Extending Idefix package

Intermediate presentation

Daniel Gil Sanchez
KU Leuven
March 2019

0 Outline

1 Outline

1 What is *idefix* R package for?

- To create optimal designs for discrete choice experiments (DCEs) based on the multinomial logit model (MNL) and
- Individually adapted designs for the mixed multinomial logit model (MIXL).
- Available on CRAN (v 0.3.3).

Discrete choice experiments

DCEs are composed by:

- Nominal or Ordinal response variable
- Choice sets
- Alternatives within each choice set
- Attributes and levels

1 Multinomial Logit Model

- Choice design matrix $\mathbf{X} = [\mathbf{x}'_{js}]$, where $\mathbf{x}_{js}$ is a $k \times 1$ vector of attributes for profile j in choice set s.
- Respondent's utility is $u_{js} = \mathbf{x}'_{js}\boldsymbol{\beta} + \epsilon_{js}$, where $\boldsymbol{\beta}$ is a vector of parameters and ϵ_{js} is an i.i.d. extreme value error term.
- Probability a respondent chooses alternative j in choice set s is

$$p_{js} = \frac{e^{\mathbf{x}'_{js}\beta}}{\sum_{t=1}^{J} e^{\mathbf{x}'_{ts}\beta}}$$

- Information Matrix is

$$\mathbf{M}(\mathbf{X}, \boldsymbol{\beta}) = N \sum_{s=1}^{S} \mathbf{X}'_s(\mathbf{P}_s - \mathbf{p}_s\mathbf{p}'_s)\mathbf{X}_s$$

where $\mathbf{X}_s$ is the design matrix of choice set s,
$\mathbf{p}_s = [p_{1s}, \cdots, p_{Js}]$ and $\mathbf{P}_s = diag[p_{1s}, \cdots, p_{Js}]$ and N is the number of respondents.

1 D-optimality

- In OLS is defined as $D = |\mathbf{X}'\mathbf{X}|$
- In MNL is defined adopting the prior distribution of $\boldsymbol{\beta}$

$$D_B = \int_{\mathcal{R}^k} \left\{ det \left(\mathbf{M}^{-1}(\mathbf{X}, \boldsymbol{\beta}) \right) \right\}^{1/k} \pi(\boldsymbol{\beta}) d\boldsymbol{\beta}$$

Where k is the number of unknown parameters in the model and $\pi(\boldsymbol{\beta})$ is the prior distribution of $\boldsymbol{\beta}$. This criterion is also called Bayesian $D-$optimality criterion or just D_B.

1 Mixed Multinomial Logit model

- MNL models assume that the respondents have the same preferences, $\boldsymbol{\beta}$, for the attributes studied in the experiment.
- MIXL models assume that the individual preferences, $\boldsymbol{\beta}_n$, follow a certain distribution across respondents ($\boldsymbol{\beta}_n \sim f(\boldsymbol{\mu}_\beta, \boldsymbol{\sigma}_\beta)$).
  - Probability a respondent chooses alternative j in choice set s is

    $$p^*_{js} = \int p_{js}(\boldsymbol{\beta}) f(\boldsymbol{\beta}) d\beta$$

    Where $p_{js}(\boldsymbol{\beta})$ is defined as in the MNL model.

MIXL model assumes that respondents choose according to an MNL model, but each with different preferences.

1 Individually Adapted designs

The proper name of the methodology is *Individually adapted sequential Bayesian* design. It consists in two stages:

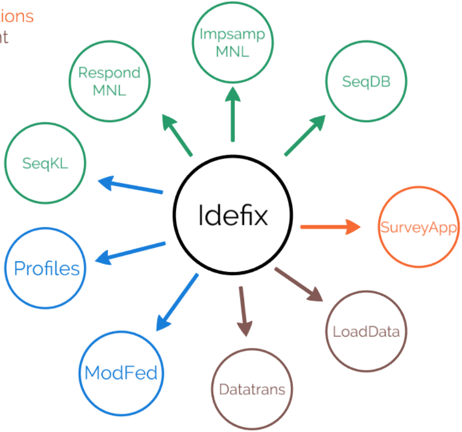
- **Initial static stage:** use a common initial prior distribution $\boldsymbol{\pi}(\beta)$ for all respondents. It is used to generate an initial design.
- **Adaptative sequential stage:** the prior information is updated sequentially after each response, and each choice set is constructed using the updated prior. Therefore, each respondent will have a different design.

