UNIVERSITY OF ESSEX
INSTITUTE FOR SOCIAL AND ECONOMIC RESEARCH
Professor Stephen P. Jenkins <stephenj@essex.ac.uk>

Essex Summer School course 'Survival Analysis'
and
EC968. Part II: Introduction to the analysis of spell duration data

Lesson 6. Estimation: (ii) discrete time models (logistic and cloglog)

Contents

1 Aim

The aim of this lesson is to illustrate how to use Stata to estimate multivariate discrete time (grouped data) survival time models of the type discussed in Lesson 2.

2 Introduction:

Stata does not have a set of specialist commands for estimating the discrete time proportional odds or proportional hazards models. But they are very easy to estimate nonetheless. All one has to do is re-organise the data set, define some new variables (to specify the baseline hazard function in particular), and then apply logit or cloglog regression. (See the Lecture

Notes for details.) Derivation of predicted survival times (median durations etc) is a little more fiddly because there are no closed-form formulae for these except in special cases, but the **predict** command makes things relatively straightforward.

The illustrations concerning discrete time models use the Cancer data set in the same way as Lesson 5 about continuous time models did. This is done deliberately in order to highlight the similarities and differences between the modelling (and the estimates).

This lesson first discusses how to re-organise the data set and define the new variables, necessary for estimation of both proportional odds and proportional hazard models. Then I discuss estimation. The discrete time models are estimated by maximum likelihood using **logit** and **cloglog** (or **logistic** and **glm**: see below). We will focus here on the discrete logistic (proportional odds) model. Estimation of the discrete complementary log-log (proportional hazard) model is very similar: see Exercise 6.1. I also show how to use **predict** to derive predicted hazard functions and survivor functions.

3 Data reorganisation and creation of new variables

Revise the material discussing this in Lesson 3. Recall that our 'easy estimation' methods for the discrete models are based on application of standard binary dependent variable models to re-organised data.

The data set must be re-organised so that, for each person, there are as many data rows as there are time intervals at risk of the event occurring for each person. We need to go from the simple data set discussed earlier, with one row of data per person, to another data set in which each person contributes T_i rows, where T_i is the number of time periods (e.g. months) *i* was at risk of the event. In effect an unbalanced panel data set-up is required.

We also require a unique identifier variable for each subject (if it doesn't already exist), plus a spell month identifier variable for each subject. The binary dependent variable also needs to be created. If subject *i*'s survival time is censored, the binary dependent variable is equal to 0 for all of *i*'s spell months; if subject *i*'s survival time is not censored, the binary dependent variable is equal to 0 for all but the last of *i*'s spell months (month 1,..., T_i–1) and equal to 1 for the last month (month T_i). To emphasize that time now refers to discrete intervals, I use the notation *j* for elapsed duration rather than *t* as in Lesson 5.

Let us illustrate this using the Cancer data set (as in Lesson 3), with the drug variable recoded into two categories (as before):

```
. use cancer
(Patient Survival in Drug Trial)

. ge id = _n

. lab var id "subject identifier"
. recode drug 1=0 2/3=1
(48 changes made)
. lab var drug "receives drug?"
. lab def drug 0 "placebo" 1 "drug"
. lab val drug drug
```

Now we do the episode splitting, producing a data set in person-month format, exactly we did in Lesson 3 (see the discussion there explaining the Stata code used).

```
. expand studytim
(696 observations created)

. bysort id: ge j = _n
. * spell month identifier, by subject
. lab var j "spell month"

. bysort id: ge dead = died==1 & _n==_N
. lab var dead "binary depvar for discrete hazard model"
```

Remember that we do *not* have to **stset** the data for estimation, because we do not use the **st** commands – they are for the continuous time case.

4 Choose the functional form for the baseline hazard function

The final step prior to estimation is to choose a functional form for the baseline hazard function. We do this by defining new time-varying covariates which are functions of survival time *t* per person (variable t in the illustration). Lesson 3 briefly discussed this. Here we consider several alternative specifications (from the many!): log(time), polynomial in time, piece-wise constant, and fully non-parametric.

For the log(time) and cubic polynomial specifications, the new variables are:

```
. ge lnj = ln(j)

. ge j2 = j^2

. ge j3 = j^3
```

For a non-parametric baseline, we need to create duration-interval-specific dummy variables, one for each spell month at risk. The maximum survival time in the Cancer data set is 39, so we need 39 dummy variables. There are 2 methods (at least) of creating them. First, you: can use **tabulate, generate(.)** where the argument of the generate option is the common prefix (‘stub’) of the 39 new variables, ‘d’ in this case:

```
. ta j, ge(d)

spell month |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         48        6.45        6.45
          2 |         46        6.18       12.63
          3 |         45        6.05       18.68
          4 |         44        5.91       24.60
          5 |         42        5.65       30.24
          6 |         40        5.38       35.62
          7 |         37        4.97       40.59
          8 |         36        4.84       45.43
          9 |         32        4.30       49.73
         10 |         31        4.17       53.90
         11 |         29        3.90       57.80
         12 |         26        3.49       61.29
         13 |         24        3.23       64.52
         14 |         23        3.09       67.61
         15 |         23        3.09       70.70
         16 |         21        2.82       73.52
         17 |         20        2.69       76.21
         18 |         18        2.42       78.63
         19 |         18        2.42       81.05
         20 |         16        2.15       83.20
         21 |         15        2.02       85.22
         22 |         15        2.02       87.23
         23 |         13        1.75       88.98
         24 |         11        1.48       90.46
         25 |         10        1.34       91.80
         26 |          8        1.08       92.88
         27 |          8        1.08       93.95
         28 |          8        1.08       95.03
         29 |          6        0.81       95.83
         30 |          6        0.81       96.64
         31 |          6        0.81       97.45
         32 |          6        0.81       98.25
         33 |          4        0.54       98.79
         34 |          3        0.40       99.19
         35 |          2        0.27       99.46
         36 |          1        0.13       99.60
         37 |          1        0.13       99.73
         38 |          1        0.13       99.87
         39 |          1        0.13      100.00
------------+-----------------------------------
      Total |        744      100.00
```

This creates 39 dummy variables called d1–d39. This is confirmed by describing the data in compact form using the **ds** command (cf. the more verbose **describe** command):

```
. ds d*
died  d1    d4    d7    d10   d13   d16   d19   d22   d25   d28   d31   d34   d37
drug  d2    d5    d8    d11   d14   d17   d20   d23   d26   d29   d32   d35   d38
dead  d3    d6    d9    d12   d15   d18   d21   d24   d27   d30   d33   d36   d39
```

