Simulating time-to-event data from parametric distributions, custom distributions, competing risk models and general multi-state models

Michael J. Crowther[1,2]

[1]Biostatistics Research Group,
Department of Health Sciences,
University of Leicester, UK
[2]Department of Medical Epidemiology and Biostatistics,
Karolinska Institutet, Stockholm, Sweden

17th September 2020

mjcrowther.co.uk michael.crowther@le.ac.uk @Crowther_MJ

Plan

- A brief overview of how to simulate a range of simple and complex time-to-event data
- A new, general framework for simulating from arbitrary multi-state models
- Lots of examples, illustrated with the `survsim` Stata package

Background

- Simulating data from a defined distribution can be simple, or can be extremely complex

Background

- Simulating data from a defined distribution can be simple, or can be extremely complex
- Assume we have a random variable T, with associated cumulative distribution function, $F(T)$

Background

- Simulating data from a defined distribution can be simple, or can be extremely complex
- Assume we have a random variable T, with associated cumulative distribution function, $F(T)$
- To simulate survival times from such a distribution, we let

$$F \sim U(0, 1)$$

Background

- Simulating data from a defined distribution can be simple, or can be extremely complex
- Assume we have a random variable T, with associated cumulative distribution function, $F(T)$
- To simulate survival times from such a distribution, we let

$$F \sim U(0, 1)$$

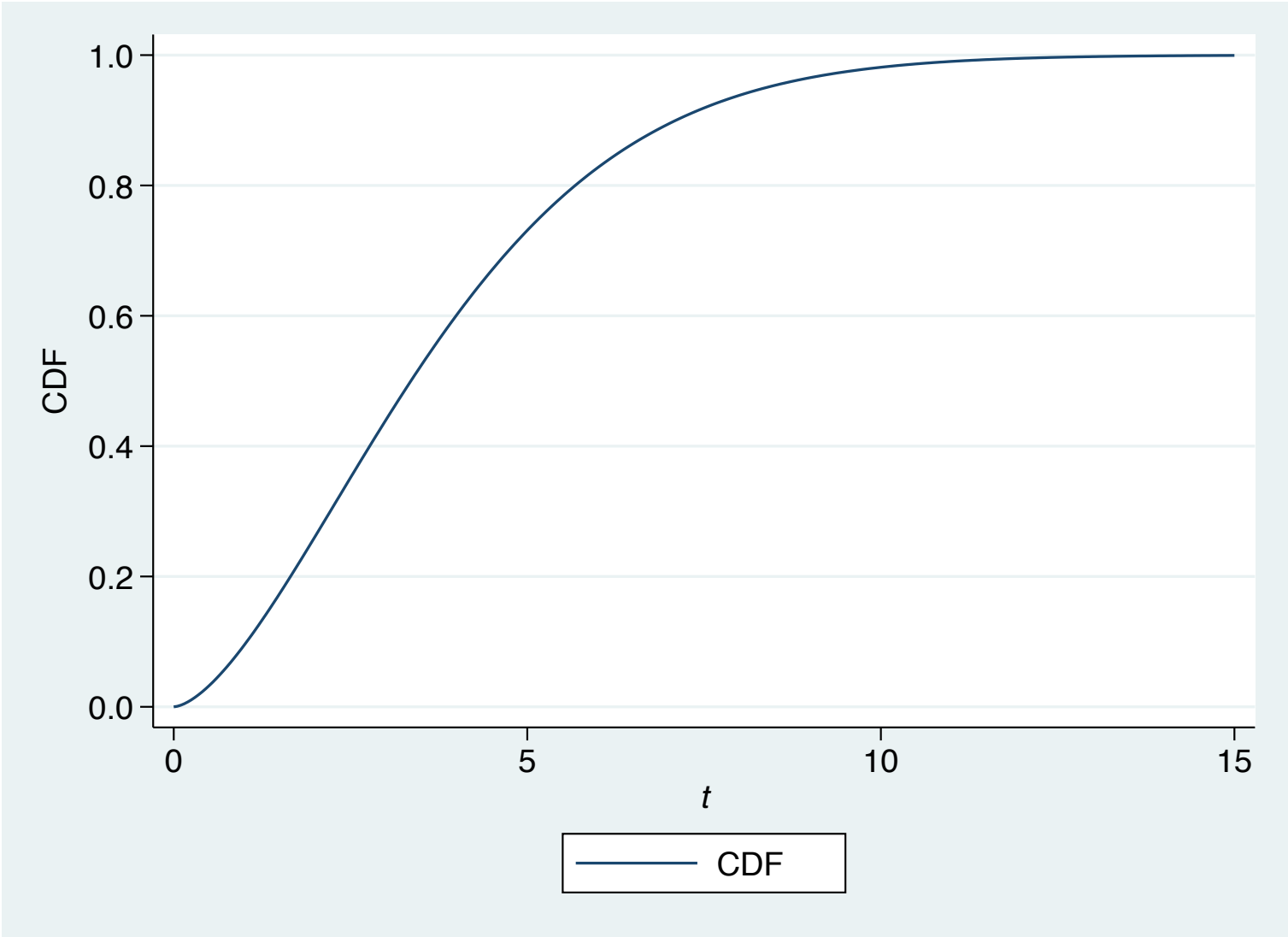
- To simulate an observation, we draw from the uniform distribution, say $u \sim U(0, 1)$, and substitute,

