Citation for published item:

Further information on publisher’s website:
http://www.stata-journal.com/article.html?article=st0312

Publisher’s copyright statement:

Use policy

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a link is made to the metadata record in DRO
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the full DRO policy for further details.
lclogit: A Stata module for estimating latent class conditional logit models via the Expectation-Maximization algorithm

Daniele Pacifico  
Italian Department of the Treasury  
daniele.pacifico@tesoro.it  

Hong il Yoo  
University of New South Wales  
h.yoo@unsw.edu.au

Abstract. This paper describes lclogit, a Stata module for estimating a discrete mixture or latent class logit model via the EM algorithm.

Keywords: st0001, lclogit, latent class model, EM algorithm, mixed logit

1 Introduction

Mixed logit or random parameter logit is used in many empirical applications to capture more realistic substitution patterns than traditional conditional logit. The random parameters are usually assumed to follow a normal distribution and the resulting model is estimated through simulated maximum likelihood, as in Hole (2007)’s Stata module mixlogit. Several recent studies, however, note potential gains from specifying a discrete instead of normal mixing distribution, including the ability to approximate the true parameter distribution more flexibly at lower computational costs.

Pacifico (2012) implements the Expectation-Maximization (EM) algorithm for estimating a discrete mixture logit model, also known as latent class logit model, in Stata. As Bhat (1997) and Train (2008) emphasize, the EM algorithm is an attractive alternative to the usual (quasi-)Newton methods in the present context, because it guarantees numerical stability and convergence to a local maximum even when the number of latent classes is large. In contrast, the usual optimization procedures often fail to achieve convergence since inversion of the (approximate) Hessian becomes numerically difficult.

With this contribution, we aim at generalizing Pacifico (2012)’s code with a Stata module that introduces a series of important functionalities and provides an improved performance in terms of run time and stability.

2 EM algorithm for latent class logit

This section recapitulates the EM algorithm for estimating a latent class logit model (LCL). Suppose that each of $N$ agents faces, for notational simplicity, $J$ alternatives

1. For example, see Hess et al. (2011), Shen (2009) and Greene and Hensher (2003).
Latent class logit model

in each of \( T \) choice scenarios. Let \( y_{njt} \) denote a binary variable which equals 1 if agent \( n \) chooses alternative \( j \) in scenario \( t \) and 0 otherwise. Each alternative is described by alternative-specific characteristics, \( x_{njt} \), and each agent by agent-specific characteristics including a constant, \( z_n \).

LCL assumes that there are \( C \) distinct sets (or classes) of taste parameters, \( \beta = (\beta_1, \beta_2, ..., \beta_C) \). If agent \( n \) is in class \( c \), the probability of observing her sequence of choices is a product of conditional logit formulas:

\[
P_n(\beta_c) = \prod_{t=1}^{T} \prod_{j=1}^{J} \left( \frac{\exp(\beta_c x_{njt})}{\sum_{k=1}^{J} \exp(\beta_c x_{nkt})} \right)^{y_{njt}}
\]

Since the class membership status is unknown, the researcher needs to specify the unconditional likelihood of agent \( n \)'s choices, which equals the weighted average of equation 1 over classes. The weight for class \( c \), \( \pi_{cn}(\theta) \), is the population share of that class and usually modeled as fractional multinomial logit:

\[
\pi_{cn}(\theta) = \frac{\exp(\theta_c z_n)}{1 + \sum_{l=1}^{C-1} \exp(\theta_l z_n)}
\]

where \( \theta = (\theta_1, \theta_2, ..., \theta_{C-1}) \) are class membership model parameters; note that \( \theta_C \) has been normalized to zero for identification.

