Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science

Evaluating students' study habits using Bayesian Multinomial Logistic Regression

Amna Areej Ahmad

August 24, 2023

Evaluating students' study habits using Bayesian Multinomial Logistic Regression

by

Amna Areej Ahmad

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 24, 2023 at 11:00 AM.

Student number:	4870441	
Project duration:	May, 2023 – August, 2023	
Thesis committee:	Dr. J. G. Spandaw,	TU Delft, supervisor
	Prof. dr. ir. G. Jongbloed,	TU Delft
	Dr. T.W.C. Vroegrijk,	TU Delft, PRIME

Abstract

The Bayesian approach is a very important approach for tackling problems in statistics. It involves choosing a distribution that reflects the prior knowledge and thus takes all knowledge into account in contrast to the frequentist approach. It also assumes that the parameters (the regression coefficients) follow a distribution called the posterior distribution instead of fixed constants. When a specific choice of this prior is made, this needs to be justified as the prior directly influences the posterior distribution of the regression coefficients. It is also possible to consider priors that do not carry a lot of information and such priors will be compared in this project.
In this thesis, the Bayesian approach will be used to apply a multinomial logistic regression model to data concerning students' study habits and beliefs. The data is provided by a research group called PRIME and they focus on mathematics education at the TU Delft. Multinomial logistic regression is used to find predictions of the choices expressed in probabilities. Bayesian statistics is not only useful in a sense that it offers the possibility to specify the prior knowledge, but also because the Bayesian way of thinking can be incorporated in evaluating results. This can be done by constructing credible intervals for the predicted probabilities. Overlap between intervals can then give insight on prediction quality.
In this project, the models are coded in R and here two packages are used: the `UPG` and the `BRMS` package. The priors that are compared are the Gaussian and Cauchy distributions. Other than that there are also default priors used in the packages, which can be compared to the Gaussian and Cauchy priors. In the end, a conclusion can be drawn about the performance of each model based on the prediction accuracy. It can be concluded that the `BRMS` package outperforms the `UPG` package in terms of accuracy both using default priors and overall using default priors gives more accurate results than specifying the prior. However, the difference in the accuracy of the model using the `BRMS` package is not significantly higher than the accuracy obtained from the `UPG` model and the running time is a lot higher for the `BRMS` package. From the models with a specified prior, the model with the Cauchy distribution as prior performed better.

Contents

Chapter 1

Introduction

In statistics, there are two approaches of tackling a problem, the frequentist and the Bayesian approach. The frequentist approach is sometimes also called the classical approach and is often the approach that is explained when being introduced to statistics. Frequentist statistics is known for being the more objective choice, because even if there is prior knowledge available, this is not taken into account. This is one of the advantages of Bayesian statistics as this approach considers this prior knowledge as well as this objective data that is given (Bolstad and Curran 2016).

The parameters are assumed to be fixed unknown constants when applying the frequentist approach. The Bayesian approach, however, assumes that the parameters follow a certain probability distribution. Using the Bayesian approach a predictive distribution is obtained, which cannot be done using the frequentist approach.

Despite the large number of advantages, due to the computational complexity, the Bayesian approach was not often used. The formula for the predictive distribution requires some integration, which was not always easily done numerically. Here, adding more parameters to the model could increase the complexity severely.

Nowadays, this is a lot easier as computers are faster and there are methods for taking random samples from the desired predictive distribution, such as the Gibbs sampler. By taking a large number of samples an approximation of the predictive distribution is found. More descriptive statistics such as the mean and standard deviation result from applying the Markov Chain Monte Carlo Method (MCMC) (Bolstad and Curran 2016).

With this emergence of algorithms to make the Bayesian approach a more realistic option, it follows that the Bayesian approach overall often performs better than the frequentist method, even when comparing the models in terms of criteria used for frequentist methods to judge performance. This fueled the rising interest for the Bayesian approach and also makes this approach useful today. Hence, in this research the focus will be on applying and incorporating the Bayesian way of thinking as good as possible (Bolstad and Curran 2016).

As mentioned before, the Bayesian approach includes prior knowledge together with the already given data, which is considered to be objective. The Bayesian approach relies solely on Bayes' theorem. This theorem combines these two components and yields the predictive distribution, which is called the posterior distribution. The prior knowledge is contained in what is called a prior distribution. This is thus also considered to be a random variable and can be chosen. This choice of the prior is fundamental for the resulting posterior distribution and is hence an important decision that needs to be taken.

It is often not the case that the prior distribution reflects all of the prior knowledge. Quite often it occurs that when choosing the prior distribution not a lot of information is available and the choice is made unconsciously without including any information. Such a prior that does not consider any prior knowledge is referred to as an uninformative prior. As there are many choices for the prior, when specifying an uninformative prior distribution with certain mean and variance, this choice of prior needs to be justified (Robert 2007).

Considering different priors can thus be interesting as the posterior distribution is directly affected by this decision. Hence, in this project different priors will be used to illustrate differences in accuracy and to find which of the uninformative or weakly-informative priors perform best.

1.1 PRIME

In this project, the Bayesian approach will be applied to a data set obtained from PRIME. PRIME is a research group that focuses on mathematics education at the TU Delft. The research team at PRIME works together in the section Statistics of the Delft Institute of Applied Mathematics (DIAM) to increase motivation for students and improve mathematics education. PRIME also aims to incorporate mathematics more into engineering. The research at PRIME is thus split into the following two lines:

1. Increasing students motivation.
2. Improve students performance and incorporating mathematics into engineering.

This research belongs to the second line as the results from this project could be interpreted and applied to further improve mathematics courses at the TU Delft, while making sure the context is interesting to the student. There are already several things that are done to keep the students engaged such as using videos that explain the mathematical concepts to actual problems and incorporating more applied examples.

The data used for this project contains survey results that were filled in by students from different programs at the TU Delft. Some of the questions were aimed at evaluating students habits and beliefs about the mathematics courses they were following at that time. These questions were mostly multiple choice and contain the information needed to conduct this research. Some other questions were also asked about the students previous results and about their expectations. These questions were used to predict the probabilities of the outcomes of the questions about the students' study study habits and beliefs.

To predict these probabilities multinomial logistic regression is applied. Logistic regression is a transformation of linear regression such that the outcome is a value between 0 and 1 and thus represents a probability for some variable. Logistic regression works in the case there are 2 possible outcomes for the question that needs to be answered. However, in this case the questions involving the students' study habits and beliefs all have more than 2 options. Hence, the extension of logistic regression called multinomial logistic regression is used. To fully understand this concept, the first research question is '*What is multinomial logistic regression?*'.

To finally evaluate the students' study habits and beliefs these two mathematical concepts are combined together and thus a question that arises is: '*How can we apply the Bayesian approach to multinomial logistic regression?*'. In (Schriemer 2023), the same data was already used to discuss the differences between the two approaches, frequentist and Bayesian, when applying multinomial logistic regression. To delve deeper into Bayesian statistics, this research mainly focuses on this. Now when applying the Bayesian approach to multinomial logistic regression, an expression for the model needs to be constructed, which is another research question. As mentioned before, the Bayesian way of thinking can be used to evaluate the results. The question that then arises is how this Bayesian way of thinking can be used to interpret results. This Bayesian approach, however, depends on the choice of the prior. Hence, another question that is answered in this project is '*How does choosing a different prior influence the results for evaluating students' study habits?*'.

1.2 Overview of this project

This project covers different subjects, so here the structure of the project is briefly summarized. In chapter 2, the background information is given that is needed to understand this thesis. First, logistic regression will be discussed, which is further extended to multinomial logistic regression. This gives a basic understanding and is fundamental in understanding the model. Since this thesis involves the Bayesian approach, the basics of Bayesian statistics are discussed. Within Bayesian statistics, in this thesis specifically the prior is chosen differently. So, in chapter 2, the choice of the prior is also discussed and the choices made in this thesis are discussed.
Chapter 3 discusses the data provided by PRIME. This chapter also involves preprocessing, where the data is cleaned so it is ready to use. Finally, most of the assumptions of the model were already checked in (Schriemer 2023). However, to give some additions, the correlations between the predictors are checked. This can be found in section 3.3.
Now applying this data, formulating the model for the predictors and outcome variables as found in chapter 3 is done in chapter 4. To evaluate results, there are two packages that are used in this project: `UPG` and `BRMS`. These are packages in R and use underlying algorithms, which are briefly discussed in chapter 4.
Finally, chapter 5 lists all results that follow from applying the model. The first model that is discussed is

the model applied to uniformly generated data. Then the `UPG` package is used to find results when applying this to the whole data set from PRIME. Now, this data set can be split up using the Train-Test split method to evaluate results. This is a way of comparing the fit and accuracy to assess validity of the model. Another method that is applied for this is K-fold cross-validation. Here, the same default prior is considered. Now, using the `BRMS` package also the default prior is first considered and then validated using K-fold cross-validation and Leave-One-Out cross-validation. In the `BRMS` package it is possible to specify the prior. Hence, finally the Normal distribution with mean 0 and variance 4 and the Cauchy distribution with center 0 and scale 2.5 are considered as prior distributions.

Chapter 6 discusses the main conclusions that can be drawn from the results and summarizes the results shortly. Other than that future research and limitations of this project are discussed. The code used for this thesis can be found in the appendix as well as some additional plots as mentioned in the results.

Chapter 2

Background information

In this chapter the main background information that is needed to understand this project will be discussed. First, logistic regression and its corresponding model is discussed. This is then extended to multinomial logistic regression. The model used in this project applies the Bayesian approach, so in this chapter Bayesian statistics will be discussed. As there are some choices made for the models used in this project, these will also be discussed in this chapter. Here, the choice of the prior will be discussed for each model.

2.1 Logistic Regression

Logistic regression is used in practice for the case in which the outcome variable is binary (i.e. either 1 or 0) and aims to predict the probability of the outcome of the binary variable Y_i. Using logistic regression the probability that the outcome variable $Y_i = 1$ can be evaluated. This probability is then denoted by π_i. Now, the probability that the outcome variable equals 0 is exactly $1-\pi_i$. As π_i is a probability, it must always hold that $\pi_i \in (0, 1)$.

As a short recap, consider a simple linear regression line $Y_i = \alpha + \beta X_i + \varepsilon_i$. Here, it is assumed that $\varepsilon_i \sim N(0, \sigma^2)$. In this regression line $i = 0, \dots, n$ is the row in the data set that is considered and there is only one predictor variable X_i involved in determining the outcome. As mentioned before, the outcome variable Y_i is in this case binary and can thus only attain values 0 and 1. This implies that $Y_i|X_i \sim Ber(\pi_i)$ for $i = 1, \dots, n$. This notation indicates that given X_i, Y_i follows a Bernoulli distribution, where the probability of a success is equal to π_i.

Now to determine probability π_i, a transformation needs to be applied to this simple linear regression line. Such a transformation function is called a link function. This transformation must make sure that π_i stays in the defined interval. A constraint for this function then is that the link function must be positive monotone. This means that the function values should always be larger than 0 and the function should be increasing. To ensure that this is the case, a cumulative probability distribution function can be used as the link function (Fox 2015). This link function is denoted by $\Lambda(\cdot)$ such that $\pi_i = \Lambda(\alpha+\beta X_i)$. This implies that now $Y_i|X_i \sim Ber(\Lambda(\alpha+\beta X_i)$.

A cumulative probability distribution can be used as it satisfies all requirements. For the transformation to be well-defined, it must be continuously differentiable and symmetric. Similarly, it must tend to $\pi_i = 0$ and $\pi_i = 1$ slowly as well as it should be strictly increasing. For the inverse transformation to be well defined the transformation $\Lambda(\cdot)$ must be one-to-one. This is that each value for Y_i is mapped to exactly one value of π_i by the link function. In this way, it follows that $\Lambda^{-1}(\pi_i) = \alpha + \beta X_i$ (Fox 2015).

Possible choices for the link function are the logit or the probit transformations, which both satisfy all the requirements as specified above. The logit link function is defined as in equation 2.1 and the probit link function is defined in equation 2.2. Here, the logit model considers the CDF of the logistic distribution. However, the probit model considers the CDF of the standard normal distribution instead, which distinguishes the two models from each other (Fox 2015).

$$\pi_i = \Lambda(\alpha + \beta X_i) = \frac{1}{1 + e^{-(\alpha+\beta X_i)}} \tag{2.1}$$

$$\pi_i = \phi(\alpha + \beta X_i) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha+\beta X_i} e^{-\frac{1}{2}Z^2} dZ \tag{2.2}$$

As these two models are very similar, it is important to highlight the key differences between the models to be able to choose which model to use. According to (Fox 2015), the most important difference between logit and probit is that considering the inverse transformations for the logit model gives us log odds which can be interpreted directly. However the inverse of the probit model can not be interpreted directly. Another advantage of the logit model is that it is less complex to evaluate as the probit model uses integral, which then needs to be evaluated. This would add to the complexity of the model. Hence, for simplicity and convenience the logit model will be used for this research.

The inverse of the logit model $\Lambda^{-1}(\pi_i)$ gives the log odds. Here $\Lambda^{-1}(\pi_i) = \log_e(\frac{\pi_i}{1-\pi_i})$ is called the logit of π_i. However, it was indicated before that the link function should be one-to-one. Hence, it also follows that $\Lambda^{-1}(\pi_i) = \alpha + \beta X_i$. Equating both sides then gives the following expression for the odds, where the odds are denoted by $\frac{\pi_i}{1-\pi_i}$:

$$\log_e\left(\frac{\pi_i}{1-\pi_i}\right) = \alpha + \beta X_i \tag{2.3}$$

$$\frac{\pi_i}{1-\pi_i} = e^{\alpha+\beta X_i} \tag{2.4}$$

This value $\frac{\pi_i}{1-\pi_i}$ represents the probability of $Y_i = 1$ divided by the probability that $Y_i = 0$. When the probability of a success is higher that the probability of failure, then the odds ratio is larger than 1. The odds are at least 0 as each probability $\pi_i > 0$ and do not have an upper bound. It can then be deduced that the logarithm of these odds increases as these odds increase since the logarithm is a monotone increasing function. Example 2.1 now gives an applied example of logistic regression. Since there are probabilities involved, this example considers a Bayesian problem, which will be explained in section 2.3.

Example 2.1 (Weather prediction) Consider the binary variable Y_i that indicates whether it is going to rain on day i or not, where

$$Y_i = \begin{cases} 1 & \text{Rain on day } i \\ 0 & \text{No rain on day } i \end{cases}$$

Say that this binary variable Y_i depends on some variable X_i, which is the amount of rain in mm on the day before. The data that is available to predict the probabilities contains values for Y_i and X_i. As explained before π_i indicates the probability that $Y_i = 1$. Therefore, the probability that it will rain on day i is π_i. The probability that it will not rain on day i is then equal to $1 - \pi_i$. The value for π_i can now be found using the formula in equation 2.1, where a training set is used to estimate α and β. Here $\pi_i = \Lambda(\alpha + \beta X_i)$. These estimations for α and β are then called $\hat{\alpha}$ and $\hat{\beta}$. Hence, it follows that

$$\hat{\pi_i} = \frac{1}{1 + e^{-(\hat{\alpha}+\hat{\beta}X_i)}}$$

When $\hat{\alpha}$ and $\hat{\beta}$ are estimated, given the X_i, $\hat{\pi_i}$ can be calculated.

This can also be elaborated to models with more than one explanatory variables $x_{i1}, \dots, x_{ik}$ that are involved in predicting the outcome. The outcome is in this case still a binary variable y_i. Then, the multiple regression line is defined as $y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_k x_{ik} + \varepsilon_i$. Here, the k explanatory variables are used to determine the outcome variable y_i. Again, applying the logit transformation as the link function yields equation 2.5.

$$\pi_i = \Lambda(\beta_0 + \beta_1 x_{i1} + \dots + \beta_k x_{ik}) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_{i1}+\dots+\beta_k x_{ik})}} \tag{2.5}$$

Now, an expression for the odds can be deduced using equation 2.5. From equation 2.5, it follows that

$$\begin{aligned} 1 - \pi_i &= 1 - \frac{1}{1 + e^{-(\beta_0+\beta_1 x_{i1}+\dots+\beta_k x_{ik})}} \\ &= \frac{1 + e^{-(\beta_0+\beta_1 x_{i1}+\dots+\beta_k x_{ik})} - 1}{1 + e^{-(\beta_0+\beta_1 x_{i1}+\dots+\beta_k x_{ik})}} \\ &= \frac{e^{-(\beta_0+\beta_1 x_{i1}+\dots+\beta_k x_{ik})}}{1 + e^{-(\beta_0+\beta_1 x_{i1}+\dots+\beta_k x_{ik})}} \end{aligned}$$

Dividing the expression for π_i by the expression for $1-\pi_i$ then yields the following

$$\frac{\pi_i}{1-\pi_i} = \pi_i \cdot \frac{1}{1-\pi_i} \tag{2.6}$$

$$= \frac{1}{1+e^{-(\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik})}} \cdot \frac{1+e^{-(\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik})}}{e^{-(\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik})}} \tag{2.7}$$

$$= \frac{1}{e^{-(\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik})}} \tag{2.8}$$

$$= e^{\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik}} \tag{2.9}$$

Hence, the expression that follows for the odds is as follows

$$\frac{\pi_i}{1-\pi_i} = e^{\beta_0+\beta_1 x_{i1}+\cdots+\beta_n x_{ik}}$$

The interpretation of the odds is still the same. The odds are the probability of a success divided by the probability of a failure. It is clear from this notation that when the value of say for example x_{i1} increases with exactly 1, then the odds are multiplied by e^{β_1}.

