U-Statistic

May 20, 2025

[2]:
```
import numpy as np
import math
import itertools
import scipy.stats as stats
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from statsmodels.stats.diagnostic import lilliefors
from scipy.stats import norm
from itertools import combinations
from scipy.special import comb
import cProfile
```

1 Simulation of the U-Statistic

[12]:
```
# U-Statistic is defined as follows for a triplet of draws X, Y, and Z:
# f(X, Y, Z) = sgn(X + Y - 2Z) + sgn(X + Z - 2Y) + sgn(Y + Z - 2X)

# This function will calculate the triples test for skewness provided a triplet␣
 ↪of data points
# The argument triplet is a 3x1 vector of sampled variables
def triples_test(data):
    triplet = np.array(list(combinations(data, 3)))
    X, Y, Z = triplet[:, 0], triplet[:, 1], triplet[:, 2]
    return np.sum(np.sign(X + Y - 2*Z) + np.sign(X + Z - 2*Y) + np.sign(Y + Z -␣
 ↪2*X))

U_syms1 = []
U_skews1 = []
U_syms2 = []
U_skews2 = []
U_syms3 = []
U_skews3 = []
cProfile.run("""for n in range(100):
    # 10000 is not a feasible value to include
    vals = [10, 100, 500]

    for val in vals:
```

```
        normal_draws = np.random.normal(0, 1, val)
        chi_squared_draws = np.random.chisquare(1, val)
        scalar = (1 / comb(val, 3, exact = True))
        U_sym = scalar * triples_test(normal_draws)
        U_skew = scalar * triples_test(chi_squared_draws)
        if (val == 10):
            U_syms1.append(U_sym)
            U_skews1.append(U_skew)
        if (val == 100):
            U_syms2.append(U_sym)
            U_skews2.append(U_skew)
        if (val == 500):
            U_syms3.append(U_sym)
            U_skews3.append(U_skew)
    print(n)
""")
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