$$F(t) = u$$

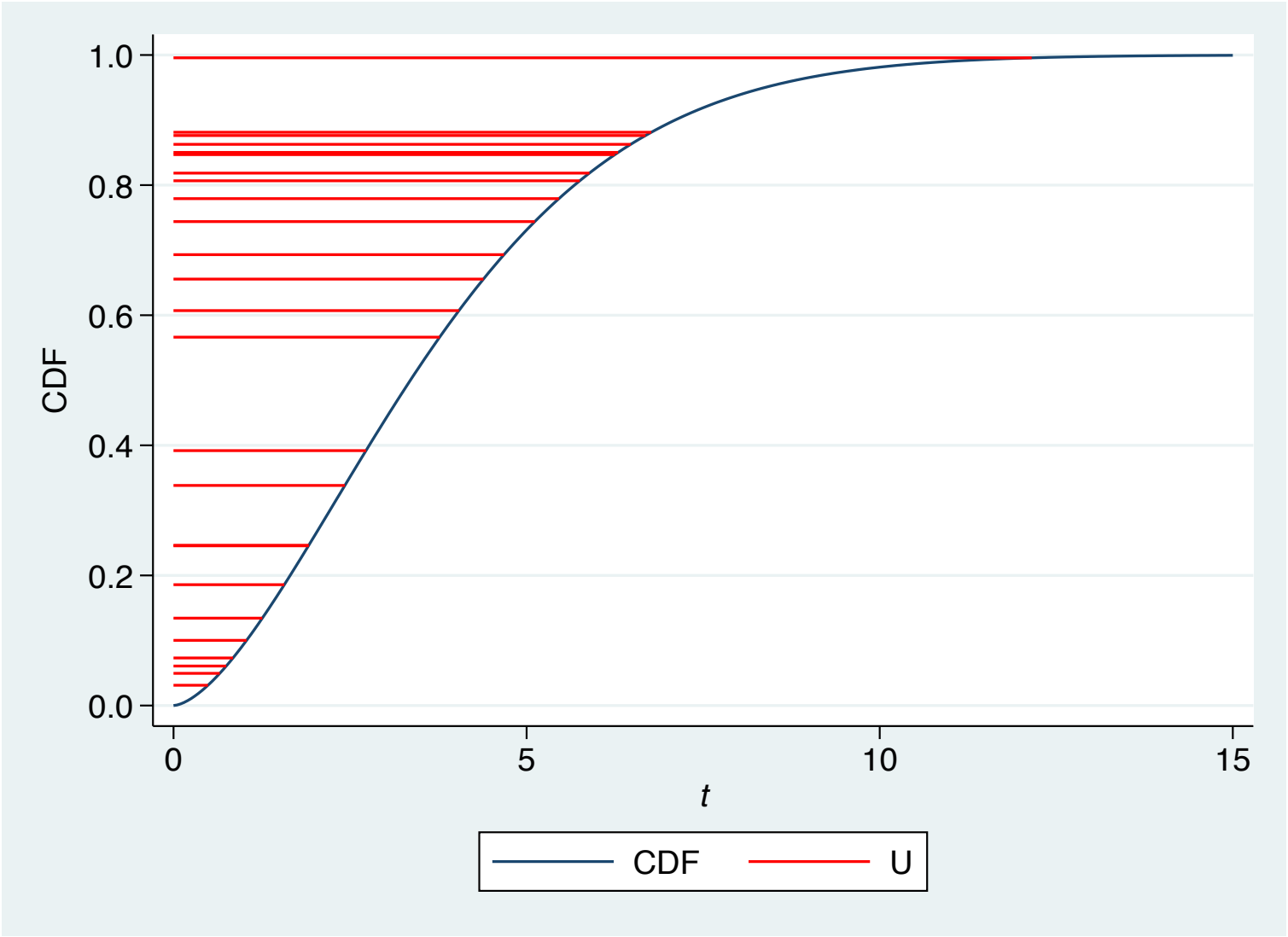
and solve for t

$$t = F^{-1}(u)$$

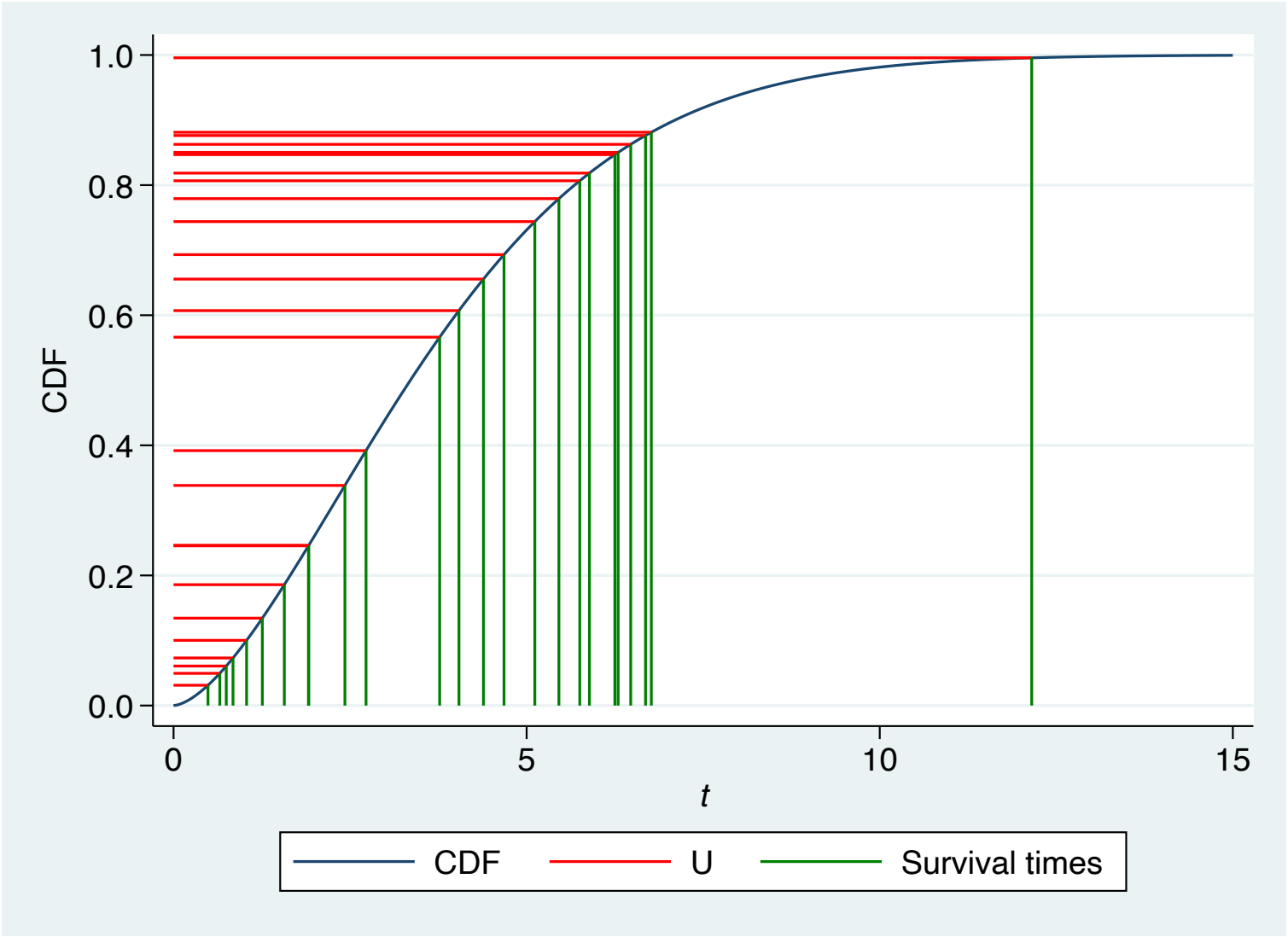
Background



Background



Background



Background

$$t = F^{-1}(u)$$

- Now solving for t relies on being able to invert the cumulative distribution function, and since it is a function of the cumulative hazard function, we must also be able to integrate our hazard function

$$F(t) = 1 - \exp[-H(t)]$$

$$u = 1 - \exp\left(-\int_0^t h(s)\, \mathrm{d}s\right)$$

Background

$$t = F^{-1}(u)$$

- Now solving for t relies on being able to invert the cumulative distribution function, and since it is a function of the cumulative hazard function, we must also be able to integrate our hazard function

$$F(t) = 1 - \exp[-H(t)]$$

$$u = 1 - \exp\left(-\int_0^t h(s)\, \mathrm{d}s\right)$$

- To accommodate these challenges, we developed a combined root-finding and numerical integration algorithm to provide an efficient method of generating event times from arbitrary distribution functions [1]

Background

- To be even more general, with this approach we can simulate from *any* hazard/cumulative hazard function

