# Monte Carlo Methods

## Some example applications in C++

# Introduction

**http://en.wikipedia.org/wiki/Monte_Carlo_method**

"Monte Carlo methods (or Monte Carlo experiments) are a class of computational algorithms that rely on *repeated random sampling* to compute their results.

Monte Carlo methods are often used in simulating physical and mathematical systems."

To illustrate the implementation of this kind of algorithm in C++ we will look at just a few basic applications where the MC method can help us solve problems in:

Maxima, Minima and Optimization

Probability and Counting Experiments

# The `rand()` function

Key to the Monte Carlo method is the generation of sequences of random numbers.

C++ has a built-in function for this:

| | |
|---|---|
| `rand()` | returns a number randomly selected in the range `0` to `RAND_MAX` |

Note that the sequence of number is not actually random, an algorithm is used to generate the numbers in a *chaotic* manner with a *large period*(the number of values returned before the sequence repeats).

Related function:

| | |
|---|---|
| `srand(n)` | sets the seed of the random number generator to `n` (to allow us to obtain different sequences of random numbers). |

# Generating and processing `rand()` values

Monte Carlo methods use large sets of random numbers, so we generally place the `rand()` function inside a large loop.

For example:

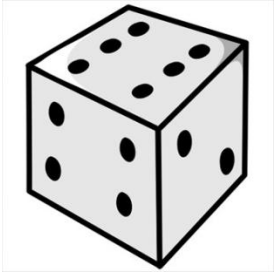
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main() {

  int n = 100000;

  cout << RAND_MAX << endl;

  for (int i=0; i<n; i++) {

    int k = rand();
    cout << k << endl;

  }
}
```

Note that the `cstdlib` header is required.

```
2147483647
1804289383
846930886
1681692777
1714636915
1957747793
424238335
719885386
1649760492
.
.
```

# Obtaining a discrete uniform distribution

For example if we want to simulate the throw of a die having six discrete random outcomes, all with equal probability:

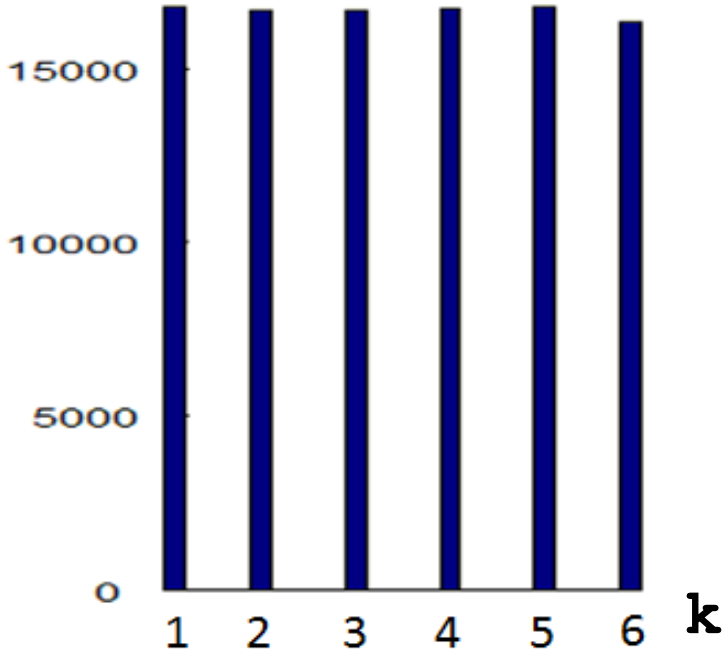


