Handling Several Batches in Bivariate Analysis

When analyzing bivariate relationships across **several batches** (e.g., multiple categories or groups within the data), we extend our analysis by breaking down the relationship into **subgroups**. This allows us to explore how the relationship between two variables differs across batches or groups.

Key Techniques

1. **Grouped Scatter Plots**:
   - Visualize the relationship between two variables for each batch.
2. **Faceted Plots**:
   - Subdivide the data into smaller groups and visualize separately for each batch using libraries like Seaborn.
3. **Grouped Aggregations**:
   - Perform statistical summaries (e.g., mean, median) for bivariate relationships within each batch.
4. **Pivot Tables**:
   - Use pivot tables to analyze relationships for each batch systematically.

Syntax and Example

1. Example Dataset

```
import pandas as pd
import numpy as np

# Sample dataset
data = {
   'Batch': ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'],
   'Study_Hours': [2, 3, 5, 1, 4, 6, 3, 7, 8],
   'Scores': [50, 55, 70, 45, 68, 80, 60, 85, 90]
}

# Create a DataFrame
df = pd.DataFrame(data)
print(df)
```

Output:

	Batch	Study_Hours	Scores
0	A	2	50
1	A	3	55
2	A	5	70
3	B	1	45
4	B	4	68
5	B	6	80

6	C	3	60
7	C	7	85
8	C	8	90

*2. **Grouped Scatter Plot***

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

# Scatter Plot with Groups
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Study_Hours', y='Scores', hue='Batch', style='Batch', palette='viridis', s=100)
plt.title("Scatter Plot of Study Hours vs Scores by Batch")
plt.xlabel("Study Hours")
plt.ylabel("Scores")
plt.grid()
plt.legend(title="Batch")
plt.show()
```

Explanation:

- hue='Batch': Differentiates points by Batch using color.
- style='Batch': Differentiates points by Batch using marker style.

3. Faceted Plot for Each Batch

```
# Faceted Scatter Plot
g = sns.FacetGrid(df, col="Batch", col_wrap=3, height=4)
g.map_dataframe(sns.scatterplot, x="Study_Hours", y="Scores", color="blue")
g.set_axis_labels("Study Hours", "Scores")
g.set_titles("Batch {col_name}")
plt.show()
```

Explanation:

- **FacetGrid** creates separate subplots for each batch.
- Allows easy comparison of patterns across batches.

*4. **Grouped Aggregations***

```
# Calculate Mean Scores for Each Batch
mean_scores = df.groupby('Batch')[['Study_Hours', 'Scores']].mean()
print("Mean Scores by Batch:\n", mean_scores)
```

Output:

Mean Scores by Batch:

Batch	Study_Hours	Scores
A	3.333333	58.333333
B	3.666667	64.333333
C	6.000000	78.333333

5. Pivot Table

```
# Create Pivot Table
pivot_table = pd.pivot_table(df, values='Scores', index='Batch', columns='Study_Hours',
aggfunc='mean', fill_value=0)
print("Pivot Table:\n", pivot_table)
```

Output:

Pivot Table:

Batch	Study_Hours 1	2	3	4	5	6	7	8
A	0	50	55	0	70	0	0	0
B	45	0	0	68	0	80	0	0
C	0	0	60	0	0	0	85	90

Explanation:

- The pivot table summarizes Scores across Study_Hours for each Batch.
- aggfunc='mean': Calculates the mean for each combination.

*6. **Regression Line for Each Batch***

```
# Regression Plot with Hue (by Batch)
sns.lmplot(data=df, x='Study_Hours', y='Scores', hue='Batch', height=6)
plt.title("Regression Line for Each Batch")
plt.show()
```

Explanation:

- lmplot fits a regression line for each batch.
- Highlights trends within each group.

Key Insights from the Analysis

1. **Scatter Plots**:
   - Show the spread of data points across batches.
   - Allow visual comparison of trends and outliers.
2. **Faceted Plots**:

- Help visualize the relationship for each batch independently.

3. **Grouped Aggregations**:
   - Provide statistical summaries (e.g., mean, median) for each batch.
4. **Pivot Tables**:
   - Present a structured view of the data, enabling easy analysis of combinations.
5. **Regression Plots**:
   - Indicate the direction and strength of relationships within batches.