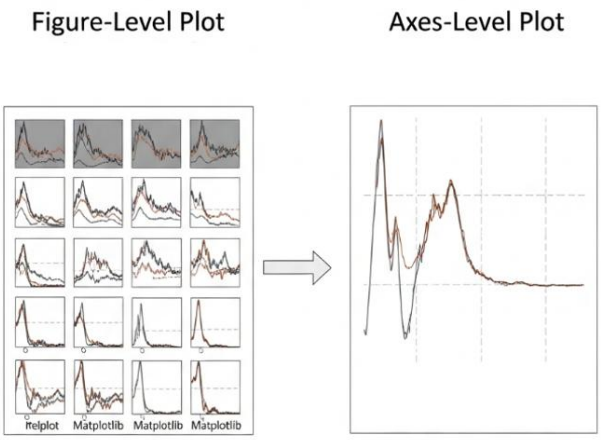
Figure level Vs Axis level plots

In Seaborn, understanding the distinction between Figure-Level and Axes-Level plots is crucial for effective visualization and for knowing how to customize your plots. This distinction primarily relates to **how the plot interacts with Matplotlib's underlying figure and axes objects.**



Matplotlib's Figure and Axes (The Foundation)

Before diving into Seaborn's levels, it's important to briefly understand Matplotlib's core components:

- **Figure:** This is the *entire canvas* or "paper" on which you draw. It can contain one or more plots (axes). Think of it as the window that pops up when you plot something.
- **Axes (or Subplot):** This is the *actual plotting area* where your data is drawn (e.g., the area with x and y axes, titles, and data points). A single figure can have multiple axes, allowing you to create grids of plots.

Figure-Level Plots in Seaborn

What they are: Figure-level plots are Seaborn functions that **manage their own Matplotlib Figure and Axes objects internally.** They are designed to create complex, multi-panel plots (like grids of subplots) where different subsets of your data are visualized in separate axes within the same figure.

Key Characteristics:

- **Creates its own Figure and Axes:** You don't explicitly pass ax (Axes) objects to these functions. They return a FacetGrid, PairGrid, or JointGrid object, which is a wrapper around the Matplotlib Figure and Axes.
- **Handles Faceting/Gridding:** They are ideal for creating "facets" or "grids" of plots, where you want to visualize the same type of relationship across different categories or conditions in your data.
- **Returns a Grid Object:** Instead of returning a Matplotlib Axes object, they return a Seaborn Grid object (e.g., FacetGrid, PairGrid, JointGrid). This object then provides methods for further customization of the entire grid.
- **Examples:** relplot(), displot(), catplot(), lmplot(), pairplot(), jointplot().

Why they are used: Figure-level plots are preferred when you want to:

- **Explore relationships across multiple categorical variables:** For example, catplot() can show box plots of sales by product category, faceted by region.
- **Visualize distributions or relationships for different subsets of data:** displot() can show histograms of a variable, separated by a third categorical variable.
- **Create complex, multi-panel dashboards quickly:** They automate the layout and linking of multiple subplots.

Analogy: Think of a Figure-Level plot as a **pre-designed poster layout** where you tell it what data to put in each section, and it handles the arrangement and individual plot types for you.

Axes-Level Plots in Seaborn

What they are: Axes-level plots are Seaborn functions that **draw directly onto a single Matplotlib Axes object**. They are designed to create individual plots that typically visualize the relationship between two or three variables.

Key Characteristics:

- **Requires an Axes Object (Implicitly or Explicitly):**
    - If you don't provide an Axes object, Seaborn will typically draw on the "current" active Axes (if one exists) or create a new default one.
    - It's best practice to explicitly pass an ax (Axes) object to these functions using the ax= parameter, especially when creating subplots manually.
- **Returns a Matplotlib Axes Object:** These functions return a Matplotlib Axes object, which you can then use for further Matplotlib-specific customizations (e.g., setting titles, labels, limits).
- **Designed for Single Plots:** They are meant for creating one plot at a time, although you can combine multiple Axes-level plots on a single Axes object (e.g., a scatter plot and a line plot on the same axes).
- **Examples:** scatterplot(), lineplot(), boxplot(), violinplot(), histplot(), kdeplot(), barplot(), heatmap(), regplot().

Why they are used: Axes-level plots are preferred when you want to:

- **Create a single, focused plot:** For a specific visualization of two variables.
- **Have fine-grained control over subplot layout:** When you're manually arranging multiple plots in a matplotlib.pyplot.subplots() grid.
- **Combine different plot types on the same axes:** For example, overlaying a KDE plot on a histogram.
- **Integrate seamlessly with existing Matplotlib figures:** When you've already set up your figure and axes and just want Seaborn to draw the data.

Analogy: Think of an Axes-Level plot as a **single drawing tool** (like a specific type of brush) that you use to draw *on a specific part of your canvas* (an Axes object) that you've already prepared.

Summary of Differences:

Feature	Figure-Level Plot	Axes-Level Plot
Output	Seaborn Grid object (e.g., FacetGrid)	Matplotlib Axes object
Axes Control	Manages its own Figure/Axes internally	Draws on (or returns) a single Axes object
Faceting	Built-in for creating multi-panel grids	Does not inherently create multiple panels
Customization	Customization often done via Grid object methods or iterating over axes	Customization done directly on the returned Axes object using Matplotlib functions
Use Case	Exploring relationships across categories, multi-panel plots	Single, focused plots; fine-grained subplot control

Choosing between Figure-Level and Axes-Level plots depends on your specific visualization goal: use Figure-Level for quick, complex grids, and Axes-Level for precise control over individual plots or when integrating into existing Matplotlib layouts.