

Point-Biserial Correlation (Specialized Case of Pearson's Correlation Coefficient)

Single Continuous Variable and Single Output Binary Variable (0 or 1)

So, we have all three levels -ve, 0(neutral) and +ve correlations

It's a non-parametric test, meaning it doesn't assume a specific distribution for the data.

The point-biserial correlation coefficient (r_{pb}) is a valuable statistical measure for examining the relationship between a dichotomous variable and a continuous variable. Let's break down its definition, properties, and applications in detail:

Definition and Characteristics

Dichotomous Variable:

This is a categorical variable that has only two possible values (e.g., 0 and 1, Yes and No, Male and Female). For instance, in a dataset where gender is represented as 0 (Male) and 1 (Female), gender is a dichotomous variable.

Continuous Variable:

This variable can take on an infinite number of values within a range (e.g., height, weight, test scores).

Point-Biserial Correlation Coefficient (r_{pb}):

It quantifies the strength and direction of the relationship between the dichotomous and continuous variables. The calculation is based on the means and standard deviations of the continuous variable for each category of the dichotomous variable.

Mathematical Representation:

The point-biserial correlation is mathematically equivalent to the Pearson correlation coefficient when the dichotomous variable is recoded to two distinct values (e.g., 0 and 1). Thus, if you have a continuous variable (X) and a dichotomous variable (Y), you can express it as:

$$r_{XY} = r_{pb}$$

The formula for calculating the point-biserial correlation coefficient (r_{pb}) is:

$$r_{pb} = \frac{M_1 - M_0}{s} \cdot \sqrt{\frac{n_1 \cdot n_0}{n^2}}$$

Where:

- (M_1) = Mean of the continuous variable for one group (e.g., ($Y=1$))
- (M_0) = Mean of the continuous variable for the other group (e.g., ($Y=0$))
- (s) = Standard deviation of the continuous variable
- (n_1) = Number of observations in group 1 (e.g., ($Y=1$))
- (n_0) = Number of observations in group 0 (e.g., ($Y=0$))
- (n) = Total number of observations

Interpretation

Weak Correlation: $|r| < 0.3$

Moderate Correlation: $0.3 \leq |r| < 0.5$

Strong Correlation: $|r| \geq 0.5$

Point-Biserial Correlation Coefficient (

✓ r_{pb}
)

Formula

The formula for the Point-Biserial Correlation Coefficient is given by:

$$r_{pb} = \frac{M_1 - M_0}{s_n} \sqrt{\frac{n_1 n_0}{n^2}}$$

Where:

- (M_1) = Mean of (X) for ($Y = 1$)
- (M_0) = Mean of (X) for ($Y = 0$)
- (s_n =

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2}$$

) (Standard deviation of the population)

- (n_1) = Number of observations in group 1
- (n_0) = Number of observations in group 0
- ($n = n_1 + n_0$)

Alternative Formula

An alternative formula for the Point-Biserial Correlation Coefficient is:

$$r_{pb} = \frac{M_1 - M_0}{s_{n-1}} \sqrt{\frac{n_1 n_0}{n(n-1)}}$$

Where:

$$s_{n-1} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2}$$

Significance Testing

The formula for significance testing of (r_{pb}) is:

$$t = r_{pb} \sqrt{\frac{n_1 + n_0 - 2}{1 - r_{pb}^2}}$$

```
import pandas as pd
from scipy import stats

# Simple dataset
data = {
    'Gender': [0, 1, 0, 1, 0, 1, 1, 0, 1, 0], # Binary variable
    'Height': [175, 160, 180, 155, 178, 162, 165, 172, 159, 170] # Continuous variable
}

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

# Calculate Point-Biserial Correlation
point_biserial_corr, p_value = stats.pointbiserialr(df['Gender'], df['Height'])

# Display the results
print(f"SciPy Implementation - Point-Biserial Correlation Coefficient: {point_biserial_corr}")
print(f"P-Value: {p_value:.4f}")

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

# Sample data
gender = np.array([0, 1, 0, 1, 0, 1, 1, 0, 1, 0]) # Binary variable
height = np.array([175, 160, 180, 155, 178, 162, 165, 172, 159, 170]) # Continuous variable

# Separate heights by gender
height_0 = height[gender == 0] # Heights for Male
height_1 = height[gender == 1] # Heights for Female
```

```

# Calculate means
M_0 = np.mean(height_0) # Mean height for Males
M_1 = np.mean(height_1) # Mean height for Females

# Calculate standard deviation of the entire height dataset
s = np.std(height, ddof=0.00) # Population standard deviation

# Calculate the number of observations in each group
n_0 = len(height_0)
n_1 = len(height_1)
n = len(height) # Total number of observations

# Calculate Point-Biserial Correlation Coefficient
r_pb = (M_1 - M_0) / s * np.sqrt((n_1 * n_0) / (n**2))

# Display the results
print(f"Custom Implementation - Point-Biserial Correlation Coefficient: {r_pb:.4f}")

# Plotting
plt.figure(figsize=(12, 6))

# Box Plot
plt.subplot(1, 2, 1)
sns.boxplot(x=gender, y=height, palette='muted')
plt.xticks([0, 1], ['Male', 'Female'])
plt.title('Height Distribution by Gender (Box Plot)')
plt.ylabel('Height (cm)')
plt.xlabel('Gender')

# Scatter Plot
plt.subplot(1, 2, 2)
plt.scatter([0]*len(height_0), height_0, color='blue', label='Male', alpha=0.6)
plt.scatter([1]*len(height_1), height_1, color='red', label='Female', alpha=0.6)
plt.axhline(M_0, color='blue', linestyle='--', label='Mean Male Height')
plt.axhline(M_1, color='red', linestyle='--', label='Mean Female Height')
plt.xticks([0, 1], ['Male', 'Female'])
plt.title('Height Scatter Plot by Gender')
plt.ylabel('Height (cm)')
plt.xlabel('Gender')
plt.legend()

plt.tight_layout()
plt.show()

# Plotting
plt.figure(figsize=(10, 6))

# Scatter Plot with Regression Line
sns.regplot(x=gender, y=height, logistic=False, ci=None,
            marker='o', scatter_kws={'alpha':0.6, 's':100},
            line_kws={'color': 'green', 'label': 'Trend Line'})

# Mean lines
plt.axhline(M_0, color='blue', linestyle='--', label='Mean Male Height')

