Iclogit: a Stata module for estimating a mixed logit model with discrete mixing distribution via the Expectation-Maximization algorithm

Daniele Pacifico, Hong il Yoo
Working Papers

The working paper series promotes the dissemination of economic research produced in the Department of the Treasury (DT) of the Italian Ministry of Economy and Finance (MEF) or presented by external economists on the occasion of seminars organised by MEF on topics of institutional interest to the DT, with the aim of stimulating comments and suggestions.

The views expressed in the working papers are those of the authors and do not necessarily reflect those of the MEF and the DT.

Copyright:

2012, Daniele Pacifico, Hong il Yoo.

The document can be downloaded from the Website www.dt.tesoro.it and freely used, providing that its source and author(s) are quoted.
Iclogit: a Stata module for estimating a mixed logit model with discrete mixing distribution via the Expectation-Maximization algorithm

Daniele Pacifico ('), Hong il Yoo ('"")

Abstract

This paper describes Iclogit, a Stata module to fit latent class logit models through the Expectation-Maximization algorithm. The stability of this estimation method allows overcoming some of the computational difficulties that normally arise when fitting such models with many latent classes. This, in turn, permits users to estimate nonparametrically the mixing distribution of the random coefficients because the more the mass points of the latent class model, the better the approximation of the unknown joint density of the random coefficients.

Keywords: st0001, Iclogit, latent class model, EM algorithm, mixed logit.

1 Introduction

Mixed logit or random parameter logit models are 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. Specifically, Bhat (1997) applies mixed logit with discrete mixing distributions to operationalize an endogenous segmentation model which jointly describe the probabilistic assignment of individuals to each of several market segments, and the probabilistic choices made by individuals following assignment to a particular segment. Other authors such as Shen (2009) or Hess et al. (2011), instead, showed how discrete mixture logit can be used to approximate the true parameter distribution more flexibly and at lower computational costs, in comparison with more common mixed logit models that require a parametric specification of the mixing distribution.

Pacifico (2012) implements the Expectation-Maximization (EM) algorithm for estimating a discrete mixture logit model, also known as latent class logit, in Stata. As discussed in Train (2008) and Bhat (1997), 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 a large number of
latent classes is specified.\(^1\) 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).\(^2\) Consider \(N\) agents facing \(J\) alternatives on each of \(T\) choice occasions. Let \(y_{njt}\) denote a binary variable which equals 1 if agent \(n\) chooses alternative \(j\) on occasion \(t\) 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 preference parameters, \(\mathbf{\beta} = (\beta_1, \beta_2, ..., \beta_C)\). If agent \(n\) is in class \(c\), the probability of observing her sequence of choices is the product of conditional logit formulas:

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

Since the class membership status is unknown, the researcher specifies the sample log likelihood using the unconditional likelihood, which equals the weighted average of these products over classes. The weight for class \(c\), \(\pi_{cn}(\mathbf{\theta})\), is the population share of that class and is specified as a multinomial logit model:

\[
\pi_{cn}(\mathbf{\theta}) = \frac{\exp(\mathbf{\theta}_c z_n)}{1 + \sum_{c=1}^{C} \exp(\mathbf{\theta}_c z_n)} \tag{2}\]

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

Bhat (1997) and Train (2008) show that the maximum likelihood estimator of \(\mathbf{\beta}\) and \(\mathbf{\theta}\) can be obtained by a typical EM recursion for maximizing a missing data log likelihood. Let us use superscript \(s\) to denote the estimates obtained at the \(s^{th}\) iteration of this recursion. Then, at iteration \(s + 1\) the estimates are updated as:

\[
\begin{align*}
\mathbf{\beta}^{s+1} &= \arg\max_{\mathbf{\beta}} \sum_{n=1}^{N} \sum_{c=1}^{C} \eta_{cn}(\mathbf{\beta}^s, \mathbf{\theta}^s) \ln P_n(\mathbf{\beta}_c) \\
\mathbf{\theta}^{s+1} &= \arg\max_{\mathbf{\theta}} \sum_{n=1}^{N} \sum_{c=1}^{C} \eta_{cn}(\mathbf{\beta}^s, \mathbf{\theta}^s) \ln \pi_{cn}(\mathbf{\theta})
\end{align*} \tag{3}\]

\(^1\) Clearly, it is always advisable to run the estimation using different starting values, so as to check for whether the local maximum is also global.

