The Stata Journal (2016)
16, Number 4, pp. 867–879

Estimating survival functions after stcox with time-varying coefficients

Constantin Ruhe
Zukunftskolleg and Department of Politics and Public Administration
University of Konstanz
Konstanz, Germany
Constantin.Ruhe@uni-konstanz.de

Abstract. In many applications of the Cox model, the proportional-hazards assumption is implausible. In these cases, the solution to nonproportional hazards usually consists of modeling the effect of the variable of interest and its interaction effect with some function of time. Although Stata provides a command to implement this interaction in `stcox`, it does not allow the typical visualizations using `stcurve` if `stcox` was estimated with the `tvc()` option. In this article, I provide a short workaround that estimates the survival function after `stcox` with time-dependent coefficients. I introduce and describe the `scurve_tvc` command, which automates this procedure and allows users to easily visualize survival functions for models with time-varying effects.

Keywords: st0458, scurve_tvc, stcox, tvc() option, stcurve, sts generate, Cox model, proportional hazards, time-varying coefficients, survival function

1 Overview

Researchers in many disciplines are interested in the time-varying effects of variables in duration analyses (Box-Steffensmeier, Reiter, and Zorn 2003; Giolo et al. 2012; Nilsson and Nivre 2013). Assume that a cancer treatment x had short-term benefits but long-term treatment caused side effects that eventually deteriorated a patient's health. In this scenario, we would like to know if the long-term side effects outweigh the short-term benefits, which is the case if after a certain time, fewer patients who receive treatment x survive compared with patients who do not receive the treatment. These groups are comparable with the respective survivor functions for each group (Putter et al. 2005). Calculating these estimates requires a short workaround, because Stata does not provide a built-in solution to plot the survival function with time-varying coefficients. I provide a solution below.

When all covariates have constant effects, it is straightforward to calculate the survivor function for different scenarios based on the estimated coefficients and baseline survival functions (Kalbfleisch and Prentice 2002; Cleves, Gould, and Marchenko 2016). Let $h_0(t)$ be the baseline hazard function. The Cox model asserts that the hazard function for an individual i with covariates x is

$$h(t|x_i) = h_0(t)\exp(x_i\beta)$$

From this, we can calculate the cumulative hazard function:

$$H(t|x_i) = \int_0^t h(u|x_i)\,\mathrm{d}u = \exp(x_i\beta)\int_0^t h_0(u)\,\mathrm{d}u = \exp(x_i\beta)H_0(t) \tag{1}$$

Because we can calculate the survivor function from the cumulative hazard function, we get

$$S(t|x_i) = \exp\left\{-\exp\left(x_i\beta\right)H_0(t)\right\} = S_0(t)^{\exp(x_i\beta)}$$

Time-varying effects complicate this calculation. To allow for a changing effect, we can include the variable's interaction with some function of time as a time-varying covariate. If we know the function $f(t)$ by which the effect varies with time, we can easily model the effect. Consider a variable z with such a time-varying effect. If we enter the interaction of z with time as a time-varying covariate, we can write the hazard as

$$h(t|z_i) = h_0(t)\exp\left\{z_i\gamma + z_i f(t)\delta\right\} = h_0(t)\exp\left[z_i\left\{\gamma + f(t)\delta\right\}\right] = h_0(t)\exp(z_i\beta_t)$$

While this easily accounts for the time-varying effect, it complicates the survival function's calculation. Once the predictor variables in the model interact with time, the linear combination of predictors $z_i\beta_t$ depends on time and remains in the integral in (1).

Below, I provide an example of how to estimate these survivor functions in Stata. Then, I describe the new command `scurve_tvc`, which automates this procedure.

