Nonparametric_statistics

December 28, 2023

```
[265]: from scipy import stats
       import numpy as np
       import random
       from matplotlib import pyplot as plt

       def mystatistics1(*data):
           return (np.mean(data[0]) - np.mean(data[1]))

       def mystatistics2(*data):
           return abs(np.mean(data[0]) - np.mean(data[1]))
```

0.1 Estimating the data from Amazon histogram

```
[211]: np.array([.63,.18,.07,.05,.07])*257
```

```
[211]: array([161.91,  46.26,  17.99,  12.85,  17.99])
```

```
[212]: black = [5]*162 + [4]*46 + [3]*18 + [2]*13 + [1]*18; len(black)
```

```
[212]: 257
```

```
[213]: np.array([.55,.16,.13,.07,.1])*114
```

```
[213]: array([62.7 , 18.24, 14.82,  7.98, 11.4 ])
```

```
[214]: blue = [5]*63 + [4]*18 + [3]*15 + [2]*8 + [1]*11; len(blue)
```

```
[214]: 115
```

0.2 Running the permutation test with 1- and 2-sided version

```
[233]: stats.permutation_test([black, blue], statistic=mystatistics1)
```

```
[233]: PermutationTestResult(statistic=0.2577228895279986, pvalue=0.0764,
       null_distribution=array([-0.03177127,  0.45911013,  0.09409575, …,
       -0.00659787,
               0.11926916, -0.22057182]))
```

```
[235]: stats.permutation_test([black, blue], statistic=mystatistics2)
```

```
[235]: PermutationTestResult(statistic=0.2577228895279986, pvalue=0.1408,
       null_distribution=array([0.01857554, 0.1450516 , 0.47169684, …, 0.08150905,
       0.16961597,
              0.08211808]))
```

0.3 Fake data: what if we 2x the group of customers who bought blue?

Assuming the same percentages of each mark.

```
[218]: stats.permutation_test([black, blue+blue], statistic=mystatistics2)
```

```
[218]: PermutationTestResult(statistic=0.2577228895279986, pvalue=0.0548,
       null_distribution=array([0.08470648, 0.01879547, 0.20005075, …, 0.05535442,
       0.15885637,
              0.12126544]))
```

0.4 Fake data: what if we take random subgroup of customers who bought black?

Random sample => should have approax. the same average.

```
[219]: small_black = random.sample(black, k=115)
       np.mean(black),np.mean(small_black)
```

```
[219]: (4.249027237354086, 4.252173913043478)
```

```
[220]: stats.permutation_test([small_black, blue], statistic=mystatistics2)
```

```
[220]: PermutationTestResult(statistic=0.26086956521739113, pvalue=0.2962,
       null_distribution=array([0.12173913, 0.22608696, 0.08695652, …, 0.15652174,
       0.20869565,
              0.36521739]))
```

```
[ ]:
```

1 11.1 – Wilcoxon test

```
[55]: stats.wilcoxon([1,3,4,-2])
```

```
[55]: WilcoxonResult(statistic=2.0, pvalue=0.375)
```

```
[56]: stats.wilcoxon([1,3,4,-2], alternative="greater")
```

```
[56]: WilcoxonResult(statistic=8.0, pvalue=0.1875)
```

```
[57]: stats.wilcoxon([1,3,4,-2], alternative="less")
```

```
[57]: WilcoxonResult(statistic=8.0, pvalue=0.875)
```

```
[ ]:
```

```
[145]: stats.wilcoxon([1,-2,3,4,5,6,7])
```

```
[145]: WilcoxonResult(statistic=2.0, pvalue=0.046875)
```

```
[ ]:
```

```
[49]: stats.wilcoxon([-1,3,4,2])
```

```
[49]: WilcoxonResult(statistic=1.0, pvalue=0.25)
```

```
[50]: stats.wilcoxon([1,3,4,-2])
```

```
[50]: WilcoxonResult(statistic=2.0, pvalue=0.375)
```

```
[51]: stats.wilcoxon([1,-3,4,2])
```

```
[51]: WilcoxonResult(statistic=3.0, pvalue=0.625)
```

```
[21]: stats.wilcoxon([1,3,-4,2])
```

```
[21]: WilcoxonResult(statistic=4.0, pvalue=0.875)
```

```
[54]: stats.wilcoxon([1,-3,4,-2])
```

```
[54]: WilcoxonResult(statistic=5.0, pvalue=1.0)
```