Notes:

- Any criterion can be used to select a new choice set.
- Different algorithms can be used to select a new choice set. Here the Modified Fedorov Algorithm is used.

1 Current state of the package

- In Blue MNL functions
- In Green MIXL functions
- In Orange real survey functions
- In Brown Data management functions



KU LEUVEN

1 Objectives

1. Improve processing time of the **Modified Fedorov algorithm** by implementing some parts of the algorithm in C++.
2. Implement the **KL criterion** and compare it with the function that is already available in the package.
3. Implement the **Coordinate Exchange algorithm** to create optimal designs.
4. Make a **simulation study** to compare processing times and optimality of designs between the Modified Fedorov algorithm, the Coordinate Exchange algorithm and the use of DB and KL criteria.
5. Reorganize some functions inside the package, remove possible redundancy in code and implement parts of the code in C++.

2 Outline

2 Modified Fedorov Algorithm

- Point exchange algorithm
- Algorithm steps:
    1. Create a random initial design from candidate set.
    2. For each row of this design
        1. Exchange this row with each row from the candidate set. Resulting in N different designs.
        2. Compute optimality criterion for each modified design and choose the best value.
        3. update initial design and start again with the following row
    3. Repeat this process until no differences are found between the initial design and the final.

2 Modified Fedorov Algorithm

Example

- Design $3^3/2/8 \Rightarrow$ Design matrix 16×6 (dummy coding).
- Candidate set has $3^3 = 27$ rows/profiles
- In each iteration $16 \times 27 = 432$ information matrices and determinants are computed to find the best optimal design. Assume that only one iteration is needed.
- But, the information matrix needs draws from the prior of β. Assuming just 10 draws, the number of information matrices and determinants is 4320.
- But, different random initial designs to avoid local optima. Assuming 10 initial designs, the final number of information matrices and determinants is 43200.

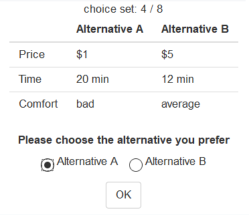
2 Processing time

First activity: Improve processing time in individually adapted designs.

`SeqDB` function selects the next DB-efficient choice set given parameter values and an initial design.

Example

Considering $3^3/2/8$ design:


choice set: 4 / 8

	Alternative A	Alternative B
Price	$1	$5
Time	20 min	12 min
Comfort	bad	average

Please choose the alternative you prefer

Alternative A Alternative B

OK

KU LEUVEN

2 Processing time

How to make it faster?

Using Hadley Wickham approach in his book *Advanced R*:

1. Find the biggest bottleneck (the slowest part of the code).
2. Try to eliminate it (you may not succeed but that is ok).
3. Repeat until your code is **fast enough**.

But, how to make it faster?

- Using faster functions in R and avoiding loops using vectorized functions.
- Implementing parts of the code in C++.

2 Processing time

Find the biggest bottleneck

- $4 \times 3 \times 2/2/8$ design.
- 10 draws from β distribution.
- Pre-defined initial design and alternatives chosen (1st stage of IASB approach).

2 Processing time

Profiling `SeqDB` function

Flame Graph | Data | Options ▾

Code	Memory (MB)		Time (ms)
▾ SeqDB	-156.3	159.2	3000
▾ apply	-156.3	159.2	3000
▾ FUN	-156.3	159.2	3000
▾ apply	-156.3	158.6	2990
▾ FUN	-143.9	157.5	2990
▸ InfoDes	-131.6	134.6	2630
▸ det	-12.3	16.1	290
rbind	0	1.8	30
▸ InfoDes	0	0	10

