xlogit: An Open-Source `Python` Package for GPU-Accelerated Estimation of Mixed Logit Models

Cristian Arteaga[a], JeeWoong Park[a,*], Prithvi Bhat Beeramoole[b], Alexander Paz[b]

[a] *Department of Civil and Environmental Engineering, University of Nevada Las Vegas, Las Vegas, NV 89154*

[b] *Department of Civil Engineering and Built Environment, Queensland University of Technology, Brisbane, Queensland 4000*

Abstract

Mixed Logit is an advanced and flexible tool for the study of discrete choice problems. However, this flexibility involves computationally intensive calculations, as the estimation of Mixed Logit models requires the simulation of integrals. In addition, the specification of Mixed Logit models requires decisions such as potential explanatory variables to be included in the model as well as their mixing distributions. This specification process involves testing and estimation of different combinations of variables and mixing distributions, which is time consuming and computationally intensive. In response, this paper introduces *xlogit*, an open-source `Python` package that leverages the performance of graphic processing units (GPU) for an efficient estimation of Mixed Logit models. For benchmarking, the performance of *xlogit* was compared against the *PyLogit* and *Biogeme* `Python` packages as well as the *mlogit*, *Apollo*, *gmnl*, and *mixl* `R` packages. Artificially generated as well as actual data were used to evaluate the performance gains provided by *xlogit*. Results suggest that using a mid-range graphics card and a regular desktop computer, *xlogit* is in average 55x faster than *Apollo*, 43x faster than *Biogeme*, 74x faster than *gmnl*, 39x faster than *mixl*, 16x faster than *mlogit*, and 27x faster than *PyLogit*, with an additional advantage of efficient memory management. The performance gains provided by *xlogit* facilitate an efficient modeling process, as it enables the testing of a large number of model specifications more efficiently relative to existing software packages. *xlogit*'s open source code, documentation, and usage examples are publicly available in the package's GitHub repository.

Keywords: mixed logit, Python, GPU acceleration, discrete choice, econometrics

---

1. Introduction

Discrete choice models are widely applied in many contexts, including analysis of consumer preferences, travel behavior, and traffic crash severity (Train, 2003; Ben-Akiva and Lerman, 1985). Mixed Logit is one of the most prominent techniques for discrete choice modeling because of its flexibility and ability to approximate any random utility specification (McFadden and Train, 2000). They allow a flexible error structure and unrestricted

*Corresponding author

Email address: `jee.park@unlv.edu` (JeeWoong Park)

substitution patterns. Unlike Probit models, Mixed Logit models are not constrained to normal parameter distributions, which enables a wider range of applications by providing extra flexibility. Various applications of Mixed Logit models include taste heterogeneity in mode choice behavior (Vij and Krueger, 2017), health study preferences (Raspa et al., 2020), neighborhood choice (Wang et al., 2020), and road crash analyses (Intini et al., 2020). Despite the advantages of Mixed Logit models, there are significant difficulties in effectively developing and applying these models in a time-efficient manner. In the process of specifying these models, a series of assumptions must be made and subsequently verified. Hence, it is important to develop tools that can help analysts to efficiently test a large number of model specifications.

Several proprietary and open-source tools are available for the estimation of Mixed Logit models (Mariel et al., 2021). Popular proprietary tools include `STATA` (StataCorp, 2019) and `NLogit` (Greene, 2012), and well-known open-source tools include the *PyLogit* `Python` package (Brathwaite and Walker, 2018), the *Biogeme* `Python` package (Bierlaire, 2020), the *mlogit* `R` package (Croissant, 2020), the *Apollo* `R` package (Hess and Palma, 2019), the *mixl* `R` package (Molloy et al., 2021), and the *gmnl* `R` package (Sarrias and Daziano, 2017). The discussion and benchmark conducted in this paper focuses on open-source tools, given that the proprietary tools have licensing requirements for usage. Existing open-source tools offer a rich set of features for specification of Mixed Logit models, such as the ability to work with panel data, handling of unbalanced panels, and inclusion of individual and alternative specific variables. In addition, these open-source tools allow the incorporation of several types of mixing distributions, such as normal, log-normal, triangular, and uniform, except for *PyLogit*, which only allows normal mixing distributions.

Although the existing open-source tools for the estimation of Mixed Logit models have significantly contributed to the research and practice in statistical analyses, these tools are limited by their running times, as Mixed Logit models are computationally intensive. This can be a strong limiting factor in several scenarios. First, slow estimation times can be inconvenient when analysts need to test several model specifications before deciding on a final specification. Second, the amount of data available for analysis have significantly increased due to the extensive adoption of technology and telecommunications. Therefore, data analysis tools need to evolve to keep up with the increasing demand in volume processing capacity. Third, despite the widespread practice of estimating models using only a few hundred random draws, past studies have highlighted the importance of using a larger number of draws (Czajkowski and Budziński, 2019; Chiou and Walker, 2007), as failing to do so may hide serious identification problems. Given that the use of many random draws can be important in model estimation, estimation efficiency becomes a crucial factor, which is one of the aspects the developed *xlogit* package seeks to enhance. Finally, slow estimation times limit the leveraging of recently proposed approaches for the assisted specification of Mixed Logit models, such as the ones proposed by Paz et al. (2019); Ortelli et al. (2020) and Rodrigues et al. (2019), that require iterative testing of a large number of model specifications, which can be excessively time consuming using existing estimation tools.

The existing *mlogit*, *PyLogit*, and *gmnl* packages lack a built-in capability to perform parallel processing to reduce estimation time. On the other hand, the *Apollo*, *Biogeme*, and *mixl* packages can leverage multiple processor threads for parallel processing, which has been shown to help reduce the estimation time (Hess and Palma, 2019). However, despite the multi-thread processing capabilities of *Apollo*, *Biogeme*, and *mixl*, their estimation times are still high, even when using many processor threads, as it will be shown later in the benchmark section. In view of the performance limitations of existing estimation packages, this paper introduces *xlogit* as an alternative estimation tool that leverages GPU processing to significantly speed-up the estimation of Mixed Logit models and escalate to estimations using hundreds of thousands of random draws.

2. Mixed Logit models

2.1. Formulation

Logit-based models, including Mixed Logit, assume that a decision maker chooses an alternative among a set of finite and mutually-exclusive alternatives based on explanatory variables that include attributes of the decision maker and the alternatives. Under the assumptions of the Random Utility Maximization theory (Manski, 1977), the decision maker chooses the alternative that maximizes their utility or benefit. The following notations are used to discuss the mathematical formulation of Mixed Logit models:

n: subscript for a decision maker; $n = \{1, 2, \ldots, N\}$

j: superscript for a choice alternative; $j = \{1, 2, \ldots, J\}$

k: subscript for an explanatory variable; $k = \{1, 2, \ldots, K\}$

x_{nk}^j: value of explanatory variable k, for decision maker n and alternative j.

y_n^j: choice variable equal to one if decision maker n chooses alternative j, zero otherwise.

β_k: coefficient for explanatory variable k.

The linear definition of utility for Logit-based models is given by Equation 1.

$$V_n^j = \beta_1 x_{n1}^j + \beta_2 x_{n2}^j + \cdots + \beta_K x_{nK}^j = \sum_{k=1}^{K} \beta_k x_{nk}^j = \beta x_n^j \tag{1}$$

where V_n^j is the observed utility that a decision maker n obtains from choosing alternative j. βx_n^j represents the linear product between all the coefficients β_k and the explanatory variables x_{nk}^j. The probability P_n^c of an individual n choosing alternative c, is given by Equation 2. Note that this probability is a function of the parameters β. The objective of the estimation process is to find the set of β parameters that best fit the input data.

$$P_n^c(\beta) = \int \frac{e^{\beta x_n^c}}{\sum_{j=1}^{J} e^{\beta x_n^j}} f(\beta) d\beta \tag{2}$$

When decision makers have repeated choices or choices across multiple time periods in a panel structure, the probability of the sequence of choices is expressed as the product of the probabilities at each period t ($t = 1, 2, \ldots T$), as shown in Equation 3 (Train, 2003), where $\beta x_n^{j(t)}$ represents the observed utility at time period t. For simplicity, the remaining equations in this section omit the panel indicator (t).

$$P_n^c(\beta) = \int \prod_{t=1}^{T} \frac{e^{\beta x_n^{j(t)}}}{\sum\limits_{j=1}^{J} e^{\beta x_n^{j(t)}}} f(\beta) d\beta \tag{3}$$

The likelihood, as expressed in Equation 4, is a measure of the goodness of fit of the parameters, computed as the product of the probabilities of the chosen alternatives for all the decision makers in the dataset.

$$\mathcal{L}(\beta) = \prod_{n=1}^{N} \prod_{j=1}^{J} (P_n^j)^{y_n^j} \tag{4}$$

2.2. Estimation

The parameter estimation for discrete choice models is defined as a non-linear minimization problem, using the likelihood function as the optimization target (Bunch, 1987). Gradient-based iterative methods are generally applied under the global concavity assumption for the parameter search. The gradient (first derivatives of the likelihood function) and Hessian (second derivatives of the likelihood function) are critical components of parameter estimation, as these determine the search direction to obtain parameters with higher log-likelihood values. Standard Newton-based methods require the estimation of gradient and Hessian at every iteration, resulting in increased computational time. Therefore, Quasi-Newton methods, which use Taylor series approximation for Hessian estimation, are popularly applied due to their computational efficiency. The methods include BHHH (Berndt et al., 1974), BFGS (Broyden-Fletcher-Goldfarb-Shanno), and Limited-Memory BFGS, as they are found to perform faster than the standard method (Liu and Nocedal, 1989). Irrespective of the method, the estimation process requires the computation of the log-likelihood, which is the objective function of the optimization routine, and the gradient vector. The following subsections provide a discussion of how the log-likelihood and gradient are computed in the context of Mixed Logit models.

