

Implementation of IPCW in Oncology Trials and Enhanced SAS Macro Tools

Jiayi Chen, Meiqing Huang, Shang Hansoh Biomedical R&D Inc.

ABSTRACT

Treatment crossover from the control arm to the experimental arm is common in oncology clinical trials and can complicate interpretation of overall survival (OS). Inverse probability of censoring weighting (IPCW) is a recognized approach for estimating a hypothetical no-switching treatment effect, but its use in production clinical programming requires more than a statistical model statement. Programmers must prepare interval-level data, align time-dependent covariates, estimate and assess weights, fit weighted survival models, and generate uncertainty estimates that reflect the full analysis workflow.

This paper presents an enhanced SAS macro framework for IPCW implementation in oncology trials. Although several R packages provide useful methodological references, SAS-native implementation remains important in clinical trial reporting environments. Building on the SAS macro framework described by Mosier (2023), we propose enhancements for time-dependent covariate handling, quality control, and bootstrap confidence interval estimation for the adjusted hazard ratio. The resulting workflow is intended to make IPCW analyses more transparent, reproducible, and practical for statistical programming teams. We also developed an independent R script of the complete IPCW workflow to assess consistency between the R and SAS implementations.

INTRODUCTION

Overall survival serves as a pivotal endpoint in oncology trials due to its clinical relevance and direct interpretability. However, evaluating this metric is confounded when patients in the control arm cross over to the treatment arm during follow-up. Such crossover events frequently result from disease progression, inadequate therapeutic response, or ethical imperatives facilitating access to promising interventions.

Although intent-to-treat (ITT) analysis is fundamental to preserving the integrity of randomized treatment comparisons, it may not adequately address the hypothetical scenario of overall survival effects in the absence of switching when a substantial proportion of control patients receive experimental therapy. Consequently, we recommend implementing supplementary treatment-switching adjustment analyses to ensure robust clinical interpretation and comprehensive sensitivity assessment.

Inverse probability of censoring weighting (IPCW) is a robust causal inference method designed to adjust for treatment switching. Conceptually, observed data are reweighted to reconstruct a hypothetical scenario wherein switching did not occur, thereby isolating the true treatment effect. This paper delineates a practical IPCW programming workflow in SAS, detailing the end-to-end process encompassing counting-process data construction, stabilized weight estimation, weighted Cox modeling, quality control (QC), and bootstrap confidence interval estimation, thus providing a comprehensive guide for statistical programmers and biostatisticians.

METHODOLOGICAL CONTEXT

Treatment switching constitutes a post-randomization event that introduces substantial complexity into survival analysis. While simple analytical approaches offer useful reference points, they are inherently biased when estimating the effect of continuous treatment exposure. For instance, as-treated analyses are subject to confounding because post-randomization treatment choices are not randomized. Similarly, naïve censoring at the time of switch, frequently employed in per-protocol analyses, relies on the assumption of non-informative censoring. Should patients who switch exhibit a systematically different prognosis compared to those who do not, this approach induces informative censoring bias.

Within the ICH E9(R1) estimand framework, IPCW supports a hypothetical strategy by addressing what the overall survival treatment effect would have been in the absence of relevant switching. To address this inquiry, IPCW constructs a counterfactual population that approximates the control group under the

assumption that no participants had switched treatment. This method extends naïve censoring techniques used in per-protocol analysis, where patients are censored at the time of treatment switching. However, such naïve censoring introduces informative censoring bias because outcomes for switchers and non-switchers are likely to differ. To mitigate this, patients with similar clinical characteristics who have not yet switched are assigned higher weights, thereby enabling them to represent their censored counterparts in subsequent analyses. These weights are derived from the inverse of the predicted probability of remaining unswitched at any given time, conditional on observed baseline and time-varying covariates. By modeling this weighted pseudo-population, differences between censored switchers and uncensored non-switchers are effectively eliminated, allowing for the estimation of the true treatment effect.

