University of Warsaw
Department of Economics
czaj.org

DCE DEMO: Script walkthrough
Version 2015-11-02

by Martín López

Contents

List of Figures

List of Tables

DCE DEMO: Script Walkthrough

1. Introduction

The main purpose of this document is to demonstrate how to use the files contained in the *DCE folder* in order to estimate the different Discrete Choice models available in the web site www.czaj.org. The author believes that by reading this document, you will be able to understand faster the process of model estimation.

This guide assumes basic programing skills and basic knowledge of Discrete Choice models. Previous exposition to Matlab language is useful but not necessarily required.

It is advised to read this guide interactively. Thus, make sure that while reading the different parts of this guide, you run the relevant sections in your computer and check that the output matches what is described in this document.

The guide is structured as follows: in the next section an overview of the folder and the *DEMO script* is presented. In the third section, it is shown how to set general options (they apply to all models). After that, in section four, it is shown how to handle and change estimation and optimization options. Finally, in the fifth section, the process of models' estimation is presented.

In order to make a more readable guide, I added important but not fundamental details in footnotes. Likewise, the material that was considered relevant, but required more space than a footnote to be treated clearly, has been gathered in the appendix. Main text body and footnotes contain explicit references to the appendix when that is required.

Finally, I would like to point out that this guide is continuously updated. Thus, make sure that the version of the DCE script matches the one in the cover page. Feedback is welcome to mlopez@wne.uw.edu.pl

I sincerely hope that this document will help you to go through the code!

2. Overview

2.1. Folder overview

The folder *DCE* contains three main subfolders: *DCE*, *DCE_demo* and DCE_*tools*[1] (see figure 1). The first subfolder (*DCE*) contains ten subfolders; each one of those subfolders contains the user-defined functions to estimate the DCE models. The second subfolder, *DCE_demo*, contains the script that we extensively explain here and the dataset used by the script. The last folder, *DCE_tools*, contains some support functions.

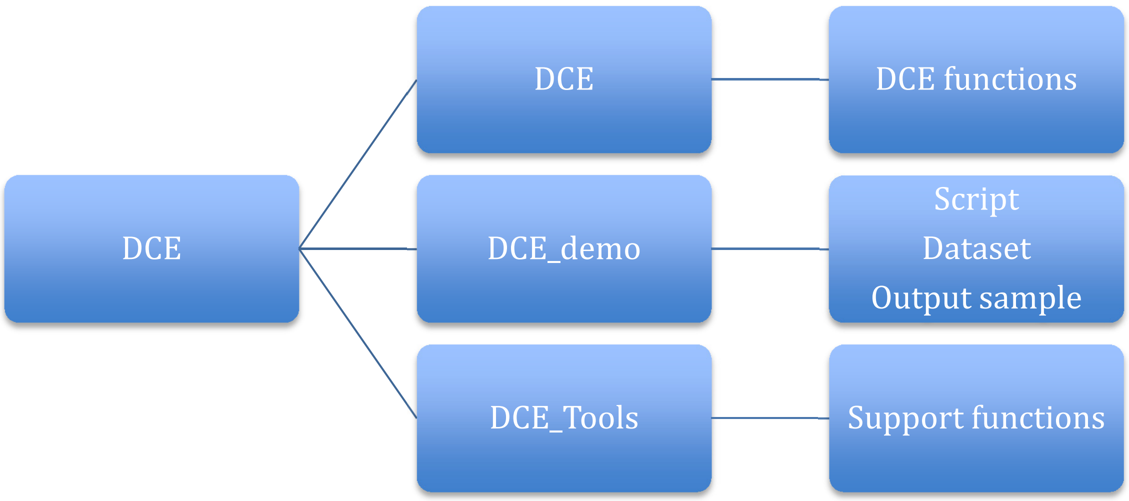

Figure 1: Folder structure

2.2. Script overview

The file *DCE_DEMO.m* is the script designed to estimate the different discrete choice models. By following the script, it is possible to estimate models from scratch (assuming only the existence of a data set properly arranged). The script is compound of eighteen sections (see table 1). For explanation purposes, these sections can be divided into three groups: general options (sections 1-4), estimation and optimization options (section 5) and models' estimation (section 6 -18)

[1] It is advised to add the folder (with subfolders) to the searching path. This will avoid future problems when using the files contained in subfolders. To do so, in the menu home, click on the button set path. A new window will open. In this window, select the location of DCE folder and select the options add with subfolders. Finally save the changes.

In the next parts of this document and based on the preloaded example, we show how to use the script and the options available to estimate the different discrete choice models.

