The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*, *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
1-year subscription	$ 79	1-year subscription	$115
2-year subscription	$155	2-year subscription	$225
3-year subscription	$225	3-year subscription	$329
3-year subscription (electronic only)	$210	3-year subscription (electronic only)	$210
1-year student subscription	$ 48	1-year student subscription	$ 79
1-year university library subscription	$ 99	1-year university library subscription	$135
2-year university library subscription	$195	2-year university library subscription	$265
3-year university library subscription	$289	3-year university library subscription	$395
1-year institutional subscription	$225	1-year institutional subscription	$259
2-year institutional subscription	$445	2-year institutional subscription	$510
3-year institutional subscription	$650	3-year institutional subscription	$750

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The *Stata Journal* (ISSN 1536-867X) is a publication of Stata Press. Stata, Stata, Stata Press, Mata, Mata, and NetCourse are registered trademarks of StataCorp LP.

The Stata Journal (2012)
12, Number 4, pp. 639–654

Tools to simulate realistic censored survival-time distributions

Patrick Royston
Hub for Trials Methodology Research
MRC Clinical Trials Unit and University College London
London, UK
pr@ctu.mrc.ac.uk

Abstract. Simulation of realistic censored survival times is challenging. Most research studies use highly simplified models, such as the exponential, that do not adequately reflect the patterns of time to event and censoring seen in real datasets. In this article, I present a general method of simulating such data based on flexible parametric survival models (Royston and Parmar, 2002, *Statistics in Medicine* 21: 2175–2197). A key component of the approach is modeling not only the time to event but also the time to censoring. I illustrate the methods in data from clinical trials and from a prognostic study. I also describe a new Stata program, `stsurvsim`, that does the necessary calculations.

Keywords: st0274, stsurvsim, survival analysis, Monte Carlo simulation, flexible parametric survival models, time to event, time to censoring, clinical trials

1 Introduction

Simulation of censored time-to-event data is often required when investigating the properties of some procedure. In my experience, such simulations have featured in many research articles and PhD projects over the years. Mostly, presumably for simplicity, researchers reach for the exponential distribution to simulate times to event (henceforth "survival times", even though the event may be other than death). In some cases, the censoring distribution may also be represented by an exponential distribution, and the censored survival time may be computed as the minimum of the survival time and time to censoring.

Most real survival distributions do not resemble an exponential. In cancer, for example, we typically find that the hazard function peaks fairly soon after diagnosis and initial treatment of the disease and then declines gradually thereafter. Such a hazard function is neither a constant nor a monotonic function of time. Probably the second most popular distribution, the Weibull, has a monotonic hazard function, so it too is not a good choice for realistic simulation.

Why does it matter whether the simulated survival distribution is realistic? For example, simulations using a model that tends to produce extreme survival times might adversely affect the performance of some new method of analysis but be so unlikely to occur in practice that it leads to unsound conclusions. In addition, getting the censoring fraction and pattern right could be important in some studies.

In this article, I describe an approach to simulation based on creating pseudorandom samples from Royston–Parmar distributions (Royston and Parmar 2002; Royston and Lambert 2011). The defining characteristic of a Royston–Parmar distribution is that its baseline distribution function is modeled as a restricted cubic spline (RCS) function of log survival-time. Because spline functions are extremely flexible, the variety of available Royston–Parmar distributions is effectively limitless. One can also simulate the time to censoring as a Royston–Parmar distribution, making it possible to replicate censored survival-time distributions closely similar to that of a given real dataset.

2 Royston–Parmar models and software: A brief summary

2.1 Description

Let $S(t;\mathbf{x})$ be the survival function of interest and $\mathbf{x}$ be a vector of covariates. We wish to simulate samples from a distribution with survival function $S(t;\mathbf{x})$, but we do not necessarily know (or wish to assume a priori) the form of $S(t;\mathbf{x})$.

Suppose we write a broad class of models for $t|\mathbf{x}$ as follows

$$g\{S(t;\mathbf{x})\} = g\{S_0(t)\} + \boldsymbol{\beta}'\mathbf{x} \tag{1}$$

where $S_0(t) = S(t;\mathbf{0})$ is the baseline survival function and $g(\cdot)$ is some monotonic link function. The covariate effects in (1) are proportional on the scale of the link function, $g(\cdot)$.

Suppose we further write

$$g\{S_0(t)\} = s(\ln t) \tag{2}$$

where $s(\cdot)$ is an RCS function of log time. Equations (1) and (2) together define a basic Royston–Parmar model class.