Example 2.2 (Weather prediction with 3 explanatory variables) Consider the same binary variable y_i from example 2.1. Say that this binary variable is now not just dependent on the rain on the day before, but also on 2 other factors, such that y_i depends on the following variables:

x_{i1} Rain on the day before in mm

x_{i2} Average amount of rain in the previous week in mm

x_{i3} Humidity percentage on day i

With these variables one can construct $\pi_i = \Lambda(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3})$, which is the probability that it will rain on day i. Using equation 2.5, the expression for π_i becomes

$$\pi_i = \frac{1}{1+e^{-(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3})}}$$

Plugging in the fitted values $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3$ that again follow from using a training set, gives the following expression for the prediction

$$\hat{\pi_i} = \frac{1}{1+e^{-(\hat{\beta}_0+\hat{\beta}_1 x_{i1}+\hat{\beta}_2 x_{i2}+\hat{\beta}_3 x_{i3})}}$$

2.2 Multinomial Logistic Regression

Logistic regression is the basic case in which the distribution of a binary variable is described in terms of covariates using a linear predictor. However, in reality there are many cases where the outcome variable is a categorical variable and thus has more than two possible outcomes. Therefore, an extension of logistic regression is useful in the case that the target variable is categorical. This is exactly the purpose of multinomial logistic regression. This results in the same model, when the logit model is used, as in equation 2.5. The only difference between logistic regression and multinomial logistic regression is that the response variable y_i can attain more than two values. This results in the following model, where π_{ij} is the resulting probability that a certain category j was chosen such that $j \in \{1, \ldots, m-1\}$ and m is the number of categories. Here, the categories are labelled with numbers from $1, \ldots, m$ (Fox 2015).

$$\pi_{ij} = \frac{e^{\beta_{0j}+\cdots+\beta_{kj} x_{ik}}}{1+\sum_{l=1}^{m-1} e^{\beta_{0l}+\cdots+\beta_{kl} x_{ik}}} \quad \text{for } 1 \leq j \leq m-1 \tag{2.10}$$

$$\pi_{im} = 1 - \sum_{j=1}^{m-1} \pi_{ij} \quad \text{for } j = m \tag{2.11}$$

Equation 2.10 shows the expression for the model that will be used in this research to determine the probabilities for the category j. Here, equation 2.10 gives the probabilities for $1 \leq j \leq m-1$. The probability of the last category, category m, is defined as the sum of all probabilities substracted from 1. This is given in equation 2.11. Considering the expression for $1 \leq j \leq m-1$, the outcome π_{ij} is a probability as the numerator is always less than the value in the denominator. Hence, it holds that $\pi_{ij} \in (0,1)$. For $j = m$, it holds that for all i, $\pi_{ij} \geq 0$ for every $j \in 1,..,m-1$. Since π_{ij} gives the probability that a category is chosen, it also holds that $\sum_{j=1}^{m-1} \pi_{ij} < 1$. Therefore, it follows that $\pi_{im} \in (0,1)$.

Example 2.3 (Weather prediction of a categorical variable with 3 explanatory variables) Example 2.2 can be further extended to the case where y_i is a categorical variable with more than 2 categories. Say y_i has 3 categories and is defined in the following way

$$y_i = \begin{cases} 1 & \text{Rain on day } i \\ 2 & \text{Cloudy on day } i \\ 3 & \text{No rain and not cloudy on day } i \end{cases}$$

The purpose is now to determine the probability that it was cloudy, it was raining or neither of them on day i. The probability π_{ij} is in this case defined as the probability of weather of category j on day i. For example π_{43} describes the probability that there was no rain and it was not cloudy on day 4. For $y_i = 1$ and $y_i = 2$ (so $1 \leq j \leq 2$), π_{ij} is given by

$$\pi_{ij} = \frac{e^{\beta_{0j}+\beta_{1j}x_{i1}+\beta_{2j}x_{i2}+\beta_{3j}x_{i3}}}{1+\sum_{l=1}^{2} e^{\beta_{0l}+\beta_{1l}x_{i1}+\beta_{2l}x_{i2}+\beta_{3l}x_{i3}}}$$

The probability of the last category $j = 3$, where there no rain and it is not cloudy on day i, is determined by the other two probabilities, i.e.

$$\pi_{i3} = 1 - (\pi_{i1} + \pi_{i2})$$

2.3 Bayesian statistics

There are two ways of tackling a problem in statistics, the frequentist approach and the Bayesian approach. The most used method for estimation is the frequentist approach as in this case the parameters are assumed to be fixed unknown values, which are then estimated e.g. using maximum likelihood estimation. The Bayesian approach assumes that these parameters follow a certain probability distribution depending on the data, which is called the posterior distribution. Here, the parameters are considered to be random variables. The two methods can be distinguished by their perception. The Bayesian approach assumes there is some information already available prior to the data gathering to find the posterior distribution and hence it can be said that this approach is found to be more subjective than the frequentist approach.

Example 2.4 (Bayesian vs Frequentist) Say a person lost some object and now needs to find this object. The object can make a sound. The possible options of where the object can be are room A, B or C. Deciding on which room to look in first is a problem that can be solved using either the Bayesian approach or the frequentist approach.
Say that the sound that the object makes comes from room C. Furthermore, it is known that the last location where the object was seen is in room B.
A frequentist would use just this information that the sound originated from room C to decide which room to check first using some model. Using the Bayesian approach, the last location of the object together with the fact that the sound came from room C will be taken into account and used in the model. Hence, here it is clear that the Bayesian approach takes prior knowledge into account in predicting where to go first, whereas the frequentist approach does not.

For this research, only the Bayesian approach will be considered as in a recent bachelor thesis (Schriemer 2023), the comparison between Bayesian and frequentist approach applied to multinomial logistic regression is discussed. According to (Bolstad and Curran 2016), Bayesian methods frequently outperform frequentist methods, which only makes it interesting to delve deeper into the Bayesian approach to find useful results.

The Bayesian approach is built upon Bayes' theorem from basic probability theory.

Theorem 2.1 (Bayes' theorem) Consider events A and B with $\mathbb{P}(A), \mathbb{P}(B) > 0$. Then

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B) \cdot \mathbb{P}(B)}{\mathbb{P}(A)}$$

Bayes' theorem follows from applying the definition of the conditional probability as defined in 2.1.

Definition 2.1 (Conditional probability) Consider the events A and B with $\mathbb{P}(A) > 0$. Then the conditional probability is defined as

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(B \cap A)}{\mathbb{P}(A)}$$

Using definition 2.1, the proof of Bayes' theorem as given in theorem 2.1 is given below.

Proof 2.1 (Bayes' theorem) Events A and B are given with $\mathbb{P}(A) > 0$ and $\mathbb{P}(B) > 0$. Now using definition 2.1 it follows that

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

This can be rewritten in the following way

$$\mathbb{P}(A \cap B) = \mathbb{P}(A|B) \cdot \mathbb{P}(B) \tag{*}$$

From definition 2.1 it also follows that

$$\begin{aligned}
\mathbb{P}(B|A) &= \frac{\mathbb{P}(B \cap A)}{\mathbb{P}(A)} \\
&= \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)} \\
&= \frac{\mathbb{P}(A|B) \cdot \mathbb{P}(B)}{\mathbb{P}(A)} && \text{(by *)}
\end{aligned}$$

$\square$

The Bayesian approach is built upon Bayes' theorem. Given a multinomial logistic regression model as defined in equation 2.10 and 2.11, using Bayes' theorem, a Bayesian expression can be found in terms of model parameters for the regression coefficients. The main idea of Bayesian statistics is that it is assumed that the regression coefficients $\beta_{1j}, \ldots, \beta_{kj}$ are not point estimates, but instead follow a probability distribution, which is called the posterior distribution. Here, there is some belief of the distribution of the β's, which is combined with some observed data y, which together forms the posterior distribution. The data that is known in the MLR model is the response variable y_i which depends on the predictors $x_{i1}, \ldots, x_{ik}$. The posterior distribution is thus denoted by $p_{\boldsymbol{\beta}|\mathbf{y}}$. Theorem 2.1 can now be applied again. Note that, in this conditional probability, the probability of a continuous variable is determined given on some discrete variable. So in proof 2.2, it is shown that Bayes' theorem also holds in case of a mixture of discrete and continuous variables.

Proof 2.2 (Bayes' theorem for continuous and discrete random variables) Consider the discrete random variable Y that can attain values in some set V and a continuous random variable β that is in set U. Then using definition 2.1 for the conditional probability, it follows that

$$\begin{aligned} p(\beta \in U | Y \in V) &= \frac{p(\beta \in U, Y \in V)}{p(\beta \in U)} \\ &= \frac{p(\beta \in U, Y \in V)}{\sum_{y \in V} p_{Y=y}} \end{aligned}$$

And similarly, the other way around it follows that

$$\begin{aligned} p_{Y \in V | \beta \in U} &= \frac{p_{\beta \in U, Y \in V}}{p_{Y \in V}} \\ &= \frac{p(\beta \in U, Y \in V)}{\int_U f_\beta(b) \mathrm{d}b} \end{aligned}$$

This can be rewritten as

$$p(\beta \in U, Y \in V) = p_{Y \in V | \beta \in U} \int_U f_\beta(b) \mathrm{d}b$$

This can be used in the first conditional probability in the following way

$$p(\beta \in U | Y \in V) = \frac{p_{Y \in V | \beta \in U} \int_U f_\beta(b) \mathrm{d}b}{\sum_{y \in V} p_{Y=y}} \tag{2.12}$$

This proves the claim. □

In proof 2.2, an expression is given for Bayes' theorem applied to the posterior distribution since equation 2.12 is the same as

$$\int_U f_{\beta|Y}(b|y) \mathrm{d}b = \frac{p_{Y \in V | \beta \in U} \cdot \int_U f_\beta(b) \mathrm{d}b}{\sum_{y \in V} p_{Y=y}} \tag{2.13}$$

This corresponds exactly to the notation that was used for the posterior distribution before. In Bayesian statistics, equation 2.13 is denoted in a more compact way, namely

$$p_{\boldsymbol{\beta}|\mathbf{y}} = \frac{p_{\mathbf{y}|\boldsymbol{\beta}} \cdot p_{\boldsymbol{\beta}}}{p_{\mathbf{y}}} \tag{2.14}$$

Now $p_{\mathbf{y}|\boldsymbol{\beta}}$ denotes the probability of a discrete random variable given some continuous random variable. This is then referred to as the likelihood. This likelihood is used to update the prior beliefs about the distribution of the β's. This prior belief of the regression coefficients is denoted by $p_{\boldsymbol{\beta}}$ and this distribution that the β's are assumed to follow is called the prior distribution. By updating this prior distribution by the likelihood, it follows that $p_{\mathbf{y}|\boldsymbol{\beta}} \cdot p_{\boldsymbol{\beta}}$. Thus, another way of denoting the above formula for the posterior is:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

Here, the evidence is a known constant. Another way of denoting this is by leaving out the evidence and denoting it as Posterior $\propto$ Likelihood $\times$ Prior, where notation suggests that the posterior is proportionate to the likelihood times the prior (Bolstad and Curran 2016).

The probability $p_{\mathbf{y}}$ from equation 2.14 functions as a constant with the purpose of rescaling. This rescaling constant is often left out of this expression of the posterior (Etz 2018). In these expressions, there is some likelihood function involved. This is defined as in definition 2.2 (Bijma, Jonker, and Van Der Vaart 2017).

Definition 2.2 (Likelihood function) Let $X = (X_1, \dots, X_n)$ be a random vector with probability density $p_\beta(x_1, \dots, x_n)$ with parameter β. Fixing $x = (x_1, \dots, x_n)$, then the likelihood function is given by

$$\ell(\beta; x_1, \dots, x_n) := \prod_{i=1}^{n} p_\beta(x_i)$$

In equation 2.14, the likelihood is given by the expression $p_{\mathbf{Y}|\boldsymbol{\beta}}$. Applying definition 2.2 then yields $p_{\mathbf{Y}|\boldsymbol{\beta}} = \mathbb{P}(Y_1 = y_1, \dots, Y_n = y_n|\boldsymbol{\beta})$. Now since each of these Y_i's are different responses depending on student i, each of the Y_i's are independent of each other. Thus,

$$\mathbb{P}(Y_1 = y_1, \dots, Y_n = y_n|\boldsymbol{\beta}) = \prod_{i=1}^{n} \mathbb{P}(Y_i = y_i|\boldsymbol{\beta}) = \prod_{i=1}^{n} p_\beta(y_i) = \ell(\beta; y_1, \dots, y_n)$$

This corresponds to the likelihood function given in definition 2.2. Now, the likelihood is fixed information that follows from the given data and the prior distribution is chosen. The term $\mathbb{P}(\boldsymbol{\beta}$, the prior distribution, then indicates what the parameter values should look like. The posterior distribution then contains the information the parameter $\boldsymbol{\beta}$ should follow. This is called Bayesian updating from the prior to the posterior distribution (Zens, Frühwirth-Schnatter, and Wagner 2022).

In the frequentist setup, confidence intervals are introduced. Given a confidence interval, this can then be interpreted such that 95% of the constructed confidence intervals contain the true mean. There is also a Bayesian variant, which is called a credible interval. The interpretation of the credible intervals is more straightforward. Given a 95% credible interval, this is interpreted as the probability that the true mean is in this credible interval is 0.95 (Clyde 2022). Such credible intervals can then be constructed for the posterior distribution, but also for the calculated probabilities by using the 0.025 and 0.975-quantiles. Say a 95%-credible interval is considered. Then for an equal two-tailed distribution this credible interval lies between the 0.025th and 0.975th quantiles. Using these credible intervals for the probabilities it can be checked when there is overlap between credible intervals for different categories of the response variable. This adds to validation of the model.

Here, this could be in the context of the posterior distribution, but also in the context of the found probabilities. As defined in section 2.2, the resulting probabilities are denoted by π_{ij}. Here, j indicates the number of categories of the categorical variable y_i. For every category j, a credible interval can be created for π_{ij} for each i. This gives insight on prediction quality of the model for each category and based on this one can reason whether the prediction was reasonable or not. Constructing these intervals for the probabilities gives more context and interpretability to the results (Bolstad and Curran 2016).

2.4 Choice of the prior

As seen in the previous section, for the Bayesian approach to find the posterior distribution, a likelihood is needed and a prior distribution is needed. A prior serves as the prior beliefs on the model parameters. Given some data, this prior can be updated to a more realistic distribution. This is called Bayesian updating, where the prior is updated using the observed data. This is a way to interpret the formula for the posterior distribution with $p_{\boldsymbol{\beta}|\mathbf{y}} \propto p_{\mathbf{y}|\boldsymbol{\beta}} \cdot p_{\boldsymbol{\beta}}$. This means that this prior is important to be able to find some posterior. The choice of the prior is thus a very important part of the model construction in Bayesian statistics.

An interesting question that now arises is how that choice of a prior can be made, because this is subjective. There are two options when choosing a prior, either an informative or an uninformative prior is chosen.

An informative prior is a prior chosen given some knowledge about the possible result of the problem. Assuming that there is some strong belief about the outcome of $\mathbf{y}$, this is then incorporated in the prior distribution and can be updated using the likelihood resulting from the data. In that way, a posterior distribution follows based on an informative prior. Based on literature one can construct a informative prior, which is usually hard to decide as there is often not much information available about the context of the problem that helps in deciding on a prior. Now, there is also a weakly-informative prior, which is an informative prior that is proper. Such a weakly-informative prior contains less information than is known as prior knowledge, hence the name. The idea is for the prior to carry less information on purpose (Gelman 2006). Here, when using a weakly-informative prior some suitable scales are identified for the model (M. Betancourt 2017).

Assuming that a certain prior can be used for a model implies that it is believed that the parameters follow this certain distribution without including any knowledge from the context of the problem to decide on this prior distribution. Now, this prior is referred to as an uninformative prior as there is no prior knowledge while choosing the prior (Bolstad and Curran 2016).

In this research, the data obtained from PRIME is applied using the Bayesian approach, so the prior must be chosen for the model. As there is no informative prior available, this project will involve default priors from packages in R which are compared to some other uninformative priors, such as the Cauchy distribution and the Gaussian distribution.

