MATHEMATICS OF COMPUTATION, VOLUME 31, NUMBER 140
OCTOBER 1977, PAGES 1009–1018

Computer Methods for Sampling From Student's t Distribution*

By A. J. Kinderman**, J. F. Monahan*** and J. G. Ramage**

Abstract. Several new algorithms for generating deviates from the t family for the degrees of freedom parameter $\geqslant 1$ are presented. Both acceptance-rejection and probability mixing procedures are developed. The new algorithms outperform traditional methods for generating deviates from the t family. Recommendations are made concerning choosing an algorithm suited to its application.

1. Introduction. The family of Student's t distributions has been widely used in both theoretical and applied work in statistics since Student's original paper in 1908 [14]. Traditional methods for generating samples from the t family make use of the representations of a t random variable as the ratio of a standard normal variable to the square root of an independent normalized χ^2 variable [7], and as the square root of an F random variable, generated as a transformation of a beta random variable [13].

Several new algorithms for generating deviates from the t family for the degrees-of-freedom parameter $\geqslant 1$ are developed in this paper. The choice of an algorithm will depend on the nature of the application. In many applications, a long sequence of deviates is needed for each of a few members selected from the family. Then it makes sense to use an algorithm which can be modified, with some setup cost, for each family member. In other cases, only a few deviates will be generated for a given choice of the degrees-of-freedom parameter, which will be frequently altered. The algorithm with repeated modifications would then be expensive relative to a fixed algorithm designed to exploit features common to the entire family.

The paper is organized as follows. Three acceptance-rejection procedures are discussed in Section 2. Two algorithms which make use of probability mixing are given in Section 3. The normal and Cauchy distributions, extreme cases of the t family considered in this paper, are discussed in Section 4. Implementation issues and timing comparisons for the algorithms are discussed in the final section. All of the algorithms outperform traditional methods for generating the t family. The choice of an algorithm will depend on whether the application requires few or many deviates for a

Received July 8, 1975.
AMS (MOS) subject classifications (1970). Primary 65C10; Secondary 68A55.
Key words and phrases. Random number generator, Student's t distribution.
*Work performed under the auspices of the ERDA.

**This research was supported in part by the National Science Foundation under Grant Number GS 38609 at Carnegie-Mellon University, Pittsburgh.
***This research was supported in part by the National Science Foundation under Grant Number SOC73–09243 and Grant Number GS 38609 at Carnegie-Mellon University.

given member of the t family. As expected, the modifiable algorithms perform best when the degrees-of-freedom parameter is held fixed.

2. **Acceptance-Rejection.** In the classical acceptance-rejection procedure [1] a variable is generated from the distribution with density function $f(x)$ by sampling a pair (u, x), where u is a uniform (0, 1) variable and x is an independent variable from a distribution with density proportional to an upper envelope $g(x)$,

$$f(x) \leqslant g(x), \qquad -\infty < x < \infty,$$

and then accepting x as a variable from the required distribution when

$$u \leqslant f(x)/g(x).$$

There is typically a tradeoff between efficiency of the procedure, which depends on the tightness of the upper envelope $g(x)$, and the ease of sampling from the distribution with density proportional to $g(x)$. Also, $f(x)/g(x)$ is usually costly to evaluate. A good acceptance-rejection algorithm will often combine a rather simple function $g(x)$ with upper and lower bounds on the ratio $f(x)/g(x)$,

$$b(x) \leqslant f(x)/g(x) \leqslant B(x), \qquad -\infty < x < \infty,$$

which are more easily computed than $f(x)/g(x)$ itself. The procedure is as follows:

1. Generate u.
2. Generate x from a distribution with density proportional to $g(x)$.
3. If $u \leqslant b(x)$, deliver x.
4. If $u > B(x)$, go to 1.
5. If $u \leqslant f(x)/g(x)$, deliver x; otherwise go to 1.

Here and in what follows, "generate u" (v, u_1, u_2) means obtain a pseudorandom deviate, uniformly distributed on the interval (0, 1). The phrase "deliver x" means that x is the desired deviate and no further computation is needed.

