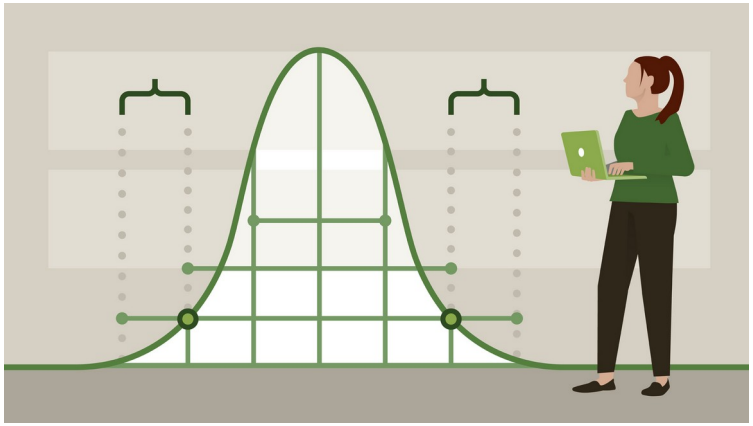
Correlation Coefficients

Tushar B. Kute,
http://tusharkute.com



Correlation Coefficients

- Correlation coefficients quantify the association between variables or features of a dataset. These statistics are of high importance for science and technology, and Python has great tools that you can use to calculate them. SciPy, NumPy, and Pandas correlation methods are fast, comprehensive, and well-documented.
- We will learn:
  - What Pearson, Spearman, and Kendall correlation coefficients are
  - How to use SciPy, NumPy, and Pandas correlation functions
  - How to visualize data, regression lines, and correlation matrices with Matplotlib

What is correlation ?

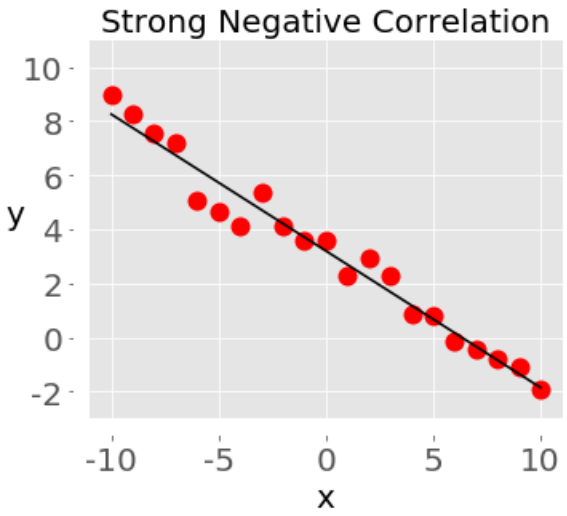
- Statistics and data science are often concerned about the relationships between two or more variables (or features) of a dataset. Each data point in the dataset is an observation, and the features are the properties or attributes of those observations.
- Every dataset you work with uses variables and observations. For example, you might be interested in understanding the following:
  - How the height of basketball players is correlated to their shooting accuracy
  - Whether there's a relationship between employee work experience and salary
  - What mathematical dependence exists between the population density and the gross domestic product of different countries

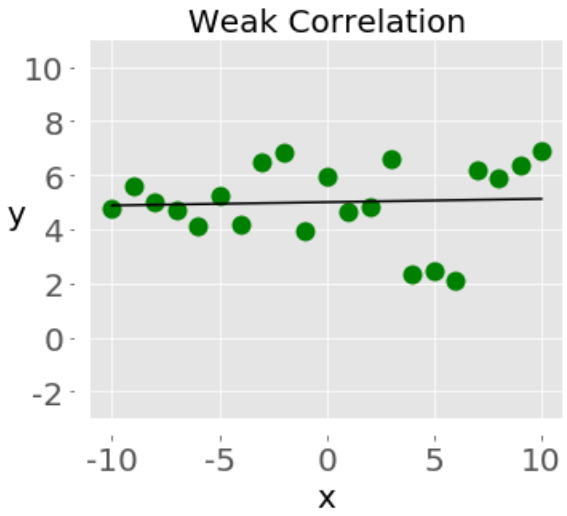
What is correlation ?

