○ 실험보고서

확률과통계 HW#3

제출일 : 06월 24일

2018008559 컴퓨터소프트웨어학부 신상윤

1. Introduction

1.1 실험목적

임의의 1000개 2차원 좌표 (X, Y)에 대하여 Least square을 이용하여 Line fitting을 한 뒤 랜덤한 X에 대하여 구한 식을 이용하여 Y를 예측해본다. 이때 Y 오차의 크기를 분석한다.
표본에서 표본평균, 표본분산을 뽑는 과정을 반복하여 모평균과 모분산을 추정하여 실제 확률 분포의 평균과 분산과 비교하여 분석한다.

1.2 배경 이론

1.2.1 Poisson 분포

푸아송분포는 단위 시간 안에 어떤 사건이 몇 번 발생할 것인지를 표현하는 이산 확률분포이다. 정해진 시간 안에 어떤 사건이 일어날 횟수에 대한 기댓값을 λ라고 했을 때, 그 사건이 n회 일어날 확률을

$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}$로 정의한다. 이때 평균과 분산은 모두 λ이다.

이항 분포의 기댓값을 λ라 하자. $np = \lambda,\ p = \frac{\lambda}{n}$ 즉,

$P(X = x) = \binom{n}{x} (\frac{\lambda}{n})^x (1 - \frac{\lambda}{n})^{n-x} = \frac{\lambda^k}{k!} \frac{n(n-1)\cdots(n-k+1)}{n^k} (1 - \frac{\lambda}{n})^{(-\frac{n}{\lambda})(-\lambda)}$에서

n이 ∞로 갈 때 $P(X = x) = \frac{\lambda^k}{k!} e^{-\lambda}$가 되고, 이는 푸아송분포와 같다.

1.2.2 normal 분포

정규분포는 연속 확률분포 중 하나이며, 수집된 자료의 분포를 근사하는 데에 자주 사용된다. 이는 중심극한정리에 의하여 독립적인 확률변수들의 평균은 정규분포에 가까워지는 성질이 있기 때문이다. 확률 밀도함수는

$N(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ 평균과 분산은 각각 $E(X) = \mu,\ V(X) = \sigma^2$이다.

1.2.3 Box-Muller transform

C에서 따로 정규분포를 따르는 랜덤한 값을 제공해주지 않으므로 직접 코드로 구현해야 한다. C에서 제공해주는 것 중에서 하나가 rand()인데 이는 연속균등분포라 볼 수 있다. 연속균등분포에서 정규분포를 만들수있게 하는 것이 **Box Muller transform** 이다.

가장 핵심적인 내용은 U_1, U_2를 (0,1)에서 균등분포라 할 때 $Z_0 = \sqrt{-2\ln U_1}\cos(2\pi U_2)$,$Z_1 = \sqrt{-2\ln U_1}\sin(2\pi U_2)$는 각각 표준정규분포를 따른다는 것이다. 이를 이용해 정규분포를 구현할 수 있다.

1.2.4 Least square

어떤 실험을 N번 반복할 때 변량 x를 변경해가며 그에 따른 실험값(y) 쌍 (x,y)를 얻었다 하자. 이 데이터의 상관관계를 알아보는 방법 중 하나이다.
어떤 일차함수 $y = ax + b$가 모든 데이터와의 편차가 최소라 하자 이때 편차의 식이 다음과 같다.

$$err = \sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

식을 각각 a와 b에 대하여 미분했을 때 둘다 0을 만족할 때 최솟값을 가질 것이다.

$$a = \frac{\sum_{i=1}^{n}(y_i - \overline{y})(x_i - \overline{x})}{\sum_{i=1}^{n}(x_i - \overline{x})^2},\ b = \overline{y} - a\overline{x}$$

이때 $\overline{x}, \overline{y}$는 각각 x, y의 표본 평균이다.