The interpretation of IPCW relies on three core identification conditions. First, conditional exchangeability requires that the decision to switch or remain unswitched be independent of potential survival outcomes, after conditioning on observed baseline and time-dependent covariates. Second, the positivity assumption necessitates a nonzero probability of remaining unswitched within the analysis population. Third, the weight model must be correctly specified, incorporating relevant time-dependent covariates, nonlinear terms, or interactions where appropriate, alongside the survival-model structure utilized for the censoring process.

IPCW WORKFLOW

The IPCW method employs two linked models. Model 1, Estimate Weights, defines a switching model within each relevant arm and estimates the probability that a patient remains unswitched by time t , based on baseline and time-varying covariates. Model 2, Compare Adjusted Survival Between Arms, incorporates the estimated weights into an outcomes model. The workflow proceeds from source data to interval construction, weight estimation, weight truncation, weighted Cox regression, quality control, and bootstrap inference.

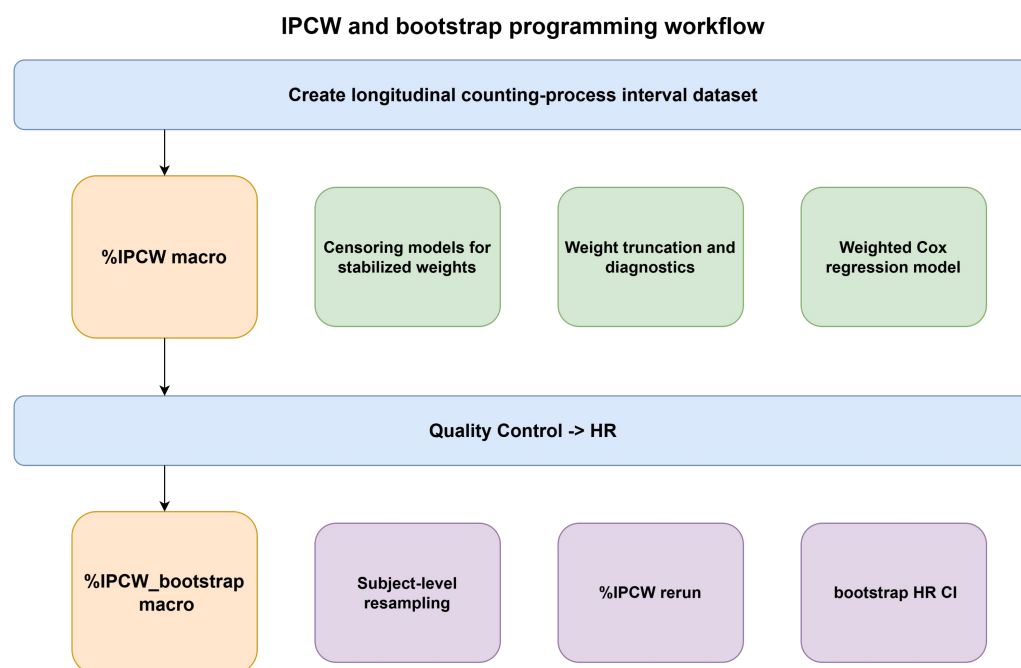


Figure 1. IPCW and bootstrap programming workflow.

DATA PREPARATION

The analytical workflow begins with source data ingestion. Baseline characteristics and treatment switching indicators are extracted from the ADSL dataset, whereas time-dependent covariates are derived from the ADQS and ADTR datasets. Event indicators are obtained from the ADTTE dataset. Table 1 outlines the integration of these sources into the counting-process style analysis dataset.

Table 1. Clinical data sources used to construct the IPCW interval dataset.

