QUANTILES FROM THE MAXIMUM STUDENTIZED RANGE DISTRIBUTION

Daniel Furtado FERREIRA[1]
Clarice Garcia Borges DEMÉTRIO[2]
Bryan Frederick John MANLY[3]
Amauri de Almeida MACHADO[4]

▪ *ABSTRACT: The Copenhaver and Holland (1988) algorithm for calculating quantiles and the distribution function of the maximum of* c *statistics having studentized range distributions of* r *sample means obtained from random a sample of size* n *from normal homocedastic distributions was adapted and implemented in Pascal. The algorithm uses Gauss-Legendre quadrature to obtain the distribution function and uses the secant method to calculate the* 100p% *quantiles from that distribution. The main advantage of the implemented algorithm is that it is not limited to the range of quantiles from 90% to 99%, and it is more accurate than the iterative procedure proposed by Lund and Lund (1983). When* c = 1 *the algorithm supplies the cumulative distribution function and the quantiles of the studentized range distribution, except for the particular case where the number of degrees of freedom is* v=1. *In general, the accuracy of the algorithm decreases as* r *increases, as* v *decreases and as* p *approaches* 1. *Also* r *and* v *have a larger effect on accuracy. Differences among the results from Tukey and maximum studentized range quantiles were shown in a real example. These differences were due to the guarantee of the overall protection for the global confidence coefficient only in the maximum studentized range distribution.*

▪ *KEYWORDS: Studentized range; quantiles*

1 Introduction

The studentized range distribution is applied in some widely used multiple comparisons tests involving sample means from normal homocedastic distributions, a subject discussed in many articles since the 1950s (Tukey, 1952; Keuls, 1952; Hochberg and Tamhane, 1987). However, these methods are available in few statistical packages.

[1]Departamento de Ciências Exatas, Universidade Federal de Lavras – UFLA, CEP 37200-000, Lavras, MG, Brazil. E-mail: *danielff@ufla.br*
[2]Departamento de Ciências Exatas, Escola Superior de Agricultura "Luiz de Queiroz" da Universidade de São Paulo – ESALQ/USP, CEP 13418-900, Piracicaba, SP, Beasil. E-mail: *clarice@carpa.ciagri.usp.br*
[3]Western EcoSystems Technology Inc. of Wyoming, USA – CAPES visiting Professor in Departamento de Ciências Exatas, Escola Superior de Agricultura "Luiz de Queiroz" da Universidade de São Paulo – ESALQ/USP, E-mail: *bmanly@compuserve.com*
[4]Departamento de Matemática Estatística e Informática, Instituto de Física e Matemática, Universidade Federal de Pelotas – UFPEL, Caixa Postal 354, CEP 96010-900, Pelotas, RS, Brasil. E-mail: *amachado@ufpel.edu.br*

One of the algorithms used for obtaining the quantiles from the studentized range is written in Fortran and was proposed by Lund and Lund (1983). This algorithm is inaccurate and limited to values of p between 0.90 and 0.99.

Copenhaver and Holland (1988) proposed a generalization of this algorithm to the distribution of the maximum of c studentized ranges, each obtained from r normal means. This distribution guarantees the overall significance level in the multiple comparison procedure for means coming from, for instance, experiments whose treatments are combinations of two qualitative factors. Although this algorithm has better accuracy and is valid for any value of p, the code in Fortran has not been published, but is available in the following electronic address: http://lib.stat.cmu.edu/general/qprob. Besides, the importance of this distribution in real situations is not well known and its use is limited.

Here in section 2 we present the cumulative distribution function of the maximum of c studentized ranges. In section 3 we present the description of the approach and algorithm proposed by Copenhaver and Holland (1988). In section 4 Pascal algorithms for obtaining the distribution function of the maximum of c studentized ranges obtained from r normal means and the inverse of the distribution function are described, together with information about their accuracy and some auxiliary algorithms. In section 5 a two-way factor analysis with c and r levels is presented to show an application of the proposed methodology.

2 The cumulative probability function of the maximum of c studentized ranges

Suppose that there are $r \times c$ normal populations being sampled, each with variance σ^2. Let $Y_{ij1}, Y_{ij2}, ..., Y_{ijn}$ be the values of a random sample of size n drawn from the ij^{th} normal population, where $i = 1, 2, ..., c$ and $j = 1, 2, ..., r$. Let S^2 be the variance estimator given by:

$$S^2 = \frac{\sum_{i=1}^{c}\sum_{j=1}^{r}\sum_{k=1}^{n}(Y_{ijk} - \overline{Y}_{ij.})^2}{\nu},$$

where $\nu = rc(n - 1)$ is the degrees of freedom, and $\overline{Y}_{ij.} = \sum_{k=1}^{n} Y_{ijk} / n$ is the estimator of the mean for the population ij. Also, for each $i = 1, 2, ..., r$, let $\overline{Y}_{i(1)} \le \overline{Y}_{i(2)} \le ... \le \overline{Y}_{i(r)}$ represent the ordered means, so that the studentized range of the i-th population Q_i is

$$Q_i = \frac{V_i}{S} = \frac{\left(\overline{Y}_{i(r)} - \overline{Y}_{i(1)}\right)\sqrt{n}}{S}. \qquad (2.1)$$