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

\[
\eta_{cn}(\beta^c, \theta^s) = \frac{\pi_{cn}(\theta^s) P_n(\beta^c_n)}{\sum_{l=1}^{C} \pi_{ln}(\theta^s) P_n(\beta^l_n)} \tag{4}
\]

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.

When \( z_n \) only includes the constant term so that each class share is the same for all agents, ie \( \pi_{cn}(\theta) = \pi_c(\theta) \), the procedure in equation 3 can be simplified because each class share can be directly updated using the following analytical solution:

\[
\pi_c(\theta^{s+1}) = \frac{\sum_{n=1}^{N} \eta_{cn}(\beta^s, \theta^s)}{\sum_{l=1}^{C} \sum_{n=1}^{N} \eta_{ln}(\beta^s, \theta^s)} \tag{5}
\]

3 The \texttt{lclogit} command

\texttt{lclogit} is a Stata module which implements the EM recursion outlined in the previous section. This module generalizes Pacífico (2012)'s step-by-step procedure and 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 \texttt{lclogit} executes fewer algebraic operations per iteration to update the estimates; uses the standard \texttt{generate} command to perform tasks which were previously executed with slightly slower \texttt{egen} functions; and works with log probabilities instead of probabilities when possible. All these changes drastically reduce computing time of the \texttt{lclogit} module, especially in the presence of a large number of parameters and/or a large sample size. Taking the 8-class model discussed in Pacífico (2012) for example, \texttt{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 \texttt{lclogit} is identical to that required by \texttt{clogit}.

The generic syntax for \texttt{lclogit} is:

\texttt{lclogit depvar [vartlist] [if][in] , group(varname) id(varname) nclasses(#) [options]}

The options for \texttt{lclogit} are:

* \texttt{group(varname)} is required and specifies a numeric identifier variable for the choice situations.
• \textit{id(varname)} is required and specifies a numeric identifier variable for the choice makers or agents. For cross section data, users may specify the same variable for both \textit{group()} and \textit{id()}.

• \textit{nclasses(\#)} is required and specifies the number of latent classes. A minimum of 2 latent classes is required.

• \textit{membership(varlist)} specifies independent variables that enter the 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 \textit{id()}.\footnote{Pacifico (2012) specified a \texttt{al} program with the method \texttt{lf} to estimate the class membership model. \texttt{1clogit} uses another user-written program from Maarten L. Buis - \texttt{fmlogit} - which performs the same estimation with the significantly faster and more accurate \texttt{d2} method. \texttt{1clogit} is downloaded with a modified version of the prediction command of \texttt{fmlogit} - \texttt{fmlogit.pr} - since we had to modify this command to obtain double-precision class shares.}

• \textit{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, \texttt{1clogit} declares convergence. The default is 0.00001.

• \textit{iterate(\#)} specifies the maximum number of iterations. If convergence is not achieved after the selected number of iterations, \texttt{1clogit} stops the recursion and notes this fact before displaying the estimation results. The default is 150.

• \textit{seed(\#)} sets the seed for pseudo uniform random numbers. The default is \texttt{c(seed)}.

The starting values for taste parameters are obtained by splitting the sample into \textit{nclasses()} different subsamples and estimating a \texttt{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.\footnote{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 \texttt{1clogit} is identical to \texttt{asprobit} 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.}

As for the starting values for the class shares, the module assumes equal shares, i.e. 1/nclasses().