The density function of the t family is given by

$$t_\alpha(x) = c_\alpha(1 + x^2/\alpha)^{-(\alpha+1)/2},$$

where

$$c_\alpha = \Gamma((\alpha + 1)/2)/[(\pi\alpha)^{1/2}\Gamma(\alpha/2)].$$

This departure from the more conventional use of ν for the degrees-of-freedom parameter emphasizes that α is not restricted to be an integer. Though $t_\alpha(x)$ is a proper density function for all $\alpha > 0$, we restrict attention in this paper to values of $\alpha \geqslant 1$. This restricted t family thus ranges from Cauchy ($\alpha = 1$) to normal ($\alpha = \infty$). It is particularly convenient to work with the unnormalized densities for the t family,

$$u_\alpha(x) = t_\alpha(x)/c_\alpha = (1 + x^2/\alpha)^{-(\alpha+1)/2},$$

since they are all symmetric about 0, where they attain a common maximum of 1. The following inequalities are basic to all of the generation algorithms developed in this paper for the t family.

THEOREM 2.1 (TAIL BOUND). *For $\alpha \geq 1$ and $-\infty < x < \infty$,*

$$u_\alpha(x) \leq 1/x^2.$$

In the extreme tails this bound is sharp, in the sense that the ratio for $\alpha = 1$,

$$x^2 u_1(x) = (1 + 1/x^2)^{-1},$$

converges rapidly to 1 as $|x|$ increases.

Proof. For $\alpha = 1$, the proof is immediate. For $\alpha > 1$, let

$$s(x) = \ln(x^2 u_\alpha(x)) = \ln(x^2) - \tfrac{1}{2}(\alpha + 1)\ln(1 + x^2/\alpha).$$

Since

$$s'(x) = 2x^{-1}[1 - x^2(\alpha + 1)/2(\alpha + x^2)]$$

has its only zeros for $x_0^2 = 2\alpha/(\alpha - 1)$ which yields a maximum, we have $s(x) \leq s(x_0)$, where

$$\begin{aligned} s(x_0) &= \ln(2\alpha/(\alpha - 1)) - \tfrac{1}{2}(\alpha + 1)\ln(1 + 2/(\alpha - 1)) \\ &= \ln(2\alpha) - [\tfrac{1}{2}(\alpha + 1)\ln(\alpha + 1) + \tfrac{1}{2}(1 - \alpha)\ln(\alpha - 1)]. \end{aligned}$$

Let $\lambda = \tfrac{1}{2}(\alpha + 1)$, $a = \alpha + 1$, $b = \alpha - 1$. Then $\tfrac{1}{2}(1 - \alpha) = 1 - \lambda$ and $\lambda a + (1 - \lambda)b = 2\alpha$. Since $\ln(x)$ is concave and $\lambda > 1$ for $\alpha > 1$, $s(x_0) < 0$. Thus, $s(x) = \ln(x^2 u_\alpha(x)) < 0$ for all x. □

THEOREM 2.2 (INNER TRIANGLE). *For $\alpha \geq 1$ and $-\infty < x < \infty$,*

$$u_\alpha(x) \geq 1 - |x|/2.$$

The triangle formed by the lower bound for $|x| \leq 2$ is the largest possible fixed triangle under the unnormalized t family.

Proof. The proof is immediate for $|x| \geq 2$. By symmetry about 0, we need only consider $x > 0$. Since $u_\alpha(x)$ has derivatives

$$u'_\alpha(x) = -x[(\alpha + 1)/(\alpha + x^2)]u_\alpha(x), \quad \text{and}$$

$$u''_\alpha(x) = [(\alpha + 2)x^2 - \alpha][(\alpha + 1)/(\alpha + x^2)^2]u_\alpha(x),$$

$u_\alpha(x)$ is concave for $x^2 \leq \alpha/(\alpha + 2) = x_0^2$, say, and convex for $x^2 \geq x_0^2$. Also, the tangent line to $u_\alpha(x)$ at $x = 1$ is

$$l_\alpha(x) = (2 - x)u_\alpha(1).$$

Note that $l_1(x) = 1 - x/2$ is the lower bound of interest. By convexity, $l_\alpha(x)$ lies below $u_\alpha(x)$ for $x^2 \geq \alpha/(\alpha + 2)$. If $u_\alpha(1)$ is written as $u_\alpha(1) = (1 - 1/(\alpha + 1))^{(\alpha+1)/2}$, it is clear that $u_\alpha(1)$ is increasing in α; and hence,

$$l_1(x) \leq l_\alpha(x) \quad \text{for } 0 \leq x \leq 2, \alpha \geq 1.$$

As noted above, by the convexity of $u_\alpha(x)$ for $x \geq x_0$, it is immediate that

$$l_1(x) \leq l_\alpha(x) \leq u_\alpha(x) \quad \text{for } x_0 \leq x \leq 2.$$