If the maximum of c studentized normal ranges is $Q = Max\{Q_1, Q_2, ..., Q_c\}$, then the distribution function of Q (Copenhaver and Holland, 1988) is given by

$$F(q) = P(Q \le q) = \int_0^{\infty} \left[H\left(q\sqrt{u}\right) \right]^c \frac{\nu^{\frac{\nu}{2}} e^{-\frac{u\nu}{2}} u^{\frac{\nu}{2}-1}}{2^{\frac{\nu}{2}} \Gamma\left(\frac{\nu}{2}\right)} du \tag{2.2}$$

where $H(q)$ is the cumulative distribution function (cdf) of the range (q) of r independent normal variates with mean zero and common variance $\sigma^2 = 1$, which given by:

$$H(q) = r \int_{-\infty}^{\infty} \theta(y) \left[\Phi(y) - \Phi(y-q)\right]^{r-1} dy \tag{2.3}$$

where

$$\theta(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2},$$

and

$$\Phi(y) = \int_{-\infty}^{y} \theta(t) dt .$$

3 The algorithm of Copenhaver and Holland

Since the integral from equation (2.3) does not have algebraic solution, Copenhaver and Holland (1988) proposed an approximation for $H(q)$. Following Hartley (1942) this is given by

$$\begin{aligned} H(w) \cong & \left[2\Phi\left(\tfrac{w}{2}\right) - 1\right]^r + \frac{2r}{\sqrt{2\pi}} \\ & \sum_{i=1}^{k} \left\{ (b_i - a_i)/2 \times \int_{-1}^{1} exp\left\{ -\left[(b_i - a_i)y + b_i + a_i\right]^2 / 8 \right\} \times \right. \\ & \left. \times \left(\Phi\left\{\left[(b_i - a_i)y + b_i + a_i\right]/2\right\} - \Phi\left\{\left[(b_i - a_i)y + b_i + a_i - 2w\right]/2\right\} \right)^{c-1} dy \right\} \end{aligned} \tag{3.1}$$

where $k = \begin{cases} 3 \text{ if } w \le 3 \\ 2 \text{ if } 3 < w < 8 \end{cases}$; $a_1 = w/2$; $a_{i+1} = b_i = \left[(k-1)(w/2) + 8i\right]/k, i < k$;

and $b_k = 8$.

This expression is a satisfactory approach using the number of intervals (k) suggested by Copenhaver and Holland (1988) for w>3. Otherwise, for small w there is a need to use more intervals.

The integral (2.2) also does not have an algebraic solution and must be evaluated numerically. Copenhaver and Holland (1988) proposed splitting the interval $(0, \infty)$ into several sub-intervals of size L. This method starts with a transformation $k(z) = Lz + 2aL + L$ to modify the integration limits to the interval (-1, 1) and, later a Gauss-Legendre quadrature is applied. The final expression is given by

$$F(q)=\sum_{a=0}^{\infty}\left\{\int_{-1}^{1}\left[H\left(q\sqrt{\frac{k(z)}{2}}\right)\right]^{c}\frac{Lv^{v/2}e^{-k(z)v/4}k(z)^{v/2-1}}{2^{v}\,\Gamma\left(\frac{v}{2}\right)}\,dz\right\}. \qquad (3.2)$$

The sub-intervals length L should be chosen depending on the number of degrees of freedom, then the same authors suggested $L = 1$ for ν from 2 to 100, L = 1/2 for ν from 101 to 800, L = 1/4 for ν from 801 to 5000, and L = 1/8 for ν from 5001 to 25000. For values of ν larger than 25,000 equation (2.2) could be approximated by $[H(q)]^c$. In the implementation of the algorithm we have verified that for ν = 2 the best choice of L is 0.968.

In multiple comparisons procedures we need to obtain quantile values $q_p(r,c,v)=F^{-1}(q)$ and the secant method offers the possibility to calculate them. This calculation is iteratively repeated until a pre-fixed criterion is reached. However this process depends on initial values, which are given by the following (Lund and Lund, 1983) equations:

$$\begin{cases}\hat{q}_p(r,v,c=1)=\sqrt{2}\;t+\left(0{,}8843\;t-0{,}2368\;t^2\right)\;ln(r-1), & v\le 120\\ \hat{q}_p(r,v,c=1)=\sqrt{2}\;t+\left(0{,}8843\sqrt{2}t-0{,}2368t^2-\frac{1{,}214t}{v}+\frac{1{,}208t^2}{v}\right)ln(r-1), & v>\end{cases} \qquad (3.3)$$

where t is the quantile from the t-Student distribution for p. An approach attributed to Peiser (1943) was used to obtain the quantiles $t_p(v)$ (Lund and Lund,1983). Note that the algorithms supply relatively accurate values of the cumulative probabilities p for different values of q, v and r with quantiles q, but are limited to the range $0.90 \le p \le 0.99$.

4 Description of the algorithms and accuracy

The algorithms for obtaining the cumulative distribution function, the quantiles of the maximum of c studentized ranges and the auxiliary functions were written in Pascal and implemented in Delphi 7 (Cantù, 2003) using the Math unit for numeric routines. We can, however, use another power routine to substitute the standard function "*Power(a, b)* = a^b", as for instance "exp(b*ln(a))".

