Transformation Techniques in Bivariate Analysis

1. **Log Transformation**:
   - Useful when data has exponential growth or wide variation in scale.
   - Converts multiplicative relationships into additive ones.

   Formula:

   $y' = \log(y)$

   where y' is the transformed variable and y is the original variable.

2. **Square Root Transformation**:
   - Applied when the data has a strong right-skew (positive skew).
   - Reduces the effect of large values and compresses the data distribution.

   Formula:

   $y' = \text{sqrt}(y)$

   where y' is the transformed variable and y is the original variable.

3. **Box-Cox Transformation**:
   - A family of transformations that can be used to stabilize variance and make data more normally distributed.
   - It includes a parameter λ\lambdaλ which determines the type of transformation applied.

   Formula:

$$y' = \frac{y^\lambda - 1}{\lambda} \quad \text{if} \quad \lambda \neq 0$$

$$y' = \log(y) \quad \text{if} \quad \lambda = 0$$

   where y' is the transformed variable, y is the original variable, and λ lambda is a parameter that is estimated.

4. **Z-Score Normalization**:
   - Standardizes the variables by subtracting the mean and dividing by the standard deviation.
   - Useful when variables have different scales and need to be compared directly.

Formula:

$z=(y-\mu)/\sigma$

where z is the transformed variable, y is the original variable, μ is the mean, and σ is the standard deviation.

5. **Power Transformation**:
   - Similar to Box-Cox but allows a more general transformation where the exponent ppp can be any real number.
   - Can be used to stabilize variance and make distributions more normal.

Formula:

$y'=y^p$

where y' is the transformed variable, y is the original variable, and p is the exponent (can be chosen based on the data).

6. **Reciprocal Transformation**:
   - Takes the reciprocal of the variable to transform it. This is often used to handle extreme positive skewness.

Formula:

Y' = 1/y

where y' is the transformed variable and y is the original variable.

Example

We will consider a dataset where we have two variables: **Study Hours** and **Test Scores**. Let's apply transformations to these variables.

1. Example Dataset

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

# Sample dataset
data = {
   'Study_Hours': [2, 3, 5, 1, 4, 6, 3, 7, 8, 10],
   'Scores': [50, 55, 70, 40, 65, 80, 60, 85, 90, 100]
}
```

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

# Plotting original data
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Study_Hours', y='Scores', data=df)
plt.title("Scatterplot: Study Hours vs Test Scores (Original Data)")
plt.xlabel("Study Hours")
plt.ylabel("Scores")
plt.grid(alpha=0.4)
plt.show()
```

2. Applying Log Transformation

```
# Apply Log Transformation
df['Log_Study_Hours'] = np.log(df['Study_Hours'])
df['Log_Scores'] = np.log(df['Scores'])

# Plot transformed data
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Log_Study_Hours', y='Log_Scores', data=df)
plt.title("Log Transformation: Study Hours vs Test Scores")
plt.xlabel("Log of Study Hours")
plt.ylabel("Log of Scores")
plt.grid(alpha=0.4)
plt.show()
```

Explanation:

- **Log Transformation**: Reduces the wide range of data and compresses large values, making the relationship more linear.

3. Apply Z-Score Normalization

```
# Apply Z-Score Normalization
df['Z_Study_Hours'] = (df['Study_Hours'] - df['Study_Hours'].mean()) / df['Study_Hours'].std()
df['Z_Scores'] = (df['Scores'] - df['Scores'].mean()) / df['Scores'].std()

# Plot normalized data
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Z_Study_Hours', y='Z_Scores', data=df)
plt.title("Z-Score Normalization: Study Hours vs Test Scores")
plt.xlabel("Z-Score of Study Hours")
plt.ylabel("Z-Score of Scores")
plt.grid(alpha=0.4)
plt.show()
```

Explanation:

- **Z-Score Normalization**: Standardizes both Study_Hours and Scores to have a mean of 0 and a standard deviation of 1. This is useful when the scales of the variables are different.

4. ***Apply Power Transformation (e.g., Square Root)***

```
# Apply Square Root Transformation
df['Sqrt_Study_Hours'] = np.sqrt(df['Study_Hours'])
df['Sqrt_Scores'] = np.sqrt(df['Scores'])

# Plot transformed data
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Sqrt_Study_Hours', y='Sqrt_Scores', data=df)
plt.title("Square Root Transformation: Study Hours vs Test Scores")
plt.xlabel("Square Root of Study Hours")
plt.ylabel("Square Root of Scores")
plt.grid(alpha=0.4)
plt.show()
```

- **Square Root Transformation**: This transformation reduces skewness and compresses the range of the data.
- **Log Transformation**: Useful when dealing with exponential growth or large ranges.
- **Z-Score Normalization**: Standardizes data for comparison.
- **Power/Square Root Transformation**: Stabilizes variance and handles skewed data.
- **Box-Cox Transformation**: A generalized transformation that can handle different types of data distributions.

When to Use Transformations in Bivariate Analysis

1. **Improving Linearity**: When the relationship between two variables is non-linear, transformations like log or square root can help.
2. **Stabilizing Variance**: If the variance of one or both variables increases as the value of the variable increases (heteroscedasticity), transformations can stabilize it.
3. **Handling Skewed Data**: For positively or negatively skewed data, transformations such as log or square root can help normalize the distribution.