The Generation of Binomial Random Variates

Wolfgang Hörmann

Inst. f. Statistik Wirtschaftuniv. Wien, Augasse 2-6, A-1090 Wien, Austria.

e-mail: whoer@statrix2.wu-wien.ac.at

Abstract:

The transformed rejection method, a combination of inversion and rejection, which can be applied to various continuous distributions, is well suited to generate binomial random variates as well. The resulting algorithms are simple and fast, and need only a short set-up. Among the many possible variants two algorithms are described and tested: BTRS a short but nevertheless fast rejection algorithm and BTRD which is more complicated as the idea of decomposition is utilized. For BTRD the average number of uniforms required to return one binomial deviate lies between 2.5 and 1.4 which is considerably lower than for any of the known uniformly fast algorithms. Timings for a C-implementation show that for the case that the parameters of the binomial distribution vary from call to call BTRD is faster than the current state of the art algorithms. Depending on the computer, the speed of the uniform generator used and the binomial parameters the savings are between 5 and 40 percent.

Key Words:

Random numbers, binomial distribution, transformed rejection method, simulation.

1. Introduction

As it was pointed out in two recent surveys (Kachitvichyanukul and Schmeiser (1988), Stadlober (1989)) and in the book of Devroye (1986) the fastest binomial generators for fixed parameters n and p are obtained by table methods like the alias method first proposed by Walker (1977). But for these methods the memory requirements and the computing time to set up the table for new values of n and p are proportional to n. On the other hand simple inversion without a table results in a short algorithm. The set up time to compute constants depending on the parameters n and p is low but the execution time is proportional to $n \cdot \min(p, 1-p)$. Therefore rejection algorithms were proposed (for example Fishman (1979), Ahrens and Dieter (1980), Kachitvichyanukul and Schmeiser (1988)) because they are both uniformly fast and well suited for the changing parameter situation. But all of these methods require an average of more than two uniform deviates to obtain one binomial random number which is large compared with the table and inversion methods wich need only one uniform deviate. So we tried to design a rejection method for the Binomial distribution which requires less than two uniform deviates by using the idea of "transformed rejection" introduced by Wallace (1976). Marsaglia (1984) called a refinement of the method "exact-approximation", Devroye (1986) used the name "almost-exact inversion".

2. Transformed rejection

If we want to use the rejection method to sample from a distribution with density function proportional to f we need a dominating density or hat function h and a real number α with $f(x) \leq \alpha h(x)\ \forall x$. Then we generate a random number X from the dominating density and a uniform random number V. If $V \leq f(X)/(\alpha h(X))$ then X is accepted as a random number from the density f, otherwise the procedure starts again. For the transformed rejection method we start with the inverse distribution function $G(u)$ of the dominating distribution. (Random numbers of this distribution are of course generated by inversion.) As the dominating density is $\left(G^{-1}\right)'(x) = 1/G'(u)$ for $x = G(u)$ the acceptance condition can be transformed into:

$$V \leq f(G(U))G'(U)/\alpha$$

In the sequel let $m = \lfloor (n+1)p \rfloor$ be the mode of the binomial distribution and

$$f(x) = \frac{f_b(\lfloor x \rfloor)}{f_b(m)} = \frac{m!(n-m)!}{\lfloor x \rfloor!(n-\lfloor x \rfloor)!} \left(\frac{p}{1-p}\right)^{\lfloor x \rfloor - m}$$

the rescaled histogram with $f(m) = 1$. To obtain high acceptance probabilities $G(u)$ should be close to the inverse distribution function of the binomial distribution. The class $G(u) =$

$\left(\frac{2a}{1/2-|u|}+b\right)u+c$, $G'(u)=\frac{a}{(1/2-|u|)^2}+b$, $-0.5\le u\le 0.5$ is very flexible and turns out to yield high acceptance probabilities for a variety of distributions including the binomial distribution. Using this transformation (the choice of the parameters a, b and c will be discussed below) we are ready to give the basic transformed rejection algorithm.