The three auxiliary functions are conversions to Pascal of published algorithms in Fortran, done in this paper, and the three functions for obtaining $H(q)$, $F(q)$ and $F^{-1}(q)$ are described below. The Pascal implementation of those functions is in the Appendix.

a) "Function apnorm(Z: extended): extended"

This auxiliary function for obtaining $F(q)$ returns the values from the cumulative probability function of the standard normal, $\Phi(Z)$, for any real argument Z. The algorithm is based on an adaptation of the algorithm to the error function. This routine is extremely accurate having error of less than 1.0×10^{-15} (based on a FORTRAN algorithm available in http://lib.stat.cmu.edu/apstat/66).

b) "Function apnorminv (p: extended): extended"
This function returns the inverse distribution function Z from the standard normal distribution.The original algorithm of Wichura (1988, AS241) was converted to Pascal. The accuracy of this algorithm is claimed to be of less than 1.0×10^{-16} (based on a FORTRAN algorithm available in http://lib.stat.cmu.edu/apstat/241).

c) "function lngammaf(z: extended; var ier: longint): extended"
This function returns the natural logarithmic of the incomplete gamma function based on the approach of Lanczos (1964). An accuracy to about 14 significant digits is obtained, except in the neighborhood of 1 and 2 (based on a FORTRAN algorithm available in http://lib.stat.cmu.edu/apstat/245).

d) "function PRange_v_inf(w,r: Extended;var ifault:Longint): Extended"
This function returns the value of the cumulative probability function *H(w)*, according to equation (2.3), but using the approximate probability of equation (3.1).

e) "function PRange(q,r,v,ci: Extended;var ifault:Longint): Extended"
This function returns the value of the cumulative probability function *F(q)* using equations (2.2) or (3.2). If $c = 1$, then the cumulative probability function of the studentized range is returned. If $c = 2$, then the distribution of the studentized maximum modulus is obtained (Dunn and Massey, 1965).

f) "Function qrange(p,r,v,ci: extended; var ifault: longint): extended"
This function returns the value of the quantile $q_p(r,c,\nu)$. To obtain quantiles from the maximum normal range distribution ($v = \infty$), argument ν can be passed to the routine with a value greater than 25000.

Other auxiliary functions will not be discussed. The accuracy of the algorithm was checked and we obtained basically the same quantiles for all situations as Copenhaver and Holland (1988). Also, we obtained the same quantiles for the studentized range ($c = 1$) as Gleason (1999). We verified that the algorithm is not applicable for $\nu = 1$ as pointed out by Gleason (1999). Adaptations in the algorithm were therefore made, using a value of L different from that suggested by Copenhaver and Holland (1988), to obtain quantiles values $q_p(r,c,\nu)$ with $\nu = 1$. The results obtained for $p = 0.95$ and $\nu = 1$ show that errors occur in the first decimal place. For the case of the studentized range distribution Ferreira (2001) presents an algorithm for the situation of $\nu = 1$ that has maximum relative absolute error of 3.5%. The author's method is based on quantiles from the t-Student distribution that is quite different from the proposed algorithm. The use of this algorithm for other values of p in the special case of $\nu = 1$ is therefore not advisable. In general, the

accuracy of the algorithm decreases as r increases, v decreases and p approaches 1. Besides that, r and v are most important for the accuracy.

5 Numerical example

To illustrate and to demonstrate the importance the application of a multiple comparison test in experiments with two factors with at least one considered fixed, an experiment of common bean varieties comparison was carried out in seven locations of the State of Rio Grande do Sul, Brazil. In each location an experiment was run with a complete block design with 20 bean varieties and 3 blocks. The common bean yield was measured as tons per hectare. The following statistical model was adopted:

$$Y_{ijk} = \mu + \alpha_i + \gamma_{i(j)} + \tau_k + \delta_{ik} + \varepsilon_{ijk} \qquad (5.1)$$

where Y_{ijk} is the observed value associated with the k-th variety, i-th location and j-th block; μ is an overall level parameter; α_i is the i-th location effect; $\gamma_{i(j)}$ is the effect of the j-th block within the i-th location; τ_k is the k-th variety effect; δ_{ik} is the interaction effect between the i-th location and the k-th variety; and ε_{ijk} is the residual effect assumed to be normal, identical and independently distributed with zero mean and common variance σ^2. The variety and the constant effect were considered fixed, and the other effects were considered random.

Table 1 shows a summary of the analysis of variance using the location x variety mean square to test the hypothesis of equal variety means. There is a significant ($P < 0.05$) variety main effect and a significant ($P < 0.01$) interaction effect.

