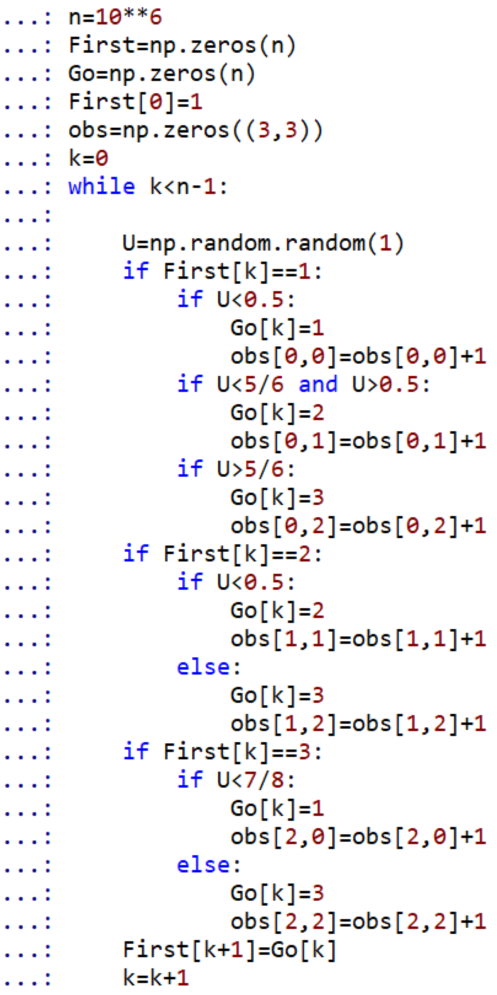
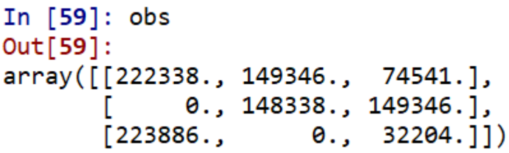
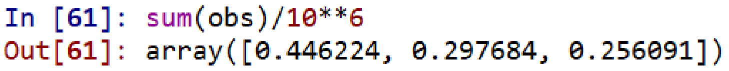
1)



```
   ...: n=10**6
   ...: First=np.zeros(n)
   ...: Go=np.zeros(n)
   ...: First[0]=1
   ...: obs=np.zeros((3,3))
   ...: k=0
   ...: while k<n-1:
   ...:
   ...:     U=np.random.random(1)
   ...:     if First[k]==1:
   ...:         if U<0.5:
   ...:             Go[k]=1
   ...:             obs[0,0]=obs[0,0]+1
   ...:         if U<5/6 and U>0.5:
   ...:             Go[k]=2
   ...:             obs[0,1]=obs[0,1]+1
   ...:         if U>5/6:
   ...:             Go[k]=3
   ...:             obs[0,2]=obs[0,2]+1
   ...:     if First[k]==2:
   ...:         if U<0.5:
   ...:             Go[k]=2
   ...:             obs[1,1]=obs[1,1]+1
   ...:         else:
   ...:             Go[k]=3
   ...:             obs[1,2]=obs[1,2]+1
   ...:     if First[k]==3:
   ...:         if U<7/8:
   ...:             Go[k]=1
   ...:             obs[2,0]=obs[2,0]+1
   ...:         else:
   ...:             Go[k]=3
   ...:             obs[2,2]=obs[2,2]+1
   ...:     First[k+1]=Go[k]
   ...:     k=k+1
```