The sample log likelihood is then obtained by summing each agent’s log unconditional likelihood:

\[
\ln L(\beta, \theta) = \sum_{n=1}^{N} \sum_{c=1}^{C} \pi_{cn}(\theta) P_n(\beta_c)
\]

Bhat (1997) and Train (2008) note numerical difficulties associated with maximizing equation 3 directly. They show that \( \beta \) and \( \theta \) can be more conveniently estimated via a well-known EM algorithm for likelihood maximization in the presence of incomplete data, treating each agent’s class membership status as the missing information. Let superscript \( s \) denote the estimates obtained at the \( s^{th} \) iteration of this algorithm. Then, at iteration \( s + 1 \), the estimates are updated as:

\[
\beta^{s+1} = \arg \max_{\beta} \sum_{n=1}^{N} \sum_{c=1}^{C} \eta_{cn}(\beta^{s+1}, \theta^{s}) \ln P_n(\beta_c)
\]

\[
\theta^{s+1} = \arg \max_{\theta} \sum_{n=1}^{N} \sum_{c=1}^{C} \eta_{cn}(\beta^{s}, \theta^{s}) \ln \pi_{cn}(\theta)
\]

where \( \eta_{cn}(\beta^{s}, \theta^{s}) \) is the posterior probability that agent \( n \) is in class \( c \) evaluated at the

---

3. iclogit is also applicable when the number of scenarios varies across agents, and that of alternatives varies both across agents and over scenarios.
\( s^\text{th} \) estimates:

\[
\eta_{cn}(\beta^s, \theta^s) = \frac{\pi_{cn}(\theta^s) P_n(\beta^s_c)}{\sum_{l=1}^{C} \pi_{ln}(\theta^s) P_n(\beta^s_l)}
\]  

(5)

The updating procedure can be implemented easily in Stata, exploiting `clogit` and `fmlogit` routines as follows. \( \beta^{s+1} \) is computed by estimating a conditional logit model (\texttt{clogit}) \( C \) times, each time using \( \eta_{cn}(\beta^s, \theta^s) \) for a particular \( c \) to weight observations on each \( n \). \( \theta^{s+1} \) is obtained by estimating a fractional multinomial logit model (\texttt{fmlogit}) which takes \( \eta_{1n}(\beta^s, \theta^s), \eta_{2n}(\beta^s, \theta^s), \cdots, \eta_{Cn}(\beta^s, \theta^s) \) as dependent variables. When \( z_n \) only includes the constant term so that each class share is the same for all agents, ie \( \pi_{cn}(\theta) = \pi_c(\theta) \), each class share can be directly updated using the following analytical solution:

\[
\pi_c(\theta^{s+1}) = \frac{\sum_{n=1}^{N} \eta_{cn}(\beta^s, \theta^s)}{\sum_{l=1}^{C} \sum_{n=1}^{N} \eta_{ln}(\beta^s, \theta^s)}
\]  

(6)

without estimating the fractional multinomial logit model.

With a suitable selection of starting values, the updating procedure can be repeated until changes in the estimates and/or improvement in the log likelihood between iterations are small enough.

An often highlighted feature of LCL is its ability to accommodate unobserved interpersonal taste variation without restricting the shape of the underlying taste distribution. Hess et al. (2011) have recently emphasized that LCL also provides convenient means to account for observed interpersonal heterogeneity in correlations among tastes for different attributes. For example, let \( \beta_q \) and \( \beta_h \) denote taste coefficients on the \( q^\text{th} \) and \( h^\text{th} \) attributes respectively. Each coefficient may take one of \( C \) distinct values, and is a random parameter from the researcher’s perspective. Their covariance is given by:

\[
cov_n(\beta_q, \beta_h) = \sum_{c=1}^{C} \pi_{cn}(\theta) \beta_{c,q} \beta_{c,h} - \left( \sum_{c=1}^{C} \pi_{cn}(\theta) \beta_{c,q} \right) \left( \sum_{c=1}^{C} \pi_{cn}(\theta) \beta_{c,h} \right)
\]  

(7)

where \( \beta_{c,q} \) is the value of \( \beta_q \) when agent \( n \) is in class \( c \), and \( \beta_{c,h} \) is defined similarly. As long as \( z_n \) in equation 2 includes a non-constant variable, this covariance will vary across agents with different observed characteristics through the variation in \( \pi_{cn}(\theta) \).

