

## August 2005

### Stata Application Tutorial 4: Discrete Models

Data Note: Code makes use of career.dta, and icpsr\_discrete1.dta. All three data sets are available on the Event History website. Code is based on Stata version 8.

**Preliminaries:** The issue of duration “dependency” in the discrete-time model. This sequence replicates the analysis from Table 5.2 of the book. I’m going to run through a sequence of models testing different functional forms for time. After each model, I generate the predicted probability of  $y=1$ .

#### “Exponential”

```
. logit _d rep, cluster(memberid)
```

```
Iteration 0: log pseudolikelihood = -1353.706
Iteration 1: log pseudolikelihood = -1352.1479
Iteration 2: log pseudolikelihood = -1352.1451
```

```
Logit estimates                               Number of obs   =       5054
                                                Wald chi2(1)    =         2.70
                                                Prob > chi2     =       0.1004
Log pseudolikelihood = -1352.1451           Pseudo R2      =       0.0012
```

(standard errors adjusted for clustering on memberid)

_d	Coef.	Robust Std. Err.	z	P> z	[95% Conf. Interval]
rep	.1894179	.1153099	1.64	0.100	-.0365853 .4154212
_cons	-2.586274	.0775683	-33.34	0.000	-2.738305 -2.434243

```
. predict pexp, p, if e(sample)
```

#### “Linear”

```
. logit _d _t rep, cluster(memberid)
```

```
Iteration 0: log pseudolikelihood = -1353.706
Iteration 1: log pseudolikelihood = -1343.2309
Iteration 2: log pseudolikelihood = -1343.0146
Iteration 3: log pseudolikelihood = -1343.0144
```

```
Logit estimates                               Number of obs   =       5054
                                                Wald chi2(2)    =       16.67
                                                Prob > chi2     =       0.0002
Log pseudolikelihood = -1343.0144           Pseudo R2      =       0.0079
```

(standard errors adjusted for clustering on memberid)

_d	Coef.	Robust Std. Err.	z	P> z	[95% Conf. Interval]
_t	-.0756131	.0203355	-3.72	0.000	-.1154701 -.0357562
rep	.1585271	.1106863	1.43	0.152	-.0584141 .3754682
_cons	-2.257402	.1144119	-19.73	0.000	-2.481645 -2.033159

```
. predict plin, p, if e(sample)
```

### "Lowess"

```
. lowess _d duration, gen(lowesst) [Creating lowess for t]  
. logit _d lowesst rep, cluster(memberid)
```

```
Iteration 0: log pseudolikelihood = -1353.706  
Iteration 1: log pseudolikelihood = -1331.3664  
Iteration 2: log pseudolikelihood = -1330.2596  
Iteration 3: log pseudolikelihood = -1330.2578
```

```
Logit estimates                               Number of obs   =       5054  
                                              Wald chi2(2)   =        46.82  
                                              Prob > chi2    =       0.0000  
Log pseudolikelihood = -1330.2578           Pseudo R2      =       0.0173
```

(standard errors adjusted for clustering on memberid)

_d	Coef.	Robust Std. Err.	z	P> z	[95% Conf. Interval]	
lowesst	13.29807	2.016537	6.59	0.000	9.345729	17.25041
rep	.1619009	.1094325	1.48	0.139	-.0525829	.3763847
_cons	-3.701575	.1890745	-19.58	0.000	-4.072155	-3.330996

```
. predict plowess, p, if e(sample)
```

### "Spline"

```
. btscs _d durat memberid, g(t) nspline(3) dummy(k)  
[Creating a spline function]  
. logit _d t _spline1 _spline2 _spline3 rep,  
cluster(memberid)
```

```
Iteration 0: log pseudolikelihood = -1353.706  
Iteration 1: log pseudolikelihood = -1330.1917  
Iteration 2: log pseudolikelihood = -1328.3452  
Iteration 3: log pseudolikelihood = -1328.3394  
Iteration 4: log pseudolikelihood = -1328.3394
```

```
Logit estimates                               Number of obs   =       5054  
                                              Wald chi2(5)   =        52.00  
                                              Prob > chi2    =       0.0000  
Log pseudolikelihood = -1328.3394           Pseudo R2      =       0.0187
```

