Problem Set 3

Research method Problem Set 3 due Wed 31th Oct, 23:00

LIU BOWEN (1004028)

For Problem Set 3, I use the following packages:

```
import matplotlib.pyplot as plt

import numpy as np

import scipy as sp

import scipy.stats as ss

import scipy.stats as ss

import math
```

Problem 1

Answer:

a)

According to equation, I can obtain the expression for n_2.

$$n_2 = n_1 \frac{\sin \theta_1}{\sin \theta_2}$$

To calculate the PoE I need to obtain the derivative of θ_1 and θ_2 which are $\frac{n_1}{\sin(\theta 1)}\cos(\theta 1)$ and $n_1 \sin(\theta 1)\frac{-\cos(\theta 2)}{(\sin\ \theta 2)^2}$ respectively. The code is shown below:

```
def calPoE(N1, mean1, var1, mean2, var2):

    deriva1 = N1 / np.sin(mean2/180.0*np.pi) * np.cos(mean1/180.0*np.pi)
```

```
    deriva2 = (N1*np.sin(mean1/180.0*np.pi)) * (-np.cos(mean2/180.0*np.pi)) / np.sin(mean2/180.0*np.pi)**2

    poe = var1**2 * deriva1**2 + var2**2 * deriva2**2

    N2 = N1 * np.sin(mean1/180.0*np.pi) / np.sin(mean2/180.0*np.pi)

    return np.sqrt(poe)/180.0*np.pi, N2

N1 = 1.0

mean1 = 22.02

var1 = 0.02

mean2 = 14.45

var2 = 0.02

print calPoE(N1, mean1, var1, mean2, var2)
```

The result is shown in Figure 1, n_2 is 1.5025 $\pm$ 0.0024

```
(0.0024133592559739776, 1.502515309544441)
```

Figure 1

b)

The block flow chart is shown in Figure 2:

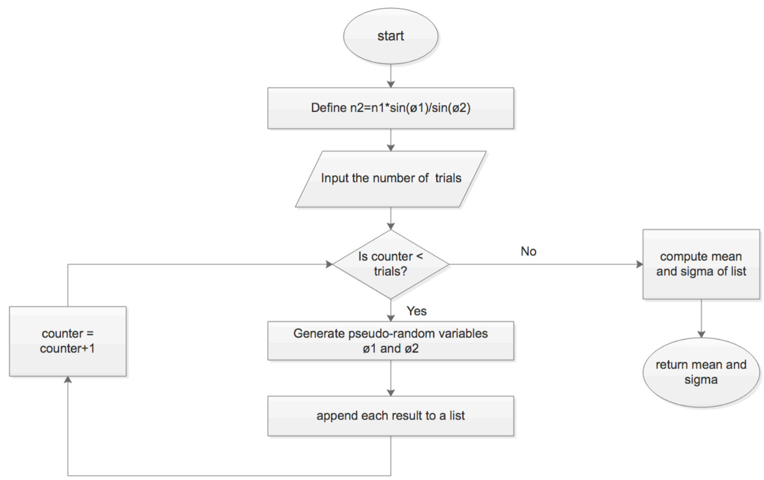

Figure 2

c)

The monte carlo code is below:

```
def tFunc(N1, rad1, rad2):

    return N1 * np.sin(rad1/180.0*np.pi) / np.sin(rad2/180.0*np.pi)

def MC(Num):

    t =[]

    for sample in range(Num):

        rad1 = ss.norm.rvs(22.02, 0.02)

        rad2 = ss.norm.rvs(14.45, 0.02)

        N1 = 1.0

        t.append(tFunc(N1, rad1, rad2))

    mean = np.mean(t)

    stdDeviat = np.std(t, ddof=1)

    return mean, stdDeviat, t
```

```
Num = 10000

his = MC(Num)[2]

plt.figure()

plt.hist(his, bins='auto',color='red', edgecolor='black', align='left')

plt.xlabel('n2')
```

When N=10,100 and 10000, the histogram result is shown:

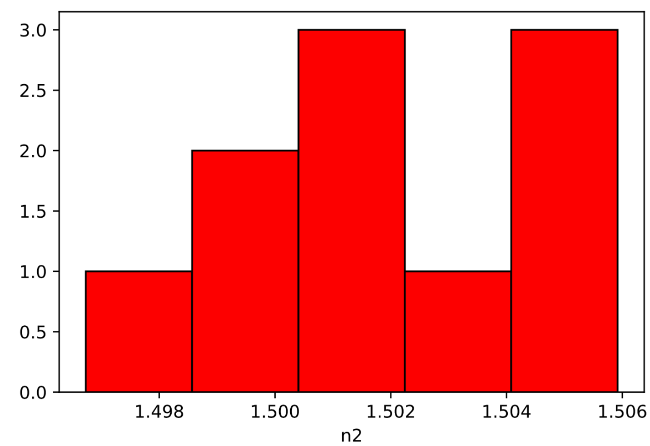

Figure 3. N=10, mean=1.5000, sigma=0.0014

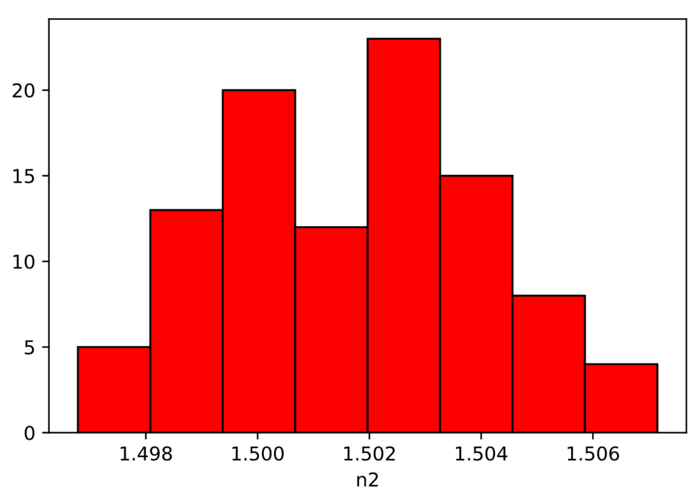

Figure 4. N=100, mean=1.5027, sigma=0.00236

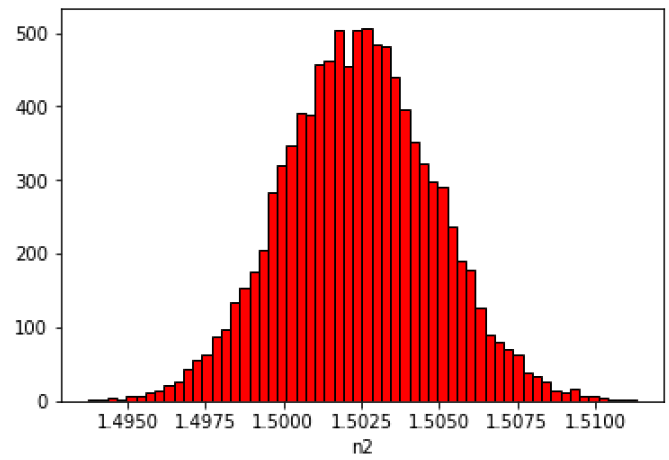

Figure 5. N=10000, mean=1.5025, sigma=0.00243

From the above three histogram, the more computations are, the more precise the measurement as standard error decreases if the sample size increases.

Problem 2

a)

H_0: The proportions from 5 years ago are representative of today's college student proportions

H_A: The proportions from 5 years ago are not representative of today's college student proportions.

b)

The obs array is from the question and expe array is calculated from the percentage for each group. The code is below:

```
def chi():

    obs = [352, 501, 371, 126, 150]

    expe = [375, 525, 375, 150, 75]

    return ss.chisquare(obs,expe)[0], ss.chisquare(obs,expe)[1]
```

```
print chi()
```

The result is shown in Figure 6:

```
(81.39047619047619, 8.83830762853637e-17)
```

Figure 6.

Therefore, the chi^2 statistic is : 81.39 and the P-value is 8.8383e-17

c)

According "if P is low, reject H_0" rule, now P-value is less than 0.01. Therefore, I need to reject H_0 which means the proportions from 5 years ago are not representative of today's college student proportions.

Problem 3

a)

From the description of question, the sample size equals 85 therefore the $\hat{p}$ = 10/85=0.118

b)

For 95% two-sided, I need to calculate the Z0.025. For python, ss.norm(0,1).ppf(1-0.025)=1.96 and ss.norm(0,1).ppf(0.025)=-1.96 respectively. The confidence interval for p is:

$$\hat{p} - z_{0.025}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \leq p \leq \hat{p} + z_{0.025}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

The code is below:

```
def conInterval(N, p, value):

    minimum = p - value * np.sqrt(p*(1-p)/N)

    maximum = p + value * np.sqrt(p*(1-p)/N)
```

```
        return minimum, maximum

N=85

p=10/85.0

value=ss.norm(0,1).ppf(1-0.025)

print conInterval(N, p, value)
```

The answer is 0.049 $\leq p \leq$ 0.186

c)

According to the question, we want to be at least 95% confident that our estimate p^ is within 0.05 of the true proportion, p. Therefore, I can obtain the upper bound on n(where p=0.5).

$$n = \left(\frac{z_{0.025}}{0.05}\right)^2 [p(1-p)]$$

The code is below:

```
def calN(value, Est, p):

        return (value/Est)**2 * p*(1-p)

value=ss.norm(0,1).ppf(1-0.025)

Est = 0.05

p = 0.5

print calN(value, Est, p)
```

The result is n=384.1 so the answer should be 385.