Examples of Constructing Confidence Interval by Likelihood Ratio Test

Mai Zhou

Department of Statistics
University of Kentucky
Lexington, KY 40536

SUMMARY

These notes details several examples that help us understand the technique of profiling the (either regular or empirical) likelihood. The likelihood is then used to produce (Wilks) confidence intervals. This provides more details to a technique used in the book *Empirical Likelihood Methods in Survival Analysis* (Zhou, 2016), and showcase the capability of some R functions inside packages `emplik`.

AMS 2000 Subject Classification: Primary 60E15; secondary 60G30.
Key Words and Phrases: Empirical likelihood, Confidence region and interval.

1 Confidence Intervals Based on the Likelihood Ratio Test[1]

1. Let's consider the binomial case. Let X be a random variable distributed as a Binomial(n, p). Then, we know that the maximum likelihood estimator of p will be $\hat{p} = \frac{x}{n}$. And, from our introductory statistics courses, we know that we can calculate a confidence interval for p by the following formula for a Wald confidence interval:

$$\hat{p} \pm z\sqrt{\frac{\hat{p}\left(1-\hat{p}\right)}{n}} \tag{1}$$

One issue with this type of confidence interval is that it's possible for it to contain values outside of the interval $(0, 1)$, which doesn't make any sense. There are some known transformations which eliminate (or reduce the likelihood) of this possibility. So, let's consider an arbitrary (monotone) transformation $g\left(p\right)$ and construct a confidence interval for it. From the formula for a Wald confidence interval, we have:

$$g\left(\hat{p}\right) \pm z\sqrt{Var\left(g\left(\hat{p}\right)\right)} \tag{2}$$

where we used the fact that the MLE is invariant, so we know $\widehat{g\left(p\right)} = g\left(\hat{p}\right)$. And, by the delta method, we have:

$$Var\left(g\left(\hat{p}\right)\right) \approx \left[g'(p)\right]^2 Var\left(\hat{p}\right) \tag{3}$$

[1]These notes were obtained using emplik version 1.0-1

This gives us the confidence interval for $g(p)$: $g(p) \in [L, U]$. Which yields the following confidence interval for p: $p \in [g^{-1}(L),\ g^{-1}(U)]$.

So, why do we use transformations and why are they useful?
One reason: If you use a "good" transformation, then you can guarantee that the confidence interval for p will be between $(0, 1)$.

But, there exist many different "good" transformations. Which one is the best? (kind of arbitrary, in terms of accurate, speed, etc.).
The selection of the best "good" transformation is somewhat arbitrary. Your decision could be based on accuracy, speed, or any other criteria that you deem important. An example of this arbitrariness is the discrepancy between the transformation used by SAS and that used by R (in their confidence intervals based on the Kaplan-Meier for survival probability). In SAS, they use $\log(-\log(1-p))$, while R uses $\log(1-p)$.

2. Hypothesis Testing
Let's consider testing the following hypotheses:

$$H_0 : p = p_0 \tag{4}$$
$$H_1 : p \neq p_0 \tag{5}$$

Using the likelihood ratio test, we would reject the null when:

$$-2 \times \log\left[\frac{likelihood(x, p_0)}{likelihood(x, \hat{p})}\right] > C = \chi^2_{1,0.95} \tag{6}$$

We can simplify the left-hand side (without the -2 and took p for p_0) of this inequality to:

$$x\log(p) + (n-x)\log(1-p) - x\log(\hat{p}) - (n-x)\log(1-\hat{p}) \tag{7}$$
$$= x\log(\frac{p}{\hat{p}}) + (n-x)\log(\frac{1-p}{1-\hat{p}}) \tag{8}$$

Now, there are two conditions the we need to take into consideration:
a. $p \neq 0$ and $p \neq 1$
b. $\hat{p} \neq 0$ and $\hat{p} \neq 1$
We can write a function in R which calculates this value for us:

```
BinLLR = function(n,x,p){

if(x >= n) stop("100 percent success?")
if(p <= 0) stop("P must between 0 and 1")
if(x <= 0) stop("no seccess?")
if(p >= 1) stop("P must < 1")

phat = x/n
```

```
-2*(x*log(p/phat)+(n-x)*log((1-p)/(1-phat)))
}

ts = BinLLR(n=100,x=30,p=0.5)
1 - pchisq(ts,df=1)

## [1] 4.977722e-05
```

This show the P-value is near zero in testing p=0.5. When we test various values of success probabilities, those that lead to a P-value larger than 0.05 form a 95% confidence interval for p. This may be accomplished by the function findUL( ) from package `emplik`. Which gives us a confidence interval of [0.2160426, 0.3940769].