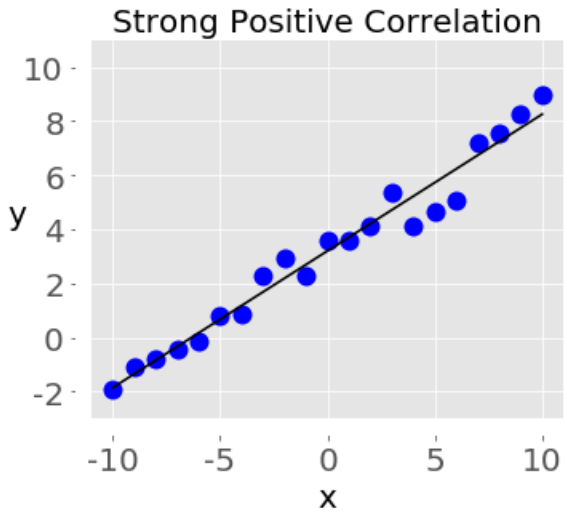
Name	Years of Experience	Annual Salary
Ann	30	120,000
Rob	21	105,000
Tom	19	90,000
Ivy	10	82,000

- In this table, each row represents one observation, or the data about one employee (either Ann, Rob, Tom, or Ivy). Each column shows one property or feature (name, experience, or salary) for all the employees.