Royston and Parmar (2002) described the model classes generated by three link functions: $\ln\{-\ln S(t)\}$, $\ln[\{1-S(t)\}/S(t)]$, and $\Phi^{-1}\{1-S(t)\}$, where $\Phi^{-1}(\cdot)$ is the inverse normal distribution function (`invnormal()` in Stata). These give rise to proportional hazards, proportional odds, and probit models, respectively. The simplest cases are where $s(\cdot)$ is a linear function of $\ln t$; then the distribution of $t|\mathbf{x}$ is Weibull, loglogistic, or lognormal, respectively, for the three link functions.

The RCS function $s(\ln t)$ has some number k of interior knots and two boundary knots. A "knot" is a join-point between two cubic polynomials on the time axis. The "restriction" in RCS refers to constraints that make the first and second derivatives continuous at the knots and force the spline function to be linear in its argument beyond the boundary knots. An RCS with k knots has $k+1$ degrees of freedom (d.f.), excluding the intercept. One of these d.f. is associated with a linear function of the argument, while the others are associated with the so-called spline basis functions.

We can write

$$s(\ln t) = \gamma_0 + \gamma_1 \ln t + \gamma_2 v_1(\ln t) + \cdots + \gamma_{k+1} v_k(\ln t)$$

where $\ln t$ and $v_1(\cdot), \ldots, v_k(\cdot)$ are the spline basis functions. Mathematical details of the basis functions are provided on pages 70–71, 97, and 102 of Royston and Lambert (2011), and a numerical example is given on pages 109–110.

The preferred software for fitting Royston–Parmar models is `stpm2` (Lambert and Royston 2009). The `stpm2` package can be downloaded from the Statistical Software Components archive by using the Stata command `ssc install stpm2`. The linear predictor for a Royston–Parmar model fit by `stpm2` is $s(\ln t) + \boldsymbol{\beta}'\mathbf{x}$; the γ and β parameters are part of a single "equation" and are estimated by maximum likelihood. A detail worth noting is that by default, `stpm2` orthogonalizes the basis functions so that their means are 0, their standard deviations are 1, and their correlations are 0. This linear transformation enhances numerical stability when fitting the model. The `stpm2` package includes a documented routine called `rcsgen`, which creates restricted cubic spline basis functions. We use `rcsgen` in this article.

Royston–Parmar models may be extended to include time-dependent effects of covariates; that is, the regression coefficients for a covariate may change over time. In this article, I do not consider time-dependent effects in a general way, and they are not supported by `stsurvsim`. There is one exception, where I model a time-dependent binary treatment effect in a clinical trial. This is done by simulating data for each arm independently. The use of Weibull distributions with a different shape parameter in each arm leads to nonproportional hazards.

2.2 Model selection

As discussed by Royston and Lambert (2011, 18–19), choosing an appropriate model for the survival data amounts to choosing an appropriate complexity (number of d.f.) for the RCS function representing the baseline distribution function. A simple way to do this is to choose the d.f. that minimizes a recognized information measure such as the Akaike's information criterion or the Bayesian information criterion (BIC). These statistics amount to penalized likelihoods. The BIC depends on sample size and is more stringent than the Akaike's information criterion, which has a fixed penalty independent of sample size. The BIC therefore tends to choose less complex RCS functions. For simulation purposes, the choice of d.f. is not absolutely critical, but underfitting is not advisable. Choosing the d.f. that approximately minimizes the BIC is a reasonable strategy in the present context.

3 Stata command stsurvsim

3.1 Syntax

`stsurvsim` *newvar* [*if*] [*in*] [, `beta(`*item* [*item* ...]`)` `bknots(`*numlist*`)`
`knots(`*numlist*`)` `knscale(`*string*`)` `rmatrix(`*matrix_name*`)` `scale(`*scale_name*`)`]

Weights are not allowed. Note that `stsurvsim` requires `stpm2` to be installed. `stpm2` can be downloaded from the Statistical Software Components archive (see [R] **ssc**).

3.2 Description

`stsurvsim` creates in *newvar* a simulated sample of uncensored survival times from the baseline distribution function of a Royston–Parmar flexible parametric survival model, implemented by `stpm2`. Covariate effects may optionally be included.