2.4.1 Default prior of the UPG package

The prior that is used in (Schriemer 2023) is the default prior of the UPG package in R. Recall the multinomial logistic regression model where the predicted probability π_{ij} is defined as in equations 2.10 and 2.11. Bayes' rule is defined as $p_{\boldsymbol{\beta}|\mathbf{y}} \propto p_{\mathbf{y}|\boldsymbol{\beta}} \cdot p_{\boldsymbol{\beta}}$, where the prior is given by $p_{\boldsymbol{\beta}}$. In the `UPG` package, it is assumed that the regression coefficients follow independent Gaussian distributions with mean equal to 0 and a specified variance. The variance for this prior can be specified separately for the intercept and the other regression coefficients. This is done by defining parameters A_0 and B_0. Here, the value of A_0 defines the variance of the intercept and the value of B_0 is the variance of all other parameters. Consider regression coefficients for a category j such that $\boldsymbol{\beta}_j = (\beta_{0j}, \ldots, \beta_{kj})^T \in \mathbb{R}^{k+1}$. Here, β_0 corresponds to the intercept and β_d is any other regression coefficient with $d \in \{1, \ldots, k\}$. In this case the total number of regression coefficients is denoted by k. Then the prior used in the `UPG` package is defined by

$$\begin{aligned} &\beta_0 \sim N(0, A_0) \\ &\beta_d \sim N(0, B_0) \quad \text{for } d \in 1, \ldots, k \end{aligned}$$

This is the case for just one category but in general for any category $j \in 1, \ldots, m$ this is

$$\beta_j \sim N_k(0, A_j)$$

In this notation, A_j is defined as the prior covariance matrix. In the default case, the regression coefficients are assumed to have variance equal to 4 (Zens, Frühwirth-Schnatter, and Wagner 2022). This yields

$$\begin{aligned} &\beta_0 \sim N(0, 4) \\ &\beta_d \sim N(0, 4) \quad \text{for } d \in 1, \ldots, k \end{aligned}$$

2.4.2 Default prior of the BRMS package

The `BRMS` package is used to fit the same model. However, when for this package it is actually possible to choose the prior by specifying a string in `stan` language, which then translates it to a prior. Now it is also possible to not specify this parameter and then the function `brm()` sets back to a default. The specifications of the default prior are given by the function `prior_summary()`. The default prior distribution for all regression coefficients and for each category is assumed to be an improper flat prior. The intercept is assumed to follow a Student t-distribution with 3 degrees of freedom, center 0 and scale 2.5. The flat prior that was set on all the regression coefficients will according to the documentation not affect this. The intercept has its own class, which makes it easy to specify the prior for this class. The improper flat prior that is considered to be the prior distribution for each of the regression coefficients refers to an improper uniform density. This prior is an example of an uninformative prior as it uses no information about the model. The uniform distribution results in an improper posterior distribution. These improper flat priors are weakly informative. The improper uniform prior is considered to be $U(0, A)$ as $A \to \infty$ (Gelman 2006). The combination of these two priors is weakly informative for the specified parameters.

2.4.3 Cauchy prior

A prior that could be used besides the flat prior and the Student t-distribution is the Cauchy distribution. According to (M. Betancourt 2017), the Cauchy distribution can be a weakly informative prior depending on the chosen parameter. For this project we take the center 0 and scale 2.5 as this corresponds to the Student t-distribution, which was used as a prior as a default in the `BRMS` package.

2.4.4 Normal prior

In section 2.4.1, the prior that is used is considered to be a Gaussian distribution with mean 0 and variance 4. For the sake of comparison of the models, it could be interesting to see which package performs better. Hence, for this part also a $N(0, 4)$-distribution will be used. The documentation mentions that a $N(0, 10)$-distribution is a very weakly informative prior to use. Therefore, this choice is reasonable.

According to (M. Betancourt 2017), the Gaussian distribution is a weakly informative prior, when choosing good values for the shape parameter. Figure 2.1 shows the priors with the Gaussian, Cauchy and Student t-distribution. The figure shows that the Cauchy and Gaussian distribution are the more heavy-tailed distribution compared to the Student t-distribution. Figure 2.1 is found using the code in appendix D.

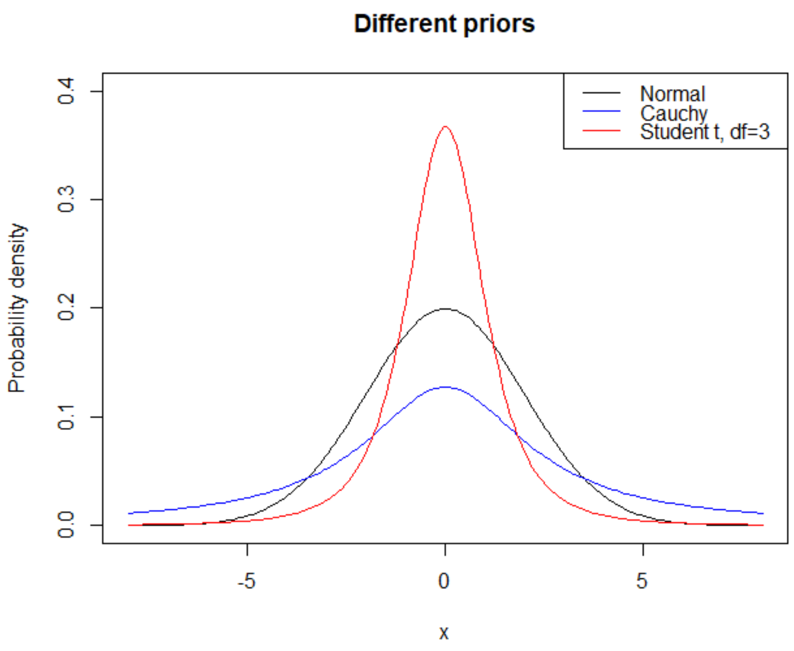


Figure 2.1: Different prior distributions used in the `BRMS` package.

Chapter 3

PRIME data analysis

In this chapter, the data provided by PRIME will be discussed. To obtain some useful insights on the data and to find results, the data needs to be preprocessed. In this section, the data is analysed and certain decisions are discussed, which were important while filtering out missing values and less important features from the data. It is also important to assess which features are more important than others for this research. This will also be done in the section preprocessing. To gain some more insight on the dependencies between the features in the data, the independence of the predictors are analysed in the section Analysis of the data,

3.1 PRIME data

PRIME distributed a survey asking questions about students' study behaviour and beliefs and some other questions related to this. The data set obtained from PRIME consists of an excel file that contains the answers that were filled in by students from different faculties studying at the TU Delft. In the context of study habits and beliefs, there are 8 questions included in the survey. These questions should give some understanding on students' study habits and beliefs.
The survey is divided in 8 subtopics:

- Informed consent
- Learning Strategies Free Report
- Prior Math Achievement
- Learning Strategies
- Math Self-Efficacy
- Study habits and beliefs about learning
- Grade goal
- Demographics

The topic of interest is study habits and beliefs about learning and in the model this is based on Prior Math Achievement, Math Self-Efficacy and Grade Goal. Hence, these three variables are called the predictors in the model, see figure 3.1.

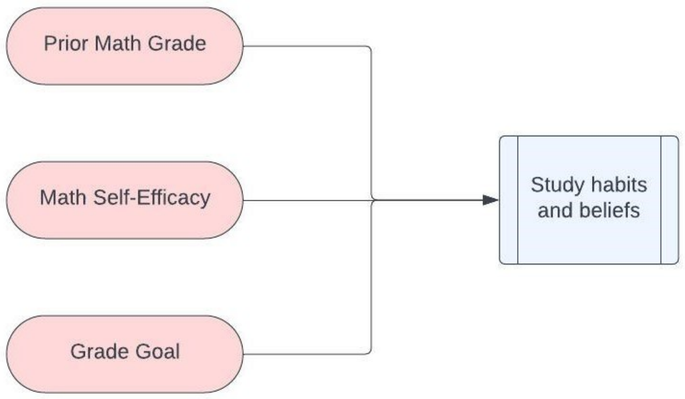


Figure 3.1: Questions corresponding to the topic study habits and beliefs.

Prior math grade: The prior math grade is a variable that is obtained from just one question. The possible answers range between 1 and 10 and are all integers as this variable is categorical. The grade represents the most recent math grade from previous education, such as secondary school mathematics.

Math Self-Efficacy: The mathematics self-efficacy gives insight on how confident a student is in the mathematics course they are taking. This variable is not just measured by one answer to one question, but there are 5 statements, which the students need to rank from 1 to 5. Here, 1 implies that the student strongly disagrees with the statement and 5 means that the student strongly agrees with the statement. These 5 rankings all together result in the Math Self-Efficacy predictor required. To combine all the rankings, in this model the average of the rankings are considered.

Grade Goal: This is the grade that the student aims to achieve for the mathematics course they are taking. To determine the grade goal, there are three questions that need to be answered by the student:

1. Which grade are you aiming for in this mathematics course? Possible answers range from 1 to 10.
2. What is the lowest grade you would be satisfied with for this mathematics course? Possible answers range from 1 to 10.
3. Which grade do you expect to earn for this mathematics course? Possible answers range from 1 to 10.

Combining the responses in one variable is done by taking the average of the responses for each student. This average will now be referred to as the grade goal, which then contains the information from the three components.

Each of these predictor variables, constructed as mentioned above, then predict the outcome variable study habits and beliefs. This outcome variable consists of 8 questions, see figure 3.2, which carry information about study habits and beliefs combined. To each question the model is applied individually.

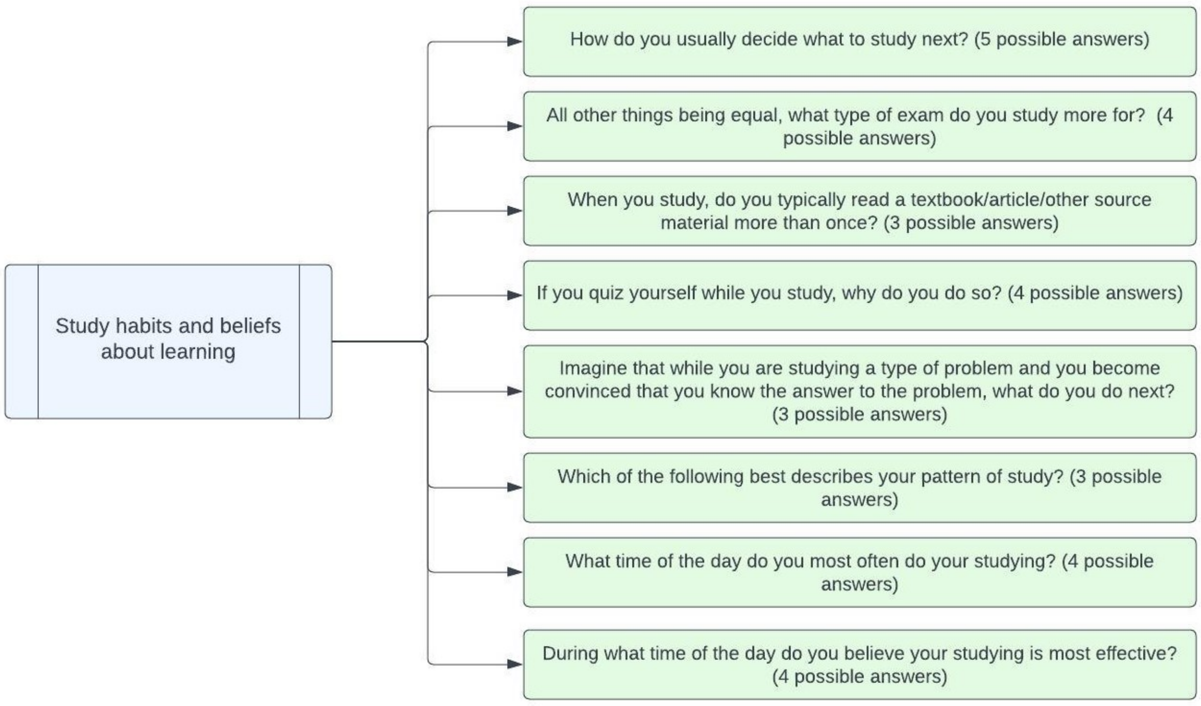
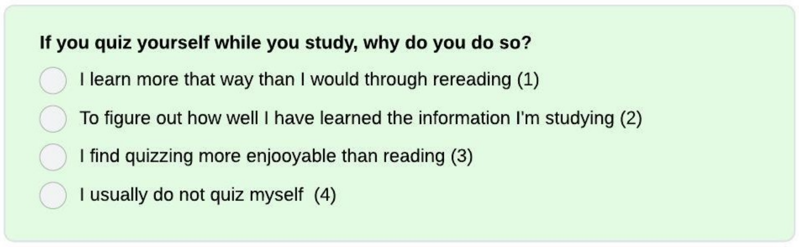

Study habits and beliefs about learning
How do you usually decide what to study next? (5 possible answers)
All other things being equal, what type of exam do you study more for? (4 possible answers)
When you study, do you typically read a textbook/article/other source material more than once? (3 possible answers)
If you quiz yourself while you study, why do you do so? (4 possible answers)
Imagine that while you are studying a type of problem and you become convinced that you know the answer to the problem, what do you do next? (3 possible answers)
Which of the following best describes your pattern of study? (3 possible answers)
What time of the day do you most often do your studying? (4 possible answers)
During what time of the day do you believe your studying is most effective? (4 possible answers)

Figure 3.2: Questions corresponding to the topic study habits and beliefs.

Consider question 4 from figure 3.2. The student can now give 4 possible answers to the question, which are shown in figure 3.3. Each of these categorical responses are encoded with numbers ranging from 1 to 4. The multinomial logistic regression model now uses the responses from the three predictors in figure 3.1 and the responses to question 4 from figure 3.3. The aim of the model is to predict the probabilities for each category 1 to 4 from figure 3.3 that the category was chosen by the student with certain choices for the predictors.


If you quiz yourself while you study, why do you do so?

- I learn more that way than I would through rereading (1)
- To figure out how well I have learned the information I'm studying (2)
- I find quizzing more enjooyable than reading (3)
- I usually do not quiz myself (4)

Figure 3.3: Question 4 from study habits and beliefs.

This can be done for all questions that influence the study habits and beliefs. Thus, the result from the model is a probability vector for each question from figure 3.2 that indicates the probability that a category will be chosen by the student.

3.2 Preprocessing

The survey conducted by PRIME contains 286 responses from students following mathematics courses at the TU Delft. As explained in the previous section, there are three predictors that predict the probabilities for the outcomes of the questions in figure 3.2. Before applying the model to the data from PRIME, the data must be preproccesed. Preprocessing includes dealing with missing values and cleaning up the data. To start off with cleaning the data, the first step is creating a dataframe consisting of only the columns that are needed for the modelling steps. Now this dataframe only contains valuable information that is needed for the model. The data is an excel file that contains the responses, so there could be questions that have not been filled in. These missing values are replaced by 0's as there are no variables that can attain the value 0. Now for each variable the number of zeros is counted to check how many missing values there are. Apparently, there are missing values for the predictor variable grade goal and some missing values for some of the study habits questions. There are exactly 7 missing values for grade goal, so these rows are deleted from the data set. Now checking the number of zeros in the data set again indicates that there are no missing values in the data set anymore. All missing values are thus taken care of and the data is ready to run for the model. The final number of rows left in the data set is 279. The code for this preprocessing can be found in appendix B.

3.3 Analysis of the data

There are a few assumptions that were made for this model. The model is essentially the same as the model used in a previous project by (Schriemer 2023). Most of the assumptions were checked in this project, so these are the still valid for this project, because the data is also the same as used in (Schriemer 2023). However, the correlations between the variables will be elaborated on.

Internal consistency was treated, however since some predictors are constructed based on other features, the internal consistency for these need to be checked as well. In order to check this the correlations are calculated for the predictors. These are calculated using the default in R which is the Pearson method. This is also the most common used method for the correlation coefficient. The correlation according to Pearson is described in definition 3.1 (Berman 2016).

Definition 3.1 (Pearson's correlation) Pearson's correlation of two variables x and y is defined as follows:

$$\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

Consider predictors x and y to which the Pearson correlation can be applied to all points (x_i, y_i). This results in a correlation matrix when applied to all predictors, which can be visualized as in figure 3.4.

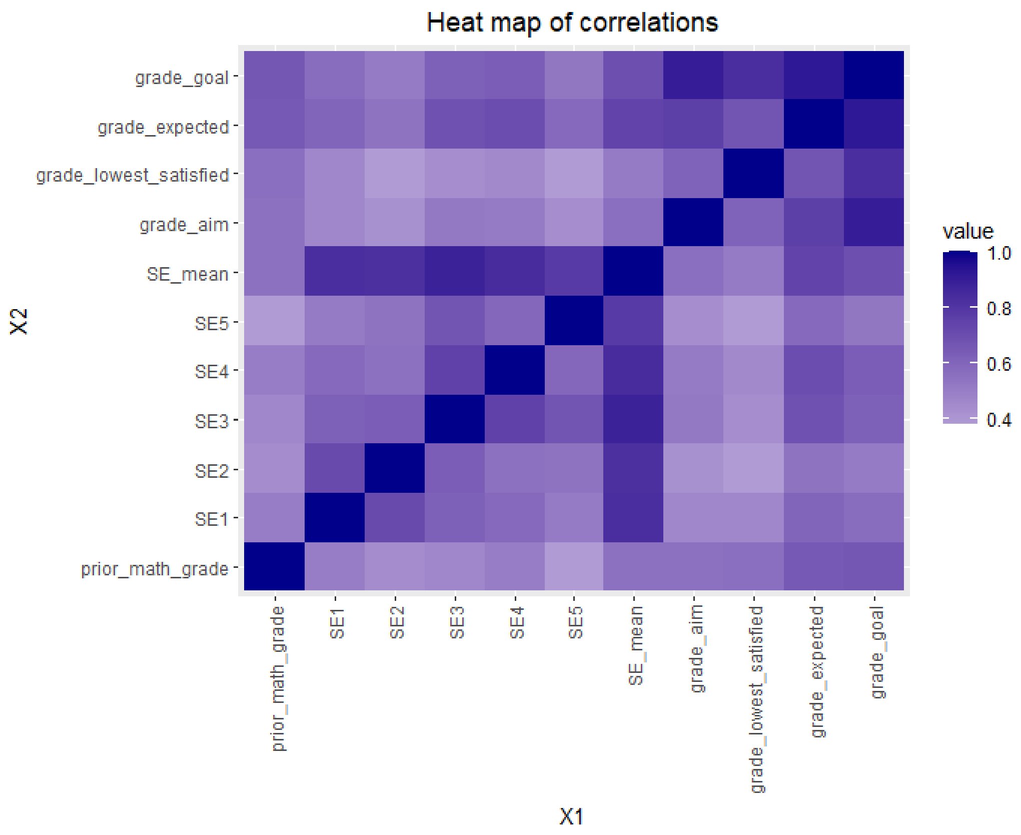


Figure 3.4: Correlation between independent variables

Figure 3.4 shows that there is clearly not a lot of correlation between the sub-questions that are used to construct the predictor Mathematics Self Efficacy. The variable used for the model is indicated by SE mean as this is the average of SE1, ..., SE5. In the figure a darker block indicates that the correlation coefficient is high and a light block refers to low correlation. High correlation is when the value is closer to 1 and a lower correlation is closer to 0.

