Example of using R functions related to emplik (empirical likelihood) package

Example 1. Find the upper and lower confidence limit of
a Wilks confidence interval related to the Kaplan-Meier estimator

We can use the function el.cen.EM from the emplik package to accomplish this.
In fact el.cen.EM is designed to test the hypothesis (via likelihood ratio test)

Ho: \int g(t) dF(t) = mu; vs. Ha: \int g(t) dF(t) != mu

where g(t) is a user supplied function. Depending on the g(t) we supply,
this hypothesis can be the test about the Kaplan-Meier mean [when g(t) = t],
or about the survival probability at time t0 [when g(t) = I[t <= t0]]
or many others.

Let us see a real example: we use the myeloma data set that comes with emplik package

```
data(myeloma)
survtimes <- myeloma[,1]
censtatus <- myeloma[,2]
```

We shall test and construct confidence interval for F(10), so define

```
myfun1 <- function(t){ as.numeric(t <= 10) }
```

Now a test based on the empirical likelihood ratio for Ho: F(10) = 0.2

```
el.cen.EM(fun=myfun1, x=survtimes, d=censtatus, mu=0.2)
```

among the output, you see a Pval (that is the P-value) of 0.3103897
so the conlusion is "not to reject the null hypothesis of F(10)=0.2".

If you want the (NPMLE) estimate of F(10), look in the output for "funMLE"
You will see a MLE of 0.2526136 (this is just the Kaplan-Meier at t= 10)

There is another function el.cen.EM2 in the emplik package
which is very similar. The function el.cen.EM is designed for a single parameter, and el.cen.EM2 can handle multiple parameters.
el.cen.EM may be a bit faster, while el.cen.EM2 may be a bit more

```
robust.
  For the subtle differences, please see the help pages in R.

  Now we can further find the 95% confidence interval for F(10). For
this purpose
  we need to define a function that just returns the -2LLR. The first
input of this
  function must be the value to be tested.

myloglikR1 <- function(mu0, x, d) {
  myfun1 <- function(t){as.numeric(t<=10)}
  el.cen.EM(fun=myfun1, x=x, d=d, mu=mu0)$"-2LLR"
}

      The real search for the lower(upper) confidence bound is done by
the
      function findL (findU)

findL(fun=myloglikR1, MLE=0.25, x=survtimes, d=censtatus)
[1] 0.1566809230 0.0000000001 3.8399999183


findU(fun=myloglikR1, MLE=0.25, x=survtimes, d=censtatus)
[1] 0.3684037790 0.0000000001 3.8399999664

    There are 3 output values, the first one is the lower(upper)
confidence bound,
    the second/third is the final step value and the final -2LLR value.
So the 95%
    confidence interval for F(10) is [0.156680923, 0.368403779]

        If you want the 90% confidence interval instead, do this

findL(fun=myloglikR1, MLE=0.25, x=survtimes, d=censtatus, level=
2.705543)

findU(fun=myloglikR1, MLE=0.25, x=survtimes, d=censtatus, level=
2.705543)



   Finally, the functions findL and findU is here: (not inside the
emplik package yet)


findU <- function(step=0.1, initStep=0, fun, MLE=0, level=3.84, ...) {
value <- 0
Ubeta <- MLE + initStep
for( i in 1:8 ) {
   while(value < level) {
    Ubeta <- Ubeta + step
    value <- fun(Ubeta, ... )
   }
   Ubeta <- Ubeta - step
```

```
   step <- step/10
   value <- fun(Ubeta, ... )
}
return( c(Ubeta, step, value) )
}


> findL
function(step=0.1, initStep=0, fun, MLE=0, level=3.84, ...) {
#### This function try to find the 1 dim confidence interval
#### (the lower confidence limit only)
#### the fun should be the -2loglik function with 1 parameter
#### (the other parameters needs to be maxed over)
#### The initStep is provided to save some initial steps, i.e. if you
know
#### approximate where the confidence bound is, (how far it is from the
MLE)
#### then you can start search from there, instead of starting from MLE.
#### You should always use a lower bound for the initStep (a
conservative value)
#### as the search is from inside the confidence interval outwards.
value <- 0
Lbeta <- MLE - initStep
for( i in 1:8 ) {
while(value < level) {
Lbeta <- Lbeta - step
value <- fun(Lbeta, ... )
}
Lbeta <- Lbeta + step
step <- step/10
value <- fun(Lbeta,  ... )
}
return( c(Lbeta, step, value) )
}
```