3 The \texttt{lclogit} command

\texttt{lclogit} is a Stata module which implements the EM iterative scheme outlined in the previous section. This module generalizes Pacifico (2012)’s step-by-step procedure and

---

4. \texttt{fmlogit} is a user-written program. See footnote 3 for a further description.
Latent class logit model

introduces an improved internal loop along with other important functionalities. The overall effect is to make the estimation process more convenient, significantly faster and more stable numerically.

To give a few examples, the internal code of lclogit executes fewer algebraic operations per iteration to update the estimates; uses the standard generate command to perform tasks which were previously executed with slightly slower egen functions; and works with log probabilities instead of probabilities when possible. All these changes substantially reduce the estimation run time, especially in the presence of a large number of parameters and/or observations. Taking the 8-class model estimated by Pacifico (2012) for example, lclogit produces the same results as the step-by-step procedure while using less than a half of the latter’s run time.

The data setup for lclogit is identical to that required by clogit.

The generic syntax for lclogit is:

```
lclogit depvar [ varlist ] [ if ][ in ] , group(varname) id(varname) nclasses(#) [ options ]
```

The options for lclogit are:

- **group(varname)** is required and specifies a numeric identifier variable for the choice scenarios.

- **id(varname)** is required and specifies a numeric identifier variable for the choice makers or agents. When only one choice scenario is available for each agent, users may specify the same variable for both `group()` and `id()`.

- **nclasses(#)** is required and specifies the number of latent classes. A minimum of 2 latent classes is required.

- **membership(varlist)** specifies independent variables that enter the fractional multinomial logit model of class membership, i.e. the variables included in the vector \( z_n \) of equation 2. These variables must be constant within the same agent as identified by `id()` 5. When this option is not specified, the class shares are updated algebraically following equation 6.

- **convergence(#)** specifies the tolerance for the log likelihood. When the proportional increase in the log likelihood over the last five iterations is less than the specified criterion, lclogit declares convergence. The default is 0.00001.

---

5. Pacifico (2012) specified a ml program with the method lf to estimate the class membership model. lclogit uses another user-written program from Maarten L. Buis - fmlogit - which performs the same estimation with the significantly faster and more accurate d2 method. lclogit is downloaded with a modified version of the prediction command of fmlogit - fmlogit.pr - since we had to modify this command to obtain double-precision class shares.
• **iterate(\#)** specifies the maximum number of iterations. If convergence is not achieved after the selected number of iterations, `lclogit` stops the recursion and notes this fact before displaying the estimation results. The default is 150.

• **seed(\#)** sets the seed for pseudo uniform random numbers. The default is `c(seed)`.

The starting values for taste parameters are obtained by splitting the sample into `nclasses()` different subsamples and estimating a `clogit` model for each of them. During this process, a pseudo uniform random number is generated for each agent to assign the agent into a particular subsample. As for the starting values for the class shares, the module assumes equal shares, i.e. $1/nclasses()$.

• **constraints(Class#1 numlist: Class#2 numlist: ..)** specifies the constraints that are imposed on the taste parameters of the designated classes, i.e. $\beta_c$ in equation 1. For instance, suppose that $x_1$ and $x_2$ are alternative-specific characteristics included in `indepvars` for `lclogit` and the user wishes to restrict the coefficient on $x_1$ to zero for Class1 and Class4, and the coefficient on $x_2$ to 2 for Class4. Then, the relevant series of commands would look like:

```
. constraint 1 x1 = 0
. constraint 2 x2 = 2
. lclogit depvar indepvars, gr() id() ncl() constraints(Class1 1: Class4 1 2)
```

• **nolog** suppresses the display of the iteration log.

4 Post-estimation command: `lclogitpr`

`lclogitpr` predicts the probabilities of choosing each alternative in a choice situation (choice probabilities hereafter), the class shares or prior probabilities of class membership, and the posterior probabilities of class membership. The predicted probabilities are stored in a variable named `stubname#` where `#` refers to the relevant class number; the only exception is the unconditional choice probability, as it is stored in a variable named `stubname`. The syntax for `lclogitpr` is:

```
lclogitpr stubname [if][in] , [options]
```

The options for `lclogitpr` are:

• **class(numlist)** specifies the classes for which the probabilities are going to be predicted. The default setting assumes all classes.

