EPFL

Comparative analysis of discrete choice models estimation using different software packages

Alexia Paratte

Master of Applied Mathematics

Semester Project

May 2024

Supervisors
Negar Rezvany
Evangelos Paschalidis
Prof. Michel Bierlaire

Contents

1 Introduction

Since their establishment in the field of econometrics, discrete choice models have been extensively used in many research fields such as transportation, environmental economics, health, energy, marketing, and others. Currently, there is a plethora of commercial and open-source software that can be used for the estimation of discrete choice models. Biogeme (Bierlaire, 2023) and Apollo (Hess & Palma, 2023) are two of the most appealing alternatives, as they are both open-source (based on Python and R respectively) and cover a broad range of advanced discrete choice models and diagnostics. More recently, other alternatives have been developed such as mixl (Molloy et al., 2021) and MO—DE.behave (Reul et al., 2023). Given the differences in the code development and the tools used, this project aims at investigating (a) differences in performance and computational speed between the two packages, (b) potential differences in the results and model statistics for the same models and datasets, and (c) available functionalities i.e., what are the common features and what are the specific features of some packages that are not available in others. Understanding the pros and cons of each package via this comparative analysis can ultimately lead to more optimised and accurate software for estimating discrete choice models.

The approach for this project is to investigate different discrete choice models and scripts from either the Biogeme or Apollo websites, translate them in the other language, make sure the models are equivalent and see how the library performs in term of estimation time. Other goals on the side are to compare features (pre and post-estimation), and to see how the Biogeme Documentation website might be improved. Data sets are the one provided by the corresponding software, that are publicly available.

This project is completed as part of a semester project by an Applied Mathematics MA student under the supervision of Negar Rezvany, Evangelos Paschalidis and Michel Bierlaire at the Ecole Polytechnique Fédérale de Lausanne (EPFL). A GitHub containing all the scripts and documentation is available.

In the following sections, we will first provide a brief overview of the Biogeme and Apollo packages. Next, we will describe the datasets used to give a clearer understanding of the focus of the models. The models under investigation include a Multinomial Logit Model (MNL) and two Mixed Multinomial Logit Models (MMNL) to introduce randomness and complexity to the analysis. An additional subsection will discuss CPU and RAM usage in relation to the previously investigated models. We will also compare the features of Apollo and Biogeme, and summarize the difficulties encountered in the final section.

2 Packages overview

2.1 Biogeme

Biogeme (BIOlogy-inspired GEneralized Method of Moments Estimation) is a Python package designed for estimating discrete choice models. These models are used primarily in transportation research to understand and predict choices made by individuals among a finite set of alternatives. It is particularly useful in fields such as transportation planning, marketing research, and environmental economics. Biogeme emphasizes user-friendly syntax and flexible model specification, making it accessible for defining utility functions and choice sets. Key features include multiple estimation methods, such as maximum likelihood estimation, and advanced capabilities like Monte Carlo simulations and handling panel data. Biogeme was developed by Michel Bierlaire, a professor at the EPFL. His work primarily focuses on transportation systems and operations research, and Biogeme is a reflection of his expertise in discrete choice modeling and econometrics.

2.2 Apollo

Apollo is an R package designed for estimating and applying choice models. It is widely used in transportation research, but its application extends to other fields involving choice modeling. Apollo is known for its extensive documentation and comprehensive support for various model types. Apollo was developed by Stephane Hess and David Palma. Stephane Hess is a prominent researcher in the field of transportation and choice modeling, known for his contributions to the understanding and application of discrete choice models. David Palma has worked closely with Hess in developing tools and methods for advanced choice modeling. Their collaboration resulted in the creation of Apollo, aiming to provide a flexible and powerful tool for choice model estimation and application.

3 Datasets description

3.1 MNL: `apollo_modeChoiceData`

Our primary resource is a synthetic dataset that examines mode choices for 500 travelers. Each individual in the dataset has made two revealed preference (RP) inter-city trips, choosing from the modes of car, bus, air, and rail, with at least two of these options available. The journey details include access time (excluding car), travel time, and cost, with times recorded in minutes and costs in dollars. Additionally, the dataset includes information on each individual's gender, whether the trip was for business, and their income. Stated preference (SP) data has been excluded in this case.

3.2 MMNL: `apollo_swissRouteChoiceData`

The dataset `apollo_swissRouteChoiceData` originates from an actual stated preference (SP) survey on public transport route choices conducted in Switzerland (Axhausen et al., 2006). It involves 388 participants who each faced 9 choices between two public transport routes, both involving train travel, resulting in 3'492 observations. The two alternatives are described based on travel time (mins), travel cost (CHF), headway (time between subsequent trains/buses), and the number of interchanges. Additionally, the dataset includes information on each individual's income, household car availability, and whether the journey was for commuting, shopping, business, or leisure.

3.3 Swissmetro: `biogeme_swissmetro`

This dataset comprises survey data collected on trains between St. Gallen and Geneva, Switzerland, in March 1998 (see Axhausen et al. (2001) and Bierlaire Michel (2018)). The respondents provided information to analyze the impact of a transportation modal innovation, the Swissmetro—a revolutionary maglev underground system—compared to the usual transport modes of car and train. Survey data were collected from 470 respondents on rail-based travels, but due to data issues, only 441 responses are used here. Each of the 441 respondents faced nine stated choice situations, offering three alternatives: rail, Swissmetro, and car (only for car owners). Time is in minutes and cost in Swiss francs.

4 Models

4.1 Preparation

Before defining the models we are working on for this project, it is crucial to specify the parameters selected in Biogeme and Apollo. This ensures that the estimation time is optimized and that the models are equivalent.

Here are some points which we need to be careful about, to avoid any mistake in translation:

- Defining the variables the same way as in the original script language
- Defining the parameters and utility functions correctly
- Make sure the availabilities are taken into account in both cases
- In a MMNL case, using the same type and same number of draws

As for the scripts parameters, we observe them in Table 1. Parameters that are not defined here are the defaults one from Biogeme and Apollo. The optimization algorithm as well as the second derivative differ from one model to another, and we will motivate how we choose

these specifications in the future. These parameters are directly selected in the script in Apollo. As for Biogeme, a `biogeme.toml` file with default values is generated during the script's initial run. The parameters should be adjusted after this file is created.

