PharmaSUG 2015 - Paper SP10

%PIC_NPMLE: A SAS Macro For Nonparametric Estimation In Partly Interval-Censored Survival Data

Liang Zhu, St. Jude Children's Research Hospital, Memphis, TN
Yimei Li, St. Jude Children's Research Hospital, Memphis, TN
Qinlei Huang, St. Jude Children's Research Hospital, Memphis, TN

ABSTRACT

Partly interval-censored (PIC) data arise frequently in follow-up studies. An example of such data is provided by the St. Jude Lifetime Cohort study (SJLIFE), which follows childhood cancer survivors for late effects such as growth hormone deficiency (GHD). For some patients, the exact time of the first occurrence of GHD is reported, but for others, the exact time is unknown and is recorded as occurring between two clinic visits. PIC data is an important topic in medical research; however, no statistical software is available for analyzing it. In this paper, we provide two SAS™ macros to calculate the nonparametric maximum-likelihood estimator (NPMLE) of the survival function in PIC data. We estimate the NPMLE by using an iterative convex minorant (ICM) algorithm and an EM iterative convex minorant (EM-ICM) algorithm based on two different likelihood functions. Our simulation studies showed that both of the proposed algorithms provide consistent estimates for survival functions and the macro %PIC_NPMLE using EM-ICM algorithm computes much faster than the other macro using EM algorithm. Finally, we illustrate how to use the %PIC_NPMLE macro by applying it to GHD data from the SJLIFE study mentioned above.

INTRODUCTION

Partly interval-censored (PIC) data arise frequently in practice and consist of exact data and interval-censored data (Kim, 2003; Zhao and Sun, 2004; Zhao et al., 2008).This type of data often occurs in medical follow-up studies in which exact failure time is observed for only some of the participants. For the remaining participants, the event is only known to have occurred between two examinations. An example of such data is provided by the St. Jude Lifetime Cohort Study (SJLIFE). All of the participants enrolled in the study had childhood cancer that was treated at St. Jude Children's Research Hospital (Memphis, TN). Upon completion of treatment, the survivors were closely monitored by the St. Jude After Completion of Treatment (ACT) program for 10 years or until the participant reached 18 years of age. After graduation from the ACT program, survivors were no longer followed by St. Jude. The SJLIFE study is now following those adult survivors prospectively for the duration of their lives to assess their health status periodically.

Time of the first occurrence of growth hormone deficiency (GHD) is one of the outcomes being investigated by the SJLIFE researchers. For some participants, history of GHD previously diagnosed during cancer treatment or ACT program was extracted from medical records, and the exact time of first occurrence of GHD is known as T. For the rest of the participants with GHD, the late effect was detected during the SJLIFE study, and the exact time of onset is unknown. Thus, time of the first occurrence of GHD is recorded as (U, V), in which U is the last monitored time before GHD was discovered, and V is the visit time when GHD was discovered. Clearly, the time of the first occurrence of GHD in this study is a mixture of exact and interval-censored time-to-event data.

We are interested in studying the survival function of the occurrence of GHD. Currently, 748 participants are enrolled in the SJLIFE pituitary study: 136 participants have exact time of occurrence of GHD; 212 have an interval-censored occurrence of GHD; and 400 have had no occurrences of GHD. An example of the data for the PIC event time of GHD is shown in Table 1. For subjects 1 and 2, GHD was discovered at an exact time (i.e., at 25.47 years and 23.98 years respectively). For subjects 3 and 4, GHD has not yet been observed, and the last time GHD status was checked was at 47.32 years and 59.32 years, respectively. For subject 5, GHD onset occurred sometime between 21.72 years and 45.70 years.

Subject	U	V	T	Indicator
1	.	.	25.47	1
2	.	.	23.98	1
3	47.32	.	.	0
4	59.32	.	.	0
5	21.72	45.70	.	0
…	…	…	…	…

Table 1. Partly Interval-Censored Data of Time of Onset of Growth Hormone Deficiency

Other examples of such data include the Framingham Heart Disease Study (Odell et al., 1992) and the Danish Diabetes Study (Ramlau et al., 1987). Although PIC data arise frequently in practice, the methods available to analyze it are limited. Peto and Peto (1972) discussed PIC data, treating an exact failure time as an interval-censored observation with a very short interval. However, Huang (1999) pointed out that the NPMLE with interval-censored data has only an $n^{1/3}$-rate of convergence, and the Kaplan-Meier estimator for exact failure time data is asymptotically normal with an $n^{1/2}$-rate of convergence. In other words, transferring PIC data to pure interval-censored data might reduce efficiency. As a result, Huang (1999) studied the NPMLE of the distribution function for PIC data and proved that the NPMLE has an $n^{1/2}$-rate of convergence when the number of exact observations is not negligible.

