

Install required package

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

```
Collecting pycodestyle
```

```
  Downloading https://files.pythonhosted.org/packages/10/5b/88879fb861ab79aef4  
  |████████████████████████████████████████| 51kB 4.8MB/s
```

```
Collecting pep257
```

```
  Downloading https://files.pythonhosted.org/packages/ec/31/e432e1aa35f692e3f6
```

```
Requirement already satisfied: pytest in /usr/local/lib/python3.6/dist-package
```

```
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-pa
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-pac
```

```
Requirement already satisfied: atomicwrites>=1.0 in /usr/local/lib/python3.6/c
```

```
Requirement already satisfied: more-itertools>=4.0.0 in /usr/local/lib/python3
```

```
Requirement already satisfied: pluggy<0.8,>=0.5 in /usr/local/lib/python3.6/di
```

```
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.6/dist-pack
```

```
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.6/dist-
```

```
Installing collected packages: pycodestyle, pep257
```

```
Successfully installed pep257-0.7.0 pycodestyle-2.6.0
```

Checking code style by pycodestyle and pep257

```
1 !pycodestyle chisq.py
```

```
2 !pep257 chisq.py
```

```
1 %%bash
```

```
2 pycodestyle chisq.py
```

```
3 pep257 chisq.py
```

Unit testing

```
1 !pytest chisq.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 9 items

chisq.py ..... [100%]

===== 9 passed in 0.52 seconds =====
```

Attached code

```
1 import chisq
2
3 %timeit -n 1000 chisq.chisq_1(chisq.sample_x, chisq.sample_y)
4 %timeit -n 1000 chisq.chisq_2(chisq.sample_x, chisq.sample_y)
5 %timeit -n 1000 chisq.chisq_3(chisq.sample_x, chisq.sample_y)
```

↳ 1000 loops, best of 3: 234 ms per loop
 1000 loops, best of 3: 118 ms per loop
 1000 loops, best of 3: 56 ms per loop

From the results of timing of single function, one can see the `chisq_1` is the slowest, `chisq_2` is the second, and `chisq_3` is the fastest. The `chisq_1` is expected to be the slowest since it used list and for each step it use list comprehension if need. However, the `chisq_1` is easy to read since for each element required, it has a separate line. The `chisq_2`, instead, using array as a data structure and by using some default numpy function it saves some time for instance, by using 'return_counts=True' one does not need to compute counts separately. Like `chisq_1`, it is also easy to read except some results are inside for saving some time. The `chisq_3` does not use any functions from other packages it uses all for loop for computing needed results. Among all three functions, it is the hardest to read, since using all for loops people need to check each loop to understand.

Overall the first method takes the most time with twice as much as second method and four time as that of the third. However the first one is the easiest to read while the third one take a while to read.

```
1 """
```

```

2 Chi-Square for two samples.
3
4 Three functions with different methods.
5 chisq_1 uses list comprehension, chisq_2 uses array, and chisq_3 uses loop.
6 """
7
8
9 import numpy as np
10 import timeit
11 import random
12 from random import randint
13 import time
14
15
16 def chisq_1(x, y):
17     """
18     Calculate a Chi-Square for two samples.
19
20     Arguments:
21     x:  list
22         sample 1
23     y:  list
24         sample 2
25
26     Returns
27     chi_s:  float
28             value of Chi-Square result
29     """
30     n, m = len(x), len(y)
31     z = x + y
32     u = np.unique(z)
33     p = [float(z.count(i)) / float(m+n) for i in u]
34     E_k = [n * j for j in p]
35     O_k = [x.count(k) for k in u]
36     chi_s = sum([float((O_k[i] - E_k[i])**2)/float(E_k[i])
37                 for i in range(len(u))])
38     return chi_s
39
40
41 def chisq_2(x, y):
42     """
43     Calculate a Chi-Square for two samples.
44
45     Keyword arguments:
46     x:  list sample 1

```

```
47 y: list sample 2
48
49 Returns
50 chi_s: float
51         value of Chi-Square result
52 """
53 n, m = len(x), len(y)
54 z = np.append(x, y)
55 u, p = np.unique(z, return_counts=True)
56 p = np.true_divide(p, (n+m))
57 E_k = p*n
58 O_k = [x.count(k) for k in u]
59 chi_s = np.nansum(np.true_divide(np.square(O_k - E_k), E_k))
60 return chi_s
61
62
63 def chisq_3(x, y):
64     """
65     Calculate a Chi-Square for two samples.
66
67     Keyword arguments:
68     x: list sample 1
69     y: list sample 2
70
71     Returns
72     chi_s: float
73             value of Chi-Square result
74     """
75     n, m = len(x), len(y)
76     z = x + y
77     u = [z[0]]
78     for i in z:
79         if i not in u:
80             u += [i]
81     p = []
82     for j in u:
83         count = 0
84         for k in z:
85             if j == k:
86                 count += 1
87         p += [float(count)/float(n+m)]
88     E_k = []
89     for freq in p:
90         E_k += [freq*n]
91     O_k = []
```

```

93     for count in range(0, len(u)):
94         for q in x:
95             if r == q:
96                 count += 1
97         O_k += [count]
98     chi_s = 0
99     for s in range(len(u)):
100         chi_s += float((O_k[s]-E_k[s])**2) / float(E_k[s])
101     return chi_s
102
103