2.2.1. Simulated probabilities and log-likelihood

Although the computation of Mixed Logit probabilities involves an integral that cannot be solved analytically, numerical simulation techniques can be applied to approximate this integral (Revelt and Train, 1998). Maximum simulated likelihood relies on the idea that integration over a density is a form of averaging (Train, 2003). Mixed Logit probabilities are simulated by taking multiple random draws R from the assumed distribution of β_k. The simulated probability $\check{P}_n^c$ is estimated by taking an average of the probabilities for each of the draws r, as shown in Equation 5, where β_r represent the coefficients at random draw r.

$$\check{P}_n^c(\beta) = \frac{1}{R} \sum_{r=1}^{R} \frac{e^{\beta_r x_n^c}}{\sum\limits_{j=1}^{J} e^{\beta_r x_n^j}} \tag{5}$$

The simulated likelihood $\check{\mathcal{L}}$ is the same as in Equation 4 after replacing the probability P_n^c by the simulated probability $\check{P}_n^c$, as shown in Equation 6. The log of the likelihood is used instead of the likelihood, given that float point precision issues arise when the computation of the likelihood includes the product of small probability numbers. Using the log of the likelihood translates the products of probabilities to summations.

$$log[\check{\mathcal{L}}(\beta)] = \sum_{n=1}^{N} \sum_{j=1}^{J} y_n^j \cdot log(\check{P}_n^j) \tag{6}$$

2.2.2. Gradient

The gradient function for logit-based models is given by Equation 7 (Train, 2003), which results in Equation 8 after some algebraic manipulations (Hasan et al., 2016).

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_k} = \sum_{n=1}^{N} \sum_{j=1}^{J} y_n^j \frac{\partial \log(\check{P}_n^j)}{\partial \beta_k} \tag{7}$$

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_k} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j \tag{8}$$

In Mixed Logit models, the gradient for the random coefficients β_k is decomposed into its mean (β_{kb}) and standard deviation (β_{kw}). Therefore, depending on the distribution of the random coefficient, the gradient with respect to the β_{kb} and β_{kw} coefficients take different forms as follows:

- A coefficient with **normal distribution** is expressed as $\beta_k = \beta_{kb} + \beta_{kw}\eta$, and the gradients for the β_{kb} and β_{kw} coefficients are shown in Equations 9 and 10, respectively, where η are standard normal random draws.

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_{kb}} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j \tag{9}$$

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_{kw}} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j \eta \tag{10}$$

- A coefficient with **lognormal distribution** is expressed as $\beta_k = e^{\beta_{kb} + \beta_{kw}\eta}$, and the gradients for the β_{kb} and β_{kw} coefficients are shown in Equations 11 and 12, respectively, where η are standard normal random draws.

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_{kb}} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j e^{\beta_{kb} + \beta_{kw}\eta} \tag{11}$$

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_{kw}} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j \eta e^{\beta_{kb} + \beta_{kw}\eta} \tag{12}$$

- A coefficient with **uniform distribution** is expressed as $\beta_k = \beta_{kb} + \beta_{kw}(2\mu - 1)$, and the gradients for the β_{kb} and β_{kw} coefficients are shown in Equations 13 and 14, respectively, where μ are uniform random draws between zero and one.

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_{kb}} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j \tag{13}$$

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_{kw}} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j (2\mu - 1) \tag{14}$$

- A coefficient with **triangular distribution** is expressed as $\beta_k = \beta_{kb} + \beta_{kw}\tau$ and the gradients for the β_{kb} and β_{kw} coefficients are shown in Equations 15 and 16, respectively, where $\tau = \sqrt{2\mu} - 1$ if $\mu < .5$ or $\tau = 1 - \sqrt{2(\mu - 1)}$ if $\mu > .5$ and μ are uniform random draws between zero and one.

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_{kb}} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j \tag{15}$$

$$\frac{\partial log[\check{\mathcal{L}}(\beta)]}{\partial \beta_{kw}} = \sum_{i=n}^{N} \sum_{j=1}^{J} (y_n^j - \check{P}_n^j) x_{nk}^j \tau \tag{16}$$

3. Proposed *xlogit* package

The main motivation behind *xlogit* is to take advantage of the fast parallel computations that GPUs offer, in order to reduce the estimation time for Mixed Logit models and escalate to estimations using hundreds of thousands of random draws. This section introduces the primary features of *xlogit*, explains how it leverages GPU processing for estimation, and describes its installation and usage.

3.1. Features

xlogit offers a comprehensive set of tools for the estimation of Mixed Logit models. However, *xlogit* is not limited to Mixed Logit or random parameter models only, it also supports the estimation of models with fixed parameters. These can be models with only individual specific parameters (i.e., Multinomial Logit models), models with only alternative specific variables (i.e., Conditional Logit models), or models that combine individual and alternative specific parameters. The current version of *xlogit* only supports models that have a linear definition of the deterministic part of the utility function. The following list summarizes the functionalities and types of models that the current version of *xlogit* supports.

- Mixed Logit with several types of mixing distributions (normal, lognormal, triangular, uniform, and truncated normal)

- Mixed Logit with panel data

- Mixed Logit with unbalanced panel data

- Mixed Logit with Halton draws
- Multinomial Logit models
- Conditional Logit models
- Handling of unbalanced availability of choice alternatives for all of the supported models
- Post-estimation tools for prediction and specification testing
- Inclusion of sample weights for all of the supported models
- Batch processing of random draws to easily escalate to model estimation using hundreds of thousands of random draws

For model estimation, *xlogit* uses the Maximum Simulated Likelihood Estimator (MSLE) and the BFGS (Fletcher, 1981) optimization routine. The optimization routine uses both the simulated log-likelihood function expressed in Equation 6 and the analytical gradients from Equations 9 to 16 to estimate the model parameters. *xlogit* uses analytical gradients, instead of finite differences approximations of the gradient, given that this enables efficient estimation by reducing the number of required objective function evaluations.

xlogit's source code is openly available at the GitHub public repository `http://github.com/arteagac/xlogit`. In this repository, it can be observed that *xlogit* has been developed following practices that seek to maximize the quality of the code. Such practices include:

(i) automated code checking with unit tests and continuous integration

(ii) compliance with styling and naming conventions defined by `Python`'s PEP8 standard

(iii) comprehensive code documentation using doc-string conventions

Detailed instructions and several usage examples are provided in the documentation of the package available at `http://xlogit.readthedocs.io`. In addition, a guide to contribute to the project is provided in *xlogit*'s repository, in order to facilitate open-source community contributions.

3.2. Leveraging GPU processing

The large number of processor cores that GPUs contain offer a significant speed advantage over those of CPUs for operations of homogeneous nature on different pieces of data. GPUs are throughput oriented, which means that they are designed for the efficient processing of many similar tasks simultaneously (data parallel) (Matthews, 2018). Although this feature was initially designed for application in computer graphics, several numerical computing tasks, such as linear algebra operations, fast Fourier transforms, random number generation, and solving differential equations, have also benefited from this fast parallel processing (Kindratenko, 2014).

The task of estimating a Mixed Logit model involves several linear algebra operations between the input data and the estimable parameters. The computation of log-likelihood and gradient values are the most time expensive operations, as they require numerous matrix products and aggregations across the random parameter draws. Using a large number of random draws can enhance the estimation results, as this benefits the numerical approximation of the parameters' mixing distributions. However, the increase in random draws imposes additional operations for the log-likelihood and gradient estimations, which consequently, increases the estimation time. To address this issue, *xlogit* leverages the fast parallel computations offered by GPUs to significantly reduce estimation times. In order to efficiently leverage GPU's capabilities, *xlogit* avoids sequential matrix operations, and instead handles extensive matrix products in parallel by expressing these in terms of multidimensional array computations that GPUs can process efficiently. For instance, to compute the log-likelihood for all the individuals in the dataset across multiple random draws, *xlogit* does not sequentially iterate over individuals or random draws but instead it operates the multidimensional arrays of random draws and input data in a single computation expression that is efficiently distributed across all the GPU cores. The current version of *xlogit* only supports GPU processing on NVIDIA CUDA-enabled cards, as CUDA has the most mature and largely adopted platform for general purpose computing with GPUs (Misic et al., 2012).

