Creating efficient designs for discrete choice experiments

Arne Risa Hole
University of Sheffield

Nordic and Baltic Stata Users Group meeting
September 2016

Outline of presentation

- Example: choice of doctor's appointment
- Theory: efficient designs
- Generating efficient designs in Stata using `dcreate`

Example choice set in doctor's appointment DCE

I want you to imagine that you have developed some itchy, flaky patches on your hands. Occasionally they become quite red and sore. The problem does not appear to be spreading but has not responded to ointment recommended by the chemist. You decide to seek a medical opinion.

Choice 3 **If you were offered options A and B below which one would you choose?**

A

- You can be seen by the Dr on the **same day**
- You are offered **a choice** of appointment times
- This appointment would cost you **£28**
- The Dr is **warm and friendly**
- This Dr has access to your medical notes **but does not know you**
- This Dr's physical examination is **not very thorough**

B

- You can be seen by the Dr in **5 days time**
- You are **offered only one** appointment time
- This appointment would cost you **£8**
- The Dr is **formal and businesslike**
- This Dr has access to your medical notes **and knows you well**
- This Dr gives you a **thorough physical examination**

Please show your selection by ticking one box:

Choose A ☐

Choose B ☐

Attributes and levels in doctor's appointment DCE

Attribute	Levels
Number of days wait for an appointment	Same day, next day, 2 days, 5 days
Cost of appointment to patient	£0, £8, £18, £28
Flexibility of appointment times	One appointment offered Choice of appointment times offered
Doctor's interpersonal manner	Warm and friendly Formal and businesslike
Doctor's knowledge of the patient	The doctor has access to your medical notes and knows you well The doctor has access to your medical notes but does not know you
Thoroughness of physical examination	The doctor gives you a thorough examination The doctor's examination is not very thorough

Constructing the choice sets

- Once the attributes and levels have been determined these need to be combined into choice sets
- The number of possible combinations of attribute levels, called the **full factorial design**, is usually very large
- E.g. in the doctor's appointment DCE the full factorial design matrix has $4 \times 4 \times 2 \times 2 \times 2 \times 2 = 256$ rows
- These can be combined into $(256 \times 255)/2 = 32,640$ pairs
- This is clearly too many to be practically feasible
- Which pairs should we choose?

The conditional logit model

- The conditional logit probability that decision maker n chooses alternative j is

$$P_{nj} = \frac{\exp(\mathbf{x}'_{nj}\boldsymbol{\beta})}{\sum_{j=1}^{J} \exp(\mathbf{x}'_{nj}\boldsymbol{\beta})}$$

- The goal of the analysis is to estimate $\boldsymbol{\beta}$, which represents the weight the respondents give to the different attributes in the experiment
- The key question is how we can construct the choice sets in such a way so as to maximise the precision of the estimates of $\boldsymbol{\beta}$

The variance-covariance matrix

- The precision of the estimates is reflected by the variance-covariance matrix of the estimated coefficients
- In the case of the conditional logit model the variance-covariance matrix is given by:

$$\Omega = \left[\sum_{n=1}^{N} \sum_{j=1}^{J} \mathbf{z}'_{nj} P_{nj} \mathbf{z}_{nj}\right]^{-1}$$

  where

$$\mathbf{z}_{nj} = \mathbf{x}_{nj} - \sum_{j=1}^{J} \mathbf{x}_{nj} P_{nj}$$

- Note that this expression depends on the coefficients since P_{nj} is a function of $\boldsymbol{\beta}$

Efficient designs

- When we design an experiment we do not know $\boldsymbol{\beta}$ - if we did we would not have to conduct a DCE
- However, we can still calculate the variance-covariance matrix given a guess, or **prior**, value for $\boldsymbol{\beta}$
- The prior can be taken from e.g. a pilot study - if no prior is available 0 is often used
- Efficient designs are based on this simple idea: minimise the size of the the variance-covariance matrix given a prior for $\boldsymbol{\beta}$
- There are various ways of calculating the size of a matrix, which lead to different **efficiency measures**