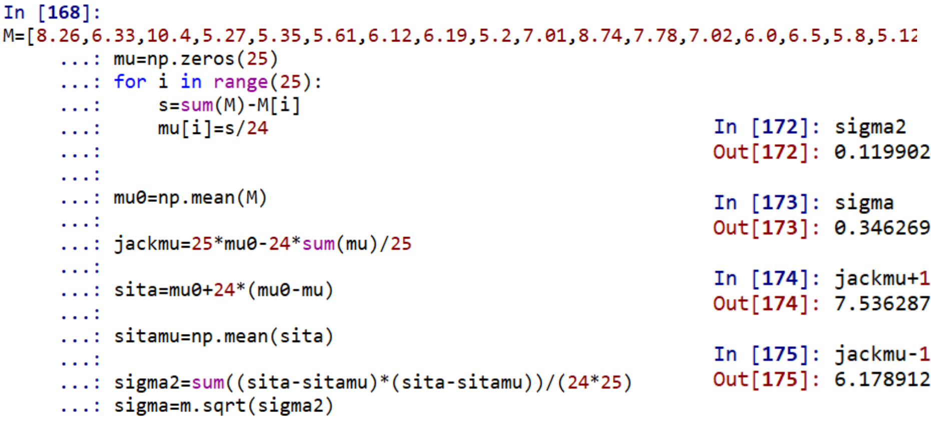


```
In [59]: obs
Out[59]:
array([[222338., 149346.,  74541.],
       [     0., 148338., 149346.],
       [223886.,      0.,  32204.]])
```

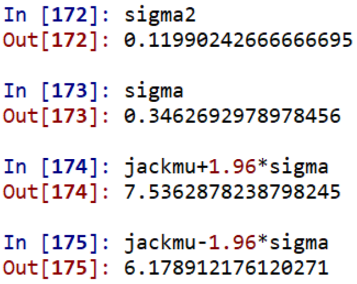


```
In [61]: sum(obs)/10**6
Out[61]: array([0.446224, 0.297684, 0.256091])
```

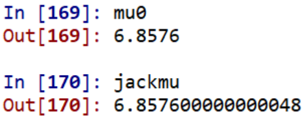
2)



```
In [168]:
M=[8.26,6.33,10.4,5.27,5.35,5.61,6.12,6.19,5.2,7.01,8.74,7.78,7.02,6.0,6.5,5.8,5.12
    ...: mu=np.zeros(25)
    ...: for i in range(25):
    ...:     s=sum(M)-M[i]
    ...:     mu[i]=s/24
    ...:
    ...:
    ...: mu0=np.mean(M)
    ...:
    ...: jackmu=25*mu0-24*sum(mu)/25
    ...:
    ...: sita=mu0+24*(mu0-mu)
    ...:
    ...: sitamu=np.mean(sita)
    ...:
    ...: sigma2=sum((sita-sitamu)*(sita-sitamu))/(24*25)
    ...: sigma=m.sqrt(sigma2)
```

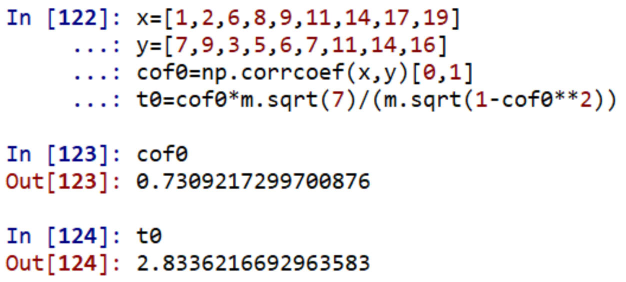


```
In [172]: sigma2
Out[172]: 0.1199024266666695

In [173]: sigma
Out[173]: 0.3462692978978456

In [174]: jackmu+1.96*sigma
Out[174]: 7.5362878238798245

In [175]: jackmu-1.96*sigma
Out[175]: 6.178912176120271
```