1.2.5 표본평균과 표본분산

어떤 모집단에서 무작위로 n개의 표본을 추출했을때 이 표본들의 평균과 분산을 각각 표본평균과 표본분산이라고 한다. 표본평균은 직관적으로 모든 표본의 합을 표본의 개수만큼 나눠주면 된다.

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

표본분산은 표본의 개수에서 1을 뺀 값으로 나눠주어야 한다.

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2$$

이는 표본분산의 기대값이 모분산의 $\frac{n-1}{n}$배에 수렴하므로 편향된 값이 나오는 것을 막기 위하여 표본분산에 $\frac{n}{n-1}$배를 한 값으로 모분산을 추정해야 편향되지 않은 값을 얻게 되기 때문이다.

2. Process

변수 X는 저번 실험에서 사용하였던 λ가 100인 푸아송분포에서 생성했고,
변수 Y는 제출일이 6월 24일이라 $Y = 6X + 24 + N(0, 24^2)$로 설정했다.

"C:₩Users₩kas12₩OneDrive₩바탕 화면₩C++₩d₩bin₩Debug₩d.exe"

```
X_mean : 99.911000, Y_mean : 623.330583
a = 5.980524
b = 25.810455
X = 114.000000, Y = 700.561071, ^Y = 707.590144, Diff = 7.029053
X = 117.000000, Y = 693.351457, ^Y = 725.531715, Diff = 32.180298
X = 93.000000, Y = 572.714804, ^Y = 581.999149, Diff = 9.284363
X = 108.000000, Y = 656.063592, ^Y = 671.707003, Diff = 15.643433
X = 95.000000, Y = 597.028410, ^Y = 593.960196, Diff = 3.068176
X = 74.000000, Y = 516.506731, ^Y = 468.369201, Diff = 48.137512
X = 110.000000, Y = 669.782826, ^Y = 683.668050, Diff = 13.885193
X = 112.000000, Y = 674.483914, ^Y = 695.629097, Diff = 21.145203
X = 94.000000, Y = 560.401559, ^Y = 587.979672, Diff = 27.578125
X = 101.000000, Y = 677.079252, ^Y = 629.843338, Diff = 47.235901
X = 106.000000, Y = 619.882979, ^Y = 659.745955, Diff = 39.862976
X = 97.000000, Y = 579.136165, ^Y = 605.921243, Diff = 26.785095
X = 93.000000, Y = 581.590426, ^Y = 581.999149, Diff = 0.408691
X = 99.000000, Y = 614.748043, ^Y = 617.882290, Diff = 3.134216
X = 103.000000, Y = 675.933716, ^Y = 641.804385, Diff = 34.129333
X = 90.000000, Y = 559.426690, ^Y = 564.057578, Diff = 4.630859
X = 115.000000, Y = 758.210826, ^Y = 713.570668, Diff = 44.640137
X = 108.000000, Y = 659.359067, ^Y = 671.707003, Diff = 12.347961
X = 112.000000, Y = 689.540172, ^Y = 695.629097, Diff = 6.088928
X = 88.000000, Y = 555.957196, ^Y = 552.096531, Diff = 3.860657
X = 79.000000, Y = 490.815558, ^Y = 498.271819, Diff = 7.456268
X = 125.000000, Y = 795.584951, ^Y = 773.375904, Diff = 22.209045
X = 93.000000, Y = 598.326074, ^Y = 581.999149, Diff = 16.326904
X = 105.000000, Y = 635.764582, ^Y = 653.765432, Diff = 18.000854
X = 110.000000, Y = 759.195036, ^Y = 683.668050, Diff = 75.526978
X = 103.000000, Y = 681.405354, ^Y = 641.804385, Diff = 39.600952
X = 106.000000, Y = 713.516530, ^Y = 659.745955, Diff = 53.770569
```

