

```

C
C ***** program wald.f *****
C
C general project      : estimate earliest time of survival differences
C specific problem     : uses Wald test and survival as response
C additional comments  : work with Dennis Boos and Walt Piegorsch
C
C latest editing date  : 06/15/91
C
C subprograms required : indata, mysort, atrisk, kme, var, tcomp, outest.
C =====
C
C      implicit real*8 (a-h,o-z)
C
C      dimension t(1200)
C      dimension c0(1200), c1(1200), d0(1200), d1(1200)
C      dimension r0(1200), r1(1200), s0(1200), s1(1200)
C      dimension v0(1200), v1(1200), z(1200)
C
C      data inunit, iout, iout2 / 21, 31, 32 /
C      data maxnj / 1200 /
C
C assign read and write unit numbers .....
C
C      open(unit=21, file='./MOR.txt')
C
C      open(unit=31, file='./wald.out')
C      open(unit=32, file='./wald_km_tab.out')
C
C read in the data .....
C
C      call indata ( maxnj, inunit, nj, t, c0, d0, c1, d1, r0, r1,
C      +           nd, n, ier )
C
C      if ( ier .gt. 0 ) go to 900
C
C compute Kaplan-Meier estimates .....
C
C      call kme ( nj, r0, d0, s0 )
C
C      call kme ( nj, r1, d1, s1 )
C
C compute the variance estimates .....
C
C      call var ( nj, r0, d0, s0, v0 )
C
C      call var ( nj, r1, d1, s1, v1 )
C
C output the results .....
C
C 900 continue
C
C      call outest ( iout, nd, n, nj, maxnj, t, s0, s1, v0, v1, z, ier )
C

```

```

c check to see if nj is even or odd .....
c
c     ntemp = 2*(nj/2)
c
c     if ( ntemp .eq. nj ) then
c       nn = nj
c     else
c       nn = nj + 1
c       t(nn) = t(nj)
c       s0(nn) = s0(nj)
c       v0(nn) = v0(nj)
c       s1(nn) = s1(nj)
c       v1(nn) = v1(nj)
c     end if
c
c     call outtab ( iout2, nn, t, s0, s1, v0, v1 )
c
c     stop
c     end
c
c *****      subroutine indata      *****
c
c this subroutine reads in the survival data.
c
c input:  maxnj, inunit
c
c output: nj, t, c0, d0, c1, d1, r0, r1, nd, n, ier
c
c subprograms: mysort, atrisk.
c =====
c
c     subroutine indata ( maxnj, inunit, nj, t, c0, d0, c1, d1, r0, r1,
c       +                nd, n, ier )
c
c     implicit real*8 (a-h,o-z)
c
c     dimension t(maxnj), c0(maxnj), d0(maxnj), c1(maxnj), d1(maxnj)
c     dimension r0(maxnj), r1(maxnj)
c
c read in the number of death times (nj) .....
c
c     read(inunit,*) nj
c
c     ier = 1
c     if ( nj .lt. 1 .or. nj .gt. maxnj ) return
c
c read in the rest of the data .....
c
c     nd = 0
c     n = 0
c
c     do 10 j = 1,nj
c
c     read(inunit,*) t(j), c0(j), d0(j), c1(j), d1(j)

```

```

c
c   nd = nd + d0(j) + d1(j)
c   n = n + d0(j) + d1(j) + c0(j) + c1(j)
c
c   ier = 2
c   if ( t(j) .lt. 0.d0 ) return
c
c   ier = 3
c   if ( c0(j) .lt. 0.d0 ) return
c
c   ier = 4
c   if ( c1(j) .lt. 0.d0 ) return
c
c   ier = 5
c   if ( d0(j) .lt. 0.d0 ) return
c
c   ier = 6
c   if ( d1(j) .lt. 0.d0 ) return
c
c 10  continue
c
c   ier = 0
c
c sort the death times into ascending order (and carry along other
arrays)
c
c   call mysort ( nj, t, c0, d0, c1, d1 )
c
c compute the numbers at risk in each group .....
c
c   call atrisk ( nj, c0, d0, r0 )
c
c   call atrisk ( nj, c1, d1, r1 )
c
c   return
c   end
c
c *****      subroutine mysort      *****
c
c this subroutine uses a bubble sort to sort the vector t into ascending
c order, while carrying along the vectors c0, d0, c1, and d1.
c
c input:  nj, t, c0, d0, c1, d1
c
c output: t, c0, d0, c1, d1
c
c subprograms required: none.
c =====
c
c   subroutine mysort ( nj, t, c0, d0, c1, d1 )
c
c   implicit real*8 (a-h,o-z)
c
c   dimension t(nj), c0(nj), d0(nj), c1(nj), d1(nj)