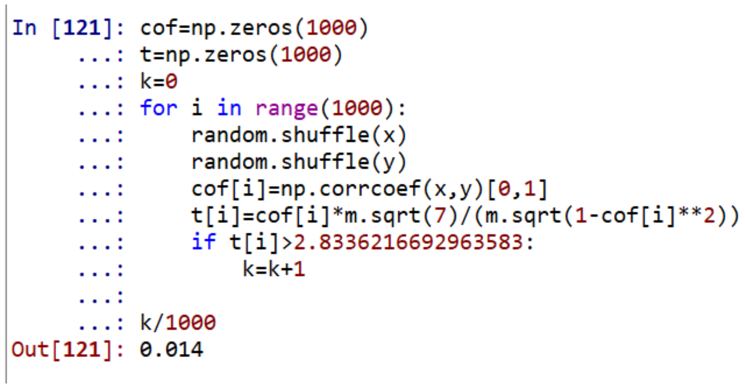


```
In [169]: mu0
Out[169]: 6.8576

In [170]: jackmu
Out[170]: 6.857600000000048
```

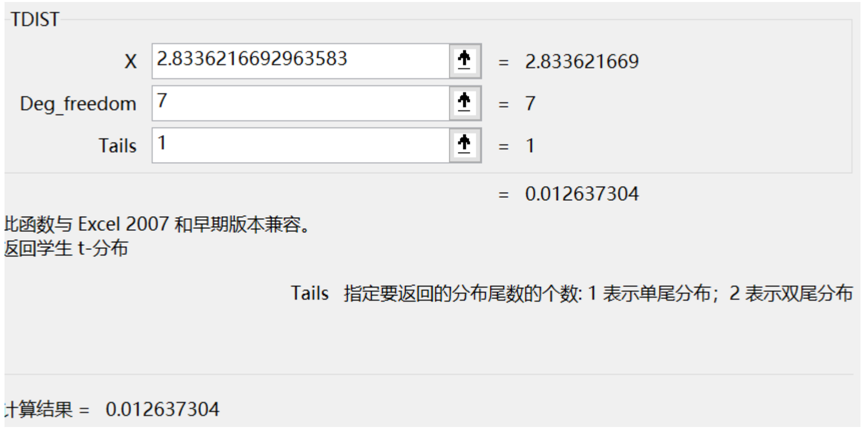
4)



```
In [122]: x=[1,2,6,8,9,11,14,17,19]
     ...: y=[7,9,3,5,6,7,11,14,16]
     ...: cof0=np.corrcoef(x,y)[0,1]
     ...: t0=cof0*m.sqrt(7)/(m.sqrt(1-cof0**2))

In [123]: cof0
Out[123]: 0.7309217299700876

In [124]: t0
Out[124]: 2.8336216692963583
```



```
In [121]: cof=np.zeros(1000)
     ...: t=np.zeros(1000)
     ...: k=0
     ...: for i in range(1000):
     ...:     random.shuffle(x)
     ...:     random.shuffle(y)
     ...:     cof[i]=np.corrcoef(x,y)[0,1]
     ...:     t[i]=cof[i]*m.sqrt(7)/(m.sqrt(1-cof[i]**2))
     ...:     if t[i]>2.8336216692963583:
     ...:         k=k+1
     ...:
     ...: k/1000
Out[121]: 0.014
```



