

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.¹

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).² 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).

2. Further details are available in Bhat (1997) and Train (2008).

in each of T choice scenarios.³ 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, \mathbf{x}_{njt} , and each agent by agent-specific characteristics including a constant, \mathbf{z}_n .

LCL assumes that there are C distinct sets (or classes) of taste parameters, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \dots, \boldsymbol{\beta}_C)$. If agent n is in class c , the probability of observing her sequence of choices is a product of conditional logit formulas:

$$P_n(\boldsymbol{\beta}_c) = \prod_{t=1}^T \prod_{j=1}^J \left(\frac{\exp(\boldsymbol{\beta}_c \mathbf{x}_{njt})}{\sum_{k=1}^J \exp(\boldsymbol{\beta}_c \mathbf{x}_{nkt})} \right)^{y_{njt}} \quad (1)$$

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 , $\pi_{cn}(\boldsymbol{\theta})$, is the population share of that class and usually modeled as fractional multinomial logit:

$$\pi_{cn}(\boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}_c \mathbf{z}_n)}{1 + \sum_{l=1}^{C-1} \exp(\boldsymbol{\theta}_l \mathbf{z}_n)} \quad (2)$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_{C-1})$ are class membership model parameters; note that $\boldsymbol{\theta}_C$ has been normalized to zero for identification.

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

$$\ln L(\boldsymbol{\beta}, \boldsymbol{\theta}) = \sum_{n=1}^N \ln \sum_{c=1}^C \pi_{cn}(\boldsymbol{\theta}) P_n(\boldsymbol{\beta}_c) \quad (3)$$

Bhat (1997) and Train (2008) note numerical difficulties associated with maximizing equation 3 directly. They show that $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ 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:

$$\begin{aligned} \boldsymbol{\beta}^{s+1} &= \operatorname{argmax}_{\boldsymbol{\beta}} \sum_{n=1}^N \sum_{c=1}^C \eta_{cn}(\boldsymbol{\beta}^s, \boldsymbol{\theta}^s) \ln P_n(\boldsymbol{\beta}_c) \\ \boldsymbol{\theta}^{s+1} &= \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^N \sum_{c=1}^C \eta_{cn}(\boldsymbol{\beta}^s, \boldsymbol{\theta}^s) \ln \pi_{cn}(\boldsymbol{\theta}) \end{aligned} \quad (4)$$

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

3. `lclogit` is also applicable when the number of scenarios varies across agents, and that of alternatives varies both across agents and over scenarios.

s^{th} estimates:

$$\eta_{cn}(\beta^s, \theta^s) = \frac{\pi_{cn}(\theta^s) P_n(\beta_c^s)}{\sum_{l=1}^C \pi_{ln}(\theta^s) P_n(\beta_l^s)} \quad (5)$$

The updating procedure can be implemented easily in Stata, exploiting `clogit` and `fmlogit` routines as follows.⁴ β^{s+1} is computed by estimating a conditional logit model (`clogit`) C times, each time using $\eta_{cn}(\beta^s, \theta^s)$ for a particular c to weight observations on each n . θ^{s+1} is obtained by estimating a fractional multinomial logit model (`fmlogit`) which takes $\eta_{1n}(\beta^s, \theta^s), \eta_{2n}(\beta^s, \theta^s), \dots, \eta_{Cn}(\beta^s, \theta^s)$ 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:

$$\pi_c(\theta^{s+1}) = \frac{\sum_{n=1}^N \eta_{cn}(\beta^s, \theta^s)}{\sum_{l=1}^C \sum_{n=1}^N \eta_{ln}(\beta^s, \theta^s)} \quad (6)$$

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 β_q and β_h denote taste coefficients on the q^{th} and h^{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:

$$cov_n(\beta_q, \beta_h) = \sum_{c=1}^C \pi_{cn}(\theta) \beta_{c,q} \beta_{c,h} - \left(\sum_{c=1}^C \pi_{cn}(\theta) \beta_{c,q} \right) \left(\sum_{c=1}^C \pi_{cn}(\theta) \beta_{c,h} \right) \quad (7)$$

where $\beta_{c,q}$ is the value of β_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 `lclogit` command

`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. `fmlogit` is a user-written program. See footnote 5 for a further description.

