KU LEUVEN

Extending Idefix package

Intermediate presentation

Daniel Gil Sanchez
KU Leuven
March 2019

0 Outline

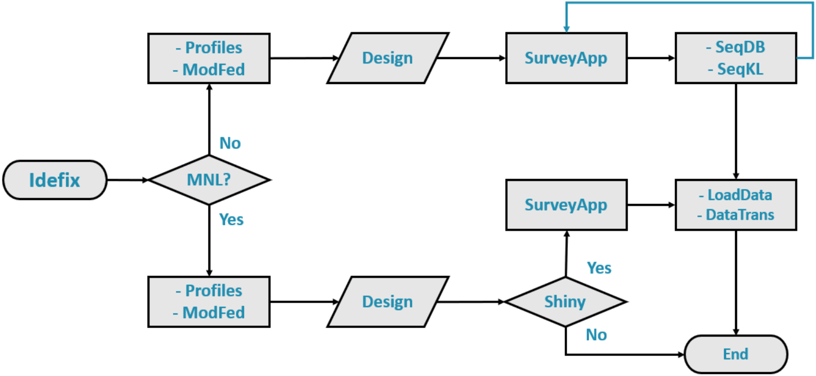
1 Outline

KU LEUVEN

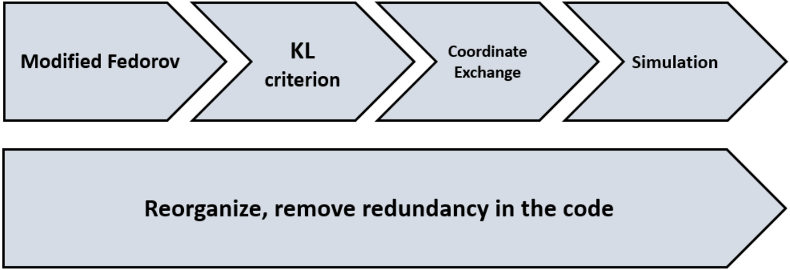
1 What is *idefix* R package for?

- 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).

1 Current state of the package



1 Objectives



1 Multinomial Logit Model

- 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

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

1 D-optimality

- To obtain precise estimates of $\boldsymbol{\beta}$
- In OLS, minimize $D = |M^{-1}(\mathbf{X})| = |(\mathbf{X}'\mathbf{X})^{-1}|$
- 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

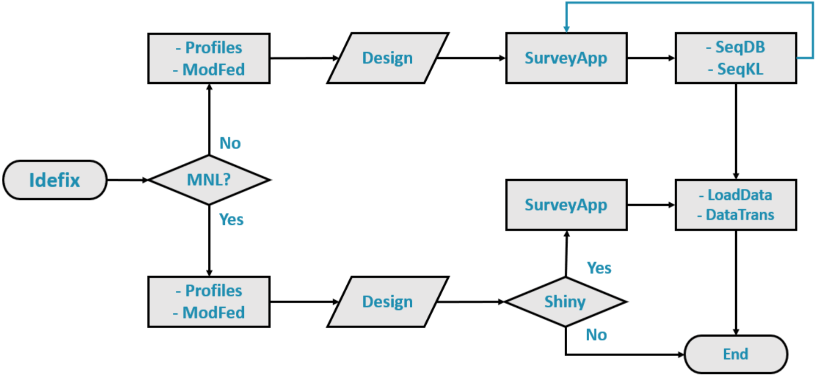
- MNL models assume that the respondents have the same preferences, $\boldsymbol{\beta}$, for the attributes studied in the experiment.
- MIXL models assume that the individual preferences, $\boldsymbol{\beta}_n$, follow a certain distribution across respondents ($\boldsymbol{\beta}_n \sim f(\boldsymbol{\mu}_\beta,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

Individually Adapted designs

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

- Initial static design
- Adaptative sequential design

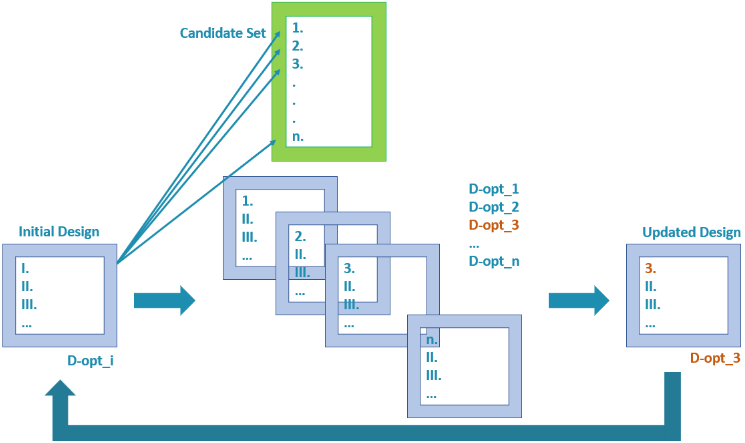
1 Current state of the package



2 Outline

2 Modified Fedorov Algorithm

Point exchange algorithm



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

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**.

