The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–14

Bootstrap pointwise confidence intervals for covariate-adjusted survivor functions in the Cox model

Constantin Ruhe
Institute of Political Science, Faculty of Social Sciences
Goethe University Frankfurt
Frankfurt am Main, Germany
Ruhe@soz.uni-frankfurt.de

Abstract. Survival functions are a common visualization of predictions from the Cox model. However, neither Stata's `stcurve` nor the user-written `scurve_tvc` command allow to estimate confidence intervals. In this article, I discuss how bootstrap confidence intervals can be formed for covariate-adjusted survival functions in the Cox model. The new command `bsurvci` automates this procedure and allows users to visualize the results. The new command enables to estimate uncertainty around survival functions estimated from Cox models with time-varying coefficients, a capability which was not previously available in Stata. Furthermore, it provides users of Stata with an additional option for survival estimates from Cox models with proportional hazards, by allowing them to chose between bootstrap confidence intervals using `bsurvci` and asymptotic confidence intervals from an existing user-written command `survci`. Since asymptotic confidence intervals make distributional assumptions when constructing confidence intervals, the bootstrap procedure proposed in this article provides a non-parametric alternative.

Keywords: st0001, `bsurvci`, `scurve_tvc`, `stcox`, `tvc()`, `survci`, confidence intervals, bootstrap, Cox model, proportional hazards, time-varying coefficients, survival function

1 Overview

The Cox Proportional Hazard model is the predominant model in survival analysis. Aside from the commonly used hazard ratios, survival functions are an intuitive way to communicate the implications of duration analyses (cf. Cleves et al. 2010; Putter et al. 2005; Ruhe 2018). Nevertheless, due to the fact that the Cox Proportional Hazard model does not directly estimate the baseline hazard, it is not straightforward to describe the uncertainty around the point estimate for the survival function. In this paper, I discuss how to calculate bootstrap pointwise confidence intervals for survival functions in Stata and introduce the `bsurvci` command which automates this procedure. If the model contains only covariates with proportional hazards, the new command `bsurvci` is a non-parametric alternative to asymptotic confidence intervals which need to make distributional assumptions about the variance of the survival function (cf. Cefalu 2011). Furthermore, for models with non-proportional hazards, the `bsurvci` command provides

the only possible uncertainty measure in Stata, since asymptotic confidence intervals are not currently available.

The `bsurvci` command introduced below enables researchers to easily compare analytical and bootstrapped confidence intervals from proportional hazards models and introduces bootstrap confidence intervals for models with time-varying coefficients, which was not possible in Stata to date. Since a survival function is a point estimate for multiple time points, Statas build-in command `bootstrap` cannot be used in this context. Hence, the bootstrap approach which I introduce in this paper provides a new capability to Stata users.

In the following, I discuss the bootstrap can improve statistical inference and outline how bootstrap pointwise confidence intervals can be estimated. Thereafter, I introduce the new command `bsurvci` which automates the method. Lastly, I demonstrate the use with examples.

2 Bootstrap vs. asymptotic confidence intervals

If the model contains only covariates with proportional hazards, several approaches for analytical, asymptotic pointwise confidence intervals exist. Cefalu (2011) provides a Stata command to calculate these types of confidence intervals using either the linear or the log-log approach. However, it has been shown that bootstrap methods can provide a better uncertainty estimation for survival functions (cf. Burr 1994). The analytically calculated intervals are derived based on assumptions regarding the asymptotic distribution of the estimated statistic. In the case of survival functions from the semiparametric Cox-Model, in which the baseline hazard is not defined, usually either the normal distribution or a transformation is assumed (Klein and Moeschberger 2003). If this assumption is incorrect, asymptotic confidence intervals will provide invalid inferences. In these cases, the bootstrap provides alternative methods to construct confidence intervals by resampling from the data used in the analysis.

Invalid inferences based on asymptotic methods are often due to deviations from a confidence interval's nominal coverage probability, i.e. the probability with which the confidence interval contains the true parameter. E.g. for a confidence interval with a nominal coverage probability of 90 percent, an accurate coverage probability would imply that in 90 percent of all samples a confidence interval calculated with this method will contain the true value (Carpenter and Bithell 2000).