Similarly, if $f_\alpha(x) = u_\alpha(x) - l_1(x)$, we have

$$f_\alpha(x) \text{ is concave for } x \leqslant x_0, \quad f_\alpha(x_0) \geqslant 0, \quad f_\alpha(0) = 0,$$

and, by the concavity of $f_\alpha(x)$,

$$f_\alpha(x) = u_\alpha(x) - l_1(x) \geqslant 0 \quad \text{for } 0 \leqslant x \leqslant x_0. \ \square$$

The first algorithm for generating the t family by rejection makes use of an upper envelope and lower bound provided by Theorems 2.1 and 2.2. Deviates with a density proportional to the upper envelope,

$$g^*(x) = \begin{cases} 1, & |x| < 1, \\ 1/x^2, & |x| \geqslant 1, \end{cases}$$

are very easily generated by computer. Since the central ($|x| < 1$) and tail areas under $g^*(x)$ are equal, it suffices to generate with equal frequency the deviates $\pm u$ (center) and $\pm 1/u$ (tail). Note that $1/u$ has density $1/x^2$ for $x \geqslant 1$. This upper envelope is also reasonably good in the sense that the area under $u_\alpha(x)$ lies between $\sqrt{2\pi}$ and π, as compared with area 4 under $g^*(x)$. Thus the expected number of uniform pairs required per deviate is $4c_\alpha$, where

$$1.27 < 4c_\alpha < 1.60.$$

The triangular lower bound

$$h(x) = 1 - |x|/2$$

is checked for $|x| < 2$ to save computation of the more complex function $u_\alpha(x)$. The complete algorithm is as follows.

Algorithm TAR: (*Acceptance-Rejection*).

1. Generate u. If $u < .5$, go to 2. Set $x = 4u - 3$. Generate v. Go to 3.
2. Set $x = .25/(u - .25)$. Generate u_1 and set $v = x^{-2}u_1$.
3. If $v < 1 - |x|/2$, deliver x. If $v < u_\alpha(x)$, deliver x. Otherwise go to 1.

The basic acceptance-rejection algorithm can be improved somewhat by means of the upper bound provided in the following result.

THEOREM 2.3 (UPPER BOUND FOR $u_\alpha(x)$). *For* $\alpha \geqslant 1$ *and* $-\infty < x < \infty$,

$$u_\alpha(x) \leqslant H_\alpha(x) \leqslant H(x),$$

where

$$H_\alpha(x) = 2u_1(x)u_\alpha(1) = 2(1 + x^2)^{-1}(1 + 1/\alpha)^{-(\alpha+1)/2}$$

and

$$H(x) = 2u_1(x)u_\infty(1) = 2(1 + x^2)^{-1}e^{-1/2}.$$

Equality is attained for the first bound only when $\alpha = 1$.

Proof. We want to show $u_\alpha(x)u_1(1) \leqslant u_1(x)u_\alpha(1)$. Let $s(x) = u_\alpha(x)/u_1(x)$, then it is sufficient to verify $s(x) \leqslant s(1)$ for $\alpha > 1$. Since the derivative of s is

$$s'(x) = x[2/(1 + x^2) - (\alpha + 1)/(\alpha + x^2)]s(x),$$

$s'(x)$ has its only zeros for $x_0 = 0$ or $x_0^2 = 1$ if $\alpha > 1$. If $s(1)$ is written as $2(1 - 1/(\alpha + 1))^{(\alpha+1)/2}$, it is clear that $s(1) > 1$ for $\alpha > 1$. Since $s(0) = 1$ and $\lim_{x\to\infty} s(x) = 0$ for $\alpha > 1$, $s(1)$ is the maximum value of $s(x)$. □

