

Estimation of discrete choice models in R: writing an R package

Hanshui ZHANG

Supervisor: Prof. M. Vandebroek

Mentor: Deniz Akinc

Thesis presented in
fulfillment of the requirements
for the degree of Master of Science
in Statistics

Academic year 2012-2013

© Copyright by KU Leuven

Without written permission of the promoters and the authors it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilize parts of this publication should be addressed to KU Leuven, Faculteit Wetenschappen, Geel Huis, Kasteelpark Arenberg 11 bus 2100, 3001 Leuven (Heverlee), Telephone +32 16 32 14 01.

A written permission of the promoter is also required to use the methods, products, schematics and programs described in this work for industrial or commercial use, and for submitting this publication in scientific contest

Preface

The *Mixed Probit* and *Mixed Logit Models* from the *Discrete Choice Models* family have been very popular for the modeling of the choice behavior of the customers in marketing research field. The Bayesian inference in conjunction with the *Monte Carlo Markov chain* techniques has been increasingly used for the estimation of the *Mixed Probit* and *Mixed Logit Models* over the past few decades. Several R packages/function can be used for the estimation of the *Mixed Probit* and *Mixed Logit Models* based on a Bayesian approach.

The objective of this thesis is to compare the functionalities and the estimation results of different R packages for the Bayesian estimations of *Mixed Probit* and *Mixed Logit Models*. A user friendly interface is implemented in this thesis which allows the user to transform the input data structure into the formats required by different R packages/function. In this case, the user can obtain the estimation result of a particular R package faster.

I wish to thank my thesis promoter Prof. M. Vandebroek, and my thesis advisor D. Akinc for suggestions that significantly improved the thesis.

I would also like to thank three anonymous professors, authors of *bayesm* package, *RSGHB* package and *ChoiceModelR* package for the thoughtful suggestions and comments.

Finally, I would like to thank my parents and friends for all the support during the studies of this Master Program.

Summary

In marketing research, making inferences on the choice behaviors of customers is very crucial. The *Discrete Choice Models*, particularly *Mixed Probit Model* and *Mixed Logit Model* are popular for modeling the choice behaviors of individuals. These model are quite complex, and in order to perform such model estimation, usually a Bayesian approach is used. Various R packages are available to be used for the Bayesian estimation of the *Mixed Probit* and *Mixed Logit Models*.

The thesis first discusses the basic concepts of the *Discrete Choice Models*, Bayesian procedures, *Monte Carlo Markov chain* techniques, and the steps of Bayesian estimations for *Mixed Probit* and *Mixed Logit Models*. After providing the readers some theoretical background of how to perform such model estimation, the functionalities of several R packages/function which can be used to estimate these models in a Bayesian approach are summarized and compared.

The R packages used for the *Mixed Logit Model* estimation are *bayesm*, *ChoiceModelR*, and *RSGHB* packages. The analysis of the estimation results indicates that the parameter estimates using these R packages are converging to similar values. The estimation accuracy for *RSGHB* package is the most accurate among the three R package, and the estimation speed of *RSGHB* package is also faster than *bayesm* and *ChoiceModelR* packages. But the specification of the likelihood function in *RSGHB* package can be time consuming when dealing with large number of model parameters, and the user should be careful when specifying the likelihood function so that the correct estimation results can be obtained. The estimation performed using *bayesm* and *ChoiceModelR* packages are similar to each other, the *bayesm* package yields better estimation accuracy than the *ChoiceModelR* package, but the estimation time using *bayesm* package is relatively long especially when estimating very large datasets.

According to the research, currently there is no R package for the *Mixed Probit* estimation, therefore a function called *rmxp* is implemented based on the R function provided in Gillbride & Allenby (2004). The *rmxp* function is used for the estimation of the *Mixed Probit Model* in this thesis. The precision criteria calculated based on the estimation results using the *rmxp* function suggests that this function gives good estimation accuracy. However, since no other R package is available to be used for the *Mixed Probit* estimation result comparison, the values of the precision criteria are not quite informative in this case. Therefore, it

is necessary to have a further exploration of the developments of R packages for *Mixed Probit* estimation.

Different R packages require different input data structures to be entered before starting the model estimation. In order to save programming time, a user interface is implemented in R so that the researcher can transform a given dataset into the required formats of the R packages/function discussed in this thesis. However, after transforming the data structure, the user is still required to specify various control settings for the model estimation depending on different R packages/function. If more R packages are available for the *Mixed Logit* or *Mixed Probit Model* estimation, the user can update the content of the user interface function so that new data input procedures will be included in this user interface.

Table of Contents

Preface	I
Summary	II
List of Figures	VI
List of Tables	VII
List of Symbols	VIII
Chapter 1 Introduction	1
Chapter 2 Discrete Choice Models	3
2.1 Probit Model	4
2.1.1 Limitation of Probit Model	5
2.1.2 Identification Problem of Probit Model	5
2.2 Mixed Probit Model	7
2.3 Logit Model	8
2.3.1 Limitations of Logit Model	9
2.4 Mixed Logit Model	10
Chapter 3 Estimation of Bayesian Hierarchical Models	12
3.1 General Concepts of Bayesian Inference	13
3.2 Monte Carlo Markov chain Methods	15
3.2.1 Metropolis-Hasting algorithm	15
3.2.2 Random-walk Metropolis Algorithm	16
3.2.3 Gibbs Sampling Algorithm	17
3.3 Data Augmentation Technique	18
3.4 Bayesian Estimation for Mixed Logit Models	19
3.5 Bayesian Estimation for Mixed Probit Models	21
Chapter 4 Overview of the Available R Packages	23
4.1 R packages for Mixed Logit Estimation	23
4.1.1 bayesm Package (Rossi & McCulloch, 2005)	23
4.1.2 ChoiceModelR Package (Sermas, 2012)	26
4.1.3 RSGHB Package (Dumont <i>et al.</i> , 2013)	28
4.1.4 Estimation Comparison	32
4.2 Mixed Probit Estimation	34

4.2.1 <i>rmxp</i> Function for Mixed Probit Estimation	34
4.2.2 Mixed Probit sample data Estimation	36
Chapter 5 User Interface for Data Structure Transformation	37
5.1 Design of the User Interface	37
5.2 R package Comparison for Mixed Logit Model	41
5.3 Simulated dataset for Mixed Probit Estimation	44
Chapter 6 Conclusion	46
Appendix A - Simulated Data Estimation in R	48
Part I - Mixed Logit Estimation	48
1. bayesm package	49
2. ChoiceModelR package	53
3. RSGHB package	57
Part II - Mixed Probit Estimation	64
Appendix B – Additional R code	67
References	80

List of Figures

Figure 1. Trace plot of bayesm package.....	33
Figure 2. Trace plot of ChoiceModelR package.....	33
Figure 3. Trace plot of RSGHB package	33
Figure 4. Trace plot of Mixed Probit estimation using <i>rmxp</i> function.....	36
Figure 5. Trace plot of bayesm package.....	43
Figure 6. Trace plot of ChoiceModelR package.....	43
Figure 7. Trace plot of RSGHB package	43
Figure 8. Trace plot of <i>rmxp</i> function.....	45
Figure 9. R code for transforming data structure into bayesm format	49
Figure 10. R code for model estimation using bayesm package	50
Figure 11. R code for obtaining estimation results using bayesm package	51
Figure 12. R code for transforming input data into ChoiceModelR format	53
Figure 13. R code for model estimation using ChoiceModelR package	54
Figure 14. R code for obtaining estimation results using ChoiceModelR package	55
Figure 15. R code for transforming data structure into RSGHB format	57
Figure 16. R code for variable specifications in RSGHB package.....	58
Figure 17. R code for control parameter specifications in RSGHB package	60
Figure 18. R code for Likelihood Function specification in RSGHB package	61
Figure 19. R code for model estimation in RSGHB package.....	63
Figure 20. R code for model estimation using <i>rmxp</i> function.....	64
Figure 21. R code for obtaining estimation results using <i>rmxp</i> function	65

List of Tables

Table 1. Metropolis-Hasting algorithm.....	16
Table 2. Gaussian Random-walk Metropolis (Rossi <i>et al.</i> , 2006).....	16
Table 3. General steps of a Gibbs Sampler	17
Table 4. Generic names for input arguments	23
Table 5. Input data structure for bayesm package	25
Table 6. Model estimation setting using <i>bayesm</i> package.....	25
Table 7. Input data structure for ChoiceModelR package.....	27
Table 8. Model estimation setting using ChoiceModelR package	27
Table 9. Example code of using ChoiceModelR package.....	28
Table 10. Input data structure for RSGHB package.....	29
Table 11. R code for model estimation setting in RSGHB package	30
Table 12. Output files of RSGHB package	32
Table 13. Estimated posterior means using different R packages	32
Table 14. Input data structure for <i>rmxp</i> function	35
Table 15. Model estimation setting using <i>rmxp</i> function.....	35
Table 16. Estimated posterior means of Mixed Probit Model	36
Table 17. Input arguments required by the <i>userin</i> function.....	37
Table 18. Functions to transform effect coded design matrix into “level” format	39
Table 19. R package comparison criteria	40
Table 20. Functions for estimation results comparison	40
Table 21. Simulated Mixed Logit dataset setup	41
Table 22. Estimated posterior means using different R packages	42
Table 23. Precision criteria for Mixed Logit estimation	44
Table 24. Simulated Mixed Probit dataset setup	44
Table 25. Estimated posterior means using <i>rmxp</i> function	45
Table 26. Precision criteria Mixed Probit estimation	45

List of Symbols

The list of symbols contains the most important symbols used in this thesis.

List of Symbols	
U_{nj}	<i>utility</i> function of decision maker n chooses alternative j
V_{nj}	<i>representative utility</i>
$\varepsilon'_n = \langle \varepsilon_{n1}, \dots, \varepsilon_{np} \rangle$	vector for the unobserved factors ε_n in the <i>utility</i> function
Ω	covariance matrix of unobserved factors ε_n
P_{ni}	choice probability of decision maker n chooses alternative i
$Y = \{y_1, \dots, y_N\}$	observed choice for N different decision makers
β	vector of fixed coefficients
β_n	vector of random coefficients
b	mean vector of the random coefficients β_n
Σ_β	covariance matrix of the random coefficients β_n
θ	general symbol for model parameters
p	number of alternatives
$nset$	number of choice sets
$nlgt$	number of respondents
$nbeta$	number of model parameters
x	the design matrix of the attributes
y	this matrix contains the responses made by the respondents
MAE_β	mean absolute error
$RMSE_\beta$	root mean squared error
$RMSE_p$	root mean squared prediction error

Chapter 1 Introduction

In the marketing research field, understanding customer preferences for certain products and making statistical inferences on the choice behavior of individuals are essential (McFadden, 1974). Discovering insight from the complex customer responses has been an interesting topic to the researchers, however, performing such analysis can be complicated since some factors which influence the behaviors of the decision makers are unable to be observed. The *Mixed Probit Model* and *Mixed Logit Model* which both belong to the *Discrete Choice Models* family are usually used to model the choice behavior of different individuals, and they have played an important role in economics, transportation, and social science fields for the past few decades.

The Bayesian inference is one of the most popular approaches for the estimation of the *Discrete Choice Models*. The implementation of Bayesian method was originally considered to be difficult, but starting from the early 1990's, the increase of computational efficiency and the development of the *MCMC* methods make the implementation of Bayesian inference much easier. In a Bayesian framework, the *Discrete Choice Model* can be formulated in a flexible way which takes a hierarchical form to represent the choice behaviors of different respondents. A generic name is usually used when a *Discrete Choice Model* is estimated based on a Bayesian approach, namely, *Hierarchical Bayes Model*. (Allenby, Rossi & McCulloch, 2005)

The *Hierarchical Bayes Models* are usually complex, but luckily certain software tools can be used for the model estimation. In particular, the Bayesian estimation algorithms of these models have been implemented in various R packages. The purpose of this thesis is to explore the current available R Packages which can be used to estimate the *Discrete Choice Models* based on a Bayesian approach. The thesis is concentrating on the estimation of the *Mixed Probit* and *Mixed Logit Models*.

This thesis consists of 6 Chapters. **Chapter 2** provides a brief theoretical introduction to the *Discrete Choice Models*. In **Chapter 3**, a general discussion about how to estimate the *Discrete Choice Models* using *Bayesian Inference* and *MCMC techniques* is provided. **Chapter 4** of this thesis provides an overview of the existing R packages. For *Mixed Logit Model* estimation, *bayesm* (Rossi & McCulloch, 2005), *ChoiceModelR* (Seramas, 2012), and *RSGHB* (Dumont *et al.*, 2013) packages are discussed. For *Mixed Probit Model* estimation, since no R

package is currently available, an R function implemented in Gillbride&Allenby (2004) is modified, and the resulting *rmxp* function is used for the estimation of the *Mixed Probit Model*. Since the data structures required by the aforementioned R packages/function are different, for the convenience of the researchers, a user friendly interface is implemented in R. This user interface could be used to transform a given dataset into different formats which are suitable for different R packages. **Chapter 5** provides a detailed explanation and illustration about how to use this user interface. The accuracy of the estimation results obtained from different R packages is also analyzed in this Chapter. Finally, the thesis is concluded in **Chapter 6**.

Chapter 2 Discrete Choice Models

A *Discrete Choice Model* is very commonly used to analyze the choice made among various alternatives from a *choice set* assuming the decision makers want to maximize their *utility* (Ben-Akiva & Boccara, 1995). A *choice set* is a set of the alternatives that a decision maker faces, and the *choice set* of a *Discrete Choice Model* contains finite number of alternatives (Bierlaire, 1997). The characteristics of each alternative are described by its *attributes*. The decision maker must choose one and only one from the available alternatives when making decisions. According to Bierlaire (1997), “*the alternative with the highest utility is supposed to be chosen.*”

The *utility* introduced in the context of the *Discrete Choice Model* could be considered as the “net benefit” a decision maker would get after making a choice (Train, 2009). The *Discrete Choice Model* is also called a *Random utility Model (RUM)* since this model is based on “*random utility maximization*” (McFadden & Train, 2000). The *utility* in the *Discrete Choice Model* is treated as a random variable, and the *utility* function consists of an observable part and an unobservable part. Given a sample of N independent decision makers, labeled as $n=1, \dots, N$, and let $j = 1, \dots, p$ be the choice of decision maker n among p alternatives the *utility* function is defined in Equation (2.1):

$$U_{nj} = V_{nj} + \varepsilon_{nj} \quad (2.1)$$

In the *utility* function (2.1), V_{nj} represents the observable part which is called the *representative utility*. The *representative utility* is usually assumed to be linear in the parameters and it is defined as $V_{nj} = \beta' x_{nj}$, where β are the parameters which need to be estimated, x_{nj} is a vector containing all the *attributes* of the alternatives. The ε_{nj} term in the *utility* function is unobservable, and it captures the unobserved factors that affect the decision maker's choice.

Furthermore, define the *choice probability* as the probability that person n chooses alternative i , this is actually the probability of the *utility* of alternative i is greater than the *utilities* of the rest of the alternatives, and it is defined in Equation (2.2):

$$P_{ni} = \text{Prob}(V_{ni} + \varepsilon_{ni} > V_{nj} + \varepsilon_{nj} \forall j \neq i) \quad (2.2)$$

Different types of *Random Utility Models* could be obtained by making different assumptions about V_{nj} and ε_n terms. For instance, by assuming ε_n to be a normal distribution, a *Probit Model* could be obtained; and if ε_n is assumed to be independently, identically distributed extreme value, and then the resulting *RUM* is a *Logit Model* (Bierlaire, 1997). Moreover, in addition to the different assumptions made for the error term, if the *representative utility* is also assumed to be distributed as certain types of distributions, a *Mixed Probit/Mixed Logit Model* is obtained.

There are certain issues with the identifications of *Discrete Choice Models*. The alternative with the highest *utility* does not change if a constant is added to the *utility* of all alternatives. Additionally, multiplying each alternative's by a constant also does not change the decision maker's choice (Train, 2009). In order to deal with the identification issue, normalization is usually required. The identification problem of the *Probit Model* is discussed in section 2.1.1. For a *Logit Model*, the identification issue is already solved by the restrictive assumption made for the error terms. The error terms for the *Logit Model* are assumed to be independently, identically distributed extreme value with a variance of $\pi^2/6$, and this assumption implies that the normalization has automatically applied to the *Logit Model*.

2.1 Probit Model

The *Probit (Probability Unit) Model* was first derived by Thurstone (1927), and later on Marschak (1960) started to use the *Probit Model* in economic field by introducing the latent *utility* variable. The *Probit Model* is based on the assumption that the unobserved factors ε_n follow a multivariate normal distribution with a mean vector of zero and the covariance matrix Ω which allows pattern of correlations. $\varepsilon'_n = \langle \varepsilon_{n1}, \dots, \varepsilon_{np} \rangle$ represents the vector for the unobserved factors ε_n .

The *utility* could be decomposed as:

$$U_{nj} = V_{nj} + \varepsilon_{nj}, \forall j, \text{ where } \varepsilon_n \sim N(0, \Omega) \quad (2.3)$$

In Equation (2.3), the density of ε_n is $\varphi(\varepsilon_n) = \frac{1}{2\pi^{p/2}|\Omega|^{1/2}} e^{-\frac{1}{2}\varepsilon'_n\Omega^{-1}\varepsilon_n}$

The choice probability for a *Probit Model* is:

$$\begin{aligned} P_{ni} &= \text{Prob} (V_{ni} + \varepsilon_{ni} > V_{nj} + \varepsilon_{nj} \forall j \neq i) \\ &= \int I(V_{ni} + \varepsilon_{ni} > V_{nj} + \varepsilon_{nj} \forall j \neq i) \varphi(\varepsilon_n) d\varepsilon_n \end{aligned} \quad (2.4)$$

Where $I(\cdot)$ in equation (2.4) is an indicator function that takes the value 1 if the statement is true, otherwise this indicator function equals 0.

The fact that the choice probability of a *Probit Model* does not have a closed form causes difficulties for the model estimation (Melvyn Weeks, 1997). In order to estimate the choice probability of a *Probit Model*, *GHK*¹ simulations are usually required. However, the *GHK* simulation procedure is beyond the scope of this thesis so it is not discussed in detail.

2.1.1 Limitation of Probit Model

One limitation of a *Probit Model* is that the model is unable to handle *random taste variation* since the parameters β in this model are assumed to be fixed. Intuitively speaking, different people value the attributes of the alternatives differently, and this could be described as *taste variation*. Dangazo (1979) discussed two types of *taste variations*: *systematic taste variation* and *random taste variation*. Sometimes the tastes of the respondents differ because of their demographic attributes (Train, 2009), this type of taste variation is referred to *systematic taste variation*. However, the taste of each decision maker might also be different “just because different people are different” (Train, 2009), thus it is unable to be predicted. According to Dangazo (1979), such type of taste variation is called *random taste variation*, or simply *taste variation*. In order to incorporate the *random taste variation* in the model, a *Mixed Probit Model* can be used by assuming the coefficients to be random over respondents.

2.1.2 Identification Problem of Probit Model

The identification problem of the *Probit Model* has always been a well known issue, because in a *Probit Model*, some of the parameters are *unidentified* such that they cannot be estimated. In 1985, Dansie pointed out that there are two reasons which lead to the identification problem of the *Probit Model* (Monfardini & Silva, 2006). The first reason is “*only differences in utility matter*” (Train, 2009),

¹ The *GHK* simulator is developed by Geweke(1989,1991), Hajivassiliou(1998), and Keane(1990,1994).

which means that the choice made by each decision maker won't be affected by adding a constant to all the *utilities*, and therefore, only the difference between the *utilities* could be identified. The second reason is that the alternative with the highest *utility* still remains the same if the *utility* of all the alternatives is multiplied by a constant, scaling the *utility* will not change the log-likelihood function (Monfardini & Silva, 2006). In order to deal with the identification issue of the *Probit Model*, normalization is needed.

There are various ways of normalizing a *Probit Model*, but under most of the situations, it is difficult for the researchers to find out which parameters in the model are unidentified. Usually two methods can be used in the specification of the *Mixed Probit Model* to deal with this problem. One method is to restrict the covariance matrix of the error terms as an identity matrix, and the other method is to use a *differenced system*.

A *differenced system* can be obtained by differencing the original system with respect to one reference alternative category, and this approach can always be used to ensure that the parameters of the *Probit Model* are identified in any cases. Suppose in a choice set, alternative p is chosen to be the reference category, then the *utilities* can be expressed in terms of differences by subtracting alternative p from the rest of the alternatives in the choice set:

$$W_{ij} = U_{ij} - U_{ip} \quad (2.5)$$

If the original structure of the attribute matrix is $X_i = \begin{bmatrix} x'_{i1} \\ \vdots \\ x'_{ip} \end{bmatrix}$, then the *differenced system* can be expressed as $X_i^d = \begin{bmatrix} x'_{i1} - x'_{ip} \\ \vdots \\ x'_{i,p-1} - x'_{ip} \end{bmatrix}$, and the covariance matrix of the previously defined model becomes $e_{ij} = \varepsilon_{ij} - \varepsilon_{ip}$.

Therefore, the difference between *utilities* $\mathbf{W}_i = (W_{i1}, \dots, W_{i,p-1})$ takes the following form:

$$\mathbf{W}_i = X_i^d \boldsymbol{\beta} + e_i, e_i \sim N(0, \tilde{\boldsymbol{\Omega}}) \quad (2.6)$$

In equation (2.6), X_i^d contains *choice-specific* variables which represent the attributes of the alternatives, and it has dimension of $(p - 1) \times k$, where k is the number of attributes. The dimension of $\boldsymbol{\beta}$ is a $k \times 1$ vector of the coefficients. $\tilde{\boldsymbol{\Omega}}$ is $(p - 1) \times (p - 1)$ positive definite matrix. In order to obtain the identified parameters, normalization with respect to σ_{11} which is the first diagonal element

of $\tilde{\Omega}$ is usually applied. As a result, the set of parameters ($\tilde{\beta} = \frac{\beta}{\sqrt{\sigma_{11}}}$, $\tilde{\Omega}^* = \tilde{\Omega} / \sigma_{11}$) are identifiable.