For survival estimates from the Cox model, simulation studies have shown that bootstrap confidence intervals have better coverage probability and outperform asymptotic confidence intervals, depending on the bootstrap method used. Whereas the asymptotic method used in these simulations provided smaller than nominal coverage probabilities, bootstrap confidence intervals based on percentile methods had approximately nominal coverage probabilities. Other methods to form bootstrap confidence intervals performed less well (Burr 1994).

Bootstrap methods are thus a good alternative or cross-check to assess if the in-

ference might be driven by inaccurate distributional assumptions. This is particularly relevant for small samples in which the asymptotic distributional assumptions are questionable, but it also applies in larger samples and complex models. In an ideal case, these methods support the inference drawn from asymptotic methods. In the worst case, bootstrap confidence intervals cast the results from asymptotic intervals in doubt (Carpenter and Bithell 2000).

Since bootstrap confidence intervals are non-parametric, it might seem advisable to always use the bootstrap. However, the bootstrap is computationally intensive, since it requires that the estimation is repeated many times based on alternative samples. In contrast, asymptotic confidence intervals are often trivial and fast to compute. Especially for large sample sizes and complex models, it is therefore advisable to apply the bootstrap procedure discussed in this paper as a validation step in the final stages of the analysis.

In some instances, methods for asymptotic intervals are not readily available in Stata. This is the case if the Cox model contains non-proportional hazards, e.g. due to time-varying effects. Non-proportional hazards can be modeled in the Cox model by interacting the respective variable with some function of analysis time. In these cases, researchers can now rely on the command provided with this paper to get uncertainty estimates for survival functions from Cox models with non-proportional hazards.[1]

3 Bootstrapping survival functions

Several methods exist to generate bootstrap samples as well as to construct confidence intervals for the desired statistic. The sampling method used in this paper is simple resampling with replacement which requires no knowledge of the censoring distribution and has generated good results in previous simulation studies (Burr 1994, 1296). Assuming data with covariate vector X_i, duration time Y_i and censoring time C_i, we observe time $T_i = min(Y_i, C_i)$ and the censoring indicator $\delta_i = I(Y_i < C_i)$. The simple bootstrap method simply resamples the triples (X_i, T_i, δ_i) (cf. Burr 1994).

In Stata, this consists of using the boostrapping command `bsample` with the option `cluster()`, which identifies a single case, if the data consists of multiple-record data. If a stratified model is estimated, the option `strata()` ensures that the triples are sampled within strata.

```
. bsample, cluster(id)
```

For each bootstrap sample, the survival function is estimated and saved. In the code below, this is done for a simple example Cox proportional hazard model with a single covariate called *var*. We may wish to estimate the survival function for $var = 1$ and calculate

1. An alternative beyond the Cox model is the flexible parametric Royston-Parmar estimator, which also allows to model non-proportional hazards and provides asymptotic confidence intervals (Royston and Lambert 2011)

$$\widehat{S}(t, var) = \widehat{S_0}(t)^{exp(\widehat{\beta} * var)} \tag{1}$$

whereby $\widehat{S_0}(t)$ is the baseline survival function estimate and $\widehat{\beta}$ the coefficient estimate from the Cox model.

```
. stcox var, nohr
         failure _d:  fail
   analysis time _t:  ftime
                 id:  id
Iteration 0:   log likelihood = -150.17452
Iteration 1:   log likelihood =  -149.9762
Iteration 2:   log likelihood = -149.97614
Refining estimates:
Iteration 0:   log likelihood = -149.97614
Cox regression -- Breslow method for ties
No. of subjects =           34                 Number of obs    =          50
No. of failures =           50
Time at risk    =  5331.148976
                                               LR chi2(1)       =        0.40
Log likelihood  =   -149.97614                 Prob > chi2      =      0.5288
------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         var |  -.1874376   .2996716    -0.63   0.532    -.7747831     .3999079
------------------------------------------------------------------------------
. matrix b=e(b)
. predict s0, basesurv
. gen S_est=s0^(exp(b[1,1]))
```

Note, that this procedure therefore produces conditional estimates (also often called adjusted predictions) rather then (average) marginal (i.e. population averaged) estimates.[2] Since marginal estimates would average over survival predictions for each individual in the study, this would require a substantive number of additional calculations for each bootstrap replication and increase the estimation time dramatically. Consequently, `bsurvci` is restricted to conditional estimates, which are computationally much more efficient.