• \textit{constraints(Class#1 numlist: Class#2 numlist: ...)} specifies the constraints that are imposed on the taste parameters of the designated classes, i.e. $\beta$, in equation 1. For instance, suppose that $x_1$ and $x_2$ are alternative-specific characteristics included in \textit{indepsvars} for \texttt{1clogit} 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:

\begin{verbatim}
   . constraint 1 x1 = 0
   . constraint 2 x2 = 2
   . 1clogit depvar indepsvars, gr() id() ncl() constraints(Class1 1: Class4 1 2)
\end{verbatim}

• \texttt{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(namelist)` specifies the classes for which the probabilities are going to be predicted. The default setting assumes all classes.
- `pr0` predicts the unconditional choice probability, which equals the average of class-specific choice probabilities weighted by the corresponding class shares.
- `pr` predicts the unconditional choice probability and the choice probabilities conditional on being in particular classes. 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.

5 Post-estimation command: lclogitcov

lclogitcov predicts the implied variances and covariances of taste parameters using lclogit estimates; see Hess et al. (2011) for details. They could be a useful tool for studying the underlying taste patterns.

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.k` is the predicted variance of the coefficient on the `kth` variable listed in `varlist`, and the predicted covariances in `cov.12, cov.13, ..., cov.23, ...` where `cov.kj` is the predicted covariance between the coefficients on the `kth` variable and the `jth` variable in `varlist`.

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

The options for \texttt{lcl} are:

- \texttt{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.
- \texttt{varname(stubname)} requests the predicted variances to be stored as \texttt{stubname1}, \texttt{stubname2},...
- \texttt{covname(stubname)} requests the predicted covariances to be stored as \texttt{stubname12}, \texttt{stubname13},...
- \texttt{matrix(name)} stores the reported average covariance matrix in a Stata matrix called \texttt{name}.

6 Application

We illustrate the use of \texttt{lcl} 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 \texttt{mixlogit} module. 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 (\texttt{price})
- The length of contract that the supplier offered, expressed in years (\texttt{contract})
- Whether the supplier is a local company (\texttt{local})
- Whether the supplier is a well-known company (\texttt{known})
- Whether the supplier offers a time-of-day rate instead of a fixed rate (\texttt{tod})
- Whether the supplier offers a seasonal rate instead of a fixed rate (\texttt{seasonal})

The dummy variable \( y \) collects the stated choice in each choice occasion whilst the numeric variables \( \texttt{pid} \) and \( \texttt{gid} \) identify customers and choice occasions respectively. To illustrate the use of \texttt{membership()} option, we generate a pseudo random regressor \( \texttt{x1} \) which mimics a demographic variable. The data are organized as follows:

```stata
. 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)
```
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:

```
di c(current_time)
12:54:16
forvalues c = 2/10 {
    lclg_y 
    x price contract local wknown tod seasonal, group(gid) id(pid)
    > nclasses(c) membership(.xl) seed(1234567890)
    matrix b = e(b)
    matrix ic = nullmat(ic)\"(e(nclasses)\",\"e(11)\",\"-colsof(b)\",\"e(caic)\",\"e(bic)\"
5.)
(output omitted)
    matrix colnames ic = \"Classes\" \"LLF\" \"Uparm\" \"CAIC\" \"BIC\"
    matlist ic, name(columns)
'''
```

<table>
<thead>
<tr>
<th>Classes</th>
<th>LLP</th>
<th>Uparm</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>2358.366</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>2302.665</td>
<td>2248.665</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>-966.3467</td>
<td>78</td>
<td>2369.904</td>
<td>2291.904</td>
</tr>
</tbody>
</table>

As the results show, the CAIC and the BIC are minimized with 5 and 7 classes respectively. Moreover, as it can be seen, the estimation of the 9 LCL took less than 5 minutes on our standard-issue PC with a quad-core CPU and 4GB RAM. If compared with the do-file routine outlined in Pacifico (2012) the increased performance is substantial, with the run time being lower than 74%. This is important, in particular when working with datasets that contain a higher number of choice makers and/or choice
occasions and/or number of parameters, all elements that could significantly increase the estimation time. In the remainder of this section, our analysis focuses on the 5-class specification to economize on space.\(^5\)

```
. lclogit y price contract local known tod seasonal, group(gid) id(pid) nclass > as(5) membership(_xi) seed(1234567890)
Iteration 0: log likelihood = -1313.967
Iteration 1: log likelihood = -1195.5476
(output omitted)
Iteration 22: log likelihood = -1039.7709
Latent class model with 5 latent classes
Choice model parameters and average class shares

<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>known</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 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>_xi</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
```