먼저 위의 조건으로 표본 1000개를 만들고 계수 a,b를 구한 뒤 랜덤하게 X를 다시 구한뒤 구한 계수를 통해 Y값을 100번 구하고 원래값과의 오차를 구하여 평균을 내 보았다.

```
"C:₩Users₩kas12₩OneDrive₩바탕 화면₩C++₩d₩bin₩Debug₩d.exe"
X = 83.000000, Y = 528.086414, ^Y = 522.193913, Diff = 5.892517
X = 94.000000, Y = 633.409114, ^Y = 587.979672, Diff = 45.429443
X = 102.000000, Y = 638.972088, ^Y = 635.823861, Diff = 3.148254
X = 89.000000, Y = 568.687699, ^Y = 558.077055, Diff = 10.610657
X = 105.000000, Y = 646.933585, ^Y = 653.765432, Diff = 6.831848
X = 101.000000, Y = 676.260318, ^Y = 629.843338, Diff = 46.416992
X = 105.000000, Y = 636.186956, ^Y = 653.765432, Diff = 17.578491
X = 107.000000, Y = 662.471017, ^Y = 665.726479, Diff = 3.255493
X = 107.000000, Y = 662.071334, ^Y = 665.726479, Diff = 3.655151
X = 89.000000, Y = 563.597614, ^Y = 558.077055, Diff = 5.520569
X = 110.000000, Y = 664.492851, ^Y = 683.668050, Diff = 19.175171
X = 108.000000, Y = 697.477452, ^Y = 671.707003, Diff = 25.770447
X = 93.000000, Y = 576.715368, ^Y = 581.999149, Diff = 5.283752
X = 90.000000, Y = 524.931602, ^Y = 564.057578, Diff = 39.125977
X = 111.000000, Y = 696.119661, ^Y = 689.648573, Diff = 6.471130
X = 101.000000, Y = 640.047580, ^Y = 629.843338, Diff = 10.204285
X = 101.000000, Y = 634.402301, ^Y = 629.843338, Diff = 4.558960
X = 93.000000, Y = 550.408551, ^Y = 581.999149, Diff = 31.590576
X = 98.000000, Y = 640.905370, ^Y = 611.901767, Diff = 29.003601
X = 109.000000, Y = 668.389564, ^Y = 677.687526, Diff = 9.297913
X = 95.000000, Y = 595.396440, ^Y = 593.960196, Diff = 1.436218
X = 120.000000, Y = 716.027025, ^Y = 743.473286, Diff = 27.446228
X = 106.000000, Y = 646.831576, ^Y = 659.745955, Diff = 12.914368
X = 98.000000, Y = 624.446421, ^Y = 611.901767, Diff = 12.544617
X = 109.000000, Y = 665.981213, ^Y = 677.687526, Diff = 11.706299
|^Y - Y| = 20.043303
Process returned 0 (0x0)   execution time : 0.246 s
Press any key to continue.
```

