The Stata Journal (2020)
20, Number 2, pp. 405–425 DOI: 10.1177/1536867X20931003

lclogit2: An enhanced command to fit latent class conditional logit models

Hong Il Yoo
Durham University Business School
Durham, UK
h.i.yoo@durham.ac.uk

Abstract. In this article, I describe the `lclogit2` command, an enhanced version of `lclogit` (Pacifico and Yoo, 2013, *Stata Journal* 13: 625–639). Like its predecessor, `lclogit2` uses the expectation-maximization algorithm to fit latent class conditional logit (LCL) models. But it executes the expectation-maximization algorithm's core algebraic operations in Mata, so it runs considerably faster as a result. It also allows linear constraints on parameters to be imposed more conveniently and flexibly. It comes with the parallel command `lclogitml2`, a new stand-alone command that uses gradient-based algorithms to fit LCL models. Both `lclogit2` and `lclogitml2` are supported by a new postestimation command, `lclogitwtp2`, that evaluates willingness-to-pay measures implied by fitted LCL models.

Keywords: st0601, lclogit2, lclogitml2, lclogitpr2, lclogitcov2, lclogitwtp2, latent class model, conditional logit, expectation-maximization algorithm, lclogit, fmm, finite mixture, mixlogit, mixed logit, willingness to pay

1 Introduction

The latent class conditional logit (LCL) model extends the conditional logit model (`clogit` in Stata) by incorporating a discrete representation of unobserved preference heterogeneity. Algebraically, the LCL likelihood function is a finite mixture of C different conditional logit likelihood functions. Stata 15 introduced the `fmm` command, which fits many finite mixture models; as of Stata 16, however, `fmm` does not support `clogit` as a component model. The community-contributed `lclogit` command (Pacifico and Yoo 2013) allows Stata users to fit LCL models. But it underuses Stata's computing capabilities available via the Mata environment and does not allow the component conditional logit models to share any parameter in common.

In this article, I describe `lclogit2`, an enhanced version of `lclogit`. Like its predecessor, `lclogit2` applies Bhat's (1997) expectation-maximization (EM) algorithm to obtain the maximum likelihood estimates (MLEs) of LCL. The EM algorithm is an attractive method to maximize the nonconcave log-likelihood function of LCL because it offers greater numerical stability than the usual Newton-type techniques that `ml maximize` applies. Train (2008) provides a masterful summary of the source of this advantage.

`lclogit2` comes with a parallel command, `lclogitml2`, that fits LCL models using the usual techniques for maximum likelihood estimation. While `lclogitml2` is fully functional as a stand-alone command, it may be also used as a postestimation tool for `lclogit2`. The EM algorithm used by `lclogit2` (and by `lclogit`) produces coefficient estimates without standard errors. To draw statistical inferences, users may pass active `lclogit2` estimates as starting values to `lclogitml2` and obtain the usual `ml maximize` output table with standard errors.

Major differences between the `lclogit2` and `lclogit` commands may be summarized as follows. To facilitate discussion, let $\boldsymbol{\beta}_c$ denote a vector of coefficients for the cth `clogit` component, or latent class, of LCL.

First, `lclogit2` estimates a given LCL specification considerably faster than `lclogit` by using Mata to execute the core algebraic operations of the EM algorithm. `lclogit` executes the same operations in the regular Stata environment.

Second, `lclogit2` allows $\boldsymbol{\beta}_c$ to include homogeneous coefficients that are identical across classes, as well as heterogeneous coefficients that vary across classes. Hole's (2007c) `mixlogit` command can fit a mixed logit model that includes a combination of nonrandom coefficients and multivariate normal random coefficients. The new feature of `lclogit2` allows estimation of a latent class counterpart to such a model specification. `lclogit` assumes that every coefficient is heterogeneous.

Third, `lclogit2` can incorporate any set of linear constraints on $\boldsymbol{\beta}_c$ for $c = 1, 2, \ldots, C$, defined using Stata's `constraint` command. The constraints may apply within a class (for example, two different coefficients in $\boldsymbol{\beta}_1$ are equal to 0) as well as across different classes (for example, a coefficient in $\boldsymbol{\beta}_1$ and the corresponding coefficient in $\boldsymbol{\beta}_2$ are the same). `lclogit` can incorporate within-class constraints only and has peculiar syntax requirements for inputting the constraints.

Fourth, `lclogitml2` is a stand-alone estimation command. `lclogitml`, which accompanies `lclogit`, is simply a wrapper that passes `lclogit` estimates to another community-contributed command, `gllamm` (Rabe-Hesketh, Skrondal, and Pickles 2002). This difference brings about several advantages:

a. `lclogitml2` uses a log-likelihood evaluator coded in Mata. It estimates a given LCL specification considerably faster than `gllamm`, which uses an evaluator coded in the regular Stata environment.

b. `lclogitml2` can inherit linear constraints defined for `lclogit2`. In contrast, to impose the same constraints across `lclogit` and `lclogitml`, users must define a set of constraints to comply with the syntax requirements of `lclogit` and another set of constraints to comply with those of `gllamm`.

c. `lclogitml2` is better suited to fitting a model with many heterogeneous coefficients. Suppose that each $\boldsymbol{\beta}_c$ consists of K heterogeneous coefficients so that there are a total of $C \times K$ heterogeneous coefficients to estimate. In `ml model`'s vernacular, `lclogitml2` will add C "equations", where each equation comprises K coefficients for a particular class. In contrast, `gllamm` will add $C \times K$ equations, where each equation's intercept is a particular coefficient. With a large $C \times K$, a call to `gllamm` (via `lclogitml`) may fail with an error message stating that some equation is not found, presumably because there is a limit on the number of equations that `ml model` can receive from `gllamm`.

Finally, when `lclogit2` or `lclogitml2` results are active, a new postestimation tool, `lclogitwtp2`, can calculate willingness-to-pay (WTP) measures implied by the coefficient estimates. Within each class c, the WTP for attribute k is calculated as the ratio of the coefficient on that attribute to another coefficient that can be interpreted as the marginal utility of money. In nonmarket valuation studies, such WTP measures are often the main parameters of interest. To derive the WTP measures from `lclogit` or `lclogitml` estimates, users need to write their own postestimation programs.