The estimation results of each bootstrap replication are then appended to the data to form a new dataset which contains a time and a survival function variable. This data records the estimated survival function for each bootstrap replication.

```
. keep S_est _t
. append using data.dta
```

This process is repeated for the desired number of replications:

```
. forvalues j=1/`replications´ {
```

2. For discussion of adjusted predictions versus marginal or population averaged effects in Stata see Williams (2012).

```
. preserve
. bsample, cluster(id)
. stcox var, nohr
. matrix b=e(b)
. predict s0, basesurv
. gen S_est=s0^(exp(b[1,1]))
. keep S_est _t
. append using data.dta
. save data.dta, replace
. restore
. }
```

Based on the bootstrap results, the confidence interval at each observed event time can be calculated. This is achieved using the percentile method, which has been found to produce good results for survival functions in simulation studies (Burr 1994, 1296). Let $i = 1, 2, \cdots, b$ be the bootstrap samples and $\widehat{S}_i(t)$ the estimated survival probability at event time t in boostrap sample i. Then the percentile method uses $[\widehat{S}_{\alpha/2}(t), \widehat{S}_{1-\alpha/2}(t)]$ as the $\alpha\%$ confidence interval, whereby $\widehat{S}_p(t)$ is the pth percentile of the bootstrap distribution $(\widehat{S}_1(t), \cdots, \widehat{S}_b(t))$ (StataCorp 2015, 234f.).

For 90% confidence intervals, this is implemented using the code below. The data containing the estimated upper and lower bound for each time point can be saved and used for visualizations or further calculations.

```
. bysort _t: egen lower=pctile(S_est), p(5)
. bysort _t: egen upper=pctile(S_est), p(95)
. collapse (min) lower (max) upper, by(_t)
. save data, replace
file data.dta saved
```

4 The `bsurvci` command

The bsurvci command provides a wrapper for the procedure using `stcox` with `predict, basesurv` as well as the `scurve_tvc` command (Ruhe 2016). In the case of proportional hazards, the `stcox` with `predict, basesurv` procedure described above is used. The `scurve_tvc` command is applied if the model contains time-varying effects. To facilitate the calculation of time-varying effects, the commands syntax closely follows the syntax for `scurve_tvc`. I describe the syntax, options and illustrative examples below.

4.1 Syntax

`bsurvci` [*if*] [*in*] , `generate(`*newvarname*`)`
`at(`*varname* # [*varname* # ...]`) id(`*varname*`)` [`tvc(`*varlist*`) texp(`*string*`)`
`replace ties(`*string*`) shared(`*varname*`) strata(`*varname*`) reps(`#`)`
`level(`#`) graph plotopts(`*string*`)`]

4.2 Description

`bsurvci` fits `stcox` and uses the bootstrap to calculate the pointwise confidence intervals of the survival curve for specific covariate values.

4.3 Options

`generate(`*newvarname*`)` creates the variable *newvarname* to store the estimated survival curve and *newvarname_lb* as well as *newvarname_ub* to store the respective upper and lower bound. If you also specify `strata()`, then bsurvci creates variables for each stratum. `generate()` is required.

`at(`*varname # [varname # ...]*`)` specifies the covariates included in the model and the values for which the survival curve should be calculated. Specifying `at1(...) [at2(...)]` etc. allows to calculate results for up to four covariate constellations simultaneously. Specifying either `at()` or `at1()` is required.

`id(`*varname*`)` specifies an ID variable to ensure that multiple-record data are treated as one subject when the bootstrap is performed. `id()` is required. For single-record data an ID variable can be created using `generate id_var=_n`.

`tvc(`*varlist*`)` specifies 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 estimated model. See `help tvc note` for more information.