---

6. More specifically, the unit interval is divided into `nclasses()` equal parts and if the agent’s pseudo random draw is in the $c^{th}$ part, the agent is allocated to the subsample whose `clogit` results serve as the initial estimates of Class $c$’s taste parameters. Note that `lclogit` is identical to `asmprobit` in that the current seed as at the beginning of the command’s execution is restored once all necessary pseudo random draws have been made.
Latent class logit model

- pr0 predicts the unconditional choice probability, which equals the average of class-specific choice probabilities weighted by the corresponding class shares. That is, \( \sum_{c=1}^{C} \pi_{cn}(\theta) \left[ \exp(\beta_c x_{njt}) / \left( \sum_{k=1}^{J} \exp(\beta_k x_{nkt}) \right) \right] \) in the context of Section 2.

- pr predicts the unconditional choice probability and the choice probabilities conditional on being in particular classes; \( \exp(\beta_c x_{njt}) / \left( \sum_{k=1}^{J} \exp(\beta_k x_{nkt}) \right) \) in equation 1 corresponds to the choice probability conditional on being in class \( c \). This is the default option when no other option is specified.

- up predicts the class shares or prior probabilities that the agent is in particular classes. They correspond to the class shares predicted by using the class membership model parameter estimates; see equation 2 in Section 2.

- cp predicts the posterior probabilities that the agent is in particular classes taking into account her sequence of choices. They are computed by evaluating equation 5 at the final estimates for each \( c = 1, 2, \ldots, C \).

5 Post-estimation command: lclogitcov

lclogitcov predicts the implied variances and covariances of taste parameters by evaluating equation 7 at the active lclogit estimates. They could be a useful tool for studying the underlying taste patterns; see Hess et al. (2011) for a related application.

The generic syntax for lclogitcov is:

```
lclogitcov [ varlist ] [ if ] [ in ], [ options ]
```

The default is to store the predicted variances in a set of hard-coded variables named var_1, var_2, ... where var_q is the predicted variance of the coefficient on the \( q^{th} \) variable listed in varlist, and the predicted covariances in cov_12, cov_13, ..., cov_23, ... where cov_qh is the predicted covariance between the coefficients on the \( q^{th} \) variable and the \( h^{th} \) variable in varlist.

The averages of these variance and covariances over agents - as identified by the required option id() of lclogit - in the prediction sample are reported as a covariance matrix at the end of lclogitcov’s execution.

The options for lclogitcov are:

- nokeep drops the predicted variances and covariances from the data set at the end of the command’s execution. The average covariance matrix is still displayed.

- varname(stubname) requests the predicted variances to be stored as stubname1, stubname2,...

- covname(stubname) requests the predicted covariances to be stored as stubname12, stubname13,...
matrix(name) stores the reported average covariance matrix in a Stata matrix called name.

6 Post-estimation command: lclogitml

lclogitml is a wrapper for gllamm (Rabe-Hesketh et al., 2002) which uses the do method to fit generalised linear latent class and mixed models including LCL via the Newton-Rhapson (NR) algorithm for likelihood maximization. This post-estimation command passes active lcatit specification and estimates to gllamm, and its primary usage mainly depends on how iterate() option is specified; see below for details.

The default setting relabels and transforms the ereturn results of gllamm in accordance with those of lclogit, before reporting and posting them. Users can exploit lclogitrpr and lclogitcov, as well as Stata’s usual post-estimation commands requiring the asymptotic covariance matrix such as nlcom. When switch is specified, the original ereturn results of gllamm are reported and posted; users gain access to gllamm’s post-estimation commands, but lose access to lclogitrpr and lclogitcov.

lclogitml can also be used as its own post-estimation command, for example to pass the currently active lclogitml results to gllamm for further NR iterations.

The generic syntax for lclogitml is:

```
lclogitml [if][in] , [options]
```

The options for lclogitml are:

- **iterate(#)** specifies the maximum number of NR iterations for gllamm’s likelihood maximization process. The default is 0 in which case the likelihood function and its derivatives are evaluated at the current lclogit estimates; this allows obtaining standard errors associated with the current estimates without bootstrapping.

