GFORMULA SAS MACRO version 4.0

The GFORMULA macro implements the non-iterative conditional expectation (NICE) algorithm of the g-formula algorithm (Robins 1986, Robins et al. 2004, Hernan and Robins 2020). The g-formula can estimate an outcome's counterfactual mean or risk under hypothetical treatment strategies (interventions) when there is sufficient information on time-varying treatments and confounders.

This macro can be used for discrete or continuous time-varying treatments and for failure time outcomes or continuous/binary end of follow-up outcomes. The macro can handle a random measurement/visit process and a priori knowledge of the data structure, as well as censoring (e.g., by loss to follow-up) and two options for handling competing events for failure time outcomes. Interventions can be flexibly specified, both as interventions on a single treatment or as joint interventions on multiple treatments.

For a quick overview of how to use the GFORMULA, see a simple example in "Example of the g-formula macro call". For a detailed list of options, see the sections "Specifying parametric model assumptions" and "Specifying the intervention."

Authors
s W. Logan, Jessica G. Young, Sarah Taubman, Yu-Han Chiu, Sara Lodi, Sally Picciotto, Goodarz Danaei, Emma E. McGee, Miguel A. Hernán

Version 4.0, October 2025
This version includes options and improvements that are not compatible with previous versions of the software. For questions and comments, email Emma McGee (emcgee@hsph.harvard.edu)

Table of Contents

Overview

The GFORMULA macro implements the parametric g-formula (Robins 1986, Robins et al. 2004) to estimate the risk, probability, or mean of an outcome under (hypothetical) sustained treatment strategies or interventions from longitudinal data with time-varying measurements of treatments and confounders. Examples of published applications of the GFORMULA macro include Taubman et al. (2009), Young et al. (2011), Danaei et al. (2013), Lajous et al. (2013), Garcia-Aymerich et al (2014), Lodi et al (2015, 2017), and Zhang et al (2017).

The g-formula is a generalization of standardization to the time-varying setting. When the measured variables are sufficient to adjust for confounding and selection bias (Robins 1986, Hernán and Robins 2016), the g-formula under a user-specified intervention identifies the outcome distribution in the study population had that intervention been, possibly contrary to fact, implemented in all individuals in the population. The parametric g-formula is an approach to estimating the g-formula under parametric model assumptions.

The GFORMULA macro allows users to specify many types of interventions, including static, dynamic, deterministic and random interventions. See Young et al. (2014) for a description of these different types of interventions. The macro also allows interventions that depend on a subject's' "natural value of treatment" at time k– i.e., the value of treatment that would have been observed at k for a given subject were the intervention discontinued right before k (Robins et al. 2004). Stronger causal assumptions are required for this special case and are given jointly in Richardson and Robins (2013) and Young et al. (2014). The structure of the g-formula macro and its submacros is presented in the Appendix.

The GFORMULA macro requires a data structure in which each row contains the values for an individual during an interval (e.g., day, month, year). The macro supports binary, categorical and continuous treatment and covariates, and 4 types of outcomes:

- A time-varying indicator of failure (*outctype = binsurv*). The macro calculates the cumulative probability (risk) of the outcome under each user-specified intervention.
- A binary outcome *at the end of follow-up* (*outctype=bineofu*). The macro calculates the probability of the outcome *at the end of follow-up* under each user-specified intervention.
- A categorical or ordinal outcome a*t the end of follow-up* (*outctype=cateofu*). The macro calculates the probability of each outcome category *at the end of follow-up* under each user-specified intervention.
- A continuous outcome at t*he end of the follow-up (outctype=conteofu, conteofu2, conteofu3, or conteofu4)*. The macro calculates the outcome mean *at the end of follow-up* under each user-specified intervention. Continuous outcomes can be modeled using 4 different approaches (see *Outline of the algorithm* Step1b for further details).

The macro supports data with censoring events. Under the assumption that the measured covariates are sufficient to control selection bias by censoring, the g-formula yields estimates under a hypothetical intervention that eliminates censoring. For outcome type *binsurv* only, the user has the choice not to treat competing risk events as censoring events. When competing risk events are not treated as censoring events, the calculation of the cumulative risk of failure under intervention corresponds to calculating a weighted average of subdistribution cumulative incidence functions for the outcome (event) of interest. When competing risk events are treated as censoring events, the calculation for the cumulative risk of failure under intervention corresponds to calculating a weighted average of the complement of Kaplan Meier survival estimates (see below and Young et al 2020 for details).

Outline of the algorithm

Let *k* denote time of follow-up measured in intervals (e.g. days, months, years) of follow-up ranging from interval *k*=0 (baseline) through *k*=*K* (end of follow-up), and *p* the number of measured covariates at each *k*. The *p* covariates *cov1, cov2,…,covp* include both treatments (i.e. variables to undergo intervention) and adjustment covariates at each *k*. The algorithm implemented by the macro consists of three main steps.

Step 1: Parametric estimation

Step 1a. The macro fits user-specified parametric models to estimate the conditional joint density of the *p* covariates at each time *k*>0 given past covariate history through *k*-1, or *f*(*cov1,…,covp* |past). This joint density is estimated via a product of conditional densities: *f*(*cov1*|past)*f*(*cov2*|*cov1*,past)…*f*(*covp*|*covp-1,…,cov1*,past). A separate user-specified parametric model is fit for each conditional density of this product. The user specifies the model for each covariate (logistic, linear…) and the ordering of the covariates *cov1,…,covp*. In the absence of model misspecification, the g-formula estimates are insensitive to the ordering (i.e. the order in which time-varying covariates are assigned to *cov1,…covp*). In practical settings where some model misspecification is likely, estimates may be sensitive to this choice.

Step 1b. The macro fits a user-specified parametric model for an outcome functional conditional on past covariate history. All regression models are pooled over time except for the regression models for the outcomes at the end of follow-up (*outctype= bineofu, cateofu, conteofu, conteofu2, conteofu3, conteofu4)*.

- For *outctype = binsurv*, the macro fits a pooled over interval logistic model to estimate the conditional discrete-time hazard of the outcome event at each time *k*. When *compevent_cens=0* (a macro parameter taking value of 0 when a competing risk event is not treated as a censoring event), the macro will also fit a logistic model to estimate the conditional discrete-time hazard of the competing event at each time *k*, which is necessary to compute the subdistribution cumulative incidence function.
- For *outctype=bineofu*, the macro fits a logistic model to estimate the conditional probability by the end of follow-up using only records at *k=K*.
- For *outctype=cateofu*, the macro fits j-1 nested logistic models (where j is the number of outcome categories) to estimate the conditional probability of each outcome category by the end of follow-up using only records at *k=K*. The first logistic model has as its dependent variable an indicator that the outcome = 1. The second logistic model has as its dependent variable an indicator that the outcome = 2 among records where the outcome is not equal to 1, and so on. Levels of the outcome variable must be coded as integers beginning at 1.
- For *outctype=conteofu*, the macro fits a linear regression model to estimate the conditional mean at the end of follow-up using only records at *k=K*. The model is fit using PROC REG.
- For *outctype=conteofu2*, the macro fits a truncated normal regression model to estimate the conditional mean at the end of follow-up using only records at *k=K*. The model is fit using PROC QLIM (truncated option) with the lower bound set to the minimum value of the outcome in the observed data plus an offset term (-0.01%) and the upper bound set to the maximum value of the outcome in the observed data plus an offset term (+0.01%).
- For *outctype=conteofu3*, the macro fits a Tobit regression model to estimate the conditional mean at the end of follow-up using only records at *k=K*. The model is fit using PROC QLIM (censored option) with

lower bound set to the minimum value of the outcome in the observed data and upper bound set to the maximum value of the outcome in the observed data.

- For *outctype=conteofu4*, the macro uses a logistic to log-linear approach. Specifically, the macro fits 1) a logistic model for an indicator that the outcome is > 0 and 2) a linear regression model for the natural log of the outcome restricted to records whether the outcome is > 0. Both models are fit at the end of follow-up using only records at *k=K*. This approach may be appropriate when the outcome has many zero values.

Step 2: Monte Carlo simulation

The macro generates a "large" number *n* (by default, set to the sample size) of covariate histories consistent with the intervention. Recursively, for each covariate history generated through *k-1*, values of the *p* covariates at time *k>0* are generated using the estimated model coefficients obtained in step 1a (with k=0 covariate values set to the observed data values). After the values of *cov1,…,covp* are generated at time *k*, the value of the covariates that are to undergo intervention are then changed according to the user-specified intervention rule.

For outcome types *bineofu, cateofu, conteofu*, *conteofu2*, *conteofu3*, and *conteofu4,* this process continues through *k=K*. The outcome mean is then generated for each of the n generated histories based on the estimated model coefficients from Step 1b. For *outctype = binsurv*, after covariates are generated and treatment(s) "assigned" according to the intervention at each *k*, the discrete hazard is estimated conditional on each of the n generated histories consistent with intervention through *k* based on the estimated model coefficients from Step 1b.

For *outctype = binsurv* and when *compevent* is not left empty, the discrete hazard of the competing risk event is also estimated at each *k* from the estimated model coefficients saved from Step 1b.

Step 3: Calculation of risk/mean under each intervention

For outcome types *bineofu, cateofu, conteofu*, *conteofu2*, *conteofu3*, and *conteofu4*, the final estimate of the population mean under the time-varying user-specified intervention is the probability or mean of the n history-specific outcome means computed from Step 2. For *outctype = binsurv*, the time- and history-specific hazard estimates are combined using the complement of Kaplan-Meier method (*compevent_cens=1)* or using an estimate of the subdistribution cumulative incidence function (Kalbfleisch and Prentice, 1980; Gooley et al., 1999) to obtain an estimate of the history-specific risk by end of follow-up (*compevent_cens*=0). The final estimate of the population risk by end of follow-up under the user-specified intervention is the mean of the history-specific cumulative risks. See Young et al (2020) for details on descriptions of competing events.

Steps 2 and 3 may be repeated for multiple interventions. Confidence intervals for risk/mean estimates and causal effect estimates are obtained by nonparametric bootstrapping, i.e., by repeating these three steps for many bootstrap samples. See a description of the algorithm in Taubman et al. (2009) and Young et al. (2014).

The natural course

The macro can provide estimates under the natural course, i.e. under no intervention on treatment at any time (but elimination of censoring events if they are present in the data). When calculating estimates under the natural course, no change is made to any of the values of the covariates *cov1,…,covp* after they are simulated at time k in step 2 above. The natural course can serve as a meaningful reference (*refint*=0). Furthermore, the natural course estimate can be used to assess model specification of the g-formula. Specifically, analysts interested in assessing model specification of the g-formula estimates can use inverse probability (IP) weighting to estimate the natural course risk from the observed data and compare these IP-weighted estimates with the parametric g-formula estimates. See Chiu et al (2022) for details. This comparison is included in the GFORMULA

Macro Version 4.0. Setting the macro parameter *rungraphs*=1 will output a .pdf file comparing the natural course estimates (both outcome risk and covariate means) via IP weighting at each time to those generated under the natural course via the parametric g-formula. “Large” difference between the natural course estimates by IP weighting and those by the parametric g-formula suggest serious model misspecification of either (i) models used for the parametric g-formula or (ii) the censoring model used for IP weighting. Of note, while a model for censoring is not required for the parametric g-formula, when using *rungraphs*=1, users need to include a column of a time-varying censoring variable in the dataset and specify the parameter *censor* because the model for censoring will be used to estimate natural course by IP weighted method (for the purpose of assessment of model misspecification).

When competing events are present, in GFORMULA Macro Version 4.0, we require the user to include a column with a time-varying indicator of competing events in the input dataset, regardless of whether competing events are treated as censoring events or not. For example, if death from CVD (variable: CVD_death) is a competing event for the outcome of interest, then users need to specify the macro parameter *compevent*=CVD_death*.* A new parameter *compevent_cens* allows users to determine whether competing events are treated as censoring events or not. Specifically, if the user chooses to treat competing events as censoring events (i.e., the algorithm targets a controlled direct effect), then set the macro parameter *compevent_cens=1.* Models for censoring and competing events will then be used to calculate natural course estimates by IP weighted method. When competing events are not treated as a censoring event (i.e., the algorithm targets a total effect), then users need to set the macro parameter *compevent_cens=0*, and models for competing events will be used in the g-formula to calculate the subdistribution cumulative incidence functions for the event of interest.

Required structure of the input data set

The data must be arranged with one record per subject (identified by the macro parameter *id*) and time k (identified by the parameter *time*). The parameter *time* must begin at 0 for each subject (the interval that subject enters the study or “baseline”) and increment by 1 for each subsequent interval. The parameter *timepoints* specifies the desired end of follow-up interval. For example, if *time* indexes month of follow-up and the 60-month risk/probability/mean is of interest then the user should set *timepoints*=60. The largest possible value of *time* allowable for any subject must be *timepoints*-1 as the *time* index must begin at 0.

For each *id*, the record *time*=0 corresponds to the first time interval in which that subject meets the eligibility criteria for the study (baseline). This baseline may or may not correspond to the same calendar time for all subjects. For outctype=*bineofu* or *conteofu*, *time*=0 should occur early enough to allow for complete follow-up. For example, if interest is in the outcome mean at 5-year follow-up and the administrative end of the study was in 2010 (all subjects stopped being followed by this time), then only subjects whose *time*=0 is in 2005 or earlier should be included in the data set.

Baseline time-fixed covariates (fixed in value across *time* records for the same subject *id*) are specified in the parameter *fixedcov*. Time-varying covariates for each *id* measured in each *time=k* are specified in the parameters *cov1-covp*.

Coding requirements for outcomes in the input data set depends on the outcome type:

i) outctype=binsurv

The time-varying indicator of the failure event of interest is specified in the parameter *outc*. For records with *time*=k the failure indicator on that line should be defined as an indicator of whether the outcome (event of interest) has occurred for that *id* between interval *k* and interval *k*+1 covariate measurements. When *outc*=1 on line *time*=*k* for a given subject, this should be the last line in the data for that subject. When time intervals are of long duration and measurements within intervals are cross-sectional, coding decisions to ensure covariates "precede" the outcome on a given line k may require assumptions.

If there are censoring events in the data, the user may have a variable in the data set that is a time-varying indicator of censoring in the time between covariate measurements in interval *k* and interval *k*+1. There may be multiple reasons for censoring. The macro expects that the input data is coded such that, if a censoring event occurs between interval k and interval *k*+1 covariate measurements, then the line *time*=*k* is the last record in the data for that subject. On such lines, *outc* may be coded as missing or 0. If coded as missing then such censored records will be excluded when fitting the hazard model in Step 1b of the algorithm (see Section *Outline of the Algorithm*) and in estimating the component hazards for nonparametric estimation of observed risks (see Section *Natural Course*). If coded as 0 they will be included in computing these estimates. This choice will make no difference to estimates when intervals are made small enough such that there are no failures in intervals where there are censoring events (and the pooled logistic model in step 1b perfectly fits the data). This decision of whether to code the *outc* as 0 or missing on line *k* if a subject is first censored between *k* and *k*+1 covariate measurements is conceptually a choice of whether to count such subjects in the *time*=*k* risk set or not, respectively.

The following is an example of how the data should be structured for a subject with *id*=66 who experiences the outcome (variable "death") after measurement of *time*=5 covariates yet before we would have measured *time*=6 covariates such that *time*=5 is this subject's last record in the data set:

obs	id	time	mt	fish	cig	cigprebl	cens	Death (outcome)
1	66	0	6	2	8	0	0	0
2	66	1	5	3	0	0	0	0
3	66	2	4	3	0	0	0	0
4	66	3	6	2	0	0	0	0
5	66	4	9	2	5	0	0	0
6	66	5	8	0	6	0	0	1

Here *outc*=death, *time*=time and *id*=id with mt, fish and cig time-varying measurements of meat intake, fish intake and number of cigarettes per day (allowed to take different values for the same *id* over different values of time) which will be assigned in some order *cov1* through *cov3*. The variable cigprebl is a time-fixed covariate which might be assigned to *fixedcov* and which takes the same value across values of *time* for the same *id*. Finally, cens is a time-varying censoring indicator in the data set, the model of which is not used in the parametric g-formula. However, in GFORMULA Macro Version 4.0, users need to specify a censoring variable (here, *censor*=cens) in order to compute nature course estimate via IP weighted method. For subject *id*=66, cens=0 at all times as this subject has the outcome of interest prior to censoring. When the parameter *censor* is left empty, the macro will compare the observed risk/mean with the natural course estimate by the parametric g-formula.

The following is an example of records from the same data set for a different subject (*id*=1) who is censored prior to the desired end of follow-up (after *time*=4 covariate measurements yet before we would have measured *time*=5 covariates) such that *time*=4 is this subject's last record in the data set. Here we have chosen to code death as missing for censored records such that these records will not be included in the estimation of death hazards.

obs	id	time	mt	fish	cig	cigprebl	cens	death (outcome)
1	1	0	6	4	12	0	0	0
2	1	1	6	4	0	0	0	0
3	1	2	6	2	6	0	0	0
4	1	3	2	6	0	0	0	0
5	1	4	5	3	0	0	1	.

When competing risk events are present in the data, the macro parameter *compevent* encodes a time-varying indicator of a competing risk event between interval k and k+1 covariate measurements. The following is an example of how the data should be structured for a subject with *id*=78 who experiences a competing event (variable "CVD_death") after measurement of *time*=2 covariates yet before we would have measured *time*=3 covariates, such that *outc* after a competing event is coded as missing and *time*=2 is this subject's last record in the data set.

obs	id	time	mt	fish	cig	cigprebl	CVD_death	cens	cancer_death (outcome)
1	77	0	6	4	12	0	0	0	0
2	77	1	6	4	0	0	0	0	0
3	77	2	6	2	6	0	0	0	0
4	77	3	2	6	0	0	0	0	0
5	77	4	5	3	0	0	0	1	.
6	78	0	5	3	0	0	0	0	0
7	78	1	5	4	0	0	0	0	0
8	78	2	2	2	0	0	1	.	.

In GFORMULA Macro Version 4.0, we use the macro parameter *compevent_cens* to indicate whether competing events are handled as a censoring event or not.

- When competing risk events are considered censoring events (i.e., the algorithm targets the controlled direct effect), the user should set the parameter *compevent_cens* to be 1. In this example, *compevent=CVD_death, censor*=cens, *compevent_cens=1*, *outc*=cancer_death.
- If competing risk events are not to be treated as censoring events (i.e., the algorithm targets a total effect on the outcome event), the user should set the parameter *compevent_cens* = 0. Risk estimates in this case will be a function of the discrete hazard for competing risk conditional on past treatment and covariates.

For details on competing events, see Young et al. (2020). For details on implementing natural course risk via IP weighting, see Chiu et al (2022).

ii) ***<u>outctype=</u> bineofu, cateofu, conteofu*, *conteofu2*, *conteofu3*, or *conteofu4***

The binary, categorical / ordinal, or continuous outcome at the end of follow-up is specified in the parameter *outc*. The value of *outc* will only be used for records with *time= timepoints*-1, and ignored for records with any other value of *time*. As above, if a subject is censored between interval k and k+1 covariate measurements for any time=k then this should be the last record in the data for that subject. In this case, these subjects will contribute to estimating the covariate densities (see Outline of the Algorithm above). Unlike survival outcomes (where users have two different options for handling competing events), for binary, categorical / ordinal, and continuous end of follow up outcomes, competing events are treated as a censoring event (*compevent_cens=1*).

Example of the g-formula macro call

We now illustrate the general structure of the g-formula macro call through a simple example with input data set "sample", *id* the name of a variable in the data set "sample" containing the subject identifier "id", *time* the name of a variable in the data set "sample" containing the follow-up time index "time"(here *id* and *time* happen to have the same name as the macro parameter but do not have to) and *timepoints*=6. The file example1.sas is a sample dataset. This calls the GFORMULA macro. The outcome is the time-varying indicator from diabetes "dia", thus we set *outctype*=binsurv. Here "dead" is the name of a variable in the data set "sample containing a time varying indicator of death from a competing risk. The call assigns this variables name to the macro parameter *compevent* in order to estimate effects on the scale of the subdistribution cumulative incidence function (see Sections *Overview* and *Outline of the Algorithm*). We wish to adjust for potential baseline confounding by age (*fixedcov*=baseage) as well as for potential time-varying confounding by the covariate hypertension (*cov1*=hbp). The treatment/exposure is time-varying physical activity (*cov2*=act). To guarantee that rerunning the code yields the same results we set *seed*=9458. We select *inttype1*=2 to estimate the risk of diabetes under a threshold intervention on exercise such that all subjects "exercise at least 30 minutes per day" (see Table 4 below).

The main components of the call to the GFORMULA macro are as follows. Notice that every parameter must be followed by a comma.

```
** Read the gformula macro;
  %include 'mypath/gformula4.0.sas';

** Specify the intervention(s) before calling the gformula macro;
%let interv1 = intno=1, nintvar=1,
   intlabel='All subjects exercise at least 30 minutes per day in all intervals',
   intvar1 = act, inttype1 = 2, intmin1=30, intpr1=1, inttimes1 = 0 1 2 3 4 5 ;

**Call to the GFORMULA macro;
title 'GFORMULA SAMPLE';
%gformula(
** Declare input dataset, time variable and indicators for the outcome event, censoring (if any) and competing risk (if any);
data= bytimes,
id=id,
time=time,
timepoints = 10,
timeptype= concat,
timeknots = 1 2 3 4 5 6 7 8 9,
outc=y,
outctype=binsurv,

** Specify a censoring variable (for the purpose of estimating IP-weighted natural course risk and covariate mean), and parameter
maxipw allows user to truncate the upper limit of the censoring weight.
censor=cens,
maxipw=1000
**Specify the variable for competing events
compevent=d,
**Specify whether competing events are treated as a censoring event (1 if yes, 0 if no)
compevent_cens=1
** Specify the number of interventions other than the natural course;
numint=1,

** List of time fixed covariates, if any;
fixedcov = ,

** Specify the time-vayring covariates ;
ncov=2,
cov1  = L,   cov1otype  = 1, cov1ptype = lag2bin,
cov2  = A,   cov2otype  = 3, cov2ptype = lag2bin,

** Specify whether to generate graphs for comparing natural course estimates by g-formula vs by IP weighting.
rungraphs = 1 ,

** Specify the seed and the number of bootstrap samples ;
seed= 9458,
nsamples = 20
);
```

The output from the above call is provided here:

```
                      PREDICTED RISK UNDER SEVERAL INTERVENTIONS

     Interv.    Description
        0       Natural course
        1       All subjects maintain A to at least 0.18 in all intervals

                                IP-weighted natural course risk= 35.73 %
           Data= sample, Sample size= 1000, Monte Carlo sample size= 1000
                           Number of bootstrap samples= 10
                             Reference intervention is 0

              Lower  Upper         Lower  Upper                           %        Aver %
        Risk  limit  limit  Risk   limit  limit Bootstrap Bootstrap Intervened Intervened
Interv.  (%)  95% CI 95% CI ratio  95% CI 95% CI Risk Mean  Risk SE       On          On

   0    35.72  35.54   35.88   1.00   1.00   1.00      35.71      0.09       0.0         0.00

   1    34.83  34.58   35.00   0.98   0.97   1.98      34.82      0.12       100        82.68


                 Lower    Upper                Lower    Upper               Lower    Upper
          Risk   limit    limit      Risk      limit    limit   # Needed    limit    limit
  Interv.  (%)   95% CI   95% CI  difference   95% CI   95% CI  to Treat    95% CI   95% CI

     0    35.72   35.54    35.88     0.00        0.0     0.00        .          .        .

     1    34.83   34.58    35.00    -0.88      -0.96    -0.79     -113       -127     -104


                     RESTRICTED MEAN SURVIVAL TIME AFTER 10 TIME POINTS
```

```
                    Data= sample, Sample size= 1000, Monte Carlo sample size= 1000
                                   Number of bootstrap samples= 20
                                     Reference intervention is 0

                                                  Restricted
              Restricted                             mean
                 mean         Lower      Upper     survival      Lower      Upper
               survival       limit      limit       time        limit      limit
       int       time        95% CI     95% CI    difference    95% CI     95% CI

        0       7.953         7.937      7.961       0.000       0.000      0.000

        1       8.002         7.987      8.016       0.049       0.043      0.055
```

The natural course risk estimate by IP-weighting is 35.73%. The parametric g-formula estimate of the natural course risk is 35.72% and the corresponding estimated risk under the activity intervention is 34.83%. The estimated risk ratio, using the natural course as the reference intervention, is 0.98 with a 95% confidence interval based on 10 bootstrap samples of (0.97, 0.98). The estimated risk difference is -0.88 with 95% CI (-0.96, -0.79). Restricted mean survival time under each intervention is the sum of the estimated cumulative survival probabilities over all follow-up times. The files example3.sas and example4.sas include other examples.

Specifying parametric model assumptions for covariates

For all outcome types, the user should specify a covXotype and covXptype for each time-varying covariate *covX* (*cov1,…covp*).

For binary, categorical / ordinal, and continuous end of follow-up outcomes (<u>*outctype=*</u> *bineofu, cateofu, conteofu*, *conteofu2*, *conteofu3*, or *conteofu4*), the user should additionally specify whether they would like to use a more complex history of time-varying covariates in the outcome model than that specified by covXptype. If the user would like to use the covariate history defined by covXptype in the outcome model, then they should set *usehistory_eof = 0.* If the user would like to use a more complex history of time-varying covariates in the outcome model than that defined by covXptype, they should set *usehistory_eof = 1* and additionally specify a *covXetype* for each time-varying covariate (see Table 3). Caution should be exercised when setting *usehistory_eof = 0* for an end of follow-up outcome because this approach may not directly account for potential time-varying confounding at all time points prior to the end of follow-up.

COVXOTYPE

The macro parameter *covXotype* is used to select the SAS regression fitting procedure for the density of each time-varying covariate *covX (cov1,…covp)*. The selection of *covXotype* also determines how the value of the time-varying covariate is simulated at each time *k*. Options available for *covXotype* are summarized in Table 1.

Note that when *covXotype* = 5 the user must specify *covXptype* = -cat type (lag1cat, lag2cat, lag3cat, or skpcat) and *covXetype* = cat (see next sections).

Table 1: Summary of SAS regression procedures and simulation rules by choice of covXotype

covXotype	**covX variable type**	**SAS procedure(s)**	**Notes on model fit**	**Rules for simulation**
1	Binary variable	PROC LOGISTIC	Fits model to all records	CovX is generated based on estimated model parameters
2	Binary variable	PROC LOGISTIC	Fits model only to records where the first lagged value of covX=0. Should be used for binary covariates that, once they switch from 0 to 1, they stay 1 (e.g. indicator of diabetes diagnosis)	CovX is generated based on the estimated model parameters until the first 1 is generated. After that the value of covX is always set to 1.
3	Continuous variable	PROC REG	Fits model to all records	CovX is generated according to a normal density with mean estimated by the estimated regression model coefficients and variance estimated by the sample variance of covX . If sim_trunc = 1, then generated values greater than the maximum (minimum) observed value of covX in the dataset are subsequently set to this maximum (minimum) value.
4	Continuous variable	PROC LOGISTIC & PROC REG	Fits a logistic model for indicator that covX is > 0 and a linear regression model for the natural log of covX restricting to records where covX>0. The procedure internally creates two variables: z_covX (indicator for covX>0) and l_covX (natural log of covX). covXotpye = 4 might be appropriate when covX has many zero values (e.g., number of cigarettes per day).	z_*covX* is simulated using coefficients estimated by PROC LOGISTIC. If the generated value of z_*covX* is 0 then *covX* is generated as 0. Otherwise if the generated value of z_*var* is 1, l_*covX* is generated using the second set of coefficients estimated by PROC REG and covX is set to the exponentiated , l_*covX*. Generated values greater than the maximum (minimum) observed value in the *dataset* are subsequently set to this maximum (minimum) value.
5	Categorical/ordinal variable	PROC LOGISTIC Calls j-1 times where j is the number of levels of covX	Dependent variable in first call is indicator that covX=1; dependent variable in second call is indicator that covX=2 among records where covX ne 1 and so on. Levels of *covX* must be coded as integers beginning at 1.	Levels of *covX* are simulated as integers beginning at 1 from the estimated model parameters.

6	Continuous variable	PROC QLIM (TRUNCATED option)	Fits a truncated normal regression model TRUNCATED option with lower bound set to the minimum value of *covX* in the observed dataset plus an offset term (-.01%) and upper bound set to the maximum value of *covX* plus an offset term (+.01%).	CovX is generated according to truncated normal using estimated regression parameters and sample variance of covX
7	Continuous variable	PROC QLIM (CENSORED option)	Fits Tobit regression model CENSORED option with lower bound set to the minimum value of covX in observed dataset and upper bound set to the maximum value of *covX*	CovX is generated according to a normal density with mean defined by the estimated regression model coefficients and variance estimated as sample variance of covX. Generated values greater than the maximum (minimum) observed value of covX in the dataset are subsequently set to this maximum (minimum) value.

COVXPTYPE

The macro parameters *covXptype* (*cov1ptype,…,covpptype*) are jointly used to create and select the functional form of the "past covariate history" included as independent variables in the parametric regression models for the *p* covariates and the outcome functional (for *outctype = binsurv* or for *outctype= bineofu, cateofu, conteofu*, *conteofu2*, *conteofu3*, or *conteofu4* and *usehistory_eof = 0*). The program contains options to incorporate different function of this history through appropriate selection of these parameters. Lagged values of *covX* must be included in the input data set data with names covX_l1, covX_l2, covX_l3, for certain choices of *covXptype* (see Table 2 below).

Table 2 summarizes different options for *covXptype*. For any given choices of *covXptype*, X=1,…,p, independent variables included in any given regression model will depend on the "order" of *covX* relative to the dependent variable of that model. For this reason, in Table 2, in describing how different choices of *covXptype* impact how the "history" of *covX* appears in a given regression model, we will distinguish between:

- a model for the density of covY, where Y<X ; heretofore "preceding covariate model"
- a model for the density of covY, where Y>X ; heretofore "succeeding covariate model"
- an "outcome regression model," which refers to the pooled over time regression model for the discrete hazard of the event or competing risk (*outctype=binsurv*) or, for *outctype= bineofu, cateofu, conteofu*, *conteofu2*, *conteofu3*, or *conteofu4* and *usehistory_eof = 0*, a regression model for the outcome using only records with time=timepoints-1.

Note: The log file will automatically display both externally and internally created variables included as independent variables in each regression fit so the user can see the implications of selecting different values of *covXptype*.

Table 2. Summary of options for covXptype

covXptype	**Description**	**Details**	**Required additional parameters**
lag1- types: *lag1bin* *lag1cat* *lag1qdc* *lag1zqdc* *lag1cub* *lag1spl*	Function of most recent lag	The algorithm uses and creates a function of *covX* and *covX_l1*. The function is determined by the suffix (bin, cat, qdc,zqdc,cub,spl). For ''preceding covariate models'', the algorithm includes the suffix-determined function of *covX*_l1 as an independent variable; for ''succeeding covariate models'', the suffix-determined function of *covX* and *covX*_l1; for ''outcome regression models'', the function of *covX*. The user must include a variable named covX_l1 that corresponds to the first lagged value of *covX* in the input data set (e.g. if *covX*=cig then must include a variable named cig_l1).	
lag2- types: *lag2bin* *lag2cat* *lag2qdc* *lag2zqdc* *lag2cub* *lag2spl*	Function of most recent two lags	The algorithm uses and creates a function of *covX*, *covX_l1* and *covX_l2*. The function is determined by the suffix (bin, cat, qdc,zqdc,cub,spl). For ''preceding covariate models'', the algorithm includes the suffix-determined function of *covX* _l1 and *covX* _l2 as independent variables; for ''succeeding covariate models'', the suffix-determined function of *covX*, *covX* _l1 and *covX* _l2; for ''outcome regression models'', the function of *covX* and *covX* _l1. The user must include variables named covX_l1 and covX_l2 that correspond to the first and second lagged values of covX, respectively, in the input data set (e.g. if *covX*=cig then must include variables named cig_l1 and cig_l2).	
lag3- types: *lag3bin* *lag3cat* *lag3qdc* *lag3zqdc* *lag3cub* *lag3spl*	Function of most recent three lags	The algorithm uses and creates a function of the lag variables *covX* _l1, *covX* _l2 and *covX* _l3 are created internally. The function is determined by the suffix (bin, cat, qdc,zqdc,cub,spl). For ''preceding covariate models'', the algorithm includes the suffix-determined function of *covX* _l1, *covX* _l2 and *covX* _l3 as independent variables; for ''succeeding covariate models'', the suffix-determined function of *covX*, *covX* _l1, *covX* _l2, and *covX* _l3; for ''outcome regression models'', the function of *covX*, *covX* _l1 and *covX* _l2. The user must include variables named covX_l1, covX_l2 and covX_l3 that correspond to the first, second and third lagged values of covX, respectively, in the input data set (e.g. if *covX*=cig then must include variables named cig_l1, cig_l2 and cig_l3).	

skp- types: *skpbin* *skpcat* *skpqdc* *skpzqdc* *skpcub* *skpspl*	Function of last measured value	The algorithm uses and creates (i) a function of *covX* and *covX* _l1 and (ii) interaction terms between each variable comprising these functions and the time between the current interval k and the last time *covX* was actually measured. The function is determined by the suffix (bin, cat, qdc,zqdc,cub,spl). This lapse of time is determined by the length of each interval — as defined by the macro parameter *interval* — and the intervals in which *covX* is not measured for any subject — as defined by the macro parameter *covXskip*. The skp type should only be used in data sets arising from cohort studies where variables *covX* are not measured for *any individual* during certain follow-up times. See FAQ 7. The user must include a variable named covX_l1 that corresponds to the first lagged value of covX in the input data set (e.g. if covX=cig then must include a variable named cig_l1).	covXskip *For skpcat and skpspl:* covXknots
-bin types: *lag1bin* *lag2bin* *lag3bin* *skpbin*	No transformation	identity function (no transformation)	
-cat types: *lag1cat* *lag2cat* *lag3cat* *skpcat*	Function of category indicators	transformed to category indicators, cutoffs for categories defined by *covXknots*,. Reference level is highest level	covXknots
-qdc types: *lag1qdc* *lag2qdc* *lag3qdc* *skpqdc*	Quadratic function	quadratic function (linear and squared terms)	
-zqdc types: *lag1zqdc* *lag2zqdc* *lag3zqdc* *skpzqdc*	Quadratic function	quadratic function (linear and squared terms). Includes a product term with “variable is greater than zero”	
-cub types: *lag1cub* *lag2cub*	Cubic function	cubic function (linear, squared and cubed terms)	

lag3cub *skpcub*			
-spl types: *lag1spl* *lag2spl* *lag3spl* *skpspl*	Splines	restricted cubic spline transformation; knots defined by covXknots	covXknots
tsswitch1	Function of time since covX switched from 0 to 1	Intended for any variable *covX* with covXotype=2 (e.g. an indicator of disease diagnosis by interval k) to allow for modelling the history of covX as a function of the time since covX last switched from 0 to 1 at each time k. Specifically, for "preceding covariate models" (excepting Y=X), the algorithm includes *covX* _l1 and an internally created variable *tscovX _l1_inter* (a linear term for the product of *covX* _l1 and the cumulative sum of *covX* from time=0 through time=k-1) as independent variables; for "succeeding covariate" and "outcome regression models", *covX* and internally created *tscovX _inter* (a linear term for the product of *covX* and the cumulative sum through k) are included as independent variables. The user must include a variable named covX_l1 that corresponds to the first lagged value of covX in the input data set (e.g. if covX=cig then must include a variable named cig_l1). To include a restricted cubic spline transformation of the *tscovX_l1_inter* and *tscovX_inter* terms (rather than linear terms), the user must specify knots via covXknots.	*For splines:* covXknots
cumavg	Cumulative average	Creates and includes a linear term for the cumulative average of the entire history of *covX* relative to interval k beginning from time=0 To include a restricted cubic spline transformation of this cumulative average (rather than a linear term), the user must specify knots via covXknots.	*For splines:* covXknots
cumavgcat *lag1-* *lag2-*	Categorical version of the cumavg ptype	Creates a categorical version of the cumavg ptype, cutoffs for categories defined by *covXknots*,. Reference level is highest level	covXknots
lag1cumavg	Cumulative average where the last term is pulled off the average	A variation of the cumavg ptype where the last term is pulled off of the average. In this case there are two generated predictors. At time = k these will be *covX* _l1 and a linear term for the cumulative average of *covX* from time = 0 to time = k-2. To include a restricted cubic spline transformation of the cumulative average term (rather than a linear term), the user must specify knots via covXknots.	*For splines:* covXknots
lag2cumavg	Cumulative average where the last two	A variation of the cumavg ptype where the last two terms are pulled off of the average. In this case there are three generated predictors. At time = k these will be covX_l1, covX_l2, and a linear term for	*For splines:*

	terms are pulled off the average	the cumulative average of *covX* from time = 0 to time = k-3. To include a restricted cubic spline transformation of the cumulative average term (rather than a linear term), the user must specify knots via covXknots.	covXknots
rcumavg	Recent cumulative average	Creates and includes the cumulative average of restricted history of *covX* relative to interval k based on two most recent values only.	

COVXETYPE

When estimating effects on binary, categorical / ordinal, or continuous end of follow-up outcomes (*outctype= bineofu, cateofu, conteofu*, *conteofu2*, *conteofu3*, or *conteofu4*), users may wish to include a more complex function of "past covariate history" in the outcome model than that defined by covXptype because, unlike for failure time outcomes, the outcome model is not pooled over time and only the most recent values of time-varying covariates at the end of follow-up will generally be included when using functional forms defined by *covXptype*. To include a more complex covariate history, users must specify *usehistory_eof = 1* and include the macro parameters *covXetype* (*cov1etype,…,covpetype*), which allow users to specify the functional form of the "past covariate history" that will be included as independent variables in the outcome regression model.

The *covXetype* is comprised of 2 components: 1) the functional form of *covX* that will be used in the outcome model at the end of follow-up, and 2) the number of lags (i.e., values from prior periods) of *covX* that will be used. Table 3 summarizes the options for each of these components.

Note that the macro parameter *usehistory_eof* must be set to *usehistory_eof = 1* in order for the *covXetypes* to be used. Note also that, unlike with the *covXptyp*e, the *covXetype* may use lagged values of *covX* that are not included in the input data set (e.g., if the *covXetype* requires up to 4 lagged values but the covXptype only requires up to 2 lagged values for covX, then the input dataset must include variables named covX_l1 and covX_l2 but it does not need to contain variables names covX_l3 or covX_l4).

The log file will automatically display both externally and internally created variables included as independent variables in each regression fit so the user can see the implications of selecting different values of *covXetype*. Because the potential number of covariates included in the outcome model may be large with certain combinations of *covXetype*, users should carefully review the coefficients and standard errors of all covariates included in the outcome model and should inspect the log files for warning messages (e.g., quasi-complete separation of data points).

Table 3. Summary of options for covXetype

covXetype	**Description**	**Details**	**Required additional parameters**
Component 1: functional form of *covX*			
bin *bin all* *bin none* *bin r*	No transformation	identity function (no transformation) for all, none, or the r most recent lags of *covX*	
cat *cat all* *cat none* *cat r*	Function of category indicators	transformed to category indicators for all, none, or the r most recent lags of *covX*. Cutoffs for categories defined by *covXeknots*. Reference level is highest level. If *covXotype* = 5 then *covXetype* must be set to *cat* and the cutoffs for the categories will be set equal to those defined by *covXknots* (see Table 2).	covXeknots
qdc *qdc all* *qdc none* *qdc r*	Quadratic function	quadratic function (linear and squared terms) for all, none, or the r most recent lags of *covX*	
zqdc *zqdc all* *zqdc none* *zqdc r*	Quadratic function	quadratic function (linear and squared terms) for all, none, or the r most recent lags of *covX*. Includes a product term with "variable is greater than zero"	
cub *cub all* *cub none* *cub r*	Cubic function	cubic function (linear, squared, and cubed terms) for all, none, or the r most recent lags of *covX*	

spl *spl all* *spl none* *spl r*	Splines	restricted cubic spline transformation for all, none, or the r most recent lags of *covX*; knots defined by covXeknots	covXeknots
skp- types: *skpbin all, none, or r* *skpcat all, none, or r* *skpqdc all, none, or r* *skpzqdc all, none, or r* *skpcub all, none, or r* *skpspl all, none, or r*	Function of last measured value	When a skp- type is specified, the macro creates lagged versions of covX, as described above, except that only lagged values that correspond to times when covX was actually measured are included in the outcome model. Covariate values from any skipped times will not appear in the outcome model. The functional form of covX at the measured times is determined by the suffix (bin, cat, qdc,zqdc,cub,spl). See FAQ 7 and the skp- covXptype for additional details.	covXskip *For skpcat and skpspl:* covXeknots
tsswitch1 *tsswitch1 all* *tsswitch1 none* *tsswitch1* ***r***	Function of time since covX switched from 0 to 1	Intended for any variable *covX* with covXotype=2 (e.g. an indicator of disease diagnosis by interval k) to allow for modelling the history of covX as a function of the time since covX last switched from 0 to 1 at each time k. For these covX, setting covXetpye = tsswitch1 is identical to setting covXetype = cumsum. In either case, the macro will create and include a linear term indicating the time since covX switched to 1 (0 if never switched to 1, otherwise time since switch). To include a restricted cubic spline transformation of the time since covX switched to 1 (rather than a linear term), the user must specify knots via covXeknots.	*For splines:* covXeknots
cumavg *cumavg all* *cumavg none* *cumavg* ***r***	Cumulative average	Creates and includes a linear term for the cumulative average of the history of *covX* over all, none, or the r most recent lags of *covX*. The second component of the covXetype is used to specify the *r* most recent values that will be used to compute the cumulative average. Specifically, the cumulative average is calculated beginning from time=*K* (end of follow-up) through time=*K*–number of lags (*r)* specified in component 2 of the *covXetype*. For example, if *covXetype = cumavg 4*, then the macro will include in the outcome model at the end of follow-up a linear term representing the cumulative average of *covx* over the last 4 time points. To include a restricted cubic spline transformation of this cumulative average (rather than a linear term), the user must specify knots via covXeknots.	*For splines:* covXeknots

cumavgcat *cumavgcat all* *cumavgcat none* *cumavgcat r*	Categorical version of the cumavg etype	Creates a categorical version of the cumavg etype, with the cutoffs for categories defined by *covXeknots*. Reference level is highest level.	covXeknots
cumavgnew *cumavgnew all* *cumavgnew none* *cumavgnew r*	Cumulative average where the last k terms are pulled off the average	A variation of the cumavg etype where the last *r* terms are not included in the cumulative average but instead are included as individual lagged covariates in the outcome model at the end of follow-up. For example, if *covXetype = cumavgnew 4* and timepoints = K = 10, then the macro will include in the outcome model at the end of follow-up 1) the lagged values of *covX* from the 4 prior time points (time = K-1, time = K-2, time=K-3, and time=K-4) and 2) a continuous variable representing the cumulative average of *covX* from the start of follow-up time = 0 to time = K - 5. In this case, there would be 5 generated predictors for covX: covX_9_eof, covX_8_eofu, cov7_eof, covX_6_eof, and covX_cumavg_5_eof, where the first 4 predictors represent the values of covX at timepoints 9, 8, 7, and 6 and the last predictor represents a linear term for the cumulative average of covX from time = 0 to time = 5. To include a restricted cubic spline transformation of the cumulative average term (rather than a linear term), the user must specify knots via covXeknots.	*For splines:* covXeknots
cumavgcatnew *cumavgcatnew all* *cumavgcatnew none* *cumavgcatnew r*	Categorical version of the cumavgnew etype	Creates a categorical version of the cumavgnew etype, with the cutoffs for categories defined by *covXeknots*. Reference level is highest level.	covXeknots
cumsum *cumsum all* *cumsum none* *cumsum r*	Cumulative sum	Creates and includes the cumulative sum of the history of *covX* over all, none, or the r most recent lags of *covX*. The second component of the covXetype is used to specify the *r* most recent values that will be used to compute the cumulative sum. Specifically, the cumulative sum is calculated beginning from time=*K* (end of follow-up) through time=*K*–number of lags (*r)* specified in component 2 of the *covXetype*. For example, if *covXetype = cumsum 4*, then the macro will include in the outcome model at the end of follow-up a linear term representing the cumulative sum of *covx* over the last 4 time points. To include a restricted cubic spline transformation of the cumulative average term (rather than a linear term), the user must specify knots via covXeknots.	*For splines:* covXeknots

cumsumcat *cumsumcat all* *cumsumcat none* *cumsumcat r*	Categorical version of the cumsum etype	Creates a categorical version of the cumsum etype, with the cutoffs for categories defined by *covXeknots*. Reference level is highest level.	covXeknots
cumsumnew *cumsumnew all* *cumsumnew none* *cumsumnew r*	Cumulative sum where the last k terms are pulled off the average	A variation of the cumsum etype where the last *r* terms are not included in the cumulative sum but instead are included as individual lagged covariates in the outcome model at the end of follow-up. For example, if *covXetype = cumsumnew 4* and timepoints = K = 10, then the macro will include in the outcome model at the end of follow-up 1) the lagged values of *covX* from the 4 prior time points (time = K-1, time = K-2, time=K-3, and time=K-4) and 2) a continuous variable representing the cumulative sum of *covX* from the start of follow-up time = 0 to time = K - 5. In this case, there would be 5 generated predictors for covX: covX_9_eof, covX_8_eofu, cov7_eof, covX_6_eof, and covX_cumsum_5_eof, where the first 4 predictors represent the values of covX at timepoints 9, 8, 7, and 6 and the last predictor represents a linear term for the cumulative sum of covX from time = 0 to time = 5. To include a restricted cubic spline transformation of the cumulative average term (rather than a linear term), the user must specify knots via covXeknots.	*For splines:* covXeknots
cumsumnewcat *cumsumnewcat all* *cumsumnewcat none* *cumsumnewcat r*	Categorical version of the cumsumnew etype	Creates a categorical version of the cumsumnew etype, with the cutoffs for categories defined by *covXeknots*. Reference level is highest level.	covXeknots
Component 2: number of lags of *covX* to use			
all *bin all* *cat all* *qdc all* *zqdc all* *cub all* *spl all* *skpbin all* *skpcat all*	All lags of covX	All lags of covX will be used in the outcome model at the end of follow-up. This is the default. and is identical to listing r = timepoints. Caution should be exercised when using all lags because this may result in a large number of model parameters relative to the number of records with time=timepoints-1 that contribute to the outcome model. Users should carefully review the coefficients and standard errors of all covariates included in the outcome model and should inspect the log files for warning messages (e.g., quasi-complete separation of data points).	

skpqdc all *skpzqdc all* *skpcub all* *skpspl all* *tsswitch1 all* *cumavg all* *cumavgcat all* *cumavgnew all* *cumavgcatnew all* *cumsum all* *cumsumcat all* *cumsumnew all* *cumsumcatnew all*		
none *none* *bin none* *cat none* *qdc none* *zqdc none* *cub none* *spl none* *skpbin none* *skpcat none* *skpqdc none* *skpzqdc none* *skpcub none* *skpspl none* *tsswitch1 none*	No lags of covX	No lags of covX will be used in the outcome model at the end of follow-up. In this case, covX will not appear at all in the outcome model. For cumavg- or cumsum- etype variables, no first component is required (e.g., the user may simply specify covXetype = none). This is identical to listing r = 0.
r, where r can be any integer from r = 0 (same as specifying none) to r = timepoints (same as specifying all) *bin r*	r lags of covX	The r most recent values of covX will be used. For (skp)bin, (skp)cat, (skp)qdc, (skp)zqdc, (skp)cub, (skp)spl, cumavg, cumavgcat, cumsum, and cumsumcat, this means that only the r most recent values of covX will be used when creating the history of covX. See component 1 for more details on how r is used for cumavgnew, cumavgcatnew, cumsumnew, and cumsumcatnew. When r = 0, this is identical to listing none and covX will not appear in the outcome model.

cat r	When r = 1, 2, or 3, this is identical to using a lag1-, lag2-, or lag3- ptype (respectively).
qdc r	When r = timepoints, this is identical to listing all and all lags of covX will be used.
zqdc r	
cub r	
spl r	
skpbin r	
skpcat r	
skpqdc r	
skpzqdc r	
skpcub r	
skpspl r	
tsswitch1 r	
cumavg r	
cumavgcat r	
cumavgnew r	
cumavgcatnew r	
cumsum r	
cumsumcat r	
cumsumnew r	
cumsumcatnew r	

The two components of *covXetype* should be separated by a space.

Specifying the intervention

A single intervention may be defined on up to 8 treatment variables in each interval k. Interventions are currently defined by setting certain macro parameter values prior to calling the GFORMULA macro using the following syntax:

```
%let interv1 = intno=1, intlabel = , intcond = , nintvar= , … ;
```

Parameters are as follows:

intno (required)

Intervention number. This should be indexed 1 to numint.

intlabel (required)

Description of the intervention to be output with the results.

intcond (optional)

Condition under which the intervention is implemented (if not always implemented).

nintvar (required)

Number of intervened on variables. A maximum value of 8 is allowed.

intvisittype (optional, default = 1)

Indicates how a variable is carried forward during skip times (i.e., when no measurement of a covariate occurs) or when there is not an associated visit. The default approach is to carry forward the previous value of that covariate under the intervention (i.e., the last intervened on value). This was the method used in previous versions of the macro. When intvisittype = 2 the previous natural course/simulated value had the intervention been discontinued right before the last measurement time is carried forward. Note: This option is only used when intvar# has a skip-type ptype or for variables with an associated visit process when there is not a simulated visit.

For #=1,…nintvar

intvar# (required)

Variable undergoing intervention.

inttype# (required for all intervention variables)

Type of intervention for intvar#. There are 6 possible types, summarized in Table 1.

inttimes# (optional, default = -1)

Times at which intvar# will be intervened on. The default is -1, which will lead to no intervention.

intpr# (optional, default=1)

probability that the intervention occurs. The default is 1, which will lead to always intervening when other conditions are met.

intmin# (required for inttype 2)

intmax# (required for inttype 2)

For inttype#= 2, the user assigns the threshold value to intmin# if the goal is to maintain treatment values under intervention above the threshold. The intmax# is parallel to intmin# but sets a maximum rather than a minimum. Both intmin# and intmax# may be specified to maintain treatment values under intervention within some range (e.g. eat at least 1 serving of fish but no more than 3 servings of fish per week)

intchg# (required for inttype 3)

Fixed amount that is added to intvar#.

intcond# (used in inttype 5. Default 1=1)

For grace period interventions, this is the condition that is used to determine when to start the grace period for initiating a treatment. This condition can include any variable that is modeled in a covX.

intgrace# (required for inttype 5)

Length of the grace period for treatment. This specifies the maximum length of time that is allowed to pass before treatment must be initiated after the condition in intcond# is satisfied. During all time points of the grace period before the end of the grace period, treatment is assigned randomly with probability based on the time to the end of the grace period. At the end of the grace period, treatment is set to 1 for all individuals who have not yet had treatment set to 1 during the grace period. Before the condition specified in intcond# is satisfied, treatment is assigned to be 0. With inttype = 5, the treatment model is not used in the Monte-Carlo step.

intvalue# (required for inttype 1)

Fixed target for the values of intvar#. There are several pre-defined interventions supported by the algorithm which are chosen using the inttype# macro parameter for # possibly ranging from 1,…4. Here we describe some choices of inttype# that have been applied in the literature with references. Other choices are more briefly summarized in Table 1.

Table 4: Summary of intervention types

inttype	**Description**	**Action**	**Required additional parameters**
1	static deterministic intervention	sets to fixed value intvalue#	intvalue#
2	threshold intervention as considered by Taubman et al. (2009), Young et al. (2014).	assigns intmin#/intmax# if natural value is below intmin#/above intmax#, otherwise sets to the natural value	intmin#/intmax#
3	fixed change	adds a fixed amount intchg# to the natural value	intchg#
4	previous value	carries forward value from previous time	

5	grace period	Allows a binary treatment, with otype = 2, to be intervened on within a grace period (with length specified by intgrace#) after a condition (specified by intcond#) is satisfied. During all time points of the grace period before the final time point, the treatment value is randomly assigned with an increasing probability based on the amount of time that has passed since the start of the grace period. At the end of the grace period, if the treatment was not previously set to 1, the intervention will set treatment to 1 at that time. (Note: This intervention type does not use the model for treatment fit during the Monte Carlo step to determine the value of treatment assigned during the grace period.)	intcond# and intgrace#
-1	user defined intervention. the user must provide a user-defined macro with code to implement the intervention.	Examples of user-defined interventions are provided in the supporting documentation and in the appendix.	User-defined macro intusermacro#. The macro should be placed above the call to %gformula. Can also use

Description of the GFORMULA macro parameters

data (required)
Input dataset to be used for the analysis.

id (required)
Unique identifier for subjects in the dataset *data*.

time (required)
Time index in the dataset *data*. This <u>must</u> begin at 0 (the interval that subject enters the study or "baseline") for each subject (indexed by *id*, see above) and increment by 1 for each subsequent interval. The largest possible value of *time* for any subject must be *timepoints*-1 (see below and also details in Section 2).

timeptype (required)
Function of time/interval to be included in pooled over time models. Choices include *conbin, concat, conqdc, concub, and conspl*. The function is determined by the choice of suffix. These are as described as Table 2 for

covXptype. For suffix -cat and -spl, the user must also define *timeknots*.

timeknots (optional)
Knots or category cutoffs when *timeptype* is selected as conspl or concat.

timepoints (required)
End of follow-up. The parameter *timepoints* specifies the desired end of follow-up interval. For example, if *time* indexes month of follow-up and the 60- month risk/mean is of interest then the user should set *timepoints*=60. The largest possible value of *time* allowable for any subject must be *timepoints*-1 as the *time* index must begin at 0.

timefuncgen (optional)
User-defined macro that provides general functions of time (other than splines and categories) which can be used in the covariate and outcome models. It should be used together with *covXaddvars* and *covXsaddvrs* when the time function is included in the model for covX, or with *eventaddvars* and *eventsaddvars* when the new time function is included in the model for the outcome. For more details, see FAQ 1.

interval (optional, default = 1)
Length of time between one unit increments in *time* (assumed constant). The value of this parameter is only used if any *covXptype* is chosen as a skp- type. See Sections 5 and 6 for details.

outc (required)
Outcome variable in the dataset *data*.

outctype (optional, default=binsurv)
Type of outcome: *binsurv* (time-varying failure indicator; default), *bineofu* (binary outcome at end of follow-up), *cateofu* (categorical or ordinal outcome at end of follow-up), *conteofu* (continuous outcome at end of follow-up modeled using linear regression), *conteofu2* (continuous outcome at end of follow-up modeled using truncated normal regression), *conteofu3* (continuous outcome at end of follow-up modeled using Tobit regression), and *conteofu4* (continuous outcome at end of follow-up modeled using a logistic to log-linear approach).

outcinteract (optional)
Product ("interaction") terms to be included as independent variables in the outcome regression model. These can include product terms between any baseline variable included in fixedcov, any time-varying covariate covX or time (note, product terms with time for the outcome model are only relevant for outctype=binsurv). The structure of the interaction is a product term bN*bM or bN*M or N*M, where the b indicates that the product term is with a baseline variable and the N or M alone indicates that the product term is with a covX or time. The N(M) in bN(bM) is the numeric location of the baseline covariate included in fixedcov. The N(M) in N(M) alone is the index X for the time-varying covariate covX or 0 for time. The *outcinteract* option can also be used to stratify the outcome model by a discrete covariate. See FAQ 2 and FAQ 3 for more details.

compevent
A variable indicating a competing risk event in the dataset *data*. This should be specified regardless of whether competing risk events are or are not treated as censoring events. See Section 2 for coding requirements and further explanation.

compevent_cens
A variable indicating how a competing event is handled for survival outcomes. When competing events are

treated as a censoring event, then *compevent_cens=1*. When competing events are treated as no event (i.e., not a censoring event), then *compevent_cens=0*. For continuous or binary end of follow up outcome, competing events are considered as a censoring event (*compevent_cens=1*).

compeventinteract (optional)
Product (“interaction’) terms to be included in the above hazard model for the competing risk event if *compevent* is assigned a value. See *outcinteract* for the required syntax.

eventaddvars (optional)
List of variables (should be baseline variables not included in all of the modeled covariate models) to add to the outcome model. For more information, see *covXaddvars*.

compeventaddvars (optional)
Additional variables and user defined macros for adding extra variables to the *compevent* models that were not included in other lists. For more information see *covXaddvars*.

censor
A variable indicating a censoring event (e.g., loss to follow up) in the dataset. The model of censoring will be used to estimate the natural course risk/mean, which will be compared with the estimates by the parametric g-formula.

maxipw
Specify the maximal value for a censoring weight. If users wish not to truncate the censoring weight, then set maxipw to a big value (e.g., maxipw=1000).

censoraddvars (optional)
Additional variables and user defined macros for adding extra variables to the censoring model that was not included in other lists. For more information see covXaddvars.

usehistory_eoof (optional, default = 0)
For binary, categorical / ordinal, or continuous end of follow-up outcomes, this option specifies whether the user would like to use a more complex history of time-varying covariates in the outcome model than that which was specified by *covXptype*. If the user would like to use the covariate history defined by *covXptype* in the outcome model, then *usehistory_eof = 0.* If the user would like to use a more complex history of time-varying covariates in the outcome model than that defined by *covXptype*, *usehistory_eof = 1* and a *covXetype* must be specified for each time-varying covariate (see Table 3). Caution should be exercised when setting *usehistory_eof = 0* for an end of follow-up outcome because this approach may not directly account for time-varying confounders at all time points prior to the end of follow-up.

fixedcov (optional)
List of time-fixed covariates at baseline (i.e. any variables defined in the input data set *data* which have constant values over time within levels of *id*) to control confounding/selection bias. Variables should be separated by spaces. For example, to define *fixedcov* to include sex, race and birthplace, define *fixedcov=sex race birthplace*. The user may also use *fixedcov* as a means of including the baseline values of any time-varying covariates *covX* in all functions of time-varying covariate history; thus allowing for a slightly more flexible function of history than that created using certain choices of *covXptype*. For example, say we define *covX*=CD4 for some X=1,…,p. Defining *fixedcov*=sex race CD4 would allow the outcome model and all time k covariate models to depend on the baseline

value of CD4 as well as the pre-baseline subject characteristics sex and race in addition to dependence on the post-baseline function of CD4 determined by the choice of *covXptype* in this example (e.g. *covXptype*=lag2cat or, for end of follow-up outcomes when *usehistory_eof = 1*, *covXetype= cat 2).*

ncov (required)
Number of time-varying covariates in the analysis (p). The maximum allowable value is 30.

For all X=1,…, *ncov*:

covX (required for X=1)
Specifies a variable corresponding to a time varying covariate value for each person-interval in the dataset *data*.
covXotype (required for X=1 and, when covX not empty, X>1)
Specifies the regression procedure where *covX* is the dependent variable and related specifications for the simulation in step 2 of the algorithm. Possible types are described in Table 1.

covXptype (required for X=1 and, when covX not empty, X>1)
Specifies how the history of *covX* will be included in all regression models for failure event outcomes and, when *usehistory_eof = 0,* for binary, categorical / ordinal, and continuous end of follow-up outcomes. Types are described in Table 2.

covXetype (required when *usehistory_eof = 1*)
Specifies how the history of *covX* will be included in the outcome regression model for binary, categorical / ordinal, and continuous end of follow-up outcomes (*outctype=* *bineofu, cateofu, conteofu*, *conteofu2*, *conteofu3*, or *conteofu4*) when *usehistory_eof = 1*. The first component specifies the functional form of covX. The second component specifies the number of lags that will be used. Types are described in Table 3. The default number of lags is all.

covXmtype (optional, default = all, can be all, some , or nocheck)
checkaddvars (optional, default = 1)
CovXmtype along with *checkaddvars* specifies how the variable covX appears in the models for covY. When covXmtype = all the variable will appear in all the models. This is the default option and covX should not appear in any covYaddvars. In this case, the variable c*heckaddvars* will not affect how covX appears as a predictor in the model for covY. If it is desired to restrict how covX appears in any model then *covXmtype* should be set to some or nocheck. When *covXmtype* = something the variable will need to be added into the desired models by using the *covYaddvars* variable (or *eventaddvars* or *censdaddvars*). When the option is set to something (ie non blank), the user can use the name of the variable set in covX in covYaddvars. When *checkaddvars* = 1 the macro will then replace the value of covX in the variable *covYaddvars* with the corresponding predictors defined by *covXptype* as if *covXmtype* was set to all. If *checkaddvars* = 0 then the user will need to list the functional form of covX that is desired in the model for covY in covYaddvars. NOTE: Care must be used so that the temporal order in conditional densities is preserved. In this case, the gformula macro will have available predictors that are based on the *ptype* listed in *covXptype.* If the user desires other forms of covX, then the *covXgenmacro* option can be used to generate these. When covXmtype = something there is a special case in determining how covX appears as a predictor in the model for covX. In the case of covXmtype = some the predictors will be the same as if covXmtype = all. If the user wants to change this the option of covXmtype = nocheck can be used and the predictors can be added in the covXaddvars list. As in the case for covYaddvars, the user can add in the desired form of covX. This can be used to model covX with no dependence of previous values of covX. Note: When covXmtype is not all any form of the baseline value of covX should not appear in the fixedcov variable. When covXmtype = some or nocheck the baseline variables should also be included in the covYaddvars when covX is

included.

covXcumint (optional for covXpytpe= *lag1cumavg* and *lag2cumavg*)
Specifies a variable to indicate which observations to include in the average terms when the ptype is lag1cumavg or lag2cumavg. CovXcumint should be a 0-1 variable which once it is 1 it is always 1 and should also be one of the CovY. When the covXcumint variable is 0, the average will be set to be 0. See Table 2.

covXknots (required for –cat and -spl ptype variables and, if splines are desired for the cumulative average variable of covX, for cumavg, lag1cumavg, and lag2cumavg ptypes)
Knots or category cutoffs when *covXptype* is a -cat or -spl type or, if splines are desired for the cumulative average variable of covX, when *covXptype* is a cumavg, lag1cumavg, or lag2cumavg type. There are two ways in which *covXknots* can be specified: i) by listing the location of the desired knots or category cutoffs (e.g., *covXknots = 0.5 5 7.5 11)*, or, for -spl, cumavg, lag1cumavg, and lag2cumav ptypes, ii) by setting *covXknots = 0 (or NONE), 3, 4, or 5*, where this integer value represents the number of percentiles to use when defining the spline knots (0 (or NONE) = no knots, 3 = 3 knots at the 10th, 50th, and 90th percentiles, 4 = 4 knots at the 5th, 25th, 75th, and 95th percentiles, and 5 = 5 knots at the 5th, 25th, 50th, 75th, and 95th percentiles). For -cat ptypes, only option i) (i.e., a list of the location of the desired category cutoffs) can be used. Note that, if the values of at least 2 of these percentiles are the same, then the macro will produce division by 0 errors when calculating the spline variables. In this case, the user must either define the exact locations of the spline knots (rather than percentiles) or consider alternative functional forms for that covX. Note also that for *covXptype = lag1cumavg or lag2cumavg* and when covXknots is specified with a value other than 0, spline terms will be created only for the cumulative average variable of covX (not for the lagged variables of *covX*). See Table 2 for more details.

covXeknots (required for (skp)cat, (skp)spl, cumavgcat, cumavgcatnew, cumsumcat, and cumsumcatnew etype variables and, if splines are desired for the cumulative average or sum variable of covX, for cumavg, cumavgnew, cumsum, and cumsumnew etypes)
Knots or category cutoffs when *covXetype* is a (skp)cat, (skp)spl, cumavgcat, cumavgcatnew, cumsumcat, or cumsumcatnew type or, if splines are desired for the cumulative average or sum variable of covX, when *covXetype* is a cumavg, cumavgnew, cumsum, or cumsumnew type. There are two ways in which *covXeknots* can be specified: i) by listing the location of the desired knots or category cutoffs (e.g., *covXeknots = 0.5 5 7.5 11)*, or, for (skp)spl, cumavg, cumavgnew, cumsum, and cumsumnew etypes, ii) by setting *covXeknots = 0 (or NONE), 3, 4, or 5*, where this integer value represents the number of percentiles to use when defining the knots (0 (or NONE) = no knots, 3 = 3 knots at the 10th, 50th, and 90th percentiles, 4 = 4 knots at the 5th, 25th, 75th, and 95th percentiles, and 5 = 5 knots at the 5th, 25th, 50th, 75th, and 95th percentiles). For (skp)cat, cumavgcat, cumavgcatnew, cumsumcat, and cumsumcatnew etypes, only option i) (i.e., a list of the location of the desired category cutoffs) can be used. Note that, if the values of at least 2 of these percentiles are the same, then the macro will produce division by 0 errors when calculating the spline variables. In this case, the user must either define the exact locations of the spline knots (rather than percentiles) or consider alternative functional forms for that covX. Note also that for *covXetype = cumavgnew or cumsumnew* and when covXknots is specified with a value other than 0, spline terms will be created only for the cumulative variable of covX (not for the lagged variables of *covX*). See Table 3 for more details.

covXskip (required for skp- type variables)
Values of *time* for which *covX* was not measured for any subject. For example if *covX* was not measured for any subject at *time*=0, *time*=2 and *time*=5, *covX* was carried forward from actual measurement times at those times and we selected *covXptype* as a skp- type then we would define *covXskip*= 0 2 5. See Table 2 for more details.

covXinteract (optional)

Product terms (interactions) to include in the covariate model. Syntax is as in *outcinteract*. Care should be taken when using interactions since *covYptype* will be used when creating the interaction terms when covY is included as an element of covXinteract (Y may or may not be equal to X, interaction terms are created with the first lagged values, regardless of where the factors appear in the list of modeled covariates).

covXclass (optional for covXotype=4)
Specifies the variable in the data that contains the lagged value of *var* that will be used to split the model for *z_var* (see section 5) into two parts based on covXclass = 0 and covXclass ne 0.

covXwherem (optional)
Specifies a condition under which to fit any regression procedures specified by the choice of *covXotype*. This is specified using standard SAS syntax, referring to the variables in the original dataset by their macro parameter names (e.g. &cov1), with the condition placed in parentheses. The conditions for modeling a covariate must depend only on "earlier"-indexed covariates if the condition depends on the values of other covariates at the same time point. For more information, see FAQ 4 for an example.

covXwherenosim (optional default = (1=0))
Indicates a condition under which not to simulate a value under an intervention but to assign a fixed value.

covXnosimelse (optional)
User defined macro used to assign a value to *covX* when the *covXwherenosim* condition holds. Caution is needed when the *covX* includes a random visit process (i.e. covXrandomvisitp is not left blank). In this case, when the *covXwherenosim* condition holds, variables assigned to *covXrandomvisitp* will also need to be assigned values in this user-defined macro . For more information, see FAQ 4.

covXrandomvisitp (optional)
Time-varying variable corresponding to an indicator that covX is measured in interval k. On lines where this variable is set to 0, covX should be carried forward from the last time in the input data set *data*. When this option is used, this time-varying indicator of "visit process" for covX is assumed an additional time-varying confounder and covX corresponds to a "last measured value" (Hernan et al., 2008). Defining this option will also change the way covX is modeled and simulated to minimize reliance on parametric model assumptions where deterministic knowledge of the data structure can be used (Young et al., 2011). Specifically, in addition to regression models specified by the choice of covXotype, a logistic regression model where *covXrandomvisitp* is dependent variable is fit using the same independent variables as in regression models specified by *covXotype*. In addition, regression models where covX is dependent variable specified by the choice of *covXotype* are restricted to records with *covXrandomvisitp*=1. In the simulation step, the visti/measurement indicator for covX is first simulated according to the estimated logistic regression parameters. When this simulated indicator is 1, covX is generated from the model parameters determined by covXoptype. When this indicator is 0, covX is carried forward from the last simulated value. See FAQ 8.

covXvisitpmaxgap (optional, default = 9e10)
Fixed value for the maximum number of periods allowed between visits before a subject will be censored based on the visit indicator covXrandomvisitp. If there is no limit (i.e. subjects will not be censored for too many missed measurements), then this must be set to a large number (larger than *timepoints*). When this value is less than timepoints, it is expected that a subject will be censored on a line k such that the previous *covXviistpmaxgap* lines all have covXrandomvisitp=0 for that subject. As an additional way to avoid over reliance on parametric models and instead rely on knowledge of the data structure, covXviistipmaxgap is used as a condition under which to model *covXrandomvisitp* process and is also used in the simulation of the random visit

process by imposing the condition that if the time-since-last-visit lagged by one period is equal to *covXvisitpmaxgap*, then the simulation will force the simulated value of *covXrandomvisitp* to be 1. Additional details are given in FAQ 8.

covXvisitpwherem (optional, default = (1=1))
Condition under which to model *covXrandomvisitp*, and is specified using standard SAS syntax, referring to the variables in the original dataset by their parameter names (e.g. &cov1), with the condition placed in (). The conditions for modeling a covariate must depend only on "earlier"-indexed covariates if the condition depends on the values of other covariates at the same time point. Default for the condition is 1=1. See FAQ 8.

covXvisitpcount (optional, required when using a visit process)
Fixed value to assign to the lagged time-since-last-visit variable at baseline. Used to initialize the time-since-last-covX counter at baseline.

covXaddvars (optional)
List of any additional variables to include as predictors of covX which are not included in *fixedcov* or as a time-varying covariate. The variables should appear in the dataset and could be functions of time (see timefuncgen), a time fixed or time-varying variable. This option should be used in combination with covXsaddvars. See FAQ 5 for an example.

covXgenmacro (optional)
User defined macro used for creating additional variables used the covariate and outcome models listed in covXaddvars or eventaddvars, etc. The code should be the same as how the variables were created prior to calling the GFORMULA macro.

wherevars (optional)
List of any variables referenced in *cov1wherem – covpwherem* (*time* is not required in this list even if referenced).

keepsimuldata (optional)
List of variables not created by any specified *covXptype* or *covXotype* that will be needed in the generation of the simulated data set.

equalitiessimuldata (optional)
User defined macro that equates pre baseline simulated variables to observed variables.

intervname (optional)
Specifies the name of a data set to hold the results from the INTERV macro for each intervention when running the bootstraps in parts. This must be set to a permanent data set so that the results can be put back together for each part. If *sample_start* is set to 0 and *sample_end* is set to a value other than *nsamples* then *intervname* must be defined.

numint (optional, default = 0)
Number of interventions. The default is 0 which produces results only for the natural course.

refint (optional, default = 0)
Intervention to be used as the reference level when calculating risk ratios and risk differences. The default is the natural course (*refint*=0).

sim_trunc (optional, default = 1)
When simulating variables of otype=3, specifies whether values should be truncated to be within the observed bound (*sim_trunc =1)*. When *sim_trunc =1*, if the simulated value of covX is outside of the observed interval [min(covX), max(covX)] then it will be truncated so it will be equal to min(covX) or max(covX), whichever is the closest bound. Otherwise, no truncation is performed (*sim_trunc = 0*).

usebetadata (optional default = 0)
Specifies whether regression procedures should be fit (*usebetadata*=0) or a saved data set with the estimated coefficients from a previous run stored in the data set *betadata* should be used (*usebetadata*=1).

betadata (optional)
Specifies a dataset to store the coefficient estimates. If *usebetadata*=1 then *betadata* must be defined as a permanent data set created on a previous run.

resultsdata (optional)
Specifies a dataset to store the results.

simuldata (optional)
Specifies a dataset to store the simulated dataset under the natural course.

survdata (optional, default = work._survdata_all)
Specifies the dataset to hold the estimated cumulative survival probabilities at each time point under each interventions. When not running the bootstraps in parts, this should include the libname for where to keep the data. The default will be the work space. When running the bootstraps in parts with *outctype* = binsurv this variable needs to be defined to calculate the restricted mean survival time. See FAQ 6 for more details of how to run the bootstraps in parts.

check_cov_models (optional, default = 0)
Specifies if the program will save the means of the observed and generated/simulated covariates under the natural course.

covmeandata (optional, default=work._covmean_all)
Specifies the name of the dataset to hold the differences between the observed and simulated means of each covariate if *check_cov_models=1*.

print_cov_means (optional, default = 0)
Specifies if the program should print out the comparison of the observed and generated variables if *check_cov_models=1*.

save_raw_covmean (optional, default = 0, default = _covmean_all_raw)
Specifies if the program should save the results of the comparisons for each sample and each time point if *check_cov_models=1*.

observed_surv (optional)
Dataset to hold the observed survival probabilities at each time point under the natural course. When not running the bootstraps in parts, this should include the libname for where to keep the data. The default will be the work space. See FAQ 6 for more details of how to run the bootstraps in parts.

print_stats (optional, default = 1)
Indicates whether to print basic proc means statistics for the generated variables for each timepoint, k, in the base sample.

outputs (optional, default = yes)
Option to suppress the model output for the base sample. Model results for all bootstrap samples are suppressed by default.

seed (optional, default = 7834)
Random numbers seed.

nsamples (optional, default = 50)
Specifies the number of bootstrap samples for construction of confidence intervals and estimated standard errors.

sample_start (optional, default = 0)
Specifies the starting sample when running bootstraps in parts. See the Appendix for additional information on running the bootstraps in parts.

sample_end (optional, default = -1)
Specifies the ending sample when running bootstraps in parts. When *sample_end* is set to be -1, it will be replaced with the value of nsamples. See the Appendix for additional information on running the bootstraps in parts.

nparam (optional, default = sample size)
Specifies the sample size drawn for estimating model parameters in each bootstrap sample (less than or equal to the sample size of the dataset *data*).

nsimul (optional, default = sample size)
Specifies the sample size of the Monte-Carlo simulation.

hazardratio (optional, default = 0, when compevent is left empty)
When *outctype* = binsurv, fits a Cox model and estimates the hazard ratio of the outcome for two interventions listed in *intcomp*.
intcomp (optional)
When *hazardratio*=1, *intcomp* is an ordered list of two interventions, int1 int2, where the reference level will be taken to be int1. The natural course can be used by setting int1 = 0.

bootstrap_hazard (optional, default = 0)
When running bootstraps and *hazardratio* = 1, the program will also calculate the bootstrap confidence interval for the hazard ratio. (Warning: with large data this will increase the time required to run the program.)

hazardname (optional)
When running the program in parts and *bootstrap_hazard* = 1, *hazardname* is the name of data set to hold the temporary hazard ratio for each iteration of the program. This needs to be the same name in each call to the macro. The data sets will be stored in the directory indicated in *savelib*.

savelib (optional, default = work)
Library to hold the intermediate results and data sets that are to be saved by the macro. Currently the *savelib* variable is only used when running bootstraps in parts (chunks), or when running the macro with the testing option. See the Appendix.

rungraphs (optional, default = 0)
Indicates whether the macro should also generate graphs for comparing the simulated (generated) data under the natural course and the observed data. Running the macro with *rungraphs* = 1 will also cause the GFORMULA macro to save more intermediate results. If the user does not set the variables *check_cov_models, covmeandata, observed_surv,* and *survdata* then the macro will set them as follows : check_cov_models = 1, covmeandata=_covmean_ , observed_surv=_obssurv_, and survdata=_survdata_. To redo the graphs later the parameters s*urvdata, observed_surv and covmeandata* need to be set to permanent data sets that include the libname in the name. When setting *rungraphs* = 1 the macro will save more data than during a typical run.

title1,title2, title3 (optional, default=)
Used in generating the three-line title on the first page of the graphs obtained when setting *rungraphs* = 1.

titledata (optional, default=)
Data set that contains the label for each covariate comparison that will be generated when setting *rungraphs* = 1. The data set should have one observation with a character string for each modeled variable (other than the *time* variable) which has the value of what the desired label should be. The macro will generate a default label of "Mean covX" (where *covX* is the variable listed in the *covX* macro variable.)

tsize (optional, default = 1)
Relative size of the titles used in generating the graphs when setting rungraphs = 1. Depending on the length of the titles that are being used, this value can be modified to alter the size of the titles so that they will fit on the page. This should typically not be larger than around 1.5.

graphfile (optional, default=gfile.pdf)
File which will hold the generated graphs when running the macro with *rungraphs*=1. This should be a pdf file.

runnc (optional, default = 1)
Indicates whether the macro should run the natural course intervention. When *runnc* = 0 and *numint* = 0, the GFORMULA macro will only calculate the model parameter estimates. This might be useful for running basic diagnostics of the covariate models. When *runnc* = 0, *numint* > 0 and *refint* ^= 0 the GFORMULA macro will skip over simulating the natural course intervention. This could be useful when the user is only interested in a single intervention (for example, *runnc* = 0, *numint* = 1 and *refint* = 1 will generate a single intervention))

weight (optional, default =)
A weight variable to be included in all of the covariate models. The value of the variable should be non-negative and can have non-integer values. See the SAS documentation on the use of this variable. This variable will be used only in the estimation of the regression models but not in the generation of the simulated data sets.

printlogstats (optional, default = 1)
Indicator of whether or not to include diagnostic output in the log file, including seeds , number intervened on for each intervention and bootstrap, number of times variables generated outside bounds, etc. This may be useful when calling the gformula macro from another macro.

minimalistic (optional = no)
Indicator of whether or not to reduce the number of variables that are saved when creating the data sets under each intervention. When set to yes, this will set the following macro variables: rungraphs = 0 , print_cov_means = 0, and check_cov_models = 0. In addition, this will reduce the number of variables that are saved when running the Monte-Carlo simulation part of the macro. Only those variables that are needed for the final table will be saved. Generally, these are variables based on the outcome.

testing (optional, default= no)
Indicates whether or not to run the macro in a testing mode. When equal to yes, only one sample will be run. In this case the macro will set *nsamples* = 0 and *mimimalistic* = no. When more than the natural course is run each simulated data set based on the intervention will be saved with all the desired variables for each observation. The results for each intervention will be saved in a data set named simuldata&intcount, where intcount ranges from 0 to &numint. These data sets will be saved in the library listed in the savelib variable. This will essentially double the time needed to run the macro with a single sample. *Simuldata* must be set to a name for this option to work.

Frequently asked questions

1) How to include a function of time for a time-varying covariate model?

Sometimes we might want to include different functions of time for different covariates. For example, when patients enrolled in a study have a visit typically every fixed amount of time (say, every 6 months), including a period function of time might be helpful to better predict and capture the visit process. In this case, we will use the following set of parameters*: timefuncgen, covXaddvard* and *covXsaddvars*.
For example, if two new variables mytime1 and mytime2 are to be included in the model (and simulations) for covariate *cov3*, the user will need to supply a macro before the GFORMULA macro call to define the simulated values of mytime1 and mytime2, as follows:

```
 %macro create_mytime ;
    mytime1 = func1(&time) ;
                      /* for example mytime1 = sin(&time) and mytime2 = cosin(&time) */
    mytime2 = func2(&time) ;
%mend ;
```

where func1 and func2 are two functions. The variables mytime1 and mytime2 must be defined in the input dataset. Then, in the g-formula call the user would set *timefuncgen = create_mytime*. Finally, the user should specify the options *cov3addvars=mytime1 mytime2*, and use the parameter *cov3saddvars=new_time* to supply a macro in *cov3saddvars=new_time* listing the two variables, also to appear before the GFORMULA macro call:

```
%macro new_time ;
          smytime1 smytime2
%mend ;
```

2) How to include a product ("interaction") term between two covariates in the outcome model?

We will use the parameter *outcinteract*. Say we defined *outctype=binsurv, fixedcov=sex race* and *cov4=CD4*. If we wished to include a product term between race and the most recent value of CD4 cell count in the regression model for the hazard of the event in any interval k, we would then define *outcinteract=b2*4*. If we wished to include a product between sex and interval k we would define *outcinteract=b1*0*. If we wished to include both product terms we would define *outcinteract=b2*4 bl*0*.

The above considerations also apply to models for a competing event or time-varying covariate.

3) How to stratify the outcome model by a discrete covariate?
To stratify the outcome model by a discrete covariate, we can add product terms between this covariate and all other covariates in the outcome model using the parameter *outcinteract*. For example, say we defined *outctype=binsurv, fixedcov=sex race*, *cov1=CD4, cov2 = RNA, cov3 = AIDS,* and *cov4 = ART.* To stratify the outcome model by race, we would define outcinteract = *b2*b1 b2*0 b2*1 b2*3 b2*4*.

The above considerations also apply to models for a competing event or time-varying covariate.

4) How to, in general, incorporate deterministic knowledge of the data in place of parametric model assumptions?
Sometimes we have information about relationships between time-varying covariates that can be used in place of arbitrary parametric model assumptions to avoid extrapolation where unnecessary. For example, consider the case where we have two time-varying covariates corresponding to indicators of whether a subject has had a myocardial infarction (MI) by a given time and whether a subject has had coronary artery bypass surgery (CABG). Suppose further that, as expected, we see in the data that subjects who have MI=0 at a given time k also have CABG=0 (as subjects who have not had MI by that time would not have the CABG procedure by that time as the procedure is not indicated). In this case, we would like to avoid fitting a model for the probability of CABG amongst records with MI=0 because we know this probability is zero. Further in the simulation, we would like to incorporate the assumption that this deterministic relationship would not change under any treatment intervention; i.e. we want to generate CABG in the simulation according to the estimated regression model coefficients when we generate MI=1 but otherwise, when we generate MI=0, we want to set CABG to 0 with probability 1.

We can use the following parameters to incorporate a priori knowledge of deterministic relationships between covariates: *covXwherem, covXwherenosim, and covXnosimelsemacro.*
Returning to our example, for covX=CABG, we can set covXwherem=(MI=1) which tells the algorithm to only fit a model for the probability of CABG using histories consistent with MI=1. We can then set covXwherenosim=(MI=0) which tells the algorithm to not simulate based on the saved regression coefficients when the condition MI=0 holds. Finally we set covXnosimelse to the name of a user-defined macro (e.g. covXnosimelse=mymacro) where mymacro contains the code with the rule of how to assign covX when the condition covXwherenosim holds, here:

```
%macro mymacro; * Force the simulated value of CABG=0 when current value of MI=0;
        CABG=0;
%end;
```

…
The macro *mymacro* should appear in the program before the GFOMULA macro call. Note the above would require that MI is listed prior to CABG in the ordering of time-varying covariates, e.g. if cov4=CABG then MI must be assigned to either cov1, cov2 or cov3.

5) **How to include an additional baseline or time-varying covariate in the models?**

We will need to use the parameters *covXaddvar and keepsimuldata*. For example, assume that the baseline covariates sex and baseage (age at baseline) are not listed in *fixedcov*, but we want them to appear in the model for the time-varying covariate cov1=hbp, blood pressure to improve model fit. In the G-FORMULA call we use the following syntax for cov1:

```
…
cov1=hbp,cov1addvars=sex baseage,
keepsimuldata=sex baseage,
…
```

6) I am running the g-formula procedure with a large number of bootstrap samples. Is there any way to speed up the process?

The GFORMULA macro allows running the bootstrap samples in parts where all the intermediate results must be saved to permanent datasets so that they can be put back together to obtain the final results. Running the bootstraps in parts will produce the same results as running all samples in a single run, but it might considerably speed up the process. The total number of bootstrap samples can then be divided into smaller parts. The program Example3.sas provided with the documentation illustrates the code.

7) How to model covariates that are not measured in every interval in interval-cohort studies?

In interval cohort studies, covariate measurements are taken at the end of regularly spaced intervals, usually of relatively great length. In such cohorts, some time-varying covariates may not be measured in every interval by design. For example, in the Nurses' Health Study (NHS), covariate data are collected every two years by return of a mailed questionnaire. Questions on physical activity were not asked in the questionnaire during some of the two-year intervals. In the construction of the NHS, in these "skipped" intervals, all subjects' activity measurements are carried forward from the last interval in which it was measured (the validity of this choice of data construction, of course, requires untestable assumptions). For any skipped interval k, we know, that, for all subjects (i.e. all covariate histories), the physical activity measurement in the data at k is equivalent to that in interval k-1 (by the choice of data construction). We can explicitly account for this information on the data structure in the g-formula call. If the user specifies skipped times (intervals) for covX in *covXskip*, then records corresponding to these times are automatically left out of the model fitting procedure specified by covXotype in the parametric model estimation. In simulation step, values of covX are carried forward at skipped times in generating the intervention covariate histories. The *intvisittype* option allows users to decide whether to carry forward (i) the previous value of a covariate under the intervention (i.e., the last intervened on value) or (ii) the previous natural course/simulated value of a covariate had the intervention been discontinued right before the last measurement time. For these variables, one might also choose to use a *covXptype* or *covXetype* with the prefix skp-. See Tables 2 and 3.

8) How to model random visit/measurement processes in clinical cohorts?

We will use the parameters: *covXrandomvisitp, covXvisitpcount,*and *covXvisitpmaxgap*. In clinical cohorts, the data are not usually recorded in 'waves' or 'sweeps', but are recorded everytime there is a visit at the clinic. Therefore, times at which the time-varying covariates and outcome are measured will vary by subject. It is typical to construct the data such that (i) the last measured value of a covariate covX is carried forward at a time where there is no visit/measurement (*covXrandomvisitp*=0) and (ii) a subject is censored when he/she is not seen for a specified consecutive number of time intervals (*covXvisitpcount)*. There are implications for both confounding control and parametric modelling in such settings. First, following Hernán et al. (2009), under such data structures, the visit process history should be considered part of the confounder history. Here causal inference relies on the assumption that given the most recent measurement of covX along with its measurement history, we can control confounding and selection bias by unmeasured risk factors. In such settings one may choose to censor subjects after a certain number of intervals without a new measurement of covX. For example, the assumption that the last measured value of covX is sufficient to control unmeasured confounding might be considered reasonable after 3 months without a visit but not after 12 months without a visit. In this case, we must additionally rely on the assumption that measured variables are sufficient to control selection bias due to this source of censoring.

Following Young et al., 2011, in these settings where (i) last measured values are carried forward at times of no new measurements (ii) the visit/measurement process is itself defined as a time-varying confounder and (iii) we censor after a certain number of consecutive times with no visit/measurement, there are certain deterministic relationships in the data that can be used in place of arbitrary parametric assumptions in both step 1 and step 2 of the parametric g-formula estimation algorithm. First, we know that if the visit indicator (*covXrandomvisitp)* is 0, covX is always equal to its lagged value. We also know that the number of consecutive records for a given subject where *covXrandomvisitp*=0 cannot exceed *covXvisitpcount-1* (as we will censor after that).

For example, in clinical cohorts of HIV-positive patients, CD4 count, a prognostic factor, is measured at every visit. Assume that visit_cd4 is a an indicator variable in the original dataset for whether an actual measurement of CD4 count was taken at time k. Assume also that we choose to censor subjects once they have not had CD4 measured for 12 consecutive intervals. By this, we can use the following syntax for *cov1* in the g-formula call:
…
cov1=cd4,   cov1otype=3, cov1ptype=lag1bin,
cov1randomvisitp = visit_cd4, cov1visitpmaxgap = 12,
…

By these specifications, a logistic regression model will be fit *for visit_cd4* in addition to a regression model for cov1 determined by *cov1otype* using the same set of independent variables. The model for cov1 (ie cd4 count) is then also automatically restricted to records where the parameter in *cov1randomvisitp (*ie visit_cd4) =1. In the data generation step, cd4 values are then carried forward in place of being generated from the estimated regression coefficients when the generated value of visit_cd4=0. Because we specified cov1visitmaxgap=12, the logistic regression model for visit_cd4 is fit only to records with time since the last visit at the previous time less than 12. This is because if this time is 12, we know with certainty that the value of visit_cd4 must be 1 (because subjects with 0 are censored by our definition of censoring).Analogous In the data generation step 2, when visit_cd4 is generated as 0 cd4 values is carried forward in place of being generated from the fitted model. Further c*ovXrandomvisitp* (visit_cd4) will be set to one in step 2 when the previous simulated value of visit_cd4 is the *covXvisitmaxgapth* (e.g. (12th) consecutive simulated 0.

Finally, when *covXrandomvisitp* is not left empty, the macro will automatically generate a variable encoding the time since covX was last measured. Either the current value or the last lag of this time since last measurement (depending on whether a preceding or succeeding covariate model — see Section 5) will be included as independent variables for all regression models. When each subject has *covXrandomvisitp* at baseline, the lagged time-since-last-visit variable can be defined to be 0. When there are subjects who do not have a visit at baseline the variable needs to be defined in such a way so that the supplied value plus 1 will be equal to the time-since- last-visit at baseline. The user does not need to supply the complete history for this variable.

Appendix

Structure of the GFORMULA macro

The GFORMULA macro is comprised of several nested macros, of which only the GFORMULA main macro and the INTERV macro require parameters.

The structure of the nested macros is outlined below

```
GFORMULA (requires parameters)
        DATAPREP
        do 0 to nsamples SAMPLES
                PARAMETERS do 0
                to numint
                        INTERV (requires parameters)
                        end
                end
           RESULTS
```

The DATAPREP macro does the preparatory generation of macro variables for later use and preparatory manipulation of the input person-time data set.

The SAMPLES macro does the bootstrap sampling of the data for nsamples bootstrap samples with sample 0 the original input data set.

The PARAMETERS macro fits regression models to implement step 1 of the algorithm. Parameter estimates are saved for use in later Monte Carlo simulations.

The INTERV macro generates the covariate histories under the intervention and calculates the final intervention risk/mean estimates (steps 2 and 3 of the algorithm). This macro is called numint times corresponding to the number of interventions specified (not including the natural course).

The RESULTS macro takes the results of the INTERV macro to create program output including the intervention mean estimates and relative effect measures, along with bootstrapped
95% confidence intervals.

The macro runs many SAS procedures, so it is often important to specify the SAS “nonotes” option.

Examples of user-defined interventions

If the user wants to define their own intervention rule for intervention variable # (which, as above can take values from 1 to 8) for a given intervention, they can select inttype#=-1 and provide a user-defined macro coding the desired intervention rule. This user-defined macro is called within a loop, which is within a SAS DATA step. The loop index is the user supplied time variable assigned to macro parameter *time*. The loop begins at time=0 and continues through *timepoints*-1.

We consider an example of specifying a user-defined intervention of the following form considered by Young et al. (2011): "Start antiretroviral treatment (art) when CD4 cell count first drops below x or diagnosis of AIDS, whichever comes first". These authors were interested in estimating the 60 month causal risk ratio comparing these interventions for different choices of CD4 cutoff x (e.g. x=500 versus x=200). Immediately below is a user-defined macro named dynamiccd4 which codes this intervention for an input data set where *month* encodes the follow-up time index (assigned to macro parameter time in the main *gformula* call), and *art* and *lncd4* are both time-varying covariates in the data set representing a a time-varying indicator of treatment initiation by a given month and lncd4 is the most recent measurement of log CD4 cell count relative to a given month. The macro *dynamiccd4* itself has a macro parameter named *intcut* which specifies the cutoff x defining the intervention.

```
/* this is the user-defined macro*/
/* it is called within a loop month=0 to timepoints-1*/
%macro dynamiccd4(intcut=);
   /*assess whether condition for treatment by time k is met*/
         if month=0 then do;
                   dyncond_l1=0;
         end;

         if dyncond_l1=1 then do;
                   dyncond=1;
         end;

         else if dyncond_l1=0 then do;
                   if lncd4 < log(&intcut) or aids = 1 then do;
                                 dyncond  = 1;
                   end;
                   else do;
                             dyncond = 0;
                   end;
   end;
                                                          dyncond_l1=dyncond;

         /*if condition is not met do not yet start treatment*/
   if dyncond = 0 then do;
                   art = 0;
   end;

         /*if condition has been met need to have started treatment*/
   if dyncond = 1 then do;
                   art = 1;
    end;

   /*this array stores time-varying treatment values */
         /*under intervention, will affect output*/
         /*the array will have the name of the intervention variable*/
                                                             /*preceded by the letter s */
                        sart[month]=art;
```

```
%mend;
```

This intervention can then be called for x=500 and x=200 with the following syntax, placed above the main call to the gformula macro

```
%let interv1 =
intno=1,
intlabel='start cART when cd4 first drops below 500', nintvar=1,
intvar1=art, inttype1=-1,
inttimes1= 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
    31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59,
intusermacro1=dynamiccd4(intcut=500) ;

%let interv2 =
intno=2,
intlabel='start cART when cd4 first drops below 200', nintvar=1,
intvar1=art, inttype1=-1,
inttimes1= 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
    31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59,
intusermacro1=dynamiccd4(intcut=200);
```

Here is a subsequent call to the gformula macro which could be used to estimate the 60 month causal risk ratio of interest (note that the variables art and lncd4 must both be listed as time-varying covariates):

```
%gformula(
data=hivdata,
id=id,
time=month,
timeptype = conbin ,
timepoints=60,
outc=death,
numint=  3,
refint=2,
fixedcov= rna_0 cd4_0 age_0  yrshiv,

ncov=4,
cov1=lncd4,     cov1otype=3, cov1ptype = lag1bin ,
cov2 = lnrna,  cov2otype=3, cov2ptype=lag1bin,
cov3=aids, cov3otype=2, cov3ptype=lag1bin,
cov4=art, cov4otype=2, cov4ptype=lag1bin,
nsamples= 0
);
```

One problem with the coding of dynamiccd4 above is that it does not track the number of times that a simulated value of art was inconsistent with interventions. Such tracking is included in pre-specified interventions included with the macro (e.g. inttype#=1). Without such tracking, parts of the program output including statistics on “percent intervened on ” and “average percent intervened on” which quantify the deviation between simulated and intervention consistent values will not be accurate. If the user would like these statistics accurately computed then the following updated version of dynamiccd4 --dynamiccd4alt--can be used which will give an equivalent estimate of the risk ratio but will additionally compute these trackers of percent intervened on.

```
/* this is the user-defined macro*/
/* it is called within a loop month=0 to timepoints-1*/
%macro dynamiccd4alt(intcut=) ;

        /* assess whether condition for treatment by time k is met*/
          if month=0 then do;
                    dyncond_l1=0;
        end;

        if dyncond_l1=1 then do;
                    dyncond=1;
        end;

        else if dyncond_l1=0 then do;
                    if lncd4 < log(&intcut) or aids = 1 then do;
                    dyncond  = 1;
                 end;
                 else do;
                    dyncond = 0;
                 end;
         end;

         dyncond_l1=dyncond;

/* if condition is not met and simulated value of art is 1, intervene and change to 0, trackers are also included*/
        if dyncond = 0 and art =1 then do;
                            /* this art is the simulated value before intervention*/
          art = 0;
                            /* this is changing the value of art to be in line with intervention rule*/
                       art_totinterv = art_totinterv + 1;
           intervened = 1;
           intervenedk[month] = 1;
           totinterv = totinterv + 1;
          end;

/* if condition has been met need to have started treatment but simulated value of art is 0 then intervene and change 1, trackers
are also included*/
        if dyncond = 1 and art = 0 then do; /* this art is simulated value before intervention*/
          art = 1; /* this is changing the value of art to be in line with intervention*/
                    art_totinterv = art_totinterv + 1;
                intervened = 1;
                intervenedk[month] = 1;
                totinterv = totinterv + 1;
          end;

    /* this array stores the final  time-varying treatment values */
    /* under intervention, will affect output*/
          /* the array will have the name of the intervention variable*/
          /* preceded by the letter s */
    sart[month]=art;

%mend;
```

Note that the tracker “art_totinterv” is named by the name of the intervention variable in the data set. This would be analogously modified for a different intervention variable. E.g. if the variable to be intervened on was named “cig” then this tracker would be renamed “cig_totinterv”.

Finally, Young et al (2011) additionally considered a slightly more complex version of the interventions above indexed by x that further allow for a grace period before requiring that treatment start once the dynamic condition to initiate treatment is met. These can be generally stated as follows: “Start antiretroviral treatment (art) within m months of CD4 cell count first dropping below x or diagnosis of AIDS, whichever comes first”. The user-defined macro below called “cutgrace” can be used to define these interventions. This has two macro parameters: intcut defining the cutoff x and intgrace defining the grace period m. Here we illustrate an alternative coding that utilizes an array called “dyncond” which holds the time-varying indicator of whether the dynamic condition to start treatment has been met by each time (from 0 to 59 in this case because end of follow-up is 60 months). This array can be defined in another user-defined macro which here we call setupdyncond. This macro will be referenced below by the macro parameter “*intsetup*”. *Intsetup* can generally be used to define any array that might be needed for the user-defined intervention. The user-defined macro assigned to *intsetup* will be called just prior to the execution of the *time* loop.

```
%macro setupdyncond ;
   array dyncond{0:59} ;
%mend;

%macro cutgrace(intcut=,intgrace=) ;

      tgrace = month - &intgrace;

      if   lncd4 < log(&intcut) or aids = 1 then do;
               dyncond[month] = 1;
      end;
      else do;
                  dyncond[month] = 0;
      end;
      if month > 0 & dyncond[month - 1]  = 1  then dyncond[month] = 1;

      if dyncond[month] = 0 and art = 1 then do;
         art = 0;
       art_totinterv = art_totinterv + 1;
                              intervened = 1;
                              intervenedk[month] = 1;
                              totinterv = totinterv + 1;
      end;

     if month >=  &intgrace then do;
         /* note if month<&intgrace and dynamic condition has been met, we are still by definition in the grace period so no
         intervention will take place even if treatment hasn’t started*/
              if dyncond[tgrace] = 1 and art = 0 then do;
         /* if dynamic condition to start treatment was met m months ago, you now have to start if you haven’t started*/
                 art = 1 ;
                 art_totinterv = art_totinterv + 1;
                              intervened = 1;
                              intervenedk[month] = 1;
                              totinterv = totinterv + 1;
              end;
         end;

       sart[month]=art;

%mend;
```

This intervention that allows a grace period of m=6 can then be called for x=500 and x=200 with the following modified syntax, again placed above the main call to the gformula macro

```
%let interv1 =
intno=1,
intlabel='x=500, grace period = 6 months', nintvar=1,
intvar1=art, inttype1=-1,
inttimes1= 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
     31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59,
intusermacro1=cutgrace(intcut=500,intgrace=6), intsetup=setupdyncond;

%let interv2 =
intno=2,
intlabel='x=200, grace period = 6 months', nintvar=1,
intvar1=art, inttype1=-1,
inttimes1= 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
     31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59,
intusermacro1=cutgrace(intcut=200,intgrace=6), intsetup=setupdyncond;
```

References

Danaei G, Pan A, Hu FB, Hernán MA. Hypothetical lifestyle interventions in middle-aged women and risk of type 2 diabetes: a 24-year prospective study. *Epidemiology* 2013; 24(1):122-128.

Garcia-Aymerich J, Varraso R, Danaei G, Camargo CA, Hernán MA. Incidence of adult-onset asthma after hypothetical interventions on body mass index and physical activity. An application of the parametric g-formula. *American Journal of Epidemiology* 2014; 179(1):20-26.

Gooley TA, Lessening W, Crowley J, Storer BE. Estimation of failure probabilities in the presence of competing risks: new representations of old estimators. *Statistics in Medicine* 1999. 18(6):695-706.

Kalbfleisch JD, Prentice RL. The Statistical Analysis of Failure Time Data, John Wiley, New York, 1980.

Hernán MA, McAdams M, McGrath N, Lanoy E, Costagliola D. Observation plans in longitudinal studies with time-varying treatments. Statistical Methods in Medical Research 2009;18(1):27-52.

Hernán MA, Robins JM. (2020). Causal Inference: What If (Chapman & Hall/CRC).

Lajous M, Willett WC, Robins JM, Young JG, Rimm E, Mozaffarian D, Hernán MA. Changes in fish consumption in midlife and the risk of coronary heart disease in men and women. American Journal of *Epidemiology* 2013; 178(3):382-91.

Lodi S, Phillips A, Logan R, Olson A, Costagliola D, Abgrall S, van Sighem A, Reiss P, Miró JM, Ferrer E, Justice A, Gandhi N, Bucher HC, Furrer H, Moreno S, Monge S, Touloumi G, Pantazis N, Sterne J, Young JG, Meyer L, Seng R, Dabis F, Vandehende MA, Pérez-Hoyos S, Jarrín I, Jose S, Sabin C, Hernán MA; HIV-CAUSAL Collaboration. Comparative effectiveness of strategies for antiretroviral treatment initiation in HIV-positive individuals in high-income countries: immediate universal treatment versus CD4-based initiation. *Lancet HIV* 2015; 2(8):e335-343.

Lodi S, Costagliola D, Sabin C, Amo JD, Logan R, Abgrall S, Reiss P, van Sighem A, Jose S, Blanco JR, Hernando V, Bucher HC, Kovari H, Segura F, Ambrosioni J, Gogos CA, Pantazis N, Dabis F, Vandenhende MA, Meyer L, Seng R, Gill J, Krentz H, Phillips A, Porter K, Grinsztejn B, Pacheco AG, Muga R, Tate J, Justice A, Hernán MA. Effect of immediate initiation of antiretroviral treatment in HIV-positive individuals aged 50 years or older. *Journal of Acquired Immune Deficiency Syndromes* 2017 (in press).

Robins JM. A new approach to causal inference in mortality studies with a sustained exposure period: application to the healthy worker survivor effect. Mathematical Modelling, 7:1393–1512, 1986. [Errata (1987) in Computers and Mathematics with Applications 14, 917-921. Addendum (1987) in Computers and Mathematics with Applications 14, 923-945. Errata (1987) to addendum in *Computers and Mathematics with Applications* 18, 477.

Richardson TS and Robins JM. Single World Intervention Graphs (SWIGs): a unification of the counterfactual and graphical approaches to causality. Center for Statistics and the Social Sciences, University of Washington Series, 2013. URL http://www.csss.washington.edu/Papers/. Working Paper Number 128.

Taubman SL, Robins JM, Mittleman MA, Hernán MA. Intervening on risk factors for coronary heart disease: an application of the parametric g-formula. *Int J Epidemiol* 2009; 38(6):1599-611.

Young JG, Cain LE, Robins JM, O'Reilly E, Hernán MA. Comparative effectiveness of dynamic treatment regimes: an application of the parametric g-formula. *Statistics in Biosciences* 2011; 3:119-143.

Young JG, Hernán MA, Robins JM. Identification, estimation and approximation of risk under interventions that depend on the natural value of treatment using observational data. *Epidemiologic Method*s 2014; 3(1):1-19.

Zhang Y, Young JG, Thamer M, Hernán MA. Comparing the effectiveness of dynamic treatment strategies using electronic health records: an application of the parametric g-formula to anemia management strategies. Health Services Research 2018; 53(3):1900-1918.

Young JG, Stensrud MJ, Tchetgen Tchetgen EJ, Hernán MA. A causal framework for classical statistical estimands in failure-time settings with competing events. *Statistics in Medicine* 2020;39:1199-236.

Chiu Y-H, Lan W, McGrath S, Logan R, Dahabreh I, Hernán MA. Evaluating model specification when using the parametric g-formula in the presence of censoring. *American Journal of Epidemiology* 2023; 192(11):1887-1895.