Algorithm Transformed Rejection (TR):

1: Generate two uniform random numbers U and V. Set $U \leftarrow U-0.5$.

2: If $V \le f(G(U))G'(U)/\alpha$ return $\lfloor G(U)\rfloor$, else go to 1.

If the parameters a, b and c of G were chosen properly $f(G(U))G'(U)/\alpha$ is close to 1 over the interval (-0.5,0.5). Therefore it is possible to find a large rectangle between the curve and the u-axis which will be denoted by $(-u_r,u_r)\times(0,v_r)$. Using this rectangle as a simple squeeze reduces the number of evaluations of the acceptance condition considerably.

Algorithm Transformed Rejection with Squeeze (TRS):

1: Generate two uniform random numbers U and V. Set $U \leftarrow U-0.5$.

2: If $|U|\le u_r$ and $V\le v_r$ return $\lfloor G(U)\rfloor$.

3: If $V \le f(G(U))G'(U)/\alpha$ return $\lfloor G(U)\rfloor$, else go to 1.

To reduce the expected number of uniform deviates required the idea of decomposition in combination with "recycling" of uniform deviates is necessary. Thus we obtain:

Algorithm Transformed Rejection with Decomposition (TRD):

1: Generate a uniform random number V. If $V\le 2u_rv_r$ return $\lfloor G(V/v_r-u_r)\rfloor$.

2: If $V\ge v_r$ generate a uniform random number U in (-0.5,0.5),

otherwise set $U\leftarrow V/v_r-(u_r+0.5)$, $U\leftarrow \mathrm{sign}(U)\,0.5-U$,

and generate a uniform random number V in $(0,v_r)$.

3: If $V \leq f(G(U))G'(U)/\alpha$ return $\lfloor G(U) \rfloor$, else go to 1.

Algorithm TRD is a uniformly fast binomial generator that needs an average of less than two uniform deviates per sample if we are able to find good values for a, b and c. Without loss of generality we restricted our attention to the case $p \leq 0.5$. We used $c = np+0.5$ and computed the optimal values of a and b for many values of the binomial parameters n and p by numerical search. As there are short and fast inversion algorithms for small μ we approximated the optimal values by simple functions for $\mu = np \geq 10$ and $p \leq 0.5$. For these values of a, b and c we computed an upper bound for α and a lower bound for v_r. The results of these computations are contained in Table 1.

Table 1: Approximations valid for $\mu \geq 10$ and $p \leq 0.5$

c	$np + 0.5$
b	$1.15 + 2.53\sqrt{np(1-p)}$
a	$-0.0873 + 0.0248b + 0.01p$
α	$(2.83 + 5.1/b)\sqrt{np(1-p)}$
u_r	0.43
v_r	$0.92 - 4.2/b$

That the bounds for α and v_r are correct was checked numerically. As the binomial distribution tends to the Poisson distribution for $np = \mu$ constant and $n \to \infty$ the asymptotic values of a, b, c, α, u_r and v_r were checked to be correct for the Poisson distribution. For p fixed and $n \to \infty$ the binomial distribution tends to the normal distribution. For this case easy calculations show that the limits of the standardized parameters a, b and c are 2.53, 0.062744 and 0. The limit of the standardized α is 2.83, of v_r 0.92 which is both close to the optimal values of the rescaled density of the standard normal distribution with the above values of a, b and c wich are 2.81 and 0.9246 respectively. That the fit achieved with the approximations of Table 1 is good follows from Table 2 that contains the low number of uniform deviates required by Algorithms TRS and TRD. For the case $n = 50$ and $p = 0.25$ the good fit can be seen in Figure 1 that shows the curve $f(G(U))G'(U)/\alpha$ and the enclosed rectangle $(-u_r, u_r) \times (0, v_r)$. Figure 2 presents the rescaled binomial histogram for the same parameter values together with α times the density function associated with G.