```

```

plt.axhline(M_1, color='red', linestyle='--', label='Mean Female Height')

plt.xticks([0, 1], ['Male', 'Female'])
plt.title('Height vs. Gender with Point-Biserial Correlation')
plt.ylabel('Height (cm)')
plt.xlabel('Gender')
plt.legend()

plt.grid()
plt.tight_layout()
plt.show()

import numpy as np

# Sample data
gender = np.array([0, 1, 0, 1, 0, 1, 1, 0, 1, 0]) # Binary variable
height = np.array([175, 160, 180, 155, 178, 162, 165, 172, 159, 170]) # Continuo

# Separate heights by gender
height_0 = height[gender == 0] # Heights for Male
height_1 = height[gender == 1] # Heights for Female

# Calculate means
M_0 = np.mean(height_0) # Mean height for Males
M_1 = np.mean(height_1) # Mean height for Females

# Calculate standard deviation of the entire height dataset
s = np.std(height, ddof=0.00) # Sample standard deviation

# Calculate the number of observations in each group
n_0 = len(height_0)
n_1 = len(height_1)
n = len(height) # Total number of observations

# Calculate Point-Biserial Correlation Coefficient
r_pb = (M_1 - M_0) / s * np.sqrt((n_1 * n_0) / (n**2))

# Display the results
print(f"Custom Implementation - Point-Biserial Correlation Coefficient: {r_pb:.4f}")

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

# Sample data
gender = np.array([0, 1, 0, 1, 0, 1, 1, 0, 1, 0]) # Binary variable
height = np.array([175, 160, 180, 155, 178, 162, 165, 172, 159, 170]) # Continuo

# Separate heights by gender
height_0 = height[gender == 0] # Heights for Male
height_1 = height[gender == 1] # Heights for Female

```

```

# Calculate means
M_0 = np.mean(height_0) # Mean height for Males
M_1 = np.mean(height_1) # Mean height for Females

# Calculate standard deviation of the entire height dataset
s = np.std(height, ddof=0) # Population standard deviation

# Calculate the number of observations in each group
n_0 = len(height_0)
n_1 = len(height_1)
n = len(height) # Total number of observations

# Calculate Point-Biserial Correlation Coefficient
r_pb = (M_1 - M_0) / s * np.sqrt((n_1 * n_0) / (n**2))

# Display the results
print(f"Custom Implementation - Point-Biserial Correlation Coefficient: {r_pb:.4f}")

# Plotting
plt.figure(figsize=(10, 5))
plt.boxplot([height_0, height_1], labels=['Male', 'Female'])
plt.title('Height Distribution by Gender')
plt.ylabel('Height (cm)')
plt.xlabel('Gender')
plt.axhline(M_0, color='blue', linestyle='--', label='Mean Male Height')
plt.axhline(M_1, color='red', linestyle='--', label='Mean Female Height')
plt.legend()
plt.grid()
plt.show()

# Plotting
plt.figure(figsize=(10, 5))
sns.boxplot(x=gender, y=height, palette='muted', width=0.5)
sns.stripplot(x=gender, y=height, color='black', alpha=0.5)
plt.xticks([0, 1], ['Male', 'Female'])
plt.title('Height Distribution by Gender (Strip Plot)')
plt.ylabel('Height (cm)')
plt.xlabel('Gender')
plt.grid()
plt.show()

# Plotting
plt.figure(figsize=(8, 5))
mean_heights = [M_0, M_1]
labels = ['Male', 'Female']
plt.bar(labels, mean_heights, color=['blue', 'red'], alpha=0.7)
plt.title('Mean Height by Gender')
plt.ylabel('Mean Height (cm)')
plt.xlabel('Gender')
plt.ylim(150, 200) # Set y-axis limits for better visibility
plt.grid(axis='y')
plt.show()

