



Cross Validated is a question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization. Join them; it only takes a minute:

Sign up


Here's how it works:



Anybody can ask a question



Anybody can answer



The best answers are voted up and rise to the top

Simulating draws from a Uniform Distribution using draws from a Normal Distribution

I recently purchased a [data science interview resource](#) in which one of the probability questions was as follows:

Given draws from a normal distribution with known parameters, how can you simulate draws from a uniform distribution?

My original thought process was that, for a discrete random variable, we could break the normal distribution into K unique subsections where each subsection has an equal area under the normal curve. Then we could determine which of K values the variable takes by recognizing which area of the normal curve the variable ends up falling into.


But this would only work for discrete random variables. I did some research into how we might do the same for continuous random variables, but unfortunately I could only find techniques like inverse transform sampling that would use as *input* a uniform random variable, and could *output* random variables from some other distribution. I was thinking that perhaps we could do this process in reverse to get uniform random variables?

I also thought about possibly using the Normal random variables as inputs into a linear congruential generator, but I'm not sure if this would work.

Any thoughts on how I might approach this question?

self-study normal-distribution simulation uniform

asked Oct 2 '14 at 22:44

 **wellington**  
157 3 10

2 Answers

In the spirit of using simple algebraic calculations *which are unrelated to computation of the Normal distribution*, I would lean towards the following. They are ordered as I thought of them (and therefore needed to get more and more creative), but I have saved the best--and most surprising--to last.

1. Reverse the **Box-Mueller technique**: from each pair of normals  $(X, Y)$ , two independent uniforms can be constructed as  $\text{atan2}(Y, X)$  (on the interval  $[-\pi, \pi]$ ) and  $\exp(-(X^2 + Y^2)/2)$  (on the interval  $[0, 1]$ ).

2. Take the normals in groups of two and sum their squares to obtain a sequence of  $\chi^2_2$  variates  $Y_1, Y_2, \dots, Y_i, \dots$ . The expressions **obtained from the pairs**

$$X_i = \frac{Y_{2i}}{Y_{2i-1} + Y_{2i}}$$

will have a  $\text{Beta}(1, 1)$  distribution, which is uniform.

That this requires only basic, simple arithmetic should be clear.

3. Because the **exact distribution of the Pearson correlation coefficient** of a four-pair sample from a standard bivariate Normal distribution is uniformly distributed on  $[-1, 1]$ , we may simply take the normals in groups of four pairs (that is, eight values in each set) and return the correlation coefficient of these pairs. (This involves simple arithmetic plus two square root operations.)

4. It has been known since ancient times that a cylindrical projection of the sphere (a surface in three-space) is **equal-area**. This implies that in the projection of a uniform distribution on the sphere, both the horizontal coordinate (corresponding to longitude) and the vertical coordinate (corresponding to latitude) will have uniform distributions. Because the trivariate standard Normal distribution is spherically symmetric, **its projection onto the sphere is uniform**. Obtaining the longitude is essentially the same calculation as the angle in the Box-Mueller method (*q.v.*), but the projected latitude is new. The projection onto the sphere merely normalizes a triple of coordinates  $(x, y, z)$  and at that point  $z$  is the projected latitude. Thus, take the Normal variates in groups of three,  $X_{3i-2}, X_{3i-1}, X_{3i}$ , and compute

https://stats.stackexchange.com/questions/117689/simulating-draws-from-a-uniform-distribution-using-draws-from-a-normal-distribut

1/5

$$\frac{X_{3i}}{\sqrt{X_{3i-2}^2 + X_{3i-1}^2 + X_{3i}^2}}$$

for  $i = 1, 2, 3, \dots$

- Because most computing systems represent numbers in **binary**, uniform number generation usually begins by producing uniformly distributed *integers* between 0 and  $2^{32} - 1$  (or some high power of 2 related to computer word length) and rescaling them as needed. Such integers are represented internally as strings of 32 binary digits. We can obtain independent random bits by comparing a Normal variable to its median. Thus, it suffices to break the Normal variables into groups of size equal to the desired number of bits, compare each one to its mean, and assemble the resulting sequences of true/false results into a binary number. Writing  $k$  for the number of bits and  $H$  for the sign (that is,  $H(x) = 1$  when  $x > 0$  and  $H(x) = 0$  otherwise) we can express the resulting normalized uniform value in  $[0, 1)$  with the formula

