DCE package
User manual & Walkthrough

https://github.com/czaj

Updated: 2022-05-12

Table of Contents

Introduction 1

Overview and comparison with other software 2

Preparing the environment 3

Loading the data 4

Model Specifications 6

Estimation and Optimization options 9

- EstimOpt 9
- OptimOpt 10

Model examples and other model-specific estimation options 12

- MNL – Multinomial Logit Model 12
- MXL – Mixed (random parameters) Logit Model 12
- GMXL – Generalized Mixed Logit Model 13
- LC – Latent Class Model 14
- LCMX – Latent Class Mixed Logit Model 14
- MIMIC – Multiple Indicators Multiple Causes Model 15
- HMNL – Hybrid Multinomial Logit Model 16
- HLC – Hybrid Latent Class Model 16
- HMXL – Hybrid Mixed Logit Model 16
- GWMNL – Geographically Weighted Multinomial Logit Model 17
- LML – Mixed Logit Model with a Flexible Mixing Distribution 18
- MDCEV – Multiple Discrete-Continous Extreme Value Model 19
- MMDCEV – Mixed Multiple Discrete-Continous Extreme Value Model 19
- MNL, MXL_d & MXL – Starting values, interactions and Bactive example 20

Introduction

This is a user manual and walkthrough for the https://github.com/czaj package with Matlab scripts and functions that allow for the estimation of models for Discrete Choice Experiments. This manual assumes the basic knowledge of Discrete Choice models. Basic programming skills and Matlab knowledge are recommended, but not necessary. If you want to power-use (e.g., modify, introduce your add-ons) these codes, be prepared to spend a significant amount of time understanding them. This guideline is intended to help you familiarize yourself with this package and outline some of the additional features.

We hope our codes will be useful to other researchers – feel free to study, apply, extend, and build upon what we have done. One of our main goals was to create fast and efficient codes. Our package is significantly faster than most (or even all) accessible Discrete Choice Models packages using different software (e.g. R, STATA, nlogit). The codes come with no warranty – we try to make them error-free and as researcher friendly as possible, however, certain errors may remain. If you find some errors or have suggestions, please contact us – mc@uw.edu.pl.

The package includes codes for:

- Multinomial (conditional) Logit (MNL)
- Mixed (random parameters) Logit (MXL)
- Generalized Mixed Logit Model (GMXL)
- Latent Class (LC)
- Latent Class Mixed Logit (LCMXL)
- Multiple Indicators Multiple Causes (MIMIC)
- Hybrid Multinomial Logit (HMNL)
- Hybrid Mixed Logit (HMXL)
- Hybrid Latent Class (HLC)
- Geographically Weighted Multinomial Logit Model (GWMNL)
- Mixed Logit Model with a Flexible Mixing Distribution (LML)
- Multiple Discrete-Continuous Extreme Value model (MDCEV)
- Mixed Multiple Discrete-Continuous Extreme Value model (MMDCEV)

The models are estimated using the maximum likelihood method and work with the following specifications:

- preference or WTP space
- multiple distribution types (for random parameters)
- non-linear transformations of explanatory variables
- covariates of means, scale, and scale variance (where applicable)
- impose equality restrictions or constraints
- flexible data types (panel structure, non-constant number of choice tasks or alternatives per respondent, and missing data)
- various estimation and numerical optimization algorithms and options
- parallel computing and more.

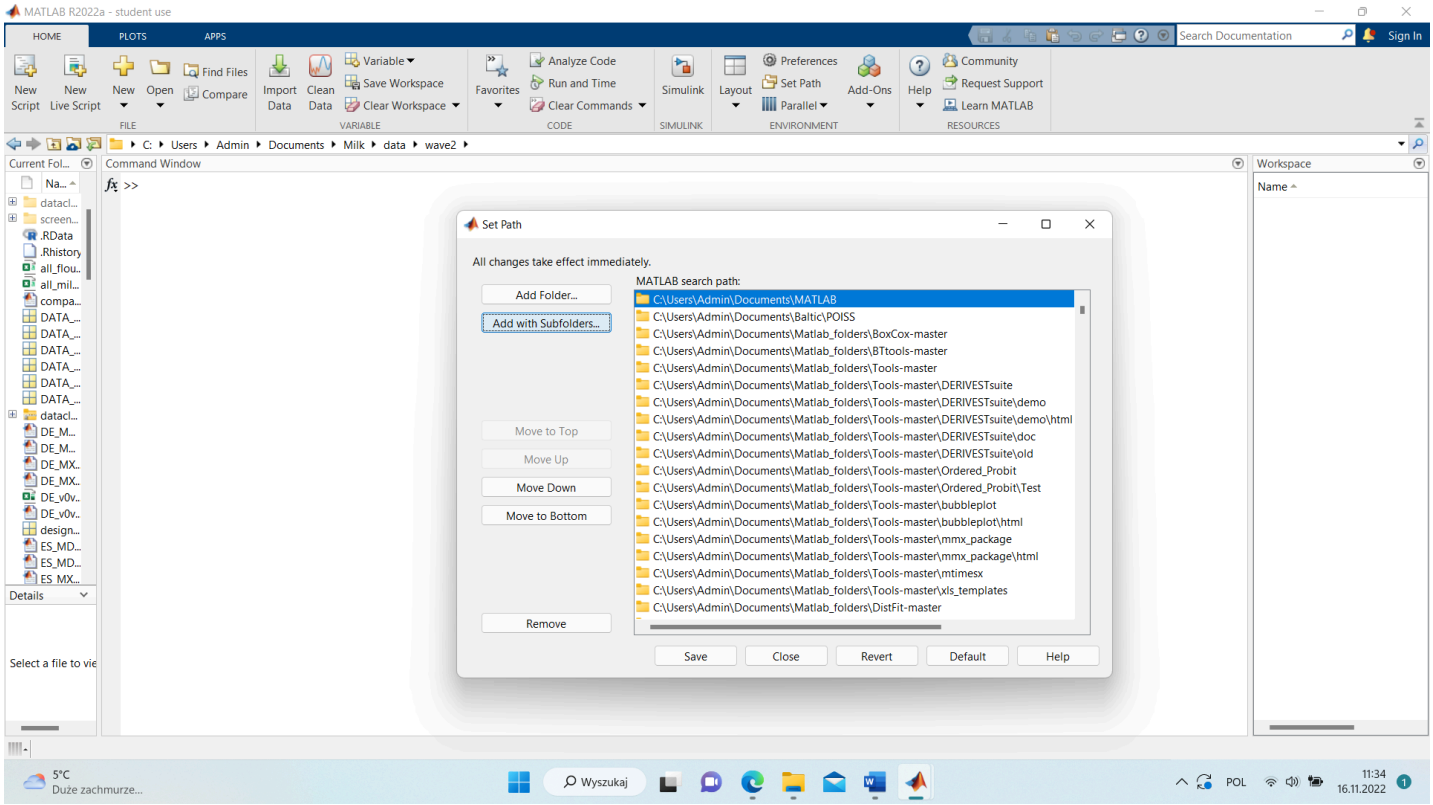
This package was prepared under the leadership of prof. Mikołaj Czajkowski from the University of Warsaw, Poland with the significant contribution of Wiktor Budziński. We also gratefully acknowledge the help of (in alphabetical order and in addition to registered GitHub contributors): Mateusz Buczyński, Danny Campbell, Richard Carson, Marek Giergiczny, William Greene, Arnie Risa Hole, Stanisław Łaniewski, Klaus Moeltner, Nada Wasi, Błażej Popławski, Kenneth Train, Maciej Wilamowski, Wojciech Zawadzki and Ewa Zawojska, whose examples, comments or suggestions we followed when working on these codes.

