scipy.stats.linregress

scipy.stats.**linregress**(*x*, *y=None*) [source]
(https://github.com/scipy/scipy/blob/v1.1.0/scipy/stats/_stats_mstats_common.py#L14-L121)

Calculate a linear least-squares regression for two sets of measurements.

Parameters: **x, y :** *array_like*

Two sets of measurements. Both arrays should have the same length. If only x is given (and y=None), then it must be a two-dimensional array where one dimension has length 2. The two sets of measurements are then found by splitting the array along the length-2 dimension.

Returns: **slope :** *float*

slope of the regression line

intercept : *float*

intercept of the regression line

rvalue : *float*

correlation coefficient

pvalue : *float*

two-sided p-value for a hypothesis test whose null hypothesis is that the slope is zero, using Wald Test with t-distribution of the test statistic.

stderr : *float*

Standard error of the estimated gradient.

See also:

scipy.optimize.curve_fit (scipy.optimize.curve_fit.html#scipy.optimize.curve_fit) Use non-linear least squares to fit a function to data.
scipy.optimize.leastsq (scipy.optimize.leastsq.html#scipy.optimize.leastsq) Minimize the sum of squares of a set of equations.

Examples

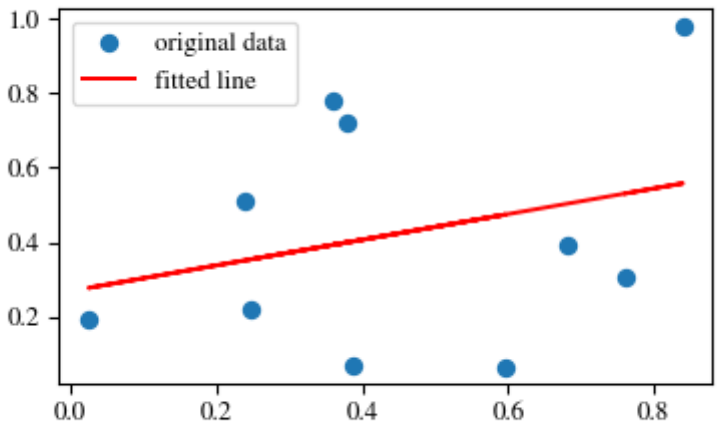
```
>>> import matplotlib.pyplot as plt
>>> from scipy import stats
>>> np.random.seed(12345678)
>>> x = np.random.random(10)
>>> y = np.random.random(10)
>>> slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
```

To get coefficient of determination (r_squared)

```
>>> print("r-squared:", r_value**2)
r-squared: 0.08040226853902833
```

Plot the data along with the fitted line

```
>>> plt.plot(x, y, 'o', label='original data')
>>> plt.plot(x, intercept + slope*x, 'r', label='fitted line')
>>> plt.legend()
>>> plt.show()
```



Previous topic

scipy.stats.weightedtau (scipy.stats.weightedtau.html)

Next topic

scipy.stats.theilslopes (scipy.stats.theilslopes.html)