PharmaSUG 2023 - Paper AP-191

Survival Methods for Crossover in Oncology Trials

Brian Mosier, EMB Statistical Solutions, LLC

ABSTRACT

In oncology trials, patients are often allowed to switch from one treatment arm to the other when their disease progresses. This crossover of patients, which is typically not at random, may lead to bias in the estimates of overall survival for the study. As such, when patient crossover is allowed, the intent-to-treat population is no longer appropriate for analysis. Various methods have been developed to handle the crossover of patients.

This paper presents two SAS® macros to implement methods that account for treatment crossover in oncology trials. The methods included are (1) the rank preserving structural failure time (RPSFT) model, and (2) the inverse probability of censoring weighting (IPCW) model. Both models have previously been implemented in SAS, but this paper expands upon those implementations and provides corrections to code for the IPCW model for both the estimates of the weights in the first step of the method and the implementation of the weights in the second step of the method, as well as accounting for repeated measures for the subjects.

INTRODUCTION

In oncology trials, patients are often allowed to switch from one treatment arm to the other when their disease progresses. This crossover of patients, which is typically not at random, may lead to bias in the estimates of overall survival for the study. As such, when patient crossover is allowed, the intent-to-treat population is no longer appropriate for analysis. Various methods have been developed to handle the crossover of patients.

This paper presents two SAS macros to implement methods that account for crossover in oncology trials, which focuses on scenarios where subjects are allowed to switch from the control arm to the experimental treatment arm. The methods included are (1) the rank preserving structural failure time (RPSFT) model, and (2) the inverse probability of censoring weighting (IPCW) model. Both models have previously been implemented in SAS.

In previous work for the RPSFT model, only code for the log-rank test was presented (Danner & Sarkar, 2018). This paper includes additional options in the macro, allowing for a Weibull or Cox proportional hazards models. The macro also includes confidence intervals for the acceleration factor, which had to be done manually in previous work.

In previous work for the IPCW model, authors have provided code that uses logistic models to estimate weights to be used in the final model (Jiménez-Moro and Gómez, 2014). In addition to the logistic model for estimating weights, the macro presented in this paper allows for Cox proportional hazards models to estimate weights. It also provides corrections to their code for the weighted Cox proportional hazards model to estimate treatment effect to (1) account for repeated measures for the subjects and (2) properly implement the weights derived from the first step of the method.

RANK PRESERVING STRUCTURAL FAILURE TIME MODEL

The rank preserving structural failure time model estimates the change in survival time due to receiving the experimental treatment. This effect assumes that survival times are multiplied by a constant, called the acceleration factor once a subject begins treatment. This acceleration factor is interpreted as the change in survival once a patient begins treatment.

Let $T_i = T_i^{off} + T_i^{on}$ be the observed event time for subject i, where T_i^{off} and T_i^{on} are the time spent off and on treatment, respectively. For patients in the treatment group, T_i^{off} will be equal to 0, and so $T_i = T_i^{on}$. For patients who never crossed over from the control to the treatment group, T_i^{on} is equal to 0, and so $T_i = T_i^{off}$. Now, consider another event time, $U_i = T_i^{off} + T_i^{on}\exp(\psi)$, which represents the survival

time that would have been observed if the treatment had never been given. This is the unobserved counterfactual time. The factor $\exp(\psi)$ is the acceleration factor, which is assumed to be constant, regardless of when a patient switched treatments. This is an untestable assumption, which may or may not be true. If $\exp(\psi)$ is equal to 1, then there is no observed treatment effect.

Because of the counterfactual times produced for patients who switch treatment, the censoring times need to be re-estimated to be in the same scale. Let the new censoring time for patient i be denoted D_i, then $D_i = \min(C_i, C_i \exp(\psi))$, which is the minimum possible censoring time.

Estimation of ψ is commonly done using various methods, including the log-rank test, the Weibull test, and the Cox test. $Z(\psi)$ is the test statistic, and the optimal value of ψ is the value that sets $Z(\psi) = 0$.