Overview and comparison with other software

…

Preparing the environment

If it is your first time using Matlab or github.com/czaj package we advise you to download the newest codes from https://github.com/czaj (especially DCE and Tool packages) and gather them in one folder (with subfolders). Then you can specify the path to the codes by clicking Home→Environment→Seth path→Add with subfolders. In that case every time you would use a function from this package, it will automatically be applied. Moreover, it is necessary in order for MS Excel VBA Macros to work correctly and generate output each time you estimate a model. If you would be interested in updating the package, you would only have to download the newest batch of codes and paste them into the currently defined folder.



Before we will describe various estimation options we will suggest you some useful steps that usually are included before calling the model's functions. For instance, at the beginning of the creation of code that will start the estimation of your model, it is recommended to clear the workspace and the command window.

```
clear all
```

```
clc
```

However, be aware that this command will erase all of the unsaved objects from Matlab's current workspace. If you do not want to clear all of the workspaces you can declare some elements that you wish to keep. This command might be also useful for simulations (e.g. if you want to repeat the calculations but do not want to use the same draws, so the results will be different).

```
clearvars -except X
```

We also suggest declaring the B_backup variable as global. It is an auxiliary variable that stores a vector of parameters from the last model iteration. In case when the estimation would be interrupted (e.g. by clicking ctrl+c) the last obtained vector with coefficients (/parameters) will be saved. You would be able to load it manually and continue the estimation from the last point (it can really save you some time).

```
global B_backup;
```

Loading the data

Next step concerns loading the data, which you can do by running the simple code below and declaring the name of the data file. In this walkthrough, we will use (as an example) the data from *Czajkowski, M., Barczak, A., Budzinski, W., Giergiczny, M., and Hanley, N., 2016. Preference and WTP stability for public forest management. Forest Policy and Economics, 71:11-22*. The data concerns the discrete choice experiment about The Białowieża Forest in Poland

```
EstimOpt.DataFile = ('NEWFOREX_DCE_demo1.mat');

DATA = load(EstimOpt.DataFile);
```

However, keep in mind that if you are not working in the folder with your code and data, you should specify the full path to the data. We also suggest transforming your data to .mat file with column vectors.

Also, it is necessary to transform the data into a long format. You can see the auxiliary diagram below, where each respondent faced 3 choice sets with 3 alternatives each.

No.	ID	CS	Alt	Y	X1	X2
1	1	CS1	Alt1	…	…	…
2	1	CS1	Alt2	…	…	…
3	1	CS1	Alt3	…	…	…
4	1	CS2	Alt1	…	…	…
5	1	CS2	Alt2	…	…	…
6	1	CS2	Alt3	…	…	…
7	1	CS3	Alt1	…	…	…
8	1	CS3	Alt2	…	…	…
9	1	CS3	Alt3	…	…	…
10	2	CS1	Alt1	…	…	…
11	2	CS1	Alt2	…	…	…
12	2	CS1	Alt3	…	…	…
13	2	CS2	Alt1	…	…	…
14	2	CS2	Alt2	…	…	…
15	2	CS2	Alt3	…	…	…
16	2	CS3	Alt1	…	…	…
17	2	CS3	Alt2	…	…	…
18	2	CS3	Alt3	…	…	…
19	3	CS1	Alt1	…	…	…
…	…	…	…	…	…	…

In order to prepare the correct form of data we recommend using built-in Matlab functions: reshape() and repmat(). We also suggest using our function ReplicateRows(), which allows you to replicate the rows a specified number of times (e.g. if you want to repeat respondent-specific variable number_of_cs*number_of_alternative times or choice set-specific variable number_of_cs times).

If you would like to have a look at the data, you can create the table. You can do that by binding several column vectors and transforming such created array into table (and you can provide some column names, so it would be much easier to check your data).

```
tableDCE = [DATA.ID, DATA.ALT, DATA.CT, DATA.Choice, DATA.SQ, DATA.GOS, DATA.CEN, DATA.VIS==2,
DATA.VIS==1, -DATA.FEE/4/10];

tableDCE = array2table(tableDCE, 'VariableNames', {'Id', 'Alt', 'CS', 'Choice', 'Status quo',
'Passive protection of all commercial forests', 'Passive protection of all natural regeneration
forests', 'Reducing the number of visitors to 7500 per day', 'Reducing the number of visitors to
5000 per day', '-Cost (10 EUR)'});
```

The data in our example concerns respondents' preferences toward applying passive protection in commercial or natural regeneration forests, reducing the number of visitors, cost, time and age.