3.
```
myfun <- function(theta, n, x){
temp <- BinLLR(n=n, x=x, p=theta)
list("-2LLR"=temp)
}

library(emplik)
## Warning: package 'emplik' was built under R version 3.2.2
## Loading required package: quantreg
## Warning: package 'quantreg' was built under R version 3.2.2
## Loading required package: SparseM
## Warning: package 'SparseM' was built under R version 3.2.2
## Loading required package: methods
##
## Attaching package: 'SparseM'
## The following object is masked from 'package:base':
##
##    backsolve

findUL(fun=myfun, MLE=0.3, n=100, x=30)

## $Low
## [1] 0.2160426
##
## $Up
## [1] 0.3940769
##
## $FstepL
## [1] 1e-10
##
```

```
## $FstepU
## [1] 1e-10
##
## $Lvalue
## [1] 3.84
##
## $Uvalue
## [1] 3.84
```

4. Good Properties of the Likelihood Ratio Test Approach to Confidence Intervals

   (a) The LRT method is transformation invariant

      i. Monotone Transformations
      If you have a confidence interval for p, ($lower$, $upper$), and you want a confidence interval for $\log(p)$ (or any monotone function), you can find it by taking the log of the lower and upper bounds for p, ($\log(lower)$, $\log(upper)$). This is much easier than the Wald method which involves calculating derivatives, using the delta method, etc.

      ii. Transformations Which Aren't Monotone
      This property even holds for transformations which aren't monotone. If you have a transformation $g(p)$ which isn't monotone and a confidence interval for p, (l, u) is available, then you can find your transformed confidence interval for$g(p)$ by the following steps:

$$upper_{new} = \max_{l<p<u} g(p)$$
$$lower_{new} = \min_{l<p<u} g(p)$$

   (b) The LRT method is range preserving
   This means that the confidence intervals it produces will always be inside of the parameter space. So, in the binomial case, the confidence intervals produced by the LRT method will always be between $(0, 1)$.

   (c) You do not have to compute/estimate the variance of the estimator (usually the MLE).

   (d) The LRT method doesn't necessarily give symmetric intervals
   The asymmetry of the intervals is more similar to the actual distributions under consideration. So, in the binomial case when we have a small value of p, we know that there are more values above p than below it, so we'd expect our interval to reflect this. The Wald interval, on the other hand, will always be symmetric which we know is not always true of the distributions under consideration.

5. Possible Drawbacks of the Likelihood Ratio Test Approach to Confidence Intervals

   (a) You may not have to use the delta method or take any derivatives, but you still have to find maximums and roots. These calculations should be easily done using computers, so this isn't too much of an issue.

2 The Ratio of Two Success Probabilities

Let's consider the case where we have two samples, each containing a parameter: success probability. From each sample we will have an n_i and an x_i, where n_i is the total number of units observed in the i^{th} sample and x_i is the total number of successes observed in the i^{th} sample. We can model this situation by two binomial random variables, X_1 and X_2, where $X_1 \sim Binomial\,(n_1, p_1)$ and $X_2 \sim Binomial\,(n_2, p_2)$, where the MLE for p_1 is $\widehat{p_1} = \frac{x_1}{n_1}$ and the MLE for p_2 is $\widehat{p_2} = \frac{x_2}{n_2}$.