Let Y_i be the outcome variable which represents the choice that decision maker i made, Equation (2.7) is a representation of Y_i in terms of W_i :

$$Y_i(W_i) = \begin{cases} 0, & \text{if } \max(W_i) < 0 \\ j, & \text{if } \max(W_i) = W_{ij} > 0 \end{cases}, \text{ for } i = 1, \dots, n, \text{ and } j = 1, \dots, p-1 \quad (2.7)$$

For the estimation algorithm of the *Mixed Probit Model* discussed in this thesis, an identity matrix is used as the covariance matrix of the error terms in the model specification. However, for the estimation of simulated *Mixed Probit* data, the dataset is simulated using the *differenced system* of the x matrix since the function provided in *bayesm* package for the data simulation requires the x matrix to be entered as a *differenced system*. Detailed explanations can be found in section 3.5 and 5.3.

2.2 Mixed Probit Model

A *Probit Model* can be extended to a *Mixed Probit Model* by including the random effects. In a *Mixed Probit Model*, the coefficients are assumed to be randomly distributed so that different tastes for each decision maker can be captured. The *Mixed Probit Model* deals with the random effect very well, especially, when the coefficients β_n are normally distributed. However, the density function of β_n it is not restricted to normal distributions, it can be *lognormal*, *uniform*, *triangular*, *gamma*, etc.

Let β_n represents the tastes of decision maker n , the utility could then be expressed as: $U_{nj} = \beta_n' x_{nj} + \varepsilon_{nj}$. Suppose $\beta_n \sim N(\mathbf{b}, \Sigma_\beta)$, then the parameters which need to be estimated are the mean \mathbf{b} and the covariance Σ_β of β_n .

The density function $f(\beta_n)$ is the mixing distribution which assigns weights for different values of β_n . As stated before, β_n is unknown to the researcher, therefore, the *unconditional* choice probability for *Mixed Probit Model* should be integrated over the distribution of β_n , and it can be expressed as follows:

$$P_{ni} = \int \int I(\beta_n' x_{ni} + \varepsilon_{ni} > \beta_n' x_{nj} + \varepsilon_{nj} \forall j \neq i) \varphi(\varepsilon_n) f(\beta_n) d\varepsilon_n d\beta_n \quad (2.8)$$

The *Mixed Probit Model* has not been used as widely as a *Mixed Logit Model* because the fact that the unconditional choice probability of the *Mixed Probit Model* does not have a closed form and the identification problem cause many computational difficulties. In addition, since the *Mixed Logit Model* “can approximate any random utility model” (Train, 2009) the researchers tend to prefer choosing a *Mixed Logit Model* for the modeling of the *Discrete Choice* data.

2.3 Logit Model

The *Logit Model* was developed by Marley (1965) and McFadden (1974) based on the derivation of the *Logit Formula* by Luce (1959). Such type of *Discrete Choice Model* is considered to be a “very fruitful innovation” (Cramer, 2003), and it has been the most popular *Discrete Choice Model*. Unlike the *Probit Model*, the choice probability of the *Logit Model* has a closed form, and the interpretation of the *Logit Model* is also relatively easier (Train, 2009).

The *Logit Model* assumes that each unobserved factor ε_{nj} is independently, identically distributed extreme value (also called *Gumbel* or *type I extreme value*). The tail of the extreme value distribution is slightly fatter than the normal distribution, and the assumption that the error terms of a *Logit Model* are independent implies that “the unobserved portion of utility for one alternative is unrelated to the unobserved portion of utility for another alternative.” (Train, 2009), and all the alternatives have the same variances.

Using the same notations that have been introduced in the previous sections, the choice probability for the *Logit Model* is:

$$P_{ni} = \int \left(\prod_{j \neq i} e^{-e^{-(\varepsilon_{ni} + V_{ni} - V_{nj})}} \right) e^{-\varepsilon_{ni}} e^{-e^{-\varepsilon_{ni}}} d\varepsilon_{ni} \quad (2.9)$$

The density for ε_{nj} in Equation (2.9) is $f(\varepsilon_{nj}) = e^{-\varepsilon_{nj}} e^{-e^{-\varepsilon_{nj}}}$

Since the *representative utility* is $V_{nj} = \beta' x_{nj}$, the closed form expression for *logit* probabilities in Equation (2.9) becomes:

$$P_{ni} = \frac{e^{\beta' x_{ni}}}{\sum_j e^{\beta' x_{nj}}} \quad (2.10)$$

The *Logit Model* has three major limitations: 1) unable to incorporate random taste variation; 2) IIA property; 3) unable to handle data when the errors are temporarily correlated (Train, 2009). These limitations are discussed in **section 2.3.1**.

2.3.1 Limitations of Logit Model

i) Taste Variation

According to the previous discussion at section 2.1.1 about the *random taste variation*, similar to a *Probit Model*, same limitation also applies to a *Logit Model*. A *Logit Model* is only able to deal with the *systematic taste variation*, it does not allow the *random taste variation* since the model parameters β are fixed.

ii) The Property of Independence from Irrelevant Alternatives (IIA)

The *Independence from Irrelevant Alternatives* (IIA) axiom was published by Luce in 1959, and Daganzo (1979) described the *Property of Independence from Irrelevant Alternatives* (IIA) as “*The relative probability of choice of two alternatives depends only on their measured attractiveness.*” For any two alternatives i and j , the relative odds of choosing i and j is independent from alternatives other than them (*irrelevant* alternatives). However, sometimes exhibiting such property is an advantage of the *Logit Model* since it somehow reflects the reality. For example, with IIA property, the model estimation could be performed using a subset of the alternatives, so it is quite useful when there are a large number of alternatives or if the researcher only would like to focus on a subset of alternatives. But in many other situations, such property might not be appropriate (Train, 2009).

iii) Panel Data

Sometimes the researchers are interested in observing how changes in attributes will affect the choice of the respondents. In order to make the observation, every time a set of alternatives with different attributes might be given to the respondent to make a choice, and the researcher might ask the respondent to choose from several sets of alternatives. Such type of repeated data is called

Panel Data. The *Logit Model* has the strict assumption regarding to the independence of the errors, therefore, if the unobserved factors which affects the choice of the respondent are correlated over time, the *Logit Model* could not be used. (Train, 2009)

The three aforementioned limitations of a *Logit Model* could be handled by either using a *Mixed Probit Model* or a *Mixed Logit Model*. For a *Mixed Probit Model*, first of all, it incorporates the *random taste variation*; secondly, the IIA property can be avoided by either using the full covariance matrix or imposing different covariance matrices for the error terms. Finally, for *Panel Data* analysis, the error terms of the *Mixed Probit Model* are also expected to be correlated over time. The discussion of the *Mixed Logit Model* can be found in the proceeding section.

2.4 Mixed Logit Model

The applications using *Mixed Logit Models* becomes increasingly popular since 1990s (Hensher & Greene, 2002). A *Mixed Logit Model* is also called a *Random Coefficient Logit Model* (Mcfadden & Train, 2000) since the coefficients β_n in this type of model are assumed to be random across different decision makers. Thus the *Mixed Logit Model* solves the first limitation of the standard *Logit Model* (section 2.3.1) by allowing *random taste variations*.

The *Mixed Logit Model* is a model in which the choice probability is expressed as a mixture of the logit function evaluated at different β_n with the density function $f(\beta_n)$ (Glasgow, 2001). The density function $f(\beta_n)$ is the mixing distribution which assigns weights for different values of β_n , and in general, $f(\beta_n)$ can be both discrete and continuous, but for most of the applications, it is specified to be continuous. This thesis only focuses on the case when β_n is normally distributed.

If the value of β_n is known, then the probability *conditional* on β_n :

$$L_{ni}(\beta_n) = \frac{e^{\beta_n' x_{ni}}}{\sum_{j=1}^p \beta_n' x_{nj}} \quad (2.11)$$

It can be seen that Equation (2.11) is the same as a *Logit Model*. Furthermore, the *unconditional* choice probability for *Mixed Logit Model* integrated over all the possible values of β_n is as follows:

$$P_{ni} = \int \left(\frac{e^{\beta_n' x_{ni}}}{\sum_j e^{\beta_n' x_{nj}}} \right) f(\beta_n) d\beta_n \quad (2.12)$$

There are two sets of parameters in a *Mixed Logit Model*. One set of parameters is the random coefficients β_n ; and since β_n follows a distribution with mean b and covariance Σ_β , the parameters b and Σ_β are considered to be another set of parameters which describe the mixing density $f(\beta_n)$. Let θ be the set of parameters b and Σ_β , a proper expression of the choice probability for the *Mixed Logit Model* should be written as follows:

$$P_{ni} = \int \left(\frac{e^{\beta_n' x_{ni}}}{\sum_j e^{\beta_n' x_{nj}}} \right) f(\beta_n | \theta) d\beta_n \quad (2.13)$$

The *unconditional* choice probability is a function of θ which does not depend on β_n since the integral is evaluated with respect to β_n .

Unlike a standard *Logit Model*, the *Mixed Logit Model* does not exhibit IIA property. When calculating the ratio P_{ni}/P_{nj} for a *Mixed Logit Model*, the denominator from equation (2.13) does not cancel out due to the existence of the integral, therefore, the relative odds of choosing i and j is not independent from alternatives other than them.

Additionally, when dealing with *Panel Data*, the *Mixed Logit Model* also allows the unobserved random coefficients β_n to be correlated over time. For example, let $t = 1, \dots, T$ represent different time period, if the tastes of each individual are correlated over time, the following expression can be used for the definition of the *utility*:

$$\begin{aligned} U_{nj} &= \beta_{nt}' x_{nj} + \varepsilon_{nj} \\ \beta_{nt} &= b + \tilde{\beta}_{nt} \\ \tilde{\beta}_{nt} &= \rho \tilde{\beta}_{nt-1} + \mu_{nt} \end{aligned} \quad (2.14)$$

where b is fixed and μ_{nt} is iid over n and t

Both *Mixed Probit* and *Mixed Logit Models* can be estimated using a Bayesian approach. In the next Chapter of this thesis, the basic concepts and steps for the Bayesian estimation of *Mixed Probit* and *Mixed Logit Models* are introduced.

Chapter 3 Estimation of Bayesian Hierarchical Models

The *Hierarchical model* is a combination of sub-models which can be expressed in a hierarchical form. Such type of model is usually written in levels, and it is very useful to deal with the heterogeneity between the decision makers in marketing research problems. For example, if the *utility* function is $U_{nj} = \beta'_n x_{nj} + \varepsilon_{nj} \forall j$, then the hierarchical representation of the *Discrete Choice Model* can be expressed as different levels of conditional distributions:

$$Y|U_n \quad (3.1a)$$

$$U_n|\beta'_n x_n \quad (3.1b)$$

$$\beta_n|b, \Sigma_\beta \quad (3.1c)$$

From the set of conditional distributions (3.1a)-(3.1c), (3.1c) represents the random coefficients β_n of the respondents which can be described by a certain distribution with mean b and covariance Σ_β . (3.1b) represents the choice probabilities conditioned on the random coefficients, and this level usually follows a *Probit* or *Logit Model*. Finally, $Y|U_n$ (3.1a) indicates the choice made by the respondent given the *utility*.

A Bayesian approach in conjunction with *MCMC* method is usually suitable for the estimation of a *Hierarchical model*, and the *Hierarchical Models* estimated using the Bayesian procedures is thus called the *Bayesian Hierarchical Models/Hierarchical Bayes Models*. The *Hierarchical Bayes Model* is particularly useful for making inferences of the *posterior distributions* of unit-level parameters, the unit-level parameters can be estimated based on the distribution of the random coefficients and the information from the observed data. Furthermore, by combining the *MCMC* methods in the Bayesian framework, the model estimation becomes simpler (Allenby, Rossi, McCulloch, 2005). The main focus of this thesis is the estimation of *Bayesian Hierarchical Logit Models* and *Bayesian Hierarchical Probit Models*.

3.1 General Concepts of Bayesian Inference

The main idea behind a Bayesian procedure is to make inferences for the *posterior distribution* given the sample and the prior information. It utilizes *Bayes Theorem*, in other words, conditional probability. There are three main components used in a Bayesian framework: 1) *Prior Distribution*; 2) *Likelihood Function*; 3) *Posterior Distribution*. In the Bayesian framework, the information obtained from the observed data is combined with the researcher's prior knowledge (represents by a *Prior distribution*), the combination of this information can be summarized using a *Posterior distribution* (Robert & Casella, 2004).

The *Prior Distribution* can be considered as the researcher's knowledge about the unknown model parameters θ , which describes the uncertainty regarding to the unknown model parameters before looking at the data. Two types of *prior distribution* are often used in a Bayesian framework, an *informative prior* and a *non-informative prior*. (Glickman & van Dyk, 2007) An *informative prior distribution* reflects that the researchers have some previous knowledge about the unknown model parameters. Usually this prior knowledge is obtained based on observing previous data. A *Non-informative Prior Distribution* is also called *vague/flat/diffuse prior*. It is often used when the researchers do not have much previous knowledge about the unknown model parameters, and usually the researchers use this type of *prior distribution* as default (Glickman & van Dyk, 2007).

In a Bayesian Framework, the information obtained from the observed data is represented by a joint probability function called a *likelihood function*, which is a function of the model parameters (Glickman & van Dyk, 2007). For a *Discrete Choice Model*, let $\mathbf{Y} = \{y_1, \dots, y_N\}$ be the observed choice for N different decision makers. If the probability of decision maker n choosing y_n is $P(y_n | \theta)$, the probability of observing the choice set \mathbf{Y} can be expressed by the *likelihood function* (3.2):

$$L(\mathbf{Y}|\theta) = P(y_1, \dots, y_n|\theta) = \prod_{n=1}^N P(y_n|\theta) \quad (3.2)$$

The *Posterior Distribution* is the distribution of the model parameters after observing the data. It reflects the information after combining the observed data

and the *prior distribution* together, which can be viewed as the updated information about θ after observing the data.

Let the *Prior Distribution* about model parameter θ be $k(\theta)$, and denote the *Posterior Distribution* by $K(\theta|Y)$ (Train, 2009). According to *Bayes' Theorem*:

$$\begin{aligned} K(\theta | Y) &= \frac{L(Y | \theta)k(\theta)}{L(Y)} \\ &= \int L(Y | \theta)k(\theta) d\theta, \text{ where } L(Y) \end{aligned} \quad (3.3)$$

In Equation (3.3), $L(Y)$ is a marginal distribution over θ , therefore, it could be considered as a constant. As a result, the *Posterior density* is proportional to the product of the *likelihood function* and the *prior density*, which takes the following form:

$$K(\theta | Y) \propto L(Y | \theta) k(\theta) \quad (3.4)$$

After determining the *Posterior Distribution*, inference with respect to the mean of the *Posterior Distribution* can be made. The *posterior mean* can be calculated using Equation (3.5):

$$\bar{\theta} = \int \theta K(\theta|Y) d\theta \quad (3.5)$$

In general, the inference of the *posterior mean* is done by performing simulations instead of directly calculating the integral. The simulated *posterior mean* can be obtained by taking average of the draws of the model parameter θ from the *Posterior Distribution*. If the simulation performed R draws, and θ^r is the r^{th} draw from the *Posterior Distribution*, then the simulated mean can be obtained using Equation (3.6) (Train, 2009):

$$\bar{\theta} = \frac{1}{R} \sum_{r=1}^R \theta^r \quad (3.6)$$

This section only provides a general introduction about the Bayesian procedure. More detailed explanations of the Bayesian inference could be found in Robert & Casella (2004) and Gelman *et al.* (2004).

3.2 Monte Carlo Markov chain Methods

Although the *Posterior Distribution* could be solved by simulating an approximation of the integral, usually the form of the *Posterior Distribution* is not very easy to make draws directly (Train, 2009). In order to overcome this problem, the *Monte Carlo Markov chain (MCMC)* technique is often used for simulating the *Posterior Distribution*. The *MCMC* methods are very useful especially when dealing with complex models.

The *Monte Carlo* method is a sampling method used in Bayesian estimation. It allows the researchers to repeatedly draw from the *Posterior Distribution* to summarize the information of the unknown model parameters θ . A *Markov chain* is invented by a Russian mathematician Andrey Markov (1856-1922) which was used primarily in physics. In a Bayesian context, it is used in conjunction with the *Monte Carlo* algorithm for the simulation of the *Posterior distribution* (Jackman, 2009). It might take some initial draws before the chain reaching equilibrium (Rossi *et al.*, 2005), this period is called a *burn-in period*. The draws made during the *burn-in period* are often discarded, and the simulations after the *burn-in period* can be used for making inferences about the *Posterior Distribution*. *Metropolis-Hasting algorithm* and *Gibbs sampler* are two major *MCMC* methods which are mainly used for Bayesian Model Estimation. *Gibbs sampling* is a special case of *Metropolis-Hasting algorithm* (Gelman, 1992) which allows the draws to be made sequentially. These two estimation methods are discussed in subsections 3.2.1 - 3.2.3.

3.2.1 Metropolis-Hasting algorithm

In simple terms, let θ represent the unknown model parameters, the *Metropolis-Hasting* algorithm moves from a starting point θ^0 to a new value θ^* which is drawn from a proposal distribution $q_t(\theta^*|\theta^0)$, and θ^* is evaluated relative to θ^0 to decide if the new value θ^* is kept. The evaluation is done based on an *acceptance ratio* r , where $r = \frac{p(\theta^*|Y)q_t(\theta^{(t-1)}|\theta^*)}{p(\theta^{(t-1)}|Y)q_t(\theta^*|\theta^{(t-1)})}$. If the new value θ^* is kept, then for the next iteration, the starting value becomes θ^* . The iteration can be repeated many times. **Table 1** displays a general *Metropolis-Hasting* algorithm.

Table 1. Metropolis-Hasting algorithm

1. Start with $\theta^{(t-1)}$
2. sample θ^* from the proposal distribution $q_t(\theta^*, \theta^{(t-1)})$
3. sample $U \sim \text{Unif}(0,1)$
4. $\theta^{(t)} = \begin{cases} \theta^*, & U \leq \alpha \\ \theta^{(t-1)}, & \text{otherwise} \end{cases}$, where $\alpha = \min(r, 1)$

The choice of the proposal distribution affects the acceptance ratio and the resulting Markov chain. Particularly, a *Random-walk* is a popular choice to generate the proposal distribution.

3.2.2 Random-walk Metropolis Algorithm

A *Random-walk Metropolis algorithm* (see **Table 2**) is usually considered to be a generic *Metropolis* Algorithm which can be used in most of the situations (Robert&Casella, 2010). A *Random Walk* is proposed in this algorithm such that the candidate $\theta^* = \theta^{(t)} + \varepsilon$. ε is a random perturbation around $\theta^{(t)}$, and it is independent of $\theta^{(t)}$. The proposal density $q(\theta^*|\theta^{(t)})$ in this case becomes $q_\varepsilon(\theta^* - \theta^{(t)})$.

The *acceptance ratio* becomes:

$$r = \frac{p(\theta^*|Y)}{p(\theta^{(t)}|Y)}$$

Table 2. Gaussian Random-walk Metropolis (Rossi et al., 2006)

1. Start with $\theta^{(t-1)}$.
2. Draw $\theta^* = \theta^{(t)} + \varepsilon, \varepsilon \sim N(0, \Omega)$
3. sample $U \sim \text{Unif}(0,1)$
4. $\theta^{(t)} = \begin{cases} \theta^*, & U \leq \alpha \\ \theta^{(t-1)}, & \text{otherwise} \end{cases}$, where $\alpha = \min(r, 1)$

Repeat, as necessary.

This algorithm has drawbacks because the resulting Markov chain can have serial autocorrelations (Jackman, 2009), and the required number of iterations during the simulation can be quite large (Robert & Casella, 2010).

3.2.3 Gibbs Sampling Algorithm

The *Gibbs sampling* technique can be used when the unknown model parameters θ has a high dimension, and sampling directly from the *Posterior Distribution* $p(\theta|Y)$ is difficult. The *Gibbs sampler* divides the unknown model parameters into a sequence of p subcomponents such that $\theta = (\theta_1, \dots, \theta_p)$, then making draws for each subcomponent of the model parameters from the *Posterior Distributions* conditional on the most recent values of the rest of the parameters rather than directly making draws from $p(\theta|Y)$. Since the unknown model parameters are divided into subcomponents, the number of the model parameters contained in the distribution is smaller, so the dimension of the conditional distribution is much lower, and making draws are relatively easier (Glickman & van Dyk, 2007).

Suppose there are T iterations in total, it takes p steps within each iteration t , and the *prior distribution* of θ_j is updated based on the following conditional distribution:

$$p(\theta_j|\theta_{-j}^{t-1}, Y), \text{ where } \theta_{-j} = (\theta_1, \dots, \theta_{j-1}, \theta_{j+1}, \dots, \theta_p)$$

The *Gibbs sampling* algorithm within one iteration t is shown in **Table 3**.

Table 3. General steps of a Gibbs Sampler

1. Sample $\theta_1^{(t+1)}$ from $p_1(\theta_1|\theta_2^{(t)}, \theta_3^{(t)}, \dots, \theta_p^{(t)}, Y)$
2. Sample $\theta_2^{(t+1)}$ from $p_2(\theta_2|\theta_1^{(t+1)}, \theta_3^{(t)}, \dots, \theta_p^{(t)}, Y)$
3. ...
4. Sample $\theta_p^{(t+1)}$ from $p_p(\theta_p|\theta_1^{(t+1)}, \theta_2^{(t+1)}, \dots, \theta_{p-1}^{(t+1)}, Y)$
5. $\theta^{t+1} = (\theta_1^{(t+1)}, \theta_2^{(t+1)}, \dots, \theta_p^{(t+1)})$

After sampling $\theta_j^{(t+1)}$, it is used to be conditioned on when sampling for $\theta_{-j}^{(t+1)}$.