(standard errors adjusted for clustering on memberid)

_d	Coef.	Robust Std. Err.	z	P> z	[95% Conf. Interval]	
t	-.6974501	.177487	-3.93	0.000	-1.045318	-.3495819
_spline1	-.1222513	.06804	-1.80	0.072	-.2556072	.0111047
_spline2	.0415769	.0409644	1.01	0.310	-.0387119	.1218658
_spline3	-.0003271	.0123737	-0.03	0.979	-.0245791	.0239249
rep	.1648343	.1096781	1.50	0.133	-.0501308	.3797994
_cons	-1.976583	.1102593	-17.93	0.000	-2.192688	-1.760479

```
. predict pspline, p, if e(sample)
```

### "log(t)"

```
. gen logdur=log(duration)
```

```
. logit _d logdur rep, cluster(memberid)
```

```
Iteration 0:  log pseudolikelihood = -1353.706
Iteration 1:  log pseudolikelihood = -1335.884
Iteration 2:  log pseudolikelihood = -1335.541
Iteration 3:  log pseudolikelihood = -1335.5409
```

```
Logit estimates                                     Number of obs   =       5054
                                                    Wald chi2(2)    =       32.24
                                                    Prob > chi2     =       0.0000
Log pseudolikelihood = -1335.5409                 Pseudo R2       =       0.0134
```

(standard errors adjusted for clustering on memberid)

_d	Coef.	Robust Std. Err.	z	P> z	[95% Conf. Interval]	
logdur	-.38962	.0720928	-5.40	0.000	-.5309192	-.2483207
rep	.15602	.1092708	1.43	0.153	-.0581469	.3701868
_cons	-2.136548	.1085036	-19.69	0.000	-2.349211	-1.923884

### "Quadratic"

```
. gen dur2=durat^2
```

```
. logit _d durat dur2 rep, cluster(memberid)
```

```
Iteration 0:  log pseudolikelihood = -1353.706
Iteration 1:  log pseudolikelihood = -1338.0372
Iteration 2:  log pseudolikelihood = -1337.7571
Iteration 3:  log pseudolikelihood = -1337.757
```

```
Logit estimates                                     Number of obs   =       5054
                                                    Wald chi2(3)    =       29.67
                                                    Prob > chi2     =       0.0000
Log pseudolikelihood = -1337.757                 Pseudo R2       =       0.0118
```

(standard errors adjusted for clustering on memberid)

_d	Coef.	Robust Std. Err.	z	P> z	[95% Conf. Interval]	
duration	-.2235403	.0462736	-4.83	0.000	-.3142349	-.1328458
dur2	.0122237	.0032315	3.78	0.000	.0058902	.0185573
rep	.1667169	.1104643	1.51	0.131	-.0497892	.3832231
_cons	-1.984996	.137335	-14.45	0.000	-2.254168	-1.715825

```
. predict pquad, p, if e(sample)
```

Obviously, there are several choices of functional form. The model chi-square from each estimate is:

