

## Random Max

Musicians and computer scientists have different ideas of what random means. The mathematician is concerned about the ultimate results of random operations, whereas a composer is interested in being somewhat surprised.

Although random processes abound in nature, they are hard to achieve in the computer. To make random seeming numbers in the computer, we use iterative algorithms that produce wildly jumping results. This is one example:

$$X_n = \text{abs}((X_{n-1} * 16807) \% 2147483646)$$

This works pretty well for 32 bit numbers. There are better ones. Here's a sample run:

```
16807
67726603
114683181
101098253
145470523
12945407
181384771
33939379
167261577
105516427
192156013
34547455
174245287
197659971
128674975
90474545
```

These are pretty big numbers, so most random generators convert this to the fractional part of a float or use a modulus operation to reduce the value to a desired range.

Any random number function will always give the same set of numbers given a particular starting value. To make it surprising, we give it something unlikely as the starting value of  $x$ . (The starting value is called the seed.) The elapsed time since the computer was turned on is a popular choice. Hitting that the same twice is very unlikely.

### Random

The random object in Max is perfectly fine:

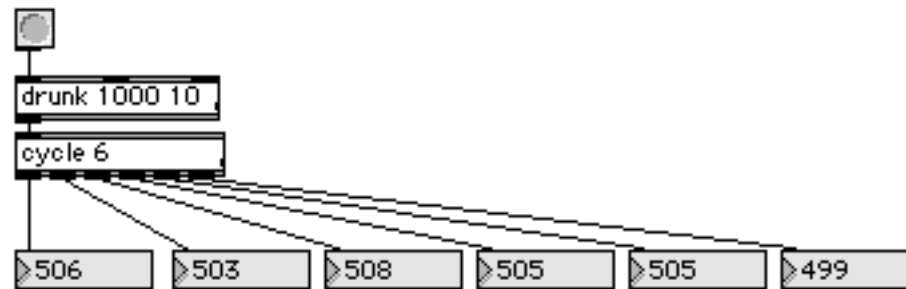


This produces numbers that bounce around with no discernable pattern. There is a tendency to favor odd numbers, however, so if you doing coin flips, there is a better object called **decide**.

Random is seeded from the computer host clock, so it's going to be surprising out of the box. If you want predictable random numbers, you can use the seed n message. Every time the seed is applied, the sequence will start over.

### Drunk

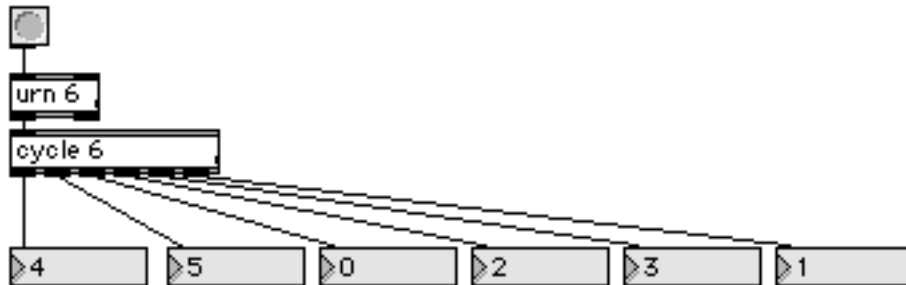
One popular variation on randomness is the Drunkard's Walk. This is produced by adding a random value to the last value computed. Thus the numbers will tend to stay close to the last value, but will ultimately wander over a wide range.



This can make for some melodic like patterns.

### Urn

Sometimes we want to get each number once, as in dealing cards. The urn (unique random number) object does this:



## Distributions

We describe random function generators on the basis of the distribution of the results they produce. That is, add up incidences of each number output and draw a graph. A perfect random number generator has a flat graph. Other shapes are appropriate in various musical circumstances.

### Gaussian

The Gaussian distribution is made by adding several<sup>1</sup> random numbers. It produces the familiar "bell curve" so beloved by statisticians.