`stsurvsim` can be used in two different modes. If any of the options `beta()`, `bknots()`, `knots()`, or `scale()` are supplied, then they must all be supplied as well as possibly `rmatrix()`. If, however, a linear model is fit (no spline terms), knots and boundary knots are not required, and only `beta()` and `scale()` must be supplied. `stsurvsim` uses their information to define the model from which to simulate. Otherwise, none of these options are provided, and `stsurvsim` uses information from the most recent fit of `stpm2`.

When the beta coefficients for the spline basis functions relate to orthogonalized basis functions, you must supply an R-matrix in the `rmatrix()` option (see *Technical note*). `stsurvsim` has no way of checking whether an R-matrix is needed; it is up to you to provide it if necessary.

3.3 Options

`beta(`*item* [*item* ...]`)` specifies the regression coefficients for the spline basis functions and also for any covariates that may be required. The syntax of *item* is *name*[`=`]# (where *name* is the name of a covariate), `_rcs`# (where # is 1, 2, ... corresponding to the spline basis functions in the model), or `_cons` (for the intercept term). For example: `beta(_rcs1=2.751 _rcs2=0.229 _cons=-2.079)`. If `beta()` is not specified, the regression coefficients are picked up from the most recent fit of `stpm2`.

`bknots(`*numlist*`)` specifies the two boundary knots for the spline function on the scale of time or, if the `knscale(log)` option is used, log time. If `bknots()` is not specified, the boundary knots are picked up from the most recent fit of `stpm2`.

`knots(`*numlist*`)` specifies interior knots for the spline function on the scale of time or, if the `knscale(log)` option is used, log time. If `knots()` is not specified, the interior knots are picked up from the most recent fit of `stpm2`.

`knscale(`*string*`)` determines the scale of the knots supplied in `bknots()` and `knots()`. If `knscale()` is not specified, knots are given in units of time. If `knscale(log)` is specified, knots are provided in units of log time. The default is `knscale("")`, meaning knots in units of time.

`rmatrix(`*matrix_name*`)` supplies the matrix used to orthogonalize the spline basis functions. If the `noorthog` option of `stpm2` was not used when fitting the original model, orthogonalization is done by default. It is then essential to save the R-matrix after running `stpm2` and supply it to `stsurvsim` as `rmatrix(`*matrix_name*`)`. `stpm2` stores the R-matrix in `e(R_bh)`.

`scale(`*scale_name*`)` specifies the scale on which the Royston–Parmar model was fit. Valid *scale_name*s are `hazard`, `odds`, and `normal`. The `scale(theta)` option of `stpm2` is not supported. If `scale()` is not specified, the scale is picked up from the most recent fit of `stpm2`.

3.4 Technical note

If a Royston–Parmar model is fit by `stpm2`, the spline basis functions, created in variables called `_rcs1`, `_rcs2`, etc., are orthogonalized by default. A linear transformation that does this (an R-matrix) is stored by `stpm2` in `e(R_bh)`. To obtain correct simulations, you need to store this matrix after running `stpm2` and supply it to `stsurvsim` in the `rmatrix(`*matrix_name*`)` option. Failure to do this will produce incorrect simulations.

You can avoid the bother of the R-matrix by specifying the `noorthog` option when running `stpm2`. The `rmatrix()` option of `stsurvsim` is then unnecessary.

4 Example 1: Simulating survival data to resemble a given dataset

Suppose we wish to generate simulated samples of censored survival-time data that are similar in distribution to a given dataset. We use the German breast cancer dataset, available in Stata through `webuse brcancer`, as an example.

For this to work, we need to approximate the time-to-event and the time-to-censoring distributions separately. First, we `stset` recurrence-free survival time in units of years (on average, 365.24 days). We then fit a Royston–Parmar model with scale hazard and a spline basis of 2 d.f.

```
. webuse brcancer
(German breast cancer data)
. stset rectime, failure(censrec) scale(365.24)
     failure event:  censrec != 0 & censrec < .
obs. time interval:  (0, rectime]
 exit on or before:  failure
    t for analysis:  time/365.24
------------------------------------------------------------------------------
      686  total obs.
        0  exclusions
------------------------------------------------------------------------------
      686  obs. remaining, representing
      299  failures in single record/single failure data
 2112.036  total analysis time at risk, at risk from t =         0
                             earliest observed entry t =         0
                                  last observed exit t =  7.280145
```

