F77 Program 15

```
c       Fortran 77 program to do a simulation of the Weibull distribution
c       of the censored model. Realistic case.
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               t is the remission time in arbitary units
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 censor time to be obtained using the equation
c               c=censorstart+((n-loop)/(n-1)) where censorstart is
c               the initial censor time 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               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 is the seed 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),t(5000),c(5000),x(5000)
      real sumln,sumpower,sumpowerln,spln2,kold,knew,kdiff
      real censorstart
      real rho,tol,ipp,ipk,ikk,ple,prcensor,surf,diff,maxdiff,cvald5
      integer loop,temp,swops,sorts,n,d,psn,iy
      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
c
c             Set the no. of trials in view at time=0 to n+1
c
      psn=n+1
      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
c     ****************************************************************
c             Simulate the Weibull distribution
c             using the two parameters to obtain the remission times.
c             However, use an equation to obtain the censor times.
c     ****************************************************************
      do 20 loop=1,n
              cdf(loop)=urand(iy)
              t(loop)=(-log(1-cdf(loop))/(rho**knew)**(1/knew)
              c(loop)=(censorstart*(n-1))+(n-loop)
              c(loop)=(c(loop))/(n-1)
c
c             Decide whether the individual is to be censored
c             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 I'
       print*
       print*,'Simulation of the Weibull distribution with '
       print*,'Index = ',knew,' and rate = ',rho
       print*
       print*,'Initial censor time = ',censorstart
       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
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             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         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         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
c
      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
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)
c
      if(d.gt.25.and.d.lt.30) cvald5=(0.27)-(0.006*(d-25))
c
      if(d.eq.30) cvald5=0.24
c
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)
c
      if(d.gt.30.and.d.lt.35) cvald5=(0.24)-(0.002*(d-30))
c
      if(d.eq.35) cvald5=0.23
c
c             For sample sizes larger than 35, use the formula
c             1.36/sqrt(n) to obtain critical values
c
      if(d.gt.35) cvald5=(1.36)/sqrt(d)
c
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
c
      print*,'Critical value of d at 5% level = ',cvald5
      print*
c
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
c
      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 15

```
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

Realistic case model I

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

Initial censor time =      .5000000E+00
```

T	C	I	x
1.339	1.500		1.339
.412	1.486		.412
.461	1.473		.461
.670	1.459		.670
.509	1.446		.509
.365	1.432		.365
.329	1.419		.329
.706	1.405		.706
.628	1.392		.628
.415	1.378		.415
.495	1.365		.495
.571	1.351		.571
.233	1.338		.233
.547	1.324		.547
.636	1.311		.636
1.207	1.297		1.207
.612	1.284		.612
1.043	1.270		1.043
.665	1.257		.665
.679	1.243		.679
.244	1.230		.244
.524	1.216		.524
.674	1.203		.674
.531	1.189		.531
.852	1.176		.852
1.147	1.162		1.147
.946	1.149		.946
.882	1.135		.882
.494	1.122		.494
.742	1.108		.742
.249	1.095		.249
.172	1.081		.172

.552	1.068		.552
.489	1.054		.489
.903	1.041		.903
.689	1.027		.689
.673	1.014		.673
.830	1.000		.830
.878	.986		.878
.194	.973		.194
.663	.959		.663
.488	.946		.488
.476	.932		.476
1.312	.919	1	.919
.598	.905		.598
.397	.892		.397
.287	.878		.287
.630	.865		.630
.341	.851		.341
.328	.838		.328
.964	.824	1	.824
.831	.811	1	.811
.467	.797		.467
.770	.784		.770
.569	.770		.569
.285	.757		.285
.502	.743		.502
.533	.730		.533
.184	.716		.184
.599	.703		.599
1.101	.689	1	.689
.537	.676		.537
.939	.662		.939
.847	.649	1	.649
.390	.635		.390
.926	.622	1	.926
.527	.608		.527
.413	.595		.413
.491	.581		.491
.580	.568	1	.568
.669	.554	1	.554
.984	.541	1	.541
.900	.527	1	.527
.616	.514	1	.514
.865	.500	1	.500