RPSFTM has an R package, which contains a dataset that is used in this paper for the implementation of the model (Allison et. al., 2017). Additionally, authors have previously implemented RPSFTM in SAS, but only provided code for the log-rank test. Appendix 1 includes code for the log-rank test that is based on theirs and additionally, it provides code for the Weibull and Cox tests. The immdef dataset from the RPSFTM R package is used to demonstrate the code in SAS. The dataset contains the following variables:

1. **id:** subject identifier.
2. **def**: indicator for deferred treatment arm. Subjects in the control arm have a value of 1. subjects in the treatment arm have a value of 9.
3. **imm:** indicator for immediate treatment. Equal to 1 - def.
4. **censyrs:** time from randomization to end of trial.
5. **xo:** indicator variable that is equal to 1 if the patient switched treatment and 0 otherwise.
6. **xoyrs:** time from randomization to treatment crossover. If imm = 1 then xoyrs = 0.
7. **prog:** indicator variable that is equal to 1 if the patient has disease progression and 0 otherwise.
8. **progyrs:** time from randomization to disease progression.
9. **entry:** time from beginning of the study to subject's participation.

The `RPSFTM` macro has the following input arguments:

1. **data:** the input dataset.
2. **subjid:** a unique subject identifier.
3. **surv_time:** time to event endpoint.
4. **trt:** which treatment arm the subject is in (0 for reference treatment, 1 for other arm that subjects can crossover into).
5. **xo:** indicator variable to indicate if subjects crossed over during the study.
6. **xo_time:** time when the subject crossed over.
7. **dth_event:** indicator variable that is equal to 1 if the subject died during the study.
8. **ct:** continuous covariates.
9. **cl:** class covariates.
10. **method:** a variable to select which method to use for the RPSFT model. Options are `'logrank'`, `'coxph'`, and `'weibull'`.

To implement the RPSFT model, we first import the data into SAS:

```
proc import datafile = 'filepath\immdef.xlsx'
dbms = excel out = immdef;
run;
```

Next, we can use the `rpsftm` macro using the logrank test with the following code:

```
%rpsftm(data       = immdef,
        subjid     = id,
        surv_time  = censyrs,
        trt        = imm,
```

```
xo          = xo,
xo_time     = xo_yrs,
dth_event   = dthevent,
ct          = entry,
cl          = ,
method      = 'logrank');
```

Output 1 contains the point estimates and confidence intervals for ψ and the acceleration factor. 0 is contained within the interval for ψ, and 1 is contained within the interval for the acceleration factor. An acceleration factor of 1 indicates that there is no difference in overall survival times for the control and treatment groups, and so for these data, the treatment effect is not significant. Figure 1 is a plot of the Z-score for the logrank test versus ψ. The optimal value of ψ is the one with a Z-score equal to 0. Because the macro uses a grid-based search, the optimal value is linearly interpolated between the largest value < 0 and the smallest value > 0.

95% Confidence Interval for Psi

Obs	Point Estimate	Lower Limit (95% CI)	Upper Limit (95% CI)
1	-0.181	-0.363	0.072

95% Confidence Interval for Acceleration Factor (exp(Psi))

Obs	Point Estimate of exp(Psi)	Lower Limit (95% CI)	Upper Limit (95% CI)
1	0.83404	0.69559	1.07466

Output 1. Point Estimates and Confidence Intervals for Psi and for the Acceleration Factor for the Logrank Test.

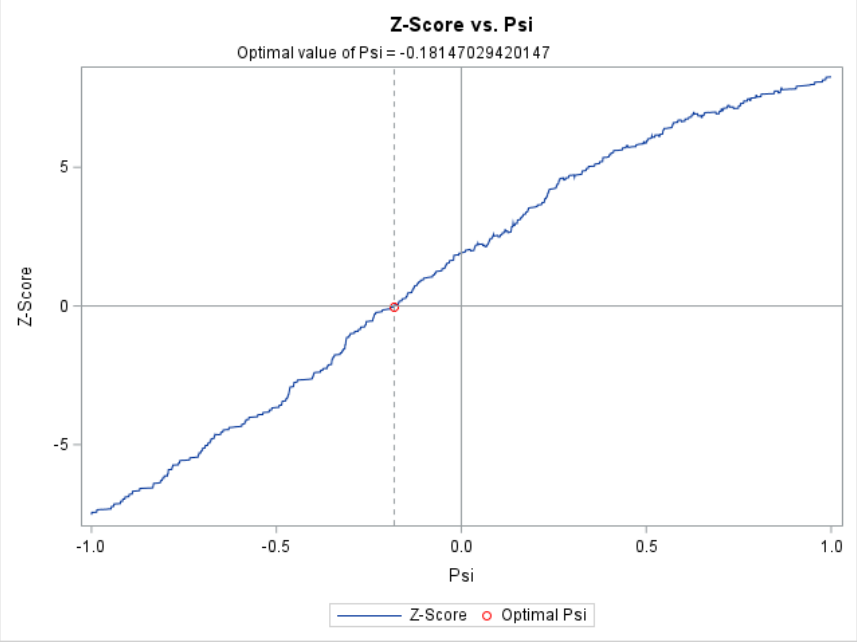

Figure 1. Plot of Z-Score vs. Psi for the Logrank Test.

