AMLET
Reference manual

Fabian Bastin

July 3, 2011

Abstract

This document explains how to install and use the software AMLET v0.10.6, designed to estimate multinomial and mixed logit models. The current documentation is unfortunately far from be completed, and is full omissions and possibly errors. Any suggestion concerning this document and/or the software should be send to `bastin@iro.umontreal.ca`.

The latest version of this document, as well as the software itself, can be retrieved at the internet address `http://amlet.slashbin.net`.

Contents

Chapter 1

Introduction

1.1 What is AMLET?

AMLET is an acronym for Another Mixed Logit Estimation Tool. As its name suggests, it is a software mainly designed to estimate various kind of mixed logit models. It can be used either as a standalone program, either as a C library. AMLET is an open source project (see Appendix C.1; the last version can be downloaded at the address `http://amlet.slashbin.net`.

1.2 Supported systems

AMLET development is primarily Unix-oriented, while the software can also be used on Windows systems, with some technical limitations[1]. Currently supported are Linux and Mac OS X systems. The software should also runs without major trouble on BSD systems. Windows users have to install the third-party package Cygwin which is a Linux-like environment; it in particular offers a `bash` text-console (see Section 1.3.3).

Unless otherwise stated, we will assume in this document that you work on a Unix-like system; the explications are still valid for the other environment. We also assume that you use `bash` as your shell, since it is the default choice for major Linux distributions, as well as `Cygwin`; some users could have to slighty adapt the explanation if they use csh shells, and its variants. If you do not know what is your current shell, you probably already use `bash`. It can also be determined by reading the file `/etc/password`.

[1] In particular, the timings will be less accurate.

1.3 Requirements

1.3.1 Hardware requirements

There is no formal minimum requirements for AMLET, since they crucially depend on the size of dataset size and the number of random variables that are present in the model. We however recommand a minimum of 1Go RAM, knowing that the available memory constrains the achievable accuracy.

1.3.2 Software requirements

Various software elements are required in order to correctly install AMLET, as well as some optional elements, if you want to install additional tools along with the main package.

C compiler We recommand to use the compiler delivered with your computer; on Linux-machine and cygwin, this is typically the GNU C Compiler (gcc). If you install the BLAS library ATLAS, gcc is required for the quick installation procedure.

Compilers The code is mainly written in C, but contains portions written in Fortran, mainly imported from LAPACK. Consequently, both a C and a Fortran compilers are requires; the code has been successfully tested with `gcc` and `gfortran`.

Unix shell In order to easily perform the installation, autotools (autoconf, automake, libtoolize) are highly recommanded. The detailed installation process can however often be pursued without them. If you are using Mac OS X, you can experience problems with `libtoolize`. Using the development environment proposed in the third-party software collections fink (`http://www.finkproject.org/`)resolves this issue.

Lex, Yacc Driver parsing with AMLET requires standard lexical analyser and parser. AMLET requires flex, and yacc or bison.

Perl Some optional analysis tools provided with AMLET rely on Perl scripting language.

BLAS-CBLAS The code required the presence of CBLAS library. See Section 2.2 for more details.

Oratio, Ophelia These are two companion libraries, that also be downloaded on `http://amlet.slashbin.net`.

1.3.3 Windows users

If Cygwin has not been yet installed, proceed first to its installation, that has to be made online, from the address `http://www.cygwin.com`, and follow the given instructions. The packages to install includes the default ones and the following ones: those in development section (be sure to inclue the packages `autoconf`, `automake`, `make`, `libtoolize`, `flex`, `bison`, `yacc`, as well a `gcc-4` and `gfortran`), —and LaTeX, as well as the `tetex` packages in the publishing section. The installation cannot be completed if the development packages are not installation; the `tetex` packages are required to produce the documentation. Please note that some problems can be encountered under Cygwin if your login name contains spaces.

It is also a good idea to install a correct text editor. Various editors are available under Windows: Vim, Emacs, Xemacs. You can find most of them of the Gnuwin software collection (`http://gnuwin.epfl.ch`). A minimalist one, available under Cygwin, is nano. You can also use notepad.exe, that should be present in most Windows configurations, while it deals with ends of lines in a different way then Cygwin. Don't use a word processor (like Microsoft Word) to edit text files, since it usually format them, making them unreadable by AMLET.

1.4 Acknowledgements

Like for many projects, the development of AMLET has only been possible with the help of many people. First of all, I would like to express my gratitude so Cinzia Cirillo, for the hard work that we have achieved together. This software could not has been written without her advices, based on her huge experience of the practical aspects of the problem. I would also like to thank Philippe L. Toint, who was my Ph.D. advisor during four years, and teached me the most important aspects of nonlinear programming. Finally, my thanks go to Belgian National Fund for Scientific Research, who gave me the grant that made the initial work on this project possible. The development of AMLET has been initiated at the University of Namur (Belgium), continued in the Imperial College London (United Kingdom) and is currently pursued at Cerfacs (Toulouse, France).