$$F(t|t_0) = 1 - \exp\left(-\int_{t_0}^{t} h(s|t_0)\ \mathrm{d}s\right)$$

Background

- To be even more general, with this approach we can simulate from *any* hazard/cumulative hazard function

$$F(t|t_0) = 1 - \exp\left(-\int_{t_0}^{t} h(s|t_0)\ \mathrm{d}s\right)$$

- For the competing risks and more general multi-state setting, this is used to simulate from the total hazard function (made up of the sum of cause/transition-specific hazard functions leaving a particular state) [2]

Examples

I will go through at least one example of how to use `survsim` to simulate survival data from each of the four main settings:

1. Simulating from standard parametric distributions
2. Simulating from a custom/user-defined hazard function
3. Simulating competing risks data
4. Simulating multi-state data

ssc install survsim

Description

survsim simulates survival data from:

help survsim parametric - a parametric distribution including the exponential, Gompertz and Weibull, and 2-component mixtures of them. Baseline covariates can be included, with specified associated log hazard ratios. Non-proportional hazards can also be included with all models; under an exponential or Weibull model covariates are interacted with log time, under a Gompertz model covariates are interacted with time. See **Crowther and Lambert (2012)** for more details.

help survsim user - a user-defined distribution. Survival times can be simulated from bespoke, user-defined [log] [cumulative] hazard functions. The function must be specified in Mata code (using colon operators), with survival times generated using a combination of numerical integration and root finding techniques. Time-dependent effects can also be specified with a user-defined function of time. See **Crowther and Lambert (2013)** for more details.

help survsim model - a fitted **merlin** model. **merlin** fits a broad class of survival models, including standard parametric models, spline-based survival models, and user-defined survival models.

help survsim msm - a competing risks or general multi-state model. Event times can be simulated from transition-specific hazards, where each transition hazard function can be a standard parametric distribution, or a user-defined complex hazard function. Covariates and time-dependent effects can be specified for each transition-specific hazard independently.