The macro also allows for Weibull or Cox-PH models. Results are similar for all three models. Output 2 and Output 3 contain the point estimates and confidence intervals for ψ and the acceleration factor for the Weibull and Cox PH models, respectively.

95% Confidence Interval for Psi

Obs	Point Estimate	Lower Limit (95% CI)	Upper Limit (95% CI)
1	-0.182	-0.373	0.073

95% Confidence Interval for Acceleration Factor (exp(-Psi))

Obs	Point Estimate of exp(Psi)	Lower Limit (95% CI)	Upper Limit (95% CI)
1	1.19898	0.93053	1.43764

Output 2. Point Estimates and Confidence Intervals for Psi and for the Acceleration Factor for the Weibull Model.

95% Confidence Interval for Psi

Obs	Point Estimate of Psi	Lower Limit (95% CI)	Upper Limit (95% CI)
1	-0.181	-0.363	0.073

95% Confidence Interval for Acceleration Factor (exp(Psi))

Obs	Point Estimate of exp(Psi)	Lower Limit (95% CI)	Upper Limit (95% CI)
1	0.83419	0.69559	1.07573

Output 3. Point Estimates and Confidence Intervals for Psi and for the Acceleration Factor for the Cox PH Model.

INVERSE PROBABILITY OF CENSORING WEIGHTING MODEL

The inverse probability of censoring weighting (IPCW) model weights patients in the group with patient crossover in order to create a pseudo-population of patients that would have been observed if no crossover had occurred. This model assumes that there are no unobserved confounders, which is an untestable assumption. Additionally, time-dependent covariates should not be perfect, or nearly perfect predictors of cross-over; this affects the estimation of weights, ruining the analysis.

The IPCW model is a two-step process. It has been implemented in both R and SAS. The first step can be accomplished using either pooled logistic regression or a Cox proportional hazards (Cox PH) model to model the probability of remaining uncensored. In previous work in R, a Cox PH model is used for the first step, whereas in SAS, a logistic regression model is used, but was not implemented correctly. The macro presented in this paper estimates the weights using the Cox PH model and full code is available in Appendix 2. Standardized weights are estimated using the ratio of the probabilities estimated from two models, one containing only baseline covariates, the other containing baseline and time-varying covariates. The denominator of the weights is estimated using the baseline covariate model and the numerator of the weights are estimated using the time-varying covariate model. Given censoring times, T^C, baseline covariates, X, and time-varying covariates, Z, the formula for the weights is given by:

$$w_{ij} = \prod_{k=1}^{j} \frac{P(T_i^C > t_k \mid T_i^C > T_{k-1}, X = x_i)}{P(T_i^C > t_k \mid T_i^C > t_{k-1}, X = x_i, Z = z_i)}.$$

To explain the meaning of the weights, imagine two patients with identical characteristics who are randomized to the same treatment. It is reasonable to assume that the patients would have similar outcomes if given the same treatment. Now, assume that one of the patients switches to the other treatment at time *t*. The remaining patient is now given a weight of 2 for all timepoints past time *t* to adjust for the fact that the other patient was censored when they switched treatments. This seeks to maintain the characteristics of the population to negate the effects of treatment censoring.

The second step of the IPCW model is to fit a Cox proportional-hazards model for overall survival with the weights to estimate the effects that would have been observed in the absence of switching or discontinuation of treatment.

To implement the IPCW model, the data must be set up in the counting process style of input, where an observation corresponding to every censoring time due to treatment switching and every event time is present for each patient until either treatment switching, or an event occurs (SAS, 2023). Instructions and an example on how to format data sets to the counting process style of input are available in Kumar, 2018. From this dataset, the macro creates two additional datasets used for the progression free survival endpoint. The first being the dataset with only the baseline covariates and the second containing time-varying covariates up until the time of treatment censoring. A snapshot of the input dataset for two patients is given in Figure 2.

