The dcm package

Oleh Komashko
Kyiv National Taras Shevchenko University
oleg.komashko@ukr.net

Riccardo (Jack) Lucchetti
Università Politecnica delle Marche
r.lucchetti@univpm.it

version 0.9

Abstract

McFadden’s choice models; includes, as special cases, multinomial and conditional logit models

Contents

1 The model
2 “Long” vs “wide” datasets
3 Examples
   3.1 A wide dataset
   3.2 A long dataset
4 The GUI interface
5 Prediction
6 Marginal effects
7 List of public functions (in alphabetical order)
8 Changelog

1 The model

We assume to have n individuals, indexed by i, who have to choose between K alternative goods, indexed by j (random utility model). The probability that individual i opts for alternative k is assumed to be:

\[ P \left[ y_i = k \right] = \frac{\exp(\mu_{ik})}{\sum_{j=1}^{K} \exp(\mu_{ij})} \]  
\[ \mu_{ij} = x_i' \beta_j + z_{ij}' \gamma \]  

where \( x_i \) is a vector with p elements containing individual-specific explanatory variables and \( z_{ij} \) is a vector with q elements containing explanatory variables describing the features of the j-th choice for individual i. For identification purposes,
one of the $\mu_{ij}$ indices is set to 1, and the corresponding integer $j$ is known as the "base". We will use the symbol $\theta$ to indicate the whole vector of model parameters $[\beta_1' \beta_2' \cdots \beta_K' \gamma']$. The number of elements in $\theta$ is $(K-1)p + q$.

Estimation is carried out by maximum likelihood via the Newton–Raphson algorithm and analytical derivatives.

## 2 “Long” vs “wide” datasets

Traditionally, data for these models can be organised in two alternative ways, that we call the "wide" and "long" (aka "stacked") format.

In the wide format, each row of the dataset is an individual and you have as many explanatory variables as parameters in the model. For example:

<table>
<thead>
<tr>
<th>id</th>
<th>choice</th>
<th>x</th>
<th>z1</th>
<th>z2</th>
<th>z3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0.3</td>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.4</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.5</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.6</td>
<td>7</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Here we have 4 individuals; individual 1 chose alternative 3 and so individual 2. Individual 3, instead, chose alternative 1 and so on. As for the explanatory variables, for example, $x_2 = 0.4$ while $z_{4,1} = 7$.

The same dataset in long format would appear as

<table>
<thead>
<tr>
<th>id</th>
<th>alt</th>
<th>choice</th>
<th>x</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0.3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0.3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0.4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0.4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0.4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0.5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0.6</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0.6</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0.6</td>
<td>3</td>
</tr>
</tbody>
</table>

Here you have as many $n \times K$ observations, for all the possible individual/choice combinations. A dummy variable ("choice" in this case) indicate what the actual choice made by individual $i$ was. For bookkeeping purposes, you also need a variable keeping track of individuals ("id" in this case).

The dcm package can handle both varieties, but a GUI hook is only available when the data are in the long format (see section [4]).

## 3 Examples

If the dependent variables has labels, they will be used automatically.
3.1 A wide dataset

Set up the model via `dcm_setup_wide` and estimate. Here, the $Z$ variables must be indicated via an array of lists; plus, you must also provide an array of labels for each of them.

The script below shows a minimal excerpt from the example script `sample-wide` that you can find in the `examples` directory; it uses the “tuna fish” dataset from [Hill et al. (2010)].

```plaintext
include dcm.gfn
open tunafish_small_wide.gdt --frompkg=dcm

### add value labels to dependent variable (optional) --------------
if 1
    alt_labels = defarray("Skist-Water", "Skist-Oil", "ChiSea-Water", "ChiSea-Oil")
    stringify(choice, alt_labels)
endif

### ----------------------------------------------------------------
list x = const income
list DISPLAY = display_br1 display_br2 display_br3 display_br4
list FEATURE = feature_br1 feature_br2 feature_br3 feature_br4
list PRICE = netprice_br1 netprice_br2 netprice_br3 netprice_br4
lists z = defarray(DISPLAY, FEATURE, PRICE)
znames = defarray("display", "feature", "price")

series id = time
bundle bun = dcm_setup_wide(choice, id, x, z, znames)
err = dcm_estimate(&bun)
dcm_printout(bun)
```

A few comments:

1. After loading the data, we endow the `choice` variable with value labels, so that output is nicer.
2. The list `x` contains the individual-specific variables $x_i$.
3. The choice-specific variables $z_{ij}$ are contained in an array of lists called `z`; we supplement it with a string array so that output will label coefficients appropriately.
4. We also need a series labelling individuals. Since we don’t have one, we just label individuals consecutively by using the built-in series `time`.
5. The function `dcm_setup_wide` is used to set up the model; note that we’re not setting a base category explicitly, so the default (1 = “Skist-Water”) applies. In the next line, `dcm_estimate` carries out the estimation. Finally, we see the printout.