As can be seen, the LCLOGIT output is organized in columns, each one showing the point estimates for a given latent class. Moreover, when the `membership()` option is included in the syntax an additional box with the same style shows the estimated class-membership parameters.\(^6\)

Here it is worth noting that the output of LCLOGIT is different with respect to other Stata estimation commands. The main reason is that the estimation through standard EM algorithms does not produce directly the standard errors of the coefficients in each class. Train (2008) overviews two approaches for the computation of standard errors and the related t-statistics, namely nonparameteric bootstrap or asymptotic formulas as in Ruud (1991). However, deriving standard errors with asymptotic formulas can be extremely time consuming when the number of parameters is high. Therefore, lclogit is fully integrated with the Stata bootstrap command, which allows obtaining standard errors for both the preference and the class-membership parameters straightforwardly.

---

5. Pacifico (2012) shows that the optimal number of latent classes is 8 when no demographics are included in the class-share probabilities. This result can be easily replicated with lclogit by simply dropping the `membership(_xi)` option from the syntax.

6. In this case, the class shares reported in the last row of the first block represent the average shares over agents, since the class shares are now agent-specific.
In order to obtain a quantitative measure of how well the model does in distinguishing different classes of preferences, we use \texttt{lclogitpr} to compute the average (over respondents) of the highest posterior probability of class membership:

\begin{verbatim}
  . bys `e(id)' : gen first = _n==1
  . lclogitpr cp, cp
  . egen double cmax = rowmax(cp1-cp5)
  . sum cmax if first, sav(0)
\end{verbatim}

\begin{verbatim}
<table>
<thead>
<tr>
<th></th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmax</td>
<td>100</td>
<td>0.959674</td>
<td>0.0860159</td>
<td>0.5899004</td>
<td>1</td>
</tr>
</tbody>
</table>
\end{verbatim}

As it can be seen, the mean highest posterior probability is about 0.96, meaning that the model does very well in capturing 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 \( c \):

\begin{verbatim}
  . lclogitpr pr, pr
  . gen byte class = .
(4760 missing values generated)
  . forvalues c - 1'/e(nclasses)' { 
    quietly replace class = `c' if cmax==cp`c'
  }
  . forvalues c - 1'/e(nclasses)' {
    qui sum pr if class == `c' & y==1
    local n=r(n)
    qui sum pr`c' if class == `c' & y==1
    local a=r(mean)
    qui sum pr`c' if class == `c' & y==1
    local b=r(mean)
    matrix pr = nulsmat(pr) \ `n', `c', `a', `b'
  }
  . matrix colnames pr = "Obs" "Class" "Uncond_Pr" "Cond_PR"
  . matlist pr, name(columns)
\end{verbatim}

\begin{verbatim}
<table>
<thead>
<tr>
<th></th>
<th>Obs</th>
<th>Class</th>
<th>Uncond_Pr</th>
<th>Cond_PR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>129</td>
<td>1</td>
<td>0.3364491</td>
<td>0.5367555</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td>2</td>
<td>0.3344088</td>
<td>0.4505939</td>
</tr>
<tr>
<td></td>
<td>191</td>
<td>3</td>
<td>0.3407353</td>
<td>0.5261553</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>4</td>
<td>0.4562778</td>
<td>0.7557497</td>
</tr>
<tr>
<td></td>
<td>229</td>
<td>5</td>
<td>0.4321717</td>
<td>0.6562177</td>
</tr>
</tbody>
</table>
\end{verbatim}

In general, the average unconditional choice probability is much higher than 0.25 which is what a naive \texttt{clogit} 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