DATA | tableDCE

28404x12 table

	1 Id	2 Alt	3 CS	4 Choice	5 Status quo	6 Passive protection of all comn	7 Passive protection of all natur	8 Reducing the number of visito	9 Reducing the number of visito	10 -Cost (10 EUR)	11 Time	12 Age
1	47411	1	1	0	1	0	0	0	0	0	0.1637	2
2	47411	2	1	1	0	1	1	1	0	-0.6250	0.1637	2
3	47411	3	1	0	0	0	0	0	1	-1.2500	0.1637	2
4	47411	1	2	0	1	0	0	0	0	0	0.1637	2
5	47411	2	2	1	0	0	0	0	1	-0.6250	0.1637	2
6	47411	3	2	0	0	1	1	0	0	-1.2500	0.1637	2
7	47411	1	3	0	1	0	0	0	0	0	0.1637	2
8	47411	2	3	1	0	0	0	0	0	-0.6250	0.1637	2
9	47411	3	3	0	0	1	1	1	0	-1.2500	0.1637	2
10	47411	1	4	0	1	0	0	0	0	0	0.1637	2
11	47411	2	4	0	0	1	1	0	0	-1.2500	0.1637	2
12	47411	3	4	1	0	0	0	1	0	-1.8750	0.1637	2
13	47411	1	5	0	1	0	0	0	0	0	0.1637	2
14	47411	2	5	0	0	0	1	1	0	-1.2500	0.1637	2
15	47411	3	5	1	0	1	0	0	1	-1.8750	0.1637	2
16	47411	1	6	0	1	0	0	0	0	0	0.1637	2
17	47411	2	6	0	0	1	0	0	1	-1.2500	0.1637	2
18	47411	3	6	1	0	0	1	0	0	-1.8750	0.1637	2
19	47411	1	7	0	1	0	0	0	0	0	0.1637	2
20	47411	2	7	0	0	1	1	0	1	-1.8750	0.1637	2
21	47411	3	7	1	0	0	0	0	0	-2.5000	0.1637	2
22	47411	1	8	0	1	0	0	0	0	0	0.1637	2
23	47411	2	8	0	0	0	1	0	0	-1.8750	0.1637	2
24	47411	3	8	1	0	1	0	1	0	-2.5000	0.1637	2

Model Specifications

Now we can move to description of some model specifications. First, we can name our project, which will be also used for naming xls results files.

```
EstimOpt.ProjectName = 'demo';
```

Later we can prepare choice variables (**Y**), attributes (**Xa**), and their names. This step is not necessary, however, might be useful if you are planning on estimating models on some subsets of your data and use filters.

```
DATA.Y = DATA.Choice; % explained variable (choice indicator)
DATA.Xa = [DATA.SQ, DATA.GOS, DATA.CEN, DATA.VIS==2, DATA.VIS==1, -DATA.FEE/4/10]; % attributes
EstimOpt.NamesA = {'Status quo';'Passive protection of all commercial forests';'Passive protection of all natural regeneration forests';'Reducing the number of visitors to 7500 per day';'Reducing the number of visitors to 5000 per day';'-Cost (10 EUR)'}; % Names of the attributes
```

You can also declare other variables, including:

- Explanatory variables of random parameters means/interactions (**Xm**)

```
% DATA.Xm = [];
% EstimOpt.NamesM = {''};
```

- Explanatory variables of scale (**Xs**)

```
% DATA.Xs = [];
% EstimOpt.NamesS = {''};
```

- Explanatory variables of scale variance (GMXL model) (**Xt**)

```
% INPUT.Xt = [];
% EstimOpt.NamesT = {''};
```

- Explanatory variables of class membership (Latent class models) (**Xc**)

```
% DATA.Xc = [];
% EstimOpt.NamesC = {''};
```

- Measurment equations variables (Hybrid models) (**Xmea**)

```
DATA.Xmea = [DATA.TIME]; %
EstimOpt.NamesMea = {'DCE time (normalized)'};
```

- Additional covariates explaining mesurment equations (Hybrid models) (Xmea_exp)

```
% DATA.Xmea_exp = [];
% EstimOpt.NamesMea_exp = {''};
```

- Structural equations variables (Hybrid models) (**Xstr**)

```
% DATA.Xstr = [];
% EstimOpt.NamesStr = {''};
```

- Coordinate variables (Geographically weighted models) (**Crds**)

```
% DATA.Crds = [];
```

- Vector of expenses (**PriceMat** e.g. price) and constraints (**I** e.g. income) (Multiple discrete-continuous models)

```
% DATA.PriceMat = []; % expenses
% DATA.I = []; % constraints
```

- Vector with weights (**W**)

```
% DATA.W = []; %
```

All of those variables can be included in INPUT and hence included in the model. Keep in mind that it is necessary to always include INPUT.Y and INPUT.Xa, because they are always used in order to estimate the simple model.

Here, you can also specify a filter variable to include only a fraction of observation (e.g. model for male respondents only), which can be used when specifying INPUT.Y and INPUT.Xa (and other explanatory variables).

```
DATA.filter = ones(size(DATA.Y)) == 1;

INPUT.Y = DATA.Y(DATA.filter);
INPUT.Xa = DATA.Xa(DATA.filter,:);

% INPUT.Xm = DATA.Xm(DATA.filter,:); % Explanatory variables of random parameters means / inreactions
% INPUT.Xs = DATA.Xs(DATA.filter,:); % Explanatory variables of scale
% INPUT.Xt = DATA.Xt(DATA.filter,:); % Explanatory variables of scale variance (GMXL model)
% INPUT.Xc = DATA.Xc(DATA.filter,:); % Explanatory variables of class membership (Latent class models)
INPUT.Xmea = DATA.Xmea(DATA.filter,:); % Measurment equations variables (Hybrid models)
% INPUT.Xmea_exp = DATA.Xmea_exp(DATA.filter,:); % Additional covariates explaining mesurment equations (Hybrid models)
% INPUT.Xstr = DATA.Xstr(DATA.filter,:); % Structural equations variables (Hybrid models)
% INPUT.Crds = DATA.Crds(DATA.filter,:); % Coordinates (GWMNL model)
% INPUT.PriceMat = DATA.PriceMat(DATA.filter,:); % vector for expenses (MDCEV models)
% INPUT.I = DATA.I(DATA.filter,:); % vector for constraints (MDCEV models)
% INPUT.W = DATA.W(DATA.filter,:); % vector with weights
```

Besides, you can indicate some missing observations (e.g. choice tasks with no answer).

```
% INPUT.MissingInd = DATA.SKIP(DATA.filter,:);
```

Here, we present a summary of possible additional explanatory variables.

	MNL	MXL	GMXL	LC	LCMXL	MIMIC	HMNL	HMXL	HLC	GWMNL	LML	MDCEV	MMDCEV
Y	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Xa	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Xm	☑	☑	☑									☑	☑
Xs	☑	☑	☑	☑	☑								
Xt			☑										
Xc				☑	☑				☑				
Xmea						☑	☑	☑	☑				
Xmea_exp						☑	☑	☑	☑				
Xstr						☑	☑	☑	☑				
PriceMat												☑	
I												☑	
W	☑	☑	☑	☑	☑	☑	☑	☑	☑		☑	☑	☑
Crds										☑			
MissingInd	☑	☑	☑	☑	☑	☑	☑	☑	☑				

After specifying INPUT, it is also needed to specify the number of choice tasks per person, the number of alternatives in a choice task, and a number of respondents.