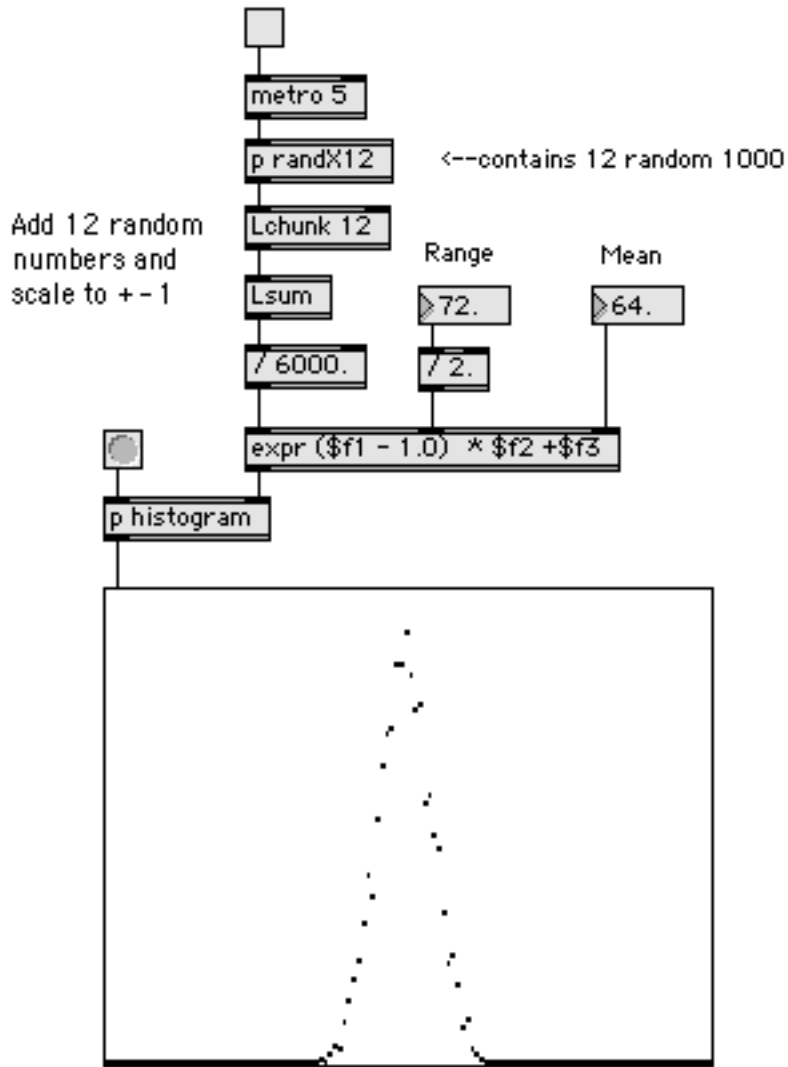


Gaussian number generators are useful for varying parameters in a natural way. Almost everything deviates from the ideal by a gaussian function, so using it on things like rhythm can de-mechanize the output of computer programs.

Here's how to generate a gaussian distribution in Max:

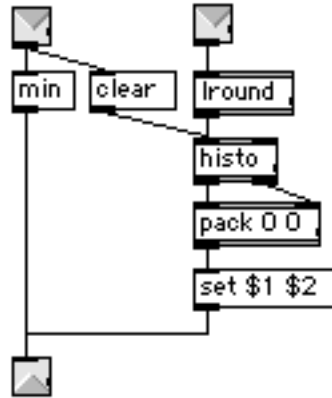
---

<sup>1</sup> Technically, an infinite number of random numbers, but 12 will usually do.



