The SightX Research Technology Platform: Choice-Based Conjoint Experiment

By
Gian Ruzzi

sightx

Contents

Chapter 1

Experiment generation

1.1 Introduction

A conjoint experiment is a statistical analysis technique designed to uncover the value people assign to different aspects of a product. It is especially useful when you are in the process of developing new products or services, and you are uncertain about which features matter most or least to consumers. The insights gained from a conjoint experiment provide valuable guidance during the creation of new products and services, ensuring they align with consumer preferences and expectations.

The initial step of conducting your conjoint experiment is to break down your product or service into distinctive features that may influence consumer preferences; these features are called "attributes." For each attribute, you then specify its respective variations, referred to as "levels." This is better explained with an example: Imagine we're in the business of crafting sophisticated men's dress shoes, and we're on a mission to create new styles that truly resonate with our customers. To achieve this, we're going to use a conjoint experiment to understand exactly what features our customers are looking for in a shoe. From experience we think that a shoe can be described by its style, material, color, and price. These will be our attributes. Based on what we can create at our factory, we define the possible levels of each attribute as follows:

Style	Material	Color	Price
Oxford	Leather	Black	$160
Monk strap	Faux leather	Brown	$170
Derby	Canvas	Chestnut	$180

For our conjoint experiment we need to generate a set of alternatives from a set of N_a attributes, where the kth attribute has ℓ_k levels (a total of $\sum_{k=1}^{N_a} \ell_k = N_L$ levels). Each alternative uses exactly one level from each attribute, with these alternatives we construct choice sets and present each choice set to respondents and ask them to choose their preferred alternative from each set.

1.2 Definitions

To make it simpler to define our methods we need some definitions [1]:

Stated choice experiment: this is a set of choice sets, each choice set consists of two or more alternatives, each respondent is shown each choice set and chooses one of the alternatives. It is

also called *discrete choice experiment* or *stated preference experiment*.

Choice set size: number of alternatives in a choice set.

Attributes and levels: alternatives are described by attributes, each with two or more levels. For example, we could construct car alternatives with three attributes: *brand*, *colour*, and *type*. Brand has levels *Chevrolet* and *Toyota*, colour has levels *black* and *white*, and type has levels *sedan* and *hatchback*. Each alternative can have at most one level from each attribute. Each alternative must have plausible levels which are varied over the range of relevance.

Attribute	Brand	Colour	Type
Level 1	Chevrolet	Black	Sedan
Level 2	Toyota	White	Hatchback

Complete factorial design: is a design that has at least one of every possible level combination. When all attributes have the same number of levels, then the design is said to be *symmetric*, and *asymmetric* otherwise.

Main effect: individual effect of each attribute on the response.

Interaction effect: when the effect of one of the attributes on the response depends on the level of other attribute.

Fractional factorial design: a design in which only a subset of the possible level combinations appears.

Orthogonal main effects plan (OMEP): this is an $N_c \times N_a$ array, where elements in column k can take as values any of the levels in attribute k, such that for any pair of columns k and q, the number of times that the ordered pair (x,y) appears in the columns is equal to $n_{xk}n_{yq}/N_c$ where n_{xk} is the number of times level x appears in column k. For example the following array is an OMEP:

A	D	G
A	E	H
A	F	I
B	D	I
B	E	G
B	F	H
C	D	H
C	E	I
C	F	G

as for every pair of columns the possible pairwise combinations between the elements of those columns appear an equal number of times. For example if we take columns 1 and 3, we see that the pairs (A,G), (A,H), (A,I), (B,G), (B,H), (B,I), (C,G), (C,H), (C,I) all appear $3 \times 3/9 = 1$ time. The main effects can be estimated from an OMEP. Similarly, a Near Orthogonal Main Effects Plan

(NOMEP) is a fractional factorial design that is as close to orthogonal as possible.

Utility: "the net benefit derived from taking some action" [2]. In a choice experiment we assume that the subjects assign a utility to each alternative in a choice set and then chooses the one with the highest utility.

Part worth: is the "value" the subjects give to an attribute level. Hence, the "utility" of an alternative is the sum of the part worths of the levels that make up the alternative.

Dominant alternative: an alternative that will be preferred by all respondents, for example a car with the cheapest price and with the best features. It is best to avoid choice sets where one option will be dominant to all respondents. It is also best to avoid choice sets where one option will be dominated by the others.

We will use these terms throughout this document, as such it is important to understand them. Now let's look at desirable properties of choice designs:

1. A manageable number of choice sets
2. Equal replication of levels of each attribute across all the choice sets.
3. Levels within each attribute should appear an equal number of times, this is known as level balance. For example if we have an attribute with 2 levels and one with 3 levels and we have 6 alternatives, then each level in attribute one should appear 3 times, and each level in attribute two should appear 2 times.
4. All combinations of levels of an attribute should appear equally often over all the choice sets.
5. Avoidance of any predictable pattern in the choice sets.
6. It is also desirable to have as many combinations of levels of an attribute as possible appearing in the choice experiment.
7. If possible, alternatives in a choice set should have nearly equal utilities. That is, we want to avoid choice sets with dominant or dominated alternatives. For this we need some *a priori* knowledge of respondent preferences.

1.3 Construction of alternatives

Ideally we want to construct OMEPs, but sometimes this is not possible. In order to construct an OMEP we need the total number of alternatives, N_c, to be a common multiple of the number of levels per attribute and of the number of level combinations per attribute combination. For example, let's say we have four attributes, the first one with 5 levels, the second and third attributes with 3 levels, and the fourth attribute with 2. Thus, the attribute combinations have the following number of level combinations:

1. Between Attribute 1 and Attribute 2 or 3: 15 level combinations.

2. Between Attribute 1 and Attribute 4: 10 level combinations.

3. Between attributes 2, 3: 9 level combinations.

4. Between Attribute 4 and Attribute 2 or 3: 6 level combinations.

Hence, we need N_c to be a common multiple of [2,3,5,6,9,10,15] in order to construct an OMEP. The least common multiplier for these is 90; for a respondent this could be too many alternatives to handle. Hence, it would be better to construct a NOMEP with less alternatives. We want our design to be level balanced, as such N_c should be a multiple of the number of levels. We could choose $N_c = 30$, as 30 is a multiple of [2, 3, 5], and then we could construct 10 choice sets each with 3 alternatives.

To construct the OMEPs or NOMEPs I will use the algorithm presented by Xu [3]. The author defines the J_2 optimality criterion, where a design is optimal under J_2 if it minimizes J_2. For a $N_c \times N_a$ matrix $\boldsymbol{X} = [x_{i,k}]$, where column k has ℓ_k levels, we let

$$\delta_{i,j}(\boldsymbol{X}) = \sum_{k=1}^{N_a} \omega_k \delta(x_{i,k}, x_{j,k}) \qquad i, j = 1, 2, \ldots, N_c \tag{1.1}$$

where $\delta(x_{i,k}, x_{j,k}) = 1$ if $x_{i,k} = x_{j,k}$, and 0 otherwise, and $\omega_k > 0$ is a weight assigned to column k. If we set $\omega_k = 1$, then $\delta_{i,j}(\boldsymbol{X})$ counts the number of level coincidences between rows i and j. For example, if we have the following matrix

$$\boldsymbol{X} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

using $\omega_k = 1$, then δ between rows 1 and 3 is $\delta_{1,3}(\boldsymbol{X}) = 2$, because for column 1 both elements are 1, and for column 3 both elements are 0, thus we have two level coincidences. With this we can define the optimality criterion

$$J_2(\boldsymbol{X}) = \sum_{i=1}^{N_c - 1} \sum_{j=i+1}^{N_c} \left[\delta_{i,j}(\boldsymbol{X})\right]^2. \tag{1.2}$$

For a $N_c \times N_a$ matrix $\boldsymbol{X}$ where the kth column has ℓ_k levels, the lower bound of $J_2(\boldsymbol{X})$ is [3]

$$J_2(\boldsymbol{X}) \geq L(N_a) = \frac{1}{2}\left[\left(\sum_{k=1}^{N_a} \frac{N_c \omega_k}{\ell_k}\right)^2 + \sum_{k=1}^{N_a} (\ell_k - 1)\left(\frac{N_c \omega_k}{\ell_k}\right)^2 - N_c \left(\sum_{k=1}^{N_a} \omega_k\right)^2\right] \tag{1.3}$$

and $J_2(\boldsymbol{X}) = L(N_a)$ if $\boldsymbol{X}$ is orthogonal (OMEP)[3]. An OMEP is J_2 optimal for any choice of weights, while the J_2 optimality depends on the choice of the weights. If the weight of column k is equal to the number of levels of column k, $\omega_k = \ell_k$, which is referred as *natural weights*, and if $\boldsymbol{X}$ is level balanced, then [3]

$$J_2(\boldsymbol{X}) = N_c^2 A_2(\boldsymbol{X}) + \frac{N_c}{2}\left[N_c N_a (N_a - 1) + N_a \sum_{k=1}^{N_a} \ell_k - \left(\sum_{k=1}^{N_a} \ell_k\right)^2\right] \tag{1.4}$$

where $A_2(\boldsymbol{X})$ is an optimality criterion introduced by Eccleston and Hedayat [4], where $A_2(\boldsymbol{X}) = 0$ if $\boldsymbol{X}$ is orthogonal, as such it is a good optimality criterion for NOMEPS. Thus, using natural weights is a good choice to construct our designs.

If we have a matrix $\boldsymbol{X}$ and we want to add a column $\boldsymbol{c}$ with ℓ_k levels and weight ω_k, then for the resultant matrix $\boldsymbol{X}_+$

$$\delta_{i,j}(\boldsymbol{X}_+) = \delta_{i,j}(\boldsymbol{X}) + \delta_{i,j}(\boldsymbol{c}) \tag{1.5}$$

Now, if we switch two elements of column $\boldsymbol{c}$ that are different, $c_a \neq c_b$, with each other, then $J_2(\boldsymbol{X_+})$ is changed by $-2\Delta(a,b)$, where

$$\Delta(a,b) = \sum_{1 \leq j \neq a,b \leq N_c} [\delta(\boldsymbol{X})_{a,j} - \delta(\boldsymbol{X})_{b,j}][\delta(\boldsymbol{c})_{a,j} - \delta(\boldsymbol{c})_{b,j}]. \tag{1.6}$$

and $\delta_{i,j}(\boldsymbol{c})$ is updated by switching $\{\delta_{a,j}(\boldsymbol{c}) \leftrightarrow \delta_{b,j}(\boldsymbol{c})\}$ and $\{\delta_{j,a}(\boldsymbol{c}) \leftrightarrow \delta_{j,b}(\boldsymbol{c})\}$ for all $j \neq a,b$.

1.3.1 Example

Let's do an example to show what we have defined so far. Let's say we have 2 attributes each with $\ell_k = 2$ and $\omega_k = 1$. If we construct matrix $\boldsymbol{X}$,

$$\boldsymbol{X} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

then we calculate $\boldsymbol{\delta}(\boldsymbol{X}) = [\delta(\boldsymbol{X})_{i,j}]$ following equation 1.1,

$$\boldsymbol{\delta}(\boldsymbol{X}) = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix},$$

and using equation 1.2,

$$J_2(\boldsymbol{X}) = 1^2 + 1^2 + 0^2 + 0^2 + 1^2 + 1^2 = 4.$$

The lower bound $L(2)$ with $\ell_k = 2$, $\omega_k = 1$, and $N_c = 4$ is calculated following equation 1.3,

$$L(2) = 4$$

As $J_2(\boldsymbol{X}) = L(2)$, $\boldsymbol{X}$ is an OMEP. If we want to add column $\boldsymbol{c}$ with $\ell_k = 2$ and $\omega_k = 1$,

$$\boldsymbol{c} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

we calculate $\boldsymbol{\delta}(\boldsymbol{c})$

$$\boldsymbol{\delta}(\boldsymbol{c}) = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

following equation 1.5 we calculate $\boldsymbol{\delta}(\boldsymbol{X}_+)$ for our augmented matrix

$$\boldsymbol{\delta}(\boldsymbol{X}_+) = \boldsymbol{\delta}(\boldsymbol{X}) + \boldsymbol{\delta}(\boldsymbol{c}) = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 2 & 1 & 0 \\ 2 & 3 & 0 & 1 \\ 1 & 0 & 3 & 2 \\ 0 & 1 & 2 & 3 \end{pmatrix}$$

and following equation 1.2

$$J_2(\boldsymbol{X}_+) = 2^2 + 1^2 + 0^2 + 0^2 + 1^2 + 2^2 = 10.$$

The lower bound $L(3)$ is calculated following equation 1.3

$$L(3) = 6,$$

since $J_2(\boldsymbol{X}_+) > L(3)$, then $\boldsymbol{X}_+$ is not an OMEP. Let's see if we can reduce J_2 by switching elements in $\boldsymbol{c}$; using equation 1.6 we find that

$$\Delta(1,3) = 2,$$

thus if we switch the first and third elements of column $\boldsymbol{c}$ with each other then we will reduce $J_2(\boldsymbol{X}_+)$ by 4. The new column $\boldsymbol{c}$ with the first and third elements switched is

$$\boldsymbol{c} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

The augmented matrix will be

$$\boldsymbol{X}_+ = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix},$$

and

$$J_2(\boldsymbol{X}_+) = 10 - 2\Delta(1,3) = 6.$$

As we switched rows 1 and 3, then we have to switch elements $\{\delta_{a=1,2}(\boldsymbol{c}) \leftrightarrow \delta_{b=3,2}(\boldsymbol{c})\}$, $\{\delta_{a=1,4}(\boldsymbol{c}) \leftrightarrow \delta_{b=3,4}(\boldsymbol{c})\}$, $\{\delta_{2,a=1}(\boldsymbol{c}) \leftrightarrow \delta_{2,b=3}(\boldsymbol{c})\}$ and $\{\delta_{4,a=1}(\boldsymbol{c}) \leftrightarrow \delta_{4,b=3}(\boldsymbol{c})\}$.
The updated $\boldsymbol{\delta}(\boldsymbol{c})$ is

$$\boldsymbol{\delta}(\boldsymbol{c}) = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

and

$$\boldsymbol{\delta}(\boldsymbol{X}_+) = \boldsymbol{\delta}(\boldsymbol{X}) + \boldsymbol{\delta}(\boldsymbol{c}) = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 1 & 1 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 3 \end{pmatrix}$$

Since $J_2(\boldsymbol{X}_+) = L(3)$, then $\boldsymbol{X}_+$ is an OMEP, and we can see that by switching elements in our new column we were able to get an orthogonal matrix.