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

After that, you can run the DataCleanDCE function (or DataCleanVCE for the MDCEV model), which helps in checking and reorganizing the data in INPUT. It also declares baseline levels for optimization options and estimates a simple MNL model with only constants (which is needed in a bunch of tests).

```
[INPUT, Results, EstimOpt, OptimOpt] = DataCleanDCE(INPUT,EstimOpt);
% [INPUT, Results, EstimOpt, OptimOpt] = DataCleanVCE(INPUT,EstimOpt);
```

Estimation and Optimization options

Here, we present a list of optimization options, which can be adjusted accordingly to the researcher's will. Most of the time those options will not be changed, so we proposed some default levels. Nonetheless, we are aware that some advanced models might need different estimation approach, therefore we present possible estimation options that are currently available.

EstimOpt

- Display output

```
% EstimOpt.Display = 1; % shows output - default
% EstimOpt.Display = 0; % hides output
```

- Overall precision level

```
% EstimOpt.eps = 1.e-9; % default
```

- Exponent of parameters (column vector of 0s, for each parameter set to 1 the $Exp(\beta)$ will be used)

```
% EstimOpt.ExpB = [0;0;0;1;0;0]; % default – only zeros
```

- Constraining model parameter to its initial values

```
% EstimOpt.ConstVarActive = 0; % without constraints - default
% EstimOpt.ConstVarActive = 1; % with constraints
```

- User-supplied Hessian approximation

```
% EstimOpt.ApproxHess = 1; % BHHH Hessian - default
% EstimOpt.ApproxHess = 0; % analytical Hessian
```

- Hessian used in the estimation

```
% EstimOpt HessEstFix = 0; % optimization Hessian - default
% EstimOpt HessEstFix = 1; % jacobian-based (BHHH) Hessian
% EstimOpt HessEstFix = 2; % high-precision jacobian-based (BHHH) Hessian
% EstimOpt HessEstFix = 3; % numerical Hessian
% EstimOpt HessEstFix = 4; % analytical Hessian
```

- Robust standard error

```
% EstimOpt.RobustStd = 0; % without rse - default
% EstimOpt.RobustStd = 1; % with rse
```

- Gradient

```
% EstimOpt.NumGrad = 0; % analytical gradient - default
% EstimOpt.NumGrad = 1; % numerical gradient
```

- Skipping rows in the Halton sequence

```
% EstimOpt.HaltonSkip = 1; % default
```

- Leaping rows in the Halton sequence

```
% EstimOpt.HaltonLeap = 0; % default
```

- Using the smallest positive normalized floating-point number in IEEE® double precision as the minimum value

```
% EstimOpt.RealMin = 0; % not using RealMin - default
```

```
% EstimOpt.RealMin = 1; % using RealMin
```

- Number of draws for numerical simulation for models with random parameters

```
% EstimOpt.NRep = 1e4; % default
```

- Draws used in numerical simulation for models with random parameters

```
% EstimOpt.Draws = 1; % pseudo-random
% EstimOpt.Draws = 2; % Latin Hypercube
% EstimOpt.Draws = 3; % Halton
% EstimOpt.Draws = 4; % Halton RR scrambled
% EstimOpt.Draws = 5; % Sobol
% EstimOpt.Draws = 6; % Sobol MAO scrambled
```

- Number of draws for numerical simulation

```
% EstimOpt.NRep = 1e3; % default
```

- Number of draws for simulating standard deviations

```
% EstimOpt.NSdSim = 1e4; % default
```

OptimOpt

The next set of optimization options concerns features for Matlab's built-in fmincon function, which is used for finding the minimum of constrained nonlinear multivariable function. Here are copy-pasted descriptions of possible optimization options of optimoptions('fmincon').

- Algorithm used by the solver

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

- Compare user-supplied analytic derivatives (gradients or Jacobian, depending on the selected solver) to finite differencing derivatives

```
% OptimOpt.CheckGradients = 0; % default
```

- Objecive gradient for Jacobian

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

- User-supplied Hessian

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

- Level of display

```
% OptimOpt.Display = 'off; % displays no output
% OptimOpt.Display = 'iter; % displays output at each iteration, and gives the default exit
message
% OptimOpt.Display = 'iter-detailed; % displays output at each iteration, and gives the
technical exit message
% OptimOpt.Display = 'notify; % displays output only if the function does not converge, and
gives the default exit message
% OptimOpt.Display = 'notify-detailed; % displays output only if the function does not
converge, and gives the technical exit message
```

```
% OptimOpt.Display = ‘final’; % displays just the final output, and gives the default exit
message  - default
% OptimOpt.Display = ‘final-detailed; % displays just the final output, and gives the technical
exit message
```

- Finite differences, used to estimate gradients

```
% OptimOpt.FiniteDifferenceType = ‘forward’; % default
% OptimOpt.FiniteDifferenceType = ‘central; % takes twice as many function evaluations but
should be more accurate. 'central' differences might violate bounds during their evaluation in
fmincon interior-point evaluations if the HonorBounds option is set to false
```

- Termination tolerance on the function value

```
% OptimOpt.FunctionTolerance = 1.0000e-06;
```

- Method of Hessian approximation

```
% OptimOpt.HessianApproximation = ‘bfgs’; % default
% OptimOpt.HessianApproximation = ‘lbfgs’;
% OptimOpt.HessianApproximation = {‘lbfgs’,Positive Integer};
% OptimOpt.HessianApproximation = ‘finite-difference’;
```

- Maximum number of function evaluations allowed

```
% OptimOpt.MaxFunctionEvaluations = 600000; % default
```

- Maximum number of iterations allowed

```
% OptimOpt.MaxIterations = 10000; % default
```

- Objective function value limit (if the objective function value goes below ObjectiveLimit and the iterate is feasible, then the iterations halt)

```
% OptimOpt.ObjectiveLimit = -1.0000e+20; % default
```

- Termination tolerance on the first-order optimality

```
% OptimOpt.OptimalityTolerance = 1.0000e-06; % default
```

- Termination tolerance on x

```
% OptimOpt.StepTolerance = 1.0000e-06; % default
```

- Applicable solvers estimate gradients in parallel

```
% OptimOpt.UseParallel = 0; % default
```

Model examples and other model-specific estimation options

MNL – Multinomial Logit Model

Let's start with simple multinomial logit model for choice data.

