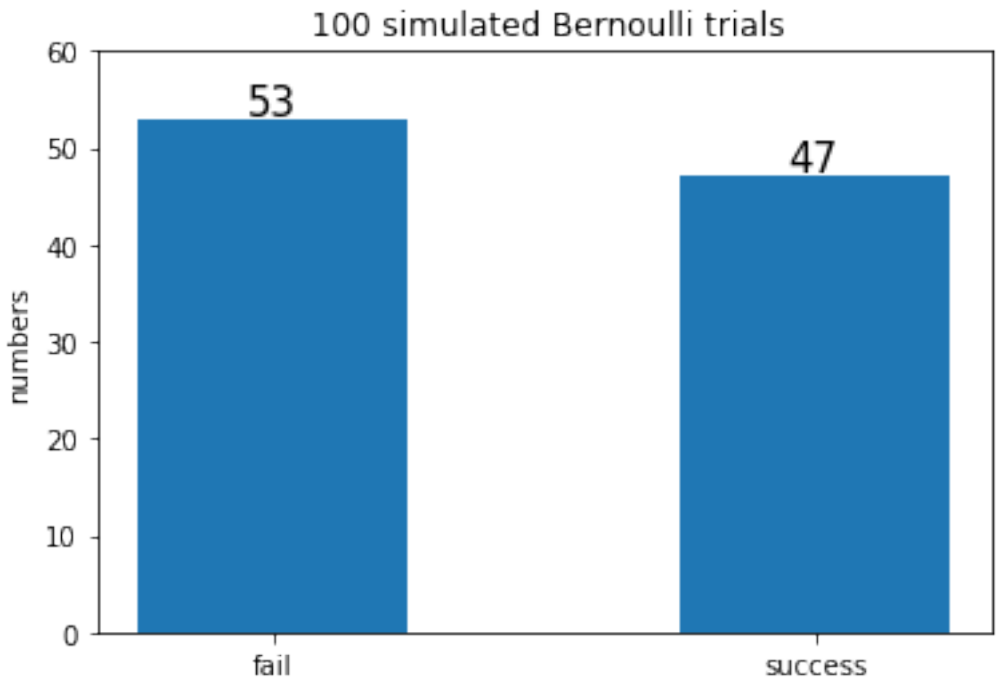
# EE503_Project1

February 7, 2019

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random

%matplotlib inline

class MyBernoulli:

    def __init__(self, p, size=10):
        self.p = p
        self.size = size

    def one_bernoulli(self):
        out = random.uniform(0, 1)
        if out <= self.p:
            out = 0
        else:
            out = 1
        return out

    def mul_bernoulli(self):
        out_list = []
        success = 0
        for i in range(0, self.size):
            tmp = self.one_bernoulli()
            if tmp == 1:
                success += 1
            out_list.append(tmp)
        return out_list, success


def cal_kth_moment(input_list, k):
    kth_moment = 0
    length = len(input_list)
    size = np.sum(input_list)
    for i in range(0, length):
        kth_moment += (input_list[i] / size) * (i**k)
```

```
    return kth_moment


def cal_var(input_list):
    return cal_kth_moment(input_list, 2) - (cal_kth_moment(input_list, 1))**2

# Three distributions based on Bernoulli trials

# 1. Write a routine to simulate a fair Bernoulli trial in your language of
# choice. Generate a histogram for 100 simulated Bernoulli trials

# Initiate
p = 0.5
size = 100
problem_1 = MyBernoulli(p, size=size)
# Experiment and Statistics
res, success = problem_1.mul_bernoulli()
p, t = np.histogram(res, bins=2, range=(0, 1))
# Calculate the expected and variance
bernoulli_1_mean = cal_kth_moment(p, 1)
bernoulli_1_var = cal_var(p)
# Plot
ind = np.arange(2)
width = 0.5
p1 = plt.bar(ind, p, width)
plt.ylabel('numbers')
plt.title(' 100 simulated Bernoulli trials')
plt.xticks(ind, ('fail', 'success'))
plt.yticks(np.arange(0, 61, 10))
for i in range(0, 2):
    plt.text(i-0.05, p[i]+0.5, str(p[i]), fontsize=15)
# Display the result
plt.show()
print("Expected value:", bernoulli_1_mean)
print("bernoulli_1_var", bernoulli_1_var)
```