1.3.2 Non-level-balanced columns

If the number rows, N_c is not divisible by the number of levels of an attribute k, ℓ_k, then the column corresponding to attribute k will not be level balanced. In this case we want to get a column that is as level balanced as possible, to do this we will use Integer Linear Programming to find the number of times each level appears in the column. We define the variable x_l as the number of times level l appears in our current column. In Operations research language we need to define our Objective function (what we want to maximize/minimize), and our constraints. For ℓ_k levels we have

$$\text{Minimize} \quad shift \tag{1.7}$$

subject to

$$\sum_{l=0}^{\ell_k} x_l = N_c \tag{1.8}$$

$$x_l \leq \left\lfloor \frac{N_c}{\ell_k} \right\rfloor + shift \qquad \forall\, l = 0,1,\ldots,\ell_k - 1 \tag{1.9}$$

$$x_l \geq \left\lfloor \frac{N_c}{\ell_k} \right\rfloor \qquad \forall\, l = 0,1,\ldots,\ell_k - 1 \tag{1.10}$$

where $\lfloor\rfloor$ is the floor function. Here we want to minimize the difference between the number of times each level appears. We will call the set of the number of times each level appears in column k as $partition_k$, $partition_k = \{x_0, x_1, \ldots, x_{\ell_k - 1}\}$. These solutions can be found using Integer Linear Programming Libraries. For example, in python we can use the *ortools.sat.python* library, see "https://developers.google.com/optimization/cp/cp_solver". If N_c is divisible by ℓ_k, then

$$partition_{k,l} = \frac{N_c}{\ell_k} \qquad \forall\, l = 0,1,\ldots,\ell_k - 1 \tag{1.11}$$

1.3.3 The algorithm

Having defined equations 1.1, 1.2, 1.3, 1.5 and 1.6 we can detail the algorithm presented by Xu [3]. For N_c alternatives with N_a attributes, where the kth attribute has ℓ_k levels and weight ω_k:

1. Calculate the lower bounds $L(k)$ for $k = 1,2,\ldots,N_a$ using equation 1.3.
2. Create an initial design $\boldsymbol{X}$ with two columns and N_c rows, these columns have the following structure

$$\boldsymbol{X}_1 = \begin{pmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & \ell_2 - 1 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & \ell_2 - 1 \\ \ell_1 - 1 & 0 \\ \vdots & \vdots \\ \ell_1 - 1 & \ell_2 - 1 \end{pmatrix}.$$

Calculate $\delta_{i,j}(\boldsymbol{X})$ using equation 1.1 and $J_2(\boldsymbol{X})$ using equation 1.2. If $J_2(\boldsymbol{X}) = L(2)$ let $n_0 = 2$, otherwise $n_0 = 0$.

3. For $k = 3, \ldots, N_a$:

   (a) Randomly generate a column as level balanced as possible, $\boldsymbol{c}$, with ℓ_k levels where each level appears as indicated by $partition_k$. Calculate $\delta_{i,j}(\boldsymbol{c})$ using equation 1.1. Calculate $J_2(\boldsymbol{X}_+)$ by

   $$J_2(\boldsymbol{X}_+) = \sum_{i=1}^{N_c-1} \sum_{j=i+1}^{N_c} \left[\delta_{i,j}(\boldsymbol{X}) + \delta_{i,j}(\boldsymbol{c})\right]^2,$$

   if $J_2(\boldsymbol{X}_+) = L(k)$ got to (d), else go to (b).

   (b) Using equation 1.6 find the largest $\Delta(a,b)$. If $\Delta(a,b) > 0$, exchange rows a and b in $\boldsymbol{c}$, reduce $J_2(\boldsymbol{X}_+)$ by $2\Delta(a,b)$, and update $\delta_{i,j}(\boldsymbol{c})$ by switching $\{\delta_{a,j}(\boldsymbol{c}) \leftrightarrow \delta_{b,j}(\boldsymbol{c})\}$ and $\{\delta_{j,a}(\boldsymbol{c}) \leftrightarrow \delta_{j,b}(\boldsymbol{c})\}$ for all $j \neq a, b$. If $J_2(\boldsymbol{X}_+) = L(k)$ go to (d), else repeat (b) until the largest $\Delta(a,b) \leq 0$.

   (c) Repeat (a) and (b) T times and choose the column $\boldsymbol{c}$ that results in the smallest $J_2(\boldsymbol{X}_+)$. T is a constant defined by the user, Xu [3] recommends $T = 100$.

   (d) Append column $\boldsymbol{c}$ as the kth column of $\boldsymbol{X}$, let $J_2(\boldsymbol{X}) = J_2(\boldsymbol{X}_+)$, and update $\delta_{i,j}(\boldsymbol{X})$ using equation 1.5. If $J_2(\boldsymbol{X}) = L(2)$ let $n_0 = k$.

4. The final design will be a $N_c \times N_a$ matrix $\boldsymbol{X}$ where the first n_0 columns form an orthogonal array.

1.3.4 Algorithm remarks

1. In order to get a level balanced design it is necessary that N_c is a common multiple of the number of levels of the attributes. If not possible, we try to get an N_c that is divisible to as many attributes as possible.

2. In order to get a level balanced orthogonal design, N_c has to also be a common multiple of the number of level combinations of the attribute combinations, see beginning of section 2. Now, this is a necessary but not sufficient condition. For example, if we had four 2 level attributes, then there is no orthogonal array with 4 rows, even though 4 is a multiple of the number of levels and of the number of level combinations.

3. According to Xu [3], the algorithm works best if the attributes are ordered in descending number of levels. In the case where N_c is not a multiple of all the number of levels of the attributes, I recommend placing the attributes that are not divisible by N_c at the end.

4. In order to relate J_2 to the A_2 criterion the user should use natural weights, see equation 1.4.

5. Conjoint designs can also incorporate level pairing prohibitions, alternative inclusions and exclusions. These will be addressed in the next section.

1.4 Constraints

In conjoint experiments it is sometimes necessary to specify level combinations or alternatives that cannot be included in the design, and alternatives that must be included in the design. For this we will use equation 1.4 to find the A_2 criterion. For a level balanced design $\boldsymbol{X}$ with N_c rows and N_a columns using natural weights we have that

$$A_2(\boldsymbol{X}) = \frac{J_2(\boldsymbol{X})}{N_c^2} - \frac{1}{2N_c}\left[N_c N_a(N_a-1) + N_a\sum_{k=1}^{N_a}\ell_k - \left(\sum_{k=1}^{N_a}\ell_k\right)^2\right], \tag{1.12}$$

we will look for designs that minimize $A_2(\boldsymbol{X})$. If the design is not level balanced then we will look for designs that minimize $J_2(\boldsymbol{X})$. For clarity we will refer to the algorithm in section 1.3.3 as the *base algorithm*.

1.4.1 Alternative inclusions without excluded level combinations

Users can specify alternative that must be included in the experiment. For example, if we have attributes and levels

Attribute	Brand	Colour	Type
Level 1	Chevrolet	Black	Sedan
Level 2	Toyota	White	Hatchback
Level 3	Nissan	Red	Pickup

and we need to include the products {Toyota, White, Hatchback} and {Nissan, White, Sedan}. To have these inclusions in our design we need to do a few tweaks to our base algorithm. The idea is to place the must include alternatives in the first rows of our design. Hence, at step 2. of our algorithm, we generate the first column as indicated, but after this column has been created we move the levels involved in the alternative inclusions to the beginning of the column. For example, for our cars inclusions using index notation we need to include alternatives $\{1, 1, 1\}$ and $\{2, 1, 0\}$, we will put these alternatives in the first and second rows respectively. For $N_c = 9$ the first column "Brand" following the base algorithm is

$$\boldsymbol{c} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

now we will move level 1 and 2 to the top as these are the Brand levels in our product inclusions,

the first column becomes

$$\boldsymbol{c} = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \\ \hline 0 \\ 1 \\ 1 \\ 2 \\ 2 \end{pmatrix}$$

Then we don't generate the second column as indicated in the base algorithm but we jump to step 3 where we start at $k = 2$ instead of $k = 3$. In step (a) we generate a level balanced random column and move the levels in inclusions to the beginning. And in step (b) we calculate the largest $\Delta(a,b)$ given that a and b are not part of the first rows that have to do with level inclusions. The rest of the algorithm is not changed.

1.4.2 Excluded level combinations

Users can specify level combinations that cannot appear in the alternatives, this makes it impossible to construct an OMEP because level pairs will not have the same number of appearances. To construct a design with excluded level combinations we first identify the attributes that are involved in exclusions. We will call them *attributes with exclusions*. We will construct a design for the attributes without exclusions following the base algorithm. After we obtain this initial design we will follow a similar approach to the base algorithm were we add new columns successively, but now step 3(a) has to be adapted such that we can generate columns that do not violate the exclusions. To achieve this we will use Integer Linear Programming to generate a set of columns that accommodate our constraints. In addition, if we have alternatives that must be included we will follow the same approach as above where we will place the levels in inclusions in the first rows of the design.

Candidate column generation

Given a $N_c \times n$ array $\boldsymbol{X}$, where $n < N_a$, we want to add a new column $\boldsymbol{c}$ corresponding to the kth attribute with ℓ_k levels. This column has to be generated such that specified level combinations do not appear in $\boldsymbol{X}_+$ ($\boldsymbol{X}_+$ is array $\boldsymbol{X}$ with column $\boldsymbol{c}$ appended at the end). We start by identifying the rows that level $l = 0, 1, \ldots, \ell_k - 1$ cannot occupy. We identify these rows by looking at $\boldsymbol{X}$ and find the rows that when adding level l will cause a violation of the level exclusions. For example, let's say we have array

$$\boldsymbol{X} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 2 \\ 1 & 0 \\ 2 & 1 \\ 2 & 2 \end{pmatrix}$$

we have the restriction that level 2 of the second column cannot be paired with level 0 of the third column, and that the third column has 2 levels. As such we cannot place level 0 on the third and

sixth rows of the new column because level 2 occupies the third and sixth rows in the second column of $\boldsymbol{X}$.

We define variable $exclusions_l$ which is a set containing the row indices that level l cannot be placed at. For our example above

$$\begin{aligned} exclusions_0 &= \{3,6\} \\ exclusions_1 &= \{\} \end{aligned}$$

In the case of product inclusions we define the variable $inclusions_l$, which contains the row indices that level l must be placed at. Let's say we have one product inclusion which has $l = 1$ at column k, following our convention of placing must include alternatives at the beginning of the design we have to place level 1 in the first row as such

$$\begin{aligned} inclusions_0 &= \{\} \\ inclusions_1 &= \{1\} \end{aligned}$$

we also define variable $x_{i,l}$ where $i = 1,2,\ldots,N_c$ and $l = 0,1,\ldots,\ell_k - 1$ such that

$$x_{i,l} = \begin{cases} 1 & \text{if level } l \text{ is in row i} \\ 0 & \text{otherwise} \end{cases} \tag{1.13}$$

We also need to calculate the partition of the current column, $partition_k$, as defined in Section 1.3.2. With these we define our constraints to generate our candidate columns

$$\sum_{l=0}^{\ell_k} x_{i,l} = 1 \qquad \forall\, i = 1,2,\ldots,N_c \tag{1.14}$$

$$\sum_{i=1}^{N_c} x_{i,l} = partition_{k,l} \qquad \forall\, l = 0,1,\ldots,\ell_k - 1 \tag{1.15}$$

$$x_{i \in exclusions_l, l} = 0 \qquad \forall\, l = 0,1,\ldots,\ell_k - 1 \tag{1.16}$$

$$x_{i \in inclusions_l, l} = 1 \qquad \forall\, l = 0,1,\ldots,\ell_k - 1 \tag{1.17}$$

we find all feasible solutions up to T solutions (T is the number of times we repeat step 3 of the base algorithm). Our columns are constructed as

$$\boldsymbol{c} = \begin{pmatrix} \{l | x_{1,l} = 1\} = \{0,1,\ldots,\ell_k - 1\} \\ \vdots \\ \{l | x_{N_c,l} = 1\} = \{0,1,\ldots,\ell_k - 1\} \end{pmatrix}.$$

These solutions can be found using Integer Linear Programming Libraries. For example, in python we can use the *ortools.sat.python* library, see "https://developers.google.com/optimization/cp/cp_solver". These will be our candidate columns at k.

Algorithm with excluded level combinations

The algorithm to generate a design with excluded level combinations with or without "must include alternatives" is as follows:

1. With the attributes that are not involved in exclusions get a set of up to T_i designs $\{\boldsymbol{X}^1,\ldots,\boldsymbol{X}^{T_i}\}$, using the *base algorithm*. We let N_a^{ne} be the number of columns in these initial designs and N_a^e the number of attributes involved in exclusions, $N_a^{ne}+N_a^e=N_a$.

2. For the attributes involved in exclusions, get T_r random orderings of these attributes, T_r is a constant to be defined by the user. For example, if we have attributes "Color", "Size" and "Price", we could have: {Color, Price, Size}, {Size, Color, Price}, {Price, Size, Color}. For each random set we randomly select one of the designs obtained in step 1., and calculate the lower bounds $L(k)$ for $k=N_a^{ne}+1,\ldots,N_a^e$. Then:

   (a) For k = $1,2,\ldots,N_a^e$:

      i. Generate up to T feasible columns given constraints on attribute k as detailed in section 1.4.2. If no feasible columns can be generated, let $k=1$ and go to the next set generated in step 2.

      ii. Randomly choose one of the feasible columns, calculate $\delta_{i,j}(\boldsymbol{c})$ using equation 1.1. Calculate $J_2(\boldsymbol{X}_+)$ by

      $$J_2(\boldsymbol{X}_+)=\sum_{i=1}^{N_c-1}\sum_{j=i+1}^{N_c}\left[\delta_{i,j}(\boldsymbol{X})+\delta_{i,j}(\boldsymbol{c})\right]^2,$$

      if $J_2=L(N_a^{ne}+k)$ go to v. else go to iii.

      iii. If the number of feasible columns is less than T go to v., else using equation 1.6 find the largest $\Delta(a,b)$, such that the switch between rows a and b does not violate constraints, i.e. the switch should not generate excluded pairs, and rows involved in alternatives inclusions must not be considered. If $\Delta(a,b)>0$, exchange rows a and b in $\boldsymbol{c}$, reduce $J_2(\boldsymbol{X}_+)$ by $2\Delta(a,b)$, and update $\delta_{i,j}(\boldsymbol{c})$ by switching $\{\delta_{a,j}(\boldsymbol{c})\leftrightarrow\delta_{b,j}(\boldsymbol{c})\}$ and $\{\delta_{j,a}(\boldsymbol{c})\leftrightarrow\delta_{j,b}(\boldsymbol{c})\}$ for all $j\neq a,b$. If $J_2(\boldsymbol{X}_+)=L(N_a^{ne}+k)$ go to v., else repeat iii. until the largest $\Delta(a,b)\leq 0$.

      iv. Repeat ii. and iii. until all generated columns in step i. are exhausted and choose the column $\boldsymbol{c}$ that results in the smallest $J_2(\boldsymbol{X}_+)$. T is a constant defined by the user, Xu [3] recommends $T=100$.

      v. Append column $\boldsymbol{c}$ as the kth column of $\boldsymbol{X}$, let $J_2(\boldsymbol{X})=J_2(\boldsymbol{X}_+)$, and update $\delta_{i,j}(\boldsymbol{X})$ using equation 1.5. If $J_2(\boldsymbol{X})=L(N_a^{ne}+k)$ let $n_0=N_a^{ne}+k$.

