The Stata Journal (2013)
13, Number 3, pp. 625–639

lclogit: A Stata command for fitting latent-class conditional logit models via the expectation-maximization algorithm

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

Hong il Yoo
Durham University
Durham, UK
h.i.yoo@durham.ac.uk

Abstract. In this article, we describe `lclogit`, a Stata command for fitting a discrete-mixture or latent-class logit model via the expectation-maximization algorithm.

Keywords: st0312, lclogit, lclogitpr, lclogitcov, lclogitml, latent-class model, expectation-maximization 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 fit through simulated maximum likelihood, as in Hole's (2007) Stata command `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 fitting a discrete-mixture logit model, also known as a latent-class logit (LCL) 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 because inversion of the (approximate) Hessian becomes numerically difficult.

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

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

2 EM algorithm for LCL

This section recapitulates the EM algorithm for fitting an LCL model.[2] Suppose that each of N agents faces, for notational simplicity, J alternatives in each of T choice scenarios.[3] Let y_{njt} denote a binary variable that equals 1 if agent n chooses alternative j in scenario t and equals 0 otherwise. Each alternative is described by alternative-specific characteristics $\boldsymbol{x}_{njt}$ and each agent by agent-specific characteristics, including a constant, $\boldsymbol{z}_n$.

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

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

Because 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 (1) over classes. The weight for class c, $\pi_{cn}(\boldsymbol{\theta})$, is the population share of that class and is usually modeled as fractional multinomial logit,

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

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{C-1})$ are class membership model parameters; note that $\boldsymbol{\theta}_C$ has been normalized to 0 for identification.

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

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

Bhat (1997) and Train (2008) note numerical difficulties associated with maximizing (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 sth iteration of this algorithm. Then at iteration $s+1$, the estimates are updated as

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

2. Further details are available in Bhat (1997) and Train (2008).
3. `lclogit` is also applicable when the number of scenarios varies across agents, and the number of alternatives varies both across agents and over scenarios.

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

$$\eta_{cn}(\boldsymbol{\beta}^s, \boldsymbol{\theta}^s) = \frac{\pi_{cn}(\boldsymbol{\theta}^s) P_n(\boldsymbol{\beta}_c^s)}{\sum_{l=1}^{C} \pi_{ln}(\boldsymbol{\theta}^s) P_n(\boldsymbol{\beta}_l^s)} \tag{4}$$

The updating procedure can be implemented easily in Stata, exploiting `clogit` and `fmlogit` routines as follows.[4] $\boldsymbol{\beta}^{s+1}$ is computed by fitting a conditional logit model (`clogit`) C times, each time using $\eta_{cn}(\boldsymbol{\beta}^s, \boldsymbol{\theta}^s)$ for a particular c to weight observations on each n. $\boldsymbol{\theta}^{s+1}$ is obtained by fitting a fractional multinomial logit model (`fmlogit`) that takes $\eta_{1n}(\boldsymbol{\beta}^s, \boldsymbol{\theta}^s)$, $\eta_{2n}(\boldsymbol{\beta}^s, \boldsymbol{\theta}^s)$, $\ldots$, $\eta_{Cn}(\boldsymbol{\beta}^s, \boldsymbol{\theta}^s)$ as dependent variables. When $\boldsymbol{z}_n$ only includes the constant term so that each class share is the same for all agents, that is, when $\pi_{cn}(\boldsymbol{\theta}) = \pi_c(\boldsymbol{\theta})$, each class share can be directly updated by using the following analytical solution without fitting the fractional multinomial logit model:

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

With a suitable selection of starting values, the updating procedure can be repeated until changes in the estimates and 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 a 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 qth and hth 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

$$\mathrm{cov}_n(\beta_q, \beta_h) = \sum_{c=1}^{C} \pi_{cn}(\boldsymbol{\theta}) \beta_{c,q} \beta_{c,h} - \left\{ \sum_{c=1}^{C} \pi_{cn}(\boldsymbol{\theta}) \beta_{c,q} \right\} \left\{ \sum_{c=1}^{C} \pi_{cn}(\boldsymbol{\theta}) \beta_{c,h} \right\} \tag{6}$$

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 $\boldsymbol{z}_n$ in (2) includes a nonconstant variable, this covariance will vary across agents with different observed characteristics through the variation in $\pi_{cn}(\boldsymbol{\theta})$.

3 The lclogit command

`lclogit` is a Stata command that implements the EM iterative scheme outlined in the previous section. This command generalizes Pacifico's (2012) 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.