2 Processing time

Implementation in C++

- Use of Rcpp package
- Use of Rcpp Armadillo: C++ linear algebra library

2 Processing time

Implementation in C++

- Use of Rcpp package
- Use of Rcpp Armadillo: C++ linear algebra library

```
InfoDes <- function(par, des, n.alts) {
  group <- rep(seq(1, nrow(des) / n.alts, 1), each = n.alts)
  # probability
  u <- des %*% diag(par)
  u <- .rowSums(u, m = nrow(des), n = length(par))
  p <- exp(u) / rep(rowsum(exp(u), group), each = n.alts)
  # information matrix
  info.des <- crossprod(des * p, des) - crossprod(rowsum( des * p, group))
  return(info.des)
}
```

10 lines of code

2 Processing time

Implementation in C++

- Use of Rcpp package
- Use of Rcpp Armadillo: C++ linear algebra library

```r
InfoDes <- function(par, des, n.alts) {
  group <- rep(seq(1, nrow(des) / n.alts, 1), each = n.alts)
  # probability
  u <- des %*% diag(par)
  u <- .rowSums(u, m = nrow(des), n = length(par))
  p <- exp(u) / rep(rowsum(exp(u), group), each = n.alts)
  # information matrix
  info.des <- crossprod(des * p, des) - crossprod(rowsum( des * p, group))
  return(info.des)
}
```

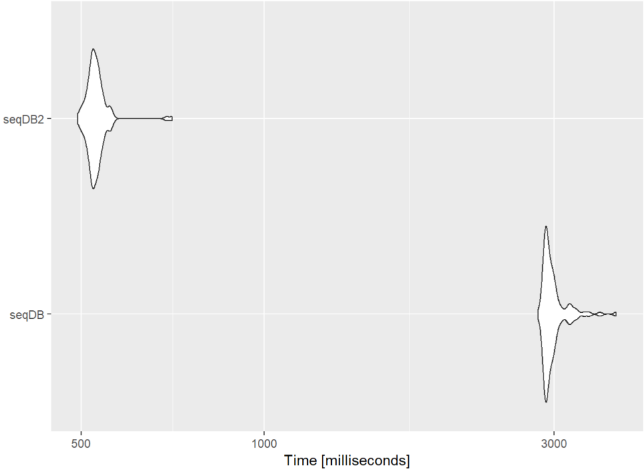
10 lines of code

```cpp
// [[Rcpp::depends(RcppArmadillo)]]

# include <RcppArmadillo.h>
using namespace Rcpp;

// [[Rcpp::export]]
NumericMatrix InfoDes_cpp(NumericVector par, NumericMatrix des,
                          double n_alts) {
  int i = 0;
  NumericVector group(des.nrow());
```

117 lines of code

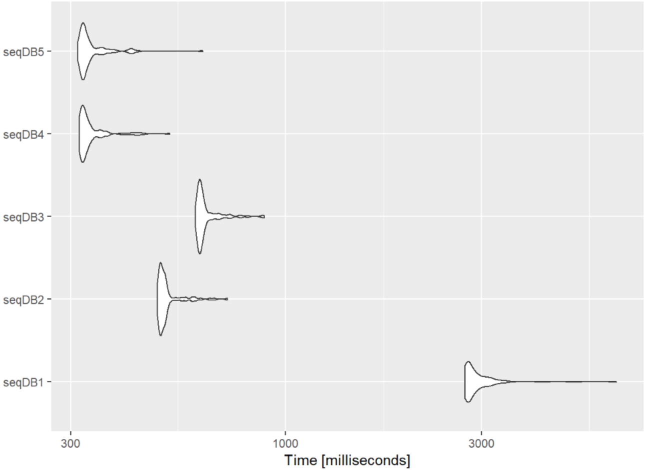
2 Processing time

Result: Implementation in C++ is almost 6x faster.