Source	Example variables	Role in IPCW
ADSL	SUBJID, AGE, SEX, TRT01P, STRAT1R, SWITCH, SWITCHDY	Baseline covariates, randomized arm, stratification, switch indicator and switch date
ADQS / ADTR	SUBJID, ADY, ECOG, SUMDIAM	Time-varying covariates mapped to the subject timeline
ADTTE	SUBJID, AVAL, CNSR	Overall survival event and censoring status

The subsequent task involves constructing longitudinal counting-process data. Each record comprises a subject identifier, interval start and stop times, event and switch statuses, randomized treatment, baseline covariates, and time-dependent covariates applicable to the specific interval. Intervals are typically split at times corresponding to events, switches, or covariate changes. Each record represents a left-open, right-closed interval, denoted as $(t_{\text{start}}, t_{\text{stop}}]$. Covariates remain constant within each interval. When multiple time-varying covariates change on different schedules, the union of all change dates defines the interval boundaries. Values are carried forward only from previously observed data. Intervals following a switch are excluded, which is equivalent to censoring at the time of the switch.

IPCW MODEL 1: WEIGHT ESTIMATION

Stabilized weights require two censoring models. The numerator model typically includes only baseline covariates, estimating the probability of remaining unswitched based on information available at randomization. The denominator model incorporates both baseline and time-dependent covariates to estimate the probability of remaining unswitched given the current clinical state. Because baseline effects are partially offset between the numerator and denominator, the final weight is primarily driven by time-dependent confounding factors, such as disease progression or ECOG.

Weight diagnostics and truncation assessments are standard outputs. Generated reports must include baseline weights (w_{bl}), time-varying weights (w_{tv}), stabilized weights (sw_{t} , where $sw_{\text{t}} = w_{\text{bl}} / w_{\text{tv}}$), post-truncation weights, and truncation cut points, along with descriptive statistics (mean, standard deviation, minimum, maximum, and selected percentiles). The presence of extreme weights necessitates a systematic review of covariate definitions, sparse categorical levels, model specification, potential non-positivity violations, and prespecified weight truncation criteria.

HOW DATA CHANGES WEIGHTS

The presentation uses Participant A to illustrate how clinical data drive interval-level weights. Table 2 integrates interval data with w_{tv} , w_{bl} , and sw_{t} . The w_{bl} column represents the numerator probability derived from baseline covariates, whereas the w_{tv} column denotes the denominator probability after incorporating time-dependent covariates.

Table 2. Participant A interval-level weights from the presentation example.

SUBJID	AGE	TSTART	TSTOP	DTH_CNSR	X0_CNSR	ECOG	SUMDIAM	w_tv	w_bl	sw_t
...	...	...	...	...	...	...	...	...	...	...
A	17	62	71	1	1	0	31.7	0.889..	0.892..	0.998..
A	17	71	78	1	1	0	31.7	0.850..	0.856..	1.003..
A	17	78	82	1	1	0	31.7	0.850..	0.856..	1.004..
A	17	82	83	1	1	0	33.4	0.848..	0.856..	1.009..

SUBJID	AGE	TSTART	TSTOP	DTH_CNSR	X0_CNSR	ECOG	SUMDIAM	w_tv	w_bl	sw_t
...	...	...	...	...	...	...	...	...	...	...

At a specific time cut-point, such as $tstop=71$, when events or switches occur among some control-arm participants, both numerator and denominator probabilities decline concurrently, partially offsetting each other; consequently, sw_t remains close to 1. When a retained patient exhibits a clinical state associated with higher switching risk, such as an increase in the sum of tumor diameters during the interval from 82 to 83, the denominator probability may decrease more than the numerator, causing the stabilized weight to increase. Finally, once the patient actually switches treatment or experiences the event (i.e., the indicator $X0_CNSR$ becomes 0), they are artificially censored, exit the no-switching risk set, and cease contributing follow-up time to the weighted Cox model.

IPCW MODEL 2: COMPARE ADJUSTED SURVIVAL BETWEEN ARMS

Following weight diagnostics, IPCW Model 2 incorporates the estimated weights into the outcome analysis. In a single-direction treatment-switching setting, intervals in the treatment arm are typically assigned a weight of 1, whereas pre-switch intervals in the control arm receive stabilized weights (without truncation). Because the counting-process data structure allows a single subject to contribute multiple time intervals with varying weights, the weighting mechanism inherently induces within-subject correlation. To properly account for this correlation, standard errors must be calculated using robust variance estimation, such as the sandwich estimator.