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)
}
```

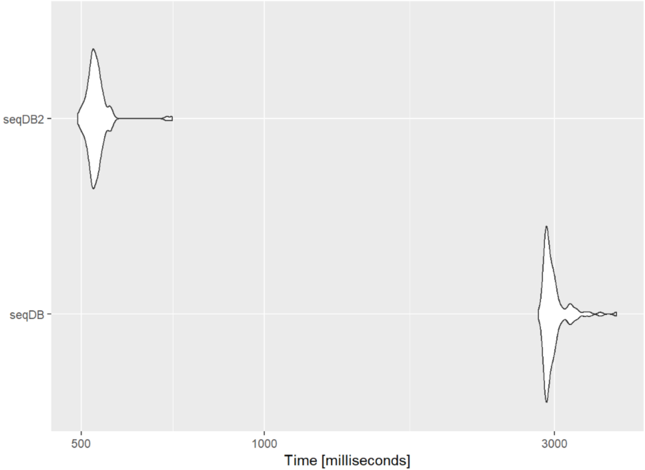
10 lines of code

```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

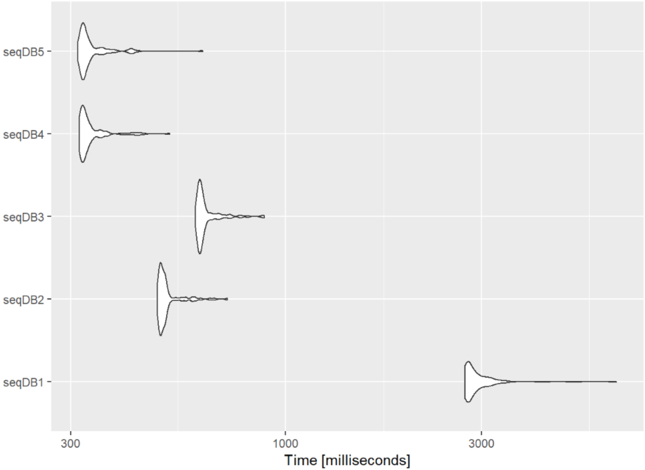
2 Processing time

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



2 Processing time

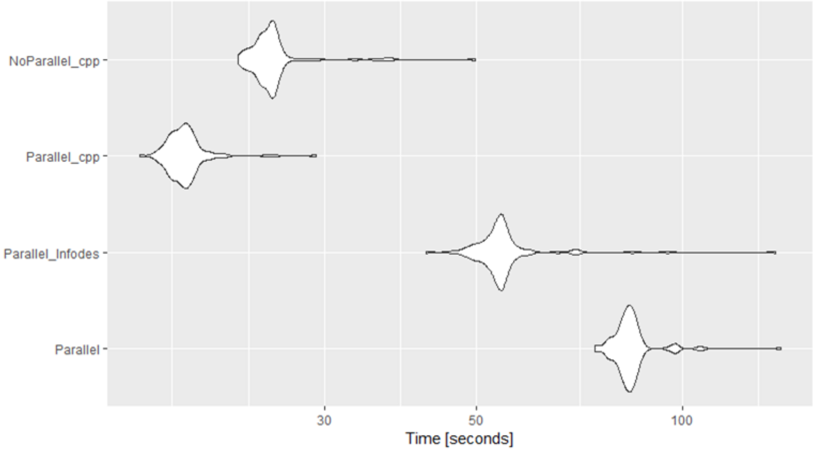
Find next bottleneck and improve it (multiple times)



KU LEUVEN

2 Processing time

What about parallel computing?