Now, let's suppose that we want to use the likelihood ratio test method of creating a confidence interval for the ratio of $\frac{p_1}{p_2}$. Since maximum likelihood estimators are invariant, we know that the MLE for $\frac{p_1}{p_2}$ will simply be $\frac{\hat{p_1}}{\hat{p_2}} = \frac{x_1 n_2}{x_2 n_1}$. We also know that since independent assumptions made the log-likelihood ratio additive, it follows that $LLR = LLR\,(\text{sample}_1) + LLR\,(\text{sample}_2)$. And since we want to use the likelihood ratio test method, we'll need to consider the following hypotheses:

$$H_0 : \frac{p_1}{p_2} = \theta$$
$$H_1 : \frac{p_1}{p_2} \neq \theta$$

Under the likelihood ratio test, our initial parameters of interest are p_1 and p_2. But this is equivalent to considering the parameters p_1 and θ, since $\theta = \frac{p_1}{p_2}$. If we take p_1 and θ to be our parameters of interest, then it follows that p_1 can be viewed as our nuisance parameter since θ is what we're truly interested in. From our previous coursework, we know that we can get rid of a nuisance parameter in the log-likelihood ratio by simply "maximizing it out." In this case, we'll have

$$LLR(\theta) = \max_{\left\{p_1, p_2 : \frac{p_1}{p_2} = \theta\right\}} \{LLR\,(\text{sample}_1) + LLR\,(\text{sample}_2)\}$$

And then we can multiply this by -2 to obtain our likelihood ratio test statistic:

$$\begin{aligned} -2\,[LLR\,(\theta)] &= -2\left[\max_{\left\{p_1, p_2 : \frac{p_1}{p_2} = \theta\right\}} \{LLR\,(\text{sample}_1) + LLR\,(\text{sample}_2)\}\right] \\ &= \min_{\left\{p_1, p_2 : \frac{p_1}{p_2} = \theta\right\}} \{-2 \cdot LLR\,(\text{sample}_1) - 2 \cdot LLR\,(\text{sample}_2)\} \end{aligned}$$

So, once we have our likelihood ratio test statistic, we can compare it to 3.84 and determine whether or not we reject H_0. Then, we can create a confidence interval for $\theta = \frac{p_1}{p_2}$ by finding the values of θ which give us a likelihood ratio test statistic less than 3.84.
Below is the R code for computing a confidence interval for the ratio of two success probabilities using the likelihood ratio test method. Note that the code is for $\theta = \frac{p_2}{p_1}$ and not $\theta = \frac{p_1}{p_2}$. But, using the invariance property of the likelihood ratio confidence intervals, we can simple take the inverse of the confidence limits to get those for $\theta = \frac{p_1}{p_2}$.

```r
BinoRatio <- function(n1, x1, n2, x2){

if(x1 >= n1) stop("all success(sample 1)?")
if(x2 >= n2) stop("all success(sample 2)?")

MLE <- (x2*n1)/(x1*n2)   #### recall we are to find CI for p2/p1, not p1/p2.
EPS <- .Machine$double.eps

Theta <- function(theta, n1=n1, x1=x1, n2=n2, x2=x2)
  {
       llr <- function(const, n1, x1, n2, x2, theta) {
              npllik1 <- -2*(x1*log((const*n1)/x1) +
                           (n1-x1)*log(((1-const)*n1)/(n1-x1)))
              npllik2 <- -2*(x2*log((const*theta*n2)/x2) +
                           (n2-x2)*log(((1-const*theta)*n2)/(n2-x2)))
        return(npllik1 + npllik2)
        }
       upBD <- min( 1-EPS, 1/theta - EPS)
       temp <- optimize(f = llr,
                        lower = EPS,
                        upper = upBD,
                        n1 = n1,
                        x1 = x1,
                        n2 = n2,
                        x2 = x2,
                        theta = theta)

    cstar <- temp$minimum
    val <- temp$objective
    pvalue <- 1 - pchisq( val, df=1)
    list(`-2LLR` = val, cstar = cstar, Pval=pvalue)
  }

temp <- findUL(step=0.1, fun=Theta, MLE=MLE, n1=n1, x1=x1, n2=n2, x2=x2)

return( temp )
}

BinoRatio(n1=100, x1=30, n2=90, x2=33)

## $Low
## [1] 0.8149795
##
## $Up
## [1] 1.844754
```

```
##
## $FstepL
## [1] 1e-09
##
## $FstepU
## [1] 1e-09
##
## $Lvalue
## [1] 3.84
##
## $Uvalue
## [1] 3.84
```

From the output, we read that the 95% confidence interval for the ratio of $\theta = \frac{p_2}{p_1}$ is [0.8149795, 1.844754].

