F77 Program 9

```
c       Fortran 77 program to do a simulation of the Weibull distribution
c       of the uncensored model.
c       This program may be adapted for censored models as well as for
c       uncensored models.
c       ***************************************************************
c                By Derek Dhammaloka FDX3 - 14th 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                knew is the new value of kappa to be entered by the user.
c                rho is the rate to be entered by the user
c                loop is used in loop counters
c                n is the no. of uncensored individuals to be entered by
c                the user.
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                iy is the seed to be entered by the user.
c       ***************************************************************
c                Find the product limit estimator
c       ***************************************************************
        real cdf(5000),t(5000)
        real knew,rho,ple,prcensor
        integer loop,temp,swops,sorts,psn,list(5000),n,iy
        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
        print*,'Seed'
        read*,iy
        print*,'Enter the index parameter
        read*,knew
        print*,'Enter the rate parameter'
        read*,rho
c       ***************************************************************
c                Simulate the Weibull distribution
c                using the two parameters to obtain the remission times
```

```
c     ***************************************************************************
      do 20 loop=1,n
              cdf(loop)=urand(iy)
              t(loop)=(-log(1-cdf(loop))/(rho**knew)**(1/knew)
              list(loop)=loop
20    continue
c     ***************************************************************************
c             Print the headings
c     ***************************************************************************
      print*
      print*, Simulation of the Weibull distribution with '
      print*,'Index = ',knew,' and rate = ',rho
      print*
      write(*,25)
25    format(t3,'cdf',t10,'time')
c     ***************************************************************************
c             Output the cdf and remission time values
c     ***************************************************************************
      do 30 loop=1,n
              write(*,40)cdf(loop),t(loop)
40            format(f7.3,t8,f7.3,t37)
30    continue
      print*
c
c             Sort the t values in ascending order
c
50    if(swops.ne.0.and.sorts.lt.n-1) then
        swops=0
        sorts=sorts+1
        do 60 loop=1,n-sorts
              if(t(list(loop)).gt.t(list(loop+1))) then
                temp=list(loop)
                list(loop)=list(loop+1)
                list(loop+1)=temp
                swops=swops+1
                endif
60      continue
        goto 50
      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*
      write(*,63)
63    format(t3,'atom',t14,'r',t27,'(1-(1/r))',t42,'ple')
c
c             Calculate the product limit estimator using a do loop
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.
c
      do 65 loop=1,n
```

```
            psn=psn-1
c
c           Only print the atom, position, censor probability and
c           the product limit if there is no censoring in the
c           individual.
c
            prcensor=psn-1
            prcensor=prcensor/psn
            ple=ple*(prcensor)
            write(*,75)t((list(loop))),psn,prcensor,ple
75          format(f7.3,t8,i7.0,t25,f7.4,t40,f7.4,t55)
65    continue
c
c           Print the final product limit estimator
c
            print*
            print*,'Product limit estimator =  ',ple
      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 9

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

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

   cdf      time
    .997   1.339
    .260    .412
    .329    .461
    .836    .670
    .399    .509
    .199    .365
    .157    .329
    .684    .706
    .577    .628
    .263    .415
    .378    .495
    .493    .571
    .070    .233
    .456    .547
    .589    .636
    .988   1.207
    .554    .612
    .953   1.043
    .630    .665
    .649    .679
    .078    .244
    .422    .524
    .642    .674
    .433    .531
    .842    .852
    .979   1.147
    .909    .946
    .866    .882
    .377    .494
    .729    .742
    .082    .249
    .033    .172
    .464    .552
    .389    .489
    .882    .903
    .662    .689
    .641    .673
    .822    .830
    .864    .878
    .045    .194
    .628    .663
```

```
.368    .488
.350    .476
.996   1.312
.533    .598
.239    .397
.115    .287
.581    .630
.170    .341
.156    .328
.919    .964
.823    .831
.336    .467
.762    .770
.491    .569
.113    .285
.388    .502
.435    .533
.039    .184
.535    .599
.970   1.101
.442    .537
.905    .939
.838    .847
.231    .390
.897    .926
.427    .527
.261    .413
.373    .491
.507    .580
.635    .669
.929    .984
.880    .900
.560    .616
.853    .865
```

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

atom	r	(1-(1/r))	ple
.172	75	.9867	.9867
.184	74	.9865	.9733
.194	73	.9863	.9600
.233	72	.9861	.9467
.244	71	.9859	.9333
.249	70	.9857	.9200
.285	69	.9855	.9067
.287	68	.9853	.8933
.328	67	.9851	.8800
.329	66	.9848	.8667
.341	65	.9846	.8533
.365	64	.9844	.8400
.390	63	.9841	.8267
.397	62	.9839	.8133
.412	61	.9836	.8000
.413	60	.9833	.7867
.415	59	.9831	.7733
.461	58	.9828	.7600
.467	57	.9825	.7467
.476	56	.9821	.7333
.488	55	.9818	.7200
.489	54	.9815	.7067
.491	53	.9811	.6933
.494	52	.9808	.6800
.495	51	.9804	.6667
.502	50	.9800	.6533
.509	49	.9796	.6400
.524	48	.9792	.6267
.527	47	.9787	.6133
.531	46	.9783	.6000
.533	45	.9778	.5867
.537	44	.9773	.5733
.547	43	.9767	.5600
.552	42	.9762	.5467
.569	41	.9756	.5333
.571	40	.9750	.5200
.580	39	.9744	.5067
.598	38	.9737	.4933
.599	37	.9730	.4800
.612	36	.9722	.4667
.616	35	.9714	.4533
.628	34	.9706	.4400
.630	33	.9697	.4267
.636	32	.9687	.4133
.663	31	.9677	.4000
.665	30	.9667	.3867
.669	29	.9655	.3733
.670	28	.9643	.3600
.673	27	.9630	.3467
.674	26	.9615	.3333
.675	25	.9600	.3200
.689	24	.9583	.3067

.706	23	.9565	.2933
.742	22	.9545	.2800
.770	21	.9524	.2667
.830	20	.9500	.2533
.831	19	.9474	.2400
.847	18	.9444	.2267
.852	17	.9412	.2133
.865	16	.9375	.2000
.878	15	.9333	.1867
.882	14	.9286	.1733
.900	13	.9231	.1600
.903	12	.9167	.1467
.926	11	.9091	.1333
.939	10	.9000	.1200
.946	9	.8889	.1067
.964	8	.8750	.0933
.984	7	.8571	.0800
1.043	6	.8333	.0667
1.101	5	.8000	.0533
1.147	4	.7500	.0400
1.207	3	.6667	.0267
1.312	2	.5000	.0133
1.339	1	.0000	.0000

Product limit estimator = .0000000E+00