CMC choice modelling code for R

www.cmc.leeds.ac.uk

The Choice Modelling Centre (CMC) at the University of Leeds has developed flexible estimation code for choice models in R. The code uses the complete opposite of a black-box approach, i.e. the user sees every step in the coding of a log-likelihood function. We believe this to be essential in ensuring a greater understanding by users of the very powerful models they have at their disposal.

CMC has developed code for all different types of choice models, including but not limited to Multinomial Logit, Nested Logit, Cross-Nested Logit, Mixed Logit, Latent Class, Mixed GEV and hybrid choice models. We have also produced code for the estimation of Multiple Discrete Continuous Extreme value models, mixed logit models allowing for both inter and intra-respondent heterogeneity, as well as code using alternative estimation approaches, including Bayesian techniques and the EM algorithm. The code has been tested extensively and is relatively robust, although the reliance on numerical derivatives will of course lead to occasional issues. We accept no liability for any use of the code.

The code we make available to users on our website covers simple implementations of Multinomial, Nested and Mixed Logit on a mode choice data set with four alternatives (car, air, rail and high speed rail). No detailed documentation is provided nor is there any support available. For users interested in learning more about our code, CMC runs two annual courses that combine theoretical lectures in choice modelling with an in-depth introduction to the R estimation code. For more details, please see www.cmc.leeds.ac.uk

A basic understanding of R is required to use the code. Background reading is for example available at https://www.r-project.org/about.html and http://www.statmethods.net/

A brief overview of the code for the three example files is provided overleaf.

We ask users of the code to cite it as follows:

CMC (2017), CMC choice modelling code for R, Choice Modelling Centre, University of Leeds, www.cmc.leeds.ac.uk

choice modelling centre

UNIVERSITY OF LEEDS

General instructions

Directory structure

- Copy the *function_input.R* file into a suitable directory on your computer. Only one copy of this file is needed per machine, and the file should not be changed.
- In each R model file, line 7 needs to be edited to set the correct directory for the location of the above *function_input.R* file
- When running a model, the base version of the code assumes that the data file is in the same directory as the R model file, and the user should thus select that directory as the working directory

Model settings

- Lines 23 to 28 contain settings to be changed for specific uses of the code
    - paneldata: set to 0 to make R treat the data as cross-sectional
    - mixing: set to 1 for any models including random parameters
    - functionality: covered in detail below
    - startingvaluesloop: set to 1 to have a loop over random starting values (with settings lower down in the code) prior to main estimation
    - saveestimates: set to 1 to save estimates and standard errors in a csv file
    - savecovar: set to 1 to save covariance matrix in a csv file

Running the code

The majority of the code should be self-explanatory. The core component is the loglike function, where the output depends on the setting used for functionality. With either functionality, the command *runmodel()* needs to be used.

Functionality = 1: model estimation

This uses maximum likelihood estimation to find values of the beta vector that maximise the log-likelihood of the model when using *runmodel()*. If this is followed by the command *modeloutput(model)*, the outputs of the model are printed on the screen, while *saveoutput(model)* saves them in a file with the same name as the model.

In the MNL example provided on our website, we also show the calculating of standard errors for willingness-to-pay (WTP) using the delta method.

Functionality = 2: prediction

This uses the values of the beta vector (i.e. should be run after finding the optimal values for beta) and applies the model to output the probabilities for each alternative in each choice situation. An example of this is implemented in the MNL code provided on the website.

Functionality = 3: saving unconditional and conditional draws

This uses the values of the beta vector (i.e. should be run after finding the optimal values for beta) and computes the conditional distributions for the random coefficients. These are then saved to a file as are the unconditional draws used in estimation. An example of this is implemented in the MMNL code provided on the website.