- Specify if you want to estimate tee the model in WTP-space (monetary attribute should come last in Xa) – works also for MXL model

```
% EstimOpt.WTP_space = 0; % default
% EstimOpt.WTP_space = 1;
```

- Specify which monetary parameter is used for which non-monetary attribute for WTP-space models – works also for MXL model

```
% EstimOpt.WTP_matrix = [];
```

- Specify which attributes are subject to non-linear transformations (vector) – works also for MXL model

```
% EstimOpt.NLTVariables = []; % choose which Xa elements to be non-linearly transformed
```

- Specify the transformation for non-linear variables – works also for MXL model

```
% EstimOpt.NLTType = 1 % Box-Cox transformations - default (if used);
% EstimOpt.NLTType = 1 % Yeo-Johnson transformations (MNL and MXL only)
```

◆ In order to estimate the model, just use the command below:

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

MXL – Mixed (random parameters) Logit Model

If you do not want to keep the model parameters fixed, you can use a model with random parameters.

- Specify if random parameters are correlated

```
% EstimOpt.FullCov = 0; % default - Mixed Logit Model without correlated random parameters
% EstimOpt.FullCov = 1; % Mixed Logit Model with correlated random parameters
```

- Specify the distribution of random parameters (for models with random parameters, i.e. model with MXL in the name). Set in a vector of numbers, each corresponding to specific distribution:
  - -1 - constant
  - 0 - normal
  - 1 - lognormal
  - 2 - spike
  - 3 - Triangular
  - 4 - Weibull (lambda - scale, k - shape) (lambda>0 & k>0) support x: [0,+infty)
  - 5 - Sinh-Arcsinh
  - 6 - Johnson Sb
  - 7 - Johnson Su
  - 9 - Uni-Log/Reciprocal (a - lower bond, b - upper bond) (0<a<b) support x: [a,b]
  - 10 - Pareto (xm - scale, alpha - shape) (xm>0 & alpha>0) support x: [xm,+infty)
  - 11 - Lomax (alpha - shape, lambda - scale) (alpha>0 & lambda>0) support x: [0,+infty)
  - 12 - Logistic (mi - location, s - scale) (s>0) support x: (-infty,+infty)
  - 13 - Log-Logistic (alpha - scale, beta - shape) (alpha>0 & beta>0) support x: [0,+infty)
  - 14 - Gumbel (mi - location, beta - scale) (beta>0) support x: (-infty,+infty)
  - 15 - Cauchy (x0 - location, gamma - scale) (gamma>0) support x: (-infty,+infty)
  - 16 - Rayleigh (sigma - scale) (sigma>0) support x: [0,+infty)
  - 17 - Exponential (lambda - scale) (lambda>0) support x: [0,+infty)

```
EstimOpt.Dist = [0;0;0;0;0;1]; % in that example we assume normal dist. for all parameters,
except lognormal dist. for cost parameter
```

- Specify if you want to predict individual-specific parameters / WTP scores using the Bayes rule

```
% EstimOpt.Scores = 0; % default
% EstimOpt.Scores = 1; %
```

◆ In order to estimate the model, just use the command below:

```
% MXL_d: Mixed Logit Model without correlated random parameters
EstimOpt.FullCov = 0;
Results.MXL_d = MXL(INPUT,Results,EstimOpt,OptimOpt);

% MXL: Mixed Logit Model with correlated random parameters
EstimOpt.FullCov = 1;
Results.MXL = MXL(INPUT,Results,EstimOpt,OptimOpt);
```

GMXL – Generalized Mixed Logit Model

Sometimes you might also be interested in estimating Generalized Mixed Logit Model that accounts for scale and taste heterogeneity.

- Specify if random parameters are correlated

```
% EstimOpt.FullCov = 0; % default – Model without correlated random parameters
% EstimOpt.FullCov = 1; % Model with correlated random parameters
```

- Specify Gamma0 type

```
% EstimOpt.Gamma0 = 0.5; % default
% EstimOpt.Gamma0 = 0; % GMXL type II
% EstimOpt.Gamma0 = 1; % GMXL type I
% Value in between allows for gamma to be freely estimated.
```

- Specify the distribution of random scale and taste heterogeneity parameters. Set in a vector of numbers, each corresponding to specific distribution:
  - -1 - constant
  - 0 - normal
  - 1 - lognormal
  - 2 - spike
  - 5 - Weibull (lambda - scale, k - shape) (lambda>0 & k>0) support x: [0,+infty)

```
EstimOpt.DistS = [0;0;0;0;0;1]; % in that example we assume normal dist. for all parameters,
except lognormal dist. for cost parameter
```

◆ In order to estimate the model, just use the command below:

```
% MXL_d: Generalized Mixed Logit Model without correlated random parameters
EstimOpt.FullCov = 0;
Results.GMXL_d = GMXL(INPUT,Results,EstimOpt,OptimOpt);

% MXL: Generalized Mixed Logit Model with correlated random parameters
EstimOpt.FullCov = 1;
Results.GMXL = GMXL(INPUT,Results,EstimOpt,OptimOpt);
```

LC – Latent Class Model

You might also be interested in estimating Latent Class Model that allows the parameter to fall into some different classes.

- Specify number of latent classes

```
% EstimOpt.NClass = 2; % default
```

- Specify if you want to predict individual-specific class membership probabilities using Bayes rule

```
% EstimOpt.ClassScores = 1; % default (yes)
```

- Specify which coefficients should be kept constant (=0)

```
% EstimOpt.BActiveClass = ones(EstimOpt.NVarA,1); % default
```

◆ In order to estimate the model, just use the command below:

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

LCMX – Latent Class Mixed Logit Model

You can also add some random parameters to the Latent Class Model.

- Specify if random parameters are correlated

```
% EstimOpt.FullCov = 0; % default – Model without correlated random parameters
% EstimOpt.FullCov = 1; % Model with correlated random parameters
```

◆ In order to estimate the model, just use the command below:

```
% LCMXL_d: Latent Class Mixed Logit Model without correlated random parameters
EstimOpt.FullCov = 0;
Results.LCMXL_d = LCMXL(INPUT,Results,EstimOpt,OptimOpt);

% MXL: Latent Class Mixed Logit Model with correlated random parameters
EstimOpt.FullCov = 1;
Results.LCMXL = LCMXL(INPUT,Results,EstimOpt,OptimOpt);
```

MIMIC – Multiple Indicators Multiple Causes Model

In our package you can also estimate Multiple Indicators Multiple Causes Model if you want to focus on the effects of the unobservable latent variables in situation when causes of the latent variable are observed.

