F77 Program 8

```
c     Fortran 77 program to do a simulation of the Weibull distribution
c     of the censored model. Realistic case Mark 2.
c     Recruitment model is to follow an exponential distribution.
c     This program may be adapted for other forms of censored models as
c     well as the simplest censoring model.
c     ***************************************************************
c           By Derek Dhammaloka FDX3 - 11th Dec. 1990
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           cdfex is as cdf but for the exponential distribution and
c           has seed iy2.
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           cp is the censor time (recruitment) that follows an
c           exponential distribution with rate rhoex
c           rhoex is to be entered by the user
c           cl is the censor time to be obtained using the equation
c           cl=censorstart+((n-loop)/(n-1)) where censorstart is
c           the initial censor time to be entered by the user
c           c is the minimum of cp and cl
c           meanex is the mean of the exponential distribution
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,iy2 are the seeds to be entered by the user.
c     ***************************************************************
      integer i(5000)
      real cdf(5000),cdfex(5000),t(5000),cl(5000),cp(5000),c(5000)
      real x(5000)
      real sumln,sumpower,sumpowerln,spln2,kold,knew,kdiff
      real censorstart,rhoex,meanex
      real rho,tol,ipp,ipk,ikk
      integer loop,n,d,iy,iy2
      tol=0.000005
c     ***************************************************************
```

```
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*
      print*,'For the censor times'
      print*,'Enter the initial censor time'
      print*
      read*,censorstart
      print*,'For the recruitment times exponentially distributed'
      print*,'Enter seed'
      read*,iy2
      print*
      print*,'Enter rate'
      read*,rhoex
      meanex=1/rhoex
c     ********************************************************************
c           Simulate the Weibull distribution
c           using the two parameters to obtain the remission times.
c           Actual censor times are obtained by taking the minimum of
c           the recruitment times obtained by using the exponential
c           distribution and the recruitment times obtained by using
c           a straight line.
c     ********************************************************************
      do 20 loop=1,n
           cdf(loop)=urand(iy)
           cdfex(loop)=urand(iy2)
           t(loop)=(-log(1-cdf(loop))/(rho**knew)**(1/knew)
           cp(loop)=(-log(1-cdfex(loop)))/(rhoex)
           cl(loop)=(censorstart*(n-1))+(n-loop)
           cl(loop)=(c(loop))/(n-1)
c
c           Find the actual censor time by taking the minimum of the
c           recruitment times obtained using a Poisson process and
c           the straight line
c
           if(cp(loop).lt.cl(loop)) c(loop)=cp(loop)
           if(cp(loop).ge.cl(loop)) c(loop)=cl(loop)
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*,'Realistic case model II'
          print*
          print*,'Simulation of the Weibull distribution with '
          print*,'Index = ',knew,' and rate = ',rho
          print*
          print*,'Initial censor time = ',censorstart
          print*
          print*,'Recruitment times exponentially distributed with'
          print*,'Rate = ',rhoex,' ie Mean = ',meanex
          print*
          write(*,25)
25        format(t3,'T',t12,'CL',t20,'CP',t30,'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),cl(loop),cp(loop),c(loop),i(loop),x(loop)
40           format(f7.3,t8,f7.3,t16,f7.3,t26,f7.3,t38,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 8

```
How many individuals
75
Seed for the Weibull distribution
2
Enter the index parameter
2.5
Enter the rate parameter
1.5

For the censor times
Enter the initial censor time
0.5
For the recruitment times exponentially distributed
Enter seed
-2