==========

In the above example, the Wald confidence interval can at least be computed
easily, with the Greenwood formula to estimate the variance. The next example
then is almost impossible to get a Wald confidence interval. But the Wilks
confidence interval is as easy as in example one.

Example 2. Find the upper and lower confidence limit of the residual median, where
the estimator is based on the Kaplan-Meier estimator.
[residual median? or median residual?]

First the definition: The residual median at a given time t0 is the number

"theta" that solve the equation

$$\frac{1-F(t_0+\theta)}{1-F(t_0)} = 0.5$$

The estimator of theta can be obtained by (a) replace F by the Kaplan-Meier
in the above and (b) solve for theta.

Notice the variance of the estimator of theta is very complicated, and even
if you get a theoretical expression for it, it is of not much use, due to the
fact that it involves density function, a quantity hard to estimate.

To use the Wilks method, we shall re-cast the problem into a testing hypothesis
problem.

Notice that a theta solves the above equation iff it solves

$$\int g_{\theta}(t)\, dF(t) = 0.5$$

where $g_{\theta}(t) = I[t <= t_0+\theta] - 0.5*I[t <= t_0]$
(simple algebraic manipulations)

and recall this hypothesis can be handled by the el.cen.EM function.

R-code:

```
library(survival)
data(cancer)
library(emplik)
cancertimes <- cancer$time
cancerstatus <- cancer$status -1
```

This estimates the median and mean residual time at t0=365.25 days

```
MMRtime(x=cancertimes, d=cancerstatus, age=365.25)
$MeanResidual
[1] 275.9997

$MedianResidual
[1] 258.75
```

Now the test that theta = 258

```
mygfun22 <- function(s, age, Mdage) {as.numeric(s<=(age+Mdage))-0.5
*as.numeric(s<=age)}
```

```
el.cen.EM(x=cancertimes, d=cancerstatus, fun=mygfun22, mu=0.5, age=
365.25, Mdage=258)$"-2LLR"


              If we are lazy, we can just repeat the last line but
change 258 to
              something else, until you get "-2LLR" as close to as
possible (but not
              exceed 3.84) to find the 95% confidence interval. Or do it
automatically
              as follows.

               To find the upper and lower confidence limit by
findL/findU, we
               need to write a function that returns the "-2LLR" and the
first
               input variable is the value to be tested.


myloglikR2 <- function(mu0, x, d) {
  mygfun22 <- function(s, age=365.25, Mdage=mu0){as.numeric(s<=
(age+Mdage))-0.5*as.numeric(s<=age)}
  el.cen.EM(x=x, d=d, fun=mygfun22, mu=0.5)$"-2LLR"
}

           Finally, we may do this (but it took too long...slow search
algorithm
           the default search step is too small.)
 takes too long:   findL(fun=myloglikR2, MLE=258, x=time, d=status,
level=2.705543)



           Better to do this (with bigger steps, search is a bit faster)
findL(step=5,fun=myloglikR2, MLE=258, x=time, d=status, level=2.705543)
[1] 184.75000000   0.00000005   2.50369332
findU(step=5,fun=myloglikR2, MLE=258, x=time, d=status, level=2.705543)
[1] 321.75000000   0.00000005   2.42779336


          So, The 90% confidence interval is [ 184.750, 321.750 ].

        If the step is too large, the hypothesis to be tested may be out
of
        the range for el.cen.EM to handle and may stop computing or too
slow.
        (here use el.cen.EM2 may be more robust)

    Notice the final level reached is 2.503 or 2.427 and somewhat away
from 2.7055.
    THis is normal, since this is a discrete case when testing the
median.
    (or any quantiles)
```