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.⁶ 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. β_c in equation 1. For instance, suppose that `x1` and `x2` are alternative-specific characteristics included in `indepvars` for `lclogit` and the user wishes to restrict the coefficient on `x1` to zero for Class1 and Class4, and the coefficient on `x2` 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.

- **pr0** predicts the unconditional choice probability, which equals the average of class-specific choice probabilities weighted by the corresponding class shares. That is, $\sum_{c=1}^C \pi_{cn}(\boldsymbol{\theta}) [\exp(\boldsymbol{\beta}_c \mathbf{x}_{njt}) / (\sum_{k=1}^J \exp(\boldsymbol{\beta}_c \mathbf{x}_{nkt}))]$ in the context of Section 2.
- **pr** predicts the unconditional choice probability and the choice probabilities conditional on being in particular classes; $\exp(\boldsymbol{\beta}_c \mathbf{x}_{njt}) / (\sum_{k=1}^J \exp(\boldsymbol{\beta}_c \mathbf{x}_{nkt}))$ 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 $c = 1, 2, \dots, C$.

5 Post-estimation command: `lologitcov`

`lologitcov` predicts the implied variances and covariances of taste parameters by evaluating equation 7 at the active `lologit` 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 `lologitcov` is:

```
lologitcov [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 `lologit` - in the prediction sample are reported as a covariance matrix at the end of `lologitcov`'s execution.

The options for `lologitcov` 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 `d0` method to fit generalised linear latent class and mixed models including LCL via the Newton-Raphson (NR) algorithm for likelihood maximization.⁷ This post-estimation command passes active `lclogit` 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 `lclogitpr` 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 `lclogitpr` 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 (1997)'s. 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.
- `nopost` restores the currently active `ereturn` results at the end of the command's execution.

7. `gllamm` can be downloaded by entering `ssc install gllamm` into the command window.

- `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)
```

	y	price	contract	local	wknown	tod	seasonal	gid	pid	_x1
1.	0	7	5	0	1	0	0	1	1	26
2.	0	9	1	1	0	0	0	1	1	26
3.	0	0	0	0	0	0	1	1	1	26
4.	1	0	5	0	1	1	0	1	1	26
5.	0	7	0	0	1	0	0	2	1	26
6.	0	9	5	0	1	0	0	2	1	26

7.	1	0	1	1	0	1	0	2	1	26
8.	0	0	5	0	0	0	1	2	1	26
9.	0	9	5	0	0	0	0	3	1	26
10.	0	7	1	0	1	0	0	3	1	26
11.	0	0	0	0	1	1	0	3	1	26
12.	1	0	0	1	0	0	1	3	1	26

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: ⁸

```
. forvalues c = 2/10 {
2.      lclogit y price contract local wknown tod seasonal, group(gid) id(pid)
> nclasses(`c`) membership(_x1) seed(1234567890)
3. matrix b = e(b)
4. matrix ic = nullmat(ic) \ `e(nclasses)` , `e(ll)` , `=colsof(b)` , `e(caic)` , `e(bic)`
5. }
(output omitted)
. matrix colnames ic = "Classes" "LLF" "Nparam" "CAIC" "BIC"
. matlist ic, name(columns)
```

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

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:

```
. lclogit y price contract local wknown tod seasonal, group(gid) id(pid) nclass
> es(5) membership(_x1) 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
```

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 $CAIC = -2\ln L + m(1 + \ln N)$.

price	-0.902	-0.325	-0.763	-1.526	-0.520
contract	-0.470	0.011	-0.533	-0.405	-0.016
local	0.424	3.120	0.527	0.743	3.921
wknown	0.437	2.258	0.325	1.031	3.063
tod	-8.422	-2.162	-5.379	-15.677	-6.957
seasonal	-6.354	-2.475	-7.763	-14.783	-6.941
Class Share	0.113	0.282	0.162	0.243	0.200

Class membership model parameters : Class5 = Reference class

Variable	Class1	Class2	Class3	Class4	Class5
_x1	0.045	0.040	0.047	0.048	0.000
_cons	-1.562	-0.544	-1.260	-0.878	0.000

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:⁹

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

Variable	Obs	Mean	Std. Dev.	Min	Max
cpmax	100	.9596674	.0860159	.5899004	1

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 :

```
. 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.