`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.

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

`ties(`*string*`)` specifies the option how `stcox` handles tied failure times. See `help stcox` for details.

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

`strata(`*varname*`)` specifies a strata ID variables. See `help stcox` for details.

`reps(`#`)` perform # bootstrap replications; default is `reps(1000)`

`level(`#`)` set confidence level; default is `level(95)`

`graph` plots the predicted survival curve. If `strata()` is specified, the survival estimates for each stratum will be plotted.

`plotopts(`*string*`)` enables to customize the plot using options allowed with `twoway line`.

5 Examples

5.1 Basic use

To use the command, we need to have an id variable which uniquely identifies each entry in the data. For multiple-record data, the id variable specified in `stset` has to be used. For single-record data, an id variable can be easily created and used to `stset` the data again. Here we use Stata's single-record example data for patient survival in a drug trial:

```
. webuse drugtr
(Patient Survival in Drug Trial)
. generate id_var=_n
. stset studytime, failure(died) id(id_var)
```

(*output omitted*)

The command can be executed without having estimated a model. The command estimates the model and provides the output for the estimated model. Subsequently, the bootstrap replications are performed and the progress of these replications is documented.

```
. bsurvci, id(id_var) generate(survival) at(drug 1 age 58)
The estimation is based on the following Cox Proportional Hazards Model:
         failure _d:  died
   analysis time _t:  studytime
                 id:  id_var
Iteration 0:   log likelihood = -99.911448
Iteration 1:   log likelihood = -83.551879
Iteration 2:   log likelihood = -83.324009
Iteration 3:   log likelihood = -83.323546
Refining estimates:
Iteration 0:   log likelihood = -83.323546
Cox regression -- Breslow method for ties
No. of subjects =           48                 Number of obs    =          48
No. of failures =           31
Time at risk    =          744
                                               LR chi2(2)       =       33.18
Log likelihood  =   -83.323546                 Prob > chi2      =      0.0000
------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        drug |  -2.254965   .4548338    -4.96   0.000    -3.146423   -1.363507
         age |   .1136186   .0372848     3.05   0.002     .0405416    .1866955
------------------------------------------------------------------------------

Bootstrap replications (1000)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................50
..................................................100
..................................................150
..................................................200
..................................................250
..................................................300
```

```
.................................................350
.................................................400
.................................................450
.................................................500
.................................................550
.................................................600
.................................................650
.................................................700
.................................................750
.................................................800
.................................................850
.................................................900
.................................................950
................................................1000
```

The command adds three new variables to the existing data: the point estimate *newvarname*, the estimated lower and upper bound *newvarname_lb* as well as *newvarname_ub*. In the example above these estimates are:

```
. sort _t
. list _t survival_lb survival survival_ub in 1/15
```

	_t	survi~lb	survival	survi~ub
1.	1	.9658983	.9898074	.9974372
2.	1	.9658983	.9898074	.9974372
3.	2	.9566204	.9841047	.9967411
4.	3	.9423828	.9781718	.9950214
5.	4	.9207626	.9647725	.9904829
6.	4	.9207626	.9647725	.9904829
7.	5	.8983182	.9481602	.9828004
8.	5	.8983182	.9481602	.9828004
9.	6	.8421911	.9295208	.977133
10.	6	.8421911	.9295208	.977133
11.	6	.8421911	.9295208	.977133
12.	7	.788717	.9199162	.9645498
13.	8	.7906908	.8889439	.9685575
14.	8	.7906908	.8889439	.9685575
15.	8	.7906908	.8889439	.9685575

These three variables can be used to create flexible customized plots of the estimated survival function. Lets assume we want to plot the survival estimates at each observed failure time, since we only have information about the survival estimate at these time points. The following code creates Figure 1, which displays this information:[3]

```
. twoway rcap survival_lb survival_ub _t, xtitle("Analysis time") /*
>         */ legend(order(2 1) label(2 "survival estimate") /*
>         */ label(1 "95% CI")) || scatter survival _t, scheme(sj)
```

3. With a limited number of replications, the confidence intervals for a later point may sometimes exceed the bounds estimated for previous failure times. This is a result of outliers in the bootstrap distribution and should be seen as a sign that a larger number of replications is needed.