	Apollo	**Biogeme**
Number of threads	8	8
Number of cores	4	4
Second derivative	1	0 *or* 1
Optimization Algorithm	BGW *or* BFGS	Newton *or* BFGS for simple bounds
Type of draws	Inter-individuals	Inter-individuals

Table 1: Scripts parameters, to choose before computing

4.2 MNL Revealed Preference Model

The first model we investigate is a basic MNL we select from Apollo's website. It is called the `RP_model` and the dataset used in this case is the `apollo_modeChoiceData` (3.1). The optimisation algorithm in Apollo and Biogeme are the default ones (BGW and Newton respectively) and the second derivative is set to 1. This option was chosen as we did not know then that it is not the best for optimization at this point of the project.

Utility functions and model parameters:

$$
\begin{aligned}
V_{\text{car}} &= \texttt{asc\_car} + \beta_{\texttt{tt\_car}} \cdot \text{time\_car} + \beta_{\texttt{cost}} \cdot \text{cost\_car} \\
V_{\text{bus}} &= \texttt{asc\_bus} + \beta_{\texttt{tt\_bus}} \cdot \text{time\_bus} + \beta_{\texttt{access}} \cdot \text{access\_bus} + \beta_{\texttt{cost}} \cdot \text{cost\_bus} \\
V_{\text{air}} &= \texttt{asc\_air} + \beta_{\texttt{tt\_air}} \cdot \text{time\_air} + \beta_{\texttt{access}} \cdot \text{access\_air} + \beta_{\texttt{cost}} \cdot \text{cost\_air} \\
V_{\text{rail}} &= \texttt{asc\_rail} + \beta_{\texttt{tt\_rail}} \cdot \text{time\_rail} + \beta_{\texttt{access}} \cdot \text{access\_rail} + \beta_{\texttt{cost}} \cdot \text{cost\_rail}
\end{aligned}
$$

Estimation and statistics:
Not all alternative specific constants (ASC) can be estimated from the dataset, we fix `asc_car` to 0. We estimate the 9 others parameters, with starting value 0 for all of them. We observe the estimation results in Table 2 and the statistics in Table 3. As the estimated parameters are almost equal in both libraries, as well as the Initial and Final log-likelihoods (LL), we can surely say that the models are equivalent. However, the optimisation times are very close to each other. The model is too simple to notice a significant difference of computing time. Therefore, we need to work on more complex models.

Parameter	Value in Apollo	Value in Biogeme
asc_car	0	0
asc_bus	0.475	0.471
asc_air	1.630	1.628
asc_rail	0.945	0.941
$\beta_{\texttt{tt\_car}}$	-0.00365	-0.00365
$\beta_{\texttt{tt\_bus}}$	-0.00885	-0.00881
$\beta_{\texttt{tt\_air}}$	-0.02069	-0.02069
$\beta_{\texttt{tt\_rail}}$	-0.0112	-0.0112
$\beta_{\texttt{access}}$	-0.0115	-0.0115
$\beta_{\texttt{cost}}$	-0.0339	-0.0339

Table 2: Parameter values for the `RP_Model` in both *Apollo* and *Biogeme* libraries.

Statistic	Value in Apollo	Value in Biogeme
Number of estimated parameters	9	9
Number of individuals	500	500
Number of observations	1000	1000
Initial log likelihood	-1170.86	-1170.86
Final log likelihood	-1025.76	-1025.756
AIC	2069.51	2069.513
BIC	2113.68	2113.683
Time of estimation (hh:mm:ss)	00:00:1,51	00:00:1,54
Optimization diagnostic	Maximum found	Maximum found

Table 3: Statistics results for the `RP_Model`

Remarks:

1. An extra step which is necessary when translating the Apollo script to Biogeme is to deal with NA values. In Apollo, NA values are replaced with 0. It is thus necessary to do the same in Biogeme, by using the `.fillna(0)` function in the script.

2. Even though the model is fairly straightforward, it is an excellent exercise for familiarizing oneself with the syntax of Biogeme and Apollo. However, the translation process can be challenging, as it is easy to lose information. For instance, overlooking the availabilities was a significant mistake that occurred multiple times.

4.3 MMNL Preference Model

In the context of a MMNL model, a random coefficient implies that the coefficient for a particular explanatory variable varies randomly across individuals or choice situations. This introduces heterogeneity in the population, allowing for differences in preferences or behaviors among individuals. Having random coefficient is a great way to add complexity to the model.

We choose a MMNL script from Apollo's website again, which we call `MMNL_Preference_Space` and which use the `apollo_swissRouteChoiceData` dataset (3.2). "`tt`", "`tc`", "`hw`" and "`ch`" respectively denote for travel time, travel cost, headway and number of interchanges. Note that in this case, we deal with panel data. The optimisation algorithm in Apollo and Biogeme are the default ones and the second derivative is set to 1

The following functions allow us to bring randomness into the script:

- `apollo_draws()` in Apollo: This function presents as a list, which takes into argument the number of draws, the type of draws, whether the draws are inter or intra-individuals.
- `bioDraws()` in Biogeme: This function takes into argument the name of the random coefficient we define, as well as the type of draws. In our case, the number of draws is chosen in the `.toml` file. The draws are inter-individual. There are other way to script this, which can be found in the Panda Documentation (Bierlaire, 2018).

We generate Halton draws which are often used in Monte-Carlo simulation. In section 4.4, we will try and implement different number of draws, but for this model we set it to 500. In the first case, we rename the `biogeme.toml` to `few_draws.toml`, where we set `[MonteCarlo] number_of_draws` = 500. In the Apollo case, we set `interNDraws = 500` directly in the script.

Utility functions and model parameters:

$$V_{\text{alt1}} = \beta_{\texttt{tt}} \cdot \text{tt}_1 + \beta_{\texttt{tc}} \cdot \text{tc}_1 + \beta_{\texttt{hw}} \cdot \text{hw}_1 + \beta_{\texttt{ch}} \cdot \text{ch}_1$$
$$V_{\text{alt2}} = \beta_{\texttt{tt}} \cdot \text{tt}_2 + \beta_{\texttt{tc}} \cdot \text{tc}_2 + \beta_{\texttt{hw}} \cdot \text{hw}_2 + \beta_{\texttt{ch}} \cdot \text{ch}_2$$

where, for $i \in \{\texttt{tt}, \texttt{tc}, \texttt{hw}, \texttt{ch}\}$:

- $\beta_{\texttt{i}} = -exp(\mu_{log(\beta_i)} + \sigma_{log(\beta_i)} \cdot \text{draws}_i)$
- $\mu_{log(\beta_i)}$ is the estimated mean for the log of β_i, set to -3 before estimation
- $\sigma_{log(\beta_i)}$ is the standard deviation of β_i, set to -0.01 before estimation

Estimation and statistics:
We estimate all parameters, initializing each at 0. The estimated parameters are presented in Table 4, and the statistical results are shown in Table 5. We note discrepancies between the estimated parameters from Apollo and Biogeme, as well as differences in the Initial and Final LL. These variations could stem from translation errors between Apollo and Biogeme, differences in optimization techniques, convergence error, or dataset specification issues. Despite the models not appearing fully equivalent, it is significant to highlight the substantial difference in estimation speed: Apollo completes the estimation in just **00:03:17**, whereas Biogeme takes **00:32:57**. This results is almost a 30-minute gap.

