- - PDF/SVG vector for journals; PNG at 300+ dpi
- - Embed charts in notebooks and documents
- - Use bbox_inches='tight' to trim excess whitespace
- - Set pdf.fonttype=42 for embedded TrueType fonts
- - Rasterize dense plot areas to keep file sizes manageable

Vector = scalable no quality loss

## PDF

- High-quality, print-ready
- Fonts embedded
- Ideal for journals

## SVG

- Scalable vector graphics
- Crisp at any size
- Best for line art

## PNG

- Raster image format
- Use at 300+ dpi
- Good for photographs and complex visuals

## HTML

- Interactive and lightweight
- Great for web sharing
- Supports interactivity

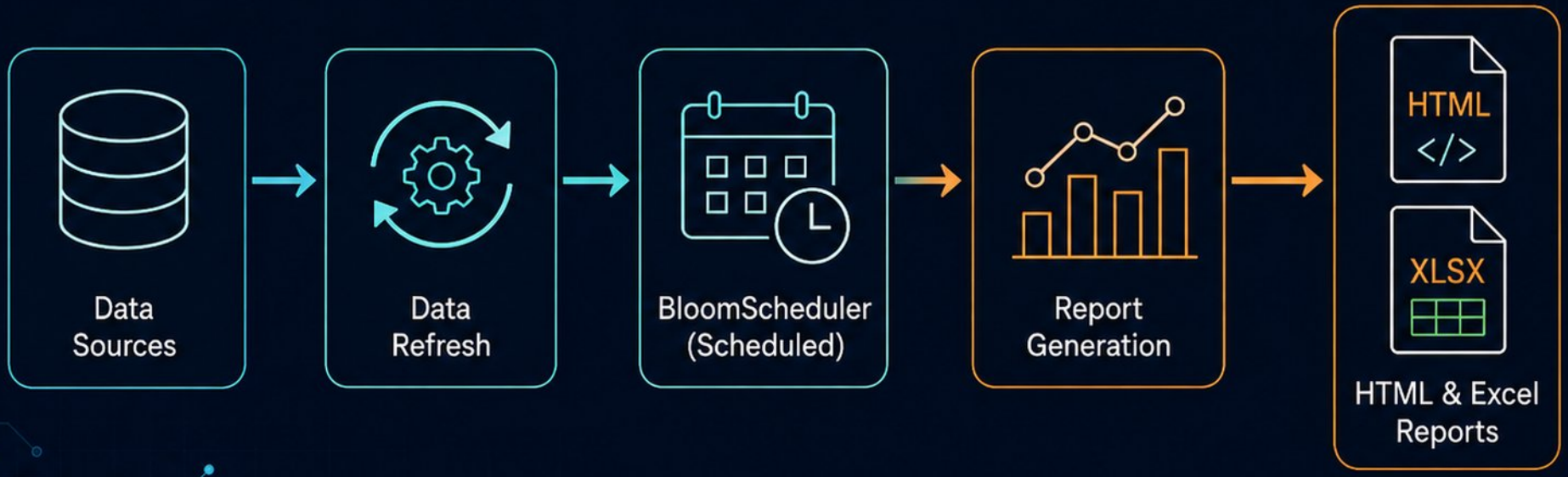
Consider audience & use case

Trim whitespace

Manage file size without losing clarity

# Automating Reports with Python

- Schedule recurring charts with BloomScheduler: daily, weekly, hourly
- Build self-contained HTML reports with Plotly or Mermaid
- Automate styled Excel reports via pandas and openpyxl
- Batch strategies: one report per run or segment
- No runtime or dependencies needed for viewer access



# Choosing the Right Tool for Your Next Chart

- - Choose by output: static for pixels, interactive for browsers
- - Seaborn for statistical plots; Plotly for dashboards
- - Matplotlib for publication; Altair for declarative charts
- - Match data size: Bokeh for 200k+ points; VegaFusion for Altair
- - Start with your workflow, then pick the library that fits

| CONSIDERATION | RECOMMENDED TOOLS | GOOD STARTING POINT |
|---|---|---|
| Choose by output: static for pixels, interactive for browsers | seaborn ; plotly | |
| Seaborn for statistical plots; Plotly for dashboards | seaborn ; plotly | |
| Matplotlib for publication; Altair for declarative charts | matplotlib ; ALTAIR | |
| Match data size: Bokeh for 200k+ points; VegaFusion for Altair | bokeh ; vegafusion | |
| Start with your workflow, then pick the library that fits | | |

PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/87e55412/public