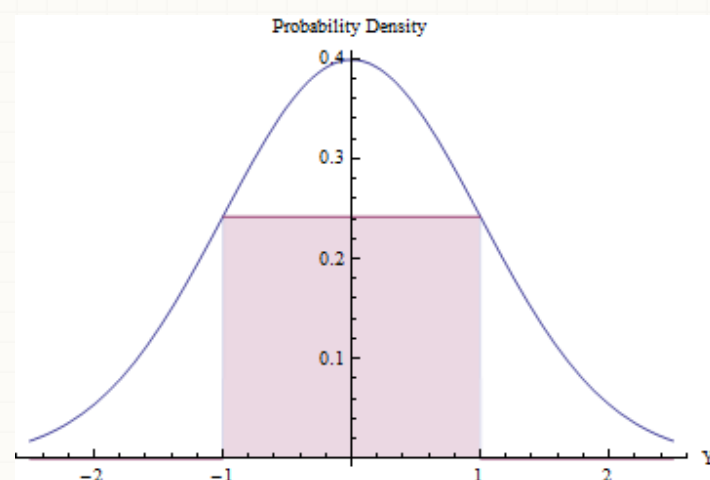
$$\sum_{j=0}^{k-1} H(X_{ki-j}) 2^{-j-1}.$$

The variates  $X_n$  can be drawn from *any* continuous distribution whose median is 0 (such as a standard Normal); they are processed in groups of  $k$  with each group producing one such pseudo-uniform value.

- Rejection sampling** is a standard, flexible, powerful way to draw random variates from arbitrary distributions. Suppose the target distribution has PDF  $f$ . A value  $Y$  is drawn according to *another* distribution with PDF  $g$ . In the rejection step, a uniform value  $U$  lying between 0 and  $g(Y)$  is drawn independently of  $Y$  and compared to  $f(Y)$ : if it is smaller,  $Y$  is retained but otherwise the process is repeated. This approach seems circular, though: how do we generate a uniform variate with a process that needs a uniform variate to begin with?

The answer is that we don't actually need a uniform variate in order to carry out the rejection step. Instead (assuming  $g(Y) \neq 0$ ) we can flip a fair coin to obtain a 0 or 1 randomly. This will be interpreted as the first bit in the binary representation of a uniform variate  $U$  in the interval  $[0, 1)$ . When the outcome is 0, that means  $0 \leq U < 1/2$ ; otherwise,  $1/2 \leq U < 1$ . *Half of the time, this is enough to decide the rejection step:* if  $f(Y)/g(Y) \geq 1/2$  but the coin is 0,  $Y$  should be accepted; if  $f(Y)/g(Y) < 1/2$  but the coin is 1,  $Y$  should be rejected; otherwise, we need to flip the coin again in order to obtain the next bit of  $U$ . Because--no matter what value  $f(Y)/g(Y)$  has--there is a  $1/2$  chance of stopping after each flip, the expected number of flips is only  $1/2(1) + 1/4(2) + 1/8(3) + \dots + 2^{-n}(n) + \dots = 2$ .

Rejection sampling can be worthwhile (and efficient) provided the expected number of rejections is small. We can accomplish this by fitting the largest possible rectangle (representing a uniform distribution) beneath a Normal PDF.



Using Calculus to optimize the rectangle's area, you will find that its endpoints should lie at  $\pm 1$ , where its height equals  $\exp(-1/2)/\sqrt{2\pi} \approx 0.241971$ , making its area a little greater than 0.48. By using this standard Normal density as  $g$  and rejecting all values outside the interval  $[-1, 1]$  automatically, and otherwise applying the rejection procedure, we will obtain uniform variates in  $[-1, 1]$  efficiently:

- In a fraction  $2\Phi(-1) \approx 0.317$  of the time, the Normal variate lies beyond  $[-1, 1]$  and is immediately rejected. ( $\Phi$  is the standard Normal CDF.)
  - In the remaining fraction of the time, the binary rejection procedure has to be followed, requiring two more Normal variates on average.
  - The overall procedure requires an average of  $1/(2 \exp(-1/2)/\sqrt{2\pi}) \approx 2.07$  steps.
- The expected number of Normal variates needed to produce each uniform result works out to

$$\sqrt{2e\pi} (1 - 2\Phi(-1)) \approx 2.82137.$$

Although that is pretty efficient, note that (1) computation of the Normal PDF requires computing an exponential and (2) the value  $\Phi(-1)$  must be precomputed once and for all. It's still a little less calculation than the Box-Mueller method (*q.v.*).