```
int k = 1 + rand() % 6;
```

Modulo **6** returns
`0 ≤ k ≤ 5`

+1 gives `1 ≤ k ≤ 6`

Store in integer `k`

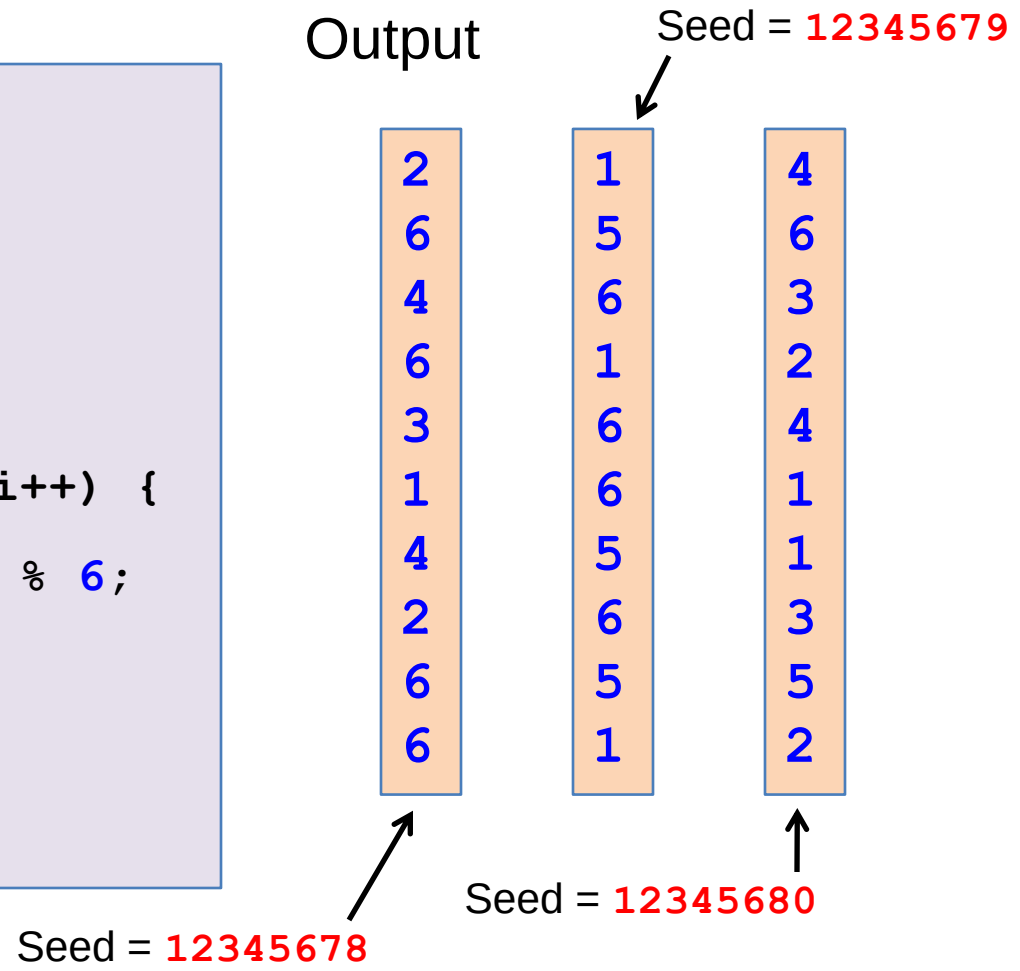



Histogram

# C++ code

Roll ten dice:

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {

  srand(12345678);

  for (int i=0; i<10; i++) {

    int d = 1 + rand() % 6;
    cout << d << endl;

  }

}
```



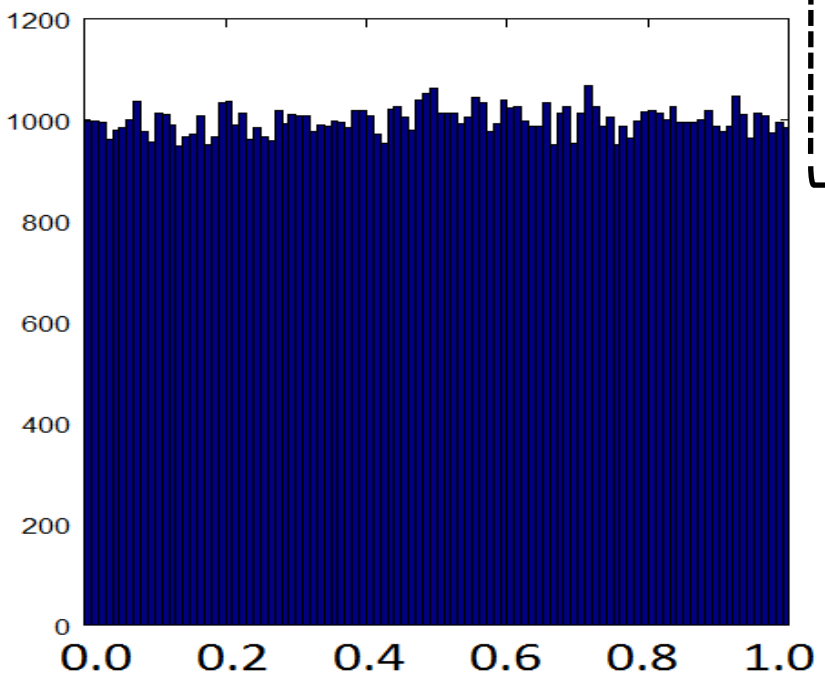
# Obtaining a continuous uniform distribution

