

Simple and complex survival analysis: New developments in merlin

Michael J. Crowther

Associate Professor of Biostatistics
Biostatistics Research Group
Department of Health Sciences
University of Leicester
michael.crowther@le.ac.uk
@Crowther_MJ

Stata Nordic and Baltic Conference
30th August 2019



Background

`survsim` (Crowther and Lambert, 2012, 2013)

Simulation of simple and complex survival data

Background

`survsim` (Crowther and Lambert, 2012, 2013)

Simulation of simple and complex survival data

`multistate` (Crowther and Lambert, 2017)

Parametric multi-state survival analysis

Background

`survsim` (Crowther and Lambert, 2012, 2013)

Simulation of simple and complex survival data

`multistate` (Crowther and Lambert, 2017)

Parametric multi-state survival analysis

`merlin` (Crowther, 2017, 2019a,b)

Extended mixed effects models for linear, non-linear and user-defined outcomes

Standard parametric distributions

```
. set obs 1000
. gen trt = runiform() $>$ 0.5
. survsim stime died, dist(weib) lambda(0.1) gamma(1.2)
>                               maxtime(5) cov(trt -0.5)
```

Standard parametric distributions

```
. set obs 1000
. gen trt = runiform()>0.5
. survsim stime died, dist(weib) lambda(0.1) gamma(1.2)
>                               maxtime(5) cov(trt -0.5)
```

Custom (log) hazard functions

```
. set obs 1000
. gen trt = runiform()>0.5
. survsim stime died, loghazard(-3 :+ #t :+ 0.1 :* log(#t) :* #t)
>                               maxtime(5) cov(trt -0.5)
```

Transition matrix

```
. mat list tmat
tmat[3,3]
      to:   to:   to:
      start rfi   osi
from:start .    1    2
from:rfi   .    .    3
from:osi   .    .    .
```

Data

pid	_from	_to	_start	_stop	_status	_trans
1	1	2	0	59.104721	0	1
1	1	3	0	59.104721	0	2
1371	1	2	0	16.558521	1	1
1371	1	3	0	16.558521	0	2
1371	2	3	16.558521	24.344969	1	3

Stacked model

```
. streg hormon _trans2 _trans3, dist(weib)  
. predictms , transmat(tmat) at1(hormon 1)
```


Stacked model

```
. streg hormon _trans2 _trans3, dist(weib)
. predictms , transmat(tmat) at1(hormon 1)
```

Separate models

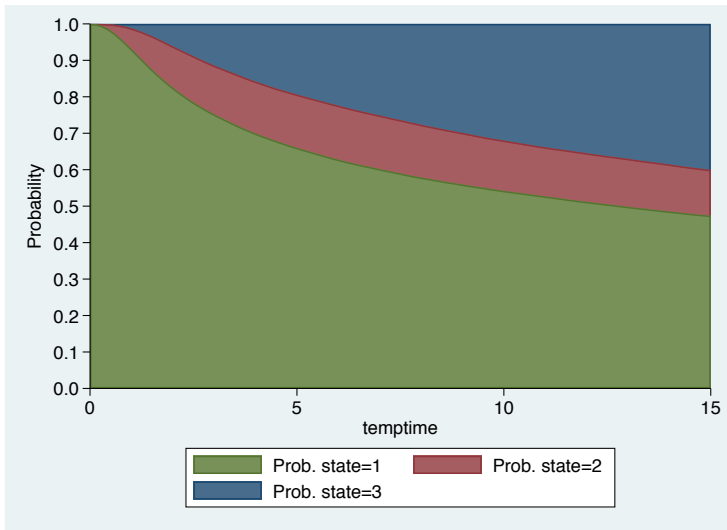
```
. streg hormon if _trans1==1, dist(weibull)
. estimates store m1

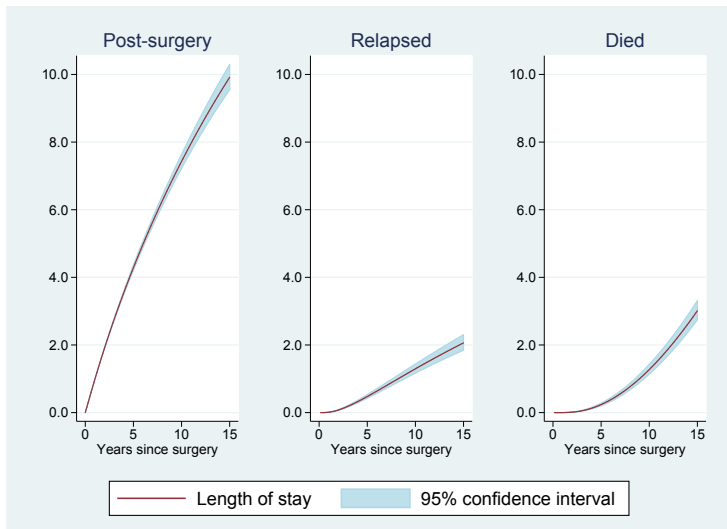
. stpm2 hormon if _trans2==1, scale(h) df(5)
. estimates store m2

. strcs hormon if _trans3==1, df(3)
. estimates store m3

. predictms , transmat(tmat) at1(hormon 1) models(m1 m2 m3)
```