The existing %IPCW macro (Mei, 2025) interface requires explicit inputs. These include the interval dataset, switching direction, subject ID, treatment contrast, interval variables, switching-censoring variable, OS event/censoring variable, covariate lists, weight truncation percentiles, and output datasets:

```
%IPCW(
  data=,
  direction=,
  subjid=,
  trt=,
  ref=,
  tstart=,
  tend=,
  xo_cnsr=,
  dth_cnsr=,
  bl_ct=,
  bl_cl=,
  tv_ct=,
  tv_cl=,
  pct_l=,
  pct_u=,
  outest=,
  outweights=
);
```

These parameters define the analysis contract. The data parameter specifies the interval-level input dataset. Direction distinguishes between single-direction and dual-direction crossover. Subjid identifies patients, while trt and ref define the treatment contrast and reference arm. Tstart and tend define the counting-process intervals. Xo_cnsr identifies censoring due to switching, whereas dth_cnsr identifies the OS event or censoring status. The covariate parameters bl_ct, bl_cl, tv_ct, and tv_cl separate baseline continuous, baseline categorical, time-dependent continuous, and time-dependent categorical covariates, respectively.

The algorithmic logic was implemented in R to facilitate technical verification. This implementation exposes intermediate components, including the numerator and denominator models. Table 3 presents the syntax mapping between R and SAS for counting-process data and robust weighted survival models:

```
den_fit <- survival::coxph(Surv(tstart, tend, xo_cnsr) ~ bl_cov + tv_cov,
                           data = anal, ties = 'efron')
num_fit <- survival::coxph(Surv(tstart, tend, xo_cnsr) ~ bl_cov,
                           data = anal, ties = 'efron')
```

Table 3. R and SAS syntax mapping for counting-process data and robust weighted survival models.

Software: function	Counting process for weight estimation	Robust standard errors for weighted survival analysis
R: survival::coxph	Surv(tstart, tend, xo_cnsr) ~ <covariates>	formula = Surv(tstart, tend, dth_cnsr) ~ <covariates> + cluster(subjid), weights = bw
SAS: PROC PHREG	model (tstart, tend)*xo_cnsr = <covariates>	model (tstart, tend)*cnsr = <covariates> / rl; id subjid; weight bw / normalize;

QUALITY CONTROL

Quality control procedures must be executed before interpreting the adjusted hazard ratio. Verification steps ensure that the start time (tstart) precedes the stop time (tstop), subject intervals are non-overlapping, and post-switch records are excluded when artificial censoring is applied. Furthermore, event and censoring codes must align with the model syntax, and time-varying covariates must utilize only information available at or before the respective interval. Because missing covariate data can alter model convergence and weight distributions, these discrepancies must be resolved before weight estimation.

A unit-weight quality control check serves as the primary model-level diagnostic. All interval weights are set to unity, and the weighted Cox model is re-estimated using the identical interval dataset and model specification:

```
proc phreg data=surv_qc covs(aggregate) covm;
  class &trt(ref=&ref) &bl_cl;
  model (&tstart,&tend)*&dth_cnsr(1)= &trt &bl_ct &bl_cl / rl;
  id &subjid;
  weight sw_qc / normalize;
run;
```

The resulting output should replicate the naive censor-at-switch analysis. Any deviation from this expectation indicates an issue with interval construction or coding, rather than a defect in the IPCW weighting mechanism.

BOOTSTRAP

The bootstrap method is employed to capture uncertainty arising from both the numerator and denominator models of the IPCW procedure. Re-fitting only the final Cox model is insufficient for this purpose. In each replicate, the workflow generates a bootstrap sample, re-estimates the numerator and denominator models, recomputes stabilized weights, applies weight truncation, refits the weighted Cox model, and records the converged treatment log-hazard ratio. Figure 2 summarizes the bootstrap loop used to obtain the distribution of the adjusted effect.