Alternatively one can use the **forvalues** command (see **help forvalues**) to generate the variables in a loop. The local macro 'x' is set equal to 1 the first time the loop and generates durat1 = 1 if the spell month identifier is equal to one, sets durat1 = 0 otherwise. The 'x' then increments to 2 and generates durat2 = 1 if the spell month identifier is equal to 2, it is set equal to 0 otherwise. And then 'x' increments by 1 again, and generates another variable, and so on, with 'x' =39 being the last loop. The incrementation is set by the 'numlist' 1(1)39.

```
. forvalues x = 1(1)39 {
          ge byte durat`x' = (j == `x')
  }

. ds durat*
durat1   durat5   durat9   durat13  durat17  durat21  durat25  durat29  durat33  durat37
durat2   durat6   durat10  durat14  durat18  durat22  durat26  durat30  durat34  durat38
durat3   durat7   durat11  durat15  durat19  durat23  durat27  durat31  durat35  durat39
durat4   durat8   durat12  durat16  durat20  durat24  durat28  durat32  durat36
```

The 39 dummy variables called durat1–durat39 are created. In fact we will use d1–d39, so we can

```
. drop durat*
```

to save on memory usage.

An example of a different piece-wise constant specification was already shown in Lessons 3 and 5. Let us assume that the interval (discrete) hazard is constant in months 1–8, 9–17, and 18+. The dummy variables we require are:

```
. ge e1 = j < 9

. ge e2 = j >= 9 & j <= 17

. ge e3 = j >= 18 & j <.
```

An alternative allowing more pieces, six in fact, would be the following:

```
. ge dur1 = d1+d2+d3+d4+d5+d6

. ge dur2 = d7+d8+d9+d10+d11+d12

. ge dur3 = d13+d14+d15+d16+d17+d18

. ge dur4 = d19+d20+d21+d22+d23+d24

. ge dur5 = d25+d26+d27+d28+d29+d30

. ge dur6 = d31+d32+d33+d34+d35+d36+d37+d38+d39
```

The reason for the splitting of survival times at the particular points chosen will become apparent shortly (it ensures there are events occurring within each of the time intervals so defined). See also Ex. 6.1.

To conserve memory space after creating these variables, you will find it useful to **compress** the data in order to conserve disk space.

```
. compress
```

If you wish to estimate a model with fully non-parametric baseline hazard, then it is essential to check whether events occur at each value of *j* (i.e. the variable ‘j’ that we created). The hazard cannot be estimated for values of *j* with no events (exactly as with the non-parametric baseline hazard in the Cox model).

If there are duration intervals with no events, then either one must refine the grouping on the survival time dimension (this was the rationale for creating the variables dur* above), or else one must drop the relevant person months from the estimation. (Cf. the discussion of identification of the logit model in the Reference Manuals under ‘perfect predictors’.)

Checking whether there are events within each of the intervals is straightforward. Cross-tabulate the spell month identifier with the censoring variable.

```
. ta j dead

           |   binary depvar for
     spell | discrete hazard model
     month |         0          1 |     Total
-----------+----------------------+----------
         1 |        46          2 |        48 
         2 |        45          1 |        46 
         3 |        44          1 |        45 
         4 |        42          2 |        44 
         5 |        40          2 |        42 
         6 |        38          2 |        40 
         7 |        36          1 |        37 
         8 |        33          3 |        36 
         9 |        32          0 |        32 
        10 |        30          1 |        31 
        11 |        27          2 |        29 
        12 |        24          2 |        26 
        13 |        23          1 |        24 
        14 |        23          0 |        23 
        15 |        22          1 |        23 
        16 |        20          1 |        21 
        17 |        19          1 |        20 
        18 |        18          0 |        18 
        19 |        18          0 |        18 
        20 |        16          0 |        16 
        21 |        15          0 |        15 
        22 |        13          2 |        15 
        23 |        11          2 |        13 
        24 |        10          1 |        11 
        25 |         9          1 |        10 
        26 |         8          0 |         8 
        27 |         8          0 |         8 
        28 |         7          1 |         8 
        29 |         6          0 |         6 
        30 |         6          0 |         6 
        31 |         6          0 |         6 
        32 |         6          0 |         6 
        33 |         3          1 |         4 
        34 |         3          0 |         3 
        35 |         2          0 |         2 
        36 |         1          0 |         1 
        37 |         1          0 |         1 
        38 |         1          0 |         1 
        39 |         1          0 |         1 
-----------+----------------------+----------
     Total |       713         31 |       744 
```

There are no deaths during months 9, 14, 18–21, 26, 27, 29–32, 34–39, and so a month-specific hazard rate cannot be estimated for these intervals. We return to this issue later.

Let us now proceed to estimation.

5 Estimation

For ML estimation of the discrete time logistic model we use **logit**. The basic syntax is

```
logit depvar varlist, [or noconstant]
```

'depvar' is the (new) event variable – dead in the illustration – and 'varlist' refers to the explanatory variables (covariates) together with the variables used to summarise the baseline hazard function.

If **logit** is used without the **or** option, coefficients are reported; with the **or** option, odds ratios (exponentiated coefficients) are reported. For an alternative way of getting the latter directly, see **help logistic**. The **noconstant** option means estimate a model without a constant term –

we mainly use this for estimating models with a fully non-parametric baseline hazard. Note that odds ratios of hazard rates refer to ratios of form $[h_1/(1–h_1)] / [h_0/(1–h_0)]$ for the one unit change in an explanatory variable from zero to one. I personally find these difficult to interpret. On the other hand, as $h \to 0$, the odds ratio tends to the hazard ratio h_1/h_0, which does have a ready interpretation.

For ML estimation of the discrete time complementary log-log model we use **cloglog**. The basic syntax is

```
cloglog depvar varlist, [or noconstant]
```

depvar and **varlist** and the **noconstant** options are as for the logit model. To produce exponentiated coefficients, simply use the **eform** option. Recall that the exponentiated coefficients can be interpreted as hazard ratios since the cloglog model is the discrete time proportional hazards model.

The cloglog model (both with or without exponentiated coefficients) can also be fitted using the **glm** command: see Exercise 6.1.

See **help logit** and **help cloglog** for the full command syntax and all the options available. As with all Stata's estimation commands, estimation output can be re-played by simply re-issuing the command name again.

6 Estimation of the cloglog (discrete time proportional hazard) model and derivation of predicted hazard and survivor functions.