```

```

c
do 20 j = 2,nj
  iflag = 0
  max = nj - j + 2
c
  do 10 i = 2,max
    if ( t(i) .lt. t(i-1) ) then
c
      temp = t(i-1)
      t(i-1) = t(i)
      t(i) = temp
c
      temp = c0(i-1)
      c0(i-1) = c0(i)
      c0(i) = temp
c
      temp = d0(i-1)
      d0(i-1) = d0(i)
      d0(i) = temp
c
      temp = c1(i-1)
      c1(i-1) = c1(i)
      c1(i) = temp
c
      temp = d1(i-1)
      d1(i-1) = d1(i)
      d1(i) = temp
c
      iflag = 1
    end if
  10 continue
c
  if ( iflag .eq. 0 ) return
  20 continue
c
  return
end

c
c ***** subroutine atrisk *****
c
c this subroutine computes the numbers at risk.
c
c input:  nj, c, d
c
c output: r
c
c subprograms required: none.
c =====
c
c   subroutine atrisk ( nj, c, d, r )
c
c   implicit real*8 (a-h,o-z)
c
c   dimension c(nj), d(nj), r(nj)

```

```

c
c   risk = 0.d0
c
c   do 10 jj = 1,nj
c     j = nj - jj + 1
c     risk = risk + c(j) + d(j)
c     r(j) = risk
10  continue
c
c   return
c   end

c
c *****      subroutine kme      *****
c
c this subroutine computes the Kaplan-Meier estimates of the survival
c curves.
c
c input:  nj, r, d
c
c output: s
c
c subprograms required: none.
c =====
c
c   subroutine kme ( nj, r, d, s )
c
c   implicit real*8 (a-h,o-z)
c
c   dimension r(nj), d(nj), s(nj)
c
c   prod = 1.d0
c   slast = 1.d0
c
c   do 10 j = 1,nj
c
c     if ( r(j) .gt. 0.d0 ) then
c       pj = 1.d0 - d(j)/r(j)
c       prod = prod*pj
c       s(j) = prod
c
c     else
c       s(j) = slast
c     end if
c
c   slast = s(j)
10  continue
c
c   return
c   end

c
c *****      subroutine var      *****
c
c this subroutine computes the estimated variances of the Kaplan-Meier
c survival estimates.

```

```

c
c input:  nj, r, d, s
c
c output: v
c
c subprograms required: none.
c =====
c
c      subroutine var ( nj, r, d, s, v )
c
c      implicit real*8 (a-h,o-z)
c
c      dimension r(nj), d(nj), s(nj), v(nj)
c
c      sum = 0.d0
c
c      do 10 j = 1,nj
c
c      if ( r(j) .gt. d(j) ) then
c      sum = sum + d(j)/( r(j)*( r(j) - d(j) ) )
c      v(j) = s(j)*s(j)*sum
c
c      else
c
c      if ( r(j) .gt. 0.d0 ) then
c      v(j) = 0.d0
c      else
c      v(j) = v(j-1)
c      end if
c
c      end if
c
c 10  continue
c
c      return
c      end
c
c *****      subroutine tcomp      *****
c
c this subroutine computes theta-hat
c
c input:  nj, zalpha, t, z
c
c output: theta
c
c subprograms required: none.
c =====
c
c      subroutine tcomp ( nj, zalpha, t, z, theta )
c
c      implicit real*8 (a-h,o-z)
c
c      dimension t(nj), z(nj)
c