```
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
         124846604 function calls in 715.394 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      600  191.286    0.319  714.506    1.191 444688049.py:6(triples_test)
        1    0.863    0.863  715.394  715.394 <string>:1(<module>)
      300    0.000    0.000    0.000    0.000 _basic.py:2682(comb)
      600    0.000    0.000    0.000    0.000
fromnumeric.py:2172(_sum_dispatcher)
      600    0.002    0.000    0.560    0.001 fromnumeric.py:2177(sum)
      600    0.002    0.000    0.558    0.001 fromnumeric.py:71(_wrapreduction)
      600    0.001    0.000    0.001    0.000 fromnumeric.py:72(<dictcomp>)
      100    0.000    0.000    0.000    0.000 iostream.py:138(_event_pipe)
      100    0.000    0.000    0.005    0.000 iostream.py:259(schedule)
      200    0.000    0.000    0.000    0.000 iostream.py:505(parent_header)
      200    0.000    0.000    0.000    0.000
iostream.py:550(_is_master_process)
      200    0.000    0.000    0.005    0.000 iostream.py:577(_schedule_flush)
      200    0.002    0.000    0.008    0.000 iostream.py:655(write)
 11889200    8.704    0.000   22.850    0.000
ipkernel.py:770(_clean_thread_parent_frames)
  5944600    5.720    0.000    8.665    0.000 ipkernel.py:785(<setcomp>)
      100    0.004    0.000    0.004    0.000 socket.py:621(send)
      100    0.000    0.000    0.000    0.000
threading.py:1125(_wait_for_tstate_lock)
 47556800    2.945    0.000    2.945    0.000 threading.py:1168(ident)
```

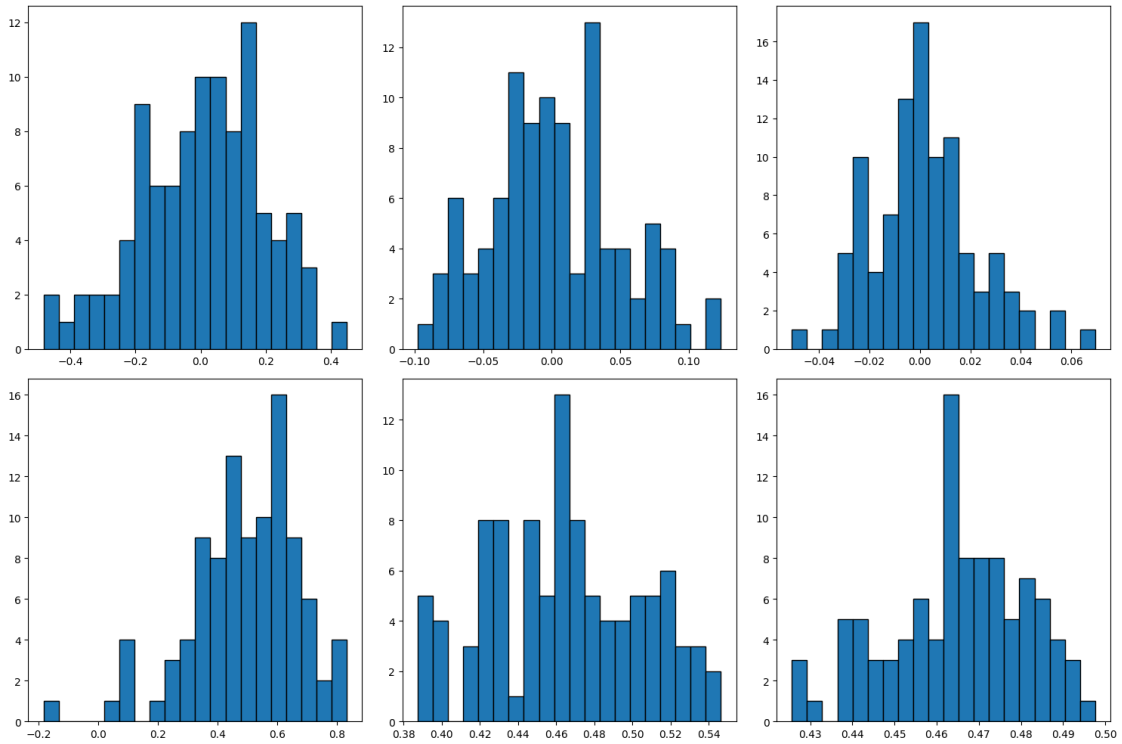
```
      100    0.000    0.000    0.000    0.000 threading.py:1192(is_alive)
  5944600    3.626    0.000    4.388    0.000 threading.py:1501(enumerate)
      100    0.000    0.000    0.000    0.000 threading.py:575(is_set)
        1    0.000    0.000    0.000    0.000 tz.py:74(utcoffset)
        1    0.000    0.000  715.394  715.394 {built-in method builtins.exec}
 11890000    0.347    0.000    0.347    0.000 {built-in method
builtins.isinstance}
      200    0.000    0.000    0.000    0.000 {built-in method builtins.len}
      100    0.001    0.000    0.008    0.000 {built-in method builtins.print}
      600  499.810    0.833  499.810    0.833 {built-in method numpy.array}
      200    0.000    0.000    0.000    0.000 {built-in method posix.getpid}
  5944800    0.311    0.000    0.311    0.000 {method '__exit__' of
'_thread.RLock' objects}
      100    0.000    0.000    0.000    0.000 {method 'acquire' of
'_thread.lock' objects}
      100    0.000    0.000    0.000    0.000 {method 'append' of
'collections.deque' objects}
      600    0.000    0.000    0.000    0.000 {method 'append' of 'list'
objects}
      300    0.014    0.000    0.014    0.000 {method 'chisquare' of
'numpy.random.mtrand.RandomState' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of
'_lsprof.Profiler' objects}
      200    0.000    0.000    0.000    0.000 {method 'get' of
'_contextvars.ContextVar' objects}
      800    0.000    0.000    0.000    0.000 {method 'items' of 'dict' objects}
 23778400    0.746    0.000    0.746    0.000 {method 'keys' of 'dict' objects}
      300    0.003    0.000    0.003    0.000 {method 'normal' of
'numpy.random.mtrand.RandomState' objects}
      600    0.556    0.001    0.556    0.001 {method 'reduce' of 'numpy.ufunc'
objects}
 11889200    0.451    0.000    0.451    0.000 {method 'values' of 'dict'
objects}
      200    0.000    0.000    0.000    0.000 {method 'write' of '_io.StringIO'
objects}
```

[14]:
```
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# Loop through combinations and plot
for i in range(2):
    for j in range(3):
        ax = axes[i, j]
        if (i == 0):
            if (j == 0):
                ax.hist(U_syms1, bins=20, edgecolor='black')
```

```
            if (j == 1):
                ax.hist(U_syms2, bins=20, edgecolor='black')
            if (j == 2):
                ax.hist(U_syms3, bins=20, edgecolor='black')
        if (i == 1):
            if (j == 0):
                ax.hist(U_skews1, bins=20, edgecolor='black')
            if (j == 1):
                ax.hist(U_skews2, bins=20, edgecolor='black')
            if (j == 2):
                ax.hist(U_skews3, bins=20, edgecolor='black')

plt.tight_layout()
```



2 Testing for skew in the one factor model

[ ]:
```
np.random.seed(42)

iter = [100, 1000, 10000]
n = 100
betas = np.random.normal(1, 0.25)
```