Obs	trt	subjid	age	tv1	tv2	xo_time	dth_time	tstart	tend
80	0	114005	32	13.3732	1.72566	3	6	0	1
81	0	114005	32	13.3732	1.72566	3	6	1	2
82	0	114005	32	13.3732	1.72566	3	6	2	3
83	0	114085	38	22.3058	1.27702	6	6	0	1
84	0	114085	38	22.3058	1.27702	6	6	1	2
85	0	114085	38	22.3058	1.27702	6	6	2	3
86	0	114085	38	22.3058	1.27702	6	6	3	4
87	0	114085	38	22.3058	1.27702	6	6	4	5
88	0	114085	38	12.4436	1.40415	6	6	5	6

Figure 2. Simulated Data in the Counting Process Style of Input.

The dataset contains the following variables:

`subjid`: unique subject ID.

`trt`: is the variable specifying which treatment the subject received.

`xo_time`: number of days since randomization that the subject switched treatments.

`xo_ind`: indicator for whether a subject crosses over.

`xo_cnsr`: variable to that indicates when a subject crosses over. Equal to 0 when **`tend`** < **`xo_time`** and 1 otherwise. If **`xo_ind`** = 0 for a subject, then **`xo_cnsr`** = 0 at all timepoints.

`dth_time`: number of months since randomization that the subject died.

`dth_ind`: indicator for whether a subject dies.

`dth_event`: variable to that indicates when a subject dies. Equal to 0 when **`tend`** < **`dth_time`** and 1 otherwise. If **`dth_ind`** = 0 for a subject, then **`dth_cnsr`** = 0 at all timepoints.

In addition, to the above variables, the dataset includes baseline and time-varying covariates. **dth_time** is the number of days since randomization until death of the patient, **age** is the age at randomization, **tv1** and **tv2** are time varying covariates, **tstart** is the number of days since randomization for the beginning of the interval, **tend** is the number of days since randomization for the end of the interval.

The ipcw macro can be ran using the following code:

```
%IPCW( data      = os,
       subjid    = subjid,
       endpnt    = dth_time,
       trt       = trt,
       ref       = 0,
       tstart    = tstart,
       tend      = tend,
       xo        = xo_time,
       xo_ind    = xo_ind,
       xo_cnsr   = xo_cnsr,
       dth_event = dth_event,
       bl_ct     = age,
       bl_cl     = ,
       tv_ct     = tv1 tv2,
       tv_cl     = );
```

To get the estimates of the weights, we run two models, one with only the baseline covariates, and one with the baseline and time varying covariates. This can be accomplished with the following code:

```
proc phreg data=&data outest=survmod covs(aggregate) covm;
  where &trt = &ref;
  class &bl_cl &tv_cl;
  model (&tstart,&tend)*xo_cnsr(0)= &bl_ct &bl_cl &tv_ct &tv_cl /rl;
  id &subjid;
  output out= probs_TV survival = w_TV;
run;
```

The text highlighted in yellow is not used for the baseline covariate model. Additionally, weights are only estimated for those receiving the control treatment (treatment 0). For the experimental treatment (treatment 1), all weights are set equal to 1 because they are not allowed to switch treatments. It is important to note that this model can accommodate patients from either arm switching to the other, but this paper focuses on scenarios where patients can switch only from the control arm to the experimental arm.

After getting the weights from the baseline and time-dependent models, we take the ratio of the weights to use in the final model. The Cox PH model is given by:

```
proc phreg data=surv_ph outest=survmod covs(aggregate) covm;
  class &bl_cl &tv_cl &trt(ref=first);
  model (&tstart,&tend)*dth_event(0)= &trt &bl_ct &bl_cl &tv_cl &tv_ct /rl;
  id &subjid;
  weight sw_t/norm;
run;
```

The macro gives the following results:

Analysis of Maximum Likelihood Estimates with Sandwich Variance Estimate										
Parameter		DF	Parameter Estimate	Standard Error	StdErr Ratio	Chi-Square	Pr > ChiSq	Hazard Ratio	95% Hazard Ratio Confidence Limits		Label
trt	1	1	-0.48779	0.22544	1.027	4.6819	0.0305	0.614	0.395	0.955	trt 1
age		1	-0.00893	0.00746	1.003	1.4345	0.2310	0.991	0.977	1.006	
tv1		1	-0.01370	0.00883	0.950	2.4075	0.1208	0.986	0.969	1.004	
tv2		1	-0.05942	0.09294	0.955	0.4087	0.5226	0.942	0.785	1.131	

In this example, the treatment provides a hazard ratio of 0.614 with a 95% confidence limit of (0.395, 0.955), and so in this simulated data, the treatment provides a significant reduction in the hazard of death.