We fit the desired `stpm2` model. We apply the `noorthog` option to avoid orthogonalization of the spline basis functions:

```
. stpm2, df(2) scale(hazard) noorthog
Iteration 0:   log likelihood = -676.96512
Iteration 1:   log likelihood = -676.24956
Iteration 2:   log likelihood = -676.24812
Iteration 3:   log likelihood = -676.24812
Log likelihood = -676.24812                     Number of obs   =        686
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
xb           |
       _rcs1 |    2.751215   .2602665    10.57   0.000     2.241102    3.261328
       _rcs2 |    .2291809   .0360612     6.36   0.000     .1585022    .2998595
       _cons |   -2.079276   .1021607   -20.35   0.000    -2.279507   -1.879045
------------------------------------------------------------------------------
```

We now display the additional information we need: knots stored in `'e(bhknots)'` and boundary knots in `'e(boundary_knots)'`:

```
. display "`e(bhknots)´"
1.768700062381288
. display "`e(boundary_knots)´"
.1971306421738497 6.724345909577711
```

By default, the interior knots are placed at certain predetermined centiles of the uncensored survival-time distribution. The model has 2 d.f. and just one knot. Referencing the `df()` option in `help stpm2` reveals that the knot is placed at the 50th centile of the uncensored survival times:

```
. centile _t if _d==1, centile(50)
                                                  -- Binom. Interp. --
    Variable |     Obs  Percentile      Centile    [95% Conf. Interval]
-------------+-------------------------------------------------------
          _t |     299         50       1.7687    1.574642    2.003994
```

The boundary knots are placed at the minimum and maximum event times:

```
. summarize _t if _d==1
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          _t |       299    2.184224    1.370769    .1971307   6.724346
```

We now have the information we need to simulate the uncensored times to event. We drop all those unnecessary decimal places:

```
. stsurvsim t_uncens, bknots(0.197 6.724) knots(1.769) scale(hazard)
> beta(_rcs1=2.751 _rcs2=0.229 _cons=-2.079)
[4 iterations performed]
```

The program informs us that four iterations were required. The algorithm in `stsurvsim` inverts the spline function to convert centiles back to the survival-time scale, which is an iterative process. Four iterations is typical and is very fast. The program fails with an error message if the maximum allowed number of 100 iterations is reached. This may well happen if you have fit a model in `stpm2` without specifying `noorthog`, and then you attempt to use the estimated parameter estimates to simulate.

We have seen all the details of how to simulate survival times in this example. In fact, because we just fit a model with `stpm2` and we are using the same model to generate the simulations, we could simply have typed

```
stpm2, df(2) scale(hazard) noorthog
stsurvsim t_uncens
```

In other words, you can use `stsurvsim` without options to get a similar result.

We now turn to simulating the censoring time distribution, for which we use a slightly more complex spline model with 3 d.f. The process is simple because all we need to do is reverse the censoring indicator, fit the `stpm2` model, and simulate:

```
replace _d = 1 - _d
stpm2, df(3) scale(hazard) noorthog
stsurvsim t_cens
```

Finally, we construct the simulated time to event and censoring indicator for the censored survival-time distribution:

```
generate t = min(t_uncens, t_cens)
generate byte d = cond(t_cens < t_uncens, 0, 1)
```

We are now ready to `stset` the dataset with `t` and `d` and to do whatever work we had in mind with the simulated distribution.

Figures 1(a) and 1(b) demonstrate that `stpm2` has achieved an excellent fit to the distribution of recurrence-free survival and of time to censoring, respectively.

Informally, figure 1(c) shows that the simulated distribution of censored time to event matches the original very well. More formally, we find that the BIC values with d.f. $= 1, 2, 3, 4, 5$ for the `censrec` failure event are 1410.8, 1369.6, 1372.3, 1375.2, and 1381.0, respectively. This confirms that `df(2)` is a good choice. For the time-to-censoring outcome, the BIC has a clear minimum at `df(3)` (values not shown).