For example, once $\theta_1^{(t+1)}$ is sampled from step 1, it becomes the conditional argument in step 2 so that $\theta_2^{(t+1)}$ could be sampled. Finally the resulting Markov chain $\theta^{t+1} = (\theta_1^{(t+1)}, \theta_2^{(t+1)}, \dots, \theta_p^{(t+1)})$ has a distribution $p(\theta|Y)$ (Jackman, 2009).

For the estimation of *Discrete Choice Models*, it is quite common to embed the *Metropolis-Hasting* steps within the *Gibbs sampler* when the conditional distribution $p(\theta_j|\theta_{-j}^{t-1}, Y)$ in the *Gibbs sampling* algorithm is not very easy to directly making draws from (Muller, 1991b; Rao, 2003; Gamerman & Lopes, 2006). By using such a method, the performance of the *Gibbs sampler* will be

improved (Besag *et al.*, 1995; Liu, 1996; Jackman, 2009). This technique is also called *Metropolis-Hasting (M-H) within Gibbs sampling* or *hybrid Gibbs sampling*. Instead of sampling directly from the full conditional distribution, a *Metropolis-Hasting* step is used to making draws for the subcomponents of the parameters from a proposal distribution close to $p(\theta_j | \theta_{-j}^{t-1}, Y)$.

3.3 Data Augmentation Technique

Before the discussion of the estimation procedures for *Discrete Choice Models*, a brief explanation about the *Data Augmentation (DA)* technique is provided in this section. The *Data Augmentation* technique was originally introduced by Tanner and Wong (1987) to deal with the missing data issue, and later on this technique is to used to deal with the unobserved latent constructs to make the computation for the *posterior distribution* easier. The general idea of the *Data Augmentation* technique is to allow simulating two conditional distributions at the same iteration, and this technique is usually used together with the *Gibbs sampling* algorithm for the estimation of the *Probit Models*.

As previously stated in **Chapter 2**, the *utilities* are unobserved latent variables used for the construction of *Discrete Choice Models*. The *posterior distributions* for the model parameters can be obtained based on the *prior distributions* and the information about the latent variables, during the iterations of *Gibbs sampling*, updated latent variables can be sampled from the *posterior distribution* obtained previously. When using a *Gibbs sampler* to sample directly from the posterior $p(\theta, U | Y)$ is non-trivial, but sampling from $p(\theta | U, Y)$ and $p(U | \theta, Y)$ is easy, the *Data Augmentation* technique can be used so that the posterior sample $p(\theta, U | Y)$ can be obtained by making draws from $p(\theta | U, Y)$ and $p(U | \theta, Y)$. As a result, the estimation of the *Hierarchical Bayes Model* is simplified.

The *Data Augmentation* technique is particularly useful for the Bayesian estimation of *Mixed Probit Models* since the conditional choice probability of the *Mixed Probit Model* does not have a closed form, making draws for the random coefficients β_n cannot be accomplished directly. For example, using the notations introduced in Equations (3.1a)-(3.1c), when performing the estimation of the *Mixed Probit Models*, the involvement of the *utility* which is an unobserved latent variable is considered to be the *Data Augmentation* step. This *Data Augmentation* step adds the latent *utility* into the set of the model parameters, thus a *Gibbs sampling* can be performed on the set of parameters $(U_n, \beta_n, b, \Sigma_\beta)$ for the model estimation. Conditional on the *utility* U_n , the inference on the

random coefficients β_n is a Bayesian linear regression analysis. Given β_n , the utility U_n can be drawn from a truncated normal distribution. Therefore, by introducing this step in the *Gibbs sampling* algorithm, the estimation becomes easier. The estimation algorithm for the *Mixed Probit Model* using *Data Augmentation* in the *Gibbs sampler* is introduced in section 3.5.

3.4 Bayesian Estimation for Mixed Logit Models

Section 3.4 and 3.5 provide a general outline of the Bayesian Estimation steps for *Mixed Logit* and *Mixed Probit Models* respectively. The algorithm discussed in section 3.4 for *Mixed Logit* estimation is a summary based on the estimation procedures provided in Train (2009), and this algorithm is implemented in the *RSGHB* package.

A diffused prior is used on b such that b is normally distributed with a sufficiently large variance. The prior on Σ_β is Inverted Wishart denoted by $IW(v, vI)$, where v is the degrees of freedom and I is the identity matrix.

Using the same notation introduced in section 2.4 for the *Mixed Logit Model*. For every iteration during the model estimation, draws of b , Σ_β and β_n are made. The *posteriors* are easy to simulate for b and Σ_β and the calculations are also very fast. The draws for b are taken from a normal distribution, and Σ_β can be drawn from an Invert Wishart distribution. However, making draws for β_n can be relatively time consuming since the draws are required to be obtained for every decision maker, therefore, *Metropolis-Hasting* is usually used just for the step of making draws for β_n . There are 3 layers in this *Gibbs sampling* algorithm which are specified in step 2-4 from the following algorithm, each layer represents one conditional posterior distribution of the model parameter:

1. Starting with the population mean b^0 , population covariance Σ_β^0 , and the coefficient values for each respondent β_n^0 with any values since it's not sensitive to the starting values.
2. Given b^0 , Σ_β^0 , and the observed data y_n , individual level β_n^1 can be obtained by making draws from the posterior distribution $K(\beta_n^1 | b^0, \Sigma_\beta^0, y_n)$. The draws of β_n^1 are made using *Metropolis-Hasting* for every respondent by the following steps:

- a) Draw K independent values from a standard normal density and stack the draws into a vector labeled η_1
 - b) Calculate a candidate for β_n^1 as $\check{\beta}_{n1} = \beta_n^0 + \rho L \eta_1$. where $\rho L \eta_1$ term can be determined by drawing from a normal proposal distribution with mean equals 0 and variance $\rho^2 \Sigma_\beta$. ρ is a scalar specified by the researcher and L is the Choleski factor of Σ_β .
 - c) Draw a random value μ_1 from a uniform distribution between 0 and 1.
 - d) The posterior of the new $\check{\beta}_n^1$ draw to the previous β_n^0 draw is compared in this step. First, calculate $L(y_n | \check{\beta}_n^1) g(\check{\beta}_n^1 | \mathbf{b}, \Sigma_\beta)$ which represents the probability of person n 's choice given the $\check{\beta}_{n1}$ draw multiplied by the probability of $\check{\beta}_n^1$ draw given \mathbf{b}^0 and Σ_β^0 ; second, calculate $L(y_n | \beta_n^0) g(\beta_n^0 | \mathbf{b}, \Sigma_\beta)$, which is the probability of person n 's choice given the β_n^0 draw multiplied by the probability of β_n^0 draw given \mathbf{b}^0 and Σ_β^0 . The following ratio is obtained:
$$r = \frac{L(y_n | \check{\beta}_n^1) g(\check{\beta}_{n1} | \mathbf{b}, \Sigma_\beta)}{L(y_n | \beta_n^0) g(\beta_n^0 | \mathbf{b}, \Sigma_\beta)}$$
 - e) If μ_1 chosen in step c) is less than the ratio r , then the new $\check{\beta}_n^1$ draw becomes β_n^1 ; otherwise, $\check{\beta}_n^1$ should be discarded, and the previous β_n^0 becomes β_n^1 . In a simple expression:
$$\beta_n^1 = \begin{cases} \check{\beta}_n^1, & \text{if } \mu_1 \leq r \\ \beta_n^0, & \text{if } \mu_1 > r \end{cases}$$
 - f) The process should be repeated as many times as necessary.
3. In step 3, the draws of \mathbf{b}^1 are made from the conditional distribution $K(\mathbf{b}^1 | \beta_n^1, \Sigma_\beta^0, y_n)$, which is its posterior condition on the β_n^1 for every respondent obtained in step 2, and the initial covariance matrix Σ_β^0 since the covariance matrix has not been updated yet, and the observed data y_n . The draws of \mathbf{b}^1 are made from a normal distribution with mean equals to the average of β_n^1 's, which is denoted as $\bar{\mathbf{b}}$, and variance Σ_β^0/N .
 4. In this step, the covariance matrix Σ_β^1 are drawn from the conditional distribution $K(\Sigma_\beta^1 | \beta_n^1, \mathbf{b}^1, y_n)$, which is the posterior distribution conditioned on the β_n^1 for every respondent obtained in step 2, and the mean \mathbf{b}^1 obtained from step 3, and the observed data y_n . This conditional posterior

distribution is a IW distribution with degrees of freedom equal to the prior specified by the researcher v plus the sample size N , and variance

$$(vI + N\bar{S}) / (v + N) \text{ where } \bar{S} = \sum_{n1} (\beta_n^1 - b^1)(\beta_n^1 - b^1)' / N$$

This *Gibbs sampler* can be repeated as many times as necessary.

3.5 Bayesian Estimation for Mixed Probit Models

In general, a *Mixed Probit Model* can also be estimated using *Gibbs sampling* methods, and the draws for \mathbf{b} and Σ_β are same as the steps for *Mixed Logit Models*. However, since that the conditional probability of the *Mixed Probit Model* does not have a closed form, for the estimation of a *Mixed Probit Model*, the utility U_{nj} is also included in the set of parameters during the *Gibbs sampling* algorithm. The following algorithm provides the steps for the estimation of the *Mixed Probit Model* implemented in the *rmxp* function which is introduced in **Chapter 4**. The *rmxp* function is a modification of the R function provided in Gilbride& Allenby (2004). For the *Mixed Probit Model* specification introduced in Gilbride& Allenby (2004), the covariance matrix of the error terms is restricted to an identity matrix, it does not use a *differenced system* for the x matrix.

The model is represented in Equation (3.7):

$$\begin{aligned} P_{nj} &= \Pr ob(U_{nj} > U_{ni} \text{ for all } i), \\ \text{where } U_{nj} &= \beta_n' x_{nj} + \varepsilon_n, \varepsilon_n \sim Normal(0, I) \\ \beta_n &\sim Normal(\mathbf{b}, \Sigma_\beta) \end{aligned} \tag{3.7}$$

The prior specification for the hyperparameters \mathbf{b} and Σ_β are as follows:

$$\begin{aligned} \mathbf{b} &\sim N(0, 100I) \\ \Sigma_\beta &\sim IW(v, vI), \text{ where } v = k + 5, k \text{ is the number of} \\ &\text{model parameters} \end{aligned}$$

Starting with the population mean \mathbf{b}^0 , population covariance Σ_β^0 , and the coefficient values for each respondent β_n^0 and U_n^0 with some initial values:

1. Let y_{nj} be an indicator variable which equals 1 if respondent n chooses alternative j , and equals 0 otherwise. Draw U_n^1 condition on y_n, β_n^0 , for $n = 1, \dots, N$. This step “Gibbs-thru” U_n^1 from a truncated normal distribution:

Starting with $j = 1$,

If $y_{nj} = 1$ then $U_{nj}^1 \sim TN(x_{nj}'\beta_n^0, \sigma_\varepsilon = 1, U_{nj} > U_{ni} \text{ for all } i)$, where TN is a truncated normal distribution.

If $y_{nj} = 0$ then $U_{nj}^1 \sim TN(x_{nj}'\beta_n^0, \sigma_\varepsilon = 1, U_{nj} < U_{ni}, \text{ where } y_{ni} = 1)$,

Else $U_{nj}^1 \sim Normal(x_{nj}'\beta_n^0, \sigma_\varepsilon = 1)$,

increment j and return to top.

2. Draw β_n^1 condition on U_n^1 . This step is a linear Bayesian regression since the *utility* function is defined as $U_{nj} = \beta_n' x_{nj} + \varepsilon_n$, and if U_n^1 is given, then the *utility* function is simply a regression of X_n on U_n^1 . The draws of β_n^1 are made from a normal distribution with mean equals to

$$\left(X_n' X_n + \Sigma_\beta^0 \right)^{-1} \left(X_n' U_n^1 + \Sigma_\beta^0 b^0 \right) \text{ and variance } \left(X_n' X_n + \Sigma_\beta^0 \right)^{-1}$$

3. In this step, the draws of b^1 are made from a normal distribution with mean \bar{b} , where $\bar{b} = \left((\Sigma_\beta^0/N)^{-1} + (100I)^{-1} \right)^{-1} (\Sigma_\beta^0^{-1} \sum_{n=1}^N \beta_n^1 + (100I)^{-1}(0))$, and variance $((\Sigma_\beta^0/N)^{-1} + (100I)^{-1})^{-1} \Sigma_\beta^0/N$.

4. In this step, the covariance matrix Σ_β^1 are drawn from $IW(v + N, vI + \sum_{n=1}^N (\beta_n^1 - b^1)'(\beta_n^1 - b^1))$

The iteration can be repeated as many times as necessary.

Chapter 4 Overview of the Available R Packages

This chapter provides an overview of the available R packages/function which can be used for the *Mixed Logit* and *Mixed Probit* estimations. The R packages such as *bayesm*, *ChoiceModelR*, *RSGHB* can be used to estimate the *Mixed Logit Model*. For *Mixed Probit Model* estimation, a function called *rmxp* is developed by modifying the R code provided by Gillbride & Allenby (2004).

Section 4.1 discusses the functionalities and the required data structures of the *bayesm*, *ChoiceModelR*, and *RSGHB* packages respectively. **Section 4.2** introduces the *rmxp* function for *Mixed Probit* estimation.

For the convenience of this thesis, the following generic input arguments displayed in **Table 4** are going to be used for all the packages. For different R packages/function, the required formats for *x* matrix and *y* matrix are different, detailed explanations of the required data structure are provided in the **Input Data Structure** part for each subsection.

Table 4. Generic names for input arguments	
<i>p</i>	number of alternatives
<i>nset</i>	number of choice sets
<i>nlgt</i>	number of respondents
<i>nbeta</i>	number of model parameters
<i>x</i>	The design matrix of the attributes
<i>y</i>	This matrix contains the responses made by the respondents

4.1 R packages for Mixed Logit Estimation

4.1.1 bayesm Package (Rossi & McCulloch, 2005)

The *bayesm* package is developed by Peter Rossi in 2005 for the Bayesian estimation of *Discrete Choice Models* in the marketing research field, and this package is a companion R package to the book titled “Bayesian Statistics and Marketing” (Rossi *et al.*, 2005). The book provides a guide for the experienced researchers about how to construct and estimate the *Hierarchical Bayes Models*. The *rhierMnlRwMixture* function included in this package can be used for the estimation of a *Mixed Logit Model*. In this thesis, *bayesm* package version 2.2-5 is used.

i) Estimation Algorithm

The *rhierMnlRwMixture* function implemented the *hybrid Gibbs Sampler* which includes a *RW Metropolis* step with increments having covariance $s^2(H_i + (\Sigma_\beta^{-1})^{-1})^{-1}$ (where H_i is the individual-level Hessian) for the estimation of a *Mixed Logit Model* with a mixture of normals prior. The estimation steps implemented in this function is similar to the procedures discussed in section 3.4. The *RW metropolis* step included in this sampler is used for making draws of the random effect β_n , and β_n can be obtained as follows:

$$\beta_{new} = \beta_{old} + \varepsilon, \varepsilon \sim N(0, s^2(H_i + (\Sigma_\beta^{-1})^{-1})^{-1})$$

ii) Input Data Structure:

In order to use the *rhierMnlRwMixture* function for *Mixed Logit Model* estimation, the user is recommended to have some experience of working with data frames in R since the input data is required to be stored in a list in R. For example, if the input dataset is named as *bayesmData*, it should have the following structure:

bayesmData=list(p=p,lgtdata=lgtdata)

In this list, the first item $p=p$ contains the total number of alternatives. The second item *lgtdata* contains the information of the attributes and the choice made by the respondents. The *lgtdata* is a list which contains *nlg* elements, and each element in the *lgtdata* list represents one respondent. Each element in the *lgtdata* list contains the x data frame and y data frame for each individual. The x data frame for each individual respondent has dimension $(nset * p) \times nbeta$ which contains the attributes information. The y data frame for each respondent contains the alternative numbers chosen for all the choice sets. The ordering of the x data frame and y data frame required by *bayesm* package is identical to the input data structure, but the user is required to assign each respondent a separate data frame which only contains the attributes and choice information for that particular person. If categorical attributes exist in the dataset, the user should use effect coding before entering the data.

For example, if a dataset has the setup: *nlg*=50, *nset*=20, *p*=3, *nbeta*=6, then the *lgtdata* list should contain 50 elements, each element represents a particular respondent. The user can use the command: *lgtdata[[1]]* to obtain the input data structure for the first respondent. **Table 5** is a visual illustration of the input data

structure required by *bayesm* package for respondent 1. Since each respondent faces 20 choice sets, the *y* data frame contains a vector of 20 integers which represent 20 choices made by respondent 1. The *x* data frame contains a 60 by 6 integer matrix that represents the attribute information.

Table 5. Input data structure for bayesm package										
<i>lgtdata[[1]]\$y</i>	A vector of <i>nset</i> integers which contains the choices made by respondent 1:									
	Choice Set	1	2	3	4	5	...	18	19	20
	Choice	3	1	1	2	1	...	2	1	2
<i>lgtdata[[1]]\$X</i>	A (<i>nset</i> * <i>p</i>) × <i>nbeta</i> matrix which contains the attribute information:									
	Choice Set	Alternative	V1	V2	V3	V4	V5	V6		
	1	1	-1	-1	1	0	-1	-1		
	1	2	0	1	-1	-1	1	0		
	1	3	-1	-1	0	1	1	0		
	2	1	-1	-1	-1	-1	-1	-1		
	2	2	0	1	0	1	0	1		
	2	3	0	1	0	1	-1	-1		
		
	20	1	1	0	0	1	0	1		
	20	2	0	1	-1	-1	1	0		
	20	3	1	0	0	1	0	1		

iii) Model Estimation

After the data has been stored in a data frame, the user is required to specify several control settings before starting the model estimation (see **Table 6**).

Table 6. Model estimation setting using bayesm package	
<i>Prior1=list(mubar,Amu,nu,V,ncomp)</i>	
<i>mubar</i>	prior mean vector for normal mean. Default=0
<i>Amu</i>	prior precision for normal mean. Default=0.01/
<i>nu</i>	degree of freedom for IW prior. Default= <i>nbeta</i> +3
<i>V</i>	location parameter for IW prior. Default= <i>nul</i>
<i>ncomp</i>	number of mixture of normal components. In this thesis, <i>ncomp</i> =1 since only one normal component of the heterogeneity distribution is discussed. (Required)
<i>Mcmc1=list(s,w,R,keep)</i>	

<i>s</i>	scaling parameter for RW Metropolis. Default=2.93/sqrt(nbeta)
<i>w</i>	fractional likelihood weighting parameter. Default=0 .1
<i>R</i>	total number of iterations (Required)
<i>keep</i>	obtain every <i>keep</i> th draws for the estimation of posterior means (Required)
Estimation can be started by invoking: <i>out=rhierMnlRwMixture(Data=testData, Prior= Prior1, Mcmc= Mcmc1)</i>	

For further analysis after the model estimation, the user can use the command *out\$betadraw* to access the individual-level parameter estimates. The *out\$betadraw* gives a $n_{lgt} \times nbeta \times R/keep$ dimension array. The *posterior means* for each iteration can be obtained using the *out\$nmix\$compdraw* command. Examples of how to perform model estimation and analyze the estimation results can be found in **Appendix A. Part I - 1. bayesm package**

4.1.2 ChoiceModelR Package (Sermas, 2012)

The *ChoiceModelR* package is developed by Ryan Sermas in 2012, and this package contains one main function *choicemodelr* which can be used for the estimation of *Mixed Logit Models*. The estimation algorithm implemented in *choicemodelr* function is written based on the *rhierMnlRwMixture* function from *bayesm* package, and it is a simplified version of the *rhierMnlRwMixture* function. Version 1.2 of the *ChoiceModelR* package is used for the model estimation in this thesis.

i) Estimation Algorithm:

The estimation algorithm implemented in the *ChoiceModelR* package is identical to the one implemented in the *rhierMnlRwMixture* function except that the length of the *Random Walk (RW) Metropolis* step within the *Gibbs Sampler* was simplified to $s^2 \Sigma_{\beta}$. Furthermore, the *prior distribution* for the individual-level random coefficient β_n is restricted to be normally distributed in this package.

ii) Input Data Structure:

The *ChoiceModelR* package requires the data to be entered in a matrix format which is similar to the example displayed in **Table 7**. The first column indicates the ID of the respondent, for each respondent, the total number of rows = $nset \times p$. The second column contains the choice set number, the third column contains

the alternative number. The last column represents the choice made by the respondent, and within each choice set, only the first cell is indicated by the number of alternative selected, the rest of the rows are indicated as “0”.

Table 7. Input data structure for ChoiceModelR package

Note: This illustration is directly taken from the documentation of *ChoiceModelR* package.

ID	Choice Set	Alternative	Attribute 1	Attribute 2	Choice
1	1	1	2	2	4
1	1	2	4	2	0
1	1	3	3	3	0
1	1	4	1	3	0
1	2	1	3	2	2
1	2	2	1	3	0
1	2	3	5	2	0
1	2	4	4	2	0

In the example from **Table 7**, the first 4 rows indicate that for respondent 1, in the first choice set, there are 4 alternatives, and the respondent has chosen Alternative 4. Each alternative has 2 categorical attributes, attribute 1 has 5 levels, and attribute 2 has 3 levels.

iii) Control Settings and Model Estimation

Before starting the model estimation, the user is required to specify the *xcoding* option to indicate if the attribute is categorical or continuous. The categorical attribute should be specified as “0”, and the continuous attribute should be specified as “1”. In the example from **Table 7**, there are 2 categorical attributes, therefore the *xcoding* option should be specified as *xcoding(0,0)*. The user is required to specify the values for *xcoding* according to the order of the attributes arranged in the matrix. If there is a third attribute which is continuous, then the *xcoding* option should be specified as *xcoding(0,0,1)*, where the last “1” indicates that the third attribute is continuous.

In addition to the *xcoding* option, the user is also required to specify several iteration settings shown in **Table 8**.

