Scatterplots and Resistant Lines in Bivariate Analysis

Scatterplots are one of the simplest ways to visualize the relationship between two quantitative variables in bivariate analysis. **Resistant lines** are a way to fit a linear trend to the scatterplot while minimizing the influence of outliers, making them robust to unusual data points.

Scatterplots

Scatterplots plot individual data points for two variables, allowing us to see patterns, correlations, and potential outliers.

Resistant Lines

Resistant lines, unlike least squares regression lines, are less influenced by extreme outliers. They are particularly useful when:

- The dataset contains outliers.
- A robust trend line is needed for analysis.

Example

Dataset

Dataset containing the number of hours studied and the corresponding test scores.

1. Scatterplot

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Sample dataset
np.random.seed(42)
study_hours = np.array([2, 3, 5, 1, 4, 6, 3, 7, 8, 10])
scores = np.array([50, 55, 70, 40, 65, 80, 60, 85, 90, 100])

# Create a scatterplot
plt.figure(figsize=(8, 6))
plt.scatter(study_hours, scores, color='blue', label='Data Points')
plt.title("Scatterplot: Study Hours vs Test Scores")
plt.xlabel("Study Hours")
plt.ylabel("Test Scores")
plt.grid(alpha=0.4)
plt.legend()
plt.show()
```

Explanation:

- plt.scatter(): Plots data points.
- Shows the relationship between study_hours and scores.

2. Adding Resistant Lines

Here, we compute and overlay a resistant line manually. We'll use **median smoothing** to estimate the resistant line.

```
# Sort the data for resistant line calculation
sorted_indices = np.argsort(study_hours)
sorted_hours = study_hours[sorted_indices]
sorted_scores = scores[sorted_indices]

# Divide the data into two halves and calculate the medians
mid_index = len(sorted_hours) // 2
x_left, x_right = sorted_hours[:mid_index], sorted_hours[mid_index:]
y_left, y_right = sorted_scores[:mid_index], sorted_scores[mid_index:]

median_x_left, median_y_left = np.median(x_left), np.median(y_left)
median_x_right, median_y_right = np.median(x_right), np.median(y_right)

# Calculate the resistant line slope and intercept
resistant_slope = (median_y_right - median_y_left) / (median_x_right - median_x_left)
resistant_intercept = median_y_left - resistant_slope * median_x_left

# Line equation: y = mx + c
resistant_line = resistant_slope * study_hours + resistant_intercept

# Plotting the scatterplot with the resistant line
plt.figure(figsize=(8, 6))
plt.scatter(study_hours, scores, color='blue', label='Data Points')
plt.plot(study_hours, resistant_line, color='red', label='Resistant Line', linewidth=2)
plt.title("Scatterplot with Resistant Line")
plt.xlabel("Study Hours")
plt.ylabel("Test Scores")
plt.grid(alpha=0.4)
plt.legend()
plt.show()
```

Explanation:

1. **Median Smoothing**:
   - The data is split into two halves.
   - Medians of x and y are computed for each half.
2. **Slope and Intercept**:

   - Calculated based on the medians.
3. **Resistant Line**:
   - Added to the scatterplot using the calculated slope and intercept.

3. Using Seaborn's Regression Plot (For Comparison)

Although Seaborn does not directly provide resistant lines, it can fit least-squares regression lines.

```
sns.regplot(x=study_hours, y=scores, ci=None, scatter_kws={"color": "blue"},
line_kws={"color": "red"})
plt.title("Scatterplot with Least Squares Regression Line")
plt.xlabel("Study Hours")
plt.ylabel("Test Scores")
plt.grid(alpha=0.4)
plt.show()
```

Output Comparison

1. **Scatterplot**: Shows raw data points and highlights patterns.
2. **Resistant Line**: More robust against outliers.
3. **Regression Line**: Suitable when the data is not heavily affected by outliers.

- regplot(): Fits a regression line using the least-squares method.
- ci=None: Disables the confidence interval shading.

Key Insights

1. **Scatterplots**:
   - Reveal linear or nonlinear relationships.
   - Highlight clusters and outliers.
2. **Resistant Lines**:
   - Robust to outliers.
   - Useful for exploratory analysis when data is noisy or contains anomalies.
3. **Regression Line**:
   - Sensitive to outliers but useful for prediction when data is clean.