Estimating nonparametric mixed logit models via EM algorithm

Daniele Pacifico

Centre for the Analysis of Public Policies and
University of Bologna
daniele.pacifico@unibo.it

Abstract

The aim of this paper is to describe a Stata routine for the nonparametric estimation of mixed logit models using a Expectation-Maximisation algorithm. We also compare the performance of our estimator with respect to more typical parametric mixed logit models estimated by means of Simulated Maximum Likelihood.

Keywords: EM algorithm, latent class, mixed logit model, unobserved heterogeneity

1. Introduction

In a recent paper Train (2008) has showed how EM algorithms can be used for the nonparametric estimation of mixing distributions in discrete choice models. In this paper we consider one of the three nonparametric methods he proposes and show how it can be implemented in Stata 11. In particular, the method we present allows estimating a discrete mixing distribution with points and shares as parameters. This constitutes a typical latent class model with a large number of classes so as to approximate the true underlying unobserved distribution. Clearly, latent class models can be estimated by standard maximum likelihood but, however, gradient-based optimisation methods as NR or DFP are often difficult, in particular if the number of classes is high. Indeed, as well explained in Train (2008), the higher the number of classes the more difficult the numerical inversion of the Hessian matrix, with the possibility of singularity at some iteration. Nevertheless, an EM procedure can help when the direct maximisation of the likelihood function becomes difficult with traditional optimisation methods since it requires the (repeated) maximisation of a function that is far easier to maximise. In this paper we will actually show the advantage of the estimation via EM algorithm with respect to standard optimisation methods. In particular, EM algorithms have been proved to be very stable and, under conditions given by Boyles (1983) and Wu (1983), these recursions always climb uphill until convergence to a local maximum.

This paper is structured as follows: section 2 presents a mixed logit model based on discrete mixed distributions; section 3 shows how this model can be estimated via EM algorithm; section 4 contains the related Stata codes and section 5 contains an example based on accessible data.

2. A mixed logit model with discrete mixing distributions

Assume there are N agents who face J alternatives on T choice occasions. Agents choose the alternative that maximises their utility in each choice occasion. The random utility of agent i from choosing alternative j in period t is defined as follows:

$$U_{ijt=}\boldsymbol{\beta}_i \boldsymbol{x}_{ijt} + \epsilon_{ijt} \tag{1}$$

Where $\boldsymbol{x}_{ijt}$ is a vector of alternative-specific attributes and ϵ_{ijt} is a stochastic term, which is assumed to be distributed IID extreme value. Importantly, each β_i in vector $\boldsymbol{\beta}_i$ is assumed to be random with unconditional density $f(\beta \,|\, \boldsymbol{\vartheta})$, $\boldsymbol{\vartheta}$ being the parameters that characterise its distribution.

Conditional on knowing $\boldsymbol{\beta}_i$ the probability of the observed sequence of choices for agent i is given by the traditional McFadden's choice model (see McFadden (1973)):

$$Pr_i(\beta_i) = \prod_{t=1}^{T} \left(\frac{exp(\boldsymbol{\beta}_i \boldsymbol{x}_{ijt})}{\sum_{j=1}^{J} exp(\boldsymbol{\beta}_i \boldsymbol{x}_{ijt}))} \right)^{d_{ijt}} \tag{2}$$

where d_{ijt} is a dummy that picks the chosen alternative in each choice occasion. However, since $\boldsymbol{\beta}_i$ is unknown, the conditional probability of the sequence of observed choices has to evaluated for any possible value of $\boldsymbol{\beta}_i$. Hence, assuming that $f(\beta \,|\, \boldsymbol{\vartheta})$ has a continuous distribution in the population, the unconditional probability of the sequence of observed choices is:

$$Pr_i(\boldsymbol{\vartheta}) = \int \prod_{t=1}^{T} \left(\frac{exp(\boldsymbol{\beta} \boldsymbol{x}_{ijt})}{\sum_{j=1}^{J} exp(\boldsymbol{\beta} \boldsymbol{x}_{ijt}))} \right)^{d_{ijt}} f(\boldsymbol{\beta}|\boldsymbol{\vartheta}) \tag{3}$$

Typically, the log likelihood function derived from this probability is estimated using simulation methods (See Train (2003)). In Stata, this can be done by using the command MIXLOGIT written by Hole (2007).

If the distribution of each β_i in the population is discrete, the probability in equation 3 becomes:

$$Pr_i(\boldsymbol{\vartheta}) = \sum_{c=1}^{C} \pi_c(\boldsymbol{\alpha_i}) \prod_{t=1}^{T} \left(\frac{exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ijt})}{\sum_{j=1}^{J} exp(\boldsymbol{\beta_c} \boldsymbol{x}_{ijt}))} \right)^{d_{ijt}} \tag{4}$$

Where $\pi_c = f(\boldsymbol{\beta_c}|\boldsymbol{\alpha})$ represents the share of the population that has coefficients $\boldsymbol{\beta_c}$, which, different from Train (2008), we define as a function of observed demographic characteristics.