Each of these correlations can be looked at more in depth. This can be done by looking at the scatter plots of the pairs with a higher correlation than expected. As it is expected that there is correlation between variables that are constructed from each other. Similarly, it is expected that the predictor variables are all independent of each other and thus have a low correlation.

Some of the pairs for which the correlation was slightly higher are plotted in figure 3.5.

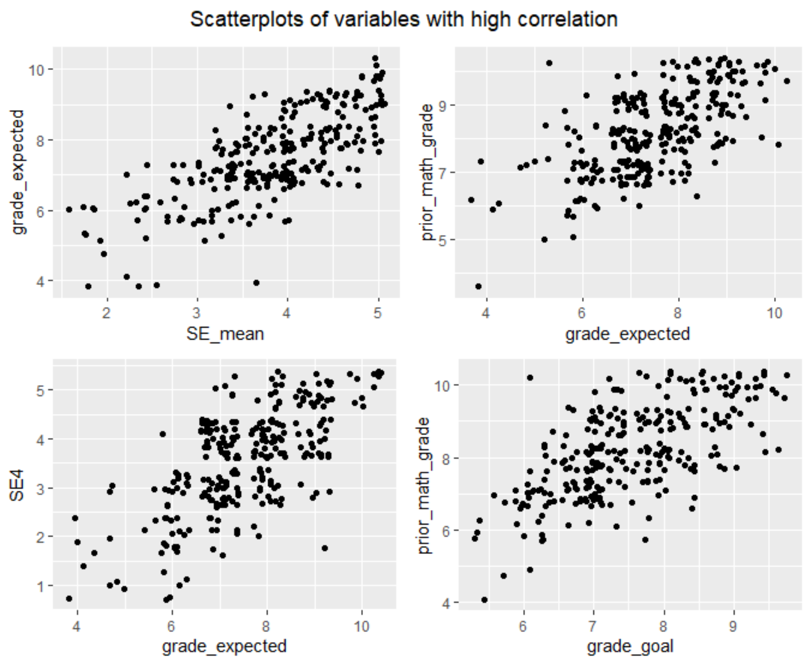


Figure 3.5: Scatter plot of pairs with high correlation.

In the figure, the slightly higher correlations are reflected in the plots. It is clear that there is a slight linear line visible in each of these scatter plots. This can be compared to some pairs that seems to have a low correlation based on the heat map in figure 3.4. This is visualized in figure 3.6.

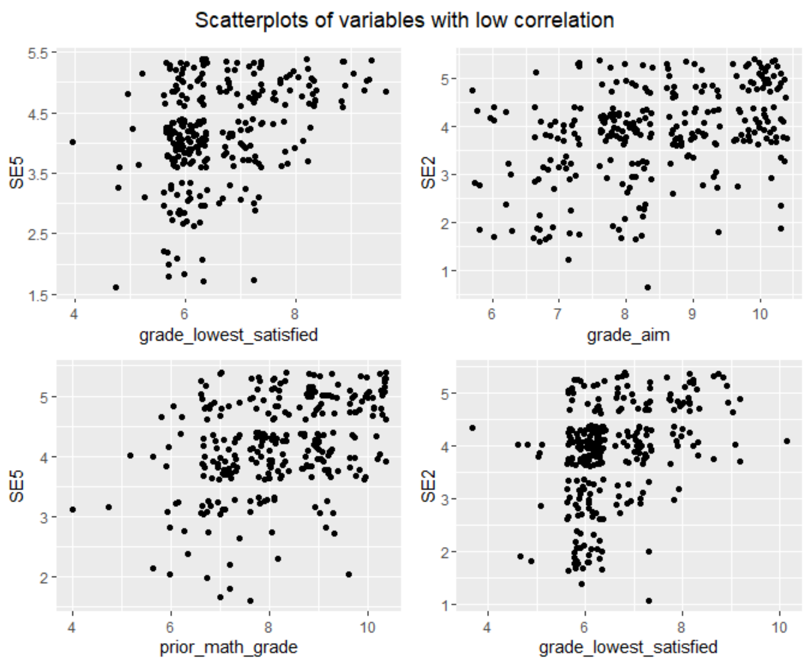


Figure 3.6: Scatter plot of pairs with low correlation.

Comparing the scatter plots from figure 3.5 to the ones in figure 3.6 leads to the conclusion that the pair in figure 3.5 are quite correlated. However, according to (Tabachnick and Fidell 2013) these are still not considered strongly correlated as this would mean a correlation of 0.90 or higher, which is not the case. Hence, here it can be concluded that there are some pairs that are moderately correlated, but the assumption of independence of predictors is still satisfied. The code needed to generate the scatterplots and the heat map can be found in appendix B.

Chapter 4

Model definition

This chapter includes a description of the model used for this project. Here, the multinomial logistic regression model is applied to the data obtained from PRIME. It is important to define variables, parameters and indices, which is also done in this chapter. The model will be applied in R by using the `UPG` package which uses Markov Chain Monte Carlo method using the Gibbs sampler. This will be elaborated on in this chapter.

4.1 Multinomial Logistic Regression Model

To apply Multinomial Logistic Regression, there are certain parameters and variables that need to be defined in the context of the data that is provided. Multinomial logistic regression is used to predict a probability π_{ij} that the outcome variable $\mathbf{y}$ is equal to some category j for a student i with its specific values of the explanatory variables. Hence, the aim is to predict a dependent variable y_i, which is for each iteration of the model one of the questions from figure 3.2. There are 8 questions that describe the students' study habits and beliefs, which implies that each student i answered each of these 8 questions. Hence, the outcome variable is an $(n\times 8)-$matrix as there are n students that answered these 8 questions. To obtain results for esch of the 8 questions the code in appendix C and D are used, where the question number needs to be specified as part of the preprocessing. The outcome variable y_i is an n-dimensional vector, which represents the responses given by the n students for a fixed question . Again, the responses $j \in \{1, \dots, m\}$ are the categories of each question. For the sake of reducing complexity, the model is built for each response variable from figure 3.2 separately.
As explained in chapter 3, there are three predictors involved in predicting the probabilities that the outcome is equal to some category j. These three predictors are denoted as X_1, X_2, X_3, where these variables represent the prior math grade, grade goal and the mathematics self-efficacy. These predictor variables are then incorporated in the multinomial logistic regression model as described in chapter 2.
Now, as the variable of interest is the probability, this is defined as $\pi_{ij} = \mathbb{P}(y_i = j)$. Hence, in words this means that π_{ij} denotes the estimation of the probability that a student with same values of explanation would choose category j for some fixed question. The expression for the multinomial logistic regression model is then described in definition 4.1.

Definition 4.1 (MLR model) Given some outcome variable y_i that can attain values $j \in \{1, \dots, m\}$ with $m > 2$ and predictor variables X_1, X_2, X_3, then for each student $i \in \{1, \dots, n\}$ the probability π_{ij} that student i has chosen category j is defined as

$$\pi_{ij} = \frac{e^{\beta_{0j}+\beta_{1j}X_{i1}+\beta_{2j}X_{i2}+\beta_{3j}X_{i3}}}{1+\sum_{l=1}^{m-1} e^{\beta_{0l}+\beta_{1l}X_{i1}+\beta_{2l}X_{i2}+\beta_{3l}X_{i3}}} \qquad \text{(for } j \in \{1, \dots, m-1\}\text{)}$$

$$\pi_{im} = 1 - \sum_{j=1}^{m-1} \pi_{im} \qquad \text{(for } j = m\text{)}$$

4.2 Markov Chain Monte Carlo method

The Markov Chain Monte Carlo method (MCMC) is the underlying method for sampling from the posterior distribution as used in the UPG package. It is a commonly used family of algorithms and has been involved in the emergence of Bayesian approach (McElreath 2020). As the name already suggests, this algorithm considers some Markov Chain which is then simulated using the Monte Carlo method. In order to understand the algorithm, there is first some prior knowledge about Markov Chains and the Monte Carlo method that needs to be reviewed. The method involves a Markov Chain which is a random walk, see definition 4.2, as defined in (Grimmett and Welsh 2014).

Definition 4.2 (Markov Chain) Consider a sequence of of random variables $\boldsymbol{Z} = (Z_n : n \geq 0)$. This sequence is now called a Markov Chain, when the Markov Property is satisfied. In words, this means that the distribution of the $(n+1)$st value of the sequence only depends on the nth value and is conditionally independent of all other values of the sequence before that. Formally, it holds that for all $n \geq 0$

$$\mathbb{P}(Z_{n+1} = i_{n+1} | Z_n = i_n, \ldots, Z_0 = i_0) = \mathbb{P}(Z_{n+1} = i_{n+1} | Z_n = i_n)$$

Such a sequence that satisfies the Markov property is then used in the MCMC algorithm. Before explaining how this is applied to MCMC, Monte Carlo simulation needs to be explained. The Monte Carlo method is a method to find an approximation of some properties of a distribution by repeatedly generating random samples from that probability distribution. Now these random samples that are generated from a probability distribution are then random variables that can be denoted by Z_i. The main idea of MCMC is using Monte Carlo to draw random samples to find some properties and each of these random samples that are taken then form a Markov Chain. Hence, the sequence $Z = (Z_n : n \geq 0)$ of Monte Carlo simulations is considered to be a Markov Chain. This means that the next random sample Z_{i+1} resulting from the Monte Carlo simulation only depends on the current sample Z_i and does not depend on all the random samples that were generated before that (Van Ravenzwaaij, Cassey, and Brown 2016).

In the Bayesian setting, the MCMC algorithm can be used to approximate the posterior distribution of the model parameter. This is the underlying method that is used in most of the packages in R. A lot of information can be derived about the posterior distribution using the MCMC method. To apply MCMC to the Bayesian approach, it can be useful to draw random samples Z_i from the posterior distribution. This sequence, denoted by $Z = (Z_n : n \geq 0)$, is then a Markov Chain. The stationary distribution of this Markov Chain is now the posterior distribution. In this way MCMC can find the sample means, quantiles and more properties of this posterior distribution, while keeping the compilation time of the program realistic (Van Ravenzwaaij, Cassey, and Brown 2016).

This is now the underlying method used for approximating the posterior distribution for the `UPG` package and the `BRMS` package. However, there is still a difference as the UPG package uses MCMC using Gibbs sampling and the `BRMS` package uses Hamiltonian Monte Carlo sampling. This will be discussed briefly in the next sections.

4.3 MLR using the UPG package

The abbreviation `UPG` refers to Ultimate Pòlya Gamma. The package uses latent variable representation based on some random variables that follow a Pòlya Gamma distribution. An observed variable contains information that can directly be observed. A latent variable influences an observable variable. This means that a latent variable is not directly observable or measurable (Price 2023).

Using this notation involving latent variables is used to find several different logistic regression models. More details on the exact representation of this can be found in (Zens, Frühwirth-Schnatter, and Wagner 2023). To find the resulting estimates Gibbs sampling algorithms are used.

The `UPG` package uses MCMC using the Gibbs sampler to fit the model. Unlike most packages, the `UPG` package doesn't require a formula as input. Recall the model notation as in definition 4.1. Instead of a formula the predictors X are given as input in the form of a matrix and the response variable y is a vector given as input. Now, there are several other parameters that need to be specified such as the model that is used, which is in this case the Multinomial Logistic Regression model. Other than that, the baseline category needs to be given. The default value for this is the category that occurs the most often. However, for consistency the baseline category is chosen to be category 1 for all of the models. Note that the categories are relabelled when fitting the model, so this does not necessarily need to be the last category. The probability for category 1 is now based on all other probabilities so the regression coefficients for category 1 are equal to 0. The number of iterations is chosen to be 10.000, which should be enough to give an accurate result without increasing the running time too much. There is also a parameter that specifies the burn ins using during Gibbs sampling. It is recommended to take this parameter equal to 2000 when estimating a model according to the documentation.

Gibbs sampling will be further explained in the following subsection.

4.3.1 Gibbs sampling

Gibbs sampling is used when it is hard to sample from the joint distribution, but easy to sample from the conditional probabilities. Here, the samples are taken from univariate Gaussian distributions. Consider example 4.1.

Example 4.1 (Gibbs Sampling) Consider some posterior distribution $p(y|x)$ that we want to obtain samples from. With Gibbs sampling it is easy to sample from the conditional distributions, $p(y|x)$ and $p(x|y)$. It is assumed that these follow a Gaussian distribution with center 0 with the covariance matrix as the variance.
Now, say the aim is to generate S samples. Start with initial values $(x^{(0)}, y^{(0)})$. Now, the algorithms applies the following steps:

1. Initialize starting values $(x^{(0)}, y^{(0)})$
2. Find $x^{(1)}$ by sampling from $p(x^{(1)}|y^{(0)})$
3. Use $x^{(1)}$ to find $y^{(1)}$ by sampling from $p(y^{(1)}|x^{(1)})$
   From this the first sample $(x^{(1)}, y^{(1)})$ follows.
4. Find $x^{(2)}$ by sampling from $p(x^{(2)}|y^{(1)})$
5. Use $x^{(2)}$ to find $y^{(2)}$ by sampling from $p(y^{(2)}|x^{(2)})$
   From this the second sample $(x^{(2)}, y^{(2)})$ follows.
   This goes on until S samples are generated.

In this case take some starting value for $\beta^{(0)}$. Now using the Gibbs sampling scheme the next point $\beta^{(1)}$ is taken from a Gaussian distribution, determined by the previous point. Now, the Gibbs sampler keeps sampling until the number of specified samples is reached.

4.4 MLR using the BRMS package

As mentioned before, the `BRMS` package uses the Hamiltonian Monte Carlo method to generate the posterior distribution. This package differs from the `UPG` package in a sense that in this package the model is translated to C++ and fitted with Stan. The results that follow from this are then converted to an object in R. Figure 4.1 from (Bürkner 2017) gives an overview of how the model exactly works.

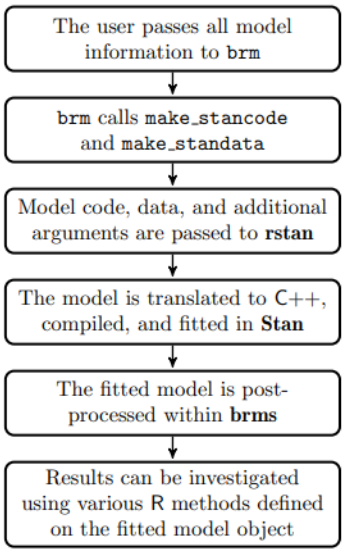

The user passes all model information to brm
brm calls make_stancode and make_standata
Model code, data, and additional arguments are passed to rstan
The model is translated to C++, compiled, and fitted in Stan
The fitted model is post-processed within brms
Results can be investigated using various R methods defined on the fitted model object

Figure 4.1: Modelling steps in the `brm()` function in R (Bürkner 2017).

From figure 4.1, it is clear that the first step that is taken is to preprocess the data to input in the model. How it works in `BRMS` for categorical response variables is that the response variable Y should be a matrix, where the number of rows is n (the number of students) and the number of columns is m (the number of categories for the specified question). The matrix is now constructed such that the results are converted in transposed vectors, where for $Y_i = j$, row i consists of a 1 in column j and 0 in all other columns.

4.4.1 Hamiltonian Monte Carlo sampling

The Hamiltonian Monte Carlo method works in a way where some auxiliary variables are introduced. This method is more physics related as the variables that are introduced are considered to be momentum. Furthermore, the Hamiltonian function is involved, which is the function of total energy in a system. There are two types of energy involved: kinetic energy and potential energy. In this section there won't be a very elaborate explanation about these concepts from physics as this is beyond the scope of the project. The main focus will be on the mathematics behind it and the relevant information for this project. Now consider Bayes' formula $p_{\boldsymbol{\beta}|\mathbf{Y}} \propto p_{\mathbf{Y}|\boldsymbol{\beta}} \cdot p_{\boldsymbol{\beta}}$ (M. J. Betancourt and Girolami 2013).