Table 1 - Summary of the two-way model (5.1) analysis of variance, considering variety main effect as fixed and other factors effects as random

Source of variation	DF	MS	F	P<F
Location (L)	6	8.4013×107	157.13	0.000
Block(Location)	14	452398.0	2.57	0.002
Variety (V)	19	442363.6	1.71	0.043
L x V	114	258029.4	1.47	0.006
Residual	266	175767.5		
Total	419	-		

Since the interaction effects are significant, we compare the variety effects within each location. We found a significant ($P<0.05$) F test for variety effects only in the location of Veranópolis. Thus, we applied Tukey's (1949) multiple comparison test for the variety effect, considering $c = 1$ (conventional analysis) and considering $c = 7$, to preserve the overall confidence nominal level of 95%. The least significant difference (Δ_l) for the first case is:

$$\Delta_1 = q_p\left(r, c=1, \nu\right)\sqrt{\frac{MSR}{b}} = 5.133\sqrt{58589.17} = 1242.5$$

where MSR is the residual mean square, p =0.95, r = 20 (varieties), b = 3 (blocks) and ν = 114. For the second case, which preserves the overall confidence level p = 0.95, the least significant difference is:

$$\Delta_2 = q_p\left(r, c=7, \nu\right)\sqrt{\frac{MSR}{b}} = 5.949\sqrt{58589.17} = 1440.0$$

The final result of the two multiple comparison procedures is presented in Table 2. Note that there is a difference between the two results: for c = 1 there were four different groups in the classification (A, B, C and D) of the varieties means and for c = 7, only three. Therefore, for instance, variety "SM_8915" is significantly (P<0.05) different from variety "Carioca" using the conventional Tukey test, but not significant different when the test used the maximum studentized range quantile. This result shows that the maximum studentized range distribution controls the overall nominal confidence level and is therefore more conservative than the Tukey test.

Table 2 - Multiple comparison results using the Tukey test with c = 1 (studentized range) and with c = 7 (maximum studentized range), where the latter controls the overall nominal 95% confidence level

Varieties	Veranópolis	c = 1*	*c = 7**
TB9501	5716.667	A	A
TB9401	4995.333	AB	AB
TB9713	4895.000	ABC	AB
LM_95102	4839.333	ABC	AB
CARIOCA	4673.000	ABC	ABC
FT_NOBRE	4617.667	ABCD	ABC
SM_89153	4584.333	ABCD	ABC
PEROLA	4584.333	ABCD	ABC
LM922041	4484.333	ABCD	ABC
SM_9708	4429.000	BCD	ABC
MT952020	4395.667	BCD	ABC
M_89852	4206.333	BCD	BC
MACOTACO	4084.667	BCD	BC
SM_8990	4062.333	BCD	BC
TB9707	4040.333	BCD	BC
TB9608	3863.000	BCD	BC
TB9502	3729.333	CD	BC
SM_9704	3718.667	CD	BC
LM_93204	3596.333	D	BC
SM_8915	3385.333	D	C

* Means followed by the same letters are not significantly (P<0.05) different.

Acknowledges

We would like to acknowledge CNPq and CAPES financial support.

FERREIRA, D. F.; DEMÉTRIO, C. G. B.; MANLY, B. F. J.; MACHADO, A. de A. Quantis da distribuição do máximo da amplitude estudentizada. *Rev. Mat. Est*., São Paulo, v.25, n.1, p.117-135, 2007.

- RESUMO: O algoritmo de Copenhaver e Holland (1988), que calcula quantis e a função de distribuição do máximo de *c* amplitudes "estudentizadas" de *r* médias provenientes de amostras aleatórias de tamanho *n* de uma distribuição normal homocedástica, foi adaptado e implementado em Pascal. Na implementação foi utilizada a quadratura Gauss-Legendre para obter a função de distribuição $F(q) = P(Q \leq q) = p$ e o método da secante para calcular os quantis $q_p(r, v, c)$ dessa distribuição. A principal vantagem do algoritmo implementado é a sua maior abrangência em relação ao procedimento iterativo proposto por Lund e Lund (1983) que, além de menos preciso, é limitado aos quantis entre 90% e 99%. Quando $c = 1$ o algoritmo fornece as probabilidades e quantis da amplitude "estudentizada" com elevada precisão, exceto para o caso particular do número de graus de liberdade $\nu = 1$. Em geral, a acurácia do algoritmo decresce à medida que *r* aumenta, *v* diminui e *p* aproxima-se de 1, sendo que *r* e *v* têm uma maior importância para a acurácia. Diferenças entre os resultados do teste Tukey e do máximo da amplitude estudentizada foram mostradas em um exemplo real. Essas diferenças foram devidas à garantia para o coeficiente de confiança global que ocorre somente quando se considera a distribuição do máximo das amplitudes estudentizadas.
- PALAVRAS-CHAVE: Amplitude *estudentizada*; quantis; algoritmo.

References

CANTÙ, M. *Mastering Delphi 7***.** Alameda: Sybex, 2003. 1011p.

COPENHAVER, M. D.; HOLLAND, B. S. Computation of the distribution of the maximum studentized range statistic with application to multiple significance testing of simple effects. *J. Stat. Comput. Simulat***.**, New York, v.30, p.1-15, 1988.

DUNN, O. J.; MASSEY Jr, F. J. Estimation of multiple contrast using t-distributions. *J. Am. Stat. Assoc.,* New York, v.60, p.573-583, 1965.

FERREIRA, D. F. Quantis aproximados da distribuição da amplitude *estudentizada***,** obtidos por meio da distribuição t de Student. *Rev.Mat.Estat.,* São Paulo, v.19, p.199-215, 2001.

GLEASON, J. R. An accurate, non-iterative approximation for studentized range quantiles. *Comput. Stat. Data Anal.***,** Amsterdam, v.31, p.147-158, 1999.