The documentation of a software is always a difficult exercise. I would therefore like to thank anyone that helped us to improve the present document. In particular, I would like to express my gratitude to Roberta "Speedy" Pellicanò for her suggestions, during my study stay in the University Frederico II of Naples, Italy (June 2004), and Jasper Knockaert.

The software has also benefit from many suggestions during its development. The following list is certainly far from be complete; my apologies go to each one I have forgotten. I will cite here (by alphabetical order) Michel Bierlaire and Stéphane Hess.

Finally, the painting reproduced the front cover is due to Pieter van Steenwyck, under the name *Ars longa, vitta brevis*.

Chapter 2

Installation

AMLET is currently distributed in source code only, so that the installation process primarily involves the compilation of the various source files. You can compile each element separatly, or the all package in one step. While the later is simpler, the former offers you more flexibility. This allows you in particular to recompile only the elements to have to be replaced by newer ones.

2.1 Environment variables

In order to allow the various libraries to be found during the compilation process and execution, please add the following lines in the file `.bashrc`, present at the root of your home directory:

```
export LIBRARY_PATH=/usr/local/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

This line can be placed at any position position in the `.bashrc` file, but be careful to use a raw text editor. From a text console, your can reach your home directory by simply enter the command

```
cd
```

If you use Cygwin under Windows, you can experience difficulties if a LaTeX distribution has been previously installed. In such case, please also add this line is the file `.bashrc`:

```
export TEXMFCNF=/usr/share/texmf/web2c
```

On Mac OS X, it is preferable to add the package `Fink`, and install the packages `autoconf`, `automake`, `libtool`, and set

```
export PATH=/sw/bin:$PATH
```

The place where the program and the libraries will be install may vary following the Operating System. If you benefit from administrator rights, the installation paths will be typically `/usr/local/bin` for the programs and `/usr/local/lib` for the libraries. If you have not these rights, the typical locations will be `$HOME/bin` and `$HOME/lib`, where `$HOME` is the user repertory. For reference, we will design by `INSTALL_PATH` the path where the programs are installed.

2.2 BLAS library

AMLET make heavy use of CBLAS routines, the C version of BLAS (`http://www.netlib.org/blas`). Many implementations exist, most of them proprietary and restricited to some specific architectures. We only review three open-source implementations: GotoBlas2, GSL Blas, and ATLAS

2.2.1 GotoBlas2

GotoBlas2, available at `http://www.tacc.utexas.edu/tacc-projects/gotoblas2`, is a Blas implementation proposed by the University of Texas at Austin. Praised as one of the best implementations, it has been released under BSD license in 2010. We therefore strongly recommend its use, and Ophelia and Amlet are compiled by default with GotoBlas if detected. A local copy is hosted at `http://amlet.slashbin.net`, that has been tested on Arch Linux, Fedora 7, SuSE 11.3, and Mac OS X 10.6.8. On Arch Linux, the compilation required a slight modification of the script `f_check`, obtained by changing `($flags =~ /^\-l/)` with `($flags =~ /^\-l[a-z,A-Z]/)`.

2.2.2 GSL CBlas

GSL (`"http://www.gnu.org/software/gsl/"`) Blas is supported only for convenience reasons, as it is available as a package on all the major distributions. The GPL is however incompatible with OSL, used for many parts of the software. However, AMLET cannot be considered as a derived work of GSL CBlas as the code can be compiled without any modifications with other CBlas implementations.

2.2.3 ATLAS

ATLAS (`"http://math-atlas.sourceforge.net/"`) is another free implementation of Blas, far better than the GSL one, but which present various issues during the installation process. AMLET will try to detect ATLAS is neither GotoBLAS2 nor GSL CBlas have been found. Note however the you have to install ATLAS as shared libraries in order to be able to run the code.

2.3 AMLET compilation

As for the libraries, you can compile AMLET with the commands

```
./configure
make
make install
```

Optimization compilation options can be specified as arguments of `configure`, for instance:

```
./configure CFLAGS=''-O3 -funroll-loops'' FFLAGS="-O2"
```

Chapter 3

First steps with AMLET

AMLET is a command-line application, and as such, has to be launched from a text-console. The basic usage is

```
amlet [OPTIONS] FILE
```

where `FILE` designs the driver file (see Section 3.1), and `[OPTIONS]` corresponds to optional options, as described in Table 3.1. The options can be used in short or long form; for instance, the command