Let us begin with the case with a log(time) baseline hazard:

```
Complementary log-log regression                Number of obs     =        744
                                                Zero outcomes     =        713
                                                Nonzero outcomes  =         31

                                                LR chi2(3)        =      35.20
Log likelihood = -111.26371                     Prob > chi2       =     0.0000

------------------------------------------------------------------------------
        dead |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        drug |   -2.18907   .4110876    -5.33   0.000    -2.994787   -1.383353
         age |    .119348   .0371648     3.21   0.001     .0465064    .1921896
         lnj |   .6402733   .2454492     2.61   0.009     .1592017    1.121345
       _cons |  -9.928747   2.272995    -4.37   0.000    -14.38374   -5.473759
------------------------------------------------------------------------------
```

Observe that the estimated coefficients are similar to those for the Weibull model estimated in Lesson 5. In that model, the coefficient on drug was –2.20 (compared to –2.19 here), the coefficient on age was 0.12 (compared to 0.12 here), and the shape parameter p was 1.68. Recall that in Lesson 2 we parameterised the baseline hazard in the discrete time log(time) case as $c(j) = (q–1).\ln(j)$. The estimate of q here turns out to be almost the same, i.e. 1.64. Clearly, according to both models, drug recipients have lower hazard rates, the hazard rate increases with age, and the baseline hazard rises with elapsed survival time.

We can replay the estimates and get the exponentiated coefficients, which are the hazard ratios from the underlying continuous time model:

```
. cloglog, eform

Complementary log-log regression                Number of obs     =        744
                                                Zero outcomes     =        713
                                                Nonzero outcomes  =         31

                                                LR chi2(3)        =      35.20
Log likelihood = -111.26371                     Prob > chi2       =     0.0000

------------------------------------------------------------------------------
        dead |     exp(b)   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        drug |   .1120209   .0460504    -5.33   0.000     .0500473    .2507365
         age |   1.126762   .0418758     3.21   0.001     1.047605      1.2119
         lnj |   1.896999    .465617     2.61   0.009     1.172574    3.068979
------------------------------------------------------------------------------
```

The hazard ratios on drug and age turn out to very similar to those the continuous time Weibull model.

6.1 Within-sample prediction

Now consider derivation of the hazard and survivor functions for persons with particular covariate combinations. There are (at least) two ways of doing this. I will show first how to do this using within-sample predictions. This is facilitated by the fact that the data are already in person-month format, so we have covariate combinations and survival times in the data set. Later, towards the end of the lesson, show how to use out-of-sample predictions.

Both methods rely on the **predict** command. This has format **predict *newvarname* [, *statistic*]** where *statistic* can be 'p' for the probability (the default), 'xb' for the linear prediction, and so on.

Thus if we type

```
. predict h, p
```

then Stata generates the predicted logistic hazard rate for each person given the values of his or her covariates and the value of *j* in the relevant spell month. The variable containing the predicted hazard is called h. The formula used to calculate the predicted cloglog hazard for each person *i* and spell month *j* is

$$h_i(t) = 1 - \exp(-\exp[\, c(j) + \beta' X_i])$$

where β and $c(j)$ have been replaced by their estimated values. See Lesson 2 for a reminder of this formula (and the corresponding logit one).

The following code generates predicted hazard and survivor functions for two persons aged 55, one who received the placebo (drug = 0) and the other who received the drug (drug = 1). The code for generating the survivor function values (variable s) uses the same Stata 'tricks' as were used in the last lesson, based on the cumulative sum function **sum(.)**. The '**bysort id (j) :**' is new. This instructs Stata to sort observations by id and then, within id, to sort them by survival time j. Putting parentheses round the 'j' variable instructs Stata to do the **generate** calculate by groups defined according to id (but not also by j).

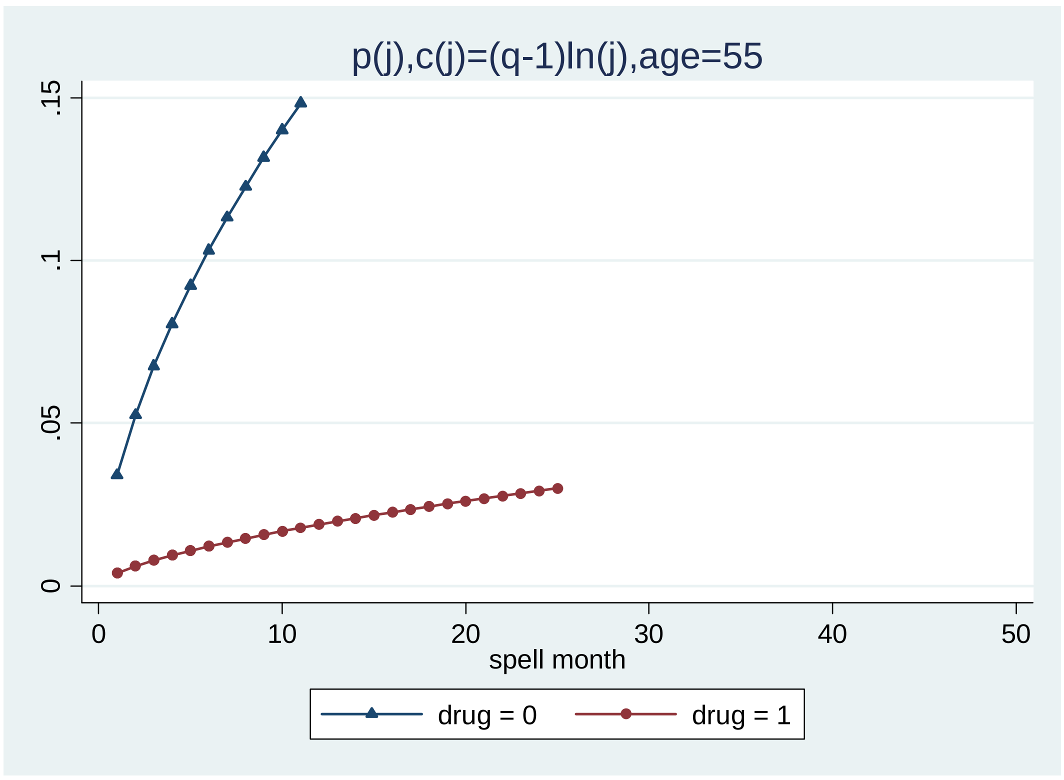
```
bysort id (j): ge s = exp(sum(ln(1-h)))
```

Now let us summarize the predictions graphically. (We could also have used **list** to inspect them.) To get separate plots for the two cases, we need to generate new and separate hazard and survivor function values for each of them.