The new upper bounds provided by Theorem 2.3 will of course be most useful where they lie beneath the upper envelope $g^*(x)$. It is easily verified that $H_\alpha(x) \leqslant g^*(x)$ iff $|x|$ lies between $(2u_\alpha(1) - 1)^{1/2}$ and $(2u_\alpha(1) - 1)^{-1/2}$. Since $u_\alpha(1)$ is increasing in α, the fixed upper bound $H(x)$ is useful for $|x|$ between $(2u_\infty(1) - 1)^{1/2}$ and $(2u_\infty(1) - 1)^{-1/2}$, or for $|x| \in [(2e^{-1/2} - 1)^{1/2}, (2e^{-1/2} - 1)^{-1/2}]$.

Two refinements of the basic acceptance-rejection algorithm TAR make use of the upper bounds of Theorem 2.3. In both, it is convenient to make use of the upper bounds only out to 2, which is only slightly smaller than $(2e^{-1/2} - 1)^{-1/2}$. The first uses the upper bound $H(x)$, which is independent of α. The second uses the tighter upper bound $H_\alpha(x)$, which requires the relatively expensive calculation $u_\alpha(1) = (1 + 1/\alpha)^{-(\alpha+1)/2}$. In many applications, a long sequence of deviates for a given α will be required, and α will be changed only infrequently. In such cases, $u_\alpha(1)$ need be computed only once for each choice of α, and it will be advantageous to use the second algorithm. In other applications, α may be changed frequently, and the first algorithm, which avoids the setup cost of computing $u_\alpha(1)$, will be preferred. The refined acceptance-rejection algorithms are as follows.

Algorithm TIR: (*Improved Rejection*).

0. Calculate the constants $b = (2e^{-1/2} - 1)^{1/2} = .4615856577$, $b/2$ and $1 + b^2 = 2e^{-1/2}$.
1. Generate u. If $u \geqslant b/2$, go to 3. Set $x = 4u - b$.
2. Generate v. If $v \leqslant 1 - |x|/2$, deliver x. If $v \leqslant u_\alpha(x)$, deliver x; otherwise go to 1.
3. If $u \geqslant .5$, go to 5. Set $x = [|4u - 1 - b| + b]\ \text{sign}(4u - 1 - b)$. Generate v.
4. If $v \leqslant 1 - |x|/2$, deliver x. If $v \geqslant (1 + b^2)/(1 + x^2)$, go to 1. If $v \leqslant u_\alpha(x)$ deliver x; otherwise go to 1.
5. If $u \geqslant .75$ go to 6. Set $x = 2/[|8u - 5| + 1]\,\text{sign}(8u - 5)$. Generate u_1, set $v = x^{-2}u_1$, and go to 4.
6. Set $x = 2/(8u - 7)$. Generate v. If $v < x^2u_\alpha(x)$, deliver x; otherwise go to 1.

Algorithm TIRS: (*Improved Rejection With Setup*).

0. Set $b = (2u_\alpha(1) - 1)^{1/2}$ and calculate $b/2$ and $(1 + b^2) = 2u_\alpha(1)$.
1.–6. Identical to algorithm TIR. For repeated calls with the same α, enter algorithm at 1.

3. Mixing. The method of probability mixing is based on generating a deviate from a randomly chosen member of a set of distributions according to some fixed weights. If the distributions $F_1, \ldots, F_k$ are chosen with probabilities $p_1, \ldots, p_k$, the generated deviate has distribution

$$F = p_1 F_1 + \cdots + p_k F_k.$$

Efficient algorithms for the target distribution F are achieved by generating deviates from very simple component distributions with high probability and correcting the overall mixture with deviates from more complex component distributions with low probability. This method has previously been applied to construct individual algorithms for specific distributions or specific members of families of distributions [1], [6], [10], [11], [12].