Parameter	Value in Apollo	Value in Biogeme
$\mu_{log(\beta_{tt})}$	-1.984	-2.541
$\sigma_{log(\beta_{tt})}$	-0.442	-0.577
$\mu_{log(\beta_{tc})}$	-1.016	-2.62
$\sigma_{log(\beta_{tc})}$	-0.991	-1.89
$\mu_{log(\beta_{hw})}$	-2.938	-3.07
$\sigma_{log(\beta_{hw})}$	-0.834	0.268
$\mu_{log(\beta_{ch})}$	0.631	0.251
$\sigma_{log(\beta_{ch})}$	0.858	0.801

Table 4: Parameter values for the `MMNL_Preference_Model` (final version)

Statistic	Value in Apollo	Value in Biogeme
Number of estimated parameters	8	8
Number of threads	8	8
Number of observations	388	388
Initial log likelihood	-2253.78	-1703.926
Final log likelihood	-1444.35	-1534.219
AIC	2904.7	3084.437
BIC	2953.97	3116.125
Time of estimation (hh:mm:ss)	00:03:17	00:32:57

Table 5: Model Estimation Results `MMNL_Preference_Model` (final version)

Remarks: Some trials and errors have been done in order to try and correct for the encountered problem. After running the Apollo script from the first time, we used the estimated parameters as starting values for the script in Biogeme. However, this was not sufficient to compensate for the unmatched results. These "test scripts" can be found on the GitHub.

4.4 Swissmetro

After selecting scripts from the Apollo website and translating them to Biogeme, we decide to reverse the process by choosing a model from the Biogeme website. This decision aims to address the non-equivalence observed in the previous model. Suspecting a dataset issue, we intend to test datasets originating from Biogeme instead (3.3).

4.4.1 First version: 1 random coefficient

The selected model is an MMNL called `swissmetro`, incorporating only one random component with Halton draws for now. We will test different draw numbers to observe optimization time variations across packages. After consulting with Michel Bierlaire, we set the second derivative to 0 in Biogeme to speed up optimization, switching the algorithm to BFGS. For reasons to be discussed later, Apollo's algorithm is also changed from BGW to BFGS.

Utility functions and model parameters:

$$V_{\text{train}} = \texttt{asc\_train} + \beta_{\texttt{time\_rnd}} \cdot train\_tt\_scaled + \beta_{\texttt{cost}} \cdot train\_cost\_scaled$$
$$V_{\text{sm}} = \texttt{asc\_sm} + \beta_{\texttt{time\_rnd}} \cdot sm\_tt\_scaled + \beta_{\texttt{cost}} \cdot sm\_cost\_scaled$$
$$V_{\text{car}} = \texttt{asc\_car} + \beta_{\texttt{time\_rnd}} \cdot car\_tt\_scaled + \beta_{\texttt{cost}} \cdot car\_cost\_scaled$$

The randomness is brought using normally distributed Haltons draws (with base 3):

$$\beta_{\texttt{time\_rnd}} = \beta_{\texttt{time}} + \beta_{\texttt{time\_s}} \cdot \texttt{bioDraws('B\_TIME\_RND', 'NORMAL\_HALTON3')}$$

Instead of assuming fixed effect for travel time, i.e. assuming that β_{time_rnd} is constant across all individuals, the model allows for the possibility that different individuals have different sensitivities or preferences regarding travel time. This reflects the heterogeneity in preferences or behaviors across the sample.

Estimation and statistics:
We start by comparing the models for 100 draws, to make sure these are equivalent before rising the number of draws. We observe in Tables 6 and 7 that the estimated parameters are close to each other, as well as the Initial and Final LL.

Parameter	Value in Apollo	Value in Biogeme
asc_sm	0	0
asc_car	0.281	0.284
asc_train	-0.572	-0.565
$\beta_{\texttt{time}}$	-3.165	-3.194
$\beta_{\texttt{time\_s}}$	3.718	3.706
$\beta_{\texttt{cost}}$	-1.650	-1.648

Table 6: Parameter values for the `swissmetro` Model, 100 draws

Statistic	Value in Apollo	Value in Biogeme
Number of estimated parameters	5	5
Number of threads	8	8
Number of individuals	441	441
Number of observations	752	752
Initial log likelihood	-5782.84	-5784.807
Final log likelihood	-4363.05	-4362.971
AIC	8736.11	8735.94
BIC	8770.21	8759.06

Table 7: Model Estimation Results `swissmetro` Model, 100 draws

Estimation time and Final LL:
Now that we know the models are equivalent, we can begin investigating estimation speed. We test various numbers of draws: 100, 500, 1'000, 2'000, 5'000, and 10'000. Table 8 shows the optimization times for Apollo and Biogeme, which are further illustrated in Figure 1. Initially, the estimation time gap between Apollo and Biogeme is small. However, as the number of draws increases, the gap widens, reaching up to **44 minutes** at 5'000 draws. Interestingly, the gap narrows again at 10'000 draws. As the estimation time for Apollo appears linear initially, the result for 10'000 draws might be an outlier. However, we can confidently assert that in this case, Apollo is better optimized than Biogeme.