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

For example, the internal code of `lclogit` executes fewer algebraic operations per iteration to update the estimates; uses the standard `generate` command to perform tasks that were previously executed with slightly slower `egen` functions; and, when possible, works with log probabilities instead of probabilities. All of these changes substantially reduce the estimation run time, especially in the presence of a large number of parameters and observations. If we take the 8-class model fit by Pacifico (2012) as an example, `lclogit` produces the same results as the step-by-step procedure while taking less than one-half of the run time.

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

3.1 Syntax

The generic syntax for `lclogit` is

`lclogit` *depvar* [*indepvars*] [*if*] [*in*], `group(`*varname*`)` `id(`*varname*`)`
`nclasses(`#`)` [`membership(`*varlist*`)` `convergence(`#`)` `iterate(`#`)` `seed(`#`)`
`constraints(Class`# *numlist*: [`Class`# *numlist*: ...]`)` `nolog`]

3.2 Options

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

`id(`*varname*`)` specifies a numeric identifier variable for the choice makers or agents. With cross-section data, users should specify the same variable for both the `group()` and the `id()` options. `id()` is required.

`nclasses(`#`)` specifies the number of latent classes used in the estimation. A minimum of two latent classes is required. `nclasses()` is required.

`membership(`*varlist*`)` specifies independent variables to enter the fractional multinomial logit model of class membership, that is, the variables included in the vector $\boldsymbol{z}_n$ of (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 (5).

`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 `convergence(0.00001)`.

5. Pacifico (2012) specified an `ml` program with the method `lf` to fit the class membership model. `lclogit` uses another user-written program from Buis (2008), `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` and `fmlogit_pr` because 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 `iterate(150)`.

`seed(#)` sets the seed for pseudouniform random numbers. The default is the `creturn` value `c(seed)`.

The starting values for taste parameters are obtained by splitting the sample into `nclasses()` different subsamples and fitting a `clogit` model for each of them. During this process, a pseudouniform 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, `lclogit` uses equal shares, that is, 1/`nclasses()`.

`constraints(Class#` *numlist*: [`Class#` *numlist*: ...]`)` specifies the constraints that are imposed on the taste parameters of the designated classes, that is, $\boldsymbol{\beta}_c$ in (1). For instance, suppose that `x1` and `x2` are alternative-specific characteristics included in *indepvars* for `lclogit` and that the user wishes to restrict the coefficient on `x1` to 0 for `Class1` and `Class4` and the coefficient on `x2` to 2 for `Class4`. Then the relevant series of commands would look like this:

```
constraint 1 x1 = 0
constraint 2 x2 = 2
lclogit depvar indepvars, group(varname) id(varname)  ///
     nclasses(8) constraints(Class1 1: Class4 1 2)
```

`nolog` suppresses the display of the iteration log.

4 Postestimation 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, which is stored in a variable named *stubname*.

4.1 Syntax

The syntax for `lclogitpr` is

`lclogitpr` *stubname* [*if*] [*in*] [, `class(`*numlist*`) pr0 pr up cp`]

6. More specifically, the unit interval is divided into `nclasses()` equal parts, and if the agent's pseudorandom draw is in the cth 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 pseudorandom draws have been made.

4.2 Options

`class(`*numlist*`)` 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. That is, $\sum_{c=1}^{C} \pi_{cn}(\boldsymbol{\theta})[\exp(\boldsymbol{\beta}_c \boldsymbol{x}_{njt})/\{\sum_{k=1}^{J} \exp(\boldsymbol{\beta}_c \boldsymbol{x}_{nkt})\}]$ in the context of section 2.

`pr` predicts the unconditional choice probability and the choice probabilities conditional on being in particular classes; $\exp(\boldsymbol{\beta}_c \boldsymbol{x}_{njt})/\{\sum_{k=1}^{J} \exp(\boldsymbol{\beta}_c \boldsymbol{x}_{nkt})\}$ in (1) corresponds to the choice probability conditional on being in class c. This is the default option.

`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 (2) in section 2.

`cp` predicts the posterior probabilities that the agent is in particular classes, taking into account his or her sequence of choices. They are computed by evaluating (4) at the final estimates for each $c = 1, 2, \ldots, C$.

5 Postestimation command: lclogitcov

`lclogitcov` predicts the implied variances and covariances of taste parameters by evaluating (6) 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*] [`, nokeep` `varname(`*stubname*`)` `covname(`*stubname*`)` `matrix(`*name*`)`]

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 qth variable listed in *varlist*, and to store the predicted covariances in `cov_12`, `cov_13`, ..., `cov_23`, ..., where `cov_`qh is the predicted covariance between the coefficients on the qth variable and the hth variable in *varlist*.

The averages of these variances 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.