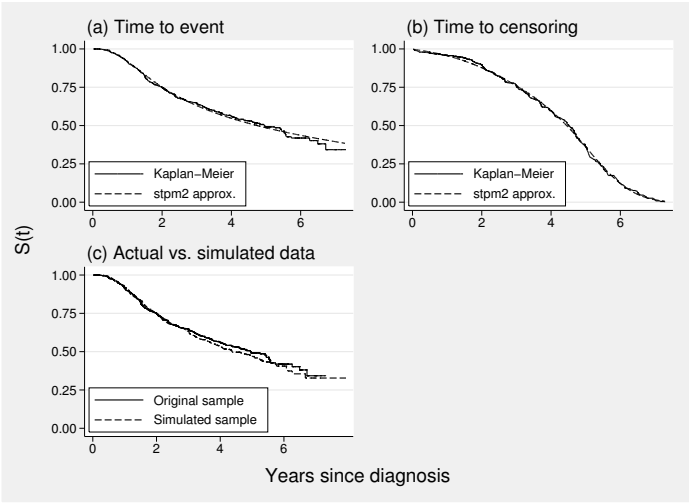

Figure 1. German breast cancer data. (a) Kaplan–Meier and `stpm` 2 d.f. fit to the survival function. (b) Kaplan–Meier and `stpm` 3 d.f. fit to the survival function for time to censoring. (c) Kaplan–Meier curves for the original sample and that simulated by `stsurvsim`.

5 Example 2: Simulating from published survival curves

The Iressa Pan-Asia Study (IPASS) (Mok et al. 2009), a randomized controlled trial in lung cancer, evoked some methodological comment in the letters section of the *New England Journal of Medicine* following its publication in 2009. The letters from four sets of correspondents and the authors' response may be found in the *New England Journal of Medicine* 361: 2485–2487 (2009). Panel A of figure 2 in the original article showed Kaplan–Meier curves for overall survival in the two treatment groups, exhibiting a treatment effect that was clearly nonproportional on the hazard scale (the survival curves crossed). A hazard ratio of 0.73 was reported. Correspondents subsequently complained about the inappropriateness of summarizing the treatment effect as a single hazard ratio when the hazard ratio is clearly varying over time.

As part of an analysis demonstrating the use of restricted mean survival time as an alternative outcome measure when the proportional hazards assumption is untenable, Royston and Parmar (2011) extracted survival values from figure 2 in Mok et al. (2009) and simulated data from the survival-time distributions in the two trial arms. They did this by fitting a Weibull distribution in each arm, which they accomplished by regressing the log cumulative hazard linearly on the log survival-time. The location and slope of the linear regressions gave rough estimates of the scale and shape parameters of the two Weibull distributions.

Table 1 gives the estimates of $S(t)$ that Royston and Parmar (2011) read in manually off the graph.

Table 1. Survival data extracted from panel A of figure 2 in Mok et al. (2009)

Control arm (carboplatin–paclitaxel)				Experimental arm (gefitinib)			
t	$S(t)$	t	$S(t)$	t	$S(t)$	t	$S(t)$
1	0.97	11	0.08	1	0.93	11	0.30
2	0.88	12	0.06	2	0.76	12	0.25
3	0.77	13	0.04	3	0.66	13	0.21
4	0.74	14	0.03	4	0.60	14	0.17
5	0.66	15	0.025	5	0.55	15	0.16
6	0.48	16	0.02	6	0.48	16	0.16
7	0.31	17	0.015	7	0.42	17	0.12
8	0.25	18	0.01	8	0.40	18	0.09
9	0.15	19	0.01	9	0.34	19	0.07
10	0.11	20	0.01	10	0.32	20	0.05

We adopted a similar approach to fitting, but instead of linear functions, we worked with spline functions of log time with 2 d.f. in each treatment group:

```
generate lnt = ln(t)
generate byte trt = (_n > 20)
rcsgen lnt, gen(_rcs) df(2)
display "`r(knots)´"
```

This gave two spline basis functions, `_rcs1` and `_rcs2`, with boundary knots at 0 and 2.996 log months and one interior knot at 2.350 log months. We next regressed the log cumulative hazard in each treatment group on the basis functions to estimate the coefficients needed by `stsurvsim`:

```
generate lnH = ln(-ln(s))
regress lnH _rcs1 _rcs2 if trt == 0
regress lnH _rcs1 _rcs2 if trt == 1
```

Using the estimated regression coefficients from these models, we simulated the survival-time distribution in each group separately and combined the results into a single variable. There were 1,217 patients in the original trial, 608 in the control arm, and 609 in the experimental arm. We suggest that to make the simulated values reproducible, you set the random-number seed before running `stsurvsim`:

```
drop _all
set obs 1217
set seed 111 /* arbitrary value */
generate byte trt = (_n > 608)
stsurvsim t0, scale(hazard) knscale(log) bknots(0 2.996) knots(2.35) ///
  beta(_rcs1=1.92 _rcs2=0.095 _cons=-3.52)
stsurvsim t1, scale(hazard) knscale(log) bknots(0 2.996) knots(2.35) ///
  beta(_rcs1=1.18 _rcs2=0.046 _cons=-2.37)
generate t = cond(trt==0, t0, t1)
drop t0 t1
```

Finally, we `stset` the data, truncating the survival time at 22 months, the final point shown in the original Kaplan–Meier graph:

```
generate byte d = 1
stset t, failure(d) exit(time 22)
```

Figure 2 shows the Kaplan–Meier plot for one resulting simulated dataset.