2 11.1 – Sign-test

```
[137]: stats.binom.cdf(1,4,p=.5)*2
```

```
[137]: 0.625
```

```
[ ]:
```

```
[146]: stats.binomtest(2,4,p=.5, alternative='two-sided')
```

```
[146]: BinomTestResult(k=2, n=4, alternative='two-sided', statistic=0.5, pvalue=1.0)
```

```
[138]: stats.binomtest(3,4,p=.5, alternative='two-sided')
```

```
[138]: BinomTestResult(k=3, n=4, alternative='two-sided', statistic=0.75, pvalue=0.625)
```

```
[42]: stats.binomtest(4,4,p=.5)
```

```
[42]: BinomTestResult(k=4, n=4, alternative='two-sided', statistic=1.0, pvalue=0.125)
```

```
[32]: 5/16.
```

```
[32]: 0.3125
```

```
[40]: stats.binomtest(3,4, alternative='greater')
```

```
[40]: BinomTestResult(k=3, n=4, alternative='greater', statistic=0.75, pvalue=0.3125)
```

```
[41]: stats.binomtest(4,4, alternative='greater')
```

```
[41]: BinomTestResult(k=4, n=4, alternative='greater', statistic=1.0, pvalue=0.0625)
```

```
[140]: 1/16.
```

```
[140]: 0.0625
```

```
[139]: stats.binomtest(3,4, alternative='less')
```

```
[139]: BinomTestResult(k=3, n=4, alternative='less', statistic=0.75, pvalue=0.9375)
```

3 11.2

```
[160]: import numpy as np
```

```
[161]: with_sample_problems = np.array([591, 621, 683, 579, 451, 680, 691, 769, 563,␣
        ↪575])
       without_sample_problems = np.array([509, 540, 688, 502, 424, 683, 568, 748,␣
        ↪530, 524])
```

```
[162]: print(with_sample_problems-without_sample_problems)
```

```
[ 82  81  -5  77  27  -3 123  21  33  51]
```

```
[165]: stats.wilcoxon(with_sample_problems, without_sample_problems,␣
        ↪alternative='less')
```

```
[165]: WilcoxonResult(statistic=52.0, pvalue=0.9970703125)
```

```
[111]: stats.wilcoxon(with_sample_problems-without_sample_problems,␣
        ↪alternative='greater')
```

```
[111]: WilcoxonResult(statistic=52.0, pvalue=0.0048828125)
```

```
[75]: np.size(with_sample_problems)
```

```
[75]: 10
```

```
[112]: np.mean(with_sample_problems), np.mean(without_sample_problems)
```

```
[112]: (620.3, 571.6)
```

```
[ ]:
```

```
[ ]:
```

```
[166]: stats.wilcoxon(with_sample_problems, without_sample_problems+20,␣
         ↪alternative='greater')
```

```
[166]: WilcoxonResult(statistic=46.0, pvalue=0.0322265625)
```

```
[167]: stats.wilcoxon(with_sample_problems-without_sample_problems-20,␣
         ↪alternative='less')
```

```
[167]: WilcoxonResult(statistic=46.0, pvalue=0.9755859375)
```

```
[156]: stats.wilcoxon(with_sample_problems-without_sample_problems-50,␣
         ↪alternative='less')
```

```
[156]: WilcoxonResult(statistic=28.0, pvalue=0.5390625)
```

```
[ ]:
```

```
[157]: with_sample_problems-without_sample_problems-50
```

```
[157]: array([ 32,  31, -55,  27, -23, -53,  73, -29, -17,   1])
```

```
[115]: with_sample_problems-without_sample_problems>=20
```

```
[115]: array([ True,  True, False,  True,  True, False,  True,  True,  True,
                True])
```

```
[159]: np.sum(with_sample_problems-without_sample_problems>=20)
```

```
[159]: 8
```

```
[117]: stats.binomtest(8, 10, alternative='greater')
```

```
[117]: BinomTestResult(k=8, n=10, alternative='greater', statistic=0.8,
       pvalue=0.0546875)
```

```
[148]: X = list(with_sample_problems)
       Y = list(without_sample_problems+20)
```

```
[169]: X
```

```
[169]: [591, 621, 683, 579, 451, 680, 691, 769, 563, 575]
```

```
[170]: Y
```

```
[170]: [529, 560, 708, 522, 444, 703, 588, 768, 550, 544]
```

```
[168]: stats.permutation_test([X,Y],  statistic=mystatistics)
```

```
[168]: PermutationTestResult(statistic=28.699999999999932, pvalue=0.4906,
       null_distribution=array([ 39.1,   0.7,  -5.5, …, -95.9,  -5.3,  38.9]))
```

```
[171]: stats.permutation_test([X*8,Y*8],  statistic=mystatistics)
```