```

```

104 # Establishing the simulation
105 fixtest_x = [1, 1, 2, 2, 2, 3, 4, 4, 4, 5]
106 fixtest_y = [2, 2, 3, 4, 4, 5, 5, 5]
107
108 random.seed(10)
109 sample_x = [randint(1, 6) for i in range(100000)]
110 sample_y = [randint(1, 6) for i in range(10000)]
111
112 coin_x = [randint(0, 1) for i in range(1000)]
113 coin_y = [randint(0, 1) for i in range(100)]
114
115 simple_x = [1, 1, 1]
116 simple_y = [1, 1, 0]
117
118 %timeit -n 20 chisq_1(sample_x, sample_y)
119 %timeit -n 20 chisq_2(sample_x, sample_y)
120 %timeit -n 20 chisq_3(sample_x, sample_y)
121
122 """

```

123 From the results of timing of single function, one can see the chisq_1 is the
 124 slowest, chisq_2 is the second, and chisq_3 is the fastest. The chisq_1 is
 125 expected to be the slowest since it used list and for each step it use list
 126 comprehension if need. However, the chisq_1 is easy to read since for each
 127 element required, it has a separate line. The chisq_2, instead, using array as
 128 a data structure and by using some default numpy function it saves some time
 129 for instance, by using 'return_counts=True' one does not need to compute counts
 130 separately. Like chisq_1, it is also easy to read except some results are
 131 inside for saving some time. The chisq_3 does not use any functions from other
 132 packages it uses all for loop for computing needed results. Among all three
 133 functions, it is the hardest to read, since using all for loops people need to
 134 check each loop to understand.

135 Overall the first method takes the most time with twice as much as second
 136 method and four time as that of the third. However the first one is the easiest

```
137 to read while the third one take a while to read.
138
139
140
141 def test_simple_1():
142     """
143     Test chisq_1 with a simple test.
144
145     Testing chisq_1 by using x = [1, 1, 1], y = [1, 0, 0]
146     """
147     assert chisq_1(simple_x, simple_y) == 0.6
148
149
150 def test_simple_2():
151     """
152     Test chisq_2 with a simple test.
153
154     Testing chisq_2 by using x = [1, 1, 1], y = [1, 0, 0]
155     """
156     assert chisq_2(simple_x, simple_y) == 0.6
157
158
159 def test_simple_3():
160     """
161     Test chisq_3 with a simple test.
162
163     Testing chisq_3 by using x = [1, 1, 1], y = [1, 0, 0]
164     """
165     assert chisq_3(simple_x, simple_y) == 0.6
166
167
168 def test_method_1():
169     """
170     Test chisq_1.
171
172     Testing chisq_1 by using random samples but fixed seed
173     """
174     assert round(chisq_1(sample_x, sample_y), 5) == 0.36701
175
176
177 def test_method_2():
178     """
179     Test chisq_2.
180
181     Testing chisq_2 by using random samples but fixed seed
```

```
183     assert round(chisq_2(sample_x, sample_y), 5) == 0.36701
184
185
186 def test_method_3():
187     """
188     Test chisq_3.
189
190     Testing chisq_3 by using random samples but fixed seed
191     """
192     assert round(chisq_3(sample_x, sample_y), 5) == 0.36701
193
194
195 def test_fixed_1():
196     """
197     Test chisq_1.
198
199     Testing chisq_1 by using fixed samples
200     """
201     assert round(chisq_1(fixtest_x, fixtest_y), 5) == 1.43
202
203
204 def test_fixed_2():
205     """
206     Test chisq_2.
207
208     Testing chisq_2 by using fixed samples
209     """
210     assert round(chisq_2(fixtest_x, fixtest_y), 5) == 1.43
211
212
213 def test_fixed_3():
214     """
215     Test chisq_3.
216
217     Testing chisq_3 by using fixed samples
218     """
219     assert round(chisq_3(fixtest_x, fixtest_y), 5) == 1.43
220
```

```
20 loops, best of 3: 234 ms per loop
20 loops, best of 3: 118 ms per loop
20 loops, best of 3: 55.9 ms per loop
```

