Analysing a Discrete Choice Experiment

Anna Heath and Petros Pechlivanoglou

Change `eval` to `TRUE` if you want to knit this document.

```
rm(list = ls())      # clear memory (removes all the variables from the workspace)
```

In this worksheet, we will be analysing the data from a discrete choice experiment (DCE) using `R`. To start, this worksheet will demonstrate how to analyse a DCE using a simple example with two potential choices. You will then complete a set of exercises to analyze a DCE with 3 potential options. For both questions, we will be using the `mlogit` package in `R` to undertake the analysis. So, to begin, we will load the relevant packages into `R`.

01 Load packages

```
if (!require('pacman')) install.packages('pacman'); library(pacman) # use this package to conveniently
# load (install if required) packages from CRAN
p_load( "dplyr", "mlogit", "boot")
```

02 Load data

The `mlogit` package contains several datasets from DCEs, which we will be using in these exercises. The advantage of using these datasets is that they are available in `R` so they do not need to be loaded from your file, as we saw in the Introduction to R exercises. The disadvantage is that these datasets are already formatted so we will not discuss how to format these datasets before loading them into `R`. In practice, you will need to format the data correctly before loading into `R` or you will require the data manipulation skills from the Introduction to `R` worksheets.

```
## Load the Train dataset from the mlogit package
data("Train", package = "mlogit")
Train$scenario <- 1:nrow(Train) # Add a "Scenario" index that is unique for each row of the data
```

The `Train` dataset is a sample of 235 Dutch individuals who made choices across 2929 scenarios to determine the relative value of four different components of train travelling; the price of ticket `price` (in cents of guilders - that's how old this dataset is!), the time taken for the journey `time` (in minutes), the comfort level of the journey `comfort` (coded 0, 1, 2 in order of decreasing comfort) and the number of changes `change`.

The `Train` dataset is in *wide* format, while the `mlogit` function requires that the data are in *long* format. The `dfidx` package and function have been developed alongside the `mlogit` package to transform the data from wide form to long form data.

```r
## Transform the data with the dfidx function
# Change data into wide format
Tr <- dfidx(Train,  # Dataset
            shape = "wide", # Shape of the dataset (wide or long)
            varying = 4:11, # The column indices where the attribute levels are stored
            sep = "_", # The separator after which the option disignation appears, _A and _B
            idx = list(c("scenario", "id")), # The variable names for the indices, scenario and patient
            choice = "choice", # Variable name for the variable that contains which choices was made
            drop.index = FALSE)

## Observe the long form data
Tr
```

Once we have formatted the data in long form, we can use the `mlogit` function to fit the logistic regression.

03 Fit a Standard Binomial Logit

We begin by fitting a standard binary logistic regression. This assumes that the errors in the utility measurements are independent and identically distribution with a Gumbel distribution. The `mlogit` function uses the standard framework for regression models in `R`. The first argument is a formula. The dependent variable is listed before the `~` and then all the independent variables are listed afterwards with a `+`. The second argument is the dataset name, in this case `Tr`.

```r
# Standard binary logit
model.standardlogit <- mlogit(choice ~ price + time + change + comfort, Tr)
summary(model.standardlogit)
```

This model summary includes the estimates for the coefficients, demonstrating that all four aspects of our journey are unlikely to have no impact on the calculation of the utility. Note that this model also includes an option specific intercept, essentially this is the component of the utility of choosing option `B` over option `A` that arises from the fact that it is option `B`. In general, this option specific intercept is valid if the options are "named" in some way, i.e., a named drug versus a generic, as it measures the additional importance of the named option.

The McFadden R^2 value offers an estimate of the goodness of fit of this model. Similar, to the R^2 measure in standard regression, the higher the value, the better the fit. However, values of McFadden R^2 are significantly smaller than standard R^2 values with values between 0.2 and 0.4 representing excellent fit.

The results from this model can be analysed to give the willingness-to-pay for a given attribute or the marginal rate of substitution if neither of the attributes are prices. The willingness-to-pay is equal to the ratio of two coefficients.

```r
## Willingness to pay or marginal rate of subsitution
coef(model.standardlogit)["time"] / coef(model.standardlogit)["price"]
# Consumer is willing to pay 19 guilder cents to reduce their travel time by one minute

coef(model.standardlogit)["change"] / coef(model.standardlogit)["price"]
# Consumer is willing to pay 219 guilder cents to avoid a change
```

The `mlogit` package also contains a function to estimate the probability that each option is taken for each scenario.

```
## The probability that each individual selects each option
head(fitted(model.standardlogit, type = "probabilities"))
```

04 Extensions to the standard model

It this example, the intercept term for option B has no interpretation as option B is not a specific names option. Thus, it would be more appropriate to fit the DCE model without the intercept term. This can be achieved by adding `+ 0` to the formula.

```
# Standard binary logit with no intercept
model.nointercept <- mlogit(choice ~ price + time + change + comfort + 0, Tr)
summary(model.nointercept)
```

This implicitly sets the utility when all attribute levels are 0 to 0, although this can be rescaled to estimate the "true" utility if alternative anchoring points are mode relevant.