This is a typical latent class model but here we follow the classification proposed by McFadden and Train (2000) and define this as a mixed logit model with discrete mixing distributions so as to maintain the parallelism with the continuous mixed logit model described in equation 3. This kind of models are normally estimated via ML but, as we discussed in the introduction, estimating such models with a high number of classes via ML is often difficult because standard gradient-based optimisation methods often fail to achieve convergence[1]. An EM algorithm could help in this case, as it requires the (repeated) estimation of a far simpler likelihood function with respect to the original one.

3. An EM algorithm for the estimation of mixed logit models with discrete mixing distributions

EM algorithms were initially proposed in the literature to deal with missing data problems, although they turned out to be a very good method to estimate latent class models (where the missing data is the class shares). Nowadays, they are widely used in many economic fields where the assumption that people can be grouped in classes with different unobserved taste heterogeneity is reasonable.

The recursion is known as "E-M" because it consists of two steps, namely an "Expectation" and a "Maximisation". As well explained in Train (2008), the term being maximised is the expectation of the *missing-data* log likelihood (i.e. the *joint* density of the observed choice and the missing data), where the expectation is over the distribution of the missing data conditional on the density of the data *and* the previous parameters estimates. Consider the conditional logit panel data model with discrete mixing distributions outlined in the previous section. The log likelihood to be maximised can be defined as follows:

$$LL = \sum_{i=1}^{N} ln \sum_{c=1}^{C} \pi_c(\boldsymbol{\alpha_c}) \prod_{t=1}^{T} \left(\frac{exp(\boldsymbol{\beta_c x}_{ijt})}{\sum_{j=1}^{J} exp(\boldsymbol{\beta_c x}_{ijt}))} \right)^{d_{ijt}} \tag{5}$$

However, the same log likelihood can be also maximised by repeatedly updating the following recursion:

$$\begin{aligned} \boldsymbol{\eta}^{s+1} &= argmax_{\eta} \textstyle\sum_i \sum_c C_i(\boldsymbol{\eta}^s) ln \cdot \pi_c(\boldsymbol{\alpha_c}) \prod_t \left(\frac{exp(\boldsymbol{\beta_c x}_{ijt})}{\sum_{j=1}^{J} exp(\boldsymbol{\beta_c x}_{ijt}))} \right)^{d_{ijt}} \\ &= argmax_{\eta} \textstyle\sum_i \sum_c C_i(\boldsymbol{\eta}^s) ln \cdot (L_i \mid class_i = c) \end{aligned} \tag{6}$$

Where $\boldsymbol{\eta}$ is as vector that contains the whole set of parameters to be estimated (i.e. those that enter the probability of the observed choice plus those that may define the class shares); L_i is the *missing-data* likelihood function and $C(\boldsymbol{\eta}^s)$ is the *posterior* probability that household i belongs to class c, conditional on the density of the data *and* the previous value of the parameters. This conditional probability, $C(\boldsymbol{\eta}^s)$, is the key future of the EM recursion and can be computed by means of the Bayes' theorem:

$$C_i(\boldsymbol{\eta}^s) = \frac{L_i | class_i = c}{\sum_{c=1}^{C} L_i | class_i = c} \tag{7}$$

Now, given the basic fact that:

$$ln\, \pi_c(\boldsymbol{\alpha_c}) \frac{exp(\boldsymbol{\beta_c x}_{ijt})}{\sum_{j=1}^{J} exp(\boldsymbol{\beta_c x}_{ijt}))} = ln\, \pi_c(\boldsymbol{\alpha_c}) + ln \frac{exp(\boldsymbol{\beta_c x}_{ijt})}{\sum_{j=1}^{J} exp(\boldsymbol{\beta_c x}_{ijt}))} \tag{8}$$

[1]This is actually the case with the database used in this paper. Indeed, a ML model with four latent classes fails to achieve convergence.

the recursion in equation 6 can be split into the following steps:

1. Form the contribution to the likelihood $(L_i \,|\, class_i = c)$ as defined in equation 6 *for each class*[2],
2. Form the *individual-specific* posterior probabilities of class membership using equation 7,
3. *For each class*, maximise the *weighted* log likelihood so as to get a new set of $\boldsymbol{\beta_c},\ c = 1, ..., C$:

$$\boldsymbol{\beta_c^{s+1}} = argmax_{\boldsymbol{\beta}} \sum_i C(\boldsymbol{\eta}^s) ln \frac{exp(\boldsymbol{\beta_c x}_{ijt})}{\sum_{j=1}^{J} exp(\boldsymbol{\beta_c x}_{ijt}))} \tag{9}$$

