Random number generators and random processes

Eugeniy E. Mikhailov

The College of William & Mary

Lecture 11

Statistics and probability intro

Mathematical and physical definition of probability (p) of event 'x'

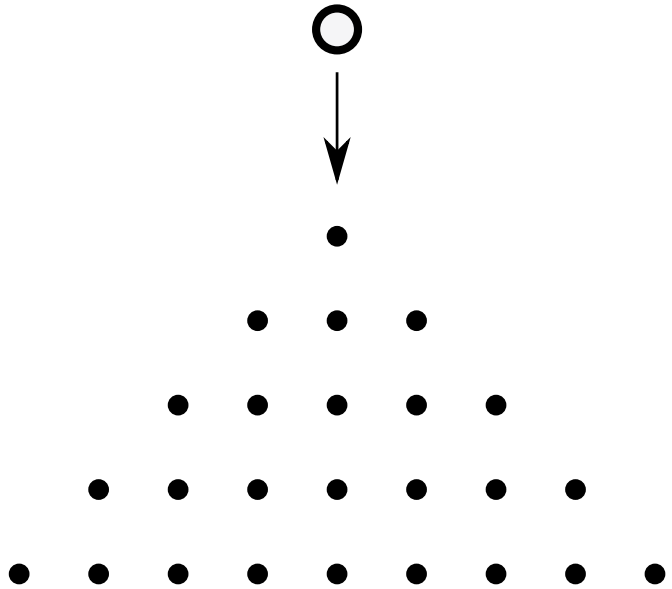
$$p_x = \lim_{N_{total} \to \infty} \frac{N_x}{N_{total}}$$

where

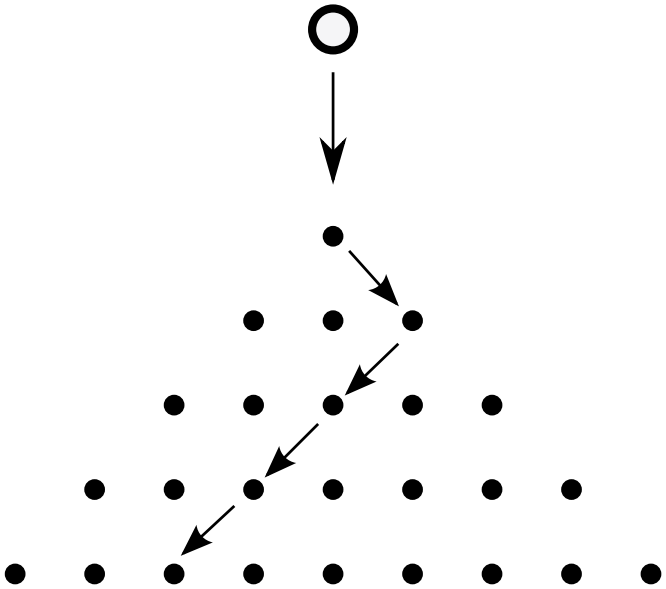
N_x the number of registered event 'x'