COMPARING THE MODELS

Both models seek to account for patients who switch treatments during a study and both rely heavily on untestable assumptions that may not be true. The IPCW model requires that covariates are collected throughout the study, while the RPSFT model does not. Additionally, the data structure may need to be adjusted to conform with the counting process data format for the IPCW model, which is a complex and time consuming process. As such, the RPSFT model is oftentimes easier to implement. This model tests for an acceleration factor of the treatment, which is assumed to be constant, regardless of when a patient switches treatments. The IPCW model censors patients who switched treatments and reweights the remaining patients in that arm in an attempt to get a new group of patients who represent the original data. Again, this model relies heavily on the assumption that this reweighted group of patients is similar to the original group.

CONCLUSION

This paper presents two SAS macros to implement models to account for patient crossover in studies, being the RPSFT model and the IPCW model. It extends the options for the RPSFT model beyond what previous authors have provided, by including not only the logrank test, but also the Weibull and Cox PH tests as options to the macro. Additionally, the macro provides 95% confidence intervals for the acceleration factor, as well as a plot to visualize the optimal acceleration factor. This paper makes corrections for the IPCW model that previous authors have provided and additionally, it has expanded the options for this model by including the option to estimate weights using the Cox PH model.

REFERENCES

Allison A, White IR, Bond S. 2017 Dec 4. “rpsftm: An R Package for Rank Preserving Structural Failure Time Models”. *R Journal*. 9(2):342-353. rpsftm: An R Package for Rank Preserving Structural Failure Time Models - PMC (nih.gov).

Danner BJ, Sarkar I. 2018. “Implementing the Rank-Preserving Structural Failure Time Model in SAS and R”. PharmaSUG 2018 Conference Proceedings. Seattle, Washington.

Jiménez-Moro, José Luis and Gómez, Javier. 2014. “Inverse Probability of Censoring Weighting for Selective Crossover in Oncology Clinical Trials.” *Pharmaceutical Users Software Exchange (PHUSE)*, London, United Kingdom. SP02.pdf (lexjansen.com).

SAS. 2023. “*Counting Process Style of Input*.” Accessed January 1, 2023. PROC PHREG: Counting Process Style of Input :: SAS/STAT(R) 9.2 User's Guide, Second Edition

Kumar, P. 2018. “A case study of fitting time dependent covariate in cox model using SAS”. *Pharmaceutical Users Software Exchange (PHUSE)*, Frankfurt, Germany.