   (b) Repeat step (a) until all T_r sets generated in step 2. are exhausted, for each of these designs calculate A_2 using equation 1.12. Choose the designs with the lowest A_2 (J_2 if design is not level balanced). Calculate the number of unique rows for these selected designs, and choose the designs with the largest number of unique products.

1.4.3 Alternative exclusions

In the case of having alternative exclusions, we will generate a set of designs following the algorithms defined above and reject the designs that have the excluded alternatives, this is possible due to the large amount of potential products a design can have. In the case we can't obtain a design without the excluded alternative we will get the design with the lowest A_2 (J_2 if design is not level balanced) and the highest number of unique alternatives and we will simply remove the alternatives that must be excluded.

1.5 Construction of choice sets

Once we have our alternatives, we need to group them in choice sets that will be shown to the respondents. What we want to achieve are choice sets with some level overlap (alternatives with some of their levels being the same) and utility balanced sets (i.e. alternatives within a choice set should have similar utilities). Utility balance is only possible if there is some *a priori* ordering to the levels. For example, usually cheaper prices have higher part worths than more expensive prices; therefore, we would prefer to gather cheaper products with other cheaper products, and expensive products with other expensive products. Finally, if we have repeated alternatives a choice set must not have any repeated alternatives. To construct the choice sets we will use simulated annealing, a concept borrowed from physics.

1.5.1 Simulated annealing

Annealing is a heat treatment where materials are heated up to a certain temperature and then cooled down, this treatment is used to change the physical properties of materials. Our simulation follows a similar approach, where we start at a given "temperature" and switch alternatives between choice sets such that we try to get our system to its minimum energy state. The probability of switching alternatives depends on how the energy changes, a switch that decreases the energy is favored over one than increases the energy. Temperature also affects the switching probability, at higher temperatures, switches are more frequent than at lower temperatures. The higher switching frequency reduces the probability of getting trapped in a local minimum, and as we decrease the temperature the system settles in a energy state.

We are going to treat our "stated choice experiment" as if it were a physical system. In our physical world systems tend to be at states that minimize their energy; therefore, we need to define the energy of our system such that the minimum energy state is an experiment with utility balanced choice sets that have the desired amount of level overlap. For N_c alternatives, choice set size N_{set}, and N_g choice sets, we define the energy of our system as

$$E = \sum_{i=1}^{N_g} E_i \tag{1.18}$$

where E_i is the energy of the ith choice set,

$$E_i = \frac{1}{N_{set}} \sum_{j=1}^{N_{set}} \left(\frac{\text{utility}_{i,j} - \text{avg-utility}_i}{\text{max-utility}} \right)^2 + \frac{1}{N_a} \sum_{k=1}^{N_a} \left(\frac{\text{overlap}_{i,k} - \text{opt-overlap}}{N_{set} - 1} \right)^2 \tag{1.19}$$

here $\text{utility}_{i,j}$ is the utility of the jth alternative in the ith choice set, avg-utility_i is the average utility of the ith choice set, max-utility is the maximum possible utility. $\text{overlap}_{i,k}$ is the number of times a level in attribute k is repeated in the ith choice set; for example, if the first choice set of an experiment is:

	Brand	**Colour**	**Type**
Alternative 1	Chevrolet	Black	Hatchback
Alternative 2	Toyota	Black	Hatchback
Alternative 3	Nissan	White	Hatchback

,

we can see that Black appears twice so for Colour we have a level that is repeated one time, Hatchback appears three times so for Type we have a level that is repeated two times, and for brand we do not have any repeated levels. Thus,

$$\begin{aligned} \text{overlap}_{1,1} &= 0 \\ \text{overlap}_{1,2} &= 1 \\ \text{overlap}_{1,3} &= 2 \end{aligned}$$

opt-overlap is the desired number of times we would like to see levels being repeated per attribute in a choice set. The first part of the energy is minimized when all alternatives in a choice set have the same utility. The second part of the energy is minimized when levels per attribute are repeated opt-overlap number of times.

The acceptance ratio of switching a pair alternatives between choice sets is given by

$$A(\mu \to \nu) = min\left(1, e^{-\frac{E(\nu)-E(\mu)}{T}}\right) \tag{1.20}$$

where μ is the system before the switch and $E(\mu)$ its corresponding energy, ν is the system after the switch and $E(\nu)$ its corresponding energy, and T is the temperature. Thus, for N_c alternatives with N_a attributes and choice set size N_{set}, the algorithm goes as follows

1. Define starting temperature T_i, temperature step ΔT, final temperature $T_f > 0.0$. Randomly assign alternatives to $N_g = N_c/N_{set}$ choice sets, making sure alternatives are not repeated within sets. Calculate the energy of each choice set following equation 1.19, and let $T = T_i$.

2. For $20N_c$ times:

   (a) Randomly choose two alternatives that are in different choice sets, $i \neq j$; making sure alternative in choice set j is not in choice set i, and alternative in choice set i is not in choice set j.

   (b) Calculate the energy of choice sets i and j as if the selected alternatives were switched, we will refer to these energies as $E(new)_i$ and $E(new)_j$; and to the energies before the switch as $E(old)_i$ and $E(old)_j$.

   (c) Calculate the change in energy given by the switch

   $$\Delta E = E(new) - E(old) = E(new)_i + E(new)_j - E(old)_i - E(old)_j$$

   (d) If $\Delta E < 0$ go to (e), else calculate $\omega = e^{-\Delta E/T}$. Generate ξ from a uniform distribution $[0,1)$, if $\xi \leq \omega$ go to (e), else go to (a).

   (e) Switch the randomly selected alternatives between groups, and set $E_i = E(new)_i$ and $E_j = E(new)_j$. Go to (a).

3. Subtract ΔT from T, if the new $T < T_f$ stop, else go to 2.

It is important to note that the resulting stated choice experiment will not always be the lowest energy state given the stochastic nature of the method, but it will have utility balanced choice sets that have the desired amount of level overlap.

1.5.2 *A priori* utilities

In order to generate utility balanced choice sets we need to have an idea of the utilities of each alternative. For a given alternative, the user might have an idea of which levels are more preferred than others; for example in the case of price, cheaper values tend to be preferred over more expensive values.

If the hierarchy of the levels of an attribute is not clear, all the levels' part worths for that attribute are set to zero. On the other hand, if there is a notion of the ordering of the levels within an attribute, we order the levels from "least preferred" to "most preferred", we assume that the part worths are equally spaced, we set the part worth of the "least preferred" level to -1 and the part worth of the "most preferred" level to 1. For example, if we had an attribute with five levels, then the part worths are set as

	Part worths
Level 1	-1.0
Level 2	-0.5
Level 3	0.0
Level 4	0.5
Level 5	1.0

The utilities of the alternatives are calculated by adding up the part worths of the levels making up the alternatives. The max-utility that we use in equation 1.19 is equal to the number of attributes that have ordered levels.

Chapter 2

Utility estimation

After we collect our responses for the choice sets as defined in Chapter 1, we need to obtain the respondents' part worths. To do this we will use a Hierarchical Bayes Multinomial Logistic Model (HB MNL). This model has two levels [5, 6, 7, 8]:

1. At the upper level we assume that the part worths are described by a multivariate normal distribution.

2. At the lower level we assume that the individual's probabilities of choosing an option, given the individual part worths, are described by a multinomial logistic model [2].

2.1 Effects coding

In order to estimate the respondents' part worths we first need to code our levels. We could use dummy coding where we describe each product with a vector of 1s and 0s, where each element represents a level; a 1 in the position of a level means that that level is included in that product, and a 0 otherwise. For example, if we have one attribute called "size", whose levels are {small, medium, large}, we will have a vector with three elements, where we can have the following products

	Small	Medium	Large
Product 1 (small)	1	0	0
Product 2 (medium)	0	1	0
Product 3 (large)	0	0	1

Table 2.1: Dummy coding of a three level attribute

The issue with this approach is that the variables are linearly dependent, i.e. the state of one of the attributes can be determined by the state of the other two. Following our example, if the product is not "small" neither "medium" then it must be "large". To solve this we will use effects coding where for each attribute we choose one level arbitrarily to be the reference level, and constraint the part worth of this level to be equal to the difference of the part worths of the other levels within their attribute [8]. For our example above, we choose Large to be the reference level, thus we have:

	Small	Medium
Product 1 (small)	1	0
Product 2 (medium)	0	1
Product 3 (large)	-1	-1

Table 2.2: Effects coding of a three level attribute

By using effects coding, the part worths within each attribute are zero centered. We will use this type of coding, and choose the reference level to be the last level of each attribute as defined by the user. Given this coding the number of parameters we have to find is equal to the total number of levels minus the number of attributes:

$$n_{ind} = \sum_{k=1}^{N_a} \ell_k - N_a \tag{2.1}$$

where N_a is the number of attributes and ℓ_k is the number of levels in the kth attribute.

2.1.1 Coding the *None* option

In some cases the user, if having the choice, will decide that they would not purchase any of displayed products, in this scenario we can have an alternative be "None of these". As such, how do we code this? The solution is to all other level columns to 0, add a column for None, and set it to 1. Following our size example, our products with the none alternative will look something like this

	Small	Medium	None
Product 1 (small)	1	0	0
Product 2 (medium)	0	1	0
Product 3 (large)	-1	-1	0
None of the others	0	0	1

Table 2.3: Effects coding of a three level attribute with the None alternative

Here we can see that the "None" attribute does not have a reference level, and it is simply a one level attribute. This way we can also determine the utility of the "None" alternative. Here, it is important to note that the only attribute that can have one level is the None attribute. We can thing of this as being the "None" product

2.2 Model

The individual part worths are independently and identically distributed from a multinomial normal distribution, $\mathcal{N}_{n_{ind}}(\boldsymbol{\gamma}, \boldsymbol{V})$. Where $\boldsymbol{\gamma}$ is a n_{ind} dimensional row vector of means of the respondents' part worths, $\boldsymbol{V}$ is a $(n_{ind} \times n_{ind})$ matrix of variances and covariances of the individuals' part worths. We define $\boldsymbol{\beta}_i$ as a n_{ind} dimensional row vector, where the elements are the part worths of the ith respondent, and for N_{res} respondents we let $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_{N_{res}}\}$ be the set containing all the individual part worths.

Following a multinomial logistic model, the probability of the ith individual choosing the qth product within the jth choice set given part worths $\boldsymbol{\beta}_i$ is defined as [2, 7, 8]

$$p_{i,j}(q|\boldsymbol{\beta}_i) = \frac{e^{\boldsymbol{\beta}_i \cdot \boldsymbol{x}_{i,j,q}}}{\sum_k e^{\boldsymbol{\beta}_i \cdot \boldsymbol{x}_{i,j,k}}} \tag{2.2}$$

where $\boldsymbol{x}_{i,j,k}$ is the effects coded kth product within the jth choice set shown to the ith respondent.

In order to estimate $\boldsymbol{\gamma}$, $\boldsymbol{V}$, and $\boldsymbol{\beta}$ we use an iterative process. We initialize $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}_i$ as zero vectors. We initialize the elements $v_{i,j}$ of $\boldsymbol{V}$ as:

$$v_{i,i} = \frac{\ell_k - 1}{\ell_k} \qquad \text{level i} \in \text{attribute k} \tag{2.3}$$

$$v_{i,j} = -\frac{1}{\ell_k} \qquad \text{if i} \neq \text{j and } \{\text{level i and level j}\} \in \text{attribute k} \tag{2.4}$$

$$v_{i,j} = 0 \qquad \text{if level i, level j belong to different attributes} \tag{2.5}$$

Each iteration consists of three steps:

1. Given the current estimates of $\boldsymbol{\beta}$, and $\boldsymbol{V}$ we generate a new $\boldsymbol{\gamma}$ by drawing randomly from a multivariate normal distribution, $\mathcal{N}_{n_{ind}}(mean(\boldsymbol{\beta}), \mathbf{V} \times 1/N_{res})$, with mean vector equal to the mean of $\boldsymbol{\beta}$ and variance covariance matrix equal to $\boldsymbol{V} \times 1/N_{res}$ [5, 6, 7].

2. Given the current estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ we generate a new $\boldsymbol{V}$ by drawing randomly from an inverse Wishart distribution, $\mathscr{W}^{-1}(df, \boldsymbol{S})$, with degrees of freedom $df = N_{res} + n_{ind}$, and scale matrix $\boldsymbol{S}$ [5, 6, 7] given by

$$\boldsymbol{S} = n_{ind} \times \boldsymbol{I} + \sum_{i=1}^{N_{res}} (\boldsymbol{\beta}_i - \boldsymbol{\gamma})^T (\boldsymbol{\beta}_i - \boldsymbol{\gamma}) \tag{2.6}$$

   where T stands for "transpose" and $\boldsymbol{I}$ is an identity matrix of order n_{ind}.

3. Given the current estimates of $\boldsymbol{\gamma}$ and $\boldsymbol{V}$ we generate $\boldsymbol{\beta}$ from a multivariate normal distribution which has variance covariance matrix proportional to $\boldsymbol{V}$, for this we use the Metropolis Hastings Algorithm described bellow [5, 6, 7].

We repeat these steps for a given number of iterations until equilibration is reached. We divide the simulation in three stages, an adaptive stage, an equilibration stage, and a measurement stage.

1. **Adaptive stage**: During this stage we run the simulation for 3000 iterations to estimate the proportionality factor used in the Metropolis Hastings Algorithm.