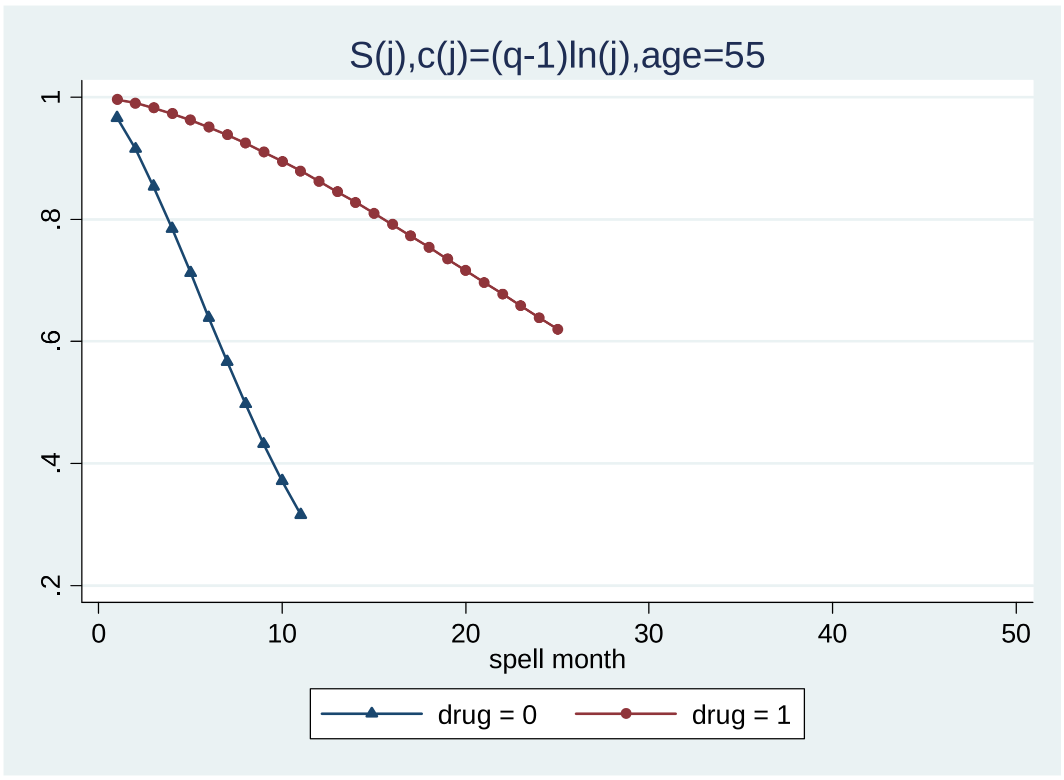
```
. ge h0 = p if age == 55 & drug == 0
(733 missing values generated)

. ge h1 = p if age == 55 & drug == 1
(693 missing values generated)

. lab var h0 "drug = 0"
. lab var h1 "drug = 1"

. ge s0 = s if age == 55 & drug == 0
(733 missing values generated)

. ge s1 = s if age == 55 & drug == 1
(693 missing values generated)

. lab var s0 "drug = 0"

. lab var s1 "drug = 1"
```

Now we can produce the graphs. Here they are; first, for the hazard functions, and second for the survivor functions. A commentary then follows.

```
twoway (connect h0 j , sort  msymbol(t) ) (connect h1 j, sort msymbol(o) ) ///
>         , title("p(j),c(j)=(q-1)ln(j),age=55") saving(dclog1, replace)
(file dclog1.gph saved)
```



```
. twoway (connect s0 j if age == 55, sort  msymbol(t) ) (connect s1 j, sort msymbol(o) ) ///
>         , title("S(j),c(j)=(q-1)ln(j),age=55") saving(dclog2, replace)
(file dclog2.gph saved)

. drop h h0 h1 s s0 s1
```



Survival times for 55 year-old placebo recipients are much shorter than for 55 year-old drug recipients (hazards are higher, as the first graph shows). The median for the former group is about 7 months, but cannot be estimated using the in-sample data for the latter group. (Extrapolation suggests a median nearer 30 months.)

The graphs are truncated at the maximum survival times in the data set for our representative persons. One might want to extrapolate beyond these times. Also the within-sample prediction method has to use covariate combinations that are already present in the data. (One couldn't derive predictions for mean age (say) unless rounded since age is an integer variable in this data set.) To handle this one needs out-of-sample predictions. I return to these later; first we redo the estimation using the non-parametric hazard specification.

One thing we have to do first is:

```
. set matsize 100
```

If we had not done this before running the command, we would have got

```
matsize too small; type -help matsize-
r(908);
```

Matsize refers to the maximum size of the matrices which Intercooled Stata can handle. The number of covariates in your regressions must be less than the value of **matsize**. (The built-in Intercooled Stata maximum is matrices 800x800, hence maximum matsize 800. For a larger maximum, you need Stata Special Edition.)

Now we can estimate the model with non-parametric baseline, making sure we exclude from the estimation those intervals in which there are no failures.

```
. cloglog dead drug age d1-d8 d10-d13 d15-d17 d22-d25 d28 d33 /*
>       */ if (j>=1 & j<=8) | (j>=10 & j<=13) /*
>       */ | (j>=15 & j<=17) | (j>=22 & j<=25) /*
>       */ | j==28 | j==33 , nocons nolog

Complementary log-log regression                Number of obs     =        573
                                                Zero outcomes     =        542
                                                Nonzero outcomes  =         31

                                                Wald chi2(23)     =     178.32
Log likelihood = -96.797174                     Prob > chi2       =     0.0000

------------------------------------------------------------------------------
        dead |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        drug |  -2.455159   .4736798    -5.18   0.000    -3.383554   -1.526763
         age |    .120896   .0376723     3.21   0.001     .0470597    .1947324
          d1 |  -9.321509   2.333704    -3.99   0.000    -13.89548   -4.747533
          d2 |  -9.888202   2.423303    -4.08   0.000    -14.63779   -5.138616
          d3 |  -9.841989   2.426089    -4.06   0.000    -14.59704   -5.086942
          d4 |  -9.008133   2.293928    -3.93   0.000    -13.50415   -4.512116
          d5 |  -8.758807   2.243463    -3.90   0.000    -13.15591     -4.3617
          d6 |  -8.617639   2.218279    -3.88   0.000    -12.96539   -4.269893
          d7 |  -9.269886    2.32007    -4.00   0.000    -13.81714   -4.722632
          d8 |  -8.075466   2.179966    -3.70   0.000    -12.34812   -3.802812
         d10 |   -8.93185   2.327972    -3.84   0.000    -13.49459   -4.369108
         d11 |  -8.144975   2.229088    -3.65   0.000    -12.51391   -3.776044
         d12 |  -7.819549   2.204512    -3.55   0.000    -12.14031   -3.498786
         d13 |  -8.275143   2.281494    -3.63   0.000    -12.74679   -3.803498
         d15 |  -8.190083   2.286423    -3.58   0.000    -12.67139   -3.708776
         d16 |  -8.068548   2.282111    -3.54   0.000     -12.5414   -3.595693
         d17 |  -7.959321   2.278477    -3.49   0.000    -12.42505   -3.493589
         d22 |  -6.799638   2.160356    -3.15   0.002    -11.03386   -2.565418
         d23 |  -6.231211   2.166325    -2.88   0.004    -10.47713   -1.985291
         d24 |  -6.597666   2.301049    -2.87   0.004    -11.10764   -2.087693
         d25 |  -6.481676   2.300701    -2.82   0.005    -10.99097   -1.972384
         d28 |  -6.293316   2.318011    -2.71   0.007    -10.83653   -1.750098
         d33 |  -5.654195    2.36762    -2.39   0.017    -10.29464   -1.013745
------------------------------------------------------------------------------
```

The coefficients on age is much the same as before, and the one on drug is slightly larger in magnitude. The estimated coefficients on the duration interval dummies tell us about the shape of the baseline hazard. Larger (less negative) values are associated with higher hazards. As we would expect from the earlier Lessons, the hazard generally rises over time but non-monotonically.

