Algorithm AS 177

Expected Normal Order Statistics (Exact and Approximate)

By J. P. Royston

MRC Clinical Research Centre, Harrow HA1 3UJ, Middx, UK

[Received January 1981. Final revision July 1981]

Keywords: RANKITS; EXPECTED NORMAL SCORES; EXPECTED NORMAL ORDER STATISTICS

Language

Fortran 66

Description and Purpose

The algorithms $NSCOR1$ and $NSCOR2$ calculate the expected values of normal order statistics in exact or approximate form respectively. $NSCOR2$ requires little storage and is fast, and hence is suitable for implementation on small computers or certain programmable calculators (HP-67, etc). This is not recommended for $NSCOR1$. Expected normal order statistics are needed in the calculation of analysis of variance tests of normality, such as W (Shapiro and Wilk, 1965) and W' (Shapiro and Francia, 1972).

In a sample of size n the expected value of the rth largest order statistic is given by

$$E(r,n) = \frac{n!}{(r-1)!\,(n-r)!}\int_{-\infty}^{\infty} x\{1-\Phi(x)\}^{r-1}\{\Phi(x)\}^{n-r}\,\phi(x)\,dx, \qquad (1)$$

where $\phi(x) = 1/\sqrt{(2\pi)}\exp(-\frac{1}{2}x^2)$ and $\Phi(x) = \int_{-\infty}^{x}\phi(z)\,dz$.

Values of $E(r,n)$ accurate to five decimal places were obtained by Harter (1961) using numerical integration, for $n = 2(1)\,100\,(25)\,250\,(50)\,400$. Subroutine $NSCOR1$ uses the same technique as Harter (1961). Rewrite the integrand in (1) as

$$I(r,n,x) = t_0(x)\exp\{\log_e n! - \log_e(n-r)! - \log_e(r-1)! + (r-1)\,t_1(x) + (n-r)\,t_2(x) + t_3(x)\},$$

where

$$t_0(x) = x, \quad t_1(x) = \log_e\{1-\Phi(x)\}, \quad t_2(x) = \log_e\Phi(x), \quad t_3 = -\tfrac{1}{2}\{\log_e(2\pi)+x^2\}.$$

Values of t_0, t_1, t_2 and t_3 are calculated in the range $x = -9{\cdot}0(h)9{\cdot}0$, using the auxiliary subroutine $INIT$, which needs to be called once only. $E(r,n)$ is obtained by summing the values of $I(r,n,x)$ and multiplying the result by h. We found $h = 0{\cdot}025$ sufficiently small. Values of $\Phi(x)$ must be supplied by a suitable algorithm, such as AS 66 (Hill, 1973). Log factorials are obtained from the auxiliary function $ALNFAC$, kindly supplied by Dr I. D. Hill. This is a modification of Pike and Hill's (1966) algorithm.

An approximation to $E(r,n)$ for $n = 2(1)\,50$ with accuracy 0·001 was given in AS 118 (Westcott, 1977). Using a different numerical method, $NSCOR2$ extends the range of n to 2000 and greatly improves the accuracy. Blom (1958) proposed the approximate formula

$$E(r,n) = -\Phi^{-1}\left(\frac{r-\alpha}{n-2\alpha+1}\right)$$

and recommended the compromise value $\alpha = 0{\cdot}375$. Harter (1961) provided values for α as functions of r and n, improving the overall accuracy to about 0·002 for $n \leqslant 400$. Defining

$$P_{r,n} = \Phi\{-E(r,n)\} \quad \text{and} \quad Q_{r,n} = \frac{r-\varepsilon}{n+\gamma},$$

we approximate $P_{r,n}$, the normal upper tail area corresponding to $E(r,n)$, as

 0035–9254/82/31161 $2.00

$$\tilde{P}_{r,n} = Q_{r,n} + \frac{\delta_1}{n} Q_{r,n}^{\lambda} + \frac{\delta_2}{n} Q_{r,n}^{2\lambda} - C_{r,n}.$$

Estimates of ε, γ, δ_1, δ_2 and λ were obtained for $r = 1, 2, 3$ and $r \geqslant 4$, and λ was further approximated as $\lambda = a + b/(r+c)$ for $r \geqslant 4$. A small correction $C_{r,n}$ to $\tilde{P}_{r,n}$ was found necessary for $r \leqslant 7$ and $n \leqslant 20$, and this is supplied by the auxiliary function *CORREC*. The approximation to $E(r,n)$ is thus given by

$$\tilde{E}(r,n) = -\Phi^{-1}(\tilde{P}_{r,n}).$$

Values of the inverse normal probability function Φ^{-1} may be obtained from Algorithm AS 111 (Beasley and Springer, 1977).