Table 1: Script sections

Group	Title
General options	1. Preamble
	2. Loading and specifying data
	3. Loading and specifying input
	4. Sample characteristics
Estimation and optimization options	5. Estimation and optimization options
Models estimation	6. MNL
	7. MXL
	8. MXL
	9. GMXL
	10. GMXL
	11. LC
	12. LCMXL
	13. LCMXL
	14. MIMIC
	15. HMNL
	16. HLC
	17. HMXL
	18. HMXL

3. General options

General options, as can be easily inferred, apply to all models estimations. These options are set in the first four sections of the script. The purpose of these initial sections is, therefore, to prepare the arguments required to estimate the DCE models. Concretely, three key structures are prepared: *INPUT, EstimOpt*, and *OptimOpt*.

The first structure stores the data in terms of chosen options and attributes. The second one is mainly used to specify sample characteristics and some other specific options, such as simulation of probabilities. The third one specifies the options to solve the optimization problem. Particular details about these structures will become clearer as we go through sections.

For illustrative purposes, the strategy that is followed in the proceeding parts of this document is to run section by section and keep track of the changes that take place in the workspace[2].

3.1. Preamble

In this section, we simply clear the command window and the workspace and create the variable *B_backup*[3]. At this moment, the variable is empty. However, as it will be shown later, this variable will be used during the estimation process, so we will come back to it later.

The code used to perform the tasks previously described is shown next

```
% DCE demo script - 2015-08-18
% (CC BY 4.0) czaj.org

clear all
clc

global B backup; % this is for storing B in case iterations are
interrupted with ctrl-c
```

After running this section your workspace should look like these

Workspace:
B_backup []

3.2. Loading and specifying data

As its name specifies it, the aim of this section is to load the data set into the workspace. For that purpose a structure[4] called *estimOpt* is created and the field *DataFile* is added to it. Here the filename of the data set must be specified (see code below). After that, the data set is loaded into the variable *DATA*.

```
EstimOpt.DataFile = (‘NEWFOREX DCE_demo.mat’);
DATA = load(EstimOpt.DataFile);
```

In our example, the name of the dataset (included in the downloadable folder) is *NEWFOREX_DCE_demo.mat*[5]. The data set contains 290 variables and it is stored as a structure.

After running this section the workspace looks like this:

[2] To run specific sections of the script you can use the button run section under the editor menu.

[3] It can be read as ‘betas backup’ because in the future it will store the values of the models’ parameters. The variable is declared as global, which means that a single copy of this variable is used for all other functions. More details about global variables can be consulted in Matlab documentation.

[4] *Structure* is a data type used in Matlab. It is normally used to store different data types within the same variable. For more details consult the section *Data types* in Matlab documentation.

[5] The data set was derived as part of a study within the project *New Ways to Value and Market Forest Externalities* (NEWFOREX). A more detailed description can be seen in the appendix of this document.

Workspace[6]
B_backup[]
DATA (1*1 structure, 290 fields)
EstimOpt (structure, 1 field)

3.3. Loading and specifying input

In this section, we specify the variables from the data set that are going to be used as dependent and independent. For that purpose, a structure called *INPUT* is created and two fields, *Y* and *Xa* are added to it. As it might be expected, *Y* works as the dependent variable and *Xa*[7] works as independent variable(s). After that, and in case there are missing observations, we need to handle them. Finally, we specify the attributes' names.

```
INPUT.Y = DATA.Y;
INPUT.Xa = DATA.X;
% INPUT.MissingInd = DATA.SKIP;
EstimOpt.NamesA = {'SQ';'GOS'; 'CEN'; 'VIS1'; 'VIS2';'FEE'}; % Specify
names of the attributes
```

In our example, *Y* represents the choices of the survey respondents. Therefore, it is a logical variable, which means that only takes values of 0 and 1. Likewise, *Xa* represent the attributes that described those choices. In this particular case, we worked with six attributes. A more detailed description of these variables and how they can be interpreted can be found in the appendix A.1 of this document.

After running this section the workspace should look like this:

Workspace
B_backup[]
DATA (structure, 290 fields)
EstimOpt (structure, 2 fields)
INPUT (1*1 structure, 2 fields)

3.4. Sample characteristics

In this section, by sample characteristics, we concretely mean the number of choice tasks per person (*NCT*), the number of alternatives (*Nalt*) and the number of respondents (*NP*). In order to handle this information, three fields are added to the structure *EstimOpt*.

```
EstimOpt.NCT = 12; % Number of choice tasks per person
EstimOpt.Nalt = 3; % Number of alternatives
EstimOpt.NP = length(INPUT.Y)/EstimOpt.NCT/EstimOpt.Nalt; % 789; %
Number of respondents
```

[6] The variables in blue represent the changes with respect to the previous section.