마찬가지로 모든 조건을 똑같이 하고 (X, Y) 좌표를 10000회 발생시켜 보았다.

```
"C:₩Users₩kas12₩OneDrive₩바탕 화면₩C++₩d₩bin₩Debug₩d.exe"
X_mean : 100.190200, Y_mean : 625.328916
a = 6.031605
b = 21.021225
X = 90.000000, Y = 548.190171, ^Y = 563.865654, Diff = 15.675476
X = 117.000000, Y = 720.839381, ^Y = 726.718983, Diff = 5.879639
X = 119.000000, Y = 717.419391, ^Y = 738.782192, Diff = 21.362793
X = 83.000000, Y = 524.372446, ^Y = 521.644421, Diff = 2.728027
X = 108.000000, Y = 671.692850, ^Y = 672.434540, Diff = 0.741699
X = 93.000000, Y = 561.843724, ^Y = 581.960468, Diff = 20.116699
X = 89.000000, Y = 551.324487, ^Y = 557.834049, Diff = 6.509583
X = 86.000000, Y = 569.682587, ^Y = 539.739235, Diff = 29.943359
X = 119.000000, Y = 733.703361, ^Y = 738.782192, Diff = 5.078796
X = 107.000000, Y = 696.441874, ^Y = 666.402935, Diff = 30.038940
X = 108.000000, Y = 687.761242, ^Y = 672.434540, Diff = 15.326660
X = 106.000000, Y = 664.848849, ^Y = 660.371330, Diff = 4.477539
X = 92.000000, Y = 534.809479, ^Y = 575.928864, Diff = 41.119324
X = 114.000000, Y = 699.648845, ^Y = 708.624168, Diff = 8.975281
X = 98.000000, Y = 583.165767, ^Y = 612.118492, Diff = 28.952698
X = 97.000000, Y = 613.609155, ^Y = 606.086887, Diff = 7.522217
X = 101.000000, Y = 616.387064, ^Y = 630.213306, Diff = 13.826233
X = 95.000000, Y = 586.016028, ^Y = 594.023678, Diff = 8.007629
X = 108.000000, Y = 639.941736, ^Y = 672.434540, Diff = 32.492859
X = 89.000000, Y = 509.293584, ^Y = 557.834049, Diff = 48.540466
X = 107.000000, Y = 658.534249, ^Y = 666.402935, Diff = 7.868713
X = 102.000000, Y = 617.870160, ^Y = 636.244911, Diff = 18.374756
X = 103.000000, Y = 650.841685, ^Y = 642.276516, Diff = 8.565186
X = 94.000000, Y = 576.774350, ^Y = 587.992073, Diff = 11.217712
X = 99.000000, Y = 601.034582, ^Y = 618.150097, Diff = 17.115479
X = 98.000000, Y = 614.802449, ^Y = 612.118492, Diff = 2.683960
X = 101.000000, Y = 631.628358, ^Y = 630.213306, Diff = 1.415039
```

```
"C:₩Users₩kas12₩OneDrive₩바탕 화면₩C++₩d₩bin₩Debug₩d.exe"
X = 97.000000, Y = 594.109085, ^Y = 606.086887, Diff = 11.977844
X = 96.000000, Y = 597.889109, ^Y = 600.055283, Diff = 2.166199
X = 115.000000, Y = 727.573548, ^Y = 714.655773, Diff = 12.917786
X = 106.000000, Y = 624.227443, ^Y = 660.371330, Diff = 36.143921
X = 84.000000, Y = 542.627606, ^Y = 527.676025, Diff = 14.951599
X = 87.000000, Y = 499.173654, ^Y = 545.770840, Diff = 46.597168
X = 89.000000, Y = 582.812343, ^Y = 557.834049, Diff = 24.978271
X = 96.000000, Y = 581.302201, ^Y = 600.055283, Diff = 18.753113
X = 90.000000, Y = 565.670567, ^Y = 563.865654, Diff = 1.804932
X = 109.000000, Y = 717.208641, ^Y = 678.466145, Diff = 38.742493
X = 87.000000, Y = 577.955127, ^Y = 545.770840, Diff = 32.184326
X = 95.000000, Y = 627.452633, ^Y = 594.023678, Diff = 33.428955
X = 110.000000, Y = 736.631274, ^Y = 684.497749, Diff = 52.133545
X = 99.000000, Y = 611.009369, ^Y = 618.150097, Diff = 7.140686
X = 97.000000, Y = 600.188639, ^Y = 606.086887, Diff = 5.898254
X = 109.000000, Y = 657.432793, ^Y = 678.466145, Diff = 21.033325
X = 108.000000, Y = 687.202086, ^Y = 672.434540, Diff = 14.767517
X = 94.000000, Y = 586.177105, ^Y = 587.992073, Diff = 1.814941
X = 85.000000, Y = 540.057888, ^Y = 533.707630, Diff = 6.350220
X = 85.000000, Y = 614.776445, ^Y = 533.707630, Diff = 81.068787
X = 100.000000, Y = 615.734084, ^Y = 624.181702, Diff = 8.447632
X = 102.000000, Y = 651.984215, ^Y = 636.244911, Diff = 15.739258
X = 84.000000, Y = 530.220359, ^Y = 527.676025, Diff = 2.544312
X = 90.000000, Y = 552.449680, ^Y = 563.865654, Diff = 11.415955
X = 116.000000, Y = 740.491224, ^Y = 720.687378, Diff = 19.803833
|^Y - Y| = 17.656181
Process returned 0 (0x0)   execution time : 0.466 s
Press any key to continue.
```