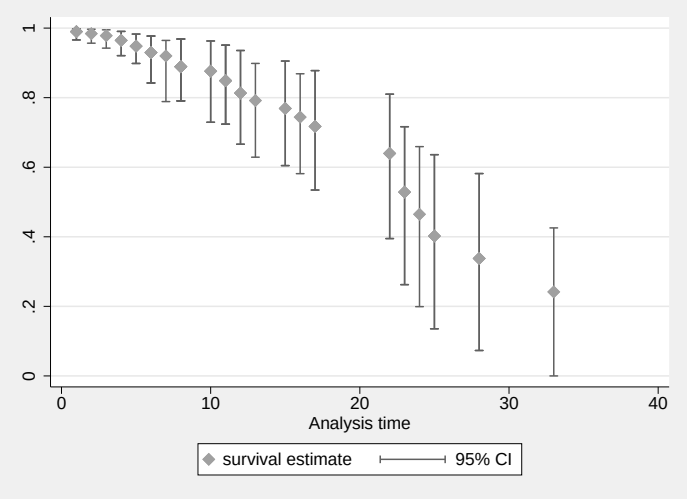

Figure 1: Survival estimate for a 58 year old patient with the drug treatment.

5.2 Comparing survival estimates

In many situations, it is useful to compare the estimates for different scenarios. In the example data, this could be a comparison of the drug treatment relative to the placebo. To generate such a comparison, we can simply rerun the command with the alternative covariate values. It is also possible to specify up to four sets of covariate values in one command through the `at1()`, `at2()` etc. options. This latter option is especially relevant for very large datasets for which model estimation is time-intensive.

In order to compare the previous estimate of a 58 year old patient with drug treatment to a person of similar age without the treatment, we can rerun the command by simply adjusting the covariate value of the treatment variable to 0.

```
. bsurvci, id(id_var) generate(survival0) at(drug 0 age 58)
```
(*output omitted*)

Alternatively, if we had not already estimated the prediction for the treatment case in the previous section, we could have specified the `at1()` and `at2()` options to save estimation time:[4]

```
. bsurvci, id(id_var) generate(survival) at1(drug 0 age 58) at2(drug 1 age 58)
```
(*output omitted*)

The estimation results for the drug and the placebo treatment are displayed in Figure 2. The following code produces the graph:

```
. twoway rcap survival_lb survival_ub _t || /*
```

4. In this case the the variables would have been called *survival1* and *survival2* respectively.

```
>         */ scatter survival _t || /*
>         */ rcap survival0_lb survival0_ub _t || /*
>         */ scatter survival0 _t, legend(order(2 1 4 3) label(4 "placebo") /*
>         */ label(2 "drug") label(1 "95% CI") label(3 "95% CI"))  scheme(sj) /*
>         */ xtitle("Analysis time") ytitle("Predicted survival probability")
```

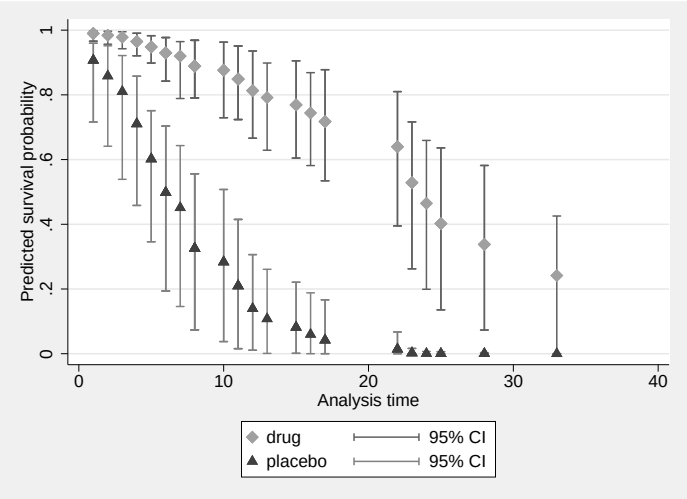

Figure 2: Survival estimate for a 58 year old patient depending on treatment.