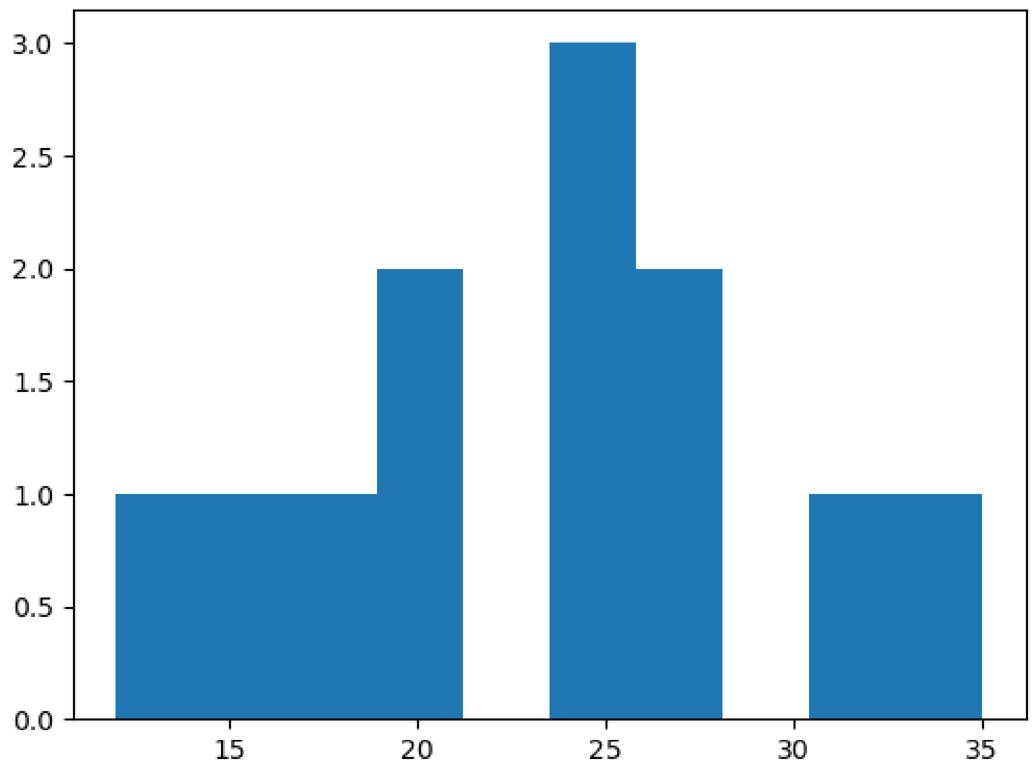
```
[171]: PermutationTestResult(statistic=28.699999999999932, pvalue=0.0538,
       null_distribution=array([-14.175,  -0.675,  14.825, …,  18.95 ,   0.775,  22.4
       ]))
```