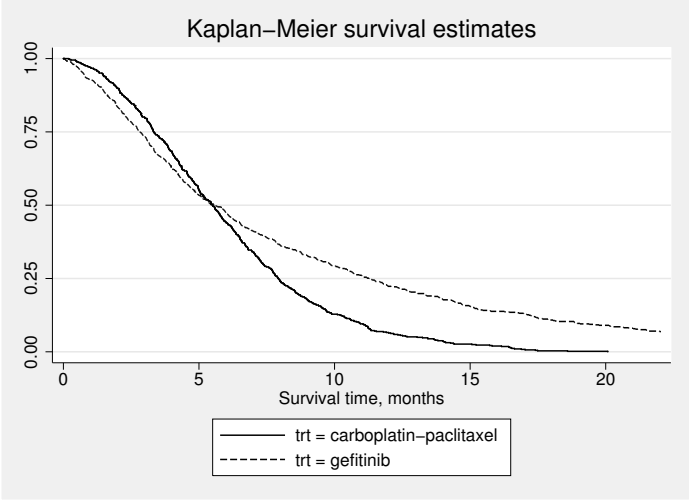

Figure 2. Simulated data resembling the IPASS trial in lung cancer

The pattern is qualitatively similar to the original. The survival curves cross at about 5–6 months.

6 Including covariates in the simulation

Including the effects of covariates on the simulated survival times is straightforward. Unlike the spline basis functions, the covariates must already exist in the data, so if they are to be generated by some additional simulation step, that process obviously needs to be completed before simulating the survival times. All that is required is to include in the `beta()` option for `stsurvsim` items of the form *varname*=#, where # is the regression coefficient for the desired effect of *varname* on the baseline distribution function. For example, if `trt` was a binary treatment variable with a log hazard-ratio of −0.3, we would include `trt=-0.3` in the `beta()` option and specify `scale(hazard)`.

Alternatively, we can simply fit a Royston–Parmar model including the covariates of interest by using `stpm2` and then run `stsurvsim` without options to generate simulations under the model, automatically including covariate effects.

As illustrated in example 1, the procedure can be applied with the censoring indicator reversed to generate realistic censoring patterns. If necessary, covariates that influence the censoring distribution can be included in the Royston–Parmar model.

7 Example 3: Multivariable modeling

Our final example is a little more complex. Again using the German breast cancer data, we simulate censored survival times from Sauerbrei and Royston's (1999) model III, a prognostic model. The variables, their fractional polynomial transformations, and the estimated regression coefficients from a Cox model are given in table 4 of Sauerbrei and Royston (1999). As in example 1, we simulate the censoring time distribution and the time to event, the main outcome of interest being recurrence-free survival time.

A simple aim of the simulation is to assess the stability of the model by estimating the selection fraction of each potentially influential covariate in the dataset. Sauerbrei and Royston (1999) did this by taking a large number of bootstrap samples of the dataset, running the `mfp` command on each sample to select a model, and counting the number of times each covariate appeared in the model after applying `mfp`'s backward-elimination selection procedure. We repeat the process with 200 replications, the only difference being that we simulate the survival times from model III rather than by bootstrap resampling. However, we still apply the bootstrap to resample the covariates.

Sauerbrei and Schumacher (1992) recommend a minimum of 200 bootstrap samples for a stability analysis. For the purpose of the present demonstration, we create 200 datasets containing the simulated times to event. We first get the parameters for simulating the times to censoring:

```
replace _d = 1 - _d
stpm2, df(3) scale(h) noorthog
local knots `e(bhknots)´
local bknots `e(boundary_knots)´
local rcs1 = [xb]_b[_rcs1]
local rcs2 = [xb]_b[_rcs2]
local rcs3 = [xb]_b[_rcs3]
local cons = [xb]_b[_cons]
local beta _rcs1=`rcs1´ _rcs2=`rcs2´ _rcs3=`rcs3´ _cons=`cons´
replace _d = 1 - _d
```