```
amlet -o log driver
```

has the same effect that

```
amlet --output log driver
```

short	long	argument	function
-d	--debug	-	activate debug mode
-h	--help	-	print help message
-l	--loose	-	permissive mode
-o	--output	file name	save the log information into a file
-u	--unsafe	-	unsafe mode
-v	--verbose	-	verbose mode
-V	--version	-	print version number

Table 3.1: command-line options

We briefly describe these options here, and cover them in more details in the relevant sections.

debug is meant for development purposes, delivering additional messages on the standard output in order to help debugging code;

help option makes AMLET displaying some brief help on the screen, and then exits.

output allows the user to define the name of the log file (see Section 6.2).

unsafe indicates to AMLET to ignore numerical safeguards during computations (see Appendix B). As it could lead to failure of the esimation process, it is highly suggested to not use this option, except if you really know what you are doing.

verbose option informs AMLET to print additional information on the screen during execution.

version prints the version number on the screen, afterwhat the program exits.

3.1 Driver file

The AMLET driver file expresses a specification of the model to calibrate, as well as of the optimization method to apply. The driver has to be written in raw text format; examples of driver files can be found in Appendix A. It basically consists of two kinds of lines: the command ones and the affectations ones. A command line has the following form:

```
command parameters;
```

In other terms if consists of a command followed by the parameters to apply to this command. An affection line has the form

```
variable = value;
```

The value can be an integer, a real, a string or a set, depending on the variable that has to be affected. In some cases, a predefined constant identifier can be used instead of a numerical quantity. String values are any sequence of characters, enclosed within double quotes; a correct string would be for example

```
''this is a string''
```

You cannot use double quotes inside a string. We give in Table 3.2 the name of the main variables, in alphabetical order. There will be discussed in more details in the relevant sections.

Parameter	Default value	Function
b	$(0,\ldots,0)^T$	starting point
btype		type of the utilities coefficients
bgen		logit parameters for synthetic population generation
bgentype		type of the utilities coefficents during observations generation
censor		0 if all individuals face to all alternatives, 1 otherwise
iddep		dependent variable index
idcensor		variable index corresponding to the alternatives availability
draws		indicates if new draws are to be generated or if they have to be read from an external file
drawsname		name of the file containing the draws
eps		tolerance used for optimization
format	Train96	format of the data file
hessian_update		Hessian approximation method
level_accuracy		significance level used for objective accuracy evaluation
method		optimization method
np		number of individuals
nalt		number of alternatives
niter		maximum number of iterations during the optimization process
nobs		total number of observations
nrep		number of draws
nvar		number of variables (coefficients)
problem		problem name
robust		indicates if robust estimation of variance-covariance matrix has to be used
rangen		random numbers generation method
sampling		sampling method
randomization		randomization strategy
seed1		seed of the random generator
validation	false	results validation

Table 3.2: variables

You can also write comments with the characters # or @. The remaining of the line is then ignored by AMLET, except if these characters are part of a string. The driver file is case unsensitive, except for filenames, depending on the operating system on which AMLET is executed.

Commands

We now briefly review the commands of AMLET. First, the command `output` expresses the name of file where the results will be saved:

```
output ``results'';
```

where `results` has to be replaced with the name that you want. If no `output` command is present in the drive file, the name will be set by default to the name of the driver file followed by `.output`.

The next two commands allow to specify an existing observations file or to generate a file that will be filled with synthetic data. Assuming that your observations file has the name `data`, you can load it into AMLET with the command

```
load ``data'';
```

On the other name, synthetic data can be obtained with the command

```
generate ``data'';
```

The structure of the data files is explained in Section 5.1.

3.2 A step-by-step example for Windows/Cygwin users

Since many Windows users are not custom to Unix-like environment, we will briefly review the classical steps that you have to make in order to use AMLET. First of all, launch Cygwin, and go to your data directory, for instance:

```
cd data
```

You can check the files that are present in the directory with the command `ls`. Do not forget to put the data in a supported format (see Section 5.1). Recall that they must be in raw text, and alternatives are described by columns. If your file is row oriented, you can reverse it by using the Perl script `amlet_reverse_data.pl` as follows:

```
amlet_reverse_data.pl file nrows ncols
```

where `file` is your data file, `nrows` is the number of alternatives and `ncols` is the number of variable (the dependent ones and the independent, as well as as the alternative availability vector, if applicable). Do not forget to create a driver file explaining to AMLET what to do! Use your preferred text editor, says xemacs, to write the corresponding file:

```
xemacs.exe driver &
```

Then launch AMLET:

```
amlet -v driver
```

The results are written in the file defined in the driver (see Section 3.1), say `results`. You can view them by opening the output file in your text editor, or with a command like `less`:

```
less results.
```

You can quit the `less` quit the 'q' key.

Chapter 4

Mixed logit models

4.1 Supperted models

AMLET allows the estimation of multinomial (or conditional) logit models as well as mixed logit models. The user has not to specify the wanted model, since AMLET will determine it automatically from the utility description. AMLET currently supports only linear-in-parameters utilities, i.e., for $i = 1, \ldots, I$, $j = 1, \ldots, J$,

$$U_{ij}(x_{ij}) = \sum_{k=1}^{n} \beta_{ik} x_{ijk} = \beta^T x_{ij},$$

where n is the number of dependant variables. Each component of the vector β can be fixed, constant or random, as summarized in Table 4.1.

`FIXED`	fixed coefficient
`CONSTANT`	constant (non-random) coefficient
`LOGNORMAL`	lognormally-distributed coefficient
`NORMAL`	normally-distributed coefficient
`TRIANGULAR`	triangularly-distributed coefficient
`UNIFORM`	uniformly-distributed coefficient

Table 4.1: parameters types

4.2 Supported distributions

The following distributions are currently supported for the coefficient of utilities in a mixed logit model and for observations when generating synthetic data.

Lognormal lognormally distributed coefficient; the coefficient is calculated as

$$e^{\mu+\sigma X}$$

where $X \sim N(0, 1)$, and μ and σ are the parameters to estimate. Therefore μ and σ are respectively the mean and the standard deviation of the logarithm of the coefficient.

Since lognormally distributed coefficients are always positive; you have to enter the negative of the explanatory variable in your dataset if necessary.

Normal normally distributed coefficient, with the mean and standard deviation being estimated.

Triangular Triangularly distributed coefficient, with the mean and "spread" being estimated. A triangular distribution with mean b and spread a has a zero density outside $[b - a, b + a]$, while it increases linearly from $b - a$ to b and it decreases linearly from b to $b + a$.

Uniform uniformly distributed coefficient, with the mean b and "spread" a being estimated. A uniformly distributed coefficient of mean b and spread a has therefore the distribution of $U[b - a, b + a]$.

You can also use the previous distributions to generate synthetic data, as well as the following one:

Bernouilli an observation can be drawn from a Bernouilli of parameter p ($0 \leq p \leq 1.0$), that has to be specified.

The used distributions are specified in the variables `btype` and `bgentype` as set of identifiers.

4.3 Sampling methods

Multi-dimensional integrals are approximated by drawings points from the underlying distributions of the random parameters. Standard Monte-Carlo draws can be used as well as some quasi-Monte Carlo techniques: Halton sequences [3], Modified Latin Hypercube Sampling as proposed in Hess et al. [5], and Sobol sequences, following the recommendations of Joe and Kuo [6]. These various quasi-Monte Carlo sequences can be generated by using the library ORATIO(). The drawing method can be defined by with the variable `sampling`, and is set by default to the standard Monte-Carlo approach. Possible values are summarized in Table 4.2.

Pseudo-random numbers	`monte_carlo`
Halton sequences	`halton`
Modified Latin Hypercube Sampling	`mlhs`
Sobol sequences	`sobol`

Table 4.2: sampling methods

The number of random draws to use per individual is defined with the variable `nrep`, and is set by default to 1000. A higher value is sometimes needed, especially if the accuracy and bias on the objective are important. To use more draws, say 10000 per individual, write

```
nrep = 10000;
```

4.3.1 Halton sequences

A popular alternative to pseudo-random draws in the field of mixed logit modelling is the use of Halton sequences (Halton [3]). While we strongly recommend the use of Sobol sequences instead of Halton ones, it is possible to select this approach by correspondingly setting the variable `sampling`:

```
sampling = halton;
```

One major drawback of Halton sequences is the correlation between sequences associated to successive dimensions when the dimensionality is quite high (order of 10 or superior). Two techniques have been proposed to break such correlations: scrambling (Bhat [1]) and shuffling (Hess et al. [4]). Both of them can be used in AMLET. It is also possible to randomize the sequences by randomly shifting the sequences, for each dimension, as proposed by Bhat [1]. While all these techniques are not strictly speaking randomizations methods, all of them can be set using the variable `randomization`, whose default value is `none`, as summarized in Table 4.3.

4.3.2 Randomized Sobol sequences

The sobol sequences can be randomized by using one of the randomization methods summarized in Table 4.4. The randomization strategy can be defined by assigning one of the corresponding constants to the variable `randomization`, for instance:

```
randomization = owen;
```

By default, no randomization is operated.

`none`	no randomization
`owen`	Owen randomization
`scrambling`	scramble the Halton sequences
`shifting`	randomly shift the Halton sequences
`shifting_scrambling`	randomly shift and scramble the Halton sequences
`shuffling`	randomly shuffle the Halton sequences
`shifting_shuffling`	randomly shift and shuffle the Halton sequences

Table 4.3: Randomization strategies for Halton sequences

`none`	no randomization
`owen`	Owen randomization
`faure_tezuka`	Faure-Tezuka randomization
`owen_faure_tezuka`	Owen randomization and Faure-Tezuka randomization

Table 4.4: Randomization strategies for Sobol sequences

4.4 Pseudo-random numbers generators

Each method requires the use of a uniform random numbers generator, whose quality is important to ensure quality of the results. Even if a quasi-Monte Carlo is used, the randomization part also involves the use of a pseudo-random numbers generator. The choice of a good one is still a tricky question (see for instance the relevant discussion in L'Ecuyer [7] and McCullough [9]). Unlike many econometric packages, as LIMDEP, AMLET allows the user to choose between several random number generators, defined in the random numbers generator library. Three generators are currently available: the standard minimal one, proposed by Park and Miller [10], the MRG32K3a generator, due to L'Ecuyer, and the generator MT19937 designed by Matsumoto and Nishimura [8]. To date no significant differences have been observed when using one technique instead of another one; by default we use the MT19937 generator, which benefit from strong theoretical properties and is well tested.

The choice of the random number generator can be defined with the variable `rangen`. Possible values are summarized in Table 4.5.

Each random generator produces a sequence of points that mimics draws from an uniform distribution on $(0, 1)$.. These sequences are however deterministic due to their numerical nature, so they can be reproduced when some initial conditions are fixed. Typically, they can be expressed by a single strictly positive integer,

Standard minimal	`standard`
L'Ecuyer's generator	`ecuyer`
MT19937	`mt19937`

Table 4.5: random number generators

known as the *seed*. This value can be defined in the driver file by means of the variable `seed1`, for example:

```
seed1 = 123456;
```

Most of the time, the seed can be recovered at any point during the random draws generation process, allowing the replication of a sequence starting at any draw of the initial sequence. This is useful for instance when dealing with a newly generated synthetic population, where the same random generator is used for creating the population and the random draws. In this case, the output file will contain both the initial seed and the seed after the population generation. Note however that the MT19937 cannot be restarted with a single seed, except the initial one.

The default value for `seed1` is 0. In this case, AMLET will determine a seed based on the computer clock, that will be used to initiate the pseudo-random draws sequence.

For more information, we refer the reader to the documentation accompying the random numbers library.

4.5 Results validation

The source file `validation.c`, associated to the header file `validation.h`, contains basic validation routines concerned with validation of the estimation results, on a statistical point of view. The validation is currently quite simple, as it simply repeats `nsim` times the simulation of the objective function, using a different set of random realizations for each simulation. The mean, standard deviation, and a confidence interval at 0.9 for the objective function are given on output. In order to activate the validation function, you need to set the variable `validation` to `true` in the driver file:

```
validation = true;
```

Chapter 5

Data files

5.1 Data files

Every model needs data in order to proceed to its calibration, so the possibilty to enter data in a convenient format is important. AMLET unfortunately currently supports basic formats only, even if development is planned on this issue. The supported formats reflect the original purpose of the package, which was only testing a new optimisation method.

5.1.1 GAUSS-like formats

Preliminary AMLET design mimics Train's GAUSS codes for mixed logit estimation (cross-sectional and panel versions). These routines are available at the address `http://elsa.berkeley.edu/~train/software.html`. We also refer the reader to Revelt and Train [11], and Train [12], for more details. AMLET has currently no capabilities to construct new data from a preexisting dataset, for instance dummy observations for alternatives specific constants, so please be sure that your data are completely specified before using AMLET.

The format of the data file is specified by assign to the variable `format` of the driver file the value `Train96` of `Train99`, the number being a reference to Train's code release year. For instance, the following line

```
format = Train96;
```

indicated that data are stored in `Train96` format, which is also taken as default if no format is specified in the driver file.

```
   0    1    0    0
   0    0    1    0
   0    0    0    1
 1.6 -2.0  3.2  4.1
   0  2.5  1.0  0.8
-1.2 -1.5 -0.4 -1.6
   0    1    0    0
   0    1    0    0
   0    0    1    0
   0    0    0    1
 2.4 -3.0  2.3  3.0
   0  1.7  1.2  1.5
-1.6 -2.0 -1.2 -2.4
   0    0    0    1
```

Figure 5.1: Train96 Format

Train96 format

This format is primarily designed for cross-sectional data, as illustrated in Figure 5.1.1. In this figure, each indiviual is facing four alternatives, corresponding four columns. The six first rows of each record (containing seven rows) corresponds to the independent variables. Since each individual is assumed to deliver one observation, this format cannot be used for panel data. An additional row is devoted to the dependent (decision) variable, with one on the column corresponding to the chosen alternatives, and 0 elsewhere. The line position in each record must be indicated with the variable `iddep` in the driver file. For Figure 5.1.1, we would have

```
iddep = 7;
```

By default, all alternatives are available to each individual. If you want to specify alternative availabilities, you have to set the censor variable to 1 in the driver file:

```
censor = 1;
```

The variable `idcensor` then represents the line describing alternative availabilities. The corresponding line in the driver file is simply a 0-1 line, with one on available alternatives, and 0 elsewhere.

Train99 format

This format is used for panel data. It adds to the base name the suffixes `_x`, `_y` and `_times`.

Chapter 6

Output files

The estimation process would be useless if no exploitable results were delivered to the user. The software therefore delivers various files at the of the optimisation process. Their content if meaning is described below

6.1 Results file

6.1.1 Definition

The result of the estimation process is written in the file defined by the command `output` inside the driver file (see Section 3.1). If no name is precised, the default name is the name of the driver file followed by `.output`.

The main characteristics of the problem are first summarized. It is always a good idea to name the problem under consideration, by setting the variable `problem` in the driver file, for instance

