# 模擬與統計計算 HW3

電機所 N26120870 林耕澤

## Homework

Write three Geometric random number generator programs, the first one is using p and q and simulating the experiment results until it is a success. The second one is use traditional **Inverse Transform method** to find it. The third one is to use X= Int(log(U)/log(q))+1. Discuss the r.v. histogram, the algorithm run time and explain why.

Input: p can change if you want to observe the change of the distribution.

## 方法一:Using P and Q

```python
def geo_p_q(p):
    count = 1
    while random.random() > p:
        count += 1
    return count
```

figure 1: python 版本

```cpp
int geo_p_q(double p) {
    int count = 1;
    while ((double)rand() / RAND_MAX > p) {
        count++;
    }
    return count;
}
```

figure 2: C++版本

程式碼簡介:

給定一個變數 count 給他一個初始值為 1，用以表示第幾次成功。每當 random.random()(用以產生 0~1 之間隨機浮點數)值大於 P(表示成功機率)表示失敗必續繼續進行實驗並且 count+1，反之則為成功並停止實驗回傳遞幾次成功。

## 方法二:Inverse Transform method

```python
def geo_r_i_t(p):
    U = random.random()
    i = 1
    count = 1
    baseline = p*(1-p)**(i-1)
    while U >= baseline:
        count += 1
        i += 1
        baseline += p*(1-p)**(i-1)

    return count
```

figure 3:python 版本

```cpp
int geo_r_i_t(double p) {
    double U = (double)rand() / RAND_MAX;
    int i = 1;
    int count = 1;
    double baseline = p * pow(1 - p, i - 1);
    while (U >= baseline) {
        count++;
        i++;
        baseline+=p * pow(1 - p, i - 1);
    }
    return count;
}
```

figure 4:C++ 版本

程式碼簡介:

首先在 0~1 之間隨機取一值 U，接著再宣告變數 i、count 以及 baseline (i 用來迭代、count 用以表示第幾次成功、baseline 用以表示第 i 次以內成功的機率)，如果隨機數 U 大於等於 baseline 則 count+1、i+1 且累加第 i 次成功的機率到 baseline，反之則回傳 count(第幾次成功)。

## 方法三:公式法

```python
def geo_formula(p):
    U = random.random()
    X = math.floor(math.log(U) / math.log(1 - p)) + 1
    return X
```

figure 5: python 版本

```cpp
int geo_formula(double p) {
    double U = (double)rand() / RAND_MAX;
    int X = floor(log(U) / log(1 - p)) + 1;
    return X;
}
```