2. **Equilibration stage**: During this stage, we run the simulation for a few thousand iterations where we will use the likelihood ratio index, also known as the McFadden pseudo R^2, R^2_{Mc} [2] to determine equilibration. This value increases as the number of iterations grows, but after a while it stabilizes, i.e. it randomly oscillates around a mean value, we take this stabilization to indicate that the system reached equilibrium.

3. **Measurement stage**: After equilibration has been reached, we run the simulation for 10 000 steps, we accumulate the individual $\boldsymbol{\beta}_i$s skipping 5 steps between measurements and at the end we get the average of these $\boldsymbol{\beta}_i$s, the resulting averages of the $\boldsymbol{\beta}_i$s will be the estimates of the respondents' part worths.

2.2.1 Metropolis Hastings Algorithm

For each respondent we draw a random n_{ind} dimensional row vector $\boldsymbol{b}$ from a multivariate normal distribution $\mathcal{N}_{n_{ind}}(\mathbf{0}, \boldsymbol{V} \times f)$, with zero mean and variance covariance matrix equal to $\boldsymbol{V}$ times an scaling factor f. For the ith respondent, the new proposed part worths vector, $\boldsymbol{\beta}_i^n$, is equal to their last estimated part worths vector, $\boldsymbol{\beta}_i^o$, plus $\boldsymbol{b}$

$$\boldsymbol{\beta}_i^n = \boldsymbol{\beta}_i^o + \boldsymbol{b}. \tag{2.7}$$

The probability of accepting this new estimate is given by

$$p_i(o \to n) = \min(1, r_i), \tag{2.8}$$

r_i is the ratio

$$r_i = \frac{\mathrm{L}(\boldsymbol{\beta}_i^n)}{\mathrm{L}(\boldsymbol{\beta}_i^o)} \times \frac{\rho(\boldsymbol{\beta}_i^n; \boldsymbol{\gamma}, \boldsymbol{V})}{\rho(\boldsymbol{\beta}_i^o; \boldsymbol{\gamma}, \boldsymbol{V})}, \tag{2.9}$$

and $\mathrm{L}(\boldsymbol{\beta}_i^n)$ and $\mathrm{L}(\boldsymbol{\beta}_i^o)$ are the likelihoods of the ith respondent answers given $\boldsymbol{\beta}_i^n$ and $\boldsymbol{\beta}_i^o$ respectively. These likelihoods are calculated by multiplying the probabilities of the responses of the ith user

$$\mathrm{L}(\boldsymbol{\beta}_i) = \prod_{j=1}^{N_g} p_{i,j}(q_j | \boldsymbol{\beta}_i) \tag{2.10}$$

where N_g is the number of choice sets seen by the respondents, and q_j is the index of the product chosen by the respondent in the jth choice set.

$\rho(\boldsymbol{\beta}_i^n; \boldsymbol{\gamma}, \boldsymbol{V})$ and $\rho(\boldsymbol{\beta}_i^o; \boldsymbol{\gamma}, \boldsymbol{V})$ are the densities of the $\mathcal{N}_{n_{ind}}(\boldsymbol{\gamma}, \boldsymbol{V})$ distribution at points $\boldsymbol{\beta}_i^n$ and $\boldsymbol{\beta}_i^o$ respectively,

$$\rho(\boldsymbol{\beta}_i; \boldsymbol{\gamma}, \boldsymbol{V}) = \frac{1}{(2\pi)^{n_{ind}/2} \det(\boldsymbol{V})^{1/2}} e^{\left(-\frac{1}{2}(\boldsymbol{\beta}_i - \boldsymbol{\gamma})\boldsymbol{V}^{-1}(\boldsymbol{\beta}_i - \boldsymbol{\gamma})^T\right)} \tag{2.11}$$

with this definition we can write r as

$$r_i = \frac{\mathrm{L}(\boldsymbol{\beta}_i^n)}{\mathrm{L}(\boldsymbol{\beta}_i^o)} \times e^{\left(-\frac{1}{2}(\boldsymbol{\beta}_i^n - \boldsymbol{\gamma})\boldsymbol{V}^{-1}(\boldsymbol{\beta}_i^n - \boldsymbol{\gamma})^T + \frac{1}{2}(\boldsymbol{\beta}_i^o - \boldsymbol{\gamma})\boldsymbol{V}^{-1}(\boldsymbol{\beta}_i^o - \boldsymbol{\gamma})^T\right)} \tag{2.12}$$

if $r_i \geq 1$ it means that the posterior probability of $\boldsymbol{\beta}_i^n$ is larger than the posterior probability of $\boldsymbol{\beta}_i^o$ and we accept $\boldsymbol{\beta}_i^n$ as the next estimate of the part worths for the ith respondent. If r_i is less than one, we accept $\boldsymbol{\beta}_i^n$ with probability equal to r_i, to do this we generate a random number ξ in the range [0,1), if $\xi \leq r_i$ we accept the new estimate, otherwise we keep $\boldsymbol{\beta}_i^o$ as the next estimate.

Scaling factor f

The scaling factor f can be determined by the proportion of accepted $\boldsymbol{\beta}_i^n$s per iteration. According to [6], the optimal acceptance ratio is 0.44 for a one dimensional $\boldsymbol{\beta}_i$, and declines to about 0.23 for dimensions larger than 5. As such we will take the optimal acceptance ratio to be 0.23. As indicated above, we use the adaptive stage to determine the value of f that will be used in the equilibration and measurement stages [6].

In the adaptive stage we will start by arbitrarily setting $f = 0.25$, we let n_s be the number of times β_i^n was accepted after each iteration of the HB MNL algorithm, and we calculate the success ratio as

$$r_s = \frac{n_s}{N_{res}}. \tag{2.13}$$

If $r_s < 0.23$, then we reduce f by ten percent and continue to the next iteration, if $r_s > 0.23$ we increase f by ten percent and continue to the next iteration; we repeat this for 3000 iterations. These iterations will give us an estimate of f that we will use during the equilibration and measurement stages. In the equilibration and measurement stages we do not change the value of f.

Given that the value of f fluctuates quite rapidly, see for example Fig. 2.1, and takes some time to settle, we discard the first 1500 iterations, and keep the last 1500. We use the iterations that we kept and take the average, this average is the value that we will use as f in the next stages.

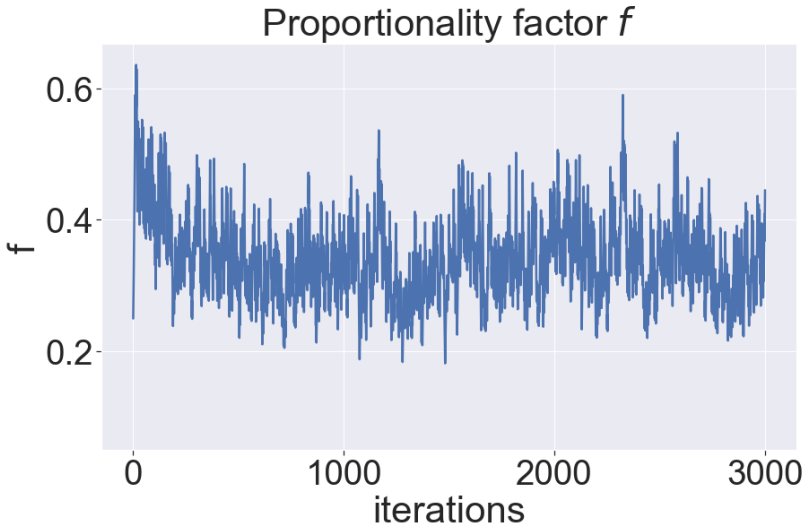


Figure 2.1: f factor for successive iterations of a conjoint experiment with 3 attributes and 3 levels each.

2.2.2 Likelihood ratio index

The likelihood ratio index R^2_{Mc} is defined as [2]

$$R^2_{Mc} = 1 - \frac{\ln(\mathrm{L}^t(\boldsymbol{\beta}))}{\ln(\mathrm{L}^t(\boldsymbol{\beta}_{null}))} \tag{2.14}$$

where $\mathrm{L}^t(\boldsymbol{\beta})$ is the value of the total likelihood using the estimates of the part worths, $\boldsymbol{\beta}$, and $\mathrm{L}^t(\boldsymbol{\beta}_{null})$ is the value of total the likelihood if we set all part worths equal to zero. The total likelihood is equal to the product of the individual likelihoods,

$$\mathrm{L}^t(\boldsymbol{\beta}) = \prod_{i=1}^{N_{res}} \mathrm{L}(\boldsymbol{\beta}_i), \tag{2.15}$$

$\mathrm{L}^t(\boldsymbol{\beta}_{null})$ corresponds to the case when all products have equal probability of being chosen. For example, if we had 5 choice sets, with 3 products per choice set the probability of choosing an individual product within a set is $1/3$, therefore the individual likelihoods will be equal to

$$\mathrm{L}(\mathbf{0}) = \prod_{i=1}^{5} \frac{1}{3} = \frac{1}{3^5} \tag{2.16}$$

as such the total likelihood is

$$\mathrm{L}^t(\boldsymbol{\beta}_{null}) = \prod_{i=1}^{N_{res}} \frac{1}{3^5} = \frac{1}{3^{5\times N_{res}}} \tag{2.17}$$

In general, for N_g choice sets, with N_{set} alternatives each, for N_{res} respondents,

$$\mathrm{L}^t(\boldsymbol{\beta}_{null}) = \frac{1}{N_{set}^{N_g \times N_{res}}} \tag{2.18}$$

so, we can write the likelihood ratio index as

$$R^2_{Mc} = 1 - \frac{\ln(\mathrm{L}^t(\boldsymbol{\beta}))}{\ln\left(\frac{1}{N_{set}^{N_g\times N_{res}}}\right)} = 1 - \frac{\ln(\mathrm{L}^t(\boldsymbol{\beta}))}{\ln(1) - \ln\left(N_{set}^{N_g\times N_{res}}\right)} = 1 + \frac{\ln(\mathrm{L}^t(\boldsymbol{\beta}))}{N_g \times N_{res}\ln(N_{set})} \tag{2.19}$$

If our model was no better than a completely random model where all products have the same probability of being chosen, then $\mathrm{L}^t(\boldsymbol{\beta}) = \mathrm{L}^t(\mathbf{0})$, and $R^2_{Mc} = 0$. This is the lowest value R^2_{Mc} can take [2]. Now, if our model was perfect, i.e. it could perfectly predict the choices of each respondent, then $\mathrm{L}^t(\boldsymbol{\beta}) = 1$ and $\ln(\mathrm{L}^t(\boldsymbol{\beta})) = 0$, therefore $R^2_{Mc} = 1$. This is the highest value R^2_{Mc} can take.

It is not possible to compare the R^2_{Mc} for two models that use different samples, or that use different choice sets for the sampled individuals. For example, we can not say that a model with $R^2_{Mc} = 0.8$ is better than a model with $R^2_{Mc} = 0.6$ that uses a different sample. But, we can compare the R^2_{Mc} of two models that use exactly the same sample, in that case we can say that the model with $R^2_{Mc} = 0.8$ fits the data better than the model with $R^2_{Mc} = 0.6$. This is important because in our Monte Carlo simulation, R^2_{Mc} tends to increase as the number of iterations grows until a certain point where it settles and oscillates around a mean value, see for example Fig. 2.3; i.e. after successive iterations our model fits the data better and better, until it settles. After this value settles we say that we have reached equilibrium and we can start taking measurements for $\boldsymbol{\beta}$. In the following section I define how we can determine equilibration.

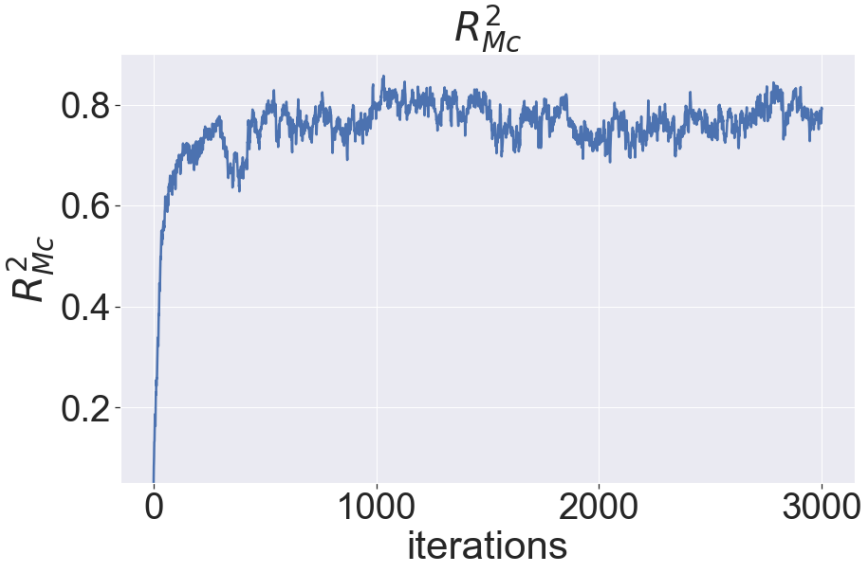


Figure 2.2: R^2_{Mc} for successive iterations of a conjoint experiment with 3 attributes and 3 levels each

2.2.3 Equilibration

There is no perfect way of determining convergence for a Monte Carlo simulation [9], here we are going to use the Geweke Diagnostic, and the moving average of R^2_{Mc}.

Geweke Diagnostic

Geweke recommended using a method based on spectral analysis to assess the convergence of a Markov Chain Monte Carlo process [9, 10]. For details on this method we encourage the reader to see [9, 10]. This method consists on comparing the means of the beginning and end of a Markov chain to detect if they are significantly different. The Geweke statistic is calculated by

$$Z_n = \frac{\overline{\theta}_A - \overline{\theta}_B}{\sqrt{\frac{S_A(0)}{n_A} + \frac{S_B(0)}{n_B}}} \tag{2.20}$$

where Z_n approaches the standard normal distribution as the number of points in the Markov chain approaches infinity [10]. As such, large values of Z_n indicate that the process has not converged, using a 1% confidence level, we say that the process has not converged if $|Z_n| > 2.6$.