```

```

# Plotting
plt.figure(figsize=(10, 5))
plt.scatter([0]*len(height_0), height_0, color='blue', label='Male', alpha=0.6)
plt.scatter([1]*len(height_1), height_1, color='red', label='Female', alpha=0.6)
plt.axhline(M_0, color='blue', linestyle='--', label='Mean Male Height')
plt.axhline(M_1, color='red', linestyle='--', label='Mean Female Height')
plt.xticks([0, 1], ['Male', 'Female'])
plt.title('Height Scatter Plot by Gender')
plt.ylabel('Height (cm)')
plt.xlabel('Gender')
plt.legend()
plt.grid()
plt.show()

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pointbiserialr

# Sample data
gender = np.array([0, 1, 0, 1, 0, 1, 1, 0, 1, 0]) # Binary variable
# Multiple continuous variables
heights = np.array([175, 160, 180, 155, 178, 162, 165, 172, 159, 170]) # Heights
weights = np.array([70, 50, 80, 45, 75, 55, 60, 68, 52, 73]) # Weights
ages = np.array([23, 25, 22, 24, 21, 26, 27, 20, 29, 28]) # Ages
exam_marks = np.array([45, 54, 65, 70, 45, 67, 56, 43, 47, 54])
# List of continuous variables
continuous_variables = [heights, weights, ages, exam_marks]
variable_names = ['Height', 'Weight', 'Age', 'Exam_Marks']
r_pb_values = []

# Calculate Point-Biserial Correlation Coefficient for each continuous variable
for variable in continuous_variables:
    # Calculate Point-Biserial Correlation Coefficient using scipy
    r_pb, _ = pointbiserialr(gender, variable)
    r_pb_values.append(r_pb)

# Define correlation categories
categories = ['Weak', 'Moderate', 'Strong']
category_values = {category: [] for category in categories}

# Categorize the correlation values
for name, r_pb in zip(variable_names, r_pb_values):
    if abs(r_pb) < 0.3:
        category_values['Weak'].append((name, r_pb))
    elif abs(r_pb) < 0.5:
        category_values['Moderate'].append((name, r_pb))
    else:
        category_values['Strong'].append((name, r_pb))

```

```

# Prepare data for plotting
category_labels = []
category_coefficients = []

for category, values in category_values.items():
    for name, coeff in values:
        category_labels.append(name)
        category_coefficients.append(coeff)

# Plotting the Point-Biserial Correlation Coefficients categorized by strength
plt.figure(figsize=(10, 6))
sns.barplot(x=category_labels, y=category_coefficients, palette='viridis')
plt.title('Point-Biserial Correlation Coefficients Categorized by Strength (Targeted Continuous Variables)')
plt.xlabel('Continuous Variables')
plt.ylabel('Point-Biserial Correlation Coefficient')
plt.ylim(-1, 1)

# Annotate the bars
for i, coeff in enumerate(category_coefficients):
    plt.text(i, coeff + 0.05, f'{coeff:.2f}', ha='center')

plt.axhline(0, color='gray', linewidth=0.8, linestyle='--') # Add a horizontal line at y=0
plt.show()

# Print categorized results
for category, values in category_values.items():
    print(f"\n{category} Correlations:")
    for name, r_pb in values:
        print(f"{name} - Point-Biserial Correlation Coefficient: {r_pb:.4f}")

-----

-----

-----

-----

-----

-----

-----

```

✓ Spearman's Rank Correlation

Spearman's rank correlation measures the strength and direction of the monotonic relationship between two variables. This means that as one variable increases, the other variable tends to either increase or decrease consistently, but not necessarily at a constant rate.

Range: The Spearman correlation coefficient ρ ranges from -1 to 1:

- **+1:** A perfect positive monotonic relationship (as one variable increases, the other also increases).
- **-1:** A perfect negative monotonic relationship (as one variable increases, the other decreases).
- **0:** No monotonic relationship between the variables.

When to Use:

- **Ordinal Data:** Suitable for data measured on an ordinal scale (i.e., ranked data).
- **Non-Normal Distribution:** Useful when the data does not meet normality assumptions.
- **Non-Linear Relationships:** Effective for capturing monotonic relationships even if they are not linear.

Calculation Steps:

1. **Rank the Data:** Convert the raw data values into ranks. If there are ties, assign the average rank to the tied values.
2. **Compute the Difference in Ranks:** For each pair of observations, calculate the difference between the ranks of the two variables.
3. **Use the Formula:**

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where:

- (d_i) is the difference between the ranks of each observation pair.
- (n) is the number of observations.