Table 8. Model estimation setting using ChoiceModelR package

<i>prior=list(mubar, Amu, df, v)</i>	
<i>mubar</i>	prior mean vector. Default is a vector of zeros
<i>Amu</i>	prior precision for normal mean. Default = 0.01
<i>df</i>	the degrees of freedom for IW prior for Sigma, must be ≥ 2 . Default=5

<i>v</i>	prior variance, must be ≥ 0 . Default=2
<i>mcmc=list(R,use,s)</i>	
<i>R</i>	total number of iterations (Required)
<i>use</i>	the number of latest iterations used for calculating the model parameters (Required)
<i>s</i>	the scaling parameter for the <i>Random Walk Metropolis</i> step. Default = 0.1
<i>options=list(save,keep)</i>	
<i>save</i>	save the draws of model parameters if <i>save</i> is set to TRUE. Default=FALSE
<i>keep</i>	retain every <i>keep</i> th draws
Estimation can be started by invoking: <i>out = choicemodelr(logitdata, xcoding, mcmc = mcmc, options = options)</i>	

For example, if the user is estimating a *Mixed Logit Model* on a dataset called *logitdata* which contains 2 categorical attributes, the commands provided in **Table 9** can be used to estimate the model for a total of 10,000 iterations, keeping everything 10th draw and use the latest 5000 draws to calculate the individual-level parameter estimates.

Table 9. Example code of using ChoiceModelR package

```
xcoding = c(0, 0)
mcmc = list(R = 10000, use = 5000)
options = list(save=TRUE, keep=10)
out = choicemodelr(logitdata, xcoding, mcmc = mcmc, options = options)
```

The estimated *posterior means* for each attribute level are also plotted automatically in the graphic window during the estimation process. After the model estimation, the individual-level parameter estimates for each attribute level are written to Rbetas.csv file and saved automatically in the working directory. The Rlog.txt file which contains the information of the iteration process can also be found in the working directory after the model estimation. The *posterior means* for each iteration can be obtained using the *out\$compdraw* command. Example of performing model estimation in R using this package can be found in **Appendix A. Part I – 2. ChoiceModelR package.**

4.1.3 RSGHB Package (Dumont *et al.*, 2013)

The *RSGHB* Package contains one main function *doHB* which can be used for the estimation of *Mixed Logit Models*. This package is developed by Dumont, J., Keller, J. & Carpenter, C. (2013), and this package optimized the Matlab/Gauss

code for Hierarchical Bayesian Estimation written by Kenneth Train, the estimation algorithm has already been discussed in section 3.4 of this thesis. This package is considered to be more flexible since distributions other than the normal distribution could be used for the prior specifications, such as: Log-normal, Normal censored from below at zero, Johnson's (1949) S_B distribution. These bounded distributions are transformations of normal distribution which are not discussed in this thesis in detail. More information about these distributions can be found in Train (2004). This thesis uses version 1.0.1 of the *RSGHB* package for the *Mixed Logit* estimation.

i) Input Data Structure:

The attribute section of the input data frame required by *RSGHB* package is organized differently compared to the formats required by *bayesm* and *ChoiceModelR* package. The user is required to store all the data information as a data frame similar to the structure shown in **Table 10**.

Table 10. Input data structure for RSGHB package						
Note: This illustration is directly taken from the documentation of <i>RSGHB</i> package.						
		Attribute 1 (A)		Attribute 2 (B)		
ID	Choice Set	Alternative 1 (A1)	Alternative 2 (A2)	Alternative 1 (B1)	Alternative 2 (B2)	choice
1	1	60	51	1.25	0	1
1	2	60	51	0.75	0	2
1	3	63	59	0.5	0	1
1	4	60	54	0.75	0	1
1	5	60	54	0.5	0	1
1	6	63	54	0.75	0	1
1	7	61	55	1.25	0	1
1	8	61	57	0.75	0	1
1	9	60	53	1.25	0	1

In **Table 10**, the first column represents the ID of a respondent, the second column indicates the choice set number. Under each attribute section, every column represents the attribute information for an alternative. The last column is the choice column, since each row represents one choice set, the number indicated in each cell of the choice column indicates the choice made within that particular choice set. If categorical attributes exist in the dataset, the user is required to use effect coding before entering the data.

ii) Model Estimation

A special feature for the *RSGHB* package is that the user is required to specify the likelihood function manually before starting the model estimation. For example, if the attribute columns in **Table 10** are named as A1,A2,B1,B2 where A1 and A2 are attributes A of alternative 1 and 2 respectively; and B1 and B2 represent attributes B of alternative 1 and 2 respectively. The estimation settings for *RSGHB* package in R can be specified by following the steps shown in **Table 11**.

Table 11. R code for model estimation setting in RSGHB package	
likelihood function specification	<p>Specify variables for utility functions A1 represents attribute “A” of alternative 1 <code>A1=RSGHBdata\$A1</code></p> <p>A2 indicates attribute “A” of alternative 2 <code>A2=RSGHBdata\$A2</code></p> <p><code>B1=RSGHBdata\$B1</code> <code>B2=RSGHBdata\$B2</code></p> <p>There are 2 alternatives, so 2 choice vectors should be specified <code>y1=(RSGHBdata\$choice==1)</code> <code>y2=(RSGHBdata\$choice==2)</code></p> <p><code>likelihood <- function(fc,b)</code> <code>{</code></p> <p>For Mixed Logit Model estimation, “b” vector is used, it is the vector of the random coefficients. “cc” is used to index the vector “b”. “fc” is only used for the Logit Model with fixed coefficients, therefore it is discarded in this example.</p> <p><code>cc <- 1</code></p> <p><i>beta1</i> and <i>beta2</i> are the model parameters to be estimated</p> <p><code>beta1 <- b[,cc];cc<-cc+1</code> <code>beta2 <- b[,cc];cc<-cc+1</code></p> <p>Specify 2 utility functions <code>v1 <- beta1*A1+beta2*B1</code> <code>v2 <- beta1*A2+beta2*B2</code></p> <p>Construct the Likelihood for Logit Model <code>p <- (exp(v1)*y1 + exp(v2)*y2)/(exp(v1) + exp(v2))</code></p>

	<pre>return(p) }</pre>
Control parameters specification	<pre>control <-list(modelname=modelname, gVarNamesNormal=gVarNamesNormal,gDIST=gDIST,svN=svN,gNCREP =gNCREP,gNEREP=gNEREP,gNSKIP=gNSKIP)</pre>
<i>modelname</i>	The user is required to specify the model name <pre>modelname <- "MNL"</pre>
<i>gVarNamesNormal</i>	This option is used to specify the name of the random parameters. <pre>gVarNamesNormal <- c("Time","Price")</pre>
<i>gDIST</i>	An integer vector for the specification of the distribution of the random coefficient, the input value for each element is 1-5: <ol style="list-style-type: none"> 1- Normal 2- Positive Log-Normal 3- Negative Log-Normal 4- Censored Normal 5- Johnson SB <p>The user is required to specify the distribution of the random parameter corresponding to the <i>gVarNamesNormal</i> using integers to indicate the distribution of the random parameters.</p>
<i>svN</i>	A vector which contains the initial mean values for the normally distributed random parameters. This vector should contain the same number of elements as previously specified in <i>gVarNamesNormal</i> .
<i>gNCREP</i>	Number of iterations used for burn-in period. (Defaults = 100000)
<i>gNEREP</i>	Number of iterations used to calculate the posterior means after the burn-in period. (Default=100000)
<i>gNSKIP</i>	If this number is set to 10, it means that keeping every 10 th draw. (Defaults=1)
Estimation can be started by invoking: <pre>doHB(likelihood,choicedata,control)</pre>	

Since various options are required to be specified manually, the user should be more cautious when the dataset contains a large number of model parameters. Misspecification of the variables or the likelihood function will lead to incorrect result of the random coefficient estimates. After specifying the likelihood function and control settings, the user can invoke the function `doHB(likelihood,choicedata,control)` to start the model estimation. A trace plot named "Markov Chain" is displayed in the graphic window during the model estimation process. The X-axis of this plot indicates the number of iterations, and the Y-axis represents the calculated *utilities* across respondents.

At the end of the model estimation, several output files will be produced (see **Table 12**).

Table 12. Output files of RSGHB package	
<i>A_file</i>	<i>posterior means</i> for the model parameters of each iteration (after the burn-in period).
<i>B_file</i>	individual-level parameter estimates
<i>Bsd_file</i>	individual-level standard deviations
<i>C_file</i>	individual-level parameter estimates for the transformed random parameters
<i>Csd_file</i>	individual-level standard deviations for the transformed random parameters
<i>D_file</i>	contains the sample covariance for each iteration
<i>F_file</i>	fixed parameter estimates

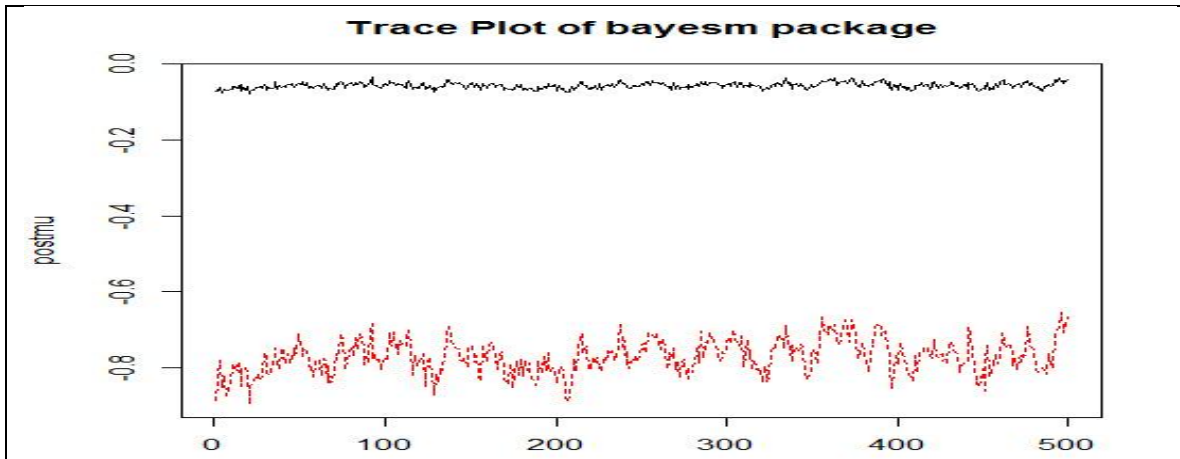
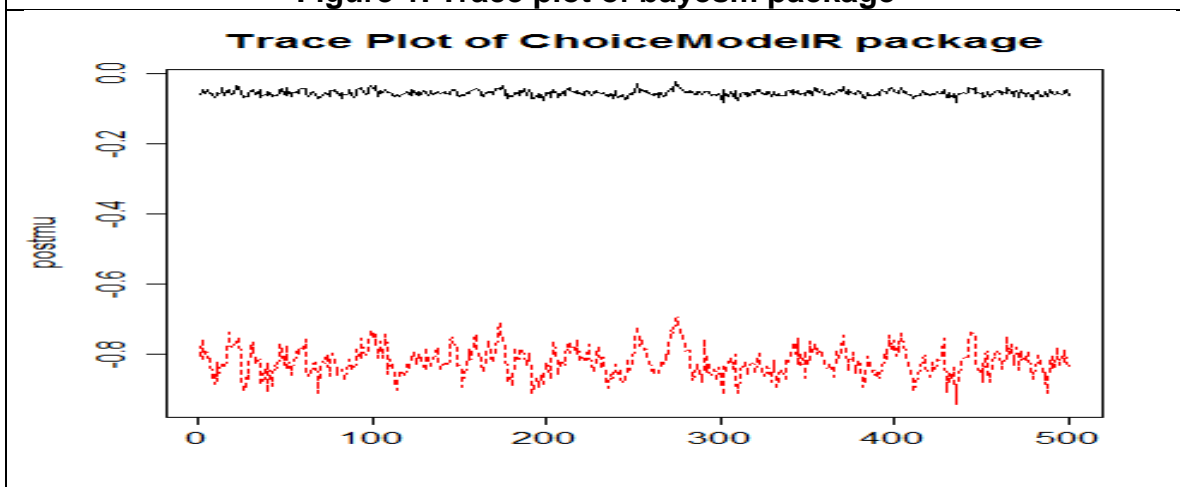
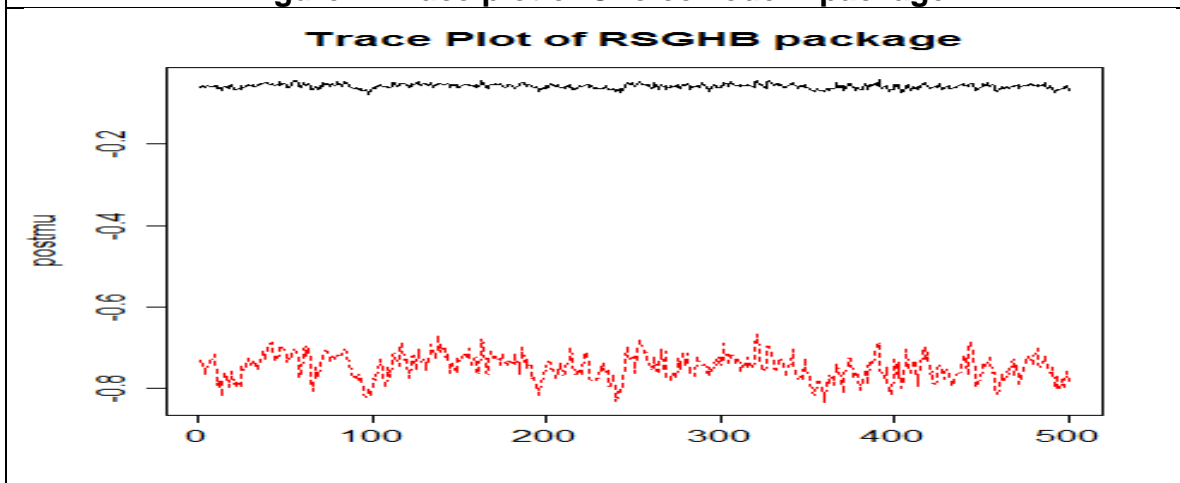
4.1.4 Estimation Comparison

The “choicedata” dataset provided in *RSGHB* package is used for a preliminary comparison of the *Mixed Logit Model* estimations using *bayesm*, *ChoiceModelR*, and *RSGHB* packages. The dataset contains 1138 respondents and two attributes (*price* and *time*). Each respondent faces 9 choice sets. The estimations for all three packages are done using 10,000 iterations (keeping every 10th draw), and the first half of the iterations are discarded as the burn-in period. The estimated *posterior means* are calculated based on the draws after the burn-in period.

Similar estimated *posterior means* for the model parameters are obtained using three R packages. According to the output information in **Table 13**, the *bayesm* package took more than an hour (56.51 minutes) to complete the model estimation, the *ChoiceModelR* package and *RSGHB* package both took less than 2 minutes to complete the model estimation which is much faster than the estimation speed using *bayesm* package.

Table 13. Estimated posterior means using different R packages			
	<i>Time</i>	<i>Cost</i>	Estimation Time (in minutes)
bayesm	-0.05629	-0.77063	56.51
ChoiceModelR	-0.05674	-0.8217	1.36
RSGHB	-0.05908	-0.74682	1.38

The default trace plot provided by each package differs slightly in their formats, in order to have a better examination of the trace plots, they are reconstructed so that all of them are displayed in the same format (Note: the “burn-in” period is not included in the trace plot). The Trace plots in **Figure 1- 3** suggest that the estimated *posterior means* converge to similar values for all three packages.

**Figure 1. Trace plot of bayesm package****Figure 2. Trace plot of ChoiceModelR package****Figure 3. Trace plot of RSGHB package**

Examples with simulated data are illustrated in **Chapter 6** using the user interface implemented in this thesis. The accuracy of the model estimation for each package is discussed based on the calculation of certain precision criteria.

4.2 Mixed Probit Estimation

For the *Mixed Probit Model* Estimation, the R function provided by Gillbride & Allenby (2004) is modified and named as *rmxp* function so that it could be used to estimate a *Mixed Probit Model*.

4.2.1 *rmxp* Function for Mixed Probit Estimation

The Estimation algorithm for the *Mixed Probit* estimation is discussed in section 3.5. A list of data which contains 5 elements are required for the model estimation: *Data=list(y=y,x=x,nlgt=nlgt,nset=nset,p=p,nbeta=nbeta)*. The *x* and *y* matrices are each stored in a two multi-dimensional array. The *x* matrix is stored in a 4 dimensional array which contains all the attribute information with dimension equals to $nbeta \times p \times nset \times nlgt$; and the *y* matrix is required to be stored in a 3 dimensional array ($p \times nset \times nlgt$) using indicator variables.

For example, if a dataset has the setup: *nlgt=100, nset=16, p=3, nbeta=5*, then the user can use *Data\$y[,1]* command to obtain the choices made by respondent 1. The *Data\$x[,1,2]* command can be used to obtain the attribute information of respondent 2 in choice set 1(see **Table 14**).

Table 14. Input data structure for <i>rmxp</i> function							
<i>Data\$y[,1]</i>	A $p \times nset$ matrix which represents the choices made by respondent 1 using indicator variables:						
		Choice Set					
	Alternative	1	2	3	...	15	16
	1	0	1	1	...	0	1
	2	0	0	0	...	0	0
	3	1	0	0	...	1	0
<i>Data\$x[,1,2]</i>	A $nbeta \times p$ matrix which contains the attribute information in choice set 1 of respondent 2:						
		Alternative					
	Parameter	1	2	3			
	1	-1	1	-1			
	2	-1	1	-1			
	3	-1	0	0			
	4	-1	1	1			
	5	-1	0	0			

In order to estimate a dataset using *Mixed Probit Model*, the code provided in **Table 15** can be used.

Table 15. Model estimation setting using <i>rmxp</i> function	
<i>Prior=list(nu,V0,betabarbar, Abeta)</i>	
<i>nu</i>	degree of freedom . Default= $nbeta+5$
<i>V0</i>	prior variance. Default= <i>nu</i>
<i>betabarbar</i>	prior normal mean. Default=0
<i>Abeta</i>	prior precision for normal mean. Default=0.01
<i>Mcmc=list(R,keep)</i>	
<i>R</i>	total number of iterations
<i>keep</i>	the number of latest iterations used for calculating the parameter estimates
Estimation can be started by invoking:	
<i>out=rmxp(Data,Prior,Mcmc)</i>	

The estimation can be started by calling *out=rmxp(Data,Prior,Mcmc)*. At the end of the model estimation, the individual-level parameter estimates can be accessed by using the *out\$betadraw* command. The *posterior means* for each iteration can be obtained using the *out\$betabardraw* command. Example of how to perform model estimation using the *rmxp* function can be found in **Appendix A. Part II**.

4.2.2 Mixed Probit sample data Estimation

The dataset used for the illustration of the *Mixed Probit* estimation is from a conjoint study of customer preference about new camera features. The dataset contains 302 respondents, each faces 14 choice sets, within each choice set, the respondents were asked to choose among 7 alternatives. The dataset contains the camera functionality attributes and price of the camera. There are 18 parameters to be estimated in this dataset. A more detailed dataset description is provided in Gilbride & Allenby (2004). 20,000 iterations are performed and the first 10,000 iterations are discarded as “burn-in” period, the time for completion is 60.32 minutes.

Table 16. Estimated posterior means of Mixed Probit Model					
$\bar{\beta}_1$	0.326449	$\bar{\beta}_7$	0.672708	$\bar{\beta}_{13}$	1.429936
$\bar{\beta}_2$	2.580976	$\bar{\beta}_8$	-0.75213	$\bar{\beta}_{14}$	-0.05709
$\bar{\beta}_3$	1.86653	$\bar{\beta}_9$	0.741207	$\bar{\beta}_{15}$	0.502433
$\bar{\beta}_4$	0.00232	$\bar{\beta}_{10}$	-0.75753	$\bar{\beta}_{16}$	0.543692
$\bar{\beta}_5$	0.118851	$\bar{\beta}_{11}$	0.409349	$\bar{\beta}_{17}$	0.707009
$\bar{\beta}_6$	0.394706	$\bar{\beta}_{12}$	1.115231	$\bar{\beta}_{18}$	-0.63221

The estimated *posterior means* are displayed in **Table 16**. The trace plot of the estimated model parameters shown in **Figure 4** indicates that convergence has been reached. The accuracy of the estimation of the *rmxp* function is discussed in **Chapter 5** by estimating a simulated *Mixed Probit* dataset.

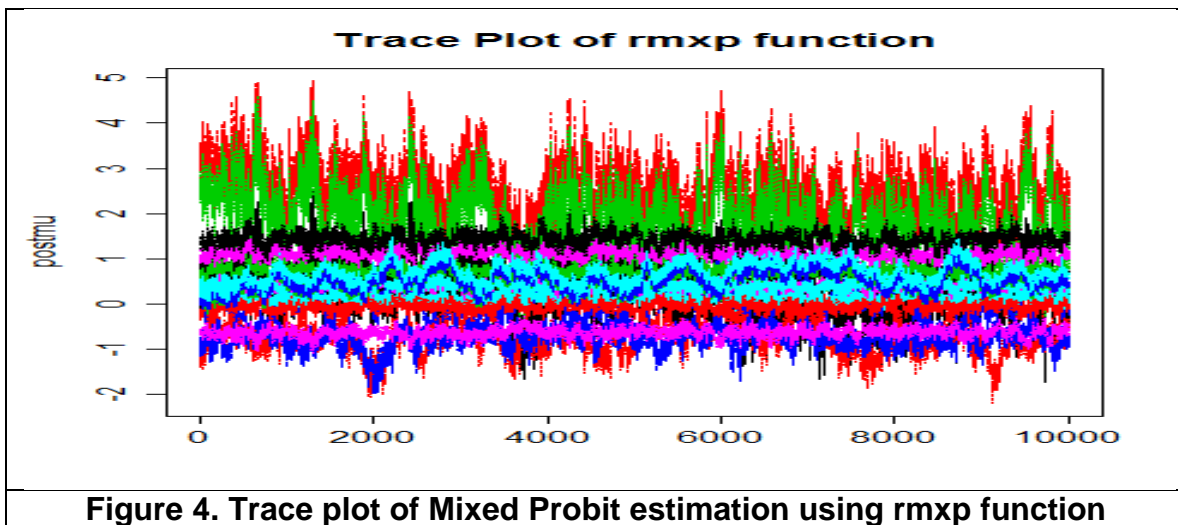


Figure 4. Trace plot of Mixed Probit estimation using rmxp function

Chapter 5 User Interface for Data Structure Transformation

The purpose of the user interface implemented in this thesis is to transform the given input data into various formats so that the researcher could use the transformed dataset instead of transforming the input data structure manually. Several functions are written which can be used in conjunction with this user interface. The user is required to specify the directory path correctly before using this interface. Additionally, *bayesm*, *ChoiceModelR*, and *RSGHB* packages should also be installed in R program.