Figure 1; Figure 2;

3. The algorithms

The above section described transformed rejection to generate binomial random variates but the question how to evaluate the acceptance condition is left open. One possibility is to compute the rescaled binomial histogram recursively from the mode m applying the formula: $f(k+1) = f(k)(n-k+1)p/(kq)$. But for k not close to m this method is slow. Therefore we agree with Stadlober (1989) that the simplest way to solve this problem is to compute the logarithm of the rescaled histogram with the Stirling approximation

$$\log(k!) = \log\sqrt{2\pi} + (k+\frac{1}{2})\log(k+1) - (k+1) + \frac{1}{12\,(k+1)} - \frac{1}{360\,(k+1)^3} + \frac{1}{1260\,(k+1)^5}$$

wich has truncation error less than $3.1 \cdot 10^{-11}$ for $k \geq 10$. For $k < 10$ the values of $\log(k!)$ can be stored in a table. We get:

$$\log(f(k)) = \log m! + \log(n-m)! - \log k! - \log(n-k)! + (k-m)\log\left(\frac{p}{q}\right)$$

As $\log(f(k))$ which is computed as a difference of large expressions can become relatively small cancellation can occur for large values of n. Obviously ${}_{10}\log(\log(n!))$ is an upper bound for the loss of accuracy in decimal digits. It is possible to reduce this loss of accuracy by substituting the Stirling approximation into the above formula. Thus we get:

$$\begin{aligned}\log(f(k)) \quad = \quad & (m+\frac{1}{2})\log\left(\frac{m+1}{n-m+1}\cdot\frac{1-p}{p}\right) + (n+1)\log\left(\frac{n-m+1}{n-k+1}\right) + \\ & (k+\frac{1}{2})\log\left(\frac{n-k+1}{k+1}\cdot\frac{p}{1-p}\right) + f_c(m) + f_c(n-m) - f_c(k) - f_c(n-k)\end{aligned}$$

where $f_c(k)$ denotes the correction term $\frac{1}{12\,(k+1)} - \cdots$ of the Stirling approximation. For $k \leq 9$ the exact values can be stored in a table. As both variants to compute the logarithm of the rescaled histogram are time consuming we also use the squeezes of Kachitvichyanukul and Schmeiser (1988) which are based on the DeMoivre-Laplace limit theorem. Now we can give the complete implementation of Algorithm TRD for the binomial distribution.

Algorithm BTRD ($n \cdot p \geq 10$ and $p \leq 0.5$):

0: [Set-up] Prepare function $f_c(0) = 0.08106146679532726$, $f_c(1) = 0.04134069595540929$, $f_c(2) = 0.02767792568499834$, $f_c(3) = 0.02079067210376509$, $f_c(4) = 0.01664469118982119$, $f_c(5) = 0.01387612882307075$, $f_c(6) = 0.01189670994589177$, $f_c(7) = 0.01041126526197209$, $f_c(8) = 0.009255462182712733$, $f_c(9) = 0.008330563433362871$ and

$f_c(k) = (1/12 - (1/360 - 1/1260/(k+1)^2)/(k+1)^2)/(k+1)$ for $k \geq 10$;

set $m \leftarrow \lfloor (n+1) * p \rfloor$, $r \leftarrow p/(1-p)$, $nr \leftarrow (n+1) * r$, $npq \leftarrow n * p * (1-p)$,

$b \leftarrow 1.15 + 2.53 * \sqrt{npq}$, $a \leftarrow -0.0873 + 0.0248 * b + 0.01 * p$, $c \leftarrow n * p + 0.5$,

$\alpha \leftarrow (2.83 + 5.1/b) * \sqrt{npq}$, $v_r \leftarrow 0.92 - 4.2/b$, $u_r v_r \leftarrow 0.86 * v_r$.

1: Generate a uniform random number v.

If $v \leq u_r v_r$ set $u \leftarrow v/v_r - 0.43$ and return $\lfloor (2a/(0.5 - |u|) + b) * u + c \rfloor$.

