Algorithm AS 204: The Distribution of a Positive Linear Combination of χ^2 Random Variables
Author(s): R. W. Farebrother
Source: *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 33, No. 3 (1984), pp. 332-339
Published by: Wiley for the Royal Statistical Society
Stable URL: http://www.jstor.org/stable/2347721

```
        comment calculation of denominators and numerators of
        new estimates;

        newmean[j] := nt[j] / dt[j];
        if j ≠ k then
            begin
            newalpha[j] := alpha[j] × dt[j] / nobvs;
            sumalpha := sumalpha + newalpha[j]
            end
        else newalpha[k] := 1.0 − sumalpha;
        if a = 1 then newsd[j] := sqrt(vt[j] / dt[j]);
        if abs(oldlogl − logl) > tol then test := true;
        oldlogl := logl; alpha[j] := newalpha[j];
        mean[j] := newmean[j];
        if a = 1 then sd[j] := newsd[j]
        end j loop;
    if c > 0 then
        begin
        if (counter − c) × c = counter then
            putout(k, alpha, mean, sd, logl)
        end
    end counter loop;
last:
  end mixture
```

Algorithm AS 204

The Distribution of a Positive Linear Combination of χ^2 Random Variables

by R. W. Farebrother

University of Manchester, UK

[Received April 1983. Revised January 1984]

Keywords: Chi-squared variables; Linear combination; Normal variables; Positive definite quadratic form; Series representation

Language

Algol 60

Description and Purpose

The algorithm presented in this paper uses Ruben's (1962) method to evaluate the expression

$$\Pr\left[\sum_{j=1}^{n} \lambda_j \chi^2(m_j, \delta_j^2) < c\right], \tag{1}$$

Present address: Dept of Economics, The University, Manchester M13 9PL, UK.

 0035–9254/84/33332 $2.00

where λ_j and c are positive constants and where $\chi^2(m_j, \delta_j^2)$ represents an independent χ^2 random variable with m_j degrees of freedom and non-centrality parameter δ_j^2.

Our algorithm is essentially an Algol 60 translation of the second part of Sheil and O'Muircheartaigh's (1977) algorithm AS 106 which eliminates the possibility of underflow in *dans*, traps the possible underflow of *ao*, amends an inner loop so that AS 204 is almost twice as fast as AS 106 and allows the user to choose a value for the parameter β. Our algorithm does not incorporate the first part of AS 106 which is concerned with the reduction of the positive definite quadratic form in normal variables

$$(x+b)' A(x+b)$$

to the linear form in noncentral χ^2 variables

$$\sum_{j=1}^{n} \lambda_j \chi^2(m_j, \delta_j^2)$$

where A is an $m \times m$ positive definite matrix, b is an $m \times 1$ matrix, x is an $m \times 1$ matrix of standard normal variables, and $m = \Sigma_{j=1}^{n} m_j$.

Method

Generalizing the earlier work of Robbins and Pitman (1949), Ruben (1962, p. 550) has shown that (1) may be expanded as the infinite series

$$\sum_{k=0}^{\infty} a_k F(m+2k, c/\beta), \tag{2}$$

where

$$F(m+2k, c/\beta) = \Pr\,[\chi^2(m+2k) < c/\beta]$$

and β is an arbitrary positive constant, and has shown that (2) converges uniformly on every finite interval of $c > 0$.

If (2) is a mixture representation of (1), that is if the a_k are nonnegative and sum to unity

$$a_k \geqslant 0, \quad \sum_{k=0}^{\infty} a_k = 1,$$

then the truncated series

$$\sum_{k=0}^{K-1} a_k F(m+2k, c/\beta) \tag{3}$$

will necessarily lie between zero and one and the truncation error will be bounded above by Ruben (1962, p. 566)

$$\left| \sum_{k=K}^{\infty} a_k F(m+2k, c/\beta) \right| \leqslant \left| 1 - \sum_{k=0}^{K-1} a_k \right| F(m+2K, c/\beta). \tag{4}$$

On the other hand, if some of the a_k are negative then (3) may take negative values and the truncation error will not necessarily satisfy (4). Ruben (1962, p. 566) gives a second upper bound for the truncation error which is usually sharper than (4) and which is applicable when $0 \leqslant \mu \leqslant 1$ where

$$\mu = \max_j |1 - \beta/\lambda_j| + (\beta/2) \sum_{j=1}^{n} \delta_j^2/\lambda_j.$$

However, this bound seems to be too inconvenient for routine application and our algorithm therefore uses (4) whether or not the chosen value of β corresponds to a mixture representation.