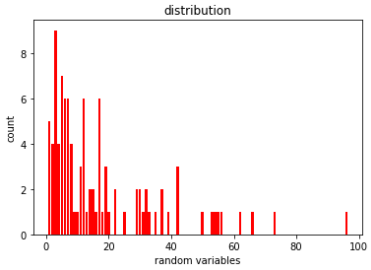
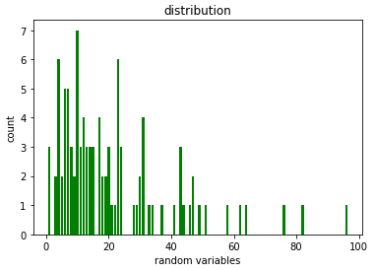
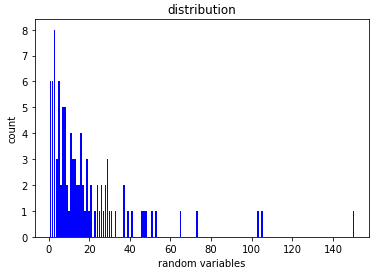
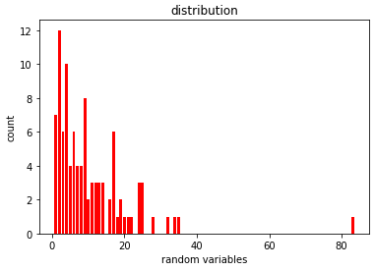
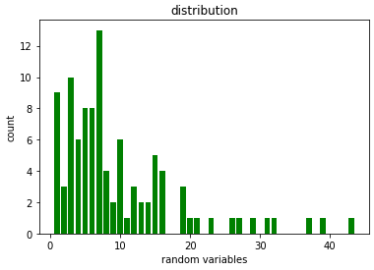
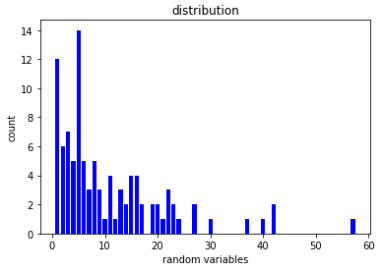
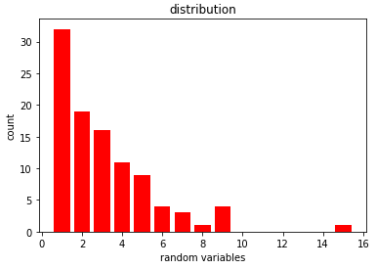
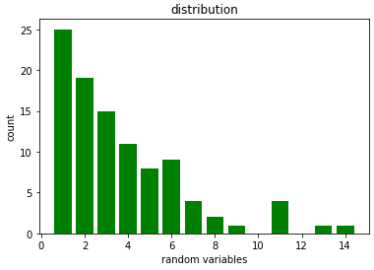
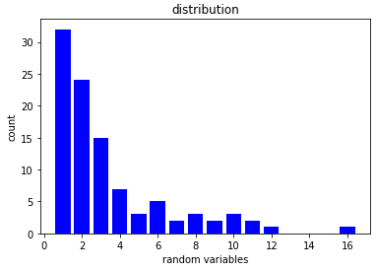
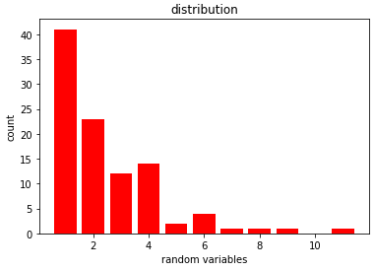
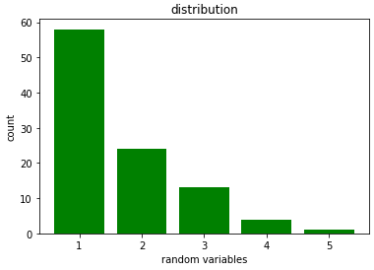
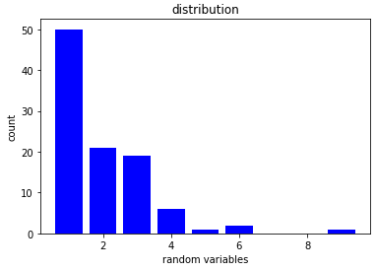
figure 6:C++版本

程式碼簡介:

首先在 0~1 之間隨機取一值 U，然後帶入公式。

## 模擬結果:

**p 值對分布之影響:**

| | Using P and Q | Inverse Transform method | 公式法 |
|---|---|---|---|
| p=0.05 |  |  |  |
| p=0.1 |  |  |  |
| p=0.3 |  |  |  |
| p=0.5 |  |  |  |

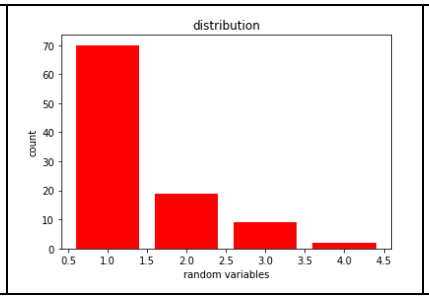
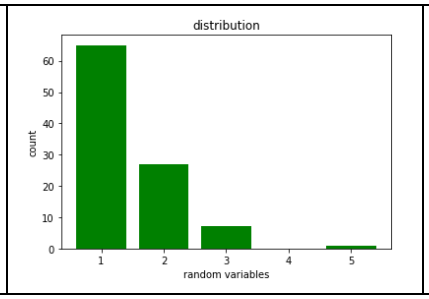
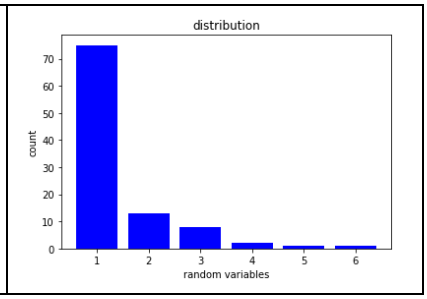
| | | | |
|---|---|---|---|
| p=0.7 |  |  |  |