다음 실험에도 역시 제출일인 6월 24일을 참고하여 $N(6, 24^2)$에서 실험을 진행하였다.

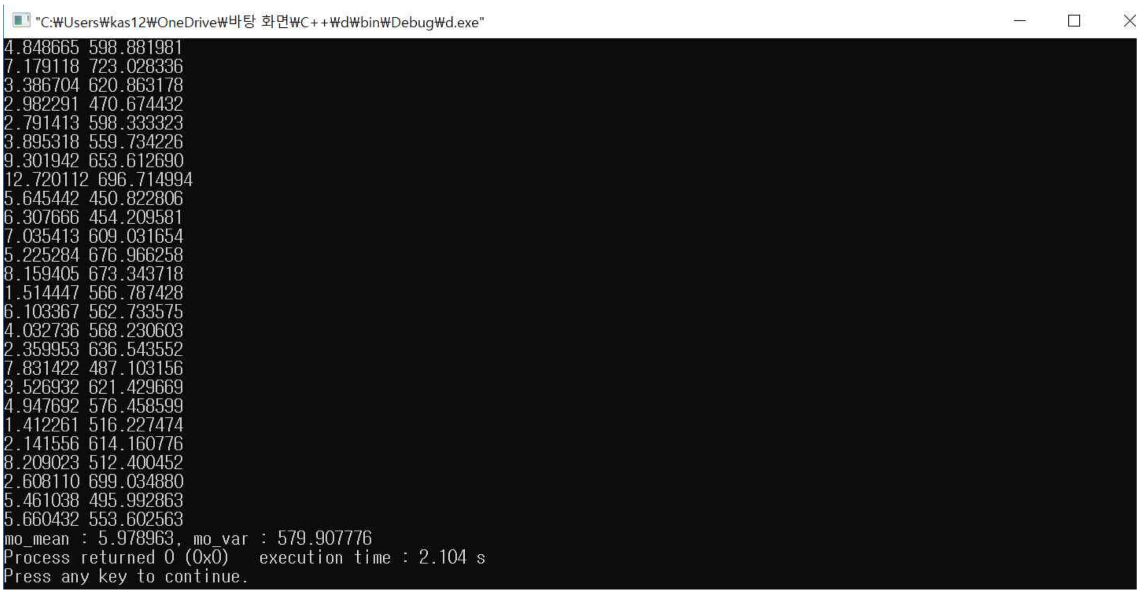
10회 반복

```
"C:₩Users₩kas12₩OneDrive₩바탕 화면₩C++₩d₩bin₩Debug₩d.exe"
5.826666 597.920780
6.816248 493.071937
7.744878 546.727698
8.880412 571.937873
8.110526 734.215415
7.264851 626.089968
4.497274 564.480355
5.057701 738.091497
8.007991 539.701914
10.041332 783.020533
mo_mean : 7.224787, mo_var : 619.525757
Process returned 0 (0x0)   execution time : 0.044 s
Press any key to continue.
```