D-efficiency

- The most commonly used efficiency measure is **D-efficiency**:

$$\left[|\Omega|^{1/K} \right]^{-1}$$

  where K is the number of parameters in the model
- The **D-error** is the inverse of the D-efficiency

$$|\Omega|^{1/K}$$

- The goal is to find a design that maximises D-efficieny or, equivalently, minimises the D-error
- Such a design is called a **D-efficient design**

Example: choice of doctor's appointment

- Let's say we want to construct a design for a simplified version of the doctor's appointment DCE
- Three attributes:
  - Waiting time (4 levels)
  - Flexibility of appointment times (2 levels)
  - Thoroughness of physical examination (2 levels)
- The model is:

$$U_{njt} = \beta_1 Wait_{njt} + \beta_2 Flex_{njt} + \beta_3 Thoro_{njt} + \varepsilon_{njt}$$

Constructing the choice sets

- In this case the full factorial design matrix has $4 \times 2 \times 2 = 16$ rows
- These can be combined into $(16 \times 15)/2 = 120$ pairs
- Still too many to present to a single respondent
- How many choice sets should we use?
- Minimum number: $K/(J-1)$
- In this example we therefore need a minimum of 3 choice sets - we choose 8

Full factorial design matrix

	wait	flex	thoro
1.	0	0	0
2.	0	0	1
3.	0	1	0
4.	0	1	1
5.	1	0	0
6.	1	0	1
7.	1	1	0
8.	1	1	1
9.	2	0	0
10.	2	0	1
11.	2	1	0
12.	2	1	1
13.	5	0	0
14.	5	0	1
15.	5	1	0
16.	5	1	1

Constructing the choice sets

- The simplest way of constructing the choice sets is combining rows from the full factorial matrix into pairs randomly

	choice_set	wait1	flex1	thoro1	wait2	flex2	thoro2
1.	1	2	0	0	5	0	0
2.	2	2	0	1	0	1	0
3.	3	5	0	0	5	0	1
4.	4	1	1	0	0	0	0
5.	5	5	1	0	1	0	0
6.	6	1	0	1	2	0	1
7.	7	2	1	1	1	1	1
8.	8	0	1	0	1	0	1

- Using the priors $\beta_1 = -0.13$, $\beta_2 = 0.2$ and $\beta_3 = 1$ the D-efficiency of this design is 1.321

Constructing D-efficient designs using dcreate

- The D-efficiency of a random design can be improved by systematically changing the levels in the alternatives using a search algorithm
- The Stata `dcreate` command uses the modified Fedorov algorithm (Cook and Nachtsheim, 1980; Zwerina et al., 1996; Carlsson and Martinsson, 2003)
- The rest of the presentation will focus on how to construct D-efficient designs using `dcreate`

Step 1: Generate full factorial design matrix

```
. matrix levmat = 4,2,2

. genfact, levels(levmat)

. list, separator(4)

     +--------------+
     | x1   x2   x3 |
     |--------------|
  1. |  1    1    1 |
  2. |  1    1    2 |
  3. |  1    2    1 |
  4. |  1    2    2 |
     |--------------|
  5. |  2    1    1 |
  6. |  2    1    2 |
  7. |  2    2    1 |
  8. |  2    2    2 |
     |--------------|
  9. |  3    1    1 |
 10. |  3    1    2 |
 11. |  3    2    1 |
 12. |  3    2    2 |
     |--------------|
 13. |  4    1    1 |
 14. |  4    1    2 |
 15. |  4    2    1 |
 16. |  4    2    2 |
     +--------------+
```

Step 2: Change variable names and recode levels

```
. rename x1 wait

. rename x2 flex

. rename x3 thoro

. recode wait (1=0) (2=1) (3=2) (4=5)

. recode flex (1=0) (2=1)

. recode thoro (1=0) (2=1)
```