Number of draws	Estimation Time Apollo	Estimation Time Biogeme
100	00:00:36	00:01:52
500	00:02:00	00:06:04
1'000	00:03:01	00:11:44
2'000	00:06:16	00:34:53
5'000	00:22:24	01:06:58
10'000	01:17:16	01:20:22

Table 8: Estimation time of `swissmetro`, MMNL model with 1 random coefficient, for different number of draws

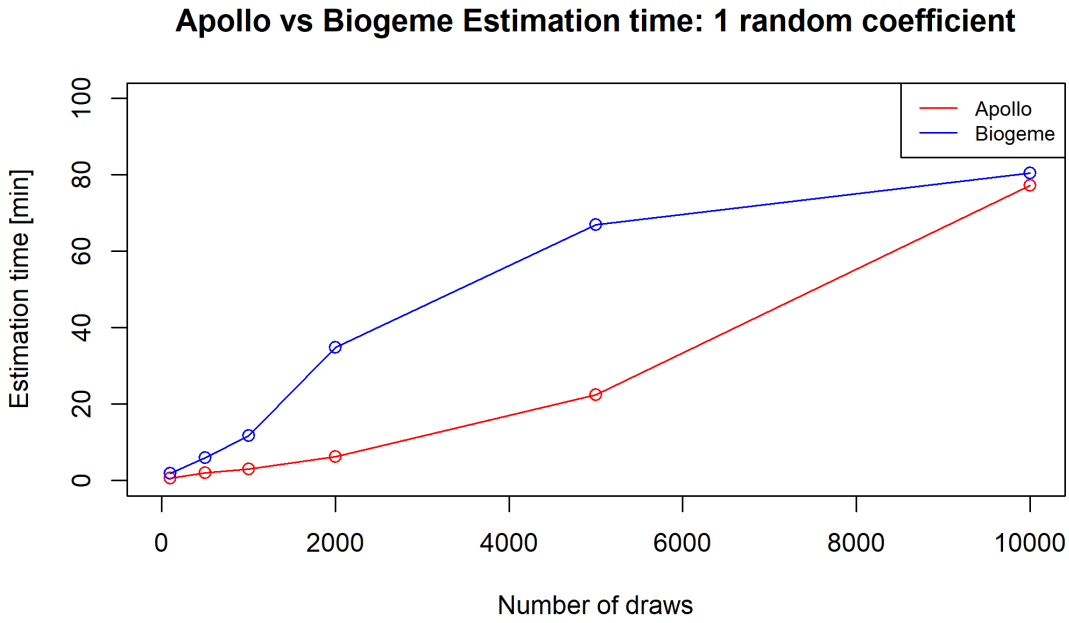

Figure 1: Estimation time plot for 100, 500, 1'000, 2'000, 5'000 and 10'000 draws, 1 random coefficient

Another interesting analysis is looking at how the Final LL changes with the growth of the number of draws. In a MMNL model with random coefficients generated by Halton draws, a higher number of draws should improve estimation by enhancing the accuracy of integral approximations, reducing simulation error, and providing more precise and stable parameter estimates. This leads to a better fit and more reliable reflection of the true underlying distribution of the random coefficients. We thus expect the Final LL to significantly decrease as we increase the draws sample size. However, this is surprisingly not the case. For Apollo, the Final LL changes only slightly from -4363.05 (100 draws) to -4359.58 (10'000 draws). Similarly, for Biogeme, it shifts from -4362.971 to -4360.266.

4.4.2 Second version: 4 random coefficients

One random coefficient effectively brings out the difference in estimation time between the packages. However, it is important to test with more than one random coefficient: adding random coefficients can lead to a more nuanced and accurate representation of the relationship between the predictors and the outcome variables, resulting in improved model performance and greater insights into the events under investigation. It is also more demanding for computation, and therefore will highlight the estimation time gap better. We create `swissmetro_2` based on the original `swissmetro` model, with additional random components.

Utility functions and model parameters:
The first thought that comes to mind regarding which random coefficient to add is the cost coefficient. Indeed, following the same logic as the first version of the `swissmetro` model, the model acknowledges that different people may have varying sensitivities or preferences regarding travel cost. We also add heterogeneity to the ASCs of train and sm directly. The new model `swissmetro_2` has the following utilities:

$$\begin{aligned} V_{\text{train}} &= \texttt{asc\_train} + \beta_{\texttt{time\_rnd}} \cdot train\_tt\_scaled + \beta_{\texttt{cost\_rnd}} \cdot train\_cost\_scaled + \sigma_{\texttt{train\_rnd}} \\ V_{\text{sm}} &= \texttt{asc\_sm} + \beta_{\texttt{time\_rnd}} \cdot sm\_tt\_scaled + \beta_{\texttt{cost\_rnd}} \cdot sm\_cost\_scaled + \sigma_{\texttt{sm\_rnd}} \\ V_{\text{car}} &= \texttt{asc\_car} + \beta_{\texttt{time\_rnd}} \cdot car\_tt\_scaled + \beta_{\texttt{cost\_rnd}} \cdot car\_cost\_scaled \end{aligned}$$

where we define

$$\begin{aligned} \beta_{\texttt{cost\_rnd}} &= \beta_{\texttt{cost}} + \beta_{\texttt{cost\_s}} \cdot \texttt{bioDraws('B\_COST\_RND', 'NORMAL')} \\ \sigma_{\texttt{train\_rnd}} &= \sigma_{\texttt{train}} \cdot \texttt{bioDraws('SIGMA\_TRAIN\_RND', 'NORMAL')} \\ \sigma_{\texttt{sm\_rnd}} &= \sigma_{\texttt{sm}} \cdot \texttt{bioDraws('SIGMA\_SM\_RND', 'NORMAL')} \end{aligned}$$

Estimation and statistics:
Following the same approach as `swissmetro`, we begin by comparing the models based on 100 draws. Although the estimated parameters in Table 6 may not appear to be exactly accurate, the Final LL, as well as the AIC and BIC in Table 10, confirm the equivalence

between the models. The translation is successful, allowing us to proceed with the analysis of estimation times. Furthermore, we notice that the `swissmetro_2` model is a better fit, as its Final LL is around -3603 (compared to -4363 for `swissmetro`).

Parameter	Value in Apollo	Value in Biogeme
asc_sm	0	0
asc_car	0.7906	0.544
asc_train	-0.788	-1.563
β_{time}	-5.359	-5.225
$\beta_{\text{time\_s}}$	2.773	3.00605
β_{cost}	-4.050	-3.864
$\beta_{\text{cost\_s}}$	3.628	3.223
σ_{car}	3.205	3.374
σ_{sm}	2.7004	2.691

Table 9: Parameter values for the `swissmetro_2` Model, 100 draws

Statistic	Value in Apollo	Value in Biogeme
Number of estimated parameters	8	8
Number of threads	8	8
Initial log likelihood	-4851.99	-4896.67
Final log likelihood	-3603.42	-3603.30
AIC	7222.85	7222.61
BIC	7277.41	7259.9

Table 10: Model Estimation Results `swissmetro_2` Model, 100 draws

Estimation time and Final LL:
This time, we run the script for 100, 500, 1'000, 2'000, 5'000, and 6'000 draws. We cannot exceed 10'000 draws due to a crashing issue in Apollo, which probably stems from either the computer's performance or from Apollo itself. Estimation times and visualization can be seen in Table 11 and Figure 2. For both Apollo and Biogeme, the estimation times are nearly linear. Initially, the time difference is small but then increases exponentially with the number of draws. This was anticipated, as it had already been observed with the `swissmetro` model. The largest variance occurs at 6'000 draws, with Biogeme taking **253 minutes** (or **4 hours and 13 minutes**) longer than Apollo.
Regarding the Final LL, we observe an improvement from 100 draws to 500. Specifically, Apollo's Final LL improves from -3603.42 to -3574.31, while Biogeme's improves from -3640.93 to -3575.82. However, from 500 to 6'000 draws, the change is minimal. For Apollo, the Final LL changes slightly from -3574.31 to -3571.19, and for Biogeme it changes from -3575.82 to -3567.34.