CONTACT INFORMATION

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

Brian Mosier
EMB Statistical Solutions, LLC
bmosier@embstats.com

APPENDIX 1

RPSFTM macro:

```
%macro
rpsftm(data,subjid,surv_time,trt,prog,prog_time,xo,xo_time,ct,cl,method,l_bou
nd,u_bound);

  data &data;
  set &data;
  length immcat $9;
    if &trt = 0 then do;
        imm = 0;
        def = 1-imm;
      immcat="Defer";
    end;
    else if &trt ne 0 then do;
        imm = 1;
        def = 1-imm;
      immcat="Immediate";
    end;
  run;

  data step1;
  set &data;

  do psi = &l_bound to &u_bound by 0.001;
    if imm = 1 then do;
      t_on = &prog_time;
        t_off = 0;
    end;
    else if imm = 0 then do;
      if &xo = 1 then do;
        t_on = &prog_time - &xo_time;
          t_off = &xo_time;
      end;
        else if &xo = 0 then do;
        t_on = 0;
      t_off = &prog_time;
      end;
    end;

    T = t_off + t_on;
    U = t_off + exp(psi)*t_on;
      cut_U = min(exp(psi)*&surv_time,&surv_time);
      if &prog = 1 then do;
        if U > cut_U then do;
          U = cut_U;
            event_U = 0;
        end;
        else do;
          event_U = 1;
        end;
      end;
      else do;
        event_U = 0;
        if U > cut_U then U = cut_U;
```

```
      end;
      output;
  end;

  run;

proc sort data = step1;
by psi &subjid;
run;

%if &method = 'logrank' %then %do;
  ods graphics off;
  ods html close;
  ods listing close;

  ods output logrankhomcov = hcov
             homtests      = htests
             homstats      = hstats;
  proc lifetest data = step1;
    by psi;
      time U*event_U(0);
      strata immcat / test=logrank;
  run;

  ods html;
  ods output;
  ods graphics;

  data step2;
   merge hstats hcov;
   by psi immcat;
   if immcat = "Defer";
   Z_value = logrank / sqrt(defer);
  run;

  data step3;
  set step2;
    label prob = 'Optimal Psi';
    by psi;
    retain last_Z_value;
    if Z_value eq 0 then output;
    if _n_ ne 1 then do;
      if Z_value * last_Z_value < 0 then do;
            psi_search = psi - Z_value/(Z_value-last_Z_value)*0.001;
            call symputx('psi_search',psi_search);
        output;
      end;
    end;
    last_Z_value = Z_value;
      keep psi psi_search prob;
  run;

  data step2;
  merge step2 step3;
  by psi;
  y=0;
  run;
```

```
  data step2;
  set step2;
  label z_value='Z-Score' y='Optimal Psi';
  run;

  ods graphics on;

  title 'Z-Score vs. Psi';
  proc sgplot data = step2;
  series x=psi y=z_value;
  scatter x=psi_search y=y / markerattrs=(color=CXFF0000);
  refline &psi_search / axis=x lineattrs=(pattern=ShortDash) label=("Optimal
value of Psi = &psi_search");
  refline 0 / axis=x;
  refline 0 / axis=y;
  xaxis label='Psi';
  yaxis label='Z-Score';
  run;
  quit;

  data htests1;
  merge htests step3;
  by psi;
  run;

  data htests1;
  set htests1;
  label psi_search='Optimal Psi' prb='Optimal Psi';
  if psi_search ne . then probchisq = 1;
  prb = probchisq;
  run;

  title 'Logrank p-Value vs. Psi';
  proc sgplot data = htests1;
  series x=psi y=probchisq;
  scatter x=psi_search y=prb / markerattrs=(color=CXFF0000);
  refline &psi_search / axis=x lineattrs=(pattern=ShortDash) label=("Optimal
value of Psi = &psi_search");
  refline 0 / axis=x;
  xaxis label='Psi';
  yaxis label='p-Value';
  run;
  quit;
  title ;

  data ci;
  set htests1;
  where probchisq >= 0.025;
  run;

  proc means data=ci noprint;
  var psi;
  output out=psi_ci min=min max=max;
  run;

  data psi_ci;
```

```
  set psi_ci;
  format point 8.3;
  point = &psi_search;
  keep point min max;
  run;

  title '95% Confidence Interval for Psi';
  proc print data=psi_ci label;
  var point min max;
  label point='Point Estimate' min='Lower Limit (95% CI)' max='Upper Limit
(95% CI)';
  run;

  data acc_fact;
  set psi_ci;
  exp_psi = exp(-point);
  exp_psi_min = exp(-max);
  exp_psi_max = exp(-min);
  keep exp_psi exp_psi_min exp_psi_max;
  run;

  title '95% Confidence Interval for Acceleration Factor (exp(-Psi))';
  proc print data=acc_fact label;
  var exp_psi exp_psi_min exp_psi_max;
  label exp_psi='Point Estimate of exp(Psi)' exp_psi_min='Lower Limit (95%
CI)' exp_psi_max='Upper Limit (95% CI)';
  run;

%end;
%else %if &method = 'weibull' %then %do;

 ods graphics off;
 ods html close;
 ods listing close;

 ods output ParameterEstimates=fit;
  proc lifereg data=step1;
    by psi;
      model U*event_U(0) = imm entry/dist=weibull;
  run;

  ods html;
  ods output;
  ods graphics;

  data fit;
  set fit;
  where parameter = upcase(symget('trt'));
  run;

  data fit1;
  set fit;
  by psi;
    retain last_estimate last_probchisq;
    if estimate eq 0 then output;
    if _n_ ne 1 then do;
      if estimate * last_estimate < 0 then do;