Forms of correlation



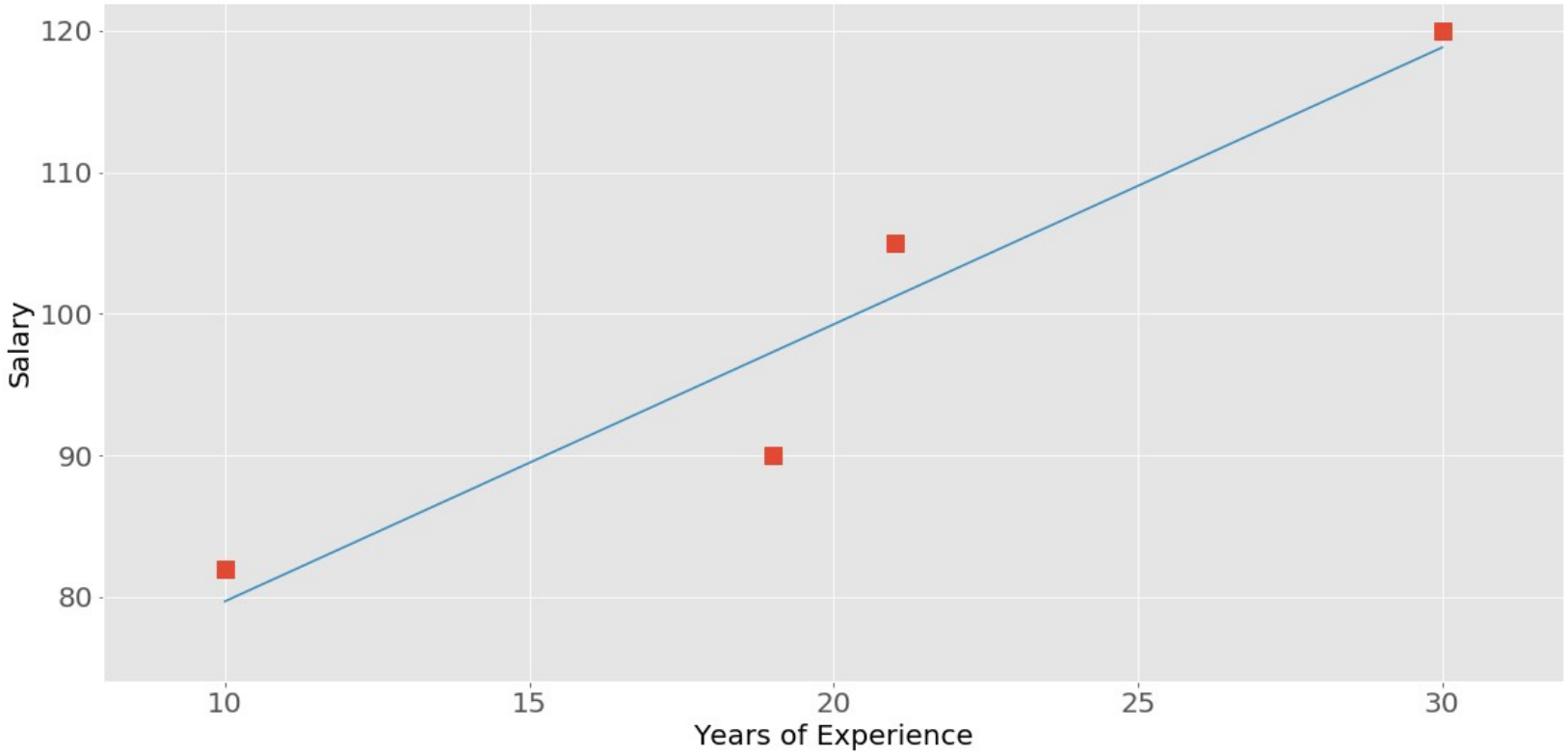




Forms of correlation

- Negative correlation (red dots): In the plot on the left, the y values tend to decrease as the x values increase. This shows strong negative correlation, which occurs when large values of one feature correspond to small values of the other, and vice versa.
- Weak or no correlation (green dots): The plot in the middle shows no obvious trend. This is a form of weak correlation, which occurs when an association between two features is not obvious or is hardly observable.
- Positive correlation (blue dots): In the plot on the right, the y values tend to increase as the x values increase. This illustrates strong positive correlation, which occurs when large values of one feature correspond to large values of the other, and vice versa.

Example: Employee table



Correlation Techniques

- There are several statistics that you can use to quantify correlation. We will be learning about three correlation coefficients:
  - Pearson's r
  - Spearman's rho
  - Kendall's tau
- Pearson's coefficient measures linear correlation, while the Spearman and Kendall coefficients compare the ranks of data.
- There are several NumPy, SciPy, and Pandas correlation functions and methods that you can use to calculate these coefficients.
- You can also use Matplotlib to conveniently illustrate the results.

Basic with numpy

```
import numpy as np
import scipy.stats
x = np.arange(10, 20)
```

```
y = np.array([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
```

```
r = np.corrcoef(x, y)
```

```
r
```

```
array([[1.        , 0.75864029],
       [0.75864029, 1.        ]])
```

```
r[0,1]
```

```
0.7586402890911867
```

```
r[1,0]
```

```
0.7586402890911869
```

	x	y
x	1.00	0.76
y	0.76	1.00

What sort of correlation ?

- The values on the main diagonal of the correlation matrix (upper left and lower right) are equal to 1.
- The upper left value corresponds to the correlation coefficient for x and x, while the lower right value is the correlation coefficient for y and y. They are always equal to 1.
- However, what you usually need are the lower left and upper right values of the correlation matrix.
- These values are equal and both represent the Pearson correlation coefficient for x and y. In this case, it’s approximately 0.76.

Correlation with scipy

```
import numpy as np
import scipy.stats
```

```
x = np.arange(10, 20)
y = np.array([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
```

```
scipy.stats.pearsonr(x, y)    # Pearson's r
```

```
(0.7586402890911869, 0.010964341301680829)
```

```
scipy.stats.spearmanr(x, y)   # Spearman's rho
```

```
SpearmanrResult(correlation=0.9757575757575757, pvalue=1.4675461874042197e-06)
```

```
scipy.stats.kendalltau(x, y)  # Kendall's tau
```

```
KendalltauResult(correlation=0.9111111111111111, pvalue=2.9761904761904762e-05)
```

```
scipy.stats.pearsonr(x, y)[0]    # Pearson's r, p-value
```

```
0.7586402890911869
```

Correlation with scipy

```
scipy.stats.pearsonr(x, y)[0]    # Pearson's r, p-value
```

```
0.7586402890911869
```

```
scipy.stats.spearmanr(x, y).correlation
```

```
0.9757575757575757
```

```
scipy.stats.kendalltau(x, y).correlation
```

```
0.9111111111111111
```

```
r, p = scipy.stats.pearsonr(x, y)
```

```
r
```

```
0.7586402890911869
```