4. Following equation 8, maximise the other part of the log likelihood in equation 6 and get the updated shares $\boldsymbol{\pi_c(\alpha_c)}$:

$$\boldsymbol{\pi_c}(\boldsymbol{\alpha_c^{s+1}}) = argmax_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \sum_{c=1}^{C} C_i(\boldsymbol{\eta}^s) ln\, \pi_c(\boldsymbol{\alpha_c}) \tag{10}$$

   - If the class shares depend on a vector of demographics $\boldsymbol{z_i}$, then the new parameters that specify the contribution of demographics in the definition of the class shares are computed as:

$$\boldsymbol{\alpha^{s+1}} = argmax_{\boldsymbol{\vartheta}} \sum_{i=1}^{N} \sum_{c=1}^{C} C_i(\boldsymbol{\eta}^s) ln \frac{exp(\boldsymbol{\alpha_c z_i})}{\sum_c exp(\boldsymbol{\alpha_c z_i})},\ \boldsymbol{\alpha_C = 0} \tag{11}$$

     This is a *grouped-data log likelihood*, where we have used a logit specification to ensure the estimated class shares to be in the right range. Moreover, notice that the set of $\boldsymbol{\alpha_c}$ c,1,2...,C are estimated jointly so as to ensure that the set of probabilities of class membership adds up to one[3].
   - Update the class shares $\pi_c(\boldsymbol{\alpha_c}),\ c = 1, 2, ..., C$ as:

$$\pi_c(\boldsymbol{\alpha^{s+1}}) = \frac{exp(\boldsymbol{\hat{\alpha}_c^{s+1} z_i})}{\sum_c exp(\boldsymbol{\hat{\alpha}_c^{s+1} z_i})}, \qquad c = 1, 2, ..., C\ ;\ \boldsymbol{\alpha_C = 0} \tag{12}$$

   - Recompute the posterior probability of class membership $C(\boldsymbol{\eta}^{s+1})$ and restart the recursion from point 3 until convergence.
   - Importantly, if the class shares do not depend on demographics, the max-

[2]For the first iteration, starting values have to be used for the densities that enter the model. Importantly, these starting values must be different in every class otherwise the recursion estimates the same set of parameters for all the latent classes.

[3]It is also worth noting that one set of this parameters is set to zero for identification.

imisation in point 4 can be avoided since its solution is simply given by:

$$\pi_c^{s+1} = \frac{\sum_i C_i(\boldsymbol{\eta}^{s+1})}{\sum_i \sum_c C_i(\boldsymbol{\eta}^{s+1})},\ c = 1, ..., C \tag{13}$$

Where $C_i(\boldsymbol{\eta}^{s+1})$ is computed using the updated values of $\boldsymbol{\beta}_c$, c=1,2,...,C (from point 3) and the previous values of the class shares.

4. Stata routine for the estimation of mixed logit models with discrete mixing distributions

In what follows we show how to implement the EM algorithm outlined in the previous section in Stata 11. We present a routine that can be easily replicated in a Stata *do* file so as to explain the procedure. However, we have also coded an *ado* file - `LCLOGIT` - that generalises the procedure and we will show how to use this command in the next section.

We begin by defining the variables that enter the model. In order to create a flexible routine we work with global variables so that the code can be easily used with other databases. The dependent variable is called "*$depvar*", the list of covariates that enter the probability of the observed choice "*$varlist*"; the list of covariates that enter the grouped-data log likelihood "*$varlist2*"; the variable that identifies the panel dimension, i.e. the choice makers "*$ind*" and the variable that defines the choice situations for each choice maker "*$group*". We also define the number of latent classes "*$nclasses*" and the number of maximum iterations "*$niter*".

```
***define global variables and environment**
global depvar "ch"
global varlist "brand1 brand2 cap1 cap2 price1 price2 filter therm"
gen _con=1
global varlist2 "_con"
global id "id"
global idt "ind"
global nclasses "2"
global niter "150"
```

In order to provide Stata with the starting values, we randomly split the sample into C different sub-samples (one for each class) and estimate a separate `CLOGIT` for each sub-sample[4]. As for the starting values for the probability of class-membership we simply define equal shares, that is $\frac{1}{C}$:

[4]If the same starting values are used for all the classes, the EM algorithm performs the same computations for each class and return the same results at each iteration.

```
***Get starting values***
sort $id $group
by $id:  gen double p=runiform() 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
}
```