Continuous developments in GPU manufacturing have significantly enhanced their processing power, while maintaining reasonable prices for general consumers. As of December 2020, there exist several low-cost and even free alternatives to access GPU computing resources. For instance, Google Colaboratory offers free GPU resources for learning purposes in a `Python` environment with no setup required, as the service can be accessed using a web browser (Colab, 2020). Cloud GPU resources are also available at less than $1 USD per hour. For instance, the Google Cloud platform offers GPU processing starting at $0.45 USD per hour for a NVIDIA Tesla K80 GPU with 4,992 CUDA cores (GCloud, 2020). Amazon Sagemaker offers virtual machine instances with the same TESLA K80 GPU at a similar price range (AWS, 2020). For non-cloud or desktop GPU options, there exist a variety of alternatives that do not require an excessively elevated budget. For instance, a low-end, yet powerful, NVIDIA 1650 Super GPU with 1,280 CUDA cores, which can significantly reduce numerical computation times, has a suggested retail price of $160 USD (DelRizzo, 2019). A much more powerful option is the recently released NVIDIA RTX 3080, which has 8,704 CUDA cores and a suggested retail price of $700 USD (NVIDIA, 2020); this is less affordable than other options, but still within the price range of computing devices for the general public. The high power and relatively low cost of GPU computing devices constitute a great alternative to address large numerical computations.

3.3. Installation

xlogit requires `Python` 3.6 or superior, and it can be installed using the *pip* package manager by executing:

```
pip install xlogit
```

The *pip* package manager is usually available as part of `Python`'s installation and can be accessed from the command line of the operating system. *xlogit* has been extensively tested, mostly in Linux and Windows operating systems; however, it is expected to work on any operating system that can provide a fully functional Python environment, such as macOS or BSD.

To enable GPU processing, it is necessary to additionally install the *CuPy* `Python` package (Okuta et al., 2017). Once *xlogit* detects that *CuPy* is properly installed, it automatically enables GPU processing without any additional setup. The installation of *CuPy* is simple, and detailed instructions can be found at this package's website.

3.4. Usage

xlogit provides an intuitive set of commands for model estimation. The following example illustrates the estimation of a Mixed Logit model using *xlogit* on a dataset of 1,182 choices among four sport fishing modes. The goal is to analyze the market shares of four alternatives (i.e., beach, pier, boat, and charter) based on their cost and fish catch rate (Cameron and Trivedi, 2005). Figure 1 illustrates how to use *xlogit* to estimate the model parameters. The data to be analyzed can be imported to `Python` using any preferred method. In this example, the data in CSV format was imported using the popular *pandas* `Python` package (McKinney, 2010) for handling dataframes. However, it is worth highlighting that *xlogit* does not depend on the *pandas* package, as *xlogit* can take any array-like structure as input. This represents an additional advantage because *xlogit* can be used with any preferred dataframe library, and not only with *pandas*.

Once the data is in the `Python` environment, *xlogit* can be used to fit the model, as shown in Figure 1. The `MixedLogit` class is imported from *xlogit*, and its constructor is used to initialize a new model. The `fit` method estimates the model using the input data and estimation criteria provided as arguments to the method's call. The following are the main arguments and data types accepted by the `fit` method. Other arguments, such as the convergence criteria, random draw type, initial coefficient values, and verbosity levels, are described in *xlogit*'s documentation.

- `X`: 2-D array of input data with choice situations as rows, and variables as columns
- `y`: 1-D array of choices
- `varnames`: 1-D array of variable names
- `alts`: 1-D array of alternative indexes or an alternatives list
- `ids`: 1-D array of the ids of the choice situations
- `randvars`: dictionary of variables and their mixing distributions
- `n_draws`: number of random draws to use in the estimation (optional, default 1,000)
- `panels`: 1-D array of indexes for panel formation (optional)
- `avail`: 1-D array of availability of alternatives for the choice situations (optional)

After the model is estimated, the `summary` method can be used to show the model's estimation results. The output of the `summary` method is shown in Figure 1. First, information about the optimization process, such as the convergence status, number of iterations, and estimation time, is presented. Second, a table with coefficient names, estimates, standard

```python
import pandas
df = pandas.read_csv("examples/data/fishing_long.csv")

from xlogit import MixedLogit
varnames = ['price', 'catch']
model = MixedLogit()
model.fit(X=df[varnames], y=df['choice'], varnames=varnames, alts=df['alt'],
          ids=df['id'], n_draws=1000, randvars={'price': 'n', 'catch': 'n'})
model.summary()
```

```
OUTPUT:
Estimation with GPU processing enabled.
Optimization terminated successfully after 42 iterations.
Estimation time= 1.6 seconds
----------------------------------------------------------------
Coefficient      Estimate      Std.Err.         z-val       P>|z|
----------------------------------------------------------------
price          -0.0272460     0.0022981   -11.8561003    1.09e-29 ***
catch           1.3271142     0.1743127     7.6134088    4.01e-13 ***
sd.price        0.0102129     0.0021859     4.6722496    1.59e-05 ***
sd.catch       -1.5706821     0.3762576    -4.1744858    0.000139 ***
----------------------------------------------------------------
Significance:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-Likelihood= -1300.511
AIC= 2609.023
BIC= 2629.323
```

Figure 1: Usage example on fishing dataset

errors, and p-values is provided. Third, goodness of fit metrics, such as the log-likelihood, AIC, and BIC, are displayed.

xlogit also provides a convenient set of post-estimation tools for prediction or forecasting. The `predict` function uses estimated parameters and a new or updated dataset to compute predicted choices. By default, `predict` returns the chosen alternative for each individual in the dataset. By including the `return_freq` parameter, `predict` additionally returns the aggregated market shares for each alternative. Also, by including the `return_proba` parameter, `predict` returns a 2-D array with individuals as rows, and the probability of choosing each alternative as columns. Figure 2 illustrates an example of the `predict` function that uses the model estimated in Figure 1 to forecast changes in market shares (choice frequency) for fishing modes caused by an increase in price for the "boat" mode. First, base market shares are computed by running `predict` on the original dataset. Then, an increase of 20% in the price for the "boat" alternative is applied to the dataset and the updated shares are predicted. The output shows that the 20% price increase would result in a decrease of almost 10% in market share for the "boat" alternative. Besides prediction, users may also need to conduct model specification tests after model estimation. For this purpose, *xlogit* provides the `lrtest` (likelihood ratio test) function, which evaluates whether the observed difference in goodness of fit between two models (one general and one restricted) is statistically significant. Usage examples and additional details and for the `lrtest` function are provided in *xlogit*'s documentation.

```
choices, freq = model.predict(X=df[varnames], varnames=varnames, ids=df['id'],
                              alts=df['alt'], return_freq=True, n_draws=1000)
print(f"base: {freq}")

df.loc[df['alt']=='boat', 'price'] *= 1.2  # 20 percent price increase
choices, freq = model.predict(X=df[varnames], varnames=varnames, ids=df['id'],
                              alts=df['alt'], return_freq=True, n_draws=1000)
print(f"updated: {freq}")


OUTPUT:
base: {'beach': 0.223, 'boat': 0.461, 'charter': 0.228, 'pier': 0.089}
updated: {'beach': 0.238, 'boat': 0.379, 'charter': 0.278, 'pier': 0.105}
```

Figure 2: Example of prediction code

4. Examples

In order to showcase *xlogit*'s features and demonstrate that it provides a comprehensive set of tools for model estimation, this section presents three estimation examples using different datasets: the swissmetro dataset, an artificially generated dataset, and a dataset of electricity supplier choices. The heterogeneous characteristics of these datasets will help to demonstrate that xlogit provides sufficient flexibility to accommodate diverse model specification needs. In addition, each example includes a comparison against estimates from the *Apollo*, *Biogeme*, *mlogit*, and *gmnl* packages, in order to show that *xlogit*'s estimates are consistent with those provided by existing estimation tools.

4.1. Swissmetro dataset

The swissmetro dataset contains stated-preferences for three alternative transportation modes that include car, train and a newly introduced mode: the swissmetro. This dataset is commonly used for estimation examples with the *Biogeme* and *PyLogit* packages. Bierlaire et al. (2001) provides a detailed discussion of the data as wells as its context and collection process. The explanatory variables in this example include the travel time (`TT`) and cost (`CO`) for each of the three alternative modes. This dataset has a panel structure, given that multiple choice situations were presented to each of the survey respondents. Figure 3 shows the code for the estimation that includes four steps: 1) reading the input data, 2) reshaping it to long format, 3) creating a model specification, and 4) running the estimation procedure.