There is an important issue of interpretation, however. We cannot say anything about the hazard in the months in which no event occurred – there is no information in the sample to identify this. We need to make some additional assumption(s) for identification and thence generated predicted hazards (and survivor functions) as we did before. One strategy might be to suppose that the hazard in this case is equal to zero, but this is rather implausible (or, rather, there is no obvious rationale for such an assumption).

A more plausible way of proceeding is to suppose that the hazard is constant over a longer interval than a month in some cases. Then, to refit the model, we need new dummy variables reflecting this coarser grouping. We have already created them: see dur1–dur6 earlier. So our new regression is:

```
. cloglog dead drug age dur1 dur2 dur3 dur4 dur5 dur6, nocons nolog

Complementary log-log regression                Number of obs     =        744
                                                Zero outcomes     =        713
                                                Nonzero outcomes  =         31

                                                Wald chi2(8)      =     240.94
Log likelihood = -111.59218                     Prob > chi2       =     0.0000

------------------------------------------------------------------------------
        dead |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        dur1 |  -8.794418   2.153186    -4.08   0.000    -13.01459    -4.57425
        dur2 |  -8.184419   2.079405    -3.94   0.000    -12.25998   -4.108861
        dur3 |   -8.18634   2.074564    -3.95   0.000    -12.25241   -4.120269
        dur4 |  -7.396003   2.064883    -3.58   0.000     -11.4431   -3.348906
        dur5 |  -7.224446   2.177626    -3.32   0.001    -11.49251   -2.956377
        dur6 |  -7.265841   2.282069    -3.18   0.001    -11.73861   -2.793069
        drug |  -2.177547   .4310298    -5.05   0.000     -3.02235   -1.332744
         age |   .1127813   .0367963     3.07   0.002     .0406619    .1849007
------------------------------------------------------------------------------
```

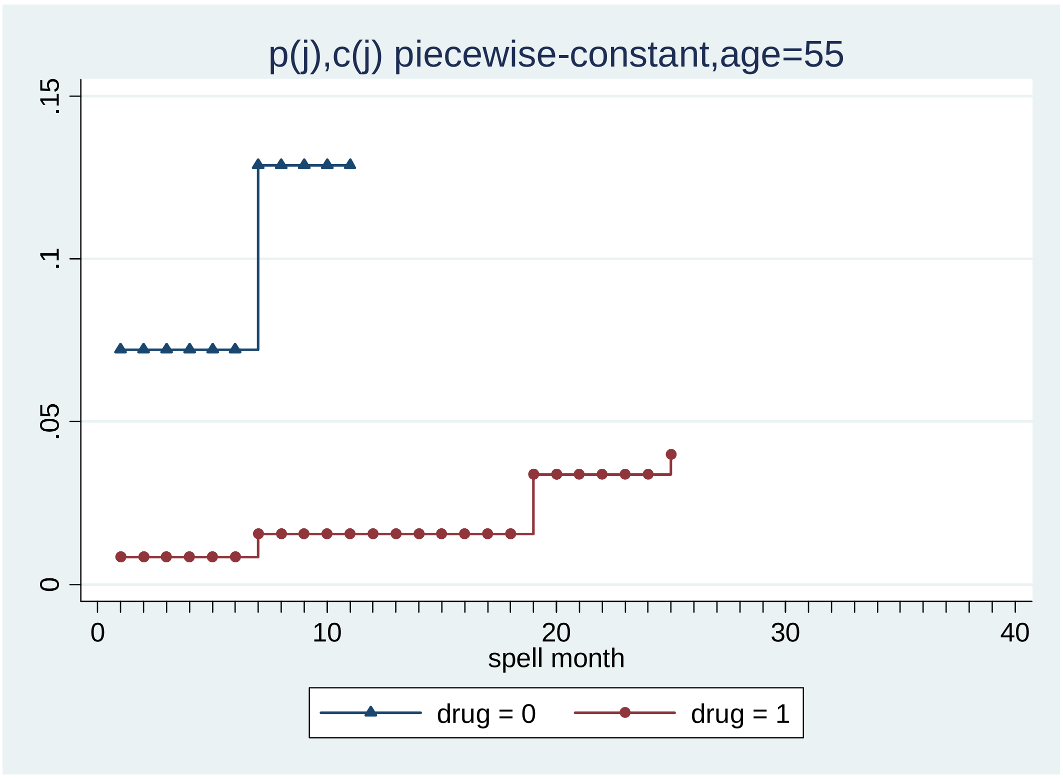
The estimated coefficients on the duration dummy variables rise in magnitude as survival time increases, broadly speaking, which suggests that the hazard rises over time.