Often we require floating-point random values:

```
double r = rand()/(1.0+RAND_MAX);
```

Cast to a double value slightly larger than `RAND_MAX`

Returns `0.0` ≤ `r` < `1.0`




Histogram of `r`

Example sequence:

| | |
|---|---|
| 0.24308 | 0.62229 |
| 0.77675 | 0.46182 |
| 0.24792 | 0.95476 |
| 0.74463 | 0.52671 |

`r` = `1.0` would be an *overflow* in the histogram

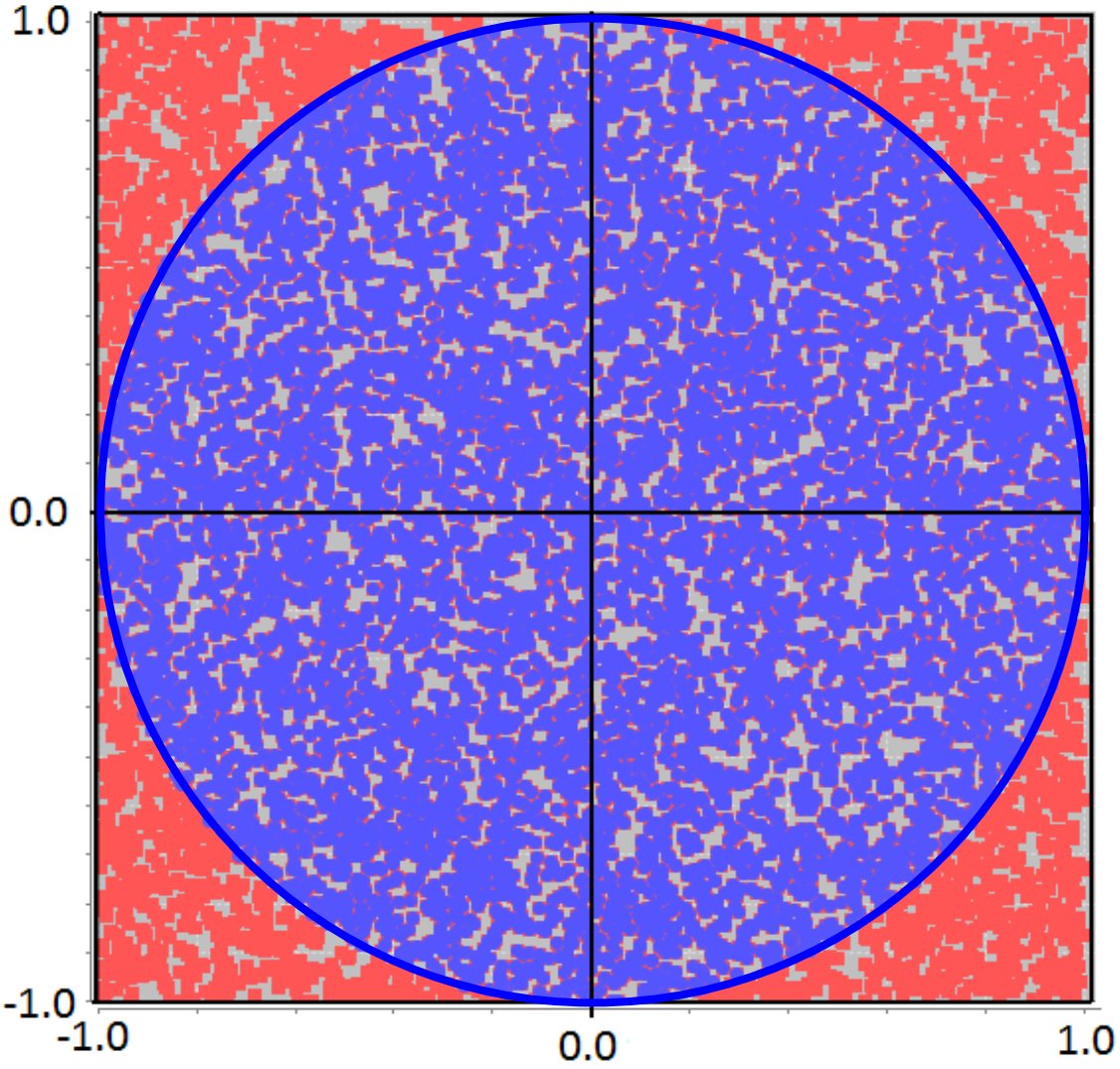
# Example

Populate a square with *n* **randomly-place points** [the blue and red points]

Count the number of points *m* that lie inside the circle [the blue points]

The ratio *4m/n* is then an approximation to the area of the circle (the area of the square being 4 units$^2$) and therefore an approximation to $\pi$.

$-1.0 \le x < +1.0$
$-1.0 \le y < +1.0$



# C++ code

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {

  int n = 100000;
  int m = 0;

  for (int i=0; i<n; i++) {

    double x = 2.0*rand()/(RAND_MAX+1.0) -1.0;