HARTLEY, H. O. The range in random samples. *Biometrika*, London, v.32, p.334-348, 1942.

HOCHBERG, Y.; TAMHANE, A. C. *Multiple comparison procedures*. New York: Wiley & Sons, 1987. 450p.

KEULS, M. The use of the “Studentized Range”: in - connection with an analysis of variance. *Euphytica*, Wageningen, v.1, p.112-122, 1952.

LANCZOS, C. A precision approximation of the gamma function. *J. SIAM Numer. Anal., B*, Philadelphia, v.1, p.86-96, 1964.

LUND, R. E.; LUND, J. R. Algorithm AS 190 - probabilities and upper quantiles for the studentized range. *J. R. Stat. Soc. Ser* C.: *Appl. Stat*., London, v.32, p.204-210, 1983. (and correction, *J. R. Stat. Soc. Ser* C.: *Appl. Stat.***,** London, v.34, p.104, 1985).

PEISER, A. M. Asymptotic formulas for significance levels of certain distributions. *Ann. Math. Stat*., Ann Arbor, v.14, p. 56-62, 1943. (and correction *Ann. Math. Stat*., v.20, p.128-129, 1944).

TUKEY, J. W. Comparing individual means in the analysis of variance. *Biometrics*, Washington, v.5, p.99-114, 1949.

WICHURA, M. J. The percentage points of the normal distribution. *J. R. Stat. Soc. Ser* C.: *Appl. Stat*., London, v.37, n.3, p.477-484, 1988.

Received in11.05.2007.

Approved after revised in 20.07.2007.