The output should look as follows:

- **Dependent variable:** choice (wide data)
- **Number of alternatives:** 4
- **Base alternative:** Skist-Water
- **Standard errors based on Hessian**
<table>
<thead>
<tr>
<th></th>
<th>coefficient</th>
<th>std. error</th>
<th>z</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skist-Oil: const</td>
<td>0.360539</td>
<td>0.444698</td>
<td>0.8108</td>
<td>0.4175</td>
</tr>
<tr>
<td>Skist-Oil: income</td>
<td>-0.0588260</td>
<td>0.0195151</td>
<td>-3.014</td>
<td>0.0026 ***</td>
</tr>
<tr>
<td>ChiSea-Water: const</td>
<td>-2.97696</td>
<td>0.655090</td>
<td>-4.544</td>
<td>5.51e-06 ***</td>
</tr>
<tr>
<td>ChiSea-Water: income</td>
<td>0.0276990</td>
<td>0.0195518</td>
<td>1.427</td>
<td>0.1536</td>
</tr>
<tr>
<td>ChiSea-Oil: const</td>
<td>-0.658224</td>
<td>0.475315</td>
<td>-1.385</td>
<td>0.1661</td>
</tr>
<tr>
<td>ChiSea-Oil: income</td>
<td>-0.00964907</td>
<td>0.0156309</td>
<td>-0.6173</td>
<td>0.5370</td>
</tr>
<tr>
<td>display</td>
<td>2.51468</td>
<td>0.399712</td>
<td>3.8034</td>
<td>0.0003 ***</td>
</tr>
<tr>
<td>feature</td>
<td>1.17844</td>
<td>0.389712</td>
<td>3.024</td>
<td>0.0025 ***</td>
</tr>
<tr>
<td>price</td>
<td>-14.8965</td>
<td>2.92766</td>
<td>-5.088</td>
<td>3.62e-07 ***</td>
</tr>
</tbody>
</table>

Log-likelihood: -209.40043  
Schwarz criterion: 468.49401  
Akaike criterion: 436.80086  
Hannan-Quinn: 449.55643  
McFadden's R-squared = 0.3260  
Likelihood ratio test: Chi-square(6) = 202.553 [0.0000]  
(H0: model has only alt-specific constants)  
Correctly predicted cases: 170 out of 250 (68.00%)  

3.2 A long dataset

Set up the model via `dcm_setup_long` and estimate. Here, the Z variables are indicated as an ordinary list. Example, with the same data as in the previous subsection (but results are different, because we're using the full dataset here):

```plaintext
set verbose off
include dcm.gfn

## long data
open tunafish.gdt --frompkg=dcm

### add value labels to dependent variable (optional) ---------------
if 1
    alt_labels = defarray("Skist-Water", "Skist-Oil", "ChiSea-Water", "ChiSea-Oil")
    stringify(alt, alt_labels)
endif

### ----------------------------------------------------------------
list x = const income
dcmlist z = display feature netprice
znames = varnames(z)
bundle bun = dcm_setup_long(alt, choice, hhid, x, z)
err = dcm_estimate(&bun)
dcm_printout(bun)
```

Note that the only changes required from the “wide” dataset script are

1. The choice-specific variables are contained in an ordinary list, instead of an array of lists; therefore, there is no need for setting up their names as a string array, because the variable names will be used.
2. We set up the model bundle via the `dcm_setup_long` function, whose arguments are somewhat different from those for the equivalent wide-dataset script.
function dcm_setup_wide (see Section 7 for details). However, once the model bundle is created, the functions for estimation and printout are the same.

The output is as follows:

- **Dependent variable:** alt (stacked data)
- **Number of alternatives:** 4
- **Base alternative:** Skist-Water
- **Standard errors based on Hessian**

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Std. Error</th>
<th>z</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skist-Oil: const</td>
<td>-0.0672745</td>
<td>0.161142</td>
<td>-0.4175</td>
</tr>
<tr>
<td>Skist-Oil: income</td>
<td>-0.02163366</td>
<td>0.00600664</td>
<td>-3.602</td>
</tr>
<tr>
<td>ChiSea-Water: const</td>
<td>-0.457149</td>
<td>0.171134</td>
<td>-2.671</td>
</tr>
<tr>
<td>ChiSea-Water: income</td>
<td>-0.00289194</td>
<td>0.00652466</td>
<td>-0.4964</td>
</tr>
<tr>
<td>ChiSea-Oil: const</td>
<td>-1.11851</td>
<td>0.211979</td>
<td>-5.276</td>
</tr>
<tr>
<td>ChiSea-Oil: income</td>
<td>-0.0126826</td>
<td>0.00753377</td>
<td>-1.683</td>
</tr>
<tr>
<td>display</td>
<td>1.62011</td>
<td>0.243009</td>
<td>6.667</td>
</tr>
<tr>
<td>feature</td>
<td>1.33832</td>
<td>0.136747</td>
<td>9.787</td>
</tr>
<tr>
<td>netprice</td>
<td>-9.96844</td>
<td>0.864795</td>
<td>-11.53</td>
</tr>
</tbody>
</table>

