Improved two-stage estimation to adjust for treatment switching in randomised trials: g-estimation to address time-dependent confounding

SUPPLEMENTARY MATERIALS

Appendix A: Simulation study data generating mechanism

Simple scenarios

Below we provide a step-by-step description of the data generation procedure used for the simple scenarios. Box A1 summarises this.

Step 1. Underlying overall survival times

We used Crowther and Lambert's [51] general survival simulation framework to simulate survival in the absence of treatment with a 2-component mixture Weibull baseline survival function, allowing us to simulate realistic hazard functions. The baseline survivor function $S_0(t)$ can be written as:

$$S_0(t) = p\exp(-\lambda_1 t^{\gamma_1}) + (1-p)\exp(-\lambda_2 t^{\gamma_2}) \quad \text{(A1)}$$

where $\lambda_1, \lambda_2 > 0$ and $\gamma_1, \gamma_2 > 0$ are scale and shape parameters, respectively. The contribution of the first Weibull to the survival model is represented by p, with $0 \leq p \leq 1$, and $1-p$ represents the contribution of the second Weibull. The related baseline hazard function is:

$$h_0(t) = \frac{\lambda_1\gamma_1 p t^{\gamma_1-1}\exp(-\lambda_1 t^{\gamma_1}) + \lambda_2\gamma_2(1-p)t^{\gamma_2-1}\exp(-\lambda_2 t^{\gamma_2})}{p\exp(-\lambda_1 t^{\gamma_1}) + (1-p)\exp(-\lambda_2 t^{\gamma_2})} \quad \text{(A2)}$$

Experimental treatment was associated with a treatment effect and we assumed no treatment effect heterogeneity. We simulated a baseline binary variable denoting prognosis group through the linear predictor of the survival model, which was incorporated as follows:

$$h_i(t) = h_0(t)\exp[\delta_1\, trt_i + \delta_2\, badprog_i] \quad \text{(A3)}$$

with δ_1 representing the log hazard ratio associated with treatment, and δ_2 representing the impact of being in the bad prognosis group.

Step 2. Disease progression times

Disease progression times were simulated to equal overall survival times multiplied by a value from a beta distribution with shape parameters (5,10). Disease progression was assumed to be observed at the first simulated visit following the progression event, with visits simulated to occur every 21 days from randomisation to death.

Step 3. Switching mechanism

In the simple scenarios switching was a function only of baseline prognosis group, and switching could only occur immediately at the visit at which disease progression was observed – switches could not happen before or after that time-point. The probability of switching was set at 0.8 for patients in the poor prognosis group, and at 0.2 for patients in the good prognosis group in scenarios with a moderate switching proportion, and at 0.9 and 0.6 for poor and good prognosis patients respectively in scenarios with a high switching proportion. In these scenarios all important information on prognostic characteristics was known at the time of disease progression (i.e. in the TSE context, at the 'secondary baseline') and therefore there was no time-dependent confounding. All patients who commenced experimental treatment (as per their randomisation or following a treatment switch) were assumed to remain on the experimental treatment until death/censoring.

Step 4. Effect of switching

In the simple scenarios switchers were simulated to benefit from switching by multiplying the survival period post switch by a factor (ω) using the following approach:

$$T_{Zi} = T_{Ci} + \omega \times T_{Ei} \quad \text{(A4)}$$

where T_{Zi} is the survival time incorporating the impact of switching, T_{Ci} represents the time from randomisation to switch and T_{Ei} represents the survival time after the switch point that was simulated to occur in the absence of switching (i.e. based on the underlying survival time generated in Step 1). This is similar to the accelerated failure time model presented in (1), but here we denote the treatment effect as ω rather than $e^{-\psi}$ and present the model in the context of estimating survival times *with* switching, rather than without switching. ω was set to the average acceleration factor observed in the experimental group, such that the common treatment effect assumption approximately held. To estimate this average acceleration factor scenario-specific survival data were generated for 1 000 000 patients without applying switching and an accelerated failure time model was applied to estimate ψ, with ω then set to equal $e^{-\psi}$. Whether or not the treatment effect in switchers is approximately similar to that in the experimental group is only important for the RPSFTM method. The approach we used resulted in an approximately common treatment effect and it is important to bear this in mind when interpreting the results of the RPSFTM. Previous studies have investigated the sensitivity of the RPSFTM to violations of the common treatment effect assumption.[12,13]

Step 5. Apply censoring

In half of our scenarios censoring was not incorporated – we did not simulate a study end date, such that death occurred for all patients. In the remainder of scenarios censoring was incorporated, by simulating a study end time set at 546 days.

Figure A1 presents an example of the Kaplan-Meier curves and hazard function produced by the simulation model (in the absence of treatment switching) from a single simulated data set in Scenario 1. By using a mixture model, we were able to simulate a hazard function that was initially low, then steadily increased and stabilised before decreasing towards the end of the trial follow-up (note that individual simulations will vary as to how closely they follow this underlying pattern). As has been previously described,[26] we believe this reflects the types of hazards often observed in an oncology RCT setting.

Complex scenarios

Box A2 provides a step-by-step description of the data generation procedure used for the complex scenarios.

Step 1. Underlying overall survival times

Underlying survival times were generated in exactly the same way as for the simple scenarios (see Step 1 for simple scenarios and Box A2).

Step 2. Disease progression times

Disease progression times were generated in exactly the same way as for the simple scenarios (see Step 2 for simple scenarios and Box A2).

Step 3. Time-dependent confounding variable (metastatic event)

In the complex scenarios we simulated a time-dependent confounding variable, which for descriptive purposes we refer to as metastatic disease, M. Hence, we placed our scenarios in the context of adjuvant treatment for cancer. The effect of M and the baseline prognosis group, V, on the switching probability, S, and on survival, Y, is illustrated in a Directed Acyclic Graph in Figure A2. Note that Figure A2 only depicts the post-progression survival period in control group patients, because we only simulated switching in the control group, and

switching could only occur post-progression. Hence, Figure A2 concentrates on illustrating the time-dependent confounding present during this time period in patients at risk of switching.

We simulated data such that *M* could only develop after disease progression – it could not happen at the same time that disease progression was observed, but could be identified at subsequent visits. Hence, the first time *M* could be observed was at the visit after that at which disease progression was observed. To reduce complexity, we restricted the possibility of *M* occurring to the 4 visits immediately following disease progression (i.e. 21, 42, 63 and 84 days after disease progression was observed). Beyond this point the value of *M* could not change. The probability that *M* developed was a function of treatment received (*A*) and prognosis group (*V*) using a two-by-two table. Because *A* and *V* were binary 0/1 variables there were four categories of patient. At each of the 4 visits at which *M* could occur the probability of *M* occurring was 0.5 in patients with bad prognosis (*V*=1*)* receiving control treatment (*A*=0), 0.25 in patients with *V*=1 and *A*=1, 0.2 in patients with *V*=0 and *A*=0, and 0.1 in patients with *V*=0 and *A*=1.

If *M* developed – irrespective of a patient's treatment status – the patient's overall survival time from the time-point of *M* occurring was halved.

Step 4. Switching mechanism

In the complex scenarios switching (*S*) was simulated as a function of baseline prognosis group, *V*, and the value of the metastatic disease variable, *M*, at an individual patient's previous visit, as illustrated in Figure A2. To reduce complexity the time-points at which switching could occur were restricted to the visit at which disease progression was observed and any one of the following 5 visits (i.e. at 0, 21, 42, 63, 84 or 105 days after disease progression). Beyond this point the value of *S* could not change.

The impact of developing metastatic disease on the probability of switching was lagged because we assumed that knowledge of the event happens before the treatment decision is made. Because the earliest observation at which *M* could be observed was the visit after progression was observed, the first time that the probability of switching could be affected by *M* was the third visit after disease progression (see Figure A2).

At the first and second visits after disease progression the probability of switching was based only on baseline prognosis group (*V* in Figure A2), and was 0.1 in poor prognosis patients (*V*=1) and 0.04 in good prognosis patients (*V*=0) in scenarios with a moderate switching proportion. In visits 3-6 after disease progression the

probability of switching was based on V and the value of M at the individual's previous visit (denoted M_L) using a two-by-two table. Since V and M_L were binary 0/1 variables there were four categories of patient. For patients in whom M_L=0 (i.e. M had not occurred at the previous visit) the probabilities of switching remained the same as in visits 1 and 2 after disease progression (0.1 if V=1 and 0.04 if V=0). For patients in whom M_L=1 (i.e. M had occurred at the previous visit) the probability of switching was 0.4 in patients with V=1 and 0.2 in patients with V=0. This implies an interaction, since the impact of M_L differs depending on whether V was 1 or 0.

In scenarios with a high switching proportion the corresponding switching probabilities for each of the four categories of patient were increased – these were 0.3 and 0.1 for poor (V=1) and good (V=0) prognosis patients who had not developed metastatic disease (M_L=0), and 0.72 and 0.40 for those who had (M_L=1).

All patients who commenced experimental treatment (as per their randomisation or following a treatment switch) were assumed to remain on the experimental treatment until death/censoring.

Step 5. Effect of switching

In the complex scenarios, switching affected survival in two ways. First, for any patients who had not developed M before switching, the probability of M occurring at future timepoints was reduced – though, again, to reduce complexity we restricted the possibility of M occurring to the 4 visits immediately following disease progression (i.e. 21, 42, 63 and 84 days after disease progression was observed). The relationship between switching (S) and M is illustrated in Figure A2. If a patient was simulated to switch before they had developed M, their future M status was re-simulated and their overall survival time was adjusted accordingly. For instance, if patient i had been simulated to develop M 84 days after observed disease progression in Step 3, above, but then was simulated to switch treatments at 42 days after observed disease progression in Step 4, their M status was re-drawn for visits at 63 and 84 days after observed disease progression. If the patient was then simulated not to develop metastatic disease their overall survival time was re-set to their originally simulated underlying survival time.

The effect of switching, S, on the probability of M at future visits was the same as the effect of receiving treatment (denoted A) described in Step 3. Hence, the probability of M was dependent on baseline prognosis (V) and whether switching (S) had occurred. V and S were binary 0/1 variables and so there were four categories of

patient: the probability of *M* occurring was 0.5 in patients with *V*=1, *S*=0; 0.25 in patients with *V*=1, *S*=1; 0.2 in patients with *V*=0, *S*=0, and; 0.1 in patients with *V*=0 and *S*=1.

In addition to the impact of switching on the future risk of *M*, the survival period of switchers post switch was multiplied by the factor (ω), as described for the simple scenarios. In this case, T_{B_i} in (7) represents the underlying survival time after the switch point generated in Step 1 if *M* was not simulated to occur, or the survival time after the switch point incorporating the reduction in survival due to *M*, if *M* was simulated to occur. ω was again set such that the common treatment effect assumption approximately held, but this was achieved by taking into account the additional treatment effect that occurred through reducing the probability of *M*. The required value for ω was calculated by first generating scenario-specific survival data for 1 000 000 patients without applying switching and comparing the hazard ratio (that is, $\exp(\delta_1)$) from (A3) to the hazard ratio estimated through an analysis of the 1 000 000 simulated patients (denoted $\exp(\delta_3)$). The contribution to the overall effect through reducing the probability of metastatic disease was then estimated as a proportion (r) of the total effect using $\frac{\exp(\delta_1)-\exp(\delta_3)}{1-\exp(\delta_3)} = r$. An accelerated failure time model was then applied to the 1 000 000 simulated patients to estimate ψ, with ω set to equal $\left(\left(e^{-\psi} - 1\right) \times (1 - r)\right) + 1 = \omega$.

Hence, if *S* occurred before *M*, switching had a dual effect – through reducing the probability of future *M* and thus increasing survival, and through increasing survival independent of *M* according to ω. If *S* occurred after *M*, switching had a singular effect, simply prolonging survival according to ω. Therefore, there was an interaction between the variable *M* and the switching effect, and so treatment effect heterogeneity was present.

Step 6. Apply censoring

As for the simple scenarios, in half of the complex scenarios censoring was not incorporated – death occurred for all patients. In the remainder of scenarios censoring was incorporated, by simulating a study end time set at 546 days.

Figure A3 presents an example of the Kaplan-Meier curves and hazard function produced by the simulation model (in the absence of treatment switching) from a single simulated data set in Scenario 9 (i.e. a complex scenario).

Box A1: Step-by-step data generation procedure for simple scenarios

Step	Description
1	**Underlying overall survival times** (a) Generate underlying survival times using survsim Stata command,[51] using a mixture Weibull survival model that incorporates a log hazard ratio treatment effect and a baseline binary prognostic covariate.
2	**Disease progression times** (a) Generate disease progression times that are a function of overall survival times. Generate observed disease progression times, whereby progression is observed at the "visit" following progression, where visits occur every 21 days.
3	**Switching mechanism** (a) Generate a binary variable for whether or not switching (*S*) occurs for each control group patient (b) *S* can only occur at the time of observed disease progression (c) The probability of *S* is based only on baseline prognosis group (i.e. there are two categories of patient – (i) poor prognosis; (ii) good prognosis). Each patient is assigned a probability of switching depending upon which category they are in.
4	**Effect of switching** (a) In switchers the remaining underlying survival time after *S* occurs is multiplied by a treatment effect time ratio.
5	**Apply censoring** (a) In scenarios in which censoring is incorporated, replace switching, progression and/or death times with administrative censoring time (and associated indicator) if those event times are greater than the administrative censoring time.