Now let's look at the individual parts of Eqn. 2.20. As stated above, we are going to use R^2_{Mc} to assess convergence, as such a Markov chain will consist of N consecutive estimates of R^2_{Mc}, i.e. we save the value of R^2_{Mc} after every iteration for N iterations, and we call this chain $\boldsymbol{\theta}$. Now we split this chain in two parts, chain A, $\boldsymbol{\theta}_A$, and chain B, $\boldsymbol{\theta}_B$. $\boldsymbol{\theta}_A$ is made of the first $0.1N$ elements of $\boldsymbol{\theta}$, and $\boldsymbol{\theta}_B$ is made of the last $0.5N$ elements of $\boldsymbol{\theta}$. The length of $\boldsymbol{\theta}_A$ will be denominated as N_A and the length of $\boldsymbol{\theta}_B$ as N_B, as such

$$\overline{\theta}_A = \frac{\sum_{i=1}^{N_A} \theta_{A,i}}{N_A}$$

$$\overline{\theta}_B = \frac{\sum_{i=1}^{N_B} \theta_{B,i}}{N_B}$$

where $\theta_{A,i}$ and $\theta_{B,i}$ are the ith elements of $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_B$ respectively.

$S_A(0)$ and $S_B(0)$ are the spectral densities of the chains at zero frequency. To estimate the spectral densities we are going to use the approach defined by Heidelberger and Welch [11]. We start by defining

$$I(n,\boldsymbol{\theta}) = \frac{1}{N}\left[\left(\sum_{i=1}^{N}\theta_i \cos\left(\frac{2\pi(i-1)n}{N}\right)\right)^2 + \left(\sum_{i=1}^{N}\theta_i \sin\left(\frac{2\pi(i-1)n}{N}\right)\right)^2\right], \tag{2.21}$$

where N is the length of $\boldsymbol{\theta}$.

$$J(n,\boldsymbol{\theta}) = 0.270 + \ln\left(\frac{1}{2}\left[I(2n-1,\boldsymbol{\theta}) + I(2n,\boldsymbol{\theta})\right]\right), \tag{2.22}$$

$$f(n,N) = \frac{4n-1}{2N}. \tag{2.23}$$

Now, given a Markov chain $\boldsymbol{\theta}$ of length N we let $K = \lfloor N/4 \rfloor$ ($\lfloor\cdot\rfloor$ is the floor function) and:

1. Calculate $J(n,\boldsymbol{\theta})$ for $n = 1,2,\ldots,K$.
2. Calculate $f(n,N)$ for $n = 1,2,\ldots,K$.
3. Using least squares, we fit the polynomial

   $$g(f(n,N)) = a_0 + a_1 f(n,N) + a_2 f(n,N)^2 + a_3 f(n,N)^3 \tag{2.24}$$

   to $J(n,\boldsymbol{\theta})$.
4. Define matrix

   $$\boldsymbol{X} = \begin{bmatrix} f(1,N)^0 & f(1,N)^1 & f(1,N)^2 & f(1,N)^3 \\ \vdots & \vdots & \vdots & \\ f(K,N)^0 & f(K,N)^1 & f(K,N)^2 & f(K,N)^3 \end{bmatrix} \tag{2.25}$$

5. Calculate

   $$\boldsymbol{S} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1} \tag{2.26}$$

6. We let

   $$\sigma^2 = 0.645 \times S_{1,1} \tag{2.27}$$

   where $S_{1,1}$ is the upper most left element of $\boldsymbol{S}$.
7. Finally, our estimate of the spectral density at zero frequency for $\boldsymbol{\theta}$ is

   $$S(0) = e^{-\sigma^2/2 + a_0} \tag{2.28}$$

Exponential moving average

Given the high frequency oscillations of R^2_{Mc} as time passes, we need a way of smoothing out these fluctuations. Here we calculate a series of averages at different subsets of the full time series in order to smooth out short-term fluctuations. The moving average will allow us to recognize long term patterns in our data in order to check if the R^2_{Mc} has settled to an stable value.

We calculate the exponential moving average at the ith iteration using formula

$$average_i = 0.99 \times average_{i-1} + 0.01 \times R^2_{Mc} \tag{2.29}$$

where R^2_{Mc} is the likelihood ratio index at the i iteration. The average is initialized by taking the the value of R^2_{Mc} at $i = 0$. For an example see the following figure where the moving average is plotted the R^2_{Mc} for successive iterations

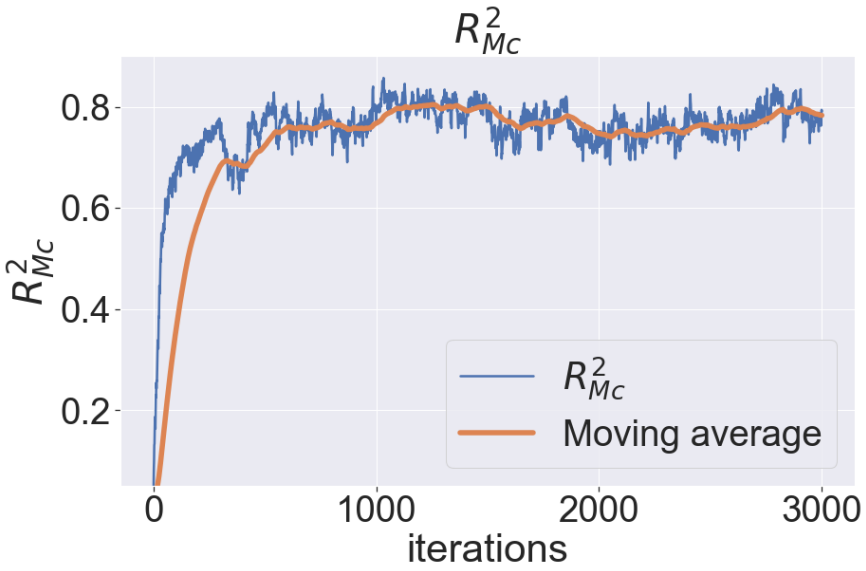


Figure 2.3: R^2_{Mc} and its moving average for successive iterations of a conjoint experiment with 3 attributes and 3 levels each

Assessing equilibration

Now that we have defined our metrics we can discuss how to assess if the process has equilibrated. We run the equilibration stage for 20 000 iteration, and discard the first 10 000 iterations. We divide the remaining iteration into 10 chains of 1000 iterations each, we calculate the Geweke statistic for each chain and count how many of them is within the range {-2.6, 2.6}. We also calculate the moving averages of this chain and calculate the standard deviation of the averages. We say that the simulation has equilibrated if the number of Geweke statistics within the range {-2.6, 2.6} is equal or larger than 5 and if the standard deviation of the moving averages is less than 0.025. If not, we run the simulation for an extra 10 000 iterations and repeat this process.

2.3 Sample size

One important question we need to consider is: What should my sample size be? This is not an easy question, and multiple rules of thumb have been proposed. Peduzzi *et al.* [7, 12], with respect to logit models, recommend the following sample size

$$N_{res} \geq 10 \frac{n_{ind}}{N_g p_{min}} \tag{2.30}$$

as defined above, n_{ind} is the total number of levels minus the number of attributes, N_g is the number of choice sets seen by each respondent. p_{min} is the probability of the least chosen option in all sets; we usually do not know this probability *a priori*, as such we could assume that all choices are equally likely (this could be a good assumption for a utility balanced design). For example, if each set had 3 products, then $p_{min} = 0.333$. The problem comes when the utilities are not balanced, in such cases $p_{min} < 1/N_{set}$ (N_{set}: number of products per choice set).

Orme and Chrzan [7] recommend a sample size given by

$$N_{res} \geq 1000 \frac{\ell_k^{max}}{N_{set} \times N_g} \tag{2.31}$$

where ℓ_k^{max} is the maximum number levels in any attribute.

We can also consider the desired margin of error of the shares of preference [7], if we assume that the shares of preference are proportions and that their errors follow a normal distribution, then the estimated error is given by

$$error = \pm z_{\alpha/2} \sqrt{\frac{\hat{p}(1-\hat{p})}{N_{res}}} \tag{2.32}$$

where $\hat{p}$ is the estimated proportion, and $z_{\alpha/2}$ is the standardized score corresponding to two tailed confidence level $1-\alpha$. For $\alpha = 0.05$, $z = 1.96$. Now for $N_{res} < 1000$, we usually use the t-score obtained from a Student's distribution where $t > z$. Since we are only interested in estimates we set $z = 2$, and we consider values of $N_{res} > 60$ as valid. Taking a conservative approach, we consider the value of $\hat{p}$ at which the *error* is a maximum, this value corresponds to $\hat{p} = 0.5$. As such, our expression reduces to

$$error = \pm \sqrt{\frac{1}{N_{res}}} \tag{2.33}$$

therefore, given a desired *error*, our sample size is

$$N_{res} = \frac{1}{|error|^2}. \tag{2.34}$$

For example, let's say we want our market share estimates be accurate to within $\pm 5\%$ or less, then our sample size should be

$$N_{res} \geq \frac{1}{0.05^2} = 400. \tag{2.35}$$

Notice that this number is inversely proportional to the square of the error, as such doubling our error means that our sample size decreases by a factor of 4. As a visual aid, I include the error for different sample sizes in the following table

N_{res}	*error*
50	±14.1%
100	±10.0%
150	±8.2%
200	±7.1%
300	±5.8%
400	±5.0%
500	±4.5%
600	±4.1%
700	±3.8%
800	±3.5%
900	±3.3%
1000	±3.2%

Chapter 3

Population segmentation

When running a conjoint experiment we will usually have respondents that come from different segments. For example, if we run a study about carbonated drinks, we could have a segment of people who like drinks with no sugar, another segment where people do not like the taste of artificial sweeteners, and a segment where people are indifferent to sugar content. Let's say we are interested in launching a new zero-sugar drink, as such our target will be the people who like no sugar drinks, and maybe also the people who are indifferent to sugar content, and we will be interested in investigating the characteristics of these population segments. That is, we would like to segment our surveyed population with respect to their responses to our conjoint experiment. *A priori*, we do not know anything about the segments, thus we need a way of identifying differences in our sample population. To divide people into segments with similar choice preferences we will use latent class multinomial logit.

3.1 Latent class multinomial logit

Given a desired number of segments, this method outputs an estimate of the levels' part worths for each segment. We do not assume that each respondent absolutely belongs to a given group, instead, each individual has a probability of belonging to each group. As the estimated part worths of a specific segment increasingly align with an individual responses, the likelihood of belonging to that segment also rises.

3.1.1 Method

For a given number of segments, N_{seg}, we need to obtain the part worths that defines each segment. To do this we will use the method defined by Greene and Hensher [13, 14]. First we need some definitions.

Following a multinomial logistic model, the probability of the ith individual choosing the qth product within the jth choice set given part worths $\boldsymbol{\beta}_s$, corresponding to segment s, is defined as [2, 7, 8, 13]

$$p_{i,j}(q|\boldsymbol{\beta}_s) = \frac{e^{\boldsymbol{\beta}_s \cdot \boldsymbol{x}_{i,j,q}}}{\sum_k e^{\boldsymbol{\beta}_s \cdot \boldsymbol{x}_{i,j,k}}} \tag{3.1}$$

and the likelihood of the ith respondent for a given segment s is calculated by multiplying the

probabilities of their responses

$$\mathrm{L}_i(\boldsymbol{\beta}_s) = \prod_{j=1}^{N_g} p_{i,j}(q_j|\boldsymbol{\beta}_i) \tag{3.2}$$

where N_g is the number of choice sets seen by the respondents, and q_j is the index of the product chosen by the respondent in the jth choice set.

The respondents classification into the different segments is unknown, as such we define the segment probabilities as logit probabilities

$$p_s = \frac{e^{\theta_s}}{\sum_{r=1}^{N_{seg}} e^{\theta_r}}. \tag{3.3}$$

Following Bayes' theorem we obtain the posterior segment probabilities given a response i,

$$p_{s|i} = \frac{\mathrm{L}_i(\boldsymbol{\beta}_s)p_s}{\sum_{r=1}^{N_{seg}} \mathrm{L}_i(\boldsymbol{\beta}_r)p_r} \tag{3.4}$$

these probabilities allow us to classify each respondent into the different segments, i.e. we assign respondent i to the segment which has the largest $p_{s|i}$.

In order to find the part worths for each segment we will use the expectation–maximization (EM) algorithm, to do this we let $u_{i,s} = 1$ if respondent i is a member of segment s, and 0 otherwise, and we treat these variables as unknown. Now the complete log likelihood of our sample is given by [13]

$$\ln(\mathrm{L}_c) = \sum_{i=1}^{N_{res}} \sum_{s=1}^{N_{seg}} \left[u_{i,s} \ln\left(\mathrm{L}_i(\boldsymbol{\beta}_s)\right) + u_{i,s} \ln(p_s) \right] \tag{3.5}$$

The EM algorithm maximizes this log likelihood by the use of two steps, expectation (E) and maximization (M). In the expectation step we obtain the expectation of the likelihood, $\mathrm{E}(\ln(\mathrm{L}_c))$, where we replace $u_{i,s}$ with $p_{s|i}$ in equation 3.5,

$$\mathrm{E}\big(\ln(\mathrm{L}_c)\big) = \sum_{i=1}^{N_{res}} \sum_{s=1}^{N_{seg}} \left[p_{s|i} \ln\left(\mathrm{L}_i(\boldsymbol{\beta}_s)\right) + p_{s|i} \ln(p_s) \right] \tag{3.6}$$

$p_{s|i}$ is calculated with our current estimates of β_s and p_s. In the maximization step we maximize the log likelihood using the estimated $p_{s|i}$s. Conditioned to these probabilities, the expectation of the log likelihood can be separated into two parts and we maximize them separately. The first part of $\mathrm{E}(\ln(\mathrm{L}_c))$ becomes a weighted log likelihood [13]

$$\frac{\partial \mathrm{E}(\ln(\mathrm{L}_c))}{\partial \boldsymbol{\beta}_s} = \sum_{i=1}^{N_{res}} p_{s|i} \frac{\partial \ln\left(\mathrm{L}_i(\boldsymbol{\beta}_s)\right)}{\partial \boldsymbol{\beta}_s} = \mathbf{0} \tag{3.7}$$

using equations 3.1 and 3.2 we get the set of equations for each β_s as

$$\sum_{i=1}^{N_{res}} p_{s|i} \frac{\partial \ln\left(\mathrm{L}_i(\boldsymbol{\beta}_s)\right)}{\partial \boldsymbol{\beta}_s} = \sum_{i=1}^{N_{res}} p_{s|i} \sum_{j=1}^{N_g} \left[\boldsymbol{x}_{i,j,q_j} - \frac{\sum_{k=1}^{N_{set}} \boldsymbol{x}_{i,j,k} e^{\boldsymbol{\beta}_s \cdot \boldsymbol{x}_{i,j,k}}}{\sum_{k=1}^{N_{set}} e^{\boldsymbol{\beta}_s \cdot \boldsymbol{x}_{i,j,k}}} \right] = \mathbf{0} \tag{3.8}$$

$$\sum_{i=1}^{N_{res}} p_{s|i} \frac{\partial \ln\left(\mathrm{L}_i(\boldsymbol{\beta}_s)\right)}{\partial \boldsymbol{\beta}_s} = \sum_{i=1}^{N_{res}} p_{s|i} \left[\left(\sum_{j=1}^{N_g} \boldsymbol{x}_{i,j,q_j} \right) - \left(\sum_{j=1}^{N_g} \frac{\sum_{k=1}^{N_{set}} \boldsymbol{x}_{i,j,k} e^{\boldsymbol{\beta}_s \cdot \boldsymbol{x}_{i,j,k}}}{\sum_{k=1}^{N_{set}} e^{\boldsymbol{\beta}_s \cdot \boldsymbol{x}_{i,j,k}}} \right) \right] = \mathbf{0} \tag{3.9}$$

where as before $\boldsymbol{x}_{i,j,q_j}$ is the chosen alternative by respondent i for set j, and N_{set} is the choice set size. This is a vector equation as $\boldsymbol{\beta}_s$ and $\boldsymbol{x}_{i,j,k}$ are vectors, and we solve for each component of $\boldsymbol{\beta}_s$.