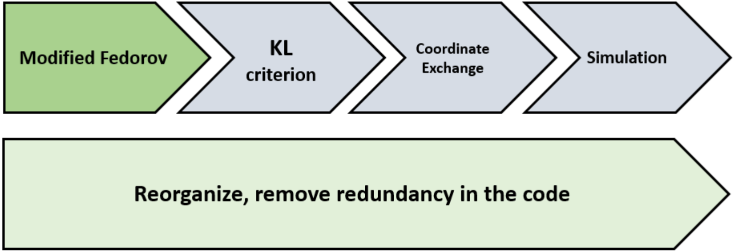


2 Processing time

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 Progress



Modified Fedorov
KL criterion
Coordinate Exchange
Simulation
Reorganize, remove redundancy in the code

3 Outline

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

KU LEUVEN

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.

3 Kullback-Leibler criterion

Implementation in DCEs

- 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.

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

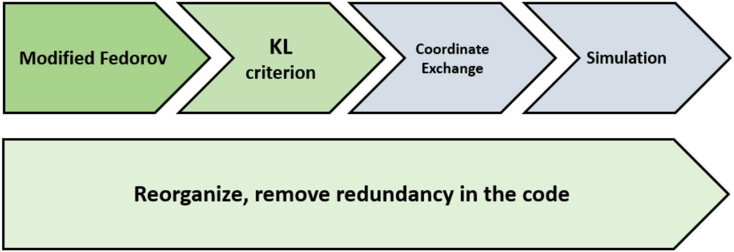
- Modified Fedorov algorithm.
- Simulations in R are not consistent with results obtained in the paper that proposed the criterion.

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.

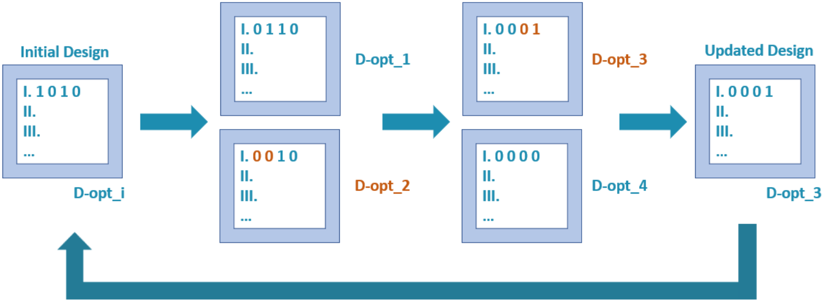
3 Progress


Modified Fedorov
KL
criterion
Coordinate
Exchange
Simulation
Reorganize, remove redundancy in the code

4 Outline

4 Coordinate Exchange Algorithm

- Compute optimality criterion $\sum_{j=1}^{J} l_j$ times for each row.
- Point exchange algorithm: Compute optimality criterion $\prod_{j=1}^{J} l_j$ times for each row.



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.

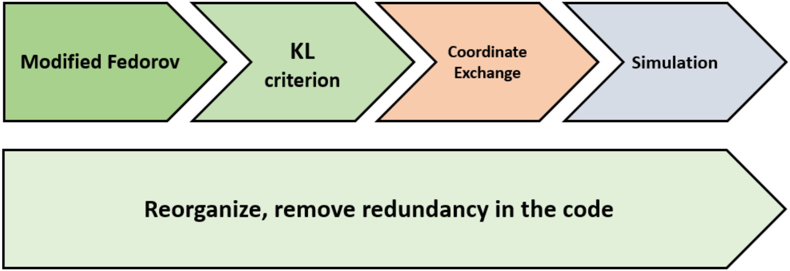
KU LEUVEN

5 Outline

5 Simulation study

- 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



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.

5 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, D_B is computed $16 \times 27 = 432$ times.
- Assuming just 10 draws, D_B is computed 4320.
- Assuming 10 initial designs, the final number of D_B computed is 43200.

5 Coordinate Exchange Algorithm

Example

- Design $3^3/2/8 \Rightarrow$ Design matrix 16×6 (dummy coding).
- No Candidate set is needed.
- In each iteration, D_B is computed $16 \times 9 = 144$ times.
- Assuming just 10 draws, D_B is computed is 1440.
- Assuming 10 initial designs, the final number of D_B computed is 14400.

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

KU LEUVEN

5 Parallel computing

1. Create set of copies of `R` running in parallel
2. Send data required to those copies
3. Split the task in chunks and send each chunk to each copy
4. Wait for all copies to complete their tasks
5. Combine results
6. Close and delete these copies

This procedure is used to compute the D_B, where for each draw from the prior a correspondent $D-$optimality value is computed.

KU LEUVEN

5 KL criterion

List of differences

- A minus sign missing in a function. ✓
- Number of degrees of freedom for the multivariate t distribution (importance distribution). ✓
- Difference in the density of the multivariate t distribution.
- Generation of lattice points.

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++. ✓