- The **order statistics** of a uniform distribution have exponential gaps. Since the sum of squares of two Normals (of zero mean) is exponential, we may generate a realization of  $n$  independent uniforms by summing the squares of pairs of such Normals, computing the cumulative sum of these, rescaling the results to fall in the interval  $[0, 1]$ , and dropping the last one (which will always

equal 1). This is a pleasing approach because it requires only squaring, summing, and (at the end) a single division.

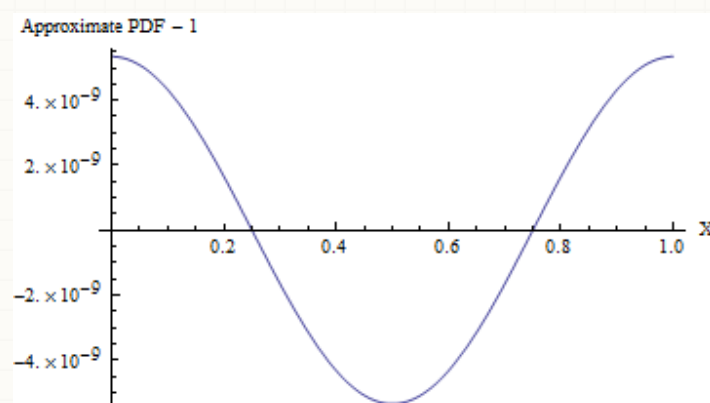
The  $n$  values will automatically be in ascending order. If such a sorting is desired, *this method is computationally superior to all the others* insofar as it avoids the  $O(n \log(n))$  cost of a sort. If a sequence of independent uniforms is needed, however, then sorting these  $n$  values randomly will do the trick. Since (as seen in the Box-Mueller method, *q.v.*) the ratios of each pair of Normals are independent of the sum of squares of each pair, we already have the means to obtain that random permutation: order the cumulative sums by the corresponding ratios. (If  $n$  is very large, this process could be carried out in smaller groups of  $k$  with little loss of efficiency, since each group needs only  $2(k+1)$  Normals to create  $k$  uniform values. For fixed  $k$ , the asymptotic computational cost is therefore  $O(n \log(k)) = O(n)$ , needing  $2n(1 + 1/k)$  Normal variates to generate  $n$  uniform values.)

8. To a superb approximation, **any Normal variate with a large standard deviation looks uniform** over ranges of much smaller values. Upon rolling this distribution into the range  $[0, 1]$  (by taking only the fractional parts of the values), we thereby obtain a distribution that is uniform for all practical purposes. This is extremely efficient, requiring one of the simplest arithmetic operations of all: simply round each Normal variate down to the nearest integer and retain the excess. **The simplicity of this approach becomes compelling** when we examine a practical `R` implementation:

```
rnorm(n, sd=10) %% 1
```