The second part of the expectation maximization is given by [13]

$$\frac{\partial \mathrm{E}(\ln(\mathrm{L}_c))}{\partial \theta_s} = \sum_{i=1}^{N_{res}} p_{s|i} \frac{\partial \ln(p_s)}{\partial \theta_s} = 0 \tag{3.10}$$

using equation 3.4 we get

$$p_s = \frac{\sum_{i=1}^{N_{res}} p_{s|i}}{N_{res}}. \tag{3.11}$$

Therefore in order to find the part worths for a given number of segments we follow these steps

1. We initialize each $\boldsymbol{\beta}_s$ with random numbers, and let each $p_s = 1/N_{seg}$.
2. We calculate the $p_{s|i}$s for each respondent given our current estimates of $\boldsymbol{\beta}_s$ and p_s.
3. We obtain new estimates of $\boldsymbol{\beta}_s$ and p_s by solving equations 3.9 and 3.11.
4. We assess if the process has converged, if not we return to step 2, else we stop. We say the process has converged if the gain of the $\mathrm{E}(\ln(\mathrm{L}_c))$, equation 3.6, is less than 0.01, i.e. if $\mathrm{E}(\ln(\mathrm{L}_c))$ has increased by less than 0.001 from the previous estimate.

We repeat the steps defined above for different initial $\boldsymbol{\beta}_s$ and keep the simulation that gives us the largest $\ln(\mathrm{L}_c)$. We do this to avoid getting stuck in a local maximum. The highest log likelihood we can get is $N_{res} \sum_s p_s \ln(p_s)$, because if we could fit the data perfectly then the individual likelihoods will all be 1 and $\ln(1) = 0$.

3.2 Number of segments

Now, one question one might ask is, how many segments should we use? To assess this we will use the Bayesian information criterion (BIC),

$$\mathrm{BIC} = \left[N_{seg} \left(\sum_{k=1}^{N_a} \ell_k - N_a \right) + N_{seg} - 1 \right] \ln(N_{res}) - 2\ln(\mathrm{L}_c) \tag{3.12}$$

where N_a is the number of attributes and ℓ_k is the number of levels in the kth attribute. In order to find the optimal number of segments we calculate the BIC for increasing number of segments, the BIC will decrease as we increase the number of segments, but the rate of change will decrease considerably after a certain number of segments. As such, we will choose the number of segments corresponding to this "elbow point". For example, for the case considered in Appendix A, we find the elbow at $s = 3$, this is expected as we generated the data from 3 different $\boldsymbol{\beta}$s, this is plotted in the figure 3.1

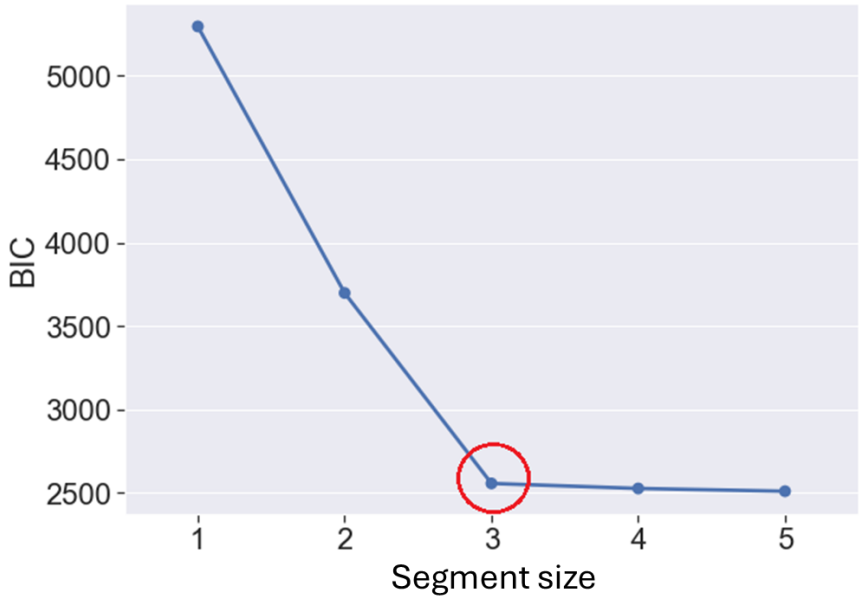


Figure 3.1: BIC for different number of segments, the elbow point is indicated by the red circle.

3.2.1 Elbow point

To find the elbow point we can compare the gradient of the lines before and after the point we are checking. For example, in figure 3.1 to see if the point at segment size = 3 is an elbow we get the gradient of the line from segment size = 2 to segment size = 3, and the gradient from segment size = 3 to segment size = 4. These gradients are

$$\Delta_{2,3} = \frac{BIC_3 - BIC_2}{3-2} = \frac{2552-3698}{3-2} = -1146$$
$$\Delta_{3,4} = \frac{BIC_4 - BIC_3}{4-3} = \frac{2526-2552}{3-2} = -26$$

Now, we can take the ratio between $\Delta_{3,4}$ and $\Delta_{2,3}$

$$r = \frac{\Delta_{3,4}}{\Delta_{2,3}} = \frac{-26}{-1146} = 0.02$$

As such we can determine a minimum r that will indicate that we have found an elbow, that is how much the gradient was reduced, here we will say that we found an elbow if $r < 0.1$. Here notice that if the gradient of the first section is negative and the gradient of the next section is positive we also determine we found an elbow, this is because in this case $r < 0$.

Chapter 4

Overall results

Once we have calculated the part worths for each respondent we can summarize our findings by calculating the performance of each level and the importance of each attribute.

4.1 Level Performance

Level performance quantifies the influence a level has on consumers when choosing a product or service. It is calculated as follows:

1. First, we calculate the average of the respondents' part worths for each level; for example, our averages could be

Brand			Color			Price		
Brand 1	**Brand 2**	**Brand 3**	**Red**	**Black**	**Blue**	**$100**	**$150**	**$200**
0.8666	-1.4576	0.5910	-1.1974	1.3227	-0.1252	0.3676	0.0250	-0.3926

2. Next, for each attribute we take the difference between its biggest level part worth and smallest level part worth. This is known as the range of an attribute. For our example we have

Brand	Color	Price
2.3242	2.5201	0.7602

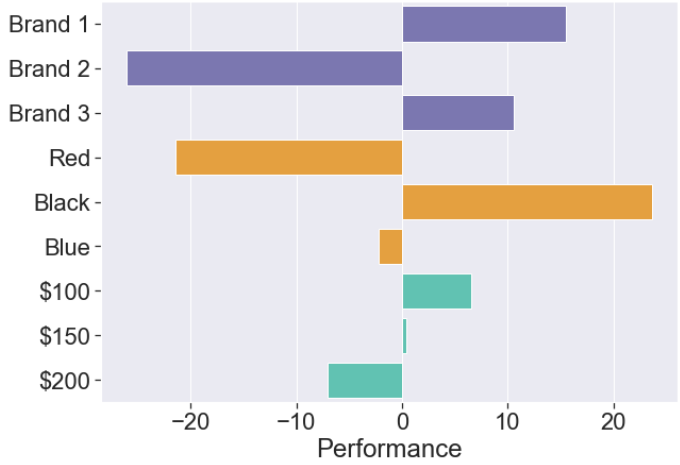
3. The performance of each level is calculated by dividing its average part worth by the sum of the ranges of the attributes. We express these proportions as percentages. Following our example, the level performance for **Brand 1** is

$$\frac{0.8666 \times 100}{2.3242 + 2.5201 + 0.7602} = 15.46\%$$

and all the level performances are

Brand			Color			Price		
Brand 1	**Brand 2**	**Brand 3**	**Red**	**Black**	**Blue**	**$100**	**$150**	**$200**
15.46%	-26.01%	10.55%	-21.37%	23.60%	-2.23%	6.56%	0.45%	-7.01%

As such, level performance is the proportion of each level average part worth with respect to the total attribute range. The higher the performance of a level, the more influence it has on consumers' choices. We can visualise level performance using a horizontal bar plot, which allows us to quickly differentiate which levels perform better and worse:



4.2 Attribute importance

Attribute importance measures how much an attribute matters to consumers when making a choice. It is calculated as follows:

1. First, for each respondent, we calculate the range of each attribute by taking the difference between its biggest level part worth and smallest level part worth. For example, we have the following individual part worths

Brand			Color			Price		
Brand 1	**Brand 2**	**Brand 3**	**Red**	**Black**	**Blue**	**$100**	**$150**	**$200**
0.6224	-1.0016	0.3792	-0.1743	0.7081	-0.5338	3.2356	-1.4937	-1.7419
0.5788	-3.1529	2.5741	-2.6824	0.9787	1.7037	0.4503	0.3087	-0.7590
0.1266	-2.8429	2.7163	-0.9462	2.5718	-1.6256	0.5569	0.1792	-0.7362
-2.3082	-2.3868	4.6950	-0.5690	1.0848	-0.5158	0.2459	0.0470	-0.2929
-0.6472	-0.4663	1.1134	-5.3739	7.2894	-1.9155	0.5686	0.0913	-0.6599

   then, the attributes ranges are

Brand	Color	Price
1.6240	1.2419	4.9774
5.7270	4.3862	1.2093
5.5593	4.1974	1.2931
7.0817	1.6538	0.5387
1.7606	12.6633	1.2285

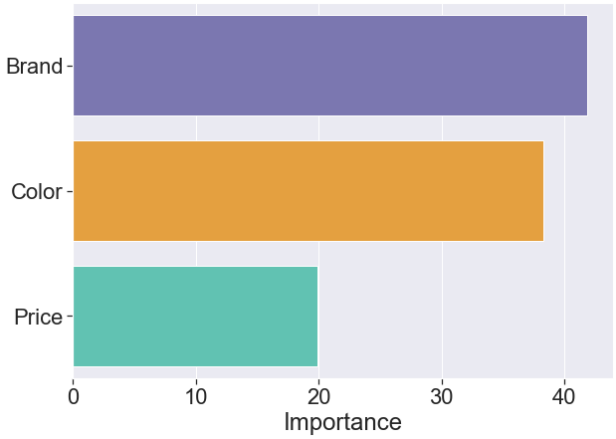
2. Next, for each respondent, we divide each attribute range by the sum of the ranges and we express these proportions as percentages. These proportions are the individual attribute importances, for our example we get

Brand	Color	Price
20.71%	15.83%	63.46%
50.58%	38.74%	10.68%
50.31%	37.99%	11.70%
76.36%	17.83%	5.81%
11.25%	80.90%	7.85%

3. Finally, we get the average of the individual attribute importances, following our example we get

Brand	Color	Price
41.84%	38.26%	19.90%

The higher the attribute importance, the more it matters to a consumers when choosing a product or service. As with level performance, we can visualise attribute importance using a horizontal bar plot, this will allow us to quickly differentiate which attributes matter the most and the least:



Chapter 5

Simulators

In the previous chapters we learned how to design our experiment, we also learned how to find the value each respondent gives to each attribute, and finally we saw how we can segment our respondents. But this information is not very useful if we cannot make predictions or inferences about our population of interest, and how this population would react and interact with our products. We could use the individual part worths directly to get information about each attribute level, but this would become overwhelming even for a relatively small number of respondents, say for example 50. We could also look at the average part worths of all respondents, but in doing so we loose information about individual preferences. This is when market simulators come into play, imagine we had a group of people to which we could ask as many questions as we want about potential products. We can do that by using their part worths and make them choose between different products, and hence obtain valuable information about market share distribution, price sensitivities, demand trends, and more. We can use this information to study hypothetical scenarios and answer questions such as: How would my product perform versus similar products in the market?, How would the demand of my product change in a competitive environment if I change X characteristic?, What can I change to make my product more appealing?, etc.

Now it is important to understand that there are many factors that affect the preferences of our buyers and that all of them cannot be captured by our simulators. Nevertheless, this should not discourage you as even given these limitations, our simulators can give us valuable information about the relative distribution of preferences isolated from other complicated factors that could be outside of our control. In this chapter I will describe what market share and demand simulators are and how to construct them using the respondents part worths.

5.1 Market share simulator

This simulator is used to find consumers' preferences about a set of competing products. Let's say we are a company that produces soft drinks and we want to introduce a new no-sugar beverage to the market, as such we would like to get an idea of how this product will perform against other four no-sugar beverages from different companies. From our simulator we find that 5% of potential buyers preferred our product and the other 95% is distributed among the other beverages, from this we will need to take a step back and analyse why our product would not be as competitive as we would like it to be; here we could change some characteristics of our product and make it compete again with the other soft drinks.