- Step 1. Read and pre-process data: The dataset is imported to the `Python` environment using *pandas*. Then, two types of samples, ones with a trip purpose different to

```python
# ====== STEP 1. READ AND PRE-PROCESS DATA =====
import pandas
import numpy as np
df_wide = pandas.read_table("swissmetro.dat", sep="\t")
df_wide = df_wide[(df_wide['PURPOSE'].isin([1, 3]) \
                  & (df_wide['CHOICE'] != 0))]  # Filter samples
df_wide['custom_id'] = np.arange(len(df_wide))  # Add unique identifier
df_wide['CHOICE'] = df_wide['CHOICE'].map({1: 'TRAIN', 2: 'SM', 3: 'CAR'})

# ===== STEP 2. RESHAPE DATA TO LONG FORMAT =====
from xlogit.utils import wide_to_long
df = wide_to_long(df_wide, id_col='custom_id', alt_name='alt',
                  alt_list=['TRAIN', 'SM', 'CAR'], empty_val=0,
                  varying=['TT', 'CO', 'AV'], alt_is_prefix=True)

# ===== STEP 3. CREATE MODEL SPECIFICATION =====
df['ASC_TRAIN'] = np.ones(len(df))*(df['alt'] == 'TRAIN')
df['ASC_CAR'] = np.ones(len(df))*(df['alt'] == 'CAR')
df['TT'], df['CO'] = df['TT']/100, df['CO']/100  # Scale variables
annual_pass = (df['GA'] == 1) & (df['alt'].isin(['TRAIN', 'SM']))
df.loc[annual_pass, 'CO'] = 0  # Cost zero for pass holders

# ===== STEP 4. ESTIMATE MODEL PARAMETERS =====
from xlogit import MixedLogit
varnames=['ASC_CAR', 'ASC_TRAIN', 'CO', 'TT']
model = MixedLogit()
model.fit(X=df[varnames], y=df['CHOICE'], varnames=varnames,
          alts=df['alt'], ids=df['custom_id'], panels=df["ID"],
          avail=df['AV'], randvars={'TT': 'n'}, n_draws=1500)
model.summary()
```

Figure 3: Estimation code for swissmetro dataset

commute or business and ones with an unknown choice, are filtered out. The original dataset contains 10,729 records, but after filtering, 6,768 records remain for following analysis. Finally, a new column that uniquely identifies each sample is added to the dataframe and the `CHOICE` column, which originally contains a numerical coding of the choices, is mapped to a description that is consistent with the alternatives in the column names.

- Step 2. Reshape data to long format: The imported dataframe is in wide format, and it needs to be reshaped to long format for processing by *xlogit*, which offers the convenient *wide_to_long* utility for this reshaping process. The user needs to specify the column that uniquely identifies each sample, the names of the alternatives, the columns that vary across alternatives, and whether the alternative names are a prefix or suffix of the column names. Additionally, the user can specify a value (`empty_val`) to be used by default when an alternative is not available for a certain variable.

- Step 3. Create model specification: Following the reshaping, users can create or update the dataset's columns in order to accommodate their model specification needs, if necessary. Step 3 in Figure 3 shows an example in which the columns `ASC_TRAIN` and `ASC_CAR` were created to incorporate alternative-specific constants in the model. In addition, the example illustrates an effective way of establishing variable interactions (e.g., trip costs for commuters with an annual pass) by updating existing columns conditional on values of other columns. Although apparently simple, column operations provide users with an intuitive and highly-flexible mechanism to incorporate model specification aspects, such as variable transformations, interactions, and alternative specific coefficients and constants.

- Step 4. Estimate model parameters: As shown in Step 4 of Figure 3, the `fit` method estimates the model by taking as input the data from Step 3 along with additional specification criteria, such as the distribution of the random parameters (`randvars`), the number of random draws (`n_draws`), and the availability of alternatives for the choice situations (`avail`). Once the estimation routine is completed, the `summary` method can be used to display the estimation results.

Table 1 shows the estimates returned by *xlogit* as well as those provided by *Apollo* and *Biogeme*. The negative signs for the cost and time coefficients suggest that decision makers experience a general disutility with alternatives that have higher waiting times and costs, which conforms to the underlying decision making theory. The estimation results indicate a high degree of similarity between the estimates returned by *xlogit* and those provided by *Apollo* and *Biogeme*. The "Ratio" column shows the average ratio between the coefficients of the three compared packages. The fact that all the ratios are very close to one (with a maximum ratio of 0.01) indicates that the estimated coefficients are highly similar across packages. The slight differences observed in the result comparisons are deemed reasonable due the different internal mechanisms that the tested packages use to process the data, such as different order of arithmetic and matrix operations, which affects floating-point precision, different ways in which Halton draws are generated, and differences in optimization sub-routines (Truong et al., 2019; Blelly et al., 2018). The comparison conducted in this example suggests that *xlogit*'s estimates are highly consistent with those provided by existing estimation tools. In addition, this example showcased convenient *xlogit*'s features that users can leverage for data-preprocessing, model specification, and estimation.

4.2. Artificial dataset

The artificial dataset contains 4,000 synthetically-generated choices among three inter-city transportation modes: bus, car, and train. The explanatory variables that influence the choices are the characteristics of the transportation modes that include price, time, convenience, comfort, food availability (meals), and pet and emissions friendliness. In addition, three variables that have no effect on the choices (statistically non-significant variables) were included in the dataset in order to simulate a more realistic analysis scenario, where potential non-significant variables need to be considered during the specification process. These

	xlogit		Apollo		Biogeme		
Variable	Coeff.	StdErr	Coeff.	StdErr	Coeff.	StdErr	Ratio
ASC Car	0.283	0.056	0.280	0.057	0.278	0.056	1.013
ASC Train	-0.572	0.079	-0.578	0.083	-0.585	0.079	0.985
Cost	-1.660	0.078	-1.660	0.078	-1.655	0.078	1.002
Time	-3.229	0.175	-3.205	0.196	-3.178	0.169	1.011
Standard Deviations							
Time	3.649	0.167	3.663	0.179	3.678	0.170	0.995
Log-Likelihood		-4359.21		-4359.46		-4360.56	

Table 1: Estimation results for the swissmetro dataset.

non-significant variables were modeled by simply not including their values in the artificial utilities; therefore, the model estimation process must properly identify these variables as statistically equal to zero. The variables for meals and pet and emission friendliness were designed with heterogeneous preferences across decision makers, using a normal distribution for their coefficients. The `Python` source code to generate the artificial dataset is available in *xlogit*'s public GitHub repository at the `examples/data` folder.

Figure 4 shows *xlogit*'s source code for estimation of coefficients for the artificial dataset. The first part imports the data into the `Python` environment using the *pandas* package and defines the list of variable names to be included in the model. The second part is the actual model estimation, which starts when the `model.fit` method is called along with the input data and estimation parameters.

```python
import pandas
df = pandas.read_csv("examples/data/artificial_long.csv")
varnames = ['price', 'time', 'conven', 'comfort', 'meals',
            'petfr', 'emipp', 'nonsig1', 'nonsig2', 'nonsig3']

from xlogit import MixedLogit
model = MixedLogit()
model.fit(X=df[varnames], y=df['choice'], varnames=varnames,
          ids=df['id'], alts=df['alt'], n_draws=1500,
          randvars={'meals': 'n', 'petfr': 'n', 'emipp': 'n'})
```

Figure 4: Estimation code for artificial dataset

Table 2 shows the estimation results for the artificial dataset using *xlogit*, *mlogit* and *gmnl*. The "Target" column lists the coefficients that were artificially designed for the different explanatory variables, which represent the coefficients to which a good estimation procedure should converge to. The results in Table 2 show that the coefficients, standard errors, and log-likelihood values estimated by *xlogit* are highly consistent with those estimated using *gmnl* and *mlogit*. The results also show a high degree of similarity between the estimated and target coefficients, demonstrating the success of *xlogit* at recovering the designed coefficients, similar to *mlogit* and *gmnl*. This can be corroborated using the "Ratio" column (average ratio between coefficients across packages), whose values deviate from one by a maximum of 0.042, thus indicating high similarity between coefficients. In addition, it can be noted that the three variables that were included as non-signicant in the dataset were properly identified as such by all the packages, which returned coefficients that are statistically equal to zero for the non-significant variables. The successful recovery of the artificially designed coefficients and the high level of similarity between the estimation results of *xlogit* and the other two estimation tools demonstrates that *xlogit* provides consistent and meaningful estimation results.

	Target	xlogit		mlogit		gmnl		
Variable	Coeff.	Coeff.	StdErr	Coeff.	StdErr	Coeff.	StdErr	Ratio
Price	-1	-1.032	0.155	-1.052	0.178	-1.056	0.185	0.985
Time	-1.5	-1.460	0.169	-1.480	0.189	-1.484	0.187	0.989
Convenience	1	0.890	0.141	0.901	0.159	0.903	0.160	0.990
Comfort	1	1.063	0.183	1.087	0.206	1.091	0.204	0.983
Meals	2	1.712	0.197	1.735	0.214	1.735	0.218	0.991
Pet friendly	4	3.875	0.374	3.946	0.429	3.959	0.427	0.986
Emissions	-2	-2.016	0.214	-2.059	0.242	-2.064	0.246	0.984
Non-signif. 1		0.073	0.127	0.070	0.151	0.069	0.148	
Non-signif. 2		0.016	0.134	0.017	0.165	0.016	0.154	
Non-signif. 3		0.019	0.113	0.025	0.141	0.026	0.136	
Standard Deviations								
Meals	1	0.684	0.237	0.714	0.271	0.731	0.248	0.958
Pet Friendly	1	1.323	0.319	1.379	0.351	1.399	0.341	0.963
Emissions	1	1.004	0.142	1.025	0.142	1.021	0.147	0.989
Log-Likelihood			-2278.79		-2278.19		-2278.39	

Table 2: Estimation results for the artificial dataset.