```
Expected value: 0.47
bernoulli_1_var 0.2491
```

1.Simulation Methodology

I create a class named MyBernoulli to simulate the Bernoulli experiment. To be specific, I initiate the class with parameter p which denotes the probability of success( or head of the coin) and size which denotes the numbers of repeating the experiment. Then I use uniform function to generate a random number between 0 and 1. If the number less than p, I assume it is the head of a coin. On the other hand, if the number is not less than p, I assume it is the tail of a coin. And I count the number of success in the size times experiments.

Secondly, I create a function called cal_kth_moment that returns the kth moment of the input list in order to calculate the expected value of the experiment and provide the value required in the function cal_var which calculates the variance of the experiment.

2.Theoretical Exploration or Analysis

If X is a random variable with Bernoulli distribution, then:

$$P(X = 1) = p = 1 - P(X = 0) = 1 - q$$

The expected value of a Bernoulli random variable X is

$$E[X] = P(X = 1) * 1 + P(X = 0) * 0 = p$$

The variance of a Bernoulli distributed X is

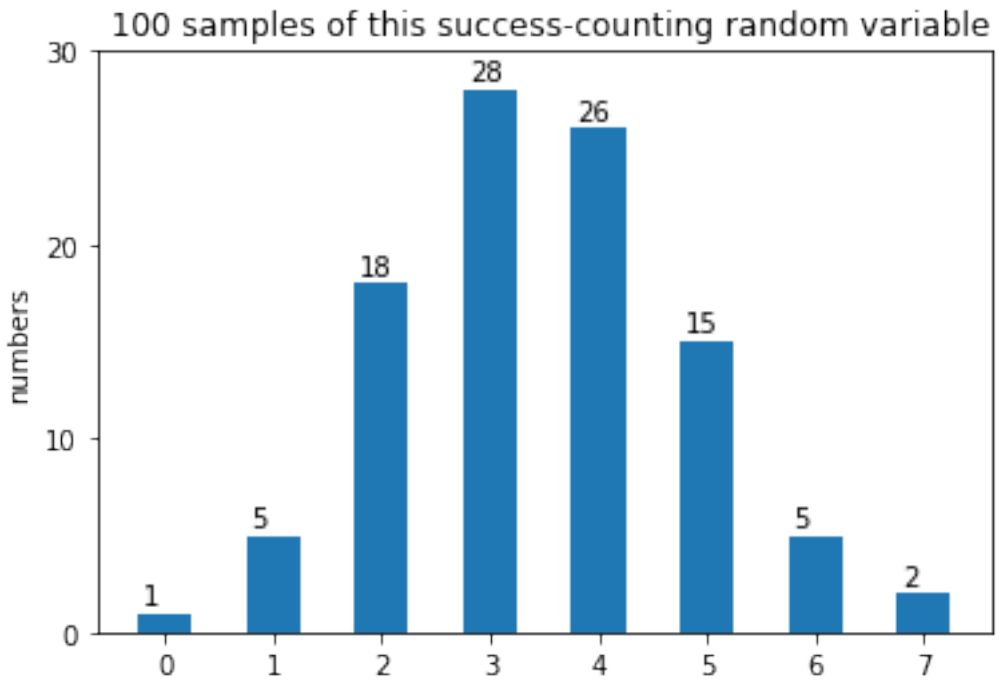
$$Var[X] = E[X^2] - E[X]^2$$

Theoretically, the number of heads in 100 times of tossing would be 50 and the expected value is 0.5, and the variance of the experiment is 0.25.

3.Experiments and Results

The result of simulation shows that the number of heads in 100 flips is around 50 which is very similar to the probability and the variance also accord with the probability. As the number of repetitions increases, the result is closer to the theoretical value.

In [17]:
```
# 2. Write a routine to count the number of successes in 7 fair Bernoulli trials.
# Generate a histogram for 100 samples of this success-counting random variable.
# Initiate
n = 100
size, p = 7, 0.5
p2 = MyBernoulli(p, size=size)
out_list = []
out_str = ''
# Experiment and Statistics
for i in range(0, n):
    res, success = p2.mul_bernoulli()
    out_list.append(success)
p, t = np.histogram(out_list, bins=8, range=(0, 8))
# Calculate the expected and variance
bernoulli_2_mean = cal_kth_moment(p, 1)
bernoulli_2_var = cal_var(p)
# Plot
ind = np.arange(8)
width = 0.5
p1 = plt.bar(ind, p, width)
plt.ylabel('numbers')
plt.title(' 100 samples of this success-counting random variable')
plt.xticks(ind, ('0', '1', '2', '3', '4', '5', '6', '7'))
plt.yticks(np.arange(0, 31, 10))
for i in range(0, 8):
    out_str += '%d:' % i + '%' + '%d, ' % p[i]
    plt.text(i-0.2, p[i]+0.5, str(p[i]), fontsize=10)
# Display the result
plt.show()
print("Probability of each elements:", out_str)
print("Expected value:", bernoulli_2_mean)
print("Variance:", bernoulli_2_var)
```