```
Problem = ``test'';
```

6.1.2 Content

t-statistics

The t-statistics can be computed in various ways, depending on the options present in the driver file. Each method involves the computation of the information matrix. However, due to the sampling errors, robust t-statistics are not necessarily better than those obtained with the outer approximation. The sampling errors are indeed amplified when computing second-order derivatives.

6.2 Log file

AMLET generates a log file during its execution. It will contain detailed information on the process, from initialisation tasks to a complete recording of the optimisation iterates. The default name of the log file is `amlet.log`, but this name can be change with the `-o` and `--option` options at the start of the software (see Chapter 3).

6.3 Analysis

Along with the main program come various tools designed to help the analysis of the results. These tools require the presente of perl development libraries in order to correctly compile, and aim to facilitate the study of the behavior of AMLET. The following files are then installed: `amlet_analysis.pl`, `amletlog` and `amletstats`. The utility `amlet_analyis.pl` cannot be used alone, but must be called as an argument of `amletlog` or `amletstats`. The tool `amletlog` allows to take some important information from the log file |amlet.log— produced along with the estimation. It must be invoked as following:

```
amletlog INSTALL_PATH/amlet_analysis.pl amlet.log
```

where `INSTALL_PATH` must be replaced by the path where `amlet_analysis.pl` has been installed, typically `/usr/local/bin` (see Section 2. The program prints then on the standard output a summary of the optimization process, where each line correspond to one iteration, and contains the iteration number, the log-likelihood objective value, the sample size and the confidence radius. The utility `amletstats` aims to analyze the behavior over several replications of similar experiments, for instance to study variations due to the random sampling. The way to invoke it is

```
amletstats INSTALL_PATH/amlet_analysis.pl < list
```

where as before `INSTALL_PATH` is the installation path, and list is a file containing the list of results files to analyze. The list must contain one filename per line. `amletstats` then prints on outputs a summary in comma separated values (CSV) format, where each line contains the iterations number, the number of function evaluations (if available), the execution time, the found optimal value, the estimated accuracy, the estimated bias, the mean log-likelihood (currently unavailable), the standard deviation (currently unavailable), the confidence radius (currently unavailable), the estimated parameters (each parameter is followed by the corresponding t-statistic).

Appendix A

Example of driver file

We illustrate the behaviour of AMLET on a 1987 dataset made of 210 non-business trips between Sydney, Canberra and Melbourne. In this dataset, each traveller chooses a mode from four available alternatives (plane, car, bus and train). This dataset is described in more details in Greene [2] (Example 19.18), and is available at the address
`http://www.ats.ucla/stat/limdep/how_to_run_statistical_analysis_.htm`.
A formatted version for AMLET is present inside the directory `testing` of the AMLET source code, as the file `model_australia`.

A.1 Multinomial logit

A.1.1 Driver file

```
# Problem name;
PROBLEM = ``Australia problem'';

# name of the output file
output "results_mnl";

# number of observations
nobs = 210;
np = 210;

# number of aternatives
nalt = 4;

# number of variables
```

```
nvar = 7;

# number of variable that is the dependant variable (choice)
iddep = 7;

# 1 if all people do not face nalt alternatives, else 0
censor = 0;

# maximum number of iterations in the maximization
niter = 150;

# tolerance for convergence
eps = 1e-6;

# type of parameters
btype = { CONSTANT, CONSTANT, CONSTANT,
          CONSTANT, CONSTANT, CONSTANT };
# starting point
b = { 0, 0, 0, 0, 0, 0 };

# data file
load "model_australia";

method = MCBTR;
hessian_update = BFGS;

iapprox = false;
robust = false;
```

A.2 Mixed logit

A.2.1 Driver file

```
# Problem name;
PROBLEM = "Australia problem";

# name of the output file
output "results_mmnl";

# number of observations
nobs = 210;
np = 210;

# number of aternatives
```

```
nalt = 4;

# number of variables
nvar = 7;

# number of variable that is the dependant variable (choice)
iddep = 7;

# 1 if all people do not face nalt alternatives, else 0
censor = 0;

# maximum number of iterations in the maximization
niter = 150;

# tolerance for convergence
eps = 1e-6;

# type of parameters
btype = { CONSTANT, CONSTANT, CONSTANT,
          CONSTANT, NORMAL, CONSTANT };
# starting point
b = { 0, 0, 0, 0, 0, 0, 0 };