multistate





A general framework for the analysis of data of all types

- Multiple outcomes of varying types
- Measurement schedule can vary across outcomes
- Any number of levels and random effects
- Sharing and linking random effects between outcomes
- Sharing functions of the expected value of other outcomes
- A reliable estimation engine
- Easily extendable by the user
- ...

Data

```
. list id time logb pro trt stime died if id==1 | id==2, noobs sepby(id)
```

id	time	logb	prothr-n	trt	stime	died
1	0	2.674149	12.2	D-penicil	1.09517	1
1	.525682	3.058707	11.2	D-penicil	.	.
2	0	.0953102	10.6	D-penicil	14.1523	0
2	.498302	-.2231435	11	D-penicil	.	.
2	.999343	0	11.6	D-penicil	.	.
2	2.10273	.6418539	10.6	D-penicil	.	.
2	4.90089	.9555114	11.3	D-penicil	.	.
2	5.88928	1.280934	11.5	D-penicil	.	.
2	6.88588	1.435084	11.5	D-penicil	.	.
2	7.8907	1.280934	11.5	D-penicil	.	.
2	8.83255	1.526056	11.5	D-penicil	.	.

```
merlin (logb          /// log serum bilirubin
        time          /// covariate
        ,             /// options
        family(gaussian) /// distribution
    )
```

```
merlin (logb          /// log serum bilirubin
        time          /// covariate
        time#trt      /// interaction
        ,             /// options
        family(gaussian) /// distribution
    )                ///
```

```
merlin (logb          /// log serum bilirubin
        time          /// covariate
        time#trt      /// interaction
        M1[id]@1      /// random intercept
        ,             /// options
        family(gaussian) /// distribution
    )                ///
```



```
merlin (logb          /// log serum bilirubin
        time          /// covariate
        time#trt      /// interaction
        M1[id]@1      /// random intercept
        time#M2[id]@1 /// random slope
        ,              /// options
        family(gaussian) /// distribution
    )
```

```

merlin (logb          /// log serum bilirubin
        time          /// covariate
        time#trt      /// interaction
        M1[id]@1      /// random intercept
        time#M2[id]@1 /// random slope
        ,             /// options
        family(gaussian) /// distribution
    )                ///
    (pro             /// prothrombin index
        rcs(time, df(3)) /// covariate
        , family(gamma)  /// distribution
    )                ///

```

```

merlin (logb          /// log serum bilirubin
        time          /// covariate
        time#trt      /// interaction
        M1[id]@1      /// random intercept
        time#M2[id]@1 /// random slope
        ,             /// options
        family(gaussian) /// distribution
    )                ///
    (pro             /// prothrombin index
        rcs(time, df(3)) /// covariate
        M3[id]@1      /// random effect
        , family(gamma) /// distribution
    )                ///

```

```

merlin (logb          /// log serum bilirubin
        time          /// covariate
        time#trt      /// interaction
        M1[id]@1      /// random intercept
        time#M2[id]@1 /// random slope
        ,             /// options
        family(gaussian) /// distribution
    )                ///
    (pro             /// prothrombin index
        rcs(time, df(3)) /// covariate
        M3[id]@1      /// random effect
        , family(gamma) /// distribution
    )                ///
    ,               /// main options
    covariance(unstructured) // vcv

```

```

merlin (logb          /// log serum bilirubin
        time          /// covariate
        time#trt      /// interaction
        M1[id]@1      /// random intercept
        time#M2[id]@1 /// random slope
        ,             /// options
        family(gaussian) /// distribution
    )                ///
    (pro             /// prothrombin index
        rcs(time, df(3)) /// covariate
        M3[id]@1      /// random effect
        , family(gamma) /// distribution
    )                ///
    ,               /// main options
    covariance(unstructured) /// vcv
    redistribution(t) df(5) /// re dist.