    double y = 2.0*rand()/(RAND_MAX+1.0) -1.0;
    if ( x*x+y*y < 1.0 ) m++;

  }

  cout << 4.0*m/n << endl;

}
```

$-1.0 \le x < +1.0$
$-1.0 \le y < +1.0$

Output for
$n = 10^5$

```
3.14376
```

Convergence

| n | |
|---|---|
| $10^2$ | 2.84 |
| $10^4$ | 3.1288 |
| $10^6$ | 3.14307 |
| $10^8$ | 3.14162 |
| $\pi$ | 3.14159 |

A factor of 100 increase in **n** yields a factor of 10 improvement in accuracy

This method works! but requires very large statistics to obtain good accuracy.

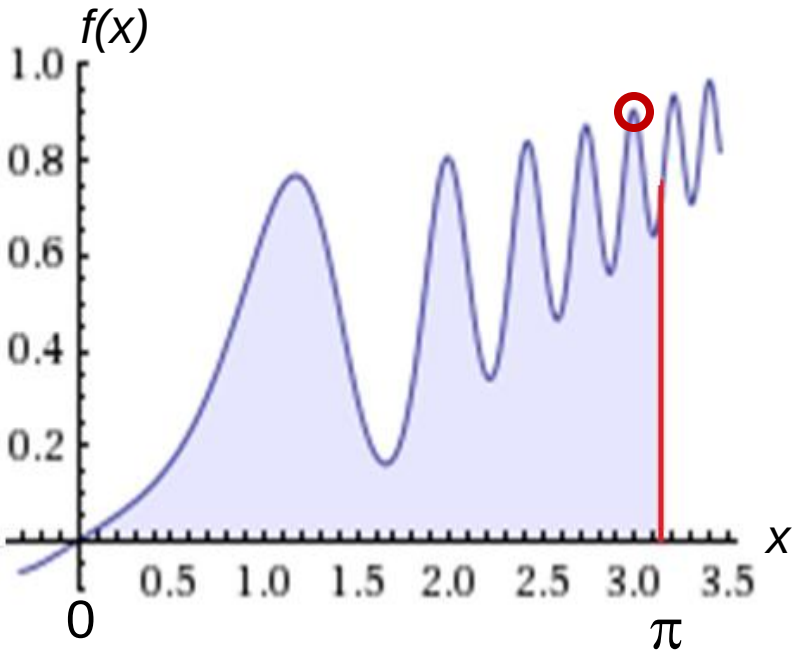
# Maxima/Minima

## Aim

We wish find the maximum or minimum of a function *f(x)*.

For example, find the maximum of the following function in the range $0 \le x < \pi$

$$f(x) = x\left(0.5 + e^{-x}\sin\left(x^3\right)\right)^2$$




How can we do this with random numbers?

## Solution

Generate a large number of random values *x* in the range $0 \le x < \pi$, evaluate *f(x)* at each point, record the position of the largest value.

# C++ code

$0 \le x < \pi$