```
Probability of each elements: 0:%1, 1:%5, 2:%18, 3:%28, 4:%26, 5:%15, 6:%5, 7:%2,
Expected value: 3.48
Variance: 1.869600000000002
```

1.Theoretical Exploration or Analysis

The binomial distribution with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent experiments, which accord with our experiment. To be specific, Bernoulli experiment is independent experiment and we count the number of successes in 7 fair Bernoulli trials which satisfies the definition of binomial distribution with n=7 and p=0.5

In general, if the random variable X follows the binomial distribution with parameters n and p [0,1], we write X ~ B(n, p). The probability of getting exactly k successes in n trials is given by the probability mass function:

$$P(k,n,p) = P(X = k) = \binom{n}{k} \frac{x}{y+2} \qquad \text{for} \qquad k = 0,1,2,...,n \tag{1}$$

The expected value of a binomial random variable X is

$$E[X] = E[X_1 + X_2 + ... + X_n] = E[X_1] + E[X_2] + ... + E[X_n] = np$$

where all X_{i} are independently Bernoulli distributed random variables

The variance of a binomial distributed X is

$$Var[X] = Var[X_1 + X_2 + ... + X_n] = Var[X_1] + Var[X_2] + ... + Var[X_n] = np(1-p)$$

where all X_{i} are independently Bernoulli distributed random variables

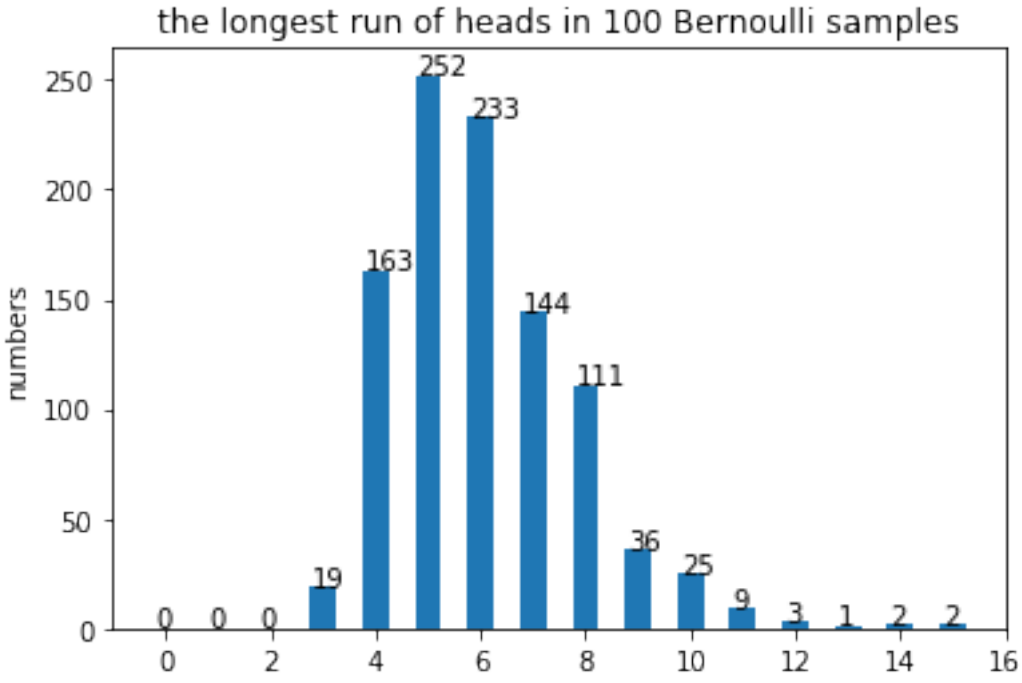
Theoretically, we get 7 heads in 7 fair Bernoulli trials so the sample space is {0, 1, 2, 3, 4, 5, 6, 7} and the probability of each of elements are 1/128, 7/128, 21/128, 35/128, 35/128, 21/128, 7/128, 1/128. Moreover, the expected value is 3.5 and the variance of the experiment is 1.75.

2.Experiments and Results

The result of simulation shows that the probability in 100 experiment are 1,5,18,28,26,15,5 and 2 which is very close to the estimated probability, and the expected value is 3.48 as well as the variance is 1.87 both of which accord with the theoretical value. In conclusion, the model is binomial distribution with parameter p which equals to 0.5 and n which equals to 7.

```
In [6]: # 3. Write a routine to count the longest run of heads in 100 Bernoulli samples.
        # Generate a histogram for this random variable
        # eg  111110=5   0000011111110=7
        import math
        # Initiate
        p = 0.5
        size = 100
        problem_3 = MyBernoulli(p, size=size)
        out_list3 = []
        # Experiment and Statistics
        out_list = []
        n = 1000
        for i in range(0, n):
            out_list3, success = problem_3.mul_bernoulli()
            longest_run = 0
            tmp = 0
            for i in range(0, 100):
                if out_list3[i] == 1:
                    tmp += 1
                    longest_run = max(longest_run, tmp)
                else:
                    tmp = 0
            out_list.append(longest_run)
        # Plot
        p3, t3 = np.histogram(out_list, bins=16, range=(0, 16))
        ind3 = np.arange(16)
        width = 0.5
        p1 = plt.bar(ind3, p3, width)
        plt.ylabel('numbers')
        plt.title('the longest run of heads in 100 Bernoulli samples')
        for i in range(0, len(p3)):
            plt.text(i-0.2, p3[i]+0.5, str(p3[i]), fontsize=10)
        plt.show()
        #Calculate the probability
        n = 101
        x = 16
        Probability_table = np.zeros((n, x))
```

```
for i in range(1, n):
    Probability_table[i][0] = 1

for i in range(0, x):
    for j in range(i, x):
        Probability_table[i][j] = np.power(2, i)

for j in range(1, x):
    for i in range(j+1, n):
        count = j + 1
        for k in range(1, count+1):
            Probability_table[i][j] += Probability_table[i-k][j]

print("The Probability that the longest run of
      heads is 0 or less in 100 coin tosses is 0.0")
for j in range(1, 16):
    print("The Probability that the longest run of
          heads is %d or less in 100 coin tosses is" % j,
          round((Probability_table[100][j]
                 - Probability_table[100][j-1])/math.pow(2, 100), 4))
```




```
The Probability that the longest run of heads is 0 or less in 100 coin tosses is 0.0
The Probability that the longest run of heads is 1 or less in 100 coin tosses is 0.0
The Probability that the longest run of heads is 2 or less in 100 coin tosses is 0.0003
The Probability that the longest run of heads is 3 or less in 100 coin tosses is 0.027
```