The mixing algorithms for the t family presented in this section are based on decomposing the t density into two components:

$$t_\alpha(x) = p_1 f_1(x) + p_2 f_2(x).$$

The first component, the triangular density

$$f_1(x) = \begin{cases} \frac{1}{2}(1 - |x|/2) & \text{for } |x| < 2, \\ 0 & \text{otherwise,} \end{cases}$$

is generated with probability $p_1 = 2c_\alpha$,

$$2c_1 = 2/\pi \leqslant p_1 \leqslant \sqrt{2/\pi} = 2c_\infty.$$

With probability $p_2 = 1 - p_1$, a random variable with density proportional to the difference function

$$\begin{aligned} p_2 f_2(x) &= t_\alpha(x) - p_1 f_1(x) \\ &= \begin{cases} c_\alpha(u_\alpha(x) - (1 - |x|/2)) & \text{for } |x| < 2, \\ c_\alpha u_\alpha(x) & \text{otherwise,} \end{cases} \end{aligned}$$

is generated by rejection techniques similar to those of the previous section.

In using a mixing technique for the t family, there is no way to completely avoid calculations dependent on α. In the setup case, where many deviates are drawn for a given α, computations independent of x are done only once per choice of α and hence are not regarded as costs. When α is to be changed frequently, however, it is worthwhile reducing the α-dependent computations as much as possible. The first mixing algorithm described below reduces dependence on α as follows. According to the bounds for the first mixing probability p_1, $f_1(x)$ will always (any α) be carried out $2/\pi$ of the time, and $f_2(x)$, $(1 - \sqrt{2/\pi})$ of the time. For the remaining $(\sqrt{2/\pi} - 2/\pi)$ of the time, $p_1 = 2c_\alpha$ must be computed exactly to attain the proper mixture.

Both versions of the mixing algorithms also reduce dependence on α by restricting the triangular component of the mixture to the range $|x| < 2$. The triangles

$f_1(x)$ are thus not the largest possible except in the extreme case $\alpha = 1$. In the worst case, however, when $\alpha = \infty$ (normal), the largest triangle extends only to approximately $|x| = 2.2$. Then the area of the largest triangle is approximately .88, as compared with $\sqrt{2/\pi} = .80$ for the fixed-base triangle [6].

The acceptance-rejection part of the mixture can fortunately be carried out in an unnormalized form comparable to those of Section 2, as is evident from the factorization of $f_2(x)$ above. The central part of the acceptance-rejection ($|x| < 2$) makes use of an outer envelope consisting of the (numerically determined) best two-piece step function above the family of difference functions,

$$u_\alpha(x) - (1 - |x|/2).$$

In the setup version, the upper bound $H_\alpha(x)$ is used as in algorithm TIRS to improve the rejection.

Algorithm TMX: (*Mixing*).

1. Generate u. If $u \leqslant 2/\pi$, go to 2. If $u \geqslant \sqrt{2/\pi}$ go to 3. Compute c_α. If $u \geqslant 2c_\alpha$, go to 3.
2. Generate u_1, u_2. Deliver $x = 2(u_1 + u_2 - 1)$.
3. Generate u. If $u \leqslant .3622520694$ go to 5. Set $x = 1/(1.0680176321 - 1.5680176321u)$. Generate v.
4. If $v \leqslant x^2 u_\alpha(x)$, deliver x; otherwise, go to 3.
5. If $u \leqslant .0530096080$, go to 7. Set $x = 11.5909050257u - 2.406629332$. Generate v.
6. If $.13528v \leqslant u_\alpha(x) - 1 + |x|/2$, deliver x; otherwise, go to 3.
7. Set $x = \text{sign}(7.840088159u - .2078)[|7.840088159u - .2078| + 1.7922]$. Generate v.
8. If $.2v \leqslant u_\alpha(x) - 1 + |x|/2$, deliver x: otherwise, go to 3.

Algorithm TMXS: (*Mixing With Setup*).

0. Compute c_α and $(1 + b^2) = 2u_\alpha(1)$.
1. Generate u. If $u \geqslant 2c_\alpha$, go to 3.
2. Generate u_2 and deliver $x = 2(u_2 - 1 + u/2c_\alpha)$.

