

Install required package

```
1 pip install pycodestyle pep257 pytest
```

```
Collecting pycodestyle
  Downloading https://files.pythonhosted.org/packages/10/5b/88879fb861ab79aef45c
    |████████████████████| 51kB 4.6MB/s
Collecting pep257
  Downloading https://files.pythonhosted.org/packages/ec/31/e432e1aa35f692e3f686
Requirement already satisfied: pytest in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: atomicwrites>=1.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: pluggy<0.8,>=0.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: more-itertools>=4.0.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.6/dist-packages
Installing collected packages: pycodestyle, pep257
Successfully installed pep257-0.7.0 pycodestyle-2.6.0
```

Checking code style including pep8 pep257 by pycodestyle and pep257

```
1 !pycodestyle fisher.py
2 !pep257 fisher.py
```

Unit testing for the function

```
1 !pytest fisher.py
```



```
===== test session starts =====
platform linux2 -- Python 2.7.17, pytest-3.6.4, py-1.8.0, pluggy-0.7.1
rootdir: /content, inifile:
collected 3 items

fisher.py .FF [100%]

===== FAILURES =====
test_2

def test_2():
    """
    second test for Fisher acceptance.

    testing if G > N
    """
    try:
        fisher_accept2(N=10, G=11, n=6)
    except AssertionError as e:
        > pytest.fail(e, pytrace=True)
E         Failed: assert 10 > 11
```

fisher.py:176: Failed

test_3

```
def test_3():
    """
    third test for Fisher acceptance.

    testing if input is a wrong data type
    """
    try:
        fisher_accept2('10', 6, 2)
    except AssertionError as e:
        > pytest.fail(e, pytrace=True)
E         Failed: assert False
E         + where False = isinstance('10', int)
```

fisher.py:188: Failed

===== 2 failed, 1 passed in 0.35 seconds =====

In the first fisher_accept function. It does not consider the if the population size is smaller than "good" items and also smaller than sample size. Moreover the datatype of the input might also be wrong. Thus, in the second algorithm, adding two assertions to avoid this situation.

Attached code

```
1 """
2 Fish accept.
3
4 Test and check algorithms for Fisher accept
5 """
6
7 import numpy as np
8 from scipy.stats import hypergeom
9 import pytest
10
11
12 def fisher_accept(N, G, n, alpha=0.05):
13     """
14     Acceptance region for randomized hypergeometric test.
15
16     Find the acceptance region for a randomized, exact level alpha test of
17     the null hypothesis  $X \sim \text{Hypergeometric}(N, G, n)$ . The acceptance region is
18     the smallest possible. (And not, for instance, symmetric.)
19
20     If a non-randomized, conservative test is desired, use the union of I and J
21     as the acceptance region.
22
23     Parameters
24     -----
25     N: integer
```

```

25     N: integer
26         population size
27     G: integer
28         number of "good" items in the population
29     n: integer
30         sample size
31     alpha : float
32         desired significance level
33
34     Returns
35     -----
36     I: list
37         values for which the test never rejects
38     J: list
39         values for which the test sometimes rejects
40     gamma : float
41         probability the test does not reject when the value is in J
42     """
43     x = np.arange(0, n+1)          # all possible values of X
44     posout = list(x)
45     # start with all possible outcomes, then remove some
46     pmf = hypergeom.pmf(x, N, G, n) # hypergeometric pmf
47     bottom = 0                     # smallest outcome still in I
48     top = n                        # largest outcome still in I
49     J = []
50     p_J = 0                        # probability of the randomized outcome
51     p_tail = 0                     # probability of outcomes excluded from I
52     while p_tail < alpha:
53         # still need to remove outcomes from the acceptance region
54         pb = pmf[bottom]
55         pt = pmf[top]
56         if pb < pt:                 # the lower possibility has smaller probability
57             J = [bottom]
58             p_J = pb
59             bottom += 1
60         elif pb > pt:               # the upper possibility has smaller probability
61             J = [top]
62             p_J = pt
63             top -= 1
64         else:
65             if bottom < top:        # the two possibilities have equal probability
66                 J = [bottom, top]
67                 p_J = pb+pt
68                 bottom += 1
69                 top -= 1
70             else:                  # there is only one possibility left
71                 J = [bottom]
72                 p_J = pb
73                 bottom += 1
74     p_tail += p_J
75     for j in J:
76         posout.remove(j)