Correlation with Pandas

```
import pandas as pd
import numpy as np
x = pd.Series(range(10, 20))
```

```
y = pd.Series([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
```

```
y.corr(x)
```

```
0.7586402890911869
```

```
x.corr(y, method='spearman')  # Spearman's rho
```

```
0.9757575757575757
```

```
x.corr(y, method='kendall')   # Kendall's tau
```

```
0.9111111111111111
```

Linear Correlation

- Linear correlation measures the proximity of the mathematical relationship between variables or dataset features to a linear function.
- If the relationship between the two features is closer to some linear function, then their linear correlation is stronger and the absolute value of the correlation coefficient is higher.

Pearson Correlation

- Consider a dataset with two features: x and y. Each feature has n values, so x and y are n-tuples. Say that the first value x_1 from x corresponds to the first value y_1 from y, the second value x_2 from x to the second value y_2 from y, and so on. Then, there are n pairs of corresponding values: (x_1, y_1), (x_2, y_2), and so on. Each of these x-y pairs represents a single observation.
- The Pearson (product-moment) correlation coefficient is a measure of the linear relationship between two features. It's the ratio of the covariance of x and y to the product of their standard deviations. It's often denoted with the letter r and called Pearson's r. You can express this value mathematically with this equation:
- $r = \Sigma_i((x_i - \text{mean}(x))(y_i - \text{mean}(y))) \, (\sqrt{\Sigma_i}(x_i - \text{mean}(x))^2 \sqrt{\Sigma_i}(y_i - \text{mean}(y))^2)^{-1}$

Pearson Correlation

- The Pearson correlation coefficient can take on any real value in the range $-1 \leq r \leq 1$.
- The maximum value $r = 1$ corresponds to the case when there's a perfect positive linear relationship between x and y. In other words, larger x values correspond to larger y values and vice versa.
- The value $r > 0$ indicates positive correlation between x and y.
- The value $r = 0$ corresponds to the case when x and y are independent.
- The value $r < 0$ indicates negative correlation between x and y.
- The minimal value $r = -1$ corresponds to the case when there's a perfect negative linear relationship between x and y. In other words, larger x values correspond to smaller y values and vice versa.

Pearson Correlation

Pearson's r Value	Correlation Between x and y
equal to 1	perfect positive linear relationship
greater than 0	positive correlation
equal to 0	independent
less than 0	negative correlation
equal to -1	perfect negative linear relationship

Linear Regression

- Linear regression is the process of finding the linear function that is as close as possible to the actual relationship between features.
- In other words, you determine the linear function that best describes the association between the features. This linear function is also called the regression line.
- You can implement linear regression with SciPy. You'll get the linear function that best approximates the relationship between two arrays, as well as the Pearson correlation coefficient.

Practical Linear Regression

```
import numpy as np
import scipy.stats
x = np.arange(10, 20)
y = np.array([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
```

```
result = scipy.stats.linregress(x, y)
```

```
result.slope
```

```
7.4363636363636365
```

```
result.intercept
```

```
-85.92727272727274
```

```
result.rvalue
```

```
0.7586402890911869
```

```
result.pvalue
```

```
0.010964341301680825
```

More on Regression

```
xy = np.array([[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
               [2, 1, 4, 5, 8, 12, 18, 25, 96, 48]])
```

```
scipy.stats.linregress(xy)
```

```
LinregressResult(slope=7.4363636363636365, intercept=-85.92727272727
274, rvalue=0.7586402890911869, pvalue=0.010964341301680825, stderr=
2.257878767543913)
```

```
xy.T;
```

```
scipy.stats.linregress(xy.T)
```

```
LinregressResult(slope=7.4363636363636365, intercept=-85.92727272727
274, rvalue=0.7586402890911869, pvalue=0.010964341301680825, stderr=
2.257878767543913)
```

```
scipy.stats.linregress(np.arange(3), np.array([2, np.nan, 5]))
```

```
LinregressResult(slope=nan, intercept=nan, rvalue=nan, pvalue=nan, s
tderr=nan)
```

