

# scipy.stats.fisher\_exact

scipy.stats.fisher\_exact(*table*, *alternative*='two-sided')

[source]

Perform a Fisher exact test on a 2x2 contingency table.

**Parameters:**

**table** : *array\_like of ints*

A 2x2 contingency table. Elements must be non-negative integers.

**alternative** : {'two-sided', 'less', 'greater'}, optional

Defines the alternative hypothesis. The following options are available (default is 'two-sided'):

- 'two-sided'
- 'less': one-sided
- 'greater': one-sided

See the Notes for more details.

**Returns:**

**oddsratio** : *float*

This is prior odds ratio and not a posterior estimate.

**p\_value** : *float*

P-value, the probability of obtaining a distribution at least as extreme as the one that was actually observed, assuming that the null hypothesis is true.

**See also**

- [chi2\\_contingency](#)  
Chi-square test of independence of variables in a contingency table. This can be used as an alternative to [fisher\\_exact](#) when the numbers in the table are large.
- [barnard\\_exact](#)  
Barnard's exact test, which is a more powerful alternative than Fisher's exact test for 2x2 contingency tables.
- [boschloo\\_exact](#)  
Boschloo's exact test, which is a more powerful alternative than Fisher's exact test for 2x2 contingency tables.

**Notes**

*Null hypothesis and p-values*

The null hypothesis is that the input table is from the hypergeometric distribution with parameters (as used in [hypergeom](#))  $M = a + b + c + d$ ,  $n = a + b$  and  $N = a + c$ , where the input table is  $\begin{bmatrix} a, & b \\ c, & d \end{bmatrix}$ . This distribution has support  $\max(0, N + n - M) \leq x \leq \min(N, n)$ , or, in terms of the values in the input table,  $\min(0, a - d) \leq x \leq a + \min(b, c)$ .  $x$  can be interpreted as the upper-left element of a 2x2 table, so the tables in the distribution have form:

$$\begin{bmatrix} x & n - x \\ N - x & M - (n + N) + x \end{bmatrix}$$

For example, if:

$$\text{table} = \begin{bmatrix} 6 & 2 \\ 1 & 4 \end{bmatrix}$$

then the support is  $2 \leq x \leq 7$ , and the tables in the distribution are:

$$\begin{bmatrix} 2 & 6 \\ 5 & 0 \end{bmatrix} \quad \begin{bmatrix} 3 & 5 \\ 4 & 1 \end{bmatrix} \quad \begin{bmatrix} 4 & 4 \\ 3 & 2 \end{bmatrix} \quad \begin{bmatrix} 5 & 3 \\ 2 & 3 \end{bmatrix} \quad \begin{bmatrix} 6 & 2 \\ 1 & 4 \end{bmatrix} \quad \begin{bmatrix} 7 & 1 \\ 0 & 5 \end{bmatrix}$$

Search the docs ...

[Input and output \(`\_scipy.io`\)](#)

[Linear algebra  
\(`\_scipy.linalg`\)](#)

[Low-level BLAS functions  
\(`\_scipy.linalg.blas`\)](#)

[Low-level LAPACK functions  
\(`\_scipy.linalg.lapack`\)](#)

[BLAS Functions for Cython](#)  
[LAPACK functions for Cython](#)

[Interpolative matrix  
decomposition  
\(`\_scipy.linalg.interpolative`\)](#)

[Miscellaneous routines  
\(`\_scipy.misc`\)](#)

[Multidimensional image  
processing](#)