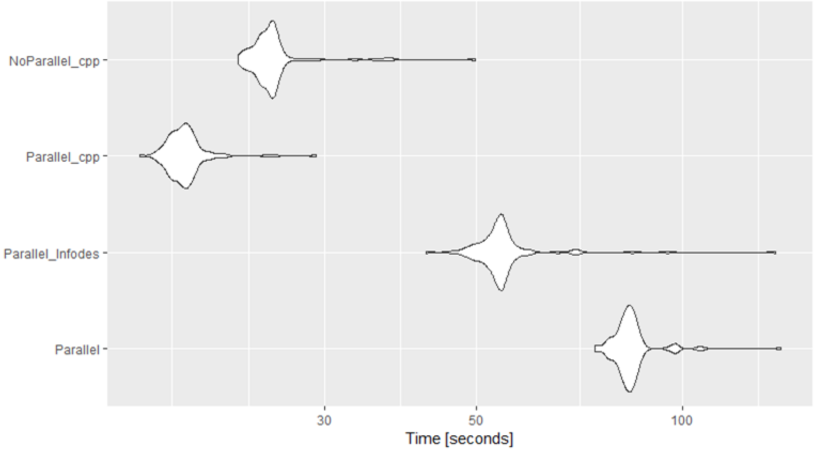
2 Processing time

Find next bottleneck and improve it (multiple times)



2 Processing time

What about parallel computing?



KU LEUVEN

2 Processing time

Second activity: Improve processing time in MNL designs

- The hardest work had already been done
  - Find bottlenecks
  - Improve functions in C++
- `ModFed` processing time was also improved by using the same functions as in `SeqDB`.

2 Processing time

Second activity: Improve processing time in MNL designs

- The hardest work had already been done
  - Find bottlenecks
  - Improve functions in C++
- `ModFed` processing time was also improved by using the same functions as in `SeqDB`.

What was needed to complete this task with success?

- Understand how Modified Fedorov algorithm works.
- Learn version control (Git and GitHub).
- Learn how C++ and C++ armadillo work in R.
- Learn about benchmarking and parallel computing.

KU LEUVEN

3 Outline

3 Kullback-Leibler criterion

- It was developed under individually adapted designs for the MIXL.
- It is an alternative to $D-$optimal criterion.
- It is faster to compute and it provides equally efficient designs.
- It is based on the Kullback-Leibler information:

$$KL(f,g) = \int f(x) log \frac{f(x)}{g(x)} dx$$

Where f and g are continuous densities of X.

  - KL is non-negative or zero $(f(x) = g(x))$
  - KL increases as the densities become more divergent
  - KL is not symmetric, $KL(f,g) \neq KL(g,f)$

3 Kullback-Leibler criterion

Implementation in DCEs

- To select next choice set, maximize the KL between the current posterior of β and the updated posterior one can obtain with the additional response from the next choice set.
- Since there are multiple alternatives, the expectation over all possible choices is maximized.

$$KLP = \sum_{j=1}^{J} \pi(y_{jsn}|\mathbf{y}_n^{s-1}) KL \left[f(\boldsymbol{\beta})_n|\mathbf{y}_n^{s-1}), f(\boldsymbol{\beta})_n|\mathbf{y}_n^{s-1}, y_{jsn}) \right]$$

Where s is the next choice set, n is a particular respondent and j is the chosen alternative. The densities $f(\boldsymbol{\beta})_n|\mathbf{y}_n^{s-1})$ and $f(\boldsymbol{\beta})_n|\mathbf{y}_n^{s-1}, y_{jsn})$ are the updated posteriors and $\pi(y_{jsn}|\mathbf{y}_n^{s-1})$ is the posterior weighted choice probabilities for the alternatives in the choice set s, given the previous responses.

3 Kullback-Leibler criterion

Implementation in the package

- Applying KL definition, KLP can be written as

$$KLP = \sum_{j=1}^{J} \pi(y_{jsn}|\mathbf{y}_n^{s-1})[log\ \pi(y_{jsn}|\mathbf{y}_n^{s-1}) -$$

$$\int log\ p_{jsn}(\boldsymbol{\beta}_n) f(\boldsymbol{\beta}_n|\mathbf{y}_n^{s-1} d\boldsymbol{\beta}_n)]$$

- Modified Fedorov algorithm is used, but instead of using $D-$optimality criterion KLP is used.
- Simulations in R are not consistent with results obtained in the paper that proposed the criterion.