N_{total} the total number of all events

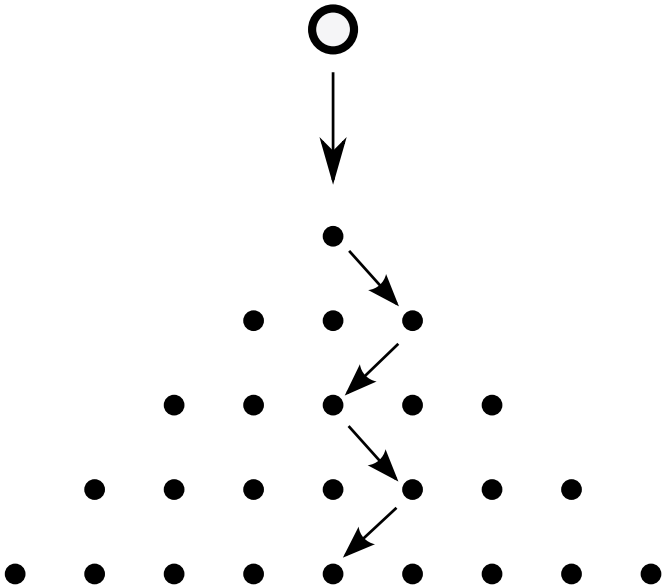
Peg board example



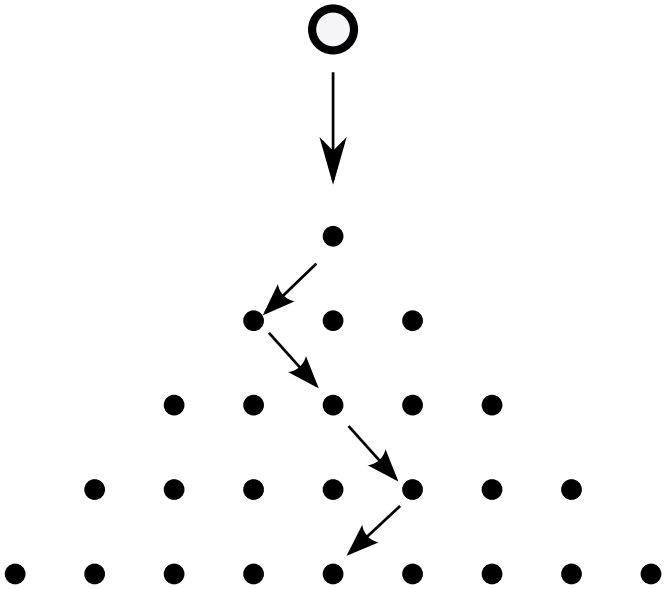
Peg board example



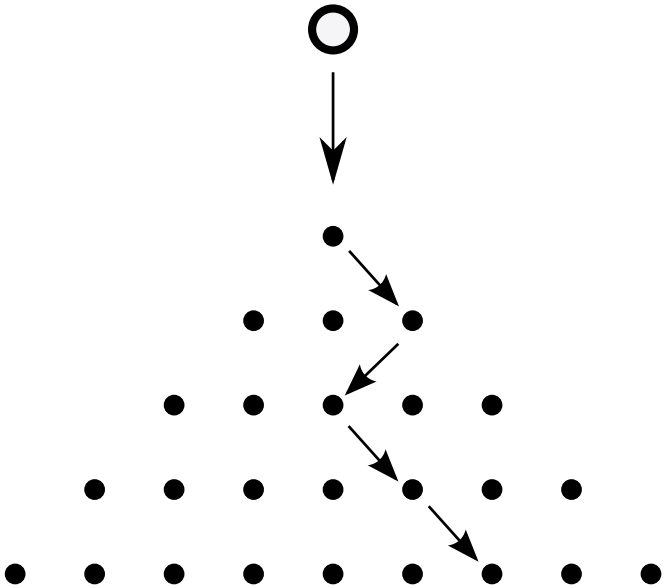
Peg board example



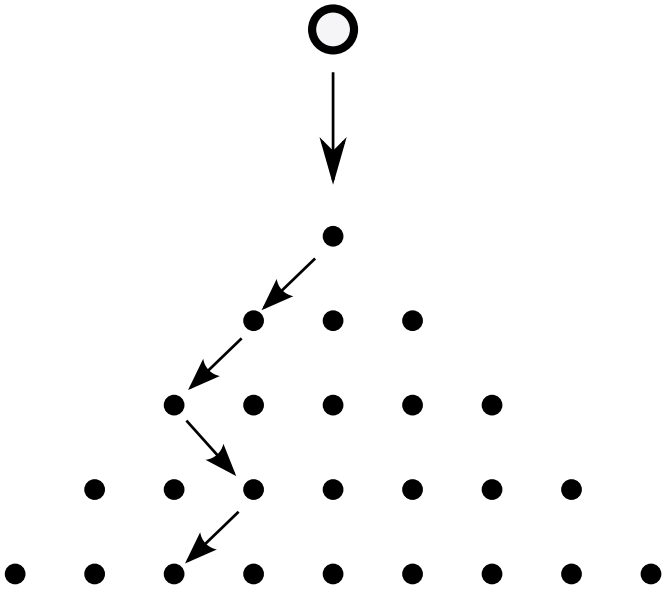
Peg board example



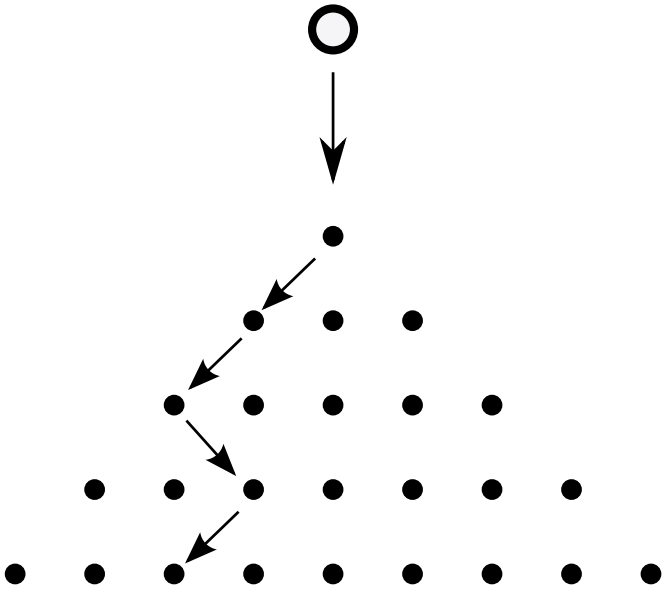
Peg board example



Peg board example

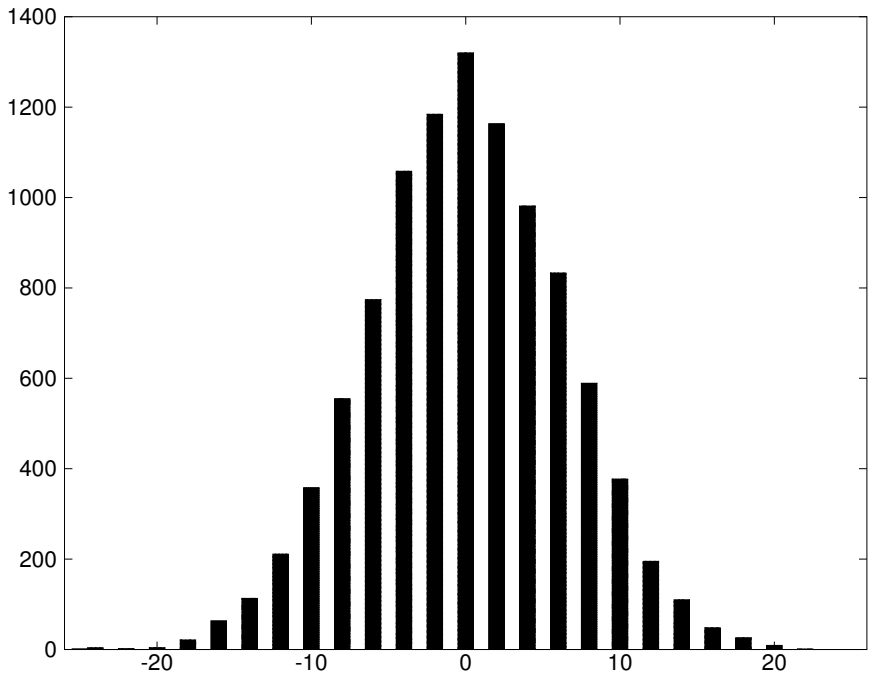


Peg board example



Example of 10^4 balls runs over 40 layers of nails

```
x=pegboard(10^4, 40);
hist(x, [-25:25]);
```



Resulting distribution is Gaussian

Probability for occurrence of the real number

It is well known that interval [0..1] has infinite amount of reals numbers (as well as any other non zero interval).
So there is very little or may be zero chance that event will repeat or even happen. Since we cannot run ∞ number of tests.
In this case we should speak about probability density $p(x)$.
The best way to estimate it is from a histogram.
Run your N tests with numbers distributed between 0 and 1. Split your interval at m bins and calculate number of events when you hit each bin $h(x_b)$. Plot this vs bin positions.