5.3 The `graph` and `plotopts()` options

The `graph` option allows to plot the survival function and its estimated confidence intervals.

```
. bsurvci, id(id_var) replace generate(survival0) at(drug 0 age 58) graph
```
(*output omitted*)

Without any further specification, the line plot uses the default line pattern and colors of the scheme. To produce a more refined graph, the `plotopts()` option allows to customize the graph. Any option allowed with `twoway line` can be used. The order of the variables is *newvarname_lb*, *newvarname_ub*, *newvarname*. Hence, the third entry in e.g. `lpattern()` modifies the line pattern of the point estimate. The first two change the look of the confidence interval. Figure 3 displays the results of the following, customized graph:

```
. bsurvci, id(id_var) replace generate(survival0) at(drug 0 age 58) graph
plotopts(scheme(sj) lpattern(dash dash solid) lcolor(gs8 gs8 black) title(""))
```
(*output omitted*)

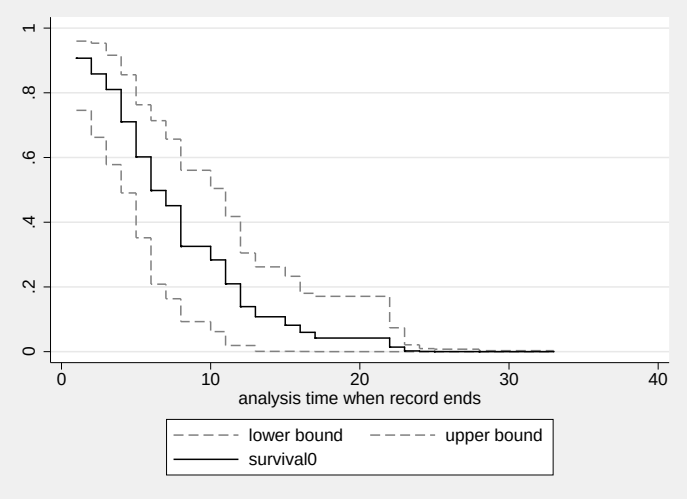

Figure 3: Survival estimate and 95%-confidence intervals produced with the `graph` and `plotopts()` options

5.4 Time-varying coefficients

The command can be used with time-varying coefficients for which confidence intervals were not yet available in Stata. The syntax is equivalent to the existing `scurve_tvc` command (Ruhe 2016). The values of the variables with time-varying coefficients are set in `at()`. Additionally, they are designated as time-varying in `tvc()`. The functional form of the time-interaction needs to be specified in `texp()`. All other options are equivalent to the use for models with proportional hazards. Figure 4 shows the results for a hypothetical time-varying gender difference in Stata's catheter example data.

```
. webuse catheter
(Kidney data, McGilchrist and Aisbett, Biometrics, 1991)
. generate id=_n
. stset time, failure(infect) id(id)
  (output omitted)
.
.
.
. bsurvci, id(id) generate(S_tvc) at(female 1 age 43) tvc(female) /*
>         */ texp(ln(_t)) graph plotopts(scheme(sj) lpattern(dash dash solid) /*
>         */ lcolor(gs8 gs8 black) title("") ytitle("P(Without infection)") /*
>         */ legend(order(3 1) label(3 "Survival estimate") label(1 "95% CI")))

Dataset has been temporarily split at failure times
(50 failure times)
(1,575 observations (episodes) created)

The estimation is based on the following Cox Proportional Hazards Model:
```

```
        failure _d:  infect
  analysis time _t:  time
                id:  id

Iteration 0:   log likelihood = -188.44736
Iteration 1:   log likelihood = -179.69769
Iteration 2:   log likelihood = -179.09006
Iteration 3:   log likelihood = -179.08784
Iteration 4:   log likelihood = -179.08784
Refining estimates:
Iteration 0:   log likelihood = -179.08784

Cox regression -- Breslow method for ties

No. of subjects =           76                 Number of obs    =       1,651
No. of failures =           58
Time at risk    =         7424
                                               LR chi2(3)       =       18.72
Log likelihood  =   -179.08784                 Prob > chi2      =      0.0003

------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      female |  -4.198875   1.158097    -3.63   0.000    -6.468704   -1.929045
         age |   .0068936   .0089764     0.77   0.443    -.0106999     .024487
   _female_t |   .9247154   .3146905     2.94   0.003     .3079333    1.541497
------------------------------------------------------------------------------
Note: tvc-interactions denoted by _varname_t were interacted with ln(_t).


Bootstrap replications (1000)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
```