In order to find the respondents' preferences we first need to define the products that we want to study. Next, for each respondent we find the utilities of each product by adding their levels'

part worths and find their individual preferences by using equation

$$p_{i,k} = \frac{e^{U_{i,k}}}{\sum_j e^{U_{j,k}}} \tag{5.1}$$

where $p_{i,k}$ is the probability of respondent k choosing product i, and $U_{i,k}$ is the utility of product i for respondent k. The preference respondent k gives to product i is equal to $p_{i,k}$, the total preference of product i, P_i, for our population is equal to

$$P_i = \sum_{k=1}^{N_{res}} p_{i,k}. \tag{5.2}$$

and the share of preference of product i is

$$\text{share}_i = \frac{P_i}{\sum_j P_j} \tag{5.3}$$

Therefore to find the shares of preference of our products of interest, we find the probabilities of each product being chosen by each respondent, we add those probabilities for all respondents, and find their relative preference with respect to all products being considered. Now, this method does not satisfy the *Independence of Irrelevant Alternatives* property, IIA, also known as the *Blue bus/Red bus paradox*. The IIA condition says that the relative likelihood of choosing between two options should not change if a third option is added. But in our case, if we add an option that is similar to an already added option, then this new option will take share from all other options. For example, let's imagine our city has the following modes of transportation with utilities

	Utility	Probability
Blue Bus	1.2	32.4%
Bike	0.9	24.0%
Car	1.5	43.6%

now let's say we add a Red Bus that is exactly the same as the Blue Bus (apart from the color), then it is expected that it will only borrow preference from the Blue bus and not the other modes of transportation, but what we get is the following

	Utility	Probability
Blue Bus	1.2	24.4%
Red Bus	1.2	24.4%
Bike	0.9	18.1%
Car	1.5	33.0%

as we can see it borrowed preference from all modes of transportation, not only the Blue Bus, and the probability of choosing a Bus is 48.8% which is higher than before.

To reduce IIA problems we will use a top-N rule, where we take the top N products, allocate share proportionally to them, and make the share of the remaining products equal to zero. The share of the top products for respondent k is calculated as

$$p_{i,k}^{new} = \frac{p_{i,k}}{\sum\limits_{j \in top} p_{j,k}} \qquad \forall\ i \in top \tag{5.4}$$

For example, if we had five products with the following preferences for two respondents

	Product 1	Product 2	Product 3	Product 4	Product 5
Respondent 1	0.20	0.25	0.05	0.10	0.40
Respondent 2	0.35	0.05	0.20	0.25	0.15

the top 3 products for respondent 1 are Product 5, Product 2, and Product 1, and for respondent 2 are Product 1, Product 4, and Product 3, as such we choose these products and find their new preferences following equation 5.4,

$$p_{1,1} = \frac{0.20}{0.20+0.25+0.40} = 0.24 \qquad p_{1,2} = \frac{0.35}{0.35+0.20+0.25} = 0.44$$
$$p_{2,1} = \frac{0.25}{0.20+0.25+0.40} = 0.29 \qquad p_{3,2} = \frac{0.20}{0.35+0.20+0.25} = 0.25$$
$$p_{5,1} = \frac{0.40}{0.20+0.25+0.40} = 0.47 \qquad p_{4,2} = \frac{0.25}{0.35+0.20+0.25} = 0.31$$

hence, the final preferences and shares of preference are

	Product 1	Product 2	Product 3	Product 4	Product 5
Respondent 1	0.24	0.29	0.0	0.0	0.47
Respondent 2	0.44	0.0	0.25	0.31	0.0
Total Preference	0.68	0.29	0.25	0.31	0.47
Share of preference	34%	14.5%	12.5%	15.5%	23.5%

5.2 Demand curves, revenue curves, and price elasticities simulator

This simulator is used to study how the demand of a product, competing with other products, is affected by changes in price. To calculate demand curves we first need to define our products of interest. In the case we want to get the demand curve of one of the products we keep the other products constant, change the price of the product of interest, and calculate the share of preference of the product of interest at each price point. For example, let's say our product competes with two other products

	Brand	Color	Price
Our product	Brand 1	Red	$110
Product 2	Brand 2	Blue	$105
Product 3	Brand 3	Black	$110

the price levels of our products are {$100, $105, $110, $115}, thus we have to get the share of preference of our product at the price of $100, $105, $110, and $115, keeping the other products fixed (including their price, we only change the price of the product of interest). Let's say the shares of preference of our product at the different price points are

	$100	$105	$110	$115
Share of preference	55%	49%	37%	15%

As such our demand curve is:

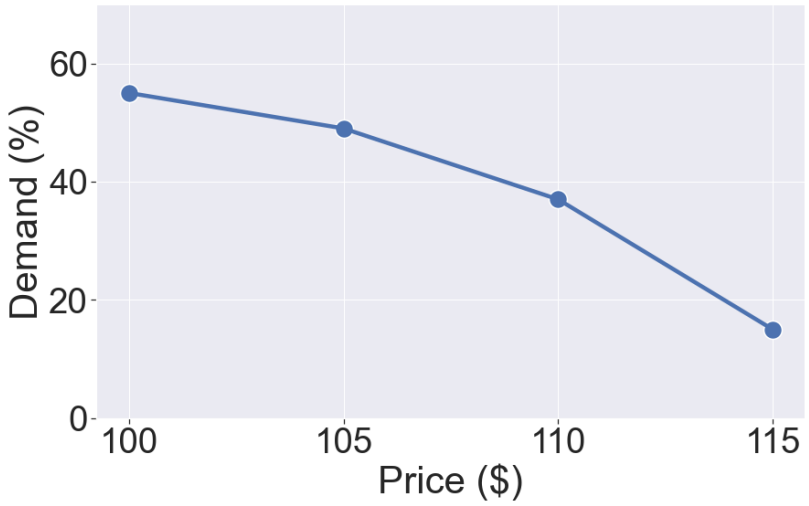


Figure 5.1: Demand curve of product of interest.

If we want to get demand curves of all the products we have to pick a fixed price point (usually the middle price), we set this price for all products, we take the first product, change its price as before, calculate its demand at each price point, then we take the next product, fix the price for the other products, change the price of the current product, calculate the demand at the different price points, and repeat until we get the demand curves of all the products.

Revenue curves

Using our demand curves we can calculate our expected revenue at each price point. To calculate the expected revenue we multiply each price by their corresponding demand and by the total number number of potential buyers. In our example above, if we had 200 potential buyers, our revenue at each price point is

	$100	$105	$110	$115
Revenue	$100 \times 0.55 \times 200 = \$11000$	$10290	$8140	$3450

as such our revenue curve is:

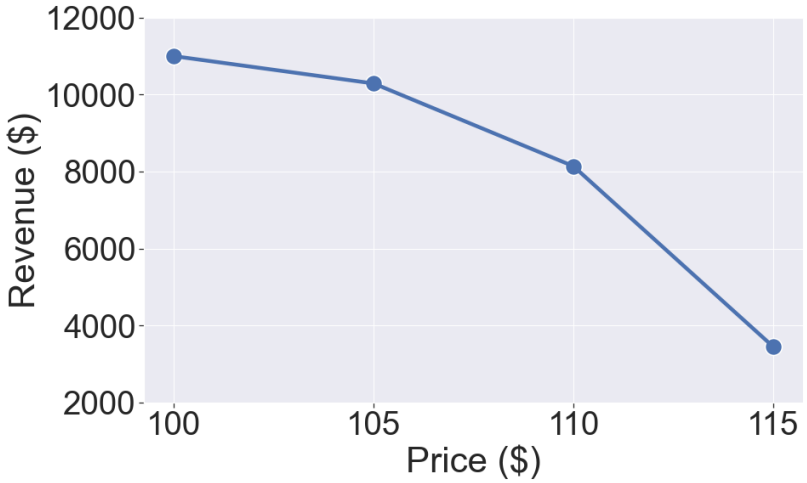


Figure 5.2: Revenue curve of product of interest.

Price elasticty of demand

Demand curves let us estimate the price elasticity of demand (PED) of our products. The PED tells us how sensitive the demand of our product is with respect to changes in price. If PED is less than 1 we say that our product is inelastic, i.e., changing the price of our product will have a small effect on the demand. On the other hand, if the PED is equal or greater than 1 we say that our product is elastic, that is changes in price will have a significant impact on the demand of the product. The price elasticity is calculated following the formula

$$\text{PED} = \frac{\%\text{ change in demand}}{\%\text{ change in price}} = \left| \frac{\Delta CF/CF_i}{\Delta P/P_i} \right| \tag{5.5}$$

where $|\ |$ stands for absolute value, CF_i stands for the initial cumulative frequency of the initial price P_i, and Δ stands for change. In our example above, the price elasticity between \$100 and \$105 is:

$$PED = \left| \frac{(49-55)/55}{(105-100)/100} \right| = \left| \frac{-6/55}{5/100} \right| = \frac{6/55}{5/100} = 2.18 \tag{5.6}$$

As the PED > 1 we say that our product is elastic between the price range \$100 - \$105. The average price elasticity is calculated using formula

$$\text{PED}_{avg} = \frac{\frac{2(d_f - d_i)}{d_f + d_i}}{\frac{2(pr_f - pr_i)}{pr_f + pr_i}} \tag{5.7}$$

where d_f is the demand of the maximum price pr_f, and d_i is the demand of the minimum price pr_i.

Price interpolation

So far we have only used the price levels defined in our experiment, but in most cases we would like to be able to consider price as a continuous variable. The issue is that we cannot ask respondents to consider every possible price, as such we will use piece-wise linear interpolation. We will fit straight lines between consecutive data points, the equation of a line going from point (x_o, y_o) to point (x_f, y_f) is

$$y = \frac{y_f - y_o}{x_f - x_o}(x - x_o) + y_o \qquad x_o \leq x \leq x_f. \tag{5.8}$$

This method of interpolation can be applied to any numeric attribute not only price, as such if other numeric attributes are need to be continuous then this method can be applied.

To obtain the demand point we will get N_p price points that include the defined price levels in our experiment. Once we get the demand curves we will use piece-wise cubic splines to interpolate between consecutive demand point, that way we can get price elasticities between any pair of price points, the price point must lie between the minimum and maximum price levels.

Bibliography

[1] D. J. Street and L. Burgess, *The construction of optimal stated choice experiments: Theory and methods*. John Wiley & Sons, 2007.

[2] K. E. Train, *Discrete choice methods with simulation*. Cambridge university press, 2009.

[3] H. Xu, "An algorithm for constructing orthogonal and nearly-orthogonal arrays with mixed levels and small runs," *Technometrics*, vol. 44, no. 4, pp. 356–368, 2002.

[4] J. Eccleston and A. Hedayat, "On the theory of connected designs: characterization and optimality," *The annals of statistics*, pp. 1238–1255, 1974.

[5] G. M. Allenby and P. J. Lenk, "Modeling household purchase behavior with logistic normal regression," *Journal of the American Statistical Association*, vol. 89, no. 428, pp. 1218–1231, 1994.

[6] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. CRC press, 2013.

[7] B. K. Orme and K. Chrzan, *Becoming an Expert in Conjoint Analysis: Choice Modeling for Pros*. Sawtooth Software, Inc., 2021.

[8] J. J. Louviere, D. A. Hensher, and J. D. Swait, *Stated choice methods: analysis and applications*. Cambridge university press, 2000.

[9] M. K. Cowles and B. P. Carlin, "Markov chain Monte Carlo convergence diagnostics: a comparative review," *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 883–904, 1996.

[10] J. Geweke, "Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments," tech. rep., Federal Reserve Bank of Minneapolis, 1991.

[11] P. Heidelberger and P. D. Welch, "A spectral method for confidence interval generation and run length control in simulations," *Communications of the ACM*, vol. 24, no. 4, pp. 233–245, 1981.

[12] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, "A simulation study of the number of events per variable in logistic regression analysis," *Journal of clinical epidemiology*, vol. 49, no. 12, pp. 1373–1379, 1996.

[13] W. H. Greene and D. A. Hensher, "A latent class model for discrete choice analysis: contrasts with mixed logit," *Transportation Research Part B: Methodological*, vol. 37, no. 8, pp. 681–698, 2003.

[14] W. H. Greene, "Fixed and random effects in nonlinear models," 2001.

Appendix A

Finding segments: an example

A.1 Finding the segments' part worths

To test our segmentation method we can generate respondents from a set of pre-defined $\boldsymbol{\beta}$s adding random errors that follow a Gumbel distribution. First we need to define our attributes and levels:

Brand	Colour	Price
Brand 1	Black	$100
Brand 2	Blue	$105
Brand 3	Red	$110
		$115

with these we construct 3 sets of $\boldsymbol{\beta}$s

		Segment 1	Segment 2	Segment 3
Brand	Brand 1	1.5	-1.5	1.5
	Brand 2	-1.0	1.0	-0.5
	Brand 3	-0.5	0.5	-1.0
Colour	Black	-1.5	-1.0	1.5
	Blue	1.0	1.5	-1.0
	Red	0.5	-0.5	-0.5
Price	$100	2.0	2.0	2.0
	$105	1.0	1.0	1.0
	$110	-1.0	-1.0	-1.0
	$115	-2.0	-2.0	-2.0

given that our latent class method will shrink or stretch the utilities of every segment depending on the goodness of fit, we will use the relative performance of the levels. To obtain the relative performance we follow Chapter 4. The relative performances for each segment are:

		Segment 1	Segment 2	Segment 3
	Brand 1	16.67%	-16.67%	16.67%
Brand	Brand 2	-11.11%	11.11%	-5.56%
	Brand 3	-5.56%	5.56%	-11.11%
	Black	-16.67%	-11.11%	16.67%
Color	Blue	11.11%	16.67%	-11.11%
	Red	5.56%	-5.56%	-5.56%
	$100	22.22%	22.22%	22.22%
Price	$105	11.11%	11.11%	11.11%
	$110	-11.11%	-11.11%	-11.11%
	$115	-22.22%	-22.22%	-22.22%

Now we need to define our experiment (using effects coding) as follows

Choice set index	Brand 1	Brand 2	Black	Blue	$100	$105	$110
1	1	0	1	0	0	0	1
	0	1	-1	-1	1	0	0
	-1	-1	0	1	0	1	0
2	1	0	1	0	0	1	0
	0	1	0	1	1	0	0
	-1	-1	-1	-1	0	0	1
3	1	0	0	1	1	0	0
	0	1	1	0	0	0	1
	-1	-1	-1	-1	0	1	0
4	1	0	1	0	-1	-1	-1
	0	1	0	1	0	1	0
	-1	-1	-1	-1	1	0	0
5	1	0	0	1	0	1	0
	0	1	-1	-1	-1	-1	-1
	-1	-1	1	0	1	0	0
6	1	0	-1	-1	1	0	0
	0	1	0	1	-1	-1	-1
	-1	-1	1	0	0	0	1
7	1	0	0	1	0	0	1
	0	1	-1	-1	0	1	0
	-1	-1	1	0	-1	-1	-1
8	1	0	0	1	-1	-1	-1
	0	1	-1	-1	0	0	1
	-1	-1	1	0	0	1	0
9	1	0	-1	-1	0	1	0
	0	1	1	0	1	0	0
	-1	-1	0	1	-1	-1	-1
10	1	0	-1	-1	-1	-1	-1