[7] If you are working with your own data set, the attributes must be previously gathered in a single variable, just like in the example.

In our example, we had twelve choice tasks (*NCT)* per respondent. Each choice task consisted of three alternatives. The survey was completed for 789 respondents[8]. Hence, these are the numbers that appeared in the code. If you are using the script with your own data set, you have to change these numbers accordingly.

After running this section the workspace should look like this

Workspace
B_backup[]
DATA (structure, 290 fields)
EstimOpt (structure, 5 fields)
INPUT (structure, 2 fields)

3.5. General options Summary

Up to this point we have loaded the data set and specified the variables that will be used as inputs for estimation pouposes. Additionally, we specified the sample characteristics. Before turning into models estimations, however, it is required to specify the optimization method that the software will employ to estimate models' parameters. That is the task that we confront in the next section.

4. Estimation and Optimization options

In this section, the first step is to call the user-defined function *DataClean* (included in the downloadable folder). The actual call to this functions looks like this:

```
[INPUT, Results, EstimOpt, OptimOpt] = DataClean(INPUT, EstimOpt);
```

This function has two main goals: First, it checks that the data is consistent and complete. This means that the **data set must include the same *NCT* and *Nalt* per person** and, additionally, **for every choice task there must be an election.** In the appendix of this document, it is explained with further details how the function evaluates the previously mentioned characteristics.

Second, the function set the optimization options. In the default case, the optimization options are as follows:

- Solver: Fminunc
- Algorithm: quasi-newton
- GradObj: on
- TolFun: 1e-6
- TolX:1e-6
- MaxIter: 1e4
- FunValCheck: on
- Diagnostics: on
- Maxfuneval: 600000

[8] Notice that the length of *INPUT.Y* is 28404, then 28404/12/3 = 789.

In the appendix, you can see an explanation of the meaning of these options and the alternatives and methods to change them.

After running this section the workspace looks like this:

Workspace
B_backup[]
DATA (structure, 290 fields)
EstimOpt (structure, 22 fields)
INPUT (structure, 4 fields)
OptimOpt
Results (structure with 1 field)

5. Models specifications

Once we have run the previous sections and in case there are no errors with the data, we are ready to estimate the DCE models. In the next sections we show how they work based on the preloaded example.

5.1. Multinomial Logit (MNL)

The first model that appears in the script is multinomial logit (MNL). The relevant section looks as follows:

```
% INPUT.Xs = [DATA.Income 1, DATA.Sex, DATA.Edu dummy(:, 2:4),
DATA.Hhadult, DATA.Age_dummy(:,2:4)];  % Covariates of scale
% INPUT.Xs = INPUT.Xs;
% EstimOpt.NamesS = {'Income'; 'Sex'; 'Edu2'; 'Edu3'; 'Edu4'; 'HHadult';
'Age2'; 'Age3';'Age4'};

% EstimOpt.WTP space = 0; % number of cost parameters for WTP-space
estimation (need to come last in Xa)

% Results.MNL.b0 = [0.1613;0.4870;0.6095;0.06885;0.1886;0.01424];

Results.MNL = MNL(INPUT,Results,EstimOpt,OptimOpt);

% xlswrite('MNL.xls', Results.MNL.R_out)
```

This part of the code allows choosing between different options in order to estimate the MNL parameters. These options will be described in further detail in this part of the guide. For now, however, it is convenient to mention that it is possible include covariates of scale. The first three non-active lines of the code control this option. Second, you can also choose to estimate parameters in *preference-space* or *WTP-space,* which is controlled by fourth non-active option of the code. Third, you can chose between providing initial values or not[9] (fifth non-activated line of the code).

[9] There is a third option: to use values from the *B_backup*. This option, however, does not apply when you estimate the model for the first time. The reason is that at this point, the variable is empty. However, if you run the section for a second time, the values will be taken form *B_backup*

In case you do not provide initial values, a linear regression is used instead[10] (see figure 2)[11]. This last option is particularly useful if, for instance, no previous information about optimal values is available. The final line of the code, allows exporting the results to an Excel file[12].