2: If $v \geq v_r$ generate a uniform random number u in (-0.5,0.5),

otherwise set $u \leftarrow v/v_r - 0.93$, $u \leftarrow \mathrm{sign}(u) * 0.5 - u$

and generate a uniform random number v in $(0, v_r)$.

3.0 Set $us \leftarrow 0.5 - |u|$, $k \leftarrow \lfloor (2 * a/us + b) * u + c \rfloor$. If $k < 0$ or $k > n$ go to 1.

Set $v \leftarrow v * \alpha/(a/(us * us) + b)$, $km \leftarrow |k - m|$; if $km > 15$ go to 3.2.

3.1 [Recursive evaluation of $f(k)$]

Set $f \leftarrow 1$.

If $m < k$ set $i \leftarrow m$ and repeat $i \leftarrow i + 1$, $f \leftarrow f * (nr/i - r)$ until $i = k$.

Otherwise if $m > k$ set $i \leftarrow k$ and repeat $i \leftarrow i+1$, $v \leftarrow v * (nr/i - r)$ until $i = m$.

If $v \leq f$ return k, otherwise go to 1.

3.2 [Squeeze-acceptance or rejection]

Set $v \leftarrow \log(v)$, $\rho \leftarrow (km/npq) * (((km/3 + 0.625) * km + 1/6)/npq + 0.5)$,

$t \leftarrow -km * km/(2 * npq)$.

If $v < t - \rho$ return(k), if $v > t + \rho$ go to 1.

3.3 [Set-up for 3.4]

Set $nm \leftarrow n - m + 1$, $h \leftarrow (m + 0.5) * \log((m+1)/(r * nm)) + f_c(m) + f_c(n-m)$.

3.4 [Final acceptance-rejection test]

Set $nk \leftarrow n - k + 1$.

If $v \leq h + (n+1) * \log(nm/nk) + (k + 0.5) * \log(nk * r/(k+1)) - f_c(k) - f_c(n-k)$

return k, otherwise go to 1.

If we want to obtain a simple, short and readable algorithm for binomial random variate generation it is best to combine Algorithm TRS with the simple Stirling approximation version to calculate $\log(f(k))$. We get:

Algorithm BTRS:($n \cdot p \geq 10$ and $p \leq 0.5$):

0: [Set-up]

Prepare function of $\log(k!)$ using Stirling approximation or table for $k \leq 9$;

Set $spq \leftarrow \sqrt{n * p * (1-p)}$, $b \leftarrow 1.15 + 2.53 * spq$, $a \leftarrow -0.0873 + 0.0248 * b + 0.01 * p$,

$c \leftarrow n * p + 0.5$, $v_r \leftarrow 0.92 - 4.2/b$.

1: Generate two uniform random numbers u and v.

Set $u \leftarrow u - 0.5$, $us \leftarrow 0.5 - |u|$, $k \leftarrow \lfloor (2 * a/us + b) * u + c \rfloor$.

2: If $us \geq 0.07$ and $v \leq vr$ return k. If $k < 0$ or $k > n$ go to 1.

3.0: [Set-up for 3.1]

Set $\alpha \leftarrow (2.83 + 5.1/b) * spq$, $lpq \leftarrow log(p/(1-p))$,

$m \leftarrow \lfloor (n+1) * p \rfloor$, $h \leftarrow \log(m!) + \log((n-m)!)$.

3.1: [Acceptance-rejection test]

Set $v \leftarrow v * \alpha/(a/(us * us) + b)$,

if $v \leq h - \log(k!) - \log((n-k)!) + (k-m) * lpq$ return k, otherwise go to 1.