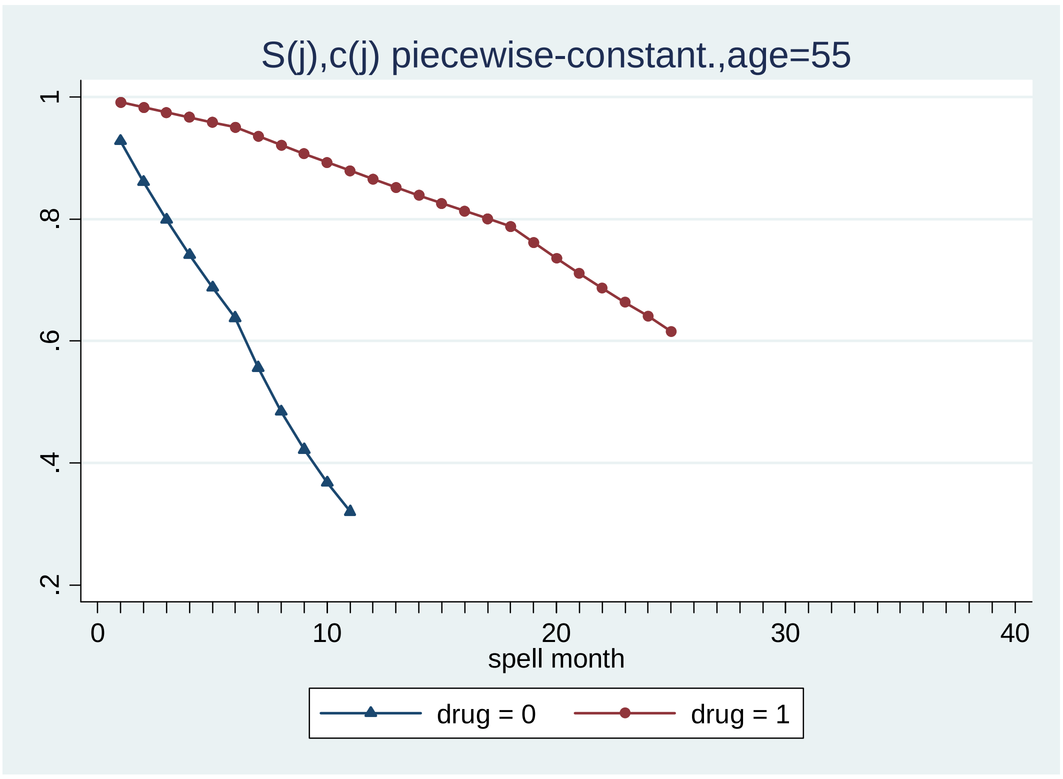
We can explore this graphically for our two 55-year olds with just a slight change to the prediction commands used earlier (after first dropping the variables we used before in order to reuse the names).

```
. drop h h0 h1 s

. predict h, p

. bysort id (j): ge s = exp(sum(ln(1-h)))
.
. ge h0 = p if age == 55 & drug == 0
(733 missing values generated)

. ge h1 = p if age == 55 & drug == 1
(693 missing values generated)

. ge s0 = s if age == 55 & drug == 0
(733 missing values generated)

. ge s1 = s if age == 55 & drug == 1
(693 missing values generated)

. lab var s0 "drug = 0"

. lab var s1 "drug = 1"

. lab var h0 "drug = 0"

. lab var h1 "drug = 1"
```

That's the calculations done; now we can draw the graphs. One small addition to the code used last time to draw the hazard rates is the **connect(J)**, which connects the points in a 'stairstep' manner to highlight the step nature of the piece-wise constant specification.

```
. twoway (connect h0 j, sort  msymbol(t) connect(J) ) ///
>         (connect h1 j, sort msymbol(o) connect(J) ) ///
>         , xtick(1(1)39) title("p(j),c(j) piecewise-constant,age=55") ///
>         saving(dlogit7, replace)
(file dlogit7.gph saved)

twoway (connect s0 j, sort  msymbol(t) ) (connect s1 j, sort msymbol(o) ) ///
>              ,  xtick(1(1)39)  title("S(j),c(j)  piecewise-constant,age=55")  saving(dlogit8,
replace)
(file dlogit8.gph saved)

. drop h h0 h1 s s0 s1
```





The hazard function is step-shaped function – by construction according to the piecewise constant assumption. The survivor function is, however, fairly similar to the one that we derived earlier using the log-time duration dependence assumption. Perhaps the most apparent difference is the slightly sharper decline in survival probabilities at longer durations for drug recipients using the current specification – but the value of the sixtieth percentile is little different nonetheless.

6.2 Out-of-sample prediction

Let us now consider out-of-sample predictions for persons with the mean age (55.89) again contrasting drug and placebo recipients. I return to the model with log(time) baseline hazard. One has to use a parametric specification for this or else how can one project beyond the range of survival times in the sample?

There are a few Stata 'tricks' involved in doing these predictions. The principal new one is the addition of some artificial new observations to the data set – these will represent the spell months for our representative persons (I have 50 spell months for the two people). Initially all the variables for all these new spell months are missing (equal to '.'). So I 'fill in' some non-missing values – for the subject identifier variable (id) and for the covariates. Then I use the **predict** command's capability to make out-of-sample predictions. The estimation of the model is based on the original data set, but predictions can be derived for all spell months including the artificial ones which I generated. Finally one can look at the results, either graphically or listed. Look through the following code:

First we get the information about mean age (see the discussion of this in the previous Lesson).

```
. su age if j==1  /* one obs per person */

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
     age |      48      55.875   5.659205         47         67

. local meana = r(mean)

. di "Average age =  "  `meana'
Average age =  55.875
```

I want to create 50 new rows in the dataset – they will hold the information about the characteristics of the hypothetical person for whom predictions are made. The new number of observations in the data set will be the current total number of observations (_N) plus 50. I also want to know the maximum value of the id variable, because later I need to create a new value of this variable for the new observation.

```
. *** drug == 0, age == mean
. local newn = _N + 50

. su id

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
      id |     744    32.21371   12.24805          1         48

. local idmax = r(max)

. set obs `newn'
obs was 744, now 794
```

Now let's do the regression:

```
. cloglog dead drug age lnj, nolog

Complementary log-log regression                Number of obs     =        744
                                                Zero outcomes     =        713
                                                Nonzero outcomes  =         31

                                                LR chi2(3)        =      35.20
Log likelihood = -111.26371                     Prob > chi2       =     0.0000

------------------------------------------------------------------------------
        dead |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        drug |   -2.18907   .4110876    -5.33   0.000    -2.994787   -1.383353
         age |    .119348   .0371648     3.21   0.001     .0465064    .1921896
         lnj |   .6402733   .2454492     2.61   0.009     .1592017    1.121345
       _cons |  -9.928747   2.272995    -4.37   0.000    -14.38374   -5.473759
------------------------------------------------------------------------------
```

Now we create the value of the id variable for the new person, and also the survivor time variable and the value of the explanatory variables (otherwise they would be left as missing). Observe how the **replace** commands are done only for the person with appropriate id number.

```
. replace id = `idmax' + 1 if id==.
(50 real changes made)

. sort id

. by id: replace j = _n if id==(`idmax' + 1)

. replace lnj = ln(j) if  id==(`idmax' + 1)
(50 real changes made)

. replace drug = 0 if  id==(`idmax' + 1)
(50 real changes made)

. replace age = `meana' if id==(`idmax' + 1)
age was byte now float
(50 real changes made)
```

We now have valid values for all the variables and for all the person-months in the data (including the new person). It's time to take advantage of the fact that **predict** will generate predictions for all observations in the active data set, and not simply the observations that were used to run the regression.