In this paper, we provide two SAS^{TM} macros for NPMLE of the survival function for PIC data by using an iterative convex minorant (ICM) algorithm (Groeneboom and Wellner,1992; Jongbloed, 1998) and an expectation-maximization iterative convex minorant (EM-ICM) algorithm (Wellner and Zhan, 1997) based on two different likelihood functions (Huang, 1999). So et al. (2010) developed the macro %EMICM to obtain NPMLE for interval-censored data, and we adapted their macro to apply it to PIC data. Our macros are shown in Appendices 1 and 2.

NONPARAMETRIC MAXIMUM-LIKELIHOOD ESTIMATOR

Consider a survival study that involves n subjects. For some subjects, the exact failure times (Ts) are observed, but for the remaining subjects, their failure times are interval-censored. In other words, their failure times were not observed and are known only to have occurred between two observation times (U, V]. Let δ_i=1 if the ith subject has the exact failure time observed, and let δ_i =0 if the failure time is interval-censored for the ith subject for i=1,…, n. Assume that the failure times and observation times are independent.

In this section, we will provide two ways to write the likelihood function, as described in Huang (1999), and hence two methods to estimate the survival functions. On the bases of the two methods, we wrote two corresponding macros.

METHOD 1

The likelihood function for PIC data is proportional to

$$L = \prod_{i=1}^{n} dF(T_i)^{\delta i}\,[S(U_i) - S(V_i)]^{(1-\delta i)}\,, \qquad (1)$$

where $F(t)=P_r(T_i \le t)$ is the distribution function, and $S(t)=1-F(t)$ is the survival curve. To estimate F(t), we define $0< s_1 <... < s_m <1$ as the collection of the exact failure times and observed intervals $\{T_i, U_i, V_i, i=1,…,n\}$, so that the nonoverlapping intervals $\{(s_1, s_2], …, (s_{m-1}, s_m]\}$ should cover all T_i for δ_i =1, and every interval of $(U_i, V_i]$ for δ_i =0 contains at least one s_j for i=1,…, n and j=1,…, m. It is assumed that the NPMLE of F(t) is a step function that jumps only at $\{s_j\}_{j=1}^{m}$, unless otherwise specified. In other words, we can determine the NPMLE by maximizing the likelihood function over $F(s_1) <… < F(s_m)$.

To estimate $F(s_j)$ for j=1,…, m, we define $p_j=F(s_j)-F(s_{j-1})$; thus, the log-likelihood function becomes

$$logL = \sum_{i=1}^{n}\left[\delta i \log\{\mathrm{I}(T_i = s_j)p_j\} + (1-\delta i)\log\left\{\sum_{j=1}^{m} I(U_i < s_j \le V_i)\, p_j\right\}\right].$$

One can take the first and second derivatives of logL on $F(s_j)$ through p_j and apply the ICM algorithm to avoid the problem of inverting the Hessian matrix when the number of parameters to be estimated is too large. On the basis of this method, we write the macro %PIC_NPMLE_ICM (see Appendix 1).

METHOD 2

The ICM algorithm described above provides one way to estimate the survival function. Another way is to use the EM-ICM algorithm. Following Huang (1999) and assuming F(t) is a step function, the likelihood function for PIC data given in (1) can be rewritten as follows:

$$L = \prod_{i=1}^{n}[S(T_{(i-1)}) - S(T_{(i)})]^{\delta i}\,[S(U_i) - S(V_i)]^{(1-\delta i)}\,, \qquad (2)$$

where $T_{(i)}$s are the ordered values of exact failure times and T_0=0. Clearly, we can let $S(T_0)=0$. If we define $\alpha_{ij} =\delta_i\, I(T_{i-1}<s_j \le T_i)+(1-\delta_i)I(U_i<s_j \le V_i)$, the log-likelihood function then becomes

$$logL = \sum_{i=1}^{n}\left[\log\left\{\sum_{j=1}^{m} \alpha_{ij}\; p_j\right\}\right].$$

With this form of log-likelihood, one can estimate F(s_j), j=1,…, m, by using the EM-ICM algorithm. In this algorithm, the self-consistency equation $p_j = 1/n \sum_{i=1}^{n} \alpha_{ij} p_j \ / \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{ij}\ p_j$, which is provided in Turnbull (1976), can be used, as well as the ICM algorithm, to maximize the likelihood function. Based on this method, the macro %EMICM (So et al., 2010) can be adapted to the macro %PIC_NPMLE (see Appendix 2) to estimate the survival function.

