F77 Program 16

```
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 - 21st 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                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                swops is the no. of swops
c                sorts is the no. of sorts
c                temp is used in sorting data values
c                list assigns the values 1,n so that it can be used in
c                sorting 1 variable in ascending or descending order and
c                the other variables have the same values as the original
c                data, but is being sorted
c                psn is the no. of trials in view at a certain time
c                ple is the product limit estimator
c                prcensor is the probability of censoring
c                surf is the survivor function with its estimated
```

```
c           parameters, kappa and rho of the Weibull distribution
c           diff is the absolute difference between the product
c           limit estimator and the survivor function (does not have
c           to be declared as an array!)
c           maxdiff is the maximum of the absolute differences
c           cvald5 is the critical value of d at the 5% level
c           iy,iy2 are the seeds to be entered by the user.
c     ************************************************************************
c           Estimate kappa and rho
c           Find the product limit estimator
c           Use the K-S test to examine the goodness of fit
c     ************************************************************************
      integer i(5000),list(5000)
      real cdf(5000),cdfex(5000),t(5000),c1(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,ple,prcensor,surf,diff,maxdiff,cvald5
      integer loop,temp,swops,sorts,n,d,psn,iy,iy2
      tol=0.000005
      ple=1
      swops=1
      sorts=0
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
                list(loop)=loop
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
c
c
c           Sort the x values in ascending order
c
85    if(swops.ne.0.and.sorts.lt.n-1) then
        swops=0
        sorts=sorts+1
        do 86 loop=1,n-sorts
              if(x(list(loop)).gt.x(list(loop+1))) then
                temp=list(loop)
                list(loop)=list(loop+1)
                list(loop+1)=temp
                swops=swops+1
                endif
86      continue
        goto 85
      endif
c
c
c           Print headings
c
      print*
      print*,'Calculation of the product limit estimator'
      print*,'(No. of failures assumed to be 1 for uncensored obs.)'
      print*
      print*,'Survivor function of the Weibull distribution with
      print*,'Index = ',knew,' and rate = ',rho
      print*
      write(*,63)
63    format(t3,'atom',t14,'r',t27,'(1-(1/r))',t42,'ple',t55,'Sur. fn')
c
c           Calculate the product limit estimator using a do loop
c           and compare it with the survivor function
c           Assume that the no. of failures for each x value is 1
c           providing that the individual is not censored since
c           the times are on a continuous scale and it is rare
c           for ties to occur (cf Poisson process)
c
      do 65 loop=1,n
         psn=psn-1
c
c
c           Only print the atom, position (no. of trials),censor
c           probability, product limit, survivor function and
c           absolute difference if there is no censoring in the
c           individual.
c
         if(i(list(loop)).eq.0) then
           prcensor=psn-1
           prcensor=prcensor/psn
           ple=ple*(prcensor)
           surf=exp(-(rho*x(list(loop)))**knew)
           diff=abs(ple-surf)
c
c           If the absolute difference exceeds the maximum
c           difference then the maximum difference equals the
```

```
c
c           absolute difference
c
        if(diff.ge.maxdiff) maxdiff=diff

c
c           Final column gives the difference between the
c           product limit estimator and the survivor function
c
        write(*,75)x((list(loop))),psn,prcensor,ple,surf,diff
75      format(f7.3,t8,i7.0,t25,f7.4,t40,f7.4,t55,f7.4,t68,f7.4,t80)
        endif
65    continue
c
c           Print the final product limit estimator and the
c           survivor function. Also print the maximum difference
c
      print*
      print*,'Product limit estimator          = ',ple
      print*,'Survivor function                = ',surf
      print*,
      print*,'Calculated maximum difference    = ',maxdiff
c
c           Lookup the critical value of d at the 5% level given the
c           number of failures, using block-IF statements.
c
c           The data is taken from Table 16, Percentage points of
c           d in the 1-sample Kolmogorov-Smirnov distribution in
c           Statistical Tables by J Murdoch and J A Barnes, 1986
c           published by Macmillan Education

      if(d.eq.1) cvald5=0.975
      if(d.eq.2) cvald5=0.842
      if(d.eq.3) cvald5=0.708
      if(d.eq.4) cvald5=0.624
      if(d.eq.5) cvald5=0.565
      if(d.eq.6) cvald5=0.521
      if(d.eq.7) cvald5=0.486
      if(d.eq.8) cvald5=0.457
      if(d.eq.9) cvald5=0.432
      if(d.eq.10) cvald5=0.41
      if(d.eq.11) cvald5=0.391
      if(d.eq.12) cvald5=0.375
      if(d.eq.13) cvald5=0.361
      if(d.eq.14) cvald5=0.349
      if(d.eq.15) cvald5=0.338
      if(d.eq.16) cvald5=0.328
      if(d.eq.17) cvald5=0.318
      if(d.eq.18) cvald5=0.309
      if(d.eq.19) cvald5=0.301
      if(d.eq.20) cvald5=0.294
c
c           d values for n=21,22,23 and 24 are not shown in
c           this table, so apply linear interpolation for these
c           n values (21,22,23 or 24)
c
      if(d.gt.20.and.d.lt.25) cvald5=(0.294)-(0.0048*(d-20))
c
```

```
      if(d.eq.25) cvald5=0.27

c           d values for n=26,27,28 and 29 are not shown in
c           this table, so apply linear interpolation for these
c           n values (26,27,28 or 29)

      if(d.gt.25.and.d.lt.30) cvald5=(0.27)-(0.006*(d-25))

      if(d.eq.30) cvald5=0.24

c           d values for n=31,32,33 and 34 are not shown in
c           this table, so apply linear interpolation for these
c           n values (31,32,33 or 34)

      if(d.gt.30.and.d.lt.35) cvald5=(0.24)-(0.002*(d-30))

      if(d.eq.35) cvald5=0.23

c           For sample sizes larger than 35, use the formula
c           1.36/sqrt(n) to obtain critical values

      if(d.gt.35) cvald5=(1.36)/sqrt(d)

c           Print the critical value of d at ther 5% level and
c           compare it with the maximum absolute difference.
c           Therefore give a conclusion about this model

      print*,'Critical value of d at 5% level = ',cvald5
      print*

c           If the calculated maximum difference is less than
c           the tabulated d value, then the product limit estimator
c           is consistent with the survivor function, otherwise the
c           product limit estimator is not consistent with the
c           survivor function

      if(maxdiff.lt.cvald5) then
         print*,'The product limit estimator is reasonably consistent'
         print*,'with the survivor function'
      else
         print*,'The product limit estimator is not consistent with'
         print*,'consistent with the survivor function'
      endif
      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 16