2 Estimating survival functions with the tvc() option

Consider a case with a binary treatment variable `x` and a binary confounder `control`, in which the effect of `x` changes over time. The following code generates duration data for this setting, and the observations are censored after 30 observations (see also Crowther and Lambert [2012]). The data come from the following exponential duration model:

$$h(t|\mathtt{x}_i) = \exp\left\{\ln(0.05) - 0.9\mathtt{x}_i + 0.6\ln(t)\mathtt{x}_i + 1.5\mathtt{control}_i\right\}$$

```
. set seed 94215841
. set obs 1000
number of observations (_N) was 0, now 1,000
. generate x=(runiform()>.5)
. generate control=(runiform()>.5)
. survsim ftime, cov(x -.9 control 1.5) tde(x .6) distribution(exponential)
>         lambda(.05)
. generate failure=(ftime<=30)
. replace ftime=30 if ftime>30
(65 real changes made)
. stset ftime, failure(failure)
     failure event:  failure != 0 & failure < .
obs. time interval:  (0, ftime]
 exit on or before:  failure
------------------------------------------------------------------------------
       1000  total observations
          0  exclusions
------------------------------------------------------------------------------
       1000  observations remaining, representing
        935  failures in single-record/single-failure data
   9693.849  total analysis time at risk and under observation
                                             at risk from t =         0
                                  earliest observed entry t =         0
                                       last observed exit t =        30
```

To model the time-dependent effect, users could use the `stcox` command with the `tvc()` option. Unfortunately, Stata cannot estimate survival functions in the presence of time-dependent effects.

```
. stcox x control, tvc(x) texp(ln(_t)) nohr nolog
        failure _d:  failure
  analysis time _t:  ftime
Cox regression -- no ties
No. of subjects =        1,000                  Number of obs    =       1,000
No. of failures =          935
Time at risk    =  9693.849402
                                                LR chi2(3)       =      468.86
Log likelihood  =   -5468.3546                  Prob > chi2      =      0.0000
------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
main         |
           x |   -.931918    .132382    -7.04   0.000    -1.191382    -.672454
     control |   1.464325   .0768318    19.06   0.000     1.313737    1.614912
-------------+----------------------------------------------------------------
tvc          |
           x |   .6156235    .065858     9.35   0.000     .4865441    .7447028
------------------------------------------------------------------------------
Note: Variables in tvc equation interacted with ln(_t).
. capture noisily stcurve, survival at(x=1) at(x=0)
this post-estimation command is not allowed after estimation with tvc();
see tvc note for an alternative to the tvc() option
```

In a randomized clinical trial, the results for both groups are comparable using, for example, the Kaplan–Meier survival estimates. However, if the data stem from observational studies and require adjustment for many covariates, this comparison is no longer useful. Moreover, we might want to predict the survival function for different subgroups to assess the relative success of an intervention in different contexts (see also Putter et al. [2005]). In this case, a manual workaround can help us determine how the two groups evolve over time and how the effect differs across subgroups in our sample. Below, I outline this workaround with example data.

First, we need to estimate the time-varying coefficient manually:

```
. generate id=_n
. stset ftime, id(id) failure(failure)
                id:  id
     failure event:  failure != 0 & failure < .
obs. time interval:  (ftime[_n-1], ftime]
 exit on or before:  failure
------------------------------------------------------------------------------
       1000  total observations
          0  exclusions
------------------------------------------------------------------------------
       1000  observations remaining, representing
       1000  subjects
        935  failures in single-failure-per-subject data
   9693.849  total analysis time at risk and under observation
                                             at risk from t =         0
                                  earliest observed entry t =         0
                                       last observed exit t =        30
. stsplit, at(failures)
(935 failure times)
(497,420 observations (episodes) created)
. generate x_t = x*ln(_t)
. stcox x x_t control, nolog nohr
        failure _d:  failure
  analysis time _t:  ftime
                id:  id
Cox regression -- no ties
No. of subjects =        1,000                 Number of obs    =     498,420
No. of failures =          935
Time at risk    =  9693.849402
                                               LR chi2(3)       =      468.86
Log likelihood  =   -5468.3546                 Prob > chi2      =      0.0000
------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   -.931918    .132382    -7.04   0.000    -1.191382    -.672454
         x_t |   .6156235    .065858     9.35   0.000     .4865441    .7447028
     control |   1.464325   .0768318    19.06   0.000     1.313737    1.614912
------------------------------------------------------------------------------
```

We could now use the `stcurve` command, but setting `x_t` to some value would ignore the fact that this variable is not constant, but rather varies with time. Hence, we need to proceed with the manual workaround and obtain the stored coefficient estimates:

```
. matrix b=e(b)
```