```
#include <iostream>
#include <cmath>
#include <cstdlib>
using namespace std;

int main() {

  int n = 100000;
  double xmax = 0., fmax = 0.;

  for (int i=0; i<n; i++) {
    double x = M_PI * rand()/(RAND_MAX+1.0);
    double f = x*pow((0.5+exp(-x)*sin(x*x*x)),2);
    if ( f > fmax ) {
      fmax = f;
      xmax = x;
    }
  }

  cout << "f=" << fmax << endl;
  cout << "x=" << xmax << endl;

}
```

Output for $n = 10^5$

```
f=0.90536
x=2.99013
```

time = 12 *ms*

Convergence

| n | |
|---|---|
| $10^1$ | f=0.837919 |
| $10^2$ | f=0.905246 |
| $10^3$ | f=0.904914 |
| $10^4$ | f=0.905358 |
| $10^5$ | f=0.905360 |

http://www.wolframalpha.com/

$$\max\left\{x\left(0.5+\frac{\sin(x^3)}{e^x}\right)^2 \;\middle|\; 0 \le x \le \pi\right\} \approx 0.90536 \text{ at } x \approx 2.99013$$

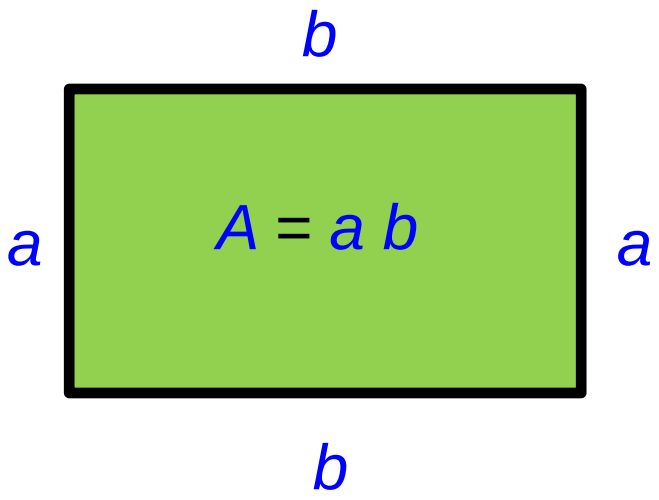
# Optimization

## Aim

Similar to the idea of obtaining the maximum or minimum of a function, we sometimes wish to optimize a system; i.e. maximise or minimize a target quantity.

## Example

We have 50 meters of fencing and wish to construct a fenced rectangular area, sides $a$ and $b$ with the target of maximizing the area $A = a\,b$ enclosed by the fence.




$2a + 2b = 50 \Rightarrow b = 25 - a$ *with* $0 < a < 25$

# C++ code

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {

  int n = 1e5;
  double max_area = 0., max_a = 0.;

  for (int i=0; i<n; i++) {

    double a = 25.0 * rand()/(RAND_MAX+1.0);
    double b = 25.0 - a;
    double area = a*b;

    if ( area > max_area ) {
      max_area = area;
      max_a = a;
    }

  }

  cout << max_a << endl;
}
```

*b* = 25 – *a*
*with* 0 < *a* < 25

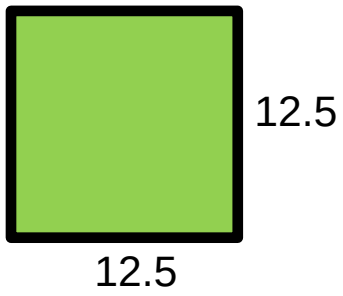
Output for
***n*** = $10^5$

```
12.4998
```

