

number of pairs of observations, $df = N - 2$, and $t = r \times \text{sqr}t(df/(1.0 - r \uparrow 2))$ (cf. e.g. [5]).

For reasonably small df , Student's cosine formula is used [3, 4]:

$$ttest = 1.0 - \text{coef} \int_0^\theta \cos^{df-1} \theta \, d\theta$$

where $\theta = \arctan(t/\text{sqr}t(df))$ and

$$\text{coef} = (df-1)/(df-2) \times (df-3)/(df-4) \dots \begin{cases} (\frac{3}{2}) \times (2/\pi) & \text{for odd } df, \\ (\frac{5}{4}) \times (\frac{3}{2}) \times (\frac{1}{2}) & \text{for even } df. \end{cases}$$

Integrated in series, this gives results which appear to be correct to very nearly the full single precision accuracy of the machine (in terms of the number of digits after the decimal point, not necessarily significant digits).

An approximation due to R. A. Fisher [1] gives results accurate to within $\pm 3 \times 10^{-7}$ when *maxn* has been set at 30. The tradeoff on time is also optimal at about this point. The **real procedure** *gauss* computes the area under the left-hand portion of the normal curve. Algorithm 209 [6] may be used for this purpose.

Thanks to the referee for many helpful suggestions, most of which have been incorporated, and to David F. Foster, who wrote an early version of part of the program.

REFERENCES:

1. FISHER, R. A. *Metron* 5 (1925), 109-112.
2. —. *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh, 1965.
3. GOSSET, W. S. (Student). The probable error of a mean. *Biometrika* 6 (1908), 1.
4. —. New tables for testing the significance of observations. *Metron* 5 (1925), 105.
5. GUILFORD, J. P. *Fundamental Statistics in Psychology and Education*. McGraw-Hill, New York, 1956, pp. 219-221.
6. IBBETSON, D. Algorithm 209, *Gauss. Comm. ACM*, 6 (Oct. 1963), 616.

```
begin
  if df < 1 then go to error;
  if x = 0 then ttest := 1.0 else
  begin real t;
    t := abs(x);
    if df < maxn then
    begin integer i, nh; real cth, sth, cthsq, xi, coef, z;
      z := t/sqr(df);
      cth := 1.0/sqr(z↑2+1.0);
      sth := z × cth;
      cthsq := cth↑2;
      nh := (df-1) ÷ 2;
      if df = 2 × (df÷2) then
      begin
        t := sth;
        if nh = 0 then go to g;
        cth := cthsq; xi := 1.0;
        coef := 0.5 × sth
      end else
      begin
        t := 0.6366197724 × arctan(z);
        comment 0.6366197723675813430755351... = 2/π;
        if nh = 0 then go to g;
        xi := 0; coef := 0.6366197724 × sth
      end
    end
  end;
  for i := 1 step 1 until nh do
  begin
    t := t + coef × cth; cth := cth × cthsq;
    xi := xi + 2.0;
    coef := coef × xi/(xi+1.0)
```

```
end;
g: t := 1.0 - t
end else
  if t > 6.0 then t := 0 else
  if df < 106 then
  begin real f, t2, t4, t6, t8, t10, t12, t14, t16, t18;
    f := df; t2 := t × t; t4 := t2 × t2; t6 := t4 × t2;
    t8 := t6 × t2; t10 := t8 × t2; t12 := t10 × t2;
    t14 := t12 × t2; t16 := t14 × t2; t18 := t16 × t2;
    comment 0.3989422804014326779399461... = 1/sqr(2 × π);
    t := 2.0 × (gauss(-t) + t × 0.3989422804 × exp(-0.5 × t2) ×
      ((t2+1.0)/(4.0 × f) + (3.0 × t6 - 7.0 × t4 - 5.0 × t2 - 3.0)/
      (96.0 × f × f) + (t10 - 11.0 × t8 + 14.0 × t6 + 6.0 × t4 - 3.0 × t2 -
      15.0)/(384.0 × f↑3) + (15.0 × t14 - 375.0 × t12 + 2225.0 × t10 -
      2141.0 × t8 - 939.0 × t6 - 213.0 × t4 - 915.0 × t2 + 945.0)/
      (92160.0 × f↑4) + (3.0 × t18 - 133.0 × t16 + 1764.0 × t14 -
      7516.0 × t12 + 5994.0 × t10 + 2490.0 × t8 + 1140.0 × t6 + 180.0 ×
      t4 + 5355.0 × t2 + 17955.0)/(368640.0 × f↑5)))
    end else t := 2.0 × gauss(-t);
    ttest := if t < 0 then 0 else t
  end
end ttest
```