Ruben (1962, p. 564) has shown that (2) is a mixture representation if $\beta \leqslant \lambda_{\min}$ and that the a_k sum to unity if and only if $\beta < 2\lambda_{\min}$ but notes (p. 567) that "it is not always advisable to choose a value of β which yields a mixture representation". Algorithm AS 106 restricts the user to the choice

$$\beta_A = 0.90625\ \lambda_{\min}$$

without explaining why this value of β was chosen whilst Ruben himself recommends (p. 567)

$$\beta_B = 2/(1/\lambda_{\min} + 1/\lambda_{\max}).$$

Structure

real procedure *RUBEN* (*lambda*, *mult*, *delta*, *n*, *c*, *mode*, *maxit*, *eps*, *dnsty*, *ifault*)

Formal parameters

lambda	Real array [1 : *n*]	input:	the weights $\lambda_1, \lambda_2, \ldots, \lambda_n$
mult	Integer array [1 : *n*]	input:	the multiplicities $m_1, m_2, \ldots, m_n$
delta	Real array [1 : *n*]	input:	the noncentrality parameters $\delta_1^2, \delta_2^2, \ldots, \delta_n^2$
n	Integer	value:	the number of terms in equation (1)
c	Real	value:	the argument of equation (1)
mode	Real	value:	if *mode* > 0 then $\beta = mode \times \lambda_{\min}$ otherwise $\beta = \beta_B$
maxit	Integer	value:	the maximum number of terms in equation (3)
eps	Real	value:	the desired level of accuracy
dnsty	Real	output:	the density of the linear form
ifault	Integer	output:	the fault indicator

Failure indications

ifault $= -i$	one or more of the constraints $\lambda_i > 0$, $m_i > 0$ and $\delta_i^2 \geqslant 0$ is not satisfied
ifault = 1	non-fatal underflow of *ao*
ifault = 2	one or more of the constraints $n > 0$, $c > 0$, *maxit* > 0 and *eps* > 0 is not satisfied
ifault = 3	the current estimate of the probability is less than -1.
ifault = 4	the required accuracy could not be obtained in *maxit* iterations
ifault = 5	the value returned by the procedure does not satisfy $0 \leqslant RUBEN \leqslant 1$
ifault = 6	*dnsty* is negative
ifault = 9	faults 4 and 5
ifault = 10	faults 4 and 6
ifault = 0	otherwise

Local constant

tol is used to trap the underflow of *dans*. It should be set at a value rather larger than the natural logarithm of the smallest positive real number.

Auxiliary Algorithm

centnorm (x) evaluates the probability that a standard normal variable lies in the interval $[-x, x]$. Our implementation is based on Hill's (1973) algorithm AS 66.

Accuracy and Time

Table 1 records the results obtained when we used AS 155, AS 204 with $\beta = \beta_A$ (AS 204A) and AS 204 with $\beta = \beta_B$ (AS 204B) to evaluate expression (1) with a desired accuracy of 0.0001. Twelve combinations of λ_j, m_j and δ_j^2 were employed. The term $\lambda_j \chi^2(m_j, \delta_j^2)$ is represented in the first column of Table 1 by the triple $\lambda_j, m_j, \delta_j^2$ if $\delta_j^2 \neq 0$ and by the pair λ_j, m_j if $\delta_j^2 = 0$. Six of these combinations were employed in a similar study by Davies (1980) and the first nine are due to Imhof (1961).