Appendix

The following Delphi unit contains all of the necessary functions for obtaining the quantiles and of the cumulative probabilities of the maximum of *c* studentized range statistic from *r* normal means.

```
unit cop_holl;

interface

Function PRange_v_inf(w,r: Extended;var ifault:Longint): Extended;
Function PRange(q,r,v,ci: Extended;var ifault:Longint): Extended;
Function qrange(p,r,v,ci: extended; Var ifault: longint): extended;
Function apnorm(Z: extended): extended;
Function apnorminv (P: extended): extended;
function lngammaf(z: extended; var ier: longint): extended;

implementation
 uses math;

Const
 Root : Array [1..20] of Extended =
 (0.993128599185095, 0.963971927277914, 0.912234428251326, 0.839116971822219,
 0.746331906460151,0.636053680726515, 0.510867001950827, 0.37370608871542,
 0.227785851141645, 0.076526521133 4973,-0.076526521133 4973, -0.227785851141645,
-0.37370608871542, -0.510867001950827, -0.636053680726515,-0.746331906460151,
-0.839116971822219, -0.912234428251326, -0.963971927277914, -0.993128599185095);

Weight : Array [1..20] of Extended =
(0.0176140071391521, 0.0406014298003869, 0.0626720483341091, 0.0832767415767048,
 0.10193011981724,0.118194531961518, 0.131688638449177, 0.142096109318382,
 0.149172986472604, 0.152753387130726,0.152753387130726, 0.149172986472604,
 0.142096109318382, 0.131688638449177, 0.118194531961518,0.10193011981724,
 0.0832767415767048, 0.0626720483341091, 0.0406014298003869, 0.0176140071391521);

Function apnorm(Z: extended): extended;
//      normal probabilities – accuracy of 1.e-15.
//      Z = number of standard deviation from mean
//      P, Q = Left and right probabilities from Z.   P + Q = 1.
//      PDF = the probability density.
//      Based upon algorithm 5666 for the error function, from:
//      Hart, J.F. et al, 'Computer Approximations', Wiley 1968
//
//      Delphi version: 04/11/2004

//
Const
```

```
P0=220.2068679123761e0;      P1=221.2135961699311e0;      P2=112.0792914978709e0;
P3=33.91286607838300e0;      P4=6.373962203531650e0;   P5=0.7003830644436881e0;
P6=0.3526249659989109e-01;     Q0=440.4137358247522e0;Q1=793.8265125199484e0;
Q2=637.3336333788311e0;Q3=296.5642487796737e0;          Q4=86.78073220294608e0;
Q5=16.06417757920695e0;      Q6=1.755667163182642e0;Q7=0.8838834764831844e-1;
CUTOFF=7.071e0; ROOT2PI=2.506628274631001e0;

var
  zabs,expntl  : extended;
  p, q, pdf    : extended;
begin
    zabs:= abs(z);
    if (zabs > 37.0e0) then  //|z| > 37.
    begin
      pdf := 0.0e0;
      if (z > 0.0e0) then
      begin
        p := 1.0e0; q := 0.0e0
      end else
      begin
        p := 0.0e0;    q := 1.0e0
      end
    end else
    begin //|z| <= 37
      expntl := exp(-0.5e0*zabs*zabs);
      pdf := expntl/root2pi;
    // |z| < cutoff = 10/sqrt(2).
      if (zabs < cutoff) then
        p := expntl*((((((p6*zabs + p5)*zabs + p4)*zabs + p3)*zabs +
           p2)*zabs + p1)*zabs + p0)/(((((((q7*zabs + q6)*zabs +
           q5)*zabs + q4)*zabs + q3)*zabs + q2)*zabs + q1)*zabs + q0)
    //       |z| >= cutoff
      else
        p := pdf/(zabs + 1.0e0/(zabs + 2.0e0/(zabs + 3.0e0/(zabs + 4.0e0/
             (zabs + 0.65e0)))));
      if (z <0.0e0) then q := 1.0e0 - p
      else
      begin
        q := p;      p := 1.0e0 - q
      end;
    end;//z<=37
    apnorm:=p;
end;
(* ----------------------------  NORMAL  ----------------------------  *)

Function apnorminv (P: extended): extended;
//      original name: PPND16
```

```
//      ALGORITHM AS241  APPL. STATIST. (1988) VOL. 37, NO. 3
//      Produces the normal deviate Z corresponding to a given lower
//      tail area of P; Z is accurate to about 1 part in 10**16.
//      The hash sums below are the sums of the mantissas of the
//      coefficients. They are included for use in checking
//      transcription.
//      Delphi version

Const
      ZERO = 0.e0; ONE = 1.e0; HALF = 0.5e0;   SPLIT1 = 0.425e0; SPLIT2 = 5.e0;
      CONST1 = 0.180625e0; CONST2 = 1.6e0;
//       Coefficients for P close to 0.5
      A0 = 3.3871328727963666080e0;       A1 = 1.3314166789178437745e+2;
      A2 = 1.9715909503065514427e+3;      A3 = 1.3731693765509461125e+4;
      A4 = 4.5921953931549871457e+4;      A5 = 6.7265770927008700853e+4;
      A6 = 3.3430575583588128105e+4;      A7 = 2.5090809287301226727e+3;
      B1 = 4.2313330701600911252e+1;      B2 = 6.8718700749205790830e+2;
      B3 = 5.3941960214247511077e+3;      B4 = 2.1213794301586595867e+4;
      B5 = 3.9307895800092710610e+4;      B6 = 2.8729085735721942674e+4;
      B7 = 5.2264952788528545610e+3;
//      Coefficients for P not close to 0, 0.5 or 1.
      C0 = 1.42343711074968357734e0;      C1 = 4.63033784615654529590e0;
      C2 = 5.76949722146069140550e0;      C3 = 3.64784832476320460504e0;
      C4 = 1.27045825245236838258e0;      C5 = 2.41780725177450611770e-1;
      C6 = 2.27238449892691845833e-2;     C7 = 7.74545014278341407640e-4;
      D1 = 2.05319162663775882187e0;      D2 = 1.67638483018380384940e0;
      D3 = 6.89767334985100004550e-1;     D4 = 1.48103976427480074590e-1;