TDIST

X 2.8336216692963583 = 2.833621669

Deg_freedom 7 = 7

Tails 1 = 1

= 0.012637304

比函数与 Excel 2007 和早期版本兼容。

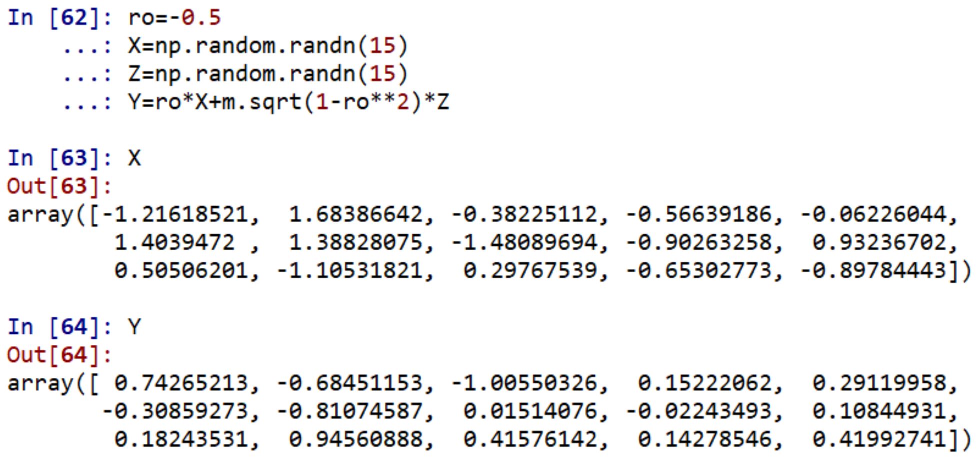
反回学生 t-分布

Tails 指定要返回的分布尾数的个数: 1 表示单尾分布；2 表示双尾分布

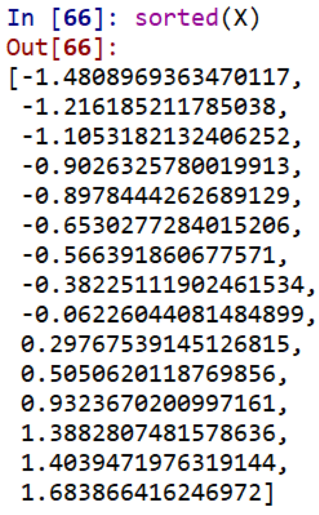
十算结果 = 0.012637304

Real p

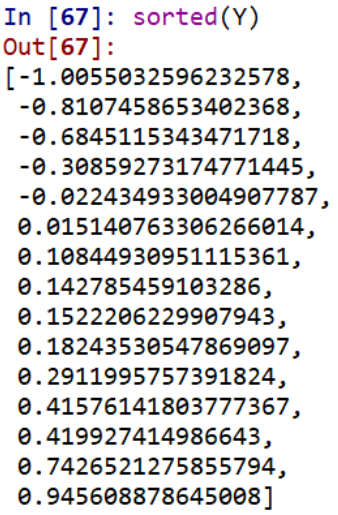
5)



```
In [62]: ro=-0.5
    ...: X=np.random.randn(15)
    ...: Z=np.random.randn(15)
    ...: Y=ro*X+m.sqrt(1-ro**2)*Z

In [63]: X
Out[63]:
array([-1.21618521,  1.68386642, -0.38225112, -0.56639186, -0.06226044,
        1.4039472 ,  1.38828075, -1.48089694, -0.90263258,  0.93236702,
        0.50506201, -1.10531821,  0.29767539, -0.65302773, -0.89784443])

In [64]: Y
Out[64]:
array([ 0.74265213, -0.68451153, -1.00550326,  0.15222062,  0.29119958,
       -0.30859273, -0.81074587,  0.01514076, -0.02243493,  0.10844931,
        0.18243531,  0.94560888,  0.41576142,  0.14278546,  0.41992741])
```



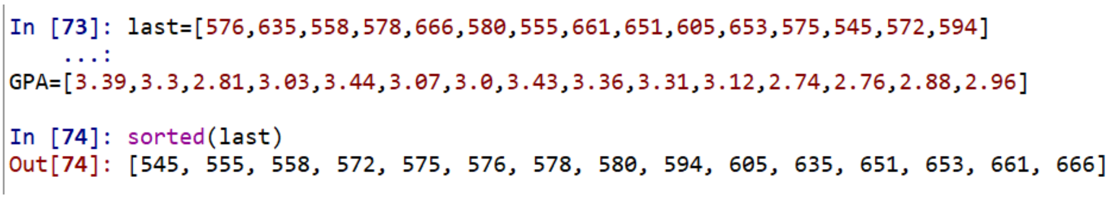
```
In [66]: sorted(X)
Out[66]:
[-1.4808969363470117,
 -1.216185211785038,
 -1.1053182132406252,
 -0.9026325780019913,
 -0.8978444262689129,
 -0.6530277284015206,
 -0.566391860677571,
 -0.38225111902461534,
 -0.06226044081484899,
 0.29767539145126815,
 0.5050620118769856,
 0.9323670200997161,
 1.3882807481578636,
 1.4039471976319144,
 1.683866416246972]
```