We now use `sts generate` (or, alternatively, `predict` with the option `basehc`) to store the estimated hazard component, $\Delta H(t_j) = H(t_j) - H(t_{j-1})$, adjusting all variables to 0. Smoothing the variable `baseline` generates an estimate of the baseline hazard function. The cumulative function of the variable gives us an estimate of the baseline cumulative hazard function.

```
. sts generate baseline=h, adjust(x x_t control)
```

Here, however, we are interested in a comparison of the survival function for four different scenarios: `x` = 1 and `control` = 0; `x` = 0 and `control` = 0; `x` = 1 and `control` = 1; and `x` = 0 and `control` = 1. Because the effect of `x` varies with time, we cannot calculate the function from the baseline cumulative hazard or survival function. We therefore calculate the scenario-specific hazard contribution based on Kalbfleisch and Prentice (2002, 114ff).

$$\Delta H(t_j|\mathbf{x}_i) = \left[1 - \{1 - \Delta H\,(t_j)\}^{\exp(\mathbf{x}_i\beta)}\right]$$

By summing up these values, we approximate the cumulative hazard function for each scenario. From this, we can calculate the estimated survival function.

$$S(t_j|\mathbf{x}_i) = \exp\{-\textstyle\sum \Delta H(t_j|\mathbf{x}_i)\}$$

The workaround implements these steps:

```
. preserve
. sort _t
. collapse (mean) baseline, by(_t)
. *scenario 1: x=1 and control=0
. generate b_x_nocontrol = 1-(1-baseline)^exp(b[1,1]+b[1,2]*ln(_t))
(1 missing value generated)
. generate H_x_nocontrol = sum(b_x_nocontrol)
. generate S_x_nocontrol = exp(-H_x_nocontrol)
. *scenario 2: x=0 and control=0
. generate b_nox_nocontrol = baseline
(1 missing value generated)
. generate H_nox_nocontrol = sum(b_nox_nocontrol)
. generate S_nox_nocontrol = exp(-H_nox_nocontrol)
. *scenario 3: x=1 and control=1
. generate b_x_control = 1-(1-baseline)^exp(b[1,1]+b[1,2]*ln(_t)+b[1,3])
(1 missing value generated)
. generate H_x_control = sum(b_x_control)
. generate S_x_control = exp(-H_x_control)
```

```
. *scenario 4: x=0 and control=1
. generate b_nox_control = 1-(1-baseline)^exp(b[1,3])
(1 missing value generated)
. generate H_nox_control = sum(b_nox_control)
. generate S_nox_control = exp(-H_nox_control)
. keep S_x_nocontrol S_nox_nocontrol S_x_control S_nox_control  _t
. rename _t _tx
. save my_surv_curve, replace
file my_surv_curve.dta saved
. restore
. merge 1:1 _n using my_surv_curve
```

(*output omitted*)

This dataset can now be merged with the original dataset and plotted using simple line plots. Figure 1 compares the workaround estimates with Kaplan–Meier estimates for each scenario. The user-written command `grc1leg` (Royston 2014) combines the individual graphs. The graph demonstrates that the estimated survival functions are consistent with the nonparametric Kaplan–Meier estimates for each scenario.

```
. sts graph if control==0, by(x) plot1opt(lpat(dash)) plot2opt(lpat(solid))
>         scheme(sj) saving(km1, replace) ylabel(0(.2)1, format(%9.1g))
>         title("") subtitle("Control=0")
         failure _d:  failure
   analysis time _t:  ftime
                 id:  id
(file km1.gph saved)
. sts graph if control==1, by(x) plot1opt(lpat(dash)) plot2opt(lpat(solid))
>         scheme(sj) legend(off) saving(km2, replace) ylabel(0(.2)1,
>         format(%9.1g)) title("") t1title("Control=1")
         failure _d:  failure
   analysis time _t:  ftime
                 id:  id
(file km2.gph saved)
. line S_nox_nocontrol S_x_nocontrol _tx, c(J J) sort lp(dash solid)
>         scheme(sj) legend(off) saving(w1, replace) ylabel(0(.2)1)
>         t1title("Control=0") xtitle("analysis time")
(file w1.gph saved)
. line S_nox_control S_x_control _tx, c(J J) sort lp(dash solid)
>         scheme(sj) legend(off) saving(w2, replace) ylabel(0(.2)1)
>         t1title("Control=1") xtitle("analysis time")
(file w2.gph saved)
. grc1leg km1.gph km2.gph, leg(km1.gph) scheme(sj) ycommon
>         saving(km, replace) subtitle("Kaplan-Meier estimates")
(file km.gph saved)
. grc1leg w1.gph w2.gph, scheme(sj) ycommon saving(w, replace)
>         subtitle("Cox estimates")
(file w.gph saved)
. grc1leg km.gph w.gph, scheme(sj) ycommon leg(km.gph) col(1)
```