table 1: p 值對分布之影響

如上圖可得知成功機率(p)下降時，隨機變數之分布的也會隨之變的比較趨緩(分布較為均勻)，反之，則變得較為陡峭(分布集中在 X 較小之處)。

**p 值對執行時間之影響:**

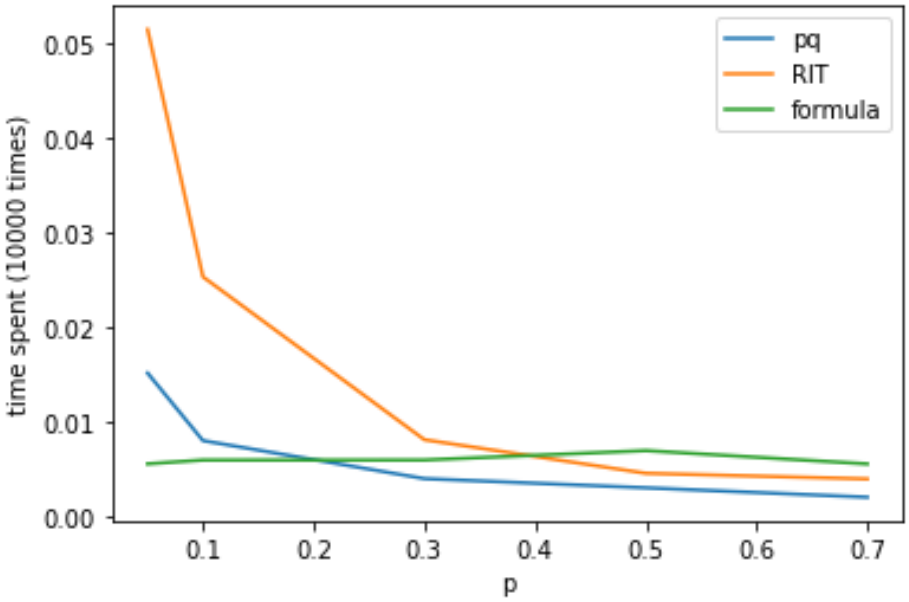



figure 7:p 值對執行時間之影響(python 生成隨機變數 10000 次之執行時間)

```
p = 0.1:
(geo_p_q)  :  2006ms
(geo_r_i_t)  :  2714ms
(geo_formula)  :  411ms
p = 0.3:
(geo_p_q)  :  800ms
(geo_r_i_t)  :  1008ms
(geo_formula)  :  288ms
p = 0.5:
(geo_p_q)  :  385ms
(geo_r_i_t)  :  588ms
(geo_formula)  :  391ms
p = 0.7:
(geo_p_q)  :  303ms
(geo_r_i_t)  :  398ms
(geo_formula)  :  286ms
```

figure 8: p 值對執行時間之影響(C++ 生成隨機變數 10000 次之執行時間)

由 figure7 以及 figure8 可以發現不論是在 python 或是 C++，公式法的執行時間大致上不太受影響，而另外兩種方會都會受到蠻大的影響，我認為主要原因為公式法之外的兩種方法都會因為 p 值低(成功機率低)而大幅上升執行次數。除此之外，我們可以觀察到當 p 值較大時在 python 版本中，相較於公式法，另外兩中方法的執行時間會比較短一些(其實也沒差多少)，我認為主要原因為 python 要 call 函式庫的函式會比較耗時，而公式法也相較其他兩種方法計算較為複雜且使用到的函式較多。