5.1 Design of the User Interface

The implemented user interface can be used by calling the *userin* function using the command:

```
userin (pkg,x,y,p,nset,nlgt,nbeta)
```

The *userin* function returns the transformed data structure required by the indicated R package/function number. The user interface can be invoked by specifying the arguments shown in **Table 17**.

Table 17. Input arguments required by the <i>userin</i> function	
<i>pkg</i>	package indicator, the user could choose a package for the estimation by entering number 1, 2, 3 or 4: 1 - <i>bayesm</i> package 2 - <i>ChoiceModelR</i> package 3 - <i>RSGHB</i> package 4 – <i>rmxp</i> function (for <i>Mixed Probit Model</i>)
<i>p</i>	number of alternatives
<i>nset</i>	number of choice sets
<i>nlgt</i>	number of respondents
<i>nbeta</i>	number of model parameters
<i>x</i>	$(p * nset * nlgt) \times nbet$ a design matrix of the attributes. The table below is a visual illustration of the required input format of the <i>x</i> matrix. Only the columns which contain the attribute information are required to be given (see shaded area), the first 3 columns are not needed for the transformation of the data structure. The discrete attributes should be “effect coded” before invoking the “ <i>userin</i> ” function.

	<table><tr><th>ID</th><th>Choice Set</th><th>Alternative</th><th>Attribute 1</th><th>Attribute 2</th><th>Attribute...</th></tr><tr><td>1</td><td>1</td><td>1</td><td>...</td><td>...</td><td>...</td></tr><tr><td>1</td><td>1</td><td>2</td><td>...</td><td>...</td><td>...</td></tr><tr><td>1</td><td>1</td><td>3</td><td>...</td><td>...</td><td>...</td></tr><tr><td>1</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr><tr><td>1</td><td>2</td><td>3</td><td>...</td><td>...</td><td>...</td></tr><tr><td>1</td><td>2</td><td>4</td><td>...</td><td>...</td><td>...</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr><tr><td><i>nlgt</i></td><td><i>nset</i></td><td><i>p</i></td><td>...</td><td>...</td><td>...</td></tr></table>	ID	Choice Set	Alternative	Attribute 1	Attribute 2	Attribute...	1	1	1	1	1	2	1	1	3	1	1	2	3	1	2	4	<i>nlgt</i>	<i>nset</i>	<i>p</i>
ID	Choice Set	Alternative	Attribute 1	Attribute 2	Attribute...																																																		
1	1	1																																																		
1	1	2																																																		
1	1	3																																																		
1																																																		
1	2	3																																																		
1	2	4																																																		
...																																																		
<i>nlgt</i>	<i>nset</i>	<i>p</i>																																																		
<i>y</i>	<p>(<i>nset</i> * <i>nlgt</i>) × 1 matrix which contains the responses made by the respondents. The table below is a visual illustration of the required format of the <i>y</i> matrix. Only the last column (the “Choice” column) is needed to be given for the transformation of the data structure.</p> <table><tr><th>ID</th><th>Choice Set</th><th>Choice</th></tr><tr><td>1</td><td>1</td><td>4</td></tr><tr><td>1</td><td>2</td><td>2</td></tr><tr><td>...</td><td>...</td><td>...</td></tr><tr><td><i>nlgt</i></td><td><i>nset</i></td><td>...</td></tr></table>	ID	Choice Set	Choice	1	1	4	1	2	2	<i>nlgt</i>	<i>nset</i>	...																																							
ID	Choice Set	Choice																																																					
1	1	4																																																					
1	2	2																																																					
...																																																					
<i>nlgt</i>	<i>nset</i>	...																																																					

The function returns the transformed data structure required by the indicated R package/function number which have been discussed previously in **Chapter 4**. For example, if the *pkg* argument in the *userin* function is set to 1, then the *bayesm* package is selected, and the *userin* function will return a data frame which is required by the *bayesm* package. A message of “*The dataset is ready to be used for the estimation of bayesm package*” is also displayed automatically in R to ensure that the user has selected the desired package for the model estimation.

It should be noticed that since *ChoiceModelR* package does not require the discrete attributes to be effect coded, the original attribute level should be entered. If the dataset is given as effect coded, the user could use the *uncodex* function to transform the coded attributes into the original format. The *uncodex* function requires the input *x* matrix only contains discrete attributes. The user could also use the *uneffect* function to transform the design matrix for only 1 attribute at a time. **Table 18** contains the description of the *uncodex* and *uneffect* functions. An example of transforming the effect coded *x* matrix into “level” format can be found in the **Appendix A. Part I – 2. ChoiceModelR package**.

Table 18. Functions to transform effect coded design matrix into “level” format	
Function Name	Input Arguments
<code>uncodex(x,nbeta,levels)</code> A function which converts a design matrix from effect coding to the original “level” format.	<p><i>x</i>: design matrix which only contains the effect coded attributes</p> <p><i>nbeta</i>: total number of model parameters</p> <p><i>levels</i>: a vector of integers which indicates the level of each attribute in <i>x</i> matrix. For example, <code>levels<-(3,3,3)</code> indicates that the design matrix has 3 attributes, each with 3 levels.</p> <p>NOTE: this function assumes that the <i>x</i> matrix only contains discrete attributes. The ordering of the attribute levels specified in <i>levels</i> argument should be the same as the design matrix.</p>
<code>uneffect(attr,l)</code> This function converts 1 attribute from effect coding to the original levels.	<p><i>attr</i>: matrix which contains one effect coded attribute</p> <p><i>l</i>: total number of levels of <i>attr</i></p>

As previously discussed in **Chapter 4**, the user is required to specify control settings for each package. For *RSGHB* package, since the user is required to construct the likelihood function, a different way of specifying the variables are used so that the user could specify the likelihood function faster (see **Appendix A. Part I – 3. RSGHB package**).

After the estimation, if the user is using simulated dataset, *compare* or *compareprobit* function can be used to calculate the precision criteria for *Mixed Logit* or *Mixed Probit Model* respectively. Three precision criteria are calculated in *compare/compareprobit* functions: *Mean Absolute Error* (MAE_{β}), *Root Mean Squared Error* ($RMSE_{\beta}$), and *Root Mean Squared Prediction Error* ($RMSE_p$). The MAE_{β} assesses the averaged difference in magnitude of parameter estimates and true parameter values regardless the signs of the parameter values, this criteria is used in the example provided in the *ChoiceModelR* package documentation. The $RMSE_{\beta}$, measures the difference between the unit-level parameter estimates and the true parameter values, $RMSE_p$ is used to assess the accuracy of prediction, both $RMSE_{\beta}$ and $RMSE_p$ have been used in Yu, et al. (2011). The precision criteria are calculated in R based on the formulas displayed in **Table 19**.

Table 19. R package comparison criteria	
Mean Absolute Error	$MAE_{\beta} = \frac{1}{N} \sum_{n=1}^N \hat{\beta}_n - \beta_n^* $
Root Mean Squared Error	$RMSE_{\beta} = \sqrt{\frac{1}{N} \sum_{n=1}^N (\hat{\beta}_n - \beta_n^*)' (\hat{\beta}_n - \beta_n^*)}$
Root Mean Squared Prediction Error	$RMSE_p = \sqrt{\frac{1}{N} \sum_{n=1}^N (p(\hat{\beta}_n) - p(\beta_n^*))' (p(\hat{\beta}_n) - p(\beta_n^*))}$

The *compare* function can be used for the computation of the precision criteria for the *Mixed Logit Model*. Note that for *ChoiceModelR* package, the *x* matrix used in the *compare* function should be the effect coded design matrix, because when calculating the $RMSE_p$, the *simmlnwX* function from *bayesm* package is used to obtain the choice probabilities, and this function requires the effect coded design matrix to be entered. For the *Mixed Probit Model*, the *compareprobit* function can be used for the calculation of precision criteria. Since a function provided in *bayesm* package is used to calculate the $RMSE_p$, and this function requires the *differenced* system of the *x* matrix to be entered, therefore the user is also required to use the *differenced* system of the *x* matrix using *p* as the base category (see **Chapter 2** for the discussion of the *difference* system). The user can use *getdiff* function to transform the given *x* matrix into the *differenced* system. **Table 20** contains the description of the functions which can be used in the comparison of estimation results part.

Table 20. Functions for estimation results comparison	
<code>compare(p,nset,nlgt,estbeta,truebeta,x)</code> This function returns the comparison criteria for <i>Mixed Logit Model</i> . The “Comparison Criteria.csv” file is saved to the working dictionary.	<code>estbeta</code> : estimated unit-level parameters <code>truebeta</code> : true unit-level parameter values
<code>compareprobit(estbeta,truebeta,Sigma,x,nlgt,p,nset)</code> This function returns the comparison criteria for <i>Mixed Probit Model</i> . The “Comparison Criteria.csv” file is saved to the working dictionary.	<code>Sigma</code> : the covariance matrix of the error terms used for the simulation of the dataset <code>estbeta</code> : estimated individual-level posterior means <code>truebeta</code> : true individual-level parameter values
<code>getdiff(p,nset,nlgt,nbeta,x)</code> This function converts the input <i>x</i> matrix	see Table 5.1 for input arguments specification.

into the <i>differenced system</i> using alternative p as the base category. It returns a $((p - 1) * nset) \times nbeta$ dimensional matrix.	
---	--

The following sections are illustrations of how to use this user interface for simulated data. **Section 5.2** provides the example of estimating simulated *Mixed Logit* dataset. **Section 5.3** is an example of *Mixed Probit* estimation using the *rmxp* function.

5.2 R package Comparison for Mixed Logit Model

In this section, a simulated dataset for *Mixed Logit Model* is used for comparing the estimation results of *bayesm*, *ChoiceModelR*, and *RSGHB* packages. **Table 21** contains the dataset information. For each attribute, there are 3 levels which have been effect coded (the highest level of each attribute is considered to be the reference category which is coded as “-1”). There are 6 model parameters to be estimated. For the *ChoiceModelR* package, since it does not require the attributes to be coded, the design matrix of each attribute has been transformed from effect coding to the original “level” format.

Table 21. Simulated Mixed Logit dataset setup	
Number of Respondents	50
Number of Choice Sets	20
Number of Alternatives	3
Number of Attributes	3
Number of Parameters	6

The dataset is converted into the *bayesm*, *ChoiceModelR*, and *RSGHB* formats respectively. The *Mixed Logit* estimations are performed separately for three packages. 30,000 iterations (keeping every 10th draw) per respondent are used for all three packages, and the first half of the iterations are discarded as the “burn-in” period. A detailed explanation of how to perform the *Mixed Logit Model* estimations in R can be found in **Appendix A. Part I**.

The estimated *posterior means* for the 6 model parameters shown in **Table 22** indicate that the estimation results obtained using *bayesm* and *ChoiceModelR* packages are very close to each other. For the *RSGHB* package, the estimated *posterior means* slightly differ from the other two packages. The values for the

estimated *posterior means* obtained from three R packages are not very different from the true parameter values.

Table 22. Estimated posterior means using different R packages						
	$\bar{\beta}_1$	$\bar{\beta}_2$	$\bar{\beta}_3$	$\bar{\beta}_4$	$\bar{\beta}_5$	$\bar{\beta}_6$
bayesm	-0.77628	0.03493	-0.16227	-0.05059	-0.62139	0.105637
ChoiceModelR	-0.81114	0.033776	-0.1664	-0.0503	-0.66061	0.112911
RSGHB	-0.63011	0.011709	-0.09382	-0.05783	-0.45832	0.071921
True Parameter Values	-0.67482	0.040027	-0.26655	0.019073	-0.55184	0.053364

The trace plots (after the burn-in period) displayed in **Figure 5-7** show that convergence has been reached for all three R packages, and these trace plots do not look very different from each other.

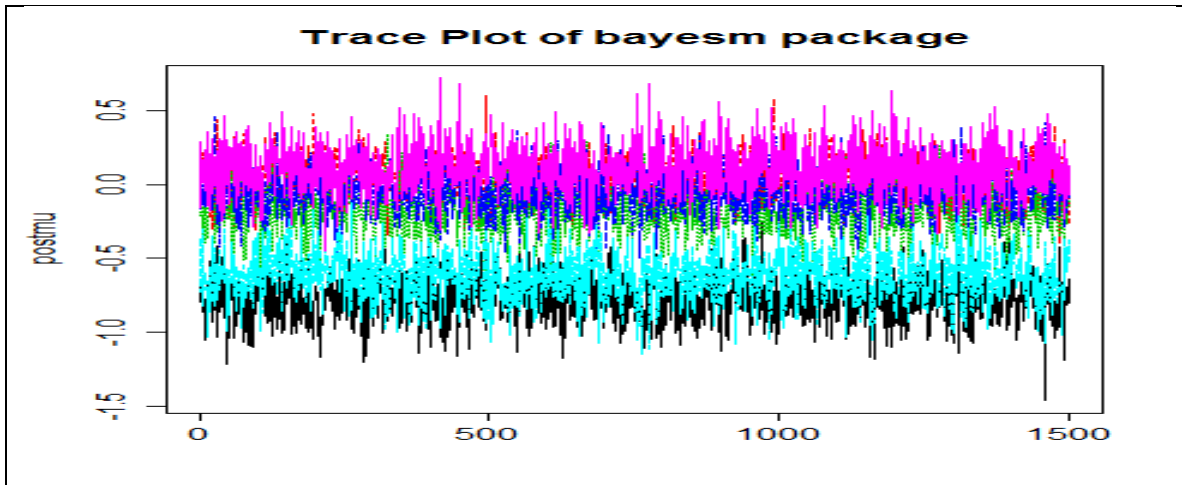


Figure 5. Trace plot of bayesm package

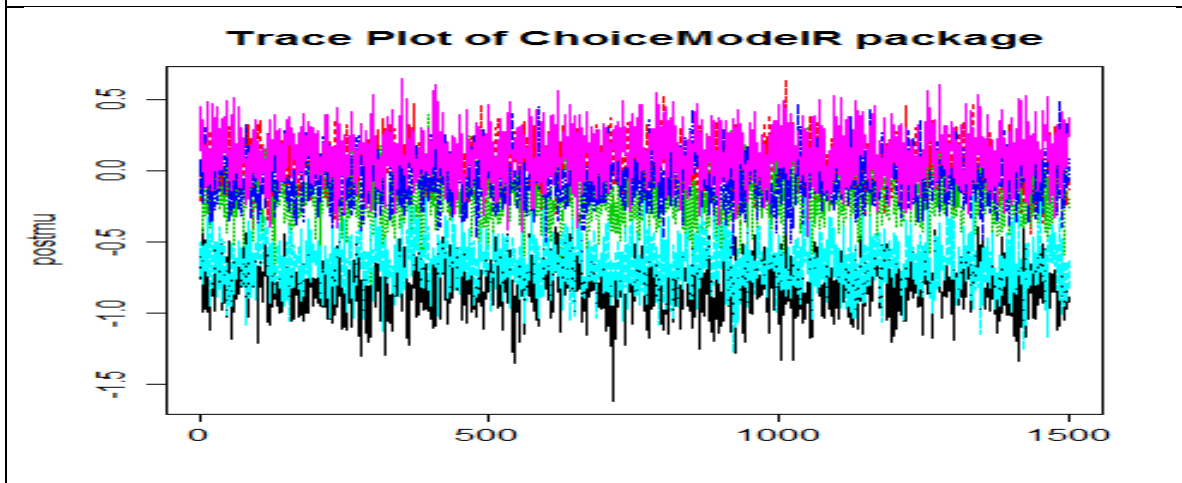


Figure 6. Trace plot of ChoiceModelR package

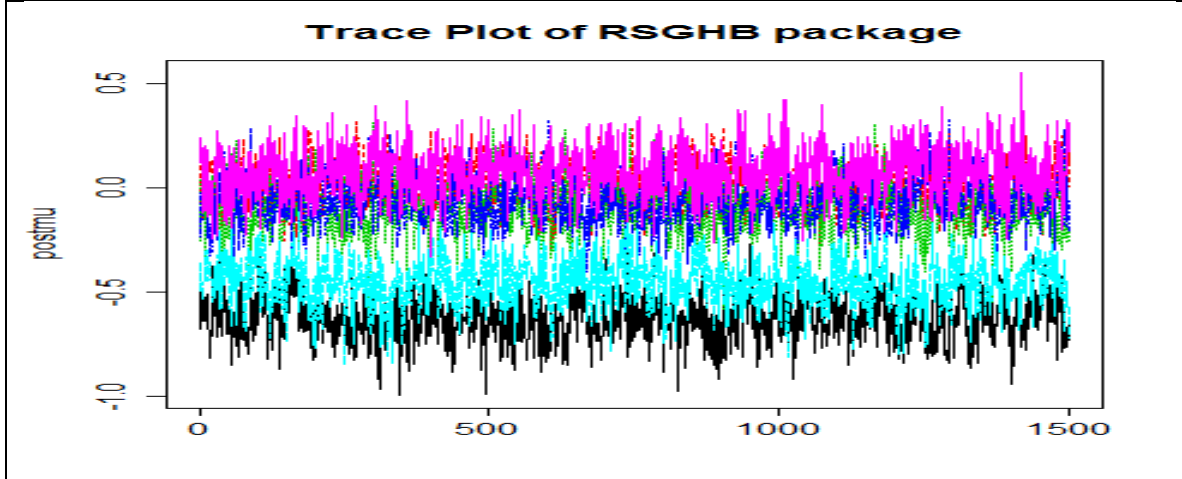


Figure 7. Trace plot of RSGHB package

The precision criteria calculated after the estimation shown in **Table 23** suggest that the *RSGHB* package yields the lowest MAE_{β} , $RMSE_{\beta}$, and $RMSE_p$ values among the three packages, this implies that the *RSGHB* package is the most accurate package and the estimation speed using this package is the fastest among all three packages (1.54 minutes). Since the precision criteria calculated using the *ChoiceModelR* package are the largest, therefore this package is the least accurate one. For the total time elapsed, the estimation performed using *bayesm* package is still the slowest which took 8.26 minutes for completion, but it is more accurate than the *ChoiceModelR* package since all the precision criteria for *bayesm* package are smaller than the ones from *ChoiceModelR* package.

Table 23. Precision criteria for Mixed Logit estimation				
	MAE	RMSE	RMSPE	Time (in minutes)
bayesm	0.341818	0.4357	0.155318	8.26
ChoiceModelR	0.359315	0.461428	0.156936	5.24
RSGHB	0.320001	0.415003	0.146367	1.54

5.3 Simulated dataset for Mixed Probit Estimation

In this section, a simulated dataset is used for the estimation of the *Mixed Probit Model*. **Table 24** contains the dataset information. The dataset is simulated based on an orthogonal x design matrix. The level for each attribute is 2, 2, 4 respectively, and the attributes are effect coded (the highest level of each attribute is considered to be the reference category which is coded as “-1”). There are 5 model parameters to be estimated. A function from *bayesm* package is used for the simulation of this dataset, and this function requires the x matrix to be entered as a *differenced system* (alternative p is treated as the base category). The details of the simulation function can be found in the *rmnpGibbs* function from the documentation of the *bayesm* package.

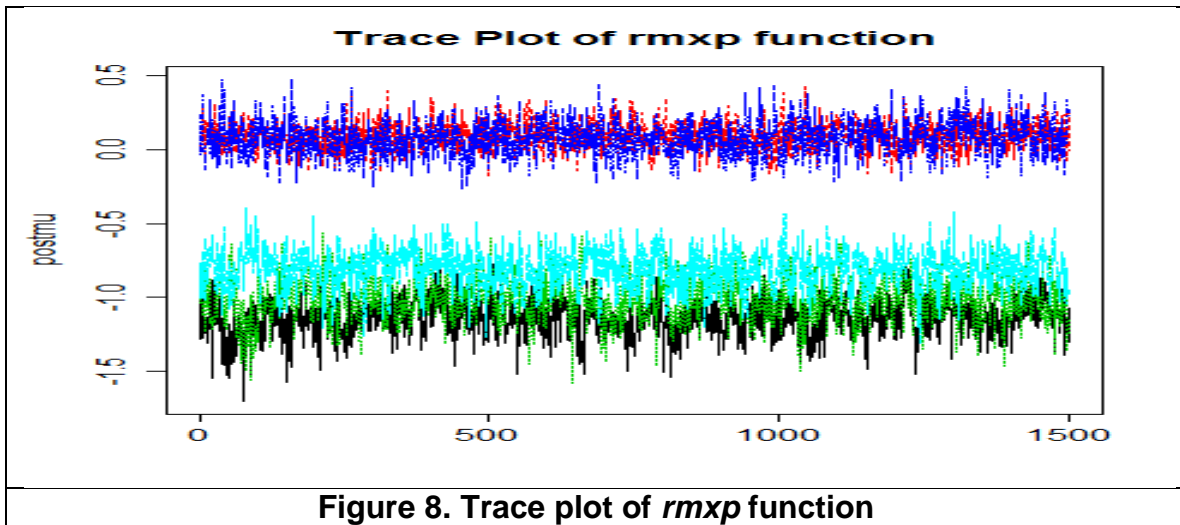
Table 24. Simulated Mixed Probit dataset setup	
Number of Respondents	100
Number of Choice Sets	16
Number of Alternatives	3
Number of Attributes	3
Number of Parameters	5

The estimation has been performed using 30,000 iterations (keeping every 10^{th} draw), and the first half of the iterations are discarded as the “burn-in” period.