TABLE 1
Accuracy and time of AS 155 and AS 204

			Accuracy			*Time*		
Quadratic form	*c*	*Probability*	*AS 155*	*AS 204A*	*AS 204B*	*AS 155*	*AS 204A*	*AS 204B*
Q_1 = 6,1; 3,1; 1,1	1	0.0542	0.009	0.005	0.002	201	0	1
	7	0.4936	0.066	0.017	0.056	172	1	1
	20	0.8760	0.137	0.055	0.135	94	3	2
Q_2 = 6,2; 3,2; 1,2	2	0.0065	0.180	0.018	0.003	25	0	0
	20	0.6002	0.032	0.062	0.101	23	3	2
	60	0.9839	0.002	0.178	0.224	20	8	3
Q_3 = 6,6; 3,4; 1,2	10	0.0027	0.165	0.010	0.012	10	1	1
	50	0.5647	0.037	0.066	0.103	10	7	3
	120	0.9912	0.022	0.300	0.435	8	20	6
Q_4 = 6,2; 3,4; 1,6	10	0.0334	0.025	0.046	0.037	13	2	1
	30	0.5804	0.003	0.123	0.207	12	3	2
	80	0.9913	0.021	0.255	0.253	10	10	4
Q_5 = 7,6,6; 3,2,2	20	0.0061	0.020	0.026	0.009	8	1	1
	100	0.5913	0.022	0.139	0.145	7	4	2
	200	0.9779	0.053	0.281	0.345	7	7	3
Q_6 = 7,1,6; 3,1,2	10	0.0451	0.003	0.019	0.036	116	1	1
	60	0.5924	0.033	0.053	0.079	63	3	2
	150	0.9777	0.087	0.293	0.319	23	6	3
$Q_7 = Q_3 + 2Q_4$	45	0.0109	0.088	0.020	0.059	16	7	3
	120	0.6547	0.000	0.083	0.116	15	28	8
	210	0.9846	0.052	0.212	0.227	14	58	16
$Q_9 = Q_5 + Q_6$	70	0.0437	0.012	0.053	0.043	11	4	3
	160	0.5848	0.012	0.199	0.119	10	8	5
	260	0.9538	0.014	0.228	0.326	10	14	6
$Q_{11} = Q_3 + Q_4 + Q_5 + Q_6$	120	0.0158	0.023	0.019	0.044	27	32	11
	240	0.5736	0.007	0.087	0.127	26	93	30
	400	0.9883	0.122	0.255	0.344	25	187	54
R_1 = 30,1; 1,10	5	0.0154	0.134	0.006	0.043	34	1	1
	25	0.5108	0.033	0.026	0.094*	33	3	3
	100	0.9163	0.001	0.062	0.123*	30	19	12
R_2 = 30,1; 1,20	10	0.0049	0.058	0.016	0.134	23	1	1
	40	0.5732	0.025	0.040	0.714*	21	4	5
	100	0.8965	0.009	0.049	0.481*	21	17	520
R_3 = 30,1; 1,30	20	0.0171	0.016	0.020	DNC*	20	2	756
	50	0.5665	0.001	0.027	DNC*	19	5	756
	100	0.8713	0.010	0.060	DNC*	18	16	756

* Trap 3 suppressed.

Our program was run on a CDC 7600 which carries 14.45 digits (48 bits) in the calculations. Accuracy is measured by the absolute error expressed as a proportion of the desired accuracy where we take the value obtained from AS 155 or AS 204 (β_A) with $acc = eps = 10^{-10}$ to be the true value. CPU timings are measured in milliseconds.

Examining the table we find that all three procedures achieved the desired accuracy but that AS 204 (β_B) did not converge in 500 iterations (fault 9) in R_3. This difficulty and the slow time for R_2 with $c = 100$ may be eliminated by introducing trap 3 which terminates the procedure if the current estimate of the probability (3) is less than minus one. In this context the seven starred entries in the table are terminated within one millisecond. Our arbitrary choice of a critical value for trap 3 may seem to be too strict but R_2 takes 503 milliseconds to achieve a desired accuracy of 0.00001 when $c = 40$ and 650 milliseconds when $c = 100$.

With this modification AS 204B achieves the desired accuracy substantially faster than AS 204A and faster than AS 155 except in Q_7 and Q_{11} when c is large. Thus it seems reasonable to recommend AS 204 with $\beta = \beta_B$ for use in exploratory studies as the accuracy of the final result may be checked by AS 204 with $\beta \leqslant \lambda_{\min}$ or by AS 155.