2 Latent class conditional logit

Consider decision maker n making a choice from J alternatives in each of T choice occasions, where $n = 1, 2, \ldots, N$. Alternative j, available to him or her in occasion t, is described by a row vector of K attributes, $\boldsymbol{x}_{njt}$. Denote by y_{njt} a binary indicator that equals 1 if his or her choice is alternative j, and 0 otherwise. Under the conditional logit model (`clogit` in Stata), the joint likelihood of his or her T choices is given by

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

where $\boldsymbol{\beta}$ is a column vector of K coefficients, which can be interpreted as the marginal utilities of the corresponding entries in $\boldsymbol{x}_{njt}$. As a matter of fact, `clogit` (as well as `lclogit2` and `lclogitml2`) can also accommodate datasets with T varying across decision makers and J varying across decision makers, choice occasions, or both. While T and J in (1) must be more accurately written as T_n and J_{nt}, the subscripts will be omitted for notational simplicity.[1]

The LCL extends the conditional logit by incorporating a discrete representation of unobserved preference heterogeneity across decision makers. Specifically, LCL assumes that there are C distinct types, or "classes", of decision makers and that each class c makes choices consistent with its own `clogit` model with utility coefficient vector $\boldsymbol{\beta}_c$. Suppose that the probability that decision maker n belongs to class c is given by a fractional multinomial logit specification

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

where $\boldsymbol{z}_n$ is a row vector of decision maker n's characteristics and the usual constant regressor (that is, 1); $\boldsymbol{\theta}_c$ is a conformable column vector of membership model coefficients for class c, with $\boldsymbol{\theta}_C$ normalized to $\mathbf{0}$ for identification; and $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{C-1})$

1. Stata's *User's Guide* also omits the subscripts from T and J when explaining conditional logit and related models.

denotes a collection of the $C-1$ identified membership coefficient vectors. Under LCL, the joint likelihood of decision maker n's choices is given by

$$L_n(\boldsymbol{B}, \boldsymbol{\Theta}) = \sum_{c=1}^{C} \pi_{nc}(\boldsymbol{\Theta}) P_n(\boldsymbol{\beta}_c) \tag{3}$$

where $\boldsymbol{B} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_C)$ denotes a collection of the C utility coefficient vectors and each $P_n(\boldsymbol{\beta}_c)$ is obtained by evaluating (1) at $\boldsymbol{\beta} = \boldsymbol{\beta}_c$.

The sample log-likelihood function under LCL can be constructed by adding up the natural log of $L_n(\boldsymbol{B}, \boldsymbol{\Theta})$ across N decision makers in the sample. The command `lclogit2` computes the MLE of $\boldsymbol{B}$ and $\boldsymbol{\Theta}$ by using Bhat's (1997) EM algorithm to maximize the sample log-likelihood function. The command `lclogitml2` computes the MLEs of the same coefficients by using gradient-based maximization techniques that Stata's `ml` programs rely on. Unless the EM algorithm has been terminated prior to achieving convergence, `lclogitml2` must produce the same estimates as the existing `lclogit2` estimates when the gradient-based maximization run uses the latter set of estimates as initial values. Train (2008) provides a lucid explanation for this equivalence.[2]

3 Estimation: lclogit2 and lclogitml2

Both `lclogit2` and `lclogitml2` require the same data structure as `clogit` and its extensions, such as `mixlogit` (Hole 2007c) and `lclogit` (Pacifico and Yoo 2013). To aid clarification, let us consider the notation introduced in section 2. The data y_{njt}, $\boldsymbol{x}_{njt}$, and $\boldsymbol{z}_n$ for each distinct triplet of indices $\{n, j, t\}$ must be organized into a separate row in the dataset (that is, an observation in Stata's vernacular). Within a block of data rows associated with consumer n, y_{njt} and $\boldsymbol{x}_{njt}$ thus vary from row to row, whereas $\boldsymbol{z}_n$ is repeated across all rows.

The syntax diagram for `lclogit2` is as follows:

`lclogit2` *depvar* [*varlist1*] [*if*] [*in*] `,` `group(`*varname*`)` `rand(`*varlist2*`)`
`nclasses(`#`)` [`id(`*varname*`)` `membership(`*varlist3*`)` `constraints(`*numlist*`)`
`seed(`#`)` `iterate(`#`)` `ltolerance(`#`)` `tolerance(`#`)` `tolcheck` `nolog`]

The syntax diagram for `lclogitml2` is similar:

`lclogitml2` *depvar* [*varlist1*] [*if*] [*in*] `,` `group(`*varname*`)` `rand(`*varlist2*`)`
`nclasses(`#`)` [`id(`*varname*`)` `membership(`*varlist3*`)` `constraints(`*numlist*`)`
`seed(`#`)` `from(`*init_specs*`)` *noninteractive_options*]

2. As a primer to Train (2008), see Fiebig and Yoo (2019) and Pacifico and Yoo (2013). The former provides an intuitive description of the surrogate objective function that Bhat's (1997) EM algorithm uses in lieu of the sample log-likelihood function. The latter provides a short summary of algebraic operations involved in maximizing the surrogate objective function.

The indicator y_{njt} in section 2 refers to each observation on the dependent variable, *depvar*. Within a block of data rows associated with consumer n and choice occasion t, *depvar* must be equal to 1 in the row describing the alternative that he or she actually chose and 0 in all the other rows.

Each command has three required options. The required option `group(`*varname*`)` is identical to the namesake option in `clogit`, `mixlogit`, and `lclogit` and specifies a numeric variable that identifies distinct choice occasions faced by different decision makers. In the context of (1), the variable in question tells Stata which J data rows to use when evaluating the `clogit` formula inside the large round brackets. The variable must take a unique numeric value for each distinct pair of n and t, and the value must be repeated across all J data rows associated with that pair.

The required option `rand(`*varlist2*`)` is similar to the namesake option in `mixlogit` and specifies attribute variables whose utility coefficients are assumed to vary from class to class. Sometimes, users may wish to constrain a subset of utility coefficients to be identical across all classes. Such constraints can be conveniently requested by using the optional *varlist1* to specify those attributes with class-invariant utility coefficients.[3] To avoid contradiction, do not place a variable in both *varlist2* and *varlist1*. The attribute vector $\boldsymbol{x}_{njt}$ in section 2 refers to each observation on *varlist2* (and, if specified, *varlist1*).

Finally, the required option `nclasses(`#`)` specifies the number of classes, C, in (3). In empirical research, it is common practice to choose the preferred number of classes by estimating an LCL specification repeatedly with different candidate values for C and inspecting which value optimizes the Bayesian information criterion (BIC). See section 5 for further discussion.

Optional options for `lclogit2` include the following:

`id(`*varname*`)` is identical to the namesake option in `mixlogit` and `lclogit` and specifies a numeric identifier variable for decision makers. This variable is expected to identify which block of data rows is associated with each decision maker n; its value must vary from decision maker to decision maker but remain constant within all data rows for the same decision maker. The default is to assume that `group()` and `id()` are identical, which is equivalent to assuming that each decision maker has faced only one choice occasion (that is, $T = 1$).

3. When specifying a mixed logit model, users sometimes assume that the coefficient on price is identical across all decision makers, while the coefficients on all other attributes are normally distributed across decision makers. As Revelt and Train (1998) have noted, the homogeneous price coefficient makes it easier for gradient-based numerical maximizers to find a solution and ensures that the implied WTP for each nonprice attribute has a finite expected value. To estimate an LCL version of this specification, users may include the price variable in *varlist1* and the rest of the attribute variables in *varlist2*.