ALGORITHM 322

F-DISTRIBUTION [S14]

EGON DORRER (Recd. 25 Jan. 1967, 3 July 1967, and 17 Oct. 1967)

Institut für Photogrammetrie und Kartographie, Technische Hochschule München, W. Germany; now: Department of Surveying Engineering, University of New Brunswick, Fredericton, N.B., Canada

KEY WORDS AND PHRASES: Fisher's *F*-distribution, Student's *t*-distribution

CR CATEGORIES: 5.5

```
real procedure Fisher(m, n, x);
  value m, n, x; integer m, n; real x;
  comment Fisher's F-distribution with m and n degrees of freedom. Computation of the probability
```

$$Pr(F < x) = \frac{\Gamma\left(\frac{m+n}{2}\right)}{\Gamma\left(\frac{m}{2}\right) \cdot \Gamma\left(\frac{n}{2}\right)} \cdot \int_0^w \frac{\xi^{m/2-1}}{(\xi+1)^{(m+n)/2}} d\xi,$$

where $w = (m/n)x$ and $F = (\sum_{i=1}^m x_i^2/m)/(\sum_{i=1}^n y_i^2/n)$. The solution results recursively from the basic integrals

$$Fisher(1,1,x) = 2 \cdot \arctan \sqrt{w}/\pi, \quad Fisher(1,2,x) = (w/(w+1))^{\frac{1}{2}},$$

$$Fisher(2,1,x) = 1 - 1/(w+1)^{\frac{1}{2}}, \quad Fisher(2,2,x) = w/(w+1).$$

π is introduced by $0.3183098862 = 1/\pi$. By calling *Fisher* (1, *n*, *t*↑2), Student's *t*-distribution will be obtained;

```
begin integer a, b, i, j; real w, y, z, d, p;
  a := 2 × (m÷2) - m + 2; b := 2 × (n÷2) - n + 2;
  w := x × m/n; z := 1/(1+w);
  if a = 1 then
  begin
    if b = 1 then
    begin
      p := sqrt(w); y := 0.3183098862;
      d := y × z/p; p := 2 × y × arctan(p)
    end else
    begin
      p := sqrt(w × z); d := 0.5 × p × z/w
    end
  end else
```

```

if  $b = 1$  then
  begin
     $p := \text{sqrt}(z)$ ;  $d := 0.5 \times z \times p$ ;  $p := 1 - p$ 
  end else
  begin
     $d := z \times z$ ;  $p := w \times z$ 
  end;
   $y := 2 \times w/z$ ;
  for  $j := b + 2$  step 2 until  $n$  do
    begin
       $d := (1 + a/(j-2)) \times d \times z$ ;
       $p := \text{if } a = 1 \text{ then } p + d \times y/(j-1) \text{ else } (p+w) \times z$ 
    end j;
     $y := w \times z$ ;  $z := 2/z$ ;  $b := n - 2$ ;
    for  $i := a + 2$  step 2 until  $m$  do
      begin
         $j := i + b$ ;  $d := y \times d \times j/(i-2)$ ;  $p := p - z \times d/j$ 
      end i;
       $\text{Fisher} := p$ 
    end Fisher