5.1 Options

`nokeep` drops the predicted variances and covariances from the dataset at the end of the command's execution. The average covariance matrix is still displayed.

`varname(`*stubname*`)` requests that the predicted variances be stored as *stubname*`1`, *stubname*`2`, ...

`covname(`*stubname*`)` requests that the predicted covariances be stored as *stubname*`12`, *stubname*`13`, ...

`matrix(`*name*`)` stores the reported average covariance matrix in a Stata matrix called *name*.

6 Postestimation command: lclogitml

`lclogitml` is a wrapper for `gllamm` (Rabe-Hesketh, Skrondal, and Pickles 2002), which uses the `d0` method to fit generalized linear latent-class and mixed models, including LCL, via the Newton–Raphson (NR) algorithm for likelihood maximization.[7] This postestimation command passes active `lclogit` specification and estimates to `gllamm`, and its primary use mainly depends on how the `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 postestimation 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 postestimation commands but lose access to `lclogitpr` and `lclogitcov`.

`lclogitml` can also be used as its own postestimation 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*] [`,` `iterate(`#`)` `level(`#`)` `nopost` `switch` *compatible_gllamm_options*]

6.1 Options

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

7. `gllamm` can be downloaded by typing `ssc install gllamm` into the Command window.

With a nonzero 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 uses 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 nonzero 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 the confidence level. The default is `level(95)`.

`nopost` restores the currently active `ereturn` results at the end of the command's execution.

`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 postestimation 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 a household's choice of electricity supplier accompanying Hole's (2007) `mixlogit` command, 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, while the numeric variables `pid` and `gid` identify customers and choice occasions, respectively. To illustrate the use of the `membership()` option, we generate a pseudorandom regressor `_x1`, which mimics a demographic variable. The data are organized as follows:

```
. use http://fmwww.bc.edu/repec/bocode/t/traindata.dta
. set seed 1234567890
. by pid, sort: 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 the Bayesian information criterion (BIC) and consistent Akaike information criterion (CAIC). The next lines show how to estimate nine LCL specifications repeatedly and obtain the related information criteria:[8]

```
. forvalues c = 2/10 {
  2.         quietly 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

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

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)
> nclasses(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

       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
```

Note 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 postestimation command `lclogitpr` with the `up` option.

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]

```
. by `e(id)´, sort: generate first = _n==1
. lclogitpr cp, cp
. egen double cpmax = rowmax(cp1-cp5)
. summarize cpmax if first, sep(0)
    Variable        Obs        Mean    Std. Dev.       Min        Max

       cpmax        100    .9596674    .0860159   .5899004          1
```

9. A dummy variable that 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.

As 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 him or 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
. generate byte class = .
(4780 missing values generated)
. forvalues c = 1/`e(nclasses)´ {
  2.         quietly replace class = `c´ if cpmax==cp`c´
  3. }
. forvalues c = 1/`e(nclasses)´ {
  2.         quietly summarize pr if class == `c´ & y==1
  3.         local n=r(N)
  4.         local a=r(mean)
  5.         quietly summarize 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 class. 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 of:  Average of:  Average of:  Average of:  Average of:
                    price     contract        local       wknown          tod
Coefficients   -.79129238   -.23755636    1.9794603    1.6029319   -7.6272765

               Average of:
                 seasonal
Coefficients   -7.6494889

. 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
```