With a non-zero argument, this option can implement a hybrid estimation strategy similar to Bhat’s (1997). He executes a relatively small number of EM iterations to obtain intermediate estimates, and use them as starting values for direct likelihood maximization via a quasi-Newton algorithm until convergence, because the EM algorithm tends to slow down near a local maximum.

Specifying a non-zero argument for this option can also be a useful tool for checking whether lclogit has declared convergence prematurely, for instance because convergence() has not been set stringently enough for an application at hand.

- **level(#)** sets confidence level; the default is 95.

- **nopo** restores the currently active ereturn results at the end of the command’s execution.

**Note:** gllamm can be downloaded by entering ssc install gllamm into the command window.


Latent class logit model

- switch displays and posts the original gllamm estimation results, without relabeling and transforming them in accordance with the lclogit output.

- compatible_gllamm_options refer to gllamm’s estimation options which are compatible with the LCL model specification. See gllamm’s own help menu for more information.

7 Application

We illustrate the use of lclogit and its companion post-estimation commands by expanding upon the example [Pacifico (2012)] uses to demonstrate his step-by-step procedure for estimating LCL in Stata. This example analyzes the stated preference data on household’s electricity supplier choice accompanying [Hole (2007)]’s mixlogit module, which in turn are a subset of data used in [Huber and Train (2001)]. There are 100 customers who face up to 12 different choice occasions, each of them consisting of a single choice among 4 suppliers with the following characteristics:

- 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)

The dummy variable y collects the stated choice in each choice occasion whilst the numeric variables pid and gid identify customers and choice occasions respectively. To illustrate the use of membership() option, we generate a pseudo random regressor _x1 which mimics a demographic variable. The data are organized as follows:

```
use http://fmwww.bc.edu/repec/bocode/t/traindata.dta, clear
set seed 1234567890
bysort pid: egen _x1=sum(round(rnormal(0.5),1))
list in 1/12, sepby(gid)
```

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>price</th>
<th>contract</th>
<th>local</th>
<th>wknown</th>
<th>tod</th>
<th>seasonal</th>
<th>gid</th>
<th>pid</th>
<th>_x1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>2.</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>3.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>4.</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>5.</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>6.</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>26</td>
</tr>
</tbody>
</table>
In empirical applications, it is common to choose the optimal number of latent classes by examining information criteria such as BIC and CAIC. The next lines show how to estimate 9 LCL specifications repeatedly and obtain the related information criteria:

```stata
forvalues c = 2/10 {
    lclogit y price contract local wknown tod seasonal, group(gid) id(pid)
    nclasses(`c') membership(_x1) seed(1234567890)
    matrix b = e(b)
    matrix ic = nullmat(ic)`e(nclasses)'`e(ll)'`e(ncols(b))'`e(caic)'`e(bic)'
}
```

<table>
<thead>
<tr>
<th>Classes</th>
<th>LLF</th>
<th>Nparam</th>
<th>CAIC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-1211.232</td>
<td>14</td>
<td>2500.935</td>
<td>2486.935</td>
</tr>
<tr>
<td>3</td>
<td>-1117.521</td>
<td>22</td>
<td>2358.356</td>
<td>2336.356</td>
</tr>
<tr>
<td>4</td>
<td>-1084.559</td>
<td>30</td>
<td>2337.273</td>
<td>2307.273</td>
</tr>
<tr>
<td>5</td>
<td>-1039.771</td>
<td>38</td>
<td>2292.538</td>
<td>2254.538</td>
</tr>
<tr>
<td>6</td>
<td>-1027.633</td>
<td>46</td>
<td>2313.103</td>
<td>2267.103</td>
</tr>
<tr>
<td>7</td>
<td>-999.9628</td>
<td>54</td>
<td>2302.605</td>
<td>2248.605</td>
</tr>
<tr>
<td>8</td>
<td>-987.7199</td>
<td>62</td>
<td>2322.96</td>
<td>2260.96</td>
</tr>
<tr>
<td>9</td>
<td>-985.1933</td>
<td>70</td>
<td>2362.748</td>
<td>2292.748</td>
</tr>
<tr>
<td>10</td>
<td>-966.3487</td>
<td>78</td>
<td>2369.901</td>
<td>2291.901</td>
</tr>
</tbody>
</table>
```

CAIC and BIC are minimized with 5 and 7 classes respectively. In the remainder of this section, our analysis focuses on the 5-class specification to economize on space.

`lclogit` reports the estimation results as follows:

```stata
lclogit y price contract local wknown tod seasonal, group(gid) id(pid) nclass
```

```
Variable | Class1 | Class2 | Class3 | Class4 | Class5
---------|--------|--------|--------|--------|--------
```

8. `lclogit` saves three information criteria in its `ereturn list`: AIC, BIC and CAIC. AIC equals $-2\ln L + 2m$, where $\ln L$ is the maximized sample log likelihood and $m$ is the total number of estimated model parameters. BIC and CAIC penalize models with extra parameters more heavily, by using penalty functions increasing in the number of choice makers, $N$: $BIC = -2\ln L + m\ln N$ and $CAIC = -2\ln L + m(1 + \ln N)$. 

D. Pacifico and H. Yoo
**Latent class logit model**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Class1</th>
<th>Class2</th>
<th>Class3</th>
<th>Class4</th>
<th>Class5</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>-0.902</td>
<td>-0.325</td>
<td>-0.763</td>
<td>-1.526</td>
<td>-0.520</td>
</tr>
<tr>
<td>contract</td>
<td>-0.470</td>
<td>0.011</td>
<td>-0.533</td>
<td>-0.405</td>
<td>-0.016</td>
</tr>
<tr>
<td>local</td>
<td>0.424</td>
<td>3.120</td>
<td>0.527</td>
<td>0.743</td>
<td>3.921</td>
</tr>
<tr>
<td>unknown</td>
<td>0.437</td>
<td>2.258</td>
<td>0.325</td>
<td>1.031</td>
<td>3.063</td>
</tr>
</tbody>
</table>

Class Share

| Class Share | 0.113 | 0.282 | 0.162 | 0.243 | 0.200 |

Class membership model parameters: Class5 = Reference class

<table>
<thead>
<tr>
<th>Variable</th>
<th>Class1</th>
<th>Class2</th>
<th>Class3</th>
<th>Class4</th>
<th>Class5</th>
</tr>
</thead>
<tbody>
<tr>
<td>_x1</td>
<td>0.045</td>
<td>0.040</td>
<td>0.047</td>
<td>0.048</td>
<td>0.000</td>
</tr>
<tr>
<td>_cons</td>
<td>-1.562</td>
<td>-0.544</td>
<td>-1.260</td>
<td>-0.878</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Note: Model estimated via EM algorithm

It is worth noting that the reported class shares are the average shares over agents, because the class shares vary across agents when the membership() option is included in the syntax. If needed, agent-specific class shares can be easily computed by using the post-estimation command `lclogitpr` with the `up` option.

In order to obtain a quantitative measure of how well the model does in differentiating several classes of preferences, we use `lclogitpr` to compute the average (over respondents) of the highest posterior probability of class membership:

```stata
. bys `e(id)´: gen first = _n==1
. lclogitpr cp, cp
. egen double cpmax = rowmax(cp1-cp5)
. sum cpmax if first, sep(0)
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpmax</td>
<td>100</td>
<td>.9596674</td>
<td>.0860159</td>
<td>.5899004</td>
<td>1</td>
</tr>
</tbody>
</table>

As it can be seen, the mean highest posterior probability is about 0.96, meaning that the model does very well in distinguishing among different underlying taste patterns for the observed choice behavior.