Simulating survival times from standard parametric distributions

Let's simulate survival times from a Weibull distribution, with a binary treatment group, `trt`, and a continuous covariate, age, under proportional hazards:

$$h(t) = \lambda\gamma t^{\gamma-1}\exp(\text{trt}\beta_1 + \text{age}\beta_2)$$

I'll simulate 300 observations, and pick some distributions for the covariates, which should be self-explanatory,

```
. clear
. set obs 300
. set seed 134987
. gen trt = runiform()>0.5
. gen age = rnormal(50,3)
```

I'll simulate 300 observations, and pick some distributions for the covariates, which should be self-explanatory,

```
. clear
. set obs 300
. set seed 134987
. gen trt = runiform()>0.5
. gen age = rnormal(50,3)
```

We then call `survsim`, setting $\lambda = 0.1$, $\gamma = 1.2$, $\beta_1 = -0.5$ and $\beta_2 = 0.01$,

```
. survsim stime, distribution(weibull) lambda(0.1) gamma(1.2)
>                covariates(trt -0.5 age 0.01)
```

which stores our simulated survival times in the new variable `stime`.

If we wanted to apply right-censoring, we could apply a common censoring time using for example `maxtime(5)`, which would censor all observations if their simulated event times was greater than 5, or we could generate observation specific potential censoring times, such as

```
. gen censtime = runiform() * 5
```

and now add the `maxtime()` option to `survsim`, remembering to also specify a second new variable name for the event indicator,

```
. survsim stime2 died2, distribution(weibull) lambda(0.1) gamma(1.2)     ///
>                       covariates(trt -0.5 age 0.01) maxtime(censtime)
```

If we wanted to apply right-censoring, we could apply a common censoring time using for example `maxtime(5)`, which would censor all observations if their simulated event times was greater than 5, or we could generate observation specific potential censoring times, such as

```
. gen censtime = runiform() * 5
```

and now add the `maxtime()` option to `survsim`, remembering to also specify a second new variable name for the event indicator,

```
. survsim stime2 died2, distribution(weibull) lambda(0.1) gamma(1.2)     ///
>                       covariates(trt -0.5 age 0.01) maxtime(censtime)
```

We could also:

- add time-dependent effects using the `tde()` option
- add left-truncation/delayed entry using the `ltruncated()` option

Simulating survival times from a user-defined (log) (cumulative) hazard function

```
. clear
. set obs 500
. set seed 134987
. gen trt = runiform()>0.5
```

The most flexible form of simulating survival data with `survsim` is by specifying a custom hazard or cumulative hazard function, such as:

$$h(t) = h_0(t)\exp(\text{trt}\beta_1)$$

where

$$h_0(t) = \exp(-1 + 0.02t - 0.03t^2 + 0.005t^3)$$

which can be done, on the `loghazard()` scale for simplicity, using

```
. survsim stime1 died1, loghazard(-1:+0.02:*{t}:-0.03:*{t}:^2:+0.005:*{t}:^3)  ///
>                       covariates(trt -0.5) maxtime(1.5)
Warning: 321 survival times were above the upper limit of maxtime()
         They have been set to maxtime()
         You can identify them by _survsim_rc = 3
```

- The `loghazard()` function is defined using Mata code, with colon operators representing element by element operations
- Time must be referred to using the {t} notation