4.3. Electricity dataset

The electricity dataset contains 4,308 choices among four electricity suppliers based on the attributes of the offered plans, which include prices(pf), contract lengths(cl), time of day rates (tod), seasonal rates(seas), as well as attributes of the suppliers, which include whether the supplier is local (loc) and well-known (wk). The data was collected through a survey conducted to 361 participants by presenting them with 12 different choice situations. The multiple responses per participants were organized into panels. Given that some participants answered less than 12 of the choice situations, some panels are unbalanced, which *xlogit* is able to handle. Revelt and Train (2000) provide a detailed description of this dataset. Figure 5 shows the source code used with *xlogit* to estimate the coefficients for the electricity dataset. Note that the parameter `panels` was included in the `fit` function in order to take into account panel structure of this dataset during estimation.

Table 3 presents the estimation results for the electricity dataset using *xlogit*, *mlogit*, and *gmnl*. Comparing the estimated coefficients using the "Ratio" column, the largest ratio deviation from one is 0.01, which is minimal, demonstrating a high level of similarity. In

```python
import pandas
df = pandas.read_csv("examples/data/electricity_long.csv")
varnames = ['pf', 'cl', 'loc', 'wk', 'tod', 'seas']

from xlogit import MixedLogit
model = MixedLogit()
model.fit(X=df[varnames], y=df['choice'], varnames=varnames,
          ids=df['chid'], panels=df['id'], alts=df['alt'], n_draws=1500,
          randvars={'pf': 'n', 'cl': 'n', 'loc': 'n', 'wk': 'n',
                    'tod': 'n', 'seas': 'n'})
```

Figure 5: Estimation code for electricity dataset

terms of conceptual interpretation, the coefficient estimates obtained from the three packages comply with the underlying human decision-making theory. Cost and time variables, such as price, seasonal rate and time of the day rate, and contract length, have negative coefficients depicting the dis-utility associated with these variables. The positive coefficients for the supplier type (i.e., local and well-known suppliers) indicate that the probability of choosing a supplier increases if they are local and well-known. The additional coefficients include the random effects of the variables on choice behavior. All of the coefficients are assumed to be normally distributed. The statistically significant standard deviations of the coefficients determine the existence of heterogeneity in individual choice preferences, which is generally caused due to some unobserved explanatory variables.

	xlogit		mlogit		gmnl		
Variable	Coeff.	StdErr	Coeff.	StdErr	Coeff.	StdErr	Ratio
Price	-1.005	0.034	-0.989	0.036	-1.005	0.039	1.000
Contract length	-0.240	0.026	-0.228	0.015	-0.242	0.026	0.994
Local supplier	2.285	0.125	2.273	0.090	2.284	0.130	1.000
Well-known supplier	1.709	0.101	1.646	0.072	1.704	0.098	1.002
Time of day rates	-9.684	0.321	-9.669	0.316	-9.635	0.345	1.003
Seasonal rates	-9.821	0.309	-9.750	0.316	-9.842	0.338	0.999
Standard Deviations							
Price	0.233	0.018	0.199	0.013	0.231	0.019	1.013
Contract length	0.422	0.025	0.406	0.021	0.420	0.025	1.004
Local supplier	1.849	0.120	1.822	0.105	1.843	0.123	1.002
Well-known supplier	1.242	0.091	1.251	0.086	1.246	0.097	0.998
Time of day rates	2.484	0.214	2.459	0.138	2.514	0.226	0.992
Seasonal rates	1.517	0.166	1.633	0.138	1.527	0.164	0.997
Log-Likelihood		-3886.17		-3886.02		-3886.39	

Table 3: Estimation results for the electricity dataset.

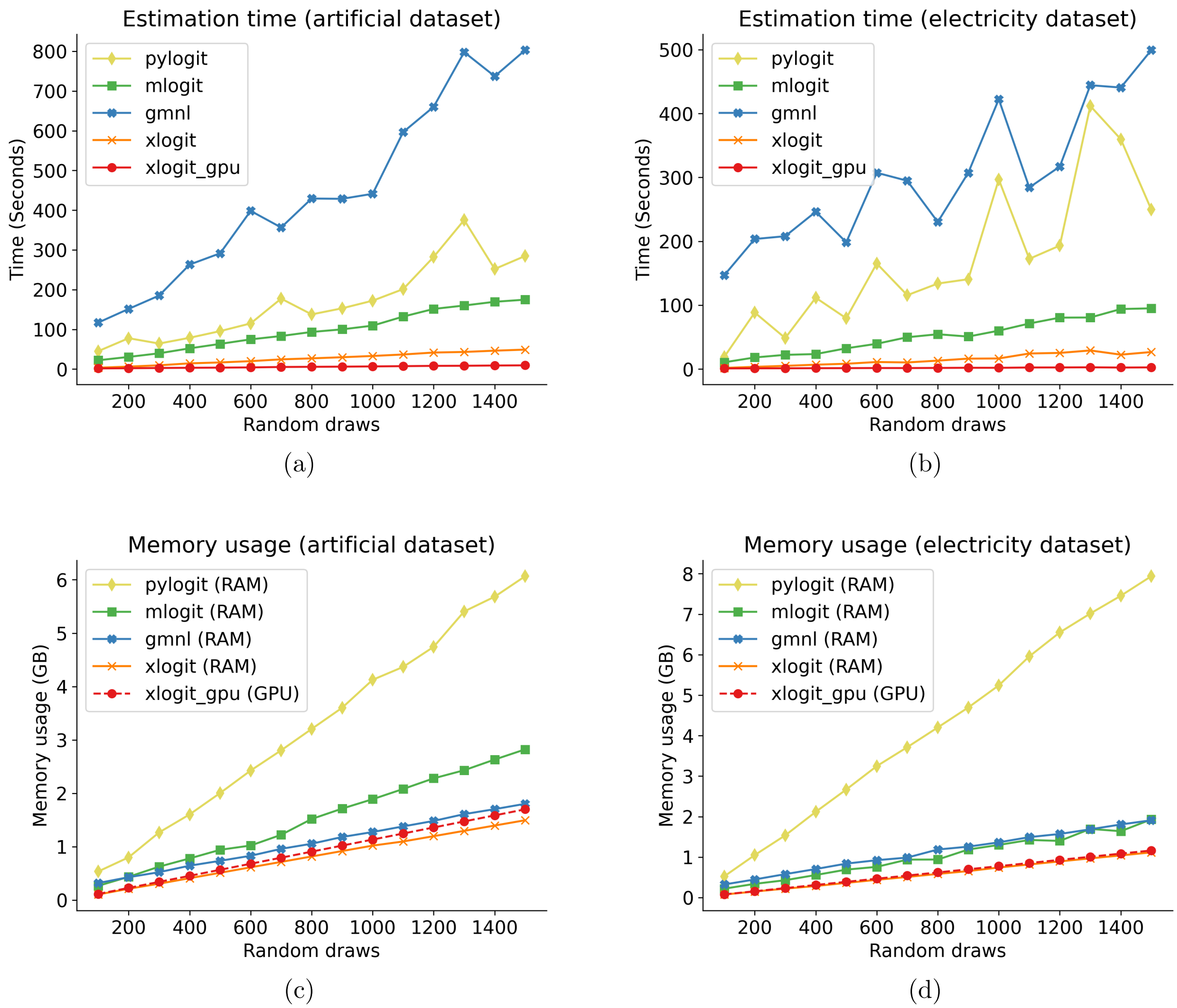
5. Benchmark

This section describes the benchmark results between *xlogit* and six open-source estimation packages to demonstrate the gains in estimation speed and memory management offered by the proposed package. First, a benchmark of estimation speed and memory usage between *xlogit* (v0.1.4), *mlogit* (v1.1.1), *PyLogit* (v0.2.2), and *gmnl* (v1.1.3.2) is discussed. Second, a benchmark of estimation speed between *xlogit* (v0.1.4), *Biogeme* (v3.2.6), *Apollo*(v0.1.0), and *mixl*(v1.3.1), focusing on parallel processing, is described. The benchmark is split into two groups based on parallel processing capabilities: 1) the first group includes *Apollo*, *Biogeme*, and *mixl*, possessing parallel processing capabilities; and 2) the second group includes *mlogit*, *PyLogit*, and *gmnl*, without them. The artificial and electricity datasets used for the benchmark incorporate a considerable number of records and heterogeneous estimation requirements, such as the use of panel data and multiple explanatory variables with mixing distributions, which offer a substantial level of complexity to demonstrate *xlogit*'s advantages. The source code, data, and instructions to execute the benchmarks are available in *xlogit*'s public GitHub repository at the `examples/benchmark` folder.