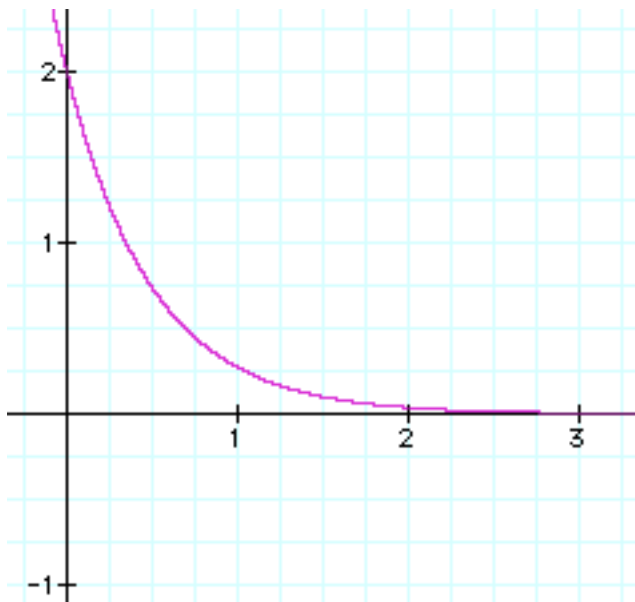
The range and mean controls modify the output, which would otherwise be between -1 and 1.

The histogram subpatcher contains a histo object and logic to display the contents in a multislider:



### Exponential

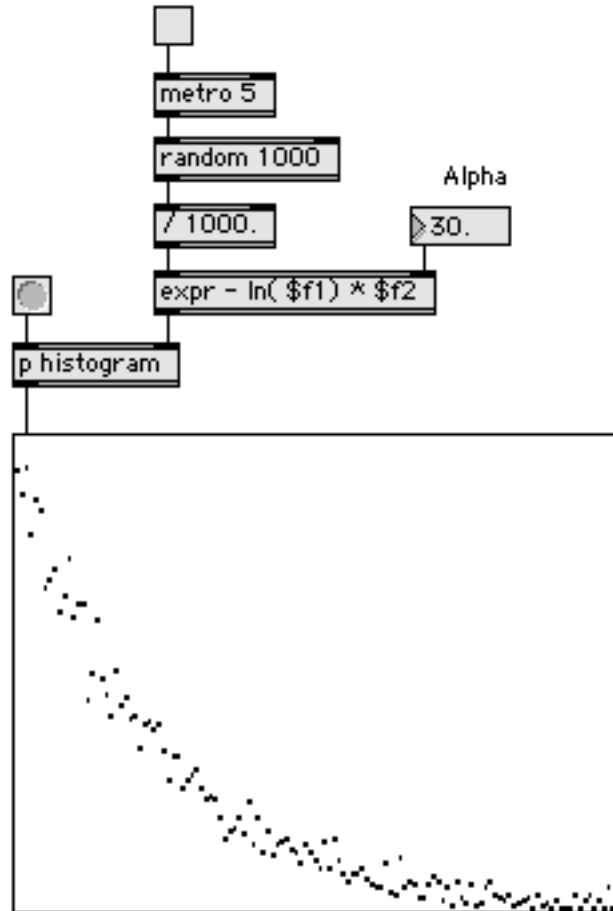
A simple exponential distribution is squeezed toward 0.



This can be generated by taking the negative<sup>2</sup> of the natural logarithm of the random numbers. This function is useful for things that should only vary occasionally, and not by much.

---

<sup>2</sup> Remember, the random numbers are between 0.0 and 1.0, so the logarithms will be negative.