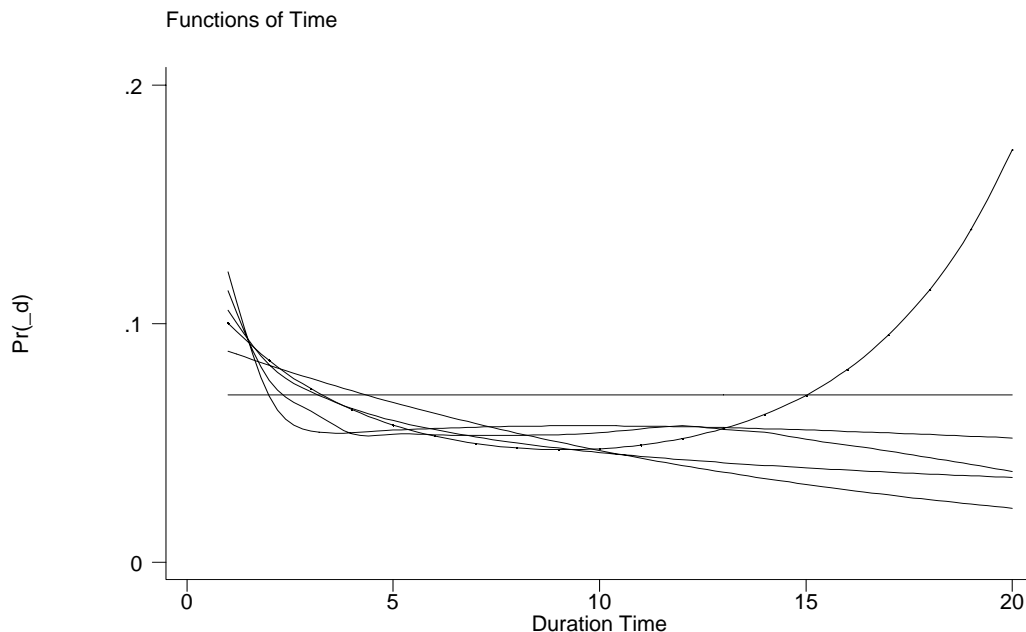
```
Exponential -1352.1451
Linear       -1343.0144
Lowess      -1330.2578
Spline      -1328.3394
Log(t)      -1335.5409
Quadratic   -1337.7570
```

We could obviously evaluate fit based on the likelihood ratio statistic. If we did this, we would conclude the spline model is slightly preferred over the lowess model.

To see what the different functions look like, I graph them (holding the “rep” covariate to 0; I use an old Stata 7 command because it had the capacity to graph all of the functions...thus producing a very ugly graph!!)

```
gr7 plowess pspline pexp plin plog pquad _t , c(ssssss), if
rep==0 , s(iiiii) ylab xlab saving(icpsr_discretet.gph, replace)
t1("Functions of Time") b2("Duration Time")
```

Which returns:



(Clear as mud, isn't it?)

## The Conditional Logit-Cox Link

There is a close connection between the Cox model and the conditional logit model. Specifically, the exact-discrete approximation is equivalent to a conditional logit estimator.

To see this, consider this hypothetical data set:

	duration	caseid	class	event
1.	6	1	1	1
2.	4	2	1	1
3.	10	3	1	1
4.	3	4	0	1
5.	2	5	0	0
6.	2	6	0	1
7.	4	7	0	1
8.	2	8	0	0
9.	2	9	0	1
10.	3	10	1	1
11.	4	11	1	0
12.	6	12	1	1
13.	2	13	0	0
14.	2	14	0	1

Let's estimate a Cox model using the exact-discrete approximation:  
(First I stset the data)

```
. stset duration, failure(event==1) id(caseid)
```

```
          id: caseid
      failure event: event == 1
obs. time interval: (duration[_n-1], duration]
exit on or before: failure
```

```
-----
14 total obs.
 0 exclusions
```

```
-----
14 obs. remaining, representing
14 subjects
10 failures in single failure-per-subject data
52 total analysis time at risk, at risk from t =      0
                                     earliest observed entry t =      0
                                     last observed exit t =      10
```

```

. stcox class, exactp nohr nolog

      failure _d:  event == 1
analysis time _t:  duration
              id:  caseid

Cox regression -- exact partial likelihood

No. of subjects =          14                Number of obs   =          14
No. of failures =          10
Time at risk    =          52
Log likelihood  = -10.348733                LR chi2(1)         =          5.37
                                                Prob > chi2        =          0.0204

-----+-----
      _t |          Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-----+-----
      class |    -2.39456    1.201183    -1.99   0.046    -4.748835    -.040285
-----+-----

```

The results are standard Cox model results. The estimated hazard ratio is .0912, meaning the risk of failure is much lower when class=1 [the inverse of this ratio is 10.91; this implies that when class=0, the risk of failure is about 11 times greater than when class=1...of course it's all made-up data!].