```

```

merlin (logb          /// log serum bilirubin
        time          /// covariate
        time#trt      /// interaction
        M1[id]@1      /// random intercept
        time#M2[id]@1 /// random slope
        ,             /// options
        family(gaussian) /// distribution
    )                ///
    (pro             /// prothrombin index
        rcs(time, df(3)) /// covariate
        M3[id]@1      /// random effect
        , family(gamma) /// distribution
    )                ///
    (stime trt       /// response + covariate
        , family(rp, df(3)) /// distribution
        failure(other)) /// event indicator
    )                ///
    ,               /// main options
    covariance(unstructured) /// vcv
    redistribution(t) df(5)   // re dist.

```

```

merlin (logb          /// log serum bilirubin
      time           /// covariate
      time#trt       /// interaction
      M1[id]@1       /// random intercept
      time#M2[id]@1  /// random slope
      ,              /// options
      family(gaussian) /// distribution
    )                ///
  (pro               /// prothrombin index
    rcs(time, df(3)) /// covariate
    M3[id]@1         /// random effect
    , family(gamma)  /// distribution
  )                ///
  (stime trt         /// response + covariate
    dEV[logb] EV[pro] /// associations
    , family(rp, df(3)) /// distribution
    failure(other)) /// event indicator
  )                ///
  ,                 /// main options
  covariance(unstructured) /// vcv
  redistribution(t) df(5)  /// re dist.

```

```

merlin (logb          /// log serum bilirubin
      time           /// covariate
      time#trt      /// interaction
      M1[id]@1      /// random intercept
      time#M2[id]@1 /// random slope
      ,             /// options
      family(gaussian) /// distribution
    )              ///
  (pro             /// prothrombin index
    rcs(time, df(3)) /// covariate
    M3[id]@1      /// random effect
    , family(gamma) /// distribution
  )              ///
  (stime trt      /// response + covariate
    trt#fp(stime, power(0)) /// tde
    dEV[logb] EV[pro] /// associations
    , family(rp, df(3)) /// distribution
      failure(other)) /// event indicator
  )              ///
  ,             /// main options
  covariance(unstructured) /// vcv
  redistribution(t) df(5)  /// re dist.

```



```

merlin (logb time time#trt M1[id]@1          /// model 1
        time#M2[id]@1 ,                      ///
        family(gaussian)                    ///
    )                                         ///
(pro   rcs(time, df(3)) M3[id]@1           /// model 2
    , family(gamma)                         ///
)                                             ///
(stime trt                                  ///
    trt#fp(stime, power(0))                /// model 3 - cause 1
    dEV[logb] EV[pro]                      /// tde
    , family(rp, df(3))                    /// distribution
        failure(other))                    /// event indicator
)                                             ///
(stime trt                                  /// model 4 - cause 2
    trt#rcs(stime, df(3) log)              /// tde
    EV[logb] iEV[pro]                      /// associations
    , family(weibull,                      /// distribution
        failure(cancer))                  /// event indicator
)                                             ///
,                                           ///
covariance(unstructured)                   ///

```

Model

```
. merlin (stime hormon , family(rp, df(3) failure(died)))  
. estimates store m1
```

Model

```
. merlin (stime hormon , family(rp, df(3) failure(died)))  
. estimates store m1
```

Simulate

```
. survsim stime died, model(m1) maxtime(5)
```

Model

```
. merlin (stime hormon , family(rp, df(3) failure(died)))  
. estimates store m1
```

Simulate

```
. survsim stime died, model(m1) maxtime(5)
```

Predict

```
. predictms, model(m1) at1(hormon 1)
```

merlin + survsim

Model

```
. merlin (stime hormon , family(weibull, failure(died)))
Mixed effects regression model          Number of obs      =      1,000
Log likelihood = -1370.8379
```

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	

stime:						
hormon	-.544245	.0967967	-5.62	0.000	-.7339631	-.3545269
_cons	-2.083643	.0958374	-21.74	0.000	-2.271481	-1.895806
log(gamma)	.110706	.0446086	2.48	0.013	.0232747	.1981372

. estimates store m1						

Simulate

```
. survsim stime2 died2, model(m1) maxtime(5)
```

merlin + survsim

Model

```
. merlin (sttime hormon , family(weibull, failure(died)))
Mixed effects regression model          Number of obs      =      1,000
Log likelihood = -1370.8379
```