100회 반복

```
선택 "C:₩Users₩kas12₩OneDrive₩바탕 화면₩C++₩d₩bin₩Debug₩d.exe"
6.443275 490.231316
7.295632 505.830061
3.712557 710.300326
5.859549 437.459097
3.153915 463.539262
-0.104099 611.596768
6.449737 486.194308
0.593088 568.631285
3.084458 749.886942
4.163756 539.086230
4.387155 531.148823
8.425580 618.632153
3.707415 651.359030
6.701302 528.648287
8.749235 523.454683
2.412662 714.740648
5.515782 539.554084
7.271991 604.906153
9.201099 577.258979
3.497526 567.970801
4.454191 704.559528
5.672541 512.375241
9.462826 563.206387
5.728530 557.959478
6.409344 575.946998
mo_mean : 6.148893, mo_var : 587.915283
Process returned 0 (0x0)   execution time : 0.150 s
Press any key to continue.
```

1000회 반복



3. Result

3.1 결과 분석

1000번, 10000번을 반복했을 때 Y의 오차를 10번씩 구해보았다.

1000번	10000번
22.18304	16.410930
19.33296	17.042852
20.34214	17.291721
21.23424	18.419890
16.75431	17.321057
19.52167	19.537652
19.52905	19.227498
19.54257	19.602632
21.18775	18.707249
19.03709	17.905762

좌표가 많을수록 오차가 많이 줄어들 줄 알았지만 큰 차이는 없었다. 또한 오차가 16 ~ 20에 머물렀다.

주제 1과 다르게 주제 2에서는 확실히 표본이 많을수록 평균과 분산이 점점 원래의 확률 분포의 평균과 분산에 가까워짐을 볼 수 있었다.

이는 표본추출의 의의를 알 수 있게 해준다. 표본을 많이 뽑는다고 표본평균, 표본분산이 원래 확률 분포에 가까워지는 것이 아니라 표본을 해석한 표본평균, 표본분산을 여러번 추출하여 구한 모평균, 모분산이 분포에 더 가깝다.

3.2.1 주제1 코드

```
#include <stdio.h>
#include <math.h>
#include <Windows.h>
#include <time.h>
float abs(float a,float b){
    if(a>b) return a - b;
    else return b - a;
}
double gaussianRandom(double average, double stdev){
  double v1, v2, s, temp;
  do {
    v1 =  2 * ((double) rand() / RAND_MAX) - 1;      // -1.0 ~ 1.0 까지의 값
    v2 =  2 * ((double) rand() / RAND_MAX) - 1;      // -1.0 ~ 1.0 까지의 값
    s = v1 * v1 + v2 * v2;
  } while (s >= 1 || s == 0);
  s = sqrt( (-2 * log(s)) / s );
  temp = v1 * s;
  temp = (stdev * temp) + average;
  return temp;
};

int main(void) {
    double a[1001] = {0};
    double b[1001] = {0};
    double sum_x = 0,sum_y = 0;
    srand(time(NULL));
    for(int i=0; i<1000; i++){
        double cnt = 0;
```

```
        for(int j=0; j<1000; j++){
            if(rand()%10 == 0)
                cnt++;
        }
        a[i] = cnt;
        double n = 6*cnt + 24 + gaussianRandom(0, 24);
        b[i] = n;
        sum_x += a[i];
        sum_y += b[i];
    }
    printf("X_mean : %f, Y_mean : %f\n",sum_x/1000,sum_y/1000);
    float mean_x = sum_x/1000, mean_y = sum_y/1000;
    float sum_x_2 = 0, sum_x_y = 0;
    for(int i=0; i<1000; i++)
    {
        sum_x_2 += (a[i] - mean_x)*(a[i] - mean_x);
        sum_x_y += (a[i] - mean_x)*(b[i] - mean_y);
    }
    float cof_a = sum_x_y/sum_x_2;
    printf("a = %f\n",cof_a);
    float cof_b = mean_y - cof_a * mean_x;
    printf("b = %f\n",cof_b);

    double sum_diff = 0;
    for(int i=0; i<100; i++){
        double cnt = 0;
        for(int j=0; j<1000; j++){
            if(rand()%10 == 0)
                cnt++;
        }
        double n = 6*cnt + 24 + gaussianRandom(0, 24);
        printf("X = %f, Y = %f, ^Y = %f, Diff =
%f\n",cnt,n,cof_a*cnt+cof_b,abs(n,cof_a*cnt+cof_b));
        sum_diff += abs(n,cof_a*cnt+cof_b);
    }
    printf("|^Y - Y| = %f",sum_diff/1000);
    return 0;
}
```