- Specify if you want to standardize mean and variance of explanatory variables in INPUT.Xstr

```
% EstimOpt.StrNorm = ones(1,9); % (default = true)
```

- Specify number of latent variables

```
% EstimOpt.NLatent = 1; % default
```

- Specify number of latent classes

```
% EstimOpt.NClass = 2; % default
```

- Specify names of classes

```
% EstimOpt.NamesC = []; % default
```

- Provide a matrix of measurement equations (by default model is assuming that every latent variable is in every measurement equation)

```
% EstimOpt.MeaMatrix = [];
```

- Provide a matrix associating additional explanatory variables with measurement equations

```
% EstimOpt.MeaExpMatrix = [];
```

- Specify measurement equation type:
  - 0 – OLS
  - 1 – MNL
  - 2 – ordered probit
  - 3 – Poisson
  - 4 – Negative Binomial
  - 5 – ZIP
  - 6 – ZINB

```
% EstimOpt.MeaSpecMatrix = [2 2 1 2 2 0 5 2]; % example
```

◆ In order to estimate the model, just use the command below:

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

HMNL – Hybrid Multinomial Logit Model

You can also estimate Hybrid choice models that include attitudinal variables.

- Specify number of latent variables

```
% EstimOpt.NLatent = 1; % default
```

- Provide a matrix associating measurement equations with LV

```
% EstimOpt.MeaMatrix = [];
```

- Provide a matrix associating additional explanatory variables with measurement equations

```
% EstimOpt.MeaExpMatrix = [];
```

- Specify measurement equation type:
  - 0 – OLS
  - 1 – MNL
  - 2 – ordered probit
  - 3 – Poisson
  - 4 – Negative Binomial
  - 5 – ZIP
  - 6 – ZINB

```
% EstimOpt.MeaSpecMatrix = [2 2 1 2 2 0 5 2]; % example
```

- Provide limit for censoring count variables (in XMea)

```
% EstimOpt.CountCut = 80; % default
```

- Specify if scale should be interacted with LV

```
% EstimOpt.ScaleLV = 1; % default
```

◆ In order to estimate the model, just use the command below:

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

HLC – Hybrid Latent Class Model

You can also include latent classes in your hybrid choice model.

◆ In order to estimate the model, just use the command below:

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

HMXL – Hybrid Mixed Logit Model

You can also include random parameters in your hybrid choice model.

- Specify if random parameters are correlated

```
% EstimOpt.FullCov = 0; % default - Model without correlated random parameters
% EstimOpt.FullCov = 1; % Model with correlated random parameters
```

◆ In order to estimate the model, just use the command below:

```
% MXL_d: Hybrid Mixed Logit Model without correlated random parameters
EstimOpt.FullCov = 0;
Results.HMXL_d = HMXL(INPUT,Results,EstimOpt,OptimOpt);

% MXL: Hybrid Mixed Logit Model with correlated random parameters
EstimOpt.FullCov = 1;
Results.HMXL = HMXL(INPUT,Results,EstimOpt,OptimOpt);
```

GWMNL – Geographically Weighted Multinomial Logit Model

You can also estimate the Geographically Weighted Multinomial Logit Model when you can also specify bandwidth.

- Specify bandwidth type

```
% EstimOpt.BandType = 1; % default – global bandwidth
% EstimOpt.BandType = 2; % spatially varying bandwidth
% EstimOpt.BandType = 3; % square root of spatially varying bandwidth
```

- Specify if model should find the optimal bandwidth value

```
% EstimOpt.BandSearch = 0; % default – model uses fixed bandwidth as provided in BandVal
% EstimOpt.BandSearch = 1; % model will find the optimal bandwidth value
```

- Specify default bandwidth value

```
% EstimOpt.BandVal = 3; % default
```

- Specify bandwidth lower bound

```
% EstimOpt.BandLower = 0; % default
```

- Specify bandwidth upper bound

```
% EstimOpt.BandLower = 5; % default
```

- Clustered standard errors

```
% EstimOpt.Clustered  = 1; % default (yes)
% EstimOpt.Clustered  = 0;
```

- Specify the type of summing

```
% EstimOpt.WeightSum  = 1; % default – no particular summing
% EstimOpt.WeightSum  = 2; % summing to 1
% EstimOpt.WeightSum  = 3; % summing to NP
```

- Bootstrappingng

```
% EstimOpt.BStrap  = 0; % default (without bootstraping)
% EstimOpt.BStrap  = 1;
```

- Number of bootstrap replications

```
% EstimOpt.NBStrap  = 100; % default
```

◆ In order to estimate the model, just use the command below:

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

LML – Mixed Logit Model with a Flexible Mixing Distribution

You can also estimate mixed logit model that allows for a flexible mixing distribution.

- Specify the distribution of random parameters (for models with random parameters, i.e. model with MXL in the name). Set in a vector of numbers, each corresponding to specific distribution:
  - 0 – approximate normal
  - 1 – approximate lognormal
  - 2 – Legendre polynomial (normal)
  - 3 – Legendre polynomial (log-normal)
  - 4 – Step function
  - 5 – Linear Spline
  - 6 – Cubic Spline
  - 7 – Piece-wise Cubic Spline
  - 8 – Piece-wise Cubic Hermite Interpolating Spline

```
EstimOpt.Dist = [];
```

- ? Auxiliary variable for distributions of random parameters
  - dist=0 or dist=1 - for approximate it is Order of approximation,
  - dist=2 or dist=3 - for Legendre polynomial(s) it is their order,
  - dist=4 - for step function it is number of step function segments,
  - dist=5,6,7,8 - for splines it's number of spline knots (including bonds)

```
EstimOpt.NOrder= [];
```

- Number of grids

```
% EstimOpt.NGrid  = 1000; % default
```

- User defined step function

```
% EstimOpt.StepFun  = []; % default
```

- Drawing plot

```
% EstimOpt.PlotIndex  = 0; % default – not drawing plot
% EstimOpt.PlotIndex  = 1; % drawing plot
```

- Creating output

```
% EstimOpt.NoOutput  = 0; % default – not creating output
% EstimOpt.NoOutput  = 1; % creating output
```

◆ In order to estimate the model, just use the command below:

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

MDCEV – Multiple Discrete-Continous Extreme Value Model

You can also estimate multiple discrete-continuous extreme value model that allows for both discrete and continuous choice.