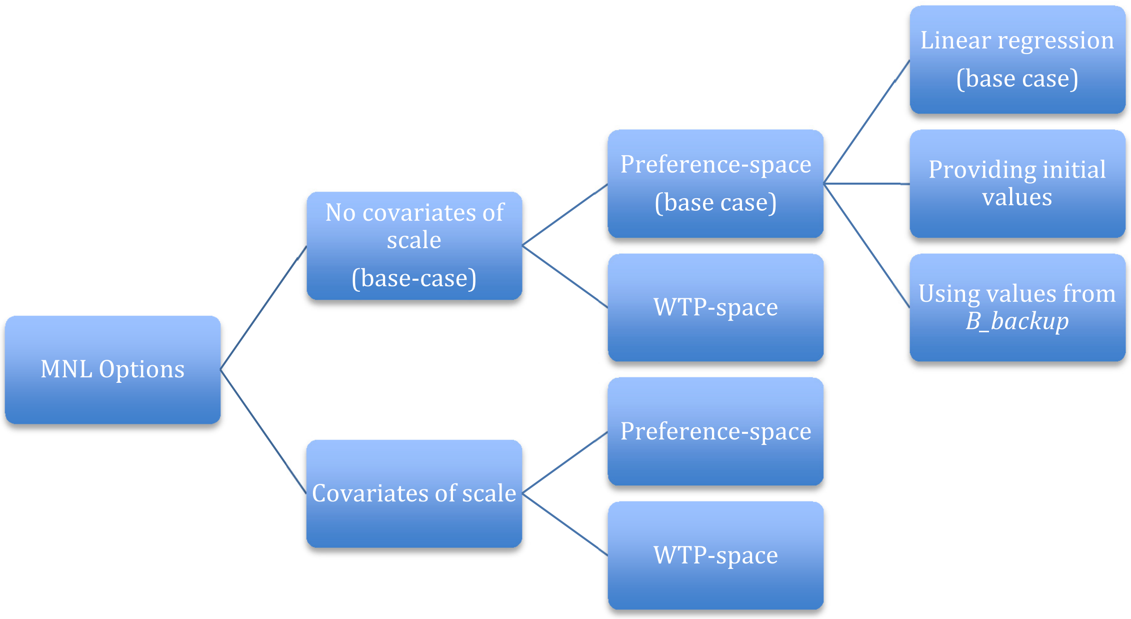
MNL Options
No covariates of scale (base-case)
Covariates of scale
Preference-space (base case)
WTP-space
Preference-space
WTP-space
Linear regression (base case)
Providing initial values
Using values from B_backup

Figure 2. MNL options available

5.1.1 Base case

In the base case, the model is estimated in *preference-space,* initial values are taken from a linear regression and no covariates of scale are estimated (see code below).

```
Results.MNL = MNL(INPUT,Results,EstimOpt,OptimOpt);
```

As it can be seen, the model is estimated by calling to the user-defined function *MNL* (contained in the downloadable folder)*.* The function uses the structures created in the previous sections of the script in order to estimate model's parameters. The output of the functions is added to the structure *Results*. A closer look to the screen output[13] is probably the best way to understand what the function does.

[10] The linear regression is estimated as follows: b0 = regress(INPUT.Y, INPUT.Xa). This produces a parameter for each attribute, which is used as an initial value.

[11] Even thought it is not explicitly included in the figure, the last column applies to all cases. This means that no matter what configuration you use, it is always necessary to provide initial values. In turn, initial values can be specified manually or taken from linear regression. Once a model has been estimated, initial values are taken from *B_backup*.

[12] The xlswrite function works only on Windows based computers with Microsoft Excel installed on them. Depending on you operating system, the output of this function may differ or not work at all.

[13] The output to which we refer here is the one that appears on the command window. The output stored in *Results.mnl* includes more information.

Assuming that no options were modified, the first part of the output should look like this:

```
Estimating MNL model ...
in preference-space.
Using linear regression estimates as starting values
Initial values: [0.41 0.21 0.22 0.12 0.11 0.00041]
Opmization algorithm: quasi-newton
Gradient: user-supplied, analytical
Hessian: off, retained from optimization
```

Here you see the options used to estimate the parameters. Concretely, we are estimating the parameters in *preference-space* and we are using the values estimated in the linear regression as initial values.

Next you will see diagnostic information:

```
____________________________________________________________
  Diagnostic Information

Number of variables: 6

Functions
Objective and gradient:          optimfcnchk/checkfun
Hessian:                         finite-differencing (or Quasi-Newton)


Algorithm selected
  quasi-newton
____________________________________________________________
  End diagnostic information
```

In this part of the output, we see information about the optimization problem. We are estimating six parameters, one per each attribute. After that, we see the objective function, gradient and hessian options. And finally we see the algorithm applied to the optimization problem.