We could make the treatment effect diminish over log time by incorporating a time-dependent effect, where

$$\beta_1(t) = \log(t)\beta_1$$

which is defined using the `tdefunction()` and `tde()` options, setting $\beta_1 = 0.03$

```
. survsim stime2 died2, loghazard(-1:+0.02:*{t}:-0.03:*{t}:^2:+0.005:*{t}:^3)
>                       covariates(trt -0.5) tde(trt 0.03) tdefunction(log({t}))
>                       maxtime(1.5)
Warning: 328 survival times were above the upper limit of maxtime()
         They have been set to maxtime()
         You can identify them by _survsim_rc = 3
```

- which will form an interaction between `trt`, its coefficient `0.03` and log time

We could make the treatment effect diminish over log time by incorporating a time-dependent effect, where

$$\beta_1(t) = \log(t)\beta_1$$

which is defined using the `tdefunction()` and `tde()` options, setting $\beta_1 = 0.03$

```
. survsim stime2 died2, loghazard(-1:+0.02:*{t}:-0.03:*{t}:^2:+0.005:*{t}:^3)
>                       covariates(trt -0.5) tde(trt 0.03) tdefunction(log({t}))
>                       maxtime(1.5)
Warning: 328 survival times were above the upper limit of maxtime()
         They have been set to maxtime()
         You can identify them by _survsim_rc = 3
```

- which will form an interaction between `trt`, its coefficient `0.03` and log time
- Alternatively, we could instead simulate from a model on the cumulative hazard scale, using the `logchazard()` option instead.

Simulating survival times from a fitted `merlin` survival model

Rather than simulating from a particular data-generating model specified essentially by hand, we can directly simulate from a fitted model.

```
. webuse brcancer, clear
(German breast cancer data)
. stset rectime, f(censrec=1) scale(365)
. stmerlin hormon , distribution(weibull)

Fitting full model:

Survival model                                  Number of obs     =        686
Log likelihood = -868.02684
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
_t:          |
      hormon |  -.3932405   .1248267    -3.15   0.002    -.6378962   -.1485847
       _cons |  -2.196012   .1094092   -20.07   0.000     -2.41045   -1.981574
  log(gamma) |   .2509974   .0496958     5.05   0.000     .1535953    .3483994
------------------------------------------------------------------------------
```

We then simply store the model object, calling it whatver we like, such as the imaginative name of `m1`,

```
. estimates store m1
```

This we then pass to `survsim` to simulate a dataset, of the same size, using our fitted results,

```
. survsim stime5 died5, model(m1) maxtime(7)
```

The option `maxtime()` is required in this case.

We can then fit the same model as before, and of course get slightly different results, because we have sampling variability.

```
. stset stime5, failure(died5)
. stmerlin hormon , distribution(weibull)

Survival model                                  Number of obs     =        686
Log likelihood = -1298.8059
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
_t:          |
      hormon |  -.3533907   .1014646    -3.48   0.000    -.5522576   -.1545238
       _cons |  -2.349859   .1100583   -21.35   0.000    -2.565569   -2.134148
  log(gamma) |   .2650626   .0421563     6.29   0.000     .1824377    .3476875
------------------------------------------------------------------------------
```

- Easy to manipulate your covariate distributions in your dataset, and then simply recall `survsim`

We can then fit the same model as before, and of course get slightly different results, because we have sampling variability.

```
. stset stime5, failure(died5)
. stmerlin hormon , distribution(weibull)

Survival model                                  Number of obs     =        686
Log likelihood = -1298.8059
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
_t:          |
      hormon |  -.3533907   .1014646    -3.48   0.000    -.5522576   -.1545238
       _cons |  -2.349859   .1100583   -21.35   0.000    -2.565569   -2.134148
  log(gamma) |   .2650626   .0421563     6.29   0.000     .1824377    .3476875
------------------------------------------------------------------------------
```

- Easy to manipulate your covariate distributions in your dataset, and then simply recall `survsim`
- `survsim` can simulate from any of the available survival models in `merlin`, but does not support simulation from multivariate models or a model containing random effects