3.–8. Identical to algorithm TMX, with the insertion

5.5. If $.13528v \geqslant (1 + b^2)/(1 + x^2) - 1 + |x|/2$, go to 3.

4. Special Cases: Normal and Cauchy. When a particular member of the t family is singled out for special attention, specific features of the distribution selected can be exploited to improve on the algorithms for the general family, even for those with setup features. For the t family, the extreme cases $\alpha = \infty$ (normal) and $\alpha = 1$ (Cauchy) are often of special interest. The normal case has received a great deal of attention (for recent discussions, see, for example, [1], [2]). A mixing algorithm (KR) based on the largest triangle beneath the normal density and exploiting a special technique for rejection over triangles [9] was developed by two of the authors [6].

The Cauchy distribution has not received comparable attention. Two traditional methods involve generating a Cauchy variable as the ratio of two independent standard

normal deviates [4] or as the tangent of a uniform deviate on $(-\pi/2, \pi/2)$ [3]. A variant of the second method, sometimes known as the synthetic tangent algorithm,† yields a Cauchy deviate by generating a point (u, v) uniformly on the half-circle $\{0 \leqslant u \leqslant 1, -1 \leqslant v \leqslant 1, u^2 + v^2 \leqslant 1\}$ and delivering $x = v/u$. Algorithm TMX was refined to exploit features peculiar to the Cauchy distribution. However, the resulting algorithm proved inferior to the synthetic tangent algorithm CST, and is not included here.

Algorithm CST: (*Synthetic Tangent*).

1. Generate u, v and set $v = 2v - 1$, $w = u^2 + v^2$.
2. If $w > 1$ go to 1, otherwise deliver $x = v/u$.

5. Discussion. All of the algorithms were written as FORTRAN functions and tested on the IBM system 360/model 67 under the operating system TSS. Uniform deviates were generated by an assembler language subroutine based on the Lewis, Goodman, and Miller [8] algorithm. For each t algorithm, 1,000 single deviates were generated in a DO-loop for twenty different choices of α. Ten repetitions were performed. The times (in microseconds per deviate) are reported in Table I for selected α.†† Algorithm TMX is reported for two versions: in the first, c_α is obtained by a call of a FORTRAN function which performed a table look-up, while in the second, the FORTRAN function computed c_α using standard function calls to EXP and ALGAMA. The table also included times for CTN, the tangent algorithm for the Cauchy distribution, CAR, algorithm TAR implemented for $\alpha = 1$ (in which $1/(1 + x^2)$ replaces $(1 + x^2/\alpha)^{-(\alpha+1)/2}$), and KR, an algorithm for the normal distribution [6].

Some general remarks can be made about the algorithms, based on the results in Table I. Good mixing algorithms can be written for a specific distribution, e.g. normal (KR), or for the t family when the necessary constants are known (TMXS). The mixing algorithms are less efficient when the constants must be computed for each call (TMX). The acceptance-rejection algorithms (TIR, TIRS) are competitive for the t family because the unnormalized densities $u_\alpha(x)$ all have a similar bell shape and, as noted in Section 2, fit between an easily generated and close fitting upper envelope and a good lower bound.

All of the new algorithms are to be preferred to the classic method for integer α of generating $\alpha + 1$ independent standard normal deviates and forming $t = \sqrt{\alpha} x_{\alpha+1}/\sqrt{\Sigma_1^\alpha x_i^2}$. Because the number of normal deviates required grows linearly in α, this algorithm will be slower than any of the new algorithms for even moderate α, say 3 or 4. Another alternative, valid for all α, is to let $t = \sqrt{\alpha} x/\sqrt{2y}$, where x is a standard normal deviate and y is a deviate from the standard gamma distribution with

---

†The authors thank David Hoaglin and the referee for communicating this algorithm which was overlooked in the original version of this paper. Generalization of this algorithm is the object of current research [5].

††Times for the complete list of $\alpha = 1, 1.5, 2, 2.5, 3, 4, 5, 6, 7, 8, 10, 15, 20, 25, 30, 40, 50, 65, 80, 100$ and FORTRAN listings of the algorithms are available from the authors upon request.

parameter $\alpha/2$ ($2y$ has a χ^2 distribution with α degrees of freedom). Dieter and Ahrens [3] have developed several algorithms for the gamma distribution. A FORTRAN function for the t distribution based on the gamma algorithm GO [3] was written which required about 1950 microseconds/deviate and 1500 microseconds/deviate for regular and setup versions, respectively. Since it is only valid for α greater than 5.07 [3], it was not included in Table I.

