stmgintcox — Marginal Cox PH model for interval-censored multiple-event data

Description

`stmgintcox` fits marginal Cox proportional hazards models to interval-censored multiple-event data or, more precisely, event-time data with multiple types of events per subject. Each event time is known only to occur over a random time interval. `stmgintcox` can be used with both single- or multiple-record-per-event interval-censored data. With interval-censored data, the event-time variables are specified directly using the `stmgintcox` command instead of using `stset`. All `st` settings are ignored by `stmgintcox`.

Quick start

Single-record-per-event interval-censored data

Marginal Cox proportional hazards model with common covariates `x1` and `x2` fit to interval-censored multiple-event data with subject ID variable `id`, event variable `event`, and lower and upper interval endpoints `t1` and `t2`

```
stmgintcox x1 x2, id(id) event(event) interval(t1 t2)
```

Add a time-varying covariate for all events by interacting `x2` with the logarithm of analysis time

```
stmgintcox x1 x2, id(id) event(event) interval(t1 t2) tvc(x2) texp(ln(_t))
```

Marginal Cox proportional hazards model with covariate `x1` for event labeled `"event1"`, covariates `x1` and `x2` for `"event2"`, and covariate `x3` for `"event3"`

```
stmgintcox ("event1": x1) ("event2": x1 x2) ("event3": x3), ///
  id(id) event(event) interval(t1 t2)
```

Same as above

```
stmgintcox x1 ("event2": x2) ("event3": x3, nocommon),      ///
  id(id) event(event) interval(t1 t2)
```

After estimation, report regression coefficients instead of hazard ratios

```
stmgintcox, nohr
```

After estimation, report robust standard errors but using a fixed step size instead of the default adaptive step size

```
stmgintcox, vce(robust, stepsize(fixed))
```

After estimation, save estimated baseline hazard contributions to `basehc.dta`, and store estimation results as `model1`

```
stmgintcox, saving(basehc)
estimates store model1
```

Multiple-record-per-event interval-censored data

Marginal Cox proportional hazards model with baseline (time-invariant) covariate `x1` and time-varying covariate `x2` fit to multiple-record-per-event interval-censored data with subject identifier `id`, event variable `event`, examination time `tvar`, and event status indicator `status`

```
stmgintcox x1 x2, id(id) event(event) time(tvar) status(status)
```

Same as above, but use the covariate values at the nearest examination time on the right, instead of the default nearest examination time on the left, to impute values of time-varying covariate `x2` between examination times

```
stmgintcox x1 x2, id(id) event(event) time(tvar) status(status)  ///
  tvcovimpute(nearright)
```

Menu

Statistics > Survival analysis > Regression models > Interval-censored multiple-event Cox PH model

Syntax

Single-record-per-event interval-censored data

Marginal Cox model with common covariates

```
stmgintcox indepvars [if] [in], id(idvar) event(eventvar)
           interval(t_l t_u) [single_options]
```

Marginal Cox model with event-specific (and optionally common) covariates

```
stmgintcox [indepvars]
           ([event1:] [indepvars1] [, event_options])
           ([event2:] [indepvars2] [, event_options])
           [...] [if] [in], id(idvar) event(eventvar)
           interval(t_l t_u) [single_options]
```

event# may be a value of variable *eventvar* or the corresponding label for a value as defined by the value label. If *event#* is a label, it should be enclosed in double quotes.

Each event specification must include at least one independent variable.

Multiple-record-per-event interval-censored data

Marginal Cox model with common covariates

```
stmgintcox indepvars [if] [in], id(idvar) event(eventvar)
           time(timevar) status(statusvar) [multiple_options]
```

Marginal Cox model with event-specific (and optionally common) covariates

```
stmgintcox [indepvars]
           ([event1:] [indepvars1] [, event_options])
           ([event2:] [indepvars2] [, event_options])
           [...] [if] [in], id(idvar) event(eventvar)
           time(timevar) status(statusvar) [multiple_options]
```

event# may be a value of variable *eventvar* or the corresponding label for a value as defined by the value label. If *event#* is a label, it should be enclosed in double quotes.

Each event specification must include at least one independent variable.

single_options	Description
Model	
* `id(`*idvar*`)`	specify subject ID variable
* `event(`*eventvar*`)`	specify event variable
* `interval(`t_l t_u`)`	specify lower and upper endpoints for the event-time interval
options	options for both single- and multiple-record-per-event interval-censored data

*`id()`, `event()`, and `interval()` are required for single-record-per-event interval-censored data and cannot be combined with `time()`, `status()`, or `tvcovimpute()`.

multiple_options	Description
Model	
† `id(`*idvar*`)`	specify subject ID variable
† `event(`*eventvar*`)`	specify event variable
† `time(`*timevar*`)`	specify examination time variable
† `status(`*statusvar*`)`	specify event status indicator variable
Model 2	
`tvcovimpute(`*type*`)`	specify how to impute unobserved covariate values between examination times for time-varying covariates; default is `tvcovimpute(nearleft)`
`statussysmissok`	retain the observations that contain system missing values (`.`)
options	options for both single- and multiple-record-per-event interval-censored data

†`id()`, `event()`, `time()`, and `status()` are required for multiple-record-per-event interval-censored data and cannot be combined with `interval()`.

event_options	Description
Model	
`nocommon`	exclude common covariates (if specified) from a specific event
Time varying	
`tvc(`*varlist*$_t$`)`	specify covariates to be interacted with a function of time for a specific event
`texp(`*exp*`)`	specify a function of time for a specific event; default is `texp(_t)`
EM options	
emopts	EM options for a specific event

options	Description
Model 2	
`strata(`*varlist*`)`	specify strata variables
`reduced`	estimate baseline hazard function using a reduced set of time intervals; the default
`full`	estimate baseline hazard function using all time intervals
`favoraccuracy`	favor accuracy of results over speed; the default
`favorspeed`	favor speed possibly over accuracy of results
Time varying	
`tvc(`*varlist*$_t$`)`	specify covariates to be interacted with a function of time
`texp(`*exp*`)`	specify a function of time; default is `texp(_t)`
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `robust` (the default) or `cluster` *clustvar*; may be specified on replay of results
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`saving(`*filename*`[ , replace ])`	save estimates of baseline hazard contributions to *filename*; use `replace` to overwrite existing *filename*
`nohr`	report regression coefficients, not hazard ratios
`noheader`	suppress header from coefficient table
`detail`	display the censoring information for each event
`[no]log`	display or suppress EM and VCE iteration logs; default is `log`
`dots`	display all EM and VCE iterations as dots
`[no]vcedots`	suppress VCE iteration dots or display a dot for every iteration; default is `vcedots`
`vcedots(`#`)`	display a dot every # VCE iterations; default is to display a dot every iteration, meaning `vcedots(1)`
`[no]emlog`	suppress EM iteration log or display the log-pseudolikelihood value every 100 iterations; default is `emlog`
`emlog(`#`)`	display the log-pseudolikelihood value for the EM algorithm every # iterations; default is `emlog(100)`
`[no]emdots`	suppress or display EM iteration dots; default is `noemdots`
`emdots(`#`)`	display every # EM iterations as dots; default is to display EM iteration logs, meaning `noemdots`
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
emopts	EM options for all events
`coeflegend`	display legend instead of statistics

indepvars, *indepvars*#, and *varlist*$_t$ may contain factor variables; see **[U] 11.4.3 Factor variables**.

`by`, `collect`, and `statsby` are allowed; see **[U] 11.1.10 Prefix commands**.

`coeflegend` does not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

emopts	Description
EM options	
emiterate(#)	maximum number of EM iterations; default is emiterate(5000)
emtolerance(#)	tolerance for the coefficient vector; default is emtolerance(1e-6)
emltolerance(#)	tolerance for the log pseudolikelihood; default is emltolerance(1e-7)
emhsgtolerance(#)	tolerance for the scaled gradient; default is emhsgtolerance(1e-5)
noemhsgtolerance	do not perform the scale-gradient convergence check
from(*init_specs*)	initial values for the regression coefficients

vcetype	Description
robust[, *vce_options*]	Huber/White/sandwich estimator; the default
cluster *clustvar*[, *vce_options*]	clustered sandwich estimator

vce_options	Description
stepsize(adaptive | fixed [#])	use adaptive or fixed step size to compute VCE; default is adaptive step size
derivopts	options to control computation of numerical derivatives when adaptive step size is used
iterate(#)	maximum number of iterations to compute VCE; default is iterate(5000)
tolerance(#)	profile log-pseudolikelihood tolerance to compute VCE; default is tolerance(1e-6)
[no]dots	synonym for novcedots and vcedots
dots(#)	synonym for vcedots(#)
post	replace the current e(V) with the specified VCE type; can be used only on replay

dots, dots(), nodots, and post do not appear in the dialog box.

Options

Model

id(*idvar*) is required with stmgintcox. It specifies the subject ID variable; observations with equal, nonmissing values of *idvar* are assumed to belong to the same subject. Observations for which *idvar* is missing are ignored.

event(*eventvar*) is required with stmgintcox. It specifies the event variable; observations with equal, nonmissing values of *eventvar* are assumed to belong to the same event. Observations for which *eventvar* is missing are ignored.

`interval(`t_l t_u`)` is required with single-record-per-event interval-censored data; see *Single- versus multiple-record-per-event interval-censored data formats* in *Remarks and examples*. It specifies two time variables that contain the endpoints of the event-time interval. t_l represents the lower endpoint, and t_u represents the upper endpoint. `interval()` may not be combined with the `time()`, `status()`, or `tvcovimpute()` option.

The interval time variables t_l and t_u should have the following form:

Type of observations		t_l	t_u
interval-censored	$(a, b]$	a	b
left-censored	$(0, b]$	.	b
left-censored	$(0, b]$	0	b
right-censored	$(a, +\infty)$	a	.
missing		.	.
missing		0	.

In the table, a and b satisfy $0 < a < b < \infty$. Also note that $t_l = t_u$ is not allowed with left-censored or interval-censored observations.

`time(`*timevar*`)` is required with multiple-record-per-event interval-censored data; see *Single- versus multiple-record-per-event interval-censored data formats* in *Remarks and examples*. It specifies the examination times for the event of interest and may not be combined with the `interval()` option.

`status(`*statusvar*`)` is required with multiple-record-per-event interval-censored data; see *Single- versus multiple-record-per-event interval-censored data formats* in *Remarks and examples*. It specifies a binary status indicator for the event of interest. For each examination time, *statusvar* indicates, by the value of 1 versus 0, whether the event has occurred between previous and current examination times. `status()` may not be combined with the `interval()` option.

In combination with the `statussysmissok` option, observations with system missing values (`.`) in the *statusvar* variable will be used during estimation but will not be used to determine the event-time information for the corresponding events and subjects; see the description of `statussysmissok`.

`time()` and `status()` together define the lower t_l and upper t_u endpoints of the event-time interval and the censoring types for different events in a multiple-record-per-event interval-censored dataset. If the event of interest occurs before the first examination time for a subject, the particular event for this subject is left-censored. In the first record of the event for this subject, time is the first examination time, and the event-status indicator is 1. The corresponding event-time interval of the particular event for this subject has 0 as the left lower endpoint, t_l, and the first examination time as the right upper endpoint, t_u. If the event occurs between two examination times, the event for this subject is interval-censored. The event-time interval has the last examination time where the status indicator is 0 as t_l and the first examination time where the status indicator is 1 as t_u. If the event does not occur during the study, the event for this subject is right-censored. The event-time interval has the last examination time as t_l and missing time (`.`) as t_u.

`nocommon` is used when specifying event-specific covariates and relevant only when common covariates are also specified following `stmgintcox`. When common covariates are specified, such as `stmgintcox x1 x2 ("event1":x3) ("event2":x4) ...`, these covariates (`x1` and `x2`) are included in the equation for each event. `nocommon` specifies that the common covariates be excluded from the specified event.

Model 2

`strata(`*varlist*`)` specifies the stratification variables. Observations with equal values of the strata variables are assumed to be in the same stratum. Stratified estimates (equal regression coefficients across strata but with a baseline hazard unique to each stratum) are then obtained.

`reduced`, the default, specifies that the baseline hazard function be estimated using a reduced (innermost) set of time intervals for a particular event type. This allows the estimator of the cumulative baseline hazard function to change its values only at the endpoints of the innermost time intervals of that particular event; these innermost time intervals were originally used by Turnbull (1976) to estimate the survivor function in the one-sample case. `reduced` may not be combined with the `full` option.

`full` specifies that the baseline hazard function be estimated using all observed time intervals for a particular event type. In this case, the estimator of the cumulative baseline hazard function can potentially change its values at the endpoints of all the observed time intervals of that particular event. This is the approach used by Zeng, Mao, and Lin (2016). It is more time consuming, but it may provide more accurate results. `full` may not be combined with the `reduced` option.

`favoraccuracy`, the default, and `favorspeed` control the tradeoff between accuracy of the results and the execution speed. `favoraccuracy` specifies that the command run longer to obtain more accurate results. `favorspeed` specifies that the command run faster at the possible expense of reduced accuracy of the results. You can use `favorspeed` for a quick initial exploration of the results and `favoraccuracy` for final reporting of the results.

When you specify `favorspeed`, `stmgintcox` uses less stringent convergence criteria to obtain the results. Specifically, it assumes lower expectation-maximization (EM) coefficient, pseudo-likelihood, and variance–covariance matrix of the estimators (VCE) tolerances of 0.0001 and implies the `noemhsgtolerance` option. In addition, it uses a fixed step size with a multiplier of 5 instead of an adaptive step size when computing VCE. That is, specifying `favorspeed` is equivalent to specifying `emtolerance(0.0001)`, `emltolerance(0.0001)`, `noemhsgtolerance`, and `vce(, tolerance(0.0001) stepsize(fixed))`. `favorspeed` may not be combined with `emtolerance()`, `emltolerance()`, `noemhsgtolerance`, and `vce(, tolerance() stepsize())`.

`tvcovimpute(`*type*`)` is used with multiple-record-per-event interval-censored data and relevant only to variables included in the model whose values vary over time for a particular event in the dataset. It specifies how to impute unobserved covariate values between two examination times for time-varying covariates in the dataset. *type* is one of `nearleft`, the default, `nearright`, `nearest`, or `first`. `tvcovimpute()` may not be combined with the `interval()` option.

`stmgintcox` requires covariate values for each event and each subject at all distinct analysis times, but the data typically record covariates only at event-specific and subject-specific examination times, and the covariate values at other analysis times need to be imputed. `stmgintcox` offers the following imputation methods.

`nearleft`, the default, uses covariate values at the nearest examination time on the left to impute covariate values at observation times that fall between two examination times for a given event and subject. This method is often preferred in practice because it does not use future covariate values to fill in the current values. This is also the method used with right-censored survival-time data.

`nearright` uses covariate values at the nearest examination time on the right to impute covariate values at analysis times that fall between two examination times for a given event and subject.

`nearest` uses covariate values at the nearest examination time, left or right, to impute covariate values at analysis times that fall between two examination times for a given event and subject.

For all three imputation methods above, for a particular event, if a subject is not examined at baseline (time 0), then the covariate value at the first examination time is used for all analysis times before the first examination time for this subject. And the covariate value at the last examination time is used for all observation times after the last examination time for a subject.

`first` replaces all covariate values for a subject with those at the first examination time, which is the same as using the baseline covariate values for all time-varying variables during estimation.

See *Methods and formulas* for details.

`statussysmissok` is used with multiple-record-per-event data. It specifies that, during estimation, the observations that contain system missing values (`.`) in the event status variable specified in the `status()` option be retained. These observations will not be used to determine the event-time intervals and censoring information for the corresponding subjects, but examination times and covariate values in these observations will be used during estimation. Without this option, `stmgintcox` omits observations with a system missing value in the status variable from estimation, similar to how an observation with a missing value in any of the specified variables would be omitted. Observations that contain extended missing values (`.a` through `.z`) are always omitted during estimation. `statussysmissok` is useful for creating time-varying covariates in a multiple-record-per-event format; see *Single- versus multiple-record-per-event interval-censored data formats* in *Remarks and examples*.

Time varying

`tvc(`*varlist*$_t$`)` specifies the variables to be included in the model as an interaction with a function of time to form time-varying covariates. During estimation, these variables are interacted with analysis time or with a function of analysis time specified in the `texp()` option. This is a convenience option to include time-varying covariates that are deterministic functions of time. Using this option speeds up calculations and avoids having to split the data over many analysis times. `tvc()` in conjunction with `texp()` is also useful for testing the proportional-hazards assumption; see example 3 in *Remarks and examples*.

`texp(`*exp*`)` is used in conjunction with `tvc(`*varlist*$_t$`)` to specify the function of time that should be used to multiply covariates specified in `tvc()`; using this pair of options allows you to include time-varying covariates that are deterministic functions of time. For example, specifying `texp(ln(_t))` would cause the covariates in the `tvc()` option to be multiplied by the logarithm of analysis time. If `tvc(`*varlist*$_t$`)` is used without `texp(`*exp*`)`, Stata understands that you mean `texp(_t)` and thus multiplies the covariates by the analysis time, denoted as `_t` here.

For a demonstration of how to use `tvc(`*varlist*$_t$`)` and `texp(`*exp*`)` to test the proportional-hazards assumption, see *Incorporating time-varying covariates using the tvc() option* in *Remarks and examples*. `tvc()` and `texp()` can be specified globally with the command and individually for a specific event. Options specified for a particular event will override globally specified options.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported. `vce()` may be specified at the time of estimation or when replaying results. If specified when replaying results, `vce()`-related stored results are not updated unless the `post` suboption is specified. *vcetype* may be one of the following:

`robust`[`,` *vce_options*] uses the robust or sandwich estimator of variance based on the profile log pseudolikelihood; see *Methods and formulas*. `vce(robust)` is the default.

`cluster` *clustvar* [, *vce_options*] specifies that the standard errors allow for intragroup correlation, relaxing the usual requirement that the observations be independent. That is, the observations are independent across groups (clusters) but not necessarily within groups. *clustvar* specifies to which group each observation belongs. The clustered sandwich estimator of variance is based on the profile log pseudolikelihood; see *Methods and formulas*.

vce_options may be `stepsize()`, *derivopts* with adaptive step size, `iterate(#)`, `tolerance(#)`, `dots`, `dots()`, `nodots`, and `post`.

`stepsize(adaptive | fixed [ # ])` specifies the step size for computing numerical derivatives. The default is `stepsize(adaptive)`, which uses adaptive step size in computations; see [M-5] **deriv()**. `stepsize(fixed)` uses a fixed step size equal to $\delta_n = 5n^{-1/2}$, where n is the number of subjects or clusters. `stepsize(fixed #)` uses a fixed step size equal to $\# \times n^{-1/2}$.

derivopts are allowed only with `stepsize(adaptive)` and may be `search()`, `h()`, `scale()`, and `bounds()`.

`search(`*search_type*`)` specifies the approach used to search for an optimal step size for computing the numerical derivatives; three approaches are offered: `bracket`, `interpolate`, and `off`; see `deriv_init_search()` in [M-5] **deriv()**. The bracket method, `search(bracket)`, is chosen as the default for `stmgintcox` because it tends to perform better with interval-censored multiple-event data.

`h(#|`*matname*`)` specifies the h values, which are multipliers for step size used to compute numerical derivatives; see `deriv_init_h()` in [M-5] **deriv()**. You can specify the same h value, #, for all parameters or parameter-specific h values as a Stata matrix (vector) *matname*.

`scale(#|`*matname*`)` specifies the starting scale values used to compute numerical derivatives; see `deriv_init_scale()` in [M-5] **deriv()**. You can specify the same initial scale value, #, for all parameters or parameter-specific initial scale values as a Stata matrix (vector) *matname*.

`bounds(`$\#_1$ $\#_2$`)` specifies the minimum and maximum values used to search for optimal scale values; see `deriv_init_bounds()` in [M-5] **deriv()**. The default is `bounds(1e-6 1e-5)`.

`iterate(#)` specifies the maximum number of iterations to compute the VCE based on the profile log pseudolikelihood. The default is `iterate(5000)`.

`tolerance(#)` specifies the tolerance for the profile log pseudolikelihood used to compute the VCE. The default is `tolerance(1e-6)`.

`dots`, `dots(#)`, and `nodots` display or suppress iteration dots showing the progress of the variance estimation. By default, the dot is displayed every iteration, but you can change this by specifying `dots(#)` or completely suppressing the dots with `nodots`.

The default is `dots`, unless you used `set iterlog off` to suppress them; see `set iterlog` in [R] ***set iter***. *vce_options* `dots`, `dots(#)`, and `nodots` override the display settings of `log`, `nolog`, and `dots`.

When a fixed step size is used, an iteration corresponds to one derivative computation with respect to a regression coefficient. When an adaptive step size is used, an iteration corresponds to one call of the Mata `deriv()` function, which may be called multiple times to compute one derivative with respect to one regression coefficient. Thus, you will typically see more iteration dots with VCE estimation using an adaptive step size than using a fixed step size.

These options do not appear in the dialog box.

`post` can be used only on replay. It replaces the current `e(V)` with the specified *vcespec*. When `vce(`*vcespec*`)` is used on replay without `post`, the coefficient table will display the standard error of the specified *vcespec*, but `e(V)` will remain unchanged. This option does not appear in the dialog box.

Reporting

`level(#)`; see [R] **Estimation options**.

`saving(`*filename*`[ , replace ])` saves the estimated baseline hazard contributions in *filename*`.dta`. `replace` specifies to overwrite *filename*`.dta` if it exists. If `saving()` is not specified, `stmgintcox` saves estimation results in a temporary file for later access by postestimation commands. This temporary file will be overridden every time `stmgintcox` is run and will also be erased if the current estimation results are cleared. `saving()` may be specified during estimation or on replay.

Because the file containing the baseline hazard contributions is considered to be part of the estimation results, you must use `saving()` before storing or saving your estimation results using `estimates store` or `estimates save`.

`nohr` specifies that regression coefficients be displayed rather than exponentiated regression coefficients or hazard ratios. This option affects only how results are displayed and not how they are estimated. `nohr` may be specified at estimation time or when replaying results.

`noheader` suppresses the output header, either at estimation or upon replay.

`detail` reports the censoring information table for each event. By default, the overall censoring information for all events is displayed above the coefficient table. If the `tvc()` option is specified, `detail` will also report the results for event-specific Wald tests for the proportional-hazards assumption at the bottom of the coefficient table. By default, we get only a Wald test for the global proportional-hazards assumption.

`log` and `nolog` display or suppress `stmgintcox`'s iteration log, which includes both the EM and VCE iterations. The EM iteration log displays the log-pseudolikelihood value every 100 iterations (the `emlog` option). The VCE iteration log displays iterations as dots (the `vcedots` option). The default is `log`, which implies `emlog` and `vcedots`, unless you used `set iterlog off` to suppress them; see `set iterlog` in [R] ***set iter***. `nolog` suppresses both EM and VCE iteration logs, and it is equivalent to specifying `noemlog` and `novcedots`.

You may override the display settings of `log` and `nolog` individually for the EM and VCE iterations with `novcedots`, `vcedots`, `vcedots(#)`, `noemlog`, `emlog`, `emlog(#)`, `noemdots`, `emdots`, and `emdots(#)`. `log` and `nolog` may not be combined with the `dots` option.

`dots` implies the `emdots` and `vcedots` options to display both the EM and VCE iteration logs as dots. The default is to display the VCE iteration log as dots and the log-pseudolikelihood value for the EM algorithm every 100 iterations; however, if you used `set iterlog off`, both the VCE iteration dots and EM iteration log are suppressed. See `set iterlog` in [R] ***set iter***. To display the EM iterations as dots, you can specify `dots` or `emdots`.

You may override the display settings of dots individually for the EM and VCE iterations with novcedots, vcedots(#), noemlog, emlog, emlog(#), noemdots, and emdots(#). dots may not be combined with log or nolog.

vcedots, vcedots(#), and novcedots are synonyms for vce(, dots), vce(, dots(#)), and vce(, nodots), respectively. The default is to display the VCE iteration log as dots; however, if you used set iterlog off, the VCE iteration dots are suppressed. See set iterlog in [R] ***set iter***. You can use vce(, nodots) to suppress the VCE iteration dots. vcedots, vcedots(#), and novcedots override the display settings of log, nolog, and dots.

emlog, emlog(#), and noemlog display or suppress an iteration log showing the progress of the EM algorithm. The default is to display the EM iteration log, unless you used set iterlog off to suppress it; see set iterlog in [R] ***set iter***. emlog, the default, displays the log-pseudolikelihood value every 100 iterations and is equivalent to emlog(100); noemlog suppresses it. emlog(#) displays the log-pseudolikelihood value every #th iterations. emlog, emlog(#), and noemlog override the display settings of log, nolog, and dots.

noemdots, emdots(#), and emdots control the display of the EM iteration log as dots. By default, the EM iteration log displays the log-pseudolikelihood value every 100 iterations; that is, noemdots is implied. Instead, you can specify emdots to display every 100 iterations as a dot or emdots(#) to display every # iterations as a dot. This is a useful alternative for long EM iteration logs.

The default is emlog, which implies noemdots; however, if you used set iterlog off, the EM iteration log is suppressed. See set iterlog in [R] ***set iter***. noemdots, emdots, and emdots(#) override the display settings of log, nolog, and dots.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

EM options

emiterate(#), emtolerance(#), emltolerance(#), emhsgtolerance(#), noemhsgtolerance, and from(); see iterate(), tolerance(), ltolerance(), nrtolerance(), nonrtolerance, and from() in [R] **Maximize**. These options control the EM optimization process. The defaults are emiterate(5000), emtolerance(1e-6), emltolerance(1e-7), and emhsgtolerance(1e-5). These options can be specified globally with the command and individually for a specific event. Options specified for a specific event will override globally specified options. These options may not be combined with the favorspeed option.

The following option is available with stmgintcox but is not shown in the dialog box:

coeflegend; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

Introduction

stmgintcox fits a marginal Cox proportional hazards model to interval-censored multiple-event data. Interval-censoring occurs when the failure time or the event time of interest is not exactly observed but is known only to lie within some interval. See *Introduction* in [ST] **stintcox** for details about interval-censored data. If you have right-censored data, see [ST] **stcox**. See [ST] **stintreg** for fitting parametric models to interval-censored data.

The Cox proportional hazards model was first introduced by Cox (1972) for right-censored survival data. For an introduction to interval-censored data, see Finkelstein and Wolfe (1985), Odell, Anderson, and D'Agostino (1992), Huang and Wellner (1997), Lindsey and Ryan (1998), and Sun (2006).

The Cox proportional hazards model specifies that the hazard function of the event time conditional on a p-vector of baseline (time-invariant or time-independent) covariates $\mathbf{x} = (x_1, \ldots, x_p)$ takes the form

$$h(t; \mathbf{x}) = h_0(t) \exp(\beta_1 x_1 + \cdots + \beta_p x_p)$$

where $\beta_1, \ldots, \beta_p$ are unknown regression coefficients and $h_0(t)$ is an arbitrary baseline hazard function. Under the proportional-hazards assumption, the hazard ratios, or exponentiated regression coefficients $\exp(\beta_1), \ldots, \exp(\beta_p)$, are constant over time. As with right-censored data, the Cox proportional hazards model is appealing for interval-censored data because it does not require parameterization of the baseline hazard function and, for low event rates, the exponentiated regression parameters approximate the relative risks.

The partial-likelihood approach (Cox 1972, 1975) is used to estimate parameters of the Cox model with right-censored data, in which some of the event times are observed exactly, while others are known to be longer than the duration of follow-up. Under interval-censoring, however, none of the event times are observed exactly. Thus, it is much more challenging to deal with interval-censored data than right-censored data, both theoretically and computationally. In particular, the traditional partial-likelihood approach is not applicable.

Zeng, Mao, and Lin (2016) developed a novel EM algorithm for efficient nonparametric maximum-likelihood estimation of the Cox proportional hazards model with univariate interval-censored data. It allows a completely arbitrary event-time distribution and results in consistent, asymptotically normal, and asymptotically efficient estimators of the regression parameters. See [ST] **stintcox** for more details.

Sometimes, we are interested in analyzing data with interval censoring in which the study subjects can experience more than one type of event or belong to some clusters. This type of data is known as multivariate interval-censored event-time data, and special methodology is required to model these data because event times may be correlated. Zeng, Gao, and Lin (2017) proposed a general framework for modeling multivariate event times using a general class of semiparametric transformation models with random effects. Several other authors (Chang, Wen, and Wu 2007; Chen et al. 2014 ; Chen, Tong, and Sun 2009; Wen and Chen 2013) proposed different types of random-effects models. However, random-effects modeling has some limitations. Statistical inference based on random-effects models may not be valid if the underlying distribution of random effects is misspecified and the relationship between event times is not appropriately modeled. In addition, results from these models do not have the population–average-effects interpretation that is often of interest because we interpret the regression parameters as conditional on the random effects. Another limitation of random-effects models is speed; the process used to fit these models is computationally intensive.

To address the above limitations, Xu, Zeng, and Lin (2023) proposed marginal proportional hazards models. Unlike random-effects models, these models do not require any specification of the dependence structure between multiple event times, and the distribution of event times is estimated nonparametri-

cally. In addition, the subjects can have arbitrary sequences of examination times. Because there is no need to model the dependence structure between event times, marginal models lead to more robust inference. Marginal models also produce estimates of parameters that can be interpreted as population-average effects. Finally, computational algorithms for marginal models are faster and more stable than for random-effects models. For more details about this method, see *Methods and formulas*.

Because each subject may experience multiple types of events, you must specify the subject identifier in the `id()` option and subjects' event-type identifiers in the `event()` option with the `stmgintcox` command. As with the `stintcox` command, event-time information can be specified with `stmgintcox` in two different ways depending on the storage format of the interval-censored event-time data. The two different storage formats are the single-record-per-event format (or time-intervals format) and multiple-record-per-event format (or examination-times format). With single-record-per-event data, you must use the `interval()` option to specify the two variables containing the lower and upper endpoints of the time interval for each event. With multiple-record-per-event data, you must use the `time()` option to specify the variable that records the examination times and the `status()` option to specify the event-status indicator. See *Single- versus multiple-record interval-censored data formats* for details.

`stmgintcox` supports time-varying (time-dependent) covariates for all or specific events. You can use the `tvc()` option to easily include time-varying covariates that are formed by multiplying covariates specified in `tvc()` with a deterministic function of time, specified in `texp()`. In a multiple-record-per-event format, you can include more general time-varying covariates; see *Single- versus multiple-record-per-event interval-censored data formats* and *Time-varying covariates*.

`stmgintcox` also supports flexible ways to specify the model. You can specify the common covariates for all the events, or you can specify different covariates for different events. You can also construct event-specific time-varying covariates; see *Flexible ways to specify the model* for details.

`stmgintcox` does not support data exhibiting delayed entry, gaps, and multiple failures per event.

Single- versus multiple-record-per-event interval-censored data formats

Interval-censored multiple-event data can be recorded in two different formats. In one format, the event-time information is recorded as interval data, with one record per event per subject containing lower and upper endpoints of the event-time interval. We call data stored in this format "single-record-per-event interval-censored data".

In the other format, the event-time information is recorded by a pair of an examination time and an event status at that time. The dataset typically contains multiple records (multiple examination times) per event per subject. We call data stored in this format "multiple-record-per-event interval-censored data". This format is often used to accommodate time-varying covariates.

`stmgintcox` supports both formats. In both formats, `stmgintcox` requires that we specify the subject identifier variable in the `id()` option and event type variable in the `event()` option.

Single-record-per-event interval-censored data. Consider the following dataset:

```
id      event  ltime  rtime   x1     x2     x3
101       1      0      6     17     22      0
102       1      4      9     12     22      1
103       1     13      .     13     22      0
101       2      3      9     17     22      1
102       2      0      4     12     22      1
103       2      7      .     13     22      1
```

Here each subject contains two types of event identified by variable `event`, and each event contains only a single record. Covariates `x1-x3` are constant over time for each event. Variables `ltime` and `rtime` record the respective lower, t_l, and upper, t_u, endpoints of the event-time interval for each event. `stmgintcox` requires that we specify these variables in the `interval()` option:

```
. stmgintcox x1 x2 x3, id(id) event(event) interval(ltime rtime) ...
```

In this format, if the data are left-censored, the lower endpoint is zero and may be represented in t_l by either a missing value (`.`) or zero. If the data are right-censored, the upper endpoint is $+\infty$ and is represented in t_u by a missing value. Uncensored data are represented by the two endpoints that are equal. If $0 < t_l < t_u < \infty$, the data are interval-censored. Truly missing values must be represented by missing values in both t_l and t_u or by a 0 in t_l and a missing value in t_u. Uncensored observations, with $t_l = t_u$, are not allowed in the presence of left-censored or interval-censored observations.

In our example, for event `1`, subject 101 is left-censored, subject 102 is interval-censored, and subject 103 is right-censored. For event `2`, subject 101 is interval-censored, subject 102 is left-censored, and subject 103 is right-censored.

This format is convenient for storing data containing interval-censored observations and covariates that are constant over time.

Multiple-record-per-event interval-censored data. In this format, we can easily record time-varying covariates for each event. Suppose that the values of variable `x3` are varying with time for each event in the following dataset:

id	event	time	status	x1	x2	x3
101	1	6	1	17	22	0
102	1	4	0	12	22	1
102	1	6	0	12	22	0
102	1	9	1	12	22	1
103	1	13	0	13	22	0
101	2	3	0	17	22	1
101	2	9	1	17	22	0
102	2	4	1	12	22	0
103	2	7	0	13	22	1
103	2	13	0	13	22	0

Here variable `time` records examination times, and variable `status` records the event status indicator for each examination time. And subjects may have multiple examination times for each event. In this format, `stmgintcox` requires that we specify the subject identifier (`id`) in the `id()` option, the event type variable (`event`) in the `event()` option, the examination time (`time`) in the `time()` option, and the event status indicator (`status`) in the `status()` option.

```
. stmgintcox x1 x2 x3, id(id) event(event) time(time) status(status) ...
```

Examination times and event status at each examination time can be used to determine the censoring type and the event-time interval for each event. In our example, for event 1, subject 101 has only one examination time, 6, and the event has already occurred by that time. Subject 101 is thus left-censored with the time interval $(t_l, t_u] = (0, 6]$. Subject 102 has three examination times, 4, 6, and 9, and the event occurred by time 9. Subject 102 is interval-censored with the time interval $(6, 9]$. Subject 103 has one examination at time 13, and the event has not occurred by that time. This subject is right-censored with the time interval $(13, +\infty)$.

For event 2, subject 101 has two examination times, 3 and 9, and the event occurred by time 9. Thus, this subject is interval-censored with time interval $(3, 9]$. Subject 102 has only one examination at time 4, and the event has already occurred by that time. Subject 102 is left-censored for event 2. Subject 103 has two examinations at times 7 and 13, and the event has not occurred by time 13. So this subject is right-censored for event 2 with the time interval $(13, +\infty)$.

For more discussion about single- versus multiple-record interval-censored data formats, see *Single- versus multiple-record interval-censored data formats* in [ST] **stintcox**.

Flexible ways to specify the model

`stmgintcox` offers different ways to specify the model depending on whether the model contains some event-specific covariates. Suppose our dataset contains three types of event with value labels `"event1"`, `"event2"`, and `"event3"` and three covariates of interest, `x1`, `x2`, and, `x3`. The following specifications work for both single- and multiple-record-per-event data formats.

If all covariates are common for all events, you can specify the model as

```
. stmgintcox x1 x2 x3, ...
```

This is the same as repeatedly specifying those covariates in all equations but offering an easier and shorter alternative.

If you have common covariate `x1` for all events, and you have event-specific covariates `x2` for `"event2"` and `x3` for `"event3"`, you can specify the model as the combination of common and event-specific covariates,

```
. stmgintcox x1 ("event2": x2) ("event3": x3), ...
```

This is the same as including the common covariate within each equation,

```
. stmgintcox ("event1": x1) ("event2": x1 x2) ("event3": x1 x3), ...
```

Suppose you want to specify the model with covariate `x1` for `"event1"`, covariates `x1` and `x2` for `"event2"`, and covariate `x3` for `"event3"`, you can use the following syntax,

```
. stmgintcox ("event1": x1) ("event2": x1 x2) ("event3": x3), ...
```

or specify `x1` as a common covariate but also use `nocommon` in equation `"event3"` to exclude the common covariate in that equation:

```
. stmgintcox x1 ("event2": x2) ("event3": x3, nocommon), ...
```

You can also specify different `tvc()` and `texp()` within the event-specific equations to include time-varying covariates that are interactions between those covariates and a deterministic function of time. For example, if you want to additionally include the interaction of `x2` and function of time `log(_t)` for equation `"event2"`, you can specify this as follows:

```
. stmgintcox x1 ("event2": x2, tvc(x2) texp(log(_t))) ("event3": x3, nocommon), ...
```

Specifying global `tvc()` and `texp()` options will allow you to add those time-varying covariates to all equations, unless you specify `tvc()` and `texp()` or `nocommon` within the event-specific equations, which will allow Stata to override the global options for that specific equation. See *Fitting models with event-specific covariates* for an example.

Single-record-per-event interval-censored data

In a single-record-per-event format, the interval that brackets the event time of interest, the event-time interval, is recorded for each event. The event of interest may occur before the first examination time, resulting in a left-censored observation; after the last examination time, resulting in a right-censored observation; or between two examination times, resulting in an interval-censored observation.

▷ Example 1: Single-record-per-event interval-censored data

Xu, Zeng, and Lin (2023) considered a study of the Atherosclerosis Risk in Communities (ARIC) on a cohort of 14,751 Caucasian and African-American individuals from four US communities (The ARIC investigators 1989). Baseline examinations for participants were performed in 1987–1989. Then follow-up examinations were performed approximately every three years. In 2011–2013, an additional examination was performed. There are several interval-censored events, including onset of diabetes and onset of hypertension.

The data used in this example are simulated based on the study described above. The dataset contains 200 subjects: 96 are females and 104 are males. All subjects are from one of the 4 US communities: 39 from Forsyth County, North Carolina; 37 from Jackson, Mississippi; 61 from suburbs of Minneapolis, Minnesota; and 63 from Washington County, Maryland. All subjects are examined to determine the onset of two different events (`event`): event 1, diabetes (`"Diabetes"`), and event 2, hypertension (`"Hypertension"`). We wish to identify the factors that influence time to onset of diabetes and hypertension. The covariates that we are interested in are `race`, sex (`male`), `community`, and five baseline risk factors: age (`age`), body mass index (`bmi`), glucose level (`glucose`), systolic blood pressure (`sysbp`), and diastolic blood pressure (`diabp`). The dataset also contains two variables, `ltime` and `rtime`, that record, respectively, the last examination time before the event happened and the first examination time after the event happened.

```
. use https://www.stata-press.com/data/r19/aric
(Simulated ARIC Study)

. describe
Contains data from https://www.stata-press.com/data/r19/aric.dta
 Observations:           400                  Simulated ARIC data
    Variables:            12                  25 Nov 2024 14:05
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
id	int	%9.0g		Subject ID
event	byte	%12.0g	evlab	Event type
ltime	int	%9.0g		Last exam time before event occurred
rtime	int	%9.0g		First exam time after event occurred
age	byte	%9.0g		Age at baseline
community	byte	%11.0g	commlab	Community of residence
male	byte	%9.0g	sexlab	Male
race	byte	%9.0g	racelab	Race
bmi	float	%9.0g		Body mass index
glucose	float	%9.0g		Glucose level
sysbp	int	%9.0g		Systolic blood pressure
diabp	byte	%9.0g		Diastolic blood pressure

```
Sorted by: id  event
```

Below, we use `stmgintcox` to fit a marginal Cox proportional hazards model in which the time to onset of diabetes and hypertension depends on `age`, `male`, `community`, `race`, `bmi`, `glucose`, `sysbp`, and `diabp`. Because we are working with single-record-per-event data, we must use the `interval()` option to specify the variables that record the lower and upper endpoints of the event-time interval. And we must specify the subject and event identifiers with the `id()` and `event()` options. For output brevity, we also specify the `vcedots(10)` option to display a dot every 10 iterations when computing standard errors. Note that because `stmgintcox` estimates baseline-hazard jumps at all distinct times across all events, it is computationally intensive and may take a few minutes to fit.

```
. stmgintcox age i.male i.community i.race bmi glucose sysbp diabp,
> id(id) event(event) interval(ltime rtime) vcedots(10)
note: using adaptive step size to compute derivatives.
note: displaying dots every 10 iterations.

Performing EM optimization for event "Diabetes" (showing every 100 iterations):
Iteration 0:   Log pseudolikelihood = -179.18329
Iteration 100: Log pseudolikelihood = -110.21215
Iteration 200: Log pseudolikelihood = -109.98083
Iteration 300: Log pseudolikelihood = -109.93752
Iteration 400: Log pseudolikelihood = -109.92532
Iteration 500: Log pseudolikelihood = -109.92091
Iteration 600: Log pseudolikelihood = -109.91902
Iteration 624: Log pseudolikelihood = -109.91874

Performing EM optimization for event "Hypertension" (showing every 100
> iterations):
Iteration 0:   Log pseudolikelihood = -208.75033
Iteration 100: Log pseudolikelihood = -160.09944
Iteration 200: Log pseudolikelihood = -159.88744
Iteration 300: Log pseudolikelihood =  -159.8376
Iteration 400: Log pseudolikelihood = -159.82201
Iteration 500: Log pseudolikelihood =  -159.8162
Iteration 600: Log pseudolikelihood = -159.81368
Iteration 610: Log pseudolikelihood = -159.81352

Computing standard errors: ........................................... done
```

```
Marginal interval-censored Cox regression          Number of events   =      2
Baseline hazard: Reduced intervals                 Number of subjects =    200
                                                   Number of obs      =    400
ID variable: id                                           Uncensored  =      0
Event variable: event                                Left-censored    =     47
Event-time interval:                                 Right-censored   =    240
  Lower endpoint: ltime                              Interval-cens.   =    113
  Upper endpoint: rtime
                                                   Wald chi2(20)      = 128.31
Log pseudolikelihood = -269.73225                  Prob > chi2        = 0.0000
```

	Haz. ratio	Robust std. err.	z	P>|z|	[95% conf.	interval]
Diabetes						
age	.95663	.0283264	-1.50	0.134	.9026917	1.013791
male						
Yes	.8067524	.2301774	-0.75	0.452	.4611911	1.411236
community						
Jackson	1.594206	.6427935	1.16	0.247	.7233276	3.513614
Minneapolis	1.036757	.3611512	0.10	0.917	.523798	2.052062
Washington	1.419528	.49871	1.00	0.319	.7130152	2.826109
race						
White	.4267559	.1203645	-3.02	0.003	.2455286	.741749
bmi	1.119352	.0262064	4.82	0.000	1.069149	1.171913
glucose	1.139996	.0198973	7.51	0.000	1.101657	1.179669
sysbp	1.020623	.0117379	1.77	0.076	.9978743	1.04389
diabp	.9920976	.0117945	-0.67	0.505	.969248	1.015486
Hypertension						
age	.9951778	.0192589	-0.25	0.803	.9581379	1.03365
male						
Yes	.6667426	.1584725	-1.71	0.088	.4184497	1.062364
community						
Jackson	.6045343	.2070578	-1.47	0.142	.3089425	1.182944
Minneapolis	.8997162	.2690857	-0.35	0.724	.5006446	1.616894
Washington	.6734203	.2210427	-1.20	0.228	.3539067	1.281397
race						
White	1.240887	.3461451	0.77	0.439	.7182731	2.143752
bmi	1.012572	.0170935	0.74	0.459	.9796174	1.046635
glucose	.9898238	.0094633	-1.07	0.285	.9714488	1.008546
sysbp	1.075034	.012595	6.18	0.000	1.05063	1.100006
diabp	1.025542	.011092	2.33	0.020	1.004031	1.047514

```
Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.
```

The header above the coefficient table provides information on the number of events, number of subjects, and the number of uncensored, left-censored, right-censored, and interval-censored observations. We can use the `detail` option to also get censoring information for each event.

In the output for event diabetes, we see that White individuals have a lower risk of diabetes. Also, higher values of the body mass index and elevated glucose levels are associated with a higher risk of diabetes. In the output for event hypertension, we see that higher values of systolic and diastolic blood pressure are associated with a greater risk of hypertension.

Before the iteration log, `stmgintcox` displays a note stating that an adaptive step size is used to compute derivatives during VCE computation. The command also displays a note following the output table about potential variability of the standard error estimates. We address all of this in more detail in *Standard error estimation with interval-censored data* in [ST] **stintcox**.

◁

▷ Example 2: Fitting models with event-specific covariates

From the model above, we can see that body mass index and glucose level are important risk factors for diabetes but not for hypertension. Conversely, systolic and diastolic blood pressure may play a role in the risk of hypertension but not the risk of diabetes. Therefore, we can use different sets of covariates to model the two events. We also use the `favorspeed` option so that the command will run more quickly in this example.

```
. stmgintcox ("Diabetes": age i.male i.community i.race bmi glucose)
> ("Hypertension": age i.male i.community i.race sysbp diabp),
> id(id) event(event) interval(ltime rtime) favorspeed vcedots(10)
note: using fixed step size with a multiplier of 5 to compute derivatives.
note: displaying dots every 10 iterations.
note: using EM and VCE tolerances of 0.0001.
note: option noemhsgtolerance assumed.

Performing EM optimization for event "Diabetes" (showing every 100 iterations):
Iteration 0:   Log pseudolikelihood = -179.18329
Iteration 71:  Log pseudolikelihood = -111.73164

Performing EM optimization for event "Hypertension" (showing every 100
> iterations):
Iteration 0:   Log pseudolikelihood = -208.75033
Iteration 61:  Log pseudolikelihood =  -161.0338

Computing standard errors: ........................... done
```

```
Marginal interval-censored Cox regression          Number of events   =      2
Baseline hazard: Reduced intervals                 Number of subjects =    200
                                                   Number of obs      =    400
ID variable: id                                           Uncensored  =      0
Event variable: event                                Left-censored    =     47
Event-time interval:                                 Right-censored   =    240
  Lower endpoint: ltime                              Interval-cens.   =    113
  Upper endpoint: rtime
                                                   Wald chi2(16)      =  77.01
Log pseudolikelihood = -272.76543                  Prob > chi2        = 0.0000
```

	Haz. ratio	Robust std. err.	z	P>|z|	[95% conf.	interval]
Diabetes						
age	.9693495	.0293552	-1.03	0.304	.9134885	1.028626
male						
Yes	.8021755	.2273265	-0.78	0.437	.4603091	1.397942
community						
Jackson	1.549902	.6274179	1.08	0.279	.7010166	3.426733
Minneapolis	.9649113	.3361108	-0.10	0.918	.4875122	1.909806
Washington	1.36829	.5112313	0.84	0.401	.6578786	2.845842
race						
White	.4412767	.135994	-2.65	0.008	.2412044	.8073037
bmi	1.112781	.0314166	3.79	0.000	1.052878	1.176092
glucose	1.141379	.0304922	4.95	0.000	1.083153	1.202735
Hypertension						
age	.9945906	.0220662	-0.24	0.807	.9522686	1.038794
male						
Yes	.6229044	.1403048	-2.10	0.036	.4005846	.9686091
community						
Jackson	.606375	.1824113	-1.66	0.096	.3362643	1.093457
Minneapolis	.8873364	.2642854	-0.40	0.688	.4949546	1.590784
Washington	.6548935	.1999546	-1.39	0.166	.3599802	1.191414
race						
White	1.26674	.4058107	0.74	0.460	.6760798	2.373433
sysbp	1.072573	.0149785	5.02	0.000	1.043614	1.102336
diabp	1.025091	.0138294	1.84	0.066	.9983414	1.052558

```
Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.
```

Following the command specification, we now see additional notes regarding the updated convergence criteria, which are the results of specifying `favorspeed`. The EM and VCE tolerances have been adjusted to 0.0001, compared with the defaults of 1e–6. The tolerance check for the Hessian scaled gradient is now suppressed. Additionally, a fixed step size with the multiplier of 5 is used to compute derivatives during the VCE computation.

Because age, male, community, and race are common covariates for both events, you could also specify the model more conveniently by using a combination of common covariates and event-specific covariates:

```
. stmgintcox age i.male i.community i.race
> ("Diabetes": bmi glucose) ("Hypertension": sysbp diabp),
> id(id) event(event) interval(ltime rtime) favorspeed
```

◁

Incorporating time-varying covariates using the tvc() option

As with stintcox, there are two ways to incorporate time-varying covariates with stmgintcox. If you already have data in which some variables vary with examination times and those data are already present in multiple-record-per-event format, you can use stmgintcox's multiple-record-per-event syntax to analyze such data. See example 4.

If a time-varying covariate can be represented as an interaction between a baseline covariate and a deterministic function of time, you can use the tvc() and texp() options to include such a covariate in the model. You can do this with either single-record or multiple-record interval-censored data. When you specify the tvc() option, stmgintcox performs a more memory-efficient computation and automatically incorporates the specified function of time in various calculations during estimation and postestimation. You can also incorporate different interactions between covariates and deterministic functions of time for different events by specifying tvc() and texp() within event-specific equations.

One way of testing the proportional-hazards assumption for a covariate is to test whether the coefficient associated with that covariate is time invariant. This can be accomplished by including an interaction between that covariate and a function of time in the model and testing whether the corresponding coefficient equals zero. This can be easily accomplished by specifying the covariates suspected of violating the proportional-hazards assumption in tvc(). The following example provides more details.

▷ Example 3: Testing the proportional-hazards assumption

Continuing with example 1, suppose that we want to test the proportional-hazards assumption for the `bmi` variable for both events. We want to interact this variable with years (`_t/365`) for both events, so we specify the covariate in `tvc()` and the function of time as `texp(_t/365)`. By default, `stmgintcox` will report the Wald test for the global proportional-hazards assumption; the null hypothesis is that all the coefficients in the `tvc` equation are equal to zero for all events. Below, we add `detail` to also get Wald tests for each event.

```
. stmgintcox age i.male i.community i.race bmi glucose sysbp diabp,
> id(id) event(event) interval(ltime rtime) tvc(bmi) texp(_t/365)
> detail nohr nolog
note: using adaptive step size to compute derivatives.
Marginal interval-censored Cox regression       Number of events   =         2
Baseline hazard: Reduced intervals              Number of subjects =       200
                                                Number of obs      =       400
ID variable: id                                         Uncensored =         0
Event variable: event                              Left-censored =        47
Event-time interval:                              Right-censored =       240
  Lower endpoint: ltime                           Interval-cens. =       113
  Upper endpoint: rtime
                                                Wald chi2(22)      = 10239.79
Log pseudolikelihood = -267.42419               Prob > chi2        =   0.0000
```

		Subjects per event				
Event	No. of obs.	Uncensored	Censored: Left	Censored: Right	Censored: Interval	Total
Diabetes	200	0	22	134	44	200
Hyperte~n	200	0	25	106	69	200

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
Diabetes						
age	-.0472282	.0285623	-1.65	0.098	-.1032092	.0087529
male						
Yes	-.267267	.2858202	-0.94	0.350	-.8274644	.2929303
community						
Jackson	.3973853	.413791	0.96	0.337	-.4136302	1.208401
Minneapolis	.0238152	.3530424	0.07	0.946	-.6681351	.7157656
Washington	.3880326	.3523592	1.10	0.271	-.3025788	1.078644
race						
White	-.8662366	.2803032	-3.09	0.002	-1.415621	-.3168524
bmi	.0382006	.0240719	1.59	0.113	-.0089795	.0853808
glucose	.1372663	.0131616	10.43	0.000	.1114701	.1630625
sysbp	.0231018	.01073	2.15	0.031	.0020714	.0441323
diabp	-.0089993	.0113405	-0.79	0.427	-.0312262	.0132277
tvc_Diabetes						
bmi	.030363	.005028	6.04	0.000	.0205083	.0402177
Hypertension						
age	-.0054012	.0203236	-0.27	0.790	-.0452348	.0344324
male						
Yes	-.4089727	.2278163	-1.80	0.073	-.8554844	.0375389
community						
Jackson	-.5071775	.3764983	-1.35	0.178	-1.245101	.2307455
Minneapolis	-.1002443	.3348892	-0.30	0.765	-.756615	.5561264
Washington	-.3958838	.3752002	-1.06	0.291	-1.131263	.339495
race						
White	.2129527	.2822622	0.75	0.451	-.340271	.7661764
bmi	.0178395	.0322651	0.55	0.580	-.0453989	.081078
glucose	-.0098966	.0098305	-1.01	0.314	-.0291642	.0093709
sysbp	.0724637	.0118817	6.10	0.000	.049176	.0957515
diabp	.0250423	.0108181	2.31	0.021	.0038392	.0462454
tvc_Hyperte~n						
bmi	-.0022996	.0092792	-0.25	0.804	-.0204865	.0158873

```
Notes: Standard error estimates may be more variable for small datasets and
       datasets with low proportions of interval-censored observations.
       Variables in tvc equations interacted with _t/365.

Wald test that [tvc] = 0:
  All events: chi2(2) = 36.8911                      Prob > chi2 = 0.0000
    Diabetes: chi2(1) = 36.4668                      Prob > chi2 = 0.0000
Hypertension: chi2(1) = 0.0614                       Prob > chi2 = 0.8043
```

The first equation reports the coefficients of the covariates for event diabetes that do not vary over time; the second equation, `tvc_Diabetes`, reports the results for `bmi` interacted with years (`_t/365`) in the model for the onset of diabetes. The third equation reports the coefficients of the covariates for event hypertension that do not vary over time; the fourth equation, `tvc_Hypertension`, reports the results of `bmi` interacted with years in the model for the onset of hypertension.

Without the nohr option, the table would present results as hazard ratios for all the equations; exponentiated coefficients can be interpreted as hazard ratios when we are actually modeling time-varying covariates. However, in this case, we are using tvc() to test time-varying coefficients for all events; in our model, the coefficient for covariate k for event i ($i = 1, 2$) is expressed as $b_{ki} + \gamma_{ki} \times t/365$, where b_{ki} corresponds to the parameter in the time-invariant equation for event i and γ_{ki} corresponds to the parameter in the event-specific tvc equation. The proportional-hazards assumption does not hold unless $\gamma_{ki} = 0$.

Looking at the output for the tvc_Diabetes and tvc_Hypertension equations, we have evidence that the proportional-hazards assumption is violated for the covariate bmi in relation to diabetes but not for hypertension. At the bottom of the table, stmgintcox also displays results for a Wald test for the global proportional-hazards assumption; the null hypothesis for this global test is that all the coefficients in both tvc equations are equal to zero. Because we specified detail, we also get a Wald test for each event. The results of the Wald tests show that the proportional-hazards assumption has been violated globally as well as for the diabetes event.

◁

Multiple-record-per-event interval-censored data

You may already have data in which some variables vary with time. These data will have multiple examination times per event for some subjects. You can use stmgintcox's multiple-record-per-event syntax to analyze data with existing time-varying covariates. And you can still include time-varying covariates that are deterministic functions of time by specifying baseline or existing time-varying variables in tvc().

▷ Example 4: Multiple-record-per-event interval-censored data

The dataset used in this example is an extended version of the dataset described in example 1; it consists of four covariates—bmi, glucose, sysbp, and diabp—that are varying with examination times. The data are recorded in the multiple-record-per-event format, where each observation records a subject identifier (id), an event type identifier (event), the examination time (time), and whether the event occurred since the last examination time (status). Each observation also records baseline (time-invariant) factors, such as age at recruitment (age), community of residence, sex (male), and race as described in example 1.

Below, we list the observations for subject 1:

```
. use https://www.stata-press.com/data/r19/aric2
(Simulated ARIC study with time-varying variables)
. list event-age bmi-diabp if id==1, compress noobs
```

event	time	sta~s	age	bmi	glucose	sysbp	diabp
Diabetes	96	0	50	31.46205	71.2087	113	69
Diabetes	1389	0	50	35.72507	88.52099	108	60
Hypertension	96	0	50	31.46205	71.2087	113	69
Hypertension	1389	0	50	35.72507	88.52099	108	60

There are two events (Diabetes and Hypertension) with multiple records per event. The examination time and event status variables define event-time intervals and censoring types for each event. Subject 1 has been examined at two different times: one at day 96 and one follow-up examination at day 1,389. She has never been diagnosed with diabetes at any of her examinations. She is right-censored with an

event-time interval $(1389, +\infty)$ for diabetes. Also, she has never been diagnosed with hypertension. She is right-censored with the same event-time interval for hypertension. Her body mass index (bmi), glucose level, systolic blood pressure, and diastolic blood pressure vary at each examination time, while other covariates remain the same as the baseline factors.

We use stmgintcox to fit a Cox proportional hazards model in which the time to onset of events depends on baseline covariates age, male, community, race, and time-varying covariates bmi, glucose, sysbp, and diabp. In a multiple-record-per-event format, we must specify id(), event(), time(), and status(). We also include detail to get detailed censoring information for each event and vcedots(10) to shorten the standard error iteration log.

```
. stmgintcox age i.male i.community i.race bmi glucose sysbp diabp,
> id(id) event(event) time(time) status(status) detail vcedots(10)
note: time-varying covariates detected in the data; using method nearleft to
      impute their values between examination times.
note: using adaptive step size to compute derivatives.
note: displaying dots every 10 iterations.

Performing EM optimization for event "Diabetes" (showing every 100 iterations):
Iteration 0:   Log pseudolikelihood = -179.18329
Iteration 100: Log pseudolikelihood = -104.79734
Iteration 200: Log pseudolikelihood = -104.54451
Iteration 300: Log pseudolikelihood = -104.49087
Iteration 400: Log pseudolikelihood = -104.47041
Iteration 500: Log pseudolikelihood = -104.46061
Iteration 600: Log pseudolikelihood = -104.45551
Iteration 700: Log pseudolikelihood = -104.45274
Iteration 800: Log pseudolikelihood =  -104.4512
Iteration 820: Log pseudolikelihood = -104.45098

Performing EM optimization for event "Hypertension" (showing every 100
> iterations):
Iteration 0:   Log pseudolikelihood = -208.75033
Iteration 100: Log pseudolikelihood = -160.62508
Iteration 200: Log pseudolikelihood = -160.39564
Iteration 300: Log pseudolikelihood = -160.34713
Iteration 400: Log pseudolikelihood = -160.33315
Iteration 500: Log pseudolikelihood = -160.32838
Iteration 563: Log pseudolikelihood = -160.32704

Computing standard errors: ..................................................... done

Marginal interval-censored Cox regression        Number of events   =        2
Baseline hazard: Reduced intervals               Number of subjects =      200
                                                 Number of obs      =      830
ID variable: id                                          Uncensored =        0
Event variable: event                                Left-censored =       47
Examination time: time                              Right-censored =      240
Status indicator: status                            Interval-cens. =      113