- Specify profiles

```
% EstimOpt.Profile = 1; % default – Alpha
% EstimOpt.Profile = 2 % Gamma

%  EstimOpt.SpecProfile = [1 1 1; 1 1 1]; % default (first row for Alphas, second for Gammas)
```

◆ In order to estimate the model, just use the command below:

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

MMDCEV – Mixed Multiple Discrete-Continous Extreme Value Model

You can also estimate multiple discrete-continuous extreme value with random parameters.

- Specify if random parameters are correlated

```
% EstimOpt.FullCov = 0; % default – Model without correlated random parameters
% EstimOpt.FullCov = 1; % Model with correlated random parameters
```

◆ In order to estimate the model, just use the command below:

```
% MMDCEV_d: Mixed Multiple Discrete-Continous Extreme Value Model without correlated random
parameters
EstimOpt.FullCov = 0;
Results. MMDCEV_d = MMDCEV(INPUT,Results,EstimOpt,OptimOpt);

% MXL: Mixed Multiple Discrete-Continous Extreme Value Model with correlated random parameters
EstimOpt.FullCov = 1;
Results.MMDCEV = MMDCEV(INPUT,Results,EstimOpt,OptimOpt);
```

MNL, MXL_d & MXL – Starting values, interactions and Bactive example

Sometimes it is needed to specify the starting values manually. In that case, some basic knowledge of discrete choice models is needed. Here we present some examples of how to specify starting values. Shortly, the vector of starting parameters should be a column vector with a length equal to the number of parameters of the declared model.

We will also provide an example for a model that includes interactions (Xm) and how to constrain coefficients values through Bactive. Remember that in that case, you need to specify EstimOpt.NamesM and INPUT.Xm,before using DataCleanDCE function.

```
DATA.Xm = [DATA.AGE==3, DATA.AGE==4, DATA.AGE==5];
EstimOpt.NamesM = {'e.g. 26-44','e.g. 45-64','e.g. 65+'};
INPUT.Xm = DATA.Xm(DATA.filter,:);
[INPUT, Results, EstimOpt, OptimOpt] = DataCleanDCE(INPUT,EstimOpt);
```

Let's start with the simple MNL model with parameters β and attributes X described as $Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6$, where the backup vector should look like this:

β_1
β_2
β_3
β_4
β_5
β_6

In that case instead of $\beta_1, \dots, \beta_6$ you should declare some numbers, for instance:

```
B_backup =
[0.159954254207451;0.486588294026178;0.609085577346533;0.188275582287589;0.0682110004987603;0.5
69893495060325];
```

If we would also add interactions for new X_7, X_8 and X_9 attributes (specified in Xm), we would consider the model described as $Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_1 X_7 + \beta_8 X_2 X_7 + \beta_9 X_3 X_7 + \beta_{10} X_4 X_7 + \beta_{11} X_5 X_7 + \beta_{12} X_6 X_7 + \beta_{13} X_1 X_8 + \beta_{14} X_2 X_8 + \beta_{15} X_3 X_8 + \beta_{16} X_4 X_8 + \beta_{17} X_5 X_8 + \beta_{18} X_6 X_8 + \beta_{19} X_1 X_9 + \beta_{20} X_2 X_9 + \beta_{21} X_3 X_9 + \beta_{22} X_4 X_9 + \beta_{23} X_5 X_9 + \beta_{24} X_6 X_9$ and the backup vector would look like this:

β_1
β_2
…
β_{24}

Where instead of $\beta_1, \beta_2, \dots, \beta_{24}$ you should declare some numbers, for instance:

```
B_backup = [-
0.0649871757079609;0.516086083936041;0.764495593032389;0.175803035783724;0.0423877876196776;0.6
56304581992665;0.328582977085665;0.0112058878541658;-0.183656735546277;-
0.0571270745882275;4.33728554883648e-05;-0.168997051214565;0.14460696302212;-
0.0611137990713113;-
0.204712689328378;0.169826650333787;0.10431372306787;0.0474591006391481;0.401634785987956;-
0.0535136809608674;-0.218576712727205;-0.0601489890120321;0.00391361371887034;-
0.20931481910203];
```

However, sometimes you might not be interested in the estimation of all the possible interactions. For example if you would be interested only with cost (X_6) interactions β_{12}, β_{18} and β_{24}, you should use EstimOpt.BActive column vector, which holds an information which attribute should be keep fixed during estimation (and hence if the starting values would be set to 0, they will not affect the results). In that case the backup vector would look like this:

β_1
β_2
β_3
β_4
β_5
β_6
0

…
0
β_{12}
0
…
0
β_{18}
0
…
0
β_{24}

Where instead of $\beta_1, \beta_2, \ldots, \beta_6, \beta_{12}, \beta_{18}, \beta_{24}$ you should declare some numbers, for instance:

```
B_backup =
[0.157151789122682;0.486591518506085;0.608999367821945;0.187839637442765;0.0683659481870268;0.4
92009465386079;0;0;0;0;0;0.0631226149552788;0;0;0;0;0;0.153313208191371;0;0;0;0;0;0.09092255375
19497];
```

In that case the EstimOpt.BActive vector should look like this (0 if we want to keep the parameters fixed):

1
1
1
1
1
1
0
0
0
0
0
1
0
0
0
0
0
1
0
0
0
0
0
1

```
EstimOpt.BActive = [1;1;1;1;1;1;0;0;0;0;0;1;0;0;0;0;0;1;0;0;0;0;0;1];
```

In the following examples, we will describe how to define starting values for MXL_d and MXL model. Remember, that accounting interactions in those models work very similarly because those parameters are added at the bottom of a vector with parameters (e.g. B_backup or Results.MNL.bhat, Results.MXL_d.bhat, Results.MXL.bhat). The application of EstimOpt.Bactive is also similar because the size of this vector has to be the same as the length of the parameter's vector with values 1 and 0 (reminder: 0 if you want to keep the value of the parameter fixed).