The Hamiltonian Monte Carlo method takes draws from the joint distribution of $p_{\mathbf{Y},\beta}$. In this joint distribution Y is used to denote the auxiliary variables. Now this joint distribution can be used to define the Hamiltonian. The Hamiltonian is a function used in physics and gives the total energy in a system. In this case the Hamiltonian becomes the following (M. J. Betancourt and Girolami 2013).

$$\begin{aligned} H(\mathbf{Y},\boldsymbol{\beta}) &= -\log(p_{\mathbf{Y},\boldsymbol{\beta}}) \\ &= -\log(p_{\mathbf{Y}|\boldsymbol{\beta}} \cdot p_{\boldsymbol{\beta}}) \\ &= -\log(p_{\mathbf{Y}|\boldsymbol{\beta}}) - \log(p_{\boldsymbol{\beta}}) \\ &= T(\mathbf{Y}|\boldsymbol{\beta}) + V(\boldsymbol{\beta}) \end{aligned}$$

Here $T(\mathbf{Y}|\boldsymbol{\beta})$ denotes the kinetic energy in the system and $V(\boldsymbol{\beta})$ denotes the potential energy and thus result in the total energy which is the Hamiltonian. What this now does is, the function $V(\boldsymbol{\beta})$ is passed on to a program written in Stan language. This is important as this contains the information of the prior distribution.

The current parameter θ is then combined with some ρ randomly drawn from a MultiNormal$(0, M)$ distribution where M is the Euclidean metric and this refers to the Multivariate Normal distribution. This then becomes the joint system (ρ, θ). Using the Hamilton equations this can be expressed as

$$\begin{aligned} \frac{\partial\theta}{\partial t} &= +\frac{\partial H}{\partial\rho} = +\frac{\partial T}{\partial\rho} \\ \frac{\partial\rho}{\partial t} &= -\frac{\partial H}{\partial\theta} = -\frac{\partial T}{\partial\theta} - \frac{\partial V}{\partial\theta} \end{aligned}$$

This yields

$$\begin{aligned} \frac{\partial\theta}{\partial t} &= +\frac{\partial T}{\partial\rho} \\ \frac{\partial\rho}{\partial t} &= -\frac{\partial T}{\partial\theta} - \frac{\partial V}{\partial\theta} \end{aligned}$$

These differential equations can now be solved numerically using the Leapfrog integrator. The Leapfrog integrator applies an integration algorithm to solve a system of differential equations numerically and specifically finds stable solutions. The Leapfrog integrator starts with (ρ, θ), where ρ is drawn from MultiNormal$(0, M)$. By now discretizing over intervals of length ϵ it applies L leapfrog steps in which ρ and θ are updated. By applying L leapfrog steps to ϵ intervals after $L \cdot \epsilon$ time the estimation (ρ^*, θ^*) is found. The next step in HMC is to apply the Metropolis Acceptance step, where the error is measured to judge whether the numerical result from the leapfrog is good enough. In this step the result (ρ^*, θ^*) are compared to (ρ, θ) and the probability of the estimated (ρ^*, θ^*) to be accepted is then

$$\min(1, H(\rho^*, \theta^*) - H(\rho, \theta))$$

If the resulting (ρ^*, θ^*) is not accepted, the previous parameter is returned back to use in the next step for another iteration through the Leapfrog. Hence, here it is clear that the parameters before the current one can decide the next parameter, so this is not a Markov Chain. (M. J. Betancourt and Girolami 2013)

Chapter 5

Results

The multinomial logistic regression model as described in chapter 4 will be applied to the data from PRIME. This is done using the `UPG` package in R. The `UPG` package gives the option to use the Bayesian approach to multinomial logistic regression. The package returns the posterior means together with an indicated credible interval, which comprises 0.025 and 0.975-quantiles of the posterior distribution of individual parameters in the model by default. For simplicity, the results for only one question from 3.2 will be discussed. For this chapter, the question: 'If you quiz yourself while you study, why do you do so?' will be considered. The credible intervals for the `UPG` model applied to all data will be discussed elaborately. Some additional plots are given in the appendix. The code for the results following from applying the `UPG` models can be found in appendix C and the code for the `BRMS` model results can be found in appendix D.

5.1 UPG model applied to uniformly generated data

In general in the case that there is no data available, the data can be generated for the sake of assessing the performance as the ground truth is known in this case. Here, the information that is already available before obtaining data is the range of values of the predictor variables and the possible category options for the outcome variable. Using this information, it is assumed that each predictor follows a uniform distribution with values in their defined range. Similarly, the predictor variable can be modelled with a random uniform distribution with categories ranging from 1 to 4 in this case. This is dependent on the number of categories that corresponds to the considered question. Here, the uniform distributions are randomly generated, which implies that the results should not be as accurate. The idea is that the model uses the uniformly generated data to predict the probabilities for each category j that students with the same given information choose that category. The category with the highest probability can now be compared to the choices that were made in reality according to the generated data. The prediction is then accurate when the category with the highest probability is equal to the category that was given from the data. The ratio of accurately predicted divided by the total number of data points n is then defined as the accuracy. This definition of accuracy will be used throughout the project to assess the performance of each model. Running the code for this, see appendix C, results in an accuracy of exactly 0.22 in approximately 34.16 seconds. Hence, as expected this model does not perform very well, as a result of the randomness of the data.

5.2 UPG model applied to PRIME data

`UPG` stands for Ultimate Pòlya Gamma as the `UPG` package is based on Ultimate Pòlya Gamma MCMC algorithms which are very efficient (Zens, Frühwirth-Schnatter, and Wagner 2022). The underlying model of the `UPG` package is a Random Utility Model, as further explained in (Schriemer 2023), using the Gibbs sampler using Markov Chain Monte Carlo method to find posterior estimates. Using the default prior as defined in the `UPG` package, the model can be applied to the data. This results in posterior means for individual parameters and their corresponding credible intervals. Now the predictions can be obtained using the built in function 'predict'. This function now gives the predicted probabilities again with their corresponding credible intervals. The predicted probabilities obtained using the posterior means for the first 15 students can be found in table 5.1. This table indicates for each student the probability for each category that a student with the same choices for the explanatory variables chooses a certain category. Then the column with actual choice according to the given data indicates the choice that was actually made. The last column indicates whether the category with the highest probability corresponds to the actual choice that was made. The category with the highest probability also corresponds to the category j that has the highest log-odds. Recall the definition of the log-odds:

$$\log\left(\frac{\pi_{ij}}{\pi_{im}}\right) = \beta_{0j} + \beta_{1j}X_{i1} + \beta_{2j}X_{i2} + \beta_{3j}X_{i3}$$

The category with the highest probability then corresponds to the category with the highest odds, $\frac{\pi_{ij}}{\pi_{im}}$, and since the logarithm is monotone increasing also the highest log-odds. So this last column in table 5.1 indicates whether or not the predicted probabilities were accurate. Trivially, a one indicates that the choice was correctly predicted and a zero indicates that the category with the highest probability is not the same as the actually chosen category. The amount of the correctly predicted probabilities divided by the total number of students gives the accuracy. In this case, using all the data, the accuracy is 0.4265233. Running the model takes more or less 36 seconds.

Student	Category 1	Category 2	Category 3	Category 4	Actual choice	Correctly predicted
1	0,30762561	0,42180346	0,07414183	0,19642911	2	1
2	0,28633805	0,41687529	0,10243466	0,19435199	2	1
3	0,29005795	0,43739646	0,08233445	0,19021115	2	1
4	0,14465704	0,46485674	0,03829729	0,35218893	4	0
5	0,30804015	0,37392481	0,13543621	0,18259883	2	1
6	0,2919788	0,39179336	0,0610939	0,25513394	1	0
7	0,19764704	0,46122461	0,0485231	0,29260525	4	0
8	0,29005795	0,43739646	0,08233445	0,19021115	2	1
9	0,29336295	0,42382782	0,0643345	0,21847473	2	1
10	0,24672062	0,40285258	0,05003779	0,30038901	4	0
11	0,44849975	0,32857179	0,10049731	0,12243116	1	1
12	0,25025216	0,47050885	0,05725808	0,22198092	2	1
13	0,26365913	0,44018033	0,06144568	0,23471486	2	1
14	0,18389395	0,50719251	0,09125116	0,21766238	4	0
15	0,2224259	0,46986441	0,0649793	0,24273038	4	0

Table 5.1: Mean of the predicted probabilities for each category with the actual choices for the first 15 students in the data set.

Now, for these rows where the choice was not correctly predicted, it is useful to consider the credible intervals for each prediction for each category. In table 5.1, the highlighted rows are the students for which the model did not predict the choice actually made by the student correctly. These are thus the rows corresponding to the zeroes in the column 'Correctly predicted'. Considering, for example, student 4 from table 5.1, it is clear that student 4 chose category 4 for this question. However, from the probabilities it is clear that a student with the same choices as student 4 has the highest probability to choose category 2. The probability for category 2 is also quite high compared to the other categories, but category 4 comes the closest. This is where the credible interval is useful. From the credible interval it can be deduced whether there is overlap and whether the prediction that was made was even close to predicting correctly. This is then a measure for validity of the predictions. For the highlighted students, the credible interval can be constructed for each category using the 0.025-quantile and the 0.975-quantile.
Table 5.2 shows the 0.025-quantile of the predicted probabilities for each of the categories.

Student	Category 1	Category 2	Category 3	Category 4
1	0,24851822	0,35538195	0,04382016	0,14569952
2	0,19290683	0,31334272	0,04693573	0,12041049
3	0,22997725	0,37255313	0,05163877	0,14050699
4	0,07214323	0,32723691	0,00953726	0,2224295
5	0,16026892	0,22197449	0,0367071	0,08203255
6	0,21635202	0,29741681	0,02911116	0,17408792
7	0,12901684	0,36670263	0,01947705	0,21115473
8	0,22997725	0,37255313	0,05163877	0,14050699
9	0,23343719	0,35356681	0,03514426	0,16260595
10	0,17652843	0,30321089	0,02212232	0,20822252
11	0,30408805	0,21164923	0,03468121	0,06248561
12	0,174679	0,38133162	0,02449079	0,15203373
13	0,20669602	0,37303814	0,03388748	0,18062659
14	0,10492841	0,38109895	0,03259404	0,12470977
15	0,16319142	0,40006606	0,03445375	0,18680728

Table 5.2: The 0.025-quantile of the predicted probabilities for each category for the first 15 students in the data set.

In table 5.3, the 0.975-quantile is shown for the first 15 students. These 15 students correspond to the same 15 students as were used in the other tables in this section. Also, the highlighted rows correspond to the students from 5.1.

Student	Category 1	Category 2	Category 3	Category 4
1	0,37003714	0,48853119	0,11093236	0,25258952
2	0,38766366	0,52204207	0,17930547	0,28513759
3	0,35186026	0,50341792	0,12035905	0,24421203
4	0,2441866	0,60658671	0,09459758	0,49726998
5	0,47915715	0,53995811	0,30707519	0,32582779
6	0,37323816	0,48878563	0,10642263	0,34842239
7	0,27677024	0,55486681	0,09283488	0,38190356
8	0,35186026	0,50341792	0,12035905	0,24421203
9	0,35882911	0,49421622	0,10087873	0,28029971
10	0,32744448	0,50464436	0,09192321	0,40451025
11	0,5958087	0,45923674	0,21157883	0,20851603
12	0,33493644	0,56112623	0,10490706	0,30281363
13	0,3242804	0,5087966	0,09638631	0,29371027
14	0,2790191	0,63366171	0,18549302	0,33074496
15	0,2855159	0,5417809	0,10479378	0,3029925

Table 5.3: The 0.975-quantile of the predicted probabilities for each category for the first 15 students.

Using the data from these highlighted students a credible interval for the probabilities can be obtained for each category. This is plotted for each of the highlighted students, see figure 5.1.

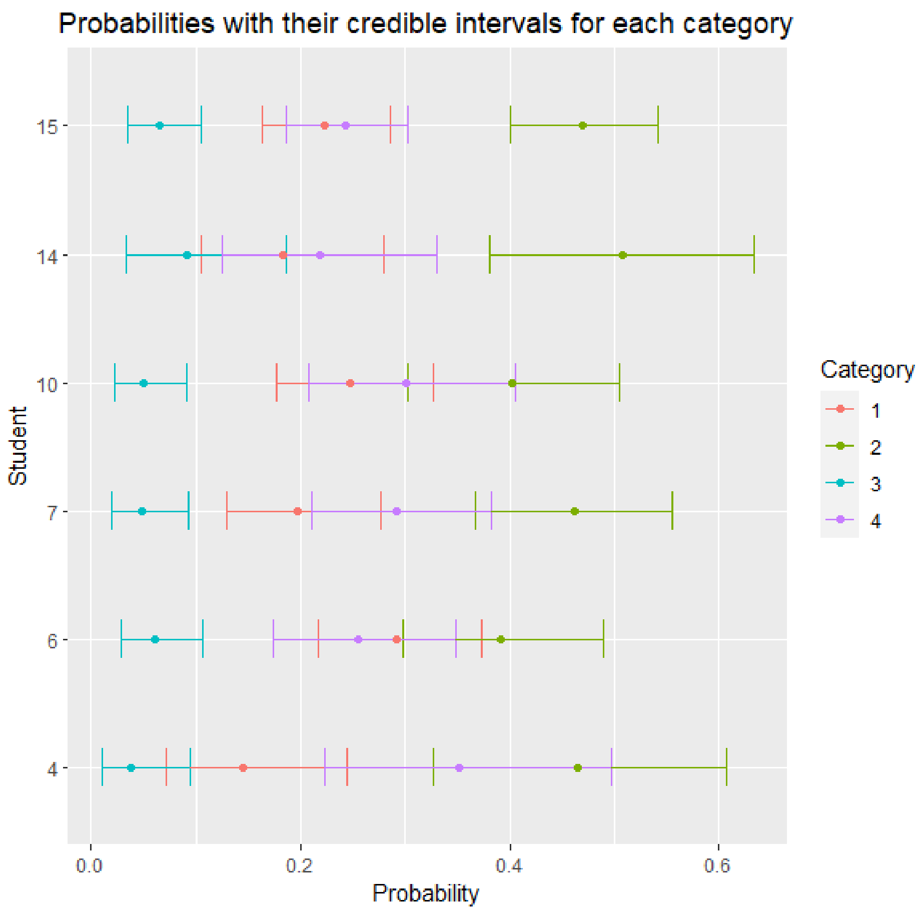


Figure 5.1: Credible intervals for those students amongst the first 15 students where the category with the highest probability does not correspond to the actual choice.

Figure 5.1 can help interpreting the results that follow from applying the model and the prediction accuracy can be assessed. Again, consider student 4. As seen before the actual choice was category category 2 and category 4 has the highest probability. In figure 5.1, we see that there is quite some overlap between the green and purple intervals. Hence, there is some overlap between the credible intervals of these categories. Since the credible intervals are interpreted as intervals with a 95% probability that the true probability is in this interval, it is clear that this prediction was close to predicting the actual choice and was not far off.

On the other hand, for student 15 this is not the case. The actual choice here is category 4 and the category with the highest probability is category 2. The second largest probability is indeed category 4, but in figure 5.1, we see that there is no overlap between the credible intervals. This prediction is thus less accurate compared to the prediction for student 4. This way of reasoning can be applied to all students for which the highest probabilities do not correspond to the actual choices and for some it can be found that the predictions were not far off.

Such credible intervals can also be constructed for the posterior distribution for each category j. These credible intervals for the posterior estimates are given in a plot in appendix A.1.

5.3 UPG model using Train-Test Split method

5.3.1 Train-Test Split method

The main idea behind the Train-Test Split method is that the total data is split up in two sets: a training set and a test set. The training set contains 90% of the data and the validation set contains the left over 10% of the data. Both, the training and the test set, are disjoint and randomly chosen. This means that once the training set is chosen randomly, the test set is fixed.

The training set is used to train the model and find the corresponding posterior distributions. This includes information about the posterior such as the posterior means and the credible regions. This information for the

$\hat{\beta}$'s for each category are used to predict the choices for the students in the test set. The test set is now a random set of students that contains 10% of the total data. The predictions that follow from this can now be compared to the actual choices to be able to assess the accuracy of the model. This is summarized in figure 5.2.

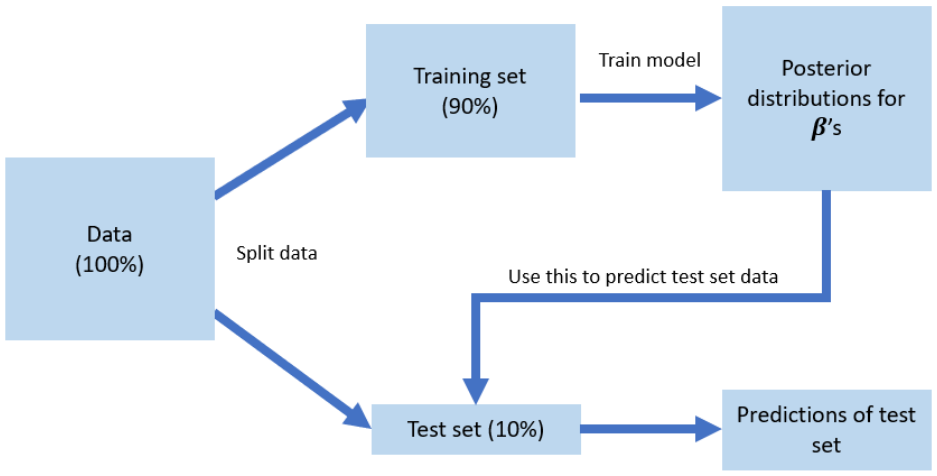


Figure 5.2: Overview of Train-Test Split method

The aim of the Train-Test Split method is to train a model and validate the model using this method of splitting the data. Besides estimating the accuracy of the model, cross validation can also be used to compare performance of different models (Refaeilzadeh, Tang, and Liu 2009).