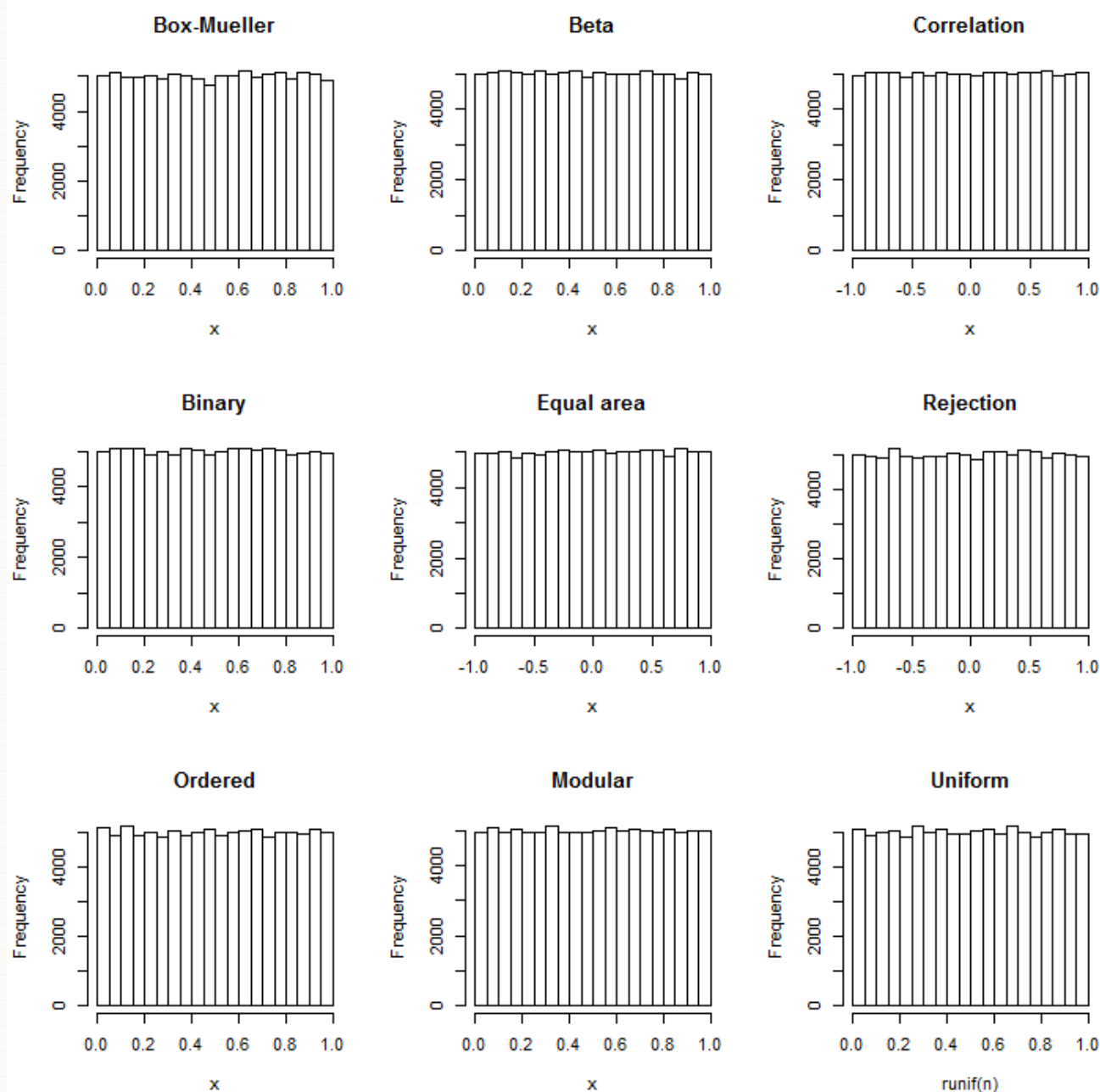
reliably produces `n` uniform values in the range  $[0, 1]$  at the cost of just `n` Normal variates and almost no computation.

(Even when the standard deviation is 1, the PDF of this approximation varies from a uniform PDF, as shown in the following figure, by less than one part in  $10^8$ ! To detect it reliably would require a sample of  $10^{16}$  values--that's already beyond the capability of any standard test of randomness. With a larger standard deviation the non-uniformity is so small it cannot even be calculated. For instance, with an SD of 10 as shown in the code, the maximum deviation from a uniform PDF is only  $10^{-857}$ .)



In every case Normal variables "with known parameters" can easily be recentered and rescaled into the Standard Normals assumed above. Afterwards, the resulting uniformly distributed values can be recentered and rescaled to cover any desired interval. These require only basic arithmetic operations.

The ease of these constructions is evidenced by the following `R` code, which uses only one or two lines for most of them. Their correctness is witnessed by the resulting near-uniform histograms based on 100,000 independent values in each case (requiring around 12 seconds for all seven simulations). For reference--in case you are worried about the amount of variation appearing in any of these plots--a histogram of uniform values simulated with `R`'s uniform random number generator is included at the end.



All these simulations were tested for uniformity using a  $\chi^2$  test based on 1000 bins; none could be considered significantly non-uniform (the lowest p-value was 3%--for the results generated by R's actual uniform number generator!).

```
set.seed(17)
n <- 1e5
y <- matrix(rnorm(floor(n/2)*2), nrow=2)
x <- c(atan2(y[2,], y[1,])/(2*pi) + 1/2, exp(-(y[1,]^2+y[2,]^2)/2))
hist(x, main="Box-Mueller")

y <- apply(array(rnorm(4*n), c(2,2,n)), c(3,2), function(z) sum(z^2))
x <- y[,2] / (y[,1]+y[,2])
hist(x, main="Beta")

x <- apply(array(rnorm(8*n), c(4,2,n)), 3, function(y) cor(y[,1], y[,2]))
hist(x, main="Correlation")

n.bits <- 32; x <- (2^-(1:n.bits)) %%% matrix(rnorm(n*n.bits) > 0, n.bits)
hist(x, main="Binary")

y <- matrix(rnorm(n*3), 3)
x <- y[1, ] / sqrt(apply(y, 2, function(x) sum(x^2)))
hist(x, main="Equal area")

accept <- function(p) { # Using random normals, return TRUE with chance `p`
  p.bit <- x <- 0
  while(p.bit == x) {
    p.bit <- p >= 1/2
    x <- rnorm(1) >= 0
    p <- (2*p) %%% 1
  }
  return(x == 0)
}
y <- rnorm(ceiling(n * sqrt(exp(1)*pi/2))) # This aims to produce `n` uniforms
y <- y[abs(y) < 1]
x <- y[sapply(y, function(x) accept(exp((x^2-1)/2)))]
hist(x, main="Rejection")

y <- matrix(rnorm(2*(n+1))^2, 2)
x <- cumsum(y)[seq(2, 2*(n+1), 2)]
x <- x[-(n+1)] / x[n+1]
x <- x[order(y[2,-(n+1)]/y[1,-(n+1)])]
hist(x, main="Ordered")

x <- rnorm(n) %%% 1 # (Use SD of 5 or greater in practice)
hist(x, main="Modular")

x <- runif(n) # Reference distribution
hist(x, main="Uniform")
```