Note that both *NSCOR*1 and *NSCOR*2 generate the $[n/2]$ largest rankits; the (symmetrical) smallest rankits are obtained via

$$E(n-r+1,n) = -E(r,n), \quad r = 1, \ldots, [n/2],$$

with $E([n/2]+1,n) = 0$ if n is odd.

Structure

*SUBROUTINE NSCOR*1 (*S*, *N*, *N*2, *WORK*, *IFAULT*)

Formal parameters

S	Real array (*N*2)	output: contains *N*2 largest rankits
N	Integer	input: sample size
*N*2	Integer	input: largest integer less than or equal to $\frac{1}{2}N$
WORK	Real array (4,721)	input: working array, values set by *INIT*
IFAULT	Integer	output: fault indicator, equal to 3 if $N2 \neq N/2$ 2 if $N > 2000$ 1 if $N \leqslant 1$ 0 otherwise

SUBROUTINE INIT (*WORK*)

Formal parameters

WORK	Real array (4,721)	output: working array required by *NSCOR*1

The user must call *INIT* once before the first call of *NSCOR*1.

REAL FUNCTION ALNFAC (*J*) calculates natural log of factorial *J*; it is called from within *NSCOR*1.

*SUBROUTINE NSCOR*2 (*S*, *N*, *N*2, *IFAULT*)

Formal parameters

Identical to *NSCOR*1, except that a working array (*WORK*) is not required.

REAL FUNCTION CORREC (*I*, *N*) is called from within *NSCOR*2.

Failure indications

The fault condition *IFAULT* = 2, occurring if $N > 2000$, still permits the calculation of rankits, but the results cannot be guaranteed to be as accurate as for lower values of N. No calculations are carried out when *IFAULT* = 1 or 3.

Auxiliary algorithms

REAL FUNCTION ALNORM (*X*, *UPPER*) calculates the upper or lower tail area under the normal distribution at *X*, e.g. Algorithm AS 66 (Hill, 1973).

REAL FUNCTION PPND (*P*) calculates the normal equivalent deviate corresponding to *P*, e.g. Algorithm AS 111 (Beasley and Springer, 1977).

Restrictions

*NSCOR*1 and *NSCOR*2 have been validated up to $N = 2000$, but *NSCOR*2 is probably accurate for much larger N. The accuracy of *NSCOR*1 for $N > 2000$ may be improved by reducing the constant h (and increasing *NSTEP*).

Precision

The algorithms were developed on a 48-bit machine (ICL 1903A). *NSCOR*1 requires *DOUBLE PRECISION* on machines of word-length 36 bits or fewer. The following changes should be made to construct a double precision version:

1. *INIT*, *NSCOR*1 and *ALNFAC*: change *REAL* variables and arrays to *DOUBLE PRECISION*, *E* exponents to *D* in DATA statements, and *ALOG* and *EXP* to *DLOG* and *DEXP* respectively.
2. *ALNFAC* becomes a *DOUBLE PRECISION FUNCTION*.

Time and Accuracy

*NSCOR*2 ran about 30 times faster than *NSCOR*1 on the ICL 1903A. The execution time is directly proportional to N for both subroutines.

*NSCOR*1 in *DOUBLE PRECISION* is accurate to at least seven decimal places on a 36-bit machine; *NSCOR*2 is accurate to 0·0001, and usually to five or six decimal places.

References

Beasley, J. D. and Springer, S. G. (1977). Algorithm AS 111. The percentage points of the normal distribution. *Appl. Statist.*, **26**, 118–121.

Harter, H. L. (1961). Expected values of normal order statistics. *Biometrika*, **48**, 151–165.

Hill, I. D. (1973). Algorithm AS 66. The normal integral. *Appl. Statist.*, **22**, 424–427.

Pike, M. C. and Hill, I. D. (1966). Algorithm 291. Logarithm of the gamma function. *Commun. Ass. Comput. Mach.*, **9**, 684.

Shapiro, S. S. and Francia, R. S. (1972). An approximate analysis of variance test for normality. *J. Amer. Statist. Ass.*, **67**, 215–216.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality. *Biometrika*, **52**, 591–611.

