Rohan Sanjay

MWF 9:00 am MATH 408

Computer Project #2

10/30/2020

In [180]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
from statsmodels.stats import stattools
from scipy import integrate
import math
import statistics
```

Part I: Generate 100 independent normal random variables with mean zero and variance 1. Call it vector V1 = (x1, . . . , x100). Then generate 100 independent normal random variables with mean zero and variance 1.5. Call it vector V1.5 = (y1, . . . , y100). Now, how can you tell which vector is which if you only look at the components? Here are four possible ways.

In [181]:

```
v1 = np.random.normal(0, 1, 100)
v1_5 = np.random.normal(0, 1.5, 100)
```

Procedure A. If you are handed only a pair of numbers xk and yk without knowing which is which, procedure A is to guess that the number with the smaller absolute value came from vectorV1. Run this procedure on your data and determine how many times you get the correct conclusion. Then compute the theoretical value of the probability that procedure A gives the correct conclusion.

In [182]:

```
count = 0

for x in range(100):
    if abs(v1[x]) < abs(v1_5[x]):
        count += 1

print('Experimental probability:', count / 100)

t, s = 0, 1
cauchy = lambda x: 1 / (math.pi * (1 + x**2))
print('Theoretical probability:', integrate.quad(cauchy, -np.sqrt(1.5), np.sqrt(1.5
))[0])
print('P( |v1| < |v1.5| ) = P( |v1| < sqrt(1.5) * |v1.5| / sqrt(1.5) ) = P( cauchy
(0, 1) < sqrt(1.5) )',
       'because the divison between two standard normal random variables is cauchy
(0, 1)')
```

```
Experimental probability: 0.61
Theoretical probability: 0.5640942168489749
P( |v1| < |v1.5| ) = P( |v1| < sqrt(1.5) * |v1.5| / sqrt(1.5) ) = P( ca
uchy(0, 1) < sqrt(1.5) ) because the divison between two standard norma
l random variables is cauchy(0, 1)
```

In [183]:

```
print('Here, we get the right conclusion because v1 is smaller 61/100 times')
```

```
Here, we get the right conclusion because v1 is smaller 61/100 times
```

Procedure B. Suppose that instead of a pair of numbers (xk, yk), you have the entire collec-tion of numbers, (x1, . . . , x100) and (y1, . . . , y100) but without knowing which collection is which.

Procedure B says that the collection with the larger sum of squares is V1.5. Apply procedure B to the data you generated. Does this procedure give the correct answer?

In [185]:

```
print('SS v1:', v1 @ v1.T)
print('SS v_1.5:', v1_5 @ v1_5.T)
print('This method gives the right answer since sum of sqaures v_1.5 > v1')
```

```
SS v1: 109.05190104765356
SS v_1.5: 242.1750704899023
This method gives the right answer since sum of sqaures v_1.5 > v1
```

In [186]:

```
print('Theoretic probability:', stats.f.cdf(1.5, dfn=100, dfd=100))
print('The sum of v1_i and v1.5_i is chi-squared with 100 df and chi-sqaured df 100
/ chi-sqaured df 100 if F distribution dfn 100, dfd 100')
```

```
Theoretic probability: 0.9780695578699148
The sum of v1_i and v1.5_i is chi-squared with 100 df and chi-sqaured d
f 100 / chi-sqaured df 100 if F distribution dfn 100, dfd 100
```

Procedure C. This is a more realistic version of Procedure B, when you pretend that you do not know that the mean in each sample is zero and say that the collection with the larger sample standard deviation is V1.5. Apply procedure C to the data you generated. Does this procedure give the correct answer?

In [187]:

```
print('sample std v1', statistics.stdev(v1))
print('sample std v1.5', statistics.stdev(v1_5))
print('This method gives the right answer since sample std v_1.5 > v1')
```

```
sample std v1 1.0486616723606663
sample std v1.5 1.5640183160931196
This method gives the right answer since sample std v_1.5 > v1
```

In [188]:

```
print('Theoretic probability:', stats.f.cdf(1.5, dfn=100, dfd=100), 'by the F test'
)
```

```
Theoretic probability: 0.9780695578699148 by the F test
```

Procedure D.

This is an even more realistic procedure, when you pretend that you do not know the distributions of the sample. Use the result of problem 6, part 4 in Homework number 12 to reduce the setting to a standard shift model, and then use sign test and any other non-parametric test to answer the question.

In [189]:

```
v1_ln = np.log( np.abs(v1) )
v1_5_ln = np.log( np.abs(v1_5) )

count = 0

for x in range(100):
    if abs(v1_ln[x]) < abs(v1_5_ln[x]):
        count += 1

print('Experimental probability sign test:', count / 100)

print('Experimental probability non-parametric test sample std v1_ln', statistics.s
tdev(v1_ln))
print('Experimental probability non-parametric test sample std v1.5_ln', statistics
.stdev(v1_5_ln))
```

```
Experimental probability sign test: 0.47
Experimental probability non-parametric test sample std v1_ln 1.3989330
829591777
Experimental probability non-parametric test sample std v1.5_ln 0.92440
08657443137
```

In [190]:

```
print('Both tests here gave us the wrong answer. In the sign test, we got 47/100, w
hich while close, could lead us to believe that the second vector has the smaller v
ariance.',
      'In the non-parametric test, we got that the first vector has a larger sample
std. This could have happened because taking the log of rvs close to zero in the fi
rst vector',
      'could have made many of the small values below 1 very large.')
```

```
Both tests here gave us the wrong answer. In the sign test, we got 47/1
00, which while close, could lead us to believe that the second vector
has the smaller variance. In the non-parametric test, we got that the f
irst vector has a larger sample std. This could have happened because t
aking the log of rvs close to zero in the first vector could have made
many of the small values below 1 very large.
```

Generate five more collections of 100 independent normal random variables with mean zero and variance 1. Then generate five more collections of 100 independent normal random variables with mean zero and variance 1.5. Apply procedures B, C, and D to each of the five pairs. How many times did you get wrong answer? Compute the theoretical value of the probabilities that procedures B, C, and D give correct answer.

In [191]:

```
print('-------------------------------------------------')

for i in range(5):
    print('Pair', i + 1)
    v1 = np.random.normal(0, 1, 100)
    v1_5 = np.random.normal(0, 1.5, 100)

    count = 0
    for x in range(100):
        if abs(v1[x]) < abs(v1_5[x]):
            count += 1

    print('Procedure A Experimental probability:', count / 100)
    print()

    print('Procedue B SS v1:', v1 @ v1.T)
    print('Procedue B SS v_1.5:', v1_5 @ v1_5.T)
    print()

    print('Procedue C sample std v1', statistics.stdev(v1))
    print('Procedue C sample std v1.5', statistics.stdev(v1_5))

    print('\nProcedure D')
    v1_ln = np.log( np.abs(v1) )
    v1_5_ln = np.log( np.abs(v1_5) )

    count = 0

    for x in range(100):
        if abs(v1_ln[x]) < abs(v1_5_ln[x]):
            count += 1

    print('Experimental probability sign test:', count / 100)

    print('Experimental probability non-parametric test sample std v1_ln', statisti
cs.stdev(v1_ln))
    print('Experimental probability non-parametric test sample std v1.5_ln', statis
tics.stdev(v1_5_ln))
    print('-------------------------------------------------')
```

```
-------------------------------------------------
Pair 1
Procedure A Experimental probability: 0.67

Procedue B SS v1: 81.2874588589245
Procedue B SS v_1.5: 214.86672565782084

Procedue C sample std v1 0.9059490080244722
Procedue C sample std v1.5 1.4660419582315676

Procedure D
Experimental probability sign test: 0.47
Experimental probability non-parametric test sample std v1_ln 1.2526585
770107541
Experimental probability non-parametric test sample std v1.5_ln 1.14943
65996320706
-------------------------------------------------
Pair 2
Procedure A Experimental probability: 0.62

Procedue B SS v1: 94.31268403326894
Procedue B SS v_1.5: 206.2815799548444

Procedue C sample std v1 0.9690915568616177
Procedue C sample std v1.5 1.4395708812660017

Procedure D
Experimental probability sign test: 0.51
Experimental probability non-parametric test sample std v1_ln 1.0669323
285599552
Experimental probability non-parametric test sample std v1.5_ln 1.15565
66627531581
-------------------------------------------------
Pair 3
Procedure A Experimental probability: 0.61