      D5 = 1.51986665636164571966e-2;     D6 = 5.47593808499534494600e-4;
      D7 = 1.05075007164441684324e-9;
//       Coefficients for P near 0 or 1.
      E0 = 6.65790464350110377720e0;      E1 = 5.46378491116411436990e0;
      E2 = 1.78482653991729133580e0;      E3 = 2.96560571828504891230e-1;
      E4 = 2.65321895265761230930e-2;     E5 = 1.24266094738807843860e-3;
      E6 = 2.71155556874348757815e-5;     E7 = 2.01033439929228813265e-7;
      F1 = 5.99832206555887937690e-1;     F2 = 1.36929880922735805310e-1;
      F3 = 1.48753612908506148525e-2;     F4 = 7.86869131145613259100e-4;
      F5 = 1.84631831751005468180e-5;     F6 = 1.42151175831644588870e-7;
      F7 = 2.04426310338993978564e-15;
var
   ppnd,q,r  : extended;
   ifault    : longint;
begin
  ifault := 0;  q := p - half;
  if (abs(q) <= split1) then
  begin
    r := const1 - q * q;
    ppnd := q * (((((((a7 * r + a6) * r + a5) * r + a4) * r + a3)
```

```
          * r + a2) * r + a1) * r + a0) / (((((((b7 * r + b6) * r + b5) * r + b4) * r + b3)
          * r + b2) * r + b1) * r + one)
  end else
  begin
    if (q < zero) then
      r := p
    else
      r := one - p;
    if (r <= zero) then
    begin
      ifault := 1;       ppnd := zero
    end else
    begin
      r := sqrt(-ln(r));
      if (r <= split2) then
      begin
        r := r - const2;
        ppnd := (((((((c7 * r + c6) * r + c5) * r + c4) * r + c3)
              * r + c2) * r + c1) * r + c0) / (((((((d7 * r + d6) * r + d5) * r + d4) * r + d3)
              * r + d2) * r + d1) * r + one)
      end
      else
      begin
        r := r - split2;
        ppnd := (((((((e7 * r + e6) * r + e5) * r + e4) * r + e3) * r + e2) * r + e1) * r + e0) /
              (((((((f7 * r + f6) * r + f5) * r + f4) * r + f3) * r + f2) * r + f1) * r + one)
      end;
      if (q < zero) then ppnd := - ppnd;
    end;
  end;
  apnorminv:=ppnd
end;//norminv


function lngammaf(z: extended; var ier: longint): extended;
//       Uses Lanczos' approximation for ln(gamma) and z > 0. Reference:
//        Lanczos, C. 'A precision approximation of the gamma
//        function', J. SIAM Numer. Anal., B, 1, 86-96, 1964.
//        Accuracy: About 14 significant digits except for small regions
//       in the vicinity of 1 and 2.
//       Programmer: Alan Miller -  CSIRO Division of Mathematics & Statistics
//       Latest revision - 17 April 1988
//      Delphi version: Date: 04/11/2004

var
  a                                  : array [1..9] of extended;
  lnsqrt2pi,tmp,lngamma              : extended;
```

```
  j                                    : longint;
Begin
  a[1]:=0.9999999999995183e0;          a[2]:=676.5203681218835e0;                  a[3]:=-
1259.139216722289e0;
  a[4]:=771.3234287757674e0;                                    a[5]:=-176.6150291498386e0;
a[6]:=12.50734324009056e0;
  a[7]:=-0.1385710331296526e0; a[8]:=0.9934937113930748e-05;
  a[9]:=0.1659470187408462e-06;  lnsqrt2pi:=0.9189385332046727e0;
  if (z <=0.e0) then
  begin
    ier := 1; exit;
  end;
  ier := 0;
  lngamma := 0.e0;
  tmp := z + 7.e0;
  for j := 9 downto 2 do
  begin
    lngamma := lngamma + a[j]/tmp;
    tmp := tmp - 1.e0
  end;
  lngamma := lngamma + a[1];
  lngamma := ln(lngamma) + lnsqrt2pi - (z+6.5e0) +
  (z-0.5e0)*ln(z+6.5e0);
  lngammaf := lngamma;
end;
(* ------------------------  Ln_da_Gama ------------------------  *)


Function PRange_v_inf(w,r: extended;var ifault:longint): extended;
var
  k,ai,bi,soma,ii            : extended;
  i                          : longint;
function fint(w,yii,aii,bii,r: extended): extended;
var
  yyi      : extended;
begin
  yyi:=(bii-aii)*yii+bii+aii;
  fint:=Power(exp(1),-yyi*yyi/8)*
  power((apnorm(yyi/2)-apnorm((yyi-2*w)/2)),r-1);
end;
function GaussLegendreQuadrature(w,yii,aii,bii,r: extended;
                          const a, b:  double;const n:  longint;
                          var ifault : longint):  extended;

var
   c,d,sum          : extended;
   j,jfirst,jlast   : longint;
```

```
begin
   jfirst := 1;
   jlast  := n;
   c := (b - a) / 2.0;
   d := (b + a) / 2.0;
   sum := 0.0;
   for j := jfirst to jlast do
   begin
     if   root[j] = 0.0
     then sum := sum + weight[j]*fint(w,d,aii,bii,r)
     else sum := sum + weight[j]*(fint(w,root[j]*c + d,aii,bii,r));
   end;
   gausslegendrequadrature := c * sum
 end {gausslegendrequadrature};

begin
  if w<=0 then
  begin
    PRange_v_inf:=0;
    exit
  end;
  if w<=3 then k:=3.0
  else k:=2.0;
  //inicializando valor de ai p/ i=1
  ai:=w/2;
  ii:=1;
  bi:=((k-ii)*(w/2)+8*ii)/k;
  soma:=0;
  for i:=1 to round(k) do //loop para soma externa de i = 1 a k
  begin
    ii:=i;
    soma:=soma+((bi-ai)/2)*
    GaussLegendreQuadrature(w,0.0,ai,bi,r,-1.0, +1.0,20,ifault);
    ai:=bi;
    if i+1=round(k) then bi:=8
    else bi:=((k-ii-1)*(w/2)+8*(ii+1))/k;
  end;
  soma:=soma*2*r/sqrt(2*Pi);
  soma:=soma+power(exp(1),r*ln(2*apnorm(w/2)-1));
   PRange_v_inf:=soma
end;

function PRange(q,r,v,ci: Extended;var ifault:Longint): Extended;
var
  precis,a,auxprob,L,probinic    : Extended;
  found                          : Boolean;
```

```