The next step is to estimate a separate CLOGIT for each sub-sample. After each estimation, we use the PREDICT command to obtain the probability to choose every alternative in each latent classes; call these vectors of probabilities as *l_ , l_ 2,...,l_ C*:

```
forvalues s=1/$nclasses{
qui clogit $depvar $varlist if ss==‘s’, group($group) technique(nr dfp)
predict double l_‘s’
}
```

Define equal shares as starting values for the probability of class membership:

```
forvalues s=1/$nclasses{
gen double prob‘s’=$prop
}
```

We now show the steps needed to calculate the posterior probability defined in equation 7, given the starting values we have computed.
Firstly, we multiply *l_ 1, l_ 2,..,l_ C* by the dummy variable that identifies the observed choice in each choice situation so as to pick only the probability of the observed choice. Secondly, for each choice maker we multiply the probabilities of the observed choices *in each choice situation.* Importantly, this latter point is performed through the Stata program *gprod*, which can be downloaded from the Internet (type: "*findit gprod*" in the Stata command and go to the *dm71* package to install it):

```
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
}
```

The third step is to construct the denominator of equation 7 by computing a weighed sum of the previous variables (*kbbb1, kbbb2, etc.*) with weights given by the probability of class membership (*prob1, prob2,..., etc.*):

```
gen double den=prob1*kbbb1
forvalues s=2/$nclasses{
replace den=den+prob‘s’*kbbb‘s’
}
```

Then we construct the ratio es defined in equation 7 as:

```
forvalues s=1/$nclasses{
gen double h`s'=(prob`s'*kbbb`s')/den
qui recode h`s' .=0
}
```

Finally, we simply rearrange the previous variables (*h1, h2, etc.*) in order to create individual-level variables. These are the conditional probabilities $C(\boldsymbol{\eta}^s)$ as explained in the text:

```
forvalues s=1/$nclasses{
by $id:  egen double H_`s'=sum(h`s')
}
```

Before starting the loop that iterates the EM recursion until convergence, we need to specify a Stata `ml` command to perform the estimation of the grouped-data model defined in equation 11. In what follows we show the commands for a model with *two* latent classes. Of course, the routine can be easily adapted to account for a higher number of latent classes even though the command `LCLOGIT` does it automatically up to 30 classes.

Importantly, it is worth noting that if the probability of class membership does not depend on demographics, then the maximisation of the grouped-data model can be avoided since its solution can be provided analytically following equation 13. This is important, in particular if the number of latent classes is high as the maximisation algorithm slows down the overall procedure.

For this reason, the routine identifies if the model contains demographics in the probability of class membership. This is performed by computing the means of the variables in `$varlist2` and by checking if the (last) mean of these variables is different from 1 (i.e. the mean of the constant term). Only in this case, the routine launches the `ml model`. Finally, notice that, in this latter case one vector of parameters in the `ml model` is set to zero for identification (so that a model with two latent classes needs only one vectors of parameters as input in the maximisation routine):

```
qui su $varlist2
global mean=r(mean)
if $mean!=1{
sort $id $group alt
global nalt=3
**$nalt identifies the number of alternatives in each choice occasion
by $id:  gen double id1=1 if _n<=$nalt
qui recode id1 .=0
capture program drop logit_lf
program logit_lf
args lnf ab
tempvar denom
qui gen double ‘denom’=1+exp(‘ab’)
qui replace ‘lnf’=[H_1*ln(1/‘denom’)+H_2*ln(exp(‘ab’)/(‘denom’))] if $depvar==1 & id1==1
qui recode ‘lnf’ .=0
capture drop prob*
qui gen double prob1=1/‘denom’
qui gen double prob2=exp(‘ab’)/(‘denom’)
end
}
```

We now present the loop that repeats the steps above until convergence:

```
local i=1
while ‘i’<= $niter{
set more off
quietly{
```

Estimate again the C `CLOGIT` models (one for each class) using the conditional probabilities - computed following equation 7 - as weights. Then, recompute the probability of each alternative (the variables l_1, l_2,.., l_C) using the updated parameters:

```
capture drop l_*
forvalues s=1/$nclasses{
qui clogit $depvar $varlist [iw=H_‘s’], group($group) technique(nr dfp)
predict double l_‘s’
}
```

Now simply update the variables constructed before the loop in order to update the conditional probabilities used as weights in the previous sets of maximisations:

```
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
}
```

Importantly, if the model contains demographics for the explanation of the probability of class membership, the routine calls the `ml model` defined before the loop and maximises the grouped-data log likelihood:

```
if $mean!=1{
ml model lf logit_lf ($varlist2, nocons)
ml max
```