```

2.      quietly replace class = `c' if cpmx==cp`c`
3. }
. forvalues c = 1/`e(nclasses)' {
2.      qui sum pr if class == `c' & y==1
3.      local n=r(N)
4.      local a=r(mean)
5.      qui sum pr`c' if class == `c' & y==1
6.      local b=r(mean)
7.      matrix pr = nullmat(pr) \ `n', `c', `a', `b'
8. }
. matrix colnames pr = "Obs" "Class" "Uncond_Pr" "Cond_PR"
. matlist pr, name(columns)

```

	Obs	Class	Uncond_Pr	Cond_PR
	129	1	.3364491	.5387555
	336	2	.3344088	.4585939
	191	3	.3407353	.5261553
	300	4	.4562778	.7557497
	239	5	.4321717	.6582177

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. `lclogit` saves these statistics as part of its `ereturn list`:

```

. matrix list e(PB)
e(PB) [1,6]

```

	Average price	Average contract	Average local	Average wknown	Average tod	Average seasonal
Coefficients	-.79129	-.23756	1.9795	1.6029	-7.62728	-7.64949

```

. matrix list e(CB)
symmetric e(CB) [6,6]

```

	price	contract	local	wknown	tod	seasonal
price	.20833629					
contract	.07611239	.05436665				
local	.48852574	.32683725	2.1078043			
wknown	.27611961	.22587673	1.4558029	1.045789		
tod	2.2090348	.65296465	4.0426714	1.9610973	25.12504	
seasonal	1.9728148	.65573999	3.8801716	2.0070985	21.845013	20.189302

Since we estimated a model with the `membership()` option, the class shares (hence the covariances; see equation 7) 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:

```
. quietly lclogitcov price contract
. sum var_1 cov_12 var_2 if _x1 >20 & first
```

Variable	Obs	Mean	Std. Dev.	Min	Max
var_1	62	.2151655	.0061303	.2065048	.2301424
cov_12	62	.0765989	.000348	.0760533	.0773176
var_2	62	.0545157	.0000987	.0543549	.0547015

```
. sum var_1 cov_12 var_2 if _x1 <=20 & first
```

Variable	Obs	Mean	Std. Dev.	Min	Max
var_1	38	.1971939	.0053252	.1841499	.2050795
cov_12	38	.0753185	.0004483	.0741831	.075949
var_2	38	.0541235	.0001431	.0537589	.0543226

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 `gllamm` (Rabe-Hesketh et al., 2002), to take advantage of the EM algorithm and obtain conventional standard errors at the same time. `lclogitml` simplifies this process.

```
. lclogitml, iter(5)
-gllamm- is initializing. This process may take a few minutes.
Iteration 0: log likelihood = -1039.7709 (not concave)
Iteration 1: log likelihood = -1039.7709
Iteration 2: log likelihood = -1039.7706
Iteration 3: 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	-.2774167	1.125685
wknown	.4370318	.2864782	1.53	0.127	-.1244552	.9985188
tod	-8.422232	1.584777	-5.31	0.000	-11.52834	-5.316126
seasonal	-6.354626	1.569516	-4.05	0.000	-9.43082	-3.278431
choice2						
price	-.3249095	.1090047	-2.98	0.003	-.5385548	-.1112642
contract	.0108523	.0384404	0.28	0.778	-.0644894	.0861941
local	3.122255	.2842558	10.98	0.000	2.565124	3.679387
wknown	2.258772	.2553446	8.85	0.000	1.758306	2.759238
tod	-2.157726	.8906932	-2.42	0.015	-3.903453	-.4119997
seasonal	-2.470511	.894278	-2.76	0.006	-4.223264	-.7177582
choice3						
price	-.7629762	.1415072	-5.39	0.000	-1.040325	-.4856271
contract	-.5331056	.0739354	-7.21	0.000	-.6780162	-.3881949