ANALYSIS OF SIMULATED DATA

We performed some simulation studies to check and compare the performance of the two macros (%PIC_NPMLE_ICM and %PIC_NPMLE). First, we generated a dataset of 200 observations (repeats=500) with the following code. Among them, 100 observations had exact time data, and the other 100 observations were interval censored.

```
data underlyingdata;
 do i=1 to 200;
    randu=rand("Uniform"); T=sqrt(-log(randu)); left=rand("Uniform");
    right=left+rand("Uniform");
    output;
 end;
run;

data rawdata;
 set underlyingdata;
 if _n_<=100 then do; indicatorT=1; T=T; U=.; V=.; end; else indicatorT=0;
 if indicatorT=0& T<=left then do; T = . ; U=0; V=left; end;
 if indicatorT=0& T>left & T<=right then do; T = . ; U=left; V=right; end;
 if indicatorT=0&  T>right then do; T = . ; U=right; V=.; end;
run;

data rawdata; set rawdata; keep U V T indicatorT; run;
```

Next, we used the two macros to estimate the survival functions:

```
%PIC_NPMLE(datain=rawdata, t=T, u=U, v=V, indicator=indicatorT, rateconv=1e-5,
title=%str(NPMLE survival curve for PIC data), xlabel=%str(T));

%PIC_NPMLE_ICM(datain=rawdata, t=T, u=U, v=V, indicator=indicatorT, rateconv=1e-5,
title=%str(NPMLE survival curve for PIC data), xlabel=%str(T));
```

Here, datain is the dataset to be analyzed; t is the variable of exact observation time; u is the left interval; v is the right interval; the indicator designates whether the data is exact time (1) or interval censored (0); rateconv is the convergence criteria; title is the title of the output plot; and xlabel is the label of the x axis.

Both methods provided estimates that approximated the true survival function (Figure 1), and as expected, when the sample size was increased, the estimates were even better. By comparing the two macros, we found that they provide essentially the same results. However, in terms of computing speed, %PIC_NPMLE was much faster than %PIC_NPMLE_ICM: the former macro required 12 iterations on average to reach convergence, while the latter required 5859 on average. As a result, we recommend using %PIC_NPMLE to analyze PIC data.

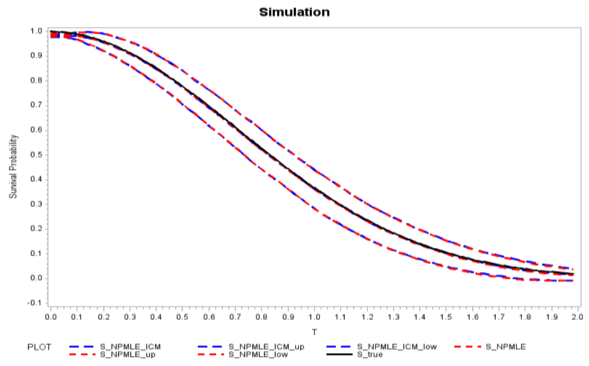

Figure 1. The Estimated Survival Functions for Simulation Data. Blue lines represent the estimate and corresponding 95% confidence interval from %PIC_NPMLE_ICM using method 1; the red lines represent the estimate and corresponding 95% confidence interval from %PIC_NPMLE using method 2; and the black

lines represent the true values of survival function.

ANALYSIS OF A REAL DATASET

We used the macro %PIC_NPMLE to analyze the SJLIFE data and calculated the estimated survival function for the first occurrence of GHD (Figure 2).

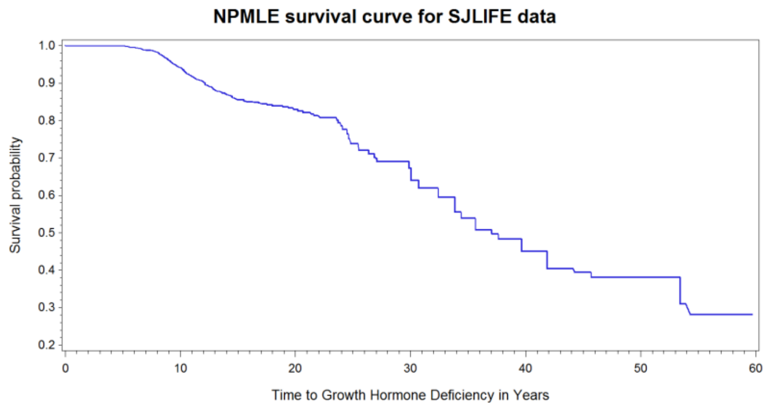

Figure 2. The Estimated Survival Function for the First GHD Occurrence

The macro also provided a list of the NPMLE of the survival functions for the PIC data. Examples of the SJLIFE data are shown in Table 2.

T	S
…	…
22.308	0.80863
23.5975	0.80168
23.707	0.79448
23.9808	0.78591
…	…

Table 2. The NPMLE of the Survival Function in PIC Data from the SJLIFE Study
*Data indicate the time points (T) for s_j, j=1, …, m
†Data indicate the probability of survival (S) at that time point.