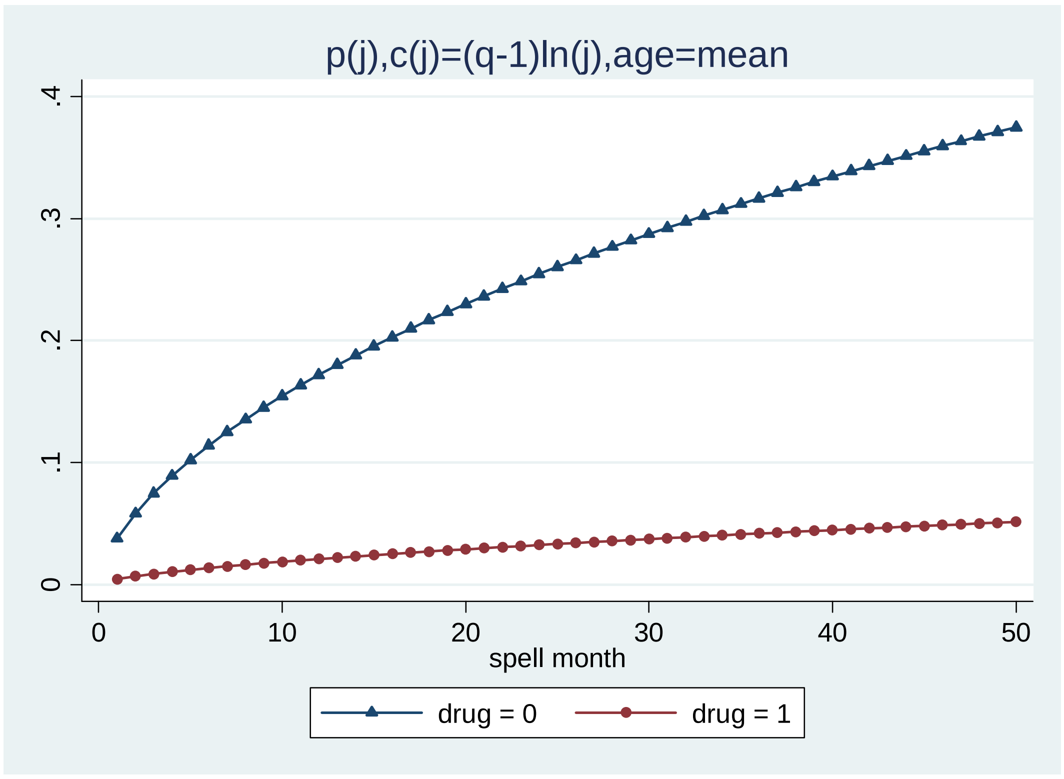
```
. predict h0 if id==(`idmax' + 1), p
(744 missing values generated)

. lab var h0 "drug = 0"

. bysort id (j): ge s0 = exp(sum(ln(1-h0))) if id==(`idmax' + 1)

. lab var s0 "drug = 0"

. * sample size right?
. su id drug age j lnj studytim

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
      id |     794    33.27078    12.5381          1         49
    drug |     794    .7103275   .4538963          0          1
     age |     794    54.67742   5.024352         47         67
       j |     794    12.44962   9.626603          1         50
     lnj |     794    2.156627   .9497978          0   3.912023
studytim |     744    22.14516   9.770883          1         39
```

That's the computations for the first person. Now we repeat all the steps for the second person (the 55-year old drug recipient). We are adding yet another 50 rows to the dataset.

```
. *** now repeat for drug == 1, age == 55
. local newn = _N + 50

. su id

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
      id |     794    33.27078    12.5381          1         49

. local idmax = r(max)

. set obs `newn'
obs was 794, now 844

. replace id = `idmax' + 1 if id==.
(50 real changes made)

. sort id

. by id: replace j = _n if id==(`idmax' + 1)

. replace lnj = ln(j) if id==(`idmax' + 1)
(50 real changes made)

. replace drug = 1 if id==(`idmax' + 1)
(50 real changes made)

. replace age = `meana' if id==(`idmax' + 1)
(50 real changes made)

. predict h1 if  id==(`idmax' + 1), p
(794 missing values generated)

. lab var h1 "drug = 1"

. bysort id (j): ge s1 = exp(sum(ln(1-h1))) if id==(`idmax' + 1)

. lab var s1 "drug = 1"

. * sample size right?
. su id drug age j lnj studytim

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
      id |     844    34.26185   12.78656          1         50
    drug |     844    .7274882   .4455158          0          1
     age |     844    54.67742   4.873072         47         67
       j |     844    13.22275   10.44173          1         50
     lnj |     844    2.204786   .9651106          0   3.912023
studytim |     744    22.14516   9.770883          1         39
```

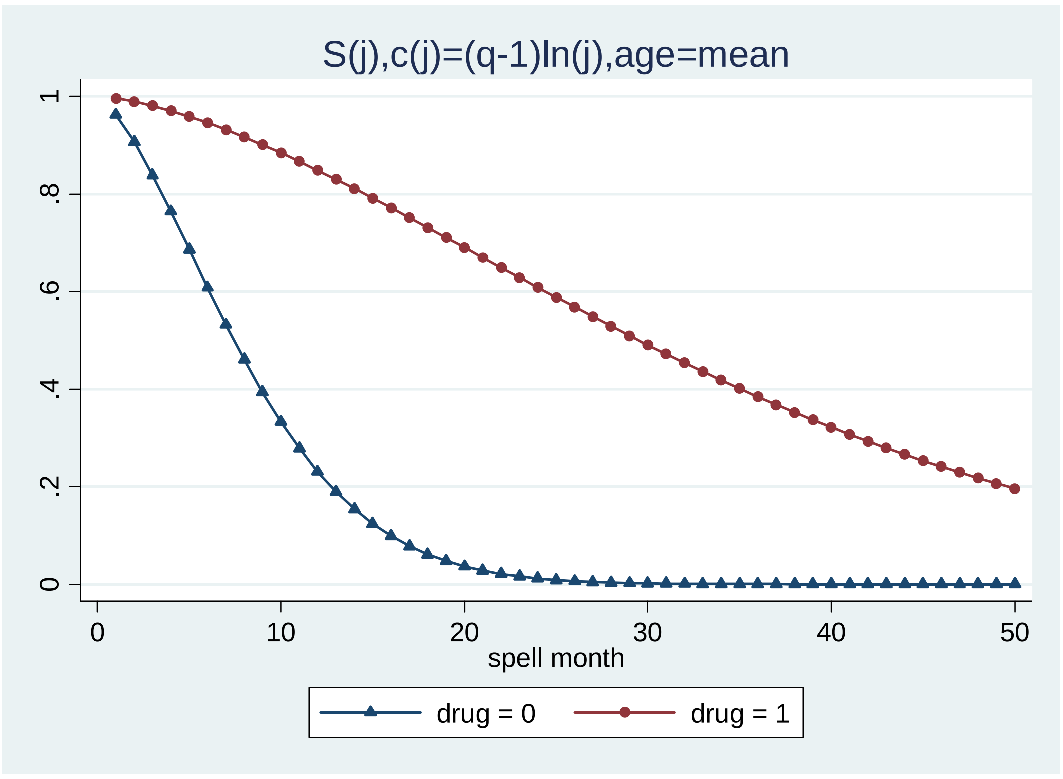
Now, finally, we can summarize the results in a graph. First comes the predicted hazard function, followed by the predicted survivor function.

```
. twoway (connect h0 j, sort  msymbol(t) ) (connect h1 j, sort msymbol(o) ) ///
>         , xlabel(0(10)50) title("p(j),c(j)=(q-1)ln(j),age=mean") saving(dclog13, replace)
(file dclog13.gph saved)
```