3 Kullback-Leibler criterion

Third activity: Check why *SeqKL* function is not working

- Check code of simulations done in the paper that proposed the criterion.
  - Made in SAS. Proc IML.
  - 490 lines of code. No comments, no indentation.
- Check code of implementation in R
- Comparison of results from each function in both implementations.
- List differences.
- Discussion.

4 Outline

4 Coordinate Exchange Algorithm

- Coordinate exchange algorithm
  - Point exchange algorithm: Compute optimality criterion $\prod_{j=1}^{J} l_j$ times for each row.
  - Here: Compute optimality criterion $\sum_{j=1}^{J} l_j$ times for each row.
- Algorithm steps:
  1. Create a random initial design
  2. For each row of this design
     1. Take the first attribute in the row, evaluate the optimality criterion over all the levels of that attribute.
     2. If the optimality criterion of any of these levels is better than the current, then it is replaced.
     3. Repeat with the remaining attributes in the row.
  3. Repeat this process until no differences are found between the initial design and the final.

4 Coordinate Exchange Algorithm

Example

- Design $3^3/2/8 \Rightarrow$ Design matrix 16×6 (dummy coding).
- No Candidate set is needed.
- In each iteration $16 \times 9 = 144$ information matrices and determinants are computed to find the best optimal design. Assume that only one iteration is needed.
- But, the information matrix needs draws from the prior of β. Assuming just 10 draws, the number of information matrices and determinants is 1440.
- But, different random initial designs to avoid local optima. Assuming 10 initial designs, the final number of information matrices and determinants is 14400.

As a reminder, in Modified Fedorov the final number was 43200*.*

4 Coordinate Exchange Algorithm

Fourth activity: Implement the Coordinate Exchange algorithm

1. Implementation with only categorical factors/attributes.
2. Implementation with continuous attributes.
3. Implementation with both categorical and continuous.
4. Improve processing time, if possible (parallel computing, C++).

Note:

$D-$optimality criterion is going to be used, so the implementation of the information matrix in C++ is also going to be used here.

KU LEUVEN

5 Outline

1. Introduction
2. Modified Fedorov Algorithm
3. Kullback-Leibler criterion
4. Coordinate Exchange Algorithm
5. Simulation study

5 Simulation study

- The idea is to compare the processing time of Modified Fedorov algorithm and the Coordinate Exchange algorithm.
  - Determine the scenarios where one outperforms the other.
  - Determine in which situations parallel computing is needed.
- Compare efficiency of designs found with $D-$optimality criterion and KL criterion.
  - Determine the scenarios where one outperforms the other.

5 Objectives

1. Improve processing time of the **Modified Fedorov algorithm** by implementing some parts of the algorithm in C++. ✓
2. Implement the **KL criterion** and compare it with the function that is already available in the package. ✓
3. Implement the **Coordinate Exchange algorithm** to create optimal designs.
4. Make a **simulation study** to compare processing times and optimality of designs between the Modified Fedorov algorithm, the Coordinate Exchange algorithm and the use of DB and KL criteria.
5. Reorganize some functions inside the package, remove possible redundancy in code and implement parts of the code in C++. ✓

5 Optimality criteria

- To obtain precise estimates of $\boldsymbol{\beta}$
  - $D-$optimality: minimize the determinant of the variance-covariance matrix of $\boldsymbol{\beta}$
  - $A-$optimality: minimize the trace of the variance-covariance matrix of $\boldsymbol{\beta}$
- To obtain precise response predictions
  - $G-$optimality: minimize the maximum prediction variance
  - $V-$optimality: minimize the average prediction variance

Note:

These criteria are based on the information matrix, which depends on the unknown values in $\boldsymbol{\beta}$ through the probabilities p_{js}. Therefore, a Bayesian strategy that integrates the design criteria over a prior parameter distribution $\boldsymbol{\pi}(\boldsymbol{\beta})$ is adopted. Usually, the prior is a multivariate normal distribution.