

F77 Program 19

```

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 - 28th 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 arbitrary 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      sep and sek are the standard errors of rho and kappa
c      cipub95 and cipub95 are the lower and upper bounds of
c      rho with 95% confidence
c      ciklb95 and cikub95 are the lower and upper bounds of
c      kappa with 95% confidence
c      rho is the rate to be entered by the user
c      l0 and l1 are the values of the likelihood function
c      under H0 and H1 respectively, where l1 must be greater
c      than l0, but how much bigger?
c      lambda is the difference between l1 and l0
c      cvalchi5 is the critical value of chi-square at the 5
c      per cent level
c      netscaleddev is the change in scaled deviance
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      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
real censorstart,sep,sek,cipub95,cipub95,ciklb95,cikub95
real rho,tol,ipp,ipk,ikk,l0,l1,lambda,cvalchi5,netscaleddev

```

APPENDIX 02 - Fortran 77 (F77) Programs

```

integer loop,n,d,iy
tol=0.000005
*****
      Input the no. of individuals
      Also the index (kappa) and the rate (rho)
*****
print*, 'How many individuals'
read*,n

      Set the no. of failures to the no. of observations

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
*****
      Simulate the Weibull distribution
      using the two parameters to obtain the remission times.
      However, use an equation to obtain the censor times.
*****
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)

      Decide whether the censored individual is to be
      eliminated using the censoring rules

      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
*****
      Print the headings
*****
print*
print*, 'Realistic case model I'
print*
print*, 'Simulation of the Weibull distribution with
print*, 'Index = ',knew,' and rate = ',rho
print*

```

APPENDIX 02 - Fortran 77 (F77) Programs

```

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.

```

APPENDIX 02 - Fortran 77 (F77) Programs

```

*****
if(abs(kdiff).gt.tol) then
    sumpower=0
    sumpowerln=0
    spln2=0
    goto 50
endif
*****
Print the elements of the information matrix
*****
print*
print*, 'Elements of the information matrix -
print*, 'ipp = ', ipp
print*, 'ipk = ', ipk
print*, 'ikk = ', ikk
*****
Assign and print the standard errors of rho and kappa
*****
print*
sep=1/(sqrt(ipp))
sek=1/(sqrt(ikk))
print*, 'Standard errors -
print*, 'rho = ', sep
print*, 'kappa = ', sek
*****
Assign and print the 95% confidence intervals of rho and
kappa. Assume that the Asymptotic Theorem for MLEs hold.
*****
print*
ciplb95=(rho)-(1.96*sep)
cipub95=(rho)+(1.96*sep)
ciklb95=(knew)-(1.96*sek)
cikub95=(knew)+(1.96*sek)
print*, 'With 95% confidence -
print*, 'rho is between ', ciplb95, ' and ', cipub95
print*, 'kappa is between ', ciklb95, ' and ', cikub95
*****
Perform log likelihood ratio test and use it to test
for exponentiality, ie by putting kappa = 1
*****
print*
Print null and alternative hypotheses
*****
print*, 'H0 : kappa = 1, for exponentiality'
print*, 'H1 : kappa <> 1, for non-exponentiality'
print*
*****
The following statements give the likelihood
function of the Weibull distribution under the null
and alternative hypotheses
*****
For 10 use the examined values of kappa and rho
For 11 use the estimated values of kappa and rho
*****
l0=(d*log(rho))-d

```

APPENDIX 02 - Fortran 77 (F77) Programs

```

l1=(d*log(knew))+(knew*d*log(rho))+((knew-1)*sumln)-d
c
c      Print the values of the likelihood function
c      under the null and alternative hypotheses
c
print*, 'Likelihood function under H0 = ',l0
print*, 'Likelihood function under H1 = ',l1
c
c      Assume that the large sample method holds and use
c      2 ln constant = chi-square, where ln constant is
c      equal to l1-l0 and 2 ln constant represents the
c      change in scaled deviance.
c
lambda=l1-l0
print*
print*, 'Value of l1 - l0 = ',lambda
netscaleddev=2*lambda
print*
c
c      Do not proceed further with this test, unless
c      l1 is greater than or equal to l0
c
if(lambda.lt.0) then
  print*, 'Cannot use the likelihood ratio test, since
  print*, 'l1 is less than l0'
else
  print*, 'Change in scaled deviance = ',netscaleddev
c
c      The following statement represents the critical
c      value of chi-square at the 5 per cent level on
c      1 degree of freedom
c
cvalchi5=3.841
print*, 'Chi-sq at 5% level on 1 df = ',cvalchi5
print*
c
c      Compare the change in scaled deviance with the
c      critical value of chi-square at the 5% level
c      on 1 degree of freedom
c
c      If the change in scaled deviance is less than
c      this critical value of chi-square, then we
c      accept this null hypothesis. Otherwise we
c      reject this null hypothesis.
c
if(netscaleddev.lt.cvalchi5) then
  print*, 'It is reasonable to assume exponentiality'
else
  print*, 'It is not reasonable to assume exponentiality'
endif
endif
stop
end

real function urand(iy)
integer iy

```

APPENDIX 02 - Fortran 77 (F77) Programs

```

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 arbitrary 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 *****
c      integer ia,ic,itwo,m2,m,mic
c      double precision halfm
c      real s
c      data m2/0/,itwo/2/
c
c      If first entry, compute machine integer word length
c      if(m2.eq.0) then
c          m=1
10  if(m.gt.m2) then
c          m2=m
c          m=itwo*m2
c          goto 10
c      endif
c      halfm=m2
c
c      Compute multiplier and increment for linear congruential method
c      ia=8*int(halfm*atan(1.d0)/8.d0)+5
c      ic=2*int(halfm*(0.5d0-sqrt(3.d0)/6.d0))+1
c      mic=(m2-ic)+m2
c
c      s is the scale factor for converting to floating point
c      s=0.5/halfm
c      endif
c
c      Compute next random number
c      iy=iy*ia
c
c      The following statement is for computers which do not allow
c      integer overflow on addition
c      if(iy.gt.mic) iy=(iy-m2)-m2
c      iy=iy+ic
c
c      The following statement is for computers where the word length
c      is greater than for multiplication
c      if(iy/2.gt.m2) iy=(iy-m2)-m2
c
c      The following statement is for computers where integer overflow
c      affects sign bit
c      if(iy.lt.0) iy=(iy+m2)+m2
c      urand=float(iy)*s
c      return
c      end

```

APPENDIX 02 - Fortran 77 (F77) Programs

Output from F77 Program 19

 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

APPENDIX 02 - Fortran 77 (F77) Programs

.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

APPENDIX 02 - Fortran 77 (F77) Programs

Sum of logs to the base e = -1.4003725E+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
2.57465	1.41836

Elements of the information matrix -
 ipp = .2042950E+03
 ipk = .1036976E+02
 ikk = .1145519E+04

Standard errors -
 rho = .6996343E-01
 kappa = .2954601E-01

APPENDIX 02 - Fortran 77 (F77) Programs

With 95% confidence -

rho is between .1281229E+01 and .1555486E+01
kappa is between .2516744E+01 and .2632654E+01

H0 : kappa = 1, for exponentiality
H1 : kappa <> 1, for non-exponentiality

Likelihood function under H0 = -.4033102E+02
Likelihood function under H1 = -.1062036E+02

Value of $l_1 - l_0$ = .2971066E+02

Change in scaled deviance = .9050021E+02
Chi-sq at 5% level on 1 df = .3841000E+01

It is not reasonable to assume exponentiality