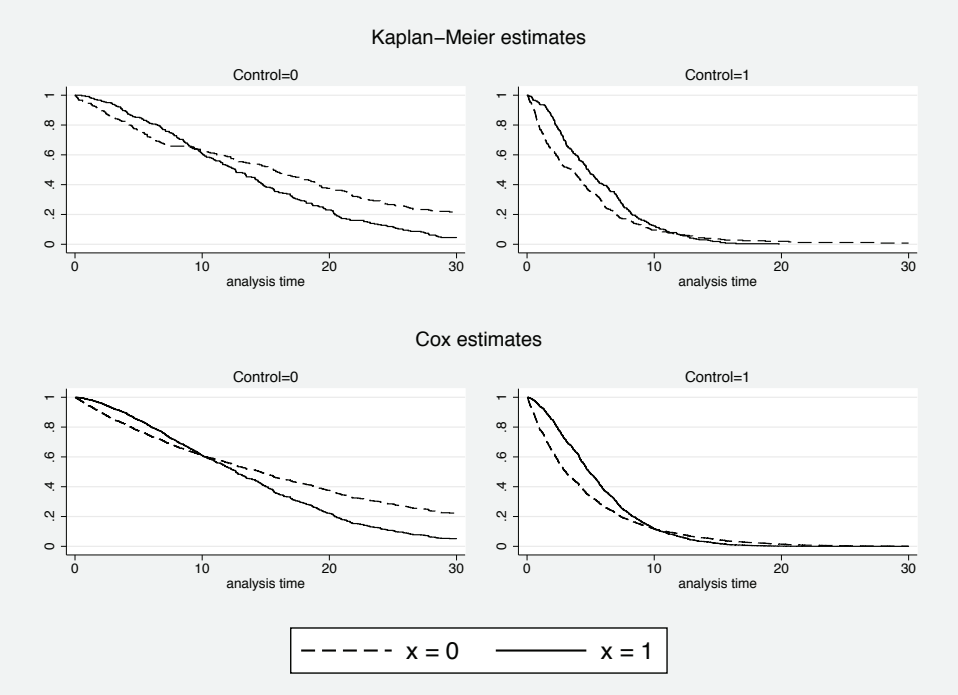

Figure 1. Comparing Kaplan–Meier survival estimates with estimated survivor functions using the workaround

3 The scurve_tvc command

This procedure is automated in the ado-file `scurve_tvc`. This command fits a Cox model with time-varying coefficients from which it estimates, saves, and plots the estimated survival function for a user-specified scenario. I describe `scurve_tvc`'s syntax and options and provide an illustrative example below.

3.1 Syntax

`scurve_tvc` [*if*] [*in*], `generate(`*newvar*`) at(`*varname* # [*varname* # ...]`)`
`tvc(`*varlist*`) texp(`*string*`)` [`replace ties(`*stcox_ties*`) shared(`*varname*`)`
`strata(`*varname*`) graph plotopts(`*options*`)`]

3.2 Description

`scurve_tvc` fits `stcox` with time-varying effects and calculates the survival curve for specific covariate values.

3.3 Options

`generate(`*newvar*`)` creates the variable *newvar* to store the estimated survival curve. If you also specify `strata()`, then `scurve_tvc` creates one variable for each stratum. The corresponding analysis-time variable, which allows us to plot the results, is saved in the new variable `_tscurve`. `generate()` is required.

`at(`*varname* # [*varname* # ...]`)` specifies the covariates included in the model and the values for which the survival curve should be calculated. `at()` is required.