The third part of the output shows the optimization process:

```
Iter.  Eval.    dB        Step       f(x)       df(x)     Opt. Cond.
  0     1      0.0000    0.0000    9718.59    0.0000    0.0027

Local minimum possible.

fminunc stopped because it cannot decrease the objective function along the current search
direction.

<stopping criteria details>
```

For simplicity, not all iterations were included. However, it is possible to notice that for every iteration, information about the optimization process is displayed. After that, we see the exit message: *local minimum possible*. This message is a brief explanation of stopping conditions[14].

Finally, we see the parameters of the attributes, their statistics (standard error and p-value) and convergence value of the function.

```
var.  coef.       st.err.   p-value
SQ    0.1600***   0.0537   0.0029
GOS  0.4866***   0.0293   0.0000
CEN  0.6091***   0.0308   0.0000
VIS1 0.0682*      0.0401   0.0886
VIS2 0.1883***   0.0377   0.0000
FEE  0.0142***   0.0006   0.0000

LL at convergence: -9718.5907
```

The MNL model, therefore, is defined as follows:

p_j = 0.16(SQ)+0.48(GOS)+0.60(CEN)+0.04(VIS1)+0.03(VIS2)-0.01(FEE)

where p_j stands for the probability of choosing option j

After running this part of the script the workspace looks like this:

B_backup[0.16, 0.48, 0.60, 0.06, 0.18, 0.01]
DATA (structure, 290 fields)
EstimOpt (structure, 22 fields)
INPUT (structure, 4 fields)
OptimOpt
Results (structure with 2 fields)

Compared to the previous section there are two changes. First, the variable *B_backup* is used to store the value of the parameters. And second, the results of model estimation are stored in *Results*.

5.1.2. MNL options

In this section we show how to change the different options to estimate the MNL model. Particularly, we show how to estimate covariates of scale, how to change to *WPT-space* and how to chose between providing initial values or using linear regression. It is important to mention that these options can be combined. For instance, it is possible to estimate the model with covariates of scale, in WTP-space, taking initial values from a linear regression.

Covariates of scale

[14] More information about stopping can be obtained by clicking the relevant link is the command window and in Matlab documentation: *Exit flags and exit messages*.

In order to estimate the parameters for covariates of scale[15], it is necessary to activate the first three options of the MNL section, as it is shown below:

```
INPUT.Xs = [DATA.Income 1, DATA.Sex, DATA.Edu dummy (:, 2:4),
DATA.Hhadult, DATA.Age_dummy(:,2:4)];  % Covariates of scale
INPUT.Xs = INPUT.Xs;
EstimOpt.NamesS = {'Income'; 'Sex'; 'Edu2'; 'Edu3'; 'Edu4'; 'HHadult';
'Age2'; 'Age3';'Age4'};
```

The main effect of this change is that the variable *Xs* is created. In this particular case, *Xs* is a matrix of dimensions 28404 * 9[16]. In addition, the names of the variables are specified in *NamesS*. When covariates of scale are included in the model estimation, it is required to change the gradient option to numerical (*EstimOpt.NumGrad = 1*). The reason is that covariates of scale do not support analytical gradient options. In addition, it is also advised to use *trust-region* algorithm.

When this option in included, in addition to the normal output, we also get the estimations for the covariates of scale

```
Covariates of scale
var.      coef.       st.err.   p-value
Income   -0.0975***   0.0280   0.0005
Sex       0.1389**    0.0645   0.0314
Edu2      0.0952      0.2547   0.7085
Edu3      0.0206      0.2414   0.9319
Edu4     -0.0728      0.2422   0.7636
HHadult  -0.0702**    0.0305   0.0214
Age2      0.1371      0.0903   0.1290
Age3      0.0046      0.0940   0.9608
Age4      0.1745**    0.0814   0.0321

LL at convergence: -9702.9368
```

WTP space

To change from *preference-space* to *WTP-space*[17], it is only required to activate the next option before calling the *MNL* function (e.g. before running the section)

---

[15] The covariates of scale are a way to control dynamics (e.g. ordering effects and changes in respondents' scale) in Discrete Choice Models. For more details the reader is referred to Czajkowski et al 2014b y Czajkowski 2014c.