Number of draws	Estimation Time Apollo	Estimation Time Biogeme
100	00:00:50	00:07:08
500	00:03:08	00:23:57
1'000	00:05:28	00:49:20
2'000	00:12:45	01:32:37
5'000	00:25:36	04:10:35
6'000	00:43:57	04:57:19

Table 11: Estimation time of `swissmetro_2`, MMNL model with 4 random coefficients, for different number of draws

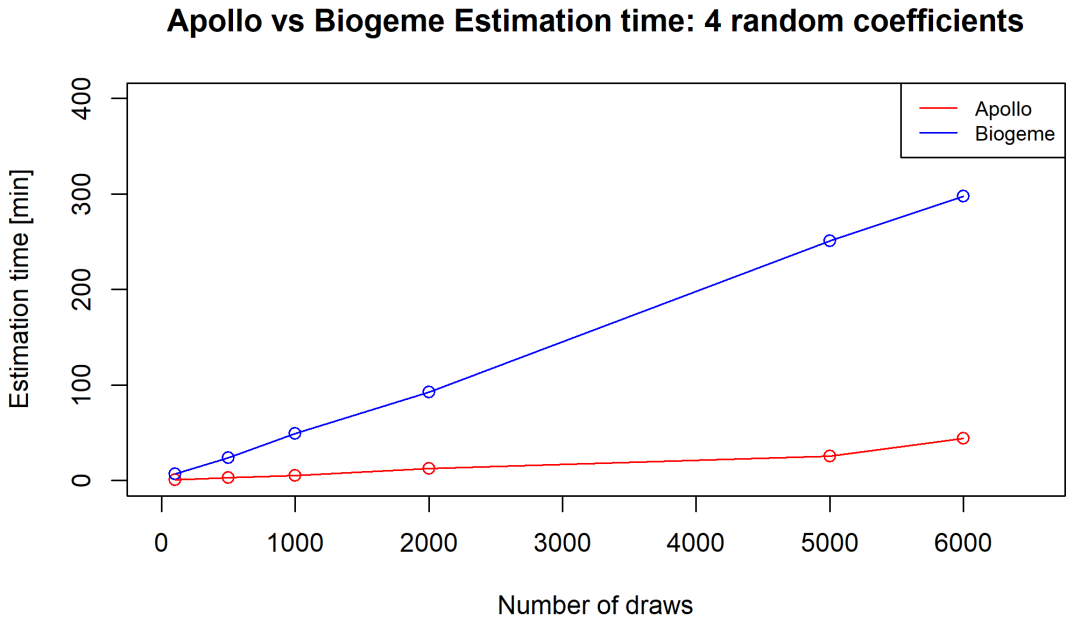

Figure 2: Estimation time of `swissmetro_2`, MMNL model with 4 random coefficients, for different number of draws

Remarks:

- To have an approximate idea of how long would 10'000 draws take, we still run the script of `swissmetro_2` on a more powerful computer, with the same parameters as in Table 1. Apollo takes **01:15:00** to run the script, and Biogeme takes **11:53:54**.
- The scripts were run only once or twice, on the same computer and in the same conditions (battery plugged in, flight mode). This is to ensure the results are not affected by anything else. However, it would be wise to run all the scripts multiple times and take a median of the estimation time. This would give more accurate results.
- The Apollo algorithm was changed from BGW to BFGS because of a convergence error. We get the message `"Model diagnosis: False convergence"` even thought

the Final LL does not seem to be affected. We are not sure why this error occurs.

- In the `swissmetro` script, the draw type is `'NORMAL_HALTON3'`, unlike in `swissmetro_2`. This change is necessary due to issues with adding a new random coefficient to the original model. When creating `swissmetro_2` from `swissmetro`, we initially use `'NORMAL_HALTON3'`. We change β_{cost} to the random coefficient β_{cost_rnd} and observe significant differences in Final LL and estimated parameters between Apollo and Biogeme, indicating non-equivalent models. Through tests, we find that using `'NORMAL'` draws in `bioDraws()` instead of `'NORMAL_HALTON3'` makes the models equivalent.

4.5 CPU and RAM

Another idea to compare both package performances is to implement a function to measure CPU and RAM usage as the script runs. Indeed, understanding CPU and RAM usage helps understanding performance. If a script consumes excessive CPU, it may indicate inefficient algorithms or operations that could be optimized. High RAM usage might suggest the need for more efficient data handling or memory management. To do so, we script the function `CPU_RAM_monitor_table()` in Python. The function measures CPU and RAM usage in percentages every x seconds, depending on the model it is used for. It also takes into argument the time of required duration of running.

RP Model (Section 4.2)**:** The function measures the CPU and RAM percentage every 0.5 seconds. This time gap is chosen because the estimation time is very short. Even this might be not very helpful and significant, it is a great way to see if the `CPU_RAM_monitor()` function is well implemented. Recall that the estimation times for the `RP_model` are 00:00:1,51 and 00:00:1,54 for Apollo and Biogeme respectively. This is very short indeed, we thus decide to run the monitoring function for 5 seconds and we get Table 12.

Time(s)	CPU(%) Apollo/Biogeme	RAM(%) Apollo/Biogeme
0	2.7 / 3.7	54.3 / 47.0
0.5	2.3 / 7.8	54.4 / 47.0
1	3.1 / 3.8	54.1 / 47.0
1.5	2.0 / 6.6	54.1 / 47.0
2	9.5 / 8.7	54.1 / 47.2
2.5	9.7 / 3.9	54.4 / 47.4
3	10.2 / 13.6	54.3 / 47.6
3.5	7.8 / 4.6	54.4 / 47.6
4	3.4 / 1.1	54.4 / 47.6
4.5	2.3 / 4.1	54.4 / 47.6

Table 12

MMNL Preference Space Model (Section 4.3)**:** In this case, the measurements are more meaningful. The computing times being 00:03:17 and 00:32:57 for Apollo and Biogeme, it leaves more time for usage measurements. We decide to measure the CPU and RAM every 20 seconds and during 4 minutes for Apollo, every minute during 50 minutes for Biogeme. The visualization of the measurements can be seen in Figures 3 and 4. We observe that the RAM usage is slightly higher for Apollo than Biogeme, however it stays approximately within the same range. As for the CPU, it is clearly more solicited by the Biogeme script. There are notable peaks as well. According to Michel Bierlaire, it might be due to the second derivative being turned to 1.

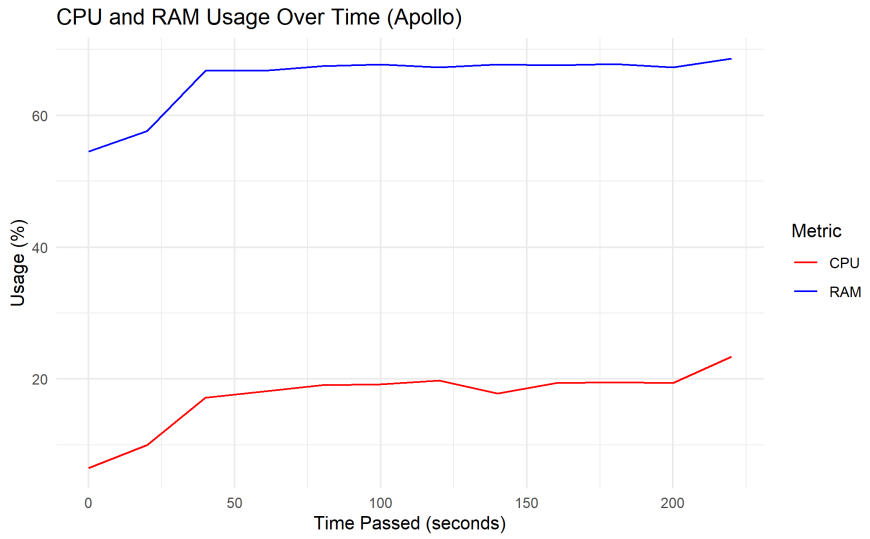

Figure 3: CPU and RAM usage for Apollo, for the `MMNL_Preference_Space` model

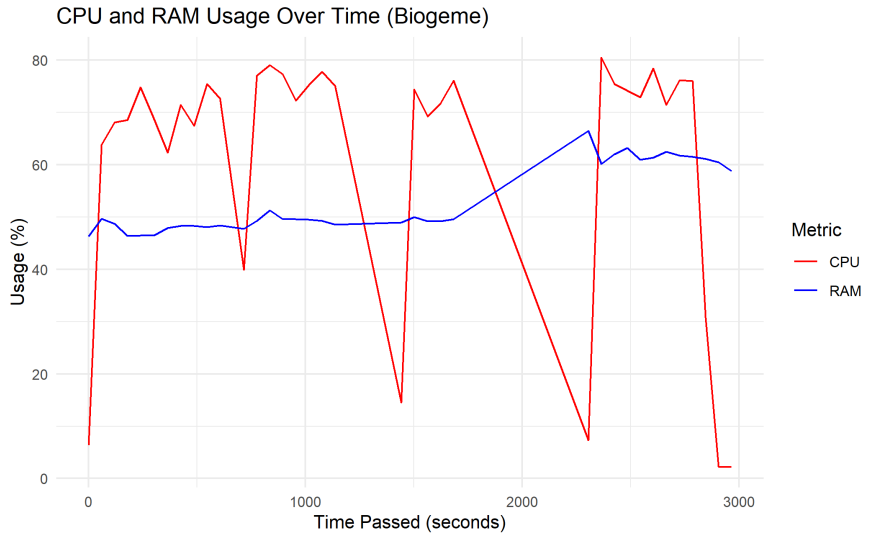

Figure 4: CPU and RAM usage for Biogeme, for the `MMNL_Preference_Space` model

Swissmetro 2 Model (Section 4.4.2)**:** For this analysis, we choose to perform CPU and RAM measurements for 1'000 draws. Given that Apollo's computation time is 00:05:28, we record measurements every 5 seconds for 10 minutes. In the case of Biogeme, with a computation time of 00:49:20, we collect measurements every 20 seconds for 55 minutes. The results are presented in Figures 5 and 6. To further our investigation, we calculate various statistics, which are summarized in Table 13.

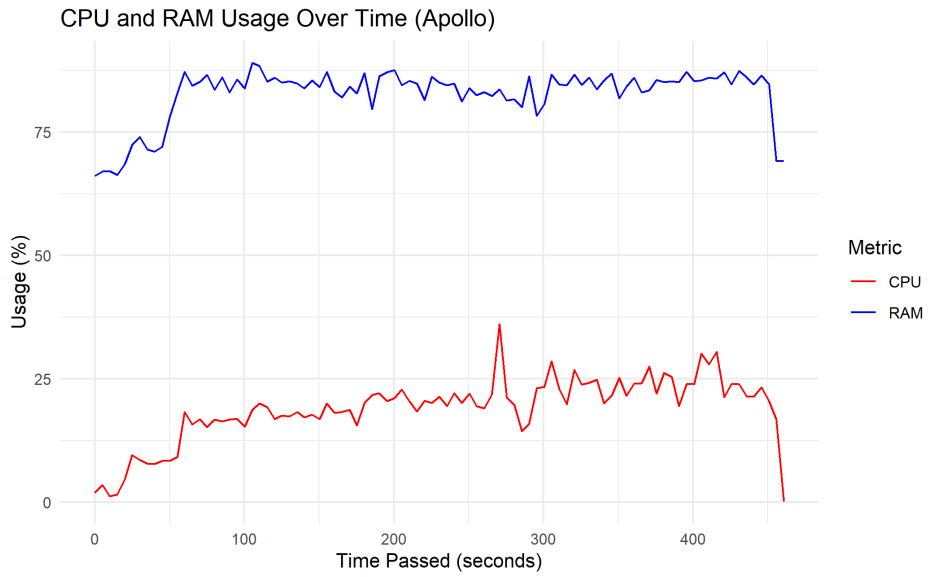

Figure 5: CPU and RAM usage for Apollo, for the `swissmetro_2` model

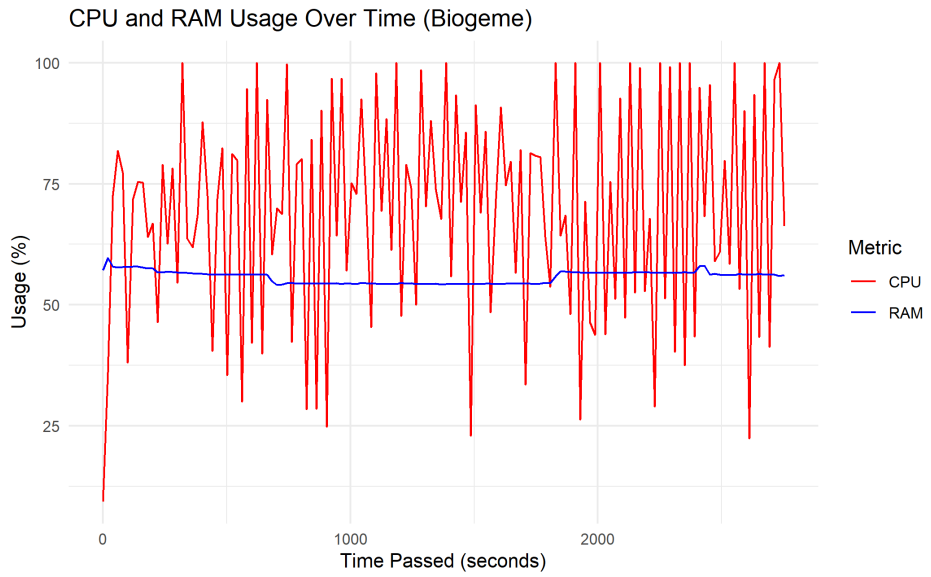

Figure 6: CPU and RAM usage for Biogeme, for the `swissmetro_2` model

Recall that for the `MMNL_Preference_Space` model, we determine that the CPU's peaks in the Biogeme script might be caused by setting the second derivative to 1. This phenomenon is also observed in the `swissmetro_2` case, even with the second derivative being set to 0. As expected, Biogeme demands significantly more CPU resources than Apollo, with an average usage of 69.41 compared to Apollo's 18.87. However, Apollo shows higher RAM usage, averaging 82.65 versus Biogeme's 55.74. This suggests that Biogeme's implementation is computation-intensive but not memory-demanding, while Apollo appears to hold a substantial amount of data in memory without performing extensive calculations or operations at any given time.

	mean	median	variance	max
Apollo (CPU)	18.87	20.00	43.84	36.10
Biogeme (CPU)	69.41	71.65	500.70	100
Apollo (RAM)	82.65	84.70	30.64	89
Biogeme (RAM)	55.74	56.20	1.52	59.70

Table 13: Statistics Summary of CPU and RAM usage for `swissmetro_2`, 1000 draws

Remarks: We extended the runtime for `CPU_RAM_monitor()` to ensure capturing the entire data. The results are stored in an Excel table and then visualized using R Studio. Any additional measurements taken after the script has finished running are removed from the Excel table before plotting. Note that the current setup for monitoring CPU and RAM is not optimal. We manually initiate the script and must predetermine its runtime duration. Consequently, prior knowledge of the script's computational time is necessary. Here are several suggestions to enhance the `CPU_RAM_monitor()` function, which unfortunately have not been implemented due to time constraints:

- Having an equivalence of the `CPU_RAM_monitor()` function RStudio. This would be helpful to have the function implemented in the Apollo scripts directly.
- Having an automatic starting and stopping time before and after the estimation part of the script, instead of a pre-chosen running duration
- Comparing CPU and RAM usage with the computations performed in the scripts side by side would help identify the reasons behind certain patterns such as peaks, drops, level, etc.

5 Estimation Features

Comparing the pre and post-estimation features of Apollo and Biogeme can provide valuable insights into the strengths and limitations of each package. Understanding how these packages handle pre and post-estimation inform researchers about the suitability of each tool for their specific needs and workflows. To compare these features, we read through Hess & Palma (2023) and Bierlaire (2018) and try to summarize the features in the following subsections.

5.1 Pre-estimation

Pre-analysis of choices: When working with choice data, either labeled or unlabeled, it is beneficial to analyze the choices before estimating the model to see if individuals' characteristics differ by the alternatives they choose. The function `apollo_choiceAnalysis()` in Apollo facilitates this.

Starting Value Search: A common strategy to mitigate the risk of converging to a suboptimal local maximum involves initiating the optimization process from multiple candidate points and selecting the solution with the highest likelihood. However, this method is computationally demanding. To address this, algorithms have been developed to dynamically discard unpromising candidates. The function `apollo_searchStart()` implements a simplified version of the algorithm proposed by Bierlaire et al. (2010) with the key distinction being that `apollo_searchStart()` employs only two of the three candidate tests described by Bierlaire et al. (2010).

5.2 Post-estimation

Likelihood Ratio Tests Between Two Models: It is possible to compare a model (unrestricted) with its restricted version via a likelihood ratio test. This is facilitated in both Apollo and Biogeme, using the `apollo_lrTest()` and `biogeme.tools.likelihood_ratio_test()` functions, respectively.

Ben-Akiva & Swait Test: Likelihood ratio tests can be restrictive, as they require comparing a model with its nested version. The Ben-Akiva & Swait test allows for non-nested model comparison using the `apollo_basTest()` function in Apollo. Currently, there does not appear to be an equivalent function in Biogeme.

Model Prediction: Model prediction involves using the estimated parameters from the model to forecast the choices that individuals will make under different scenarios or conditions. This is done in Apollo via the `apollo_prediction()` function. As for Biogeme, it is done through `Biogeme.simulate()`.

Market share: In Biogeme, we can compute market shares using sample enumeration. We need a representative sample of individuals from the population, where the real values of all variables involved in the model are known. This sample may be same as the one used for model estimation only if it contains revealed preferences data, as real values are necessary for the calculation of indicators. The implementation is explained in Section 2 of Bierlaire (2018). As for Apollo, the function `apollo_choiceAnalysis()`compares market shares across subsamples in dataset, and conducts statistical tests. Furthermore, it is possible to compare the shares predicted by the model with the shares observed in the data, and conducts statistical tests. The `apollo_sharesTest()` function is a tool for performing this test, which is useful for refining the model.

Comparison of model fit: In Apollo, with the `apollo_fitsTest()`, it is possible to compare the fit of the estimated model with different subset of the data.