We next examine the model’s ability to make in-sample predictions of the actual choice outcomes. For this purpose, we first classify a respondent as a member of class $c$ if class $c$ gives her highest posterior membership probability. Then, for each subsample of such respondents, we predict the unconditional probability of actual choice and the probability of actual choice conditional on being in class $c$:

```stata
. lclogitpr pr, pr
. gen byte class = .
(4780 missing values generated)
. forvalues c = 1/`e(nclasses)´ {
```

9. A dummy variable which equals 1 for the first observation on each respondent is generated because not every agent faces the same number of choice situations in this specific experiment.
In general, the average unconditional choice probability is much higher than 0.25 which is what a naive model would predict given that there are 4 alternatives per choice occasion. The average conditional probability is even better and higher than 0.5 in all but one classes. Once again we see that the model describes the observed choice behavior very well.

When taste parameters are modeled as draws from a normal distribution, the estimated preference heterogeneity is described by their mean and covariances. The same summary statistics can be easily computed for LCL by combining class shares and taste parameters; see Hess et al. (2011) for a detailed discussion. \texttt{lclogit} saves these statistics as part of its \texttt{ereturn list}:

\begin{verbatim}
. matrix list e(PB)
   e(PB)[1,6]
   Average Average Average Average Average Average
   price contract local unknown tod seasonal
   Coefficients  -.79129  -.23756  1.9795  1.6029  -7.62728  -7.64949
. matrix list e(CB)
   symmetric e(CB)[6,6]
   price contract local unknown tod seasonal
   price  .20833629
   contract .07611239  .05436665
   local  .48852574  .32683725  2.1078043
   unknown .27611961  .22587673  1.4558029  1.045789
   tod  2.2090348  .65296465  4.0426714  1.9610973  25.12504
   seasonal  1.9728148  .6573999  3.8801716  2.0070985  21.845013  20.189302
\end{verbatim}

Since we estimated a model with the \texttt{membership()} option, the class shares (hence the covariances; see equation \[\text{7}\]) now vary across respondents and the matrix \texttt{e(CB)} above is an average covariance matrix. In this case, the post-estimation command \texttt{lclogitcov} can be very useful for studying variation in taste correlation patterns within and across different demographic groups. To illustrate this point, we compute the covariances of the coefficients on \texttt{price} and \texttt{contract}, and then summarize the results for two groups.
defined by whether \_x1 is greater or less than 20:

\[
\text{quietly lclogitcov price contract}
\]

\[
\text{sum var\_1 cov\_12 var\_2 if \_x1 >20 & first}
\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_1</td>
<td>62</td>
<td>0.2151655</td>
<td>0.0061303</td>
<td>0.2065048</td>
<td>0.2301424</td>
</tr>
<tr>
<td>cov_12</td>
<td>62</td>
<td>0.0765989</td>
<td>0.000348</td>
<td>0.0760533</td>
<td>0.0773176</td>
</tr>
<tr>
<td>var_2</td>
<td>62</td>
<td>0.0545157</td>
<td>0.000987</td>
<td>0.0543549</td>
<td>0.0547015</td>
</tr>
</tbody>
</table>

\[
\text{sum var\_1 cov\_12 var\_2 if \_x1 <=20 & first}
\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_1</td>
<td>38</td>
<td>0.1971939</td>
<td>0.0053252</td>
<td>0.1841499</td>
<td>0.2050795</td>
</tr>
<tr>
<td>cov_12</td>
<td>38</td>
<td>0.0753185</td>
<td>0.0004483</td>
<td>0.0741831</td>
<td>0.0759492</td>
</tr>
<tr>
<td>var_2</td>
<td>38</td>
<td>0.0541235</td>
<td>0.0001431</td>
<td>0.0537598</td>
<td>0.0543226</td>
</tr>
</tbody>
</table>

Standard errors associated with any results provided by \texttt{lclogit} can be obtained via \texttt{bootstrap}. However, the bootstrapped standard errors of class-specific results are much less reliable than those of averaged results because the class labeling may vary arbitrarily across bootstrapped samples; see \cite{Train2008} for a detailed discussion.