The estimated *posterior means* of the model parameters displayed in **Table 25** are similar to the true parameter values.

Table 25. Estimated posterior means using <i>rmxp</i> function					
	$\bar{\beta}_1$	$\bar{\beta}_2$	$\bar{\beta}_3$	$\bar{\beta}_4$	$\bar{\beta}_5$
Estimated Posterior Means	-1.13884	0.093337	-1.00956	0.077872	-0.8117
True Parameter Values	-0.83438	0.102166	-0.70273	0.106559	-0.76069

The trace plot displayed in **Figure 8** indicates that convergence has been reached.



The precision criteria for the *rmxp* function are calculated and displayed in **Table 26**. The values of these precision criteria are not very large, and this implies that the *Mixed Probit Model* estimation using *rmxp* function has a good accuracy. However, since no other R packages have been found for the *Mixed Probit* estimation, the precision criteria are not quite informative in this case.

Table 26. Precision criteria Mixed Probit estimation				
	MAE	RMSE	RMSPE	Time (in minutes)
<i>rmxp</i> function	0.364152	0.478601	0.125429	17.41

Chapter 6 Conclusion

The thesis discussed the estimation results of *Bayesian Hierarchical Logit/Probit Models* using various R packages and function. The estimation results of the simulated dataset suggest that for *Mixed Logit Model* estimation, the estimated individual-level model parameters converge to similar values for *bayesm*, *ChoiceModelR*, and *RSGHB* packages.

According to the analysis, the *RSGHB* package is considered to be the most accurate package since it has the lowest precision criteria, and the estimation speed is relatively faster than the other two packages. However, when there are a large number of model parameters, the users should avoid making typos for the specifications of variables and likelihood function to ensure the correctness of the estimation results.

The estimation time for the *rhierMnlRwMixture* function from *bayesm* package is the longest among three R packages, especially when estimating large sized datasets. However, this package should still be considered as a more general package to be used for *Mixed Logit* Estimation since it allows the random coefficients to be distributed as a mixture of normals.

As a modification of the *rhierMnlRwMixture* function, the *ChoiceModelR* package does not require the discrete attributes to be effect coded. The input data structure for *ChoiceModelR* package is also simplified as matrix format instead of using data frame which helps to reduce the run time of the model estimation (Burn, 2011). But the analysis of the estimation results indicates that this package gives the lowest precision criteria among three R packages.

For *Mixed Probit Model*, since currently no R package has been found to perform such model estimation, and only one function for the estimation of *Mixed Probit Model* is discussed, further explorations of the R packages for *Mixed Probit Model* estimation is essential so that the precision criteria could be more meaningful.

The user interface implemented in this thesis transforms the given data format into different structures so that less programming steps are needed before starting the model estimation. However, basic knowledge of working with matrices and data frames in R is still quite necessary for using the R packages discussed in this thesis. If more R packages for the estimation of *Mixed Logit* or *Mixed Probit Models* are available in the future, new data transformation functions could be implemented and added to this user interface to make it more comprehensive.

Appendix A - Simulated Data Estimation in R

This appendix provides the R code and the explanations about how to perform the estimation using each R package/function discussed in **Chapter 5**. The *demouserin.rar* file contains the R code which can be used for the Model Estimations. The user is required to have *bayesm*, *ChoiceModelR*, and *RSGHB* packages installed in their R program.

Part I - Mixed Logit Estimation

The user is required to specify the path of the following functions:

The *userin* function is the user interface for the transformation of the data structure

```
source("C:/Users/.../demouserin/userin.R",local=TRUE)
```

The *uncodex* function can be used to transform an effect coded design matrix into the "level" format

```
source("C:/Users/.../demouserin/mxl/uncodex.R",local=TRUE)
```

The *uneffect* function can be used to transform an effect coded attribute into the "level" format.

```
source("C:/Users/.../demouserin/mxl/uneffect.R",local=TRUE)
```

The *compare* function is used to calculate the precision criteria after the *Mixed Logit* estimation

```
source("C:/Users/.../demouserin/useroutput/compare.R",local=TRUE)
```

The *rmxp* function is used for the estimation of the *Mixed Probit Model*

```
source("C:/Users/.../demouserin/mxp/rmxp.R",local=TRUE)
dyn.load("C:/Users/.../demouserin/mxp/screen.dll")
```

The *compareprobit* function is used to calculate the precision criteria after the *Mixed Probit* estimation

```
source("C:/Users/.../demouserin/useroutput/compareprobit.R",local=TRUE)
```

Before invoking the user interface, the user should specify the setup of the dataset:

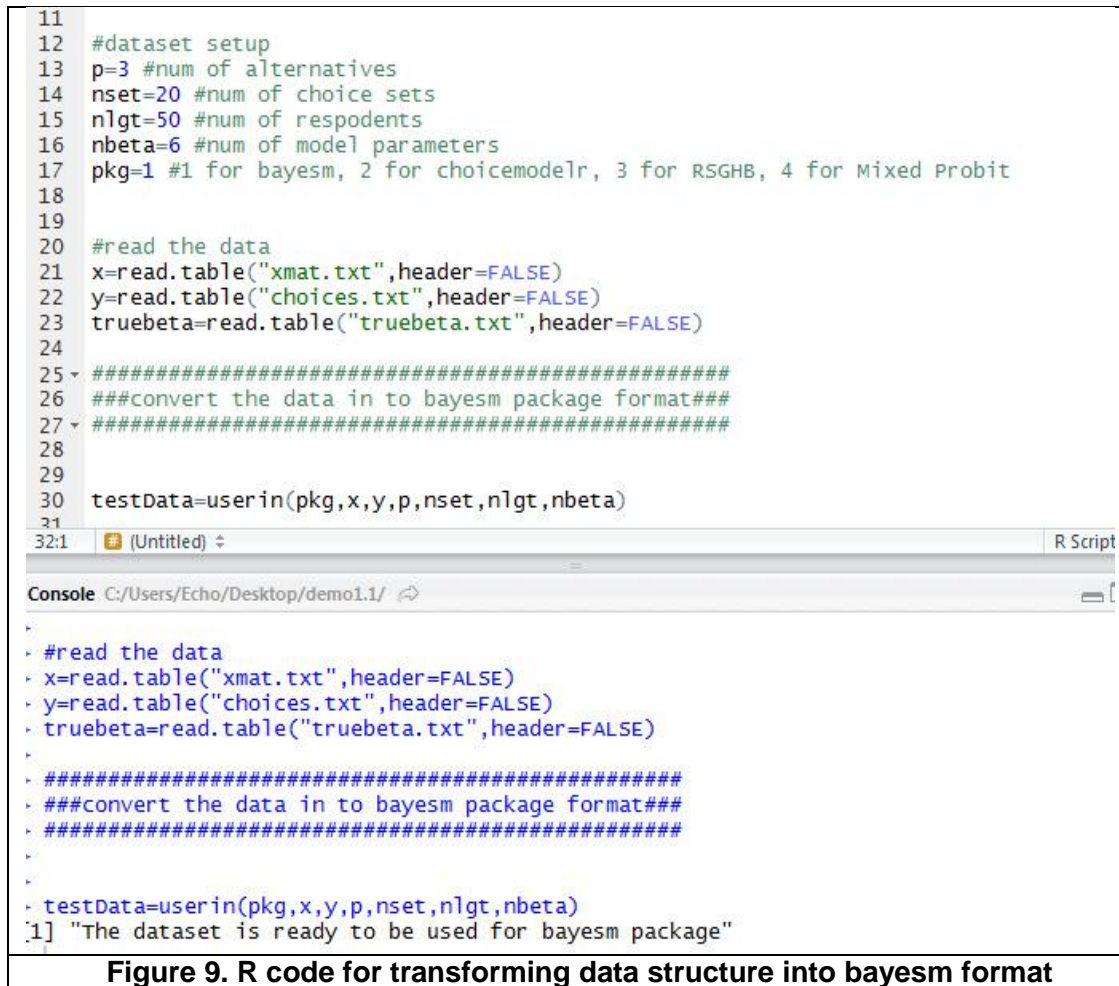
```
p=3 #number of alternatives
nset=20 #number of choice sets
nigt=50 #number of respondents
nbeta=6 #number of model parameters
```

pkg should be specified differently for different packages

```
pkg= 1 for bayesm, 2 for choicemodelr, 3 for RSGHB, 4 for Mixed Probit
```

1. bayesm package

i) Perform Data Transformation (Figure 9)



```

11
12 #dataset setup
13 p=3 #num of alternatives
14 nset=20 #num of choice sets
15 nlgt=50 #num of respodents
16 nbeta=6 #num of model parameters
17 pkg=1 #1 for bayesm, 2 for choicemodelr, 3 for RSGHB, 4 for Mixed Probit
18
19
20 #read the data
21 x=read.table("xmat.txt",header=FALSE)
22 y=read.table("choices.txt",header=FALSE)
23 truebeta=read.table("truebeta.txt",header=FALSE)
24
25 #####
26 ###convert the data in to bayesm package format###
27 #####
28
29
30 testData=userin(pkg,x,y,p,nset,nlgt,nbeta)
31
32:1 (Untitled) R Script

```

```

Console C:/Users/Echo/Desktop/demo1.1/
>
> #read the data
> x=read.table("xmat.txt",header=FALSE)
> y=read.table("choices.txt",header=FALSE)
> truebeta=read.table("truebeta.txt",header=FALSE)
>
> #####
> ###convert the data in to bayesm package format###
> #####
>
>
> testData=userin(pkg,x,y,p,nset,nlgt,nbeta)
[1] "The dataset is ready to be used for bayesm package"

```

Figure 9. R code for transforming data structure into bayesm format

The dataset can be read into R using the following commands:

```

x=read.table("xmat.txt",header=FALSE)
y=read.table("choices.txt",header=FALSE)
truebeta=read.table("truebeta.txt",header=FALSE)

```

The input dataset is transformed to *testData* which is the format required by *bayesm* package using the following command:

```

#pkg=1 indicates bayesm package is selected
testData=userin(pkg=1,x,y,p,nset,nlgt,nbeta)

```

ii) Mixed Logit Model Estimation using bayesm package (Figure 10)

```
#####
###bayesm Estimation###
#####

#1 normal component
Prior1=list(ncomp=1)

#30000 iterations
R=30000
#keeping every 10th draw
keep=10

Mcmc1=list(R=R,keep=keep)

#start model estimation using rhierMnIRwMixture function from bayesm package
out=rhierMnIRwMixture(Data=testData,Prior=Prior1,Mcmc=Mcmc1)
```

Figure 10. R code for model estimation using bayesm package

1 normal component is specified:

Prior1=list(ncomp=1)

Run 30,000 iterations by keeping every 10th draw

Mcmc1=list(R=30000,keep=10)

The model estimation can be started using the *rhierMnIRwMixture* function:

out=rhierMnIRwMixture(Data=testData, Prior= Prior1,Mcmc= Mcmc1)

iii) Output from bayesm package (Figure 11)

```
#####
###bayesm Output###
#####

#obtain the list that contains posterior means for every iteration
mudraw=out$nmix$compdraw

#the first half of the total iterations are treated as the "burn-in" period
#itnum=1500 means the last 1500 iterations are used to calculate the posterior means
itnum=1500
#The first iteration after the "burn-in" period starts at the iteration number ind
ind=3000-itnum+1
#totalit is the ending index from mudraw
totalit=R/keep

#postmu is used to store the posterior means
#for each iteration after the "burn-in" period
postmu=NULL
#the following loop store the posterior means from iteration 1501 to 3000
for (i in ind:totalit){
  #mudraw[[i]][[1]]$mu gives the posterior means at iteration i
  #for the first normal mixture component
  #(Note: the thesis is only dealing with 1 normal component)
  postmu=rbind(postmu,mudraw[[i]][[1]]$mu)
  #postmu is a 1500 by 6 matrix
  #postmu contains the posterior means from iteration 1501 to 3000
}
#trace plot after the burn-in period
matplot(postmu, type="l",main="Trace Plot of bayesm package")

#estbeta is the individual-level parameter estimates from iteration 1501 to 3000
estbeta<-apply(out$betadraw[,1501:3000], c(1,2), mean)
write.csv(estbeta, file = "bayesmbeta.csv", row.names = FALSE)

#calculate precision criteria
compare(p,nset,nlgt,estbeta,truebeta,x)
```

Figure 11. R code for obtaining estimation results using bayesm package

The list that contains the posterior means for every iteration can be obtained using the following command:

```
mudraw=out$nmix$compdraw
```

itnum indicates the number of iterations used for posterior mean calculation after the burn-in period

```
itnum=1500
```

The first iteration after the "burn-in" period starts at the *ind*th iteration:

```
ind=3000-itnum+1
```

The last iteration after the "burn-in" period is the *totalit*th iteration:

```
totalit=R/keep
```

postmu is used to store the posterior means for each iteration after the burn-in period:

```
postmu=NULL
```

Since the thesis is only dealing with 1 normal component, *mudraw[[i]][[1]]\$mu*

gives the posterior means at iteration i for the first normal mixture component.

The following loop store the posterior means from iteration 1501 to 3000.

```
for (i in ind:totalit){
  postmu=rbind(postmu,mudraw[[i]][[1]]$mu)
}
```

At the end of the loop, *postmu* contains the posterior means from iteration 1501 to 3000.

The trace plot can be plotted using the following command:

```
matplot(postmu, type="l",main="Trace Plot of bayesm package")
```

The user could obtain the individual-level parameter estimates using the *apply* function from R. The *out\$betadraw* is a $nlgt \times nbeta \times R/keep$ dimension array, in order to obtain the individual-level parameter estimates averaging over iterations, the following command can be used:

```
estbeta<-apply(out$betadraw, c(1,2), mean)
```

If the first 1500 iterations are discarded as the burn-in period, the user can obtain the individual-level parameter estimates based on the iterations after the burn-in period using the following command:

```
estbeta<-apply(out$betadraw[,1501:3000], c(1,2), mean)
```

The estimated individual-level parameter estimates can be saved as a csv file using the following command:

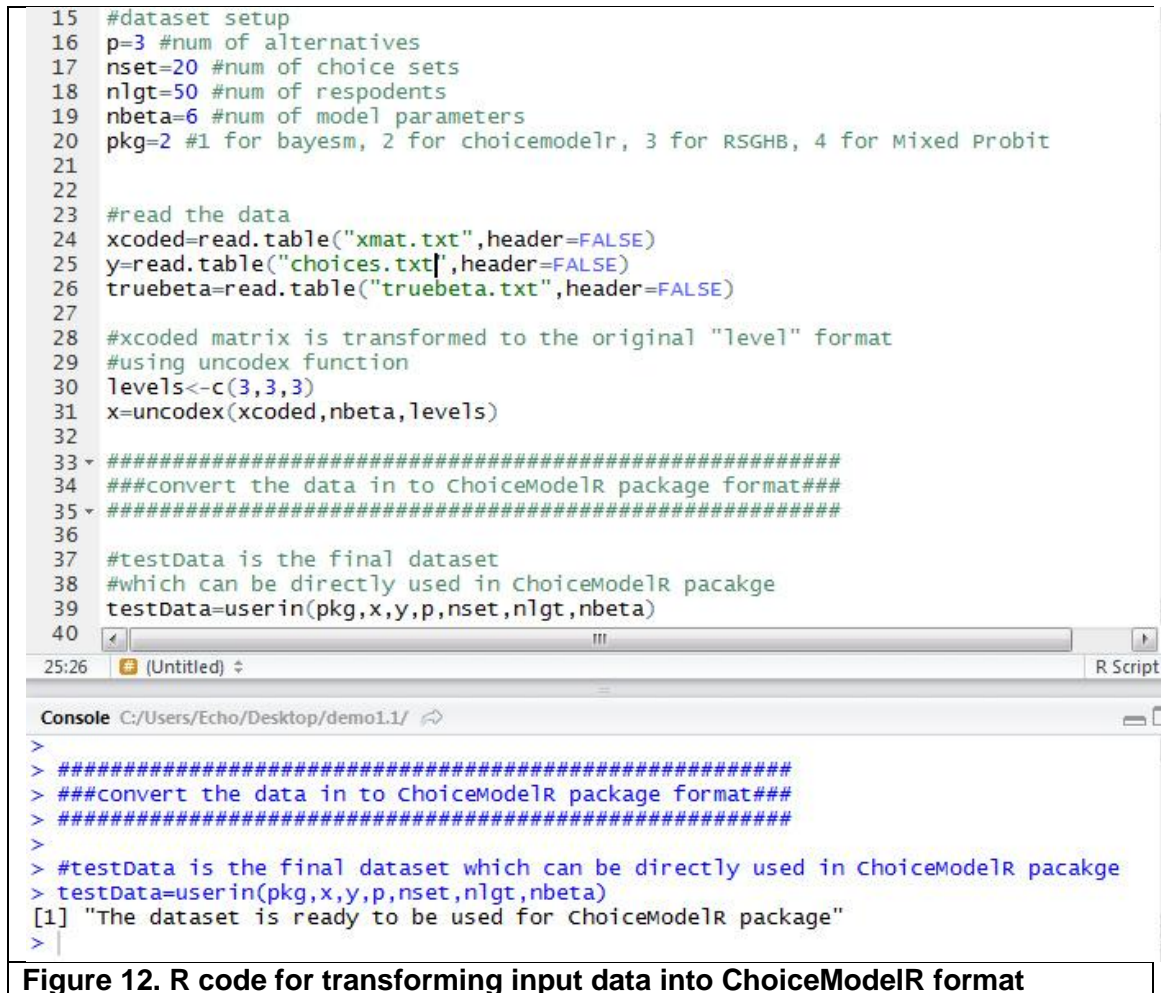
```
write.csv(estbeta, file = "bayesmbeta.csv", row.names = FALSE)
```

The precision criteria can be calculated using the following command:

```
compare(p,nset,nlgt,estbeta,truebeta,x)
```

2. ChoiceModelR package

i) Perform Data Transformation (Figure 12)



```

15 #dataset setup
16 p=3 #num of alternatives
17 nset=20 #num of choice sets
18 nlgt=50 #num of respondents
19 nbeta=6 #num of model parameters
20 pkg=2 #1 for bayesm, 2 for choicemodelr, 3 for RSGHB, 4 for Mixed Probit
21
22
23 #read the data
24 xcoded=read.table("xmat.txt",header=FALSE)
25 y=read.table("choices.txt",header=FALSE)
26 truebeta=read.table("truebeta.txt",header=FALSE)
27
28 #xcoded matrix is transformed to the original "level" format
29 #using uncodex function
30 levels<-c(3,3,3)
31 x=uncodex(xcoded,nbeta,levels)
32
33 #####
34 ###convert the data in to ChoiceModelR package format###
35 #####
36
37 #testData is the final dataset
38 #which can be directly used in ChoiceModelR package
39 testData=userin(pkg,x,y,p,nset,nlgt,nbeta)
40

```

25:26 (Untitled) R Script

Console C:/Users/Echo/Desktop/demo1.1/

```

>
> #####
> ###convert the data in to ChoiceModelR package format###
> #####
>
> #testData is the final dataset which can be directly used in ChoiceModelR package
> testData=userin(pkg,x,y,p,nset,nlgt,nbeta)
[1] "The dataset is ready to be used for ChoiceModelR package"
>

```

Figure 12. R code for transforming input data into ChoiceModelR format

The user specify $pkg=2$ since *ChoiceModelR* package is going to be used for the model estimation:

The effect coded x matrix is transformed to the original "level" format using *uncodex* function:

```
levels<-c(3,3,3)
x=uncodex(x,nbeta,levels)
```

In order to transform the input data into the *ChoiceModelR* format, specify $pkg=2$, and the *testData* from the following command is the final dataset which can be directly used in *ChoiceModelR* package:

```
testData=userin(pkg=2,x,y,p,nset,nlgt,nbeta)
```


ii) Mixed Logit Model Estimation using ChoiceModelR package (Figure 13)

```
#####
##ChoiceModelR Estimation##
#####

#several options should be specified before starting the model estimation
#3 discrete attributes, the values for "xcoding" are all zeros
xcoding=c(0,0,0)

#indicate mcmc parameters
#30000 iterations in total
#the first half of the 30000 iterations are discarded as the "burn-in" period
mcmc=list(R=30000,use=15000)
#keep 10 random draws to save
options=list(none=FALSE, save=TRUE, keep=10)

#start model estimation by calling "choicemodelr" function from ChoiceModelR package
out=choicemodelr(testData,xcoding,mcmc=mcmc,options=options)
```

Figure 13. R code for model estimation using ChoiceModelR package

There are 3 discrete attributes in the dataset, therefore, the values for *xcoding* are all zeros:

```
xcoding=c(0,0,0)
```

The following iteration setting is used for the model estimation. *R=30000* indicates that 30,000 iterations are performed, *use=15000* indicates that the last 15,000 iterations are used for the estimation of the calculation of the posterior means:

```
mcmc=list(R=30000,use=15000)
```

keep 10 random draws to save

```
options=list(none=FALSE, save=TRUE, keep=10)
```

The model estimation can be started by calling the *choicemodelr* function from the *ChoiceModelR* package using the following command:

```
out=choicemodelr(testData,xcoding,mcmc=mcmc,options=options)
```


iii) Output from ChoiceModelR package (Figure 14)

```
#####
###ChoiceModelR Output###
#####

#initialize posterior mean matrix
postmu=NULL
#postmu is the matrix which contains the posterior mean for each iteration
#after the burn-in period
for(i in 1:1500)
{
  #compdraw[[i]][[1]] gives the posterior mean for iteration i
  postmu=rbind(postmu,out$compdraw[[i]][[1]])
}

#create trace plot
matplot(postmu, type="l",main="Trace Plot of ChoiceModelR package")

#Please discard Rbeta file after the estimation
#read the posterior means for each individual
estbeta<-apply(out$betadraw, c(1,2), mean)

#create individual-level posterior means
write.csv(estbeta, file = "ChoiceModelRbeta-uneffect.csv", row.names = FALSE)

#calculate precision criteria
#Note: when calculating the precision criteria, use the xcoded matrix
compare(p,nset,nlgt,estbeta,truebeta,xcoded)
```

Figure 14. R code for obtaining estimation results using ChoiceModelR package

The `out$compdraw[[i]][[1]]` command can be used to obtain the posterior means for the i^{th} iteration.