Exercise: Similar to the above BinoRatio( ) function, write an R function BinoDiff( ), that will produce the Wilks confidence interval for the difference of two binomial success probabilities.

3 Confidence Intervals related to the Kaplan-Meier Estimator

For any quantities, if they can be written as an integral with respect to the Kaplan-Meier estimator, then we can find its confidence interval. Below are just two examples.

3.1 Survival Probability at a given time

Usually the five-year survival rate and 3-year survival rate are used in breast cancer analysis. The five-year survival rate can be written as an integration wrt the Kaplan-Meier:

$$F(5) = \int_0^\infty I[s \leq 5] dF(s) \qquad and \qquad \hat{F}(5) = \int_0^\infty I[s \leq 5] d\hat{F}(s)$$

Notice this is the five year death rate, so when we got the confidence interval we need to take a 1- F to get the survival rate. Next we test the hypothesis the death rate is 0.3.

```
library(survival)

## Warning:  package 'survival' was built under R version 3.2.5
##
## Attaching package:  'survival'
## The following object is masked from 'package:quantreg':
##
##    untangle.specials
```

```r
library(emplik)
data(pbc)
FiveSurvfun <- function(x) {as.numeric(x <= 5*365.25)}
el.cen.EM2(x=pbc$time, d=(pbc$status==2), fun=FiveSurvfun, mu=0.3)
```

```
## $loglik
## [1] -1033.999
##
## $times
##   [1]   41   41   43   51   71   77   94  110  111  130  131  140  179  186
##  [15]  191  193  198  207  216  221  223  249  264  264  304  321  326  334
##  [29]  348  359  388  400  460  466  489  515  549  552  559  597  597  611
##  [43]  625  662  673  681  694  703  708  727  733  750  762  769  778  785
##  [57]  786  790  791  797  799  824  850  853  859  890  904  930  935  943
##  [71]  971  974  980  990  999 1000 1012 1037 1077 1080 1083 1095 1152 1165
##  [85] 1168 1170 1191 1191 1197 1212 1217 1235 1297 1350 1356 1360 1413 1427
##  [99] 1434 1444 1462 1478 1487 1492 1518 1536 1576 1616 1657 1682 1690 1690
## [113] 1741 1746 1786 1827 1847 1925 2011 2055 2071 2081 2090 2105 2111 2224
## [127] 2256 2286 2288 2297 2386 2400 2419 2466 2503 2540 2583 2598 2689 2769
## [141] 2796 2812 2847 3086 3090 3170 3222 3244 3282 3358 3395 3428 3445 3561
## [155] 3574 3584 3762 3839 3853 4079 4191 4795
##
## $prob
##   [1] 0.002412920 0.002412920 0.002412920 0.002412920 0.002412920
##   [6] 0.002412920 0.002412920 0.002412920 0.002412920 0.002412920
##  [11] 0.002412920 0.002412920 0.002412920 0.002412920 0.002412920
##  [16] 0.002412920 0.002412920 0.002412920 0.002412920 0.002412920
##  [21] 0.002412920 0.002412920 0.002412920 0.002412920 0.002412920
##  [26] 0.002412920 0.002412920 0.002412920 0.002412920 0.002412920
##  [31] 0.002412920 0.002412920 0.002412920 0.002412920 0.002412920
##  [36] 0.002412920 0.002419313 0.002419313 0.002419313 0.002419313
##  [41] 0.002419313 0.002419313 0.002425844 0.002425844 0.002425844
##  [46] 0.002425844 0.002432482 0.002432482 0.002432482 0.002432482
##  [51] 0.002439231 0.002446036 0.002446036 0.002446036 0.002446036
##  [56] 0.002446036 0.002446036 0.002452996 0.002452996 0.002452996
##  [61] 0.002452996 0.002452996 0.002467240 0.002467240 0.002467240
##  [66] 0.002474487 0.002481799 0.002481799 0.002481799 0.002489220
##  [71] 0.002496707 0.002496707 0.002504286 0.002511934 0.002519653
##  [76] 0.002519653 0.002519653 0.002535379 0.002567530 0.002567530
##  [81] 0.002567530 0.002592423 0.002652624 0.002661483 0.002661483
##  [86] 0.002661483 0.002670492 0.002670492 0.002670492 0.002670492
##  [91] 0.002688945 0.002717207 0.002775760 0.002890359 0.002890359
##  [96] 0.002890359 0.002980882 0.003016449 0.003028542 0.003065560
## [101] 0.003103652 0.003103652 0.003116668 0.003116668 0.003129906
## [106] 0.003143314 0.003212418 0.003284954 0.003361194 0.003408885
```

```
## [111] 0.003408885 0.003408885 0.003475583 0.003475583 0.003581743
## [116] 0.003553299 0.003589931 0.003685408 0.003871749 0.003938504
## [121] 0.003938504 0.003938504 0.003938504 0.003961808 0.003985532
## [126] 0.004268376 0.004382199 0.004441821 0.004441821 0.004472667
## [131] 0.004773071 0.004773071 0.004809507 0.004962190 0.005042877
## [136] 0.005258388 0.005647918 0.005647918 0.006047321 0.006372545
## [141] 0.006442604 0.006515029 0.006745127 0.007982833 0.007982833
## [146] 0.008747527 0.008747527 0.009046707 0.009207121 0.009552609
## [151] 0.009739220 0.009937574 0.009937574 0.011693034 0.011693034
## [156] 0.012361492 0.013578060 0.014587256 0.015184368 0.023689194
## [161] 0.030921914 0.341963933
##
## $lam
## [1] -5.092003
##
## $iters
## [1] 25
##
## $`-2LLR`
## [1] 0.09037645
##
## $Pval
## [1] 0.7636991
```