3.2.1 주제2 코드

```
#include <stdio.h>
#include <math.h>
#include <Windows.h>
#include <time.h>
```

```
double gaussianRandom(double average, double stdev){
  double v1, v2, s, temp;
  do {
    v1 =  2 * ((double) rand() / RAND_MAX) - 1;      // -1.0 ~ 1.0 까지의 값
    v2 =  2 * ((double) rand() / RAND_MAX) - 1;      // -1.0 ~ 1.0 까지의 값
    s = v1 * v1 + v2 * v2;
  } while (s >= 1 || s == 0);
  s = sqrt( (-2 * log(s)) / s );
  temp = v1 * s;
  temp = (stdev * temp) + average;
  return temp;
};

int main(void) {
    double a[10001] = {0};
    double b[10001] = {0};
    double mean[1001] = {0};
    double var[1001] = {0};
    double sum_x = 0,sum_y = 0;
    srand(time(NULL));

    for(int j=0; j<1000; j++){
        for(int i=0; i<100; i++){
            a[i] = gaussianRandom(6, 24);
            sum_x += a[i];
        }
        sum_x /= 100;
        for(int i=0; i<100; i++){
            sum_y += (a[i] - sum_x)*(a[i] - sum_x);
        }
        sum_y /= 99;
        mean[j] = sum_x;
        var[j] = sum_y;
    }
    float mo_x = 0, mo_y = 0;
    for(int i=0; i<1000; i++){
        printf("%f %f\n",mean[i],var[i]);
        mo_x += mean[i];
        mo_y += var[i];
    }
    mo_x /= 1000;
    mo_y /= 1000;
    printf("mo_mean : %f, mo_var : %f",mo_x,mo_y);
    return 0;
}
```

4. Discussion

4.1 주제1에서의 오차의 값

주제1 실험에서 오차는 대략 16 ~ 22의 값이 나타났다. a의 값의 오차가 작다고 할 때 오 차는 b의 값에서의 오차와 정규분포 $N(0,24^2)$에서의 랜덤값 만큼의 차이가 날 것이다. 정규분포의 분산이 클수록 넓게 퍼져 0에서 먼 값이 나올 확률이 높아진다. 따라서 정규분포의 분산이 작을수록 주제1에서의 오차의 값도 줄어들 것으로 예상된다.

4.2 a와 b

a의 값은 표본들의 평균을 이용하여 직접 계산을 한 것에 반해 b의 값은 단순하게 각 표본의 x, y 평균과 앞에서 구한 a를 통하여 구한 값이다. 따라서 a의 값보다 b의 값에서의 오차가 훨씬 크다. 이는 기울기를 여러 표본을 통해서 구하고 나면 y절편인 b값은 자동적으로 정해지기 때문이라고 예상된다.

4.3 σ의 값

σ의 값이 작을수록 실험의 변화를 알기 힘들 것이라 예상해 σ의 값을 크게 잡았다. 주제2는 단순하게 모집단을 통해 평균, 분산을 구하는 것 이므로 그 어떤 값도 상관없을 것이다. 주제1에서는 앞에서 예상했듯이 σ가 작아질수록 오차가 작아질 것이고, σ가 커질수록 오차가 커질 것이다.

5. Reference

[1] https://ko.wikipedia.org/

[2] https://homepage.stat.uiowa.edu/~mbognar/applets/

[3] https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform