Lab 3: Introduction to Biogeme

Meritxell Pacheco Paneque

Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
École Polytechnique Fédérale de Lausanne

September 29, 2020

Outline

1. Using Biogeme
    - Running Biogeme
    - Data file
    - Model file
2. Today's lab
    - Case study (binary logit)
    - Reading and modifying files

Using Biogeme

BIOGEME

Import the modules

```
import pandas as pd
import biogeme.database as db
import biogeme.biogeme as bio
from biogeme.expressions import Beta, DefineVariable
from biogeme.models import loglogit
```

Jupyter notebooks

- We recommend using Jupyter notebooks for working with Biogeme
- Open a terminal or Anaconda prompt window and type:

```
jupyter notebook
```

- Feel free to use any coding environment/editor you are familiar with

Biogeme files

- The Biogeme model notebook `model.ipynb` (or model python file `model.py`) contains the model specification, and reads:
  - a file containing the data `data.dat`
- Biogeme automatically generates:
  - A report file containing the results of the maximum likelihood estimation: `model.html`
  - A binary `pickle` file which can be loaded into Python, containing the same information: `model.pickle`

How to run Biogeme?

- **Jupyter notebook:** simply select the cell, and hit Run (or Shift + Enter on keyboard)
- **Command line (using Python):** Open a terminal or Anaconda prompt window and type

```
python model.py
```

Data file (1)

- File extension `.dat`
- It contains the data (*observations*)
- One observation per row
- First row contains column (variable) names
- Each row must contain a choice indicator

- **Example:** Netherlands transportation mode choice data
  - Choice between rail and car
  - 228 observations

Data file (2)

netherlands.dat

```
id       choice  rail_cost   rail_time   car_cost   car_time
1        0       40          2.5         5          1.167
2        0       35          2.016       9          1.517
3        0       24          2.017       11.5       1.966
4        0       7.8         1.75        8.333      2
5        0       28          2.034       5          1.267
...
219      1       35          2.416       6.4        1.283
220      1       30          2.334       2.083      1.667
221      1       35.7        1.834       16.667     2.017
222      1       47          1.833       72         1.533
223      1       30          1.967       30         1.267
```

Data file (3)

netherlands.dat

id	choice	rail_cost	rail_time	car_cost	car_time
1	0	40	2.5	5	1.167
2	0	35	2.016	9	1.517
3	0	24	2.017	11.5	1.966
4	0	7.8	1.75	8.333	2
5	0	28	2.034	5	1.267
...					
219	1	35	2.416	6.4	1.283
220	1	30	2.334	2.083	1.667
221	1	35.7	1.834	16.667	2.017
222	1	47	1.833	72	1.533
223	1	30	1.967	30	1.267

Unique identifier of observations

Data file (4)

netherlands.dat

id	choice	rail_cost	rail_time	car_cost	car_time
1	0	40	2.5	5	1.167
2	0	35	2.016	9	1.517
3	0	24	2.017	11.5	1.966
4	0	7.8	1.75	8.333	2
5	0	28	2.034	5	1.267
...					
219	1	35	2.416	6.4	1.283
220	1	30	2.334	2.083	1.667
221	1	35.7	1.834	16.667	2.017
222	1	47	1.833	72	1.533
223	1	30	1.967	30	1.267

Choice indicator, 0: car and 1: train

Model file

- Jupyter notebook with file extension `.ipynb` (or python file with extension `.py`)
- Must be consistent with the data file
- Contains deterministic utility specifications, model type, etc.
- **Example:** Netherlands transportation mode choice data
  - Travel times and travel costs are used as explanatory variables
  - The deterministic utility specifications are

$$\begin{aligned} V_{\text{car}} &= \text{ASC}_{\text{car}} + \beta_{\text{cost}}\text{cost}_{\text{car}} + \beta_{\text{time}}\text{time}_{\text{car}} \\ V_{\text{rail}} &= \beta_{\text{cost}}\text{cost}_{\text{rail}} + \beta_{\text{time}}\text{time}_{\text{rail}} \end{aligned}$$

Today's lab


Alkmaar
E22
Amsterdam
E35
E232
E232
E30
Netherlands
Hague
Rotterdam
E31
E311
E25
Eindhoven
Duisburg
E19

Binary Logit (the Netherlands case study) - files

1. Open the instructions file `03Lab2020.pdf` (under *Instructions Lab 3*)
2. Download the files for this case study (under *Case study Lab 3*) and copy them in a directory of your choice (e.g., Desktop)
   - `binary_generic_netherlands.ipynb` (model notebook)
   - `binary_generic_netherlands.py` (model Python file if needed)
   - `03Lab2020_solution.pdf` (description file)
3. The dataset (and its description) can be found in http://biogeme.epfl.ch/data.html

Binary logit (the Netherlands case study) - tasks

1. Go through the Jupyter notebook (check the file `03Lab2020_solution.pdf` to see the model specifications)
2. Run the notebook
3. Open the generated `.html` file and interpret the results (check the file `03Lab2020_solution.pdf` for help with the interpretations)
4. Develop other model specifications following the instructions file

Appendix

Model file: setup (binary_generic_netherlands.py)

```
import pandas as pd
import biogeme.database as db
import biogeme.biogeme as bio
from biogeme.expressions import Beta, DefineVariable
from biogeme.models import loglogit

pandas = pd.read_table("netherlands.dat")
database = db.Database("netherlands",pandas)
pd.options.display.float_format = '{:.3g}'.format

globals().update(database.variables)

exclude = sp != 0
database.remove(exclude)
```

Model file: parameters (binary_generic_netherlands.py)

```
# Parameters to be estimated
# Arguments:
#   1  Name for report. Typically, the same as the variable
#   2  Starting value
#   3  Lower bound
#   4  Upper bound
#   5  0: estimate the parameter, 1: keep it fixed
ASC_CAR  = Beta('ASC_CAR',0,None,None,0)
ASC_RAIL  = Beta('ASC_RAIL',0,None,None,1)
BETA_COST  = Beta('BETA_COST',0,None,None,0)
BETA_TIME  = Beta('BETA_TIME',0,None,None,0)
```

Model file: expressions (binary_generic_netherlands.ipynb)

```
# Define here arithmetic expressions for name that are not directly
available from the data
rail_time  = DefineVariable('rail_time', (rail_ivtt + rail_acc_time)
              +  rail_egr_time, database)
car_time  = DefineVariable('car_time', car_ivtt  + car_walk_time,database)
rate_G2E = DefineVariable('rate_G2E', 0.44378022,database)
car_cost_euro = DefineVariable('car_cost_euro', car_cost*rate_G2E,
                 database)
rail_cost_euro = DefineVariable('rail_cost_euro', rail_cost*rate_G2E,
                 database)
```

Model file: utilities, model (binary_generic_netherlands.ipynb)

```
# Utilities
Car = ASC_CAR  + BETA_COST * car_cost_euro + BETA_TIME * car_time
Rail = ASC_RAIL + BETA_COST * rail_cost_euro + BETA_TIME * rail_time
V = {0: Car,1: Rail}
av = {0: 1,1: 1}

# The choice model is a logit, with availability conditions
logprob = loglogit(V,av,choice)
biogeme  = bio.BIOGEME(database,logprob)
biogeme.modelName = "binary_generic_netherlands"
results = biogeme.estimate()
```

Model file: estimation, output (binary_generic_netherlands.ipynb)

```
# Get the results in a pandas table
pandasResults = results.getEstimatedParameters()
print(pandasResults)
print(f"Nbr of observations: {database.getNumberOfObservations()}")
print(f"LL(0) =  {results.data.initLogLike:.3f}")
print(f"LL(beta) = {results.data.logLike:.3f}")
print(f"rho bar square = {results.data.rhoBarSquare:.3g}")
print(f"Output file: {results.data.htmlFileName}")
```

Output: .html file (binary_generic_netherlands.html)

Estimation report

Number of estimated parameters:	3
Sample size:	228
Excluded observations:	1511
Init log likelihood:	-158.0376
Final log likelihood:	-123.1331
Likelihood ratio test for the init. model:	69.80899
Rho-square for the init. model:	0.221
Rho-square-bar for the init. model:	0.202
Akaike Information Criterion:	252.2661
Bayesian Information Criterion:	262.5542
Final gradient norm:	1.1699E-03
Nbr of threads:	8
Algorithm:	BFGS with trust region for simple bound constraints
Proportion analytical hessian:	0.0%
Relative projected gradient:	4.962596e-06
Number of iterations:	16
Number of function evaluations:	43
Number of gradient evaluations:	14
Number of hessian evaluations:	0
Cause of termination:	Relative gradient = 5e-06 <= 6.1e-06
Optimization time:	0:00:00.020760

Estimated parameters

Name	Value	Std err	t-test	p-value	Rob. Std err	Rob. t-test	Rob. p-value
ASC_CAR	-0.798	0.27	-2.95	0.00314	0.275	-2.9	0.00379
BETA_COST	-0.113	0.0232	-4.85	1.21e-06	0.0241	-4.67	3.03e-06
BETA_TIME	-1.33	0.344	-3.86	0.000115	0.354	-3.75	0.00018

Correlation of coefficients

Coefficient1	Coefficient2	Covariance	Correlation	t-test	p-value	Rob. cov.	Rob. corr.	Rob. t-test	Rob. p-value
BETA_COST	ASC_CAR	0.00434	0.693	2.69	0.00712	0.00473	0.713	2.65	0.00813
BETA_TIME	ASC_CAR	0.0455	0.491	-1.67	0.0949	0.0464	0.476	-1.6	0.109
BETA_TIME	BETA_COST	0.000664	0.0834	-3.54	0.000398	0.000701	0.0822	-3.44	0.000584

Smallest eigenvalue: 6.82736

Largest eigenvalue: 4322.14

Condition number: 633.062