Notice that the probability of class membership (“prob1”, “prob2”, etc.) are updated internally when the `ml model` is called (i.e. in the previous two lines). Finally, update the weights using the conditional probabilities:

```
replace den=prob1*kbbb1
forvalues s=2/$nclasses{
replace den=den+prob`s'*kbbb`s'
}
forvalues s=1/$nclasses{
replace h`s'=(prob`s'*kbbb`s')/den
recode h`s' .=0
}
drop H_*
forvalues s=1/$nclasses{
by $id:  egen double H_`s'=sum(h`s')
}
}
```

However, if the model does not contains demographics that help explaining the probability of class membership, the routine provides the solution of the `ml model` analytically according to equation 13 so that the maximisation step can be avoided, increasing significantly the estimation performance:

```
if $mean==1{
replace den=prob1*kbbb1
forvalues s=2/$nclasses{
replace den=den+prob`s'*kbbb`s'
}
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')
}
}
```

When the iteration ends, the routine displays the value of the maximised log likelihood (which is the variable *sumll* defined below) and restarts the loop:

```
capture drop sumll
egen sumll=sum(ln(den))
sum sumll
global z=r(mean)
local i=‘i’ +1
}
display as green "Iteration " ‘i’ ":  log likelihood = " as yellow $z
}
```

Convergence is normally declared when the maximised log likelihood does not change from one iteration to the next.
Since EM algorithms do not compute standard errors, they can be easily obtained by bootstrap. However, when the model contains a large number of classes, summary statistics for the coefficients over the C classes can be more relevant than the values of the coefficients in each class. In this case, the bootstrap command in Stata can be used to compute standard errors for - let's say - the arithmetic average of each coefficient over the C classes. In the next section, we introduce the `LCLOGIT` command, which generalises the procedure outlined here.

5. The `LCLOGIT` command

`LCLOGIT` is an ado file that does not contain an internal *ml* evaluator. Indeed, as we have seen in section 4, the recursion uses only the command `CLOGIT`, which really speeds up the overall computation. Following Weeks and Lange (1989), convergence is declared when the last 5 values of the maximised log likelihood are equal. This is needed because, as discussed in Dempster et al. (1977), EM algorithms can move slowly toward the maximum. When convergence is achieved, `LCLOGIT` stops the internal loop and shows the estimated coefficients in matrix `C`. The user can also list matrix `P`, which contains the corresponding probabilities of class membership, while the global number `$z` contains the value of the maximised log likelihood.

Importantly, at the end of the estimation `LCLOGIT` produces a series of relevant variables: the predicted probabilities for each choice occasion (`_pr`, which correspond to what is normally computed with the postestimation command `predict`); C variables that contain the weights used internally with the `CLOGIT` estimations (`_H1`, `_H2`,...,`_HC`) and C variables that contain the estimated probability of class membership (`_prob1`,`_prob2`,...,`_probC`)[5]

Finally, it is worth noting that the internal routine produces global numbers and ancillary variables (all of them starting with an underscore). Hence, some of these variables might appear in the data when the estimation process is stopped in the middle of a computations. `LCLOGIT` can be downloaded from the Internet through this link:
http://www.danielepacifico.com/LCLOGIT.zip[6].

[5]For this reason, the user may want to increase the virtual memory that Stata can use, in particoular when the number of classes is high. This can be done with the command `set mem`.

[6]The ado file has to be saved in a folder where Stata can find it. By typing `sysdir` in the command line, one can find out which directories Stata looks in for '.ado' files. If the user wants to estimate a model with demographics in the the probabilities of class membership, then an ancillary do file containing the ml program "lclogit_lf.do" has to be saved in the Stata working folder (which can be found by typing `pwd`).

The generic syntax for LCLOGIT is:

```
lclogit depvar [varlist] [if], group(varname) id(varname) nclasses(real) niter(real)
[options]
```

The required options are:

- `id`(varname); it specifies the identifier variable for the choice-makers.
- `idt`(varname); it specifies the identifier variable for the choice occasions.
- `nclasses`(real); it specifies the number of latent classes to be considered.
- `niter`(real); it specifies the number of maximum iterations.

Other not-required options are:

- `seed`(real) so that the user can change the starting values, the default is `1234567890`.
- `varlist2`(namelist), which can be used to introduce demographics in the specification of the probabilities of class membership. Notice that the constant is provided internally so that the user has to specify only individual-specific covariates[7].

6. Application

For our application we use accessible data from Huber and Train (2000) on household's choice of electricity supplier[8]. Importantly, this is the same database used by Hole (2007) for an application of his Stata command MIXLOGIT.