5.3.2 Train-Test split method applied to the UPG model

The model from section 5.2 can also be applied to the data set that is split in a training and test set. The training and test set are chosen randomly using the function `sample()` in R. The validation set is constructed with 10% of the data, which is approximately 28 students chosen at random from the data set. The function `sample()` then generates 28 random numbers between 1 and 279, which is the number of rows of the data set. These random numbers are used as indices for the validation set and data set without this test set is then the training set. As explained in the previous section, the training set is now used to find posterior means and these are then used to predict probabilities using data from the validation set. Table 5.4, now gives the predictions for all the students from the test set. The table is constructed in a similar way as table 5.1. Training the model is finished is about 31 seconds.

Student	Category 1	Category 2	Category 3	Category 4	Actual choice	Correctly predicted
1	0,17776604	0,46947536	0,04822984	0,30452876	4	0
2	0,33521651	0,40124712	0,06472232	0,19881405	2	1
3	0,32309713	0,38821366	0,08294683	0,20574238	1	0
4	0,18069915	0,4606003	0,07292425	0,2857763	2	1
5	0,1138878	0,43971946	0,0390997	0,40729304	4	0
6	0,42720611	0,37585594	0,08929937	0,10763858	2	0
7	0,28188949	0,43829902	0,07296223	0,20684926	2	1
8	0,19011478	0,51853316	0,05981828	0,23153378	2	1
9	0,2868299	0,44604615	0,12885767	0,13826629	1	0
10	0,37806923	0,43188906	0,08016808	0,10987363	3	0
11	0,36026058	0,44223135	0,10465463	0,09285345	4	0
12	0,10536097	0,54366177	0,04229408	0,30868318	2	1
13	0,41755365	0,38207035	0,07802115	0,12235485	2	0
14	0,16527084	0,48779713	0,05491158	0,29202045	4	0
15	0,28768386	0,46169146	0,03764851	0,21297617	4	0
16	0,32433886	0,4185819	0,0863189	0,17076034	1	0
17	0,17261293	0,4416646	0,09262478	0,29309769	4	0
18	0,1114361	0,38792498	0,05523286	0,44540607	4	1
19	0,24508402	0,42725273	0,04503342	0,28262983	1	0
20	0,36026058	0,44223135	0,10465463	0,09285345	4	0
21	0,14068083	0,44932465	0,04935739	0,36063714	1	0
22	0,07564188	0,44263235	0,05067304	0,43105273	2	1
23	0,24058312	0,45219933	0,06093451	0,24628304	2	1
24	0,34601554	0,41260166	0,05024424	0,19113856	2	1
25	0,26641693	0,5135383	0,10152501	0,11851977	1	0
26	0,34601554	0,41260166	0,05024424	0,19113856	2	1
27	0,130542	0,46753873	0,05643323	0,34548604	2	1
28	0,2062584	0,46892298	0,11725376	0,20756486	2	1

Table 5.4: Predictions of the test set using the trained model from the training set

From the results in 5.4, the accuracy can be found. For the test set this is 0.4285714. This is slightly higher than the accuracy from section 5.2. For these estimates for the probabilities credible intervals can be constructed. Figure 5.3 shows the credible intervals for the first 15 students from table 5.4 whose actual choices were different from the category with highest predicted probability. So, the credible intervals are constructed for the first few students that have a 0 in the last column in table 5.4.

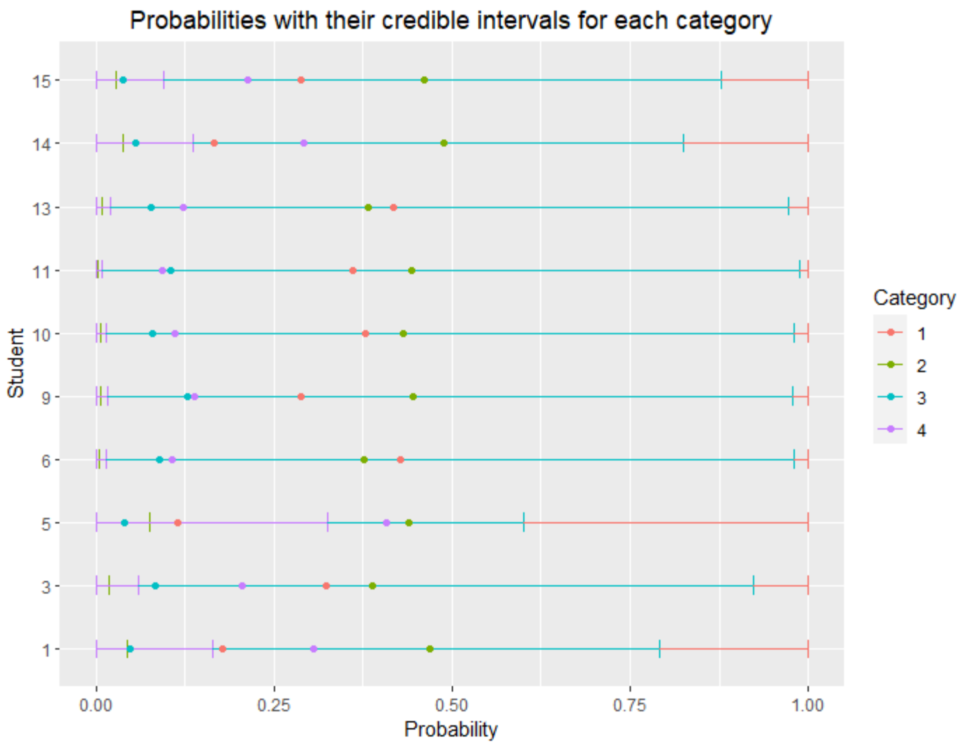


Figure 5.3: Credible intervals for the first few students from the test set for who the category with the predicted probability don't correspond to the actual choices.

From figure 5.3 it can be noted that the credible intervals are quite large and thus carry less information. The posterior estimates are very similar to the ones that were found in section 5.2. The credible intervals for the posterior estimates are given in a plot in appendix A.2.

5.4 UPG Model using K-fold cross-validation

Cross validation can also be applied in such a way that every data point used for validation once. This is called K-fold cross validation. Here, the data is split up in K sets of approximately equal size. Now, cross validations is applied in K times. Each iteration uses 1 of the K sets for the validation set, also called the test set. The other $K - 1$ sets are then used for the training set. In this way, each data point is used for the test set exactly once and $K - 1$ times to train the model. This is summarized in figure 5.4. The figure depicts the splitting process of K-fold cross-validation, where in this case $K = 10$.

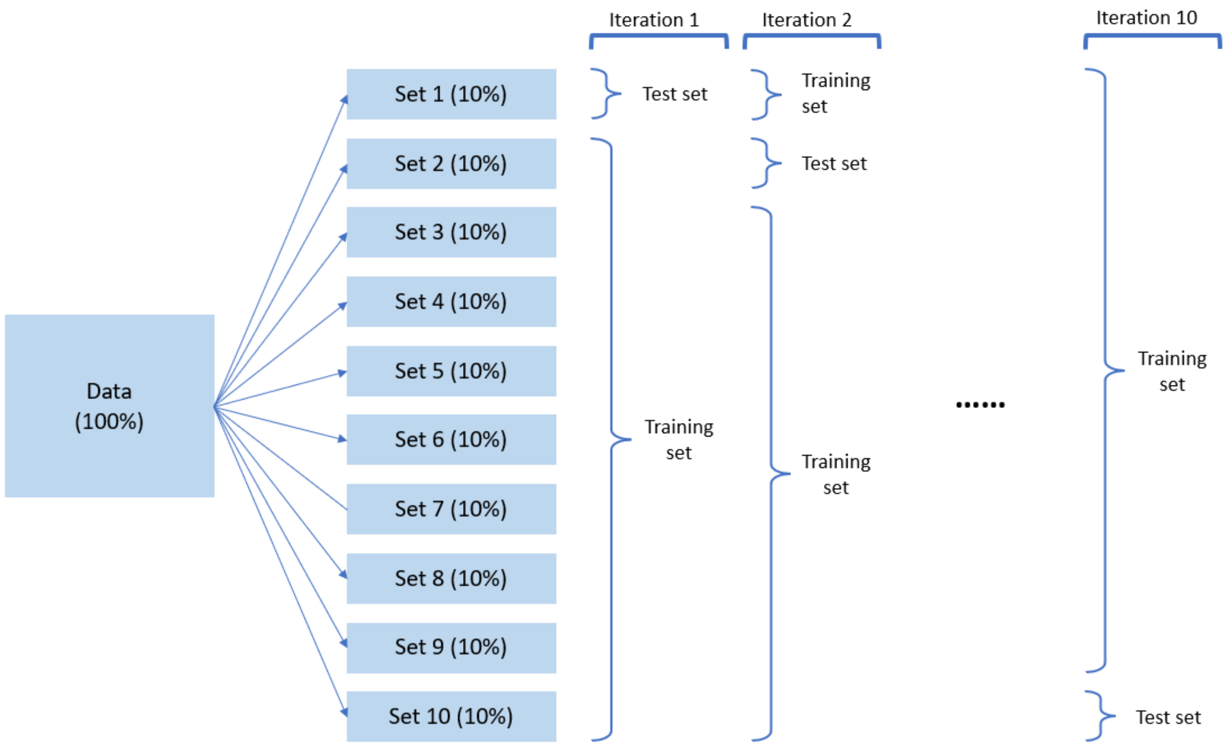


Figure 5.4: Overview of K-fold cross-validation with $K = 10$.

Applying K-fold cross-validation gives more insight into the validity of the accuracy. For each iteration in the K-fold cross-validation algorithm, the model is fitted based on different data and validated using different data. Hence, the prediction accuracy and estimated posterior distributions are independent of each other, since each of the sets are chosen disjoint from each other. In case there would be overlap between the training and test sets, this would lead to over-estimation of the model. For the K-fold cross-validation in this project, K is chosen to be equal to 10. Taking a too large number of folds results in a test set that is too small, which leads to a less accurate result (Refaeilzadeh, Tang, and Liu 2009). Taking $K = 10$ seems reasonable according to (Refaeilzadeh, Tang, and Liu 2009).

In this way, this can be applied to the `UPG` package to measure the validity of the accuracy that was found before. Here, the number of folds is considered to be 10. For each iteration, the training set is used to find the posterior means. These posterior means are used to then apply this to the data from the test set using the model from equation 2.10 to obtain the corresponding predictions of the choices. Each of these iterations take approximately 6 seconds and in total running the whole K-fold cross-validation takes roughly 53 seconds. The code in R used to obtain these results can be found in appendix C.

Iteration	Accuracy
1	0.4642857
2	0.3333333
3	0.3214286
4	0.4074074
5	0,5
6	0.4642857
7	0.4642857
8	0.4137931
9	0,25
10	0.3928571

Table 5.5: Accuracy for each iteration of 10-fold cross-validation.

For each iteration the accuracy is given in table 5.5. The average accuracy is then 0.4011677. It can be concluded that on average the predicting performance deviates from the predicting performance for all data with approximately 0.02. This is a close approximation and hence the model performs approximately the same when taking different training and test sets. The performance of the model is hence reliable. The model with the highest accuracy is saved and for this model the credible intervals of the posterior estimates could be compared to the credible intervals posterior estimates of the other two `UPG` models that were given before. The credible interval plot can be found in appendix A.3. It appears that there are very slight differences which are listed in this appendix. However, these differences are so small, there are not really any strong conclusions that can be drawn from this. It also depends on the data that is tested on, because this is a small set the accuracy can go up faster and in the same way the accuracy can also go down quickly.

5.5 BRMS model applied to PRIME data

The `BRMS` package can be used to predict the probabilities for each category using multinomial logistic regression in a Bayesian sense. The function `brm()` can be used to predict these probabilities for each category. The default prior is defined as in section 2.4.2 and multinomial logistic regression is applied as explained in section 4.4. Running the `BRMS` model takes approximately 410.121 seconds to compile. Now, applying the model again results in posterior distributions containing the posterior means and the corresponding 95% credible intervals. This can be fitted to end up with the predicted probabilities and their credible intervals. Table 5.6 gives the predicted probabilities for the first 15 students and as done before a column with the actual choices and a column that indicates whether the category with the highest probability corresponds to the choice.

Student	Category 1	Category 2	Category 3	Category 4	Actual choice	Correctly predicted
1	0,29679697	0,32852474	0,08628344	0,28839485	2	1
2	0,27510595	0,32644088	0,11450306	0,28395011	2	1
3	0,28378922	0,33893246	0,10000423	0,27727409	2	1
4	0,13694059	0,36941548	0,0456932	0,44795073	4	1
5	0,2909513	0,29840634	0,13681975	0,27382261	2	1
6	0,26563155	0,309407	0,06049773	0,36446372	1	0
7	0,18863001	0,36206071	0,05734032	0,39196896	4	1
8	0,28378922	0,33893246	0,10000423	0,27727409	2	1
9	0,28016226	0,33097417	0,07321085	0,31565272	2	1
10	0,22147263	0,31782365	0,04943551	0,4112682	4	1
11	0,42909306	0,25952202	0,11014913	0,2012358	1	1
12	0,24916957	0,36443852	0,07468166	0,31171025	2	1
13	0,25289309	0,34349572	0,07142909	0,33218211	2	1
14	0,18833785	0,39065245	0,12623612	0,29477359	4	0
15	0,21793781	0,36590804	0,08050891	0,33564524	4	0

Table 5.6: Mean of the predicted probabilities for each category with the actual choices for the first 15 students in the data set.

The 95% credible intervals that is now constructed for these 15 students consists of the 0.025 and 0.975-quantile. Table 5.7 gives the 0.025-quantile for each category.

Student	Category 1	Category 2	Category 3	Category 4
1	0,01210225	0,01099995	0,00170133	0,00127572
2	0,01132761	0,01023301	0,00219318	0,00116792
3	0,01184256	0,01127962	0,00206847	0,00110893
4	0,00188403	0,00649393	0,000299	0,00247671
5	0,0141112	0,00829148	0,00257582	0,00107297
6	0,00874795	0,00846661	0,00072388	0,00223391
7	0,0037401	0,00784055	0,00052715	0,00209945
8	0,01184256	0,01127962	0,00206847	0,00110893
9	0,00974833	0,01017628	0,00121148	0,0015279
10	0,00556719	0,00721336	0,00048965	0,00281309
11	0,0342663	0,00946477	0,0033552	0,00078436
12	0,00669428	0,01066005	0,00105445	0,00132411
13	0,00763825	0,00964868	0,00105174	0,00161777
14	0,00473022	0,01035497	0,00193598	0,00093359
15	0,00579626	0,00953629	0,00108393	0,00146799

Table 5.7: The 0.025-quantile of the predicted probabilities for each category for the first 15 students in the data set

The 0.975-quantile is given in table 5.8. The blue rows are again the rows that are incorrectly predicted. There are only three students amongst the first 15 in the data set for which the prediction was not accurate.

Student	Category 1	Category 2	Category 3	Category 4
1	0,59807704	0,83727451	0,46681917	0,95439893
2	0,57910255	0,8379709	0,59054836	0,95502238
3	0,5806534	0,84341946	0,51686596	0,95404906
4	0,37185497	0,88491734	0,28633675	0,98553344
5	0,62826617	0,81479043	0,65590373	0,94953513
6	0,56667267	0,82710835	0,37099454	0,9693191
7	0,44594667	0,87709947	0,33954178	0,97868818
8	0,5806534	0,84341946	0,51686596	0,95404906
9	0,5836333	0,83872124	0,40857821	0,96206286
10	0,49760781	0,83794503	0,31155193	0,97757336
11	0,75845152	0,76608082	0,53951427	0,90279226
12	0,55136968	0,86536783	0,43314187	0,96588072
13	0,54008835	0,85119026	0,4129339	0,96727116
14	0,46353612	0,88274125	0,6212479	0,96642961
15	0,4882453	0,8720537	0,44733368	0,9697377

Table 5.8: The 0.975-quantile of the predicted probabilities for each category for the first 15 students in the data set

These three given tables can be combined to construct the credible intervals. This is done for the rows where the highest probability does not correspond to the choice, which is only the case for three students in this case. For each category there is then an interval with a probability of 95% that the true value for this probability lies in this interval. This is visualized in figure 5.5.

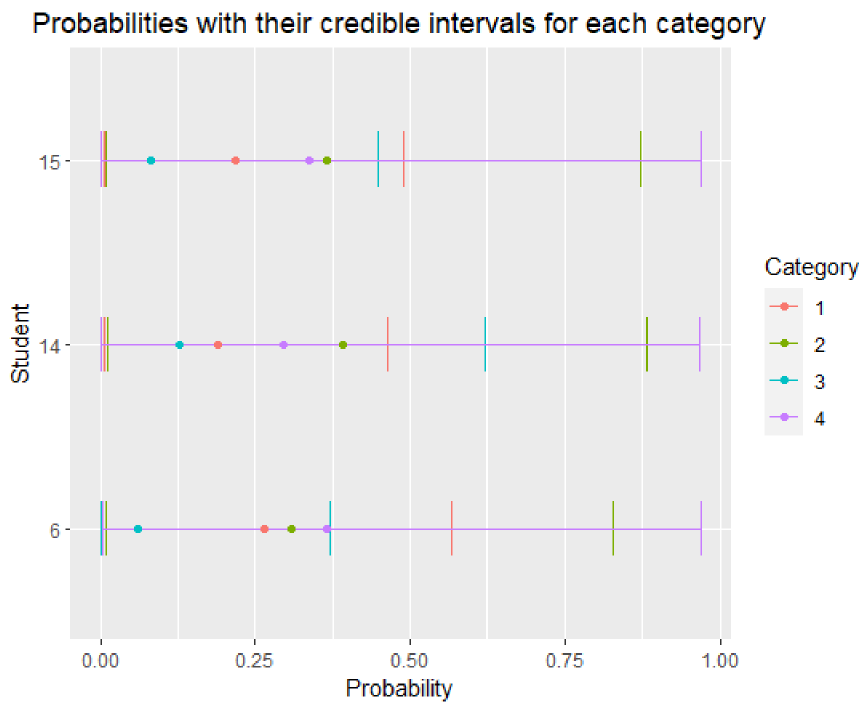


Figure 5.5: Credible intervals for those students amongst the first 15 students where the category with the highest probability does not correspond to the actual choice.