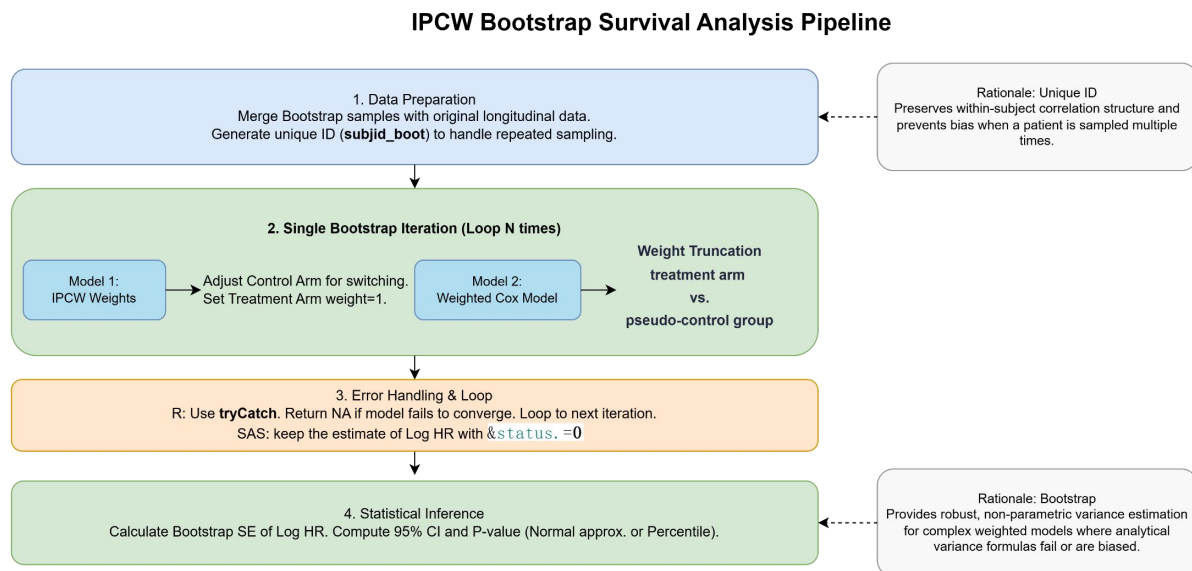


Figure 2. IPCW Bootstrap Survival Analysis Pipeline.

Resampling is performed at the subject level rather than the interval level to preserve longitudinal history, ensuring that all intervals associated with a selected patient are retained. Sampling is stratified by treatment arm to maintain the integrity of the randomized structure. When a subject is selected multiple times, a unique bootstrap subject identifier is generated to treat each copy as a distinct individual in the analysis.

For each replicate, bootstrap subject identifiers are reconstructed, numerator and denominator models are re-estimated, stabilized weights are recomputed and truncated, and the weighted Cox model is refitted. The treatment log-HR is stored only upon model convergence; non-converged estimates are recorded with an explicit failure reason rather than being silently excluded.

Bootstrap estimates are summarized on the log-HR scale by combining all converged treatment-effect estimates. The default approach uses the 2.5th and 97.5th percentiles of the empirical bootstrap log -HR distribution, exponentiating these values to obtain the 95% CI for the HR. In practice, the workflow can employ a two-stage execution strategy: a small bootstrap run (e.g., $n=100$) for program debugging and runtime assessment, followed by a larger run (e.g., $n=1000$) for the primary or sensitivity analysis.