[Processing](#)[\( `scipy.ndimage` \)](#)[Orthogonal distance regression](#)[\( `scipy.odr` \)](#)[Optimization and root finding](#)[\( `scipy.optimize` \)](#)[Nonlinear solvers](#)[Cython optimize zeros API](#)[Signal processing](#)[\( `scipy.signal` \)](#)[Sparse matrices](#)[\( `scipy.sparse` \)](#)[Sparse linear algebra](#)[\( `scipy.sparse.linalg` \)](#)[Compressed sparse graph routines](#)[\( `scipy.sparse.csgraph` \)](#)[Spatial algorithms and data structures \( `scipy.spatial` \)](#)[Distance computations](#)[\( `scipy.spatial.distance` \)](#)[Special functions](#)[\( `scipy.special` \)](#)[Statistical functions](#)[\( `scipy.stats` \)](#)[Result classes](#)[Statistical functions for masked arrays](#)[\( `scipy.stats.mstats` \)](#)[Quasi-Monte Carlo submodule](#)[\( `scipy.stats.qmc` \)](#)[Low-level callback functions](#)

The probability of each table is given by the hypergeometric distribution `hypergeom.pmf(x, M, n, N)`. For this example, these are (rounded to three significant digits):

x	2	3	4	5	6	7
p	0.0163	0.163	0.408	0.326	0.0816	0.00466

These can be computed with:

```
>>> from scipy.stats import hypergeom
>>> table = np.array([[6, 2], [1, 4]])
>>> M = table.sum()
>>> n = table[0].sum()
>>> N = table[:, 0].sum()
>>> start, end = hypergeom.support(M, n, N)
>>> hypergeom.pmf(np.arange(start, end+1), M, n, N)
array([0.01631702, 0.16317016, 0.40792541, 0.32634033, 0.08158508,
       0.004662  ])
```

The two-sided p-value is the probability that, under the null hypothesis, a random table would have a probability equal to or less than the probability of the input table. For our example, the probability of the input table (where  $x = 6$ ) is 0.0816. The  $x$  values where the probability does not exceed this are 2, 6 and 7, so the two-sided p-value is  $0.0163 + 0.0816 + 0.00466 \approx 0.10256$ :

```
>>> from scipy.stats import fisher_exact
>>> oddsr, p = fisher_exact(table, alternative='two-sided')
>>> p
0.10256410256410257
```

The one-sided p-value for `alternative='greater'` is the probability that a random table has  $x \geq a$ , which in our example is  $x \geq 6$ , or  $0.0816 + 0.00466 \approx 0.08626$ :

```
>>> oddsr, p = fisher_exact(table, alternative='greater')
>>> p
0.08624708624708627
```

This is equivalent to computing the survival function of the distribution at  $x = 5$  (one less than  $x$  from the input table, because we want to include the probability of  $x = 6$  in the sum):

```
>>> hypergeom.sf(5, M, n, N)
0.08624708624708627
```

For `alternative='less'`, the one-sided p-value is the probability that a random table has  $x \leq a$ , (i.e.  $x \leq 6$  in our example), or  $0.0163 + 0.163 + 0.408 + 0.326 + 0.0816 \approx 0.9949$ :

```
>>> oddsr, p = fisher_exact(table, alternative='less')
>>> p
0.9953379953379957
```

This is equivalent to computing the cumulative distribution function of the distribution at  $x = 6$ :

```
>>> hypergeom.cdf(6, M, n, N)
0.9953379953379957
```

### Odds ratio

The calculated odds ratio is different from the one R uses. This SciPy implementation returns the (more common) “unconditional Maximum Likelihood Estimate”, while R uses the “conditional Maximum Likelihood Estimate”.

### Examples

Say we spend a few days counting whales and sharks in the Atlantic and Indian oceans. In the Atlantic ocean we find 8 whales and 1 shark, in the Indian ocean 2 whales and 5 sharks. Then our contingency table is:

	Atlantic	Indian
whales	8	2
sharks	1	5

We use this table to find the p-value:

```
>>> from scipy.stats import fisher_exact
>>> oddsratio, pvalue = fisher_exact([[8, 2], [1, 5]])
>>> pvalue
0.0349...
```

The probability that we would observe this or an even more imbalanced ratio by chance is about 3.5%. A commonly used significance level is 5%—if we adopt that, we can therefore conclude that our observed imbalance is statistically significant; whales prefer the Atlantic while sharks prefer the Indian ocean.

<< [scipy.stats.contingency.association](#)

[scipy.stats.barnard\\_exact](#) >>