Because we fit a model with the `membership()` option, the class shares [hence, the covariances; see (6)] now vary across respondents, and the matrix `e(CB)` above is an average covariance matrix. In this case, the postestimation 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 than or less than 20:

```
. quietly lclogitcov price contract

. summarize 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

. summarize var_1 cov_12 var_2 if _x1 <=20 & first

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       var_1 |        38    .1971939    .0053252   .1841498   .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 such as `gllamm` (Rabe-Hesketh, Skrondal, and Pickles 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   .2012346    -4.48   0.000    -1.296719   -.5078943
    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   1.584778    -5.31   0.000    -11.52834   -5.316125
    seasonal |  -6.354626   1.569516    -4.05   0.000    -9.430821    -3.27843
-------------+----------------------------------------------------------------
choice2      |
       price |  -.3249095   .1090047    -2.98   0.003    -.5385547   -.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   .8906931    -2.42   0.015    -3.903453   -.4119999
    seasonal |  -2.470511   .8942779    -2.76   0.006    -4.223263   -.7177583
-------------+----------------------------------------------------------------
choice3      |
       price |  -.7629762   .1415072    -5.39   0.000    -1.040325   -.4856272
    contract |  -.5331056   .0739354    -7.21   0.000    -.6780162    -.388195
       local |    .526889   .2633905     2.00   0.045     .0106531    1.043125
      wknown |   .3249201   .2391513     1.36   0.174    -.1438078    .7936479
         tod |  -5.379464   1.100915    -4.89   0.000    -7.537217    -3.22171
    seasonal |  -7.763171   1.191777    -6.51   0.000    -10.09901   -5.427331
-------------+----------------------------------------------------------------
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
-------------+----------------------------------------------------------------
choice5      |
       price |  -.5194972   .1357407    -3.83   0.000    -.7855442   -.2534503
    contract |  -.0141426   .0915433    -0.15   0.877    -.1935642     .165279
       local |   3.907502     .70797     5.52   0.000     2.519906    5.295098
      wknown |   3.055901   .4653006     6.57   0.000     2.143928    3.967873
         tod |  -6.939564   1.428878    -4.86   0.000    -9.740112   -4.139015
    seasonal |   -6.92799   1.363322    -5.08   0.000    -9.600052   -4.255928
-------------+----------------------------------------------------------------
share1       |
         _x1 |   .0443861   .0510411     0.87   0.385    -.0556525    .1444247
       _cons |  -1.562361   1.197298    -1.30   0.192    -3.909022    .7843005
-------------+----------------------------------------------------------------
share2       |
         _x1 |   .0400449   .0427769     0.94   0.349    -.0437962    .1238861
       _cons |  -.5443567   .9566361    -0.57   0.569    -2.419329    1.330616
-------------+----------------------------------------------------------------
             |
```

```
share3       |
         _x1 |   .0470822   .0458336     1.03   0.304    -.0427501    .1369145
       _cons |  -1.260251   1.061043    -1.19   0.235    -3.339857    .8193545
-------------+----------------------------------------------------------------
share4       |
         _x1 |   .0479228    .042103     1.14   0.255    -.0345976    .1304431
       _cons |  -.8794649   .9718417    -0.90   0.365     -2.78424     1.02531
------------------------------------------------------------------------------
```

The fitted choice model or taste parameters $\boldsymbol{\beta}_c$ and class membership model parameters $\boldsymbol{\theta}_c$ 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.[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 three iterations, though the parameter estimates remain almost the same. This may happen because `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.[11]

When `lclogit` is used in a final production run, you should 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 coauthored this article when he was studying for a PhD in economics at the University of New South Wales in Sydney, Australia. Yoo's work was supported under the Australian Research Council's Discovery Projects funding scheme (project number DP0881205).

10. The original output table `gllamm` report 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`.
11. The benefit of using `lclogit` beforehand cannot be overstated. Because `gllamm` uses the `d0` evaluator and the LCL log likelihood is not amenable to direct maximization, each iteration tends to last for a long time, and finding initial values that lead to convergence often involves a laborious search. `lclogit` exploits the EM algorithm, which in theory guarantees convergence to a local maximum, and takes the estimates to a local maximum or its close neighborhood in a relatively fast way in practice.

9 References

Bhat, C. R. 1997. An endogenous segmentation mode choice model with an application to intercity travel. *Transportation Science* 31: 34–48.

Buis, M. L. 2008. fmlogit: Stata module fitting a fractional multinomial logit model by quasi maximum likelihood. Statistical Software Components S456976, Department of Economics, Boston College. http://ideas.repec.org/c/boc/bocode/s456976.html.

Greene, W. H., and D. A. Hensher. 2003. A latent class model for discrete choice analysis: Contrasts with mixed logit. *Transportation Research, Part B* 37: 681–698.

Hess, S., M. Ben-Akiva, D. Gopinath, and J. Walker. 2011. Advantages of latent class over continuous mixture of logit models. http://www.stephanehess.me.uk/papers/Hess_Ben-Akiva_Gopinath_Walker_May_2011.pdf.

Hole, A. R. 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401.

Huber, J., and K. Train. 2001. On the similarity of classical and Bayesian estimates of individual mean partworths. *Marketing Letters* 12: 259–269.

Pacifico, D. 2012. Fitting nonparametric mixed logit models via expectation-maximization algorithm. *Stata Journal* 12: 284–298.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2002. Reliable estimation of generalized linear mixed models using adaptive quadrature. *Stata Journal* 2: 1–21.

Shen, J. 2009. Latent class model or mixed logit model? A comparison by transport mode choice data. *Applied Economics* 41: 2915–2924.

Train, K. E. 2008. EM algorithms for nonparametric estimation of mixing distributions. *Journal of Choice Modelling* 1: 40–69.

About the authors

Daniele Pacifico works with the Italian Department of the Treasury in Rome, Italy.

Hong il Yoo is a lecturer in economics at Durham University Business School in Durham, United Kingdom.