We see that the function `el.cen.EM2` is testing the null hypothesis

$$H_0 : \int_0^\infty fun(x)dF(x) = mu$$

where you specify the function `fun` and the value `mu`. Here the P-value of the test is 0.7637. Next we use the function findUL to find the 95% confidence interval.

```
myfun9 <- function(theta, x, d){
el.cen.EM2(x, d, fun=function(x) {as.numeric(x <= 5*365.25)}, mu=theta)
}
findUL(step=0.2, fun=myfun9, MLE=0.3, x=pbc$time, d=(pbc$status==2))

## $Low
## [1] 0.2528338
##
## $Up
## [1] 0.3444268
##
## $FstepL
## [1] 2e-09
##
```

```
## $FstepU
## [1] 2e-09
##
## $Lvalue
## [1] 3.839999
##
## $Uvalue
## [1] 3.839999
```

We see that the confidence interval for F(5 year) is [0.25283, 0.34442]. Thus the confidence interval for the 5 year *survival* rate is [1-0.34442, 1-0.25283]=[0.65558, 0.74717]. The step option in the `findUL` function is the search step size. When we search a confidence interval inside [0,1] then this step should be small like 0.1 or 0.2. The MLE entry is the MLE of the estimator, but do not have to be exact, any value inside the eventual confidence interval should work.

Exercise: Given two samples of censored survival times, find the difference of two 5-year survival probabilities.

3.2 Difference of two Survival Probabilities at a given time

Take the pbc data from survival package. The two treatments lead to two samples.

```
mypbc <- pbc[1:312,]
pbcX1 <- mypbc$time[mypbc$trt==1]
pbcX2 <- mypbc$time[mypbc$trt==2]
pbcD1 <- mypbc$status[mypbc$trt==1]
pbcD2 <- mypbc$status[mypbc$trt==2]
pbcd1 <- as.numeric(pbcD1==2)
pbcd2 <- as.numeric(pbcD2==2)
```

