

August 2005 Stata Application Tutorial 5: Issues in Model Selection

Data Note: Code makes use of the file `adopt_singleevent.dta`. This data set is available on the Event History website. Code is based on Stata version 8.

Let's estimate some simple one-covariate models. Using the single-spell version of the restrictive abortion adoption data we'll start with a Cox model.

COX

```
. stcox preroe, nohr exactp basehc(baseline_hazard)

      failure _d:  failed
analysis time _t:  event
              id:  stcode

Iteration 0:  log likelihood = -100.00895
Iteration 1:  log likelihood = -96.609943
Iteration 2:  log likelihood = -96.599246
Iteration 3:  log likelihood = -96.599246
Refining estimates:
Iteration 0:  log likelihood = -96.599246

Cox regression -- exact partial likelihood

No. of subjects =          50                Number of obs =          50
No. of failures =          43
Time at risk   =          386
Log likelihood  = -96.599246                LR chi2(1)    =          6.82
                                                Prob > chi2   =          0.0090

-----+-----
      _t |          Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-----+-----
mooneymean |   -.220654   .0878145    -2.51   0.012   - .3927673   - .0485408
-----+-----
```

The command “basehc” computes the Cox baseline hazard function. Note that it is a very close cousin of the nonparametric Nelson-Aalen estimator we previously discussed. As such, it is non-smooth. We'll see this in a moment. Now, let's try a Weibull.

WEIBULL

```
. streg mooneymean, nohr dist(weib) robust
```

```
      failure _d: failed  
analysis time _t: event  
              id: stcode
```

Fitting constant-only model:

```
Iteration 0:  log pseudolikelihood = -81.651323  
Iteration 1:  log pseudolikelihood = -81.394283  
Iteration 2:  log pseudolikelihood = -81.394041  
Iteration 3:  log pseudolikelihood = -81.394041
```

Fitting full model:

```
Iteration 0:  log pseudolikelihood = -81.394041  
Iteration 1:  log pseudolikelihood = -77.542228  
Iteration 2:  log pseudolikelihood = -77.356367  
Iteration 3:  log pseudolikelihood = -77.355825  
Iteration 4:  log pseudolikelihood = -77.355825
```

Weibull regression -- log relative-hazard form

```
No. of subjects      =          50          Number of obs      =          50  
No. of failures      =          43  
Time at risk        =          386  
Log pseudolikelihood = -77.355825          Wald chi2(1)         =          6.23  
                                                              Prob > chi2          =          0.0126
```

(standard errors adjusted for clustering on stcode)

	Coef.	Robust Std. Err.	z	P> z	[95% Conf. Interval]	
mooneymean	-.2227881	.0892734	-2.50	0.013	-.3977607	-.0478154
_cons	-2.096628	.2488421	-8.43	0.000	-2.58435	-1.608906
/ln_p	-.0171348	.1059015	-0.16	0.871	-.2246979	.1904283
p	.9830112	.1041024			.7987575	1.209768
1/p	1.017282	.1077317			.826605	1.251944

I want the baseline hazard estimate. I'm going to compute this "by hand" and not use Stata options (why I do this will be explained in class). To do this, I reestimate the Weibull as an AFT model and then generate the function:

```
. streg mooneymean, nohr time dist(weib) robust [output suppressed]  
  
. gen lambda_base=exp(-(_b[_cons])) if e(sample)  
. gen haz_weib=lambda_base*e(aux_p)*(lambda_base*_t)^(e(aux_p)-1)
```

This gets the job done.

Now let's estimate a Royston-Parmer model:

```
. stpm mooneymean, scale(h) df(3)
```

```
initial:      log likelihood = -82.472475
rescale:      log likelihood = -82.472475
rescale eq:   log likelihood = -82.472475
Iteration 0:  log likelihood = -82.472475
Iteration 1:  log likelihood = -75.051963
Iteration 2:  log likelihood = -74.16672
Iteration 3:  log likelihood = -74.155585
Iteration 4:  log likelihood = -74.155581
```

```
Log likelihood = -74.155581
```

Number of obs	=	50
Wald chi2(1)	=	6.62
Prob > chi2	=	0.0101

```
-----+-----
```

	_t	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
s0	_cons	1.855173	.4420958	4.20	0.000	.9886813 2.721665
s1	_cons	.6191146	.3254337	1.90	0.057	-.0187237 1.256953
s2	_cons	-.3176424	.2211123	-1.44	0.151	-.7510145 .1157297
xb	mooneymean	-.2126251	.0826363	-2.57	0.010	-.3745893 -.0506609
	_cons	-2.386486	.393081	-6.07	0.000	-3.15691 -1.616061

```
-----+-----
Deviance = 148.311 (50 observations.)
```

```
. predict haz_rp, haz zero
```

The above command generates the estimated hazard rate from the Royston-Parmer model. Note that the “stpm” command is not a standard-issue Stata executable. It must be downloaded.

