F77 Program 6

```
c       Fortran 77 program to do a simulation of the Weibull distribution
c       of the simplest censoring model -
c                                       x = t if t < constant
c                                       x = constant if t >= constant
c       (Where constant = mean of something, etc)
c       This program may be adapted for censored models as well as the
c       simplest censoring model.
c       ************************************************************************
c               By Derek Dhammaloka FDX3 - 9th Jan. 1991
c       ************************************************************************
c               Define the following variables
c
c               cdf is the cumulative density function of the probability
c               function and is between 0 and 1. The function urand will
c               generate the random numbers between 0 and 1. It has 1
c               parameter iy, the seed to initialise the generator.
c               t is the remission time in arbitary units
c               sumln is the sum of the logs to the base e
c               sumpower is the sum of the powers
c               sumpowerln is the sum of product of (t**k) and ln t
c               spln2 is the same as sumpowerln, but has (ln pt)**2
c               kold is the old value of kappa in the iteration loop
c               knew is the new value of kappa to be entered by the
c               user, but is changed in the iteration loop until it is
c               almost equal to kold.
c               kdiff is the difference between the new and old
c               values of kappa
c               ipp,ipk and ikk are the elements of the info. matrix
c               rho is the rate to be entered by the user
c               tol is the tolerance value
c               loop is used in loop counters
c               n is the no. of individuals to be entered by the user
c               d is the no. of uncensored individuals
c               c is the constant censor time
c               const is the minimum time for censoring to occur. It is
c               to be entered by the user.
c               i is the indicator variable (1 if censored, 0 otherwise)
c               x is equal to t if t is less than C, C otherwise
c               iy is the seed to be entered by the user.
c       ************************************************************************
      integer i(5000)
      real cdf(5000),t(5000),c(5000),x(5000)
      real sumln,sumpower,sumpowerln,spln2,kold,knew,kdiff,const
      real rho,tol,ipp,ipk,ikk
      integer loop,n,d,iy
      tol=0.000005
c     ************************************************************************
c               Input the no. of individuals
c               Also the index (kappa) and the rate (rho)
c     ************************************************************************
      print*,'How many individuals'
      read*,n
```

```
c
c           Set the no. of failures to the no. of observations
c
      d=n
      print*, Seed for the Weibull distribution'
      read*,iy
      print*,'Enter the index parameter'
      read*,knew
      print*,'Enter the rate parameter'
      read*,rho
      print*,'Enter the minimum time for censoring to occur'
      read*,const
c     ***************************************************************
c           Simulate the Weibull distribution
c           using the two parameters to obtain the remission times.
c           However, the censor times are constant.
c     ***************************************************************
      do 20 loop=1,n
           cdf(loop)=urand(iy)
           t(loop)=(-log(1-cdf(loop))/(rho**knew)**(1/knew)
           c(loop)=const
c
c          Decide whether the censored individual is to be
c          eliminated using the censoring rules
c
           if(t(loop).ge.c(loop)) then
                                  i(loop)=1
                                  d=d-1
                                  x(loop)=c(loop)
           endif
           if(t(loop).lt.c(loop)) then
                                  i(loop)=0
                                  x(loop)=t(loop)
           endif
20    continue
c     ***************************************************************
c           Print the headings
c     ***************************************************************
      print*
      print*,'For the simplest censored model'
      print*
      print*,'Simulation of the Weibull distribution with '
      print*,'Index = ',knew,' and rate = ',rho
      print*
      print*,'Minimum time for censoring to occur = ',const
      print*
      write(*,25)
25    format(t3,'T',t10,'C',t40,'I',t55,'x')
c     ***************************************************************
c           Output the remission and censor times
c           Also the indicator variable
c     ***************************************************************
      do 30 loop=1,n
           write(*,40)t(loop),c(loop),i(loop),x(loop)
40         format(f7.3,t8,f7.3,t37,i7.0,t50,f7.3,t68)
30    continue
```

```
      print*
c                Update the sumln statistic over uncensored observations
      do 35 loop=1,n
                if(i(loop).eq.0) then
                        sumln=sumln+(log(x(loop))
                endif
35    continue
c                Print statistics
      print*,'Sum of logs to the base e = ',sumln
      print*,'No. of failures = ',d
      print*
c                Print headings
      write(*,45)
45    format(t9,'kappa',t20,'rho')
c                Iteration loop
50    kold=knew
c                Update the statistics of sumpower,sumpowerln over
c                all observations
      do 60 loop=1,n
                sumpower=sumpower+(x(loop)**kold)
                sumpowerln=sumpowerln+((x(loop)**kold)*log(x(loop)))
60    continue
      knew=1/((sumpowerln/sumpower)-(sumln/d))
      kdiff=knew-kold
c     Use the new value of kappa to give the new rho value
      rho=(d/sumpower)**(1/knew)
c                Obtain sumpowerln2 using this new rho value
      do 70 loop=1,n
c                Update the statistics of sumpower2 over all obs.
                   spln2=spln2+(sumpower*(log(rho**x(loop)))**2)
70    continue
c     Use the new value of rho to obtain the elements of the
c     information matrix. ipk will have to appear in 2 lines, as it
c     is difficult to get this whole expression in 66 characters.
      ipp=(knew*d/(rho**2))+((knew*(knew-1)*sumpower*(rho**(knew-2))))
      ipk=-((d/rho)-((rho**(knew-1)*(1+(knew*log(rho)))+sumpower)))
      ipk=ipk+(knew*(rho**(knew-1))*sumpowerln)
      ikk=-((-d/(knew*knew))-((rho**knew)*spln2))
      write(*,80)knew,rho
80    format(f12.5,f12.5)
c     ***************************************************************
c                If the old and new values of kappa do not agree to a
c                certain no. of decimal places, zero the sums (not sumln)
c                and carry on with the iteration.
c     ***************************************************************
      if(abs(kdiff).gt.tol) then
                        sumpower=0
                        sumpowerln=0
                        spln2=0
                        goto 50
      endif
```

```
c     ***************************************************************************
c              Print the elements of the information matrix
c     ***************************************************************************
      print*
      print*, Elements of the information matrix -
      print*, 'ipp =  ',ipp
      print*, 'ipk =  ',ipk
      print*, 'ikk =  ',ikk
      stop
      end

      real function urand(iy)
      integer iy
c     ***************************************************************************
c              Urand is a uniform random number generator based on
c              theory and suggestions given by kNUTH (1969). The
c              integer iy should be initialised to an arbitary integer
c              prior to the first call to urand. The calling program
c              should not alter the value of iy between subsequent
c              calls to urand. Values of urand will be returned in the
c              interval (0,1).
c     ***************************************************************************
c              Reference - Problem solving with Fortran 77
c                          Brian D.Hahn 1987
c     ***************************************************************************
      integer ia,ic,itwo,m2,m,mic
      double precision halfm
      real s
      data m2/0/,itwo/2/

c     If first entry, compute machine integer word length
      if(m2.eq.0) then
                  m=1
10    if(m.gt.m2) then
                  m2=m
                  m=itwo*m2
                  goto 10
      endif
      halfm=m2

c     Compute multiplier and increment for linear congruential method
      ia=8*int(halfm*atan(1.d0)/8.d0)+5
      ic=2*int(halfm*(0.5d0-sqrt(3.d0)/6.d0))+1
      mic=(m2-ic)+m2

c     s is the scale factor for converting to floating point
      s=0.5/halfm
      endif

c     Compute next random number
      iy=iy*ia

c     The following statement is for computers which do not allow
c     integer overflow on addition
      if(iy.gt.mic) iy=(iy-m2)-m2
      iy=iy+ic
```

```
c     The following statement is for computers where the word length
c     is greater than for multiplication
      if(iy/2.gt.m2) iy=(iy-m2)-m2
c
c     The following statement is for computers where integer overflow
c     affects sign bit
      if(iy.lt.0) iy=(iy+m2)+m2
      urand=float(iy)*s
      return
      end
```