CONCLUSION

Many methods have been developed to analyze exact failure-time data and interval-censored data. Although software is widely available to analyze exact failure-time data, it is not available to analyze interval-censored data. This disparity may reflect the fact that methods for analyzing interval-censored data are more complicated and harder to implement. As a result, there is even less literature about and software for analyzing a mixture of the two types of data, though PIC data are very common in real data sets. In this paper, we provided a SAS macro to address the important issue of NPMLE of the survival function in PIC data. One possible extension of this current paper would be to develop the test procedures to compare multiple survival functions.

REFERENCES

- Groeneboom, P. and Wellner, J. A. (1992). Information bounds and nonparametric maximum likelihood estimation, volume 19. Springer Science & Business Media.
- Huang, J. (1999). Asymptotic properties of nonparametric estimation based on partly interval-censored data. Statistica Sinica, 9(2):501-519.
- Jongbloed, G. (1998). The iterative convex minorant algorithm for nonparametric estimation. Journal of Computational and Graphical Statistics, 7(3):310-321.
- Kim, J. S. (2003). Maximum likelihood estimation for the proportional hazards model with partly interval-censored data. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 65(2):489-502.
- Odell, P. M., Anderson, K. M., and D'Agostino, R. B. (1992). Maximum likelihood estimation for interval-censored data using a weibull-based accelerated failure time model. Biometrics, pages 951-959.

- Peto, R. and Peto, J. (1972). Asymptotically efficient rank invariant test procedures. Journal of the Royal Statistical Society. Series A (General), pages 185-207.
- Ramlau-Hansen, H., Chr. Bang Jespersen, N., Kragh Andersen, P., Borch-Johnsen, K., and Deckerf, T. (1987). Life insurance for insulin-dependent diabetics. Scandinavian Actuarial Journal, 1987(1-2):19-36.
- So, Y. et al. (2010). Analyzing interval-censored survival data with... In SAS GLOBALFORUM 2010. Citeseer.
- Turnbull, B. W. (1976). The empirical distribution function with arbitrarily grouped, censored and truncated data. Journal of the Royal Statistical Society. Series B (Methodological), pages 290-295.
- Wellner, J. A. and Zhan, Y. (1997). A hybrid algorithm for computation of the nonparametric maximum likelihood estimator from censored data. Journal of the American Statistical Association, 92(439):945-959.
- Zhao, Q. and Sun, J. (2004). Generalized log-rank test for mixed interval-censored failure time data. Statist. Med., 23(10):1621-1629.
- Zhao, X., Zhao, Q., Sun, J., and Kim, J. S. (2008). Generalized log-rank tests for partly interval-censored failure time data. Biometrical Journal, 50(3):375-385.

ACKNOWLEDGMENTS

We thank Dr. Melissa Hudson for sharing the GHD data from the SJLIFE study.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Liang Zhu
Enterprise: St. Jude Children's Research Hospital
Address: Department of Biostatistics, MS 768, 262 Danny Thomas Place
City, State ZIP: Memphis, TN 38103
Work Phone: 901-595-5240
Fax: 901-595-8843
E-mail: liang.zhu@stjude.org
Web: http://www.stjude.org/zhu
Twitter: NA