```

```

        if ( z(nj) .gt. zalpha ) then
            tmax = -1.d0
            zlast = 0.d0
c
            do 10 j = 1,nj
                if ( zlast .le. zalpha .and. t(j) .gt. tmax ) tmax = t(j)
                zlast = z(j)
10          continue
c
            theta = tmax
c
            else
                theta = -999.d0
            end if
c
            return
            end
c
c *****      subroutine outest      *****
c
c this subroutine outputs the mle's.
c
c input:  iout, nd, n, nj, maxnj, t, s0, s1, v0, v1, ier
c
c subprograms required: tcomp.
c =====
c
c      subroutine outest ( iout, nd, n, nj, maxnj, t, s0, s1, v0, v1, z,
+
c          ier )
c
c      implicit real*8 (a-h,o-z)
c
c      character*6 iblank, istar, itheta(2)
c      character*6 istar1, istar2, istar3, istar4, istar5, istar6
c
c      dimension t(nj), s0(nj), s1(nj), v0(nj), v1(nj), z(nj)
c
c      data itheta / 'No Sta','tement' /
c
c output the date and program name .....
c
c      call idate ( imonth, iday, iyear )
c
c      write(iout,5) imonth, iday, iyear
5      format(1x,'Source: wald.f                Date: ',i2,'/',i2,'/',i2,
+
c          '                Output: wald.out')
c
c output any error messages .....
c
c      if ( ier .ne. 0 ) then
c          if ( ier .eq. 1 ) write(iout,10) nj, maxnj
10          format(//,2x,'nj = ',i5,' , which is .lt. 1 or .gt. maxnj =
',i5)
c

```

```

        if ( ier .eq. 2 ) write(iout,20)
20      format(//,2x,'t(j) is .lt. 0 for some j')
c
        if ( ier .eq. 3 ) write(iout,30)
30      format(//,2x,'c0(j) is .lt. 0 for some j')
c
        if ( ier .eq. 4 ) write(iout,40)
40      format(//,2x,'c1(j) is .lt. 0 for some j')
c
        if ( ier .eq. 5 ) write(iout,50)
50      format(//,2x,'d0(j) is .lt. 0 for some j')
c
        if ( ier .eq. 6 ) write(iout,60)
60      format(//,2x,'d1(j) is .lt. 0 for some j')
      end if
c
c output general information .....
c
      write(iout,110) nd, n
110    format(///,2x,'Number of natural deaths   = ',i5,
+          /,2x,'Number of animals on study = ',i5)
c
c initialize variables .....
c
      zero = 0.d0
      one  = 1.d0
c
      z1 = 1.282
      z2 = 1.645
      z3 = 1.960
      z4 = 2.326
      z5 = 2.576
      z6 = 3.090
c
      iblank = '      '
      istar1 = '*      '
      istar2 = '**     '
      istar3 = '***    '
      istar4 = '****   '
      istar5 = '***** '
      istar6 = '*****'
c
c write out time-specific estimates .....
c
      write(iout,120)
120    format(//,2x,'
+          /,2x,'
+          /,2x,' j   t(j)
score',
+          /,2x,'___  ___  ___  ___  ___  ___  ___
_____')
c
      write(iout,130) zero, one, one, zero, zero
130    format(/,4x,f7.0,3x,f6.3,1x,f6.3,3x,f8.5,1x,f8.5)

```

```

c
do 150 j = 1,nj
tj = t(j)
c
denom = dsqrt( v0(j) + v1(j) )
c
if ( denom .gt. 0.d0 ) then
zj = ( s0(j) - s1(j) )/denom
c
else
zj = -999.9999
end if
c
istar = iblank
c
if ( zj .gt. z1 ) istar = istar1
if ( zj .gt. z2 ) istar = istar2
if ( zj .gt. z3 ) istar = istar3
if ( zj .gt. z4 ) istar = istar4
if ( zj .gt. z5 ) istar = istar5
if ( zj .gt. z6 ) istar = istar6
c
write(iout,140) j, tj, s0(j), s1(j), v0(j), v1(j), zj, istar
140 format(1x,i3,f7.0,3x,f6.3,1x,f6.3,3x,f8.5,1x,f8.5,2x,f9.4,3x,a6)
c
z(j) = zj
150 continue
c
c compute theta-hats for various alpha levels .....
c
call tcomp ( nj, z1, t, z, theta1 )
c
call tcomp ( nj, z2, t, z, theta2 )
c
call tcomp ( nj, z3, t, z, theta3 )
c
call tcomp ( nj, z4, t, z, theta4 )
c
call tcomp ( nj, z5, t, z, theta5 )
c
call tcomp ( nj, z6, t, z, theta6 )
c
c output the estimates of theta .....
c
write(iout,200)
200
format(2x,'_____','//)
c
if ( theta1 .ge. 0.d0 ) then
write(iout,210) theta1
210 format(2x,' * ==> Z > 1.282 (P < .100) Theta-hat
=' ,f6.0)
else
write(iout,215) itheta