```
The Probability that the longest run of heads is 4 or less in 100 coin tosses is 0.1626
The Probability that the longest run of heads is 5 or less in 100 coin tosses is 0.264
The Probability that the longest run of heads is 6 or less in 100 coin tosses is 0.2286
The Probability that the longest run of heads is 7 or less in 100 coin tosses is 0.1473
The Probability that the longest run of heads is 8 or less in 100 coin tosses is 0.0826
The Probability that the longest run of heads is 9 or less in 100 coin tosses is 0.0434
The Probability that the longest run of heads is 10 or less in 100 coin tosses is 0.0221
The Probability that the longest run of heads is 11 or less in 100 coin tosses is 0.0111
The Probability that the longest run of heads is 12 or less in 100 coin tosses is 0.0055
The Probability that the longest run of heads is 13 or less in 100 coin tosses is 0.0027
The Probability that the longest run of heads is 14 or less in 100 coin tosses is 0.0014
The Probability that the longest run of heads is 15 or less in 100 coin tosses is 0.0007
```

1.Theoretical Exploration or Analysis

Let A be the event that the longest run of heads in ten coin tosses has length 3, $B_3$ be the event the longest run of heads is less than or equal to 3 and $B_2$ be the event the longest run of heads is less than or equal to 2. Easily, We know that $A = B_3 \cap B_2^c$ and thus $P(A) = P(B_3) - P(B_2)$. In fact, The elements of B3 can be partitioned by how the sequences begin. To be specific,

$B_{30}$ = {The sequences starting with T where B occurs}
$B_{31}$ = {The sequences starting with HT where B occurs}
$B_{32}$ = {The sequences starting with HHT where B occurs}
$B_{33}$ = {The sequences starting with HHHT where B occurs}

They cannot begin with four or more heads otherwise the longest run of consecutive will be more than three. Also these four subsets are disjoint. Therefore, P($B_3$)=P($B_{30}$)+P($B_{31}$)+P($B_{32}$)+P($B_{33}$)

If we define the permutation of n coin tosses there are where the longest run of heads is x or less as $N_n(x)$, then P($B_3$)=$N_{10}(3)$=210 and P($N_{10}(3)$)=P($N_9(3)$)+P($N_8(3)$)+P($N_7(3)$)+P($N_6(3)$).But for n<3, all sequences will have the longest run of heads of length 3 or less.

$$N_n(3) = \begin{cases} 2^n, & \text{if } n \leq 3 \\ N_{n-1}(3)) + N_{n-2}(3) + N_{n-3}(3) + N_{n-4}(3), & \text{else} \end{cases} \tag{2}$$

Similarlywhen x=4

$$N_n(4) = \begin{cases} 2^n, & \text{if } n \leq 4 \\ N_{n-1}(4)) + N_{n-2}(4) + N_{n-3}(4) + N_{n-4}(4) + N_{n-5}(4), & \text{else} \end{cases} \tag{3}$$

Stacking the result and we can get table of $N_n(x)$

| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 1 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 3 | 1 | 5 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 4 | 1 | 8 | 13 | 15 | 16 | 16 | 16 | 16 | 16 | 16 |
| 5 | 1 | 13 | 24 | 29 | 31 | 32 | 32 | 32 | 32 | 32 |
| 6 | 1 | 21 | 44 | 56 | 61 | 63 | 64 | 64 | 64 | 64 |

| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 1 | 34 | 81 | 108 | 120 | 125 | 127 | 128 | 128 | 128 |
| 8 | 1 | 55 | 149 | 208 | 236 | 248 | 253 | 255 | 256 | 256 |
| 9 | 1 | 89 | 274 | 401 | 464 | 492 | 504 | 509 | 511 | 512 |
| 10 | 1 | 144 | 504 | 773 | 912 | 976 | 1004 | 1016 | 1021 | 1023 |

Therefore,

$$P(A) = P(B_3) - P(B_2) \tag{4}$$
$$= \frac{773}{1027} - \frac{504}{1024} \tag{5}$$
$$= 0.263 \tag{6}$$

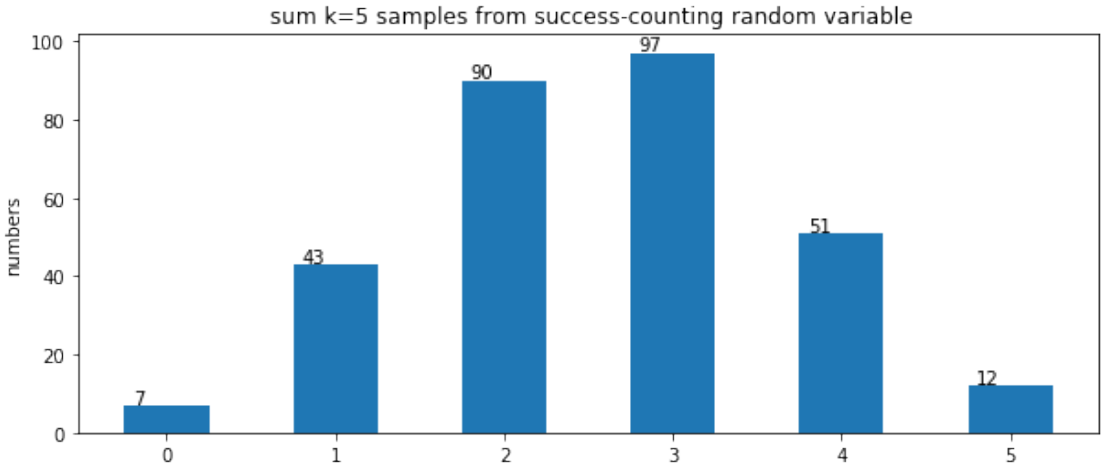
Similarly, We know that the probability that the longest run of heads is from 0 to 15 is 0, 0, 0.0003, 0.027, 0.1626, 0.264, 0.2286, 0.1473, 0.0826, 0.0434, 0.0221, 0.0111, 0.0055, 0.0027, 0.0014, 0.0007.

2.Experiments and Results The result of simulation shows that the probability in 1000 experiment are 0,0,0,19,163,252,233,144,111,36,25,9,3,1,2 and 2 which is very close to the estimated probability. So far we verify our theory successfully.

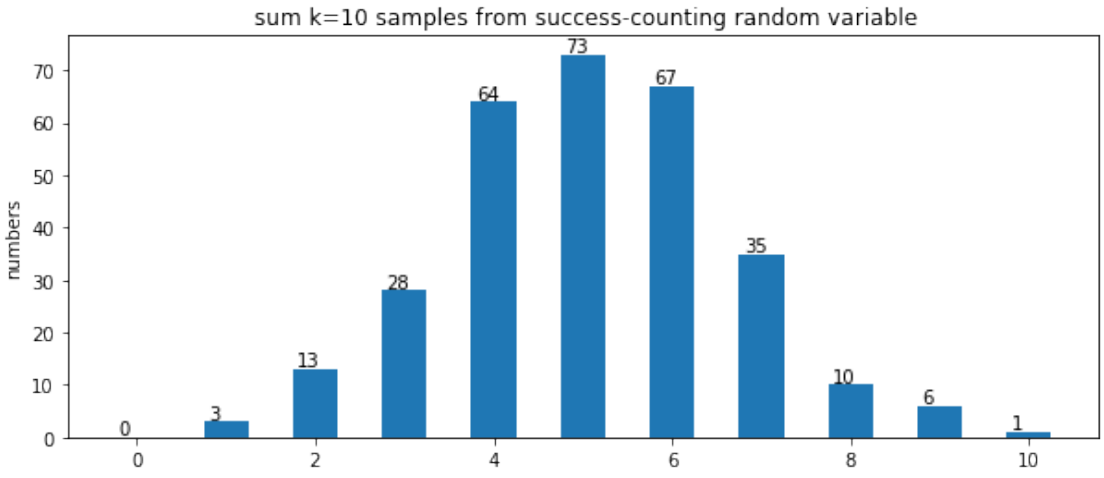
```
In [7]: # [Counting Successes]
        # Take your Bernoulli success-counting random variable
        # Generate and sum k=5 samples from this routine.
        # Generate 300 such sums and histogram your results.
        # Repeat for k={10, 30, 50}.
        # Comment on the histograms you observe for the different values of k.