`tvc(`*varlist*`)` specifies the covariates with time-varying coefficients. The variables in `tvc()` must also appear in `at()`. `scurve_tvc` will automatically `stsplit` the data at failure times to ensure a correctly fit model. Type `help tvc note` within Stata for more information. `tvc()` is required.

`texp(`*string*`)` specifies the function of analysis time according to which the effect varies with time. For example, specifying `texp(ln(_t))` would cause the variables with time-varying coefficients to be multiplied by the logarithm of analysis time. `texp()` is required.

`replace` specifies to replace the existing variable(s) with the new estimates.

`ties(`*stcox_ties*`)` specifies how `stcox` handles tied failure times. See [ST] **stcox** for details.

`shared(`*varname*`)` specifies a shared-frailty ID variable. See [ST] **stcox** for details.

`strata(`*varname*`)` specifies a strata ID variable. See [ST] **stcox** for details.

`graph` plots the predicted survival curve. If you also specify `strata()`, then `graph` plots the survival estimates for each stratum.

`plotopts(`*options*`)` customizes the plot by using options allowed with `graph twoway line`.

3.4 Example

Consider a case with a binary treatment variable `x`, a binary confounder `control1`, and a continuous confounder `control2`, where the effect of `x` changes over time. The following code generates duration data for this setting, where the observations are censored after 30 observations (see also Crowther and Lambert [2012]). The data come from the following Weibull duration model:

$$h(t|\mathtt{x}_i) = 1.3t^{1.3-1}\exp\left\{\ln(0.05) - 0.9\mathtt{x}_i + 0.6\ln(t)\mathtt{x}_i - 1.3\mathtt{control1}_i - 0.4\mathtt{control2}_i\right\}$$

```
. clear
. set seed 581265456
. set obs 1000
number of observations (_N) was 0, now 1,000
. generate x=(runiform()>.5)
. generate control1=(runiform()>.5)
. generate control2=rnormal()
. survsim ftime, cov(x -.9 control1 -1.3 control2 -.4) tde(x .6)
>         distribution(weibull) lambda(.05) gamma(1.3)
. generate failure=(ftime<=30)
. replace ftime=30 if ftime>30
(109 real changes made)
. generate id=_n
. stset ftime, id(id) failure(failure)
                id:  id
     failure event:  failure != 0 & failure < .
obs. time interval:  (ftime[_n-1], ftime]
 exit on or before:  failure
------------------------------------------------------------------------------
       1000  total observations
          0  exclusions
------------------------------------------------------------------------------
       1000  observations remaining, representing
       1000  subjects
        891  failures in single-failure-per-subject data
  13122.843  total analysis time at risk and under observation
                                              at risk from t =         0
                                   earliest observed entry t =         0
                                        last observed exit t =        30
```

Assume that `x` is a cancer treatment that causes severe long-term side effects. We want to know when the treatment is beneficial and whether the survival functions cross at a certain point in time. Here, we use `scurve_tvc` to fit a Cox model in which the effect of `x` varies with time and to predict the estimated survival function for a specific scenario, which here is `x` = 1, `control1` = 0, and `control2` = 0:

```
. scurve_tvc, generate(S_x) at(x 1 control1 0 control2 0) tvc(x) texp(ln(_t))
> replace
(note: variable S_x not found)
Dataset has been temporarily split at failure times
(891 failure times)
(493,614 observations (episodes) created)
The estimation is based on the following Cox Proportional Hazards Model:
         failure _d:  failure
   analysis time _t:  ftime
                 id:  id
Iteration 0:   log likelihood = -5506.5059
Iteration 1:   log likelihood = -5299.4786
Iteration 2:   log likelihood = -5298.9218
Iteration 3:   log likelihood = -5298.9218
Refining estimates:
Iteration 0:   log likelihood = -5298.9218
Cox regression -- no ties
No. of subjects =        1,000                 Number of obs    =     494,614
No. of failures =          891
Time at risk    =  13122.84325
                                               LR chi2(4)       =      415.17
Log likelihood  =   -5298.9218                 Prob > chi2      =      0.0000
------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |  -.9929653   .1971679    -5.04   0.000    -1.379407   -.6065234
    control1 |  -1.230347   .0740657   -16.61   0.000    -1.375513   -1.085181
    control2 |  -.3876476   .0352886   -10.99   0.000     -.456812   -.3184833
        _x_t |     .63722   .0861443     7.40   0.000     .4683803    .8060597
------------------------------------------------------------------------------
Note: tvc-interactions denoted by _varname_t were interacted with ln(_t).
```

`scurve_tvc` automatically executes all steps of the workaround presented above. It splits the data at failure times, generates interaction variables based on the specified function, and shows the model output. It stores the estimated survival function in a new variable called `S_x`.