Now comparing AS 204A with AS 155 we find that AS 204A is faster than AS 155 except in Q_7 and Q_{11} and in Q_3 and Q_9 when c is large. Following Davies (1980) it therefore seems reasonable to recommend AS 204 with $\beta = \beta_A$ rather than AS 155 except when Σm_j is large or $\lambda_{\max}/\lambda_{\min}$ is very large.

Related Algorithms

Kotz, Johnson and Boyd (1967a, p. 837; 1967b) have noted that the coefficients of (2) may be calculated from powers of $Q = I - \beta A^{-1}$ when $b = 0$. In fact they may also be calculated in this way when $b \neq 0$. However the cost of evaluating Q^p accurately when p is large restricts the useful application of this technique to occasions when A is nearly diagonal, b is zero and c is small.

Press (1966, p. 486) has generalized (2) to indefinite quadratic forms. However, the results of Table 2 suggest that his approximation is a very expensive method of achieving a level of

TABLE 2

The accuracy and time of AS 155 and Press for $Q_{12} = Q_3 - Q_5 + 2Q_6 - 2Q_4$

	Probability			Time			
Algorithm:	*AS 155*	*AS 155*	*Press*	*AS 155*	*AS 155*	*Press*	*Press*
Desired accuracy:	$10^{-10}, 10^{-6}$	10^{-4}	$10^{-4}, 10^{-6}$	10^{-6}	10^{-4}	10^{-4}	10^{-6}
$c =$ 240	0.9847959	0.9847948	0.9999516	29	23	45	54
300	0.9952305	0.9952273	0.9999818	30	23	59	70
360	0.9986005	0.9985978	0.9999938	30	24	72	88
420	0.9996114	0.9996094	0.9999981	32	24	87	104
500	0.9999344	0.9999323	0.9999996	32	25	102	126
550	0.9999792	0.9999764	0.9999999	33	26	111	140
600	0.9999935	0.9999909	0.9999999	33	26	117	152

accuracy that could have been obtained more easily by setting

$$\Pr\left[\sum_{j=1}^{n} \lambda_j \chi^2(m_j, \delta_j^2) < c\right] = \begin{cases} 0 & \text{if} \quad c < 0 \\ 1 & \text{if} \quad c > 0 \end{cases}$$

Additional Comment

Our implementation of AS 204 assumes that exp (z) will be set equal to zero if z is large and negative.

Acknowledgement

I am indebted to Grant Hillier of Monash University for initiating this research and to Robert Davies of D.S.I.R. in Wellington and the referee for many useful suggestions.