4 11.3/12.3 – Waiting times

```
[303]: times = np.array([17, 15, 20, 20, 32, 28, 12, 26, 25, 25, 35, 24])
```

```
[304]: plt.hist(times)
```

```
[304]: (array([1., 1., 1., 2., 0., 3., 2., 0., 1., 1.]),
        array([12. , 14.3, 16.6, 18.9, 21.2, 23.5, 25.8, 28.1, 30.4, 32.7, 35. ]),
        <BarContainer object of 10 artists>)
```



```
[305]: times.size
```

```
[305]: 12
```

```
[306]: np.mean(times), np.median(times)
```

```
[306]: (23.25, 24.5)
```

```
[299]: np.sum(times<20), np.sum(times>20)
```

```
[299]: (3, 7)
```

```
[409]: stats.binomtest(7,10,.5, alternative='greater')
```

```
[409]: BinomTestResult(k=7, n=10, alternative='greater', statistic=0.7,
       pvalue=0.171875)
```

```
[384]: stats.binomtest(70,100,.5, alternative='greater')
```

```
[384]: BinomTestResult(k=70, n=100, alternative='greater', statistic=0.7,
       pvalue=3.925069822796835e-05)
```

[ ]:

[382]:
```
stats.wilcoxon(times-20, alternative='greater')
```

[382]:
```
WilcoxonResult(statistic=42.5, pvalue=0.0625234400655502)
```

[ ]:

5 Wilcoxon fails terribly for Exp distribution

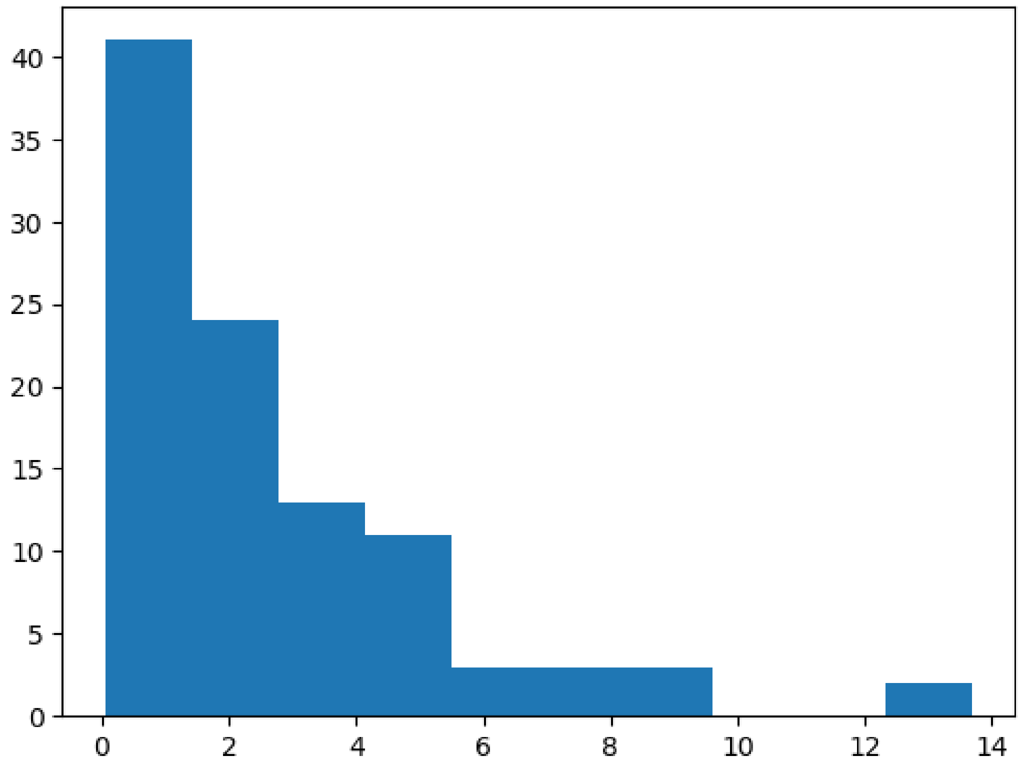
H_0: We have data from Exp(3) distribution.

We generate our own data, thus this really is true. We will sample the type-I error.