```

```

77     gamma = (p_tail-alpha)/p_J
78     # probability of accepting H_0 when X in J to get exact level alpha
79     return posout, J, gamma
80
81
82 def fisher_accept2(N, G, n, alpha=0.05):
83     """
84     Acceptance region for randomized hypergeometric test.
85
86     Find the acceptance region for a randomized, exact level alpha test of
87     the null hypothesis X~Hypergeometric(N, G, n). The acceptance region is
88     the smallest possible. (And not, for instance, symmetric.)
89
90     If a non-randomized, conservative test is desired, use the union of I and J
91     as the acceptance region.
92
93     Parameters
94     -----
95     N:  integer
96         population size
97     G:  integer
98         number of "good" items in the population
99     n:  integer
100        sample size
101     alpha : float
102         desired significance level
103
104     Returns
105     -----
106     I:  list
107         values for which the test never rejects
108     J:  list
109         values for which the test sometimes rejects
110     gamma : float
111         probability the test does not reject when the value is in J
112     """
113     assert N > n
114     assert N > G
115     assert isinstance(N, int)
116     assert isinstance(G, int)
117     assert isinstance(n, int)
118     assert isinstance(alpha, float)
119     x = np.arange(0, n+1)          # all possible values of X
120     posout = list(x)
121     # start with all possible outcomes, then remove some
122     pmf = hypergeom.pmf(x, N, G, n) # hypergeometric pmf
123     bottom = 0                     # smallest outcome still in I
124     top = n                       # largest outcome still in I
125     J = []
126     p_J = 0                       # probability of the randomized outcome
127     p_tail = 0                    # probability of outcomes excluded from I
128     while p_tail < alpha:

```

```

129     # still need to remove outcomes from the acceptance region
130     pb = pmf[bottom]
131     pt = pmf[top]
132     if pb < pt:                # the lower possibility has smaller probability
133         J = [bottom]
134         p_J = pb
135         bottom += 1
136     elif pb > pt:            # the upper possibility has smaller probability
137         J = [top]
138         p_J = pt
139         top -= 1
140     else:
141         if bottom < top:      # the two possibilities have equal probability
142             J = [bottom, top]
143             p_J = pb+pt
144             bottom += 1
145             top -= 1
146         else:                # there is only one possibility left
147             J = [bottom]
148             p_J = pb
149             bottom += 1
150     p_tail += p_J
151     for j in J:
152         posout.remove(j)
153     gamma = (p_tail-alpha)/p_J
154     # probability of accepting H_0 when X in J to get exact level alpha
155     return posout, J, gamma
156
157
158 def test_1():
159     """
160     first test for Fisher acceptance.
161
162     testing for a regular case
163     """
164     assert fisher_accept(N=10, G=2, n=5) == ([1], [0, 2], 0.8875000000000001)
165
166
167 def test_2():
168     """
169     second test for Fisher acceptance.
170
171     testing if G > N
172     """
173     try:
174         fisher_accept2(N=10, G=11, n=6)
175     except AssertionError as e:
176         pytest.fail(e, pytrace=True)
177
178
179 def test_3():
180     """

```

```
180     """
181     third test for Fisher acceptance.
182
183     testing if input is a wrong data type
184     """
185     try:
186         fisher_accept2('10', 6, 2)
187     except AssertionError as e:
188         pytest.fail(e, pytrace=True)
189
```