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
sttime:						
hormon	-.544245	.0967967	-5.62	0.000	-.7339631	-.3545269
_cons	-2.083643	.0958374	-21.74	0.000	-2.271481	-1.895806
log(gamma)	.110706	.0446086	2.48	0.013	.0232747	.1981372

```
. estimates store m1
```

Simulate

```
. survsim stime2 died2, model(m1) maxtime(5)
```

Modify

```
. mat b2 = -0.5,-2,0.1
. erepost b = b2
. estimates store m1
. survsim stime3 died3, model(m1) maxtime(5)
```

Age as the timescale

```
. merlin (eventage hormon , family(rp, df(3) failure(died)  
      ltruncated(diagage)))  
. estimates store m1
```

Age as the timescale

```
. merlin (eventage hormon , family(rp, df(3) failure(died)
                                             ltruncated(diagage)))
. estimates store m1
```

Age and time since diagnosis as timescales

```
. merlin (eventage hormon rcs(eventage, df(2) log moffset(diagage))
          , family(rp, df(3) failure(died) ltruncated(diagage)))
. estimates store m1
```


Age as the timescale

```
. merlin (eventage hormon , family(rp, df(3) failure(died)
                                             ltruncated(diagage)))
. estimates store m1
```

Age and time since diagnosis as timescales

```
. merlin (eventage hormon rcs(eventage, df(2) log moffset(diagage))
          , family(rp, df(3) failure(died) ltruncated(diagage)))
. estimates store m1
```

Predict

```
. predictms, model(m1) at1(hormon 1)
```

merlin - interval censoring

Data

id	left	time	event	status	trt
1	.	72	0	Censor	0
2	0	1	2	Cancer	0
3	.	40	1	CVD	1
4	19	20	2	CVD	0
5	.	65	0	Censor	0

Model

```
. merlin (time trt , family(rp, df(3) failure(event) linterval(left)))  
. estimates store m1
```

merlin - interval censoring & competing risks

Data

id	left	time	event	status	cause1	cause2	cause3
1	.	72	0	Censor	1	1	1
2	0	1	2	Cancer	1	0	0
3	.	40	1	CVD	0	1	0
4	19	20	2	Other	0	0	1
5	.	65	0	Censor	1	1	1

merlin - interval censoring & competing risks

Data

id	left	time	event	status	cause1	cause2	cause3
1	.	72	0	Censor	1	1	1
2	0	1	2	Cancer	1	0	0
3	.	40	1	CVD	0	1	0
4	19	20	2	Other	0	0	1
5	.	65	0	Censor	1	1	1

Model

```
. merlin (time trt if cause1==1, family(weibull, fail(event) linterval(left)))  
> (time trt if cause2==1, family(rp, df(3) fail(event) linterval(left)))  
> (time trt if cause3==1, family(gompertz, fail(event) linterval(left)))  
> , transmatrix(tmat)
```

merlin - interval censoring & competing risks

Data

id	left	time	event	status	cause1	cause2	cause3
1	.	72	0	Censor	1	1	1
2	0	1	2	Cancer	1	0	0
3	.	40	1	CVD	0	1	0
4	19	20	2	Other	0	0	1
5	.	65	0	Censor	1	1	1

Model

```
. merlin (time trt if cause1==1, family(weibull, fail(event) linterval(left)))  
> (time trt if cause2==1, family(rp, df(3) fail(event) linterval(left)))  
> (time trt if cause3==1, family(gompertz, fail(event) linterval(left)))  
> , transmatrix(tmat)
```

Simulate

```
. estimates store m1  
. survsim stime died, model(m1) outcome(2) maxtime(5)
```

We can predict the cause-specific cumulative incidence functions

```
range tvar 0 60 500
predict cif1, cif outcome(1) timevar(tvar) at(trt 1)
predict cif2, cif outcome(2) timevar(tvar) at(trt 1)
predict cif3, cif outcome(3) timevar(tvar) at(trt 1)
```