References

Davies, R. B. (1980) Algorithm AS 155. The distribution of a linear combination of χ^2 random variables. *Appl. Statist.*, **29**, 323–333.
Hill, I. D. (1973) Algorithm AS 66. The normal integral. *Appl. Statist.*, **22**, 424–427.
Imhof, J. P. (1961) Computing the distribution of quadratic forms in normal variables. *Biometrika*, **48**, 419–426.
Kotz, S., Johnson, N. L. and Boyd, D. W. (1967a) Series representations of distributions of quadratic forms in normal variables. I. Central case. *Ann. Math. Statist.*, **38**, 823–837.
——— (1967b) Series representations of distributions of quadratic forms in normal variables. II Noncentral case. *Ann. Math. Statist.*, **38**, 838–848.
Press, S. J. (1966) Linear combinations of non-central chi-square variates. *Ann. Math. Statist.*, **37**, 480–487.
Robbins, H. and Pitman, E. J. G. (1949) Applications of the method of mixtures to quadratic forms in normal variables. *Ann. Math. Statist.*, **20**, 552–560.
Ruben, H. (1962) Probability content of regions under spherical normal distributions. IV The distribution of homogeneous and non-homogeneous quadratic functions of normal variables. *Ann. Math. Statist.*, **33**, 542–570.
Sheil, J. and O'Muircheartaigh, I. (1977) Algorithm AS 106. The distribution of non-negative quadratic forms in normal variables. *Appl. Statist.*, **26**, 92–98.

```
real procedure RUBEN(lambda, mult, delta, n, c, mode, maxit,
     eps, dnsty, ifault);

comment Algorithm AS 204 Appl. Statist. (1984) Vol. 33, No. 3;

value n, c, mode, maxit, eps;
real c, mode, eps, dnsty; integer n, maxit, ifault;
array lambda, delta; integer array mult;

comment RUBEN evaluates the probability that a positive definite
quadratic form in Normal variates is less than a given value;

if n < 1 ∨ c ⩽ 0.0 ∨ maxit < 1 ∨ eps ⩽ 0.0 then
     begin
     RUBEN := −2.0; ifault := 2
     end
else
     begin integer i, k, m;
     real ao, aoinv, z, beta, eps2, hold, hold2, sum, sum1,
          dans, lans, pans, prbty, tol;
     array gamma, theta[1 : n], a, b[1 : maxit];
     tol := −200.0;

     comment preliminaries;

     beta := sum := lambda[1];
     for i := 1 step 1 until n do
          begin
          hold := lambda[i];
          if hold ⩽ 0.0 ∨ mult[i] < 1 ∨ delta[i] < 0.0 then
               begin
               RUBEN := −7.0; ifault := −i;
               goto EXIT
               end;
```

```
        if beta > hold then beta := hold;
        if sum < hold then sum := hold
        end;
beta := if mode > 0.0 then mode × beta
   else 2.0 / (1.0 / beta + 1.0 / sum);
k := 0; sum := 1.0;
sum1 := 0.0;
for i := 1 step 1 until n do
        begin
        hold := beta / lambda[i]; gamma[i] := 1.0 − hold;
        sum := sum × hold ↑ mult[i]; sum1 := sum1 + delta[i];
        k := k + mult[i]; theta[i] := 1.0
        end i;
ao := exp(0.5 × (ln(sum) − sum1));
if ao ⩽ 0.0 then
        begin
        RUBEN := dnsty := 0.0; ifault := 1
        end
else
        begin
        z := c / beta;

        comment evaluate probability and density of chi-squared
        on k degrees of freedom. The constant 0.22579135264473
        is ln(sqrt(pi/2));

        if k = k ÷ 2 × 2 then
                begin
                i := 2; lans := −0.5 × z;
                dans := exp(lans); pans := 1.0 − dans
                end
        else
                begin
                i := 1; lans := −0.5 × (z + ln(z)) − 0.22579135264473;
                dans := exp(lans); pans := centnorm(sqrt(z))
                end;
        k := k − 2;
        for i := i step 2 until k do
                begin
                if lans < tol then
                        begin
                        lans := lans + ln(z / i); dans := exp(lans)
                        end
                else dans := dans × z / i;
                pans := pans − dans
                end i;

        comment evaluate successive terms of expansion;

        prbty := pans; dnsty := dans;
        eps2 := eps / ao; aoinv := 1.0 / ao;
        sum := aoinv − 1.0;
```

```
for m := 1 step 1 until maxit do
     begin
     sum1 := 0.0;
     for i := 1 step 1 until n do
          begin
          hold := theta[i]; theta[i] := hold2 := hold × gamma[i];
          sum1 := sum1 + hold2 × mult[i] +
               m × delta[i] × (hold − hold2)
          end i;
     b[m] := sum1 := 0.5 × sum1;
     for i := m − 1 step −1 until 1 do
          sum1 := sum1 + b[i] × a[m − i];
     a[m] := sum1 := sum1 / m; k := k + 2;
     if lans < tol then
          begin
          lans := lans + ln(z / k); dans := exp(lans)
          end
     else dans := dans × z / k;
     pans := pans − dans; sum := sum − sum1;
     dnsty := dnsty + dans × sum1; sum1 := pans × sum1;
     prbty := prbty + sum1;
     if prbty < −aoinv then
          begin
          RUBEN := −3.0; ifault := 3;
          goto EXIT
          end;
     if abs(pans × sum) < eps2 then
          begin
          if abs(sum1) < eps2 then
               begin
               ifault := 0; goto L
               end
          end
     end m;
ifault := 4;
L: dnsty := ao × dnsty / (beta + beta); prbty := ao × prbty;
if prbty < 0.0 ∨ prbty > 1.0 then ifault := ifault + 5
else if dnsty < 0.0 then ifault := ifault + 6;
RUBEN := prbty
end;
EXIT:
     end RUBEN
```