`membership(`*varlist3*`)` specifies independent variables for the class membership model in (2), except the constant regressor of 1, which is always assumed to be included. Together with the constant regressor, each observation on *varlist3* makes up $\boldsymbol{z}_n$, the vector of decision maker n's characteristics. Within a block of data rows associated with decision maker n, the numerical values of *varlist3* must remain constant across all rows. The default is to assume that *varlist3* is empty; that is, $\boldsymbol{z}_n$ includes only the constant regressor.

`constraints(`*numlist*`)` specifies linear constraints to be applied during estimation. The constraints must be defined using the Stata command `constraint` prior to estimation. The default is to impose no such constraints.

When using `constraint`, note that equation names for the utility coefficients on *varlist1* and *varlist2* are `Fix` and `Class`*c*, respectively, where *c* refers to a particular class number. Therefore, the coefficient on *varname* in *varlist1* is `[Fix]`*varname*. The coefficient on *varname* in *varlist2* is `[Class1]`*varname* for class 1, `[Class2]`*varname* for class 2, and so on.

`seed(`*#*`)` sets the seed for pseudouniform random numbers used in computing starting values. See Pacifico and Yoo (2013) for the detailed procedure. The default seed is `c(seed)`.

`iterate(`*#*`)` specifies the maximum number of iterations. The default is `iterate(1000)`.

`ltolerance(`*#*`)` specifies the tolerance for the log likelihood. When the relative increase in the log likelihood over the last five iterations is less than the specified value, `lclogit2` declares convergence. The default is `ltolerance(0.00001)`.

`tolerance(`*#*`)` specifies the tolerance for the coefficient vector. The default is `tolerance(0.0004)`.

`tolcheck` requests the use of an extra convergence criterion to reduce the chance of false declaration of convergence. If this option is used, `lclogit2` will declare convergence when 1) the relative increase in the log likelihood is smaller than `ltolerance()` and 2) the relative difference in the coefficient vector is smaller than `tolerance()` over the last five iterations.

`nolog` suppresses the display of an iteration log.

As the syntax diagram above shows, many of the optional options for `lclogit2` are also available for `lclogitml2`. Optional options unique to `lclogitml2` are as follows:

`from(`*init_specs*`)` is identical to the namesake option of `mixlogit` (Hole 2007c) and supplies custom starting values for the utility and membership coefficients, that is, $\boldsymbol{B}$ and $\boldsymbol{\Theta}$ in (3). The default starting values are obtained by applying the same procedure as what Pacifico and Yoo (2013) describe for `lclogit`.

noninteractive_options refers to extra options for use with `ml model` in noninteractive mode; see [R] **ml**.

4 Postestimation: lclogitpr2, lclogitcov2, and lclogitwtp2

Both `lclogit2` and `lclogitml2` are supported by three postestimation commands: `lclogitpr2`, `lclogitcov2`, and `lclogitwtp2`. For each decision maker, `lclogitpr2` predicts choice probabilities associated with each alternative in each choice situation that he or she has faced, as well as class membership probabilities. `lclogitcov2` computes variances and covariances of class-specific utility coefficients $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_C$, by considering them as a discrete random variable with probability masses given by class membership probabilities $\pi_{n1}(\boldsymbol{\Theta}), \pi_{n2}(\boldsymbol{\Theta}), \ldots, \pi_{nC}(\boldsymbol{\Theta})$. Finally, `lclogitwtp2` converts estimated utility coefficients into implied WTP measures similarly to how Hole's (2007a) `wtp` works on `clogit` coefficients.

The remainder of this section focuses on the syntax diagram for `lclogitwtp2`, which provides a new postestimation tool that is not available for `lclogit`. The other two postestimation commands have the same functionalities and syntax diagrams as `lclogitpr` and `lclogitcov` which support `lclogit`, apart from the "2" suffix. Pacifico and Yoo (2013) describe `lclogitpr` and `lclogitcov` in detail.

4.1 Syntax for lclogitwtp2

The attribute vector $\boldsymbol{x}_{njt}$ typically includes a pecuniary attribute, which allows the researcher to estimate a utility coefficient that can be associated with the marginal utility of money. Often, the pecuniary attribute measures the cost of acquiring a particular alternative. For example, in Oviedo and Yoo (2017), each alternative is a reforestation project, and the cost is a required increase in the decision maker's tax liabilities to finance that project. In some applications, the pecuniary attribute may measure income generated by a particular alternative instead. For example, in Doiron and Yoo (2017), each alternative is a junior nursing job, and the amount of income is salary earned from that job.

In most nonmarket valuation studies, the index function $\boldsymbol{x}_{njt}\boldsymbol{\beta}$ is assumed to be linear in the pecuniary attribute. The marginal utility of money is then equal to -1 times the cost coefficient or, alternatively, to the income coefficient itself. Let $\beta_{k,c}$ be an entry in $\boldsymbol{\beta}_c$ that is the utility coefficient on attribute k. The WTP for attribute k can be evaluated as $-1 \times \beta_{k,c}/\beta_{\text{cost},c}$ or $\beta_{k,c}/\beta_{\text{income},c}$, depending on whether the pecuniary attribute measures cost or income.[4]

In the "cost" case, the syntax diagram for `lclogitwtp2` is

`lclogitwtp2, cost(`*varname*`)` [`nonlcom` *nlcom_options*]

Similarly, in the "income" case, the syntax diagram for `lclogitwtp2` is

`lclogitwtp2, income(`*varname*`)` [`nonlcom` *nlcom_options*]

4. Note that the WTP measure is class invariant only when numerator and denominator coefficients are both class invariant. Even when the numerator (denominator) coefficient is constrained to be class invariant, the WTP measure varies from class to class as long as the denominator (numerator) coefficient does.

The required option `cost(`*varname*`)` or `income(`*varname*`)` identifies the pecuniary attribute variable, whose coefficient enters the denominator of the WTP formula. When `lclogit2` estimates are active, `lclogitwtp2` simply reports the implied WTP measures. When `lclogitml2` results are active, it also acts as a wrapper for Stata's `nlcom` command, which it uses to compute standard errors and confidence intervals for the implied WTP measures.

The two optional options are relevant only when `lclogitml2` results are active:

`nonlcom` requests that the command skip the `nlcom` step and report the WTP measures without test statistics. The default is to execute the `nlcom` step.

nlcom_options refers to options of `nlcom`; see [R] **nlcom**.