Simulating competing risks data from specified cause-specific hazard functions

Let's simulate from a competing risk model with 2 competing events. The first cause-specific hazard has a Weibull distribution, with no covariates. The second cause-specific hazard model has an exponential distribution, with a beneficial treatment effect. Right censoring is applied at 10 years.

```
. clear
. set seed 398
. set obs 1000
. gen trt = runiform()>0.5
. survsim time state event , hazard1(dist(weibull) lambda(0.1) gamma(0.8))  ///
>                            hazard2(dist(exponential) lambda(0.02)         ///
>                            covariates(trt -0.5)) maxtime(10)
variables time0 to time1 created
variables state0 to state1 created
variables event1 to event1 created
```

- Each `hazard#()` defines a cause-specific hazard function, with specified `distribution()` and associated baseline parameters, covariate effects and time-depdent effects

- Each `hazard#()` defines a cause-specific hazard function, with specified `distribution()` and associated baseline parameters, covariate effects and time-depdent effects
- Each of the `hazard#()` functions can be as similar, or different, as required

- Each `hazard#()` defines a cause-specific hazard function, with specified `distribution()` and associated baseline parameters, covariate effects and time-depdent effects
- Each of the `hazard#()` functions can be as similar, or different, as required
- `survsim` creates some new variables, based on the *newvarstubs* that we specified in the call.

```
. list if _n<=5
     +-----------------------------------------------------+
     | trt   time0   state0       time1   state1   event1 |
     |-----------------------------------------------------|
  1. |   0       0        1   4.5792847        2        1 |
  2. |   1       0        1          10        1        0 |
  3. |   1       0        1          10        1        0 |
  4. |   1       0        1   2.8415219        3        1 |
  5. |   0       0        1    1.576534        2        1 |
     +-----------------------------------------------------+
```

- The starting time can be changed using the `ltruncated()` option.

From the starting state, observations have two places to go:

- State 1 to State 2, with the transition rate governed by `hazard1()`
- State 1 to State 3, with the transition rate governed by `hazard2()`

From the starting state, observations have two places to go:

- State 1 to State 2, with the transition rate governed by `hazard1()`
- State 1 to State 3, with the transition rate governed by `hazard2()`

We can see which events occurred with

```
. tabulate state1 event1

           |        event1
    state1 |         0           1 |     Total
-----------+----------------------+----------
         1 |       484           0 |       484
         2 |         0         414 |       414
         3 |         0         102 |       102
-----------+----------------------+----------
     Total |       484         516 |     1,000
```

which shows that at by ten years, 484 observations were right-censored, 414 are in State 2, and 102 are in State 3.

Now let's simulate from a competing risk model with 3 competing events...

```
. cap drop time* state* event*
. set seed 32984575
. survsim time state event,                                          ///
>         hazard1(user(exp(-2 :+ 0.2:* log({t}) :+ 0.1:*{t}))  ///
>                 covariates(trt 0.1))                               ///
>         hazard2(dist(weibull) lambda(0.01) gamma(1.3)        ///
>                 covariates(trt -0.5))                              ///
>         hazard3(user(0.1 :* {t} :^ 1.5) covariates(trt -0.5) ///
>                 tde(trt 0.1) tdefunction(log({t})))          ///
>         maxtime(3)
variables time0 to time1 created
variables state0 to state1 created
variables event1 to event1 created
```

Where did they go...

```
. tabulate state1 event1

           |        event1
    state1 |         0           1 |     Total
-----------+----------------------+----------
         1 |       341           0 |       341
         2 |         0         345 |       345
         3 |         0          30 |        30
         4 |         0         284 |       284
-----------+----------------------+----------
     Total |       341         659 |     1,000
```

I currently let you use up to 50 cause-specific hazards, just in case you're feeling particularly adventurous.

Simulating from an illness-death model

We first define the transition matrix for an illness-death model. It has three states:

- State 1 - A "healthy" state. Observations can move from state 1 to state 2 or 3.
- State 2 - An intermediate "illness" state. Observations can come from state 1, and move on to state 3.
- State 3 - An absorbing "death" state. Observations can come from state 1 or 2, but not leave.