```

ALGORITHM 323
GENERATION OF PERMUTATIONS IN
LEXICOGRAPHIC ORDER [G6]

R. J. ORD-SMITH (Recd. 27 Apr. 1967 and 26 July 1967)
Computing Laboratory, University of Bradford, Bradford,
Yorkshire, England

KEY WORDS AND PHRASES: permutations, lexicographic
order, lexicographic generation, permutation generation
CR CATEGORIES: 5.39

Author's Remark. Lexicographic generation involves more than the minimum of $n!$ transpositions for generation of the complete set of $n!$ permutations of n objects. The actual number of transpositions required can be shown to tend asymptotically to $(\cosh 1) n! \doteq 1.53n!$ However, lexicographic generation can be described by an algorithm requiring very simple book-keeping. The author is indebted to Professor H. F. Trotter for suggesting an improvement to an original algorithm, which now results in a process more than twice as fast as the previously fastest lexicographic Algorithm 202 [Comm. ACM 6 (Sept. 1963), 517]. Tabulated results below show *BESTLEX* to be only 9.3 percent slower than the transposition Algorithm 115 [Comm. ACM 5 (Aug. 1962), 434] when $n = 8$.

The usual practice is adopted of using a nonlocal Boolean variable called *first* which may be assigned the value *true* to initialize generation. On procedure call this is set *false* and remains so until it is again set *true* when complete generation of permutations has been achieved. Table I gives results obtained for *BESTLEX*. The times given in seconds are for an I.C.T. 1905 computer. t_n is the time for complete generation of $n!$ permutations. r_n has the usual definition $r_n = t_n/(n \cdot t_{n-1})$.

TABLE I

| Algorithm | t_n | t_8 | r_8 | Number of transpositions |
|----------------|-------|-------|-------|--------------------------|
| <i>BESTLEX</i> | 6 | 47 | 0.98 | $\rightarrow 1.53n!$ |
| 202 | 12.4 | 100 | 1.00 | ? |
| 115 | 5.6 | 43 | 0.98 | $n!$ |

```

procedure BESTLEX ( $x, n$ ); value  $n$ ; integer  $n$ ; array  $x$ ;
begin own integer array  $q[2:n]$ ; integer  $k, m$ ; real  $t$ ;
comment own dynamic arrays are not often implemented. The
upper bound will then have to be given explicitly;
if first then
  begin first := false;
    for  $m := 2$  step 1 until  $n$  do  $q[m] := 1$ 
  end of initialization process;
  if  $q[2] = 1$  then
    begin  $q[2] := 2$ ;
       $t := x[1]$ ;  $x[1] := x[2]$ ;  $x[2] := t$ ;
    go to finish
  end;
  for  $k := 2$  step 1 until  $n$  do
    if  $q[k] = k$  then  $q[k] := 1$  else go to trstart;
  first := true;  $k := n$ ; go to trinit;
  trstart:  $m := q[k]$ ;  $t := x[m]$ ;  $x[m] := x[k]$ ;  $x[k] := t$ ;
     $q[k] := m + 1$ ;  $k := k - 1$ ;
  trinit:  $m := 1$ ;
  transpose:  $t := x[m]$ ;  $x[m] := x[k]$ ;  $x[k] := t$ ;
     $m := m + 1$ ;  $k := k - 1$ ;
    if  $m < k$  then go to transpose;
  finish:
end of procedure BESTLEX

```

ALGORITHM 324
MAXFLOW [H]