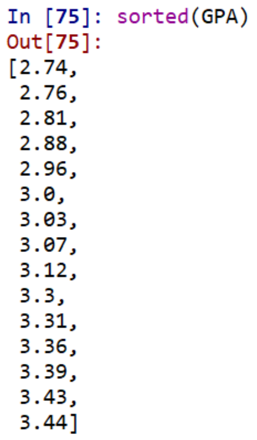
```
In [67]: sorted(Y)
Out[67]:
[-1.0055032596232578,
 -0.8107458653402368,
 -0.6845115343471718,
 -0.30859273174771445,
 -0.022434933004907787,
 0.015140763306266014,
 0.10844930951115361,
 0.142785459103286,
 0.1522206229907943,
 0.18243530547869097,
 0.2911995757391824,
 0.41576141803777367,
 0.419927414986643,
 0.7426521275855794,
 0.945608878645008]
```

RANK X: 2, 15, 8, 7, 9, 14, 13, 1, 4, 12, 11, 3, 10, 6, 5.

RANK Y: 14, 3, 1, 9, 11, 4, 2, 6, 5, 7, 10, 15, 12, 8, 13



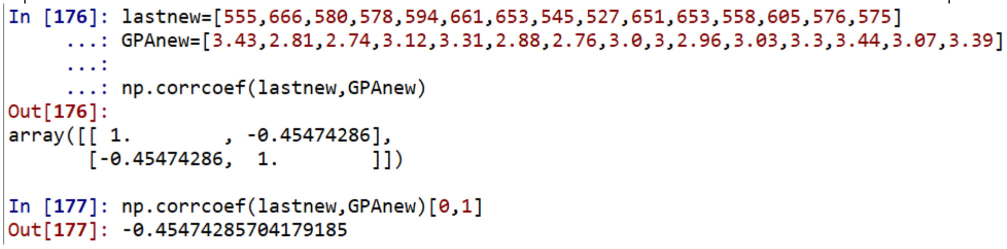
```
In [73]: last=[576,635,558,578,666,580,555,661,651,605,653,575,545,572,594]
    ...:
GPA=[3.39,3.3,2.81,3.03,3.44,3.07,3.0,3.43,3.36,3.31,3.12,2.74,2.76,2.88,2.96]

In [74]: sorted(last)
Out[74]: [545, 555, 558, 572, 575, 576, 578, 580, 594, 605, 635, 651, 653, 661, 666]
```



```
In [75]: sorted(GPA)
Out[75]:
[2.74,
 2.76,
 2.81,
 2.88,
 2.96,
 3.0,
 3.03,
 3.07,
 3.12,
 3.3,
 3.31,
 3.36,
 3.39,
 3.43,
 3.44]
```

LAST 555,666,580,578,594,661,653,545,527,651,653,558,605,576,575

GPA 3.43,2.81,2.74,3.12,3.31,2.88,2.76,3.0,3,2.96,3.03,3.3,3.44,3.07,3.39



```
In [176]: lastnew=[555,666,580,578,594,661,653,545,527,651,653,558,605,576,575]
     ...: GPAnew=[3.43,2.81,2.74,3.12,3.31,2.88,2.76,3.0,3,2.96,3.03,3.3,3.44,3.07,3.39]
     ...:
     ...: np.corrcoef(lastnew,GPAnew)
Out[176]:
array([[ 1.        , -0.45474286],
       [-0.45474286,  1.        ]])

In [177]: np.corrcoef(lastnew,GPAnew)[0,1]
Out[177]: -0.45474285704179185
```