We then repeat the process for `x` = 0. We plot this variable against the corresponding analysis time stored in the new variable `_tscurve`. Figure 2 shows that, for the specified scenarios, the survival curves cross after about nine time units. Hence, the negative side effects of treatment `x` outweigh its benefits after this time.

```
. quietly scurve_tvc, generate(S_nox) at(x 0 control1 0 control2 0) tvc(x)
>         texp(ln(_t)) replace
. label var S_x "x=1"
. label var S_nox "x=0"
. twoway line S_x S_nox _tscurve, c(J J) scheme(sj) title("")
>         xtitle("Analysis time") ytitle("Probability of survival")
>         ylabel(0(.2)1)
```

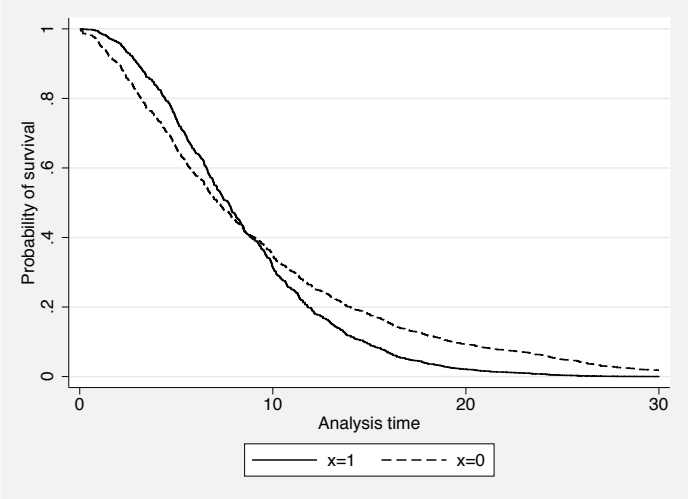

Figure 2. Predicted survival probabilities with and without treatment `x`; remaining covariates held at 0

For a single scenario, using the option `graph` along with `plotopts()` automatically produces a graph for the scenario specified in `at()`. However, we are often interested in a comparison of different scenarios, similar to Putter et al. (2005). Hence, we can use `scurve_tvc` to estimate the survival functions for `x` = 0 and `x` = 1 at various values of the remaining covariates. In the hypothetical cancer treatment example, we could calculate the estimated survival probabilities for specific patient characteristics with different risk levels. We could then plot the estimated survival functions for each scenario to transparently communicate the estimated survival probabilities and highlight potential trade-offs for treatments with time-varying effects.

Figure 3 provides these estimates for four scenarios. The substantive interpretation for our hypothetical cancer treatment implies that treatment `x` is ineffective and even harmful for low- and moderate-risk patients. However, high-risk patients benefit from the procedure. The graphical representation therefore helps clearly communicate the complex results from the model with nonproportional hazards.