3.3 Ratio of two Survival Probabilities at a given time

We compute the ratio of 5 year survival ratio confidence interval. Here we take the nuisance parameter `r` as the 5-year survival probability of sample two, while `theta` is the parameter of interest: the ratio sample 1 surv prob / sample 2 surv prob.

```
Ratiofun3 <- function(theta, x1, d1, x2, d2){
       survRatio <- function(r, x1, d1, x2, d2, theta){
      temp1 <- el.cen.EM2(x=x1,d=d1,fun=function(x){as.numeric(x > 5*365.25)},mu=r*theta)
      temp2 <- el.cen.EM2(x=x2,d=d2,fun=function(x){as.numeric(x > 5*365.25)},mu=r)
       return(temp1$"-2LLR" + temp2$"-2LLR")
           }

          temp <- optimize(f=survRatio,
                           lower=0.4,
```

```
                            upper=0.9,
                            x1=x1, d1=d1, x2=x2, d2=d2,
                            theta=theta)
          cstar <- temp$minimum
          val <- temp$objective
          list("-2LLR"=val, cstar=cstar)
}

findUL(step=0.1, fun=Ratiofun3, MLE=1, x1=pbcX1, d1=pbcd1, x2=pbcX2, d2=pbcd2)

## $Low
## [1] 0.8530551
##
## $Up
## [1] 1.149156
##
## $FstepL
## [1] 1e-09
##
## $FstepU
## [1] 1e-09
##
## $Lvalue
## [1] 3.84
##
## $Uvalue
## [1] 3.84
```

We see the 95% confidence interval for $S_1(5)/S_2(5)$ is [0.8530551, 1.149156]. Exercise: find the 95% confidence interval for the ratio $F_1(5)/F_2(5)$.

3.4 Restricted Mean Survival Time

When the largest survival time in a sample is right censored, the Kaplan-Meier estimator is un-defined beyond the last observed time. Thus the mean survival time estimator based on the Kaplan-Meier estimator of the survival time cannot be defined. In those cases we often use the 'restricted mean' as a measure of the average survival.

The restricted mean survival time is used often in the comparison of cancer treatments. By definition, the restricted mean survival time (RMST) is

$$\mu(\tau) = \int_0^\tau 1 - F(s)ds$$

where τ is a prespecified time. An obvious estimate of the RMST is to replace $F(s)$ by the Kaplan-Meier $\hat{F}(s)$. By an integration by parts, we can rewrite the RMST as

$$\mu(\tau) = \int_0^\infty \min(s, \tau)\, dF(s)$$

and the estimate as (assume we always have Kaplan-Meier drop down to zero at infinity)

$$\hat{\mu}(\tau) = \int_0^\infty \min(s, \tau)\, d\hat{F}(s)$$

For more discussions and references, please see the R package survRM2 and references therein.We used the dataset `ovarian` from the `survival` package and pre-select the time restriction τ= 600. Suppose we are trying to test the hypothesis H_0: $\hat{\mu}(600) = 530$.

```
library(survival)
library(emplik)
data(ovarian)
RMSTfun <- function(x) {pmin(x, 600) - 530}
el.cen.EM2(x=ovarian$futime, d=ovarian$fustat, fun=RMSTfun, mu=0)

## $loglik
## [1] -48.13337
##
## $times
##  [1]   59  115  156  268  329  353  365  431  464  475  563  638 1227
##
## $prob
##  [1] 0.01669385 0.01789823 0.01889634 0.02229227 0.02471097 0.02581288
##  [7] 0.02640152 0.03250394 0.03681496 0.03792030 0.05351697 0.06241253
## [13] 0.62412525
##
## $lam
## [1] -0.0719794
##
## $iters
## [1] 25
##
## $`-2LLR`
## [1] 2.9891
##
## $Pval
## [1] 0.08382675
```

We see that the p-value of the test is 0.0838. Next we try to construct the 95% confidence interval for RMST. This is accomplished by the function `findUL` from the package `emplik`, which in turn requires a function that returns log likelihood in testing a hypothesis about `theta`.

```
myfun8 <- function(theta, x, d) {
el.cen.EM2(x, d, fun=function(t){pmin(t, 600) - theta}, mu=0)
}
findUL(step=10, fun=myfun8, MLE=500, x=ovarian$futime, d=ovarian$fustat)
```

```
## $Low
## [1] 406.4569
##
## $Up
## [1] 535.4033
##
## $FstepL
## [1] 1e-07
##
## $FstepU
## [1] 1e-07
##
## $Lvalue
## [1] 3.84
##
## $Uvalue
## [1] 3.84
```

The input MLE in the `findUL` is supposed to be the MLE of the parameter that we are try to find its confidence interval, in our case it is the MLE of RMST. But it do not have to be very accurate, a ballpark value will work. We set the option step=10 in finfUL, since the numbers here are large. Ideally step should be similar to (or 1/2 of) the standard deviation of the MLE.