A sample of 100 residential electricity customers were asked up to 12 choice experiments. Since some people stopped before answering all 12 experiments, there are a total of 1195 choice occasions in the sample. In each experiment, the person was asked which of the four suppliers he/she would prefer among four hypothetical electricity suppliers. In the experiments, the characteristics of each offer were stated:

- The price of the contract (in cents per kWh) whenever the supplier offers a contract with a fixed rate (`price`)
- The length of contract that the supplier offered, expressed in years (`contract`)
- Whether the supplier is a local company (`local`)
- Whether the supplier is a "well-known" company (`wknown`)
- Whether the supplier offers a time-of-day rate instead of a fixed rate (`tod`)
- Whether the supplier offers a seasonal rate instead of a fixed rate (`seasonal`)

[7]Importantly, the covariates must not change from one choice occasion to the other.

[8]The complete database is available from Kenneth Train's website in Matlab format for a sample of 361 costumers (http://elsa.berkeley.edu/~train/). In this paper we use a sub-sample composed by the first 100 costumers, which is also the sub-sample used in Hole (2007) for the estimation of parametric mixed logit models by Simulated Maximum Likelihood in Stata.

Notice that when the supplier does not offer a fixed rate, the variable `price` is equal to zero. Each costumer is identified by the variable "`pid`". For each costumer, the variable that identifies a given choice occasion is "`gid`" and the dummy that identifies the stated choice in each choice occasion is "y". The data is already organised according to the setup needed for the command `CLOGIT`. The first 12 rows - as they appear in the Stata editor - are presented below:

```
use http://fmwww.bc.edu/repec/bocode/t/traindata.dta, clear
list in 1/12, sepby(gid)
```

```
     +--------------------------------------------------------------------+
     | y   price   contract   local   wknown   tod   seasonal   gid   pid |
     |--------------------------------------------------------------------|
  1. | 0       7          5       0        1     0          0     1     1 |
  2. | 0       9          1       1        0     0          0     1     1 |
  3. | 0       0          0       0        0     0          1     1     1 |
  4. | 1       0          5       0        1     1          0     1     1 |
     |--------------------------------------------------------------------|
  5. | 0       7          0       0        1     0          0     2     1 |
  6. | 0       9          5       0        1     0          0     2     1 |
  7. | 1       0          1       1        0     1          0     2     1 |
  8. | 0       0          5       0        0     0          1     2     1 |
     |--------------------------------------------------------------------|
  9. | 0       9          5       0        0     0          0     3     1 |
 10. | 0       7          1       0        1     0          0     3     1 |
 11. | 0       0          0       0        1     1          0     3     1 |
 12. | 1       0          0       1        0     0          1     3     1 |
     +--------------------------------------------------------------------+
```

We begin by estimating a conditional logit model using the Stata command `clogit`:

```
clogit y price contract local wknown tod seasonal, group(gid)
Iteration 0:   log likelihood =   -1379.3159
(output omitted)
Iteration 4:   log likelihood =   -1356.3867
Conditional (fixed-effects) logistic regression   Number of obs   =      4780
                                                  LR chi2(6)      =    600.47
                                                  Prob > chi2     =    0.0000
Log likelihood = -1356.3867                       Pseudo R2       =    0.1812
------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       price |  -.6354853   .0439523   -14.46   0.000    -.7216302   -.5493403
    contract |    -.13964   .0161887    -8.63   0.000    -.1713693   -.1079107
       local |   1.430578   .0963826    14.84   0.000     1.241672    1.619485
      wknown |   1.054535    .086482    12.19   0.000     .8850338    1.224037
         tod |  -5.698954   .3494016   -16.31   0.000    -6.383769    -5.01414
    seasonal |  -5.899944     .35485   -16.63   0.000    -6.595437   -5.204451
------------------------------------------------------------------------------
```

From the results above, we can see that, on average, costumers prefer lower prices, shorter contracts length, a local and well-known company and fixed rather than variable rate plans. We now use the command `MIXLOGIT` from Hole (2007) to estimate a parametric mixed logit model with independent normally distributed coefficients by Simulated Maximum Likelihood with 300 Halton draws[9]:

[9] `MIXLOGIT` can be installed in Stata by typing "finditd mixlogit" in the Stata command. See the help file for the syntax.

```
mixlogit y, group(gid) id(pid) rand(price contract local wknown tod seasonal) nrep(300)
Iteration 0:  log likelihood = -1249.8219 (not concave)
(output omitted)
Iteration 7:  log likelihood = -1101.6085

Mixed logit model                                   Number of obs     =     4780
                                                    LR chi2(6)        =   509.56
Log likelihood = -1101.6085                         Prob > chi2       =   0.0000

------------------------------------------------------------------------------------
            y |      Coef.   Std.  Err.       z     P>|z|    [95% Conf.   Interval]
--------------+---------------------------------------------------------------------
Mean          |
        price |  -1.004329    .0721185    -13.93   0.000    -1.145679   -.8629798
     contract |  -.2274985     .047386     -4.80   0.000    -.3203735   -.1346236
        local |   2.208746    .2439681      9.05   0.000     1.730578    2.686915
       wknown |   1.656329    .1707167      9.70   0.000      1.32173    1.990927
          tod |  -9.364151    .5858618    -15.98   0.000    -10.51242   -8.215883
     seasonal |  -9.496181    .5792009    -16.40   0.000    -10.63139   -8.360968
--------------+---------------------------------------------------------------------
SD            |
        price |   .2151655    .0311095      6.92   0.000      .154192    .2761389
     contract |    .384136     .044778      8.58   0.000     .2963728    .4718992
        local |   1.788806    .2370063      7.55   0.000     1.324282     2.25333
       wknown |   1.185838    .1731652      6.85   0.000     .8464401    1.525235
          tod |     1.6553    .2094545      7.90   0.000     1.244777    2.065824
     seasonal |  -1.119371    .2836182     -3.95   0.000    -1.675252   -.5634893
------------------------------------------------------------------------------------
```

As it can be seen from the maximised log likelihood, we can safely reject the conditional logit specification in favour of the parametric mixed logit model[10]. Moreover, the magnitude of the coefficients is significantly different with respect to the model without random coefficients, which gives an indication of the bias produced by the IIA property of conditional logit models[11].

We now show how to use `LCLOGIT` to estimate a nonparametric mixed logit model. As we have explained in section 1, the main idea is to use a latent class framework with a relatively high number of classes so as to approximate the true unobserved distributions of the coefficients. Following Greene and Hensher (2003) and Train (2008), the choice of the appropriate number of latent classes is done by means of some information criteria. From the loop below we estimate latent class mixed logit models up to 15 latent classes. We set a relatively high number of maximum iterations (200) since, as we have explained, EM algorithms may move slowly toward the maximum, in particular if the number of latent classes is high. Notice also that we save the maximised log likelihood in the scalars sll1,sll2,...,sll15:

[10]The two models are nested and a comparison with a log likelihood ratio test is therefore meaningful.

[11]See Bhat (2000)

```
forvalues s=2/15{
lclogit y price contract local wknown tod seasonal, group(gid) id(pid) nclasses(‘s’) niter(200)
scalar sll‘s’=$z
}
forvalues s=2/15{
display sll‘s’
}
```

The routine lasts for about 30 minutes to estimate the whole set of 14 models on our standard-issue PC[12]. In what follows we use the Consistent-AIC and the BIC criteria in order to select the right number of latent classes. According to both these criteria, a model with 8 latent classes is chosen[13]:

Classes	Log likelihood	parameters	CAIC	BIC
1	-1356.39	6	2737.24	2731.24
2	-1211.35	13	2475.71	2462.71
3	-1118.23	20	2318.02	2298.02
4	-1085.3	27	2280.7	2253.7
5	-1040.49	34	2219.61	2185.61
6	-1028.56	41	2224.29	2183.29
7	-1006.37	48	2208.45	2160.45
8	**-990.24**	**55**	**2204.73**	**2149.73**
9	-983.64	62	2220.08	2158.08
10	-979.23	69	2239.8	2170.8
11	-965.76	76	2241.4	2165.4
12	-952.68	83	2243.78	2160.78
13	-947.24	90	2261.44	2171.44
14	-945.59	97	2286.69	2189.69
15	-943.42	104	2310.89	2206.89

The list of parameters for a model with 8 classes can be displayed by typing:

```
matrix list C8
C8[8,6]
             y:          y:          y:          y:          y:          y:
         price    contract       local      wknown         tod    seasonal
y1  -.91040978   -.4381791   .37045253   .36884693   -8.256523  -6.4399279
y1  -.73728307   .21832437   2.4164109   2.8395595  -6.6905693  -7.2133156
y1   -.4883505  -.59163456   .78155976   .70969229  -4.1326455   -6.560601
y1  -2.1095744   -.6621888   .71708341   .24117339  -14.190615  -17.207319
y1  -.64251826   .09629113   2.1856728   1.2073454  -3.8365207  -4.0524238
y1  -1.2084713  -.19842282   6.5797129   5.1040843  -14.852122  -15.340282
y1   -1.532864  -.40906556   .62090963   .93044277  -16.007181  -14.818061
y1  -.08169616  -.15640971   4.9367671   3.4441044  -1.0876165  -1.0597307
```

As Train (2008) points out, when the number of classes rises, summary statistics for the distirbution of each coefficient become more informative than the single coefficients. In this case, the bootstrap command allows computing standard errors

[12]We used a PC with a 2.2GHz Intel core 2 duo and 4MB RAM.