Among the binomial generators that utilize transformed rejection BTRD is the fastest algorithm, BTRS the shortest one. Of course it is possible to use other combinations of the different ideas; for example adding the recursive evaluation in step 3.1 of Algorithm BTRD to Algorithm BTRS would speed up this generator considerably for small values of $\mu = n \cdot p$ with only little extra code. We also tested the accuracies of the two algorithms and computed the approximate probabilities that a wrong random number is returned due to numerical inaccuracies when 64-bit floating point numbers are used. For Algorithm BTRD this probability is lower than $n \cdot 6 \cdot 10^{-17}$ (about the same as for Algorithm BTPE of Kachitvichyanukul and Schmeiser (1988)) which means that the method is exact at least for $n \leq 10^7$ and can be used for $n \leq 2 \cdot 10^9$ (that are all positive integers representable on a 32-bit computer). As pointed out above the loss of accuracy due to cancellation is larger for Algorithm BTRS. The probability to generate a "wrong" random number is about 10^{-9} for $n = 10^6$ and about 10^{-6} for $n = 10^9$.

4. Comparison of algorithms

As Kachitvichyanukul and Schmeiser (1988) report that their binomial generator BTPE is much faster than any other binomial generator in the changing parameter situation we compared our new algorithms BTRD and BTRS only with BTPE and with two recent methods not included in their comparison: The search from the mode algorithm BKEMP (Kemp (1986)) and the ratio of uniforms algorithm BRUA (Stadlober (1989) and (1991)). As any of these algorithms is replaced by the simple inversion algorithm BIN for small values of μ we included BIN in our comparison. A first important theoretical measure is the time complexity of the different methods: Algorithms BTRD, BTRS, BTPE and BRUA have bounded computation times whereas the average execution time of BKEMP increases with $\sqrt{np(1-p)}$ the time of BIN with $\mu = np$. As a second important theoretical measure we computed the mean number of uniform deviates required to return one binomial variate.

Table 2: Mean number of uniforms required

$n \cdot p$	10		50		100		1000		10000	
p	0.5	0.001	0.5	0.001	0.5	0.001	0.5	0.001	0.5	0.001
BTRD	2.45	2.15	1.87	1.73	1.73	1.62	1.48	1.45	1.40	1.39
BTRS	2.82	2.68	2.55	2.47	2.47	2.41	2.33	2.31	2.28	2.27
BTPE	3.99	3.80	3.00	2.73	2.74	2.55	2.29	2.25	2.29	2.28
BRUA	3.47	3.23	3.04	2.95	2.95	2.88	2.80	2.78	2.76	2.75
BKEMP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
BIN	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 2 shows that BTRD needs by far the lowest number of uniforms among the uniformly fast algorithms. BTRS has the best fit among the rejection algorithms. To compare the execution times of the algorithms we coded them in C and tested them on three different computers (a DEC-Station 5200, a PC-386/25 with UNIX and a PC-386/20 with DOS). The relative execution times were about the same. Table 2 of course implies that the speed of the uniform generator has an important influence on the relative speed of the generators. We tested three different uniform generators: Two linear congruential generators with modulus 2^{32} and $2^{31}-1$ taking 1.4 and 2.4 μ-seconds and a multiple recursive linear congruential generator with modulus $2^{31}-1$ taking 4.5 μ-seconds, which has by far the longest period and the best lattice properties (c.f. L'Ecuyer 1990). Table 3 contains – as an example – our timing results on the DEC-Station using the multiple recursive uniform generator for the case that n and p remain fixed and that they vary after each call by a very small quantity. The execution times when using one of the two faster uniform generators can be easily deduced from Table 2. As a crude measure for the length and complexity of the algorithms column L of Table 3 contains the number of C statements of the algorithms including the extern functions necessary for BTRD and BTRS.