[388]:
```
expdata = stats.expon.rvs(scale=3, size=100)
```

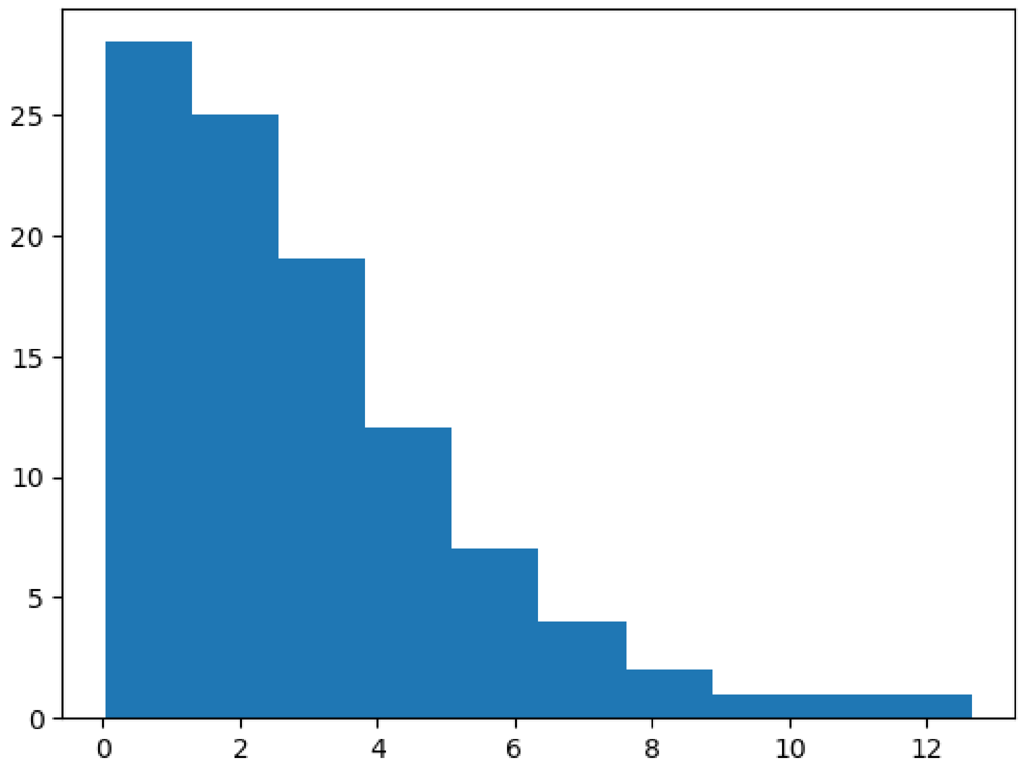
[386]:
```
plt.hist(expdata)
```

[386]:
```
(array([41., 24., 13., 11.,  3.,  3.,  3.,  0.,  0.,  2.]),
 array([ 0.06338363,  1.42545899,  2.78753436,  4.14960972,  5.51168508,
         6.87376045,  8.23583581,  9.59791117, 10.95998654, 12.3220619 ,
        13.68413726]),
 <BarContainer object of 10 artists>)
```



[370]:
```
plt.hist(expdata)
```

[370]:
```
(array([28., 25., 19., 12.,  7.,  4.,  2.,  1.,  1.,  1.]),
 array([ 0.04347253,  1.3051205 ,  2.56676847,  3.82841643,  5.0900644 ,
         6.35171236,  7.61336033,  8.8750083 , 10.13665626, 11.39830423,
        12.65995219]),
 <BarContainer object of 10 artists>)
```



[389]:
```
np.median(expdata), np.mean(expdata)
```

[389]:
```
(2.1125229336199345, 2.818046296052764)
```

[387]:
```
stats.expon.median(scale=3), stats.expon.mean(scale=3)
```

[387]:
```
(2.0794415416798357, 3.0)
```

[396]:
```
expdata = stats.expon.rvs(scale=3, size=100)
stats.wilcoxon(expdata-2.079).pvalue
```