local	.526889	.2633905	2.00	0.045	.0106531	1.043125
wknown	.3249201	.2391513	1.36	0.174	-.1438078	.793648
tod	-5.379464	1.100915	-4.89	0.000	-7.537218	-3.22171
seasonal	-7.763171	1.191777	-6.51	0.000	-10.09901	-5.427331
<hr/>						
choice4						
price	-1.526036	.1613542	-9.46	0.000	-1.842284	-1.209787
contract	-.4051809	.0754784	-5.37	0.000	-.5531158	-.2572459
local	.7413859	.3599632	2.06	0.039	.0358711	1.446901
wknown	1.029899	.3032522	3.40	0.001	.4355353	1.624262
tod	-15.68543	1.523334	-10.30	0.000	-18.67111	-12.69975
seasonal	-14.78921	1.463165	-10.11	0.000	-17.65696	-11.92146
<hr/>						
choice5						
price	-.5194972	.1357407	-3.83	0.000	-.7855442	-.2534503
contract	-.0141426	.0915433	-0.15	0.877	-.1935641	-.165279
local	3.907502	.7079699	5.52	0.000	2.519907	5.295097
wknown	3.055901	.4653005	6.57	0.000	2.143928	3.967873
tod	-6.939564	1.428877	-4.86	0.000	-9.740112	-4.139015
seasonal	-6.92799	1.363322	-5.08	0.000	-9.600052	-4.255928
<hr/>						
share1						
_x1	.0443861	.0510411	0.87	0.385	-.0556525	.1444247
_cons	-1.562361	1.197298	-1.30	0.192	-3.909022	.7843001
<hr/>						
share2						
_x1	.0400449	.0427769	0.94	0.349	-.0437962	.1238861
_cons	-.5443567	.956636	-0.57	0.569	-2.419329	1.330615
<hr/>						
share3						
_x1	.0470822	.0458336	1.03	0.304	-.0427501	.1369144
_cons	-1.260251	1.061043	-1.19	0.235	-3.339857	.8193542
<hr/>						
share4						
_x1	.0479228	.042103	1.14	0.255	-.0345976	.1304431
_cons	-.8794649	.9718415	-0.90	0.365	-2.784239	1.025309

The estimated choice model or taste parameters, β_c , and class membership model parameters, θ_c , are grouped under equations `choicec` and `sharec` 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.¹⁰ 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

10. The original output table `gllamm` reports is lengthier and somewhat less intuitive in comparison. For instance, it splits the six estimates displayed under equation `choice1` over six different equations, labeled `z.1.1`, `z.2.1`, `z.3.1`, `z.4.1`, `z.5.1` and `z.6.1` respectively.

likelihood or the parameter vector are smaller than a given tolerance level.¹¹

When `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 `seed()`. Train (2008) contains references highlighting the importance of these issues for applications exploiting EM algorithms.

8 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).

9 References

- Bhat, C., 1997. An endogenous segmentation mode choice model with an application to intercity travel. *Transportation Science*, 3, pp. 34-48.
- Greene, W. and Hensher, D., 2003. A latent class model for discrete choice analysis: contrasts with mixed logit. *Transportation Research Part B*, 37 (8), pp. 681-698.
- Hess, S., Ben-Akiva, M, Gopinath, D. and Walker, J., 2011. Advantages of latent class over mixture of logit models, mimeo.
- Hole, A.R., 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal*, 7 (3), pp. 388-401.
- Huber, J. and K. Train, 2001. On the similarity of classical and bayesian estimates of individual mean partworths. *Marketing Letters*, 12, pp. 259-269.
- Pacifico, D., 2012. Estimating nonparametric mixed logit models via EM algorithm. *Stata Journal* 12 (2), pp. 284-298.
- Rabe-Hesketh, S., Skrondal, A. and Pickles, A. 2002. Reliable estimation of generalized linear mixed models using adaptive quadrature. *Stata Journal*, 2 (1), pp. 1-21.
- Shen, J., 2009. Latent class model or mixed logit model? A comparison by transport mode choice data. *Applied Economics*, 41 (22), pp. 2915-2924.
- Train, K., 2008. EM Algorithms for Nonparametric Estimation of Mixing Distributions. *Journal of Choice Modelling*, 1 (1), pp. 40-69.

11. The benefit of pre-use of `lclogit` cannot be overstated. Since `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. `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.

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).