Next we fit model III by using `stpm2` and create the 200 datasets in 200 temporary files. The `bsample` command creates a bootstrap sample. The time-to-event variable `t_uncens` is created through the parameters that `stpm2` leaves behind:

```
webuse brcancer, clear
stset rectime, failure(censrec) scale(365.24)
fracgen x1 -2 -0.5
fracgen x6 0.5
stpm2 x1_1 x1_2 x4a x5e x6_1 hormon, df(2) scale(hazard)
local reps 200
forvalues j = 1/`reps´ {
  preserve
  bsample
  tempfile f`j´
  stsurvsim t_cens, bknots(`bknots´) knots(`knots´) beta(`beta´) scale(hazard)
  stsurvsim t_uncens
  generate t = min(t_cens, t_uncens)
  generate byte d = cond(t_cens < t_uncens, 0, 1)
  stset t d
  drop x4 x1_1 x1_2 x6_1 t_cens _rcs* _d_rcs* t_uncens t d
  save `f`j´´
  restore
}
```

We append the files together to produce a single file, indexing the replicates by a variable called `simind`:

```
use `f1´, replace
generate int simind = 1
forvalues j = 2 / `reps´ {
  append using `f`j´´
  replace simind = `j´ if missing(simind)
  erase `"`f`j´´"´
}
save mfpsim, replace
```

Finally, we run the user-written command `mfpboot` (Royston and Sauerbrei 2009) to apply `mfp` to the simulation replicates and record which variables are selected as well as their fractional polynomial transformations, if any:

```
use mfpsim, clear
mfpboot, select(0.05, hormon:1) clear outfile(mfpbootsim) replace seed(101) ///
  simid(simind): stcox x1 x2 x3 x4a x4b x5e x6 x7 hormon
```

A nominal significance level of 5% was used in the backward-elimination procedure, as indicated by the `select(0.05)` option. All models included hormonal treatment (`hormon`). The results are stored in `mfpbootsim.dta`. We can now count the selection fractions for the potential covariates. These are compared in table 2 with the corresponding results in table 6 of Sauerbrei and Royston (1999).

Table 2. Simulation-with-bootstrap and bootstrap-only results for prognostic modeling of the German breast cancer data

Variable	Description	Selection %		FP1 or linear %	
		Sim.	Boot.*	Sim.	Boot.*
`x1`	age	90	94	9	15
`x2`	menopausal status	10	18	–	–
`x3`	tumor size	7	43	5	38
`x4a`	tumor grade dummy 1	71	59	–	–
`x4b`	tumor grade dummy 2	7	9	–	–
`x5e`	transformed nodes	100	100	99	100
`x6`	progesterone receptors	100	99	95	92
`x7`	estrogen receptors	5	13	2	2

* Bootstrap-only results from table 6 of Sauerbrei and Royston (1999)

The selection fractions for `x1`, `x2`, `x4a`, `x4b`, `x5e`, `x6`, and `x7` are broadly similar between the simulation and bootstrap studies (although `x4a` is selected more often in the simulation study).

Note that the selection fraction of the uninfluential variables `x3`, `x4b`, and `x7` in the simulation study is close to the nominal 5%. The selection fraction of the uninfluential variable `x2` is 10%, which is above the nominal 5%. The reason is that `x2` is highly correlated with the influential variable `x1`. As a result, `x2` may be selected incorrectly in preference to `x1` in some replications.

The striking difference between the simulation and bootstrap results is for `x3` (tumor size). This variable was selected in nearly half of the bootstrap replications but in only 7% of simulations. There is a clear indication that `x3` should be included in a "final" model for the original data. Presumably, it was not selected more often because of a lack of power.

The simulation-with-bootstrap study is a nice sensitivity analysis of the bootstrap-only study under circumstances in which we know the true model. It helps us to judge which variables really are important. A strong case emerges from the bootstrap-only study for the prognostic importance of tumor size. This is in keeping with results from other, larger medical studies.

8 Further notes on stsurvsim

8.1 Use of the rmatrix() option

We have seen in examples how to use `stsurvsim`. As described above, its options are `bknots()`, `knots()`, `knscale()`, `beta()`, `scale()`, and `rmatrix()`. These options are only needed when we are providing the required values ourselves rather than relying on `stpm2` to supply them.