[13]It is worth notinge that the routine took about 3 minutes on our PC to estimate a model with 8 classes.

for these statistics straightforwardly. Let's consider the matrix of coefficients C8. A possible manner to compute standard errors for various summary statistics is to clear the data in Stata and paste this matrix of coefficients in the Stata editor. Then, for example, the standard errors for the mean of the distribution of the coefficient of price can be computed as follows:

```
bootstrap t=r(mean), rep(1000):  sum price
```

The next matrix shows the mean and the standard deviation for the distribution of each coefficient, along with the corresponding z-statistic computed using the bootstrapped standard error:

	Mean	z	St Dev	z
price	-0.964	-4.65	0.639	4.3
contract	-0.268	-2.49	0.315	5.32
local	2.326	3.13	2.288	4.01
wknown	1.856	3.20	1.751	4.37
tod	-8.632	-4.65	5.708	6.27
seasonal	-9.086	-4.77	5.911	6.05

Interestingly, the means of the coefficients are surprisingly similar in magnitude to those obtained with normal mixing distribution[14]. Moreover, we can see that there is a great amount of unobserved preference heterogeneity, as the standard deviations and their standard errors show. By using again the bootstrap command, we can also compute the following correlation matrix with the subsequent standard errors:

	price	contract	local	wknown	tod	seasonal
price	1.000					
contract	0.481* (0.305)	1.000				
local	0.314 (0.369)	0.453* (0.238)	1.000			
wknown	0.303 (0.357)	0.527* (0.256)	0.96* (0.072)	1.000		
tod	0.896* (0.069)	0.382 (0.301)	0.072 (0.467)	0.009 (0.435)	1.000	
seasonal	0.929* (0.058)	0.456* (0.271)	0.080 (0.469)	0.036 (0.450)	0.96* (0.034)	1.000

Note: * means that the coefficient is significant at 95%.

From the correlation matrix we can see that there are significant meaningful correlations. In particular, people who prefer very low prices prefer a lot more fixed rate plans with respect to either a seasonal rate or a TOD rate; moreover, people who strongly prefer a local company have also stronger preferences for well-known local company; furthermore, costumers who have stronger preferences for shorter contracts length have also lower preferences for local companies.

[14]Actually, also some of the standard deviations reported in Hole (2007) are surprisingly similar to those found here, in particular when the coefficients are allowed to be correlated. See Hole (2007) for details.

7. Conclusions

In this article we have shown how to estimate nonparametric mixed logit models in Stata 11 following the approach proposed by Train (2008). In particular, the Stata command - `LCLOGIT` - runs an EM algorithm that, thanks to its desiderable properties, allows estimating latent class models with a high number of classes so that the true, unobserved distribution of the coefficients can be well approximated.

Acknowledgements

I am grateful to Kenneth Train for helpful suggestions during the implementation of the EM algorithm in Stata. Thanks also to Maria Zindili for her important comments and suggestions.

References

Bhat, C., 2000. Flexible model structures for discrete choice analysis. Handbook of transport modelling 1, 71-90.

Boyles, R., 1983. On the convergence of the em algorithm, Journal of the Royal Statistical Society B 45, 47-50.

Dempster, A., Laird, N., Rubin, D., et al., 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological) 39 (1), 1-38.

Greene, W., Hensher, D., 2003. A latent class model for discrete choice analysis: contrasts with mixed logit. Transportation Research Part B 37 (8), 681-698.

Hole, A.R., 2007. Fitting mixed logit models by using maximum simulated likelihood. Stata Journal, 7 (3), 388-401.

Huber, J. and K. Train, 2001. On the similarity of classical and bayesian estimates of individual mean partworths, Marketing Letters 12, 259-269

McFadden, D., 1973. Conditional logit analysis of qualitative choice models. Frontiers of Econometrics, ed. P. Zarembka. New York: Academic Press.

McFadden, D., Train, K., 2000. Mixed MNL models for discrete responses. Journal of Applied Econometrics 15 (5), 447-470.

Train, K., 2003. Discrete choice methods with simulation. Cambridge Books.

Train, K., 2008. EM Algorithms for Nonparametric Estimation of Mixing Distributions. Journal of Choice Modelling 1 (1).

Weeks, D. and K. Lange 2007. Trials,tribulations, and triumphs of the EM algorithm in pegigree analysis. Journal of Mathematics Applied in Medicine and Biology 6, 209-232.

Wu, C., 1983. On the convergence properties of the EM algorithm. The Annals of Statistics 11 (1), 95-103.