Box A2: Step-by-step data generation procedure for complex scenarios

Step	Description
1	**Underlying overall survival times** (a) Generate underlying survival times using survsim Stata command,[51] using a mixture Weibull survival model that incorporates a log hazard ratio treatment effect and a baseline binary prognostic covariate.
2	**Progression-free survival times** (a) Generate disease progression times that are a function of overall survival times. Generate observed disease progression times, whereby progression is observed at the "visit" following progression, where visits occur every 21 days.
3	**Time-dependent confounding variable (metastatic event)** (a) Generate a binary variable for whether or not the metastatic event (M) occurs for each patient (b) M can only occur at 4 observations (visits) after disease progression (i.e. days 21, 42, 63 and 84 after disease progression has been observed). Beyond this point the value of M could not change. (c) Probability of M occurring is based on baseline prognosis group (V, where V=0 represents good prognosis and V=1 represents bad prognosis) and treatment received (A, where A=0 represents control treatment and A=1 represents experimental treatment). Therefore, there are four categories of patient – (i) V=0, A=0; (ii) V=1, A=0; (iii) V=0, A=1; (iv) V=1, A=1. Each patient is assigned a probability of developing M depending upon which of the four categories they are in. (d) At each of the four visits (21, 42, 63 and 84 days after observed disease progression) a binomial random variate draw is used to ascertain whether or not M occurs for each patient, using the probability of M described in (c) if the patient remains alive and if M has not already occurred. (e) If M occurs, the remaining survival time is halved.
4	**Switching mechanism** (a) Generate a binary variable for whether or not switching (S) occurs for each control group patient (b) S can only occur at the time of observed disease progression or at one of the 5 following observations (i.e. days 0, 21, 42, 63, 84 and 105 after disease progression has been observed). Beyond this point the value of S could not change. (c) At 0 and 21 days after observed disease progression the probability of S is based only on baseline prognosis group (i.e. there are two categories of patient – (i) poor prognosis (V=1); (ii) good prognosis (V=0)). Each patient is assigned a probability of switching depending upon which category they are in. (d) At 42, 63, 84 and 105 days after observed progression the probability of S is based on baseline prognosis group (V) and whether or not M=1 at the previous visit (denoted M_L). Therefore, there are four categories of patient – (i) V=0, M_L=0; (ii) V=0, M_L=1; (iii) V=1, M_L=0; (iv) V=1, M_L=1. Each patient is assigned a probability of switching depending upon which of the four categories they are in. (e) At each of the six visits (0, 21, 42, 63, 84 and 105 days after observed progression) a binomial random variate draw is used to ascertain whether or not S occurs for each patient, using the probability of S described in (c) and (d) if the patient remains alive and if S has not already occurred. See Figure 2 for an illustration of the relationship between variables V, M and S.
5	**Effect of switching** (a) If switching for patient i happens before M has occurred, the probability of M occurring at the remaining visits at which M could occur is re-assigned (e.g. if S=1 and M=0 at the consultation 21 days after observed disease progression, the probability of M is amended to reflect the fact that the patient is now "on treatment" at visits 42, 63 and 84 days after observed disease progression – i.e. having S=1 is the same as having A=1 as described in Step 3). (b) For these patients the binomial random variate draw is performed again to ascertain whether or not M occurs at visits after switching. (c) If M occurs, the remaining underlying survival time is halved. If M no longer occurs in patients who were simulated to experience M without switching, overall survival time is re-

	set to the originally simulated underlying survival time for that patient. Hence, a benefit of switching for patients who have not yet developed *M* is the reduced probability of developing *M* at future visits, thereby preventing substantial reductions in survival times. (d) Steps 5(a) to 5(c) mean that some switching patients may live past an additional visit (because they no longer experience *M*, or experience *M* at a later time-point). For these patients the probability of *M* at the additional visit(s) is assigned according to the prognosis (*V*) and switching (*S*) group they are in and a binomial random variate draw is performed for each extra visit to ascertain whether or not *M* occurs. If *M* occurs, the remaining underlying survival time is halved. (e) Following this, in all switchers the remaining survival time after *S* occurs is multiplied by a treatment effect time ratio. Hence, switchers who have not yet developed *M* receive a dual treatment effect – the probability of future *M* is reduced and survival is increased independent of *M* according to the time ratio associated with *S*. For patients who switch after *M* has been developed switching had a singular effect, simply prolonging survival according to the time ratio associated with *S*. See Figure 2 for an illustration of the relationships between *V*, *M*, *S* and survival, *Y*. (f) After extending the survival time in switchers, some patients may live past an additional visit. For these patients the probability of *M* at the additional visit(s) is assigned according to the prognosis (*V*) and switching group (*S*) they are in and a binomial random variate draw is performed for each extra visit to ascertain whether or not *M* occurs. If *M* occurs, the remaining underlying survival time is halved.
6	**Apply censoring** (a) In scenarios in which censoring is incorporated, replace switching, progression, metastatic disease and/or death times with administrative censoring time (and associated indicator) if those event times are greater than the administrative censoring time.

Figure A1. One simulated dataset from Scenario 1 with no switching: (a) Overall survival Kaplan–Meier; (b) Smoothed hazard rate

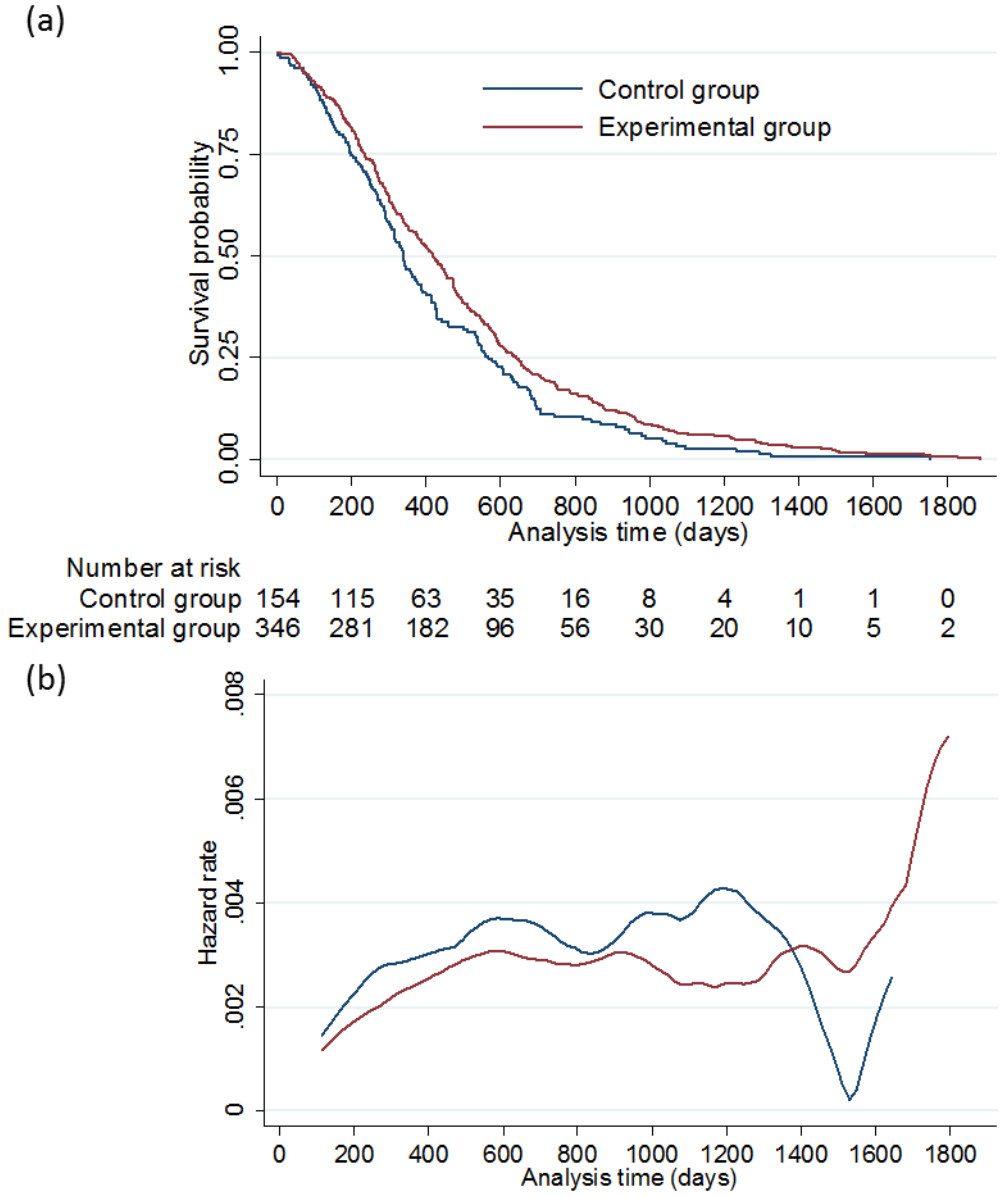
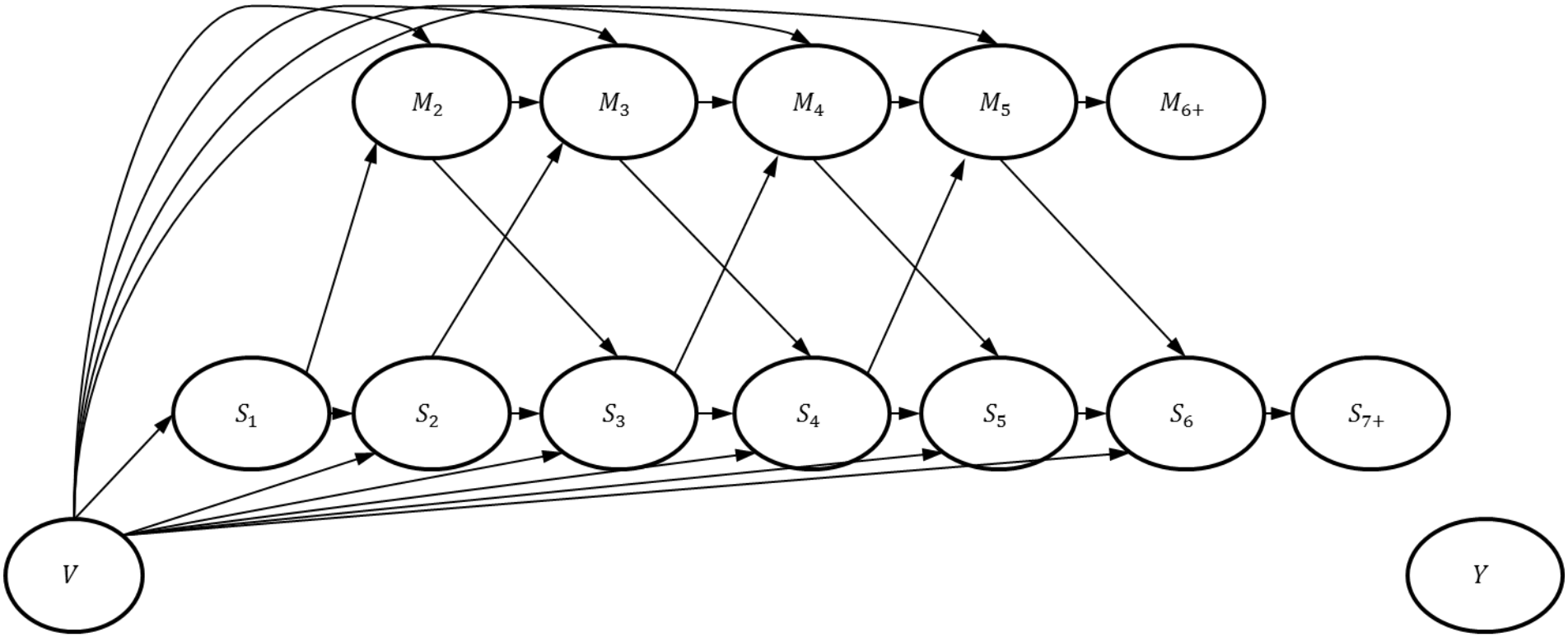
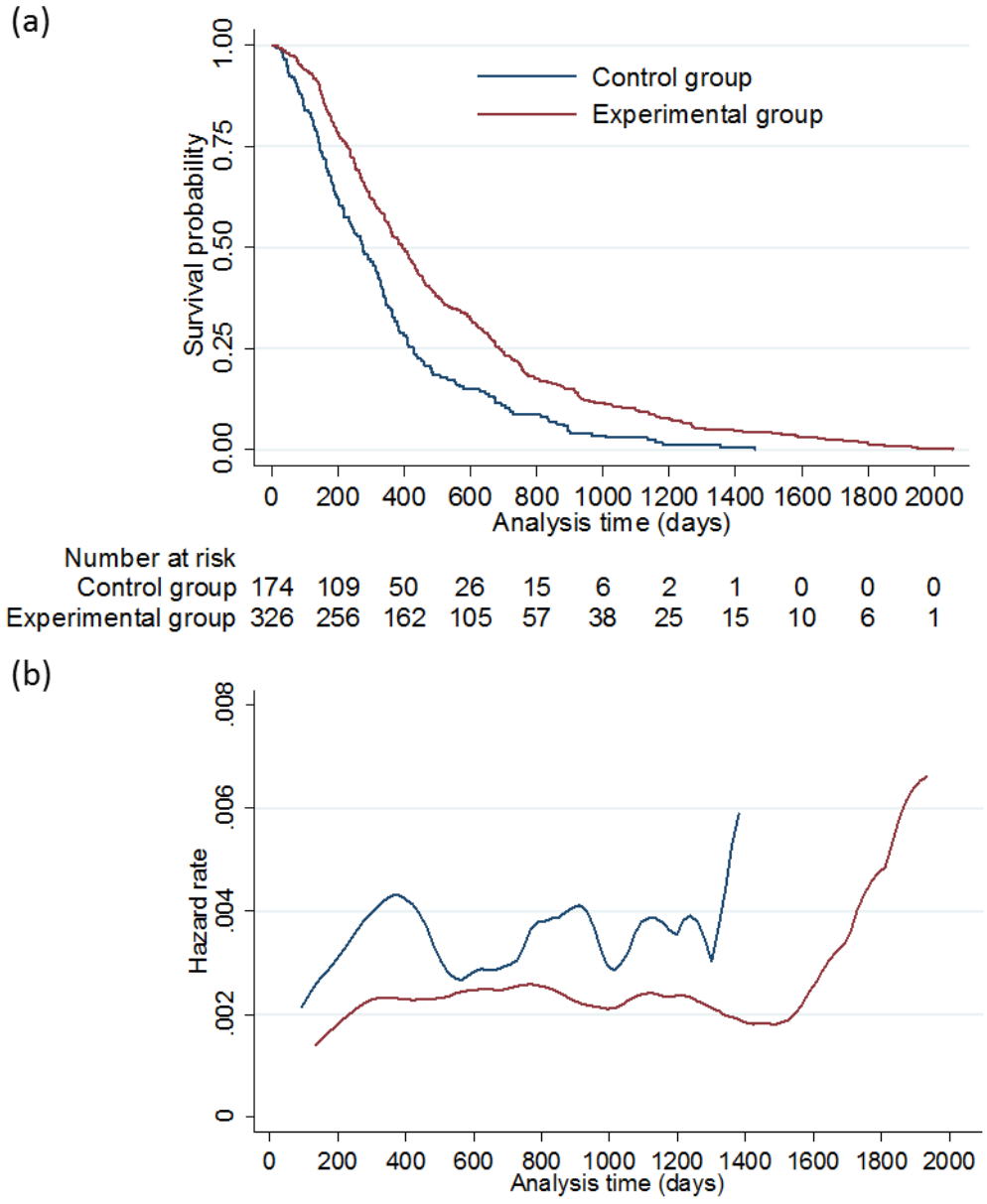


Figure A2. Simulated post-progression disease and switching pathway over time, control group patients



V= baseline prognosis group
M = metastatic disease indicator
S = switch treatment indicator
Y = death indicator
Note: all variables have an arrow to Y

Figure A3. One simulated dataset from Scenario 9 with no switching: (a) Overall survival Kaplan–Meier; (b) Smoothed hazard rate



Appendix B: Simulation study scenario parameter values

In Table B1, values for each variable in Scenario 1 are specified, as are alternative values for the 17 other scenarios.

Table B1: Simulated scenarios – Parameter values and alternatives tested

Variable	Value (Scenario 1)	Alternative Values
Sample size	500 (2:1 randomisation)	10,000 (2:1 randomisation) in Scenarios 17 and 18.
Number of prognosis groups (prog)	2	-
Probability of good prognosis	0.5	-
Probability of poor prognosis	0.5	-
Maximum follow-up time	5000 days (no censoring)	546 days
Impact of bad prognosis on survival	Log hazard ratio = 0.3	-
Survival time distribution	Weibull parameters: Mix 1: Scale parameter 0.000025 Shape parameter 1.8 Mix 2: Scale parameter 0.000015 Shape parameter 1.7 p = 0.5 (mix parameter)	-
Progression free survival	Overall survival (OS) time multiplied by a value from a beta distribution with shape parameters (5,10) – this implies the assumption that time to progression is 33% of OS. This is not an important assumption – time to progression is only included because we model a situation where switching cannot occur before disease progression	-
Baseline treatment effect (note: this the true treatment effect in Scenarios 1-8, but not in Scenarios 9-18 as this does not take into account the effect of the treatment that occurs through the time-dependent confounder, metastatic disease)	Baseline log hazard ratio -0.20 for a small treatment effect	Alter log hazard ratio to -0.50 for a large treatment effect <u>Scenarios with time-dependent confounding (including effect through metastatic disease variable)</u> Baseline log hazard ratio -0.08 for a small treatment effect Alter log hazard ratio to -0.35 for a large treatment effect
Probability of developing metastatic disease (note, the first time metastatic disease could be developed is one visit after disease progression has been observed)	0 (zero probability in Scenarios 1-8)	Bad prognosis and no treatment: 0.5 Good prognosis and no treatment: 0.2 Bad prognosis and treated: 0.25 Good prognosis and treated: 0.1 (metastatic disease could only occur at the visit after disease progression was observed, or at the three subsequent visits)
Probability of switching (note: switching only possible in control group)	Moderate switching proportion Bad prognosis: 0.8 Good prognosis: 0.2 (switching only possible at the visit at which disease progression is observed)	Scenarios with high switching proportion, no time-dependent confounding: Bad prognosis: 0.9 Good prognosis: 0.6 (switching only possible at the visit at which disease progression is observed) Scenarios with moderate switching

		proportion and time-dependent confounding: Bad prognosis, no metastatic disease: 0.1 Good prognosis, no metastatic disease: 0.04 Bad prognosis, metastatic disease: 0.4 Good prognosis, metastatic disease: 0.2 (switch could occur at the visit at which disease progression was observed or at any of the following 5 visits) Scenarios with high switching proportion and time-dependent confounding: Bad prognosis, no metastatic disease: 0.3 Good prognosis, no metastatic disease: 0.1 Bad prognosis, metastatic disease: 0.72 Good prognosis, metastatic disease: 0.4 (switch could occur at the visit at which disease progression was observed or at any of the following 5 visits)
Survival impact of metastatic disease	Survival time from the time-point of metastases developing to individual's originally generated survival time is shrunk by a factor of 2	-
Assumed frequency of visits	One every 3 weeks (21 days)	-
Treatment effect in switching patients	Equal to baseline treatment effect multiplied by ω. Set ω such that treatment effect received by switching patients is 100% of the average effect received by experimental group patients.	-

Appendix C: Simulation study scenario settings

Table C1: Simulation study scenario settings

Scenario	Treatment effect	Switch proportion	Censoring	Time-dependent confounding	Sample size
1	Low	Moderate	None	None	500
2	High	Moderate	None	None	500
3	Low	High	None	None	500
4	High	High	None	None	500
5	Low	Moderate	Moderate	None	500
6	High	Moderate	Moderate	None	500
7	Low	High	Moderate	None	500
8	High	High	Moderate	None	500
9	Low	Moderate	None	Present	500
10	High	Moderate	None	Present	500
11	Low	High	None	Present	500
12	High	High	None	Present	500
13	Low	Moderate	Moderate	Present	500
14	High	Moderate	Moderate	Present	500
15	Low	High	Moderate	Present	500
16	High	High	Moderate	Present	500
17	Low	High	Moderate	Present	10,000
18	High	High	Moderate	Present	10,000

Appendix D: Adjustment methods applied in the simulation study

TSEsimp, TSEsimpTDC, TSEgest, RPSFTM and IPCW were included.

TSEsimp was applied using disease progression as the secondary baseline time-point. A Weibull model was fitted to post-progression control group survival times to estimate the switching effect. The Weibull model included a time-dependent switch indicator variable, and a variable for baseline prognosis group. The RPSFTM was applied using a log-rank test within the g-estimation procedure using the Stata command strbee.[52] For the RPSFTM and other methods that involve a counterfactual survival model (TSEsimp, TSEsimpTDC and TSEgest) re-censoring was applied in scenarios where censoring was simulated.

In the simple scenarios, switching models for TSEgest and IPCW only included the baseline prognosis variable. In the complex scenarios we matched the specification of the switching models to the switching mechanism described previously. The probability of switching depended on the value of M at the previous visit and the baseline prognosis group (V), with the four categories created by these interacting binary variables each assigned a switching probability which remained constant over 6 post-progression visits for each patient. Switching could not occur beyond visit 6 after disease progression (as illustrated in Figure A2). Hence, we created four dummy variables to indicate which of the four categories each patient was in at visits 1-6 after disease progression (where visit 1 was the observation at which progression was first observed), i.e. Dummy 1: V=0, M_L=0; Dummy 2: V=0, M_L=1; Dummy 3: V=1, M_L=0; Dummy 4: V=1, M_L=1. As illustrated by Figure A2, all patients will have M_L=0 at visits 1 and 2. Another dummy variable was created to indicate visit 7 onwards, to account for the fact that switching could not occur beyond visit 6, irrespective of a patient's prognosis group and metastatic disease status. In addition, a lagged variable was created for the switching indicator, because, in our simulations, prior switching is a perfect predictor of the treatment variable equalling '1' in subsequent observations (because we did not simulate subsequent treatment discontinuation).

We updated the Stata program stgest to apply TSEgest,[20] with disease progression used as the secondary baseline time-point. The switching model included the four patient group dummy variables and the visit 7 onwards dummy variable, as well as the lagged switching variable. This meant that, for each patient, only observations up to (and including) the point of switch were included in the switching model. Subsequent observations would be excluded due to the inclusion of the perfectly predicting lagged switching variable. Also, for non-switchers, observations beyond visit 6 would be excluded (because the visit 7 onwards dummy perfectly

predicts no switching). The potential outcome (i.e. counterfactual survival) was entered into the switching model using the martingale residual from the null model (that is, the event indicator minus the cumulative hazard), thus combining the time to event and censoring indicators. The g-test used was a Wald test with sandwich variance to allow for dependence of observations for the same individual.

For IPCW we used stabilised weights using a similar approach to that described by Fewell *et al*.[53] Baseline prognosis group was the only covariate included in V and the model for the numerator of the stabilised weight was fitted to all time periods. In the model used for the denominator of the weight the four patient group dummy variables were included in $\bar{L}(k)$ and the model was only fitted to visits 1-6 after progression – thus, the dataset used to model the denominator of the weight exactly matched that used for the TSEgest switching model. The denominator of the weight was set to 1 in periods before disease progression and after visit 6 after progression, reflecting the impossibility of switching occurring in these observations. 21-day intervals were used and time was incorporated using restricted cubic splines using the Stata command rcsgen.[54] Interior knots were placed at the 33rd and 67th centiles of the distribution of censoring (switching) times and 2 boundary knots were placed at the minimum and maximum values of the censoring (switching) times.

TSEsimpTDC was included in Scenarios 9-18. In this analysis the four time-dependent dummy variables for baseline prognosis/metastatic disease categories at visits 1-6 after disease progression were included in the Weibull model used to estimate the effect of switching on survival.

We used the Stata command stpm2 [40] to fit flexible parametric models to the counterfactual datasets provided by TSEsimp, TSEsimpTDC, TSEgest and RPSFTM, and to the weighted survival times provided by IPCW, to obtain the survivor function extrapolated (if necessary) to 546 days, ensuring that our RMST comparisons were comparing "like with like". Flexible parametric models were fit on the log cumulative hazard scale, with 3 interior knots placed at the 25th, 50th and 75th centiles of the distribution of log survival times and boundary knots placed at the minimum and maximum of the distribution of uncensored survival times. If the final observed counterfactual survival time was less than 546 days, RMST at 546 days was estimated through extrapolation using stpm2, which involves a linear extrapolation of log time on the log cumulative hazard scale, based on the fitted function and extrapolating from the last knot.

Appendix E: Overview of simulation scenarios

Table E1 presents key details associated with each of the scenarios simulated. The true area under the curve (restricted mean survival time (RMST)) unconfounded by treatment switching is presented, along with the average treatment effect in terms of a hazard ratio (HR) and an acceleration factor (AF). In Scenarios 1-8 the true RMST was calculated by integrating the survivor function using equations (A1), (A2) and (A3) given in Appendix A. The true HR was known, as $\exp(\delta_1)$ from (A3). The average AF was estimated by generating scenario data for 1,000,000 patients without simulating switching and applying an RPSFTM under no switching. For scenarios 9-18 the survivor function was not analytically tractable and so this simulation technique was used to estimate true RMST, HR and AF. RMST was estimated directly from the data generated for 1,000,000 patients without simulating switching, Cox models were used to estimate the HR, and an RPSFTM was again used to estimate the AF. The HR and AF represent only an approximation of the true treatment effect as the proportional hazards and constant acceleration factor assumptions did not hold in Scenarios 9-18. In terms of a hazard ratio, the average treatment effect varied between 0.61 and 0.82.

The proportion of control group patients that switched, averaged across the 1000 simulations that made up each scenario, is also presented. The switching proportion varied between 49% and 75% of all control group patients. Table E1 also presents the switching proportion as a percentage of the control group patients that became 'at-risk' of switching. In our simulations control group patients could only switch treatments if they were alive at their first 'visit' at 21 days and if their disease progressed before the end of the simulated follow-up. The switching proportion as a percentage of patients that became at-risk of switching is higher than when it is measured as a percentage of all control group patients – it ranged from 50% to 75%. We estimated the proportion of patients who became at risk of switching in each scenario by collecting data on the number of patients for whom disease progression was observed in each simulation and taking the mean. This is approximate, but appropriately indicative for our purposes. We took a similar approach to estimate the proportion of control group patients who developed metastatic disease for the scenarios that included time-dependent confounding.

Table E1 also presents details on the mean proportion of control group patients that were censored in each scenario – that is, the proportion for whom death was not observed. This varied between 0% and 35%.

Table E1: Overview of simulated scenarios

Scenario	Treatment effect	Switch proportion	Censoring	Time-dependent confounding	Sample size	Truth (years)		Average treatment effects		Treatment effect in switchers (AF)	Mean switcher % of total	Mean switcher % of at risk	Mean censoring proportion, control group	Mean metastatic disease % at risk, control group
						RMST (Control group)	RMST (Exp group)	HR	AF					
1	Low	Moderate	None	None	500	423.62	486.25	0.82	1.15	1.15	50%	50%	0%	0%
2	High	Moderate	None	None	500	423.62	600.32	0.61	1.42	1.42	50%	50%	0%	0%
3	Low	High	None	None	500	423.62	486.25	0.82	1.15	1.15	74%	75%	0%	0%
4	High	High	None	None	500	423.62	600.32	0.61	1.42	1.42	74%	75%	0%	0%
5	Low	Moderate	Moderate	None	500	350.15	375.19	0.82	1.14	1.14	49%	50%	28%	0%
6	High	Moderate	Moderate	None	500	350.15	409.24	0.61	1.38	1.38	49%	50%	32%	0%
7	Low	High	Moderate	None	500	350.15	375.19	0.82	1.14	1.14	74%	75%	29%	0%
8	High	High	Moderate	None	500	350.15	409.24	0.61	1.38	1.38	74%	75%	35%	0%
9	Low	Moderate	None	Present	500	338.11	387.73	0.83	1.15	1.06	51%	51%	0%	69%
10	High	Moderate	None	Present	500	338.00	465.44	0.65	1.38	1.30	51%	51%	0%	69%
11	Low	High	None	Present	500	338.11	387.73	0.83	1.15	1.06	75%	75%	0%	64%
12	High	High	None	Present	500	338.00	465.44	0.65	1.38	1.30	74%	75%	0%	64%
13	Low	Moderate	Moderate	Present	500	301.36	328.71	0.82	1.15	1.06	49%	50%	17%	67%
14	High	Moderate	Moderate	Present	500	301.39	362.44	0.64	1.36	1.28	50%	51%	18%	67%
15	Low	High	Moderate	Present	500	301.36	328.71	0.82	1.15	1.06	73%	74%	18%	62%
16	High	High	Moderate	Present	500	301.39	362.44	0.64	1.36	1.28	73%	75%	21%	63%
17	Low	High	Moderate	Present	10,000	301.36	328.71	0.82	1.15	1.06	73%	75%	18%	62%
18	High	High	Moderate	Present	10,000	301.39	362.44	0.64	1.36	1.28	73%	75%	21%	63%

Appendix F: Simulation study code

What follows is the code used to run the simulation study in Stata (version 14.2).[39] Note that this is for scenario 2, which had no censoring. In scenarios that included censoring, the value for 'admin' was 546, and analysis programs had to be amended to account for censoring.

First the simulation program is presented, followed by the analysis programs called by the simulation program.

Simulation program

```
set seed 36504189
***base***
cd "C:\crabstera"
capture program drop crabster1
program define crabster1, rclass
version 14.2
syntax [, obs(int 500) bprog(real 0.5) trtlghr(real -0.50) bprogsl(real 0.3) ///
admin(real 5000) ppoor(real 0.80) pgood(real 0.20) xomult(real 1.4188308) ///
pcatnotrtbprog(real 0.0) pcattrtbprog(real 0.0) catmult(real 0.5) tdxo(real 0)  ///
pcatnotrt(real 0.0) pcattrt(real 0.0) pcatxo(real 10.0) adminpps(real 4979) ///
                                                LAMBDA1(real 0.000025)          ///
                                                LAMBDA2(real 0.000015)          ///
                                                GAMMA1(real 1.8)        ///
                                                GAMMA2(real 1.7)        ///
                                                Pmix(real 0.5)                  ///
]

clear

adopath ++ "C:\ado\plus\s\survsim"

*** 1. Mixture model ***
  clear
  pr drop _all
  scalar state1 = substr(c(rngstate), 1,    2000)
  scalar state2 = substr(c(rngstate), 2001, 4000)
  scalar state3 = substr(c(rngstate), 4001, .)
                set obs `obs'
                gen trtrand = rbinomial(1,0.66)
                gen bprog = rbinomial(1,`bprog')
                gen admin = `admin'

                survsim timeOS event, hazard(                                                                                   ///
                                        (`lambda1':*`gamma1':*`pmix':*#t:^(`gamma1':-1):*exp(-`lambda1':*#t:^`gamma1') :+ ///
                                        `lambda2':*`gamma2':*(1:-`pmix'):*#t:^(`gamma2':-1):* ///
                                        exp(-`lambda2':*#t:^`gamma2') ):/(`pmix':*exp(-`lambda1':*#t:^`gamma1') ///
                                        :+ (1:-`pmix'):*exp(-`lambda2':*#t:^`gamma2'))      )                                    ///
                cov(trtrand `trtlghr' bprog `bprogsl') maxtime(5000) nodes(100)

*** 2. GENERATE ROUNDED SURVIVAL TIMES AND RANDOM ENTRY ***
                gen id = _n
                rename event dead
                replace timeOS = round(timeOS)
                replace timeOS=1 if timeOS==0

*** 3. GENERATE TIME TO DISEASE PROGRESSION ***
                stset timeOS, failure(dead) id(id)
                gen timePFS = round(timeOS*rbeta(5,10))
                stsplit timeOS2, every(21)
                sort id
                by id: gen PFSobsind=1 if timePFS<timeOS2
                by id: replace PFSobsind=. if timePFS>admin
                by id: gen timePFSobst=timeOS2 if PFSobsind==1
                by id: egen timePFSobs=min(timePFSobst)
                by id: egen timeOSreal=max(timeOS)
                by id: gen progressed = 1 if timePFSobs!=.
                by id: replace progressed = 0 if timePFSobs==.
                by id: replace timePFSobs = timeOSreal if progressed==0
                by id: replace timePFSobs = admin if (timeOSreal > admin & progressed==0)
                by id: replace timePFSobs = admin if (timePFSobs > admin & progressed==1)
                by id: replace progressed = 0 if timePFSobs==admin
                collapse (max) trtrand bprog timeOS timePFS timePFSobs admin progressed dead, by(id)

*** 4. GEN CATASTROPHIC EVENT THAT CAN HAPPEN AFTER PROGRESSION RELATED TO TRT AND BPROG AND AMEND SURVIVAL TIMES ***
                by id: gen probcat = `pcatnotrtbprog' if trtrand==0 & bprog==1
                by id: replace probcat = `pcattrtbprog' if trtrand==1 & bprog==1
                by id: replace probcat = `pcatnotrt' if trtrand==0 & bprog==0
                by id: replace probcat = `pcattrt' if trtrand==1 & bprog==0
```

```
		gen cat2 = rbinomial(1,probcat) if progressed==1 & timeOS > timePFSobs+(21)
		gen cat3 = rbinomial(1,probcat) if cat2==0 & progressed==1 & timeOS > timePFSobs+(42)
		gen cat4 = rbinomial(1,probcat) if cat2==0 & cat3==0 & progressed==1 & timeOS > timePFSobs+(63)
		gen cat5 = rbinomial(1,probcat) if cat2==0 & cat3==0 & cat4==0 & progressed==1 & timeOS > timePFSobs+(84)
		gen cattime = timePFSobs +(21) if cat2==1
		replace cattime = timePFSobs +(42) if cat3==1
		replace cattime = timePFSobs +(63) if cat4==1
		replace cattime = timePFSobs +(84) if cat5==1
		gen catevent= 1 if (cat2==1 | cat3==1 | cat4==1 | cat5==1)
		gen catOSloss = timeOS-cattime if catevent==1
		replace catOSloss = round(catOSloss*`catmult')
		gen timeOS2 = cond(catevent==1, catOSloss + cattime,timeOS)
		gen timeOS5 = timeOS
		replace timeOS = timeOS2
		drop timeOS2

*** 5. CALCULATE HR AND RMST WITH NO SWITCHING ***
		preserve
		replace dead = 0 if timeOS>admin
		replace timeOS = admin if timeOS>admin
		stset timeOS, failure(dead) id(id)

		***RMST stci***
		stci if trtrand==0, rmean
		return scalar simtrueconstcimean = r(rmean)
		return scalar simtrueconstcilb = r(lb)
		return scalar simtrueconstciub = r(ub)
		return scalar simtrueconstcise = r(se)
		stci if trtrand==1, rmean
		return scalar simtrueexpstcimean = r(rmean)
		return scalar simtrueexpstcilb = r(lb)
		return scalar simtrueexpstciub = r(ub)
		return scalar simtrueexpstcise = r(se)

		***HR FPM with bprog***
		capture stpm2 trtrand bprog, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand bprog, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand bprog, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand bprog, scale(h) df(1) lininit iterate(200)
}
}
}
return scalar simtrue_fpmwbprog_hr = exp(_b[trtrand])
		gen conv1 = e(converged)
		summ conv1
		return scalar simtrue_fpmwbrog_conv = r(mean)

		***HR FPM without bprog***
		capture stpm2 trtrand, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand, scale(h) df(1) lininit iterate(200)
}
}
}
		return scalar simtrue_fpm_hr = exp(_b[trtrand])
		return scalar simtrue_fpm_hr_SE = exp(_b[trtrand])*_se[trtrand]
  return scalar simtrue_fpm_hr_LB = exp((_b[trtrand])-(1.96*_se[trtrand]))
  return scalar simtrue_fpm_hr_UB = exp((_b[trtrand])+(1.96*_se[trtrand]))
		gen conv2 = e(converged)
		summ conv2
		return scalar simtrue_fpm_conv = r(mean)

  ***HR Weibull with bprog***
		streg trtrand bprog, dist(weibull) iterate(200)
		return scalar simtrue_weibwbprog_hr = exp(_b[trtrand])
		gen conv3 = e(converged)
		summ conv3
		return scalar simtrue_weibwbprog_conv = r(mean)

		***HR Weibull without bprog***
		streg trtrand, dist(weibull) iterate(200)
		return scalar simtrue_weib_hr = exp(_b[trtrand])
		gen conv4 = e(converged)
		summ conv4
		return scalar simtrue_weib_conv = r(mean)

		***RMST FPM experimental group***
capture stpm2 if trtrand==1, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==1, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
```

```
		capture stpm2 if trtrand==1, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==1, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv5 = e(converged)
capture predict meansurv, rmst tmax(`admin') ci
if _rc==498 {
capture predict meansurv, rmst tmax(`admin')
summ meansurv
return scalar simtruefpmexpauc=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc!=498 {
summ meansurv2
return scalar simtruefpmexpauc=r(mean)
summ meansurv2_lci
return scalar simtruefpmexpauc_LB=r(mean)
summ meansurv2_uci
return scalar simtruefpmexpauc_UB=r(mean)
}
summ conv5
return scalar simtrue_fpmexp_conv = r(mean)
		restore

		***RMST FPM control group***
		preserve
		replace dead = 0 if timeOS>admin
		replace timeOS = admin if timeOS>admin
		stset timeOS, failure(dead) id(id)

capture stpm2 if trtrand==0, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==0, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==0, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==0, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv6 = e(converged)
capture predict meansurv, rmst tmax(`admin') ci
if _rc==498 {
capture predict meansurv, rmst tmax(`admin')
summ meansurv
return scalar simtruefpmconauc=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc!=498 {
summ meansurv2
return scalar simtruefpmconauc=r(mean)
summ meansurv2_lci
return scalar simtruefpmconauc_LB=r(mean)
summ meansurv2_uci
return scalar simtruefpmconauc_UB=r(mean)
}
summ conv6
return scalar simtrue_fpmcon_conv = r(mean)
restore

*** 6. APPLY SWITCH AND EFFECT ***
		gen tdxo = `tdxo'
		gen p1 = (invlogit(logit(`ppoor'))) if bprog==1
		replace p1 = (invlogit(logit(`pgood'))) if bprog==0
		gen p2 = (invlogit(logit(`ppoor'))) if bprog==1
		replace p2 = (invlogit(logit(`pgood'))) if bprog==0
		gen p3 = (invlogit(logit(`ppoor') + log(`pcatxo'))) if bprog==1 & (cat2==1)
		replace p3 = (invlogit(logit(`pgood') + log(`pcatxo'))) if bprog==0 & (cat2==1)
  replace p3 = (invlogit(logit(`ppoor'))) if bprog==1 & cat2!=1
  replace p3 = (invlogit(logit(`pgood'))) if bprog==0 & cat2!=1
		gen p4 = (invlogit(logit(`ppoor') + log(`pcatxo'))) if bprog==1 & (cat2==1 | cat3==1)
		replace p4 = (invlogit(logit(`pgood') + log(`pcatxo'))) if bprog==0 & (cat2==1 | cat3==1)
  replace p4 = (invlogit(logit(`ppoor'))) if bprog==1 & cat2!=1 & cat3!=1
  replace p4 = (invlogit(logit(`pgood'))) if bprog==0 & cat2!=1 & cat3!=1
		gen p5 = (invlogit(logit(`ppoor') + log(`pcatxo'))) if bprog==1 & (cat2==1 | cat3==1 | cat4==1)
		replace p5 = (invlogit(logit(`pgood') + log(`pcatxo'))) if bprog==0 & (cat2==1 | cat3==1 | cat4==1)
  replace p5 = (invlogit(logit(`ppoor'))) if bprog==1 & cat2!=1 & cat3!=1 & cat4!=1
  replace p5 = (invlogit(logit(`pgood'))) if bprog==0 & cat2!=1 & cat3!=1 & cat4!=1
		gen p6 = (invlogit(logit(`ppoor') + log(`pcatxo'))) if bprog==1 & (cat2==1 | cat3==1 | cat4==1 | cat5==1)
		replace p6 = (invlogit(logit(`pgood') + log(`pcatxo'))) if bprog==0 & (cat2==1 | cat3==1 | cat4==1 | cat5==1)
  replace p6 = (invlogit(logit(`ppoor'))) if bprog==1 & cat2!=1 & cat3!=1 & cat4!=1 & cat5!=1
  replace p6 = (invlogit(logit(`pgood'))) if bprog==0 & cat2!=1 & cat3!=1 & cat4!=1 & cat5!=1

		gen xo1 = rbinomial(1,p1) if trtrand==0 & timeOS>timePFSobs & progressed==1
		gen xo2 = rbinomial(1,p2) if trtrand==0 & xo1==0 & timeOS>timePFSobs+(21) & progressed==1 & tdxo==1
		gen xo3 = rbinomial(1,p3) if trtrand==0 & xo1==0 & xo2==0 & timeOS>timePFSobs+(42) & progressed==1 & tdxo==1
```

```
gen xo4 = rbinomial(1,p4) if trtrand==0 & xo1==0 & xo2==0 & xo3==0 & timeOS>timePFSobs+(63) & progressed==1 & tdxo==1
gen xo5 = rbinomial(1,p5) if trtrand==0 & xo1==0 & xo2==0 & xo3==0 & xo4==0 & timeOS>timePFSobs+(84) & progressed==1 & tdxo==1
gen xo6 = rbinomial(1,p6) if trtrand==0 & xo1==0 & xo2==0 & xo3==0 & xo4==0 & xo5==0 & timeOS>timePFSobs+(105) & progressed==1 & tdxo==1

gen xotime = timePFSobs if xo1==1
replace xotime = timePFSobs +(21) if xo2==1
replace xotime = timePFSobs +(42) if xo3==1
replace xotime = timePFSobs +(63) if xo4==1
replace xotime = timePFSobs +(84) if xo5==1
replace xotime = timePFSobs +(105) if xo6==1

gen xo= 1 if (xo1==1 | xo2==1 | xo3==1 | xo4==1 | xo5==1 | xo6==1)

***Apply switch effect first through reducing probability of catastrophic event***
  gen xoprecat = 1 if (xotime<cattime | catevent==.) & xo==1
replace probcat = `pcattrtbprog' if trtrand==0 & xo==1 & bprog==1 & xoprecat==1
replace probcat = `pcattrt' if trtrand==0 & xo==1 & bprog==0 & xoprecat==1
replace cat2=. if xo==1 & xoprecat==1 & xotime==timePFSobs
replace cat3=. if xo==1 & xoprecat==1 & (xotime==timePFSobs | xotime==timePFSobs+(21))
replace cat4=. if xo==1 & xoprecat==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42))
replace cat5=. if xo==1 & xoprecat==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42) | xotime==timePFSobs+(63))
replace cat2 = rbinomial(1,probcat) if progressed==1 & timeOS > timePFSobs+(21) & xo==1 & xotime==timePFSobs
replace cat3 = rbinomial(1,probcat) if cat2==0 & progressed==1 & timeOS > timePFSobs+(42) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21))
replace cat4 = rbinomial(1,probcat) if cat2==0 & cat3==0 & progressed==1 & timeOS > timePFSobs+(63) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) |
xotime==timePFSobs+(42))
replace cat5 = rbinomial(1,probcat) if cat2==0 & cat3==0 & cat4==0 & progressed==1 & timeOS > timePFSobs+(84) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21)
| xotime==timePFSobs+(42) | xotime==timePFSobs+(63))

replace cattime = . if xo==1 & xoprecat==1
replace cattime = timePFSobs +(21) if cat2==1 & xo==1 & xotime==timePFSobs
replace cattime = timePFSobs +(42) if cat3==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21))
replace cattime = timePFSobs +(63) if cat4==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42))
replace cattime = timePFSobs +(84) if cat5==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42) | xotime==timePFSobs+(63))

replace catevent = . if xo==1 & xoprecat==1
replace catevent= 1 if (cat2==1 | cat3==1 | cat4==1 | cat5==1)

replace catOSloss = . if xo==1 & xoprecat==1
replace catOSloss = timeOS5-cattime if catevent==1 & xo==1 & xoprecat==1
replace catOSloss = round(catOSloss*`catmult') if catevent==1 & xo==1 & xoprecat==1
replace timeOS = catOSloss + cattime if catevent==1 & xoprecat==1
replace timeOS = timeOS5 if catevent!=1 & xoprecat==1

***Apply chance of catastrophic event for people who don't have cat event or have later cat event and now live past an additional observation. Only replacing for EXTRA
observations - others don't change***
gen extra2v1 = 1 if progressed==1 & timeOS > timePFSobs+(21) & xo==1 & xotime==timePFSobs & cat2==.
gen extra3v1 = 1 if cat2==0 & progressed==1 & timeOS > timePFSobs+(42) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21)) & cat3==.
gen extra4v1 = 1 if cat2==0 & cat3==0 & progressed==1 & timeOS > timePFSobs+(63) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42))
& cat4==.
gen extra5v1 = 1 if cat2==0 & cat3==0 & cat4==0 & progressed==1 & timeOS > timePFSobs+(84) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) |
xotime==timePFSobs+(42) | xotime==timePFSobs+(63)) & cat5==.
gen extraobsv1 = 1 if (extra2v1==1 | extra3v1==1 | extra4v1==1 | extra5v1==1)

replace cat2 = rbinomial(1,probcat) if progressed==1 & timeOS > timePFSobs+(21) & xo==1 & xotime==timePFSobs & cat2==.
replace cat3 = rbinomial(1,probcat) if cat2==0 & progressed==1 & timeOS > timePFSobs+(42) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21)) & cat3==.
replace cat4 = rbinomial(1,probcat) if cat2==0 & cat3==0 & progressed==1 & timeOS > timePFSobs+(63) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) |
xotime==timePFSobs+(42)) & cat4==.
replace cat5 = rbinomial(1,probcat) if cat2==0 & cat3==0 & cat4==0 & progressed==1 & timeOS > timePFSobs+(84) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21)
| xotime==timePFSobs+(42) | xotime==timePFSobs+(63)) & cat5==.

replace cattime = . if xo==1 & xoprecat==1 & extraobsv1==1
replace cattime = timePFSobs +(21) if cat2==1 & xo==1 & xotime==timePFSobs  & extraobsv1==1
replace cattime = timePFSobs +(42) if cat3==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21)) & extraobsv1==1
replace cattime = timePFSobs +(63) if cat4==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42)) & extraobsv1==1
replace cattime = timePFSobs +(84) if cat5==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42) | xotime==timePFSobs+(63)) &
extraobsv1==1

replace catevent = . if xo==1 & xoprecat==1 & extraobsv1==1
replace catevent= 1 if (cat2==1 | cat3==1 | cat4==1 | cat5==1) & extraobsv1==1
***remember, these are only patients who hadn't had a cat event before, and now may have done in their additional observations. So survival times cannot go up, this only
allows for possible additional events***
replace catOSloss = . if xo==1 & xoprecat==1 & extraobsv1==1
replace catOSloss = timeOS5-cattime if catevent==1 & xo==1 & xoprecat==1 & extraobsv1==1
replace catOSloss = round(catOSloss*`catmult') if catevent==1 & xo==1 & xoprecat==1    & extraobsv1==1
replace timeOS = catOSloss + cattime if catevent==1 & xoprecat==1 & extraobsv1==1
replace timeOS = timeOS5 if catevent!=1 & xoprecat==1 & extraobsv1==1

***Apply switch effect second through xomult***
gen xoOSgainobs = timeOS-xotime if xo==1
replace xoOSgainobs = round(xoOSgainobs*`xomult')
gen timeOS2 = cond(xo==1, xoOSgainobs + xotime,timeOS)

***Apply chance of catastrophic event for xo patients who now live past an additional observation. Only replacing for EXTRA observations in switchers, so no additional chance
to switch and OS can't increase as currently in these observations there will be zero chance of cat event***
gen extra2v2 = 1 if progressed==1 & timeOS2 > timePFSobs+(21) & xo==1 & xotime==timePFSobs & cat2==.
gen extra3v2 = 1 if cat2==0 & progressed==1 & timeOS2 > timePFSobs+(42) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21)) & cat3==.
gen extra4v2 = 1 if cat2==0 & cat3==0 & progressed==1 & timeOS2 > timePFSobs+(63) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) |
xotime==timePFSobs+(42)) & cat4==.
```

```
        gen extra5v2 = 1 if cat2==0 & cat3==0 & cat4==0 & progressed==1 & timeOS2 > timePFSobs+(84) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) |
xotime==timePFSobs+(42) | xotime==timePFSobs+(63)) & cat5==.
        gen extraobsv2 = 1 if (extra2v2==1 | extra3v2==1 | extra4v2==1 | extra5v2==1)

        replace cat2 = rbinomial(1,probcat) if progressed==1 & timeOS2 > timePFSobs+(21) & xo==1 & xotime==timePFSobs & cat2==.
        replace cat3 = rbinomial(1,probcat) if cat2==0 & progressed==1 & timeOS2 > timePFSobs+(42) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21)) & cat3==.
        replace cat4 = rbinomial(1,probcat) if cat2==0 & cat3==0 & progressed==1 & timeOS2 > timePFSobs+(63) & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) |
xotime==timePFSobs+(42)) & cat4==.
        replace cat5 = rbinomial(1,probcat) if cat2==0 & cat3==0 & cat4==0 & progressed==1 & timeOS2 > timePFSobs+(84) & xo==1 & (xotime==timePFSobs |
xotime==timePFSobs+(21) | xotime==timePFSobs+(42) | xotime==timePFSobs+(63)) & cat5==.

        replace cattime = . if xo==1 & xoprecat==1 & extraobsv2==1
        replace cattime = timePFSobs +(21) if cat2==1 & xo==1 & xotime==timePFSobs  & extraobsv2==1
        replace cattime = timePFSobs +(42) if cat3==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21)) & extraobsv2==1
        replace cattime = timePFSobs +(63) if cat4==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42)) & extraobsv2==1
        replace cattime = timePFSobs +(84) if cat5==1 & xo==1 & (xotime==timePFSobs | xotime==timePFSobs+(21) | xotime==timePFSobs+(42) | xotime==timePFSobs+(63)) &
extraobsv2==1

        replace catevent = . if xo==1 & xoprecat==1 & extraobsv2==1
        replace catevent= 1 if (cat2==1 | cat3==1 | cat4==1 | cat5==1) & extraobsv2==1

        replace catOSloss = . if xo==1 & xoprecat==1 & extraobsv2==1
        ***remember, these are only patients who hadn't had a cat event before, and now may have done in their additional observations, which previously they did not live past. So if
they have an event, cattime will be >OS5. Hence only makes sense to apply cat event survival reduction to extended survival beyond this point, i.e. OS2.***
        replace catOSloss = timeOS2-cattime if catevent==1 & xo==1 & xoprecat==1 & extraobsv2==1
        replace catOSloss = round(catOSloss*`catmult') if catevent==1 & xo==1 & xoprecat==1    & extraobsv2==1
        replace timeOS2 = catOSloss + cattime if catevent==1 & xoprecat==1 & extraobsv2==1
        ***don't replace OS2 for those with an extraobsv2 who did not have a cat event because OS has not changed for them***

*** 7. APPLY CENSORING ***
        ***we censor assuming a max follow-up time of admin days***
        gen died=0
        replace died = 1 if timeOS2<=admin & dead==1
        drop dead
        replace timeOS2=admin if timeOS2>admin
        replace xoOSgainobs=. if xotime>admin
        replace xo=. if xotime>=admin
        replace xotime=. if xotime>=admin
        replace catevent=. if cattime>=admin
        replace cattime=. if cattime>=admin

*** 8. CREATE PANEL ***
        stset timeOS2, failure(died) id(id)
        ***split the data every 3 weeks to create a panel***
        stsplit timeOS3, every(21)
        sort id
        by id: gen obsno=_n
        tsset id obsno
        replace died=0 if died==.
        sort id
  by id: gen finalobs = 0
        by id: replace finalobs = 1 if _n==_N
        gen cens=0
        replace cens=1 if finalobs==1 & _t==admin & died==0
        replace died=. if cens==1
        ***make time-dependent switch, progression and catastrophic event variables***
        by id: gen xotdc = 0
        by id: replace xotdc = 1 if xo==1 & (timeOS3)>=(xotime)
        by id: gen cattdc = 0
        by id: replace cattdc = 1 if catevent==1 & (timeOS3)>=cattime
        by id: gen progtdc = 0
        by id: replace progtdc = 1 if progressed==1 & (timeOS3)>=timePFSobs
        by id: replace catevent = 0 if catevent==.
        ***gen xo proportion***
        by id: egen xoind=max(xo)
        replace xoind=0 if xoind==.
        by id: egen xoprop=max(timeOS3)
        replace xoprop=xoprop/(21)
        replace xoprop=xoprop+1
        replace xoprop=xoind/xoprop
        summ xoprop if trtrand==0
        return scalar xo_number=r(mean)*r(N)
        summ cens
        return scalar cens_number=r(mean)*r(N)
        summ cens if trtrand==0
        return scalar cens_number_con=r(mean)*r(N)

***ANALYSIS***

itt_crabster  , admin(`admin') adminpps(`adminpps')
return scalar prog_number=r(prog_number)
return scalar prog_number_con=r(prog_number_con)
return scalar con_number=r(con_number)
return scalar cens_number_xo=r(cens_number_xo)
return scalar cat_number = r(cat_number)
return scalar cat_number_xo = r(cat_number_xo)
return scalar cat_number_noxo = r(cat_number_noxo)
return scalar itt_stci_con_mean = r(itt_stci_con_mean)
```

```
return scalar itt_stci_con_se = r(itt_stci_con_se)
return scalar itt_stci_con_lb = r(itt_stci_con_lb)
return scalar itt_stci_con_ub = r(itt_stci_con_ub)
return scalar itt_auc_con = r(itt_auc_con)
return scalar itt_auc_con_LB = r(itt_auc_con_LB)
return scalar itt_auc_con_UB = r(itt_auc_con_UB)
return scalar itt_auc_con_conv = r(itt_auc_con_conv)
return scalar itt_cox_hr = r(itt_cox_hr)
return scalar itt_cox_hr_SE = r(itt_cox_hr_SE)
return scalar itt_cox_hr_LB = r(itt_cox_hr_LB)
return scalar itt_cox_hr_UB = r(itt_cox_hr_UB)
return scalar itt_fpm_hr = r(itt_fpm_hr)
return scalar itt_fpm_hr_SE = r(itt_fpm_hr_SE)
return scalar itt_fpm_hr_LB = r(itt_fpm_hr_LB)
return scalar itt_fpm_hr_UB = r(itt_fpm_hr_UB)
return scalar itt_fpm_hr_conv = r(itt_fpm_hr_conv)
return scalar exp_fpm_conv = r(exp_fpm_conv)

tse_simple_crabster  , admin(`admin') adminpps(`adminpps')
        return scalar tses_weib_af = r(tses_weib_af)
        return scalar tses_weib_af_SE = r(tses_weib_af_SE)
        return scalar tses_weib_af_LB = r(tses_weib_af_LB)
        return scalar tses_weib_af_UB = r(tses_weib_af_UB)
        return scalar tses_weib_af_conv = r(tses_weib_af_conv)
        return scalar tses_stci_con_mean = r(tses_stci_con_mean)
        return scalar tses_stci_con_lb = r(tses_stci_con_lb)
        return scalar tses_stci_con_ub = r(tses_stci_con_ub)
        return scalar tses_stci_con_se = r(tses_stci_con_se)
        return scalar tses_adj_auc_con = r(tses_adj_auc_con)
        return scalar tses_adj_auc_con_LB = r(tses_adj_auc_con_LB)
        return scalar tses_adj_auc_con_UB = r(tses_adj_auc_con_UB)
        return scalar tses_adj_fpm_con_conv = r(tses_adj_fpm_con_conv)
        return scalar tses_adj_fpm_hr = r(tses_adj_fpm_hr)
        return scalar tses_adj_fpm_hr_SE = r(tses_adj_fpm_hr_SE)
        return scalar tses_adj_fpm_hr_LB = r(tses_adj_fpm_hr_LB)
        return scalar tses_adj_fpm_hr_UB = r(tses_adj_fpm_hr_UB)
        return scalar tses_adj_fpm_conv = r(tses_adj_fpm_conv)

tse_complex_crabster  , admin(`admin') adminpps(`adminpps')
        return scalar tsec_psi = r(tsec_psi)
        return scalar tsec_af = r(tsec_af)
        return scalar tsec_af_LB = r(tsec_af_LB)
        return scalar tsec_af_UB = r(tsec_af_UB)
        return scalar tsec_conv = r(tsec_conv)
        return scalar tsec_stci_con_mean = r(tsec_stci_con_mean)
        return scalar tsec_stci_con_lb = r(tsec_stci_con_lb)
        return scalar tsec_stci_con_ub = r(tsec_stci_con_ub)
        return scalar tsec_stci_con_se = r(tsec_stci_con_se)
        return scalar tsec_adj_auc_con = r(tsec_adj_auc_con)
        return scalar tsec_adj_auc_con_LB = r(tsec_adj_auc_con_LB)
        return scalar tsec_adj_auc_con_UB = r(tsec_adj_auc_con_UB)
        return scalar tsec_adj_fpm_con_conv = r(tsec_adj_fpm_con_conv)
        return scalar tsec_adj_fpm_hr = r(tsec_adj_fpm_hr)
        return scalar tsec_adj_fpm_hr_SE = r(tsec_adj_fpm_hr_SE)
        return scalar tsec_adj_fpm_hr_LB = r(tsec_adj_fpm_hr_LB)
        return scalar tsec_adj_fpm_hr_UB = r(tsec_adj_fpm_hr_UB)
        return scalar tsec_adj_fpm_conv = r(tsec_adj_fpm_conv)

rpsftm_crabster  , admin(`admin') adminpps(`adminpps')
        return scalar RPSFTM = r(RPSFTM)
        return scalar RPSFTM_LB = r(RPSFTM_LB)
        return scalar RPSFTM_UB = r(RPSFTM_UB)
        return scalar RPSFTM_fpm_hr = r(RPSFTM_fpm_hr)
        return scalar RPSFTM_fpm_hr_SE = r(RPSFTM_fpm_hr_SE)
        return scalar RPSFTM_fpm_hr_LB = r(RPSFTM_fpm_hr_LB)
        return scalar RPSFTM_fpm_hr_UB = r(RPSFTM_fpm_hr_UB)
        return scalar RPSFTM_fpm_hr_conv = r(RPSFTM_fpm_hr_conv)
        return scalar RPSFTM_stci_con_mean = r(RPSFTM_stci_con_mean)
        return scalar RPSFTM_stci_con_lb = r(RPSFTM_stci_con_lb)
        return scalar RPSFTM_stci_con_ub = r(RPSFTM_stci_con_ub)
        return scalar RPSFTM_stci_con_se = r(RPSFTM_stci_con_se)
        return scalar RPSFTM_sfunc_auc_con=r(RPSFTM_sfunc_auc_con)
        return scalar RPSFTM_sfunc_auc_con_LB=r(RPSFTM_sfunc_auc_con_LB)
        return scalar RPSFTM_sfunc_auc_con_UB=r(RPSFTM_sfunc_auc_con_UB)
        return scalar RPSFTM_adj_fpm_con_conv = r(RPSFTM_adj_fpm_con_conv)

ipcw_crabster  , admin(`admin') adminpps(`adminpps')
return scalar ipcwn1_conv = r(ipcwn1_conv)
return scalar ipcwn1_omit = r(ipcwn1_omit)
return scalar ipcwn1_cds = r(ipcwn1_cds)
return scalar ipcwn1_cdf = r(ipcwn1_cdf)
return scalar ipcw_cox_hr = r(ipcw_cox_hr)
return scalar ipcw_cox_hr_SE = r(ipcw_cox_hr_SE)
return scalar ipcw_cox_hr_LB =  r(ipcw_cox_hr_LB)
return scalar ipcw_cox_hr_UB = r(ipcw_cox_hr_UB)
return scalar ipcw3_conv = r(ipcw3_conv)
return scalar ipcw3_omit = r(ipcw3_omit)
return scalar ipcw3_cds = r(ipcw3_cds)
```

```
return scalar ipcw3_cdf = r(ipcw3_cdf)
return scalar ipcw_weight_min = r(ipcw_weight_min)
return scalar ipcw_weight_max = r(ipcw_weight_max)
return scalar ipcw_weight_mean = r(ipcw_weight_mean)
return scalar ipcw_weight_sd = r(ipcw_weight_sd)
return scalar ipcw_weight_cv = r(ipcw_weight_cv)
return scalar ipcw_adj_auc_con=r(ipcw_adj_auc_con)
return scalar ipcw_adj_auc_con_LB=r(ipcw_adj_auc_con_LB)
return scalar ipcw_adj_auc_con_UB=r(ipcw_adj_auc_con_UB)
return scalar ipcw_adj_fpm_con_conv = r(ipcw_adj_fpm_con_conv)

post buffer (return(xo_number)) ///
                (return(cens_number)) ///
                (return(cens_number_con)) ///
                (return(prog_number)) ///
                (return(prog_number_con)) ///
                (return(con_number)) ///
                (return(cens_number_xo)) ///
                (return(cat_number)) ///
                (return(cat_number_xo)) ///
    (return(cat_number_noxo)) ///
                (return(simtrueconstcimean)) ///
                (return(simtrueconstcilb)) ///
                (return(simtrueconstciub)) ///
                (return(simtrueconstcise)) ///
                (return(simtrueexpstcimean)) ///
                (return(simtrueexpstcilb)) ///
                (return(simtrueexpstciub)) ///
                (return(simtrueexpstcise)) ///
                (return(simtrue_fpmwbprog_hr)) ///
                (return(simtrue_fpmwbrog_conv)) ///
                (return(simtrue_fpm_hr)) ///
                (return(simtrue_fpm_hr_SE)) ///
                (return(simtrue_fpm_hr_LB)) ///
                (return(simtrue_fpm_hr_UB)) ///
                (return(simtrue_fpm_conv)) ///
                (return(simtrue_weibwbprog_hr)) ///
                (return(simtrue_weibwbprog_conv)) ///
                (return(simtrue_weib_hr)) ///
                (return(simtrue_weib_conv)) ///
                (return(simtruefpmexpauc)) ///
                (return(simtruefpmexpauc_LB)) ///
                (return(simtruefpmexpauc_UB)) ///
                (return(simtrue_fpmexp_conv)) ///
                (return(simtruefpmconauc)) ///
                (return(simtruefpmconauc_LB)) ///
                (return(simtruefpmconauc_UB)) ///
                (return(simtrue_fpmcon_conv)) ///
                (return(itt_stci_con_mean)) ///
                (return(itt_stci_con_se)) ///
                (return(itt_stci_con_lb)) ///
                (return(itt_stci_con_ub)) ///
                (return(itt_auc_con)) ///
                (return(itt_auc_con_LB)) ///
                (return(itt_auc_con_UB)) ///
                (return(itt_auc_con_conv)) ///
                (return(itt_cox_hr)) ///
                (return(itt_cox_hr_SE)) ///
                (return(itt_cox_hr_LB)) ///
                (return(itt_cox_hr_UB)) ///
                (return(itt_fpm_hr)) ///
                (return(itt_fpm_hr_SE)) ///
                (return(itt_fpm_hr_LB)) ///
                (return(itt_fpm_hr_UB)) ///
                (return(itt_fpm_hr_conv)) ///
                (return(exp_fpm_conv)) ///
                (return(tses_weib_af)) ///
                (return(tses_weib_af_SE)) ///
                (return(tses_weib_af_LB)) ///
                (return(tses_weib_af_UB)) ///
                (return(tses_weib_af_conv)) ///
                (return(tses_stci_con_mean)) ///
                (return(tses_stci_con_lb)) ///
                (return(tses_stci_con_ub)) ///
                (return(tses_stci_con_se)) ///
                (return(tses_adj_auc_con)) ///
                (return(tses_adj_auc_con_LB)) ///
                (return(tses_adj_auc_con_UB)) ///
                (return(tses_adj_fpm_con_conv)) ///
                (return(tses_adj_fpm_hr)) ///
                (return(tses_adj_fpm_hr_SE)) ///
                (return(tses_adj_fpm_hr_LB)) ///
                (return(tses_adj_fpm_hr_UB)) ///
                (return(tses_adj_fpm_conv)) ///
                (return(tsec_psi)) ///
                (return(tsec_af)) ///
                (return(tsec_af_LB)) ///
                (return(tsec_af_UB)) ///
```

```
		(return(tsec_conv)) ///
		(return(tsec_stci_con_mean)) ///
		(return(tsec_stci_con_lb)) ///
		(return(tsec_stci_con_ub)) ///
		(return(tsec_stci_con_se)) ///
		(return(tsec_adj_auc_con)) ///
		(return(tsec_adj_auc_con_LB)) ///
		(return(tsec_adj_auc_con_UB)) ///
		(return(tsec_adj_fpm_con_conv)) ///
		(return(tsec_adj_fpm_hr)) ///
		(return(tsec_adj_fpm_hr_SE)) ///
		(return(tsec_adj_fpm_hr_LB)) ///
		(return(tsec_adj_fpm_hr_UB)) ///
		(return(tsec_adj_fpm_conv)) ///
		(return(RPSFTM)) ///
		(return(RPSFTM_LB)) ///
		(return(RPSFTM_UB)) ///
		(return(RPSFTM_fpm_hr)) ///
		(return(RPSFTM_fpm_hr_SE)) ///
		(return(RPSFTM_fpm_hr_LB)) ///
		(return(RPSFTM_fpm_hr_UB)) ///
		(return(RPSFTM_fpm_hr_conv)) ///
		(return(RPSFTM_stci_con_mean)) ///
		(return(RPSFTM_stci_con_lb)) ///
		(return(RPSFTM_stci_con_ub)) ///
		(return(RPSFTM_stci_con_se)) ///
		(return(RPSFTM_sfunc_auc_con)) ///
		(return(RPSFTM_sfunc_auc_con_LB)) ///
		(return(RPSFTM_sfunc_auc_con_UB)) ///
		(return(RPSFTM_adj_fpm_con_conv)) ///
		(return(ipcwn1_conv)) ///
		(return(ipcwn1_omit)) ///
		(return(ipcwn1_cds)) ///
		(return(ipcwn1_cdf)) ///
		(return(ipcw_cox_hr)) ///
		(return(ipcw_cox_hr_SE)) ///
		(return(ipcw_cox_hr_LB)) ///
		(return(ipcw_cox_hr_UB)) ///
		(return(ipcw3_conv)) ///
		(return(ipcw3_omit)) ///
		(return(ipcw3_cds)) ///
		(return(ipcw3_cdf)) ///
		(return(ipcw_weight_min)) ///
		(return(ipcw_weight_max)) ///
		(return(ipcw_weight_mean)) ///
		(return(ipcw_weight_sd)) ///
		(return(ipcw_weight_cv)) ///
		(return(ipcw_adj_auc_con)) ///
		(return(ipcw_adj_auc_con_LB)) ///
		(return(ipcw_adj_auc_con_UB)) ///
		(return(ipcw_adj_fpm_con_conv)) ///
		(state1) (state2) (state3)
end

postfile buffer double xo_number ///
		cens_number ///
		cens_number_con ///
		prog_number ///
		prog_number_con ///
		con_number ///
		cens_number_xo ///
		cat_number ///
		cat_number_xo ///
		cat_number_noxo ///
		simtrueconstcimean ///
		simtrueconstcilb ///
		simtrueconstciub ///
		simtrueconstcise ///
		simtrueexpstcimean ///
		simtrueexpstcilb ///
		simtrueexpstciub ///
		simtrueexpstcise ///
		simtrue_fpmwbprog_hr ///
		simtrue_fpmwbrog_conv ///
		simtrue_fpm_hr ///
		simtrue_fpm_hr_SE ///
		simtrue_fpm_hr_LB ///
		simtrue_fpm_hr_UB ///
		simtrue_fpm_conv ///
		simtrue_weibwbprog_hr ///
		simtrue_weibwbprog_conv ///
		simtrue_weib_hr ///
		simtrue_weib_conv ///
		simtruefpmexpauc ///
		simtruefpmexpauc_LB ///
		simtruefpmexpauc_UB ///
		simtrue_fpmexp_conv ///
		simtruefpmconauc ///
```

simtruefpmconauc_LB ///
simtruefpmconauc_UB ///
simtrue_fpmcon_conv ///
simtruefpmconppsauc ///
simtruefpmconppsauc_LB ///
simtruefpmconppsauc_UB ///
simtrue_fpmconpps_conv ///
itt_stci_con_mean ///
itt_stci_con_se ///
itt_stci_con_lb ///
itt_stci_con_ub ///
itt_auc_con ///
itt_auc_con_LB ///
itt_auc_con_UB ///
itt_auc_con_conv ///
itt_cox_hr ///
itt_cox_hr_SE ///
itt_cox_hr_LB ///
itt_cox_hr_UB ///
itt_fpm_hr ///
itt_fpm_hr_SE ///
itt_fpm_hr_LB ///
itt_fpm_hr_UB ///
itt_fpm_hr_conv ///
exp_fpm_conv ///
tses_weib_af ///
tses_weib_af_SE ///
tses_weib_af_LB ///
tses_weib_af_UB ///
tses_weib_af_conv ///
tses_stci_con_mean ///
tses_stci_con_lb ///
tses_stci_con_ub ///
tses_stci_con_se ///
tses_adj_auc_con ///
tses_adj_auc_con_LB ///
tses_adj_auc_con_UB ///
tses_adj_fpm_con_conv ///
tses_adj_fpm_hr ///
tses_adj_fpm_hr_SE ///
tses_adj_fpm_hr_LB ///
tses_adj_fpm_hr_UB ///
tses_adj_fpm_conv ///
tsec_psi ///
tsec_af ///
tsec_af_LB ///
tsec_af_UB ///
tsec_conv ///
tsec_stci_con_mean ///
tsec_stci_con_lb ///
tsec_stci_con_ub ///
tsec_stci_con_se ///
tsec_adj_auc_con ///
tsec_adj_auc_con_LB ///
tsec_adj_auc_con_UB ///
tsec_adj_fpm_con_conv ///
tsec_adj_fpm_hr ///
tsec_adj_fpm_hr_SE ///
tsec_adj_fpm_hr_LB ///
tsec_adj_fpm_hr_UB ///
tsec_adj_fpm_conv ///
RPSFTM ///
RPSFTM_LB ///
RPSFTM_UB ///
RPSFTM_fpm_hr ///
RPSFTM_fpm_hr_SE ///
RPSFTM_fpm_hr_LB ///
RPSFTM_fpm_hr_UB ///
RPSFTM_fpm_hr_conv ///
RPSFTM_stci_con_mean ///
RPSFTM_stci_con_lb ///
RPSFTM_stci_con_ub ///
RPSFTM_stci_con_se ///
RPSFTM_sfunc_auc_con ///
RPSFTM_sfunc_auc_con_LB ///
RPSFTM_sfunc_auc_con_UB ///
RPSFTM_adj_fpm_con_conv ///
ipcwn1_conv ///
ipcwn1_omit ///
ipcwn1_cds ///
ipcwn1_cdf ///
ipcw_cox_hr ///
ipcw_cox_hr_SE ///
ipcw_cox_hr_LB ///
ipcw_cox_hr_UB ///
ipcw3_conv ///
ipcw3_omit ///
ipcw3_cds ///

```
		ipcw3_cdf ///
		ipcw_weight_min ///
		ipcw_weight_max ///
		ipcw_weight_mean ///
		ipcw_weight_sd ///
		ipcw_weight_cv ///
		ipcw_adj_auc_con ///
		ipcw_adj_auc_con_LB ///
		ipcw_adj_auc_con_UB ///
		ipcw_adj_fpm_con_conv ///
		str2000(state1 state2 state3) using crabstersim1, replace

simulate, reps(1000): crabster1, obs(500) bprog(0.5) trtlghr(-0.50) bprogsl(0.3) admin(5000) ppoor(0.80) pgood(0.20) xomult(1.4188308) pcatnotrtbprog(0.0) pcattrtbprog(0.0) catmult(0.5)
tdxo(0) pcatnotrt(0.0) pcattrt(0.0) pcatxo(10.0) adminpps(4979)
postclose buffer

use crabstersim1, clear
gen strL state = state1+state2+state3
drop state1 state2 state3
save crabstersim1, replace
```

ITT program

```
program define itt_crabster, rclass
version 14.2

syntax [varlist] [if] [in] [, admin(real 5000) adminpps(real 4979)]

***Naive - ITT***
***Cox Proportional Hazards Analysis***
stcox trtrand
return scalar itt_cox_hr = exp(_b[trtrand])
return scalar itt_cox_hr_SE = exp(_b[trtrand])*_se[trtrand]
return scalar itt_cox_hr_LB = exp((_b[trtrand])-(1.96*_se[trtrand]))
return scalar itt_cox_hr_UB = exp((_b[trtrand])+(1.96*_se[trtrand]))

***some information***
preserve
collapse (max) trtrand bprog timeOS2 xotime xo cattime catevent died cens timePFSobs admin progressed, by(id)
summ progressed
return scalar prog_number=r(mean)*r(N)
summ progressed if trtrand==0
return scalar prog_number_con=r(mean)*r(N)
return scalar con_number=r(N)
summ cens if trtrand==0 & xo==1
return scalar cens_number_xo=r(mean)*r(N)
summ catevent
return scalar cat_number = r(mean)*r(N)
summ catevent if trtrand==0 & xo==1
return scalar cat_number_xo = r(mean)*r(N)
summ catevent if trtrand==0 & xo==.
return scalar cat_number_noxo = r(mean)*r(N)

***survival analysis***
stset timeOS2, failure(died) id(id)

***RMST stci***
		stci if trtrand==0, rmean
		return scalar itt_stci_con_mean = r(rmean)
		return scalar itt_stci_con_lb = r(lb)
		return scalar itt_stci_con_ub = r(ub)
		return scalar itt_stci_con_se = r(se)
***FPM HR***
capture stpm2 trtrand, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 trtrand, scale(h) df(1) lininit iterate(200)
}
}
}
return scalar itt_fpm_hr = exp(_b[trtrand])
return scalar itt_fpm_hr_SE = exp(_b[trtrand])*_se[trtrand]
return scalar itt_fpm_hr_LB = exp((_b[trtrand])-(1.96*_se[trtrand]))
return scalar itt_fpm_hr_UB = exp((_b[trtrand])+(1.96*_se[trtrand]))
return scalar zittnl = _b[trtrand] / _se[trtrand]
gen conv1 = e(converged)
summ conv1
return scalar itt_fpm_hr_conv = r(mean)

***FPM RMST***
capture stpm2 if trtrand==1, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==1, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
```

```
		capture stpm2 if trtrand==1, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==1, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv2 = e(converged)
summ conv2
return scalar exp_fpm_conv = r(mean)

capture stpm2 if trtrand==0, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==0, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==0, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 {
		capture stpm2 if trtrand==0, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv3 = e(converged)
capture predict meansurv, rmst tmax(`admin') ci
if _rc!=0 {
capture predict meansurv, rmst tmax(`admin')
summ meansurv
return scalar itt_auc_con=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc==0 {
summ meansurv2
return scalar itt_auc_con=r(mean)
summ meansurv2_lci
return scalar itt_auc_con_LB=r(mean)
summ meansurv2_uci
return scalar itt_auc_con_UB=r(mean)
}
summ conv3
return scalar itt_auc_con_conv = r(mean)

restore

end
```