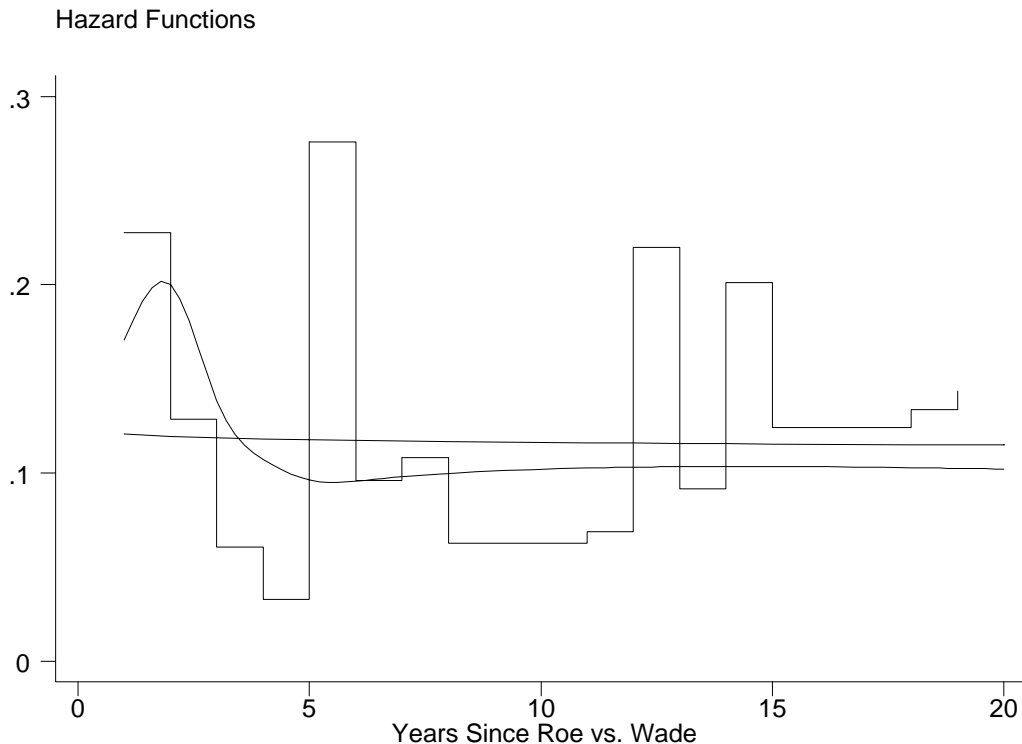
Looking at the models, we see all three yield similar conclusions about the effect of the “mooneymean” covariate. In general the hazard, or risk for adopting a restrictive abortion law decreases as the Pre-Roe permissiveness index increases. The issue here has to do with how time is conceptualized.

Under the Cox, the baseline hazard is highly adapted to the observed data. Under the Weibull (which coincidentally reduces to the exponential [why?]), the baseline hazard is smooth, but is tied to the particular F(t) chosen. Under the Royston-Parmer model, the function is smooth, but more closely adapted to the observed data (but not so close as the Cox model).

To see this, let's graph the functions:

```
. gr7 haz_rp haz_weib baseline_hazard _t, ylab xlab c(s1J)
s(iii) t1("Hazard Functions") b2(" Years Since Roe vs.
Wade")
```

Note that above, I'm using a Stata 7 command. (Although Stata graphics have improved, this code fragment more easily produces the graph of interest than does Stata 8 [or now 9] commands do.) This command returns:



What are the major features of this graph?

The “skyline” plot is the Cox model. The “flat line” is the Weibull. The smooth and non-monotonic function is the Royston-Parmer estimate (this is the same graph found in our book [p. 92, Fig. 6.1] and is based on the estimates computed above [which are found in Table 6.1 on p. 91 in the book]).