Estimation of a latent class discrete choice panel data model via Maximum Likelihood and EM algorithms in Stata

Pacifico, Daniele

University of Bologna - Department of Economics, Center for the Analysis of Public Policy - University of Modena and Reggio Emilia

20 November 2009

```
                        *******QUOTE THIS FILE IF YOU USE IT!*******

***This file shows how to estimate a latent class model for discrete choice panel
data models
***The (same) model is estimated in three different ways (each of them with pros and
cons):

*** - Via GLLAMM (a user-written program, see www.gllamm.org)

*** - Via Maximum Likelihood (you need to know how to use the optimisation tool in
Stata, see
***      Gould, W. and Pitblado, J. and Sribney, W. (2005), "Maximum Likelihood
Estimation with Stata, Stata Corporation.

*** - Via an EM algorithm

***Note that the underlying theory for both the ML and the EM estimation can be found
in:
***Pacifico D. (2009) "Modelling unobserved heterogeneity in Discrete choice models
of labour supply"

*** Finally,notice that this do files works with data that you can find here:
***http://www.gllamm.org/books/coffee.html , "coff.dat"
***put this data into your Stata editor and then you can load this file (I saved the
data in the dta file called "coffee").

clear
set more off
use coffee,clear

                        ************************************
                        *******One latent class model *******
                        *** (no unobserved heterogeneity) ***
                        ************************************
clogit  ch brand1 brand2 cap1 cap2 price1 price2 therm filter, group(ind)



                        ***************************************
                        ***Two latent class model via GLLAMM***
                        ***************************************
**Note: GLLAMM can be installed through the following command: "ssc install gllamm"
**See: Skrondal, A. and Rabe-Hesketh, S. (2004), "Generalized Latent Variable
Modeling: Multilevel, Longitudinal and Structural Equation Models", Boca Raton,
Chapman & Hall/CRC
**to understand how to use GLLAMM..

eq brand1: brand1
eq brand2: brand2
eq cap1: cap1
eq cap2: cap2
eq price1: price1
eq price2: price2
eq therm: therm
eq filter: filter
matrix a=(-.3741151,-.4046775,-2.484026,.0602947,1.96566,1.482599,1.13633, /*
  */ .9201916,.9212055,.1210968,-1.434993,-.2476023,.0675716,-.489632, /*
  */ -.0361355,.346111,1.00166)
set more off
gllamm alt, i(id) nrf(8) eqs(brand1 brand2 cap1 cap2 price1 price2 therm filter) /*
```

```
  */ nocons l(mlogit) f(binom) expand(ind ch o) nip(2) ip(fn) from(a) copy trace allc



                ************************************************
                ***Two latent class model via Maximum Likelihood***
                ************************************************
***Plese, note that you need to install the Stata program "gprod"
***Type: "findit gprod" for more information


sort ind alt
capture program drop clogit_sim_d0
program define clogit_sim_d0
  sort ind alt
  args todo b lnf
  tempvar e1 L1 L11 l1 L22 l2 beta1 lnfi last beta2 numer1 sum1 denom1 L2 numer2 sum2
denom2
  tempname p1

local d "$ML_y1"
mleval `beta1' = `b', eq(1)
mleval `beta2' = `b', eq(2)
mleval `e1'    = `b', eq(3) scalar
qui gen double `p1'=invlogit(`e1')

sort ind alt
qui gen double `numer1'=exp(`beta1')
qui by ind: gen double `sum1'=sum(`numer1')
qui by ind: gen double `denom1'=`sum1'[_N]
qui gen double `L1'=(`numer1'/`denom1') if `d'==1
sort id ind alt
qui by id: egen double `L11'=prod(`L1')
qui by id: gen double `l1'=`L11'[_N] if _n==_N

sort ind alt
qui gen double `numer2'=exp(`beta2')
qui by ind: gen double `sum2'=sum(`numer2')
qui by ind: gen double `denom2'=`sum2'[_N]
qui gen double `L2'=(`numer2'/`denom2') if `d'==1
sort id ind alt
qui by id: egen double `L22'=prod(`L2')
qui by id: gen double `l2'=`L22'[_N] if _n==_N

sort ind alt
qui gen double `lnfi'=ln((1-`p1')*`l1'+`p1'*`l2')
qui recode `lnfi' .=0
qui mlsum `lnf'=`lnfi'
if (`todo'==0 | `lnf'>=.) exit
end

ml model d0 clogit_sim_d0 (ch=brand1 brand2 cap1 cap2 price1 price2 therm filter,
nocons) (brand1 brand2 cap1 cap2 price1 price2 therm filter, nocons) ()
ml init -.3741151 -.4046775 -2.484026 .0602947 1.96566 1.482599 1.13633 .9201916
.9212055 .1210968 -1.434993 -.2476023 .067571 -.489632 -.0361355 .346111 1.00166,
