stintcox — Cox proportional-hazards model for interval-censored survival-time data

Description | Quick start | Menu | Syntax
Options | Remarks and examples | Stored results | Methods and formulas
Acknowledgments | References | Also see

Description

`stintcox` fits semiparametric Cox proportional-hazards models to interval-censored survival-time data or, more precisely, event-time data, which may contain right-censored, left-censored, and interval-censored observations. `stintcox` can be used with single- or multiple-record interval-censored data. With interval-censored data, the event-time variables are specified with the `stintcox` command instead of using `stset`. All `st` settings are ignored by `stintcox`.

Quick start

Single-record-per-subject interval-censored data

Cox proportional-hazards model with covariates `x1` and `x2` fit to interval-censored data with lower and upper interval endpoints `t1` and `t2`

```
stintcox x1 x2, interval(t1 t2)
```

Same as above, but estimate the baseline hazard function using all observed intervals instead of the default reduced set

```
stintcox x1 x2, interval(t1 t2) full
```

Use less stringent convergence criteria to explore initial results more quickly

```
stintcox x1 x2, interval(t1 t2) favorspeed
```

Fit a stratified Cox model with strata defined by levels of `svar`

```
stintcox x1 x2, interval(t1 t2) strata(svar)
```

Include a time-varying covariate by interacting `x2` with the logarithm of analysis time

```
stintcox x1 x2, interval(t1 t2) tvc(x2) texp(ln(_t))
```

Report the log-likelihood model test instead of the default Wald model test, and report regression coefficients instead of hazard ratios

```
stintcox x1 x2, interval(t1 t2) lrmodel nohr
```

Report OIM standard errors instead of the default OPG standard errors

```
stintcox x1 x2, interval(t1 t2) vce(oim)
```

After estimation, report regression coefficients instead of hazard ratios

```
stintcox, nohr
```

After estimation, report OPG standard errors using fixed step size instead of the default adaptive step size

```
stintcox, vce(opg, stepsize(fixed))
```

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

```
stintcox, saving(basehc)
estimates store intcox
```

Multiple-record-per-subject interval-censored data

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

```
stintcox x1 x2, id(idvar) 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 two examination times

```
stintcox x1 x2, id(idvar) time(tvar) status(status)  ///
  tvcovimpute(nearright)
```

Report robust standard errors instead of the default OPG standard errors

```
stintcox x1 x2, id(idvar) time(tvar) status(status) vce(robust)
```

Report cluster–robust standard errors with the cluster identifier `clustvar`

```
stintcox x1 x2, id(idvar) time(tvar) status(status)  ///
  vce(cluster clustvar)
```

Menu

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

Syntax

Single-record-per-subject interval-censored data

```
stintcox [ indepvars ] [ if ] [ in ], interval(t_l t_u) [ single_options ]
```

Multiple-record-per-subject interval-censored data

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

single_options	Description
Model	
* interval(t_l t_u)	specify lower and upper endpoints for the event-time interval
options	options for both single- and multiple-record interval-censored data

*interval(t_l t_u) is required for single-record interval-censored data and cannot be combined with id(), time(), status(), or tvcovimpute().

multiple_options	Description
Model	
† id(*idvar*)	specify multiple-record ID variable
† time(*timevar*)	specify examination time variable
† status(*statusvar*)	specify event status indicator variable
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 interval-censored data

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

options	Description
Model	
`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)`
`lrphtest`	perform the likelihood-ratio test for covariates interacted with time; default is to perform Wald test
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be one of `opg` (the default), `oim`, `robust`, or `cluster` *clustvar*; may be specified on replay of results
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`lrmodel`	perform the likelihood-ratio model test instead of the default Wald model test
`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
`[no]log`	display or suppress EM and VCE iteration logs; default is `log`
`dots`	display all EM and VCE iterations as dots
`[no]vcedots`	display or suppress VCE iteration dots 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`	display or suppress the log-likelihood value of the EM algorithm for every 100 iterations; default is `emlog`
`emlog(`#`)`	display the log-likelihood value of the EM algorithm every # iterations; default is `emlog(100)`
`[no]emdots`	suppress or display EM iteration dots; default is `noemdots`
`emdots(`#`)`	display a dot every # EM iteration; 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

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 likelihood; 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
coeflegend	display legend instead of statistics

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.

vcetype	Description
opg[, *vce_options*]	outer product of the gradient (OPG) vectors; the default
oim[, *vce_options*]	observed information matrix (OIM)
robust[, *vce_options*]	Huber/White/sandwich estimator
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-likelihood tolerance to compute VCE; default is tolerance(1e-6)
[no]dots[(#)]	synonym for vcedots, vcedots(), and novcedots
post	replace the current e(V) with the specified VCE type; can be used only on replay with opg, oim, robust, or cluster

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

Options

Model

interval(t_l t_u) is required with single-record-per-subject interval-censored data; see *Single- versus multiple-record 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 id(), 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.

`id(`*idvar*`)` is required with multiple-record-per-subject interval-censored data; see *Single- versus multiple-record interval-censored data formats* in *Remarks and examples*. 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.

When `id()` is not specified, each observation is assumed to represent a different subject and thus constitutes a single-record-per-subject dataset.

When you specify `id()`, the data are said to be multiple-record-per-subject data, even if it turns out that there is only one record per subject. Multiple-record-per-subject data can be used to accommodate time-varying covariates that exist in the dataset.

If you specify `id()`, `stintcox` requires that you also specify `time()` and `status()`. `id()` may not be combined with the `interval()` option.

`time(`*timevar*`)` is required with multiple-record-per-subject interval-censored data; see *Single- versus multiple-record 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-subject interval-censored data; see *Single- versus multiple-record 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 subjects; see the description of `statussysmissok`.

Recurrent events are not allowed. After the first event occurs, the subject is removed from the analysis, even if the subject has subsequent records in the data.

`time()` and `status()` together define the lower and upper endpoints of the event-time intervals (t_l and t_u) and the censoring types in a multiple-record-per-subject interval-censored dataset. If the event of interest occurs before the first examination time for a subject, the subject is left-censored. In the first record for this subject, time is the first examination time, and the event-status indicator is 1. The corresponding event-time interval 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 subject is interval-censored. The event-time interval for this subject 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 subject is right-censored. The event-time interval for this subject has the last examination time as t_l and missing time (`.`) as t_u.

`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. This allows the estimator of the cumulative baseline hazard function to change its values only at the endpoints of the innermost time intervals, which 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. 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. 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`, `stintcox` uses less stringent convergence criteria to obtain the results. Specifically, it assumes lower EM coefficient, likelihood, and 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))`.

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

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

For all three imputation methods above, 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-subject format. 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, `stintcox` omits system missing observations in `status()` from estimation, like any other missing value in any of the specified variables. Observations that contain extended missing values (`.a` through `.z`) are always omitted during estimation. `statussysmissok` is useful to create time-varying covariates in a multiple-record-per-subject format; see *Single- versus multiple-record 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 *Testing the proportional hazards assumption using the tvc() option* 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 option `tvc()` 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.

`lrphtest` is used in conjunction with `tvc(`*varlist*$_t$`)`. It performs the likelihood-ratio test between the full model and the model without covariates interacted with time, that is, without specifying the `tvc()` option. By default, the Wald test of coefficients on time-varying covariates equal to zero is reported when `tvc(`*varlist*$_t$`)` is specified.

`tvc(`*varlist*$_t$`)`, `texp(`*exp*`)`, and `lrphtest` are explained more in *Testing the proportional hazards assumption using the tvc() option* in *Remarks and examples*.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error estimate 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 `post` is specified. *vcetype* may be one of the following:

`vce(opg[ , `*vce_options*`])` uses the sum of the OPG vectors based on the profile log likelihood; see *Methods and formulas*. `vce(opg)` is the default.

`vce(oim[ , `*vce_options*`])` uses the sum of the OIM vectors based on the profile log likelihood; see *Methods and formulas*.

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

`vce(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 likelihood; 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 with methods `opg`, `oim`, `robust`, or `cluster`. 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 the number of 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 default is `search(interpolate)`. In some cases, such as when factor variables have highly unbalanced levels, the search may lead to the step size that is too small or too large, which may lead to the error message that the estimates of baseline hazard contributions cannot be computed because the VCE matrix is close to being singular. Trying `search(bracket)` may be helpful in this case.

`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 likelihood. The default is `iterate(5000)`.

`tolerance(#)` specifies the tolerance for the profile log likelihood 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. The dots are displayed by default, but you can use `nodots` to suppress them; see `set iterlog` in [R] ***set iter***. By default, the dot is displayed every iteration, but you can change this by specifying `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 with `vce(opg)`, `vce(oim)`, `vce(robust)`, or `vce(cluster` *clustvar*`)`. It replaces the current `e(V)` with the specified *vcetype*. When `vce(`*vcetype*`)` is used on replay without `post`, the coefficient table will display the standard error of the specified *vcetype*, but `e(V)` will remain unchanged. This option does not appear in the dialog box.

Reporting

`level(#)`, `lrmodel`; 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, `stintcox` saves estimation results in a temporary file for later access by postestimation commands. This temporary file will be overridden every time `stintcox` 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.

`log` and `nolog` display or suppress `stintcox`'s iteration log, which includes both the EM and VCE iterations. The EM iteration log displays the log-likelihood value every 100 iterations (the `emlog` option). The VCE iteration log displays iterations as dots (the `vcedots` option). `log`, the default, implies `emlog` and `vcedots`. `nolog` suppresses both EM and VCE iteration logs; see `set iterlog` in [R] ***set iter***. `nolog` is equivalent to specifying `noemlog` and `novcedots`. `log` and `nolog` may not be combined with any other options that control an iteration log.

`dots` implies the `emdots` and `vcedots` options to display both the EM and VCE iteration logs as dots. The VCE iteration log is always displayed as dots. However, the EM iteration log, by default, displays the log-likelihood value every 100 iterations (the `emlog` option). To display the EM iterations as dots, you can specify `dots` or `emdots`. `dots` may not be combined with any other options that control an iteration log.

`vcedots`, `vcedots(#)`, and `novcedots` are synonyms for `vce(, dots)`, `vce(, dots(#))`, and `vce(, nodots)`, respectively.

`emlog`, `emlog(#)`, and `noemlog` display or suppress an iteration log showing the progress of the EM algorithm. The log is displayed by default, and `noemlog` suppresses it; see `set iterlog` in [R] ***set iter***. `emlog`, the default, displays the log-likelihood value every 100 iterations and is equivalent to `emlog(100)`. `emlog(#)` displays the log-likelihood value every #th iterations.

noemdots, emdots(#), and emdots control the display of the EM iteration log as dots. By default, the EM iteration log displays the log-likelihood 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.

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).

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

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

Remarks and examples

Remarks are presented under the following headings:

Introduction
Single- versus multiple-record interval-censored data formats
Case II interval-censored data
Time-varying covariates
Standard error estimation with interval-censored data
Current status or case I interval-censored data
Testing the proportional hazards assumption using the tvc() option

Introduction

stintcox fits the Cox proportional-hazards model to interval-censored survival-time data. In the context of interval-censored data, the term "failure-time data" or "event-time data" is more appropriate, so we will use it in that context.

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] **stintreg** 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), Rabinowitz, Tsiatis, and Aragon (1995), Huang and Wellner (1997), Lindsey (1998), Lindsey and Ryan (1998), Sun (2006), and Sun and Li (2014).

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.

Several authors (Cai and Betensky 2003; Zhang, Hua, and Huang 2010; Wang et al. 2016) have proposed spline methods to fit the Cox proportional-hazards model to interval-censored data. Spline methods have limitations, however. First, the choices for the type of spline and the number and positions of knots are arbitrary, and different choices may yield conflicting results. Second, the analysis will be biased if the event-time distribution is not well approximated by the chosen spline function. Finally, the variance estimation is difficult given the data-dependent choices of spline functions (Zhang, Hua, and Huang 2010).

Direct maximum-likelihood optimization for the Cox model with interval-censored data using, for instance, the Newton–Raphson algorithm is highly unstable (Sun 2006; Finkelstein 1986).

Zeng, Mao, and Lin (2016) developed a novel EM algorithm for efficient nonparametric maximum-likelihood estimation (NPMLE) of the Cox proportional-hazards model with 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. And it reduces to the classical maximum partial-likelihood estimation in the special case of right-censored data. For more details about this method, see *Methods and formulas*.

Unlike with right-censored data, the estimation of regression coefficients must be performed jointly with estimation of the baseline cumulative hazard function for interval-censored data. Stata provides two ways to estimate the baseline cumulative hazard function. One is to use all distinct lower and upper interval endpoints as time points for estimating the baseline cumulative hazard function. This is available by specifying the `full` option.

For large datasets with many distinct time points, this approach may become time consuming. An alternative is to estimate the baseline cumulative hazard at fewer time points. Turnbull (1976) proposed a method for estimating the one-sample survivor function at a subset of time intervals, known as Turnbull's intervals, or innermost intervals, or regions of the maximal cliques. Thus, one can allow the baseline cumulative hazard function to change its values only at the endpoints of those time intervals and set baseline hazard contributions to zero for the other times. This is available via the `reduced` option and, for computational reasons, is the default in `stintcox`.

As mentioned above, NPMLE is a computationally intensive approach, so `stintcox` may take some time to run, especially for large datasets. The speed of the command depends on the desired accuracy of the computations, among other things. The higher the accuracy, the more iterations are needed to achieve that accuracy, and thus the longer the command runs. It is important to have high accuracy for the final reporting of the results, but the speed may become an issue during the exploratory stage of

the project. Thus, you may consider using the favorspeed option to expedite the command execution. When you specify this option, stintcox uses less stringent convergence criteria to produce the results more quickly; see the description of the favorspeed option for details.

Unlike many st commands, stintcox requires that you specify event-time ("survival") information directly with the command instead of using stset. Typing stset is unnecessary, and stintcox will ignore any settings of stset for the usual trivariate response variable (t_0, t, d). Event-time information can be specified with stintcox in two different ways depending on the storage format of the interval-censored event-time data. The two different storage formats are single-record-per-subject format (or time-intervals format) and multiple-record-per-subject format (or examination-times format). With single-record-per-subject data, you must specify two variables containing time intervals for each subject in stintcox's interval() option. With multiple-record-per-subject data, you must specify subjects' identifiers in the id() option, examination times in the time() option, and event-status indicators at each examination time in the status() option. See *Single- versus multiple-record interval-censored data formats* for details.

stintcox supports time-varying (time-dependent) covariates. You can use the tvc() option to more easily include time-varying covariates that are formed by multiplying covariates specified in tvc() with a deterministic function of time specified in the texp() option. In a multiple-record-per-subject format, you can include more general time-varying covariates; see *Single- versus multiple-record interval-censored data formats* and *Time-varying covariates*. In the presence of general time-varying covariates, stintcox offers several methods for imputing the values of these covariates between the examination times; see the description of the tvcovimpute() option for details.

stintcox does not support data exhibiting delayed entry, gaps, and multiple failures.

Single- versus multiple-record interval-censored data formats

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

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) for a subject. We call data stored in this format "multiple-record-per-subject interval-censored event-time data" or "multiple-record interval-censored data". This format is common for what is called "case I interval-censored data or current status data" and is often used to accommodate time-varying covariates.

stintcox supports both formats.

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

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

Here variables ltime and rtime record the respective lower, t_l, and upper, t_u, endpoints of the event-time interval for each subject. stintcox requires that we specify these variables in the interval() option:

```
. stintcox x1 x2 x3, 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, subject 101 is left-censored, subject 102 is interval-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 interval-censored data. We can provide the same event-time information as in the previous dataset in the following format:

id	time	status	x1	x2	x3
101	6	1	17	22	0
102	4	0	12	22	1
102	9	1	12	22	1
103	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. In this format, `stintcox` requires that we specify the subject identifier (`id`) in the `id()` option, examination time (`time`) in the `time()` option, and the event status indicator (`status`) in the `status()` option.

```
. stintcox x1 x2 x3, id(id) 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 subject. In our example, 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 two examination times, 4 and 9, and the event occurred by time 9. Subject 102 is interval-censored with the time interval $(4, 9]$. 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 $(13, +\infty)$. Uncensored observations are not allowed in this format.

In this format, we can easily record time-varying covariates. For instance, suppose that subject 102 had an additional examination at time 6 with a status of 0 and the value for covariate `x3` was recorded to be 0. `x3` is no longer constant within subject 102 but varies discretely with time. (The event-time interval for this subject also changed to $(6, 9]$! More about this below.) We can accommodate varying `x3` by adding an extra record for subject 102 at time 6:

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

Potential caveat when creating time-varying covariates with interval-censored data. In the above, when we added an extra record for subject 102 at time 6 with status equal 0, we modified the actual event-time interval for this subject from the original $(4, 9]$ to $(6, 9]$. This is not a problem if this new record we added corresponds to the actual examination time for this subject. Sometimes, how-

ever, we may need to add extra records to specify varying covariate values at intermediate times without changing the event-time information for subjects. This is useful, for instance, when we create more complicated functions of time and covariates to assess the proportional hazards assumption.

Returning to our previous example, how can we incorporate the varying values of covariate x3 without changing the event-time interval for subject 102? We can replace the zero status with a system missing value (.) in the new observation and use the statussysmissok option with stintcox to specify that this observation should be included in the estimation but should not be used to determine the subject's event-time interval and censoring information:

id	time	status	x1	x2	x3
101	6	1	17	22	0
102	4	0	12	22	1
102	6	.	12	22	0
102	9	1	12	22	1
103	13	0	13	22	0

and

```
. stintcox x1 x2 x3, id(id) time(time) status(status) statussysmissok ...
```

Without the statussysmissok option, the record with a system missing status value would have been omitted from the analysis entirely, like any missing value in any of the specified variables. If you need to omit any values in the status() variable from your analysis in the presence of statussysmissok, you should record them as extended system missing values (.a through .z).

For an example of fitting stintcox to multiple-record interval-censored data, see *Time-varying covariates*. The multiple-record-per-subject format is also convenient for fitting current status data; see *Current status or case I interval-censored data*.

Case II interval-censored data

Case II interval-censored data arise when there are potentially two or more examination times for each study subject. With baseline covariates, case II interval-censored data are typically recorded in a single-record-per-subject format. In the presence of time-varying covariates in the dataset, a multiple-record-per-subject format is used; see *Time-varying covariates*.

In a single-record-per-subject format, the interval that brackets the event time of interest, the event-time interval, is recorded for each subject. 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 a truly interval-censored observation.

▷ Example 1: Single-record-per-subject case II interval-censored data

Zeng, Mao, and Lin (2016) considered a cohort study of injecting drug users in Thailand. Subjects were initially seronegative for the HIV-1 virus. They were followed and assessed for HIV-1 seropositivity through blood tests approximately every four months. The event of interest was time to HIV-1 seropositivity. Because the subjects were tested approximately every four months, the exact time of HIV-1 seropositivity was not observed but was known to fall only in the interval between blood tests.

The data used in this example are the data provided in supplementary materials of Zeng, Mao, and Lin (2016), which are based on the study described above. The dataset contains 1,124 subjects: 76 are females and 1,048 are males. We wish to identify the factors that influence HIV-1 infection. The covariates that we are interested in are age at recruitment (age), sex (male), history of needle sharing

(needle), history of drug injection before recruitment (inject), and whether a subject has been in jail at the time of recruitment (jail). The dataset also contains two variables, ltime and rtime, that record, respectively, the last time of blood test when the HIV-1 was seronegative and the first time of blood test when the HIV-1 was seropositive.

```
. use https://www.stata-press.com/data/r19/idu
(Modified Bangkok IDU Preparatory Study)
. describe
Contains data from https://www.stata-press.com/data/r19/idu.dta
 Observations:         1,124                  Modified Bangkok IDU Preparatory
                                                Study
    Variables:             8                  15 Dec 2024 13:34
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
age             byte    %8.0g                 Age (in years)
male            byte    %8.0g      yesno      Male
needle          byte    %8.0g      yesno      Shared needles
jail            byte    %8.0g      yesno      Imprisoned at recruitment
inject          byte    %8.0g      yesno      Injected drugs before recruitment
ltime           double  %10.0g                Last time seronegative for HIV-1
rtime           double  %10.0g                First time seropositive for HIV-1
age_mean        double  %10.0g                Centered age (in years)
------------------------------------------------------------------------------
Sorted by:
```

We want to use stintcox to fit the Cox proportional-hazards model in which the time to HIV-1 infection depends on age, male, needle, inject, and jail. To make the interpretation of the baseline hazard function more meaningful, we will use the centered age variable, age_mean. (Remember that a baseline hazard function corresponds to all covariates equal to zero, and age of zero would not make sense for our sample of subjects. See *Making baseline reasonable* in [ST] **stcox postestimation** for more details.)

Unlike stcox's specification, in which the survival variables are set using stset and do not appear in the command, for single-record-per-subject, the interval time variables ltime and rtime must be specified in the stintcox's option interval(). Also, recall that the command implements an NPMLE method, which is computationally intensive, so it may take a little longer to run on our dataset of 1,124 observations:

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -597.65634
Iteration 200: Log likelihood = -597.57555
Iteration 295: Log likelihood = -597.56443

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

Interval-censored Cox regression                  Number of obs     =  1,124
Baseline hazard: Reduced intervals                       Uncensored =      0
                                                   Left-censored =     41
Event-time interval:                               Right-censored =    991
  Lower endpoint: ltime                            Interval-cens. =     92
  Upper endpoint: rtime
                                                  Wald chi2(5)      =  17.10
Log likelihood = -597.56443                       Prob > chi2       = 0.0043
```

	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf.	interval]
age_mean	.9684341	.0126552	-2.45	0.014	.9439452	.9935582
male						
Yes	.6846949	.1855907	-1.40	0.162	.4025073	1.164717
needle						
Yes	1.275912	.2279038	1.36	0.173	.8990401	1.810768
inject						
Yes	1.250154	.2414221	1.16	0.248	.8562184	1.825334
jail						
Yes	1.567244	.3473972	2.03	0.043	1.014982	2.419998

```
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 summarizes censored observations. Because this is a single-record-per-subject dataset, the number of observations equals the number of subjects. There are 991 subjects who did not test positive for HIV-1 by the last visit, resulting in right-censored observations. There are 41 subjects who tested positive for HIV-1 at their first follow-up, resulting in left-censored observations. The remaining 92 subjects are interval-censored.

As seen in the table, age is associated with lower risk of HIV-1 infection, and being in jail at enrollment is associated with higher risk of HIV-1 infection. Being a female, needle sharing, and history of drug injection are associated with the higher risk of HIV-1 infection, although we do not have strong evidence that the hazard ratios for these variables are different from 1.

The command displays a note following the command 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*.

`stintcox` uses the EM algorithm to estimate parameters. EM algorithms are known to require many iterations. Thus, by default, `stintcox` displays log-likelihood values every 100 iterations. You can change how often to display the iteration log by specifying the `emlog(#)` option. For example, `emlog(1)` will display every iteration. You can also use the `noemlog` option to suppress the iteration log.

Similarly, the progress of the VCE computation is displayed with a dot for each iteration. With more regression coefficients and with larger datasets, you may see many more dots. In this case, you may consider displaying a dot every # iterations by specifying `vcedots(#)`. To suppress the dots, use `novcedots`. ◁

▷ Example 2: Speed versus accuracy

By default, `stintcox` uses Stata's standard convergence rules for estimation of parameters. For instance, the parameter tolerance of 1e–6 is used as a stopping rule, and the Hessian scale-gradient tolerance of 1e–5 is used to check for convergence. More stringent criteria typically require more iterations and thus lead to longer execution times.

Although high accuracy of the results is important for final reporting, it may be reasonable to consider less stringent criteria during exploratory analysis in favor of speed. `stintcox` provides the `favorspeed` option for this.

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
> favorspeed
note: using fixed step size with a multiplier of 5 to compute derivatives.
note: using EM and VCE tolerances of 0.0001.
note: option noemhsgtolerance assumed.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 32:  Log likelihood = -598.60293

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

Interval-censored Cox regression                  Number of obs     =  1,124
Baseline hazard: Reduced intervals                       Uncensored =      0
                                                   Left-censored =     41
Event-time interval:                               Right-censored =    991
  Lower endpoint: ltime                            Interval-cens. =     92
  Upper endpoint: rtime
                                                  Wald chi2(5)      =  16.95
Log likelihood = -598.60293                       Prob > chi2       = 0.0046
```

	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf.	interval]
age_mean	.9684228	.0126481	-2.46	0.014	.9439476	.9935326
male						
Yes	.6853044	.1873617	-1.38	0.167	.4010197	1.17112
needle						
Yes	1.275045	.2272609	1.36	0.173	.899103	1.80818
inject						
Yes	1.251637	.2414583	1.16	0.245	.8575707	1.826784
jail						
Yes	1.566873	.3479667	2.02	0.043	1.013914	2.421398

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

The above takes only a few seconds to run. And the results happen to be very similar to the results from the previous two examples.

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

◁

▷ Example 3: Full versus reduced sets of time intervals

By default, `stintcox` uses a reduced set of intervals (the `reduced` option) to estimate the baseline hazard function. You can specify the `full` option to estimate the baseline hazard using all time intervals. This approach may be much more time consuming, especially for large datasets. For demonstration purposes, we will also use the `favorspeed` option from example 2 to speed up execution.

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime) full
> favorspeed
note: using fixed step size with a multiplier of 5 to compute derivatives.
note: using EM and VCE tolerances of 0.0001.
note: option noemhsgtolerance assumed.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -951.11659
Iteration 42:  Log likelihood = -599.05659

Computing standard errors: ..... done

Interval-censored Cox regression                  Number of obs     =  1,124
Baseline hazard: All intervals                          Uncensored  =      0
                                                    Left-censored   =     41
Event-time interval:                                Right-censored  =    991
  Lower endpoint: ltime                             Interval-cens.  =     92
  Upper endpoint: rtime
                                                  Wald chi2(5)      =  16.94
Log likelihood = -599.05659                       Prob > chi2       = 0.0046
```

	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf.	interval]
age_mean	.9685754	.0126454	-2.45	0.014	.9441053	.9936797
male						
Yes	.6845339	.1871941	-1.39	0.166	.4005194	1.169947
needle						
Yes	1.276357	.2276023	1.37	0.171	.8998792	1.810339
inject						
Yes	1.252501	.2416295	1.17	0.243	.8581566	1.828058
jail						
Yes	1.566614	.3479196	2.02	0.043	1.013734	2.42103

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

The results are very similar between the two ways of estimating the baseline hazard in this example. And EM optimization iterations change from 32 to 42 when the `full` option is specified.

◁

Time-varying covariates

There are two ways to incorporate time-varying covariates with stintcox. 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 tvc(), stintcox provides a more memory-efficient computation and automatically incorporates the specified function of time in various calculations during estimation and postestimation.

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

▷ Example 4: Multiple-record-per-subject case II interval-censored data

The dataset used in this example is an extended version of the dataset described in example 1; it has the imprisonment indicator (jail_vary) that varies with time. The data are recorded in the multiple-record-per-subject format, where each observation records a subject identifier (id), the examination time of the blood test (time), whether the blood test changes to positive for HIV-1 since last examination time (is_seropos), and whether the subject has been imprisoned since the last clinic visit (jail_vary), which is a time-varying covariate. Each observation also records baseline (time-invariant) factors, such as centered age at recruitment (age_mean) and sex (male), as described in example 1.

Here is a subset of the dataset:

```
. use https://www.stata-press.com/data/r19/idu2
(Modified Bangkok IDU Preparatory Study with time-varying variable jail_vary)
. format time age_mean %6.2f
. list id time is_seropos age_mean male needle inject jail_vary
> if id >= 271 & id <= 274, sepby(id) noobs compress abbreviate(10)
```

id	time	is_seropos	age_mean	male	needle	inject	jail_vary
271	4.89	No	-6.46	Yes	Yes	No	No
271	9.31	No	-6.46	Yes	Yes	No	No
271	13.38	No	-6.46	Yes	Yes	No	Yes
271	17.97	No	-6.46	Yes	Yes	No	Yes
271	22.00	No	-6.46	Yes	Yes	No	No
272	3.80	No	8.54	No	No	No	Yes
272	9.41	Yes	8.54	No	No	No	No
273	3.93	No	-11.46	Yes	Yes	No	No
273	8.00	No	-11.46	Yes	Yes	No	No
273	12.07	No	-11.46	Yes	Yes	No	Yes
273	15.97	No	-11.46	Yes	Yes	No	Yes
273	20.66	No	-11.46	Yes	Yes	No	Yes
274	3.87	Yes	-4.46	Yes	Yes	Yes	Yes

There are subjects with multiple records. The examination time and event status is_seropos define subjects' event-time intervals and censoring types. For example, subject 271 never tested positive for HIV-1 through all his examinations. He is right-censored with an event-time interval $(22, +\infty)$. Subject

272 tested negative at her first clinic visit but then tested positive at her second clinic visit. She is interval-censored with an event-time interval $(3.80, 9.41]$. Like subject 271, subject 273 is right-censored with an event-time interval $(20.66, +\infty]$. Finally, subject 274 tested positive at his first clinic visit, so he is left-censored with an event-time interval $(0, 3.87]$.

We use `stintcox` to fit a Cox proportional-hazards model in which the time to HIV-1 infection depends on baseline covariates `age_mean`, `male`, `needle`, `inject`, and time-varying covariate `jail_vary`. In a multiple-record-per-subject format, we must specify options `id()`, `time()`, and `status()` with `stintcox`.

```
. stintcox age_mean i.male i.needle i.inject i.jail_vary, id(id) time(time)
> status(is_seropos)
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.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -598.45375
Iteration 200: Log likelihood = -598.35872
Iteration 285: Log likelihood = -598.34887

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

Interval-censored Cox regression                  Number of obs      =  6,453
Baseline hazard: Reduced intervals                Number of subjects =  1,124
                                                         Uncensored =      0
ID variable: id                                     Left-censored =     41
Examination time: time                             Right-censored =    991
Status indicator: is_seropos                       Interval-cens. =     92

                                                  Wald chi2(5)       =  17.03
Log likelihood = -598.34887                       Prob > chi2        = 0.0044
```

time	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf.	interval]
age_mean	.9714605	.012757	-2.20	0.027	.9467762	.9967884
male						
Yes	.6678044	.1816576	-1.48	0.138	.3918353	1.138138
needle						
Yes	1.271409	.2275426	1.34	0.180	.8952546	1.805609
inject						
Yes	1.370672	.2575405	1.68	0.093	.9484142	1.980928
jail_vary						
Yes	1.440966	.2916178	1.81	0.071	.9691488	2.142481

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

There are 6,453 observations on 1,124 subjects, but the censoring information (as expected) is the same as in example 1. `stintcox` identified one time-varying covariate in this dataset—`jail_vary`.

Compared with example 1, after we account for time-varying imprisonment, the hazard ratio for drug injection `inject` increases from 1.25 to 1.37, but the effect of imprisonment decreases from 1.57 for baseline `jail` to 1.44 for time-varying `jail_vary`.

◁

Standard error estimation with interval-censored data

With interval-censored data, the NPMLE approach estimates regression coefficients jointly with the contributions to the baseline cumulative hazard function, which is infinite dimensional. The inverse of the entire information matrix for all the parameters, which is typically used to estimate standard errors, does not provide a valid estimator of the variance–covariance matrix in this case.

Murphy and van der Vaart (2000) and Zeng, Mao, and Lin (2016) propose to estimate the VCE for regression coefficients using the profile log likelihood, which is obtained by maximizing the likelihood by holding the regression coefficients fixed. Four different methods based on this profile log likelihood are offered to calculate the VCE for regression coefficients. The `vce(opg)` method, the default, uses the first-order numerical derivatives, whereas the `vce(oim)` method uses the second-order numerical derivatives. When there are sufficient data to estimate the second-order derivatives reliably, the OIM method will generally provide more accurate results. With small samples, however, Zeng, Gao, and Lin (2017) found that the OIM method may lead to a negative definite matrix of second-order derivatives, which is not invertible. In addition, `vce(robust)` provides a VCE that is robust to some types of misspecification, and `vce(cluster` *clustvar*`)` provides a VCE that allows for observations to be correlated within groups defined by *clustvar*.

For some datasets, the choice of the step size used to compute numerical derivatives may also affect the estimates. By default, `stintcox` uses an adaptive step size, `stepsize(adaptive)`, which is calculated from the data and updated during the computation of numerical derivatives. Zeng, Mao, and Lin (2016) and Zeng, Gao, and Lin (2017) found a fixed ad hoc step size of $\delta_n = 5n^{-1/2}$ to work well in the examples they considered. You can specify the `stepsize(fixed)` option to use that step size. And you can also provide your own multiplier instead of the above 5 by specifying `stepsize(fixed` #`)`.

When an adaptive step size is used, `stintcox` searches for an optimal step size to use to compute the numerical derivatives. In some cases, such as in the presence of covariates of different magnitudes, the search may lead to step sizes that are too large or too small such that the VCE matrix becomes close to being singular. In that case, you may consider trying a different search method, for example, `vce(, search(bracket))`, or using a fixed step size, `vce(, stepsize(fixed))`.

For small datasets or datasets with low proportions of interval-censored observations, the standard error estimates may be more variable between different estimation methods. In that case, you may want to compare several VCE estimation methods. `stintcox` provides the OPG, OIM, robust, and cluster–robust methods on replay so that you do not need to rerun the estimation command.

▷ Example 5: Variance estimation

Continuing with example 1, we note that the dataset contains only 92 interval-censored observations out of a total of 1,124 observations. Let's compare the OPG and OIM methods using both an adaptive and a fixed step size.

Let's refit our model using the default settings and save the estimation results for later comparisons.

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
```
(*output omitted*)

To save estimation results after stintcox, we must save the estimated baseline hazard contributions in a dataset before using estimates store because they are an integral part of the estimation results. This can be done after estimation, on replay, or we could have specified the saving() option during estimation above.

```
. stintcox, saving(basehc, replace)
note: file basehc.dta not found; file saved.
. estimates store opg_adapt
```

We do not need to refit the model to compute OPG and OIM VCEs. To compute OIM estimates using an adaptive step size, we simply type

```
. stintcox, vce(oim, post)
note: using adaptive step size to compute derivatives.
Computing standard errors: ............................................. done
Interval-censored Cox regression                  Number of obs     =  1,124
Baseline hazard: Reduced intervals                       Uncensored =      0
                                                     Left-censored =     41
Event-time interval:                                Right-censored =    991
  Lower endpoint: ltime                             Interval-cens. =     92
  Upper endpoint: rtime
                                                  Wald chi2(5)      =  17.10
Log likelihood = -597.56443                       Prob > chi2       = 0.0043
```

	Haz. ratio	OIM std. err.	z	P>|z|	[95% conf.	interval]
age_mean	.9684341	.0142537	-2.18	0.029	.9408965	.9967776
male						
Yes	.6846949	.2175931	-1.19	0.233	.3672728	1.276455
needle						
Yes	1.275912	.2688546	1.16	0.248	.8442274	1.928334
inject						
Yes	1.250154	.3363611	0.83	0.407	.7378069	2.118284
jail						
Yes	1.567244	.4679699	1.50	0.132	.8729166	2.813847

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

We also specified the post suboption with vce(oim) to store the oim estimates in e(V) and stored the updated estimation results.

We repeat the same steps using a fixed step size for oim,

```
. stintcox, vce(oim, stepsize(fixed) post)
  (output omitted)
. estimates store oim_fixed
```

and for opg,

```
. stintcox, vce(opg, stepsize(fixed) post)
  (output omitted)
. estimates store opg_fixed
```

We compare the results using `estimates table`:

```
. estimates table opg* oim*, b(%9.4f) se(%9.4f) t p
```

Variable	opg_adapt	opg_fixed	oim_adapt	oim_fixed
age_mean	-0.0321	-0.0321	-0.0321	-0.0321
	0.0131	0.0131	0.0147	0.0120
	-2.45	-2.46	-2.18	-2.67
	0.0141	0.0141	0.0293	0.0077
male				
Yes	-0.3788	-0.3788	-0.3788	-0.3788
	0.2711	0.2736	0.3178	0.2994
	-1.40	-1.38	-1.19	-1.27
	0.1623	0.1662	0.2333	0.2058
needle				
Yes	0.2437	0.2437	0.2437	0.2437
	0.1786	0.1784	0.2107	0.1824
	1.36	1.37	1.16	1.34
	0.1725	0.1719	0.2475	0.1817
inject				
Yes	0.2233	0.2233	0.2233	0.2233
	0.1931	0.1933	0.2691	0.1961
	1.16	1.16	0.83	1.14
	0.2476	0.2480	0.4066	0.2548
jail				
Yes	0.4493	0.4493	0.4493	0.4493
	0.2217	0.2221	0.2986	0.2379
	2.03	2.02	1.50	1.89
	0.0427	0.0431	0.1324	0.0589

Legend: b/se/t/p

As expected, the regression coefficient estimates are the same for all VCE methods. The standard error estimates are fairly similar across all methods but a little more variable for `oim_adapt`. This is not surprising because the OIM method is based on the second-order derivatives, which are estimated numerically and thus require higher tolerances to produce more accurate estimates. For instance, we can specify a slightly lower tolerance (1e–7 instead of the default 1e–6) for the `oim` method using an adaptive step size:

```
. stintcox, vce(oim, tolerance(1e-7)) nohr
note: using adaptive step size to compute derivatives.
Computing standard errors: ..................................................
> ..... done
Interval-censored Cox regression                  Number of obs     =  1,124
Baseline hazard: Reduced intervals                       Uncensored =      0
                                                    Left-censored =     41
Event-time interval:                                Right-censored =    991
  Lower endpoint: ltime                             Interval-cens. =     92
  Upper endpoint: rtime
                                                  Wald chi2(5)      =  17.10
Log likelihood = -597.56443                       Prob > chi2       = 0.0043
```

	Coefficient	OIM std. err.	z	P>|z|	[95% conf.	interval]
age_mean	-.0320749	.012257	-2.62	0.009	-.0560982	-.0080516
male						
Yes	-.378782	.288278	-1.31	0.189	-.9437965	.1862326
needle						
Yes	.2436616	.1848383	1.32	0.187	-.1186149	.605938
inject						
Yes	.2232666	.1981754	1.13	0.260	-.1651501	.6116832
jail						
Yes	.4493186	.2299995	1.95	0.051	-.0014721	.9001094

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

The standard errors are now closer to those of the other methods.

◁

Current status or case I interval-censored data

Current status data or case I interval-censored data arise when each study subject is examined only once, such that the event of interest is known to occur before or after the examination time, resulting in a left- or right-censored observation. These data can be viewed as a special case of interval-censored data without truly interval-censored observations. Current status data can be recorded in either single-record-per-subject format or multiple-record-per-subject format, but the latter is more common.

▷ Example 6: Case I interval-censoring

Sun (2006) investigated a dataset about calcification of the hydrogel intraocular lenses (IOL), a rarely reported complication of cataract treatment. The dataset contains 379 patients who had IOL implantation and were examined by an ophthalmologist. The `status` variable indicates the degree of severity of IOL calcification: 0 means no or little calcification, and 1 means mild or serious calcification. The study contains 237 females and 142 males. We want to test whether the IOL calcification is different between males and females.

Current status data usually contain two variables: one that records the examination time (`time`) and one that records the status of the event of interest (`status`). So we can analyze these data using the multiple-record-per-subject format, even though each subject has only one observation.

Alternatively, we can create two interval time variables based on examination times and event statuses and analyze the data using the single-record-per-subject format. We already have the respective interval variables, `ltime` and `rtime`, in our dataset, but see example 3 of [ST] **stintreg** on how to generate these variables.

In the following, we will demonstrate how to analyze current status data using both formats.

Let us fit a Cox proportional-hazards model on `gender` using the multiple-record-per-subject format first:

```
. use https://www.stata-press.com/data/r19/iol
(Hydrogel Intraocular Lenses (IOL) Study)
. stintcox i.gender, id(id) time(time) status(status) nohr
note: using adaptive step size to compute derivatives.
Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood =  -191.3847
Iteration 100: Log likelihood = -137.70653
Iteration 200: Log likelihood = -137.65739
Iteration 300: Log likelihood = -137.64799
Iteration 400: Log likelihood = -137.64509
Iteration 425: Log likelihood = -137.64472
Computing standard errors: ..... done
Interval-censored Cox regression                  Number of obs      =    379
Baseline hazard: Reduced intervals                Number of subjects =    379
                                                         Uncensored =      0
ID variable: id                                     Left-censored =     48
Examination time: time                             Right-censored =    331
Status indicator: status                           Interval-cens. =      0
                                                  Wald chi2(1)       =   0.52
Log likelihood = -137.64472                       Prob > chi2        = 0.4719
```

time	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
gender						
Male	-.2242553	.3117387	-0.72	0.472	-.835252	.3867415

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

We specified the `nohr` option to report the regression coefficient estimates instead of the hazard ratios for the independent variables to more easily compare results from the literature.

The above table reports $\hat{\beta} = -0.2243$ and its estimated standard error of 0.312. This yields a test of $\beta = 0$ with a p-value of 0.47, which suggests that there is no difference between males and females in terms of the time to IOL calcification.

Sun (2006) reports $\hat{\beta} = -0.2241$ with its standard error of 0.295, which are obtained by direct maximum-likelihood optimization. Our EM-based estimates are comparable.

Let's fit the same model but now use the single-record-per-subject format, where we specify the event-time intervals `ltime` and `rtime` in the `interval()` option:

```
. stintcox i.gender, interval(ltime rtime) nohr
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood =  -191.3847
Iteration 100: Log likelihood = -137.70653
Iteration 200: Log likelihood = -137.65739
Iteration 300: Log likelihood = -137.64799
Iteration 400: Log likelihood = -137.64509
Iteration 425: Log likelihood = -137.64472

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

Interval-censored Cox regression                  Number of obs     =    379
Baseline hazard: Reduced intervals                       Uncensored =      0
                                                     Left-censored =     48
Event-time interval:                                 Right-censored =    331
  Lower endpoint: ltime                              Interval-cens. =      0
  Upper endpoint: rtime
                                                  Wald chi2(1)      =   0.52
Log likelihood = -137.64472                       Prob > chi2       = 0.4719
```

	OPG					
	Coefficient	std. err.	z	P>|z|	[95% conf.	interval]
gender						
Male	-.2242553	.3117387	-0.72	0.472	-.835252	.3867415

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

We obtained the same results in the multiple-record-per-subject format.

◁

Testing the proportional hazards assumption using the tvc() option

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. For instance, consider a model with one baseline covariate x_1. We wish to include an interaction between the covariate and a function of time $g(t)$:

$$h(t) = h_0(t)\exp\{\beta_1 x_1 + g(t)\gamma_1 x_1\}$$

Rearranging terms results in

$$h(t) = h_0(t)\exp\left[\{\beta_1 + \gamma_1 g(t)\}\, x_1\right]$$

Given this new arrangement, we consider that $\beta_1 + \gamma_1 g(t)$ is a (possibly) time-varying coefficient on the covariate x_1 for some specified function of time $g(t)$. The coefficient has a time-invariant component, β_1, with γ_1 determining the magnitude of the time-varying deviations from β_1. Thus, a test of $\gamma_1 = 0$ is a test of time invariance for the coefficient on x_1. Proportional hazards imply that the relative hazard is fixed over time, and this assumption would be violated if a time interaction proved significant.

The above can be easily accomplished by using the `tvc()` option with `stintcox`, where you specify the covariates suspected to violate the proportional hazards assumption. By default, $g(t) = t$ is used, but you can use the `texp()` option to specify other functions of time.

▷ Example 7: Testing the proportional hazards assumption

Continuing with example 1, we now include time interaction terms for all covariates to test the proportional hazards assumption for individual covariates and globally:

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
> tvc(age_mean i.male i.needle i.inject i.jail) nohr
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -590.53655
Iteration 200: Log likelihood = -590.45163
Iteration 300: Log likelihood = -590.43665
Iteration 340: Log likelihood = -590.43386

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

Interval-censored Cox regression                  Number of obs     =  1,124
Baseline hazard: Reduced intervals                       Uncensored =      0
                                                     Left-censored =     41
Event-time interval:                                Right-censored =    991
  Lower endpoint: ltime                             Interval-cens. =     92
  Upper endpoint: rtime
                                                  Wald chi2(10)     =  31.99
Log likelihood = -590.43386                       Prob > chi2       = 0.0004
```

	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
main						
age_mean	-.0310177	.0233817	-1.33	0.185	-.076845	.0148097
male						
Yes	-1.271583	.4604788	-2.76	0.006	-2.174105	-.3690615
needle						
Yes	-.1819587	.3297493	-0.55	0.581	-.8282554	.464338
inject						
Yes	.6852961	.3431924	2.00	0.046	.0126513	1.357941
jail						
Yes	-.529615	.4021087	-1.32	0.188	-1.317734	.2585036
tvc						
age_mean	-.000129	.0017099	-0.08	0.940	-.0034804	.0032224
male						
Yes	.0884102	.042994	2.06	0.040	.0041434	.1726769
needle						
Yes	.0358545	.0238562	1.50	0.133	-.0109027	.0826118
inject						
Yes	-.0361192	.0228754	-1.58	0.114	-.0809541	.0087157
jail						
Yes	.0916036	.0348915	2.63	0.009	.0232176	.1599896

```
Notes: Standard error estimates may be more variable for small datasets and
       datasets with low proportions of interval-censored observations.
       Variables in tvc equation interacted with _t.
Wald test that [tvc] = 0: chi2(5) = 13.3282                Prob > chi2 = 0.0205
```

The first equation, `main`, reports the coefficients for the covariates that do not vary over time; the second equation, `tvc`, reports the results for covariates interacted with time. We used the default function of time, $g(t) = t$, although we could have specified other functions with the `texp()` option.

Without the `nohr` option, the table would present results as hazard ratios for the two equations; exponentiated coefficients can be interpreted as hazard ratios when we are actually modeling time-varying covariates. However, in this case, we are using the `tvc()` option to incorporate time-varying coefficients; in our model, the coefficient for covariate k is expressed as $b_k + \gamma_k \times t$, where b_k corresponds to the parameter in the `main` equation and γ_k corresponds to the parameter in the `tvc` equation. The proportional hazards assumption does not hold unless $\gamma_k = 0$.

There is little evidence to dispute the proportionality of hazards for the covariates `age_mean`, `needle`, and `inject` with respect to this specific form of misspecification. But the proportional hazards assumption appears to be violated for covariates `male` and `jail`.

At the bottom of the table, `stintcox` displays results for a Wald test for the global proportional hazards assumption (that is, the null hypothesis is that all the coefficients in the `tvc` equation are equal to zero). Alternatively, you can specify the `lrphtest` option to perform the likelihood-ratio test between the full model and the model without the time interaction terms:

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
> tvc(age_mean i.male i.needle i.inject i.jail) lrphtest nohr
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):

Fitting main model:
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -597.65634
Iteration 200: Log likelihood = -597.57555
Iteration 295: Log likelihood = -597.56443

Fitting full model:
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -590.53655
Iteration 200: Log likelihood = -590.45163
Iteration 300: Log likelihood = -590.43665
Iteration 340: Log likelihood = -590.43386

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

```
Interval-censored Cox regression                  Number of obs     =  1,124
Baseline hazard: Reduced intervals                       Uncensored =      0
                                                     Left-censored =     41
Event-time interval:                                 Right-censored =    991
  Lower endpoint: ltime                              Interval-cens. =     92
  Upper endpoint: rtime
                                                  Wald chi2(10)     =  31.99
Log likelihood = -590.43386                       Prob > chi2       = 0.0004
```

	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
main						
age_mean	-.0310177	.0233817	-1.33	0.185	-.076845	.0148097
male						
Yes	-1.271583	.4604788	-2.76	0.006	-2.174105	-.3690615
needle						
Yes	-.1819587	.3297493	-0.55	0.581	-.8282554	.464338
inject						
Yes	.6852961	.3431924	2.00	0.046	.0126513	1.357941
jail						
Yes	-.529615	.4021087	-1.32	0.188	-1.317734	.2585036
tvc						
age_mean	-.000129	.0017099	-0.08	0.940	-.0034804	.0032224
male						
Yes	.0884102	.042994	2.06	0.040	.0041434	.1726769
needle						
Yes	.0358545	.0238562	1.50	0.133	-.0109027	.0826118
inject						
Yes	-.0361192	.0228754	-1.58	0.114	-.0809541	.0087157
jail						
Yes	.0916036	.0348915	2.63	0.009	.0232176	.1599896

```
Notes: Standard error estimates may be more variable for small datasets and
       datasets with low proportions of interval-censored observations.
       Variables in tvc equation interacted with _t.
LR test that [tvc] = 0: chi2(5) = 14.2611                 Prob > chi2 = 0.0140
```

The results of the likelihood-ratio test agree with the previous Wald test—the proportional hazards assumption has been violated globally. The likelihood-ratio test may be more stable for sample sizes and when the sampling distribution of regression coefficients is not symmetric.

Specifying the `lrphtest` option can substantially increase computational time because both the full model and the reduced `main` model must be fit.

◁

Stored results

stintcox stores the following in e():

Scalars

e(N)	number of observations
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(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 likelihood
e(ll_0)	log likelihood, constant-only model
e(ll_c)	log likelihood, main model
e(N_clust)	number of clusters
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(ic)	number of iterations
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
e(emtolerance)	EM coefficient tolerance
e(emltolerance)	EM log-likelihood tolerance
e(emhsgtolerance)	EM scaled-gradient tolerance
e(noemhsgtolerance)	1 if noemhsgtolerance, 0 otherwise
e(vceiterate)	maximum VCE iterations
e(vcetolerance)	VCE log-likelihood tolerance

Macros

e(cmd)	stintcox
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(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 or LR; type of model χ^2 test
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(tvc)	covariates interacted with time from option tvc()
e(texp)	function of time used for covariates from tvc()

e(phtest)	Wald or LR; type of proportional-hazards assumption test
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

Matrices

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(vce), e(vcetype), e(stepsize), e(delta) and e(delta_n) (with fixed step size) and e(chi2_tvc) and e(p_tvc) (with the tvc() option).

Methods and formulas

Methods and formulas are presented under the following headings:

Data and model
EM algorithm for computing parameter estimates
Variance estimation using the profile log-likelihood function
Stratified estimation
Option tvc()

Data and model

For a comprehensive review of the methods in this entry, see Zeng, Mao, and Lin (2016).

Let T denote the event time, and let $\mathbf{x}(\cdot)$ denote a $1 \times p$ vector of covariates that can potentially depend on time. Under the Cox proportional-hazards model, the hazard function of T conditional on $\mathbf{x}(\cdot)$ is

$$h(t;\mathbf{x}) = h_0(t)\exp\{\mathbf{x}(t)\boldsymbol{\beta}\}$$

where $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown regression parameters and $h_0(t)$ is an arbitrary baseline hazard function. Let $H_0(t) = \int_0^t h_0(s)ds$, which is the baseline cumulative hazard function.

The occurrence of an asymptomatic event can be detected only through periodic examinations. Let $(T_l, T_u]$ denote the shortest time interval that brackets T, with $T_l < T_u$. Left-censoring is indicated by $T_l = 0$, and right-censoring by $T_u = \infty$.

Consider a study with n subjects and N observations. In a single-record-per-subject format, $n = N$ and the data consist of $(t_{li}, t_{ui}, \mathbf{x}_i)$ for $i = 1, \ldots, n$, where $\mathbf{x}_i$ records covariate values for subject i and t_{li} and t_{ui} define the observed time interval.

In a multiple-record-per-subject format, $n < N$ (typically) and the data consist of $(t_{ij}, \delta_{ij}, \mathbf{x}_{ij})$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n_i$, where $N = \sum_{i=1}^{n} n_i$, t_{ij} is the examination time j for subject i, δ_{ij} denotes whether the event of interest occurs between $t_{i(j-1)}$ and t_{ij} ($t_{i(j-1)} = 0$ if $j = 1$), and $\mathbf{x}_{ij}$ records covariate values at t_{ij}. Here $\mathbf{x}_{ij}$ includes both baseline and time-varying covariates, where for baseline covariates $\mathbf{x}_{ij} = \mathbf{x}_i$. Pairs (t_{ij}, δ_{ij})'s are used to form the event-time interval endpoints t_{li}'s and t_{ui}'s as follows. If an event occurs before the first examination time t_{i1} ($\delta_{i1} = 1$), then $t_{li} = 0$, $t_{ui} = t_{i1}$, and subject i is left-censored. If an event does not occur by the last examination time t_{in_i} ($\delta_{in_i} = 0$), then $t_{li} = t_{in_i}$, $t_{ui} = +\infty$, and subject i is right-censored. If, for $1 < j < n_i$, an event occurs between two examination times $t_{i(j-1)}$ and t_{ij} ($\delta_{ij} = 1$), then $t_{li} = t_{i(j-1)}$, $t_{ui} = t_{ij}$, and subject i is interval-censored.

The observed-data likelihood function for $\boldsymbol{\beta}$ and $H_0(t)$ is

$$L_n(\boldsymbol{\beta}, H_0) = \prod_{i=1}^{n} \Big(\exp[-\int_0^{t_{li}} \exp\{\mathbf{x}_i(s)\boldsymbol{\beta}\} dH_0(s)] - \exp[-\int_0^{t_{ui}} \exp\{\mathbf{x}_i(s)\boldsymbol{\beta}\} dH_0(s)] \Big)$$

where the integral is ∞ if $t_{ui} = \infty$ and $\mathbf{x}_i(\cdot)$ is a $1 \times p$ vector of potentially time-varying covariates for subject i. Under the NPMLE approach, H_0 is regarded as a step function with nonnegative jumps $h_1, \ldots, h_m$ at $t_1, \ldots, t_m$, respectively, where $t_1 < \cdots < t_m$ are the distinct time points for all $t_{li} > 0$ and $t_{ui} < \infty$ for $i = 1, \ldots, n$. Thus, we need to maximize the function

$$L_n(\boldsymbol{\beta}, \{h_k\}) = \prod_{i=1}^{n} \exp\Big\{ - \sum_{t_k \le t_{li}} h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta}) \Big\} \Big[1 - \exp\Big\{ - \sum_{t_{li} < t_k \le t_{ui}} h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta}) \Big\} \Big]^{I(t_{ui} < \infty)} \tag{1}$$

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

Direct maximization of (1) is difficult because of the lack of an analytic expression for the parameters h_k ($k = 1, \ldots, m$). And an even greater challenge is that not all t_{li} and t_{ui} are informative about the event times, so many h_k's are zeros and thus lie on the boundary of the parameter space.

To address these challenges, Zeng, Mao, and Lin (2016) construct some latent Poisson variables that yield the same observed-data likelihood as (1). They propose the EM algorithm, in which the E-step involves simple calculations and the M-step amounts to the maximization of a weighted sum of Poisson log-likelihood functions that is strictly concave and has a closed-form solution for h_k's.

Turnbull (1976) showed that the NPMLE of the survival distribution is unique only up to a set of intervals, which is called Turnbull's intervals (the innermost intervals or the regions of the maximal cliques). Therefore, for computational reasons, by default or if the `reduced` option is specified, we estimate the baseline cumulative hazard function at the endpoints of Turnbull's reduced set of intervals. Alternatively, you can specify the `full` option to estimate the baseline cumulative hazard function at the endpoints of all observed time intervals, which corresponds to the approach of Zeng, Mao, and Lin (2016).

EM algorithm for computing parameter estimates

Let W_{ik} ($i = 1, \ldots, n; k = 1, \ldots, m$) be independent latent Poisson random variables with means $h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta})$. Define $A_i = \sum_{t_k \le t_{li}} W_{ik}$ and $B_i = I(t_{ui} < \infty) \sum_{t_{li} < t_k \le t_{ui}} W_{ik}$. The likelihood for the observed data $\{t_{li}, t_{ui}, \mathbf{x}_i(\cdot), A_i = 0, B_i > 0\}$ ($i = 1, \ldots, n$) is

$$\prod_{i=1}^{n} \prod_{t_k \le t_{li}} \Pr(W_{ik} = 0) \Big\{ 1 - \Pr\Big(\sum_{t_{li} < t_k \le t_{ui}} W_{ik} = 0 \Big) \Big\}^{I(t_{ui} < \infty)} \tag{2}$$

which is exactly equal to (1). Thus, we can maximize (2) through an EM algorithm treating W_{ik} as missing data.

The complete-data log likelihood is

$$\sum_{i=1}^{n}\sum_{k=1}^{m} I(t_k \le t^*_{ui})\Big[W_{ik}\log\{h_k\exp(\mathbf{x}^*_{ik}\boldsymbol{\beta})\} - h_k\exp(\mathbf{x}^*_{ik}\boldsymbol{\beta}) - \log W_{ik}!\Big]$$

where $t^*_{ui} = I(t_{ui} < \infty)t_{ui} + I(t_{ui} = \infty)t_{li}$. In the E-step, we evaluate the posterior means of W_{ik} as

$$w_{ik} = \begin{cases} \dfrac{h_k\exp(\mathbf{x}^*_{ik}\boldsymbol{\beta})}{1 - \exp\{-\sum_{t_{li}<t_j\le t_{ui}} h_j\exp(\mathbf{x}^*_{ij}\boldsymbol{\beta})\}} & \text{if } t_{li} < t_k \le t_{ui} < \infty \\ 0 & \text{otherwise} \end{cases}$$

In the M-step, we update $\boldsymbol{\beta}$ by solving the following equation via the one-step Newton–Raphson method,

$$\sum_{i=1}^{n}\sum_{k=1}^{m} I(t_k \le t^*_{ui})\, w_{ik}\Big\{\mathbf{x}^*_{ik} - \frac{\sum_{j=1}^{n} I(t_k \le t^*_{uj})\exp(\mathbf{x}^*_{jk}\boldsymbol{\beta})\mathbf{x}^*_{jk}}{\sum_{j=1}^{n} I(t_k \le t^*_{uj})\exp(\mathbf{x}^*_{jk}\boldsymbol{\beta})}\Big\} = 0$$

and then update h_k $(k = 1, \ldots, m)$ using

$$h_k = \frac{\sum_{i=1}^{n} I(t_k \le t^*_{ui})w_{ik}}{\sum_{i=1}^{n} I(t_k \le t^*_{ui})\exp(\mathbf{x}^*_{ik}\boldsymbol{\beta})} \tag{3}$$

Setting the initial values of $\boldsymbol{\beta}$ to zeros and the initial values of h_k's to $1/m$, we iterate between the E- and M-steps until the desired convergence criteria are achieved. The convergence tolerances are described in detail in [R] **Maximize**, where `emtolerance()` is analogous to `tolerance()`, `emltolerance()` to `ltolerance()`, `hsgtolerance()` to `nrtolerance()`, and `nonrtolerance` to `nohsgtolerance`.

The observed-data log-likelihood function is calculated as

$$\log L = \sum_{i=1}^{n}\log\Big\{I(t_{li} < t_{ui})(S_{li} - S_{ui}) + I(t_{li} = t_{ui})\sum_{k=1}^{m} I(t_{li} = t_k)\, h_k\exp(\mathbf{x}^*_{ik}\boldsymbol{\beta})S_{li}\Big\}$$

where $S_{li} = \exp\{-\sum_{t_k\le t_{li}} h_k\exp(\mathbf{x}^*_{ik}\boldsymbol{\beta})\}$ and $S_{ui} = \exp\{-\sum_{t_k\le t_{ui}} h_k\exp(\mathbf{x}^*_{ik}\boldsymbol{\beta})\}I(t_{ui} < \infty)$.

When there are no covariates, the above algorithm becomes iteratively updating h_k as

$$h_k = \frac{\sum_{i=1}^{n}\{h_k I(t_{li} < t_k \le t_{ui} < \infty)S_{li}/(S_{li} - S_{ui}) + I(t_{li} = t_{ui} = t_k)\}}{\sum_{i=1}^{n} I(t_k \le t^*_{ui})}$$

where $S_{li} = \exp\{-\sum_{t_k\le t_{li}} h_k\}$ and $S_{ui} = \exp\{-\sum_{t_k\le t_{ui}} h_k\}I(t_{ui} < \infty)$.

With multiple-record interval-censored data, because uncensored observations are not supported, the terms with $t_{li} = t_{ui}$ are ignored in the formulas above.

Variance estimation using the profile log-likelihood function

Denote the estimators of $\boldsymbol{\beta}$ and h_k's by $\widehat{\boldsymbol{\beta}}$ and $\widehat{h}_k$'s, respectively, for $k = 1, \ldots, m$. The NPMLEs $\widehat{\boldsymbol{\beta}}$ and $\widehat{H}_0(t) = \sum_{t_k\le t}\widehat{h}_k$ are strongly consistent, and $\widehat{\boldsymbol{\beta}}$ is asymptotically normal and asymptotically efficient (Zeng, Mao, and Lin 2016). The covariance matrix of $\widehat{\boldsymbol{\beta}}$ can be estimated using profile likelihood (Murphy and van der Vaart 2000; Zeng, Mao, and Lin 2016; Zeng, Gao, and Lin 2017).

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

$$pl(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log\Big[\exp\Big\{ -\sum_{t_k \le t_{li}} \tilde{h}_k \exp(\mathbf{x}_{ik}^{*}\boldsymbol{\beta}) \Big\} - I(t_{iu} < \infty) \exp\Big\{ -\sum_{t_k \le t_{ui}} \tilde{h}_k \exp(\mathbf{x}_{ik}^{*}\boldsymbol{\beta}) \Big\} \Big]$$

where $\tilde{h}_k$ $(k = 1, \ldots, m)$ are the maximizers of (1) for the given $\boldsymbol{\beta}$. These maximizers are obtained from the EM algorithm described in the previous section with $\tilde{h}_k$ $(k = 1, \ldots, m)$ as the initial values and with $\boldsymbol{\beta}$ fixed over the iterations. Specifically, we apply the same EM algorithm but hold $\boldsymbol{\beta}$ fixed during the iterations. Thus, the only steps in the EM algorithm are to explicitly evaluate w_{ik} and to update h_k using (3).

Two likelihood-based methods, `vce(opg)` and `vce(oim)`, are available to estimate the covariance matrix of $\widehat{\boldsymbol{\beta}}$.

The `oim` method estimates the covariance matrix of $\widehat{\boldsymbol{\beta}}$ by the negative inverse of the Hessian matrix, $-\{D^2 pl(\widehat{\boldsymbol{\beta}})\}^{-1}$, where D^2 is the second-order numerical derivative and its (j,k)th element is

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

where $\mathbf{e}_j$ is the jth canonical vector in the space of $\boldsymbol{\beta}$ and δ_n is the step size chosen for numerical difference. Zeng, Gao, and Lin (2017) found that the above estimated matrix may be negative definite, especially in small samples. Therefore, they proposed the `opg` method, which estimates the covariance matrix using $\big[\sum_{i=1}^{n} \{Dpl_i(\widehat{\boldsymbol{\beta}})\}^{\otimes 2}\big]^{-1}$, where D is the first-order numerical derivative and $\mathbf{a}^{\otimes 2} = \mathbf{a}\mathbf{a}'$. Specifically, the jth element of D is

$$\big[Dpl_i(\widehat{\boldsymbol{\beta}})\big]_j = \frac{pl_i(\widehat{\boldsymbol{\beta}} + \delta_n \mathbf{e}_j) - pl_i(\widehat{\boldsymbol{\beta}})}{\delta_n}$$

where $pl_i(\boldsymbol{\beta})$ is the contribution of subject i to $pl(\boldsymbol{\beta})$. The resulting covariance matrix estimator is guaranteed to be positive semidefinite and more robust with respect to the choice of the step size than the estimator based on the second-order numerical difference.

The `stepsize()` suboption of the `vce()` option offers different ways for you to choose the step size with the `opg` and `oim` methods 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). And `stepsize(fixed #)` uses a fixed step size equal to $\# \times n^{-1/2}$.

This command also supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster` *clustvar*`)`, respectively. The sandwich estimator is calculated as $\{D^2 pl(\widehat{\boldsymbol{\beta}})\}^{-1} \sum_{m=1}^{M} Dpl_m(\widehat{\boldsymbol{\beta}})^{\otimes 2} \{D^2 pl(\widehat{\boldsymbol{\beta}})\}^{-1}$, where $pl_m(\boldsymbol{\beta})$ is the contribution of the mth cluster to $pl(\boldsymbol{\beta})$ and M is the number of clusters. For datasets where the `id(`*idvar*`)` option has been specified, the `vce(robust)` option has been implemented as equivalent to `vce(cluster` *idvar*`)`.

Stratified estimation

`stintcox` with the `strata()` option will produce a stratified proportional-hazards model for interval-censored data. Assume that within stratum q $(q = 1, \ldots, Q)$, the hazard function of T is

$$h_q(t; \mathbf{x}) = h_{q0}(t) \exp\{\mathbf{x}(t)\boldsymbol{\beta}\}$$

where $h_{q0}(t)$ is an unknown baseline hazard function for stratum q.

Let S_i be the stratum for subject i, which can take value from $\{1, \ldots, Q\}$. Analogously to (1), we maximize the function

$$L_n(\boldsymbol{\beta}, \{h_{qk}\}) = \prod_{i=1}^{n} \prod_{q=1}^{Q} \exp\Big\{ - \sum_{t_{qk} \le t_{li}} h_{qk} \exp(\mathbf{x}^*_{iqk}\boldsymbol{\beta}) \Big\} \Big[1 - \exp\Big\{ - \sum_{t_{li} < t_{qk} \le t_{ui}} h_{qk} \exp(\mathbf{x}^*_{iqk}\boldsymbol{\beta}) \Big\} \Big]^{I(t_{ui} < \infty)}$$

where $\mathbf{x}^*_{iqk} = \mathbf{x}_i(t_{qk})$. Within the NPMLE framework, a stratum-specific baseline cumulative hazard function $H_{q0}(t) = \int_0^t h_{q0}(s)\, ds$ for $q = 1, \ldots, Q$ is regarded as a step function, $\sum_{t_{qk} \le t} h_{qk}$, with nonnegative jumps h_{qk} at the distinct values t_{qk}, ordered from smallest to largest, of the observed interval endpoints from all subjects in stratum q.

Option tvc()

Let $\mathbf{z}(\cdot)$ be a $1 \times q$ vector of potentially time-varying covariates specified in the `tvc()` option. Let $g(\cdot)$ be the function of time specified in the `texp()` option. By default, $g(t) = t$. Let $\mathbf{z}_i$'s and $\mathbf{z}_{ij}$'s be the observed covariate values for $\mathbf{z}(\cdot)$ in the respective single-record-per-subject and multiple-record-per-subject formats.

When the `tvc()` option is specified, we replace $\mathbf{x}^*_{ik}\boldsymbol{\beta}$ in all formulas above with

$$\mathbf{x}^*_{ik}\boldsymbol{\beta} + g(t_k)\mathbf{z}^*_{ik}\boldsymbol{\gamma}$$

where $\boldsymbol{\gamma}$ is a $q \times 1$ vector of unknown regression parameters for $\mathbf{z}(\cdot)$ and $\mathbf{z}^*_{ik} = \mathbf{z}_i(t_k)$ are the covariate values of the `tvc()` variables for subject i at time t_k. For baseline covariates, $\mathbf{z}^*_{ik} = \mathbf{z}_i$. For time-varying covariates, if $t_k = t_{ij}$ is one of the examination times for subject i, then $\mathbf{z}^*_{ik} = \mathbf{z}_{ij}$. Otherwise, $\mathbf{z}^*_{ik}$ is imputed as described in the `tvcovimpute()` option in *Options*.

Acknowledgments

The development of `stintcox` was partially supported by SBIR awards 1R43CA233159-01 and 2R44CA233159-02. We also 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

Cai, T., and R. A. Betensky. 2003. Hazard regression for interval-censored data with penalized spline. *Biometrics* 59: 570–579. https://doi.org/10.1111/1541-0420.00067.

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. 1986. A proportional hazards model for interval-censored failure time data. *Biometrics* 42: 845–854. https://doi.org/10.2307/2530698.

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.

Lindsey, J. K. 1998. A study of interval censoring in parametric regression models. *Lifetime Data Analysis* 4: 329–354. https://doi.org/10.1023/A:1009681919084.

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.

Rabinowitz, D., A. A. Tsiatis, and J. Aragon. 1995. Regression with interval-censored data. *Biometrika* 82: 501–513. https://doi.org/10.2307/2337529.

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

Sun, J., and J. Li. 2014. "Interval censoring". In *Handbook of Survival Analysis*, edited by J. P. Klein, H. C. van Houwelingen, J. G. Ibrahim, and T. H. Scheike, 369–390. Boca Raton, FL: CRC Press.

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.

Wang, L., C. S. McMahan, M. G. Hudgens, and Z. P. Qureshi. 2016. A flexible, computationally efficient method for fitting the proportional hazards model to interval-censored data. *Biometrics* 72: 222–231. https://doi.org/10.1111/biom.12389.

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.

Zhang, Y., L. Hua, and J. Huang. 2010. A spline-based semiparametric maximum likelihood estimation method for the Cox model with interval-censored data. *Scandinavian Journal of Statistics* 37: 338–354. https://doi.org/10.1111/j.1467-9469.2009.00680.x.

Also see

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

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

[ST] **stmgintcox** — Marginal Cox PH model for interval-censored multiple-event data

[U] 20 Estimation and postestimation commands

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.

For suggested citations, see the FAQ on citing Stata documentation.