The following loop stores the posterior means for each iteration after the burn-in period in the `postmu` matrix

```
postmu=NULL
for(i in 1:1500)
{
  postmu=rbind(postmu,out$compdraw[[i]][[1]])
}
```

The trace plot can be plotted using the following command:

```
matplot(postmu, type="l",main="Trace Plot of ChoiceModelR package")
```

The individual-level parameter estimates can be obtained using the following command, note that the *ChoiceModelR* package automatically used the iterations after the burn-in period to calculate the individual-level parameter estimates.

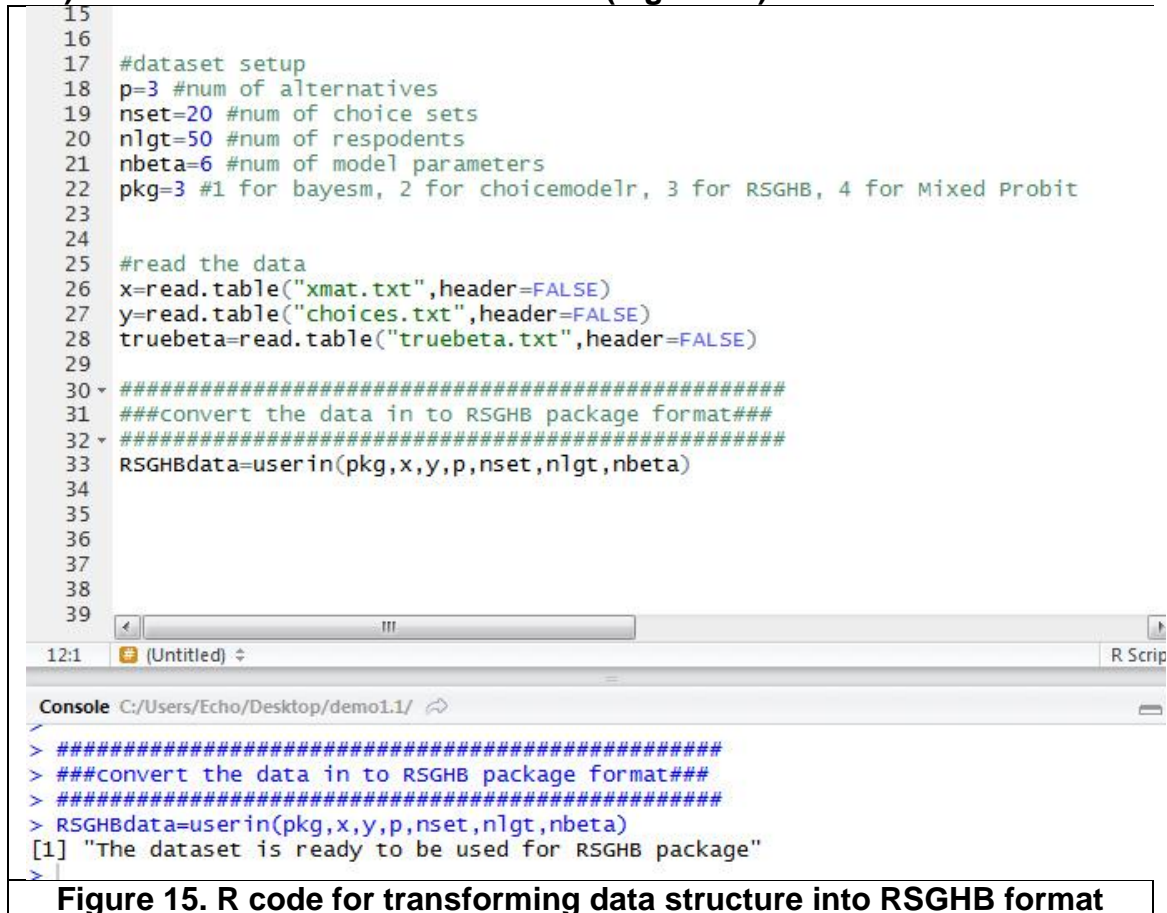
```
estbeta<-apply(out$betadraw, c(1,2), mean)  
write.csv(estbeta, file = "ChoiceModelRbeta.csv", row.names = FALSE)
```

The precision criteria can be calculated using the following command. **Note:** the user should use the original effect coded x matrix for the calculation of the precision criteria.

```
compare(p,nset,nlgt,estbeta,truebeta,xcoded)
```

3. RSGHB package

i) Perform Data Transformation (Figure 15)



```

15
16
17 #dataset setup
18 p=3 #num of alternatives
19 nset=20 #num of choice sets
20 nlgt=50 #num of respondents
21 nbeta=6 #num of model parameters
22 pkg=3 #1 for bayesm, 2 for choicemodlr, 3 for RSGHB, 4 for Mixed Probit
23
24
25 #read the data
26 x=read.table("xmat.txt",header=FALSE)
27 y=read.table("choices.txt",header=FALSE)
28 truebeta=read.table("truebeta.txt",header=FALSE)
29
30 ~ #####
31 ~ ###convert the data in to RSGHB package format###
32 ~ #####
33 RSGHBdata=userin(pkg,x,y,p,nset,nlgt,nbeta)
34
35
36
37
38
39

```

12:1 (Untitled) R Script

Console C:/Users/Echo/Desktop/demo1.1/

```

> #####
> ###convert the data in to RSGHB package format###
> #####
> RSGHBdata=userin(pkg,x,y,p,nset,nlgt,nbeta)
[1] "The dataset is ready to be used for RSGHB package"
>

```

Figure 15. R code for transforming data structure into RSGHB format

The input dataset is transformed in the format required by *RSGHB* package using the following command (specify *pkg=3* for *RSGHB* package), the *RSGHBdata* can be used for the model estimation.

RSGHBdata=userin(pkg=3,x,y,p,nset,nlgt,nbeta)

Specify variables to construct likelihood function (Figure 16)

```

#attrnum is number of columns in the attribute section
#it equals total number of columns of RSGHBdata -3
#since the first two columns are ID and choice set
#the last one is choice column

last=ncol(RSGHBdata)
attrnum=last-3

#specify variables for utility functions
RSGHBattr=NULL
for(attr in 1:attrnum)
{RSGHBattr[[attr]]=RSGHBdata[[attr+2]]}
|
nr=nrow(RSGHBdata)
xvar=array(rep(0,p*nr*nbeta), dim=c(p,nr,nbeta))

for(i in 1:nbeta){

  temp=NULL
  for (j in 1:p){
    index=j+p*(i-1)
    temp=rbind(temp,RSGHBattr[[index]])

  }#end of for j
  xvar[, ,i]=temp
}#end of for i

#specify choice vectors for p alternatives
RSGHBy=NULL
for(alt in 1:p)
{
  RSGHBy[[alt]]=(RSGHBdata[[last]]==alt)
}

```

Figure 16. R code for variable specifications in RSGHB package

The *RSGHB* package requires the user to specify the likelihood function, the original way of specifying the model variables which is discussed in **Chapter 4** is provided in the documentation of *RSGHB* package, but it can be time consuming if the dataset contains many parameters. Therefore, in order to avoid making errors and simplify the steps, the data frames are used to define the variables.

attrnum is the number of columns in the attribute section of the *RSGHBdata*, since the first two columns are ID and choice set, the last one is choice column, *attrnum* = total number of columns of *RSGHBdata* -3.

```
last=ncol(RSGHBdata)
attrnum=last-3
```

The following loop stores the attribute variables in a data frame named *RSGHBattr*, the following commands are used for illustrations about how the attribute variables are defined, the user does **NOT** need to make any changes to the following command for the estimation of different datasets.

```
RSGHBattr=NULL
for(attr in 1:attrnum)
{RSGHBattr[[attr]]=RSGHBdata[[attr+2]]}
```

The attribute variables are stored in a multidimensional array *xvar*, this array is going to be used for the specification of the likelihood function.

```
nr=nrow(RSGHBdata)
xvar=array(rep(0,p*nr*nbeta), dim=c(p,nr,nbeta))
```

```
for(i in 1:nbeta){
  temp=NULL
  for (j in 1:p){
    index=j+p*(i-1)
    temp=rbind(temp,RSGHBattr[[index]])
  }#end of for j
  xvar[,i]=temp
}#end of for i
```

For the specification of the likelihood function, the user is also required to specify a list of the choice *RSGHBy* using the following loop.

```
RSGHBy=NULL
for(alt in 1:p)
{
  RSGHBy[[alt]]=(RSGHBdata[[last]]==alt)
}
```

Specify control parameters (Figure 17)

```
#####
##Specify control parameters##
#####

#Note: the documentation of the control parameters
#is written by the author of RSGHB package
#####

#specify model name
modelName<-"Simulated Mixed Logit Model"

#names for the random parameters
gVarNamesNormal<-c("A","B","C","D","E","F")
#distributions for the random parameters
gDIST<-c(1,1,1,1,1,1)

# svN contains the starting values for the means of the normal distributions for each of the random parameters
svN <- c(0,0,0,0,0,0)
# ITERATION SETTINGS

# gNCREP contains the number of iterations to use prior to convergence
gNCREP <- 15000
# gNEREP contains the number of iterations to keep for averaging after convergence has been reached
gNEREP <- 1500
# gNSKIP contains the number of iterations to do in between retaining draws for averaging
gNSKIP <- 10
# gINFOSKIP controls how frequently to print info about the iteration process
gINFOSKIP <- 250

# To simplify the doHB functional call, we put all of the control parameters into a single list
control <- list(modelname=modelname,gVarNamesNormal=gVarNamesNormal,
               gDIST=gDIST,svN=svN,gNCREP=gNCREP,gNEREP=gNEREP,
               gNSKIP=gNSKIP,gINFOSKIP=gINFOSKIP)
```

Figure 17. R code for control parameter specifications in RSGHB package

Note: the documentation of the control parameters is written based on the original documentation provided by the author of *RSGHB* package

The user could specify the name of the estimated model using the following command:

```
modelName<-"Simulated Mixed Logit Model"
```

The names of the model parameters are required to be specified using characters. There are 6 parameters in the model, therefore the *gVarNamesNormal* vector contains 6 parameter names.

```
gVarNamesNormal<-c("A","B","C","D","E","F")
```

The model parameters are all assumed to have normal priors, therefore, each element of *gDIST* is specified as "1", there are 6 elements in this vector.

```
gDIST<-c(1,1,1,1,1,1)
```

svN contains the starting values for the means of the normal distributions for each of the random parameters

```
svN <- c(0,0,0,0,0,0)
```

The number of iterations used for the burn-in period can be specified as follows:

```
gNCREP <- 15000
```

The number of iterations used for the estimation of model parameters after the

convergence:

```
gNEREP <- 1500
```

The following command can be used for keeping every 10th draw of the iterations after convergence for the calculation of the posterior mean:

```
gNSKIP <- 10
```

The user could choose the frequency of the screen update, the following command can be used to display the information about the iteration process for every 250 iterations:

```
gINFOSKIP <- 250
```

The control setting is stored in the following list:

```
control <-
```

```
list(modelname=modelname,gVarNamesNormal=gVarNamesNormal,gDIST=gDIST,svN=svN,gNCREP=gNCREP,gNEREP=gNEREP,gNSKIP=gNSKIP,gINFOSKIP=gINFOSKIP)
```

Likelihood Function specification for RSGHB package (Figure 18)

```
#####
###RSGHB likelihood function###
#####

likelihood <- function(fc,b)
{
  cc <- 1

  beta1 <- b[,cc]; cc<-cc+1
  beta2 <- b[,cc]; cc<-cc+1
  beta3 <- b[,cc]; cc<-cc+1
  beta4 <- b[,cc]; cc<-cc+1
  beta5 <- b[,cc]; cc<-cc+1
  beta6 <- b[,cc]; cc<-cc+1

  #utility functions
  v1 <- beta1*(xvar[1,,1])+beta2*(xvar[1,,2])+beta3*(xvar[1,,3])+beta4*(xvar[1,,4])+beta5*(xvar[1,,5])+beta6*(xvar[1,,6])
  v2 <- beta1*(xvar[2,,1])+beta2*(xvar[2,,2])+beta3*(xvar[2,,3])+beta4*(xvar[2,,4])+beta5*(xvar[2,,5])+beta6*(xvar[2,,6])
  v3 <- beta1*(xvar[3,,1])+beta2*(xvar[3,,2])+beta3*(xvar[3,,3])+beta4*(xvar[3,,4])+beta5*(xvar[3,,5])+beta6*(xvar[3,,6])

  #mnl probability statement
  p <- (exp(v1)*RSGHBy[[1]] + exp(v2)*RSGHBy[[2]]+exp(v3)*RSGHBy[[3]])/ (exp(v1) + exp(v2) + exp(v3))

  return(p)
}#end of likelihood setup
```

Figure 18. R code for Likelihood Function specification in RSGHB package

```
likelihood <- function(fc,b)
```

```
{
```

Since a *Mixed Logit Model* is estimated, the user is only required to use *b*. *fc* can be discarded since it is used for a *Fixed Effect Logit Model*.

#using cc var to index the b vector simplifies the addition/subtraction of new parameters

```
cc <- 1
```

The model parameters should be specified in the following format. In the

example, there are 6 model parameters, therefore 6 vectors are created using the following command:

```
beta1 <- b[,cc];cc<-cc+1
beta2 <- b[,cc];cc<-cc+1
beta3 <- b[,cc];cc<-cc+1
beta4 <- b[,cc];cc<-cc+1
beta5 <- b[,cc];cc<-cc+1
beta6 <- b[,cc];cc<-cc+1
```

The utility function is specified using the following command:

```
v1 <-
beta1*(xvar[1,,1])+beta2*(xvar[1,,2])+beta3*(xvar[1,,3])+beta4*(xvar[1,,4])+beta5
*(xvar[1,,5])+beta6*(xvar[1,,6])
v2 <-
beta1*(xvar[2,,1])+beta2*(xvar[2,,2])+beta3*(xvar[2,,3])+beta4*(xvar[2,,4])+beta5
*(xvar[2,,5])+beta6*(xvar[2,,6])
v3 <-
beta1*(xvar[3,,1])+beta2*(xvar[3,,2])+beta3*(xvar[3,,3])+beta4*(xvar[3,,4])+beta5
*(xvar[3,,5])+beta6*(xvar[3,,6])
```

Finally the user is required to specify the likelihood function:

```
p <- (exp(v1)*RSGHBy[[1]] + exp(v2)*RSGHBy[[2]]+exp(v3)*RSGHBy[[3]])/
(exp(v1) + exp(v2) + exp(v3))

return(p)

}#end of likelihood setup
```

Model Estimation using RSGHB package

The estimation can be performed using the following command:

```
doHB(likelihood,RSGHBdata,control)
```


Output from RSGHB package (Figure 19)

```
#####
###RSGHB Output###
#####

#read estimated beta values
#the user should rename the "B_file" as "RSGHBbeta"
dat<- read.csv(file="RSGHBbeta.csv",head=TRUE)
cn=ncol(dat)

estbeta=dat[,2:cn]
estbeta=t(t(estbeta))

#trace plot after burn-in period
#the user should rename the "A_file" as "RSGHBit"
ite<- read.csv(file="RSGHBit.csv",head=TRUE)
trace=ite[,2:cn]
matplot(trace, type="l",main="Trace Plot of RSGHB package")
dev.copy(png,paste("RSGHB trace plot.png",sep=""))
dev.off()

#calculate comparison criteria
compare(p,nset,nlgt,estbeta,truebeta,x)
```

Figure 19. R code for model estimation in RSGHB package

For *RSGHB* package, the estimated *posterior means* are saved automatically to the working dictionary after the model estimation as a csv file, in order to do further calculations, the user should manually read the file to R. It is suggested to first rename the *B_file* as *RSGHBbeta* in the working dictionary, then use the following command to read the file in R:

```
dat<- read.csv(file="RSGHBbeta.csv",head=TRUE)
cn=ncol(dat)
```

The *estbeta* is the individual-level parameter estimates, this is used for the estimation of the precision criteria in the last step.

```
estbeta=dat[,2:cn]
```

This step transform *estbeta* into the matrix format:

```
estbeta=t(t(estbeta))
```

In order to obtain the trace plot after the burn-in period, , the user should rename the *A_file* as *RSGHBit*, then use the following command:

```
ite<- read.csv(file="RSGHBit.csv",head=TRUE)
trace=ite[,2:cn]
matplot(trace, type="l",main="Trace Plot of RSGHB package")
```

The following commands export the trace plot and saves the plot directly to the working dictionary with name "RSGHB trace plot":

```
dev.copy(png,paste("RSGHB trace plot.png",sep=""))
dev.off()
```

The precision criteria can be calculated using the following command:

```
compare(p,nset,nlgt,estbeta,truebeta,x)
```

Part II - Mixed Probit Estimation

The user should read in the data:

```
x=read.table("orth_xmat_100resp.txt",header=FALSE)
y=read.table("ymxp.txt",header=FALSE)
truebeta=read.table("truebetamxp.txt",header=FALSE)
```

The *xdf* matrix is the difference system using alternative *p* as the reference category, and this matrix is needed for the calculation of precision criteria. The user could use the *getdiff* function (see section 5.1) to transform the input *x* matrix into the *differenced* system.

```
xdf=read.table("xdf_orth.txt",header=FALSE)
```

i) Perform Data Transformation

The input dataset is transformed to the format required by *rmxp* function using the following command (set *pkg=4*), the *testData* can be used for the model estimation:

```
testData=userin(pkg=4,x,y,p,nset,nlgt,nbeta)
```

ii) Estimation using *rmxp* function (Figure 20)

```
#####
###specify priors and MCMC parameters###
#####
nu=nbeta+5
Prior=list(nu=nu,V0=nu*diag(rep(1,nbeta)),
           betabarbar=as.vector(rep(0,nbeta)),
           Abeta=.01*diag(rep(1,nbeta)))
Mcmc=list(R=30000,keep=10)

#start model estimation###
out=rmxp(testData,Prior,Mcmc)
```

Figure 20. R code for model estimation using *rmxp* function

The user could specify the priors and MCMC parameters using the following command:

```
nu=nbeta+5
Prior=list(nu=nu,V0=nu*diag(rep(1,nbeta)),
           betabarbar=as.vector(rep(0,nbeta)),
```

```
Abeta=.01*diag(rep(1,nbeta)))
```

30,000 iterations are used, keeping every 10th draw.

```
Mcmc=list(R=30000,keep=10)
```

The model estimation can be started using the *rmxp* function:

```
out=rmxp(testData,Prior,Mcmc)
```

iii) Output from *rmxp* function (Figure 21)

```
#####
###Mixed Probit output###
#####

#mudraw is an R/keep by nbeta matrix
#mudraw contains the posterior means for each iteration
mudraw=out$betabardraw

#the first half of the total iterations are treated as the "burn-in" period
#postmu is a matrix which contains the posterior means from iteration 1501 to 3000
postmu=mudraw[1501:3000,]

#trace plot after the burn-in period
matplot(postmu, type="l",main="Trace Plot of rmxp function")

#The individual-level parameter estimates from iteration 1501 to 3000
estbeta<-apply(out$betadraw[,1501:3000], c(1,2), mean)
write.csv(estbeta, file = "Mixed Probit Betas.csv", row.names = FALSE)

#calculate precision criteria
#for simulated data, the user should provide the variance of the error terms
Sigma=matrix(c(1,0.5,0.5,1),ncol=2)

#Note: when calculating precision criteria, xdf matrix is needed
compareprobit(estbeta,truebeta,Sigma,xdf,nlgt,p,nset)
```

Figure 21. R code for obtaining estimation results using *rmxp* function

The posterior means for each iteration can be obtained using the *out\$betabardraw* command. This command returns a matrix which contains the posterior means for each iteration. In this example, *mudraw* is a 3000 × 5 matrix:
mudraw=out\$betabardraw

postmu is a matrix which contains the posterior means from iteration 1501 to 3000:

```
postmu=mudraw[1501:3000,]
```

The trace plot after the burn-in period can be obtained using the following command:

```
matplot(postmu, type="l",main="Trace Plot of rmxp function")
```

The individual-level parameter estimates from iteration 1501 to 3000 can be obtained using the following command:

```
estbeta<-apply(out$betadraw[,1501:3000], c(1,2), mean)
write.csv(estbeta, file = "Mixed Probit Posterior Means.csv", row.names = FALSE)
```

In order to calculate precision criteria of the *Mixed Probit Model*, the user is required to provide the covariance matrix of the error terms used for the data simulation, the following Sigma for the *Mixed Probit* data simulation is used.
`Sigma=matrix(c(1,0.5,0.5,1),ncol=2)`

For the calculation of the precision criteria of the *Mixed Probit Model*, the *differenced system* of the original x matrix should be used. The reason is because in order to obtain the probabilities of the *Mixed Probit Model*, the *mnp* function from *bayesm* package is used, and this function requires the x matrix to be expressed in a *differenced system*. The following command is used to calculate the precision criteria of *Mixed Probit Model*:
`compareprobit(estbeta,truebeta,Sigma,xdf,nlgt,p,nset)`

Appendix B – Additional R code

Appendix B contains the R code of the functions implemented in this thesis.