# data file
load "model_australia";

method = MCBTRDA;
hessian_update = BFGS;

iapprox = false;
robust = true;
```

Appendix B

Numerical issues

B.1 Choice probability computation

We start from the logit expression

$$P_{ij} = \frac{e^{U_\ell}}{\sum_{k \in \mathcal{A}_i} e^{U_k}}. \tag{B.1}$$

If some of the utilities are too large, the computation of (B.1) can face some overflow. A simple way to avoid this problem is to substract some quantity α since

$$\frac{e^{U_\ell - \alpha}}{\sum_{k \in \mathcal{A}_i} e^{U_k - \alpha}} = \frac{e^{U_\ell}}{\sum_{k \in \mathcal{A}_i} e^{U_k}}.$$

Moreover, is all utilities are too small, they could be evaluated to 0, leading to the ratio $0/0$ in (B.1). A simple way to circumvent this issue is to normalize with respect to the highest utility, i.e. take

$$\alpha = \max\{U_k,\ k \in \mathcal{A}_i\} - \beta.$$

We want to keep each term e^{U_k}, $k \in \mathcal{A}_i$, in the same range, so β has to be kept low. Since $de^x/dx = e^x$, the precise value of β does not significantly affect small utilities, so a simple, while practical, choice is to set β to 0. As a consequence, $e^{U_k - \alpha} \in (0, 1]$ for all $k \in \mathcal{A}_i$.

B.2 Data file problems

Appendix C

Legal aspects

C.1 Licenses and acknoledgements

All of the code, is placed under the OSL. v2.1, or BSD license, as reproduced below. The specific choice for a specific portion of code is indicated in the header of the corresponding file. The software as a all is covered by the Open Software License.

This product includes software developed by the NetBSD Foundation, Inc. and its contributors.

C.1.1 Open Software License v 2.1

This Open Software License (the ”License”) applies to any original work of authorship (the ”Original Work”) whose owner (the ”Licensor”) has placed the following notice immediately following the copyright notice for the Original Work:

Licensed under the Open Software License version 2.1

1) Grant of Copyright License. Licensor hereby grants You a world-wide, royalty-free, non-exclusive, perpetual, sublicenseable license to do the following:

   a) to reproduce the Original Work in copies;

   b) to prepare derivative works (”Derivative Works”) based upon the Original Work;

   c) to distribute copies of the Original Work and Derivative Works to the public, with the proviso that copies of Original Work or Derivative Works that You distribute shall be licensed under the Open Software License;

   d) to perform the Original Work publicly; and

   e) to display the Original Work publicly.

2) Grant of Patent License. Licensor hereby grants You a world-wide, royalty-free, non-exclusive, perpetual, sublicenseable license, under patent claims owned or controlled by the Licensor that are embodied in the Original Work as furnished by the Licensor, to make, use, sell and offer for sale the Original Work and Derivative Works.

3) Grant of Source Code License. The term ”Source Code” means the preferred form of the Original Work for making modifications to it and all available documentation describing how to modify the Original Work. Licensor hereby agrees to provide a machine-readable copy of the Source Code of the Original Work along with each copy of the Original Work that Licensor distributes. Licensor reserves the right to satisfy this obligation by placing a machine-readable copy of the Source Code in an information repository reasonably calculated to permit inexpensive and convenient access by You for as long as Licensor continues to distribute the Original Work, and by publishing the address of that information repository in a notice immediately following the copyright notice that applies to the Original Work.

4) Exclusions From License Grant. Neither the names of Licensor, nor the names of any contributors to the Original Work, nor any of their trademarks or service marks, may be used to endorse or promote products derived from this Original Work without express prior written permission of the Licensor. Nothing in this License shall be deemed to grant any rights to trademarks, copyrights, patents, trade secrets or any other intellectual property of Licensor except as expressly stated herein. No patent license is granted to make, use, sell or offer to sell embodiments of any patent claims other than the licensed claims defined in Section 2. No right is granted to the trademarks of Licensor even if such marks are included in the Original Work. Nothing in this License shall be interpreted to prohibit Licensor from licensing under different terms from this License any Original Work that Licensor otherwise would have a right to license.

5) External Deployment. The term ”External Deployment” means the use or distribution of the Original Work or Derivative Works in any way such that the Original Work or Derivative Works may be used by anyone other than You, whether the Original Work or Derivative Works are distributed to those persons or made available as an application intended for use over a computer network. As an express condition for the grants of license hereunder, You agree that any External Deployment by You of a Derivative Work shall be deemed a distribution and shall be licensed to all under the terms of this License, as prescribed in section 1(c) herein.

6) Attribution Rights. You must retain, in the Source Code of any Derivative Works that You create, all copyright, patent or trademark notices from the Source Code of the Original Work, as well as any notices of licensing and any descriptive text identified therein as an ”Attribution Notice.” You must cause the Source Code for any Derivative Works that You create to carry a prominent Attribution Notice reasonably calculated to inform recipients that You have modified the Original Work.

7) Warranty of Provenance and Disclaimer of Warranty. Licensor warrants that the copyright in and to the Original Work and the patent rights granted herein by Licensor are owned by the Licensor or are sublicensed to You under the terms of this License with the permission of the contributor(s) of those copyrights and patent rights. Except as