```

```
            psi_search = psi - estimate/(estimate-last_estimate)*0.001;
            probchisq_search = probchisq +
last_probchisq/(probchisq+last_probchisq)*(last_probchisq-probchisq);
            call symputx('psi_search',psi_search);
        output;
      end;
    end;
    last_estimate = estimate;
      last_probchisq = probchisq;
      keep psi psi_search probchisq_search;
  run;

  data fit;
  merge fit fit1;
  by psi;
  y=0;
  label y='Optimal Psi' chisq='Chi-Square';
  run;

  title 'Treatment Effect vs. Psi';
  proc sgplot data = fit;
  series x=psi y=estimate;
  scatter x=psi_search y=y / markerattrs=(color=CXFF0000);
  refline &psi_search / axis=x lineattrs=(pattern=ShortDash) label=("Optimal
value of Psi = &psi_search");
  refline 0 / axis=x;
  refline 0 / axis=y;
  xaxis label='Psi';
  yaxis label='Treatment Effect';
  run;

  title 'Chi-Square Statistic vs. Psi';
  proc sgplot data = fit;
  series x=psi y=chisq;
  scatter x=psi_search y=chisq / markerattrs=(color=CXFF0000);
  refline &psi_search / axis=x lineattrs=(pattern=ShortDash) label=("Optimal
value of Psi = &psi_search");
  refline 0 / axis=x;
  refline 0 / axis=y;
  xaxis label='Psi';
  run;

  data ci;
  set fit;
  where probchisq >= 0.025;
  run;

  proc means data=ci noprint;
  var psi;
  output out=psi_ci min=min max=max;
  run;

  data psi_ci;
  set psi_ci;
  format point 8.3;
  point = &psi_search;
  keep point min max;
```

```
  run;

  title '95% Confidence Interval for Psi';
  proc print data=psi_ci label;
  var point min max;
  label point='Point Estimate' min='Lower Limit (95% CI)' max='Upper Limit
(95% CI)';
  run;

  data acc_fact;
  set psi_ci;
  exp_psi = exp(-point);
  exp_psi_min = exp(-max);
  exp_psi_max = exp(-min);
  keep exp_psi exp_psi_min exp_psi_max;
  run;

  title '95% Confidence Interval for Acceleration Factor (exp(-Psi))';
  proc print data=acc_fact label;
  var exp_psi exp_psi_min exp_psi_max;
  label exp_psi='Point Estimate of exp(Psi)' exp_psi_min='Lower Limit (95%
CI)' exp_psi_max='Upper Limit (95% CI)';
  run;

%end;
%else %if &method = 'coxph' %then %do;

  ods output ParameterEstimates=fit;
  ods graphics off;
  ods html close;
  ods listing close;
  proc phreg data=step1;
       by psi;
       model U*event_U(0)= imm entry /rl;
  run;

  ods html;
  ods listing;

  data fit;
  set fit;
  where parameter = upcase(symget('trt'));
  run;

  data fit1;
  set fit;
  by psi;
    retain last_estimate last_probchisq;
    if estimate eq 0 then output;
    if _n_ ne 1 then do;
      if estimate * last_estimate < 0 then do;
            psi_search = psi - estimate/(estimate-last_estimate)*0.001;
            probchisq_search = probchisq +
last_probchisq/(probchisq+last_probchisq)*(last_probchisq-probchisq);
            call symputx('psi_search',psi_search);
        output;
      end;
```

```
    end;
    last_estimate = estimate;
      last_probchisq = probchisq;
      keep psi psi_search probchisq_search;
  run;

  data fit;
  merge fit fit1;
  by psi;
  y=0;
  label y='Optimal Psi' probchisq_search='Optimal Psi';
  run;

  ods graphics on;

  title 'Treatment Effect vs. Psi';
  proc sgplot data = fit;
  series x=psi y=estimate;
  scatter x=psi_search y=y / markerattrs=(color=CXFF0000);
  refline &psi_search / axis=x lineattrs=(pattern=ShortDash) label=("Optimal
value of Psi = &psi_search");
  refline 0 / axis=x;
  refline 0 / axis=y;
  xaxis label='Psi';
  yaxis label='Treatment Effect';
  run;

  title 'Chi-Square Statistics vs. Psi';
  proc sgplot data = fit;
  series x=psi y=chisq;
  scatter x=psi_search y=chisq / markerattrs=(color=CXFF0000);
  refline &psi_search / axis=x lineattrs=(pattern=ShortDash) label=("Optimal
value of Psi = &psi_search");
  refline 0 / axis=x;
  refline 0 / axis=y;
  xaxis label='Psi';
  run;

  data ci;
  set fit;
  where probchisq >= 0.025;
  run;

  proc means data=ci noprint;
  var psi;
  output out=psi_ci min=min max=max;
  run;

  data psi_ci;
  set psi_ci;
  format point 8.3;
  point = &psi_search;
  keep point min max;
  run;

  title '95% Confidence Interval for Psi';
  proc print data=psi_ci label;
```

```
  var point min max;
  label point='Point Estimate of Psi' min='Lower Limit (95% CI)' max='Upper
Limit (95% CI)';
  run;

  data acc_fact;
  set psi_ci;
  exp_psi = exp(-point);
  exp_psi_min = exp(-max);
  exp_psi_max = exp(-min);
  keep exp_psi exp_psi_min exp_psi_max;
  run;

  title '95% Confidence Interval for Acceleration Factor (exp(-Psi))';
  proc print data=acc_fact label;
  var exp_psi exp_psi_min exp_psi_max;
  label exp_psi='Point Estimate of exp(Psi)' exp_psi_min='Lower Limit (95%
CI)' exp_psi_max='Upper Limit (95% CI)';
  run;


%end;

%mend rpsftm;
```