APPENDIX 1: %PIC_NPMLE_ICM, A SAS MACRO FOR NONPARAMETRIC ESTIMATION FOR PARTLY INTERVAL-CENSORED DATA

```
%macro PIC_NPMLE_ICM(datain=, u=, v=, t=, indicator=, rateconv =1e-5, title = %str(NPMLE survival curve for PIC
data), xlabel = %str(time to a));
proc iml;
   use &datain;
   read all var{&U &V &T &indicator};
   L= &U; R= &V; T= &T; indicatorT = &indicator;

   MissR = sum(R <= . );
   if MissR> 0 then do;
      locR = loc(R <= .); maxR= max(R); maxL= max(L); maxT=max(T);
      if (maxR > maxL) & (maxR>maxT) then maxR2= maxR*2;
      if (maxL > maxR) & (maxL>maxT) then maxR2= maxL*2;
      if (maxT > maxL) & (maxT>maxR) then maxR2= maxT*2;
      R[locR, ] = maxR2  ;
   end;
   nobs=nrow(L);
   n1=0; n2=0;
   do i=1 to nobs;
      if indicatorT[i]=1 then n1=n1+1; if indicatorT[i]=0 then n2=n2+1;
   end;

   *1: handle exact event times;
   if n1>0 then do;
      sj1a=unique(T); msj1a=ncol(sj1a);
      if sj1a[1]=. then sj1=sj1a[2:msj1a];
      mT=nrow(sj1);
   end;

   *2: handle (L,R);
   if n2>0 then do;
      indicover=J(1,n2,0);
      do i=(n1+1) to nobs;
         do j=1 to mT;
            if (L[i]<sj1[j] & sj1[j]<=R[i]) then indicover[i-n1]=1;
         end;
      end;
      nuncovered=n2-sum(indicover);
      if nuncovered>0 then do;
         Luncovered=J(1, nuncovered,.); Runcovered=J(1, nuncovered,.); count=0;
            do i=1 to n2;
               if indicover[i]=0 then do;
                  count=count+1; Luncovered[count]=L[n1+i]; Runcovered[count]=R[n1+i];
               end;
            end;
         pp= unique(Runcovered); npp= ncol(pp); qq= unique(Luncovered); nqq= ncol(qq); q= J(1, npp, .);
         do;
            do i=1 to npp;
               do j=1 to nqq;
               if (qq[j] < pp[i]) then q[i]= qq[j];
               end;
               if q[i]= qq[nqq] then goto lab1;
            end;
            lab1:
         end;
         if i > npp then nq= npp;
         else nq= i;
         q= unique(q[1:nq]); m= ncol(q); p= J(1, m, .);
         do i=1 to m;
            do j= npp to 1 by -1;
```

```
                if (pp[j] > q[i]) then p[i] = pp[j];
             end;
          end;
          sj22a = unique(p); m22=ncol(sj22a); sj22=J(m22,1,.);
          do i=1 to m22;
             sj22[i]=sj22a[i];
          end;
       end;
       if nuncovered=0 then sj=unique(t(sj1));
       if nuncovered>0 then do; sjall=t(sj1//sj22);sj=unique(sjall);  end;
    end;
    if n2=0 then sj=unique(t(sj1));
    m=ncol(sj);

    if n1>0 then do;
       dY1=J(n1,m,0);
       do i=1 to n1;
          dY1[i,]=(sj=T[i,1]);
       end;
    end;

    if n2>0 then do;
       alpha=J(n2,m,0);
       do i=1 to n2;
          alpha[i, ]= (sj > L[n1+i,1])#(sj <= R[n1+i,1]);
       end;
    end;

    start isotonic(x,weight);
       n_X = nrow(x); phat=x; wt=weight; plevel= 1:n_X ; dummy=max(phat)*1000;
       is_vlt = ((phat//dummy) <(-dummy//phat)) # t(1:(n_X +1));
       vlt_loc =min( (^is_vlt )*10*n_X  + is_vlt);
       do while (vlt_loc < 10*n_X);
          idx= t(loc(phat[1:vlt_loc, ]=phat[vlt_loc-1]) || vlt_loc) ;
          phat[idx, ] = sum(wt[idx] #phat[idx]) / sum(wt[idx]) ;
          is_vlt = ((phat//dummy) < (-dummy//phat)) # t(1:(n_X +1));
          vlt_loc =min( (^is_vlt )*10*n_X  + is_vlt);
       end;
       return(phat);
    finish isotonic;

    error1=1; RATECONV=1e-7;
    epsilon= 0.2;

    * STEP 1:  set up the initials for beta;
    iter=1;
    beta_old=J(m-1,1,1/m)#t(1:m-1); *beta is F(t), cdf. # means elementwise product, between 1/m*j, j=1,...,m-1;
    phat_old= (beta_old//1) - (0//beta_old);
    loglik=sum(log(dY1*phat_old))+ sum(log(alpha*phat_old)) ;
    alpha_dff = (alpha[ ,1:m-1]-alpha[ ,2:m]) ;

   do while (error1 > RATECONV);
       beta_up = beta_old ;
       dYp=dY1*phat_old;*dYp is sum_k=1^m dY1_{ik}*p_k, for all dF(T_i);
       if sum(dYp =0) > 0 then dYp [loc(dYp =0)] =0.00001;
       ap= alpha*phat_old ; *ap is sum_k=1^m alpha_{ik}*p_k, for all F(V_i)-F(U_i);
       if sum(ap =0) > 0 then ap [loc(ap =0)] =0.00001;
      d1l=J(m-1,1,0); d2l=J(m-1,1,0);
       if n1>0 then do;
          d1l_n1= dY1[,1:m-1]/ repeat(dYp, 1, m-1)-dY1[,2:m]/ repeat(dYp, 1, m-1);
          d1l_1= t(d1l_n1[+, ]);
          d2l_n1= - (d1l_n1##2);
```

```
        d2l_1=t(d2l_n1[+, ]);
          d1l=d1l+d1l_1;