Figure 5.5 shows that for each category, the intervals are quite large and thus not very informative as the true value lies in a larger interval with a probability of 95%. There are some intervals that are smaller than others, so for these the intervals do say more. When considering student 6 for example, the interval for category 3 is significantly smaller compared to for example category 4. It is hence less likely that category 3 is chosen. Comparing figure 5.5 to figure 5.1, it is clear that the intervals were smaller and thus predicted the values better using the `UPG` package. However, comparing the accuracy from the `BRMS` package to the `UPG` package, it follows that the accuracy of the `BRMS` model slightly higher as it is equal to 0.4336918.

5.6 BRMS Model using K-fold cross-validation

The `BRMS` model from the previous section resulted in a higher accuracy compared to the `UPG` model. Thus, it is useful to apply K-folds cross validation to this model to verify the accuracy. Here, K-fold cross validation works in the same way as in section 5.4 and we again consider $K = 10$.

To apply 10-fold cross validation, for the `BRMS` package there is a function `kfold()` in R that takes in a `BRMS`-object and applies K-fold cross validation. In this function the number of folds can be specified. This takes approximately 4 hours in total to run for the 10 different iterations. After running the object returns the expected log pointwise predictive density, the effective number of parameters and the K-fold information criterion.

Consider the split into the K-folds where $r \in 1, \ldots, K$ is a subset which consists of 10% of the data as $K = 10$ in this case. Then $\log(p(y_i|y_{-r}))$ denotes the log predictive density for each observation $i \in r$, where r is in this case the validation set. This is then defined in the following way

$$\log(p(y_i|y_{-r})) = \log\left(\int p(y_i|\boldsymbol{\beta})p(\boldsymbol{\beta}|y_{-r})\mathrm{d}y_i\right)$$

The posterior distribution is determined by a number of samples $s \in 1, \ldots, S$ and the number of samples S is taken as 4000 in this package with $\boldsymbol{\beta}^{r,s}$. Here, r indicates the subset taken as validation and s is the concerned simulation. The posterior draws can now be denoted as $p(\boldsymbol{\beta}|y_{-r})$ and summarized over these s draws, the following expression is obtained for the $elpd_i$ (Vehtari, Gelman, and Gabry 2016)

$$\widehat{elpd}_i = \log \frac{1}{S}\sum_{s=1}^{S} p(y_i|\beta^{r,s}) \quad i \in r$$

To obtain the estimated value, this needs to be summed up such that for each i that corresponds to a subset r, the expression for the estimate yields

$$\widehat{elpd}_{kfold} = \sum_{i=1}^{n} \widehat{elpd}_i$$

This is because each observation i is at exactly once in some subset $r \in 1, \ldots, K$. Note that the function `kfold()` does not take a factor of $\frac{1}{n}$ into account when measuring the predictive accuracy (McElreath 2020). Now, this value of *elpd* can be interpreted in such a way that when the *elpd* is higher, or also less negative, the model performs better and thus has higher accuracy (Nicenboim, Schad, and Vasishth 2023).

Another value that is returned by `kfold()` is the *p_loo*, which indicates the effective number of parameters in the model. As explained in (McElreath 2020), this is a penalty term which is proportional to the variability present in the posterior distribution. This implies that when this penalty term is small the variance is also small and the other way around.

Finally, *looic* is the K-fold information criterion. This is -2 times the *elpd* and hence the estimate does not necessarily carry more information. The output that follows from applying the function `kfold()` is now summarized in table 5.9.

	Estimate	SE
elpd_loo	-369.7	7.5
p_loo	20.8	4.5
looic	739.4	14.9

Table 5.9: Results following from 10-fold cross-validation.

In the table the value the estimate for the *elpd* seems quite low. This gives some doubt about the predictive performance of the model. Also, the penalty term for the variance of the posterior is equal to 20.8, which is again quite high.

5.7 BRMS Model using Leave-One-Out Cross-Validation

To assess how well this model from section 5.6 performs, this K-fold cross validation can be compared to leave-one-out cross-validation using a function `loo()` in R. Leave-one-out cross validation is a version of K-fold cross-validation in which the number of folds is equal to the number of observations. In the data set there are n observations, so when using leave-one-out cross-validation the model is trained using the whole set while leaving out exactly one observation which is used for validation. This can then be seen as K-fold cross-validation, where $K = n$ and thus in this case there are n iterations. This function `loo()` now gives a similar output as $kfold()$ and can thus be compared to see which one works better. As the model is trained using more data points and only one observation is used for validation, we expect that the value for the average *elpd* is lower and the variance is high (Refaeilzadeh, Tang, and Liu 2009).

The output from the function `loo()` are similar to the results from section 5.6. There is a small difference in the definition of the expected log pointwise predictive density (*elpd*). In the case of K-folds cross validation, the subsets $r \in 1, \ldots, K$ are considered. However, when applying leave-one-out cross-validation, only one observation is left out and the rest is used for training the model. This implies that every subset r is of size exactly equal to 1. Incorporating this change into the definition of the log predictive density yields the folowing expression:

$$\log(p(y_i|y_{-i})) = \log\left(\int p(y_i|\boldsymbol{\beta})p(\boldsymbol{\beta}|y_{-i})\mathrm{d}y_i\right)$$

Now, this implies that each $elpd_i$ becomes

$$\widehat{elpd}_i = \log \frac{1}{S} \sum_{s=1}^{S} p(y_i|\beta^{i,s}) \quad i \in r$$

Finally, to find the estimate, all of these values need to be summed up such that:

$$\widehat{elpd}_{loo} = \sum_{i=1}^{n} \log(p(y_i|y_{-i}))$$

The other components from the output are interpreted in the same way as explained in section 5.6.

Table 5.10 summarized the results that follow from applying leave-one-out cross-validation to the model from section 5.5.

	Estimate	SE
elpd_loo	-984.2	15.4
p_loo	641.7	12.9
looic	1968.4	30.9

Table 5.10: Results after applying leave-one-out cross-validation.

As expected the estimate for *elpd* is significantly lower, so more negative, and thus less accurate and the variance is higher compared to the variance in table 5.9 for *elpd_loo*. It seems that the *elpd* is doing twice as bad compared to the model using K-fold cross validation. The effective number of parameters is quite high as it is equal to 641.7 and hence so is its' variance. This is therefore not very accurate. It is clear that the K-fold method works better in this sense.

5.8 BRMS using different priors

5.8.1 BRMS model using the Cauchy prior

According to, it is common to take a Cauchy prior with center 0 and scale 2.5 as an uninformative prior. An advantage of the `BRMS` package is that it allows you to specify a different prior by using the function `set_prior()`. By specifying the prior for the regression coefficients including the intercept, the model can be fitted again. The accuracy that follows from this model is 0.4229391 and the run time is approximately 74.533 seconds. This model thus does worse than the `UPG` model using the default prior and the `BRMS` model also does better with the default prior.

To assess this model leave-one-out cross validation can be applied again using the `loo()` function as the running time is low.

	Estimate	SE
elpd_loo	-345.4	9.3
p_loo	11.4	0.7
looic	691.1	18.7

Table 5.11: Results after applying leave-one-out cross-validation applied to `BRMS` model with Cauchy prior.

From table 5.11 it follows that the *elpd* is higher than the one where K-fold cross-validation was used in table 5.9. This means that the result is more accurate and the predictive performance is doing better than the overall performance after K-fold cross-validation. The penalty for the variance of the posterior distribution is almost half of the value in table 5.9, so using a different prior improved the model in that aspect too.

5.8.2 BRMS model using the Gaussian prior

To compare the performance of the different packages, the `BRMS` package can be used with a Gaussian prior with mean 0 and variance 4 for all regression coefficients including the intercept. The accuracy that follows from applying this is 0.4301075, which runs in 62.928 seconds. Here, the model performs slightly better than the `UPG` model using the same prior. However, the default prior in the `BRMS` package still does a better job in predicting accurately. The accuracy using a Gaussian prior is then still better than the prior that follows a Cauchy distribution.

	Estimate	SE
elpd_loo	-345.9	9.4
p_loo	11.9	0.7
looic	691.8	18.9

Table 5.12: Results after applying leave-one-out cross-validation applied to `BRMS` model with Gaussian prior.

The estimate of the *elpd* is slightly higher than the *elpd* that result from using a Cauchy prior. However, the difference is only 0.3 and the standard deviation of the *elpd* is also quite similar. Since the accuracy of this model is higher than the one from section 5.8.1, these results that follow from the leave-one-out cross-validation confirm that this model does slightly better. Hence, using the Gaussian prior as uninformative prior results in slightly better predictive accuracy compared to the Cauchy prior.

Chapter 6

Conclusion and discussion

In this chapter,the conclusion that follows from the results will be discussed and the model results will be briefly summarized. Finally, some limitations of this project and options for future research will be discussed.

6.1 Conclusion

Navigating back to the research questions of the project, the goal is to give a Bayesian interpretation of the results, where the Bayesian approach depends on a choice of a prior distribution. In the end, several different models were constructed and compared with respect to their predictive performance. Chapter 5 lists the results for each of the models used in this project. To draw a conclusion from all the comments made in chapter 5, it is important to briefly summarize the main point. This is done in table 6.1, where for each model the accuracy is listed together with the running time. The running time is important as it gives insight on whether the model is realistic to use. Two different packages, `UPG` and `BRMS`, are used. The package `BRMS` allows the user to specify different prior distribution, which is why the default prior distribution of this package is compared to the same package, but with different prior distributions such as the Gaussian distribution and the Cauchy distribution. The `UPG` package generates 10.000 posterior samples and the `BRMS` packages generates 4000 posterior samples.

	Accuracy	Run time
`UPG`, Uniformly generated data	0.22	$\approx$ 34.16 sec
`UPG`, PRIME data	0.4265233	$\approx$ 36.88 sec
`UPG`, Training and test	0.4285714	$\approx$ 31 sec
`UPG`, K-fold	0.4011677	$\approx$ 53 sec
`BRMS`, PRIME data	0.4336918	$\approx$ 410.121 sec
`BRMS`, Cauchy prior	0.4229391	$\approx$ 74.533 sec
`BRMS`, Gaussian prior	0.4301075	$\approx$ 62.928 sec

Table 6.1: Summary of the accuracy of each model and the running time.

From table 6.1 it follows that the model that has the best accuracy, as in accurately predicted the outcome, is the `BRMS` model applied to all data from PRIME. Considering the running time, however, it seems that this might not be the optimal solution in the case that the data set is larger as this would increase the running time further. The high running time for the `BRMS` package applied to all the PRIME data can be explained by the fact that the default considers different priors for the regression coefficients and intercept, which takes more computational effort.

Another model with a good prediction is the `BRMS` model using a Gaussian prior instead of the default prior. This Gaussian prior is the same as the default used in the `UPG` model, but it appears that the `BRMS` model yields a higher accuracy. The `BRMS` model using the Cauchy did have a higher value for the *elpd* and should thus give a better prediction. The difference between the *elpd*'s is still very small and thus it can be concluded that the Gaussian prior does better than the Cauchy prior taking all aspects of the models into account. The *elpd* resulting from the leave-one-out cross validation is also very low for the model with the default prior compared to the *elpd* resulting from the `BRMS` model with the Cauchy prior. These results are summarized in table 6.2

	elpd Estimate	elpd SD
BRMS, K-fold	-369.7	7.5
BRMS, loo	-984.2	15.4
BRMS, Cauchy prior loo	-345.4	9.3
BRMS, Gaussian prior loo	-345.9	9.4

Table 6.2: Summary of the *elpd* estimate of each model and the corresponding running time.

It can be concluded that the model applied to the generated data does the worst, which is not a surprise. The data is generated based on a uniform distribution and the prior is Gaussian. This results in a less good estimate for the posterior and thus the final predictions are not as accurate.

Constructing credible intervals can give an insight on how close the predictions were and whether there was some overlap between intervals. In the case that there was indeed overlap, it can be argued that the model was close to predicting correctly and not totally far off. However, in some cases the credible intervals are too large. As a result the intervals don't really provide more information and are no longer that useful. This is the case for the BRMS package applied to the whole data set. The credible intervals have a lot of overlap and are very widely spread over the interval (0,1), so there is not much information given here as this can be interpreted that the actual value is in this large interval with probability 0.95.

6.2 Future research and limitations

This project considers several different prior distributions to compare in terms of predictive performance. However, this project considers only uninformative and weakly informative priors. It could be useful to consider informative priors for future research. This could be done using the posterior distribution as defined in this project as the prior to fit on new data. This could give a better fit to the data and could result in higher accuracy of the results as the accuracy in this project still lies below 0.5. To further investigate the impact of the prior distribution on the accuracy it could be interesting to consider the uninformative prior called Jeffreys prior as this is a well-known uninformative prior.

Bibliography

Berman, Jules J. (2016). "Chapter 4 - Understanding Your Data". In: *Data Simplification*. Ed. by Jules J. Berman. Boston: Morgan Kaufmann, pp. 135–187. ISBN: 978-0-12-803781-2. DOI: https://doi.org/10.1016/B978-0-12-803781-2.00004-7. URL: https://www.sciencedirect.com/science/article/pii/B9780128037812000047.

Betancourt, M. J. and Mark Girolami (2013). *Hamiltonian Monte Carlo for Hierarchical Models*. arXiv: 1312.0906 [stat.ME].

Betancourt, Michael (Jan. 2017). *How the Shape of a Weakly Informative Prior Affects Inferences*.

Bijma, F., M. Jonker, and A. Van Der Vaart (May 2017). *An introduction to mathematical statistics*. Amsterdam University Press.

Bolstad, William M. and James M. Curran (Sept. 2016). *Introduction to Bayesian Statistics*. John Wiley & Sons.

Bürkner, Paul-Christian (2017). "brms: An R Package for Bayesian Multilevel Models Using Stan". In: *Journal of Statistical Software* 80.1, pp. 1–28. DOI: 10.18637/jss.v080.i01. URL: https://www.jstatsoft.org/index.php/jss/article/view/v080i01.

Clyde, Merlise (2022). *An introduction to Bayesian thinking*. URL: https://statswithr.github.io/book/index.html.

Etz, Alexander (Mar. 2018). "Introduction to the Concept of Likelihood and Its Applications". In: *Sage Journals* 1.1, pp. 60–69. DOI: 10.1177/2515245917744314. URL: https://doi.org/10.1177/2515245917744314.

Fox, John (Apr. 15, 2015). *Applied Regression Analysis and Generalized Linear Models*. SAGE Publications, Incorporated.

Gelman, Andrew (Sept. 2006). "Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper)". In: *Bayesian Analysis* 1.3. DOI: 10.1214/06-ba117a. URL: https://doi.org/10.1214/06-ba117a.

Grimmett, Geoffrey and Dominic Welsh (Jan. 2014). *Probability*. Oxford University Press.

McElreath, Richard (Mar. 13, 2020). *Statistical Rethinking. A Bayesian Course with Examples in R and STAN*. CRC Press.

Nicenboim, B, D Schad, and S Vasishth (Feb. 2023). *An Introduction to Bayesian Data Analysis for Cognitive Science*.

Price, Larry R. (2023). "Confirmatory factor analysis: foundations and extensions". In: *International Encyclopedia of Education (Fourth Edition)*. Ed. by Robert J Tierney, Fazal Rizvi, and Kadriye Ercikan. Fourth Edition. Oxford: Elsevier, pp. 607–618. ISBN: 978-0-12-818629-9. DOI: https://doi.org/10.1016/B978-0-12-818630-5.10016-8. URL: https://www.sciencedirect.com/science/article/pii/B9780128186305100168.

Refaeilzadeh, Payam, Lei Tang, and Huan Liu (2009). "Cross-Validation". In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Springer US, pp. 532–538. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_565. URL: https://doi.org/10.1007/978-0-387-39940-9_565.

Robert, Christian (Jan. 2007). "The Bayesian Choice: From Decision Theoretic Foundations to Computational Implementation". In.

Schriemer, S.H. (Jan. 2023). *A comparison of the frequentist and Bayesian approach to multinomial logistic regression in statistics: an application to study habits data from PRIME*.

Tabachnick, Barbara G. and Linda S. Fidell (Jan. 2013). *Using multivariate statistics*.

Van Ravenzwaaij, Don, Pete Cassey, and Scott D. Brown (Mar. 2016). "A simple introduction to Markov Chain Monte–Carlo sampling". In: *Psychon Bull Rev* 25.1, pp. 143–154. DOI: 10.3758/s13423-016-1015-8. URL: https://doi.org/10.3758/s13423-016-1015-8.