5.1. Benchmark against mlogit, PyLogit, and gmnl

This benchmark was conducted by iteratively evaluating the effect of a different number of random draws on the estimation speed and memory usage for the tested packages on the artificial and electricity datasets. The number of tested random draws started at 100 up to 1,500, with increments of 100. The stopping criteria set by default in the tested packages were kept. The benchmarks were executed on a desktop computer with a Linux Ubuntu 20.04 operating system, an Intel i5 8400 processor, and 16GB of RAM memory. The graphics card is a NVIDIA GTX1060 with 6GB of Memory and 1,280 CUDA cores, which is a low-cost GPU, in order to show that even with a budget graphics card, *xlogit* can offer significant estimation speed gains. The experiments were executed while running the operating system in command line mode (without graphical user interface) to minimize the potential influence of other programs on the estimation time. The estimation time was measured as the elapsed time between the beginning and end of the estimation. The memory usage was measured as the maximum amount of memory used at any point during the estimation process (sampling every 50 milliseconds).

Figure 6 shows the benchmark results for *xlogit*, *mlogit*, *PyLogit*, and *gmnl*. *xlogit* was evaluated with and without GPU processing, and the results are labeled as "xlogit_gpu" and "xlogit", respectively. Additionally, the memory benchmark for *xlogit* with GPU processing includes an evaluation of the RAM and GPU memory used. The results in Figure 6 indicate that, in terms of estimation speed, *xlogit* with GPU processing is significantly faster than *mlogit*, *gmnl*, and *PyLogit*, and it scales better when the the number of random draws is increased. In addition, it can be noted that, even without using GPU processing, *xlogit* still offers important estimation speed gains compared to the *mlogit*, *gmnl*, and *PyLogit*. However, the most significant speed gains are provided by the use of GPU processing, as it will be shown later in the "Summary of time benchmark" subsection. This suggests that, in terms of memory usage, *xlogit* requires less memory than *mlogit*,*gmnl*, and *PyLogit* for estimation. In addition, the GPU memory required by *xlogit* is minimal, and it conveniently scales linearly with the number of random draws. For both datasets, the GPU memory used is less than 1.5GB, even when using a large number of random draws. This is an important result, as it shows that *xlogit* can run even on graphics cards that may have limited GPU memory.



(a) (b) (c) (d)

Figure 6: Results for benchmark against *mlogit*, *PyLogit*, and *gmnl*.

5.2. Benchmark against Apollo, Biogeme, and mixl

Similar to the previous subsection, the effect of different numbers of random draws on the estimation speed of the tested packages was evaluated. However, this benchmark focuses on the parallel processing capabilities of *Apollo*, *Biogeme*, and *mixl* by testing their performance using 16, 32, and 64 processor cores. The tested number of random draws included 100, 500, 1,000, and 1,500. This benchmark did not evaluate memory usage, given that profiling memory consumption in multi-thread processing is more cumbersome for replication. In addition, only the artificial dataset was used in this benchmark in order to reduce execution times, as estimation tends to be time-consuming using *Apollo*, *Biogeme*, and *mixl*. The benchmarks were executed on a server with a Linux Centos 7.8 operating system, four Intel Xeon E7-4870 processors (80 total cores), and 256GB of RAM memory. During the execution of the experiments, no graphical interface or other processes were running on the server, in order to minimize the potential influence of external programs on the estimation times. The use of this powerful server was necessary to compare the estimation times of *Apollo*, *Biogeme*, and *mixl* using multiple processor cores against *xlogit* using a GPU.

Figure 7 shows the benchmark results for *xlogit*, *Apollo*, *Biogeme*, and *mixl*. These results indicate that additional processor cores can help *Apollo* and *Biogeme* to reduce estimation times. However, *xlogit* is significantly faster than all the compared packages, even when these use 64 processor cores, with the additional advantage of a better scaling with the number of random draws. The results for "xlogit" and "xlogit_gpu" were reused from the previous benchmark (on a desktop computer), given that the server used for this benchmark did not have a GPU for estimation with *xlogit*.

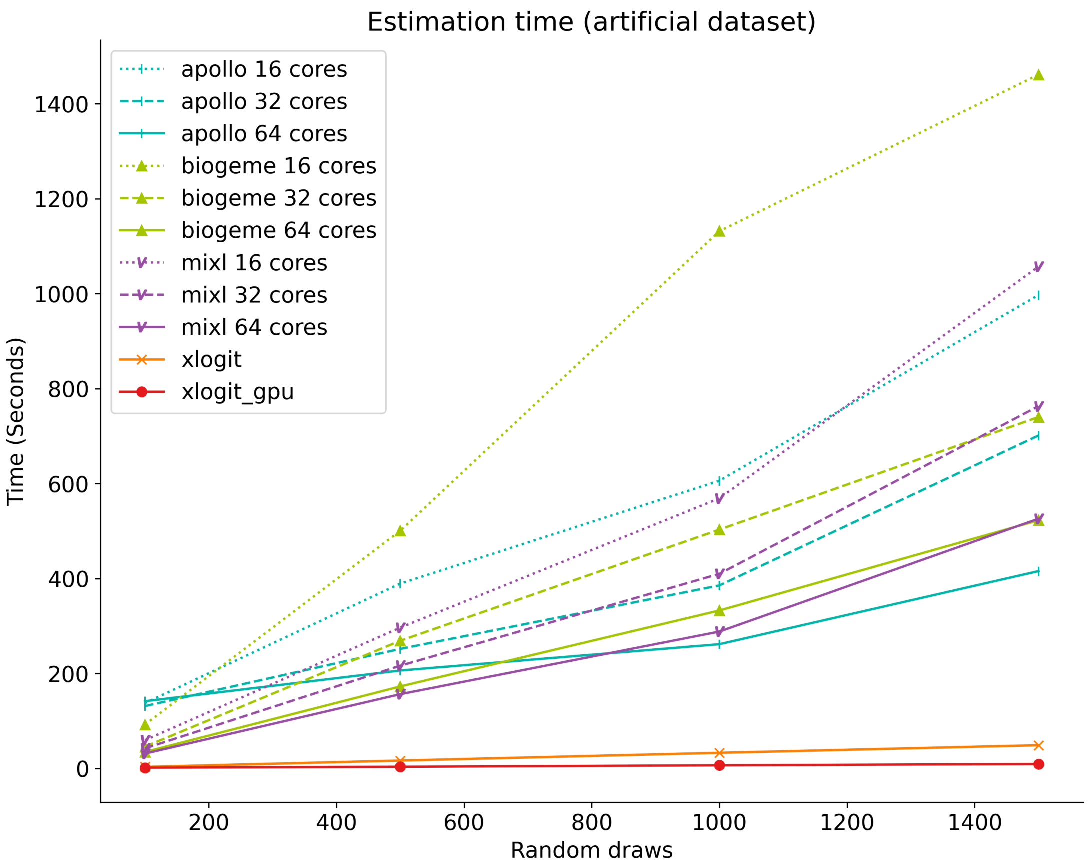


Figure 7: Results for benchmark against *Apollo*, *Biogeme*, and *mixl*.

5.3. Summary of time benchmark

Table 4 summarizes the estimation time for the artificial dataset using all the tested packages. The presented results for *Apollo* and *Biogeme* correspond to estimation using 64 processor cores. The set of columns titled "Estimation time (sec)" show how many seconds the estimation took using the tested packages at different numbers of random draws. The set of columns titled "Times slower than xlogit_gpu" show how the estimation speed compares to *xlogit* with GPU processing enabled. The "Average" column shows the fraction at which the evaluated packages are slower than "xlogit_gpu" averaged across the tested random draws. These results indicate that *xlogit* with GPU processing enabled is, on average, 55x faster than *Apollo*, 43x faster than *Biogeme*, 74x faster than *gmnl*, 39x faster than *mixl*, 16x faster than *mlogit*, and 27x faster than *PyLogit*. It is important to highlight that the presented results for *xlogit* were obtained using a modest GPU with only 1,280 CUDA cores; therefore, it is expected that more recent and sophisticated GPUs, which usually contain more than 5,000 CUDA cores (e.g. NVIDIA RTX 3070 and 3090), will provide considerably more performance gains for model estimation with *xlogit*.