An option specific slope for a given attribute or individual-level characteristic can be added using a `|` in the formula for the regression model. In this case, we add a 0 after the bar to indicate that the model should only include an option-specific slope and not an option-specific intercept.

```
# Standard binary logit with no intercept and option specific slope for comfort
model.optionspecific <- mlogit(choice ~ price + time + change + 0 | 0 + comfort, Tr)
summary(model.optionspecific)
```

The next extension to the standard model is a mixed logit model. A mixed logit model will allow the impact of each attribute to vary across individuals. This is equivalent to using a random slope in a mixed effect model. In the `mlogit` package, this random slope is assigned using the `rpar` argument. This argument lists all the attributes for which we would like to use a random effect and then states that the random effect can have one of four distributions (`n` for normal, `l` for log-normal, `t` for truncated normal and `u` for uniform).

At least two additional arguments are used to define how these random effects should be estimated and defined. Firstly `R` defines the number of random numbers that should be used to estimate the random effects. `panel` is then a logical that states whether the data consists of repeated observations from the same individuals (as we have in this setting) and if this should be included in the estimation procedure.

```
# Binary logit with random effect at the subject level
model.randomeffect <- mlogit(choice ~ price + time + change + comfort + 0, Tr,
                             rpar = c(price = "n", time = "n", change = "n",
                                      comfort = "n"),
              R = 100, panel = TRUE)
summary(model.randomeffect)
```

The random effects parameters are provided in this summary. In particular, the mean and standard deviation for each of the parameters are listed with each of the standard deviations estimated as 1. This indicates that there are some difficulties estimating these variance parameters.

The `mlogit` package can also extend to the nested logit model, which allows for more complex interdependencies using different arguments in the `mlogit()` function. The `?mlogit` help file can be used to explore these arguments or a full worked example can be found at this website `https://cran.r-project.org/web/packages/mlogit/vignettes/e2nlogit.html`.

05 Exercises

In this section, you will analyse a different DCE dataset from the `mlogit` package.

Load the Data

Firstly, we will load the `Heating` dataset which contains a sample of 900 Californian households to understand the preferences for different types of heating. to understand the dataset better you can type `?Heating`

```
data("Heating", package = "mlogit")
head(Heating)
```

The Heating dataset is also in *wide* form with the `idcase` column denoting the id for the individual household. Following this, the `depvar` variable contains the chosen alternative among 5 potential options:

1. `gc`: gas central heating
2. `gr`: gas heating in a specific room(s)
3. `ec`: electric central heating
4. `er`: electric heating in a specific room(s)
5. `hp`: a heat pump.

There are then two key variables that are alternative specific:

1. `ic`: the insulation cost of the five alternatives for a given house
2. `oc`: the annual operating costs of the five alternatives for a given house.

There are then four household specific characteristics:

1. `income`: the annual income of the household
2. `agehed`: the age of the head of the household
3. `rooms`: the number of rooms in the house
4. `region`: a factor variable with 4 levels; `ncostl` (northern coastal region), `scostl` (southern coastal region), `mountn` (mountain region), `valley` (central valley region).

Exercise 1: Using the `dfidx` function, turn the `Heating` dataset into a long form dataset

```
### Your turn
Heat <- dfidx(Heating, # Dataset
              shape = "wide", # Shape of the dataset (wide or long)
              varying = 3:12, # The column indices where the attribute levels are stored
              sep = ".", # The separator after which the option disignation appears, _A and _B
              idx = list(c("idcase")), # The variable names for the indices, scenario and patient id
              choice = "depvar" # Variable name for the variable that contains which choices was made
              )
head(Heat)
```

Exercise 2: Fit a standard multinomial logistic regression with `ic` and `oc` the independent variables. Calculate the marginal rate of substitution for the installation cost against the operating cost. Determine if this model is a good fit for the data.

```
## Your turn
# Fit a simple linear regression
model.simple <- mlogit(depvar ~ ic + oc, data = Heat)

# Marginal rate of subsitution
model.simple$coefficients["ic"] / model.simple$coefficients["oc"]

# Summary to give goodness of fit
summary(model.simple)
# R2 is 0.014 so not great model fit.
```

Exercise 3: Include the four household specific characteristics in your model as option-specific slopes. Are any of these household specific characteristics important to the model fit?

```
## Your turn
model.all <- mlogit(depvar ~ ic + oc | income + rooms + agehed + region, data = Heat)
summary(model.all)
# All parameters are non-significant so we can exclude.
```

Exercise 4 Include a random slope for `ic` and `oc`. Are the variances well estimated? Is there evidence that a random slope is valid in this example?

```
## Your turn
model.random <- mlogit(depvar ~ ic + oc, data = Heat,
                       rpar = c(ic = "n", oc = "n"),
                       R = 100, panel = FALSE)
summary(model.random)
#
```