        def count_success(size=5, p=0.5, n=300):
            # Initiate
            problem4 = MyBernoulli(p, size=size)
            out_list4 = []
            # Experiment and Statistics
            for i in range(0, n):
                tmp, success = problem4.mul_bernoulli()
                out_list4.append(success)
            p4, t = np.histogram(out_list4, bins=size+1, range=(0, size+1))
            # Calculate the expected and variance
            bernoulli_4_mean = cal_kth_moment(p4, 1)
            bernoulli_4_var = cal_var(p4)
            # plot
            ind4 = np.arange(size+1)
            width = 0.5
            plt.figure(figsize=(10, 4))
            plt.bar(ind4, p4, width)
            plt.ylabel('numbers')
```

```
    plt.title(' sum k=%d samples from success-counting random variable' % size)
    plt.yticks()
    for i in range(0, size+1):
        plt.text(i-0.2, p4[i]+0.5, str(p4[i]), fontsize=10)
    # Display the result
    plt.show()
    print("Expected value:", bernoulli_4_mean)
    print("Variance:", bernoulli_4_var)
```

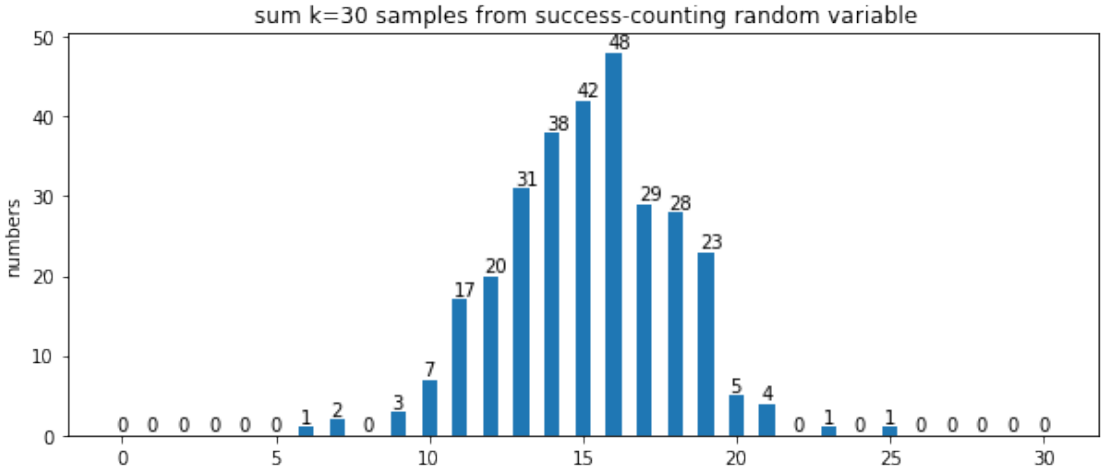
```
count_success(size=5)
count_success(size=10)
count_success(size=30)
count_success(size=50)
```



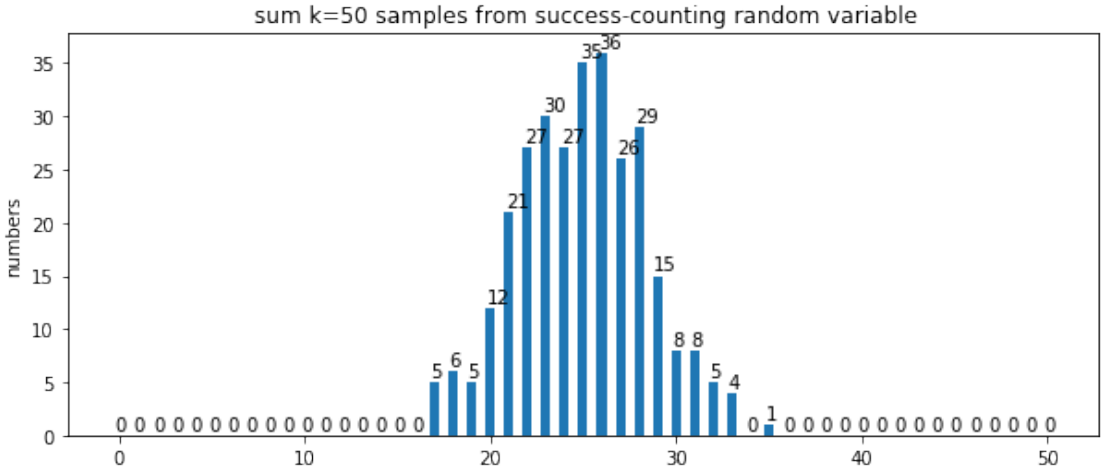