5 Examples

Just as in `clogit`, both `lclogit2` and `lclogitml2` require that the data y_{njt}, $\boldsymbol{x}_{njt}$, and $\boldsymbol{z}_n$ for each distinct triplet of indices $\{n, j, t\}$ (see section 2 for the notation) be organized into a separate row in the dataset. As an example, consider `transport.dta`, available on the Stata Press website.[5] This fictitious dataset has been generated to imitate a sample of $N = 500$ decision makers choosing from $J = 4$ alternative transport modes (car, public transport, bicycle, or walk) in each of $T = 3$ choice situations. Each choice occasion refers to a different time of the year, so the decision maker's age in decades (`age`), income in $10,000s (`income`), and full- or part-time employment status (`parttime`) may vary from occasion to occasion. Each alternative mode is described by its cost (`trcost`) in dollars and required travel time (`trtime`) in hours. Before we proceed, the contents of `trtime` will be modified to measure savings in travel time relative to walking. Following this change, the coefficient on `trtime` can be interpreted as the marginal utility of one hour saved in travel time relative to walking.

```
. use https://www.stata-press.com/data/r16/transport
(Transportation choice data)
. quietly by id t: replace trtime = trtime[4] - trtime[_n]
```

The first 12 rows of the dataset are displayed below. The variables `id`, `t`, and `alt` identify decision makers ($n = 1, 2, \ldots, 500$), choice occasions ($t = 1, 2, 3$), and alternatives ($j = 1, 2, 3, 4$), respectively. Each row of `choice` is y_{njt}, and each row of `trcost` and `trtime` is $\boldsymbol{x}_{njt}$. Decision maker 1 turns out to be someone who traveled by car in all three occasions. While each row of `age`, `income`, and `parttime` records the decision maker's characteristics, it is repeated only within a choice occasion, not across all data rows associated with the same decision maker. In other words, the row does not make up $\boldsymbol{z}_n$, and the three variables cannot be included in *varlist3* to model membership probabilities. Instead, users may consider interacting each demographic variable with `trcost` and `trtime` and including the interaction terms in *varlist1* or *varlist2*. As Train

5. This is an example dataset used by Stata 16's new `cmxtmixlogit` command, which allows users to fit mixed logit models for panel data. Compared with Hole's (2007c) `mixlogit`, the new command provides more choices for mixing distributions and Monte Carlo integration methods.

(2009, chap. 3) explains, including the interaction terms is equivalent to specifying the utility coefficient on each attribute as a linear function of the demographic variables.

```
. list in 1/12, sepby(t)

     +---------------------------------------------------------------------------------+
     | id   t       alt   choice   trcost   trtime   age   income    parttime |
     |---------------------------------------------------------------------------------|
  1. |  1   1       Car        1     4.14     0.01   3.0        3   Full-time |
  2. |  1   1    Public        0     4.74    -0.29   3.0        3   Full-time |
  3. |  1   1   Bicycle        0     2.76    -0.23   3.0        3   Full-time |
  4. |  1   1      Walk        0     0.92     0.00   3.0        3   Full-time |
     |---------------------------------------------------------------------------------|
  5. |  1   2       Car        1     8.00     0.25   3.2        5   Full-time |
  6. |  1   2    Public        0     3.14     0.27   3.2        5   Full-time |
  7. |  1   2   Bicycle        0     2.56     0.21   3.2        5   Full-time |
  8. |  1   2      Walk        0     0.64     0.00   3.2        5   Full-time |
     |---------------------------------------------------------------------------------|
  9. |  1   3       Car        1     1.76     0.41   3.4        5   Part-time |
 10. |  1   3    Public        0     2.25     0.09   3.4        5   Part-time |
 11. |  1   3   Bicycle        0     0.92    -0.47   3.4        5   Part-time |
 12. |  1   3      Walk        0     0.58     0.00   3.4        5   Part-time |
     +---------------------------------------------------------------------------------+
```

Like `clogit`, both `lclogit2` and `lclogitml2` require a variable that identifies all data rows associated with each distinct pair of decision maker n and choice occasion t. As the first command line below shows, such a variable can be generated using the `egen` command's `group()` function. To include alternative-specific intercepts in the LCL model, we create in the second command line alternative-specific constants. Variable `asc1` is set to 1 in all data rows for car and 0 everywhere else. Variables `asc2`, `asc3`, and `asc4` are similarly defined in relation to public transport, bicycle, and walk, respectively. The last variable will be excluded from the model to achieve identification.

```
. egen gid = group(id t)
. quietly tabulate alt, generate(asc)
```

How many classes, C, should LCL allow for? In many empirical studies, including my own (Yoo and Doiron 2013; Doiron and Yoo 2017, 2020; Oviedo and Yoo 2017), this question is addressed by repeatedly fitting the same LCL model with different numbers of classes and inspecting which number leads to the best model in terms of the BIC. `lclogit2` calculates and stores the fitted model's BIC in `e(bic)`, facilitating this specification search.[6]

6. The command also stores the Akaike information criterion (AIC) in `e(aic)` and the consistent Akaike information criterion (CAIC) in `e(caic)`. AIC equals $-2\ln L + 2m$, where $\ln L$ is the maximized sample log likelihood and m is the total number of estimated parameters, that is, linearly independent coefficients in $\boldsymbol{B}$ and $\boldsymbol{\Theta}$ in (3). BIC and CAIC penalize inclusion of extra parameters using penalty functions that increase in the number of decision makers, N: BIC $= -2\ln L + m\ln N$ and CAIC $= -2\ln L + m(1+\ln N)$. In my own experience, BIC and CAIC often favor the same number of classes. AIC almost always prefers more classes than BIC, but I have often found the convergence of AIC-preferred models dubious: their log-likelihood function is often not concave at the supposed maximum.

The `lclogit2` example below shows that BIC is 2316.537 with two classes. The two-class model appears to be an optimal model for this fictitious dataset. While not reported, raising the number of classes (in `ncl()`) to three slightly worsens BIC to 2318.831, and raising it further to four results in numerical convergence problems. For each class c, the output table reports the estimates of utility coefficients $\boldsymbol{\beta}_c$ and membership probability (that is, class share) $\pi_{nc}(\boldsymbol{\Theta})$. Users interested in the estimates of $\boldsymbol{\Theta}$ can inspect the full coefficient vector stored in `e(b)`. In the present application, because $\boldsymbol{z}_n$ includes only the constant regressor (that is, *varlist3* is empty), $\pi_{nc}(\boldsymbol{\Theta})$ is the same across all decision makers. If $\pi_{nc}(\boldsymbol{\Theta})$ varies from decision maker to decision maker, the output table will report sample average class shares.

```
. lclogit2 choice, ncl(2) rand(trcost trtime asc1 asc2 asc3) group(gid) id(id)
> seed(1234)
Iteration 0:  log likelihood = -1235.2979
  (output omitted)
Iteration 38:  log likelihood = -1124.0883
Iteration 39:  log likelihood = -1124.0881
Latent class model with 2 latent classes
```

Variable	Class1	Class2
trcost	-0.421	-1.338
trtime	1.127	0.599
asc1	5.213	4.769
asc2	2.185	2.567
asc3	1.265	0.851
Class Share	0.528	0.472

```
Note: Model estimated via EM algorithm
. display e(bic)
2316.537
```

To obtain standard errors for the `lclogit2` estimates, users can pass the estimates through to `lclogitml2` as initial values, as shown below. In the `lclogitml2` output table, equations `Class1`, `Class2`, and `Share1` correspond to $\boldsymbol{\beta}_1$, $\boldsymbol{\beta}_2$, and $\boldsymbol{\theta}_1$, respectively.[7] The estimation results are stored in Stata's memory under the name `ML_2` to be recalled later in other examples.