These data do not “look” discrete...but in fact just about any type of duration data can be thought of as being discrete. Suppose we “discretize” these data through Stata’s stsplrit option:

```

. stsplrit, at(failures) riskset(RS)
(5 failure times)
(18 observations (episodes) created)

```

What does this do? By splitting the data at the failure times, we are creating separate records of data for each risk period.

```
. list RS event duration caseid
```

	RS	event	duration	caseid
1.	1	0	2	5
2.	1	0	2	8
3.	1	0	2	13
4.	1	1	2	6
5.	1	1	2	9
6.	1	1	2	14
7.	1	.	2	1
8.	1	.	2	2
9.	1	.	2	3
10.	1	.	2	4
11.	1	.	2	7
12.	1	.	2	10
13.	1	.	2	11
14.	1	.	2	12
15.	2	1	3	4
16.	2	1	3	10
17.	2	.	3	1
18.	2	.	3	2
19.	2	.	3	3
20.	2	.	3	7
21.	2	.	3	11
22.	2	.	3	12
23.	3	0	4	11
24.	3	1	4	2
25.	3	1	4	7
26.	3	.	4	1
27.	3	.	4	3
28.	3	.	4	12
29.	4	1	6	1
30.	4	1	6	12
31.	4	.	6	3
32.	5	1	10	3

What are the main features of these data? They look a lot like “case-control” data...and Cox data. Look at lines 1—14. These cases denote the “first risk period.” All 14 observations enter the risk pool at time 0. At time 2, the first failure occurred (to cases 6, 9, and 14); however, three cases are right-censored here (5, 8, and 13). Hence, the second risk period contains only 8 observations (why?). This continues until the last observed failure time.

Because conditional logit is an appropriate estimator for case-control data and because we now have duration data constructed as case-control data, the conditional logit estimator is a suitable estimator for duration data:

```
. clogit _d class, group(RS) nolog
```

```
note: multiple positive outcomes within groups encountered.
note: 1 group (1 obs) dropped due to all positive or
      all negative outcomes.
```

```
Conditional (fixed-effects) logistic regression   Number of obs   =           31
                                                    LR chi2(1)      =           5.37
                                                    Prob > chi2     =           0.0204
Log likelihood = -10.348733                       Pseudo R2       =           0.2061
```

_d	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
class	-2.394559	1.201183	-1.99	0.046	-4.748834	-.0402848

Here, I use a standard conditional logit estimator. The results are *equivalent* to my Cox model results from before. This should not be surprising. Once the data are constructed as matched case-control data, the two models are one in the same.

...this means the usual issues conditional logit apply. Note that it’s a fixed effects estimator. The “effect” that is “fixed” is time. Again, this isn’t a surprise: Cox only works with the ordered failure times to derive the risk pool. This is exactly what conditional logit does.

Since this estimator is a fixed effects estimator (and the effect for the risk pool is fixed), any covariate that is perfectly time dependent *cannot be estimated*. To illustrate, suppose I have a variable called “tfactor” that changes values at each observed failure time. Let’s try the conditional logit:

```
. clogit _d class tfactor, group(RS) nolog
```

```
note: multiple positive outcomes within groups encountered.  
note: 1 group (1 obs) dropped due to all positive or  
      all negative outcomes.  
note: tfactor omitted due to no within-group variance.
```

```
Conditional (fixed-effects) logistic regression   Number of obs   =       31  
                                                  LR chi2(1)      =       5.37  
                                                  Prob > chi2     =       0.0204  
Log likelihood = -10.348733                    Pseudo R2       =       0.2061
```

```
-----  
      _d |          Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]  
-----+-----  
      class | -2.394559   1.201183    -1.99   0.046   -4.748834   -.0402848  
-----
```

Oops. This can be an issue with the conditional logit (or any fixed effects model).

In general, though, the connection between the Cox model and standard discrete models (like the conditional logit or the complementary log-log) is reasonably close. This, in turn, raises issues (in my view) about the utility of starkly distinguishing “discrete” from “continuous” models.