[396]:
```
0.3055421446183436
```

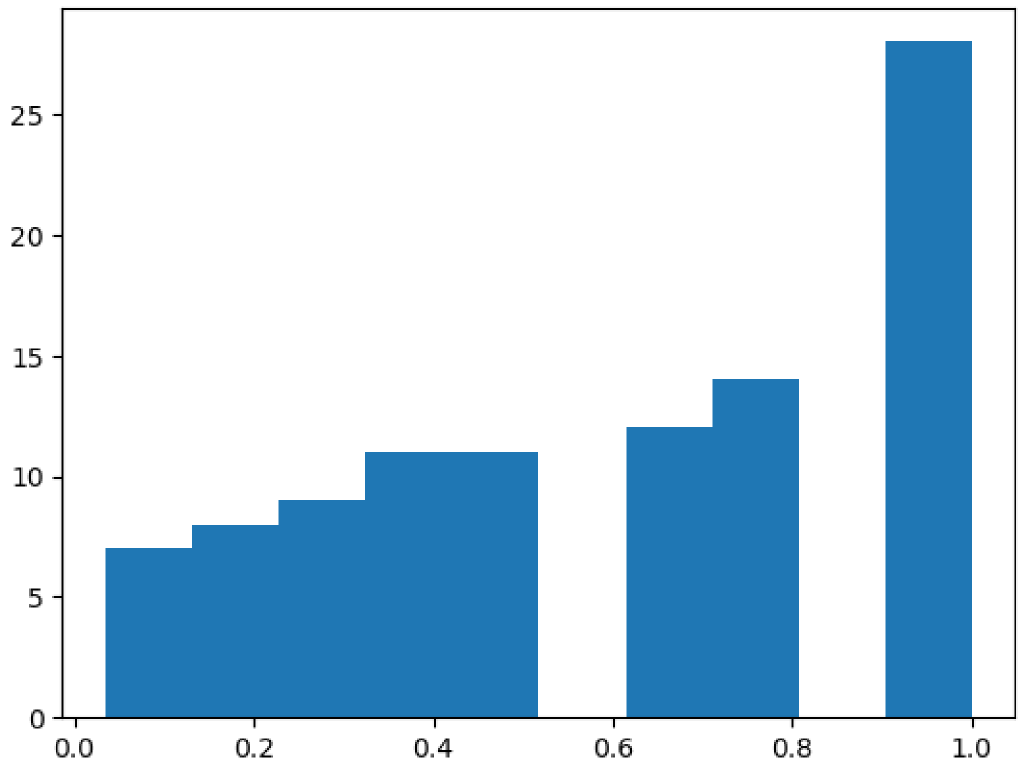
[400]:
```
L = np.array([stats.wilcoxon(stats.expon.rvs(scale=3, size=100)-2.079).pvalue␣
 ↪for _ in range(100)])
```

[410]:
```
def signtest(L):
    neg = np.sum(L<0)
    pos = np.sum(L>0)
    return stats.binomtest(pos,neg+pos,.5)
```

[415]:
```
L2 = np.array([signtest(stats.expon.rvs(scale=3, size=100)-2.079).pvalue for _␣
 ↪in range(100)])
```

[416]:
```
plt.hist(L2)
```

[416]:
```
(array([ 7.,  8.,  9., 11., 11.,  0., 12., 14.,  0., 28.]),
 array([0.0352002 , 0.13168018, 0.22816016, 0.32464014, 0.42112012,
        0.5176001 , 0.61408008, 0.71056006, 0.80704004, 0.90352002,
        1.        ]),
 <BarContainer object of 10 artists>)
```



[417]:
```
np.mean(L2<.05)
```

[417]:
```
0.02
```

```
[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[255]: a = np.array([531, 621, 663, 579, 451, 660, 591, 719, 543, 575])
       b = np.array([509, 540, 688, 502, 424, 683, 568, 748, 530, 524])

[176]: a.mean(), b.mean()

[176]: (593.3, 571.6)

[179]: np.sum(a-b>=50)

[179]: 3

[180]: stats.binomtest(3,10,.5,alternative='less')

[180]: BinomTestResult(k=3, n=10, alternative='less', statistic=0.3, pvalue=0.171875)

[ ]:
```