expressly stated in the immediately proceeding sentence, the Original Work is provided under this License on an ”AS IS” BASIS and WITHOUT WARRANTY, either express or implied, including, without limitation, the warranties of NON-INFRINGEMENT, MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY OF THE ORIGINAL WORK IS WITH YOU. This DISCLAIMER OF WARRANTY constitutes an essential part of this License. No license to Original Work is granted hereunder except under this disclaimer.

8) Limitation of Liability. Under no circumstances and under no legal theory, whether in tort (including negligence), contract, or otherwise, shall the Licensor be liable to any person for any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or the use of the Original Work including, without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses. This limitation of liability shall not apply to liability for death or personal injury resulting from Licensor's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

9) Acceptance and Termination. If You distribute copies of the Original Work or a Derivative Work, You must make a reasonable effort under the circumstances to obtain the express assent of recipients to the terms of this License. Nothing else but this License (or another written agreement between Licensor and You) grants You permission to create Derivative Works based upon the Original Work or to exercise any of the rights granted in Section 1 herein, and any attempt to do so except under the terms of this License (or another written agreement between Licensor and You) is expressly prohibited by U.S. copyright law, the equivalent laws of other countries, and by international treaty. Therefore, by exercising any of the rights granted to You in Section 1 herein, You indicate Your acceptance of this License and all of its terms and conditions. This License shall terminate immediately and you may no longer exercise any of the rights granted to You by this License upon Your failure to honor the proviso in Section 1(c) herein.

10) Termination for Patent Action. This License shall terminate automatically and You may no longer exercise any of the rights granted to You by this License as of the date You commence an action, including a cross-claim or counterclaim, against Licensor or any licensee alleging that the Original Work infringes a patent. This termination provision shall not apply for an action alleging patent infringement by combinations of the Original Work with other software or hardware.

11) Jurisdiction, Venue and Governing Law. Any action or suit relating to this License may be brought only in the courts of a jurisdiction wherein the Licensor resides or in which Licensor conducts its primary business, and under the laws of that jurisdiction excluding its conflict-of-law provisions. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any use of the Original Work outside the scope of this License or after its termination shall be subject to the requirements and penalties of the U.S. Copyright Act, 17 U.S.C. Â§101 et

seq., the equivalent laws of other countries, and international treaty. This section shall survive the termination of this License.

12) Attorneys Fees. In any action to enforce the terms of this License or seeking damages relating thereto, the prevailing party shall be entitled to recover its costs and expenses, including, without limitation, reasonable attorneys' fees and costs incurred in connection with such action, including any appeal of such action. This section shall survive the termination of this License.

13) Miscellaneous. This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable.

14) Definition of "You" in This License. "You" throughout this License, whether in upper or lower case, means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, "You" includes any entity that controls, is controlled by, or is under common control with you. For purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

15) Right to Use. You may use the Original Work in all ways not otherwise restricted or conditioned by this License or by law, and Licensor promises not to interfere with or be responsible for such uses by You.

C.1.2 BSD License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT

Bibliography

[1] Chandra R. Bhat. Simulation estimation of mixed discrete choice models using randomized and scrambled Halton sequences. *Transportation Research B*, 37(3):837–855, 2003.

[2] William H. Greene. *Econometric Analyis*. Prentice Hall, Upper Saddle River, USA, fifth edition, 2003.

[3] John H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.

[4] Stéphane Hess, John Polak, and Andrew Daly. On the performance of shuffled Halton sequences in the estimation of discrete choice models. In *Proceedings of European Transport Conference*, Strasbourg, France, 2003. PTRC.

[5] Stéphane Hess, Kenneth Train, and John Polak. On the use of a modified latin hypercube sampling (mlhs) approach in the estimation of a mixed logit model for vehicle choice. *Transportation Research B*, Forthcoming.

[6] Stephen Joe and France Y. Kuo. Remark on algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 29(1):49–57, 2003.

[7] Pierre L'Ecuyer. Uniform random numbers generators: a review. In S. Andradóttir and K. J. Healy and, editors, *Proceedings of the 29th conference on Winter simulation*, pages 127–134, New York, USA, 1997. ACM Press.

[8] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[9] B. D. McCullough. A review of TESTU01. *Journal of Applied Econometrics*, 21:677–682, 2006.

[10] Stephen K. Park and Keith W. Miller. Random number generators: good ones are hard to find. *Communications of the ACM*, 31(10):1192–1201, 1988.

[11] David Revelt and Kenneth Train. Mixed logit with repeated choices of appliance efficiency levels. *Review of Economics and Statistics*, 80(4):647–657, 1998.

[12] Kenneth Train. Halton sequences for mixed logit. Working paper No. E00-278, Department of Economics, University of California, Berkeley, 1999.