```
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	CP	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
2.82583	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	1.39798
2.82733	1.39798

Elements of the information matrix -
ipp = .1595192E+03
ipk = -.2420006E+01
ikk = .2295474E+04

Calculation of the product limit estimator
(No. of failures assumed to be 1 for uncensored obs.)

Survivor function of the Weibull distribution with
Index = .2827326E+01 and rate = .1397983E+01

x	r	(1-(1/r))	ple	Sur. fn	
.172	69	.9855	.9855	.9823	.0032
.184	66	.9848	.9706	.9786	.0080
.194	64	.9844	.9554	.9753	.0199
.233	61	.9836	.9397	.9590	.0193
.244	59	.9831	.9238	.9532	.0294

.249	57	.9825	.9076	.9504	.0428
.328	52	.9808	.8902	.8958	.0057
.329	51	.9804	.8727	.8945	.0218
.341	50	.9800	.8553	.8844	.0291
.390	48	.9792	.8374	.8349	.0026
.413	47	.9787	.8196	.8094	.0102
.415	46	.9783	.8018	.8074	.0056
.461	44	.9773	.7836	.7487	.0348
.467	42	.9762	.7649	.7416	.0233
.488	41	.9756	.7463	.7121	.0342
.489	40	.9750	.7276	.7111	.0165
.495	39	.9744	.7089	.7026	.0063
.502	37	.9730	.6898	.6932	.0034
.509	36	.9722	.6706	.6826	.0120
.524	34	.9706	.6509	.6598	.0089
.537	31	.9677	.6299	.6406	.0107
.547	30	.9667	.6089	.6263	.0174
.569	27	.9630	.5864	.5917	.0053
.571	26	.9615	.5638	.5894	.0256
.598	25	.9600	.5413	.5479	.0067
.612	24	.9583	.5187	.5256	.0069
.630	22	.9545	.4951	.4969	.0019
.636	21	.9524	.4715	.4884	.0169
.663	18	.9444	.4453	.4458	.0004
.673	17	.9412	.4192	.4310	.0118
.674	16	.9375	.3930	.4295	.0365
.679	15	.9333	.3668	.4218	.0550
.706	11	.9091	.3334	.3818	.0484
.742	10	.9000	.3001	.3301	.0300
.770	9	.8889	.2667	.2919	.0251
.852	4	.7500	.2000	.1946	.0055
.878	3	.6667	.1334	.1675	.0341
.903	2	.5000	.0667	.1450	.0783
1.147	1	.0000	.0000	.0223	.0223

```
Product limit estimator              =     .0000000E+00
Survivor function                    =     .2230304E-01

Calculated maximum difference        =     .7831535E-01
Critical value of d at 5% level      =     .2177743E+00

The product limit estimator is reasonably consistent
with the survivor function
```