==========

Example 3:

The folowing is another example of how the code findL and findU works with parametric MLE: (the parametric regression of survreg( ) and getting the confidence interval for the slope in the regression)

```
> findL(fun=loglikfun, MLE=-1.255066, y=myel$dur, d=myel$status, x1=myel
$treat, x2=myel$renal)
[1] -2.7499611050  0.0000000001  3.8399999981
```

So, the lower 95% confidence bound is -2.74996

The two functions involved: findL is general (as above), loglikfun is data specific.
The MLE value can be obtained by first run a survreg, and getting the estimate,
it is not critical, any value near -1.255 will work.

```
> loglikfun
function(beta, y, d, x1, x2) {
loglikmax <- survreg(Surv(y,d)~ x1 + x2)$loglik[2]
loglikH0 <- survreg(Surv(y,d)~ offset(beta*x1)+ x2)$loglik[2]
return( -2*(loglikH0 - loglikmax) )
}
```

Assumeing of course the data frame myel is already loaded there.

==========

Now getting the interval for both regression slopes:

```
> findU(fun=loglikfun2, MLE=-4.18, level=2.7, y=myel$dur, d=myel$status,
x1=myel$treat, x2=myel$renal)
[1] -3.1652483440  0.0000000001  2.6999999991
> findL(fun=loglikfun2, MLE=-4.18, level=2.7, y=myel$dur, d=myel$status,
x1=myel$treat, x2=myel$renal)
[1] -5.2571687790  0.0000000001  2.6999999965
>
> findU(fun=loglikfun, MLE=-1.2, level=2.7, y=myel$dur, d=myel$status,
x1=myel$treat, x2=myel$renal)
[1] -0.2800085190  0.0000000001  2.6999999965
> findL(fun=loglikfun, MLE=-1.2, level=2.7, y=myel$dur, d=myel$status,
```

```
x1=myel$treat, x2=myel$renal)
[1] -2.4572724310  0.0000000001  2.6999999991
>



> loglikfun
function(beta0, y, d, x1, x2) {
loglikmax <- survreg(Surv(y,d)~ x1 + x2)$loglik[2]
loglikH0 <- survreg(Surv(y,d)~ offset(beta0*x1)+ x2)$loglik[2]
return( -2*(loglikH0 - loglikmax) )
}
>
> loglikfun2
function(beta0, y, d, x1, x2) {
loglikmax <- survreg(Surv(y,d)~ x1 + x2)$loglik[2]
loglikH0 <- survreg(Surv(y,d)~ x1 + offset(beta0*x2))$loglik[2]
return( -2*(loglikH0 - loglikmax) )
}
>
```

==========

Example 4. Test the equality of two medians with right censored data.

The emplik package contains a function ROCnp( ) that can be used to test the equality of two medians from two independent samples.

```
library(survival)
times1 <- cancer$time[cancer$sex==1]
status1 <- cancer$status[cancer$sex==1] -1
times2 <- cancer$time[cancer$sex==2]
status2 <- cancer$status[cancer$sex==2] -1

ROCnp(t1=times1, d1=status1, t2=times2, d2=status2, b0=0.5, t0=0.5)
$`-2LLR`
[1] 11.02725

$cstar
[1] 312.105
```

which lead us to reject the null hypothesis of equal median (-2LLR larger than 3.84)

If we want to construct a confidence interval for the RATIO of the two medians, we need to work harder. (no existing code) But the idea is similar:

Let us test the hypothesis that m1/m2 = 0.6 where m1 is the median for male

and m2 is the median for female. This is same as testing
   m1 = 0.6*m2 = a   for some a, or m1=a, 0.6*m2=a; or m1=a, m2=a/0.6
.