And quantify the impact of treatment

```
. predict cifdiff1, cifdifference outcome(1) timevar(tvar) ///
> at1(trt 1) at2(trt 0) ci
. predict cifdiff2, cifdifference outcome(2) timevar(tvar) ///
> at1(trt 1) at2(trt 0) ci
. predict cifdiff3, cifdifference outcome(3) timevar(tvar) ///
> at1(trt 1) at2(trt 0) ci
```

merlin - interval censoring & competing risks

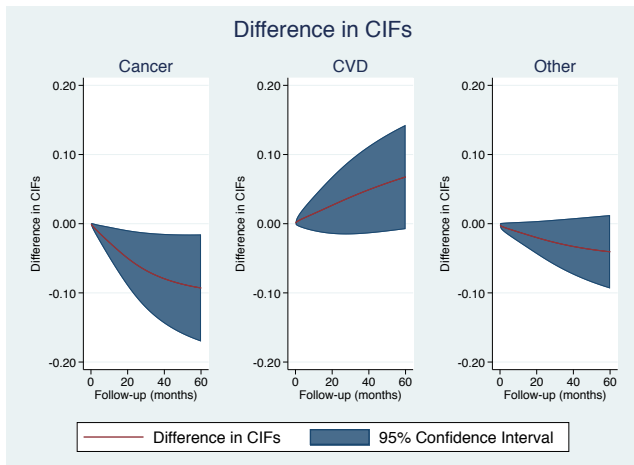


Figure 1: Difference in cumulative incidence functions, and associated 95% confidence interval, across treatment groups, for each cause of death.

We can calculate the loss in (restricted) life expectancy for each cause

```
predict lile1, timelost outcome(1) timevar(tvar) at(trt 1) ci  
predict lile2, timelost outcome(2) timevar(tvar) at(trt 1) ci  
predict lile3, timelost outcome(3) timevar(tvar) at(trt 1) ci
```

Or directly calculate the total

```
predict lile0, totaltimelost outcome(1) timevar(tvar) at(trt 0) ci  
predict lile1, totaltimelost outcome(1) timevar(tvar) at(trt 1) ci
```


merlin - interval censoring & competing risks

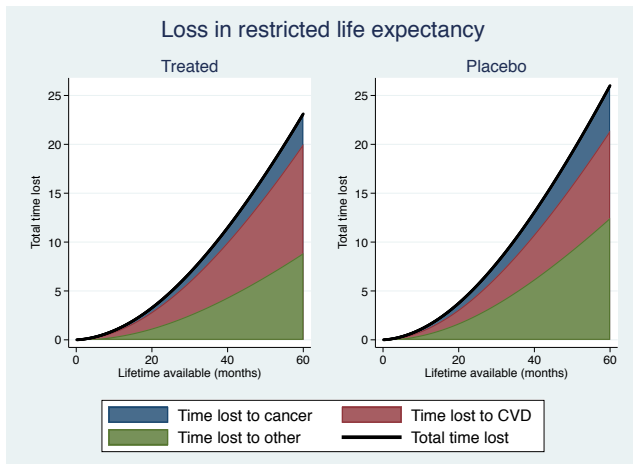


Figure 2: Loss in restricted life expectancy due to each cause of death.

Discussion and future work

- `survsim` now talks to `merlin`
- `predictms` now talks to `merlin`
- By syncing up the codebases, extensions to `merlin` will filter down
- I also showed some extensions for interval censored data
- We can still derive clinically useful predictions, regardless of the complexity of the underlying models
- More examples on mjcrowther.co.uk/software/merlin

Discussion and future work

- `survsim` now talks to `merlin`
- `predictms` now talks to `merlin`
- By syncing up the codebases, extensions to `merlin` will filter down
- I also showed some extensions for interval censored data
- We can still derive clinically useful predictions, regardless of the complexity of the underlying models
- More examples on mjcrowther.co.uk/software/merlin

...most of this talk isn't released yet

- Crowther, M. J. 2017. Extended multivariate generalised linear and non-linear mixed effects models. *arXiv* . URL <https://arxiv.org/abs/1710.02223>.
- . 2019a. merlin - a unified modelling framework for data analysis and methods development in Stata. *Stata Journal* . URL <https://arxiv.org/abs/1806.01615>.
- . 2019b. Multilevel mixed effects parametric survival analysis: Estimation, prediction and software . URL <https://arxiv.org/abs/1709.06633>.
- Crowther, M. J., and P. C. Lambert. 2012. Simulating complex survival data. *Stata J* 12(4): 674–687.
- . 2013. Simulating biologically plausible complex survival data. *Stat Med* 32(23): 4118–4134. URL <http://dx.doi.org/10.1002/sim.5823>.
- . 2017. Parametric multistate survival models: Flexible modelling allowing transition-specific distributions with application to estimating clinically useful measures of effect differences. *Statistics in medicine* 36: 4719–4742.