	Estimation time (sec)				**Times slower than xlogit_gpu**				
	Random draws				Random draws				**Average**
	100	500	1000	1500	100	500	1000	1500	
apollo	141.7	206.4	261.8	415.8	83.4	55.8	38.9	44	**55.5**
biogeme	34.8	172.8	332.8	522.7	20.4	46.7	49.4	55.3	**43**
gmnl	117	291.5	441.3	803	68.8	78.7	65.6	84.9	**74.5**
mixl	31.4	156.5	288.4	526.1	18.4	42.3	42.8	55.7	**39.8**
mlogit	22.2	63.4	109.4	174.8	13.1	17.1	16.3	18.5	**16.2**
pylogit	45	95.6	172.2	284.8	26.5	25.8	25.6	30.1	**27**
xlogit	3.5	16.8	33.1	49	2.1	4.5	4.9	5.2	**4.2**
xlogit_gpu	1.7	3.7	6.7	9.5	1	1	1	1	**1**

Table 4: Summary of estimation times for the artificial dataset.

5.4. Escalating the number draws

This section describes the convenient scaling that *xlogit* offers, as it can efficiently estimate models using hundreds of thousands of random draws on a regular desktop computer. Figure 8 (a) and (b) show that, using the setup presented in Section 5.2 (i5-8400 processor, 16GB RAM, and GTX1060 GPU), *xlogit* with GPU processing enabled estimates the model for the electricity dataset, using a half-million random draws in around 12 minutes, and the model for the artificial dataset, using 50,000 draws in around 7 minutes. Additionally, even without using GPU processing, *xlogit* yields reasonable estimation times (below two hours) considering the large number of draws that are used. However, the largest performance gains are provided by the GPU-enabled estimation.

In terms of memory usage, Figure 8 (c) and (d) illustrate that *xlogit* handles GPU memory very efficiently, as it uses batch processing of the random draws to avoid overflowing the available GPU memory (6GB in this case). When the input data and random draws are too big to fit in the GPU memory, *xlogit* keeps the data in the RAM memory and splits it into different processing batches of smaller size, which are iteratively moved to and processed in the GPU. This convenient feature enables analysts use *xlogit* to estimate models using thousands and even millions of random draws on devices with low GPU memory, with the only limitation being the available RAM memory. For instance, Figure 8 (c) and (d) indicate the usage of GPU memory never goes above 6GB, as a result of the batch processing;

however, the usage of RAM memory keeps linearly increasing with the number of random draws, as RAM needs to constantly store all of the input data and random draws. The experiments presented in this section stopped increasing the number of random draws at the point where RAM memory usage was getting close to 16GB, as this was the available RAM memory in the system used for the experiments. The difference in RAM memory usage between the electricity and artificial datasets is due to the number of individuals that each dataset contains. The electricity dataset contains 361 individuals, whereas the artificial dataset contains 4,000 individuals. Therefore, the artificial dataset requires about 10 times more random draws than the electricity dataset at each iteration, which results in 10 times larger RAM memory usage.

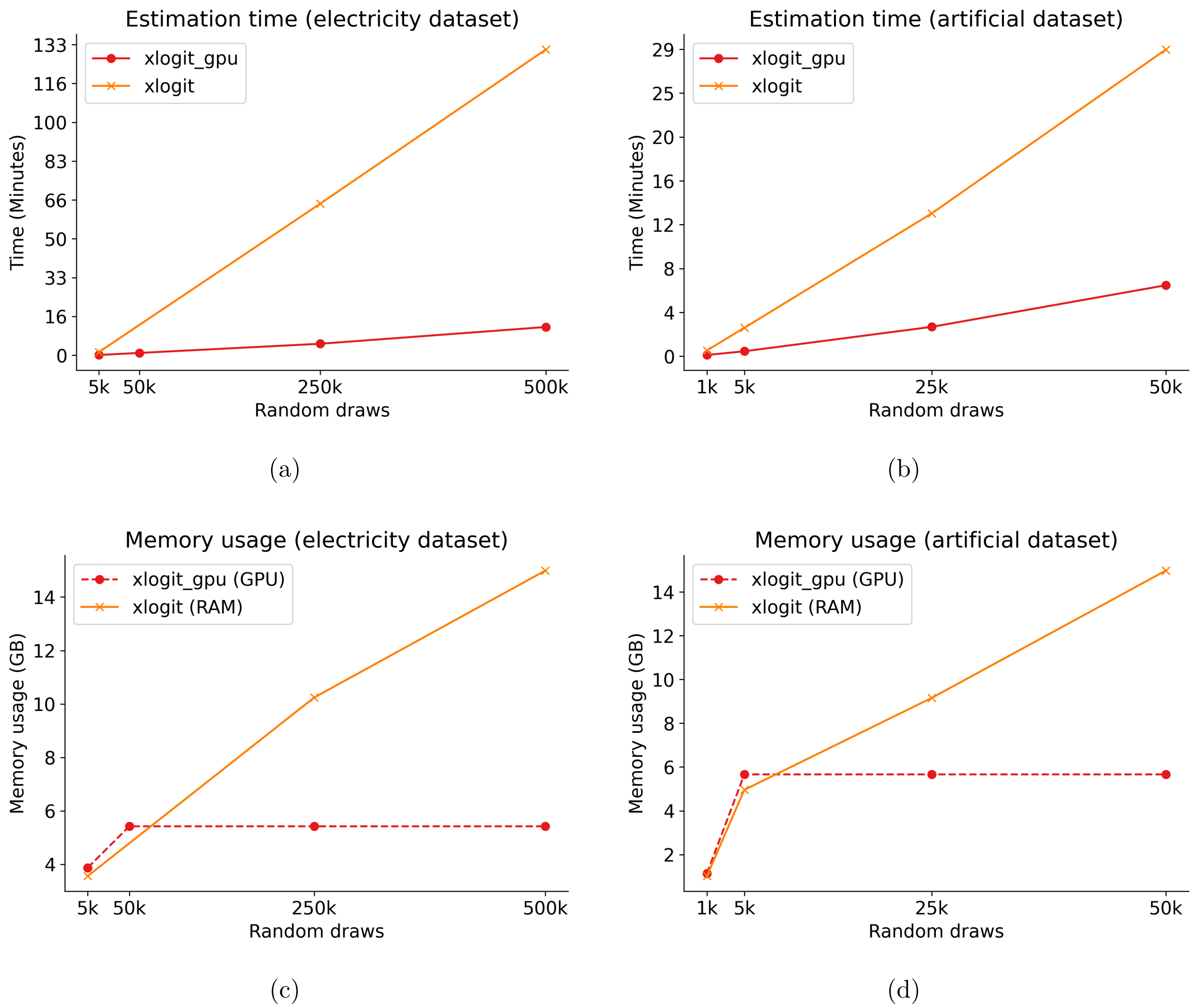


Figure 8: Performance and memory usage of xlogit for massive number of draws

6. Conclusions

This paper introduces *xlogit*, an open-source `Python` package for GPU-accelerated estimation of Mixed Logit models. The package's features were described and usage examples were provided. In addition, this paper describes a benchmark between *xlogit* and four existing estimation packages. The benchmark results suggest that, even using a mid-range graphics card and a regular desktop computer, *xlogit* with GPU processing enabled is, on average, 55x faster than *Apollo*, 43x faster than *Biogeme*, 74x faster than *gmnl*, 39x faster than *mixl*, 16x faster than *mlogit*, and 27x faster than *PyLogit*. In addition, the conducted experiments suggest that *xlogit* provides efficient scaling when the number of random draws increases, as well as convenient memory usage. The significant reduction in estimation times and the ability to escalate to estimation using hundreds of thousands of random draws offered by *xlogit* is expected to help analysts in the model specification process, as more specifications can be rapidly tested and evaluated while using a large number of draws for estimation. In addition, the accelerated estimation enabled by *xlogit* is expected to open up additional analysis opportunities for approaches that assist the model specification process by strategically testing a large set of specifications, which can be efficiently achieved using *xlogit*.

The current version of *xlogit* offers a comprehensive set of tools for the estimation of Mixed Logit models, along with extra functionalities to estimate Multinomial and Conditional Logit models. However, there are other discrete choice models (e.g., Probit, Nested Logit, and Latent Class) that can potentially benefit users of the library. Therefore, future development efforts will focus on expanding *xlogit*'s features to enable the estimation of additional Logit-based discrete choice models. Also, the plan for future releases of *xlogit* include the implementation of additional utilities for post-estimation and model specification testing. Finally, the current version of *xlogit* only supports GPU acceleration on CUDA-enabled NVIDIA graphics cards. Therefore, future efforts will seek to add support to GPU processing technologies beyond CUDA and NVIDIA cards.

7. Acknowledgements

This project was partially supported by the National Cooperative Highway Research Program (NCHRP) (IDEA 231). Any opinions, findings, and conclusions or recommendations expressed on this publication are those of the authors and do not necessarily reflect the views of the NCHRP.

References

AWS, 2020. Amazon ec2 p2 instances. URL: `https://aws.amazon.com/ec2/instance-types/p2/`.

Ben-Akiva, M., Lerman, S.R., 1985. Discrete Choice Analysis: Theory and Application to Travel Demand. volume 9. MIT press.

Berndt, E., Hall, B., Hall, R., Hausman, J., 1974. Estimation and inference in nonlinear structural models. Annals of Economic and Social Measurement 3/4, 653–665.