          d2l=d2l+d2l_1;
    end;
    if n2>0 then do;
        d1l_n2= alpha_dff  / repeat(ap, 1, m-1); d1l_2= t(d1l_n2[+, ]);
        d2l_n2= - (d1l_n2##2); d2l_2=t(d2l_n2[+, ]);
        d1l=d1l+d1l_2; d2l=d2l+d2l_2;
    end;
    W=diag(-d2l);
    x_new= beta_old + d1l/vecdiag(W);
    if sum(x_new<0) then x_new[loc(x_new<0)]=0.00001;
    beta_up = isotonic(x_new, vecdiag(W));
    phat_up = (beta_up//1) - (0//beta_up);
    dYp_up = dY1*phat_up;
    ap_up = alpha*phat_up;
    if sum(dYp_up<=0)> 0 then dYp_up[loc(dYp_up<=0)] =0.00001;
    if sum(ap_up<=0)> 0 then ap_up[loc(ap_up<=0)] =0.00001;
    loglik_up=0;
   if n1>0 then loglik_up =loglik_up+sum(log(dYp_up));
   if n2>0 then loglik_up=loglik_up+sum(log(ap_up)) ;

   if loglik_up >= loglik then beta_new= beta_up;
    if loglik_up < loglik then do;
         lambda=1; gamma= 0.5; zz= beta_up; phat_old= (beta_old//1) - (0//beta_old) ; dYp_old  = dY1*phat_old ;
         ap_old  = alpha*phat_old ;
         if sum(dYp_old=0) > 0 then dYp_old[loc(dYp_old=0)] =0.00001;
         if sum(ap_old=0) > 0 then ap_old[loc(ap_old=0)] =0.00001;
         d1l_old=J(m-1,1,0);  loglik_old=0;
         if n1>0 then do;
              d1l_n1_old= dY1[,1:m-1]/ repeat(dYp_old, 1, m-1)-dY1[,2:m]/ repeat(dYp_old, 1, m-1);
              d1l_old=d1l_old+ t(d1l_n1_old[+, ]);
              loglik_old=loglik_old+sum(log(dYp_old));
          end;
          if n2>0 then do;
              d1l_n2_old= alpha_dff  / repeat(ap_old , 1, m-1);
              d1l_old=d1l_old+  t(d1l_n2_old[+, ]);
              loglik_old=loglik_old+sum(log(ap_old)) ;
          end;
         loglik_zz=loglik_up  ;
         lb = loglik_old + epsilon*t(d1l_old) *(zz-beta_old);
        ub= loglik_old + (1-epsilon)*t(d1l_old) *(zz-beta_old) ;
        do while ( (loglik_zz < lb) | (loglik_zz > ub) );
              if loglik_zz < lb then lambda= lambda-gamma;
              if loglik_zz > ub then lambda= lambda+gamma;
              zz= beta_old + lambda*(beta_up - beta_old);
              gamma = gamma/2 ;
              ap_zz = alpha*( (zz//1) - (0//zz) );
              if sum(ap_zz=0) > 0 then ap_zz[loc(ap_zz=0)] =0.00001;
              loglik_zz= sum( log(ap_zz) ) ;
              lbtemp = loglik_old + epsilon*t(d1l_old) *(zz-beta_old);
              ubtemp= loglik_old + (1-epsilon)*t(d1l_old) *(zz-beta_old) ;
              lb= min(lbtemp, ubtemp);
              ub=max(lbtemp, ubtemp);
        end;
        beta_new = zz;
    end;
    beta_up = beta_new;

    p_new = (beta_new //1) - (0//beta_new);
    dYp_new = dY1*p_new ;
    if sum(dYp_new =0) > 0 then dYp_new[loc(dYp_new =0)] =0.00001;
```

```
        ap_new = alpha*p_new ;
        if sum(ap_new =0) > 0 then ap_new[loc(ap_new =0)] =0.00001;
        loglik_new=0; d1l_new=J(m-1,1,0);
        if n1>0 then do;
            loglik_new=loglik_new+sum(log(dYp_new));
            d1l_n1_new= dY1[,1:m-1]/ repeat(dYp_new, 1, m-1)-dY1[,2:m]/ repeat(dYp_new, 1, m-1);
            d1l_1_new= t(d1l_n1_new[+, ]);
            d1l_new=d1l_new+d1l_1_new;
          end;
        if n2>0 then do;
            loglik_new=loglik_new+sum(log(ap_new)); d1l_n2_new= alpha_dff  / repeat(ap_new, 1, m-1);
            d1l_2_new= t(d1l_n2_new[+, ]); d1l_new=d1l_new+d1l_2_new;
        end;
        error1= max(abs(beta_new - beta_old));
        if error1 > RATECONV then do;
            beta_old = beta_new; iter= iter +1; phat_old = p_new; loglik=loglik_new;
        end;
    end;
            print iter;
    beta_new = beta_new // 1;
    surv = 1-beta_new ;

    if sj[m]=maxR2 then do;
        finalsj=J(m-1,1,0);finalsurv=J(m-1,1,0);
        do j=1 to (m-1); finalsj[j,1]=sj[j]; finalsurv[j,1]=surv[j]; end;
    end;

    if sj[m]<maxR2 then do;
        finalsj=J(m,1,0);finalsurv=J(m,1,0);
        do j=1 to m; finalsj[j,1]=sj[j]; finalsurv[j,1]=surv[j]; end;
    end;
    output=finalsj||finalsurv; coln={'t' 'S'};
    create _dataB_ from output [colname=coln];
    append from output;
    close _dataB_;
    quit;

    goptions reset=all;
    symbol i=step width=3;
    title "&title";
    axis1 label=("&xlabel");
    axis2 label=(a=90 "Survival Probability");
    proc gplot data=_dataB_;
    plot S*t/haxis=axis1 vaxis=axis2;
    run;
    quit;

    proc print data = _dataB_ noobs ;
    run ;
%mend PIC_NPMLE_ICM;
```

