```
REM   Contingency test for N by M Chi-square with p-value result
REM
REM      Implementation in True BASIC where all arithmetic is IEEE 32-bit floating point.

REM   Copyright 1981, 2001, 2002, 2013 by Colin James III All Rights Reserved
REM   Permission for re-use with inclusion of the above notice is hereby granted.

DECLARE FUNCTION GammLn, Gammq

PRINT

INPUT PROMPT "Number of columns ": columns
INPUT PROMPT "Number of rows    ": rows

DIM table( 0, 0)
MAT REDIM table( rows, columns)

DIM row( 0)
MAT REDIM row( rows)
MAT row    = 0.00000000001

DIM column( 0)
MAT REDIM column( columns)
MAT column = 0.00000000001

FOR row_index = 1 TO rows
    FOR column_index = 1 TO columns
        LET input = RND * 10 + 25
        REM READ input
        LET table( row_index, column_index) = input
        LET table( row_index, column_index) = input
    NEXT column_index
NEXT row_index

FOR row_index = 1 TO rows
    FOR column_index = 1 TO columns
        LET row( row_index)       = row( row_index)       + table( row_index, column_index)
        LET column( column_index) = column( column_index) + table( row_index, column_index)
```

```
    NEXT column_index
    LET chi2_multiplier = chi2_multiplier + row( row_index)
NEXT row_index

FOR row_index = 1 to rows
    FOR column_index = 1 to columns
        LET chi2_part = chi2_part + ( ( table( row_index, column_index) ^ 2) / ( row( row_index) *
column( column_index)))
    NEXT column_index
NEXT row_index

LET chi2_part = chi2_part - 1
LET chi2 = chi2_multiplier * chi2_part

LET df = ( rows - 1) * ( columns - 1)

PRINT
PRINT "chi2 = "; chi2; " df ="; df
PRINT
PRINT "P <= "; Gammq( 0.5 * df, 0.5 * chi2)      ! Q( chi2|df) = Gammq( v/2, chi2/2) p 215 FORTRAN

! Volatility index [VI] (a meaningless statistic) - 72 values
!     For testing of N by M as: 36 x 2; 24 x 3; 18 x 4; 12 x 6; 9 x 8
!
DATA 54.920
DATA 31.180,23.990,24.670,23.250,23.590,24.430,26.010,25.620,24.160,23.150
DATA 20.860,20.900,21.380,22.250,22.930,22.950,22.680,24.350,22.510,21.680
DATA 22.220,21.930,21.640,22.570,23.220,24.760,25.120,26.410,27.200,26.700
DATA 26.510,26.730,26.400,29.720,28.950,28.410,29.220,26.670,25.330,25.320
DATA 24.790,25.100,25.280,23.610,24.740,24.170,24.410,24.400,23.360,24.830
DATA 24.280,23.570,25.170,24.010,22.720,23.130,24.060,25.380,27.080,27.240
DATA 26.690,26.590,25.640,25.780,25.840,26.590,27.170
DATA 26.610,26.530,26.290,28.080

REM END Contingency NxMchi2

REM pass degrees of freedom and Chi-square value to Gammq
```

```
FUNCTION GammLn( xx)
    REM From the FORTRAN portion of Numerical Recipes, 2nd ed, p 207 ff;
    REM Sprott's Companion (in Qbasic), p 93 ff; and Numerical Recipes in True BASIC (printed 2/92).
    REM     Note that the two language versions differ but produce the same p-value result.
    REM     Because the FORTRAN has higher precision, those constants are used here.
    REM
    DATA 76.18009172947146, -86.50532032941677
    DATA 24.01409824083091, -1.231739572450155
    DATA 0.1208650973866179, -0.5395239384953
    DATA 2.5066282746310005       ! stp

    DIM cof( 6)
    RESTORE
    FOR i = 1 TO 6
        READ cof( i)
    NEXT i
    READ stp

    LET x = xx - 1.0
    LET y = x
    LET tmp = x + 5.5
    LET tmp = ( x + 0.5) * LOG( tmp) - tmp
    LET ser = 1.000000000190015
    FOR j = 1 TO 6
        LET y = y + 1
        LET ser = ser + cof( j) / y
    NEXT j
    LET GammLn = tmp + LOG( stp * ser)
END FUNCTION
REM FUNCTION GammLn( xx)

SUB gser( gamser, a, x, gLn)
    LET itmax = 100
    LET eps = 0.0000003
    LET gLn = GammLn( a)
    IF x <= 0.0 THEN
       IF x < 0.0 THEN
          PRINT "Abnormal exit: x < 0 in gser"
```

```
          EXIT SUB
       END IF
       LET gamser = 0.0
       EXIT SUB
    END IF
    LET ap = a
    LET sum = 1.0 / a
    LET del = sum
    FOR n = 1 TO itmax
        LET ap = ap + 1.0
        LET del = del * x / ap
        LET sum = sum + del
        IF ABS( del) < ABS( sum) * eps THEN
           EXIT FOR
        END IF
    NEXT n
    IF ABS( del) >= ABS( sum) * eps THEN
       PRINT "A too large, Itmax too small in gser"
       EXIT SUB
    END IF
    LET gamser = sum * EXP( -x + a * LOG( x) - gLn)
END SUB
REM SUB gser( gamser, a, x, gLn)

SUB gcf( gammcf, a, x, gln)
    LET itmax = 100
    LET eps = 0.0000003
    LET gLn = GammLn( a)
    LET gold = 0.0
    LET a0 = 1.0
    LET a1 = x
    LET b0 = 0.0
    LET b1 = 1.0
    LET fac = 1.0
    FOR n = 1 TO itmax
        LET ana = an - a
        LET a0 = ( a1 + a0 * ana) * fac
        LET b0 = ( b1 + b0 * ana) * fac
```

```
        LET anf = an * fac
        LET a1 = x * a0 + anf * a1
        LET b1 = x * b0 + anf * b1
        IF a1 <> 0.0 THEN
           LET fac = 1.0 / a1
           LET g = b1 * fac
           IF ABS( ( g - gold) / g) < eps THEN
              LET gammcf = EXP( -x + a * LOG( x) - gLn) * g
              EXIT SUB
           END IF
           LET gold = g
        END IF
    NEXT n
    PRINT "A too large, Itmax too small"
END SUB
REM SUB gcf( gammcf, a, x, gln)

FUNCTION Gammq( a, x)
    IF ( x < 0.0) OR ( a <= 0.0) THEN
       PRINT "Bad arguments in Gammq"
       EXIT FUNCTION
    END IF
    IF x < a + 1.0 THEN
       CALL gser( gamser, a, x, gLn)
       LET Gammq = 1.0 - gamser
    ELSE
       CALL gcf( gammcf, a, x, gLn)
       LET Gammq = gammcf
    END IF
END FUNCTION
REM FUNCTION Gammq( a, x)

END
```