```
Expected value: 2.5933333333333337
Variance: 1.2479555555555528
```



```
Expected value: 5.083333333333333
Variance: 2.523055555555562
```




```
Expected value: 15.143333333333333
Variance: 7.722788888888914
```




```
Expected value: 24.920000000000005
Variance: 11.826933333333159
```

1.Theoretical Exploration or Analysis

The problem requires us to sum k=5 samples from the binomial random variable with parameter k and p which means k=5 and then generate 300 such sums and histogram results. To be specific, the sample space of a binomial random experiment is [0, k], and thus the sample space of k=5 times binomial random experiment is [0, 5]. Similarly, we can convert k=10,30,50 to count the number of successes in 70,210,350 fair Bernoulli trials respectively.

Thereforewe can easily get their expected value and variance from problem 2: When k equals to 5the expected value is 5$p$=2.5 *and the variance is* 5p1-$p$= *1.25. When k equals to 10the expected value is* 10p=5 and the variance is 10$p$1-p= 2.5. When k equals to 30the expected value is 30$p$=*15 and the variance is* 30p1-$p$= *7.5. When k equals to 50the expected value is* 50p=25 and the variance is 350$p$1-p= 12.5.

From the pmf of binomial distribution, there is a k value that maximizes it. This k value can be found by calculating

$$\frac{P(k+1,n,p)}{P(k,n,p)} = \frac{(n-k)p}{(k+1)(1-p)} \tag{7}$$

and comparing it to 1. There is always an integer M that satisfies

$$(n+1)p - 1 \leq M < (n+1)p \tag{8}$$

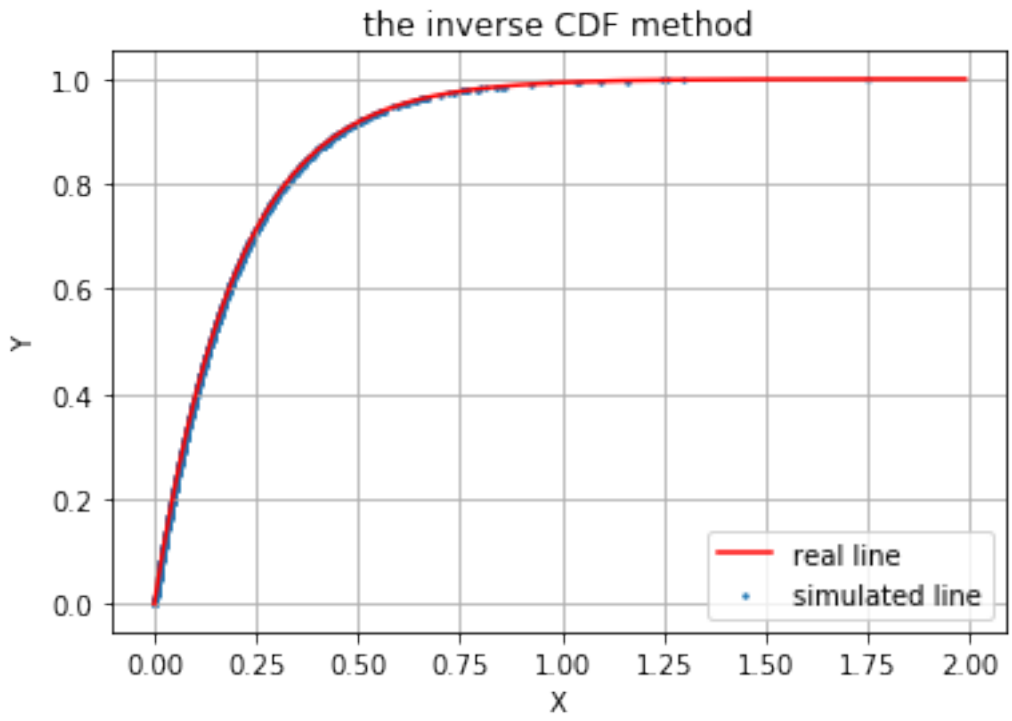
P(k, n, p) is monotone increasing for k < M and monotone decreasing for k > M, with the exception of the case where (n + 1)p is an integer. In this case, there are two values for which P is maximal: (n + 1)p and (n + 1)p  1. Therefore, values which occurs most frequently in experiemtn are 3, 5(6), 15(16), and 25(26) respectively.

2.Experiments and Results When k equals to 5the expected value is 2.6 and the variance is 1.28. When k equals to 10the expected value is 5.08 and the variance is 2.52. When k equals to 30the expected value is 15.14 and the variance is 7.72. When k equals to 50the expected value is 24.92 and the variance is 11.82. All of the result accord with the estimated value.

From the figures, all of figures look like pyramid which increases as x increases and then decrease as x increases. The x-coordinate of the each highest points of 4 pictures are 3, 5, 16 and 26. All the result accord with our expectation. Apparentlythe highest point in each figure decrease as the k increase and each figure become flatter with the increasement of k in the case we generate 300 such sums. And as k increases, the sample space becomes larger.