A much faster version of the calculation (replacing `el.cen.EM2`) is available in the package KMC.

Exercise: find the 95% confidence interval for the ratio of RMST when we have two samples of censored survival times.

By using the R package KMC, we get the same result but the speed is vastly improved.

```
library(survival)
library(emplik)
library(kmc)

## Warning:  package 'kmc' was built under R version 3.2.5
## Loading required package:  compiler
## Loading required package:  rootSolve
## Warning:  package 'rootSolve' was built under R version 3.2.5

data(ovarian)

 junkf <- function(x){ pmin(x,600) - 532.6 }
 kmc.solve(x=ovarian$futime, d=ovarian$fustat, g=list(junkf) )

## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced

##
## ------------------------------------------------------------------------------
```

```
## A Recursive Formula for the Kaplan-Meier Estimator with Constraint
## Information:
## Number of Constraints:  1
## lamda(s):  0.07839129
##
## ---------------------------------------------------------------------------
##      Log-likelihood(Ha)  Log-likelihood(H0)  -2LLR    p-Value(df=1)
## Est  -46.639             -48.329              3.380    0.066
## ---------------------------------------------------------------------------

myfun9 <- function(theta, x, d){
ff <- function(t){pmin(t, 600) - theta}
temp <- kmc.solve(x, d, g=list(ff) )
list("-2LLR" = temp$"-2llr")
}

findUL(step=15, fun=myfun9, MLE=532.22, x=ovarian$futime, d=ovarian$fustat)

## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
```

```
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
```

```
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
## Warning in qchisq(-2 * (loglik.ha - loglik.null), df = length(g)):  NaNs produced
```

```
## $Low
## [1] 406.4569
##
## $Up
## [1] 535.4033
##
## $FstepL
## [1] 1.5e-07
##
## $FstepU
## [1] 1.5e-07
##
## $Lvalue
## [1] 3.84
##
## $Uvalue
## [1] 3.84
```

3.5 Difference of two Restricted Mean Survival Times

We try to find the confidence interval for the difference of two RMST from two samples. We use dataset pbc from survival package. Here we take τ = 10 years = 3652.5 days. In order to compare with the output of survRM2, we use the exact the same data cases as their example.

```
mypbc <- pbc[1:312,]
pbcX1 <- mypbc$time[mypbc$trt==1]
pbcX2 <- mypbc$time[mypbc$trt==2]
pbcD1 <- mypbc$status[mypbc$trt==1]
pbcD2 <- mypbc$status[mypbc$trt==2]
pbcd1 <- as.numeric(pbcD1==2)
pbcd2 <- as.numeric(pbcD2==2)

Thetafun <- function(theta, x1, d1, x2, d2) {
            RMSTdiff <- function(r, x1, d1, x2, d2, theta){
     temp1 <- el.cen.EM2(x=x1, d=d1, fun=function(x){pmin(x, 3652.5)}, mu=r)
     temp2 <- el.cen.EM2(x=x2, d=d2, fun=function(x){pmin(x, 3652.5)}, mu=r-theta)
```

```
            return(temp1$"-2LLR" + temp2$"-2LLR")
            }

            temp <- optimize(f=RMSTdiff,
                             lower=1500,
                             upper=3100,
                             x1=x1,
                             d1=d1,
                             x2=x2,
                             d2=d2,
                             theta=theta)
            cstar <- temp$minimum
            val <- temp$objective
            list("-2LLR"=val, cstar=cstar)
}

findUL(step=100, fun=Thetafun, MLE=0, x1=pbcX1, d1=pbcd1, x2=pbcX2, d2=pbcd2)

## $Low
## [1] -340.4272
##
## $Up
## [1] 243.4025
##
## $FstepL
## [1] 1e-06
##
## $FstepU
## [1] 1e-06
##
## $Lvalue
## [1] 3.84
##
## $Uvalue
## [1] 3.84
```