Easy to do with Matlab

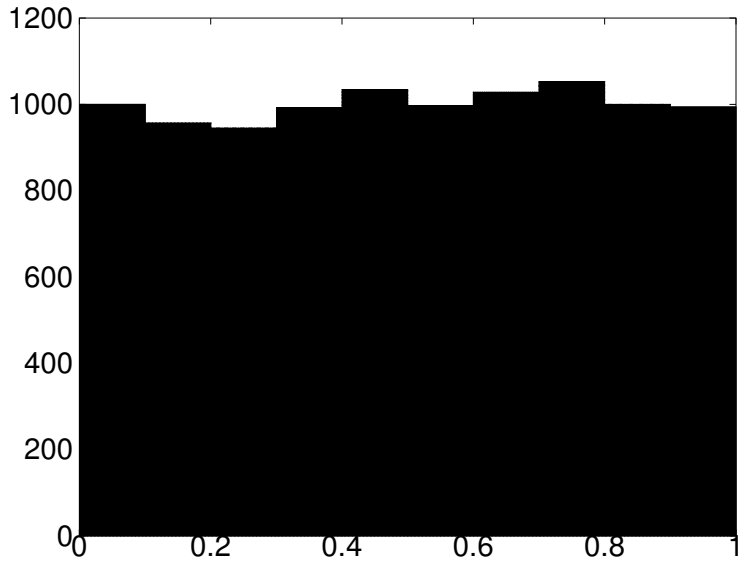
$r =$ `rand`$(1, N)$; `hist`(r, m);

Then

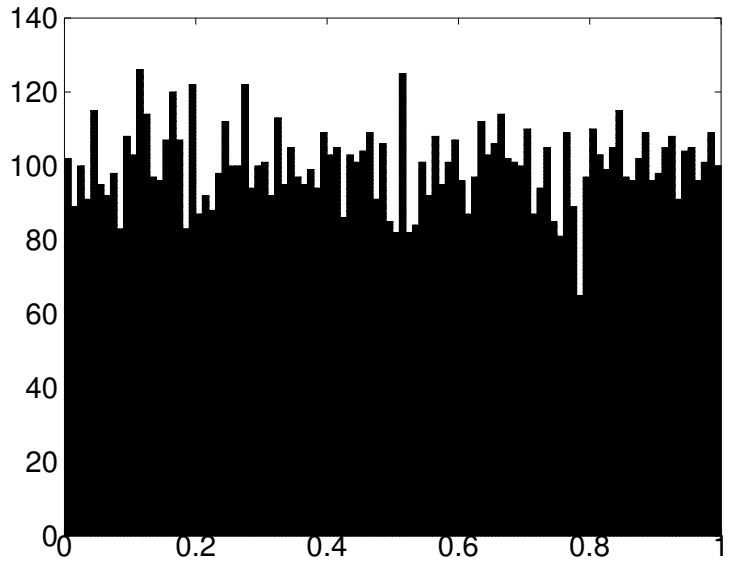
$$p(x) = \lim_{N,m\to\infty} \frac{h(x_b \text{ nearest to } x)}{N}$$

Uniform random distribution

```
r = rand(1,10000);
hist(r,10);
```




```
r = rand(1,10000);
hist(r,100);
```



Random number generators

How can a computer, which is very accurate, precise, and deterministic, generate random numbers?

It cannot!