Cross Validation: Cross-validation helps in detecting overfitting, provides insight into how the model will perform on unseen data, and ensures that the evaluation is more robust and reliable. A way to do cross-validation is by splitting the dataset into an estimation sample and a validation sample. It estimates the model using the estimation sample, then assess its fit on the validation sample. To avoid bias from a single split, it repeats this process with multiple random splits, which also allows for calculating a confidence interval for the out-of-sample fit. The implementation can be done in both Apollo and Biogeme, with the functions `apollo_outOfSample()` and `validate()` respectively.

Delta method: It is interesting to compute derived functions, such as ratios of coefficients. These ratios can reveal marginal rates of substitution and, when a cost coefficient is used as the denominator, measures of willingness-to-pay (WTP). It is crucial to calculate standard errors for these derived measures accurately. The Delta method, allows for this calculation in a straightforward and precise manner, providing exact results rather than approximations. This is possible in Apollo with the `apollo_deltaMethod()` function. As for Biogeme, indicators (such as WTP and elasticity) are manually computed.

Conditionals for random parameters: Apollo has a method for estimating posteriors for continuous mixture models and as well as latent class. This is done through the function `apollo_conditionals()`. Further information on the theorical aspect as well as the implementation can be found in Section 9.14 (Hess & Palma, 2023).

6 Encountered difficulties and improvements

We encountered several challenges throughout the project, many of which were previously mentioned.

Firstly, ensuring the equivalence of models in Apollo and Biogeme proved difficult. Despite matching optimization parameters (Table 1), differences in results persisted, as seen with the `MMNL_Preference_Space` model. There is no direct method to verify model equivalence; the only approach is to estimate and compare the results.

Another problem we encountered is the convergence error for the `swissmetro_2` model in Apollo. This error might indicate that the optimization process was stuck in a local minimum or an area of the parameter space where the gradients are very small (but not zero). According to Wikipedia, BFGS has mechanisms to escape such regions more effectively than BGW. This might explain why the convergence problem was solved by switching algorithm.

Furthermore, we cannot explain why we needed to change the draw type from Normal Halton to Normal in the `swissmetro_2` model. We created the `swissmetro_2` script by adding three new random coefficients to the `swissmetro` script, which used Normal Halton draws and was verified as equivalent in both Apollo and Biogeme. This issue arose even when adding just one random coefficient, suggesting that Biogeme struggles with multiple random coefficients when paired with Normal Halton draws.

The computer we use for this project has the following specifications:

- Processor: 11^{th} Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz
- RAM: 16.0 GB

We encounter numerous crash issues, obstructing our ability to obtain the necessary results. For instance, not being able to measure the estimation time for the 10'000 draws of `swissmetro_2` with Apollo makes it difficult to determine whether the time difference between Apollo and Biogeme would decrease over time. After testing on a computer with less RAM (8GB), the script continued to crash, indicating an error related to insufficient memory. This confirms our hypothesis that Apollo is not managing memory effectively.

Beyond scripting issues, it would be valuable to delve deeper into the CPU and RAM usage in both Apollo and Biogeme. This project's goal is to compare Biogeme's performance to Apollo in terms of estimation time. While Biogeme's efficient RAM usage is a notable strength, its heavy CPU utilization likely contributes to the longer estimation times observed compared to Apollo. Understanding the specific computational demands on the CPU and RAM for each software could reveal optimization opportunities and clarify the performance differences between the two.

7 Conclusion

The comparative analysis between Biogeme and Apollo for estimating discrete choice models has shed light on several key aspects regarding their performance, computational efficiency, and ease of use. Despite the initial aim to investigate differences in performance and computational speed between the two packages, as well as potential differences in results and model statistics, the project encountered various challenges that impacted the scope and conclusions. While Biogeme and Apollo offer valuable tools for estimating discrete choice models, the project highlighted the complexity and nuances involved in comparing their performance and results. Addressing challenges related to model equivalence, convergence errors, draw type requirements, and resource utilization is crucial for optimizing software performance and facilitating more informed model selection in practice.

Moving forward, continued efforts to enhance documentation, improve script translation processes, and refine optimization algorithms could contribute to more robust and efficient estimation of discrete choice models using both Biogeme and Apollo.

A Appendix

Thanks

I extend my heartfelt gratitude to my supervisors Negar Rezvany and Evangelos Paschalidis for their support throughout this project. Their weekly meetings provided invaluable guidance, allowing me to navigate through challenges with clarity. Their willingness to address my questions promptly and offer assistance whenever needed was truly commendable, fostering an environment of collaboration and growth. Moreover, I am grateful for their enthusiasm in exploring new aspects of choice modeling and optimization, which broadened my understanding and enhanced the quality of my work. I am deeply appreciative of their dedication and encouragement! I also thank Michel Bierlaire for accepting this project and providing valuable insights. His clarifications addressed my questions, allowing us to continue with a clearer perspective and greater ease.

References

Axhausen, K. W. ., Bierlaire, M. ., & Abay, G. (2001). The acceptance of modal innovation The case of Swissmetro Conference Paper. Retrieved from https://doi.org/10.3929/ethz-a-004238511 doi: 10.3929/ethz-a-004238511

Axhausen, K. W. ., Hess, S. ., König, A. ., Abay, G. ., Bates, J. ., Bierlaire, M., ... Bates, J. J. (2006). State of the art estimates of the Swiss value of travel time savings Conference Paper State of the art estimates of the Swiss value of travel time savings. Retrieved from https://doi.org/10.3929/ethz-a-005240029 doi: 10.3929/ethz-a-005240029

Bierlaire, M. (2018). *Calculating indicators with PandasBiogeme* (Tech. Rep.).

Bierlaire, M. (2023). *A short introduction to Biogeme* (Tech. Rep.).

Bierlaire, M., Themans, M., & Zufferey, N. (2010, 12). A heuristic for nonlinear global optimization. *INFORMS Journal on Computing*, *22*(1), 59–70. doi: 10.1287/ijoc.1090.0343

Bierlaire Michel. (2018, 6). biogeme_swissmetro.

Hess, S., & Palma, D. (2023). *Apollo: a flexible, powerful and customisable freeware package for choice model estimation and application version 0.2.9 User manual* (Tech. Rep.). Retrieved from www.ApolloChoiceModelling.com

Molloy, J., Becker, F., Schmid, B., & Axhausen, K. W. (2021, 6). mixl: An open-source R package for estimating complex choice models on large datasets. *Journal of Choice Modelling*, *39*. doi: 10.1016/j.jocm.2021.100284

Reul, J. P., Grube, T., Linßen, J., & Stolten, D. (2023, 8). MODE.behave: A Python Package for Discrete Choice Modeling. *Journal of Open Source Software*, *8*(88), 5265. doi: 10.21105/joss.05265