```
In [43]: # [Continuous Distributions]
         # Use the inverse CDF method to generate 1000 samples of the X~exp(5)
         # and Y~Cauchy(0,2). Generate histograms of the samples.
         # Suggests other methods for testing goodness-of-fit.

         def exp_cdf(x, lamda):
             if x >= 0:
                 return 1.0 - np.exp(-lamda*x)
             else:
                 return 0


         def inverse_exp_cdf(y, lamda):
             return -1.0/lamda * np.log(1.0 - y)
```

```
def new_cal_kth_moment(input_list, k, partition=1):
    kth_moment = 0
    length = len(input_list)
    size = np.sum(input_list)
    for i in range(0, length):
        kth_moment += (input_list[i] / size) * ((i/partition)**k)
    return kth_moment


def new_cal_var(input_list, partition=1):
    return new_cal_kth_moment(input_list, 2, partition) -
(new_cal_kth_moment(input_list, 1, partition))**2
# Generate exponential distribution via inverse CDF method
y = np.random.uniform(0, 1, size=1000)
x = inverse_exp_cdf(y, 5.0)
p, t = np.histogram(x, bins=2000, range=(0, 2))
# Generate exponential distribution via its CDF function
x3 = np.arange(0, 2, 0.01)
func = np.frompyfunc(exp_cdf, 2, 1)
y3 = func(x3, 5)
# plot
plt.plot(x3, y3, "red", label="real line")
plt.scatter(x, y, s=2, label="simulated line")
ax = plt.gca()
ax.set(xlabel='X', ylabel='Y',
       title='the inverse CDF method')
ax.grid()
plt.legend()
plt.show()
# Expected Value and Var Testing
print('Expected value of the sample:', new_cal_kth_moment(p, 1, 1000))
print('Variance of the Random Variance:', new_cal_var(p, 1000))
```




```
Expected value of the sample: 0.19449499999999997
Variance of the Random Variance: 0.03809673197500001
```

1.Theoretical Exploration or Analysis

If U is a uniform random variable on (0, 1) then $F^{-1}(U)$ has F as its CDF. Suppose we have a random variable U~Unif(0,1) and a cumulative distribution function

$$F_X(x) = 1 - e^{-\lambda x} \text{for} \qquad x \geq 0 \quad (and \quad 0 \quad otherwise). \tag{9}$$

By solving y=F(x) we obtain the inverse function

$$x = F^{-1}(y) = -\frac{1}{\lambda} \ln 1 - y \tag{10}$$

It means that if we draw some $y_0$ from a U~Unif(0,1) and compute $x_0 = F^{-1}(y_0) = -\frac{1}{\lambda} \ln 1 - y_0$, and finally $x_0$ has exponential distribution. Proof:

$$Pr(F^{-1}(U) \leq x) = Pr(U \leq F(x)) \tag{11}$$
$$= F(x) \tag{12}$$
$$\tag{13}$$

The mean or expected value of an exponentially distributed random variable X with parameter $\lambda$ is given by

$$E(X) = \frac{1}{\lambda} \tag{14}$$

Therefore, the estimated expected value is 0.2.And the variance of X is given by

$$Var(X) = \frac{1}{\lambda^2} \tag{15}$$

So, the estimated value is 0.04.

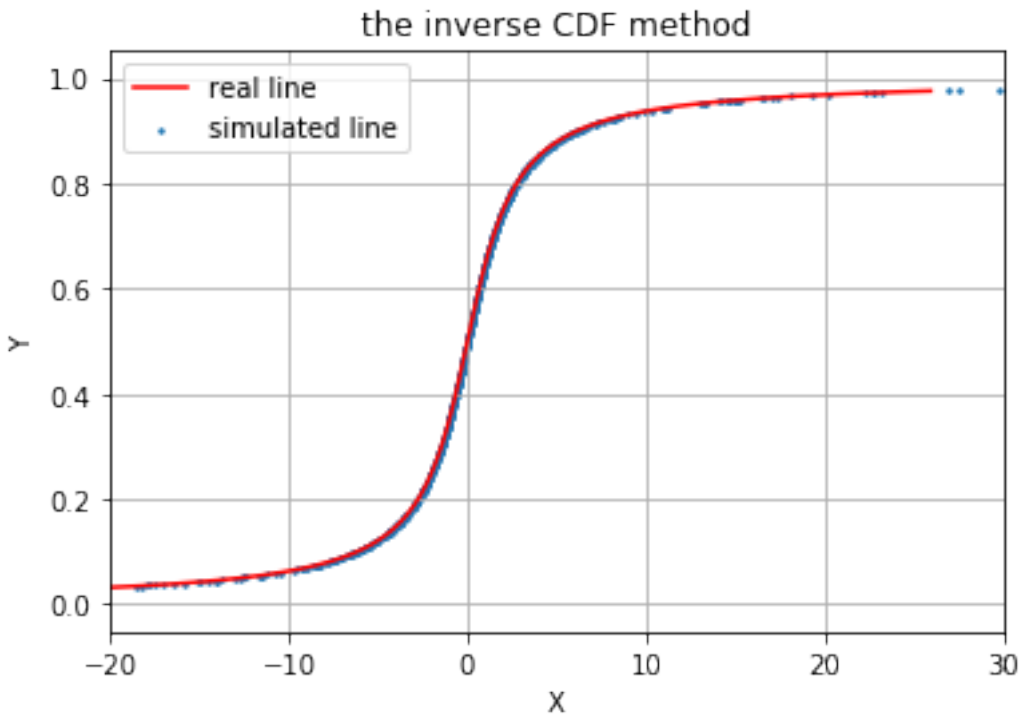
2.Experiments and Results

I use scatter diagram rather than histogram since scatter diagram is clear and more likely to fit a real line. And then I plot a theoretical line to make a contrast with the scatter diagram. Apparently two lines almost coincide, which means that random variable generated by inverse CDF method is definitely exponentially distributed.

In order to test goodness-of-fit, I partition the interval into 2000 small equally interval and count the number of points placed in each interval so that I can simulate the progress of integral and calculate the expected value and variance. In the experiment, the expected value is 0.195 and the variance is 0.038. The resultt correspond to our expectation.