We shall modify the code of ROCnp to accomplish this.

myROCnp(t1, d1, t2, d2, b0=0.5, t0=0.5)

```
> myROCnp
function (t1, d1, t2, d2, b0, t0)
{
    if (length(b0) != 1)
        stop("check length of b0")
    if (length(t0) != 1)
        stop("check length of t0")
    if (b0 >= 1 | b0 <= 0)
        stop("check the value of b0")
    if (t0 >= 1 | t0 <= 0)
        stop("check the value of t0")
    tempnp2 <- WKM(x = t2, d = d2)
    place2 <- sum(tempnp2$surv >= t0)
    c2 <- tempnp2$times[place2]
    tempnp1 <- WKM(x = t1, d = d1)
    place1 <- sum(tempnp1$surv > b0)
    c1 <- tempnp1$times[place1]
    if (c2 <= c1)
        c1 <- tempnp1$times[place1 + 1]
    llr <- function(const, t1, d1, t2, d2, b0, t0) {
        npllik1 <- el.cen.EM2(x = t1, d = d1, fun = function(x,
            theta) {
            as.numeric(x <= theta)
        }, mu = 1 - b0, theta = const)$"-2LLR"
        npllik2 <- el.cen.EM2(x = t2, d = d2, fun = function(x,
            theta) {
            as.numeric(x <= theta/0.6)              #### here is the
ratio
        }, mu = 1 - t0, theta = const)$"-2LLR"
        return(npllik1 + npllik2)
    }
    temp <- optimize(f = llr, lower = min(c2, c1), upper = max(c2,
        c1), t1 = t1, d1 = d1, t2 = t2, d2 = d2, b0 = b0, t0 = t0)
    cstar <- temp$minimum
    val <- temp$objective
    list(`-2LLR` = val, cstar = cstar)
}
```

==========

Due to the discreteness of median, the optimize( ) function may not
always
find the min. An exhaust search can be done: [slower but always find the
min]

```
ROCnp2 <- function (t1, d1, t2, d2, b0, t0)
{
    if (length(b0) != 1)
        stop("check length of b0")
    if (length(t0) != 1)
        stop("check length of t0")
    if (b0 >= 1 | b0 <= 0)
        stop("check the value of b0")
    if (t0 >= 1 | t0 <= 0)
        stop("check the value of t0")
    tempnp2 <- WKM(x = t2, d = d2)
    place2 <- sum(tempnp2$surv >= t0)
    c2 <- tempnp2$times[place2]
    tempnp1 <- WKM(x = t1, d = d1)
    place1 <- sum(tempnp1$surv > b0)
    c1 <- tempnp1$times[place1]
    if (c2 <= c1)
        {c1 <- tempnp1$times[place1 + 2]
         c2 <- tempnp2$times[place2 - 1] }
    llr <- function(const, t1, d1, t2, d2, b0, t0) {
        npllik1 <- el.cen.EM(x = t1, d = d1, fun = function(x,
            theta) {
            as.numeric(x <= theta)
        }, mu = 1 - b0, theta = const)$"-2LLR"
        npllik2 <- el.cen.EM(x = t2, d = d2, fun = function(x,
            theta) {
            as.numeric(x <= theta)
        }, mu = 1 - t0, theta = const)$"-2LLR"
        return(npllik1 + npllik2)
    }
    lower <- min(c1, c2)
    upper <- max(c1, c2)
    timesALL <- c(tempnp2$times, tempnp1$times)
    midpts <- timesALL[(timesALL>lower) & (timesALL<upper)]
    midpts <- sort(midpts)
    midpts <- (midpts[-1] + midpts[-length(midpts)])/2
    k <- length(midpts)
    val2 <- rep(-9, k)
    for(i in 1:k) val2[i] <- llr(const=midpts[i], t1=t1, d1=d1, t2=t2,
d2=d2, b0=b0, t0=t0)
    val <- min(val2)
    cstar <- midpts[ which(val2==val) ]
    list(`-2LLR` = val, cstar = cstar)
}
```

==========

A similar but easier problem: test the equality of the survival probabilities
at a given time t0; based on two independent samples.
The problem is cont. and well behaved. The minimum is easy to find.