Instead we can generate a sequence of pseudo random numbers. By pseudo we mean that starting from the same initial conditions the computer will generate the same sequence of numbers (*very handy for debugging*). But otherwise it will look like random numbers and will have statistical properties of random numbers.

Linear Congruential Generator (LCG)

Recursive formula

$$r_{i+1} = (ar_i + c) \bmod m$$

here

- m the modulus
- a multiplier, $0 < a < m$
- c increment, $c \leq c < m$
- r_1 seed value, $0 \leq r_1 < m$

All pseudo random generators have a period and this one is no exception. Note that once r_i repeat one of the previous values the sequence will restart.
This one can have at most a period of m distinct numbers (0..m).

Linear Congruential Generator (LCG) continued

Bad choice of a, c, m will lead to even shorter periods.

Example

$m = 10$, $a = 2$, $c = 1$,$r_1 = 1$
$r = [1, 3, 7, 5, 1]$

While the LCG has advantage of speed and simplicity.

Do not use the LCG whenever your money or reputation are at stake!

While Matlab does not use LCG, many other programming languages use it as default in their libraries so be aware of it.

Random number generators period

Even the best pseudo random generators cannot have a period larger than 2^B, where B is number of memory bits. Can you prove it?

While the period can be huge its not infinite. For example for Matlab R2007a the period is $2^{19937} - 1$.

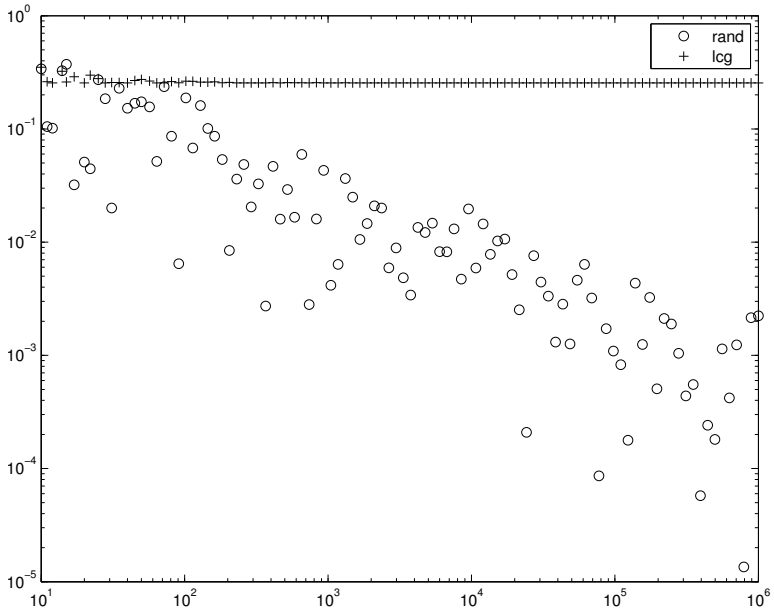
Why bother?
Recall that for example error of the Monte Carlo integration method is $\sim 1/\sqrt{N}$.
This holds true only when $N <$ than the period of random number generator (T).
Otherwise the MC method cannot give uncertainty better than $\sim 1/\sqrt{T}$. Further increase of the number of random points will not bring any extra improvement.

How to check the random generator

Generally it is uneasy and probably impossible.
However for us only statistical properties are of importance.
So the easiest is to check that integral deviation calculated with Monte Carlo algorithm drops as $1/\sqrt{N}$.

Simple LCG check

Let's see how the LCG will look with very bad coefficient
$m = 10$, $a = 2$, $c = 1$,$r_1 = 1$



Lines are from MC with `rand` and line-crosses for `lcgrand`(Nrows, Ncols, 2,1,10);

```
function r= ...
lcgrand(Nrows,Ncols, ...
a,c,m, seed)

r=zeros(Nrows, Ncols);
r(1)=seed;
cntr=1;
for i=2:Nrows*Ncols;
r(i)= mod( (a*r(i-1)+c), m);
end
r=r/(m-1); %normalization
end
```

```
check_lcgrand
```