Example: Let's assume we have a dataset of two variables (X) and (Y) , and we want to determine their Spearman correlation. We would follow the steps outlined above to calculate ρ .

In statistics, **Spearman's rank correlation coefficient** or **Spearman's ρ** , named after Charles Spearman, and often denoted by the Greek letter ρ or as r_s , is a nonparametric measure of rank correlation (statistical dependence between the rankings of two variables). It assesses how well the relationship between two variables can be described using a monotonic function.

The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables. While Pearson's correlation assesses linear relationships,

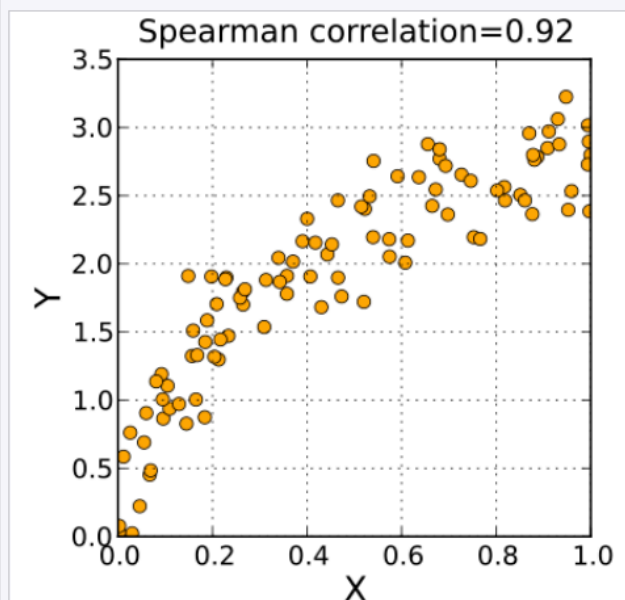
Spearman's correlation assesses monotonic relationships (whether linear or not).

If there are no repeated data values, a perfect Spearman correlation of +1 or -1 occurs when each of the variables is a perfect monotone function of the other.

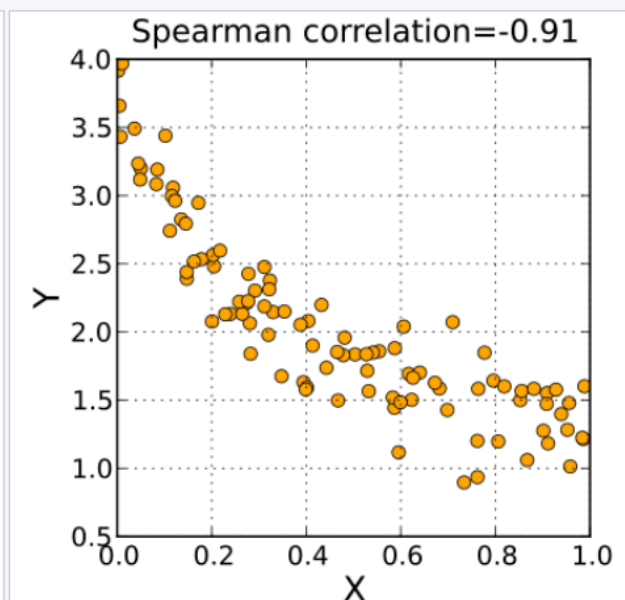
Intuitively, the Spearman correlation between two variables will be high when observations have a similar (or identical for a correlation of 1) rank (i.e., relative position label of the observations within the variable: 1st, 2nd, 3rd, etc.) between the two variables, and low when observations have a dissimilar (or fully opposed for a correlation of -1) rank between the two variables.

Spearman's coefficient is appropriate for both continuous and discrete ordinal variables. Both Spearman's ' ρ ' and Kendall's ' τ ' can be formulated as special cases of a more general correlation coefficient.

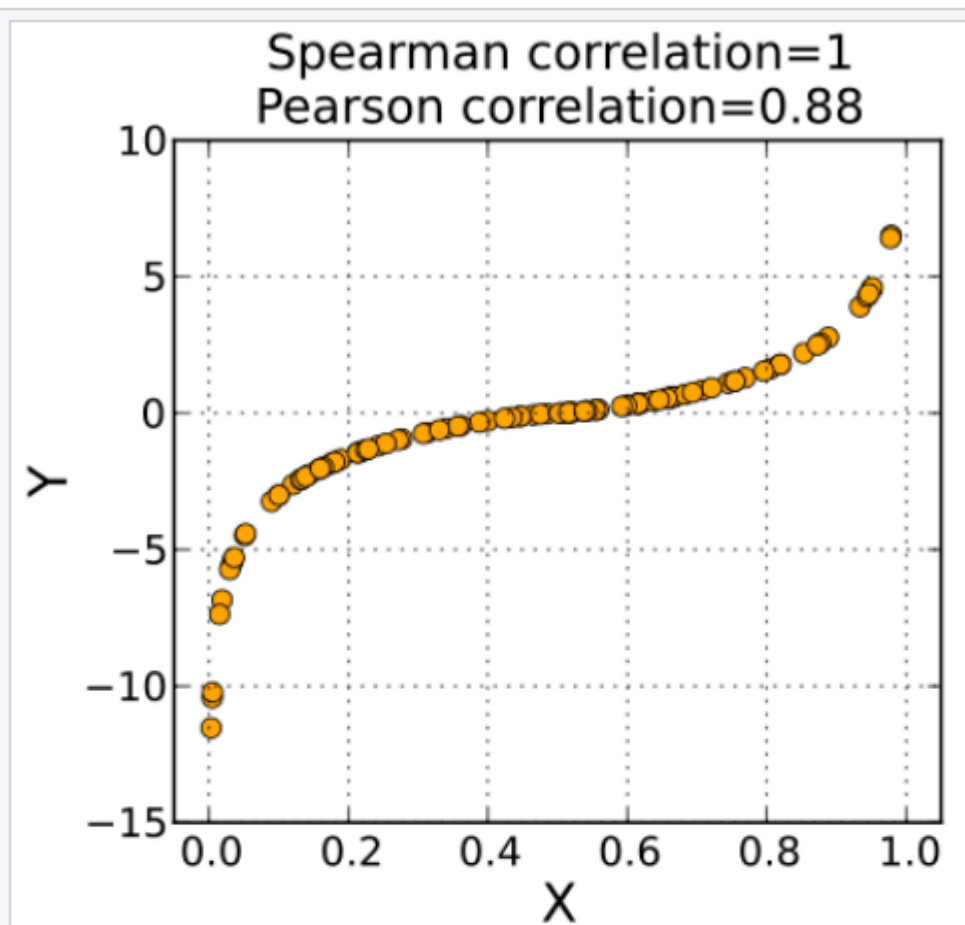
Positive and negative Spearman rank correlations



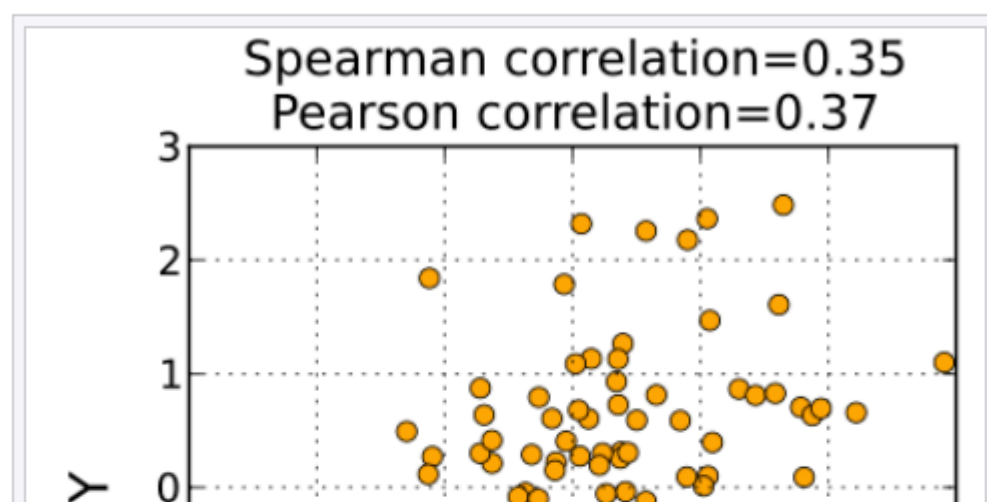
A positive Spearman correlation coefficient corresponds to an increasing monotonic trend between X and Y .

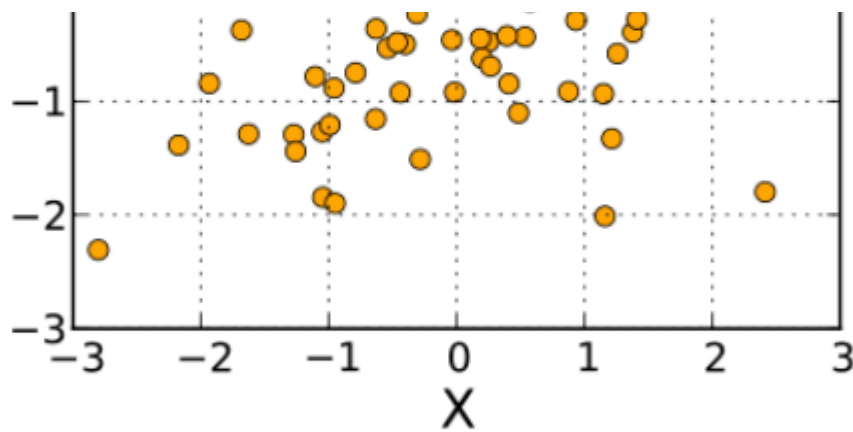



A negative Spearman correlation coefficient corresponds to a decreasing monotonic trend between X and Y .



A Spearman correlation of 1 results when the two variables being compared are monotonically related, even if their relationship is not linear. This means that all data points with greater x values than that of a given data point will have greater y values as well. In contrast, this does not give a perfect Pearson correlation.





When the data are roughly elliptically distributed and there are no prominent outliers, the Spearman correlation and Pearson correlation give similar values. 

✓ Custom Spearman's Rank Correlation

```

# Sample data
X = [10, 20, 30, 40, 50]
Y = [200, 150, 100, 250, 300]

# X = [ 106 , 100 , 86 , 101 , 99 , 103 , 97 , 113 , 112 ,110 ]

# Y = [ 7 , 27, 2, 50, 28, 29, 20, 12, 6, 17]

# Step 1: Rank the data
def rank_data(data):
    sorted_data = sorted((value, index) for index, value in enumerate(data))
    ranks = [0] * len(data)
    for rank, (value, index) in enumerate(sorted_data, start=1):
        ranks[index] = rank
    return ranks

rank_X = rank_data(X)
rank_Y = rank_data(Y)

# Step 2: Calculate d_i and d_i^2
d_i = [rank_X[i] - rank_Y[i] for i in range(len(X))]
d_i_squared = [d**2 for d in d_i]

# Step 3: Apply Spearman's formula
n = len(X)
rho = 1 - (6 * sum(d_i_squared)) / (n * (n**2 - 1))

print("Custom Spearman's Rank Correlation:", rho)

```

✓ Scipy Spearman's Rank Correlation

```

from scipy.stats import spearmanr

# Sample data
X = [10, 20, 30, 40, 50]
Y = [200, 150, 100, 250, 300]

# X = [ 106 , 100 , 86 , 101 , 99 , 103 , 97 , 113 , 112 ,110 ]
# Y = [ 7 , 27, 2, 50, 28, 29, 20, 12, 6, 17]
# Using scipy to calculate Spearman's Rank Correlation
rho, p_value = spearmanr(X, Y)

print("Built-in Spearman's Rank Correlation:", rho)
print("P-value:", p_value)

```

✓ Multiple X's with Y (One at a time)

```
from scipy.stats import spearmanr

# Sample data: multiple X variables and a single Y
X_list = [
    [10, 20, 30, 40, 50], # First X variable
    [15, 25, 35, 45, 55], # Second X variable
    [5, 10, 15, 20, 25]   # Third X variable
]
Y = [200, 150, 100, 250, 300]

# Calculate Spearman's Rank Correlation for each X with Y
correlations = []
for X in X_list:
    rho, p_value = spearmanr(X, Y)
    correlations.append((rho, p_value))

# Output the results
for i, (rho, p_value) in enumerate(correlations):
    print(f"Spearman's Rank Correlation for X{i+1} and Y: {rho}, P-value: {p_valu
```

✓ Multiple X's with Y (One at a time) with Visualizations

```
import matplotlib.pyplot as plt
from scipy.stats import spearmanr

# Sample data: multiple X variables and a single Y
X_list = [
    [10, 20, 30, 40, 50], # First X variable
    [15, 25, 35, 45, 55], # Second X variable
    [5, 10, 15, 20, 25]   # Third X variable
]
Y = [200, 150, 100, 250, 300]

# Step 1: Calculate Spearman's Rank Correlation for each X with Y
correlations = []
for X in X_list:
    rho, p_value = spearmanr(X, Y)
    correlations.append((rho, p_value))

# Extracting correlation coefficients and p-values
correlation_values = [rho for rho, _ in correlations]
p_values = [p for _, p in correlations]

# Step 2: Visualize results

# Scatter Plots
```

```

fig, axes = plt.subplots(1, len(X_list), figsize=(15, 5))
fig.suptitle("Scatter Plots of X Variables vs Y")

for i, X in enumerate(X_list):
    axes[i].scatter(X, Y, color='blue', edgecolor='black')
    axes[i].set_title(f"X{i+1} vs Y")
    axes[i].set_xlabel(f"X{i+1}")
    axes[i].set_ylabel("Y")

plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # Adjust layout

# Bar Chart for Correlation Coefficients
plt.figure(figsize=(8, 5))
plt.bar([f"X{i+1}" for i in range(len(X_list))], correlation_values, color='skybl
plt.title("Spearman's Rank Correlation Coefficients")
plt.xlabel("X Variables")
plt.ylabel("Correlation Coefficient ( $\rho$ )")
plt.ylim(-1, 1) # Spearman's correlation range is from -1 to 1
plt.axhline(0, color='black', linestyle='--', linewidth=0.5) # Add a horizontal

# Display p-values on the bar chart
for i, v in enumerate(correlation_values):
    plt.text(i, v + 0.05, f"P-value: {p_values[i]:.4f}", ha='center')

plt.show()

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import spearmanr

# Generate random datasets with varying strengths of correlation
np.random.seed(0)
x = np.random.rand(1000) * 100 # Random values scaled to 100

# y_weak: Another set of 1000 random values, representing a weak correlation (com
""" y_moderate: Creates a dataset that has a moderate correlation with x.
    It combines a linear relationship ( $x * 0.5$ ) with some random noise (n
    This means that y_moderate will generally increase with x, but with a
# y_strong: Similar to y_moderate, but it introduces a stronger linear relationsh

# Weak correlation dataset
y_weak = np.random.rand(1000) * 100

# Moderate correlation dataset (introducing some dependency)
y_moderate = x * 0.5 + np.random.rand(1000) * 50

# Strong correlation dataset (more direct dependency)
y_strong = x * 0.8 + np.random.rand(1000) * 10

# Calculate Spearman's rank correlation for each
rho_weak, _ = spearmanr(x, y_weak)

```



```

rho_moderate, _ = spearmanr(x, y_moderate)
rho_strong, _ = spearmanr(x, y_strong)

# Plotting the results
plt.figure(figsize=(18, 6))

# Weak correlation
plt.subplot(1, 3, 1)
sns.scatterplot(x=x, y=y_weak, alpha=0.6, edgecolor="black", color="pink")
plt.title(f"Weak Correlation:  $\rho = \{\text{rho\_weak:.4f}\}$ ", fontsize=12)
plt.xlabel('X Values')
plt.ylabel('Y Values')
plt.grid(visible=True, linestyle='--', alpha=0.5)

# Moderate correlation
plt.subplot(1, 3, 2)
sns.scatterplot(x=x, y=y_moderate, alpha=0.6, edgecolor="black", color="green")
plt.title(f"Moderate Correlation:  $\rho = \{\text{rho\_moderate:.4f}\}$ ", fontsize=12)
plt.xlabel('X Values')
plt.ylabel('Y Values')
plt.grid(visible=True, linestyle='--', alpha=0.5)

# Strong correlation
plt.subplot(1, 3, 3)
sns.scatterplot(x=x, y=y_strong, alpha=0.6, edgecolor="black", color="red")
plt.title(f"Strong Correlation:  $\rho = \{\text{rho\_strong:.4f}\}$ ", fontsize=12)
plt.xlabel('X Values')
plt.ylabel('Y Values')
plt.grid(visible=True, linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()

```

✓ How rank() Works in Pandas

When you call the `rank()` method on a pandas Series (like `data['X']` or `data['Y']`), the method performs the following operations:

Sorting:

The values in the Series are sorted in ascending order. The ranks are assigned based on the position of each value in this sorted list.

Assigning Ranks: Ranks are assigned as follows:

The smallest value gets a rank of 1.

The second smallest value gets a rank of 2, and so on.

If there are duplicate values (ties), they are assigned the average rank of their positions. For instance, if two values are tied for rank 2, both will receive a rank of 2.5 (average of ranks 2 and

3).

Return a New Series: The `rank()` function returns a new Series containing the ranks corresponding to the original values.

Example Walkthrough

Let's take a specific example to see how it works:

Example Data Suppose we have the following values in X:

Index X

0 20

1 10

2 30

3 20

Step-by-Step Process Initial Series: The original values in the Series `data['X']` are [20, 10, 30, 20].

Sorting the Values: The values are sorted:

Sorted values: [10, 20, 20, 30] Assigning Ranks:

10: Rank 1 (smallest)

20: There are two occurrences of 20. They are tied, so they will share the average rank:

The ranks for both 20s are calculated as follows:

Their positions in the sorted list are 2 and 3 (ranks are 2 and 3), so:

Average rank for 20 = $(2 + 3) / 2 = 2.5$

30: Rank 4 (largest)

Final Ranks

After the ranking process, the ranks assigned to the original values in X are:

Index X Rank (X_rank)

0 20 2.5

1 10 1

2 30 4

3 20 2.5

Code Implementation

In your original code, the lines:

```
data['X_rank'] = data['X'].rank()
```

```
data['Y_rank'] = data['Y'].rank()
```

are executing this ranking process for the columns X and Y in the DataFrame data. After these lines run, the DataFrame data will include two new columns: X_rank and Y_rank, which contain the ranks of the original values in X and Y, respectively.

Importance in Spearman's Rank Correlation

The ranks calculated in this way are then used to compute Spearman's rank correlation coefficient (ρ), which assesses the strength and direction of the relationship between the two variables. Since Spearman's correlation is based on ranks rather than raw data values, it is particularly useful when dealing with non-parametric data or when the relationship may not be linear.

This makes the ranking process a crucial step in preparing the data for correlation analysis.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import spearmanr

# Generate a random dataset with 1000 data points
np.random.seed(0)
x = np.random.rand(1000) * 100 # Random values scaled to 100
y = np.random.rand(1000) * 100 # Random values scaled to 100

# print(x[:15])
print("-----")
# print(y[:15])
# Create a DataFrame
data = pd.DataFrame({'X': x, 'Y': y})

# Calculate Spearman's rank correlation
rho, p_value = spearmanr(data['X'], data['Y'])

# Rank the data points
data['X_rank'] = data['X'].rank()
data['Y_rank'] = data['Y'].rank()

# Define color categories based on the rank correlation similarity within section
# Weak, Medium, Strong classifications for illustrative purposes (artificial cate
conditions = [
    (abs(data['X_rank'] - data['Y_rank']) < 100), # Strong correlation (close ra
    (abs(data['X_rank'] - data['Y_rank']) < 300), # Medium correlation (moderate
    (abs(data['X_rank'] - data['Y_rank']) >= 300) # Weak correlation (large rank
]
choices = ['red', 'green', 'pink']
data['color'] = np.select(conditions, choices)
```

```

print("-----")
# Visualization
plt.figure(figsize=(10, 6))
for color, label in zip(['red', 'green', 'pink'], ['Strong', 'Medium', 'Weak']):
    subset = data[data['color'] == color]
    sns.scatterplot(x=subset['X'], y=subset['Y'], color=color, label=label, alpha

plt.title(f"Spearman's Rank Correlation:  $\rho = \{{\rho:.4f}\}$ ", fontsize=14)
plt.xlabel('X Values')
plt.ylabel('Y Values')
plt.legend(title="Correlation Level")
plt.grid(visible=True, linestyle='--', alpha=0.5)
plt.show()

```

Spearman's Rank Correlation Coefficient

1. Definition

Spearman's Rank Correlation Coefficient, denoted as ρ , is a nonparametric measure of rank correlation. It assesses how well the relationship between two variables can be described using a monotonic function. Unlike Pearson's correlation, which measures linear relationships, Spearman's correlation evaluates whether the relationship is monotonic, meaning it either consistently increases or decreases but does not need to be linear.

2. Mathematical Formula

The Spearman correlation is calculated based on the ranks of the data rather than the actual values. The formula is:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

Where:

- (d_i) is the difference between the ranks of each observation pair,
- (n) is the number of observations.

3. When to Use Spearman's Correlation

- **Data Type:** It is suitable for ordinal data (data that can be ranked) or continuous data that does not meet the assumptions of normality (i.e., non-normally distributed).
- **Monotonic Relationships:** Use it when you suspect a monotonic relationship rather than a linear one.

4. Understanding Jackknife Resampling

The **jackknife resampling technique** is a method for estimating the precision of sample statistics. By systematically leaving out one observation at a time, it creates a pseudo-sample. For Spearman's correlation, jackknife pseudo-values help assess the stability of the correlation estimate and construct confidence intervals.

5. Confidence Intervals for Spearman's (ρ)

Purpose

A confidence interval provides a range of values that is likely to contain the true population parameter (in this case, Spearman's correlation) with a specified level of confidence (commonly 95%).

Wilks' Theorem

The confidence interval can be constructed using Wilks' theorem, which relates the distribution of the test statistic to a chi-square distribution. The confidence interval is defined by:

$$\left[\theta : \frac{(\sum_{i=1}^n (Z_i - \theta))^2}{\sum_{i=1}^n (Z_i - \theta)^2} \leq \chi_{1,\alpha}^2 \right]$$

Where:

- (Z_i) are the jackknife pseudo-values,
- $\chi_{1,\alpha}^2$ is the α quantile of a chi-square distribution with one degree of freedom,
- θ represents the Spearman correlation coefficient.

6. Calculation Steps

1. **Calculate Spearman's correlation** for your data.
2. **Compute jackknife pseudo-values** by excluding each observation one at a time and recalculating the correlation.
3. **Use the Wilks' theorem formula** to derive the lower and upper bounds of the confidence interval.

7. Interpreting the Results

- A **correlation coefficient** close to +1 indicates a strong positive monotonic relationship, while a value close to -1 indicates a strong negative monotonic relationship.
- A correlation around 0 suggests little to no monotonic relationship between the variables.
- The **confidence interval** provides insight into the reliability of the correlation estimate. If the interval does not include 0, the correlation is considered statistically significant.

8. Applications of Spearman's Correlation

Spearman's rank correlation is widely used in various fields such as:

- **Social Sciences:** Assessing relationships between ranked preferences or scores.
- **Biology:** Evaluating the correlation between ranked measurements in ecological studies.
- **Economics:** Understanding relationships between ranked economic indicators.

Summary

Understanding Spearman's rank correlation and its confidence intervals enables researchers to explore relationships between variables that do not meet the assumptions required for parametric tests. This nonparametric approach is powerful for analyzing ordinal data and establishing whether variables tend to increase or decrease together.

✓ Wilks' Theorem

Start coding or [generate](#) with AI.

Methods for Determining the Significance of Spearman's Rank Correlation Coefficient (ρ)