APPENDIX 2

```
%macro
IPCW(data,subjid,dth_time,trt,ref,tstart,tend,xo_time,xo_ind,xo_cnsr,dth_ind,
dth_event,bl_ct,bl_cl,tv_ct,tv_cl);
  data &data;
  set &data;
    if &tstart >= &dth_time then delete;
    if &xo_ind = 1 then do;
      if &tstart >= &xo_time then delete;
    end;
  run;

  data weight_est;
  set &data;
  where &trt = &ref;
  run;

  proc sort data=weight_est;
  by &subjid &tstart;
  run;

  proc phreg data=weight_est outest=survmod covs(aggregate) covm;
    by &trt;
      class &bl_cl;
      model (&tstart,&tend)*xo_cnsr(0)= &bl_ct &bl_cl/rl;
      id &subjid;
      output out= probs_BL survival = w_BL;
  run;

  proc phreg data=weight_est outest=survmod covs(aggregate) covm; * covs
option and id statement adjust for repeated measures;
    by &trt;
    class &bl_cl &tv_cl;
      model (&tstart,&tend)*xo_cnsr(0)= &bl_ct &bl_cl &tv_ct &tv_cl /rl;
      id &subjid;
      output out= probs_TV survival = w_TV;
  run;

  data probs_BL;
    set probs_BL;
    keep &subjid &trt w_BL &tstart &tend;
  run;

  data probs_TV;
    set probs_TV;
    keep &subjid &trt w_TV &tstart &tend;
  run;

  data weights;
  merge probs_TV probs_BL;
    by &subjid &tstart &tend;
    sw_t = w_BL/w_TV;
      w_BL = 1/w_BL;
      w_TV = 1/w_TV;
  run;
```

```
  * Truncation Percentiles;
  %let pctl_l = 1;
  %let pctl_u = 99;
  proc univariate data=weights noprint;
  var sw_t;
  output out=trunc
  pctlpts = &pctl_l &pctl_u
  pctlpre = p_;
  run;

  data trunc;
  set trunc;
  &trt = 0;
  output;
  run;

  data weights;
  merge weights trunc;
  by &trt;
  run;

  data weights;
  set weights;
    if sw_t < p_&pctl_l then sw_t = p_&pctl_l;
    else if sw_t > p_&pctl_u then sw_t = p_&pctl_u;
  run;

  data surv_ph;
      merge &data weights;
      by &trt &subjid &tstart &tend;
      if &trt ne &ref then do;
      sw_t = 1;
        w_TV = 1;
      end;
      log_w_TV = log(w_TV);
      log_sw_t = log(sw_t);
  run;

  proc sort data=surv_ph;
  by &trt;
  run;

  proc phreg data=surv_ph outest=survmod covs(aggregate) covm; * covs option
and id statement adjust for repeated measures;
      title 'Final Model using weights from Cox PH Model';
      class &bl_cl &tv_cl &trt(ref=first);
      model (&tstart,&tend)*dth_event(0)= &trt &bl_ct &bl_cl &tv_cl &tv_ct
/rl;
      id &subjid;
      weight sw_t/norm;
  run;
%mend IPCW;
```