Since the code is similar to ROCnp, I call it ROCnp3 [lack of a better

name??]

```
ROCnp3 <- function (t1, d1, t2, d2, t0)
{
    tempnp2 <- WKM(x = t2, d = d2)
    place2 <- sum(tempnp2$times <= t0)
    c2 <- tempnp2$surv[place2]
    tempnp1 <- WKM(x = t1, d = d1)
    place1 <- sum(tempnp1$times <= t0)
    c1 <- tempnp1$surv[place1]
    llr <- function(const, t1, d1, t2, d2, t0) {
        npllik1 <- el.cen.EM(x = t1, d = d1, fun = function(x) {
            as.numeric(x <= t0)
        }, mu = const)$"-2LLR"
        npllik2 <- el.cen.EM(x = t2, d = d2, fun = function(x) {
            as.numeric(x <= t0)
        }, mu = const)$"-2LLR"
        return(npllik1 + npllik2)
    }
    lower <- min(1-c1, 1-c2)
    upper <- max(1-c1, 1-c2)
  temp <- optimize(f=llr, lower=lower, upper=upper, t1=t1, d1=d1, t2=t2,
d2=d2, t0=t0)
  cstar <- temp$minimum
  val <- temp$objective
    list(`-2LLR` = val, cstar = cstar, Prob1=1-c1, Prob2=1-c2)
}
```

  The code runs like this: first I assume we took the cancer data set from
  package survival. take the male and female as two samples. The resulting
  -2LLR should be compared to chisq(df=1) so we reject the Ho of equal prob
  at t0=200 days

```
> ROCnp3(t1=times1, d1=status1, t2=times2, d2=status2, t0=200)
$`-2LLR`
[1] 8.913334

$cstar
[1] 0.3200662

$Prob1
[1] 0.3926928

$Prob2
[1] 0.2054065
```

==========

Some thoughts when testing the ratio of two probabilities or medians.

If the actual ratio is further away from 1 then the value being tested (as in Ho), then we may search within the two prob/medians.
Since we have to push the two together to make them equal or having the ratio stipulated in Ho.
(i.e. if the two MLE's has ratio 0.5, and we are testing if the ratio is equal to 0.6 Or if the two MLE's has ratio 1.4 and we are testing Ho ratio = 1.2)
We know where to search the common value.

When the value tested (in Ho) is further away from 1 compared to the ratio of MLE's, then we must search out side the interval formed by two MLE's. How much outside, will depends on the actual value to be tested.
If we are testing ratio equal to 2, then we should search from value a and value b, where a = larger of MLE/2, b= smaller of MLE x2.
etc.

==========

A final note: When define median, it may be better to use a "smoothed" indicator
function.

Due to discreteness nature of the Kaplan-Meier estimator. The median may be non-unique. (even after the smoothing)

Here is the indicator function and 2 smoothed version of it.

```
myfun6 <- function(x, theta) {as.numeric(x <= theta)}


myfun7 <- function(x, theta=0, eps=0.2) {
if(eps <= 0) stop("eps must > 0")
u <- (x-theta)/eps
return( pmax(0, pmin(1-u, 1)) )
}

myfun5 <- function(x, theta=0, eps=0.1) {
if(eps <= 0) stop("eps must > 0")
u <- (x-theta)*sqrt(5)/eps
INDE <- (u < sqrt(5)) & (u > -sqrt(5))
u[u >= sqrt(5)] <- 0
u[u <= -sqrt(5)] <- 1
y <- 0.5 - (u - (u)^3/15)*3/(4*sqrt(5))
u[ INDE ] <- y[ INDE ]
return(u)
}
```