```
Sum of logs to the base e =      -.4003725E+02
No. of failures =                 62

        kappa        rho
      2.63392      1.39685
      2.52975      1.43570
      2.60993      1.40538
      2.54771      1.42865
      2.59568      1.41055
      2.55851      1.42449
      2.58721      1.41367
      2.56499      1.42201
      2.58215      1.41555
      2.56887      1.42054
      2.57913      1.41668
      2.57119      1.41966
      2.57733      1.41735
      2.57258      1.41914
      2.57626      1.41776
      2.57341      1.41882
      2.57561      1.41800
      2.57391      1.41864
      2.57523      1.41814
      2.57421      1.41853
      2.57499      1.41823
      2.57439      1.41846
      2.57486      1.41828
      2.57449      1.41842
      2.57478      1.41831
      2.57456      1.41839
      2.57472      1.41833
      2.57460      1.41838
      2.57469      1.41834
      2.57462      1.41837
      2.57468      1.41835
      2.57463      1.41837
      2.57467      1.41835
      2.57464      1.41836
      2.57466      1.41836
      2.57465      1.41836
      2.57466      1.41836
      2.57465      1.41836
      2.57466      1.41836
      2.57465      1.41836
      2.57465      1.41836

Elements of the information matrix -
ipp =      .2042950E+03
ipk =      .1036976E+02
ikk =      .1145519E+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 = .2574654E+01 and rate = .1418358E+01

x	r	(1-(1/r))	ple	Sur. fn	
.172	75	.9867	.9867	.9738	.0069
.184	74	.9865	.9733	.9688	.0045
.194	73	.9863	.9600	.9646	.0046
.233	72	.9861	.9467	.9440	.0027
.244	71	.9859	.9333	.9368	.0035
.249	70	.9857	.9200	.9334	.0134
.285	69	.9855	.9067	.9075	.0009
.287	68	.9853	.8933	.9058	.0124
.328	67	.9851	.8800	.8702	.0098
.329	66	.9848	.8667	.8687	.0020
.341	65	.9846	.8533	.8574	.0041
.365	64	.9844	.8400	.8319	.0081
.390	63	.9841	.8267	.8039	.0228
.397	62	.9839	.8133	.7963	.0170
.412	61	.9836	.8000	.7777	.0223
.413	60	.9833	.7867	.7771	.0096
.415	59	.9831	.7733	.7750	.0017
.461	58	.9828	.7600	.7149	.0451
.467	57	.9825	.7467	.7077	.0389
.476	56	.9821	.7333	.6951	.0382
.488	55	.9818	.7200	.6783	.0417
.489	54	.9815	.7067	.6773	.0294
.491	53	.9811	.6933	.6739	.0194
.494	52	.9808	.6800	.6703	.0097
.495	51	.9804	.6667	.6689	.0023
.502	49	.9796	.6531	.6596	.0065
.509	48	.9792	.6395	.6493	.0098
.524	46	.9783	.6256	.6270	.0015
.527	44	.9773	.6113	.6229	.0116
.531	43	.9767	.5971	.6174	.0203
.533	42	.9762	.5829	.6149	.0320
.537	41	.9756	.5687	.6084	.0397
.547	39	.9744	.5541	.5946	.0405
.552	38	.9737	.5395	.5870	.0475
.569	35	.9714	.5241	.5616	.0375
.571	34	.9706	.5087	.5594	.0507
.598	33	.9697	.4933	.5203	.0270
.599	32	.9687	.4779	.5185	.0406
.612	31	.9677	.4624	.4993	.0369
.628	29	.9655	.4465	.4762	.0297
.630	28	.9643	.4306	.4726	.0420
.636	27	.9630	.4146	.4647	.0501
.663	24	.9583	.3973	.4254	.0281
.665	23	.9565	.3801	.4229	.0429
.670	22	.9545	.3628	.4167	.0539
.673	21	.9524	.3455	.4118	.0663
.674	20	.9500	.3282	.4104	.0822
.679	19	.9474	.3110	.4034	.0924
.689	18	.9444	.2937	.3896	.0960

.706	16	.9375	.2753	.3669	.0915
.742	15	.9333	.2570	.3198	.0629
.770	14	.9286	.2386	.2852	.0466
.830	11	.9091	.2169	.2186	.0017
.852	10	.9000	.1952	.1967	.0015
.878	9	.8889	.1735	.1719	.0016
.882	8	.8750	.1518	.1688	.0170
.903	7	.8571	.1302	.1511	.0209
.946	5	.8000	.1041	.1189	.0148
1.043	4	.7500	.0781	.0645	.0136
1.147	3	.6667	.0521	.0301	.0220
1.207	2	.5000	.0260	.0184	.0076
1.339	1	.0000	.0000	.0054	.0054

```
Product limit estimator          =     .0000000E+00
Survivor function                =     .5418546E-02

Calculated maximum difference    =     .9596300E-01
Critical value of d at 5% level  =     .1727202E+00

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