copy
ml max, diff
```

```
                        ******************************************
                        ***Two latent class model via EM Algorithm***
                        ******************************************
**Note that the EM routine I wrote here is quite general: here below you have to
insert the inputs (the name of the dependent variable, the number of latent calsses
etc etc..
**then the routine should work fine also with other datasets..

global depvar "ch"
global varlist "brand1 brand2 cap1 cap2 price1 price2 therm filter"
global alternative "alt"
global id "id"
**note $id defines the panel dimension (the choice makers)
global idt "ind"
**note: $idt defines the choice situations for each choice maker
global nclasses "2"
global niter "60"
**NOTE: EM algorithms have been proved to be very slow and they may converge to local
maxima. Hence, I suggest you to
**try with different starting values (the routine already computes random starting
values whenever it is launched).. Set
**also a (relative) large number of iterations...

                                ************************

**Get starting values
display "Partition the sample into $nclasses subsamples in orrder to get different
starting values for each class"
sort $id $idt $alt
by $id: gen double p=uniform() if _n==_N
by $id: egen double pr=sum(p)
global prop 1/$nclasses
gen double ss=1 if pr<=$prop
forvalues s=2/$nclasses{
replace ss=`s' if pr>(`s'-1)*$prop & pr<=`s'*$prop
}
drop p pr
ta ss


display "Estimate a separate CLOGIT for each subsample; after each estimation, use
the predict command to get the probability of each alternative"
forvalues s=1/$nclasses{
clogit $depvar  $varlist  if ss==`s', group($idt)
predict double l_`s'
}

**1.3) define (equal) shares for the starting values:
forvalues s=1/$nclasses{
gen double prob`s'=$prop
}


display "Multiply l_1,l_2,..,ls by the dummy variable that identifies the observed
choice so as to pick only the probability of the observed choice"
display "then, for each choice maker, multiply the probabilities of the observed
choices in each choice situation"
forvalues s=1/$nclasses{
qui gen double kbb`s'=l_`s'*$depvar
qui recode kbb`s' 0=.
```

```
by $id: egen double kbbb`s'=prod(kbb`s')
by $id: replace kbbb`s'=. if _n!=_N
}


**compute a weighed sum of the previous variables (kbbb1, kbbb2, etc) with weights
given by the shares s1, s2,..etc:
gen double den=prob1*kbbb1
forvalues s=2/$nclasses{
replace den=den+prob`s'*kbbb`s'
}

display "Now compute the individual-specific weights:"
forvalues s=1/$nclasses{
gen double h`s'=(prob`s'*kbbb`s')/den
qui recode h`s' .=0
}

display "Rearrange the previous variables in order to create individual-level
variables with the values of h1,h2,...,etc"
forvalues s=1/$nclasses{
by $id: egen double H_`s'=sum(h`s')
}

display "Compute the value of the likelihood at the first iteration"
qui egen sumll=sum(ln(den))
qui sum sumll
display as green "Iteration " 1 ":  log likelihood = " as yellow r(mean)

display "Start loop"
set more off
local i= 1
while `i'<= $niter {
quietly{
drop l_*
***estimate again the C clogit models (one for each class using the updated weights:
forvalues s=1/$nclasses{
clogit $depvar  $varlist  [iw=H_`s'], group($idt)
predict double l_`s'
}

***update the contribution to the likelihood for each choice maker:
capture drop kbbb*
forvalues s=1/$nclasses{
replace kbb`s'=l_`s'*$depvar
qui recode kbb`s' 0=.
by $id: egen double kbbb`s'=prod(kbb`s')
by $id: replace kbbb`s'=. if _n!=_N
}
replace den=prob1*kbbb1
forvalues s=2/$nclasses{
replace den=den+prob`s'*kbbb`s'
}

***update the weights using the conditional probabilities:
forvalues s=1/$nclasses{
replace h`s'=(prob`s'*kbbb`s')/den
recode h`s' .=0
capture drop nums`s'
egen double nums`s'=sum(h`s')
}
```

```
capture drop dens
gen double dens=nums1
forvalues s=2/$nclasses{
replace dens=dens+nums`s'
}
forvalues s=1/$nclasses{
replace prob`s'=nums`s'/dens
}
drop H_*
forvalues s=1/$nclasses{
by $id: egen double H_`s'=sum(h`s')
}

***repeate maximizations:
local i=`i' +1

***recompute the value of the likelihood:
drop sumll
egen sumll=sum(ln(den))
 sum sumll
global z=r(mean)
}
display as green "Iteration " `i' ":  log likelihood = " as yellow $z
}


**display the final parameters:
drop l_*
forvalues s=1/$nclasses{
clogit $depvar  $varlist  [iw=H_`s'], group($idt)
predict double l_`s'
}
```