TSEsimp program

```
program define tse_simple_crabster, rclass
version 14.2

syntax [varlist] [if] [in] [, admin(real 5000) adminpps(real 4979)]

***Two-stage Weibull with recensoring***
preserve
***check that any switchers died***
collapse (max) trtrand bprog timeOS2 xotime xo catevent cattime died cens timePFSobs admin progressed, by(id)
gen check = 1 if died==1 & xo==1
replace check = 0 if check==.
sum check
restore
if r(mean)!=0 {

gen trtnew=0
replace trtnew=1 if trtrand==0 & xotdc==1
replace trtnew=1 if trtrand==1
preserve

drop if trtrand==1
drop if progtdc==0
by id: replace obsno = _n
by id: egen minrisk=min(timeOS3)
by id: replace timeOS3=timeOS3-minrisk
by id: replace xotime=xotime-minrisk
by id: replace timeOS2=timeOS2-minrisk
by id: replace admin=admin-minrisk

stset timeOS2, failure(died) id(id)
capture streg trtnew bprog, dist(weibull) time iterate(200)
return scalar tses_weib_af = exp(_b[trtnew])
return scalar tses_weib_af_SE = exp(_b[trtnew])*_se[trtnew]
return scalar tses_weib_af_LB = exp((_b[trtnew])-(1.96*_se[trtnew]))
return scalar tses_weib_af_UB = exp((_b[trtnew])+(1.96*_se[trtnew]))
gen conv1 = e(converged)
summ conv1
return scalar tses_weib_af_conv = r(mean)

restore
sort id
preserve
```

```
***Analysis on overall survival***
collapse (max) trtrand bprog timeOS2 xotime died cens timePFSobs admin, by(id)
by id: replace timePFSobs = 0 if timePFSobs==.
by id: replace xotime=0 if xotime==.
***below allows for recensoring***
gen cfact = timeOS2 if trtrand==1
gen dcfact = died if trtrand==1

replace cfact = (xotime + ((timeOS2-xotime)/return(tses_weib_af))) if (trtrand==0 & xotime>0)
replace cfact = timeOS2 if (trtrand==0 & xotime==0)
gen OSadminc = admin/return(tses_weib_af) if (trtrand==0 & return(tses_weib_af)>1.00)
replace dcfact = died if trtrand==0
replace dcfact=0 if (OSadminc<=cfact & trtrand==0)
replace cfact = OSadminc if (OSadminc<=cfact & trtrand==0)

***do survival analysis on re-estimated survival times***
stset cfact, failure(dcfact) id(id)

***FPM HR***
capture stpm2 trtrand, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 trtrand, scale(h) df(3) lininit iterate(200)
                if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 trtrand, scale(h) df(2) lininit iterate(200)
                if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 trtrand, scale(h) df(1) lininit iterate(200)
}
}
}
if _rc==0 {
return scalar tses_adj_fpm_hr = exp(_b[trtrand])
return scalar tses_adj_fpm_hr_SE = exp(_b[trtrand])*_se[trtrand]
return scalar tses_adj_fpm_hr_LB = exp((_b[trtrand])-(1.96*_se[trtrand]))
return scalar tses_adj_fpm_hr_UB = exp((_b[trtrand])+(1.96*_se[trtrand]))
gen conv1 = e(converged)
summ conv1
return scalar tses_adj_fpm_conv = r(mean)
}
***RMST stci***
                stci if trtrand==0, rmean
                return scalar tses_stci_con_mean = r(rmean)
                return scalar tses_stci_con_lb = r(lb)
                return scalar tses_stci_con_ub = r(ub)
                return scalar tses_stci_con_se = r(se)

***FPM RMST***
capture stpm2 if trtrand==0, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 if trtrand==0, scale(h) df(3) lininit iterate(200)
                if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 if trtrand==0, scale(h) df(2) lininit iterate(200)
                if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 if trtrand==0, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv2 = e(converged)
capture predict meansurv, rmst tmax(`admin') ci
if _rc!=0 {
capture predict meansurv, rmst tmax(`admin')
summ meansurv
return scalar tses_adj_auc_con=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc==0 {
summ meansurv2
return scalar tses_adj_auc_con=r(mean)
summ meansurv2_lci
return scalar tses_adj_auc_con_LB=r(mean)
summ meansurv2_uci
return scalar tses_adj_auc_con_UB=r(mean)
}
summ conv2
return scalar tses_adj_fpm_con_conv = r(mean)

restore

}
End
```

TSEgest program

```
program define tse_complex_crabster, rclass
version 14.2

syntax [varlist] [if] [in] [, admin(real 5000) adminpps(real 4979)]

***Two-stage SNM with recensoring***
```

```
preserve
***check that any switchers died***
collapse (max) trtrand bprog timeOS2 xotime xo catevent cattime died cens timePFSobs admin progressed, by(id)
gen check = 1 if died==1 & xo==1
replace check = 0 if check==.
sum check
restore
if r(mean)!=0 & tdxo==0 {

preserve

drop if trtrand==1
drop if progtdc==0
by id: replace obsno = _n
by id: egen minrisk=min(timeOS3)
by id: replace timeOS3=timeOS3-minrisk
by id: replace xotime=xotime-minrisk
by id: replace timeOS2=timeOS2-minrisk
by id: replace admin=admin-minrisk

stset timeOS2, failure(died) id(id)
capture stgest3 trtnew bprog, visit(obsno) nograph nolist saveres(gest1) range (-3 3) model(all) outcome(mgale) test(cluster) replace
return scalar tsec_psi = r(psi)
return scalar tsec_af = exp(-r(psi))
return scalar tsec_af_LB = exp(-r(ucipsi))
return scalar tsec_af_UB = exp(-r(lcipsi))
gen conv1 = e(converged)
summ conv1
return scalar tsec_conv = r(mean)

restore
sort id
preserve

***Analysis on overall survival***
collapse (max) trtrand bprog timeOS2 xotime died cens timePFSobs admin, by(id)
by id: replace timePFSobs = 0 if timePFSobs==.
by id: replace xotime=0 if xotime==.
***below allows for recensoring***
gen cfact = timeOS2 if trtrand==1
gen dcfact = died if trtrand==1

replace cfact = (xotime + ((timeOS2-xotime)/return(tsec_af))) if (trtrand==0 & xotime>0)
replace cfact = timeOS2 if (trtrand==0 & xotime==0)
gen OSadminc = admin/return(tsec_af) if (trtrand==0 & return(tsec_af)>1.00)
replace dcfact = died if trtrand==0
replace dcfact=0 if (OSadminc<=cfact & trtrand==0)
replace cfact = OSadminc if (OSadminc<=cfact & trtrand==0)

***do survival analysis on re-estimated survival times***
stset cfact, failure(dcfact) id(id)

***FPM HR***
capture stpm2 trtrand, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 trtrand, scale(h) df(3) lininit iterate(200)
                if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 trtrand, scale(h) df(2) lininit iterate(200)
                if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 trtrand, scale(h) df(1) lininit iterate(200)
}
}
}
if _rc==0 {
return scalar tsec_adj_fpm_hr = exp(_b[trtrand])
return scalar tsec_adj_fpm_hr_SE = exp(_b[trtrand])*_se[trtrand]
return scalar tsec_adj_fpm_hr_LB = exp((_b[trtrand])-(1.96*_se[trtrand]))
return scalar tsec_adj_fpm_hr_UB = exp((_b[trtrand])+(1.96*_se[trtrand]))
gen conv1 = e(converged)
summ conv1
return scalar tsec_adj_fpm_conv = r(mean)
}
***RMST stci***
                stci if trtrand==0, rmean
                return scalar tsec_stci_con_mean = r(rmean)
                return scalar tsec_stci_con_lb = r(lb)
                return scalar tsec_stci_con_ub = r(ub)
                return scalar tsec_stci_con_se = r(se)

***FPM RMST***
capture stpm2 if trtrand==0, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 if trtrand==0, scale(h) df(3) lininit iterate(200)
                if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 if trtrand==0, scale(h) df(2) lininit iterate(200)
                if e(converged)==. | e(converged)==0 | _rc!=0 {
                capture stpm2 if trtrand==0, scale(h) df(1) lininit iterate(200)
}
```

```
}
}
gen conv2 = e(converged)
capture predict meansurv, rmst tmax(`admin') ci
if _rc!=0 {
capture predict meansurv, rmst tmax(`admin')
summ meansurv
return scalar tsec_adj_auc_con=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc==0 {
summ meansurv2
return scalar tsec_adj_auc_con=r(mean)
summ meansurv2_lci
return scalar tsec_adj_auc_con_LB=r(mean)
summ meansurv2_uci
return scalar tsec_adj_auc_con_UB=r(mean)
}
summ conv2
return scalar tsec_adj_fpm_con_conv = r(mean)

restore

}
if r(mean)!=0 & tdxo==1 {

preserve

drop if trtrand==1
drop if timeOS3<timePFSobs
drop if progressed==0
by id: replace obsno = _n
by id: egen minrisk=min(timeOS3)
by id: replace timeOS3=timeOS3-minrisk
by id: replace xotime=xotime-minrisk
by id: replace timeOS2=timeOS2-minrisk

sort id timeOS3
by id: gen catlag = cattdc[_n-1]
replace catlag=0 if catlag==.
by id: gen trtlag = trtnew[_n-1]
replace trtlag=0 if trtlag==.
gen visit1to6 = 0
replace visit1to6 = 1 if obsno==1 | obsno==2 | obsno==3 | obsno==4 | obsno==5 | obsno==6
gen visit7on = 0
replace visit7on = 1 if obsno>6
gen goodprognomlag = 0
replace goodprognomlag = 1 if bprog==0 & catlag==0
gen badprognomlag = 0
replace badprognomlag = 1 if bprog==1 & catlag==0
gen goodprogmlag = 0
replace goodprogmlag = 1 if bprog==0 & catlag==1
gen badprogmlag = 0
replace badprogmlag = 1 if bprog==1 & catlag==1
gen goodprognomlag1to6 = goodprognomlag*visit1to6
gen badprognomlag1to6 = badprognomlag*visit1to6
gen goodprogmlag1to6 = goodprogmlag*visit1to6
gen badprogmlag1to6 = badprogmlag*visit1to6

stset timeOS2, failure(died) id(id)
capture stgest3 trtnew goodprognomlag1to6 badprognomlag1to6 goodprogmlag1to6 badprogmlag1to6 visit7on trtlag, visit(obsno) nograph nolist saveres(gest1) range (-3 3) model(all)
outcome(mgale) test(cluster) nocheckobs replace
return scalar tsec_psi = r(psi)
return scalar tsec_af = exp(-r(psi))
return scalar tsec_af_LB = exp(-r(ucipsi))
return scalar tsec_af_UB = exp(-r(lcipsi))

gen conv1 = e(converged)
summ conv1
return scalar tsec_conv = r(mean)

restore
sort id
preserve

***Analysis on overall survival***
collapse (max) trtrand bprog timeOS2 xotime died cens timePFSobs admin, by(id)
by id: replace timePFSobs = 0 if timePFSobs==.
by id: replace xotime=0 if xotime==.
***below allows for recensoring***
gen cfact = timeOS2 if trtrand==1
gen dcfact = died if trtrand==1

replace cfact = (xotime + ((timeOS2-xotime)/return(tsec_af))) if (trtrand==0 & xotime>0)
replace cfact = timeOS2 if (trtrand==0 & xotime==0)
gen OSadminc = admin/return(tsec_af) if (trtrand==0 & return(tsec_af)>1.00)
replace dcfact = died if trtrand==0
replace dcfact=0 if (OSadminc<=cfact & trtrand==0)
```

```
replace cfact = OSadminc if (OSadminc<=cfact & trtrand==0)

***do survival analysis on re-estimated survival times***
stset cfact, failure(dcfact) id(id)

***FPM HR***
capture stpm2 trtrand, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 trtrand, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 trtrand, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 trtrand, scale(h) df(1) lininit iterate(200)
}
}
}
if _rc==0 {
return scalar tsec_adj_fpm_hr = exp(_b[trtrand])
return scalar tsec_adj_fpm_hr_SE = exp(_b[trtrand])*_se[trtrand]
return scalar tsec_adj_fpm_hr_LB = exp((_b[trtrand])-(1.96*_se[trtrand]))
return scalar tsec_adj_fpm_hr_UB = exp((_b[trtrand])+(1.96*_se[trtrand]))
gen conv1 = e(converged)
summ conv1
return scalar tsec_adj_fpm_conv = r(mean)
}
***RMST stci***
		stci if trtrand==0, rmean
		return scalar tsec_stci_con_mean = r(rmean)
		return scalar tsec_stci_con_lb = r(lb)
		return scalar tsec_stci_con_ub = r(ub)
		return scalar tsec_stci_con_se = r(se)

***FPM RMST***

capture stpm2 if trtrand==0, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 if trtrand==0, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 if trtrand==0, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 if trtrand==0, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv2 = e(converged)
capture predict meansurv, rmst tmax(`admin') ci
if _rc!=0 {
capture predict meansurv, rmst tmax(`admin')
summ meansurv
return scalar tsec_adj_auc_con=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc==0 {
summ meansurv2
return scalar tsec_adj_auc_con=r(mean)
summ meansurv2_lci
return scalar tsec_adj_auc_con_LB=r(mean)
summ meansurv2_uci
return scalar tsec_adj_auc_con_UB=r(mean)
}
summ conv2
return scalar tsec_adj_fpm_con_conv = r(mean)

restore

}
end
```