[16] A More detailed explanation about variables meaning can be found in the appendix

[17] In *preference-space* the utility function takes the form:

$$U_j = \beta' x_j - \alpha p_j$$

Where β are the coefficients of the non-price attributes and α is the parameter of the price attribute. The change to *WTP-space*, implies that the non-price attributes will be scaled by the price attribute. In our example, this means that every non-price attribute is divided by the *FEE* attribute coefficient (0.0142). For more details abut preference-space and WTP-space, the reader is referred to Train and Weeks (2005).

```
EstimOpt.WTP_space = 1;
```

The change will be visible in the screen output of the function

```
Estimating MNL model ...
in WTP-space.
Using the starting values from Backup
Initial values: [0.16 0.49 0.61 0.068 0.19 0.014]
Opmization algorithm: quasi-newton
Gradient: user-supplied, analytical
Hessian: off, ex-post calculated using BHHH
```

User-defined initial values

In order to use user-defined initial values instead of linear regression, it is required to activate the line *Results.MNL.b0*, as it is shown below.

```
Results.MNL.b0 = [0.1613;0.4870;0.6095;0.0688;0.1886;0.0142];
```

In this case, the optimized values are provided, but any number can be use instead. This option is useful, for instance, when previous information about approximate value of parameters is available. The change will be visible in the screen output of the function

```
Estimating MNL model ...
in preference-space.
Initial values: [0.16 0.49 0.61 0.069 0.19 0.014]
WARNING: Setting user-supplied Hessian off - quasi-newton algorithm does not use it anyway
Opmization algorithm: quasi-newton
Gradient: user-supplied, analytical
Hessian: off, retained from optimization
```

It is important to mention that the option of providing initial values is relevant only when the model is estimated for the first time. After that, the script will take the values stored in *b_backup* as initial values.

Appendix

A.1. Data set and variables description

The data set was derived as part of a study within the project *New Ways to Value and Market Forest Externalities* (NEWFOREX). The data set is the result of a survey applied to 789 respondents in two phases. The first one conducted on December 2011 and the second one six months later.

The first survey contained twelve choice tasks repeated twice. The second one contained 12 choice tasks. Each choice task consisted of three alternatives. The purpose of the analysis was to test the stability of preferences within and between samples. More details about the study, survey and dataset can be consulted in Czajkowski et al. (2014a).

In our example, we used the variables *Y* and *X* of the data set as a dependent and independent variable respectively. As it was mentioned earlier, *Y* recorded the choices of the respondents. Therefore, it is defined as a logical variable (takes only 1 for true and 0 for false). The variable contains 28404 observations, which corresponds to the survey design. This is the number of respondents (789) times the number of choice tasks (12) times the number of alternatives (3).

Likewise, *Xa* recorded the attributes of the alternatives. Each alternative was specified in terms of 6 attributes: *SQ*, *GOS*, *CEN*, *VIS1*, *VIS2*, *FEE* (see section 3.3). *SQ* stands for status quo, which means no changes with respect to the current situation. *GOS* stands for commercial forest. *CEN* stands for natural regenerated forest. *VIS1* and *VIS2* represent the number of visits per day. For *VIS 1* the visits per day are restricted to 5000 and for *VIS 2* the visits per day are restricted to 7500. Finally, *FEE* represents the annual cost per household of each alternative. The next table illustrates the attributes of each alternative.

Table 2: Alternatives and attributes

Alternative/attributes	**SQ**	**GOS**	**CEN**	**VIS1**	**VIS2**	**FEE**
Status quo	1	0	0	0	0	0
Alternative 1	0	1	1	0	1	-25
Alternative 2	0	0	0	1	0	-50
Alternative 3	0	0	0	0	1	-75
Alternative 4	0	1	0	0	1	-100

As an example of how these data can be interpreted, it is useful to see the first three observations of each variable (see table 3). In this example, the choice task consisted in choosing among the status quo (no change), alternative 1 and alternative 2. The respondent chose the alternative 1.

Table 3: Example of a choice task

Response (y)	**SQ**	**GOS**	**CEN**	**VIS1**	**VIS2**	**FEE**
0	1	0	0	0	0	0
1	0	1	1	0	1	-25
0	0	0	0	1	0	-50

Variables used to estimate Covariates of scale

Income

This variable refers to the personal income of the respondent. It is a categorical variable with eleven levels described in the next table

Table 4: Income levels of the survey

Level	Income range (PLN)
1	0 -1000
2	1001-2000
3	2001-3000
4	3001-4000
5	4001-5000
6	5001-6000
7	6001-7000
8	7001-8000
9	8001-9000
10	9001-10000
11	> 11 000

In order to check the frequency of each response, you can use the function tabulate, as it is indicated in the example below:

```
tabulate (DATA.Income_1)
```

The screen output looks like this:

```
Value   Count   Percent
    1    5076    17.87%
    2    9684    34.09%
    3    8352    29.40%
    4    2808     9.89%
    5    1332     4.69%
    6     756     2.66%
    7     108     0.38%
    8     108     0.38%
    9       0     0.00%
```

10	36	0.13%
11	144	0.51%

Sex
Determines the gender of the respondent, the number zero stands for male and one for females.