Specifying starting values for MXL_d model is slightly different than for the MNL model. In that case, parameters come from different distributions, so they can be described by a set of distributional parameters. Keep in mind that for normal distribution the first parameter (μ) describe the mean of the distribution and the second pparameter(σ^2) describe its variance. However, it is not the case for other distributions (including Lognormal, Spike, Triangular, Weibull, Sing-Arcsinh, Johnson Sb, Johnson Su, Uni-Log, Pareto, Lomax, Logistic, Log-Logistic, Gumbel, Cauchy, Rayleigh and Exponential). Hence B_backup variable holds information about the value of 1st and 2nd distributional parameters respectively. In that case for a model as $Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6$, where $\beta_1, \ldots, \beta_6$ should be derived from specific distributions, the starting values should be inserted as:

β_{1_par1}
β_{2_par1}

β_{3_par1}
β_{4_par1}
β_{5_par1}
β_{6_par1}
β_{1_par2}
β_{2_par2}
β_{3_par2}
β_{4_par2}
β_{5_par2}
β_{6_par2}

Where instead of $\beta_{1\_par1}, \dots, \beta_{6\_par2}$ you should declare some numbers, for instance:

```
B_backup = [-
1.08619793320482;0.642321348105622;0.865070353929383;0.260738389970861;0.107349312574969;-
0.0992115616773407;3.04470273652455;0.872246737738571;0.893815340001261;0.611809485841118;-
0.236528015369361;1.26423897882879];
```

Adding interactions is similar to the MNL model, because it concerns interacting with means of random parameters.

If we would want to add some interactions (continuing the previous example with X_4 and X_5), we would consider the model described as $Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_1 X_7 + \beta_8 X_2 X_7 + \beta_9 X_3 X_7 + \beta_{10} X_4 X_7 + \beta_{11} X_5 X_7 + \beta_{12} X_6 X_7 + \beta_{13} X_1 X_8 + \beta_{14} X_2 X_8 + \beta_{15} X_3 X_8 + \beta_{16} X_4 X_8 + \beta_{17} X_5 X_8 + \beta_{18} X_6 X_8 + \beta_{19} X_1 X_9 + \beta_{20} X_2 X_9 + \beta_{21} X_3 X_9 + \beta_{22} X_4 X_9 + \beta_{23} X_5 X_9 + \beta_{24} X_6 X_9$ and the backup vector would look like this:

β_{1_par1}
β_{2_par1}
β_{3_par1}
β_{4_par1}
β_{5_par1}
β_{6_par1}
β_{1_par2}
β_{2_par2}
β_{3_par2}
β_{4_par2}
β_{5_par2}
β_{6_par2}
β_7
…
β_{24}

Where instead of $\beta_{1\_par1}, \dots, \beta_{24}$ you should declare some numbers, for instance:

```
B_backup = [-
1.23685884591626;0.679719111089912;1.06706905559448;0.233782437696263;0.0670126109050347;-
0.0947684440480008;3.28854699236889;0.86847941847826;0.88563006752372;0.631869482261344;-
0.265050188044538;1.05668467646537;0.598254738657584;0.0224204349313097;-0.19815409257359;-
0.0540320181435001;0.00397739850289974;-0.136819869420428;-0.0537098579662287;-
0.0725358644643713;-
0.273618863194524;0.244079388341653;0.165361901927099;0.243590109184194;0.600681298692093;-
0.0938455827541106;-0.288702111002723;-0.0584351324695277;0.00873834035928742;-
0.287535655048998];
```

The EstimOpt.BActive works the same as before, hence its size should be the same as the size of B_backup. As before, you just need to specify which value should be kept fixed and change corresponding elements of B_backup and EstimOpt.BActive to 0.

The aforementioned MXL_d model does not concern the correlated random parameters. In the MXL model (with correlated random parameters) the parameters are drawn from distributions by using draws already used from previous parameters. For instance let's assume that $\beta_1, \dots, \beta_5$ come from normal distribution and β_6 from lognormal distribution. In that case:

$\beta_1 = \quad \beta_{1_{par1}} + \beta_{1_{par2-1}} * draws_{\beta_1}$

$$\beta_2 = \beta_{2_{par_1}} + \beta_{1_{par2-2}} * draws_{\beta_1} + \beta_{2_{par2-1}} * draws_{\beta_2}$$

$$\beta_3 = \beta_{3_{par_1}} + \beta_{1_{par2-3}} * draws_{\beta_1} + \beta_{2_{par2-2}} * draws_{\beta_2} + \beta_{3_{par2-1}} * draws_{\beta_3}$$

$$\beta_4 = \beta_{4_{par_1}} + \beta_{1_{par2-4}} * draws_{\beta_1} + \beta_{2_{par2-3}} * draws_{\beta_2} + \beta_{3_{par2-2}} * draws_{\beta_3} + \beta_{4_{par2-1}} * draws_{\beta_4}$$

$$\beta_5 = \beta_{5_{par_1}} + \beta_{1_{par2-5}} * draws_{\beta_1} + \beta_{2_{par2-4}} * draws_{\beta_2} + \beta_{3_{par2-3}} * draws_{\beta_3} + \beta_{4_{par2-2}} * draws_{\beta_4} + \beta_{5_{par2-1}} * draws_{\beta_5}$$

$$\beta_6 = \exp(\beta_{6_{par_1}} + \beta_{1_{par2-6}} * draws_{\beta_1} + \beta_{2_{par2-5}} * draws_{\beta_2} + \beta_{3_{par2-4}} * draws_{\beta_3} + \beta_{4_{par2-3}} * draws_{\beta_4} + \beta_{5_{par2-2}} * draws_{\beta_5} + \beta_{6_{par2-1}} * draws_{\beta_6})$$

Therefore, the backup vector should be defined in such situation as:

$\beta_{1_{par1}}$
$\beta_{2_{par1}}$
$\beta_{3_{par1}}$
$\beta_{4_{par1}}$
$\beta_{5_{par1}}$
$\beta_{6_{par1}}$
$\beta_{1_{par2}-1}$
$\beta_{1_{par2}-2}$
$\beta_{1_{par2}-3}$
$\beta_{1_{par2}-4}$
$\beta_{1_{par2}-5}$
$\beta_{1_{par2}-6}$
$\beta_{2_{par2}-1}$
$\beta_{2_{par2}-2}$
$\beta_{2_{par2}-3}$
$\beta_{2_{par2}-4}$
$\beta_{2_{par2}-5}$
$\beta_{3_{par2}-1}$
$\beta_{3_{par2}-2}$
$\beta_{3_{par2}-3}$
$\beta_{3_{par2}-4}$
$\beta_{4_{par2}-1}$
$\beta_{4_{par2}-2}$
$\beta_{4_{par2}-3}$
$\beta_{5_{par2}-1}$
$\beta_{5_{par2}-2}$
$\beta_{6_{par2}-1}$

You can also add interactions or keep parameters fixed by using EstimOpt.BActive for this model. However, keep in mind that currently the model with correlated random parameters works only for normal and lognormal distributions.