                                                 Wald chi2(20)      = 13197.36
Log pseudolikelihood = -264.77802                Prob > chi2        =   0.0000
```

		Subjects per event				
Event	No. of obs.	Uncensored	Left	Censored Right	Interval	Total
Diabetes	420	0	22	134	44	200
Hyperte~n	410	0	25	106	69	200

time	Haz. ratio	Robust std. err.	z	P>|z|	[95% conf.	interval]
Diabetes						
age	.9551756	.028196	-1.55	0.120	.9014808	1.012069
male						
Yes	.8063021	.2397021	-0.72	0.469	.4502437	1.443936
community						
Jackson	1.495371	.6396808	0.94	0.347	.646587	3.458365
Minneapolis	1.022231	.3737445	0.06	0.952	.4992699	2.092967
Washington	1.457473	.505914	1.09	0.278	.7381318	2.877844
race						
White	.4329106	.1197329	-3.03	0.002	.2517537	.7444243
bmi	1.119833	.0211927	5.98	0.000	1.079057	1.16215
glucose	1.160905	.0161295	10.74	0.000	1.129718	1.192952
sysbp	1.024272	.0124467	1.97	0.048	1.000165	1.048959
diabp	.9822802	.0117356	-1.50	0.135	.959546	1.005553
Hypertension						
age	1.000677	.0172332	0.04	0.969	.967464	1.03503
male						
Yes	.7086539	.1573446	-1.55	0.121	.4586054	1.095038
community						
Jackson	.6784416	.2244297	-1.17	0.241	.354759	1.297453
Minneapolis	1.00722	.3078072	0.02	0.981	.553346	1.833379
Washington	.7480168	.2412699	-0.90	0.368	.3975206	1.407548
race						
White	1.321735	.3545515	1.04	0.298	.7812895	2.236025
bmi	1.020171	.0171845	1.19	0.236	.9870395	1.054414
glucose	.9954429	.0094096	-0.48	0.629	.9771702	1.014057
sysbp	1.077175	.0099561	8.04	0.000	1.057837	1.096866
diabp	1.022089	.010188	2.19	0.028	1.002315	1.042254

Time varying: **bmi glucose sysbp diabp**
Note: Standard error estimates may be more variable for small datasets and
datasets with low proportions of interval-censored observations.

There is a total of 830 observations for both events on 200 subjects. The table above the coefficient table shows us the detailed censoring information for each event. For diabetes, there are 22 left-censored, 134 right-censored, and 44 interval-censored subjects. And for hypertension, there are 25 left-censored, 106 right-censored, and 69 interval-censored subjects.

After accounting for time-varying risk factors, we see that elevated glucose levels, a higher body mass index, and elevated systolic blood pressure are associated with an increased risk of diabetes onset. Being White is associated with a lower risk of developing diabetes. Additionally, elevated systolic and diastolic blood pressure are associated with a higher risk of developing hypertension.

stmgintcox identified four time-varying covariates in this dataset. In the presence of general time-varying covariates, stmgintcox offers several methods for imputing the values of these covariates between the examination times. By default, it uses covariate values at the nearest examination time on the left (nearleft) to impute unobserved covariate values between two examination times for each event. See the description of the tvcovimpute() option for details.

◁

Stored results

stmgintcox stores the following in e():

Scalars

e(N)	number of observations
e(N_events)	number of events
e(N_sub)	number of subjects
e(N_unc)	number of uncensored subjects
e(N_lc)	number of left-censored subjects
e(N_rc)	number of right-censored subjects
e(N_int)	number of interval-censored subjects
e(N_clust)	number of clusters
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_dv)	number of dependent variables
e(df_m)	model degrees of freedom
e(df_tvc)	degrees of freedom for proportional-hazards test
e(ll)	log pseudolikelihood
e(chi2)	χ^2
e(chi2_tvc)	χ^2 for proportional-hazards test
e(p)	p-value for model test
e(p_tvc)	p-value for proportional-hazards test
e(rank)	rank of e(V)
e(converged)	1 if converged, 0 otherwise
e(delta)	multiplier used with fixed step size
e(delta_n)	fixed step size
e(emiterate#)	maximum EM iterations for event #
e(emtolerance#)	EM coefficient tolerance for event #
e(emltolerance#)	EM log-pseudolikelihood tolerance for event #
e(emhsgtolerance#)	EM scaled-gradient tolerance for event #
e(noemhsgtolerance#)	1 if noemhsgtolerance, 0 otherwise
e(vceiterate)	maximum VCE iterations
e(vcetolerance)	VCE profile log-pseudolikelihood tolerance

Macros

e(cmd)	stmgintcox
e(cmdline)	command as typed
e(depvar)	names of time interval variables specified in interval() or name of examination time variable specified in time()
e(id)	id variable specified in id()
e(event)	event variable specified in event()
e(status)	name of status variable specified in status()
e(strata)	strata variables
e(clustvar)	name of cluster variable
e(title)	title in estimation output
e(title2)	secondary title in estimation output
e(chi2type)	Wald
e(vce)	*vcetype* specified in vce()

e(vcetype)	title used to label Std. err.
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(evnames)	event names
e(tvc_events)	events that include covariates interacted with time from tvc()
e(tvc#)	covariates interacted with time from tvc() for event # in e(tvc_events)
e(texp#)	function of time used for covariates from texp() for event # in e(tvc_events)
e(tvvariables)	time-varying variables detected in the data
e(tvcovimpute)	imputation method used for variables in e(tvvariables)
e(intervals)	reduced or full
e(filename)	name of the file with estimated baseline hazard contributions
e(stepsize)	adaptive or fixed
e(deriv_search)	search method used in search() with stepsize(adaptive)

Matrices

e(detail)	detailed censoring information for events
e(b)	coefficient vector
e(V)	variance–covariance matrix of the estimators

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

If the vce() option is specified on replay, the following estimation results will be updated only if vce()'s post suboption is specified: e(N_clust), e(chi2), e(p), e(clustvar), e(vce), e(stepsize), e(delta), and e(delta_n) (with fixed step size) and e(chi2_tvc) and e(p_tvc) (with tvc()).

Methods and formulas

Methods and formulas are presented under the following headings:

Data and model

For a comprehensive review of the methods in this entry, see Xu, Zeng, and Lin (2023).

Suppose that there are n subjects in the dataset and that each subject can experience K types of events. For $i = 1, \ldots, n$, and $k = 1, \ldots, K$, let T_{ik} denote the kth event time for the ith subject, and let $\mathbf{x}_{ik}(\cdot)$ denote a $1 \times p_k$ vector of covariates that can potentially depend on time. Under the Cox proportional hazards model, the hazard function of T_{ik} conditional on $\mathbf{x}_{ik}(\cdot)$ is

$$\lambda_{ik}(t; \mathbf{x}_{ik}) = \lambda_k(t) \exp\{\mathbf{x}_{ik}(t)\boldsymbol{\beta}_k\}$$

where $\boldsymbol{\beta}_k$ is a $p_k \times 1$ vector of unknown regression parameters and $\lambda_k(t)$ is an arbitrary baseline hazard function for event type k. Let $\Lambda_k(t) = \int_0^t \lambda_k(s)ds$, which is the baseline cumulative hazard function.

The occurrence of an asymptomatic event can be detected only through periodic examinations. Let $U_{ik,1} < \cdots < U_{ik,M_{ik}}$ denote a random sequence of positive examination times on T_{ik}, and assume that those examination times are independent of the event time. Let $(L_{ik}, R_{ik}]$ denote the shortest time interval that brackets T_{ik}, with $L_{ik} < R_{ik}$. Left-censoring is indicated by $L_{ik} = 0$, and right-censoring by $R_{ik} = \infty$. The pseudolikelihood function for $\boldsymbol{\beta}_k$ and $\Lambda_k(t)$ is

$$\prod_{i=1}^{n} \Big(\exp[-\int_0^{L_{ik}} \exp\{\mathbf{x}_{ik}(s)\boldsymbol{\beta}_k\} d\Lambda_k(s)] - \exp[-\int_0^{R_{ik}} \exp\{\mathbf{x}_{ik}(s)\boldsymbol{\beta}_k\} d\Lambda_k(s)] \Big)$$

where the integral is ∞ if $R_{ik} = \infty$.

EM algorithm for computing parameter estimates

Xu, Zeng, and Lin (2023) proposed a nonparametric maximum-pseudolikelihood estimation approach as an extension of the nonparametric maximum-likelihood approach described in Zeng, Mao, and Lin (2016). For each event type k, Λ_k is regarded as a step function with nonnegative jumps $h_{k1}, \ldots, h_{km_k}$ at $t_{k1}, \ldots, t_{km_k}$, respectively, where $t_{k1} < \cdots < t_{km_k}$ are the distinct time points for all $L_{ik} > 0$ and $R_{ik} < \infty$ for $i = 1, \ldots, n$. Thus, we need to maximize the function

$$\prod_{i=1}^{n} \exp\Big\{ - \sum_{t_{kq} \le L_{ik}} h_{kq} \exp(\mathbf{x}^*_{ikq}\boldsymbol{\beta}_k) \Big\} \Big[1 - \exp\Big\{ - \sum_{L_{ik} < t_{kq} \le R_{ik}} h_{kq} \exp(\mathbf{x}^*_{ikq}\boldsymbol{\beta}_k) \Big\} \Big]^{I(R_{ik} < \infty)} \tag{1}$$

where $\mathbf{x}^*_{ikq} = \mathbf{x}_{ik}(t_{kq})$ are the covariate values for event k of subject i at time t_{kq}. For baseline covariates, $\mathbf{x}^*_{ikq} = \mathbf{x}_{ik}$. For time-varying covariates, if $t_{kq} = t_{ikq}$ is one of the examination times for event k of subject i, then $\mathbf{x}^*_{ikq} = \mathbf{x}_{ikq}$. Otherwise, $\mathbf{x}^*_{ikq}$ is imputed as described in the `tvcovimpute()` option in *Options*.

For each event type k, we use the EM algorithm described in the `stintcox` command for computing parameter estimates; see *EM algorithm for computing parameter estimates* in [ST] **stintcox** for more details. The resulting maximum-pseudolikelihood estimators $\widehat{\boldsymbol{\beta}}_k$ and $\widehat{\Lambda}_{k0}(t) = \sum_{t_{kq} < t} \hat{h}_{kq}$ are strongly consistent. In addition, $\widehat{\boldsymbol{\beta}} = (\widehat{\boldsymbol{\beta}}_1^T, \ldots, \widehat{\boldsymbol{\beta}}_K^T)^T$ is asymptotically multivariate normal.