Table 3: Execution times in μ-seconds

$n \cdot p$	10		50		100		1000		10000		L
p	0.5	0.001	0.5	0.001	0.5	0.001	0.5	0.001	0.5	0.001	
					fixed parameter situation						
BTRD	22.7	20.9	20.1	19.2	19.2	18.0	15.9	15.2	14.0	13.7	60
BTRS	36.1	33.5	30.3	27.2	27.2	25.2	22.4	21.8	20.9	20.7	31
BTPE	37.3	40.5	28.4	29.1	26.6	26.8	19.8	18.8	17.3	16.9	67
BRUA	28.2	30.1	33.7	37.1	43.5	43.2	42.5	42.4	42.3	42.2	40
BKEMP	12.6	15.5	21.4	27.9	28.0	37.3	76.6	99.7	-	-	37
BIN	20.8	20.9	79.3	79.4	-	-	-	-	-	-	20
					varying parameter situation						
BTRD	35.2	33.7	33.2	32.6	32.6	31.5	29.2	28.2	26.8	26.4	
BTRS	61.5	56.0	49.4	45.0	44.9	41.7	37.1	36.2	34.7	34.5	
BTPE	53.6	56.8	44.7	45.4	42.9	43.1	36.1	35.1	33.6	33.1	
BRUA	62.0	62.1	66.0	68.9	75.7	75.1	73.9	74.1	73.5	74.2	
BKEMP	42.3	45.2	51.2	57.7	57.8	67.0	106.8	136.4	-	-	
BIN	38.3	37.8	96.4	95.9	-	-	-	-	-	-	

Our timing experiences on different computers with different uniform generators showed that in the varying parameter situation BTRD is faster than any other method. For the fixed parameter situation it is well known that the alias method and other table methods are much faster but among the methods without tables BTRD remains the fastest for $\mu \geq 50$, for smaller values of μ BKEMP performs better. Table 3 shows that – when using a high-quality but comparable slow uniform generator – the savings of BTRD compared with BTPE lie between 40 percent for small values of μ and 20 percent for large ones. BTRS has only half the code of BTPE but has about the same speed. On the other hand BTRS is on average considerably faster and has slightly less code than BRUA and BKEMP. An additional advantage of BTRS is the fact that it is toghether with BIN the only of the compared methods that transforms the uniform random numbers monotonely. Thus BTRS can be used for correlation induction almost without changes. Only synchronisation must be obtained by using two random number streams as it is described in Kachitvichyanukul et al. (1988).

We are convinced that due to its simplicity and speed BTRS in combination with BIN can be recommended for standard simulations that need binomial random variates. BTRD combined with BIN is – at least for $n \leq 2 \cdot 10^9$ – numerically stable, very fast and has small memory requirements. Therefore it is especially well suited for software libraries and for the case that the parameters are random variates themselves which are computed during execution.

References

Ahrens, J. H. and Dieter, U. (1980), Sampling from binomial and Poisson distributions: a method with bounded computation times, Computing 25, 193-208.

Devroye, L. (1986) Non-Uniform Random Variate Generation, Springer-Verlag, New York.

Fishman, G. S. (1979), Sampling from the binomial distribution on a computer, J. American Statistical Assosciation 74, 418-423.

Kachitvichyanukul, V., Cheng, S. J. and Schmeiser, B. W. (1988) Fast Poisson and binomial algorithms for correlation induction, Journal of Statistical Computation and Simulation 29, 17-33.

Kachitvichyanukul, V. an Schmeiser, B. W. (1988), Binomial random variate generation, Communications of the ACM 31, 216-222.

Kemp, C. D. (1986), A modal method for generating binomial random variates, Commun. Statist. A. - Theory Methods 15, 805-813.

P. L'Ecuyer (1990), Random numbers for simulation, Communications of the ACM 33, 85-97.

Marsaglia, G. (1984) The exact-approximation method for generating random variables in a computer, Journal of the American Statistical Association 79, 218-221.

Stadlober, E. (1989) Sampling from Poisson, binomial and hypergeometric distributions: Ratio of uniforms as a simple and fast alternative, Berichte der Math.-Stat. Sektion in der Forschungsgesellschaft Joanneum, Nr. 303, Graz.

Stadlober, E. (1991), Binomial random variate generation: a method based on ratio of uniforms, in: The Frontiers of Statistical Computation, Simulation & Modelling, P. R. Nelson Ed., American Sciences Press.

Walker, A. J. (1977), An efficient method for generating discrete random variables with general distributions, ACM Trans. Math. Software 3, 253-256.

Wallace, C. S. (1976) Transformed rejection generators for gamma and normal pseudo-random variables, Australian Computer Journal 8, 103-105.