\footnotereferences
7. 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.
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. **lclogit** saves these statistics as part of its `ereturn` list:

```stata
. matrix list e(PB)

  e(PB)[1,6]
     price contract local  known  tod  seasonal
     Coefficients  -.79129  -.23756   1.9795  1.65929  -7.62728  -7.64949

. matrix list e(CB)

  symmetric e(CB)[6,6]
     price contract local  known  tod  seasonal
     price   .20833629         0.05436665
     contract .07611239         .26952742
     local     .49652574         .22598773
     known    .22611961         .14550929
     tod      2.2090348         .65296465
     seasonal 1.9726148         .65573999
```

Since we estimated a model with the `membership()` option, the class shares (hence the covariances) now vary across respondents and the matrix `e(CB)` above is an average covariance matrix. In this case, the post-estimation command `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 `price` and `contract`, and then summarize the results for two groups defined by whether `x1` is greater or less than 20:

```stata
. quietly lclogitcov price contract
. 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>.2151655</td>
<td>.0961303</td>
<td>.2065048</td>
<td>.2301424</td>
</tr>
<tr>
<td>cov_12</td>
<td>62</td>
<td>.0765989</td>
<td>.0004384</td>
<td>.0765033</td>
<td>.0773176</td>
</tr>
<tr>
<td>var_2</td>
<td>62</td>
<td>.0645157</td>
<td>.0000907</td>
<td>.0543549</td>
<td>.0547015</td>
</tr>
</tbody>
</table>

. 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>.1971939</td>
<td>.0053252</td>
<td>.1841499</td>
<td>.2050795</td>
</tr>
<tr>
<td>cov_12</td>
<td>38</td>
<td>.0753185</td>
<td>.0004483</td>
<td>.0741831</td>
<td>.0759494</td>
</tr>
<tr>
<td>var_2</td>
<td>38</td>
<td>.0541235</td>
<td>.0001431</td>
<td>.0537589</td>
<td>.0543226</td>
</tr>
</tbody>
</table>
```

Standard errors associated with any results provided by `lclogit` can be obtained via `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 Train (2008) for a detailed discussion. Users interested in class-specific inferences may consider passing the `lclogit` results to user-written `ml` programs like Sophia Rabe-Hesketh’s `gllamm`, to take advantage of the
EM algorithm and obtain conventional standard errors at the same time. The use of 
`lcllogit's ereturn list may simplify this process. For instance, our current results 
can be passed to `gllamm` by entering:

```
 . foreach v of varlist `{o(indepvars)}` {
   2.   eq `{o}v' : `{o}v'
   3. }
 . eq share : `{o(indepvars2)}`
 . matrix B = `o(B)
 . matrix CMB = `o(CMB)
 . forvalues c = 1/=`o(nclasses')'-1' {
   2.   matrix init = nullmat(init), `B[^c',`c',..., CMB[^c',`c',...]
   3. }
 . matrix init = init, `B[^n(classes)',`c',...]
 . gllamm `{o(depvar)', nocons i(`o(id)') expand(`o(group)', `{o(depvar)'`o) ///
 >   1(mllogit) i(binom) nip(`o(nclasses)') lp(3n) pe(share) from(init) copy ///
 >   nlrf(-wordcount(`o(indepvars)')) eq(`o(indepvars)') allc iter(5)
```

7 Acknowledgments

We are grateful to an anonymous referee for the relevant comments and suggestions on 
how to improve this paper.

8 References

Bhat, C., 1997. An endogenous segmentation mode choice model with an application to 

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

over mixture of logit models, mimeo.

Stata Journal, 7 (3), 388-401.

Stata Journal (forthcoming).

Econometrics 49, 305-341.


Ministry of Economy and Finance
Department of the Treasury
Directorate I: Economic and Financial Analysis

Address:
Via XX Settembre, 97
00187 - Rome

Websites:
www.mef.gov.it
www.dt.tesoro.it

e-mail:
dt.segreteria.direzione1@tesoro.it

Telephone:
+39 06 47614202
+39 06 47614197

Fax:
+39 06 47821886