Output from F77 Program 6

```
How many individuals
75
Seed for the Weibull distribution
2
Enter the index parameter
2.5
Enter the rate parameter
1.5
Enter the minimum time for censoring to occur
1.25

For the simplest censored model

Simulation of the Weibull distribution with
 Index =      .2500000E+01 and rate =      .1500000E+01

Minimum time for censoring to occur =      .1250000E+01

   T       C                                    I                 x
  1.339   1.250                                    1          1.250
   .412   1.250                                                .412
   .461   1.250                                                .461
   .670   1.250                                                .670
   .509   1.250                                                .509
   .365   1.250                                                .365
   .329   1.250                                                .329
   .706   1.250                                                .706
   .628   1.250                                                .628
   .415   1.250                                                .415
   .495   1.250                                                .495
   .571   1.250                                                .571
   .233   1.250                                                .233
   .547   1.250                                                .547
   .636   1.250                                                .636
  1.207   1.250                                               1.207
   .612   1.250                                                .612
  1.043   1.250                                               1.043
   .665   1.250                                                .665
   .679   1.250                                                .679
   .244   1.250                                                .244
   .524   1.250                                                .524
   .674   1.250                                                .674
   .531   1.250                                                .531
   .852   1.250                                                .852
  1.147   1.250                                               1.147
   .946   1.250                                                .946
   .882   1.250                                                .882
   .494   1.250                                                .494
   .742   1.250                                                .742
   .249   1.250                                                .249
   .172   1.250                                                .172
   .552   1.250                                                .552
   .489   1.250                                                .489
   .903   1.250                                                .903
```

.689	1.250			.689
.673	1.250			.673
.830	1.250			.830
.878	1.250			.878
.194	1.250			.194
.663	1.250			.663
.488	1.250			.488
.476	1.250			.476
1.312	1.250		1	1.250
.598	1.250			.598
.397	1.250			.397
.287	1.250			.287
.630	1.250			.630
.341	1.250			.341
.328	1.250			.328
.964	1.250			.964
.831	1.250			.831
.467	1.250			.467
.770	1.250			.770
.569	1.250			.569
.285	1.250			.285
.502	1.250			.502
.533	1.250			.533
.184	1.250			.184
.599	1.250			.599
1.101	1.250			1.101
.537	1.250			.537
.939	1.250			.939
.847	1.250			.847
.390	1.250			.390
.926	1.250			.926
.527	1.250			.527
.413	1.250			.413
.491	1.250			.491
.580	1.250			.580
.669	1.250			.669
.984	1.250			.984
.900	1.250			.900
.616	1.250			.616
.865	1.250			.865

Sum of logs to the base e = -.4245930E+02
No. of failures = 73

```
     kappa          rho
   2.55149       1.39966
   2.51731       1.41145
   2.53984       1.40363
   2.52492       1.40879
   2.53477       1.40537
   2.52826       1.40763
   2.53256       1.40613
   2.52972       1.40712
   2.53160       1.40647
   2.53035       1.40690
   2.53118       1.40661
   2.53063       1.40680
   2.53099       1.40668
   2.53075       1.40676
   2.53091       1.40670
   2.53081       1.40674
   2.53088       1.40672
   2.53083       1.40673
   2.53086       1.40672
   2.53084       1.40673
   2.53085       1.40672
   2.53085       1.40673
   2.53085       1.40673
   2.53085       1.40673

Elements of the information matrix -
ipp =       .2362843E+03
ipk =       .2032484E+02
ikk =       .1398275E+04
```