Variance estimation using the profile log-pseudolikelihood function

The covariance matrix of $\widehat{\boldsymbol{\beta}}$ can be estimated using a robust or clustered sandwich estimator based on the profile log pseudolikelihood (Murphy and van der Vaart 2000; Zeng, Mao, and Lin 2016; Zeng, Gao, and Lin 2017; Xu, Zeng, and Lin 2023).

The profile log-pseudolikelihood function for $\boldsymbol{\beta}_k$ takes the form

$$\begin{aligned} \mathrm{pl}_k(\boldsymbol{\beta}_k) = \sum_{i=1}^{n} \log\Big[& \exp\Big\{ - \sum_{t_{kq} \le L_{ik}} \tilde{h}_{kq} \exp(\mathbf{x}^*_{ikq}\boldsymbol{\beta}_k) \Big\} \\ & - I(R_{ik} < \infty) \exp\Big\{ - \sum_{t_{kq} \le R_{ik}} \tilde{h}_{kq} \exp(\mathbf{x}^*_{ikq}\boldsymbol{\beta}_k) \Big\} \Big] \end{aligned}$$

where $\tilde{h}_{kq}$ $(q = 1, \ldots, m_k)$ are obtained from the EM algorithm described in the previous section by updating only h_{kq} $(q = 1, \ldots, m_k)$ in the M-step while holding $\boldsymbol{\beta}_k$ fixed over the iterations.

Let $\text{pl}_{ki}(\boldsymbol{\beta}_k)$ be the contribution of the ith subject to $\text{pl}_k(\boldsymbol{\beta}_k)$. We can estimate the covariance matrix between $\widehat{\boldsymbol{\beta}}_k$ and $\widehat{\boldsymbol{\beta}}_l$ $(1 \le k, l \le K)$ by the sandwich estimator

$$\{D^2\text{pl}_k(\widehat{\boldsymbol{\beta}}_k)\}^{-1} \sum_{i=1}^{n} D\text{pl}_{ki}(\widehat{\boldsymbol{\beta}}_k) D\text{pl}_{li}(\widehat{\boldsymbol{\beta}}_l)^T \{D^2\text{pl}_l(\widehat{\boldsymbol{\beta}}_l)\}^{-1}$$

where D and D^2 are the first- and second-order numerical derivatives with step size δ_n chosen for numerical difference. Specifically, the jth element of D for function $f(\boldsymbol{\gamma})$ is

$$\big[Df(\boldsymbol{\gamma})\big]_j = \frac{f(\boldsymbol{\gamma} + \delta_n \mathbf{e}_j) - f(\boldsymbol{\gamma})}{\delta_n}$$

where $\mathbf{e}_j$ is the jth canonical vector in the parameter space. And the (j,k)th element of D^2 is

$$\big[D^2 f(\boldsymbol{\gamma})\big]_{j,k} = \frac{f(\boldsymbol{\gamma}) - f(\boldsymbol{\gamma} + \delta_n \mathbf{e}_k) - f(\boldsymbol{\gamma} + \delta_n \mathbf{e}_j) + f(\boldsymbol{\gamma} + \delta_n \mathbf{e}_k + \delta_n \mathbf{e}_j)}{\delta_n^2}$$

The `stepsize()` suboption of the `vce()` option offers different ways for you to choose the step size when computing numerical derivatives. `stepsize(adaptive)`, the default, uses an adaptive step size; see [M-5] **deriv()**. `stepsize(fixed)` uses a fixed step size $\delta_n = 5n^{-1/2}$ (Zeng, Mao, and Lin 2016; Zeng, Gao, and Lin 2017; Xu, Zeng, and Lin 2023). And `stepsize(fixed #)` uses a fixed step size equal to $\# \times n^{-1/2}$, where n is the number of subjects. With clustered data, when `vce(cluster` *clustvar*`)` is specified, `stepsize(fixed #)` uses a fixed step size equal to $\# \times J^{-1/2}$, where J is the number of clusters.

Clustered data estimation

In a more general case, suppose that there are J independent clusters, let T_{ijk} denote the kth event time for the ith subject in the jth cluster, and let $\mathbf{x}_{ijk}(\cdot)$ denote a $1 \times p_k$ vector of covariates for $i = 1, \ldots, n_j$, $j = 1, \ldots, J$, and $k = 1, \ldots, K$. Also, let $(L_{ijk}, R_{ijk}]$ denote the shortest time interval that brackets T_{ijk}. Thus, the function that needs to be maximized is

$$\prod_{j=1}^{J} \prod_{i=1}^{n_j} \exp\Big\{ - \sum_{t_{kq} \le L_{ijk}} h_{kq} \exp(\mathbf{x}^*_{ijkq} \boldsymbol{\beta}_k) \Big\} \Big[1 - \exp\Big\{ - \sum_{L_{ijk} < t_{kq} \le R_{ijk}} h_{kq} \exp(\mathbf{x}^*_{ijkq} \boldsymbol{\beta}_k) \Big\} \Big]^{I(R_{ijk} < \infty)}$$

where $\mathbf{x}^*_{ijkq} = \mathbf{x}_{ijk}(t_{kq})$ are the covariate values for event k of subject i in the jth cluster at time t_{kq}.

The covariance matrix of $\widehat{\boldsymbol{\beta}}$ can be estimated using a clustered sandwich estimator based on the profile log pseudolikelihood, where the profile log pseudolikelihood for $\boldsymbol{\beta}_k$ becomes

$$\begin{aligned} \text{pl}_k(\boldsymbol{\beta}_k) = \sum_{j=1}^{J} \sum_{i=1}^{n_j} \log \Big[& \exp\Big\{ - \sum_{t_{kq} \le L_{ijk}} \tilde{h}_{kq} \exp(\mathbf{x}^*_{ijkq} \boldsymbol{\beta}_k) \Big\} \\ & - I(R_{ijk} < \infty) \exp\Big\{ - \sum_{t_{kq} \le R_{ijk}} \tilde{h}_{kq} \exp(\mathbf{x}^*_{ijkq} \boldsymbol{\beta}_k) \Big\} \Big] \end{aligned}$$

Let $\text{pl}_{kj}(\boldsymbol{\beta}_k)$ be the contribution of the jth cluster to $\text{pl}_k(\boldsymbol{\beta}_k)$. We can estimate the covariance matrix between $\widehat{\boldsymbol{\beta}}_k$ and $\widehat{\boldsymbol{\beta}}_l$ $(1 \leq k, l \leq K)$ by the clustered sandwich estimator

$$\{D^2\text{pl}_k(\widehat{\boldsymbol{\beta}}_k)\}^{-1} \sum_{j=1}^{J} D\text{pl}_{kj}(\widehat{\boldsymbol{\beta}}_k) D\text{pl}_{lj}(\widehat{\boldsymbol{\beta}}_l)^T \{D^2\text{pl}_l(\widehat{\boldsymbol{\beta}}_l)\}^{-1}$$

where D and D^2 are the same first- and second-order numerical derivatives functions defined in *Variance estimation using the profile log-pseudolikelihood function*.

Acknowledgments

The development of `stmgintcox` was partially supported by SBIR awards 1R43CA233159-01 and 2R44CA233159-02. We thank our consultants Danyu Lin of the University of North Carolina at Chapel Hill and Donglin Zeng of the University of Michigan for their contributions to the command.

References

Chang, I.-S., C.-C. Wen, and Y.-J. Wu. 2007. A profile likelihood theory for the correlated gamma-frailty model with current status family data. *Statistica Sinica* 17: 1023–1046.

Chen, M.-H., L.-C. Chen, K.-H. Lin, and X. Tong. 2014. Analysis of multivariate interval censoring by diabetic retinopathy study. *Communications in Statistics—Simulation and Computation* 43: 1825–1835. https://doi.org/10.1080/03610918.2012.745557.

Chen, M.-H., X. Tong, and J. Sun. 2009. A frailty model approach for regression analysis of multivariate current status data. *Statistics in Medicine* 28: 3424–3436. https://doi.org/10.1002/sim.3715.

Cox, D. R. 1972. Regression models and life-tables (with discussion). *Journal of the Royal Statistical Society*, B ser., 34: 187–220. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x.

———. 1975. Partial likelihood. *Biometrika* 62: 269–276. https://doi.org/10.1093/biomet/62.2.269.

Finkelstein, D. M., and R. A. Wolfe. 1985. A semiparametric model for regression analysis of interval-censored failure time data. *Biometrics* 41: 933–945. https://doi.org/10.2307/2530965.

Huang, J., and J. A. Wellner. 1997. "Interval censored survival data: A review of recent progress". In *Proceedings of the First Seattle Symposium in Biostatistics: Survival Analysis*, edited by D. Y. Lin and T. R. Fleming, 123–169. New York: Springer. https://doi.org/10.1007/978-1-4684-6316-3_8.

Lindsey, J. C., and L. M. Ryan. 1998. Methods for interval-censored data. *Statistics in Medicine* 17: 219–238. https://doi.org/10.1002/(SICI)1097-0258(19980130)17:2<219::AID-SIM735>3.0.CO;2-O.

Murphy, S. A., and A. W. van der Vaart. 2000. On profile likelihood. *Journal of the American Statistical Association* 95: 449–465. https://doi.org/10.2307/2669386.

Odell, P. M., K. M. Anderson, and R. B. D'Agostino. 1992. Maximum likelihood estimation for interval-censored data using a Weibull-based accelerated failure time model. *Biometrics* 48: 951–959. https://doi.org/10.2307/2532360.

Sun, J. 2006. *The Statistical Analysis of Interval-Censored Failure Time Data*. New York: Springer. https://doi.org/10.1007/0-387-37119-2.

The ARIC investigators. 1989. The Atherosclerosis Risk in Communities (ARIC) study: Design and objectives. *American Journal of Epidemiology* 129: 687–702. https://doi.org/10.1093/oxfordjournals.aje.a115184.

Turnbull, B. W. 1976. The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society*, B ser., 38: 290–295. https://doi.org/10.1111/j.2517-6161.1976.tb01597.x.

Wen, C.-C., and Y.-H. Chen. 2013. A frailty model approach for regression analysis of bivariate interval-censored survival data. *Statistica Sinica* 23: 383–408.

Xu, Y., D. Zeng, and D. Y. Lin. 2023. Marginal proportional hazards models for multivariate interval-censored data. *Biometrika* 110: 815–830. https://doi.org/10.1093/biomet/asac059.

Zeng, D., F. Gao, and D. Y. Lin. 2017. Maximum likelihood estimation for semiparametric regression models with multivariate interval-censored data. *Biometrika* 104: 505–525. https://doi.org/10.1093/biomet/asx029.

Zeng, D., L. Mao, and D. Y. Lin. 2016. Maximum likelihood estimation for semiparametric transformation models with interval-censored data. *Biometrika* 103: 253–271. https://doi.org/10.1093/biomet/asw013.

Also see

[ST] **stmgintcox postestimation** — Postestimation tools for stmgintcox

[ST] **PH plots (interval-censored)** — PH-assumption plots for interval-censored data

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[ST] **stcox** — Cox proportional hazards model

[ST] **stintcox** — Cox proportional hazards model for interval-censored survival-time data

[ST] **stintreg** — Parametric models for interval-censored survival-time data

[U] **20 Estimation and postestimation commands**