Example calls are:

```
%IPCW_bootstrap(
  data=anall,
  outdat=fhr_100,
  n=100,
  seed=20262026,
  unit=subjidn,
  trt=,
  tstart=,
  tend=,
  xo_cnsr=,
  dth_cnsr=,
  bl_ct=,
  bl_cl=,
  tv_ct=,
  tv_cl=,
```

```

    robustv=Y,
    outliere=N,
    tpvlu= Y
);

%IPCW_bootstrap(
    data=anall,
    outdat=fhr_1000,
    n=1000,
    seed=20262026,
    unit=subjidn,
    trt=,
    tstart=,
    tend=,
    xo_cnsr=,
    dth_cnsr=,
    bl_ct=,
    bl_cl=,
    tv_ct=,
    tv_cl=,
    robustv=Y,
    outliere=Y,
    tpvlu=N
);

```

Note: For brevity, only the function call is shown here. The complete function implementation is provided in the Appendix.

The corresponding R benchmark resamples subjects with replacement, reconstructs each bootstrap dataset by retaining all intervals for selected subjects, re-estimates IPCW weights, and refits the weighted Cox model in each replicate. Bootstrap SE, model-based CI, percentile CI, or BCa CI may be used if the analysis plan prespecifies the method :

```

bootstrap <- bootstrap_ipcw (
    data, R = 500, verbose = TRUE,
    sw_formula_numer = sw_event ~ age + sex + ecog,
    sw_formula_denom = sw_event ~ age + sex + ecog + tstart,
    sw_method = "cox",
    outcome_formula =
        Surv(tstart, tstop, event) ~ treat + age + sex + ecog +
strata(stratlrn),
    id_var = "id",
    treat_var = "treat",
    sw_var = "swtrt",
    sw_time_var = "swtrt_time",
    time_var = "tstart")

```

PROGRAMMING CONSIDERATIONS

Several implementation issues must be addressed before interpreting macro outputs. First, interval boundary validity is essential. Zero-length intervals may occur when events such as death, censoring, treatment switching, or covariate changes coincide with an intervalstart. Counting-process models often fail when start and stop times are identical. Any minor temporal adjustment, such as adding one day or a half-day increment for same-day events, must be explicitly documented.

Second, a prespecified protocol for handling missing time-varying covariates is required. Although more complex imputation methods may be considered, their integration with bootstrap inference increases computational burden.

Third, the runtime associated with bootstrap procedures requires careful planning. Each replicate necessitates repeating weight estimation and weighted outcome modeling. The number of replicates should balance the stability of interval estimates against computational runtime and project timelines.

CONCLUSION

The IPCW methodology begins by structuring interval data, mapping all events, treatment switches, censoring times, and covariate modifications onto a left-open, right-closed temporal framework. Before interpreting weighted outcomes, a quality control procedure using unit weights verifies that the interval dataset accurately replicates the results of a naive censor-at-switch analysis.

Subsequently, stabilized weights are estimated, their distributions examined, and weight truncation applied according to prespecified criteria. The weighted Cox proportional hazards model incorporates subject-level clustering and robust sandwich variance estimation. For bootstrap-based inference, subjects are sampled within their respective treatment arms, and the complete IPCW pipeline is re-executed for each replicate.

A robust IPCW implementation is defined not by a single model specification, but by a reproducible workflow integrating source data management, interval construction, weight estimation, weighted Cox modeling, quality control, and bootstrap inference.

REFERENCES

- Watkins C, Kleine E, Miranda M, Bourmaud E, Doehring O. Further Practical Guidance on Adjusting Time-To-Event Outcomes for Treatment Switching. *Pharmaceutical Statistics*. 2025;24(4):e70019. doi:10.1002/pst.70019.
- Huang M, Harite S, Zhang L, Ma H. Implementation of the Inverse Probability of Censoring Weighting (IPCW) Model in Oncology Trials. *PharmaSUG 2025*, Paper ST-343.
- Mosier M. Survival Methods for Crossover in Oncology Trials: RPSFTM and IPCW. *PharmaSUG*. 2023.
- Liang Z, Xu L, Li K, Yu M, An S. Comparison of different treatment-switching analysis methods in clinical trials. *Journal of Southern Medical University*. 2025;45(05):1093-1102.
- Robins JM, Finkelstein DM. Correcting for Noncompliance and Dependent Censoring in an AIDS Clinical Trial with IPCW Log-Rank Tests. *Biometrics*. 2000;56(3):779-788.

