独立 2 標本の並べ替え検定

青木繁伸

2020 年 3 月 17 日

1　目的

独立 2 標本データに対する並べ替え検定（permutation test）を行う。
サンプルサイズが大きい場合にはモンテカルロ法による方がよい。

2　使用法

```
import sys
sys.path.append("statlib")
from exact import permutation_test
permutation_test(x, y, FUNC, monte_carlo=False, loop=10000, verbose=True)
```

2.1　引数

x	第 1 群のデータ
y	第 2 群のデータ
FUNC	検定統計量を返す任意の検定関数。既存の関数でも自作関数でもよいが，検定統計量を戻り値で返すものでなければならない
monte_carlo	モンテカルロ法により並べ替え検定を行う場合に True にする。ただし，完全な並べ替え検定を行う回数が loop より少ない場合は，完全な並べ替え検定を行う
loop	モンテカルロ法の試行回数
verbose	必要最小限のプリント出力をする

2.2　戻り値

p 値

3　使用例

3.1　van der Waerden 検定

```
x = [2, 1, 3, 2, 5]
```

```
y = [4, 3, 2, 5, 4]

# van der Waerden 検定
import numpy as np
from scipy.stats import norm, rankdata

def vdw2_test(x, y):
  n1 = len(x) # 第一群のサンプルサイズ
  z = np.hstack((x, y)) # データベクトルを一つにまとめる
  n = len(z) # 合計したサンプルサイズ
  S = sum(norm.ppf((rankdata(z)[:n1])/(n+1))) # 第一群のデータに対する
      正規化得点の合計
  return S

import sys
sys.path.append("statlib")
from exact import permutation_test

a = permutation_test(x, y, vdw2_test)
```

```
Permutation test, using 'vdw2_test'
p value = 0.34127
```

3.2 *t* 検定

```
import numpy as np

def t_test(x, y):
    x = np.ravel(x)
    y = np.ravel(y)
    n_a = len(x)
    n_b = len(x)
    mean_a = np.mean(x)
    mean_b = np.mean(y)
    U_a = np.var(x, ddof=1)
    U_b = np.var(y, ddof=1)
    return (mean_a-mean_b)/np.sqrt(U_a/n_a+U_b/n_b)

b = permutation_test(x, y, t_test)
```

```
Permutation test, using 't_test'
p value = 0.37302
```

scipy.stats にある二標本検定関数は，統計量と p 値の二つの要素を持つタプルを返すので，以下のようなシュガーコート関数を定義してやればよい。

```
from scipy.stats import ttest_ind
```

```
def sugar(x, y):
    return ttest_ind(x, y)[0]

print(ttest_ind(x, y))
```

```
Ttest_indResult(statistic=-1.178511301977579, pvalue=0.272455713973288)
```

```
print(sugar(x, y))
```

```
-1.178511301977579
```

```
d = permutation_test(x, y, sugar)
```

```
Permutation test, using 'sugar'
p value = 0.37302
```

3.3 Brunner-Munzel 検定

```
# Brunner-Munzel test
def bm2_test(x, y):
    n1 = len(x)
    n2 = len(y)
    r = rankdata(np.hstack((x, y)))
    return (np.mean(r[n1:]) - (n2 + 1) / 2) / n1

x = [1,2,1,1,1,1,1,1,1,1,2,4,1,1]
y = [3,3,4,3,1,2,3,1,1,5,4]

np.random.seed(1234) # 乱数の種の設定（普通は必要ない）
e = permutation_test(x, y, bm2_test, monte_carlo=True, loop=20000)
```

```
Permutation test, using 'bm2_test', by 20000 times Monte Carlo simulations
p value = 0.00675
```

3.4 Mann-Whitney の U 検定

```
# Mann-Whitney U test
def mw2_test(x, y):
    n1 = len(x)
    n2 = len(y)
    r = rankdata(np.hstack((x, y)))
    R2 = sum(r[n1:])
    return n1 * n2 + n2 * (n2 + 1) / 2 - R2

np.random.seed(1234) # 乱数の種の設定（普通は必要ない）
f = permutation_test(x, y, mw2_test, monte_carlo=True, loop=20000)
```

```
Permutation test, using 'mw2_test', by 20000 times Monte Carlo simulations
p value = 0.00675
```