Westcott, B. (1977). Algorithm AS 118. Approximate rankits. *Appl. Statist.*, **26**, 362–364.


```
      SUBROUTINE NSCOR1(S, N, N2, WORK, IFAULT)
C
C        ALGORITHM AS 177  APPL. STATIST. (1982) VOL.31, NO.2
C
C        EXACT CALCULATION OF NORMAL SCORES
C
      REAL S(N2), WORK(4, 721)
      REAL ZERO, ONE, C1, D, C, SCOR, AI1, ANI, AN, H, ALNFAC
      DATA ONE /1.0E0/, ZERO /0.0E0/, H /0.025E0/, NSTEP /721/
      IFAULT = 3
      IF (N2 .NE. N / 2) RETURN
      IFAULT = 1
      IF (N .LE. 1) RETURN
      IFAULT = 0
      IF (N .GT. 2000) IFAULT = 2
      AN = N
C
C        CALCULATE NATURAL LOG OF FACTORIAL(N)
C
      C1 = ALNFAC(N)
      D = C1 - ALOG(AN)
C
C        ACCUMULATE ORDINATES FOR CALCULATION OF INTEGRAL FOR RANKITS
C
      DO 20 I = 1, N2
      I1 = I - 1
      NI = N - I
      AI1 = I1
      ANI = NI
      C = C1 - D
      SCOR = ZERO
      DO 10 J = 1, NSTEP
```

```
   10 SCOR = SCOR + EXP(WORK(2, J) + AI1 * WORK(3, J) + ANI * WORK(4, J)
     *  + C) * WORK(1, J)
      S(I) = SCOR * H
      D = D + ALOG((AI1 + ONE) / ANI)
   20 CONTINUE
      RETURN
      END
C
      SUBROUTINE INIT(WORK)
C
C        ALGORITHM AS 177.1  APPL. STATIST. (1982) VOL.31, NO.2
C
      REAL WORK(4, 721)
      REAL XSTART, H, PI2, HALF, XX, ALNORM
      DATA XSTART /-9.0E0/, H /0.025E0/, PI2 /-0.918938533E0/,
     *     HALF /0.5E0/, NSTEP /721/
      XX = XSTART
C
C        SET UP ARRAYS FOR CALCULATION OF INTEGRAL
C
      DO 10 I = 1, NSTEP
      WORK(1, I) = XX
      WORK(2, I) = PI2 - XX * XX * HALF
      WORK(3, I) = ALOG(ALNORM(XX, .TRUE.))
      WORK(4, I) = ALOG(ALNORM(XX, .FALSE.))
      XX = XSTART + FLOAT(I) * H
   10 CONTINUE
      RETURN
      END
C
      REAL FUNCTION ALNFAC(J)
C
C        ALGORITHM AS 177.2  APPL. STATIST. (1982) VOL.31, NO.2
C
C        NATURAL LOGARITHM OF FACTORIAL FOR NON-NEGATIVE ARGUMENT
C
      REAL R(7), ONE, HALF, A0, THREE, FOUR, FOURTN, FORTTY,
     *  FIVFTY, W, Z
      DATA R(1), R(2), R(3), R(4), R(5), R(6), R(7) /0.0E0, 0.0E0,
     *  0.69314718056E0, 1.79175946923E0, 3.17805383035E0,
     *  4.78749174278E0, 6.57925121101E0/
      DATA ONE, HALF, A0, THREE, FOUR, FOURTN, FORTTY, FIVFTY /
     *  1.0E0, 0.5E0, 0.918938533205E0, 3.0E0, 4.0E0, 14.0E0, 420.0E0,
     *  5040.0E0/
      IF (J .GE. 0) GOTO 10
      ALNFAC = ONE
      RETURN
   10 IF (J .GE. 7) GOTO 20
      ALNFAC = R(J + 1)
      RETURN
   20 W = J + 1
      Z = ONE / (W * W)
      ALNFAC = (W - HALF) * ALOG(W) - W + A0 + (((FOUR - THREE * Z)
     *  * Z - FOURTN) * Z + FORTTY) / (FIVFTY * W)
      RETURN
      END
C
      SUBROUTINE NSCOR2(S, N, N2, IFAULT)
C
C        ALGORITHM AS 177.3  APPL. STATIST. (1982) VOL.31, NO.2
C