```

```

215 format(2x,'      *   ==>  Z > 1.282   (P < .100)   ',2a6)
    end if
c
    if ( theta2 .ge. 0.d0 ) then
    write(iout,220) theta2
220 format(2x,'      **   ==>  Z > 1.645   (P < .050)   Theta-hat
=',f6.0)
    else
    write(iout,225) itheta
225 format(2x,'      **   ==>  Z > 1.645   (P < .050)   ',2a6)
    end if
c
    if ( theta3 .ge. 0.d0 ) then
    write(iout,230) theta3
230 format(2x,'      ***  ==>  Z > 1.960   (P < .025)   Theta-hat
=',f6.0)
    else
    write(iout,235) itheta
235 format(2x,'      ***  ==>  Z > 1.960   (P < .025)   ',2a6)
    end if
c
    if ( theta4 .ge. 0.d0 ) then
    write(iout,240) theta4
240 format(2x,'     ****  ==>  Z > 2.326   (P < .010)   Theta-hat
=',f6.0)
    else
    write(iout,245) itheta
245 format(2x,'     ****  ==>  Z > 2.326   (P < .010)   ',2a6)
    end if
c
    if ( theta5 .ge. 0.d0 ) then
    write(iout,250) theta5
250 format(2x,'     ***** ==>  Z > 2.576   (P < .005)   Theta-hat
=',f6.0)
    else
    write(iout,255) itheta
255 format(2x,'     ***** ==>  Z > 2.576   (P < .005)   ',2a6)
    end if
c
    if ( theta6 .ge. 0.d0 ) then
    write(iout,260) theta6
260 format(2x,'     ***** ==>  Z > 3.090   (P < .001)   Theta-hat
=',f6.0)
    else
    write(iout,265) itheta
265 format(2x,'     ***** ==>  Z > 3.090   (P < .001)   ',2a6)
    end if
c
    return
    end
c
c ***** subroutine outtab *****
c
c this subroutine outputs the info for a table of survival estimates.

```

```

c
c input:  iout2, nn, t, s0, s1, v0, v1
c
c subprograms required: none.
c =====
c
c      subroutine outtab ( iout2, nn, t, s0, s1, v0, v1 )
c
c      implicit real*8 (a-h,o-z)
c
c      dimension t(nn), s0(nn), s1(nn), v0(nn), v1(nn)
c
c write out time-specific estimates .....
c
c      n2 = nn/2
c
c      do 20 j = 1,n2
c        j1 = j
c        j2 = n2 + j
c
c        itj1 = t(j1)
c        itj2 = t(j2)
c
c        v0j1 = 1000.d0*v0(j1)
c        v0j2 = 1000.d0*v0(j2)
c
c        v1j1 = 1000.d0*v1(j1)
c        v1j2 = 1000.d0*v1(j2)
c
c        write(iout2,10) j1, itj1, s0(j1), v0j1, s1(j1), v1j1,
+          j2, itj2, s0(j2), v0j2, s1(j2), v1j2
c
c 10  format(2x,i2,1x,i3,1x,f4.2,1x,'(,f4.2,)',1x,f4.2,1x,'(,f4.2,)',
+        3x,i2,1x,i3,1x,f4.2,1x,'(,f4.2,)',1x,f4.2,1x,'(,f4.2,)',)
c
c 20  continue
c
c      return
c      end

```