Edu
This variable stands for education. It is a categorical variable with four levels, which are specified in the next table.

Table 5: Levels of the variable education

Level	Meaning
1	Primary
2	Vocational
3	Secondary
4	Higher

In the model, this variable is transformed to a dummy version and the first level is excluded from the estimation. An example can make this clearer: if we open the variable *Edu*, we see that the first entry is 4. In its dummy version this is captured as follows (you can now open *dummy_edu* to make sure it is like that)

Table 6: Example of how the variable *Edu* is transformed into a dummy

Edu 1	Edu 2	Edu 3	Edu 4
0	0	0	1

The same procedure is repeated for all entries and the result is *Edu_dummy*. In the model estimation, we make no use of the first column of this variable.

Hhadult
This variable refers to the number of adults in the household. In the survey, the answers were between one and seven.

Age
This variable represents the age of the respondent. It is a categorical variable with six levels (see table 7). In the model with covariates of scale, we use a dummy version of this variable[18]. The first and last columns of the dummy version of the variable are excluded from the estimation.

[18] For an example, see the explanation of the variable *Edu*. The dummy version of this variable is constructed in analogous way.

Table 7. Levels of the variable *Age*

Variable level	Age range
1	< 19
2	20-29
3	30-39
4	40-49
5	50-59
6	> 60

A.2. DataClean Function

In section four (*estimation and optimization options*), we said that the *DataClean* function checks if the data loaded is consistent and complete. Here we show how this option is handle by the function and how it applies to our example.

To begin with, the function checks that the data set is consistent (same *NAlt* and *NCT* per person). For that purpose, the field *ROWS* is added to *EstimOpt*. The data is consistent if the number of attributes observations divided by *NAlt* is equal *NP * NCT*. In case the dimensions are not consistent an error message is shown.

```
EstimOpt.Rows = size(INPUT.Xa, 1)/EstimOpt.NAlt ;
if EstimOpt.Rows ~= EstimOpt.NP * EstimOpt.NCT
    error ('Dataset needs to include the same number of choice tasks and
alternatives per person. Some can later be skipped with
EstimOpt.DataComplete and EstimOpt.MissingInd')
end
```

In the *DEMO*, we can se that the field *ROWS* is defined as the division of the number of *Xa* observations by the number of alternatives (28404/3 = 9468). This number is compared to the product of number of respondents times the choice tasks per person (789*12 = 9468).

The next step is to check if the data set is complete (for every *CT* one alternative was chosen). In order to achieve it, the field *TIMES* is added to the structure. If there are no missing indexes (*sum(INPUT.MissingInd) = 0*), then the field *TIMES* is defined as vector of dimension equal to *NP*. All the entries in the vector are equal to *CT*. The data set is complete if the sum of the field *TIMES* is equal to the sum of *Y*.

```
if sum(INPUT.MissingInd) == 0
    INPUT.TIMES = EstimOpt.CT * ones(EstimOpt.NP,1);
    if sum(INPUT.TIMES) ~= sum(INPUT.Y)
        error ('Dataset not complete (missing Y?) - provide index for
rows to skip (EstimOpt.MissingInd).')
    end
```

In the script, *TIMES* is defined as vector of dimension 789 and it is filled with 12, hence, its sum is 9468 (789*12). The sum of *Y* implies that from the 28404 observations, 9468 must have been chosen (remember that each choice task contained three options). Therefore, we are able to continue without errors.

A.3. Optimization options

The default optimization options contained in the script were shown in section four (*estimation and optimization options*). In this part of the appendix, we explain the meaning of those options and we also show the different options available to choose from.

Choosing solver

Solver is the method that is going to be used to estimate the optimal solution. In the script there are two options for solver: *fminunc* and *fmincon*. *Fminunc* is for unconstrained optimization and it must be chosen where there are no constrains on parameters. On the other hand, *fmincon* is for constrained optimization and it must be selected when there are some constraints on parameters.

The default option is *fminunc*. However, in case you required to use *fmincon*, you must add manually the field *EstimOpt.ConstVarActive = 1* before calling the function *DataClean*. As the examples that are shown in this guide are non-constrained ones, we will further explain the options available for this particular solver.

Algorithm

The solver *fminunc* works with two algorithms: *quasi-newton* and *trust-region*. The main difference between these two options is that for *trust-region*, a gradient need to be provided by the user. Conversely, *quasi-newton* algorithm does not require the gradient. Further details about these two algorithms can be found in *fminunc* documentation files.