(*output omitted*)

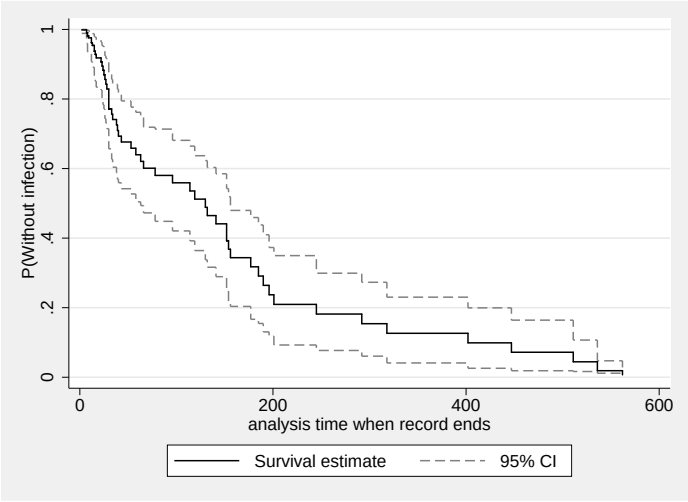

Figure 4: Survival estimate for 43 year old female patient, assuming a time-varying gender difference.

6 Conclusion

In this paper, I have described how to estimate bootstrap pointwise confidence intervals for covariate-adjusted survival functions based on the Cox model. The new user-written command allows Stata users to estimate and visualize survival functions and the associated uncertainty for Cox models with either proportional or non-proportional hazards. This fills a gap for Stata users who had no available methods to produce uncertainty estimates if the model contained covariates with non-proportional hazards, e.g. due to time-varying coefficients. Moreover, the new command provides users with the option of bootstrap confidence interval if the model contains only covariates with proportional hazards. This allows users to validate the assumptions and inference from asymptotic confidence intervals which assume specific distributions for the survival function's variance.

7 References

Burr, D. 1994. A Comparison of Certain Bootstrap Confidence-Intervals in the Cox Model. *Journal of the American Statistical Association* 89(428): 1290–1302.

Carpenter, J., and J. Bithell. 2000. Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Statistics in medicine* 19(9): 1141–1164.

Cefalu, M. 2011. Pointwise confidence intervals for the covariate-adjusted survivor function in the Cox model. *Stata Journal* 11(1): 64–81.

Cleves, M. A., W. Gould, R. G. Gutierrez, and Y. V. Marchenko. 2010. *An introduction to survival analysis using Stata*. 3rd ed. College Station, Tex.: Stata Press.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival analysis: Techniques for censored and truncated data*. 2nd ed. Statistics for biology and health, New York: Springer.

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(18): 2807–2821.

Royston, P., and P. C. Lambert. 2011. *Flexible parametric survival analysis using Stata: Beyond the Cox model*. College Station, Tex.: Stata Press.

Ruhe, C. 2016. Estimating survival functions after stcox with time-varying coefficients. *Stata Journal* 16(4): 867–879.

———. 2018. Quantifying Change Over Time: Interpreting Time-varying Effects In Duration Analyses. *Political Analysis* 26(01): 90–111.

StataCorp. 2015. *Stata 14 Base Reference Manual*. College Station, TX: Stata Press.

Williams, R. 2012. Using the margins command to estimate and interpret adjusted predictions and marginal effects. *Stata Journal* 12(2): 308–331.

About the authors

Constantin Ruhe is an Assistant Professor of Political Science at the Goethe University Frankfurt am Main, Germany. His main research interest are the quantitative analysis of political violence and conflict management as well as applied statistics.