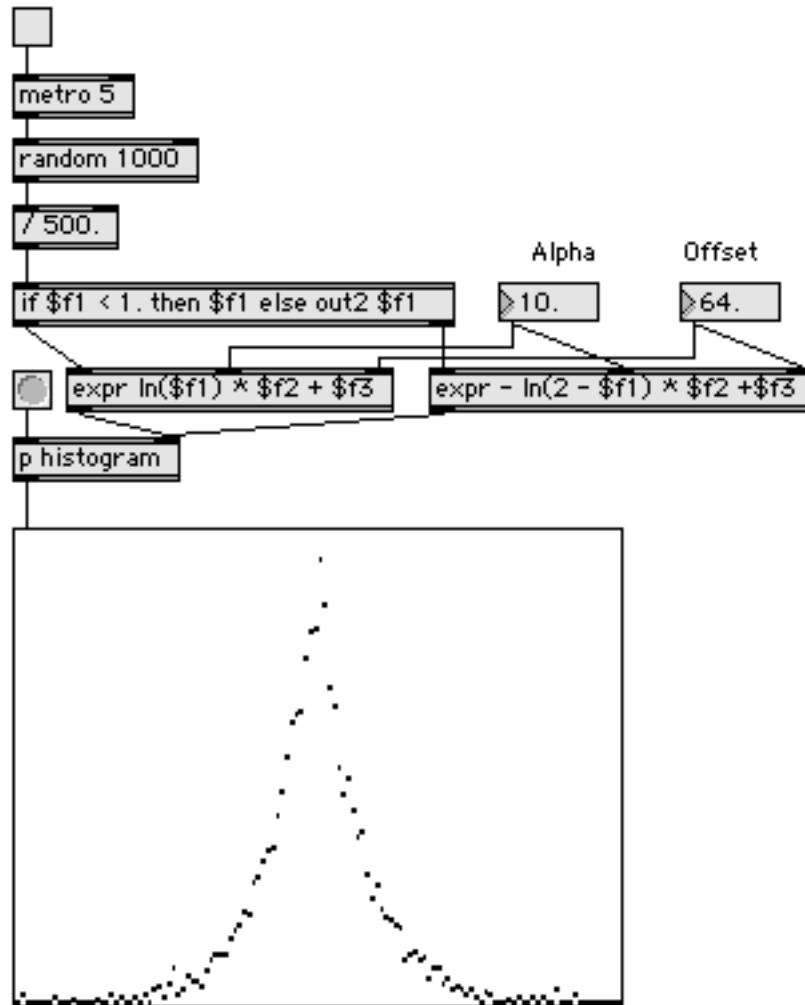


Alpha is a number greater than 0 that will control the width of the skirt of the function.. The mean of the numbers generated will be  $0.69 \cdot \alpha$ .

**Bilateral exponential**



To make an exponential function with a symmetrical shape, we break the random numbers into two ranges and find the natural log:



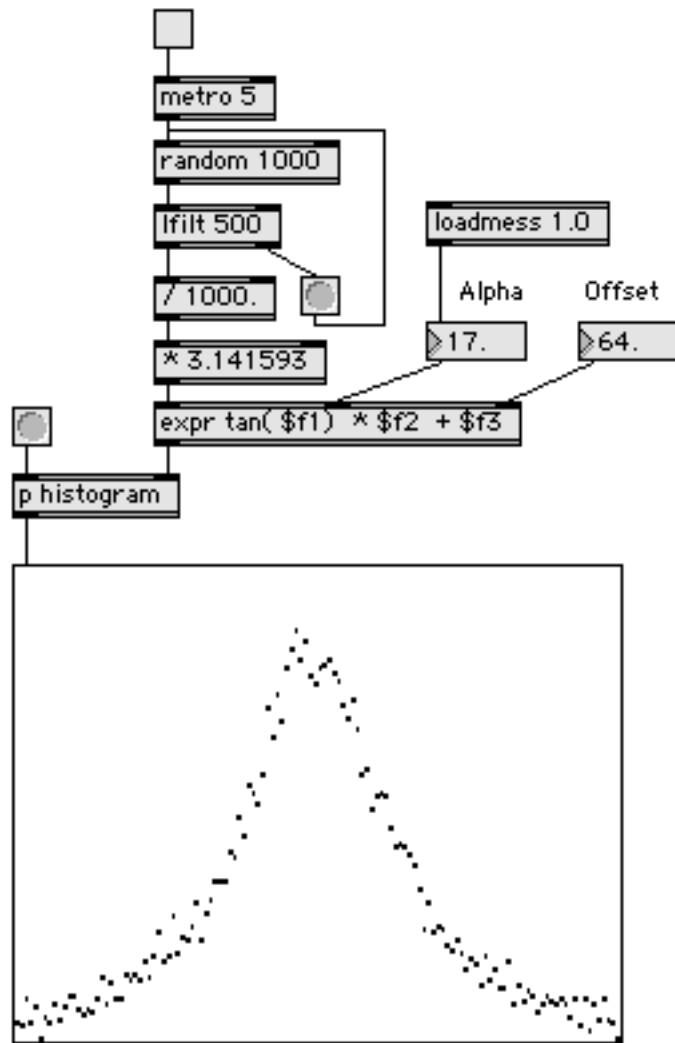
Alpha is a number greater than 0 that will control the width of the skirt of the function. Offset is added to shift the mean value. This may be positive or negative.