Clearly this coding consist of three blocs. (In the binomial success probabilities ratio example in section two, there are also these three functional blocs). The first bloc is the `RMSTdiff` function, which computes the log likelihood ratio from each sample and sum them together. Please note here we introduced a nuisance parameter (RMST of sample one, r) to make the calculation of empirical likelihood easy. The second bloc is the optimize function. We use `optimize` to profile out the nuisance parameter r, in this case it is the RMST of the sample one. Two points of caution: the log likelihood ratio from the first bloc may not be smooth [but here it is smooth]. An example is the testing of quantile/medians. If that is the case, optimize will have to be applied after some smoothing, otherwise it does not work. Another point is the lower and upper bound of the search. This must take into account the

value and approximate confidence interval for the nuisance parameter. Here the RMST of sample 1 has value approximate 2612 and (95%) confidence interval approximately 2400 to 2815. This make us supply the lower = 1500 days and upper = 3100 days. When in doubt, a larger range can be used. The drawback is the search will be slower.

The third bloc is the `findUL` function. Where you need to input the step. Here we can roughly know the width of final confidence interval should be at least 400 days. So we use step =100. The final confidence interval of the difference, RMST1 - RMST2 is: [-340.4272, 243.4025].

3.6 Ratio of two Restricted Mean Survival Times

Continue from the example in section 3.5, we try to find the confidence interval for the ratio of two RMST from two samples. We still use dataset pbc from survival package.

```
Thetafun <- function(theta, x1, d1, x2, d2) {
            RMSTratio <- function(r, x1, d1, x2, d2, theta){
     temp1 <- el.cen.EM2(x=x1,d=d1,fun=function(x){pmin(x, 3652.5)},mu=r*theta)
     temp2 <- el.cen.EM2(x=x2,d=d2,fun=function(x){pmin(x, 3652.5)},mu=r)
            return(temp1$"-2LLR" + temp2$"-2LLR")
            }

            temp <- optimize(f=RMSTratio,
                             lower=1500,
                             upper=3100,
                             x1=x1,
                             d1=d1,
                             x2=x2,
                             d2=d2,
                             theta=theta)
            cstar <- temp$minimum
            val <- temp$objective
            list("-2LLR"=val, cstar=cstar)
}

findUL(step=0.5, fun=Thetafun, MLE=1, x1=pbcX1, d1=pbcd1, x2=pbcX2, d2=pbcd2)

## $Low
## [1] 0.8782903
##
## $Up
## [1] 1.09761
##
## $FstepL
## [1] 5e-09
##
## $FstepU
```

```
## [1] 5e-09
##
## $Lvalue
## [1] 3.839998
##
## $Uvalue
## [1] 3.839997
```

Please note this is the 95% confidence interval for the ratio of RMST1/RMST2. In this case it is [0.8782903, 1.09761]. To see more discussions, please see the R package RMST2 and references there in. The method used there is the Wald confidence interval. The advantage of the method used here is that we do not have to worry what transformation to use when work on the confidence interval for the difference of two RMST, for example.

4 AUC of ROC Curve:

This is also a two-sample setting but the statistic is a U-statistics type: $E(I[X \geq Y]) = AUC$. The computation of the empirical likelihood is achieved by the so called ELseesaw method.

```
library(pROC)

## Warning: package 'pROC' was built under R version 3.2.5
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(pAUC)

## Error in library(pAUC): there is no package called 'pAUC'

data(aSAH)

auc(aSAH$outcome, aSAH$s100b)

## Area under the curve: 0.7314

xx <- aSAH[,6]
xxx <- xx[aSAH[,2]=="Good"]
yyy <- xx[aSAH[,2]=="Poor"]

eltest2step.all(true=0.72, x=xxx, y=yyy, tol=0.0001)

## Error in eval(expr, envir, enclos): could not find function "eltest2step.all"
```

```
find.UL(step=0.6, fun=eltest2step.all, MLE=0.73, x=xxx, y=yyy, tol=0.0001)

## Error in eval(expr, envir, enclos):  could not find function "find.UL"

find.UL(step=0.002, fun=neighb.xy3p12, MLE=0.031, vp=c(0,0.1), x=xxx, y=yyy,
          tol=0.0001, eps=0.02, level=0.9)

## Error in eval(expr, envir, enclos):  could not find function "find.UL"
```

5 Reference:

Reference:

Zhao, Y. and Zhou, M. (2016) Partial AUC for ROC curve by Empirical Likelihood

Zhou, M. (2016) *Empirical Likelihood Methods in Survival Analysis* CRC Press