7. Users can request that the `lclogitml2` results be reported in the `lclogit2` output table by typing `lclogit2` (without any other input) at the command prompt while `lclogitml2` results are active.

```
. matrix start = e(b)

. lclogitml2 choice, ncl(2) rand(trcost trtime asc1 asc2 asc3) group(gid)
> id(id) from(start)
Iteration 0:   log likelihood = -1124.0881
  (output omitted)
Iteration 3:   log likelihood = -1124.0873
Latent class model with 2 latent classes
```

choice	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Class1						
trcost	-.4226611	.0742005	-5.70	0.000	-.5680915	-.2772308
trtime	1.118927	.5770978	1.94	0.053	-.0121644	2.250017
asc1	5.213921	.6156628	8.47	0.000	4.007244	6.420597
asc2	2.184848	.5333783	4.10	0.000	1.139446	3.230251
asc3	1.263461	.5262117	2.40	0.016	.2321045	2.294817
Class2						
trcost	-1.341305	.1165183	-11.51	0.000	-1.569677	-1.112934
trtime	.5990337	.1978343	3.03	0.002	.2112856	.9867817
asc1	4.769738	.3333981	14.31	0.000	4.116289	5.423186
asc2	2.570557	.2447134	10.50	0.000	2.090928	3.050187
asc3	.8512455	.1683559	5.06	0.000	.521274	1.181217
Share1						
_cons	.1174352	.1941178	0.60	0.545	-.2630287	.4978992

```
. estimates store ML_2
```

Note that in the present example, `lclogitml2` manages to locate a slightly higher sample log likelihood than `lclogit2`, even though theoretically the EM algorithm should have located a local maximum. This type of numerical difference may arise because the default of `lclogit2` is to declare convergence when the relative increase in the log likelihood is smaller than `ltolerance()` (see section 3), whereas `lclogitml2` uses Stata's gradient-based optimizers, which apply a more strict set of convergence criteria (see the help file for `ml maximize`). The `tolcheck` option of `lclogit2`, which was not available for `lclogit`, requests that the EM algorithm add the relative change in the coefficient vector as another criterion. Users who favor numerical accuracy over computational speed may execute `lclogit2` with `tolcheck` to minimize, if not eliminate, the numerical difference.[8]

The new postestimation tool `lclogitwtp2` allows users to convert the utility coefficients for `Class1` and `Class2` into their monetary equivalents or WTP measures. Because `trcost` measures the cost of each transport mode, the marginal utility of money is given by $-1 \times$ its coefficient. Thus, `lclogitwtp2` must be executed with `cost(trcost)`, instead of `income(trcost)`, as the required option. The output is displayed below and includes standard errors and confidence intervals produced by `nlcom` because the active

8. Based on my experience, if users plan on using `lclogit2` as a tool to obtain initial values for `lclogitml2`, the use of `tolcheck` is unlikely to alter the final results. Even without this option, `lclogit2` can find a solution that is close to a local maximum, so toggling on `tolcheck` does not affect which maximum `lclogitml2` finally converges to.

results are for `lclogitml2`.[9] Had the active results been for `lclogit2` instead, only the first table in the output would have been displayed. The coefficient on `trtime` in each class measures how much (in dollars) each person in that class is willing to pay for one hour saved in travel time relative to walking. To test a hypothesis involving two or more WTP coefficients, users may execute `lclogitwtp2` with `nlcom`'s `post` option and then use the `test` command.

```
. lclogitwtp2, cost(trcost)
Willingness-to-pay (WTP) coefficients
```

WTP for	Class1	Class2
trtime	2.647	0.447
asc1	12.336	3.556
asc2	5.169	1.916
asc3	2.989	0.635

```
Please wait: -nlcom- is calculating standard errors for the WTP coefficients.
   C1_trtime:  _b[Class1:trtime] / (-1 * _b[Class1:trcost])
     C1_asc1:  _b[Class1:asc1] / (-1 * _b[Class1:trcost])
     C1_asc2:  _b[Class1:asc2] / (-1 * _b[Class1:trcost])
     C1_asc3:  _b[Class1:asc3] / (-1 * _b[Class1:trcost])
   C2_trtime:  _b[Class2:trtime] / (-1 * _b[Class2:trcost])
     C2_asc1:  _b[Class2:asc1] / (-1 * _b[Class2:trcost])
     C2_asc2:  _b[Class2:asc2] / (-1 * _b[Class2:trcost])
     C2_asc3:  _b[Class2:asc3] / (-1 * _b[Class2:trcost])
```

choice	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
C1_trtime	2.647337	1.474857	1.79	0.073	-.2433292	5.538003
C1_asc1	12.33594	2.028533	6.08	0.000	8.360084	16.31179
C1_asc2	5.169268	1.391943	3.71	0.000	2.441109	7.897426
C1_asc3	2.989299	1.301158	2.30	0.022	.439076	5.539522
C2_trtime	.446605	.1479973	3.02	0.003	.1565356	.7366744
C2_asc1	3.556041	.250291	14.21	0.000	3.06548	4.046603
C2_asc2	1.916459	.150539	12.73	0.000	1.621408	2.21151
C2_asc3	.6346396	.1225004	5.18	0.000	.3945432	.8747359

Both `lclogit2` and `lclogitml2` allow users to impose any set of linear constraints, defined by Stata's `constraint` command in the usual manner. The constraints may apply within the same class, as well as between different classes. In contrast, `lclogit` can incorporate within-class constraints only and has peculiar syntax requirements for inputting the constraints.[10] The `lclogitml2` example below constrains the coefficient on `trcost` to be the same across class 1 and class 2. The output is omitted from reporting because it is identical in substance to another output example to follow immediately.