APPENDIX 2: %PIC_NPMLE, A SAS MACRO FOR NONPARAMETRIC ESTIMATION FOR PARTLY INTERVAL-CENSORED DATA

```
%macro PIC_NPMLE(datain=, u= , v= , t=, indicator=, rateconv=1e-5, title= , xlabel=);

proc sort data = &indata(where = (&indicator = 1))  out = t(keep = &t) ;   by &t ; run ;
data u;   set t ;   id = _n_ + 1;   rename &t = &u; run ;
data t;    set t ;   id = _n_ ; run ;
data part1;   merge t u;    by id;   &v = &t;    if &u = . then &u = 0;   if &t = . then delete;   drop &t id ; run ;
data part2;   set &indata(where = (&indicator = 0)) ;   keep &u &v ; run ;
data new;    set part1 part2;    rename &u = ufinal  &v = vfinal; run ;
proc datasets; delete t u part1 part2; quit;

proc iml;
   use new;
   read all var{ufinal vfinal}; L= ufinal; R=vfinal;

   ExactT= sum(L=R);
   if ExactT>0 then do;
      ExactTime = loc(L=R);  L[ExactTime, ] = L[ExactTime, ] - 0.00001;
   end;

   MissR = sum(R <= . );
   if MissR> 0 then do;
      locR = loc(R <= .); maxR= max(R); maxL= max(L);
      if maxR > maxL then maxR2= maxR*2;
         else maxR2= maxL*2;
      R[locR, ] = maxR2  ;
   end;

   nobs=nrow(L);

   pp= unique(r); npp= ncol(pp); qq= unique(l); nqq= ncol(qq); q= J(1, npp, .);
   do;
      do i=1 to npp;
         do j=1 to nqq;
            if (qq[j] < pp[i]) then q[i]= qq[j];
         end;
         if q[i]= qq[nqq] then goto lab1;
      end;
      lab1:
   end;
   if i > npp then nq= npp;
      else nq= i;
   q= unique(q[1:nq]); m= ncol(q); p= J(1, m, .);
   do i=1 to m;
      do j= npp to 1 by -1;
         if (pp[j] > q[i]) then p[i] = pp[j];
      end;
   end;
   sj = unique(p);

   m=ncol(sj); alpha= J(nobs, m, 0);
   do i=1 to nobs;
      alpha[i, ]= (sj > L[i,1])#(sj <= R[i,1]);
   end;

   start isotonic(x,weight);
      n_X = nrow(x); phat=x;  wt=weight; plevel= 1:n_X ; dummy=max(phat)*1000;
      is_vlt = ((phat//dummy) <(-dummy//phat)) # t(1:(n_X +1));
      vlt_loc =min( (^is_vlt )*10*n_X  + is_vlt);
```

```
        do while (vlt_loc < 10*n_X);
            idx= t(loc(phat[1:vlt_loc, ]=phat[vlt_loc-1]) || vlt_loc) ;
            phat[idx, ] = sum(wt[idx] #phat[idx]) / sum(wt[idx]) ;
            is_vlt = ((phat//dummy) < (-dummy//phat)) # t(1:(n_X +1));
            vlt_loc =min( (^is_vlt )*10*n_X  + is_vlt);
        end;
        return(phat);
    finish isotonic;

    error1=1;
    epsilon= 0.2;

    * STEP 1:  set up the initials for beta;
    iter=1;
    beta_old=J(m-1,1,1/m)#t(1:m-1);
    phat_old= (beta_old//1) - (0//beta_old);
    loglik= sum(log(alpha*phat_old)) ;
    alpha_dff = (alpha[ ,1:m-1]-alpha[ ,2:m]) ;

    do while (error1> &rateconv);
        beta_up = beta_old ;

       * STEP 2: Apply ICM ;
        ap= alpha*phat_old ;
        if sum(ap =0) > 0 then ap [loc(ap =0)] =0.00001;
        d1l_n= alpha_dff  / repeat(ap, 1, m-1); d1l= t(d1l_n[+, ]);   d2l_n= - (d1l_n##2); d2l=t(d2l_n[+, ]);  W=diag(-d2l);
        x_new= beta_old + d1l/vecdiag(W);
        if sum(x_new<0) then x_new[loc(x_new<0)]=0.00001;
        beta_up = isotonic(x_new, vecdiag(W)) ;   phat_up = (beta_up//1) - (0//beta_up);