Similarly, algorithms based on generating an F deviate from a beta deviate would require two standard gamma deviates or special methods for generating beta($\frac{1}{2}, \alpha/2$) deviates. Since special methods are known only for beta distributions with parameters greater than 1 [3], this algorithm would require two gamma deviates and hence, would take more than 3000 microseconds/deviate.

TABLE I

	degrees of freedom α					
Algorithm	1	2	5	10	30	100
TAR	749	823	866	880	898	1027
TIR	709	798	838	862	876	1015
TIRS	607	688	754	773	803	931
TMX	688	756	796	803	822	967
TMX	906	958	993	1020	990	1164
TMXS	577	645	695	714	731	877
CTN	424	–	–	–	–	–
CAR	414	–	–	–	–	–
CST	320	–	–	–	–	–
KR	–	–	–	–	–	275($\alpha = \infty$)

Times in microseconds/deviate

The algorithms presented here are all easily programmed and valid for all $\alpha \geqslant 1$. In particular, TAR is the shortest and simplest and is only 20 to 30 percent slower than the fastest algorithm, TMXS, and that occurs only when repeated calls are made for the same α. Any one of TAR, TIR, and TMX would serve as a good general-purpose algorithm for generating deviates from the t family, and TIRS or TMXS would perform well for the user who desires long sequences of deviates from the same t distribution.

Department of Management Science
California State University, Northridge
Northridge, California 91330

Applied Mathematics Department
Brookhaven National Laboratory
Upton, New York 11973

Department of Statistics
University of Pennsylvania
Philadelphia, Pennsylvania 19174

1. J. H. AHRENS & U. DIETER, "Computer methods for sampling from the exponential and normal distributions," *Comm. ACM*, v. 15, 1972, pp. 873–881. MR **49** #1728.

2. J. H. AHRENS & U. DIETER, "Extensions of Forsythe's method for random sampling from the normal distribution," *Math. Comp.*, v. 27, 1973, pp. 927–937. MR **48** #7532.

3. J. H. AHRENS & U. DIETER, "Computer methods for sampling from gamma, beta, Poisson, and binomial distributions," *Computing (Arch. Electron. Rechnen.)*, v. 12, 1974, pp. 223–246. MR **52** #15949.

4. D. F. ANDREWS ET AL., *Robust Estimates of Location*, Princeton Univ. Press, Princeton, N. J., 1972. MR **48** #9927.

5. A. J. KINDERMAN & J. F. MONAHAN, "Computer generation of random variables using the ratio of uniform deviates," *ACM Trans. Math. Software.* (To appear.)

6. A. J. KINDERMAN & J. G. RAMAGE, "Computer generation of normal random variables," *J. Amer. Statist. Assoc.*, v. 71, 1976, pp. 893–896.

7. D. E. KNUTH, *The Art of Computer Programming*. Vol. 2: *Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1969. MR **44** #3531.

8. P. A. W. LEWIS, A. S. GOODMAN & J. M. MILLER, "A pseudo-random number generator for the system/360," *IBM Systems J.*, v. 8, 1969, pp. 136–146.

9. G. MARSAGLIA, "Random variables and computers," *Trans. Third Prague Conf. on Information Theory, Statistics, Decision Functions, and Random Processes*, Publ. House Czech. Acad. Sci., Prague, 1964, pp. 499–512. MR **29** #1721.

10. G. MARSAGLIA, "One-sided approximations by linear combinations of functions," *Approximation Theory* (A. Talbot, Editor), Academic Press, New York, 1970, pp. 233–242. MR **42** #1307.

11. G. MARSAGLIA & T. A. BRAY, "A convenient method for generating normal variables," *SIAM Rev.*, v. 6, 1964, pp. 260–264. MR **30** #2660.

12. G. MARSAGLIA, M. D. MACLAREN & T. A. BRAY, "A fast procedure for generating normal random variables," *Comm. ACM*, v. 7, 1964, pp. 4–10.

13. T. G. NEWMAN & P. L. ODELL, *The Generation of Random Variables*, Hafner, New York, 1971.

14. "STUDENT", "The probable error of a mean," *Biometrika*, v. 6, 1908, pp. 1–25.