This gives us three potential transitions between states:

- Transition 1 - State 1 $\rightarrow$ State 2
- Transition 2 - State 1 $\rightarrow$ State 3
- Transition 3 - State 2 $\rightarrow$ State 3

which is defined by the following matrix:

```
. matrix tmat = (.,1,2\.,.,3\.,.,.)
```

This gives us three potential transitions between states:

- Transition 1 - State 1 $\rightarrow$ State 2
- Transition 2 - State 1 $\rightarrow$ State 3
- Transition 3 - State 2 $\rightarrow$ State 3

which is defined by the following matrix:

```
. matrix tmat = (.,1,2\.,.,3\.,.,.)
```

Let's make it more clear

```
. mat colnames tmat = "healthy" "ill" "dead"
. mat rownames tmat = "healthy" "ill" "dead"
. mat list tmat
tmat[3,3]
         healthy      ill     dead
healthy        .        1        2
    ill        .        .        3
   dead        .        .        .
```

Now we've defined the transition matrix, we can use `survsim` to simulate some data

```
. clear
. set obs 1000
number of observations (_N) was 0, now 1,000
. set seed 9865
. gen trt = runiform()>0.5
```

This time I add the `transmat()` option...

```
. survsim time state event, transmatrix(tmat)                    ///
>         hazard1(user(exp(-2 :+ 0.2:* log({t}) :+ 0.1:*{t}))  ///
>                 covariates(trt 0.1))                           ///
>         hazard2(dist(weibull) lambda(0.01) gamma(1.3)          ///
>                 covariates(trt -0.5))                          ///
>         hazard3(user(0.1 :* {t} :^ 1.5) covariates(trt -0.5) ///
>                 tde(trt 0.1) tdefunction(log({t})))            ///
>         maxtime(3)
variables time0 to time2 created
variables state0 to state2 created
variables event1 to event2 created
```

We can see what `survsim` has created:

```
. list if inlist(_n,1,4,16,112), compress

     +-------------------------------------------------------------------------+
     | trt   time0   sta~0       time1   sta~1   eve~1       time2   sta~2   eve~2 |
     |-------------------------------------------------------------------------|
  1. |   0       0       1           3       1       0           .       .       . |
  4. |   1       0       1   .95636156       2       1           3       2       0 |
 16. |   0       0       1   1.0755764       2       1   2.4401409       3       1 |
112. |   1       0       1   2.3290322       3       1           .       .       . |
     +-------------------------------------------------------------------------+
```

All observations start initially in state 1 at time 0, which are stored in `state0` and `time0`, respectively. Then,

- Observation 1 is right-censored at 3 years, remaining in state 1
- Observation 4 moves to state 2 at 0.956 years, and is subsequently right-censored at 3 years, still in state 2
- Observation 16 moves to state 2 at 1.076 years, and then moves to state 3 at 2.440 years. Since state 3 is an absorbing state, there are no further transitions
- Observation 112 moves to state 3 at 2.329 years. Again, since state 3 is absorbing, there are no further transitions

There's a variety of extensions we could incorporate,

- we could simulate from a semi-Markov model by using the `reset` option in `hazard3()`, which would reset the clock when State 2 is entered. The simulated event times that `survsim` returns will still be calculated on the main timescale in this case, time since initial `startstate()`
- We could of course have a much more complex multi-state structure, i.e. more states or reversible transitions. Both of these are supported by `survsim`

Conclusion

- Simulating biologically plausible time to event data is crucial to understanding, evaluating and developing methods
- `survsim` can simulate survival data from pretty much anything you can think of...
- Paper - https://www.mjcrowther.co.uk/publication/survsim/
- Code and examples - https://www.mjcrowther.co.uk/software/survsim/

References

[1] Crowther MJ, Lambert PC. Simulating biologically plausible complex survival data. *Stat Med* 2013;**32**:4118–4134.

[2] Beyersmann J, Latouche A, Buchholz A, Schumacher M. Simulating competing risks data in survival analysis. *Stat Med* 2009;**28**:956–971.