	0	1	1	0	0	1	0
	-1	-1	0	1	0	0	1
11	1	0	-1	-1	0	0	1
	0	1	1	0	-1	-1	-1
	-1	-1	0	1	1	0	0
12	1	0	1	0	1	0	0
	0	1	0	1	0	0	1
	-1	-1	-1	-1	-1	-1	-1

Next, we generate 80, 100, and 90 respondents following the $\boldsymbol{\beta}$s of Segment 1, Segment 2, and Segment 3 respectively. In order to generate these responses we calculate the utilities for each product within each choice set and add to each utility a random number that follows a Gumbel distribution with a location $\mu = 0$ and a scale $\gamma = 0.5$, and select the product with the highest utility. The generated set of responses can be found in the excel file *segmentation_data_set.xlsx*, where each column represents a respondent, a 1 means the product in that row was selected and 0 otherwise, the document rows follow the same order as defined above. With these artifical responses we run the method defined in Chapter 3. We set our number of segments to 3 and run five different simulations starting at different random $\boldsymbol{\beta}$s to avoid getting stuck in a local maximum. After each step we compute the complete log likelihood of our sample, $\ln(L_c)$, and we stop after the gain is less than 0.01. The results are summarized in the following table

	Simulation 1	**Simulation 2**	**Simulation 3**	**Simulation 4**	**Simulation 5**
$\ln(L_c)$	-3803.53	-6674.17	-5723.52	-4593.47	-10524.32
	-2357.79	-1989.06	-2139.05	-1782.18	-1938.26
	-1989.37	-1542.64	-1540.48	-1214.45	-1470.51
	-1977.43	-1240.07	-1250.72	-1213.49	-1224.84
	-1973.48	-1215.05	-1218.32	-1213.49	-1213.49
	-1974.45	-1213.49	-1213.86		-1213.49
		-1213.49	-1213.55		
			-1213.49		
			-1213.49		

as we can see the maximum $\ln(L_c)$ is -1213.49, the method classified all respondents correctly and estimated the relative performances as follows

		Segment 1	Segment 2	Segment 3
Brand	Brand 1	16.82%	-16.87%	17.22%
	Brand 2	-10.58%	11.18%	-5.64%
	Brand 3	-6.24%	5.69%	-11.58%
Color	Black	-18.13%	-9.56%	15.74%
	Blue	11.00%	16.62%	-11.12%
	Red	7.14%	-7.04%	-4.61%
Price	$100	20.83%	21.95%	23.07%
	$105	11.93%	11.26%	9.11%
	$110	-10.12%	-9.39%	-10.89%
	$115	-22.64%	-23.82%	-21.27%

As we can see the part worths are not exactly the same as the ones defined above, but this is expected due to the random error we added to the utilities when generating the artificial responses.

A.2 Optimal number of segments

To test if we can estimate the optimal number of segments we run our latent class method from 1 to 6 segments, for each number of segments we run 5 simulations to avoid getting stuck in a global maximum. For the best simulations at each number of segments we calculate the BIC, the results are plotted in figure A.1

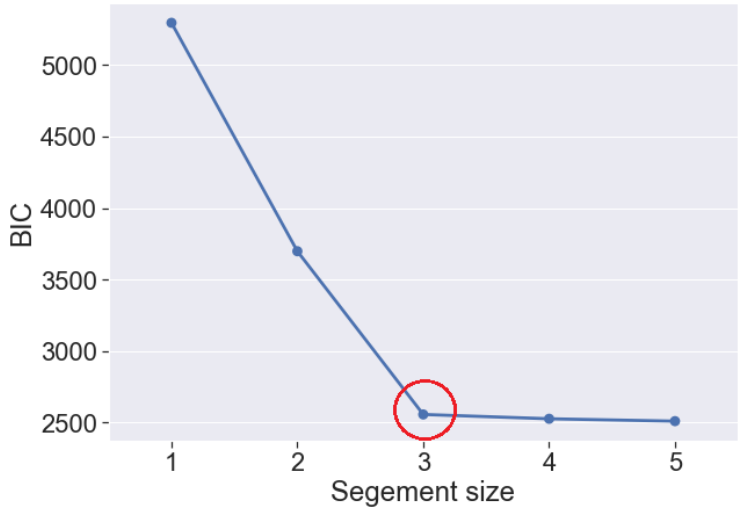


Figure A.1: BIC for different number of segments, the elbow point is indicated by the red circle.

here we can see that the elbow point is located a 3 segments which is expected as our data was generated from 3 sets of $\boldsymbol{\beta}$s.

Appendix B

Finding utilities: an example

B.1 Finding individual part worths

To test our utility estimation method we can generate respondents from a set of pre-defined part worths adding random errors that follow a Gumbel distribution. First we need to define our attributes and levels:

Brand	Colour	Price
Brand 1	Black	$100
Brand 2	Blue	$105
Brand 3	Red	$110
		$115

with these we define the part worths

Attribute	Level	Part worths
Brand	Brand 1	1.5
	Brand 2	-1.0
	Brand 3	-0.5
Colour	Black	-1.5
	Blue	1.0
	Red	0.5
Price	$100	2.0
	$105	1.0
	$110	-1.0
	$115	-2.0

given that our method will shrink or stretch the utilities depending on the goodness of fit, we will use the relative performance of the levels. To obtain the relative performance we follow Chapter 4. The relative performances are:

Attribute	Levels	Part worths
Brand	Brand 1	16.67%
	Brand 2	-11.11%
	Brand 3	-5.56%
Color	Black	-16.67%
	Blue	11.11%
	Red	5.56%
Price	$100	22.22%
	$105	11.11%
	$110	-11.11%
	$115	-22.22%

Now we need to define our experiment (using effects coding) as follows

Choice set index	Brand 1	Brand 2	Black	Blue	$100	$105	$110
1	1	0	1	0	0	0	1
	0	1	-1	-1	1	0	0
	-1	-1	0	1	0	1	0
2	1	0	1	0	0	1	0
	0	1	0	1	1	0	0
	-1	-1	-1	-1	0	0	1
3	1	0	0	1	1	0	0
	0	1	1	0	0	0	1
	-1	-1	-1	-1	0	1	0
4	1	0	1	0	-1	-1	-1
	0	1	0	1	0	1	0
	-1	-1	-1	-1	1	0	0
5	1	0	0	1	0	1	0
	0	1	-1	-1	-1	-1	-1
	-1	-1	1	0	1	0	0
6	1	0	-1	-1	1	0	0
	0	1	0	1	-1	-1	-1
	-1	-1	1	0	0	0	1
7	1	0	0	1	0	0	1
	0	1	-1	-1	0	1	0
	-1	-1	1	0	-1	-1	-1
8	1	0	0	1	-1	-1	-1
	0	1	-1	-1	0	0	1
	-1	-1	1	0	0	1	0
9	1	0	-1	-1	0	1	0
	0	1	1	0	1	0	0
	-1	-1	0	1	-1	-1	-1
10	1	0	-1	-1	-1	-1	-1

	0	1	1	0	0	1	0
	-1	-1	0	1	0	0	1
11	1	0	-1	-1	0	0	1
	0	1	1	0	-1	-1	-1
	-1	-1	0	1	1	0	0
12	1	0	1	0	1	0	0
	0	1	0	1	0	0	1
	-1	-1	-1	-1	-1	-1	-1

Next, we generate 100 respondents following the part worths defined above. In order to generate these responses we calculate the utilities for each product within each choice set and add to each utility a random number that follows a Gumbel distribution with a location $\mu = 0$ and a scale $\gamma = 0.8$, and select the product with the highest utility. The generated set of responses can be found in the excel file *utility_estimation_data_set.xlsx*, where each column represents a respondent, a 1 means the product in that row was selected and 0 otherwise, the document rows follow the same order as defined above. With these artificial responses we run the method defined in Chapter 2.

B.1.1 Adaptive stage

We initialize $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}_i$ as zero vectors, and $\boldsymbol{V}$ as indicated in section 2.2 of chapter 2. We run our simulation for 3000 iterations, after each iteration we adjust the proportionality factor f used in the Metropolis Hastings Algorithm. The evolution of f can be seen in the following plot

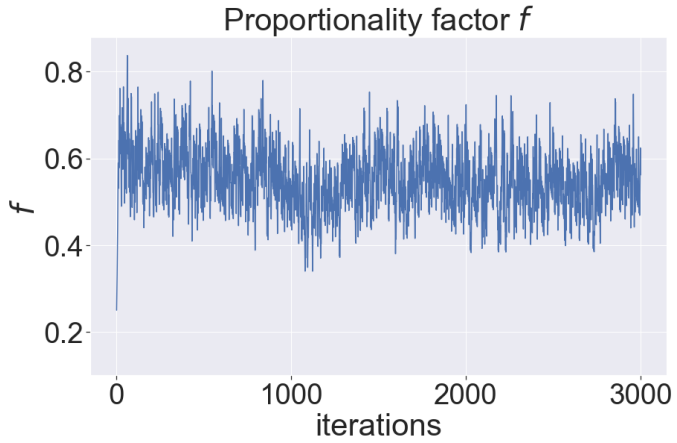


Figure B.1: f factor for successive iterations of the conjoint experiment defined above.

We discard the first 1 500 iterations and take the average of the remaining 1 500 iterations to calculate the f that will be used in the next two stages; for our example $f = 0.542$.

B.2 Equilibration stage

We set the proportionality factor to $f = 0.542$ an run our simulation for 20 000 steps, after every iteration we calculate R^2_{Mc}. The evolution of R^2_{Mc} can be seen in the following plot

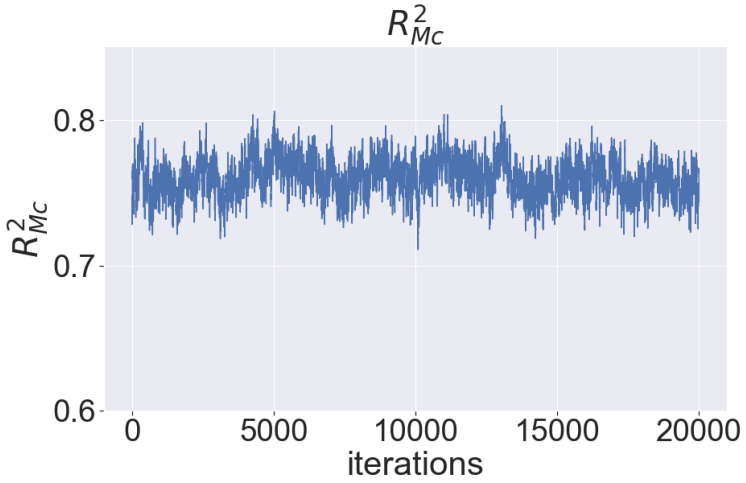


Figure B.2: R^2_{Mc} for successive iterations of the conjoint experiment defined above.

To see if our simulation has equilibrated we discard the first 10 000 iterations, and divide the remaining ones into 10 Markov chains each of length $N = 1\,000$, for each chain we calculate the Geweke statistic. The results are shown in the following plot

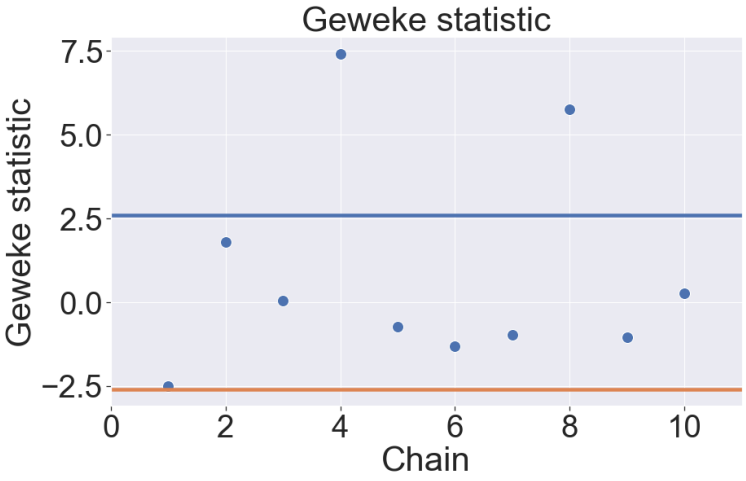


Figure B.3: Geweke statistics for successive Markov chains of the conjoint experiment defined above.

Here we can see that 8 of them fall within the {-2.6,2.6} range. We also calculate calculate the exponential moving average as shown bellow

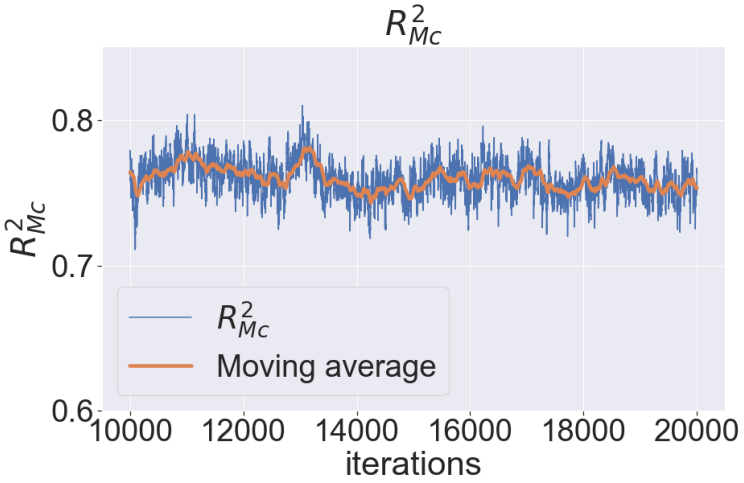


Figure B.4: R^2_{Mc} and its moving average for successive iterations of the conjoint experiment defined above.

The standard deviation of the moving average is 0.007. As both conditions defined in Chapter 2 section 2.2.3 are satisfied, we say that the simulation has reach equilibrium.

B.2.1 Measurement stage

We run the simulation for 10 000 more iterations and record the $\boldsymbol{\beta}$ of every 5th fifth iteration. The final individual part worths are the averages of the recorded $\boldsymbol{\beta}$. The individual part worths can be found in the file *part_worths_example.xlsx*. The relative performances are presented in the following table

Attribute	Level	Part worths
Brand	Brand 1	16.83%
	Brand 2	-11.50%
	Brand 3	-5.33%
Colour	Black	-16.42%
	Blue	11.02%
	Red	5.40%
Price	$100	21.61%
	$105	11.57%
	$110	-10.56%
	$115	-22.62%

As we can see the part worths are not exactly the same as the ones defined above, but this is expected due to the random error we added to the utilities when generating the artificial responses.