        ap_up = alpha*phat_up; if sum(ap_up<=0) > 0 then ap_up[loc(ap_up<=0)] =0.00001;
        loglik_up = sum(log(ap_up)) ;

        /*** line search ***/
        if loglik_up >= loglik then beta_new= beta_up;
        if loglik_up < loglik then do;
            lambda=1;  gamma= 0.5; zz= beta_up; phat_old= (beta_old//1) - (0//beta_old) ; ap_old  = alpha*phat_old ;
            if sum(ap_old=0) > 0 then ap_old[loc(ap_old=0)] =0.00001;
            d1l_n_old= alpha_dff  / repeat(ap_old , 1, m-1); d1l_old= t(d1l_n_old[+, ]);
            loglik_old= sum(log(alpha*(phat_old))) ;
            loglik_zz=        loglik_up  ;
            lb = loglik_old + epsilon*t(d1l_old) *(zz-beta_old); ub= loglik_old + (1-epsilon)*t(d1l_old) *(zz-beta_old) ;
            do while ( (loglik_zz < lb) | (loglik_zz > ub) );
                if loglik_zz < lb then lambda= lambda-gamma;
                if loglik_zz > ub then lambda= lambda+gamma;
                zz= beta_old + lambda*(beta_up - beta_old); gamma = gamma/2 ;
                ap_zz = alpha*( (zz//1) - (0//zz) ); if sum(ap_zz=0) > 0 then ap_zz[loc(ap_zz=0)] =0.00001;
                loglik_zz= sum( log(ap_zz) ) ;
                lbtemp = loglik_old + epsilon*t(d1l_old) *(zz-beta_old);
                ubtemp= loglik_old + (1-epsilon)*t(d1l_old) *(zz-beta_old) ;
                lb= min(lbtemp, ubtemp); ub=max(lbtemp, ubtemp);
            end;
            beta_new = zz;
        end;
        beta_up = beta_new;

       * STEP 3: Apply Self-consistency ;
        phat_old=  (beta_up //1) - (0//beta_up ); ap = alpha*phat_old; if sum(ap =0) > 0 then ap[loc(ap =0)] =0.00001;
        p_temp= (alpha#t(phat_old))/repeat(ap , 1, m); p_new = t(p_temp[+, ]/nobs); beta_new=cusum(p_new)[1:m-1];

       * STEP 4: Convergence checking ;
        p_new = (beta_new //1) - (0//beta_new);
        ap_new = alpha*p_new ;  if sum(ap_new =0) > 0 then ap_new[loc(ap_new =0)] =0.00001;
        loglik_new= sum( log(ap_new) ) ;
        d1l_n= alpha_dff  / repeat(ap_new, 1, m-1);  d1l= t(d1l_n[+, ]);
        error1= max(abs(beta_new - beta_old));
        if error1 > &rateconv then do; beta_old = beta_new; iter= iter +1; phat_old = p_new; loglik=loglik_new; end;
    end;
    beta_new = beta_new // 1;
    surv = 1-beta_new ;

    if sj[m]=maxR2 then do;
        finalsj=J(m-1,1,0);finalsurv=J(m-1,1,0);
        do j=1 to (m-1); finalsj[j,1]=sj[j]; finalsurv[j,1]=surv[j]; end;
    end;

    if sj[m]<maxR2 then do;
       finalsj=J(m,1,0);finalsurv=J(m,1,0);
       do j=1 to m;  finalsj[j,1]=sj[j]; finalsurv[j,1]=surv[j]; end;
    end;
    output=finalsj||finalsurv;coln={'t' 'S'};
    create _dataA_ from output [colname=coln]; append from output; close _dataA_;
    quit;

    goptions reset=all;
    symbol i=step width=3;
    title "&title";
    axis1 label=("&xlabel");
    axis2 label=(a=90 "Survival Probability");
    proc gplot data=_dataA_;
    plot S*t/haxis=axis1 vaxis=axis2;
    run;
    quit;

    proc print data = _dataa_ noobs ;
    run ;
%mend PIC_NPMLE;
```