RPSFTM program

```
program define rpsftm_crabster, rclass
version 14.2

syntax [varlist] [if] [in] [, admin(real 5000) adminpps(real 4979)]

preserve
collapse (max) trtrand bprog timeOS2 xotime died cens timePFSobs admin progressed xo, by(id)
stset timeOS2, failure(died) id(id)

capture stpm2 trtrand, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 trtrand, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 trtrand, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 trtrand, scale(h) df(1) lininit iterate(200)
}
}
```

```
}
return scalar zittnl = _b[trtrand] / _se[trtrand]
restore
***RPSFTM with recensoring***
***note need to collapse data first***
preserve
collapse (max) trtrand xotdc bprog admin timeOS2 progressed timePFSobs xotime died cens, by(id)
stset timeOS2, failure(died) id(id)
replace xotime=0 if xotdc==0
capture {
strbee trtrand, xo0(xotime xotdc) test(logrank) endstudy(admin) gen(counterf) nokmgraph savedta(strbee1,replace)
return scalar RPSFTM = exp(-r(psi))
return scalar RPSFTM_UB = exp(-r(psi_low))
return scalar RPSFTM_LB = exp(-r(psi_upp))
replace counterf = timeOS2 if trtrand==1
replace dcounterf = died if trtrand==1
stset counterf, failure(dcounterf) id(id)

capture stpm2 trtrand, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
        capture stpm2 trtrand, scale(h) df(3) lininit iterate(200)
        if e(converged)==. | e(converged)==0 | _rc!=0 {
        capture stpm2 trtrand, scale(h) df(2) lininit iterate(200)
        if e(converged)==. | e(converged)==0 | _rc!=0 {
        capture stpm2 trtrand, scale(h) df(1) lininit iterate(200)
}
}
}
if _rc==0 {
return scalar RPSFTM_fpm_hr = exp(_b[trtrand])
return scalar RPSFTM_fpm_hr_SE = exp(_b[trtrand])*_se[trtrand]
return scalar seadjnl = _b[trtrand]/return(zittnl)
return scalar RPSFTM_fpm_hr_LB = exp(_b[trtrand]-1.96*return(seadjnl))
return scalar RPSFTM_fpm_hr_UB = exp(_b[trtrand]+1.96*return(seadjnl))
gen conv1 = e(converged)
summ conv1
return scalar RPSFTM_fpm_hr_conv = r(mean)
}
***Estimate mean survival directly from counterfactuals, fpm and weibull***

capture stpm2 if trtrand==0, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
        capture stpm2 if trtrand==0, scale(h) df(3) lininit iterate(200)
        if e(converged)==. | e(converged)==0 | _rc!=0 {
        capture stpm2 if trtrand==0, scale(h) df(2) lininit iterate(200)
        if e(converged)==. | e(converged)==0 | _rc!=0 {
        capture stpm2 if trtrand==0, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv2 = e(converged)
capture predict meansurv, rmst tmax(`admin') ci
if _rc==498 {
capture predict meansurv, rmst tmax(`admin')
summ meansurv
return scalar RPSFTM_sfunc_auc_con=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc!=498 {
summ meansurv2
return scalar RPSFTM_sfunc_auc_con=r(mean)
summ meansurv2_lci
return scalar RPSFTM_sfunc_auc_con_LB=r(mean)
summ meansurv2_uci
return scalar RPSFTM_sfunc_auc_con_UB=r(mean)
}
summ conv2
return scalar RPSFTM_adj_fpm_con_conv = r(mean)
}
***RMST stci***
        stci if trtrand==0, rmean
        return scalar RPSFTM_stci_con_mean = r(rmean)
        return scalar RPSFTM_stci_con_lb = r(lb)
        return scalar RPSFTM_stci_con_ub = r(ub)
        return scalar RPSFTM_stci_con_se = r(se)

restore

end
```

IPCW program

```
program define ipcw_crabster, rclass
version 14.2

syntax [varlist] [if] [in] [, admin(real 5000) adminpps(real 4979)]

***IPCW***
```

```
preserve
if tdxo==0 {

by id: drop if timeOS3 >(xotime)
by id: replace finalobs = 0
by id: replace finalobs = 1 if _n==_N
gen infcensOS=0
replace infcensOS=1 if xotdc==1 & trtrand==0
by id: replace died=. if (infcensOS==1 | cens==1)

***Deriving IPCWs***
summ timeOS3 if infcens==1
return scalar mink = r(min)
return scalar maxk = r(max)
_pctile timeOS3 if infcens==1, percentile(33,66)
if r(r1)== return(mink) {
rcsgen timeOS3, df(2) if2(infcens==1) gen(spline)
}
_pctile timeOS3 if infcens==1, percentile(33,66)
if r(r2)==return(maxk) {
rcsgen timeOS3, df(2) if2(infcens==1) gen(spline)
}
_pctile timeOS3 if infcens==1, percentile(33,66)
if r(r1)!=return(mink) & r(r2)!= return(maxk) {
rcsgen timeOS3, df(3) if2(infcens==1) gen(spline)
}
sort id obsno
capture logistic infcensOS bprog spline* if trtrand==0 & timeOS3==timePFSobs, iterate(200)
gen conv1 = e(converged)
gen omit1 = e(k_autoCns)
summ omit1
return scalar ipcwn1_omit = r(mean)
gen cds1 = e(N_cds)
summ cds1
return scalar ipcwn1_cds = r(mean)
gen cdf1 = e(N_cdf)
summ cdf1
return scalar ipcwn1_cdf = r(mean)

predict ptrtrec if e(sample), pr

**The above code estimates the probability of each individual recieving crossover treatment (and therefore being informatively censored) each day.  For the IPCW we need the probability of remaining uncensored, so we submit the probabilities from 1:**
replace ptrtrec=ptrtrec*infcensOS+(1-ptrtrec)*(1-infcensOS)
replace ptrtrec=1 if ptrtrec==.
**Now we estimate each individual's probability of their complete censoring history up to each day**
sort id obsno
by id: replace ptrtrec=ptrtrec*ptrtrec[_n-1] if _n!=1
rename ptrtrec censdenom

***Use stabilised weights***
capture logistic infcensOS bprog spline* if trtrand==0, iterate(200)
gen conv2 = e(converged)
gen conv3 = conv1*conv2
summ conv3
return scalar ipcwn1_conv = r(mean)

predict ptrtrec1 if e(sample), pr

**The above code estimates the probability of each individual recieving crossover treatment (and therefore being informatively censored) each day.  For the IPCW we need the probability of remaining uncensored, so we submit the probabilities from 1:**
replace ptrtrec1=ptrtrec1*infcensOS+(1-ptrtrec1)*(1-infcensOS)
replace ptrtrec1=1 if ptrtrec1==.
**Now we estimate each individual's probability of their complete censoring history up to each day**
sort id obsno
by id: replace ptrtrec1=ptrtrec1*ptrtrec1[_n-1] if _n!=1
rename ptrtrec1 censnum

***The stabilised weight is derived by dividing the numerator by the denominator:***
gen weightxo=censnum/censdenom
replace weightxo=1 if trtrand==1
rcsgen timeOS3, df(4) gen(splineOS)
capture logistic died trtrand bprog splineOS*[pw=weightxo], cluster(id) iterate(200)
return scalar ipcw_cox_hr = exp(_b[trtrand])
return scalar ipcw_cox_hr_SE = exp(_b[trtrand])*_se[trtrand]
return scalar ipcw_cox_hr_LB = exp((_b[trtrand])-(1.96*_se[trtrand]))
return scalar ipcw_cox_hr_UB = exp((_b[trtrand])+(1.96*_se[trtrand]))

gen conv4 = e(converged)
summ conv4
return scalar ipcw3_conv = r(mean)
gen omit3 = e(k_autoCns)
summ omit3
return scalar ipcw3_omit = r(mean)
gen cds3 = e(N_cds)
summ cds3
return scalar ipcw3_cds = r(mean)
gen cdf3 = e(N_cdf)
```

```
summ cdf3
return scalar ipcw3_cdf = r(mean)

summ weightxo if died!=. & trtrand==0
return scalar ipcw_weight_min = r(min)
return scalar ipcw_weight_max = r(max)
return scalar ipcw_weight_mean = r(mean)
return scalar ipcw_weight_sd = r(sd)
return scalar ipcw_weight_cv = r(sd)/r(mean)

***AUC mean***
stset timeOS2 died if infcens==0 [iw=weightxo], time0(timeOS3)

***FPM RMST***
capture stpm2 bprog if trtrand==0, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 bprog if trtrand==0, scale(h) df(3) lininit iterate(200)
		if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 bprog if trtrand==0, scale(h) df(2) lininit iterate(200)
		if e(converged)==. | e(converged)==0 | _rc!=0 {
		capture stpm2 bprog if trtrand==0, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv5 = e(converged)
if e(converged)==1 {
capture predict meansurv, rmst tmax(`admin') ci
if _rc!=0 {
capture predict meansurv, rmst tmax(`admin')
sort id obsno
by id: gen ind=1 if _n==1
summ meansurv if (ind==1 & trtrand==0)
return scalar ipcw_adj_auc_con=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc==0 {
sort id obsno
by id: gen ind2=1 if _n==1
summ meansurv2 if (ind2==1 & trtrand==0)
return scalar ipcw_adj_auc_con=r(mean)
summ meansurv2_lci if (ind2==1 & trtrand==0)
return scalar ipcw_adj_auc_con_LB=r(mean)
summ meansurv2_uci if (ind2==1 & trtrand==0)
return scalar ipcw_adj_auc_con_UB=r(mean)
}
}
summ conv5
return scalar ipcw_adj_fpm_con_conv = r(mean)

restore
}
if tdxo==1 {

by id: drop if timeOS3 >(xotime)
by id: replace finalobs = 0
by id: replace finalobs = 1 if _n==_N
gen infcensOS=0
replace infcensOS=1 if xotdc==1 & trtrand==0
by id: replace died=. if (infcensOS==1 | cens==1)
sort id timeOS3
by id: gen catlag = cattdc[_n-1]
replace catlag=0 if catlag==.

***Deriving IPCWs***
summ timeOS3 if infcens==1
return scalar mink = r(min)
return scalar maxk = r(max)
_pctile timeOS3 if infcens==1, percentile(33,66)
if r(r1)== return(mink) {
rcsgen timeOS3, df(2) if2(infcens==1) gen(spline)
}
_pctile timeOS3 if infcens==1, percentile(33,66)
if r(r2)==return(maxk) {
rcsgen timeOS3, df(2) if2(infcens==1) gen(spline)
}
_pctile timeOS3 if infcens==1, percentile(33,66)
if r(r1)!=return(mink) & r(r2)!= return(maxk) {
rcsgen timeOS3, df(3) if2(infcens==1) gen(spline)
}
*denom model
sort id obsno
gen visit1to6 = 0
by id: replace visit1to6 = 1 if
((timePFSobs)==timeOS3|(timePFSobs+21)==timeOS3|(timePFSobs+42)==timeOS3|(timePFSobs+63)==timeOS3|(timePFSobs+84)==timeOS3|(timePFSobs+105)==timeOS3)
gen goodprognomlag = 0
replace goodprognomlag = 1 if bprog==0 & catlag==0
gen badprognomlag = 0
replace badprognomlag = 1 if bprog==1 & catlag==0
```

```
gen goodprogmlag = 0
replace goodprogmlag = 1 if bprog==0 & catlag==1
gen badprogmlag = 0
replace badprogmlag = 1 if bprog==1 & catlag==1
gen goodprognomlag1to6 = goodprognomlag*visit1to6
gen badprognomlag1to6 = badprognomlag*visit1to6
gen goodprogmlag1to6 = goodprogmlag*visit1to6
gen badprogmlag1to6 = badprogmlag*visit1to6

capture logistic infcensOS goodprognomlag1to6 badprognomlag1to6 goodprogmlag1to6 badprogmlag1to6 spline* if trtrand==0 &
((timePFSobs==timeOS3)|((timePFSobs+21)==timeOS3)|((timePFSobs+42)==timeOS3)|((timePFSobs+63)==timeOS3)|((timePFSobs+84)==timeOS3)|((timePFSobs+105)==timeOS3)),
iterate(200)
gen conv2 = e(converged)
return scalar ipcwn2_conv = r(mean)
gen omit1 = e(k_autoCns)
summ omit1
return scalar ipcwn1_omit = r(mean)
gen cds1 = e(N_cds)
summ cds1
return scalar ipcwn1_cds = r(mean)
gen cdf1 = e(N_cdf)
summ cdf1
return scalar ipcwn1_cdf = r(mean)

predict ptrtrec if e(sample), pr
**The above code estimates the probability of each individual recieving crossover treatment (and therefore being informatively censored) each day.  For the IPCW we need the probability
of remaining uncensored, so we submit the probabilities from 1:**
replace ptrtrec=ptrtrec*infcensOS+(1-ptrtrec)*(1-infcensOS)
replace ptrtrec=1 if ptrtrec==.
**Now we estimate each individual's probability of their complete censoring history up to each day**
sort id obsno
by id: replace ptrtrec=ptrtrec*ptrtrec[_n-1] if _n!=1
rename ptrtrec censdenom
*model numerator
capture logistic infcensOS bprog spline* if trtrand==0, iterate(200)
gen conv3 = e(converged)
gen conv4 = conv2*conv3
summ conv4
return scalar ipcwn1_conv = r(mean)

predict ptrtrec3 if e(sample), pr
replace ptrtrec3=ptrtrec3*infcensOS+(1-ptrtrec3)*(1-infcensOS)
replace ptrtrec3=1 if ptrtrec3==.
sort id obsno
by id: replace ptrtrec3 = ptrtrec3*ptrtrec3[_n-1] if _n!=1
rename ptrtrec3 censnum

***The stabilised weight is derived by dividing the numerator by the denominator:***
gen weightxo=censnum/censdenom
replace weightxo=1 if trtrand==1
rcsgen timeOS3, df(4) gen(splineOS)
capture logistic died trtrand bprog splineOS*[pw=weightxo], cluster(id) iterate(200)
return scalar ipcw_cox_hr = exp(_b[trtrand])
return scalar ipcw_cox_hr_SE = exp(_b[trtrand])*_se[trtrand]
return scalar ipcw_cox_hr_LB = exp(_b[trtrand])-(1.96*_se[trtrand]))
return scalar ipcw_cox_hr_UB = exp((_b[trtrand])+(1.96*_se[trtrand]))

gen conv5 = e(converged)
summ conv5
return scalar ipcw3_conv = r(mean)
gen omit3 = e(k_autoCns)
summ omit3
return scalar ipcw3_omit = r(mean)
gen cds3 = e(N_cds)
summ cds3
return scalar ipcw3_cds = r(mean)
gen cdf3 = e(N_cdf)
summ cdf3
return scalar ipcw3_cdf = r(mean)

summ weightxo if died!=. & trtrand==0
return scalar ipcw_weight_min = r(min)
return scalar ipcw_weight_max = r(max)
return scalar ipcw_weight_mean = r(mean)
return scalar ipcw_weight_sd = r(sd)
return scalar ipcw_weight_cv = r(sd)/r(mean)

***AUC mean***
stset timeOS2 died if infcens==0 [iw=weightxo], time0(timeOS3)

***FPM RMST***

capture stpm2 bprog if trtrand==0, scale(h) df(4) lininit iterate(200)
if e(converged)==. | e(converged)==0 | _rc!=0 {
        capture stpm2 bprog if trtrand==0, scale(h) df(3) lininit iterate(200)
        if e(converged)==. | e(converged)==0 | _rc!=0 {
        capture stpm2 bprog if trtrand==0, scale(h) df(2) lininit iterate(200)
        if e(converged)==. | e(converged)==0 | _rc!=0 {
```

```
        capture stpm2 bprog if trtrand==0, scale(h) df(1) lininit iterate(200)
}
}
}
gen conv6 = e(converged)
if e(converged)==1 {
capture predict meansurv, rmst tmax(`admin') ci
if _rc!=0 {
capture predict meansurv, rmst tmax(`admin')
sort id obsno
by id: gen ind=1 if _n==1
summ meansurv if (ind==1 & trtrand==0)
return scalar ipcw_adj_auc_con=r(mean)
}
capture predict meansurv2, rmst tmax(`admin') ci
if _rc==0 {
sort id obsno
by id: gen ind2=1 if _n==1
summ meansurv2 if (ind2==1 & trtrand==0)
return scalar ipcw_adj_auc_con=r(mean)
summ meansurv2_lci if (ind2==1 & trtrand==0)
return scalar ipcw_adj_auc_con_LB=r(mean)
summ meansurv2_uci if (ind2==1 & trtrand==0)
return scalar ipcw_adj_auc_con_UB=r(mean)
}
}
summ conv6
return scalar ipcw_adj_fpm_con_conv = r(mean)

restore
}
end
```