9. Hole's (2007a) `wtp` allows users to choose from three different approaches to computing confidence intervals for WTP that have been described in Hole (2007b). By acting as a wrapper for `nlcom`, `lclogitwtp2` adopts the first of the three approaches, known as the delta method.
10. See Pacifico and Yoo (2013) for further information.

```
. constraint 1 [Class1]trcost = [Class2]trcost
. estimates restore ML_2
(results ML_2 are active now)
. matrix start = e(b)
. lclogitml2 choice, ncl(2) rand(trcost trtime asc1 asc2 asc3) group(gid)
> id(id) from(start) constraint(1)
  (output omitted)
```

In a two-class model, constraining a coefficient to be the same across class 1 and class 2 is equivalent to making that coefficient class invariant. Users can introduce class-invariant coefficients more conveniently by moving relevant attribute variables from *varlist2* in `rand()` to *varlist1* as illustrated below. The required option `rand()` and associated distinction between *varlist1* and *varlist2* are irrelevant to `lclogit`. The older command assumes that all coefficients vary from class to class and expects all attribute variables to be specified in the position of *varlist1*.

```
. estimates restore ML_2
(results ML_2 are active now)
. lclogitml2 choice trcost, ncl(2) rand(trtime asc1 asc2 asc3) group(gid)
> id(id) continue
Iteration 0:   log likelihood = -1237.2847  (not concave)
  (output omitted)
Iteration 7:   log likelihood = -1145.5241
Latent class model with 2 latent classes
```

choice	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Class1						
trtime	.5789143	.1667984	3.47	0.001	.2519954	.9058331
asc1	3.829321	.2540867	15.07	0.000	3.33132	4.327321
asc2	1.998434	.1799878	11.10	0.000	1.645664	2.351203
asc3	.6756294	.1481346	4.56	0.000	.385291	.9659679
Class2						
trtime	1.448878	.8711226	1.66	0.096	-.2584905	3.156247
asc1	8.841724	.9494208	9.31	0.000	6.980894	10.70255
asc2	3.434056	.86774	3.96	0.000	1.733316	5.134795
asc3	1.995432	.8943154	2.23	0.026	.242606	3.748258
Share1						
_cons	.235064	.160667	1.46	0.143	-.0798374	.5499655
Fix						
trcost	-.9392836	.0506779	-18.53	0.000	-1.038611	-.8399567

The EM algorithm used by `lclogit2` fits an unconstrained model faster than a more parsimonious model that includes class-invariant coefficients or other types of between-class constraints on utility coefficients (Fiebig and Yoo 2019).[11] As usual, the `ml maximize` techniques used by `lclogitml2` tend to fit constrained models faster than unconstrained models, and users may therefore consider the sequence of estimation runs above as the default approach: using `lclogit2` to fit an unconstrained model and then feeding the unconstrained estimates as starting values to `lclogitml2`, which imposes desired constraints. When the constrained maximum is far away from the unconstrained maximum, the default approach may result in convergence failure. In such cases, users may let `lclogit2` impose the constraints despite the resulting slowdown and exploit the EM algorithm's numerical stability to locate the constrained maximum.

The new `lclogit2` and `lclogitml2` commands take advantage of Mata and can reduce computer run times considerably relative to their predecessors, especially when there are many estimated parameters. On a Windows 10 laptop with Intel i5-8250U CPU and 16 GB RAM, for example, the new commands can fit the unconstrained two-class model above almost twice as quickly as their predecessors: the `lclogit2` run achieves convergence in about 11 seconds, and the subsequent `lclogitml2` run in 9 seconds, whereas the equivalent `lclogit` and `lclogitml` runs take about 24 and 17 seconds, respectively. The run-time difference becomes more perceptible when the number of classes is increased to 3: the `lclogit2` and `lclogitml2` runs take about 75 and 30 seconds, whereas the `lclogit` and `lclogitml` runs take about 160 and 70 seconds. Of course, using Mata does not alter the fact that fitting a finite mixture model like LCL is a computer-intensive task. `lclogit2` and `lclogitml2` estimation runs in authentic applications (as opposed to the present application using an example dataset) may still take several hours, if not days, of computer time.[12]

6 Applications to other types of logit models

As explained by Cameron and Trivedi (2005, 498) and reiterated by Yan and Yoo (2019), the conditional logit (`clogit` in Stata) formula inside the big parentheses of (1) nests binary logit (`logit`) and multinomial logit (`mlogit`) formulas as special cases. Thus,

11. The EM algorithm fits an unconstrained model by fitting C separate `clogit` models to compute parameters $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_C$. This class-by-class estimation approach, however, becomes no longer viable when some constraints apply between different classes. For example, whenever there is at least one class-invariant coefficient, the EM algorithm must carry out a computationally demanding task of estimating all C utility coefficient vectors simultaneously. Train (2009, 308) reports a similar drawback of the Bayesian procedure for fitting mixed logit models; the procedure achieves convergence much faster when the model involves only random coefficients than when it involves a combination of random and nonrandom coefficients.

12. Doiron and Yoo (2020) report a four-class latent class model for a sample of 234 individuals making choices in a collective total of 11,208 occasions. The model specification was more specialized than LCL because it incorporated a variant of LCL known as latent class heteroskedastic rank-ordered logit (LHROL) (Yoo and Doiron 2013), but the estimation routine was based on essentially the same Mata codes as `lclogit2` and `lclogitml2`. On a Windows 7 desktop with Intel i7-4790 CPU and 32 GB RAM, estimating 291 parameters of the four-class model took 21 hours at the EM algorithm step and an additional 110 hours at the subsequent gradient-based optimization step that used `technique(nr)`.

in principle, users can use `clogit` to obtain the same estimation results as `logit` and `mlogit`. In practice, this requires reorganization of data beforehand. In the `reshape` command's vernacular, `clogit` requires that the data be in "long" form, with multiple rows per each group identified by `group()`, whereas `logit` and `mlogit` require that the data be in "wide" form, with one row per each group. Adkins (2011) provides a detailed Stata example showing how to reorganize `logit` and `mlogit` data for the `clogit` analysis, which he attributes to Cameron and Trivedi (2010).

`lclogit2` and `lclogitml2` can estimate latent class extensions of `logit` and `mlogit` once the data have been suitably reorganized in accordance with Adkins's (2011) example. Stata 15 introduced a new command, `fmm`, that can fit latent class extensions of several baseline models, including `logit` and `mlogit`.[13] For cross-sectional data ($T = 1$), the latent class `logit` and `mlogit` models that `lclogit2` and `lclogitml2` fit are equivalent to what `fmm` fits. But `fmm` cannot fit models for panel data ($T \geq 2$) that consider preference class membership as the decision maker's time-invariant characteristic, that is, models that assume that someone from class c has the utility coefficient vector of that class throughout all time periods or choice occasions.[14] `lclogit2` and `lclogitml2` can fit such panel models once a variable identifying decision makers has been specified in the option `id()`.