G. BAYER (Recd. 31 July 1967)
Technische Hochschule, Braunschweig, Germany

KEY WORDS AND PHRASES: network, linear programming,
maximum flow
CR CATEGORIES: 5.41

procedure *maxflow* ($from, to, cap, flow, v, n, mflow, source, sink,$
 inf, eps);

```

  value  $v, n, source, sink, inf$ ;
  integer  $v, n, source, sink$ ; real  $inf, eps, mflow$ ;
  integer array  $from, to$ ; array  $cap, flow$ ;
comment The nodes of the network are numbered from 1 to  $sn$ .
It is not necessary but reasonable that each number represent a
node. The data of the network are given by arrays  $from, to, cap$ 
in the following manner. There is a maximum possible flow of
 $cap[i]$ , nonnegative, leading from  $from[i]$  to  $to[i]$ ,  $i = 1, \dots, v$ .
Compute the maximum flow  $mflow$  from  $source$  to  $sink$ ,
( $source$  and  $sink$  given by their node numbers).  $inf$  represents
the greatest positive real number within machine capacity.
 $flow[i]$  gives the actual flow from  $from[i]$  to  $to[i]$ . Flows abso-
lutely less than  $eps$  are considered to be zero. Literature: G.
Hadley, Linear Programming, Addison-Wesley, Reading (Mass.)
and London, 1962, pp. 337-344.

```

```

  Multiple solutions are left out of account;
begin integer  $l, j, k, r, lk, ek, u, s$ ; real  $gjk, d$ ;
  integer array  $low, up, klist, labj[1:n], ind[1:v]$ ; real array
   $labf[1:n]$ ;
comment Note structure of data lists in  $up$  and  $low$ ;
   $l := 1$ ;
  for  $j := 1$  step 1 until  $n$  do
    begin  $low[j] := l$ ;
      for  $r := 1$  step 1 until  $v$  do
        begin if  $from[r] = j$  then
          begin  $ind[l] := r$ ;
             $flow[l] := cap[l]$ ;  $l := l + 1$ 
          end
        end
    end

```

```

    end;
    up[j] := l - 1
end;
mflow := 0.0;
lab;;
comment Prepare lists for new labeling;
for j := 1 step 1 until n do
begin labj[j] := klist[j] := 0;
      labf[j] := 0.0
end;
labf[source] := inf;
comment labeling;
j := source; lk := ek := 0;
path:
u := up[j];
for s := low[j] step 1 until u do
begin l := ind[s];
      k := to[l]; gjk := flow[l];
      if labj[k] ≠ 0 ∨ abs(gjk) < eps
      then go to end;
      labj[k] := j;
      labf[k] := if gjk < labf[j] then gjk else labf[j];
      if k = sink then go to reached;
      lk := lk + 1; klist[lk] := k;
end;
ek := ek + 1; j := klist[ek];
if j ≠ 0 then go to path else go to max;
comment sink is labeled, find path and possible
flow, reduce excess capacities along path;
reached:
j := sink; d := labf[j]; mflow := mflow + d;
look: k := labj[j]; u := up[k];
for s := low[k] step 1 until u do
begin l := ind[s];
      if to[l] = j then flow[l] := flow[l] - d
end;
u := up[j];
for s := low[j] step 1 until u do
begin l := ind[s];
      if to[l] = k then flow[l] := flow[l] + d
end;
j := k; if j ≠ source then go to look;
go to lab;
max;; comment maximal flow found;
for l := 1 step 1 until v do
flow[l] := cap[l] - flow[l]
end
end

```

ALGORITHM 325

ADJUSTMENT OF THE INVERSE OF A SYMMETRIC MATRIX WHEN TWO SYMMETRIC ELEMENTS ARE CHANGED [F1]

GERHARD ZIELKE (Recd. 24 Aug. 1967)

Institut für Numerische Mathematik der Martin Luther
Universität Halle-Wittenberg, German Democratic
Republic

KEY WORDS AND PHRASES: symmetric matrix, matrix inverse, matrix perturbation, matrix modification