The only option we have not illustrated is `rmatrix()`. Here is an example, an alternative to the first example in which we now omit the `noorthog` option of `stpm2`. Because the spline basis functions are (by default) being orthogonalized, we need to store the R-matrix and use it when generating the simulated data.

```
stpm2, df(2) scale(hazard)
matrix R = e(R_bh)
stsurvsim t_uncens, bknots(0.197 6.724) knots(1.769) scale(hazard) ///
  beta(_rcs1=1.447 _rcs2=0.418 _cons=-1.284) rmatrix(R)
```

Notice that because of orthogonalization, the regression coefficients for the spline basis functions have completely changed.

If we use `stsurvsim` without arguments after running `stpm2`, the R-matrix is handled automatically. The following would work correctly:

```
stpm2, df(2) scale(hazard)
stsurvsim t_uncens
```

`stsurvsim` also simulates survival times in the `df(1)` case. This amounts to simulating Weibull, loglogistic, or lognormal distributions for the three `scale()` cases (`hazard`, `odds`, and `normal`, respectively). The `bknots()` and `knots()` options are not needed.

8.2 Simulating samples of different sizes

In real-life simulations, we invariably wish to choose the sample size ourselves. You can start with an empty workspace and set the sample size to any desired number if you use `stsurvsim` and provide parameter values in the options `beta()`, etc. After you fit a model with `stpm2`, however, `stsurvsim` without options produces a simulated sample with the same number of observations as the "parent" dataset. This is unlikely to be what is wanted.

A way around the problem that retains the convenience of not specifying the parameter values explicitly is as follows:

1. Choose the desired sample size, say, n. Let the number of observations in the current dataset be $N \neq n$.
2. Fit the required `stpm2` model.

3. Place the data in random order by using `runiform()` to create a variable containing random numbers and sorting on it.

4. If $n \leq N$, then drop observations $n+1, \ldots, N$.

5. If $n > N$, then use Stata's `expand` m command to expand the data the requisite number $m > 1$ of times, namely, $m = 1 + \text{int}\{(N-1)/n\}$. There are now mN observations. Create a new variable containing random numbers and sort on it. Drop observations $N+1, \ldots, mN$.

6. Run `stsurvsim` without arguments. Finished.

9 Conclusion

I hope `stsurvsim` will set a new standard in simulating realistic censored survival data in Stata and will thereby be found useful in many research projects in survival analysis.

10 References

Lambert, P. C., and P. Royston. 2009. Further development of flexible parametric models for survival analysis. *Stata Journal* 9: 265–290.

Mok, T. S., Y.-L. Wu, S. Thongprasert, C.-H. Yang, D.-T. Chu, N. Saijo, P. Sunpaweravong, B. Han, B. Margono, Y. Ichinose, Y. Nishiwaki, Y. Ohe, J.-J. Yang, B. Chewaskulyong, H. Jiang, E. L. Duffield, C. L. Watkins, A. A. Armour, and M. Fukuoka. 2009. Gefitinib or Carboplatin–Paclitaxel in Pulmonary Adenocarcinoma. *New England Journal of Medicine* 361: 947–957.

Royston, P., and P. C. Lambert. 2011. *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*. College Station, TX: Stata Press.

Royston, P., and M. K. B. Parmar. 2002. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in Medicine* 21: 2175–2197.

———. 2011. The use of restricted mean survival time to estimate the treatment effect in randomized clinical trials when the proportional hazards assumption is in doubt. *Statistics in Medicine* 30: 2409–2421.

Royston, P., and W. Sauerbrei. 2009. Bootstrap assessment of the stability of multivariable models. *Stata Journal* 9: 547–570.

Sauerbrei, W., and P. Royston. 1999. Building multivariable prognostic and diagnostic models: Transformation of the predictors by using fractional polynomials. *Journal of the Royal Statistical Society, Series A* 162: 71–94.

Sauerbrei, W., and M. Schumacher. 1992. A bootstrap resampling procedure for model building: Application to the Cox regression model. *Statistics in Medicine* 11: 2093–2109.

About the author

Patrick Royston is a medical statistician with more than 30 years of experience and a strong interest in biostatistical methods and in statistical computing and algorithms. He works largely with methodological issues in the design and analysis of clinical trials and observational studies. He is currently focusing on alternative outcome measures in trials with a time-to-event outcome; on problems of model building and validation with survival data, including prognostic factor studies; on parametric modeling of survival data; on multiple imputation of missing values; and on novel clinical trial designs.