Some stated preference surveys ask the decision makers to rank order all alternatives from most to least preferred, instead of simply asking them to choose their most preferred alternative. A popular baseline model for analyzing rank-ordered data is the rank-ordered logit (`rologit`) model.[15] Suppose that the decision maker rank orders three different jobs described by salary, availability of on-site parking (1 for abundant and 0 for limited), and full-time contract status (1 for yes and 0 for no).[16] The data organization example below satisfies `rologit`'s requirements, and the dependent variable `rank` shows that the decision maker's most preferred job is job A (`rank` = 3) and least preferred job is job B (`rank` = 1), with job C coming in between (`rank` = 2).[17]

13. As of Stata 16, `fmm` cannot fit the LCL model, because it does not support `clogit` as a component model. But `fmm` supports another type of logit model known as ordered logit (`ologit`). `clogit` and `ologit` are nonnested models, albeit both of them nest `logit` as special cases: there is no data reorganization trick that allows users to apply `clogit` to replicate `ologit` results. Consequently, `lclogit2` and `lclogitml2` cannot estimate latent class extensions of `ologit`, whereas `fmm` can.
14. The assumption of time-invariant class membership parallels how unobserved individual heterogeneity is handled in continuous mixture models such as random-effects probit (`xtprobit`) and panel-data mixed logit (`cmxtmixlogit`). The latent dependent variable model for `xtprobit` assumes that the intercept randomly varies across decision makers but remains constant within a decision maker. The latent dependent variable model for `cmxtmixlogit` assumes that the utility coefficients randomly vary across decision makers but remain constant within a decision maker.
15. In Stata 16, `rologit` was renamed to `cmrologit`. Because of slight variations to the syntax diagrams, my discussion refers to `rologit`.
16. This example is motivated by Yoo and Doiron (2013) and Doiron and Yoo (2020), who analyze rank-ordered data on entry-level nursing jobs at Australian hospitals. In the actual data, each job is described by salary and 11 nonsalary attributes that include, inter alia, parking availability, full-time contract status, the hospital's reputation, and opportunities for professional development.
17. The default assumption of `rologit` is that a higher level of `rank` indicates a more preferred alternative.

```
. list, sepby(group)

     ┌──────────────────────────────────────────────────────┐
     │ group   rank     job   salary   parking   fulltime   │
     ├──────────────────────────────────────────────────────┤
  1. │     1      3   [1] A     1500         1          1   │
  2. │     1      1   [2] B     2500         1          0   │
  3. │     1      2   [3] C     2000         0          1   │
     └──────────────────────────────────────────────────────┘
```

As Train (2009, 156–158) points out, `rologit` is so closely related to `clogit` that users may apply `clogit` to replicate `rologit`, and users may apply the extensions of `clogit` such as `mixlogit` to estimate the corresponding extensions of `rologit`. It follows that users can use `lclogit2` and `lclogitml2` to fit what Yoo and Doiron (2013) call the latent class rank-ordered logit model.[18] This requires that the rank-ordered data be reorganized in a way that allows `clogit` to replicate `rologit`. Under `rologit`, the probability of ranking job A, job C, and job B as best, second-best, and worst, respectively, is given by a product of two `clogit` probabilities: the probability of choosing job A from {A, B, C} and that of choosing job C from {B, C}.[19] Therefore, in Train's vernacular, the rank-ordered data above can be "exploded" into data on two "pseudochoices", where the first pseudochoice is made from {A, B, C} and the second pseudochoice is made from {B, C}. The command block below explodes the rank-ordered data as suggested and displays the resulting pseudochoice data that satisfy the data organization requirements of `clogit`.

```
. generate choice = [rank == 3]
. expand 2, generate(sbest)
(3 observations created)
. drop if rank == 3 & sbest == 1
(1 observation deleted)
. replace choice = [rank == 2] if sbest == 1
(1 real change made)
. egen gid = group(group sbest)
```

18. Yoo and Doiron (2013) also describe a variant of latent class rank-ordered logit called LHROL, which accounts for the notion that decision makers may find it easier (or harder) to tell what their best alternative is than what their second-best alternative is. A command for estimating LHROL is available on the *Canadian Journal of Economics* website for Doiron and Yoo (2020). The command does not come with any help file, but it shares similar syntax diagrams with `lclogit2` and `lclogitml2`.
19. In general, when there are J alternatives, a `rologit` probability is given by a product of $J-1$ `clogit` probabilities. The component `clogit` probabilities are the probability of choosing the best from all J alternatives; that of choosing the best from $J-1$ alternatives excluding the best; that of choosing the best from $J-2$ alternatives excluding the first and second best; and so on.

```
. list, sepby(gid)
```

	group	rank	job	salary	parking	fulltime	choice	sbest	gid
1.	1	3	[1] A	1500	1	1	1	0	1
2.	1	1	[2] B	2500	1	0	0	0	1
3.	1	2	[3] C	2000	0	1	0	0	1
4.	1	1	[2] B	2500	1	0	0	1	2
5.	1	2	[3] C	2000	0	1	1	1	2

Given several pseudochoice data blocks organized as above, `clogit`, which replicates `rologit`, must be executed with the option `group(gid)` so that Stata can correctly identify data rows to be used in evaluating each `clogit` probability. `lclogit2` and `lclogitml2` must be executed with the options `group(gid)` and `id(group)`, where the variable `group` in the option `id()` allows Stata to recognize that the utility coefficients remain invariant across all pseudochoice situations arising from the same choice situation. If more than one choice situation is observed per decision maker, `id()` can be altered to specify a variable that identifies individual decision makers instead.

There is a well-known variant of rank ordering known as best–worst scaling (BWS) (Louviere, Flynn, and Marley 2015). In an "object case" BWS task, the decision maker examines a set of attributes, say, {salary, parking, contract type}, and states which of those attributes are the most important (best) and least important (worst) to his or her decision making. A popular baseline model for analyzing object case BWS data is the maximum-difference (max-diff) logit model. Once its psychological foundations are stripped away, the max-diff logit model is algebraically identical to `clogit`, meaning that users can apply `lclogit2` and `lclogitml2` to fit what Yoo and Doiron (2013) call the latent class max-diff logit model. Specifically, when there are K attributes, the max-diff logit model is algebraically identical to a `clogit` model defined over $K \times (K-1)$ alternatives, where each alternative is a particular two-permutation of the K attributes, that is, a distinct candidate pair of the best and worst attributes. To facilitate the max-diff analysis, we may organize the BWS data for the three-attribute example as follows:

```
. list, sepby(group)
```

	group	choice	salary	parking	contract
1.	1	0	1	-1	0
2.	1	0	1	0	-1
3.	1	0	-1	1	0
4.	1	1	0	1	-1
5.	1	0	-1	0	1
6.	1	0	0	-1	1

In the present example, each data row describes one of the $3 \times 2 = 6$ candidate best–worst pairs. An attribute takes a value of 1 in the row where it makes up the most important or "best" element of the pair and -1 where it makes up the least important or "worst" element. The decision maker's BWS response appears in row 4, where the dependent variable `choice` takes a value of 1 and attributes `parking` and `contract` take values of 1 and −1, respectively; the decision maker has stated that parking is the best attribute and contract type the worst attribute. Given several BWS data blocks organized in this way, the max-diff logit model can be fit by running a `clogit` regression of `choice` on any $K-1=2$ out of the $K=3$ attributes, where one attribute is omitted to achieve identification and the option `group(group)` must be specified to identify choice situations. `lclogit2` and `lclogitml2` can be used to extend the baseline `clogit` model as usual. Note that the `clogit` index ($\boldsymbol{x}_{njt}\boldsymbol{\beta}$ in section 2) for each row is now the best–worst utility difference of the pair that it describes, for example, $\beta_{\texttt{parking}} - \beta_{\texttt{contract}}$ in row 4. The term "maximum difference" refers to the assumption that the decision maker chooses the pair that maximizes the best–worst utility difference.