- **Log-likelihood:** -1528.86183
- **Schwarz criterion:** 3123.53664
- **Akaike criterion:** 3075.72365
- **Hannan-Quinn:** 3093.53632
- **McFadden’s R-squared:** 0.2209
- **Likelihood ratio test:** Chi-square(6) = 866.958 [0.0000] (H0: model has only alt-specific constants)
- **Correctly predicted cases:** 844 out of 1499 (56.30%) 

4 The GUI interface

The model can be estimated via the GUI as well, **provided the dataset is in the long format**. The menu entry can be found under **Model >Limited dependent variable>Logit>Conditional Logit Model**.

Figure 1 replicates the same model used as an example in section 3.2.

5 Prediction

After estimation, the bundle contains a phat matrix with n rows and K columns, holding predicted values for the probabilities as defined in equation (1).

6 Marginal effects

Use dcm_meff; see the meff_sample script in the examples directory; also, the sample script fishing reproduces page 493 from Cameron and Trivedi (2005).

The function takes two mandatory arguments and an optional one: a string for the variable you want marginal effects for and the bundle containing the model. An optional Boolean third argument evaluates the marginal effects at the average, in which case an “estimation-like” format is displayed.
A matrix is returned (see section 7 for details). Note that the dcm_meff checks automatically if the regressor is a dummy variable and uses the appropriate formula accordingly.

7 List of public functions (in alphabetical order)

\texttt{dcm\_estimate (bundle *db)}

\textbf{Return type} : \texttt{scalar}

*db : a pointer to a model bundle, previously set up via \texttt{dcm\_setup\_long} or \texttt{dcm\_setup\_wide} (see below);

Performs estimation.

\texttt{dcm\_meff(string vname, bundle b, bool at\_average)}

\textbf{Return type} : \texttt{scalar}

vname : string, the name of the variable for which you want to compute marginal effects;

b : model bundle, as produced by \texttt{dcm\_estimate};

at\_average : Boolean, whether to compute marginal effects at the average of the regressors (default: no);
Computes marginal effects for the model contained in the bundle \( b \) relative to the explanatory variable referred to by the string \( \text{vname} \); it returns a matrix. The behaviour of this function depends on the value for the third parameter, which is an optional Boolean flag and defaults to "no".

In the default case, the function returns an \( n \times K \) matrix, containing the marginal effect of the variable on the probability that the \( i \)-th individual makes the \( k \)-th choice. Nothing is printed.

On the contrary, if the \( \text{at\_average} \) flag is set, then a \( K \times 2 \) matrix is returned, containing the marginal effects at the average and the corresponding standard errors, calculated via the delta method. These are also printed on output.

\[
\text{dcm\_print\_out} (\text{bundle } b, \text{ bool } \text{prnLR})
\]

**Return type**: none

\( b \): model bundle, as produced by \text{dcm\_estimate};

\( \text{prnLR} \): Boolean, whether to print the LR test (default: yes);

Prints out the model.

\[
\text{dcm\_setup\_long}(\text{series } y, \text{ series } \text{choice}, \text{ series } \text{id}, \text{ list } X, \text{ list } Z)
\]

**Return type**: bundle

\( y \): series, the variable holding the choices (must be discrete);

\( \text{choice} \): series, the dependent variable (must be binary);

\( \text{id} \): series, the variable holding the individuals ID;

\( X \): list, the \( x_i \) variables, where \( i \) is the unit contained in variable \( \text{id} \);

\( Z \): list, the \( z_{i,j} \) variables, where \( i \) is the unit contained in variable \( \text{id} \) and \( j \) is the choice in variable \( y \);

Sets up a \text{dcm} bundle from a long dataset.

\[
\text{dcm\_setup\_wide}(\text{series } y, \text{ series } \text{id}, \text{ list } X, \text{ lists } Z, \text{ strings } Z\text{names})
\]

**Return type**: bundle

\( y \): series, the variable holding the choices (must be discrete);

\( \text{id} \): series, the variable holding the individuals ID;

\( X \): list, the \( x_i \) variables, where \( i \) is the unit contained in variable \( \text{id} \);
$Z$: array of lists: the $j$-th element of the array contains the $z_{i,j}$ variables, where $i$ is the unit contained in variable $id$ and $j$ is the choice in variable $y$;

$Z\text{names}$: array of strings: one for each element of $Z$, is taken as the collective name for that group of variables.

Sets up a dcm bundle from a wide dataset.

8 Changelog

v 0.9 Initial release

References