Bierlaire, M., 2020. A Short Introduction to *PandasBiogeme*. Technical Report. Ecole Polytechnique Federale de Lausanne. URL: http://transp-or.epfl.ch/documents/technicalReports/Bier20.pdf.

Bierlaire, M., Axhausen, K., Abay, G., 2001. The acceptance of modal innovation: The case of swissmetro, in: Swiss Transport Research Conference, Ascona, Switzerland. pp. 1–15. doi:10.3929/ethz-a-004238511.

Blelly, A., Felipe-Gomes, M., Auger, A., Brockhoff, D., 2018. Stopping criteria, initialization, and implementations of bfgs and their effect on the bbob test suite, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery, New York, NY, USA. p. 1513–1517. URL: https://doi.org/10.1145/3205651.3208303, doi:10.1145/3205651.3208303.

Brathwaite, T., Walker, J.L., 2018. Asymmetric, closed-form, finite-parameter models of multinomial choice. Journal of Choice Modelling 29, 78–112. doi:10.1016/j.jocm.2018.01.002.

Bunch, D.S., 1987. Maximum likelihood estimation of probabilistic choice models. SIAM Journal on Scientific and Statistical Computing 8, 56–70. URL: http://epubs.siam.org/doi/10.1137/0908006, doi:10.1137/0908006.

Cameron, A.C., Trivedi, P.K., 2005. Microeconometrics : methods and applications. Cambridge University Press, Cambridge ; New York.

Chiou, L., Walker, J.L., 2007. Masking identification of discrete choice models under simulation methods. Journal of Econometrics 141, 683–703. URL: https://www.sciencedirect.com/science/article/pii/S0304407606002119, doi:https://doi.org/10.1016/j.jeconom.2006.10.012.

Colab, 2020. Google colaboratory faq. URL: https://research.google.com/colaboratory/faq.html.

Croissant, Y., 2020. *mlogit*: Random utility models in R. Journal of Statistical Software 95, 1–41. URL: https://www.jstatsoft.org/index.php/jss/article/view/v095i11/v95i11.pdf, doi:10.18637/jss.v095.i11.

Czajkowski, M., Budziński, W., 2019. Simulation error in maximum likelihood estimation of discrete choice models. Journal of Choice Modelling 31, 73–85. URL: https://www.sciencedirect.com/science/article/pii/S1755534517302002, doi:https://doi.org/10.1016/j.jocm.2019.04.003.

DelRizzo, B., 2019. Turing now starts at $149: Introducing geforce gtx 1650. URL: https://blogs.nvidia.com/blog/2019/04/23/geforce-gtx-1650/.

Fletcher, R., 1981. Practical methods of optimization: Vol. 2: Constrained optimization. JOHN WILEY & SONS, INC., ONE WILEY DR., SOMERSET, N. J. 08873, 1981, 224 .

GCloud, 2020. Google cloud gpus pricing. URL: https://cloud.google.com/compute/gpus-pricing.

Greene, W.H., 2012. NLOGIT Version 5 Reference Guide. Plainview, NY.

Hasan, A., Wang, Z., Mahani, A.S., 2016. Fast estimation of multinomial logit models: R package *mnlogit*. Journal of Statistical Software 75. URL: http://www.jstatsoft.org/v75/i03/, doi:10.18637/jss.v075.i03.

Hess, S., Palma, D., 2019. *Apollo*: A flexible, powerful and customisable freeware package for choice model estimation and application. Journal of Choice Modelling 32, 100170. doi:10.1016/j.jocm.2019.100170.

Intini, P., Berloco, N., Fonzone, A., Fountas, G., Ranieri, V., 2020. The influence of traffic, geometric and context variables on urban crash types: A grouped random parameter multinomial logit approach. Analytic Methods in Accident Research 28, 100141. URL: https://linkinghub.elsevier.com/retrieve/pii/S2213665720300312, doi:10.1016/j.amar.2020.100141.

Kindratenko, V., 2014. Numerical Computations with GPUs. Springer-Verlag International Publishing. doi:10.1007/978-3-319-06548-9.

Liu, D.C., Nocedal, J., 1989. On the limited memory bfgs method for large scale optimization. Mathematical Programming 45, 503–528. URL: http://link.springer.com/10.1007/BF01589116, doi:10.1007/BF01589116.

Manski, C.F., 1977. The structure of random utility models. Theory and Decision 8, 229.

Mariel, P., Hoyos, D., Meyerhoff, J., Czajkowski, M., Dekker, T., Glenk, K., Jacobsen, J.B., Liebe, U., Olsen, S.B., Sagebiel, J., Thiene, M., 2021. Environmental Valuation with Discrete Choice Experiments : Guidance on Design, Implementation and Data Analysis. Springer Nature. doi:10.1007/978-3-030-62669-3.

Matthews, D., 2018. Supercharge your data wrangling with a graphics card. Nature 562, 151–152. doi:10.1038/d41586-018-06870-8.

McFadden, D., Train, K., 2000. Mixed mnl models for discrete response. Journal of Applied Econometrics 15, 447–470. doi:10.1002/1099-1255(200009/10)15:5<447::AID-JAE570>3.0.CO;2-1.

McKinney, W., 2010. Data structures for statistical computing in Python, in: Stéfan van der Walt, Millman, J. (Eds.), Proceedings of the 9th Python in Science Conference, pp. 56 – 61. doi:10.25080/Majora-92bf1922-00a.

Misic, M.J., Durdevic, D.M., Tomasevic, M.V., 2012. Evolution and trends in gpu computing, in: 2012 Proceedings of the 35th International Convention MIPRO, Opatija, Croatia. pp. 289–294.

Molloy, J., Becker, F., Schmid, B., Axhausen, K.W., 2021. mixl: An open-source r package for estimating complex choice models on large datasets. Journal of Choice Modelling 39, 100284. URL: https://www.sciencedirect.com/science/article/pii/S1755534521000178, doi:https://doi.org/10.1016/j.jocm.2021.100284.

NVIDIA, 2020. Geforce rtx 3080. URL: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3080/.

Okuta, R., Unno, Y., Nishino, D., Hido, S., Loomis, C., 2017. *CuPy*: A numpy-compatible library for nvidia gpu calculations, in: Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS), pp. 1–7. URL: http://learningsys.org/nips17/assets/papers/paper_16.pdf.

Ortelli, N., Hillel, T., Camara Pereira, F., De Lapparent, M., Bierlaire, M., 2020. Assisted Specification of Discrete Choice Models. Technical Report. Ecole Polytechnique Fédérale de Lausanne.

Paz, A., Arteaga, C., Cobos, C., 2019. Specification of mixed logit models assisted by an optimization framework. Journal of Choice Modelling 30, 50–60. doi:10.1016/j.jocm.2019.01.001.

Raspa, M., Paquin, R.S., Brown, D.S., Andrews, S., Edwards, A., Moultrie, R., Wagner, L., Frisch, M., Turner-Brown, L., Wheeler, A.C., 2020. Preferences for accessing electronic health records for research purposes: Views of parents who have a child with a known or suspected genetic condition. Value in Health 23, 1639–1652. URL: https://linkinghub.elsevier.com/retrieve/pii/S1098301520343576, doi:10.1016/j.jval.2020.06.016.

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

Revelt, D., Train, K., 2000. Customer-specific taste parameters and mixed logit: Households' choice of electricity supplier. University of California, Berkeley .

Rodrigues, F., Ortelli, N., Bierlaire, M., Pereira, F., 2019. Bayesian automatic relevance determination for utility function specification in discrete choice models. URL: http://arxiv.org/abs/1906.03855.

Sarrias, M., Daziano, R., 2017. Multinomial logit models with continuous and discrete individual heterogeneity in R: The gmnl package. Journal of Statistical Software 79, 1–46. doi:10.18637/jss.v079.i02.

StataCorp, 2019. Stata Statistical Software: Release 16. College Station, TX.

Train, K.E., 2003. Discrete Choice Methods with Simulation. Cambridge University Press, Cambridge. URL: http://ebooks.cambridge.org/ref/id/CBO9780511753930, doi:10.1017/CBO9780511753930.

Truong, B.C., Nguyen, V.B., Truong, H.V., Ho, T.D.C., 2019. Comparison of optim, nleqslv and maxlik to estimate parameters in some of regression models. Journal of Advanced Engineering and Computation 3, 532–550. URL: http://jaec.vn/index.php/JAEC/article/view/262, doi:10.25073/jaec.201934.262.

Vij, A., Krueger, R., 2017. Random taste heterogeneity in discrete choice models: Flexible nonparametric finite mixture distributions. Transportation Research Part B: Methodological 106, 76–101. URL: https://linkinghub.elsevier.com/retrieve/pii/S0191261516301138, doi:10.1016/j.trb.2017.10.013.

Wang, J., Dane, G.Z., Timmermans, H.J.P., 2020. Carsharing-facilitating neighbourhood choice: a mixed logit model. Journal of Housing and the Built Environment URL: http://link.springer.com/10.1007/s10901-020-09791-z, doi:10.1007/s10901-020-09791-z.