APPENDIX

This appendix details the programming implementations for bootstrap resampling and confidence interval estimation within the IPCW workflow. The complete R code is omitted from this manuscript; interested readers may contact the authors to obtain the relevant scripts.

SAS CODE FOR IPCW BOOTSTRAP ANALYSIS

```
*****
*****;
* * IPCW Bootstrap macro to perform IPCW analysis using stratified balanced
* bootstrap resampling. It estimates the treatment effect (Hazard Ratio)
* and its confidence intervals/p-values by iteratively calling the base
* %IPCW macro. It includes advanced options for robust variance filtering,
* outlier removal, and flexible inference methods.
*
*****
*****;
* Macro variable:
* data: input dataset with time-dependent covariates, fixed baseline
*       covariates and time to event data in counting process style.
* outdat: output dataset containing the final estimated HR, 95% confidence
*         limits (hr_lci, hr_uci), p-value, and log(HR).
* n: number of bootstrap resamples.
* seed: random number seed for the bootstrap resampling to ensure reproducibility.
* unit: variable for the sampling unit
* trt: variable for treatment group
* tstart: variable for start time of the time intervals
* tend: variable for end time of the time intervals
* xo_cnsr: time to crossover censoring variable (1: censoring, 0: crossover)
* bl_ct: variables for baseline continuous covariates
* bl_cl: variables for baseline categorical covariates
* tv_ct: variables for time-dependent continuous covariates
* tv_cl: variables for time-dependent categorical covariates
* robustv: flag to apply robust variance filtering. "robustv = Y" excludes
*          bootstrap samples where the robust standard error ratio is missing.
* outliere: flag to remove outlying bootstrap log(HR) estimates.
*          "outliere = Y" removes outliers using the 1.5*IQR rule and
*          checks for normality; "outliere = N" retains all estimates.
* tpvlu: flag for the method of calculating p-value and confidence interval.
*       "tpvlu = Y" uses the t-distribution based on the bootstrap
*       standard error; "tpvlu = N" uses the percentile method for CI
```

```

*           and normal approximation for the p-value.
*****
*****;

%macro IPCW_bootstrap(data=, outdat=, n=, seed=, unit=, direction=,
    trt=, tstart=tstart, tend=tstop, xo_cnsr=, dth_cnsr=, bl_ct= , bl_cl=,
    tv_ct=, tv_cl=, robustv=, outliere=N, tpvlu= N);

proc sort data=&data. ;by &trt.n;run;

proc surveyselect data=&data. out=sample_all
method=balbootstrap outhits seed=&seed. reps=&n. ;
samplingunit &unit. ;
strata &trt.n;
run;

%do i=1 %to &n. ;
    data sample0;
        set sample_all;
        if replicate = &i. ;
        order=_n_;
    run;
    proc sql;
        create table sample as
        select *, min(tstart) as tstart_min from sample0
        group by subjdn order by order;
    Quit;

    data sample_re;
        length subjdn_re $200. ;
        set sample;
        by order;
        retain subjdn_re;
        ***Create unique bootstrap subject identifier;

        if tstart=tstart_min then subjdn_re=_n_;

    run;
    dm "out;clear;log;clear";
    ods results off;
    %IPCW(data=sample_re,
        direction=&direction.,
        subjdn=subjdn_re,
        trt=&trt., ref="&control.", tstart=tstart, tend=tstop,
        xo_cnsr=&xo_cnsr., dth_cnsr=&dth_cnsr.,