C        APPROXIMATION FOR RANKITS
C
      REAL S(N2), EPS(4), DL1(4), DL2(4), GAM(4), LAM(4), BB, D, B1, AN,
     *  AI, E1, E2, L1, CORREC, PPND
      DATA EPS(1), EPS(2), EPS(3), EPS(4)
     *  /0.419885E0, 0.450536E0, 0.456936E0, 0.468488E0/,
     *     DL1(1), DL1(2), DL1(3), DL1(4)
     *  /0.112063E0, 0.121770E0, 0.239299E0, 0.215159E0/,
     *     DL2(1), DL2(2), DL2(3), DL2(4)
     *  /0.080122E0, 0.111348E0, -0.211867E0, -0.115049E0/,
```

```
     *      GAM(1), GAM(2), GAM(3), GAM(4)
     *   /0.474798E0, 0.469051E0, 0.208597E0, 0.259784E0/,
     *      LAM(1), LAM(2), LAM(3), LAM(4)
     *   /0.282765E0, 0.304856E0, 0.407708E0, 0.414093E0/,
     *      BB /-0.283833E0/, D /-0.106136E0/, B1 /0.5641896E0/
      IFAULT = 3
      IF (N2 .NE. N / 2) RETURN
      IFAULT = 1
      IF (N .LE. 1) RETURN
      IFAULT = 0
      IF (N .GT. 2000) IFAULT = 2
      S(1) = B1
      IF (N .EQ. 2) RETURN
C
C        CALCULATE NORMAL AREAS FOR 3 LARGEST RANKITS
C
      AN = N
      K = 3
      IF (N2 .LT. K) K = N2
      DO 5 I = 1, K
      AI = I
      E1 = (AI - EPS(I)) / (AN + GAM(I))
      E2 = E1 ** LAM(I)
      S(I) = E1 + E2 * (DL1(I) + E2 * DL2(I)) / AN - CORREC(I, N)
    5 CONTINUE
      IF (N2 .EQ. K) GOTO 20
C
C        CALCULATE NORMAL AREAS FOR REMAINING RANKITS
C
      DO 10 I = 4, N2
      AI = I
      L1 = LAM(4) + BB / (AI + D)
      E1 = (AI - EPS(4)) / (AN + GAM(4))
      E2 = E1 ** L1
      S(I) = E1 + E2 * (DL1(4) + E2 * DL2(4)) / AN - CORREC(I, N)
   10 CONTINUE
C
C        CONVERT NORMAL TAIL AREAS TO NORMAL DEVIATES
C
   20 DO 30 I = 1, N2
   30 S(I) = -PPND(S(I))
      RETURN
      END
C
      REAL FUNCTION CORREC(I, N)
C
C        ALGORITHM AS 177.4  APPL. STATIST. (1982) VOL.31, NO.2
C
C        CALCULATES CORRECTION FOR TAIL AREA OF NORMAL DISTRIBUTION
C        CORRESPONDING TO ITH LARGEST RANKIT IN SAMPLE SIZE N.
C
      REAL C1(7), C2(7), C3(7), AN, MIC, C14
      DATA C1(1), C1(2), C1(3), C1(4), C1(5), C1(6), C1(7)
     *   /9.5E0, 28.7E0, 1.9E0, 0.0E0, -7.0E0, -6.2E0, -1.6E0/,
     *      C2(1), C2(2), C2(3), C2(4), C2(5), C2(6), C2(7)
     *   /-6.195E3, -9.569E3, -6.728E3, -17.614E3, -8.278E3, -3.570E3,
     *    1.075E3/,
     *      C3(1), C3(2), C3(3), C3(4), C3(5), C3(6), C3(7)
     *   /9.338E4, 1.7516E5, 4.1040E5, 2.157E6, 2.376E6, 2.065E6,
     *    2.065E6/,
     *      MIC /1.0E-6/, C14 /1.9E-5/
      CORREC = C14
      IF (I * N .EQ. 4) RETURN
      CORREC = 0.0
      IF (I .LT. 1 .OR. I .GT. 7) RETURN
      IF (I .NE. 4 .AND. N .GT. 20) RETURN
      IF (I .EQ. 4 .AND. N .GT. 40) RETURN
      AN = N
      AN = 1.0 / (AN * AN)
      CORREC = (C1(I) + AN * (C2(I) + AN * C3(I))) * MIC
      RETURN
      END
```