<i>userin</i>	This function transforms the input data into different formats required by different R packages/function.
---------------	---

```
#####
#The user is required to have 3 packages installed#
#####

userin=function(pkg,x,y,p,nset,nlgt,nbeta){

  #####
  #Purpose:
  #
  #An user interface to convert the data into desired format
  #
  #Note: The user should have bayesm,choicemodelr,RSGHB packages installed
  #
  #Author:
  #
  # Hanshui Zhang (Last Modified on July 25, 2013)
  #
  #Arguments:
  #
  # pkg: package indicator, the user could choose a package for the estimation
  # by entering number 1, 2, or 3
  #   1 - bayesm package
  #   2 - ChoiceModelR package
  #   3 - RSGHB package
  #   4 - Mixed Probit
  #
  # p: num of alternatives
  # nset: num of choice sets
  # nlgt: num of respondents
  # nbeta: num of model parameters to be estimated
  #
  #x: design matrix which contains the attributes
  # if the user is using bayesm/RSGHB, effect coding should be done before
  # entering x
  #
  #y: (nset*nlgt) by 1 matrix, this matrix contains the responses made by the
  # respondents
  #
  #####
}
```

```

if(pkg==1){

  library(bayesm)
  print("The dataset is ready to be used for bayesm package")

  #####
  ###an input procedure for bayesm package###
  #####

  bayesmin=function(x,y,p,nset,nlgt){

    #transform y and x into matrix form
    y=t(t(y))
    x=t(t(x))

    lgtdata=NULL

    ni=rep(nset,nlgt)
    csa=nset*p
    for (i in 1:nlgt)
    {
      #obtain y
      ychoice=NULL
      ybeg=nset*(i-1)+1
      yend=nset*i
      for(c in 1:nset){ychoice[1:nset]=y[ybeg:yend]}

      #transform x into dataframe
      xmat=NULL
      xbeg=csa*(i-1)+1
      xend=csa*i
      xmat[[i]]=x[xbeg:xend,]

      lgtdata[[i]]=list(y=ychoice,X=xmat[[i]])
    }
    #end of converting data

    #the bayesmin function returns a list of 2
    return(bayesmdata=list(p=p,lgtdata=lgtdata))

  }#end of bayesmin function

  #if user call pkg 1, the data will be converted to bayesm format

```

```

return(bayesmin(x,y,p,nset,nlgt))

}#end of calling bayesm package

else if(pkg==2){

  library(ChoiceModelR)
  print("The dataset is ready to be used for ChoiceModelR package")

  #####
  ###an input procedure for ChoiceModelR package###
  #####

  choicemodelrin=function(x,y,p,nset,nlgt){

    #transform y and x into matrix form
    y=t(t(y))
    x=t(t(x))

    #input procedure for ChoiceModelR
    #setup choiceset column for each individual
    #choiceset row, total num of rows=p*nset*nlgt
    set=rep(1:nset, each = p, times=nlgt)

    #id row, total number of rows=p*nset*nlgt
    id=rep(1:nlgt,each=nset*p)

    #create alternative indicator, total number of rows=p*nset*nlgt
    alt=rep(c(1:p), nset*nlgt)

    #beginning of the matrix
    initialmat=t(rbind(id, set,alt))

    #combine xmat and attrmat
    xmat=cbind(initialmat,x)

    #make choice columns
    newchoice=y
    zeromat=matrix(0,nset*nlgt,p-1)
    choicemat=cbind(newchoice,zeromat)

    #This is the final y column representing choice
    choicecol=matrix(c(t(choicemat)))

```

```

#bind everything together
return(choicemodelrdata=cbind(xmat,choicecol))

}#end of choicemodelrin function

#if user call pkg 2, the data will be converted to ChoiceModelR format
return(choicemodelrin(x,y,p,nset,nlgt))

}#end of calling ChoiceModelR

else if (pkg==3){
  library(RSGHB)
  print("The dataset is ready to be used for RSGHB package")

  #####
  ###an input procedure for RSGHB package###
  #####

  rsghbin=function(x,y,p,nset,nlgt,nbeta){

    #transform y and x into matrix form
    y=t(t(y))
    x=t(t(x))

    #nbeta is number of model parameters
    nbeta=ncol(x)

    #RSGHB id row, total number of rows=p*nset*nlgt
    rsghbid=rep(1:nlgt,each=nset)

    #define choiceset column
    ncs=rep(1:nset, times=nlgt)

    #first 2 columns of the RSGHB data structure
    rsghbinitialmat=t(rbind(rsghbid,ncs))

    #attribute matrix
    rsghbattrmat=NULL
    indset=nset*nlgt
    for (cs in 1:indset){
      beg=p*(cs-1)+1
      end=p*cs

      xtemp=NULL
      for(col in 1:nbeta){

```



```

    xtemp=cbind(xtemp,t(x[beg:end,col]))
  }
  rsghbattrmat=rbind(rsghbattrmat,xtemp)
}

#choice column
RSGHBchoice=y

#combine columns to form the dataset
RSGHBdata=data.frame(cbind(rsghbinitialmat,rsghbattrmat,RSGHBchoice))

#first column is named as "ID", second column is called "choice set"
colnames(RSGHBdata)[[1]]="ID"
colnames(RSGHBdata)[[2]]="Choice Set"

#name the last column (choice column) of the data structure
cy=ncol(RSGHBdata)
colnames(RSGHBdata)[[cy]]="Choice"

#the RSGHB data is returned as a data frame
return(RSGHBdata)

}#end of rsghbin function

#if user call pkg 3, the data will be converted to RSGHB format
return(rsghbin(x,y,p,nset,nlgt,nbeta))

}#end of calling RSGHB

else if (pkg==4){
  print("The dataset is ready to be used for Mixed Probit Estimation")
  library(bayesm)

  #####
  ###an input procedure for rmxp function###
  #####
  mxpin=function(x,y,p,nset,nlgt,nbeta){

    x=t(t(x))
    y=t(t(y))

    #mxpin function transform the input data into the structure which can be used
    in rmxp function
    #code y matrix using indicator variables

```

```

ynum=nrow(y)
yind=NULL
for (i in 1:ynum){
  zerotemp=matrix(0,p,1)
  index=y[i,]
  zerotemp[index]=1
  yind=rbind(yind,zerotemp)
}#end for i

#format y using the coded y matrix
y = array(t(yind),dim=c(p,nset,nlgt))

X = array(t(x),dim=c(nbeta,p,nset,nlgt))

return(Data=list(y=y,X=X,nlgt=nlgt,nset=nset,p=p,nbeta=nbeta))

}# end of mxpin

return(mxpin(x,y,p,nset,nlgt,nbeta))

}# end else if pkg==4

else{

  return(print("please specify: 1-bayesm, 2-ChoiceModelR, 3-RSGHB, 4-Mixed
Probit") )

}#end else

}#end of userin function

```

uncodex

This function converts an effect coded design matrix into the original "level" format.

```

uncodex=function(x,nbeta,levels){
#####
#Purpose:
#
#A function which convert a design matrix from effect coding to the original
levels
#NOTE: this function assumes that the x matrix only contains discrete attributes
#Author:
#

```

```

# Hanshui Zhang (Last Modified on July 31, 2013)
#
#Arguments:
#
# x: design matrix which only contains the effect coded attributes
# nbeta: total number of model parameters
# levels: a vector of attributes
# if there are 3 attributes in the design matrix, each 3 levels
# specify levels<-(3,3,3)
# Note: the order of the attributes of levels should be the same as the design
matrix

#####

#indicate number of attributes
numattr=length(levels)

#number of parameters in the design matrix
nparam=levels-1

#for more than 1 attributes
if(numattr>1){

  uncodedattr=NULL
  uncodedx=NULL
  for (i in 1:numattr){

    #the ending index of each attribute matrix
    #if i is 1 then ending index is the first element of nparams
    if (i==1){endi=nparam[i]}
    #if i is not 1 then ending index is the sum of the first i elements of nparams
    if (i!=1){endi=apply(matrix(nparam[1:i]),2,sum)}
    #define beginning index for attribute i
    begi=endi-(nparam[i]-1)

    if(begi==endi){attrbi=x[begi]}#end of if (beg==end)
    else {attrbi=x[,begi:endi]}#end of else

    uncodedattr[[i]]=uneffect(attrbi,levels[i])
    uncodedx=cbind(uncodedx,uncodedattr[[i]])
  }#end of looping through each column of x matrix

}#end if numnattr>1

#if only 1 attribute
if(numattr==1){

```

```

uncodedx=uneffect(x,levels)

}#end if numattr==1

return(uncodedx)
}#end of uncodedx

```

uneffect

This function converts 1 attribute from effect coding to the original “level” format.

```

uneffect= function(attr,l) {
#####
#Purpose:
#
#A function which convert an attribute from effect coding to the original levels
#
#Author:
#
# Hanshui Zhang (Last Modified on July 18, 2013)
#
#Arguments:
# attr: matrix which contains one effect coded attribute
# l: total number of levels of attr

#####

#get number of rows
nr=nrow(attr)
#get number of columns
nc=ncol(attr)

#initialize a nr by 1 matrix of zeros to store the attribute levels
xuneffect=mat.or.vec(nr,1)

#looping through each row of matrix attr
for(i in 1:nr){

  #looping through each column within row i
  for(j in 1:nc){

    #if an element from attr is -1, then this element has the highest level
    if (attr[i,j]==-1){xuneffect[i]=l}#end if

    else {

```

```

    #if an element of attr is 1, then the level equals to the index of the column
    if(attr[i,j]==1){
      xuneffect[i]=j}#end if

  }#end else

  }#end of looping through each column within row i

}#end of looping through each row of matrix attr

#the function returns a nr by 1 column vector which contains levels of the input
attribute
return(matrix(xuneffect))

}#end of uneffect function

```

getdiff	This function transforms the attribute matrix into a differenced system (base category= <i>p</i>)
----------------	--

```

getdiff=function(p,nset,nlgt,nbeta,x){
#####
#Purpose:
#
#A function which convert the x matrix into the differenced system
#alternative p is used as the base category
#Author:
#
# Hanshui Zhang (Last Modified on July 31, 2013)
#
#Arguments:
# p: num of alternatives
# nset: num of choice sets
# nlgt: num of respodents
# nbeta: num of model parameters to be estimated
# x: p by nbeta dim attribute matrix

#####

#store the original x matrix in dataframe
xset=NULL
#xdiff is the dataframe contains the differenced values
xdiff=NULL
#xdf is the matrix form of xdiff
#dim(xdf)=(nset*nlgt) by (p-1)
xdf=NULL

```

```

totalset=nlgt*nset
#looping through each choice set
for (i in 1:totalset)
{

  #transform x into dataframe

  xbeg=p*(i-1)+1
  xend=p*i
  xset[[i]]=x[xbeg:xend,]

  #transform xmat into matrix
  xtemp=NULL
  xtemp=t(t(xset[[i]]))

  #within each choice set, take difference
  xdiffi=NULL
  for(nr in 1:(p-1)){

    xdiffi=rbind(xdiffi,xtemp[nr,]-xtemp[p,])
  }#end for (p-1)
  xdiff[[i]]=xdiffi
  xdf=rbind(xdf,xdiff[[i]])

}#end of for totalset

return(xdf)

}#end getdiff function

```

rmxp	This function performs the <i>Mixed Probit Model</i> Estimation
-------------	---

```

rmxp =
function(Data,Prior,Mcmc){
# revision history:
#   created 12/04 by Allenby and McCulloch
#
#   Modified July 21, 2013 by Hanshui zhang
# the rmxp function is a modification of the rScreen function
#
# purpose: estimate Mixed Probit Model
#
# Arguments:
#   Data contains a list of (X, y,nlgt, nset, p,nbeta)
#     X : design matrix
#       dim(X) = (nbeta,p,nset,nlgt)
#       nbeta= number of attribute level variables (discrete attributes are effect

```

```

coded)
#      p= number of choice alternatives
#      nset= number of choice tasks
#      nlgt= number of respondents
#      y : matrix of indicators for responses
#      dim(y) = (p,nset,nlgt)
#      y_i,j,k = 1 if alternative i is chosen in choice task j for respondent k
#      Prior is a list of (nu,V0,betabar,Abeta)
#      beta ~ N(mu,Vbeta)
#      mu ~ N(betabar,Abeta^-1_)
#      Vbeta ~ IW(nu,V0)
#      Mcmc is a list of (R,keep)
#
# Output:
# betabardraw: R/keep by nbeta
# Vbetadraw: R/keep by nbeta**2
# betadraw: nlgt x nbeta x R/keep

X=Data$X
y=Data$y

nlgt=Data$nlgt
nset=Data$nset
p=Data$p
nbeta=Data$nbeta

V0=Prior$V0
nu=Prior$nu
betabarbar=Prior$betabarbar
Abeta=Prior$Abeta

R=Mcmc$R
keep=Mcmc$keep

#
# -----
#
# define functions needed
#
cdrawz = function(V,Y,C,Z){
dd = dim(V)
array(.C('drawz',as.integer(dd[1]),as.integer(dd[2]),as.double(V),as.integer(Y),as.i
nteger(C),z=as.double(Z))$z,dim=dd)

```

```

}

#Initialize vector and matrix for mean and covariance for beta's
# e.g. the distribution of heterogeneity

bmean = array(0, dim=c(nbeta))
V = as.matrix(diag(1,nbeta))

z = array(0, dim=c(p,nset,nlgt))
Vbetadraw = matrix(double(floor(R/keep)*nbeta*nbeta),ncol=nbeta*nbeta)
betabardraw = array(0, dim=c(R/keep,nbeta))
oldbetadraw = matrix(double(nlgt*nbeta),ncol=nbeta)
betadraw = array(0,dim=c(nlgt,nbeta,floor(R/keep)))

bbar = matrix(0, ncol=1,nrow=nbeta)
Bdbar = matrix(0, nbeta,1)
Vh = matrix(0,nbeta,nbeta)
Vi = as.matrix(diag(1,nbeta))
sigma = 1
iota=matrix(rep(1,nlgt),ncol=1)

#Initialize choice sets
cset = array(1, dim=c(p,nset,nlgt))

itime=proc.time()[3]
cat("MCMC Iteration (est time to end - min)",fill=TRUE)
flush.console()

for(o in 1:R){

#    Draw B-h|B-bar, V
#    Prior is distribution of heterogeneity

    for(i in 1:nlgt){

#        z|Beta, sigma, y, cset

        XX=matrix(X[,,,i],nrow=nbeta,ncol=p*nset)
        VV=matrix(t(oldbetadraw[i,])%*%XX,nrow=p,ncol=nset)

        z[,i]=cdrawz(VV,y[,i],cset[,i],z[,i])

#        Beta|z, sigma
#        have to stack up the obs to do matrix manipulations

```



```

Xd=t(XX)
w=matrix(z[,i],nrow=nset*p,ncol=1)

XTw = t(Xd) %*% w
Vh = chol2inv(chol(crossprod(Xd,Xd) + Vi))
root=chol(Vh)
Bdbar = Vh %*% ((XTw) + Vi%*%bbar)
oldbetadraw[i,] = Bdbar + t(root)%*%rnorm(length(Bdbar))
}

# Draw B-bar|B-h,V
# betabar ~ N(betabarbar,Abeta-1)
vv=chol2inv(chol(nlgt*Vi+Abeta))
mu=vv%*%(nlgt*Vi%*%(t(oldbetadraw)%*%iota/nlgt)+Abeta%*%betabarbar)
ar)
bbar=mu + t(chol(vv))%*%rnorm(nbeta)

# Draw V|B-h,B-bar
# Prior is IW(nu,V0)
S=crossprod(oldbetadraw-t(array(bbar,dim=c(nbeta,nlgt))))
W=rwishart(nlgt+nu,chol2inv(chol(V0+S)))
V=W$IW
Vi=W$W

# update screen
if(o%%100==0)
{
ctime=proc.time()[3]
timetoend=((ctime-itime)/o)*(R-o)
cat(" ",o," (",round(timetoend/60,1),")",fill=TRUE)
flush.console()
}
mkeep=o/keep
if(mkeep*keep == (floor(mkeep)*keep))
{betabardraw[mkeep,]=bbar
Vbetadraw[mkeep,]=as.vector(V)
betadraw[,mkeep]=oldbetadraw

}

}
ctime=proc.time()[3]
cat(" Total Time Elapsed: ",round((ctime-itime)/60,2),fill=TRUE)

list(betabardraw=betabardraw, Vbetadraw=Vbetadraw, betadraw=betadraw)
}

```

References

- Albert, J.H. and Chib, S. (1993). Bayesian Analysis of Binary and Polychotomous Response Data. *Journal of the American Statistical Association*, 88(422): 669-679.
- Allenby, G., & Rossi, P. (2006). Hierarchical Bayes models: A practitioner's guide. In R. Grover & M. Vriens (Eds.), *The Handbook of Marketing Research*. Thousand Oaks, CA: Sage.
- Ben-Akiva, M. and Bruno Boccara. (1995). Discrete Choice Models with Latent Choice Sets. *International Journal of Research in Marketing*, 12(1): 9-24.
- Bierlaire, M. (1997). Discrete choice models. *NATO Advanced Studies Institute on Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*, Hungary.
- Bunch, D. S. and R. Kitamura. (1989). Multinomial Probit Model Estimation Revisited: Testing of New Algorithms and Evaluation of Alternative Model Specifications for Trinomial Models of Household Car Ownership. Transportation Research Group Report UCD-TRG-RR-4, University of California.
- Davis Bunch, D. S. (1991). Estimability in the Multinomial Probit Model. *Transportation Research*. 25(1):1–12.
- Burda, M., Harding, M., and Hausman, J. (2008). A Bayesian Mixed Logit-Probit Model for Multinomial Choice. *Journal of Econometrics*, 147(2): 232-246.
- Burgette, L.F. and Hahn, P. R. (2013). A Symmetric Prior for Multinomial Probit Models. University of Chicago Booth School of Business.
- Carlin, B. P., and Louis, T. A. (2000). *Bayes and Empirical Bayes Methods for Data Analysis*, London: Chapman & Hall.
- Chib, S., Greenberg, E. and Chen, Y. (1998). MCMC Methods for Fitting and Comparing Multinomial Response Models, Economics Working Paper Archive, Washington, University of St. Louis.
- Cramer, J.S. (2003). *Logit Models from Economics and Other Fields*, Cambridge: Cambridge University Press.
- Danganzo, C. (1979). *Multinomial Probit: The Theory and its Applications to Demand Forecasting*, New York: Academic Press.

- Fox, J.P. (2010). *Bayesian Item Response Modeling: Theory and Applications*, New York: Springer.
- Glasgow, G. (2001). Mixed Logit Models in Political Science. *Prepared for presentation at the Eighteenth Annual Political Methodology Summer Conference*, Emory University.
- Glickman, Mark E. and Van Dyk, David A. (2007). Basic Bayesian Methods. *Topics in Biostatistics (Methods in Molecular Biology)*, 404: 319-338.
- Gilbride, T. J. and Allenby, G.M. (2004). A Choice Model with Conjunctive, Disjunctive, and Compensatory Screening Rules. *Marketing Science*, 23(3): 391-406.
- Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. (1995). *Markov Chain Monte Carlo in Practice*, London: Chapman & Hall.
- Hensher, D.A. and Greene, W.H. (2003). The Mixed Logit Model: The State of Practice. *Transportation*, 30(20): 133-176.
- Hobert, J. P. (2011). The Data Augmentation Algorithm: Theory and Methodology, *Handbook of Markov Chain Monte Carlo*, London: Chapman & Hall.
- Imai, K. and van Dyk, D. A. (2004). A Bayesian Analysis of the Multinomial Probit Model Using Marginal Data Augmentation. *Journal of Econometrics*, 124(2): 311-334.
- McFadden, D. (1973). Conditional Logit Analysis of Qualitative Choice Behavior. *Frontiers in Econometrics* (Edited by P. Zarembka), 105-42.
- McFadden, D. and Train, K. E. (2000). Mixed MNL Models of Discrete Response. *Journal of Applied Econometrics*, 15(5): 447-470.
- McFadden, D. (2001). Disaggregate Behavioural Travel Demand's RUM Side – A 30 Years Retrospective. *Travel Behaviors Research: The Leading Edge*, Oxford: Pergamon Press, 17-64.
- McCulloch, R.E., Polson, N. G. and Rossi, P. E. (2000). A Bayesian Analysis of the Multinomial Probit Model with Fully Identified Parameters. *Journal of Econometrics*, 99(1): 173-193.
- Monfardini, C. and Santos Silva, J.M.C. (2008). What Can We Learn About Correlations from Multinomial Probit Estimates. *Economics Bulletin*, 3(28): 1-9.
- Nobile, A. (1998). A Hybrid Markov Chain for the Bayesian Analysis of the Multinomial Probit Model. *Statistics and Computing*, 8(3): 229-242.

- Rao, J.N.K. (2003). *Small Area Estimation*, New York: Wiley.
- Robert, C. and Casella, G. (2004). *Monte Carlo Statistical Methods*, New York: Springer-Verlag.
- Robert, C. and Casella, G. (2009). *Introducing Monte Carlo Methods with R, Use R*, New York: Springer-Verlag.
- Rossi, P. E., Allenby, G. M. and McCulloch, R. E. (2006). *Bayesian Statistics and Marketing*, England: John Wiley & Sons Ltd.
- Rossi, P. E., McCulloch, R. E. and Allenby, G. M. (1996). The Value of Purchase History Data in Target Marketing. *Marketing Science*, 15(4): 321-340.
- Scott, S.L. (2011). Data Augmentation for the Bayesian Analysis of Multinomial Logit Models. *Statistical Papers*, 52(1): 87-109.
- Tanner, M. A. and Wong, W. H. (1987). The Calculation of Posterior Distributions by Data Augmentation (with discussion). *Journal of the American Statistical Association*, 82(398): 528-550.
- Train, K. E. (2009). *Discrete Choice Models with Simulation*, New York: Cambridge University Press.
- Train, K. E. and Sonnier, G. (2003). Mixed Logit with Bounded Distributions of Partworths. Working paper, University of California, Department of Economics.
- van Dyk, D. A. and Meng, X.-L. (2001). The Art of Data Augmentation. *Journal of Computational and Graphical Statistics*, 10(1): 1-111.
- van Dyk, D. A. and Meng, X.-L. (2000). Algorithms based on data augmentation. In *Computing Science and Statistics: Proceedings of the 31st Symposium on the Interface* (Edited by M. Pourahmadi and K. Berk), 230-239. Interface Foundation of North America, Fairfax Station, VA.
- Yu, J., Goos, P. and Vandebroek, M. (2011). Individually Adapted Sequential Bayesian Conjoint-Choice Designs in the Presence of Consumer Heterogeneity. *International Journal of Research in Marketing*, 28(4): 378-388.

AFDELING
Kasteelpark Arenberg 11 bus 2100
3001 LEUVEN, BELGIË
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
@kuleuven.be
www.kuleuven.be