Here is the predicted survivor function.

```
twoway (connect s0 j, sort  msymbol(t) ) (connect s1 j, sort msymbol(o) ) ///
>          , xlabel(0(10)50) title("S(j),c(j)=(q-1)ln(j),age=mean") saving(dclog14, replace)
(file dclog14.gph saved)
```



Now I list the predicted survival times for our two representative cases, and ‘tidy up’ the data set, dropping the extra variables and spell months. The listing is used to find the median survival time and uses the **abs(.)** function to limit the number of cases printed out (see Lesson 2 for further discussion).

```
. sort j

. list id j s0 s1 if ((abs(s0-.5) < .1)|(abs(s1-.5) < .1)) & id > `idmax'-1

     +--------------------------------+
     | id    j          s0         s1 |
     |--------------------------------|
310. | 49    7   .5263065          . |
339. | 49    8   .4538711          . |
725. | 50   25          .    .5899806 |
742. | 50   26          .    .5696623 |
744. | 50   27          .    .5495532 |
     |--------------------------------|
758. | 50   28          .    .5296869 |
766. | 50   29          .    .5100948 |
778. | 50   30          .    .4908051 |
780. | 50   31          .    .4718437 |
794. | 50   32          .    .4532338 |
     |--------------------------------|
798. | 50   33          .     .434996 |
803. | 50   34          .    .4171489 |
     +--------------------------------+

. drop h0 h1 s0 s1

. drop if id > `idmax'-1
(100 observations deleted)
```

Consistent with the pictures for the 55 year-olds shown earlier, the median for the mean-age placebo recipient is between 7 and 8 months and for the mean-age drug recipient between 29 and 30 months.

7 Estimation of the logit (proportional odds) model and derivation of predicted hazard and survivor functions.

Estimation in this case is so similar to that for the cloglog model that it is left as an exercise (see Exercise 6.1). It is virtually as easy as substituting **logit** for **cloglog** in the code above, and prediction uses the same code.

To take just a single illustration of the **logit** model, consider the following specification which is a discrete time analogue of the piece-wise constant exponential model considered in Lessons 3 and 5. In that case, we assumed that the (continuous time) hazard rate was constant between survival times (0, 8], (8, 17], and (17, ∞) where the numbers refer to exact dates. Now we are assuming that the interval (discrete) hazard is constant in months 1–8, 9–17, and 18+; we created the necessary dummy variables corresponding to these intervals earlier in the Lesson. They are the variables called e1, e2, and e3. For the reasons discussed in the previous Lesson, we include only two of these as regressors and also include a constant term. (Alternatively one could include all three dummies, but then one must also exclude the constant term using the **nocons** option.)

```
. logit dead drug age e2 e3, nolog

Logistic regression                               Number of obs   =        744
                                                  LR chi2(4)      =      32.39
                                                  Prob > chi2     =     0.0000
Log likelihood = -112.66968                       Pseudo R2       =     0.1257

------------------------------------------------------------------------------
        dead |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        drug |  -2.179088   .4349941    -5.01   0.000    -3.031661   -1.326516
         age |   .1130895   .0380582     2.97   0.003     .0384967    .1876823
          e2 |   .5014397   .4645537     1.08   0.280    -.4090689    1.411948
          e3 |   1.181932   .5274603     2.24   0.025     .1481289    2.215735
       _cons |  -8.610281   2.187779    -3.94   0.000    -12.89825   -4.322313
------------------------------------------------------------------------------
```

Let’s look at the exponentiated coefficients – these give us the odds ratios. We replay the results, adding the **or** option:

```
. logit, or

Logistic regression                               Number of obs   =        744
                                                  LR chi2(4)      =      32.39
                                                  Prob > chi2     =     0.0000
Log likelihood = -112.66968                       Pseudo R2       =     0.1257

------------------------------------------------------------------------------
        dead | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        drug |   .1131446   .0492172    -5.01   0.000     .0482354    .2654004
         age |   1.119732    .042615     2.97   0.003     1.039247     1.20645
          e2 |   1.651097   .7670232     1.08   0.280     .6642685    4.103943
          e3 |   3.260668   1.719873     2.24   0.025     1.159662    9.168149
------------------------------------------------------------------------------
```

The estimates imply that the odds of dying for drug recipients is about one-tenth the odds for placebo recipients. For age, the estimate implies that the odds of dying increase by about 13% with each extra year of age. But ask yourself if you understand what these changes in odds mean. I find it easier to understand hazard ratios. As it happens, the odds ratios from the **logit** estimates happens to be very similar to the hazard ratios from the **cloglog** estimates. For an explanation of this, see the beginning of the Lesson.

8 Exercise 6.1

1. Redo the logistic hazard analysis shown in the illustrations above, but this time use different baseline hazard functions: (a) piece-wise constant, and (b) cubic polynomial in *j*. In both cases, not only run the regressions but also compare the hazard and survival functions for the same two people (age = 55; drug = 0, 1).
2. Experiment with derivation of the hazard and survivor functions for persons with different characteristics, e.g. compare persons aged 50 and 60.
3. Repeat the estimation of the logistic model with fully non-parametric baseline but this time dropping the **if** qualifiers, i.e. use the shorter command **logit dead drug age d1-d39, nocons**. Explain the ‘perfect prediction’ notes which Stata reports.
4. Repeat the illustrations above but this time using the cloglog specification rather than the logistic. Focus on the case for which $c(j) = (q–1)\ln(j)$ if you are short of time. How similar are the **logistic** and **cloglog** results?
5. Run the following commands in sequence (i) **cloglog dead drug age lnj**, (ii) **cloglog, eform**, (iii) **glm dead drug age lnj, f(b) l(c)**, and (iv) **glm, eform**. (You might want to **help glm** first.) Comment on the relationship between the estimates from the commands.
6. Consider again the cloglog model at the end of the Lesson which assumed the hazard was constant within in each of three time intervals. Consider how you might use Wald and likelihood ratio tests to test whether the discrete hazard was the same for the first two intervals. See **help test** and **help lrtest**.