```

```

        bl_ct=&bl_ct. ,bl_cl=&bl_cl.,
        tv_ct=&tv_ct.,tv_cl=&tv_cl.);
    ods results on;
***Check whether the IPCW model converged;

    data _null_;
        set converg;
        call symputx('status',status);
    run;
    %if &status.=0 %then %do;
        *95% CI for HR;
        data _ParameterEstimates&i.;
            set ParameterEstimates;
            replicate = &i.;
        run;
    %end;
    %if %eval(&i.)>1 %then %do;
        proc append base=_ParameterEstimates1 data=_ParameterEstimates&i. force;
        run;
        proc delete data=_ParameterEstimates&i. converg;
        run;
    %end;
%end;

data hr_comb;
    set _ParameterEstimates1;
    where parameter=upcase("&trt.")    and %if &robustv.=Y %then not;
missing(stderrratio);
    _type_=0;
run;

***Remove outlying bootstrap log HR estimates if requested;
%if &outliere.=Y %then %do;
    proc means data=hr_comb noprint;
        var estimate;
        output out=hr_comb_d q1=q1 q3=q3;
    quit;

    data hr_comb1;
        merge hr_comb hr_comb_d;
        by _type_;
        iqr = q3 - q1;
        lower_bound = q1 - 1.5 * iqr;
        upper_bound = q3 + 1.5 * iqr;

```

```

        if lower_bound <= estimate <= upper_bound;
        drop q1 q3 iqr lower_bound upper_bound;
run;

***Check the distribution of Log(HR);
ods trace on;
ods graphics on;
ods output TestsForNormality=TestsForNormality;
proc univariate data=hr_combl plot normal;
    var estimate ;
    qqplot;
    output out=estimate_un
           pctlpts = 2.5 97.5
           pctlpre = P_;
run;
ods graphics off;
ods trace off;
***Check whether the Log(HR) follows a normal distribution;
data _null_;
    set TestsForNormality;
    where test="Shapiro-Wilk";
    call symputx('hynormality', ifn(pvalue>=0.05, 1, 0));
run;

%if &hynormality=1 %then %do;
proc univariate data=hr_combl noprint;
    var estimate;
    output out=trunc pctlpts = 2.5 97.5 pctlpre = p_;
    by _type_;
run;
proc means data=hr_combl noprint;
var estimate;
output out=hr_summ mean=mean std=std;
quit;

data &outdat.;
    merge hr_ori trunc hr_summ;
    hr_lci=exp(p_2_5);
    hr_uci=exp(p_97_5);
    p_value=2*(1-probnorm(abs(log_hr/std)));
run;
%end;
%if &outliere.=N %then %do;

```

```

%if &tpvluе.=Y %then%do;
    proc means data=hr_comb noprint;
    var estimate;
    output out=hr_summ mean=mean std=std n=n_boot;
    quit;

    data &outdat.;
        merge hr_ori hr_summ;
        ***the bootstrap estimate of standard error of
log(hr)(estimate) is the standard deviation of the bootstrap replications;
        se_botstrap=std;
        t_crtit=tinv(1-0.05/2,n_boot-1);
        hr_lci=exp(log_hr-t_crtit*se_botstrap);
        hr_uci=exp(log_hr+t_crtit*se_botstrap);
        p_value=2*(1-probt(abs(log_hr/se_botstrap),n_boot-1));
        keep hr hr_lci hr_uci p_value log_hr;
    run;
%end;
%else %do;
    proc univariate data=hr_comb noprint;
    var estimate;
    output out=trunc pctlpts = 2.5 97.5 pctlpre = p_;
    run;
    proc means data=hr_comb noprint;
    var estimate;
    output out=hr_summ mean=mean std=std ;
    quit;

    data &outdat.;
        merge hr_ori trunc hr_summ;
        hr_lci=exp(p_2_5);
        hr_uci=exp(p_97_5);
        p_value=2*(1-probnorm(abs(log_hr/std)));
        keep hr hr_lci hr_uci p_value log_hr;
    run;
%end;
%end;
%mend IPCW_bootstrap;

```

CONTACT INFORMATION

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

Jiayi Chen
Chenji8@hspharm.com