### Cauchy



The Cauchy distribution is similar to the Gaussian, but spread out a bit more.

To create this, we expand the random function to  $\pi$ , and take the tangent:



Alpha is a number greater than 0 that will control the width of the skirt of the function. This distribution is centered on 0. We filter 500 out of the random values because the tangent of  $\pi/2$  is not defined.

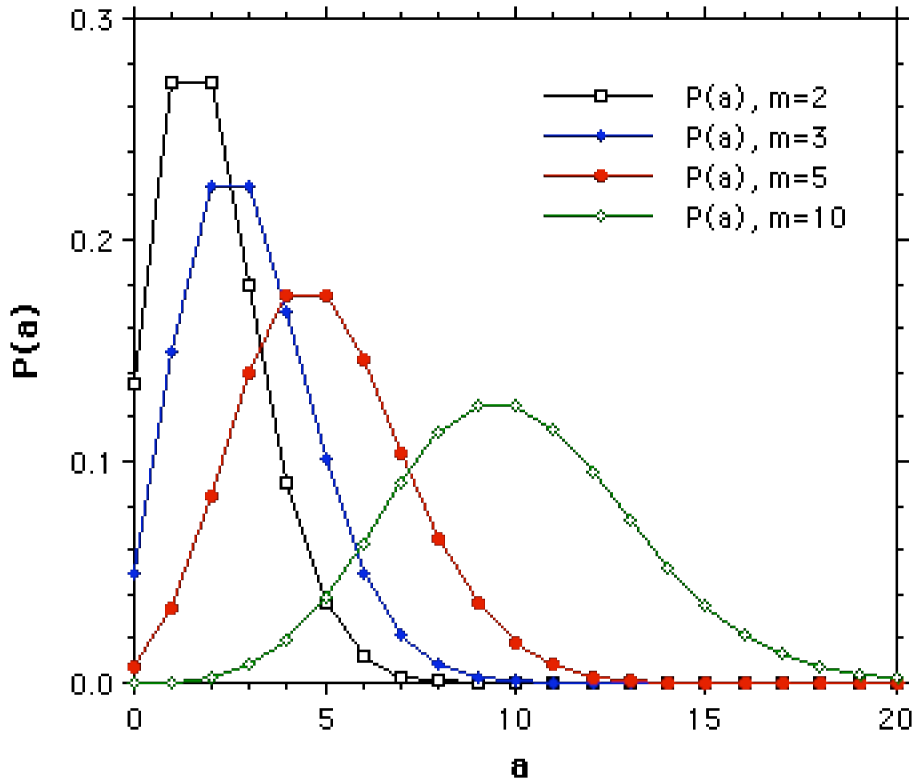


**Poisson**

A curve similar to the Gaussian, found in many natural phenomena, such as radioactive decay. It's the probable number of randomly timed events happening within a particular time. Say a process emits electrons at the average rate of 300/second. If you count electrons for 100 one second tests, you are mostly going to count 300, often 299 or 301, occasionally 250 or 350, rarely 20 or 580. The Poisson distribution describes this situation.

The probability **P** of **a** events happening during a period in which an average of **m** events usually occur is:

$$P(a) = e^{-m} \left[ \frac{m^a}{a!} \right]$$



You don't really need an algorithm to generate this, it's what you will get with ordinary random numbers applied to rhythm. The only difference between this and Gaussian distribution is that this is limited to discrete numbers, because events either happen or they don't.

## Arbitrary Distributions

For many musical purposes we want to be able to define arbitrary distributions. In other words, draw a curve and generate values that (over the long term) follow the curve. This is easy, because the code to do it is built into the table object. If you draw a curve directly into a table, and bang the table, the output will be addresses from the table with a probability determined by the value at each address. Ltotab is an Lobject that copies a list into a table. This weird curve drawn into a multislider controls the output nicely.