edited Apr 13 '17 at 12:44



Community ♦

1

answered Oct 3 '14 at 4:47



whuber ♦

189k 30 401 746



- 2 (+1) If I were asking this question in an interview, I'd modify it to ask about the case where the parameters are fixed, but *unknown*, which strikes me as more interesting. The Pearson correlation approach (#3) goes through unchanged, but is perhaps slightly esoteric. The Beta approach (#2) requires only slight modification by considering the squares of differences of disjoint pairs. Similarly,  $Z = (X_1 - X_2)/(X_3 - X_4)$  is standard Cauchy (regardless of the mean and variance of  $X$ ), which has a nice cdf. – cardinal ♦ May 14 '15 at 17:30
- 1 More generally, the principle is to find a *pivotal quantity* from the sample with a computationally amenable cdf. This ties in nicely with constructing confidence intervals and hypothesis tests, with the twist that we might seek to optimize the number of elements used rather than the latter cases which focus more on maximizing the information from a fixed sample size. – cardinal ♦ May 14 '15 at 17:37

@Cardinal Thank you for the interesting comments, as well as the ninth method (Cauchy). Even finding a pivotal quantity is unnecessary when only a good approximation is sought. For instance, (8) works perfectly well if you reserve a small number of initial results to establish a rough scale. – whuber ♦ May 14 '15 at 18:49

Get personalized job matches now






Get started

You can use a trick very similar to what you mention. Let's say that  $X \sim N(\mu, \sigma^2)$  is a normal random variable with known parameters. Then we know its distribution function,  $\Phi_{\mu, \sigma^2}$ , and  $\Phi_{\mu, \sigma^2}(X)$  will be uniformly distributed on  $(0, 1)$ . To prove this, note that for  $d \in (0, 1)$  we see that

$$P(\Phi_{\mu, \sigma^2}(X) \leq d) = P(X \leq \Phi_{\mu, \sigma^2}^{-1}(d)) = d.$$

The above probability is clearly zero for non-positive  $d$  and 1 for  $d \geq 1$ . This is enough to show that  $\Phi_{\mu, \sigma^2}(X)$  has a uniform distribution on  $(0, 1)$  as we have shown that the corresponding measures are equal for a generator of the Borel  $\sigma$ -algebra on  $\mathbb{R}$ . Thus, you can just tranform the normally distributed data by the distribution function and you'll get uniformly distributed data.

answered Oct 2 '14 at 23:09

 **swmo**  
1,477 7 17

- 4 It's the inverse of inverse transform sampling! – Roger Fan Oct 2 '14 at 23:10

Could you possibly elaborate on the second sentence of your second paragraph? I am having trouble understanding the following: "This is enough to show that  $\Phi_{\mu, \sigma^2}(X)$  has a uniform distribution on  $(0,1)$  as we have shown that the corresponding measures are equal for a generator of the Borel  $\sigma$ -algebra on  $\mathbb{R}$ ." – wellington Oct 3 '14 at 2:56

To show that some real random variable,  $X$ , has a uniform distribution, we should show that its corresponding measure,  $X(P)$  equals that of the uniform distribution for all measurable sets of the real line. However, it's actually enough to consider some generator of the  $\sigma$ -algebra, due to a uniqueness of measures-theorem. If they are equal on sets of the generator, they'll be equal for all measurable sets. This is just a measure-theoretic attachment to the answer. – swmo Oct 3 '14 at 7:23