```
In [7]: import numpy as np
        import matplotlib.pyplot as plt


        def cauchy_cdf(x, x0, y):
            return (1.0/np.pi) * np.arctan((x-x0)/y) + 1/2


        def inverse_cauchy_cdf(p, x0, y):
            return x0 + y * np.tan(np.pi*(p - 1/2))


        size = 1000
        y = np.random.uniform(0, 1, size=size)
        x1 = inverse_cauchy_cdf(y, 0, 2)

        x3 = np.arange(-25, 26, 0.1)
        func1 = np.frompyfunc(cauchy_cdf, 3, 1)
        y4 = func1(x3, 0, 2)  # cauchy
        # plot
        plt.plot(x3, y4, "red", label="real line")
        plt.scatter(x1, y, s=2, label="simulated line")
        ax = plt.gca()
        ax.set(xlabel='X', ylabel='Y',
               title='the inverse CDF method')
        ax.grid()
        plt.xlim(-20, 30)
        plt.legend()
        plt.show()
        # Chi-Squared Test
```

```
# Determine the number of intervals and Count elements
p, t = np.histogram(x1, bins=4, range=(-20, 20))
k = len(p)
total_number = np.sum(p)
# Calculate the expected Value in each interval
p_estimated = []
for i in range(-20, 20, 10):
    p_estimated.append(total_number * (cauchy_cdf(i+10, 0, 2) - cauchy_cdf(i, 0, 2)))


def chi_squared(f, f0, k):
    res = 0
    for i in range(0, k):
        res += np.power((f[i] - f0[i]), 2)/f0[i]
    return res


chi2 = chi_squared(p, p_estimated, k)

degree_of_freedom = k - 1
# Table of 2 values vs p-values when degree of freedom is 3
chi_squared_dict = {
    0.35: 0.95,
    0.58: 0.90,
    1.01: 0.80,
    1.42: 0.70,
    2.37: 0.50,
    3.66: 0.30,
    4.64: 0.20,
    6.25: 0.10,
    7.82: 0.05,
    11.34: 0.01,
    16.27: 0.001
}
# Look for the probability
tmp = list(chi_squared_dict.keys())
tmp.append(chi2)
tmp.sort()
ind = tmp.index(chi2)
probability_of_truth = 1 - chi_squared_dict[tmp[ind - 1]]
print('chi2', chi2)
print('There is a %f probability that the observed random variable is
      Cauchy random variable ' % probability_of_truth)
```




```
chi2 6.734108347317552
There is a 0.900000 probability that the observed random variable is Cauchy random variable
```

1.Theoretical Exploration or Analysis

Similarly, If U is a uniform random variable on (0, 1) then $F^{-1}(U)$ has F as its CDF. Suppose we have a random variable U~Unif(0,1) and a cumulative distribution function

$$F_X(x; x_0, \gamma) = \frac{1}{\pi} \arctan \frac{x - x_0}{\gamma} + \frac{1}{2} \tag{16}$$

By solving y=F(x) we obtain the inverse function

$$x = F^{-1}(p; x_0, \gamma) = x_0 + \gamma \tan \pi(p - \frac{1}{2}) \tag{17}$$

It means that if we draw some $y_0$ from a U~Unif(0,1) and compute $x = F^{-1}(p; x_0, \gamma) = x_0 + \gamma \tan \pi(p - \frac{1}{2})$, and finally $x$ has Cauchy distribution, but both its expected value and its variance are undefined. Therefore, we choose Chi-Squared Test method to test its goodness-of-fit. Suppose observed random variable is Cauchy random variable, and then we partition the interval into k small equally intervals and count the number of elements in each interval which notes observed value. Given the CDF of Cauchy distribution we could calculate the expected value in each intervals. The next step to calculate the test statistic

$$\sum_{i=1}^{k} \frac{(observed - expected)^2}{expected} \tag{18}$$

Under the null hypothesis, it has approximately a chi-squared distribution whose number of degrees of freedom are k-1. Finallywe look up the p-value in Table of $\chi^2$ values vs p-values according to the test statistic and number of degrees of freedom. The p-value is the probability of observing a test statistic at least as extreme in a chi-squared distribution and the probability that the observed random variable is Cauchy random variable equals to 1 - p-value.

2.Experiments and Results

Apparently two lines almost coincide, which means that random variable generated by inverse CDF method is Cauchy distributed. Let k=4and then number of degrees of freedom are 3. From the result that the test statistic is 6.734 we know that There is a 90% probability that the observed random variable is Cauchy random variable