Using Multidimensional Data

```
xy = np.array([[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
               [2, 1, 4, 5, 8, 12, 18, 25, 96, 48]])
```

```
np.corrcoef(xy)
```

```
array([[1.        , 0.75864029],
       [0.75864029, 1.        ]])
```

```
xyz = np.array([[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
                [2, 1, 4, 5, 8, 12, 18, 25, 96, 48],
                [5, 3, 2, 1, 0, -2, -8, -11, -15, -16]])
```

```
np.corrcoef(xyz)
```

```
array([[ 1.        ,  0.75864029, -0.96807242],
       [ 0.75864029,  1.        , -0.83407922],
       [-0.96807242, -0.83407922,  1.        ]])
```

Using Pandas

```
x = pd.Series(range(10, 20))
```

```
y = pd.Series([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
```

```
z = pd.Series([5, 3, 2, 1, 0, -2, -8, -11, -15, -16])
```

```
xy = pd.DataFrame({'x-values': x, 'y-values': y})
```

```
xy.corr()
```

	x-values	y-values
x-values	1.00000	0.75864
y-values	0.75864	1.00000

Using Pandas

```
xyz = pd.DataFrame({'x-values': x, 'y-values': y, 'z-values': z})
```

```
xyz.corr()
```

	x-values	y-values	z-values
x-values	1.000000	0.758640	-0.968072
y-values	0.758640	1.000000	-0.834079
z-values	-0.968072	-0.834079	1.000000

Using Pandas

```
corr_matrix = xy.corr()
```

```
corr_matrix
```

	x-values	y-values
x-values	1.00000	0.75864
y-values	0.75864	1.00000

```
corr_matrix.at['x-values', 'y-values']
```

```
0.7586402890911869
```

```
corr_matrix.iat[0, 1]
```

```
0.7586402890911869
```

Using corrwith

```
xy.corrwith(z)
```

```
x-values   -0.968072
y-values   -0.834079
dtype: float64
```

```
x.corr(y, method='spearman')
```

```
0.9757575757575757
```

```
xyz.corr(method='spearman')
```

	x-values	y-values	z-values
x-values	1.000000	0.975758	-1.000000
y-values	0.975758	1.000000	-0.975758
z-values	-1.000000	-0.975758	1.000000

```
xy.corrwith(z, method='spearman')
```

```
x-values   -1.000000
y-values   -0.975758
```

```
xy.corrwith(z, method='spearman')
```

```
x-values   -1.000000
y-values   -0.975758
dtype: float64
```

```
x.corr(y, method='kendall')
```

```
0.911111111111111
```

```
xy.corr(method='kendall')
```

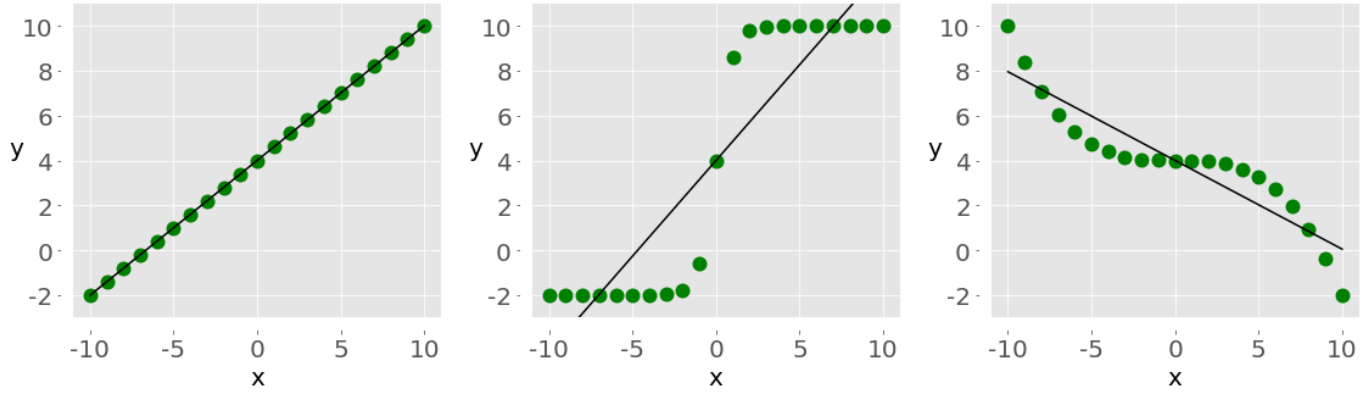
	x-values	y-values
x-values	1.000000	0.911111
y-values	0.911111	1.000000

```
xyz.corr(method='kendall')
```

Rank Correlation

- Rank correlation compares the ranks or the orderings of the data related to two variables or dataset features.
- If the orderings are similar, then the correlation is strong, positive, and high. However, if the orderings are close to reversed, then the correlation is strong, negative, and low.
- In other words, rank correlation is concerned only with the order of values, not with the particular values from the dataset.

Rank Correlation



- The left plot has a perfect positive linear relationship between x and y, so r = 1. The central plot shows positive correlation and the right one shows negative correlation. However, neither of them is a linear function, so r is different than –1 or 1.
- When you look only at the orderings or ranks, all three relationships are perfect! The left and central plots show the observations where larger x values always correspond to larger y values. This is perfect positive rank correlation. The right plot illustrates the opposite case, which is perfect negative rank correlation.

The Spearman Correlation

- The Spearman correlation coefficient between two features is the Pearson correlation coefficient between their rank values.
- It's calculated the same way as the Pearson correlation coefficient but takes into account their ranks instead of their values. It's often denoted with the Greek letter rho (ρ) and called Spearman's rho.
- Say you have two n-tuples, x and y, where (x_1, y_1), (x_2, y_2), … are the observations as pairs of corresponding values.
- You can calculate the Spearman correlation coefficient ρ the same way as the Pearson coefficient. You'll use the ranks instead of the actual values from x and y.

The Spearman Correlation

- It can take a real value in the range $-1 \leq \rho \leq 1$.
- Its maximum value $\rho = 1$ corresponds to the case when there's a monotonically increasing function between x and y. In other words, larger x values correspond to larger y values and vice versa.
- Its minimum value $\rho = -1$ corresponds to the case when there's a monotonically decreasing function between x and y. In other words, larger x values correspond to smaller y values and vice versa.

Kendall Correlation Coefficient

- Let's start again by considering two n-tuples, x and y. Each of the x-y pairs (x_1, y_1), (x_2, y_2), … is a single observation. A pair of observations (x_i, y_i) and (x_j, y_j), where $i < j$, will be one of three things:
  - concordant if either ($x_i > x_j$ and $y_i > y_j$) or ($x_i < x_j$ and $y_i < y_j$)
  - discordant if either ($x_i < x_j$ and $y_i > y_j$) or ($x_i > x_j$ and $y_i < y_j$)
  - neither if there's a tie in x ($x_i = x_j$) or a tie in y ($y_i = y_j$)
- The Kendall correlation coefficient compares the number of concordant and discordant pairs of data.
- This coefficient is based on the difference in the counts of concordant and discordant pairs relative to the number of x-y pairs. It's often denoted with the Greek letter tau (τ) and called Kendall's tau.

Kendall Correlation Coefficient

- It can take a real value in the range $-1 \leq \tau \leq 1$.
- Its maximum value $\tau = 1$ corresponds to the case when the ranks of the corresponding values in x and y are the same. In other words, all pairs are concordant.
- Its minimum value $\tau = -1$ corresponds to the case when the rankings in x are the reverse of the rankings in y. In other words, all pairs are discordant.