time = 1 ms

Convergence

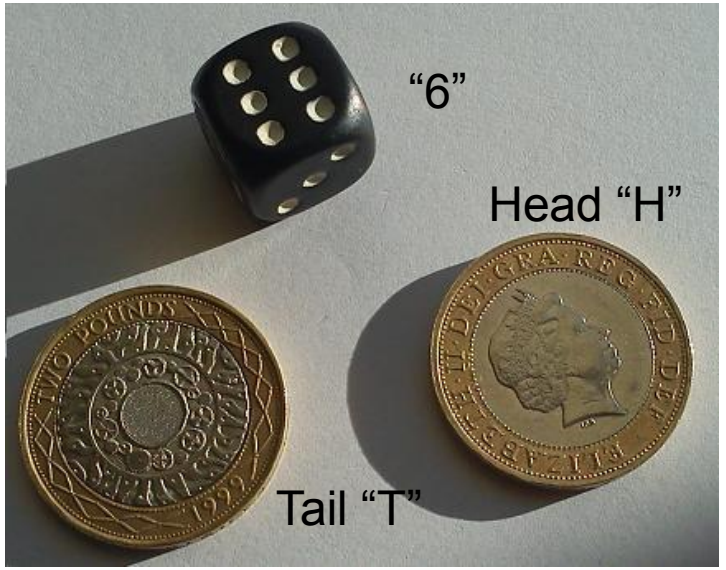
| **n** | |
|---|---|
| $10^1$ | **1**3.8492 |
| $10^2$ | **12.**3396 |
| $10^3$ | **12.4**631 |
| $10^4$ | **12.500**5 |
| $10^5$ | **12.499**8 |
| $10^6$ | **12.5000** |



# Probability and Counting Experiments

## Aim

Many physical systems are governed by random processes.
Common examples are the tossing of a coin and the throw of dice.
Monte Carlo methods allow us to simulate such systems and calculate outcomes in terms of probabilities.




For example, two coins and one die are thrown. What is the probability of obtaining a "Tail", "Head" and a "6" in any order: *P("T","H","6")* ?

Solution; throw *n* times and count the number of times *m* that we get the outcome.

Then $m / n \to P(T,H,6)$ as $n \to \infty$ ← *Experimental definition of probability*

# C++ code

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {

  int n = 1e5;
  int m = 0;
  for (int i=0; i<n; i++) {
    int   die = rand() % 6 + 1;
    int coin1 = rand() % 2;
    int coin2 = rand() % 2;
    if ( die==6 && (coin1 != coin2) ) m++;
  }

  cout << "1/" << 1/(double(m)/n) << endl;

}
```

"T" "H" or "H" "T"

Output for
$n = 10^5$

```
1/11.925
```

Convergence

| n | |
|---|---|
| $10^2$ | 1/7.6923 |
| $10^3$ | 1/10.870 |
| $10^4$ | 1/11.933 |
| $10^5$ | 1/11.925 |
| $10^6$ | 1/11.997 |
| $10^7$ | 1/12.017 |
| $10^8$ | 1/11.996 |
| $10^9$ | 1/12.001 |

This method works! but requires very large statistics to obtain good accuracy.

# Example

Bacteria are grown in culture dishes in a laboratory.
Experience tells us that on average in this lab 20% of the dishes become contaminated by unwanted bacteria (thus spoiling the culture).

Question:
If the lab is growing bacteria in ten dishes, what is the probability that more then half of the dishes will become contaminated?

Solution:
We have ten dishes, *P("contamination of each dish") = 0.2*

Use a Monte Carlo experiment to test each dish against the probability of 0.2. Repeat this *n* times and count the number of times *m* where more than 5 dishes become contaminated.

Then $n / m \to P(\text{"more than 5 dishes are contaminated"})$ as $n \to \infty$

*Experimental definition of probability*

# C++ code

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {

  int n = 1e8;
  int m = 0;
  for (int i=0; i<n; i++) {
    int k=0;
    for (int j=0; j<10; j++) { // 10 dishes
      double r = rand()/(RAND_MAX+1.0);
      if ( r < 0.2 ) k++; // contaminated
    }
    if (k>5) m++; // > 5 dished spoiled
  }

  cout << m/double(n) << endl;

}
```

*Ten dishes*
*P("contamination") = 0.2*
*P("> 5 dishes contaminated")?*

Output for
$n = 10^8$

```
0.006365
```

time = 12 s

Convergence

| n | |
|---|---|
| $10^3$ | 0.01 |
| $10^4$ | 0.0057 |
| $10^5$ | 0.00672 |
| $10^6$ | 0.006291 |
| $10^7$ | 0.006374 |
| $10^8$ | 0.006365 |
| $10^9$ | 0.006368 |

true = 0.637%

*Binomial distribution.*