function f26(q,za,aii,c,r,v: extended): extended;
Var
  yyi,aux,aux1      : extended;
begin
  yyi:=(za*L+2*aii*L+L);
  aux1:=PRange_v_inf(sqrt(yyi/2)*q,r,ifault);
  if aux1=0 then aux1:=1e-37;
  aux:=c*ln(aux1)+ln(L)+(v/2)*ln(v)+(-yyi*v/4)+(v/2-1)*ln(yyi)-
  (v*ln(2)+lngammaf(v/2,ifault));
  if abs(aux)>=1e30 then f26:=0
  else f26:=exp(aux);
end;

function gausslegdquad(q,yii,aii,r,ci: extended;
                    const a, b:  double;const n:  longint;
                    var ifault : longint):  extended;
var
   cmm,d,sum          : extended;
   j,jfirst,jlast   : longint;
begin
    jfirst := 1;
    jlast  := n;
    cmm := (b - a) / 2.0;
    d  := (b + a) / 2.0;
    sum := 0.0;
    for j := jfirst to jlast do
    begin
      if   root[j] = 0.0
      then sum := sum + weight[j]*f26(q,d,aii,ci,r,v)
      else sum := sum + weight[j]*(f26(q,root[j]*cmm + d,aii,ci,r,v));
    end;
    gausslegdquad := cmm * sum
 end {gausslegendrequadrature};
begin
 precis:=1e-10;
 ifault:=0;
 if v=1 then
 begin
   if r<10 then l:=1+1/(2*r+3)
   else if r<=100 then l:=1.0844+(1.119-1.0844)/90*(r-10)
   else l:=1.119+1/r;
 end else
 if (v=2) then l:=0.968 else
 if v<=100 then l:=1
 else if v<=800 then l:=1/2
 else if v<=5000 then l:=1/4
 else l:=1/8; //if v>25000 use (H(q))^c as approximation to the probability
```

```
if v>25000 then
begin
    PRange:=Power(PRange_v_inf(q,r,ifault),ci);
    exit
end
else auxprob:=0;
found:=false; a:=0;  probinic:=0;
while not found do
 begin
   auxprob:=auxprob+GaussLegdQuad(q,0,a,r,ci,-1.0,+1.0,20,ifault);
   if abs(auxprob-probinic)/auxprob<=precis then found:=true
   else probinic:=auxprob;
   a:=a+1;
 end;
 PRange:=auxprob;
end;

(*****************************************************************)
(* algorithm AS 190.2 Appl. Stat. (1983) vol. 32 no.2            *)
(* Calculate a initial percentile P from studentized range dist. with v DF  *)
(* and r samples, and cumulative probabilities: P [0.80..0.995]  *)
(* uses normal inverse functions                                 *)
(*****************************************************************)
Function qtrngo(p,v,r: extended; Var ifault: longint):extended;
  Var
    q,t,Vmax,half,one,four,c1,c2,c3,c4,c5 : extended;

  Procedure inic_val;
  Begin
    vmax:=120.0;    half:=0.5;      one:=1.0;       four:=4.0;      c1:=0.8843;       c2:=0.2368;
c3:=1.214;
    c4:=1.208;   c5:=1.4142
  End;
Begin
  inic_val;
  t:= apnorminv(half+half*p);
  if (v < vmax) then t:=t+(t*t*t+t)/v/four;
  q:=c1-c2*t;
  if (v < vmax) then q:=q-c3/v+c4*t/v;
  qtrngo:=t*(q*ln(r-one)+c5)
end;

(*****************************************************************
***)
(* Adapted from Algorithm AS 190.1 Appl. Stat. (1983) vol. 32 no.2  *)
(* approximate the percentile P from studentized range dist. with v DF  *)
(*and r samples, and cumulative probabilities: P [0.0..1.00]    *)
```

```
(* uses functions: normal inverse, normal pdf, prange and qtrngo   *)
function qrange(p,r,v,ci: extended; var ifault: longint): extended;
  var
    jmax, pcut, one, two, five : extended;
    j,q1,q2,p1,p2              : extended;
    aux,e1,e2                  : extended;
  procedure valores_inic;
  begin
    jmax:=28;   pcut:=1e-8;     one:=1.0;     two:=2.0;     five:=5.0;
  end;
begin
    valores_inic;
    (* verifying initial values                                  *)
    ifault:=0;
    if (v<one) or (r<two) then ifault:=1;
    if ifault<>0 then
    begin
      qrange:=0;
      exit;
    end else
    begin
      (* obtaining initial values                                 *)
      q1:=qtrngo(p,v,r,ifault);
      if ifault<>0 then
      begin
        if ifault<>0 then ifault:=9;
        qrange:=0;  exit;
      end;
      repeat
        p1:=PRange(q1,r,v,ci,Ifault);
        if p1>p then q1:=q1-0.4;
        if q1<0 then q1:=0.1;
      until p1<p;
      if ifault<>0 then
      begin
        if ifault<>0 then ifault:=9;
        qrange:=0;   exit;
      end;
      aux:=q1;
      if abs(P1-P)<pcut then
      begin
        if ifault<>0 then ifault:=9;
        qrange:=q1;  exit;
      end;
      q2:=q1+0.5;
      repeat
        p2:=PRange(q2,r,v,ci,ifault);
```

```
      if p2<p then q2:=q2+0.4;
      if q2<0 then q2:=1;
    until p2>p;
    if q2<q1 then q2:=q1+0.01;
    if ifault<>0 then
    begin
      if ifault<>0 then ifault:=9;
      qrange:=0;
      exit;
    end;
(* Refining the procedure                          *)
    j:=2;
    while j<=jmax do
    begin
      p2:=PRange(q2,r,v,ci,ifault);
      if ifault<>0 then
      begin
        if ifault<>0 then ifault:=9;
        j:=jmax+1;
      end else
      begin
        e1:=p1-p;
        e2:=p2-p;
        if e2-e1<>0 then aux:=(e2*q1-e1*q2)/(e2-e1);
        if abs(e1)< abs(e2) then
        begin
          if (abs(p1-p) <pcut*five) then
          begin
            if ifault<>0 then ifault:=9;
            j:=jmax+2
          end;
          q1:=aux;
          p1:=PRange(q1,r,v,ci,ifault);
        end else
        begin
          q1:=q2;        p1:=p2;       q2:=aux;
        end;
      end;
      j:=j+1;
    end;
    qrange:=aux
  end
end;
end. //unit
```