Users interested in class-specific inferences may consider passing the \texttt{lclogit} results to user-written \texttt{ml} programs like \texttt{gllamm} (\cite{Rabe-Hesketh2002}), to take advantage of the EM algorithm and obtain conventional standard errors at the same time. \texttt{lclogitml} simplifies this process.

\[
\text{. lclogitml, iter(5) gllamm- is initializing. This process may take a few minutes.}
\]

\[
\text{Iteration 0: log likelihood = -1039.7709 (not concave)}
\]

\[
\text{Iteration 1: log likelihood = -1039.7709}
\]

\[
\text{Iteration 2: log likelihood = -1039.7706}
\]

Latent class model with 5 latent classes

| y        | Coef.  | Std. Err. | z     | P>|z| | [95% Conf. Interval] |
|----------|--------|-----------|-------|------|----------------------|
| choice1  |        |           |       |      |                      |
| price    | -.9023068 | .2012345 | -4.48 | 0.000 | -.1.296719 -.5078945 |
| contract | -.4698861 | .089774  | -5.23 | 0.000 | -.64584 -.2939322    |
| local    | .4241342 | .3579407 | 1.18  | 0.236 | -.2.774167 1.125685 |
| wknown   | .4370318 | .2864782 | 1.53  | 0.127 | -.1.124452 0.985188  |
| tod      | -.8.22232 | 1.384777 | -5.31 | 0.000 | -.11.528346 -.518126 |
| seasonal | -.6.35626 | 1.359516 | -4.05 | 0.000 | -.9.43082 -.3.278431 |

| choice2  |        |           |       |      |                      |
| price    | -.3249095 | .1090047 | -2.98 | 0.003 | -.5.385548 -.1.112642 |
| contract | .0108523  | .0384404 | 0.28  | 0.778 | .0644894 .0861941   |
| local    | 3.122255  | .2842558 | 10.98 | 0.000 | 2.565124 3.679587   |
| wknown   | 2.258772  | .2553446 | 8.85  | 0.000 | 1.758306 2.759238   |
| tod      | -.2.15772 | .8906932 | -2.42 | 0.015 | -.3.903453 -.4.119997 |
| seasonal | -.2.470511| .894278  | -2.76 | 0.006 | -.4.223264 -.7.177582 |

| choice3  |        |           |       |      |                      |
| price    | -.7629762 | .1415072 | -5.39 | 0.000 | -.1.040325 -.4.856271 |
| contract | -.5331056 | .0739354 | -7.21 | 0.000 | -.6.780162 -.3.881949 |
The estimated choice model or taste parameters, $\beta_c$, and class membership model parameters, $\theta_c$, are grouped under equations choice$_c$ and share$_c$ respectively. lclogitml relabels and transforms the original gllamm estimation results in accordance with the lclogit’s ereturn list (see Section 6), facilitating interpretation of the new output table. The active lclogitml coefficient estimates can also be displayed in the standard lclogit output format, by entering lclogit into the command window without any additional statement.

Note that the log likelihood increases slightly after 3 iterations, though the parameter estimates remain almost the same. This may happen since lclogit uses only the relative change in the log likelihood as convergence criterion. gllamm works with the standard ml command with a d0 evaluator, which declares convergence in a more stringent manner: specifically, when the relative changes in both the scaled gradient and either the log
likelihood or the parameter vector are smaller than a given tolerance level.\footnote{The benefit of pre-use of \texttt{lclogit} cannot be overstated. Since \texttt{gllamm} uses the $d0$ evaluator and the LCL log likelihood is not amenable to direct maximization, each iteration tends to last for long and finding initial values which lead to convergence often involves a laborious search. \texttt{lclogit} exploits the EM algorithm that guarantees convergence to a local maximum in theory, and takes the estimates to a local maximum or its close neighborhood in a relatively fast way in practice.}

When \texttt{lclogit} is used in a final production run, it is advisable to specify more stringent convergence() than the default, and experiment with alternative starting values by changing \texttt{seed()}. \cite{Train2008} contains references highlighting the importance of these issues for applications exploiting EM algorithms.

\section{Acknowledgments}

We thank an anonymous referee for useful comments and suggestions. Hong il Yoo’s work was supported under Australian Research Council’s Discovery Projects funding scheme (project number: DP0881205).

\section{References}


About the author

Daniele Pacifico works with the Italian Department of the Treasury (Rome, Italy). Hong il Yoo is a PhD student at the School of Economics, the University of New South Wales (Sydney, Australia).