Enter rate
0.8

Realistic case model II

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

Initial censor time =      .5000000E+00

Recruitment times exponentially distributed with
Rate =     .8000000E+00 ie Mean =       .1250000E+01
```

T	CL	CF	C	I	x
1.339	1.500	.694	.694	1	.694
.412	1.486	.363	.363	1	.363
.461	1.473	1.393	1.393		.461
.670	1.459	.031	.031	1	.031
.509	1.446	3.358	1.446		.509
.365	1.432	.074	.074	1	.074
.329	1.419	.976	.976		.329
.706	1.405	1.452	1.405		.706
.628	1.392	.247	.247	1	.247
.415	1.378	2.588	1.378		.415
.495	1.365	1.702	1.365		.495
.571	1.351	2.230	1.351		.571
.233	1.338	.604	.604		.233
.547	1.324	3.532	1.324		.547
.636	1.311	.866	.866		.636
1.207	1.297	.808	.808	1	.808
.612	1.284	2.989	1.284		.612
1.043	1.270	.817	.817	1	.817
.665	1.257	.184	.184	1	.184
.679	1.243	1.334	1.243		.679
.244	1.230	.963	.963		.244
.524	1.216	.733	.733		.524
.674	1.203	1.933	1.203		.674
.531	1.189	.268	.268	1	.268

.852	1.176	2.666	1.176		.852
1.147	1.162	1.161	1.161		1.147
.946	1.149	.317	.317	1	.317
.882	1.135	.130	.130	1	.130
.494	1.122	.424	.424	1	.424
.742	1.108	.765	.765		.742
.249	1.095	3.173	1.095		.249
.172	1.081	2.293	1.081		.172
.552	1.068	.097	.097	1	.097
.489	1.054	2.830	1.054		.489
.903	1.041	3.302	1.041		.903
.689	1.027	.240	.240	1	.240
.673	1.014	.740	.740		.673
.830	1.000	.191	.191	1	.191
.878	.986	1.364	.986		.878
.194	.973	1.921	.973		.194
.663	.959	3.394	.959		.663
.488	.946	2.736	.946		.488
.476	.932	.289	.289	1	.289
1.312	.919	.699	.699	1	.699
.598	.905	1.470	.905		.598
.397	.892	.177	.177	1	.177
.287	.878	.266	.266	1	.266
.630	.865	2.098	.865		.630
.341	.851	.899	.851		.341
.328	.838	1.336	.838		.328
.964	.824	2.879	.824	1	.824
.831	.811	3.392	.811	1	.811
.467	.797	2.774	.797		.467
.770	.784	1.271	.784		.770
.569	.770	1.254	.770		.569
.285	.757	.086	.086	1	.086
.502	.743	5.082	.743		.502
.533	.730	.531	.531	1	.531
.184	.716	4.342	.716		.184
.599	.703	.224	.224	1	.224
1.101	.689	2.042	.689	1	.689
.537	.676	.644	.644		.537
.939	.662	6.154	.662	1	.662
.847	.649	2.363	.649	1	.649
.390	.635	.921	.635		.390
.926	.622	1.058	.622	1	.622
.527	.608	.127	.127	1	.127
.413	.595	1.675	.595		.413
.491	.581	.211	.211	1	.211
.580	.568	.981	.568	1	.568
.669	.554	.725	.554	1	.554
.984	.541	.465	.465	1	.465
.900	.527	1.060	.527	1	.527
.616	.514	1.482	.514	1	.514
.865	.500	1.382	.500	1	.500

Sum of logs to the base e = -.2805671E+02
No. of failures = 39

kappa	rho
3.09779	1.30413
2.65131	1.48196
2.96326	1.34644
2.73454	1.43947
2.89652	1.37050
2.77891	1.41892
2.86279	1.38356
2.80219	1.40867
2.84557	1.39047
2.81431	1.40347
2.83673	1.39409
2.82060	1.40081
2.83217	1.39597
2.82385	1.39944
2.82983	1.39694
2.82553	1.39873
2.82862	1.39744
2.82640	1.39837
2.82799	1.39771
2.82685	1.39818
2.82767	1.39784
2.82708	1.39809
2.82750	1.39791
2.82720	1.39804
2.82742	1.39794
2.82726	1.39801
2.82738	1.39796
2.82729	1.39800
2.82735	1.39797
2.82731	1.39799
2.82734	1.39798
2.82732	1.39799
2.82733	1.39798
2.82732	1.39799
2.82733	1.39798
2.82732	1.39798
2.82733	1.39798

Elements of the information matrix -
ipp = .1595192E+03
ipk = -.2420006E+02
ikk = .2295474E+04