Ranking with scipy

```
import numpy as np
import scipy.stats
x = np.arange(10, 20)
y = np.array([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
z = np.array([5, 3, 2, 1, 0, -2, -8, -11, -15, -16])
```

```
scipy.stats.rankdata(x)
```

```
array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

```
scipy.stats.rankdata(y)
```

```
array([ 2.,  1.,  3.,  4.,  5.,  6.,  7.,  8., 10.,  9.])
```

```
scipy.stats.rankdata(z)
```

```
array([10.,  9.,  8.,  7.,  6.,  5.,  4.,  3.,  2.,  1.])
```

```
scipy.stats.rankdata([8, 2, 0, 2])
```

```
array([4. , 2.5, 1. , 2.5])
```

Implementation

```
result = scipy.stats.kendalltau(x, y)
```

```
result
```

```
KendalltauResult(correlation=0.9111111111111111, pvalue=2.9761904761904762e-05)
```

```
result.correlation
```

```
0.9111111111111111
```

```
tau, p = scipy.stats.kendalltau(x, y)
```

```
tau
```

```
0.9111111111111111
```

Scipy with 2D

```
xyz = np.array([[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
                [2, 1, 4, 5, 8, 12, 18, 25, 96, 48],
                [5, 3, 2, 1, 0, -2, -8, -11, -15, -16]])
```

```
corr_matrix = np.corrcoef(xyz).round(decimals=2)
```

```
corr_matrix
```

```
array([[ 1.  ,  0.76, -0.97],
       [ 0.76,  1.  , -0.83],
       [-0.97, -0.83,  1.  ]])
```

Scipy with 2D

```
corr_matrix, p_matrix = scipy.stats.spearmanr(xyz, axis=1)
```

```
corr_matrix
```

```
array([[ 1.        ,  0.97575758, -1.        ],
       [ 0.97575758,  1.        , -0.97575758],
       [-1.        , -0.97575758,  1.        ]])
```

```
p_matrix
```

```
array([[6.64689742e-64, 1.46754619e-06, 6.64689742e-64],
       [1.46754619e-06, 6.64689742e-64, 1.46754619e-06],
       [6.64689742e-64, 1.46754619e-06, 6.64689742e-64]])
```

Using pandas

```
xy.corr(method='kendall')
```

	x-values	y-values
x-values	1.000000	0.911111
y-values	0.911111	1.000000

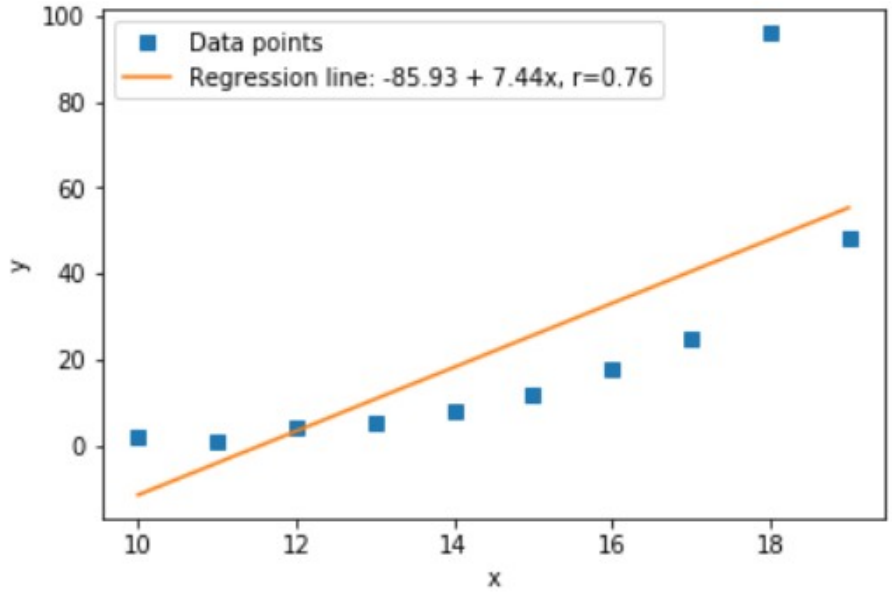
```
xyz.corr(method='kendall')
```

	x-values	y-values	z-values
x-values	1.000000	0.911111	-1.000000
y-values	0.911111	1.000000	-0.911111
z-values	-1.000000	-0.911111	1.000000

Visualization of Regression Line

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(x, y, linewidth=0, marker='s', label='Data points')
ax.plot(x, intercept + slope * x, label=line)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend(facecolor='white')
```

```
<matplotlib.legend.Legend at 0x7f03ce882668>
```



Useful resources

- https://realpython.com
- https://scipy.org
- www.towardsdatascience.com
- www.medium.com

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com