Another type of BWS known as "profile case" BWS is identical to the object case, except that each attribute in question is associated with a particular level descriptor. For example, the decision maker may examine and state the best and worst out of three attribute levels, {salary of $2,000, limited on-site parking, part-time contract}.[20] The max-diff logit model for this type of response is algebraically identical to `clogit` too, and the data can be organized similarly to the object case example. For a full example of how to organize profile case data, see the *Canadian Journal of Economics* website for Doiron and Yoo (2020).

`lclogit2` and `lclogitml2` assume that the LCL model has been specified in what Train and Weeks (2005) classify as the "preference space". Each estimated coefficient on an attribute is a utility coefficient, and `lclogitwtp2` should be used to obtain the corresponding WTP measure. An alternative approach is to reparameterize the model in the "WTP space" by specifying the sample log likelihood directly as a function of the WTP measures. Hole's (2007c, 2015) `mixlogit` and `mixlogitwtp` commands allow users to fit multivariate normal mixture logit models in the preference space and WTP space, respectively. The two commands lead to substantively different estimation results because, as explained by Train and Weeks (2005), multivariate normal utility coefficients do not imply multivariate normal WTP measures, and vice versa, unless the marginal utility of money is constant across all decision makers.[21]

In the context of a finite or discrete mixture logit model, which LCL is, whether users fit the model in one space or another is less critical. As Oviedo and Yoo (2017) point out, the set of mass points in a discrete mixing distribution that maximizes the

20. Yoo and Doiron (2013) and Doiron and Yoo (2020) provide further information on identification and interpretation of the max-diff logit models' coefficients and their comparisons with the traditional `clogit` utility coefficients. Neither the max-diff logit model nor the BWS elicitation method is our own contribution, though I believe that empirical economists may find our exposition more accessible than other comparable sources. The statistical and data-collection methods originate from a series of articles by Louviere, Flynn, and Marley, which is referenced in their book (Louviere, Flynn, and Marley 2015).

21. A brief explanation for the difference between the preference space and WTP space results would be that a WTP measure is a ratio of two utility coefficients, and a ratio of two normal random variables is not a normal random variable.

sample log-likelihood function is invariant to whether the model is parameterized in the preference space or the WTP space. Therefore, the WTP measures derived from the utility coefficients (using `lclogitwtp2`) are the same as what users would have obtained if they reparameterized the model to fit the WTP measures directly.

7 Acknowledgments

I thank an anonymous reviewer for helpful comments and constructive suggestions. I also thank users of `lclogit` for their feedback on the older command.

8 Programs and supplemental materials

To install a snapshot of the corresponding software files as they existed at the time of publication of this article, type

```
. net sj 20-2
. net install st0601     (to install program files, if available)
. net get st0601         (to install ancillary files, if available)
```

9 References

Adkins, L. C. 2011. Alternative specific logit. http://www.learneconometrics.com/class/6243/notes/Alternative Specific Logit.pdf.

Bhat, C. R. 1997. An endogenous segmentation mode choice model with an application to intercity travel. *Transportation Science* 31: 34–48. https://doi.org/10.1287/trsc.31.1.34.

Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.

———. 2010. *Microeconometrics Using Stata*. Rev. ed. College Station, TX: Stata Press.

Doiron, D., and H. I. Yoo. 2017. Temporal stability of stated preferences: The case of junior nursing jobs. *Health Economics* 26: 802–809. https://doi.org/10.1002/hec.3350.

———. 2020. Stated preferences over job characteristics: A panel study. *Canadian Journal of Economics* 53: 43–82. https://doi.org/10.1111/caje.12431.

Fiebig, D. G., and H. I. Yoo. 2019. Econometrics of stated preferences. In *The Oxford Encyclopedia of Health Economics*, ed. A. M. Jones. Oxford: Oxford University Press. https://doi.org/10.1093/acrefore/9780190625979.013.92.

Hole, A. R. 2007a. wtp: Stata module to estimate confidence intervals for willingness to pay measures. Statistical Software Components S456808, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s456808.html.

———. 2007b. A comparison of approaches to estimating confidence intervals for willingness to pay measures. *Health Economics* 16: 827–840. https://doi.org/10.1002/hec.1197.

———. 2007c. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401. https://doi.org/10.1177/1536867X0700700306.

———. 2015. mixlogitwtp: Stata module to estimate mixed logit models in WTP space. Statistical Software Components S458037, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s458037.html.

Louviere, J. J., T. N. Flynn, and A. A. J. Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge: Cambridge University Press.

Oviedo, J. L., and H. I. Yoo. 2017. A latent class nested logit model for rank-ordered data with application to cork oak reforestation. *Environmental and Resource Economics* 68: 1021–1051. https://doi.org/10.1007/s10640-016-0058-7.

Pacifico, D., and H. I. Yoo. 2013. lclogit: A Stata command for fitting latent-class conditional logit models via the expectation-maximization algorithm. *Stata Journal* 13: 625–639. https://doi.org/10.1177/1536867X1301300312.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2002. Reliable estimation of generalized linear mixed models using adaptive quadrature. *Stata Journal* 2: 1–21. https://doi.org/10.1177/1536867X0200200101.

Revelt, D., and K. Train. 1998. Mixed logit with repeated choices: Households' choices of appliance efficiency level. *Review of Economics and Statistics* 80: 647–657. https://doi.org/10.1162/003465398557735.

Train, K., and M. Weeks. 2005. Discrete choice models in preference space and willingness-to-pay space. In *Applications of Simulation Methods in Environmental and Resource Economics*, ed. R. Scarpa and A. Alberini, 1–16, 1–16. Dordrecht: Springer.

Train, K. E. 2008. EM algorithms for nonparametric estimation of mixing distributions. *Journal of Choice Modelling* 1: 40–69. https://doi.org/10.1016/S1755-5345(13)70022-8.

———. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. Cambridge: Cambridge University Press.

Yan, J., and H. I. Yoo. 2019. Semiparametric estimation of the random utility model with rank-ordered choice data. *Journal of Econometrics* 211: 414–438. https://doi.org/10.1016/j.jeconom.2019.03.003.

Yoo, H. I., and D. Doiron. 2013. The use of alternative preference elicitation methods in complex discrete choice experiments. *Journal of Health Economics* 32: 1166–1179. https://doi.org/10.1016/j.jhealeco.2013.09.009.

About the author

Hong Il Yoo is an associate professor in economics at Durham University Business School, Durham University, UK.