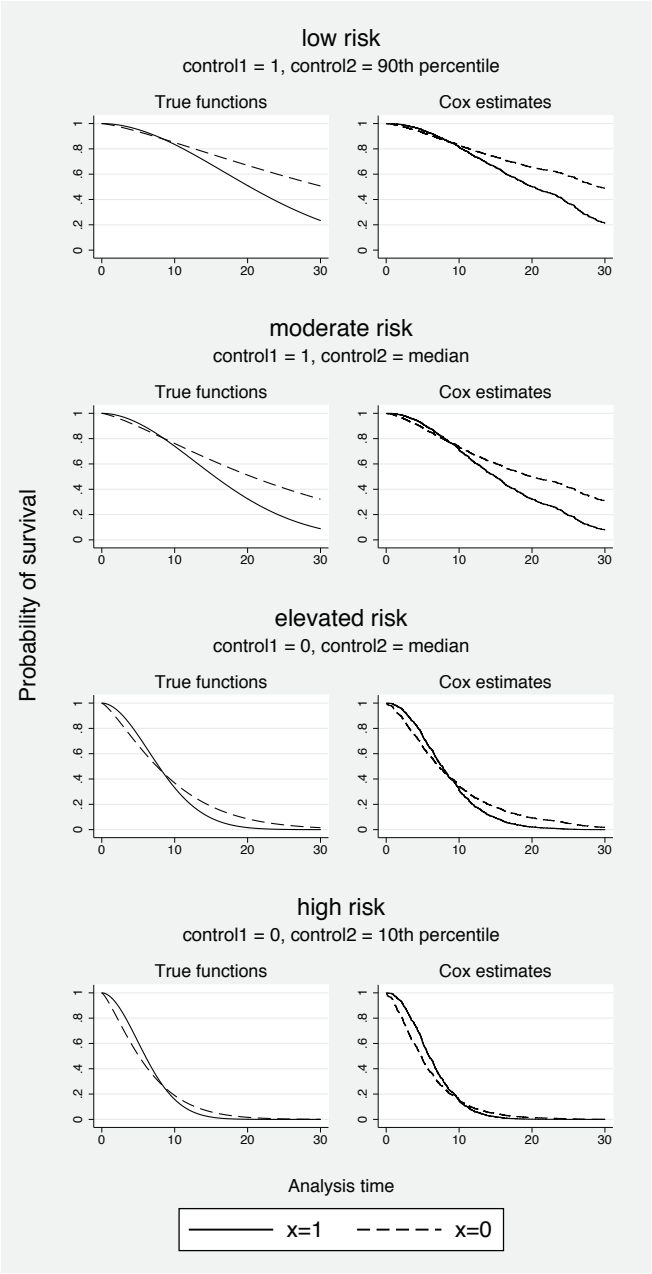

Figure 3. Comparing true, analytically calculated survivor functions (left) for the simulated data with empirical estimates calculated with `scurve_tvc` (right)

To highlight that the procedure accurately captures the data-generating process, figure 3 shows the analytically calculated, true survival functions and compares these with the estimates from `scurve_tvc`. The plot indicates that the results closely agree with the underlying data-generating process.

4 Conclusion

In this article, I demonstrated how to estimate survival functions from Cox models with time-varying coefficients. The code is automated in the new `scurve_tvc` command. The procedure allows us to visualize the predictions of models with nonproportional hazards and enables us to effectively communicate model predictions for different covariate scenarios to a broader audience.

5 References

Box-Steffensmeier, J. M., D. Reiter, and C. Zorn. 2003. Nonproportional hazards and event history analysis in international relations. *Journal of Conflict Resolution* 47: 33–53.

Cleves, M., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Revised 3rd ed. College Station, TX: Stata Press.

Crowther, M. J., and P. C. Lambert. 2012. Simulating complex survival data. *Stata Journal* 12: 674–687.

Giolo, S. R., J. E. Krieger, A. J. Mansur, and A. C. Pereira. 2012. Survival analysis of patients with heart failure: Implications of time-varying regression effects in modeling mortality. *PLOS ONE* 7: e37392.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Nilsson, M., and J. Nivre. 2013. Proportional hazards modeling of saccadic response times during reading. *Topics in Cognitive Science* 5: 541–563.

Putter, H., M. Sasako, H. H. Hartgrink, C. J. H. van de Velde, and J. C. van Houwelingen. 2005. Long-term survival with non-proportional hazards: Results from the Dutch Gastric Cancer Trial. *Statistics in Medicine* 24: 2807–2821.

Royston, P. 2014. Tools for checking calibration of a Cox model in external validation: Approach based on individual event probabilities. *Stata Journal* 14: 738–755.

About the author

Constantin Ruhe is a postdoctoral researcher in the Zukunftskolleg and in the Department of Politics and Public Administration at the University of Konstanz in Germany. His main research interests are quantitative analyses of political violence, conflict management, and applied statistics.