The default option in the script is *quasi-newton*. In order to change to *trust-region*, you just need to activate the relevant option in the script and change the option accordingly. See the example below.

First erase the *%* sign

```
% OptimOpt.Algorithm = 'quasi-newton'; % 'trust-region'
```

This will activate the option

```
OptimOpt.Algorithm = 'quasi-newton'; % 'trust-region'
```

Finally, change to trust region as indicated

```
OptimOpt.Algorithm = 'trust-region';
```

GradObj

This option allows the user to specify the gradient of the objective function. Therefore, the decision whether activate it or not depends on which algorithm you chose previously: for *trust-region* algorithm, it must be set to 'on'. For *quasi-newton* algorithm this options is not required (*'off'*).

In the script the default option is '*on*'. In order to change the option, you must follow the same steps that indicated in the previous section in the relevant option

```
% OptimOpt.GradObj = 'on'; % 'off'
```

When the algorithm and the *GradObj* option are not compatible (e.g. *quasi-newton* algorithm and *GradObj = 'on'*), a warning message is displayed. Nonetheless, the model is estimated.

TolFun

TolFun is a positive integer that determines the termination tolerance of the function value. The default value in the script is 1e-06. For more details on *TolFun*, it is advised to consult *Tolerance and stopping criteria* in Matlab documentation.

In the script, at least that otherwise is indicated, the value of this options is determined by the field *EPS* of the structure *EstimOpt*. Therefore, in order to change it, it is necessary to activate this option and change the number to its desired value.

```
% EstimOpt.eps = 1.e-9; % overall precision level
```

TolX

TolX is the lower bound on the size of a step, the default value in the script is 1e-06. For more details, please refer to *tolerance and stopping criteria* in Matlab documentation.

The value of this option is also determined by the field *EPS*, therefore, if you modified it, the value of TolX is automatically modified.

MaxIter

This option allows setting the maximum number of iterations allowed. In the script the default value is 10 thousand. In order to change this option you can activate the relevant option and change the number.

```
% OptimOpt.MaxIter = 1e3; % maximum number of iterations
```

FunValCheck

This option displays an error in case the objective function reaches a not valid value (e.g. *inf* or *NaN*). In the script the default value is 'on'.

Diagnostics

The option *Diagnostics* displays diagnostics information (e.g. number of variables, objective function and methods to calculate gradient and hessian) about the function to be minimized or solved. In the script the default value is 'on'.

MaxfunEval

MaxFunEvals allows setting the maximum number of function evaluations. The default number in the script is 60 thousand.

FinDiffType
This option allows choosing the method to estimate the gradient. The default option is *forward*. The alternative option, *central*, is more accurate but it requires more time to perform. In the script you can control this options activating this option

```
% OptimOpt.FinDiffType = 'central'; % 'forward'
```

Hessian
This option is relevant only for *trust-region* algorithm. When the mentioned algorithm is selected, this option is automatically activated. In addition, it is possible to choose between approximate Hessian (default) or user-supplied Hessian. In the default case, the software estimates the Hessian using finite differences. In order to change to user supplied the next option must be activated

```
OptimOpt.Hessian = 'user-supplied'; % 'off'
```

The Hessian is taken form the function call.

Finally, it is important to mention that if Hessian and algorithm options are not compatible (e.g. *quasi-newton* algorithm with Hessian on), a warning message is displayed. Nonetheless, the model estimation is performed.

References

Czajkowski, M., A. Bartczak, V. Budziński, M. Giergiczny and N. Hanley. 2014a. Within- and between- sample tests of preference stability and willingness to pay for forest management. Working papers no 24. University of Warsaw, Faculty of Economic Sciences.

Czajkowski, M., Giergiczny, M. and Greene, W. H. 2014b. Learning and fatigue effects revisited. Investigating the effects of accounting for unobservable preference and scale heterogeneity. Land Economics, 90(2): 323-350.

Czajkowski, M., Hanley, N. and La Riviere, J. 2014c. The Effects of Experience on Preferences: Theory and Empirics for Environmental Public Goods. American Journal of Agricultural Economics, 97(1): 333-351.

Train, K. 2009. Discrete Choice Methods with Simulation, 2nd edition. Cambridge University Press.

Train, K. E., and Weeks, M., 2005. Discrete Choice Models in Preference Space and Willingness-to-pay Space. In: Applications of Simulation Methods in Environmental and Resource Economics, R. Scarpa and A. Alberini, eds., Springer, Dordrecht, 1-16.