Step 3: Run dcreate

```
. matrix b = -0.13,0.2,1

. dcreate c.wait i.flex i.thoro, nalt(2) nset(8) bmat(b)

The D-efficiency of the random starting design is:  1.5894558843

D-efficiency after iteration 1:     4.0359646736
Difference:                         2.4465087893

D-efficiency after iteration 2:     4.2422926121
Difference:                         0.2063279385

D-efficiency after iteration 3:     4.2422926121
Difference:                         0.0000000000

The algorithm has converged.
```

Step 3: Run dcreate

```
. list, separator(4) abbreviate(16)

     +------------------------------------------+
     | wait   flex   thoro   choice_set    alt |
     |------------------------------------------|
  1. |    5      0       1            1      1 |
  2. |    0      1       0            1      2 |
  3. |    0      0       0            2      1 |
  4. |    5      1       1            2      2 |
     |------------------------------------------|
  5. |    0      0       1            3      1 |
  6. |    5      1       0            3      2 |
  7. |    0      1       0            4      1 |
  8. |    5      0       1            4      2 |
     |------------------------------------------|
  9. |    5      0       0            5      1 |
 10. |    0      1       1            5      2 |
 11. |    5      1       1            6      1 |
 12. |    0      0       0            6      2 |
     |------------------------------------------|
 13. |    5      0       1            7      1 |
 14. |    0      1       0            7      2 |
 15. |    5      1       0            8      1 |
 16. |    0      0       1            8      2 |
     +------------------------------------------+
```

- We can instead treat *wait* as a categorical variable:

```
. matrix b = -0.13,-0.26,-0.65,0.2,1

. dcreate i.wait i.flex i.thoro, nalt(2) nset(8) bmat(b)

The D-efficiency of the random starting design is:  0.3539218239

D-efficiency after iteration 1:    0.8871050846
Difference:                        0.5331832607

D-efficiency after iteration 2:    0.8952664336
Difference:                        0.0081613490

D-efficiency after iteration 3:    0.8952664336
Difference:                        0.0000000000

The algorithm has converged.
```

- This forces the algorithm to include all levels of the attribute (not just the extremes)

```
. list, separator(4) abbreviate(16)

     +-----------------------------------------+
     | wait   flex   thoro   choice_set   alt |
     |-----------------------------------------|
  1. |    1      1       0            1     1 |
  2. |    0      0       1            1     2 |
  3. |    0      1       0            2     1 |
  4. |    5      0       1            2     2 |
     |-----------------------------------------|
  5. |    1      0       1            3     1 |
  6. |    5      1       1            3     2 |
  7. |    0      1       0            4     1 |
  8. |    2      0       1            4     2 |
     |-----------------------------------------|
  9. |    0      0       1            5     1 |
 10. |    2      1       0            5     2 |
 11. |    5      1       1            6     1 |
 12. |    1      0       0            6     2 |
     |-----------------------------------------|
 13. |    2      0       0            7     1 |
 14. |    1      1       1            7     2 |
 15. |    5      0       0            8     1 |
 16. |    2      1       1            8     2 |
     +-----------------------------------------+
```

Extensions

- Interaction effects can be specified using factor-variable syntax (e.g. `i.flex##i.thoro`)
- `dcreate` has options for including alternative-specific constants, opt-out alternatives etc.
- Uncertainty in the priors can be taken into accocunt by using Baysian designs (Sándor and Wedel, 2001)
- The `blockdes` and `evaldes` commands can be used to block or evaluate the efficiency of existing designs
- The `dcreate` help file describes these options/commands in more detail

References

- Carlsson F, Martinsson P. 2003. Design techniques for stated preference methods in health economics. Health Economics 12: 281-294.
- Cook RD, Nachtsheim CJ. 1980. A comparison of algorithms for constructing exact D-optimal designs. Technometrics 22: 315-324.
- Sándor Z, Wedel M. 2001. Designing conjoint choice experiments using managers' prior beliefs. Journal of Marketing Research 38: 430–444.
- Zwerina K, Huber J, Kuhfeld W. 1996. A general method for constructing efficient choice designs. Working Paper, Fuqua School of Business, Duke University