Procedue B SS v1: 116.6856224506015
Procedue B SS v_1.5: 212.92936262517227

Procedue C sample std v1 1.0743835356980542
Procedue C sample std v1.5 1.4655385571203565

Procedure D
Experimental probability sign test: 0.47
Experimental probability non-parametric test sample std v1_ln 0.9428140
152137873
Experimental probability non-parametric test sample std v1.5_ln 1.26681
64144268117
-------------------------------------------------
Pair 4
Procedure A Experimental probability: 0.63

Procedue B SS v1: 101.36377076686574
Procedue B SS v_1.5: 273.9280262787014

Procedue C sample std v1 1.0085819818134856
Procedue C sample std v1.5 1.640317793255577
```

```
Procedure D
Experimental probability sign test: 0.49
Experimental probability non-parametric test sample std v1_ln 1.1726831
385920493
Experimental probability non-parametric test sample std v1.5_ln 1.00106
92035110893
-------------------------------------------------
Pair 5
Procedure A Experimental probability: 0.65

Procedue B SS v1: 83.94409306753595
Procedue B SS v_1.5: 215.88362083726918

Procedue C sample std v1 0.9207643772123681
Procedue C sample std v1.5 1.4704222039413373

Procedure D
Experimental probability sign test: 0.51
Experimental probability non-parametric test sample std v1_ln 1.1542770
678411236
Experimental probability non-parametric test sample std v1.5_ln 1.10762
9179200343
-------------------------------------------------
```

In [192]:

```
print('Procedures A-C in all pairs gave us the correct answer every time.')
print('The sign test in procedure d was wrong for 3 of the pairs, however very clos
e each time.')
print('The non-parametric test in procedure d was wrong for 3 of the pairs as wel
l.')
```

```
Procedures A-C in all pairs gave us the correct answer every time.
The sign test in procedure d was wrong for 3 of the pairs, however very
close each time.
The non-parametric test in procedure d was wrong for 3 of the pairs as
well.
```

Part II

Now repeat Part I when the vector V1 consists of independent standard Cauchy random variables, and V1.5 consists of independent Cauchy random variables with location parameter equal to zero and the scale parameter equal to √1.5. Compare and contrast the results with what you got in Part I.

In [193]:

```
v1 = np.random.standard_cauchy(100)
v1_5 = stats.cauchy.rvs(loc=0, scale=np.sqrt(1.5), size=100)
```

Procedure A. If you are handed only a pair of numbers xk and yk without knowing which is which, procedure A is to guess that the number with the smaller absolute value came from vectorV1. Run this procedure on your data and determine how many times you get the correct conclusion. Then compute the theoretical value of the probability that procedure A gives the correct conclusion.

In [195]:

```
count = 0

for x in range(100):
    if abs(v1[x]) < abs(v1_5[x]):
        count += 1

print('Experimental probability:', count / 100)

cauchy_product = lambda x: np.log(x**2) / ( math.pi**2 * (x**2 - 1) )
print('Theoretical probability:', 2 * integrate.quad(cauchy_product, 0, np.sqrt(1.5
))[0])
print('P( |v1| < |v1.5| ) = P( |v1| < sqrt(1.5) * |v1.5 / sqrt(1.5)| ) = P( |cauchy
(0, 1)| * |1 / cauchy(0, 1)| < sqrt(1.5) )',
      ' = P( -sqrt(1.5) < cauchy(0, 1) * 1 / cauchy(0, 1) < sqrt(1.5) ). The pdf of
product and quotient of two inp cauchy',
      'is log u^2 / (pi^2(u^2 - 1)), which can be integrated from 0 to sqrt(1.5) an
d multiplied by 2')
```

```
Experimental probability: 0.54
Theoretical probability: 0.5409886682458542
P( |v1| < |v1.5| ) = P( |v1| < sqrt(1.5) * |v1.5 / sqrt(1.5)| ) = P( |c
auchy(0, 1)| * |1 / cauchy(0, 1)| < sqrt(1.5) )  = P( -sqrt(1.5) < cauc
hy(0, 1) * 1 / cauchy(0, 1) < sqrt(1.5) ). The pdf of product and quoti
ent of two inp cauchy is log u^2 / (pi^2(u^2 - 1)), which can be integr
ated from 0 to sqrt(1.5) and multiplied by 2
```

In [196]:

```
print('This procedure gave us the correct answer since v1 was smaller than v1.5 54/
100 times')
```

```
This procedure gave us the correct answer since v1 was smaller than v1.
5 54/100 times
```

Procedure B. Suppose that instead of a pair of numbers (xk, yk), you have the entire collec-tion of numbers, (x1, . . . , x100) and (y1, . . . , y100) but without knowing which collection is which.

Procedure B says that the collection with the larger sum of squares is V1.5. Apply procedure B to the data you generated. Does this procedure give the correct answer?

In [197]:

```
print('SS v1:', v1 @ v1.T)
print('SS v_1.5:', v1_5 @ v1_5.T)
```

```
SS v1: 4162.778574195615
SS v_1.5: 2412380.506041548
```

In [207]:

```
print('This method gives the right answer since sum of sqaures v_1.5 > v1. However,
this method can be especially unreliable with cauchy since cauchy rvs can',
      'be very large numbers as seen by the much the much larger SS for the second
 vector in comparison to the first.')
```

```
This method gives the right answer since sum of sqaures v_1.5 > v1. How
ever, this method can be especially unreliable with cauchy since cauchy
rvs can be very large numbers as seen by the much the much larger SS fo
r the second vector in comparison to the first.
```

Procedure C. This is a more realistic version of Procedure B, when you pretend that you do not know that the mean in each sample is zero and say that the collection with the larger sample standard deviation is V1.5. Apply procedure C to the data you generated. Does this procedure give the correct answer?

In [198]:

```
print('sample std v1', statistics.stdev(v1))
print('sample std v1.5', statistics.stdev(v1_5))
```

```
sample std v1 6.461385439328347
sample std v1.5 155.36219021761374
```

In [200]:

```
print('This method gives the right answer since sample std v_1.5 > v1')
print('However, given how much larger the std of the second vector is, it is likely
that a few of the cauchy rvs in the 2nd vectors are very large.')
```

```
This method gives the right answer since sample std v_1.5 > v1
However, given how much larger the std of the second vector is, it is l
ikely that a few of the cauchy rvs in the 2nd vectors are very large.
```

Procedure D.

This is an even more realistic procedure, when you pretend that you do not know the distributions of the sample. Use the result of problem 6, part 4 in Homework number 12 to reduce the setting to a standard shift model, and then use sign test and any other non-parametric test to answer the question.

In [201]:

```
v1_ln = np.log( np.abs(v1) )
v1_5_ln = np.log( np.abs(v1_5) )

count = 0

for x in range(100):
    if abs(v1_ln[x]) < abs(v1_5_ln[x]):
        count += 1

print('Experimental probability sign test:', count / 100)

print('Experimental probability non-parametric test sample std v1_ln', statistics.s
tdev(v1_ln))
print('Experimental probability non-parametric test sample std v1.5_ln', statistics
.stdev(v1_5_ln))
```

```
Experimental probability sign test: 0.45
Experimental probability non-parametric test sample std v1_ln 1.6404320
085302728
Experimental probability non-parametric test sample std v1.5_ln 1.41906
878001138
```

In [202]:

```
print('Here, both tests gave us the wrong answer. The sign test had only 45/50 valu
es in vector1 smaller and the std of vector 1 is larger than std vector 2.')
```

```
Here, both tests gave us the wrong answer. The sign test had only 45/50
values in vector1 smaller and the std of vector 1 is larger than std ve
ctor 2.
```

Generate five more collections of 100 independent normal random variables with mean zero and variance 1. Then generate five more collections of 100 independent normal random variables with mean zero and variance 1.5. Apply procedures B, C, and D to each of the five pairs. How many times did you get wrong answer? Compute the theoretical value of the probabilities that procedures B, C, and D give correct answer.

In [204]:

```
print('-------------------------------------------------')

for i in range(5):
    print('Pair', i + 1)
    v1 = np.random.standard_cauchy(100)
    v1_5 = stats.cauchy.rvs(loc=0, scale=np.sqrt(1.5), size=100)

    count = 0
    for x in range(100):
        if abs(v1[x]) < abs(v1_5[x]):
            count += 1

    print('Procedure A Experimental probability:', count / 100)
    print()

    print('Procedue B SS v1:', v1 @ v1.T)
    print('Procedue B SS v_1.5:', v1_5 @ v1_5.T)
    print()

    print('Procedue C sample std v1', statistics.stdev(v1))
    print('Procedue C sample std v1.5', statistics.stdev(v1_5))

    print('\nProcedure D')
    v1_ln = np.log( np.abs(v1) )
    v1_5_ln = np.log( np.abs(v1_5) )

    count = 0

    for x in range(100):
        if abs(v1_ln[x]) < abs(v1_5_ln[x]):
            count += 1

    print('Experimental probability sign test:', count / 100)

    print('Experimental probability non-parametric test sample std v1_ln', statistics.stdev(v1_ln))
    print('Experimental probability non-parametric test sample std v1.5_ln', statistics.stdev(v1_5_ln))
    print('-------------------------------------------------')
```

```
-------------------------------------------------
Pair 1
Procedure A Experimental probability: 0.49

Procedue B SS v1: 3057.905614793219
Procedue B SS v_1.5: 6364.2849787782525

Procedue C sample std v1 5.537781012915894
Procedue C sample std v1.5 7.796693295343969

Procedure D
Experimental probability sign test: 0.51
Experimental probability non-parametric test sample std v1_ln 1.4828380
255140596
Experimental probability non-parametric test sample std v1.5_ln 1.36007
99748889003
-------------------------------------------------
Pair 2
Procedure A Experimental probability: 0.48

Procedue B SS v1: 13291.268777981762
Procedue B SS v_1.5: 5458.307509436001

Procedue C sample std v1 11.428831237450709
Procedue C sample std v1.5 7.318070273584836

Procedure D
Experimental probability sign test: 0.5
Experimental probability non-parametric test sample std v1_ln 1.6779724
580204438
Experimental probability non-parametric test sample std v1.5_ln 1.72960
09586838703
-------------------------------------------------
Pair 3
Procedure A Experimental probability: 0.6

Procedue B SS v1: 2525.301793087224
Procedue B SS v_1.5: 5766.5768518474815

Procedue C sample std v1 5.016007410489193
Procedue C sample std v1.5 7.484848608364343

Procedure D
Experimental probability sign test: 0.43
Experimental probability non-parametric test sample std v1_ln 1.5506240
897019974
Experimental probability non-parametric test sample std v1.5_ln 1.31783
5163726397
-------------------------------------------------
Pair 4
Procedure A Experimental probability: 0.5

Procedue B SS v1: 4537.900727765797
Procedue B SS v_1.5: 92684.68493468183

Procedue C sample std v1 6.70151267633837
Procedue C sample std v1.5 30.448394855128022
```

```
Procedure D
Experimental probability sign test: 0.51
Experimental probability non-parametric test sample std v1_ln 1.6213627
111274302
Experimental probability non-parametric test sample std v1.5_ln 1.88883
25234360848
-------------------------------------------------
Pair 5
Procedure A Experimental probability: 0.49

Procedue B SS v1: 13786.886923538052
Procedue B SS v_1.5: 1027762.4767475245

Procedue C sample std v1 11.800417399610653
Procedue C sample std v1.5 101.28553180234832

Procedure D
Experimental probability sign test: 0.5
Experimental probability non-parametric test sample std v1_ln 1.5924845
772064282
Experimental probability non-parametric test sample std v1.5_ln 1.79104
25775071017
-------------------------------------------------
```

In [205]:

```
print('Procedure a is wrong 3 times')
print('Procedure b is wrong 1 time')
print('Procedure c is wrong 1 time')
print('The sign test in procedure d was wrong for 1 of the pairs.')
print('The non-parametric test in procedure d was wrong for 2 of the pairs.')
```

```
Procedure a is wrong 3 times
Procedure b is wrong 1 time
Procedure c is wrong 1 time
The sign test in procedure d was wrong for 1 of the pairs.
The non-parametric test in procedure d was wrong for 2 of the pairs.
```

Compared to part 1, procudures a - c were wrong more often in part 2. This is likely because cauchy rv can more easily take on larger values, making these procedures more prone to being errenous. Procedure d, however, seemed to be a little more accurate in part 2. This could potentially be because cauchy is less likely to be less than 1 and consequently a very large negative number when the natural log is taken.

In [ ]:

```

```