CR CATEGORIES: 5.14

procedure *INVSYM 2* (*n, i, j, c, a, b*);

value *n, i, j, c*; integer *n, i, j*; real *c*; array *a, b*;

comment *INVSYM 2* computes the inverse $A^{-1} = a$ of a non-singular symmetric n th order matrix $A = B + c(e_i e_j' + e_j e_i')$ which arises from a symmetric matrix B by a change c in two elements B_{ij} and $B_{ji} = B_{ij}$ ($i \neq j$). The inverse matrix $B^{-1} = b$ is assumed to be known. The calculation with the new formula

$$a = b - \frac{c}{d} [b_{.i}(h_1 b_{.j} + h_2 b_{.i}) + b_{.j}(h_3 b_{.i} + h_1 b_{.i})]$$

where

$$h_1 = 1 + c b_{ij}, \quad h_2 = -c b_{jj}, \quad h_3 = -c b_{ii}, \quad d = h_1^2 - h_2 h_3$$

requires $n^2 + O(n)$ multiplications, therefore only about the same number of operations as if the well-known Sherman-Morrison formula for a change in one element (see Algorithm 51 [*Comm. ACM* 4 (Apr. 1961), 180]) is used. In these equations e_i denotes the i th column and e_i' the i th row of the unit matrix, $b_{.i} = b e_i$ denotes the i th column and $b_{.i} = e_i' b$ the i th row of the matrix b ;

begin integer *k, l*; real *h1, h2, h3, d*;

array *r, s*[1:*n*];

h1 := 1 + *c* × *b*[*i, j*]; *h2* := -*c* × *b*[*j, j*];

h3 := -*c* × *b*[*i, i*]; *d* := *h1* ↑ 2 - *h2* × *h3*; *d* := *c*/*d*;

h1 := *h1* × *d*; *h2* := *h2* × *d*; *h3* := *h3* × *d*;

for *k* := 1 step 1 until *n* do

begin

r[*k*] := *h1* × *b*[*j, k*] + *h2* × *b*[*i, k*];

s[*k, l*] := *h3* × *b*[*j, k*] + *h1* × *b*[*i, k*]

end;

for *k* := 1 step 1 until *n* do

for *l* := 1 step 1 until *k* do

a[*k, l*] := *a*[*l, k*] := *b*[*k, l*] - *b*[*k, i*] × *r*[*l*] - *b*[*k, j*] × *s*[*l*]

end *INVSYM 2*

MODIFIED SHARE CLASSIFICATIONS

[Designations follow algorithm titles.]

| | | | | | |
|----|---|----|--|----|--|
| A1 | Real Arithmetic, Number Theory | D4 | Differentiation | G7 | Subset Generators and Classifications |
| A2 | Complex Arithmetic | E1 | Interpolation | H | Operations Research, Graph Structures |
| B1 | Trig and Inverse Trig Functions | E2 | Curve and Surface Fitting | I5 | Input—Composite |
| B2 | Hyperbolic Functions | E3 | Smoothing | J6 | Plotting |
| B3 | Exponential and Logarithmic Functions | E4 | Minimizing or Maximizing a Function | K2 | Relocation |
| B4 | Roots and Powers | F1 | Matrix Operations, Including Inversion | M1 | Sorting |
| C1 | Operations on Polynomials and Power Series | F2 | Eigenvalues and Eigenvectors of Matrices | M2 | Data Conversion and Scaling |
| C2 | Zeros of Polynomials | F3 | Determinants | O2 | Simulation of Computing Structure |
| C5 | Zeros of One or More Transcendental Equations | F4 | Simultaneous Linear Equations | S | Approximation of Special Functions . . . |
| C6 | Summation of Series, Convergence Acceleration | F5 | Orthogonalization | | Functions are Classified S01 to S22, Following |
| D1 | Quadrature | G1 | Simple Calculations on Statistical Data | | Fletcher-Miller-Rosenhead, Index of Math. |
| D2 | Ordinary Differential Equations | G2 | Correlation and Regression Analysis | | Tables |
| D3 | Partial Differential Equations | G5 | Random Number Generators | Z | All Others |
| | | G6 | Permutations and Combinations | | |