Vehtari, Aki, Andrew Gelman, and Jonah Gabry (Aug. 2016). "Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC". In: *Statistics and Computing* 27.5, pp. 1413–1432. DOI: 10.1007/s11222-016-9696-4. URL: https://doi.org/10.1007%2Fs11222-016-9696-4.

Zens, Gregor, Sylvia Frühwirth-Schnatter, and Helga Wagner (2022). *Efficient Bayesian Modeling of Binary and Categorical Data in R: The UPG Package*. arXiv: 2101.02506 [stat.CO].

— (2023). *Ultimate Pólya Gamma Samplers – Efficient MCMC for possibly imbalanced binary and categorical data*. arXiv: 2011.06898 [stat.CO].

Appendix A

Results, additional plots

A.1 UPG applied to PRIME data

Figure A.1 depicts the estimates of the posterior distribution as follows from applying the UPG model to all PRIME data. Together with the estimates also a credible interval is shown for each of the categories and for each of the predictors. The legend only indicates categories 2, 3 and 4 since category 1 was chosen as the baseline category. This implies that the posterior is set to 0 for category 1.

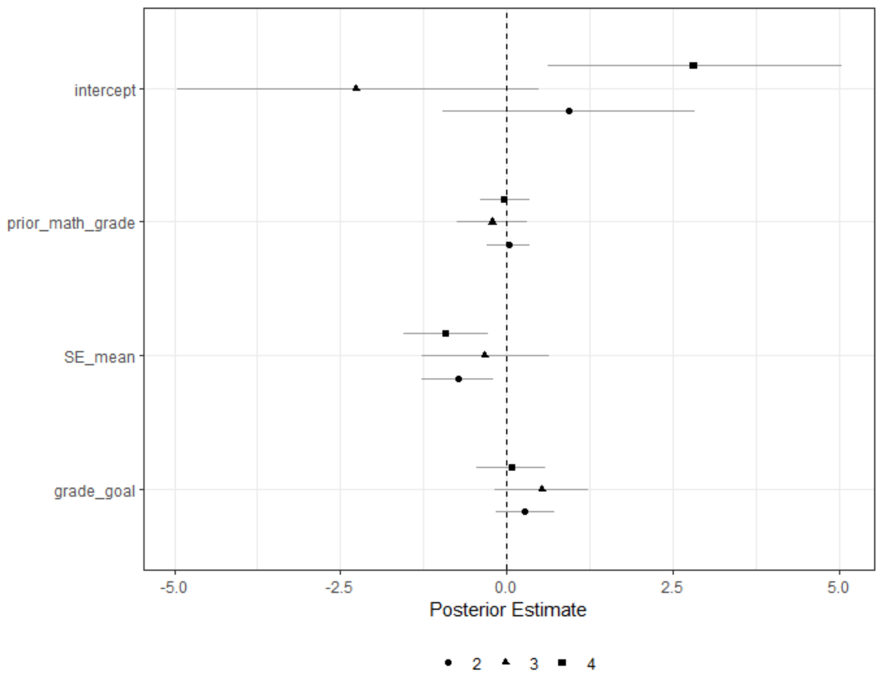


Figure A.1: Posterior estimates for the UPG Model applied to PRIME data.

A.2 UPG Model using Train-Test Split method

Again, such a plot with posterior estimates is constructed for the parameters found from fitting on the training set. The interpretation is the same as in section A.1. Comparing the estimates in figure A.2 to the estimates from figure A.1, it follows that the posterior estimates are very similar.

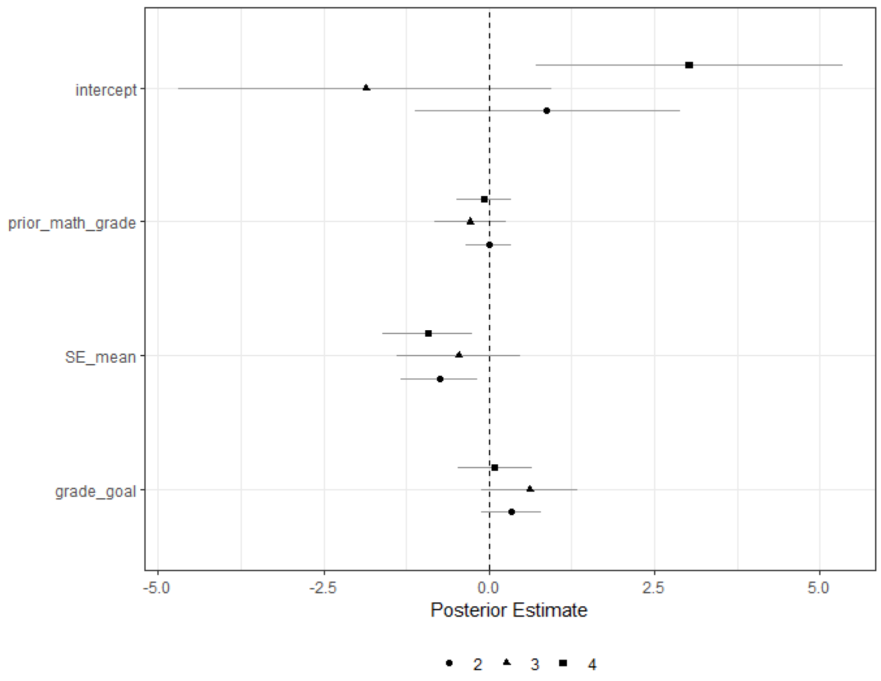


Figure A.2: Posterior estimates for the UPG Model using Train-Test Split method

A.3 UPG Model using K-fold cross-validation

For K-fold cross-validation, the posterior estimates of the best prediction can be compared to the other posterior estimates. This plot again is interpreted in the same way as in appendix A.1 and A.2.

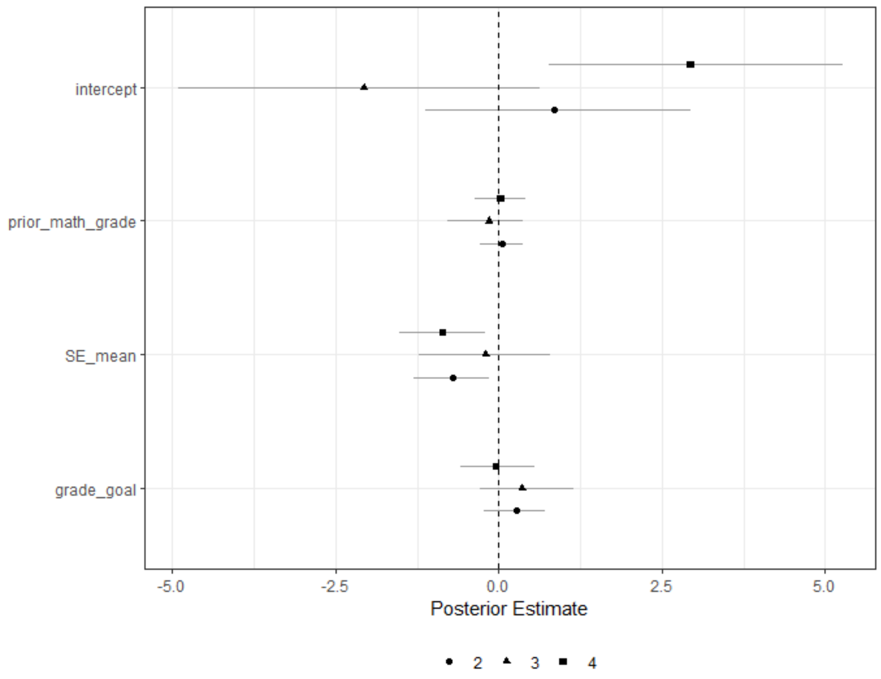


Figure A.3: Posterior estimates for the UPG Model using K-fold cross-validation.

From comparing figure A.3 to figures A.1 and A.2 it can be noted that here there are some slight differences in their estimates and credible intervals. This is interesting as this model has a higher accuracy compared to the other two models. The following differences can be noted:

1. The posterior estimate corresponding to the predictor prior math grade seems to be slightly closer to 0 for category 2 and category 3 is less negative.
2. For the predictor SE mean the corresponding posterior estimate of category 3 is closer to 0.
3. The posterior estimate for grade goal is negative instead of positive for category 2.

A.4 BRMS model using different priors

A.4.1 BRMS model using the Cauchy prior

Using the BRMS package with the Cauchy prior similarly as in section 5.5 the credible intervals can be constructed. In the same way, the first 15 students are considered. The credible intervals for the students that have a 0 in the column 'Correctly predicted' are constructed in figure A.4.

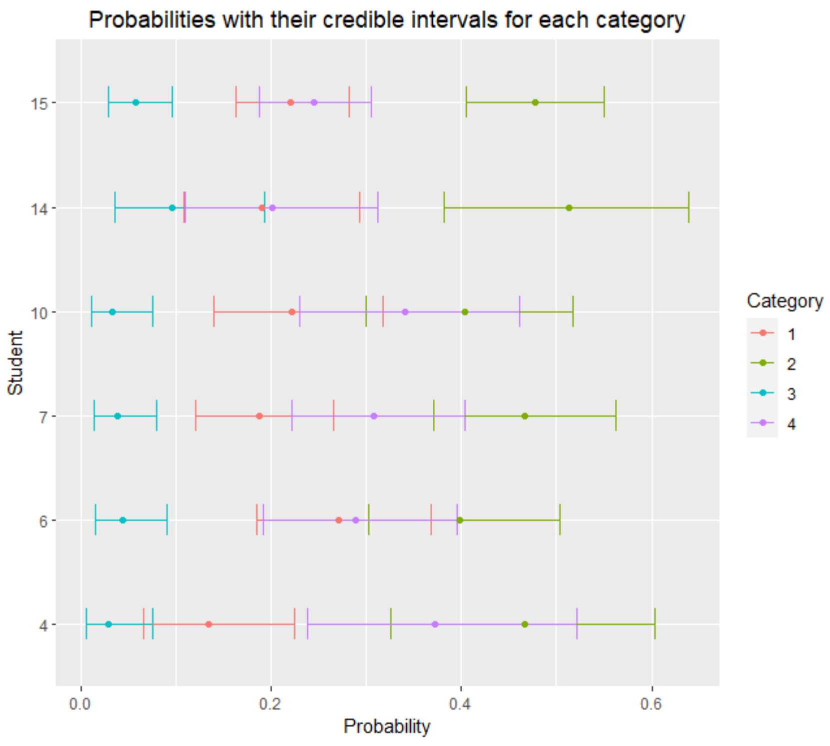


Figure A.4

These students appear to be the same as in section 5.2. Comparing these two it seems like the credible intervals are similar and there are very slight differences that are barely visible. Hence, these credible interval approximately carry the same information.

A.4.2 BRMS model using the Gaussian prior

Now, again credible intervals can be constructed. These are given in figure A.5.

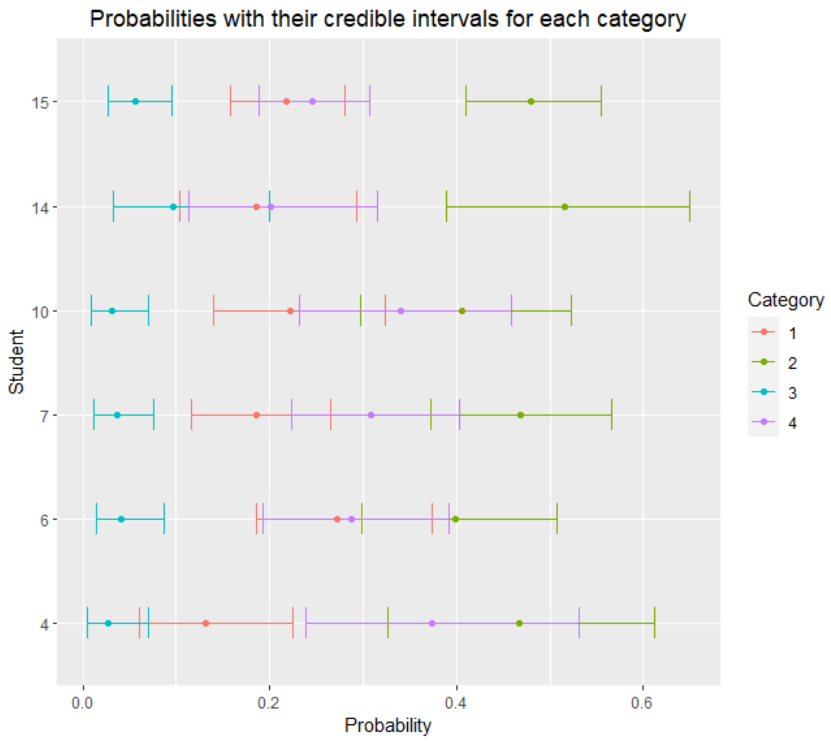


Figure A.5

Here, the students are again the same as in sections 5.2 and A.4.1 and the intervals are also similar.

Appendix B

Code for preprocessing and application to PRIME data

```r
#######PACKAGES########
library("readxl")
library("xlsx")
library("UPG")
library("ggplot2")
library("PerformanceAnalytics")
library("MASS")
library("geometry")
library("openxlsx")
library("data.table")
library("caret")
library("patchwork")


#######PREPROCESSING########
full_data <- read_xlsx("C:\\Users\\mahii\\OneDrive\\Documenten\\BEP\\data_BEP.xlsx", 1)
full_data <-data.frame(full_data)
#Removing the variables that are non-numerical and only keeping the variables of interest

df <- full_data[c(16, 45:49, 51:59, 61:63)]
grade_goal = round((df$grade_aim + df$grade_expected + df$grade_lowest_satisfied)/3, 2)
df <- cbind(df, grade_goal)

#Data cleaning
#Remove the rows with grade_goal == 0
df[is.na(df)] <- 0
colSums(df==0)
df <- df[df$grade_goal!=0, ]

#Checking the number of 0 in the dataframe
colSums(df == 0)
dim(df)


#grade_goal, self_efficacy, prior_math_grade and the response variables
predictors = c(1, 7, 19)
outcome_vars = c(8:15)

#Data set used is called data_bayes
Y = df[,outcome_vars]
intercept <- rep(1, nrow(df)) #vector of ones
data_bayes <- cbind(intercept, df[, predictors])
data_bayes <- data.matrix(data_bayes)

#Question chosen for this model: 'why_quiz'
q_no = 4
cat_no <- max(Y[,q_no])
m = as.character(cat_no)

#######HEATMAP########
heatmap_data <- as.data.frame(sapply(df[, c(1:7, 16:19)], as.numeric))
```

```r
cors <- round(cor(heatmap_data, method = "pearson" ), 2)
cors
heatmap(cors, Rowv = NA, Colv = NA, margins = c(12, 12))
melt_data <- melt(cors)
ggp <- ggplot(melt_data, aes(X1, X2)) +
  ggtitle("Heat map of correlations") +
  geom_tile(aes(fill = value)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        plot.title = element_text(hjust = 0.5)) +
  scale_fill_gradient2(low = "white", high = "darkblue")
ggp


#Scatterplots for the pairs that have a higher correlation than expected
p1 <- ggplot(heatmap_data, aes(x = SE_mean, y = grade_expected)) +
  geom_point(position = "jitter")
p2 <- ggplot(heatmap_data, aes(x = grade_expected, y = prior_math_grade)) +
  geom_point(position = "jitter")
p3 <- ggplot(heatmap_data, aes(x = grade_expected, y = SE4)) +
  geom_point(position = "jitter")
p4 <- ggplot(heatmap_data, aes(x = grade_goal, y = prior_math_grade)) +
  geom_point(position = "jitter")

plot_all_high <- (p1 + p2) / (p3 + p4) +
  plot_annotation(title = 'Scatterplots of variables with high correlation') &
  theme(plot.title = element_text(hjust = 0.5))

plot_all_high


#Scatterplots for the pairs that have a low correlation
p11 <- ggplot(heatmap_data, aes(x = grade_lowest_satisfied, y = SE5)) +
  geom_point(position = "jitter")
p22 <- ggplot(heatmap_data, aes(x = grade_aim, y = SE2)) +
  geom_point(position = "jitter")
p33 <- ggplot(heatmap_data, aes(x = prior_math_grade, y = SE5)) +
  geom_point(position = "jitter")
p44 <- ggplot(heatmap_data, aes(x = grade_lowest_satisfied, y = SE2)) +
  geom_point(position = "jitter")

plot_all_low <- (p11 + p22) / (p33 + p44) +
  plot_annotation(title = 'Scatterplots of variables with low correlation') &
  theme(plot.title = element_text(hjust = 0.5))

plot_all_low
```

Appendix C

Code for the UPG package

```
#######PACKAGES########
library("readxl")
library("xlsx")
library("UPG")
library("ggplot2")
library("PerformanceAnalytics")
library("MASS")
library("geometry")
library("openxlsx")
library("data.table")
library("caret")
library("patchwork")


#######PREPROCESSING########
full_data <- read_xlsx("C:\\Users\\mahii\\OneDrive\\Documenten\\BEP\\data_BEP.xlsx", 1)
full_data <-data.frame(full_data)
#Removing the variables that are non-numerical and only keeping the variables of interest

df <- full_data[c(16, 45:49, 51:59, 61:63)]
grade_goal = round((df$grade_aim + df$grade_expected + df$grade_lowest_satisfied)/3, 2)
df <- cbind(df, grade_goal)

#Data cleaning
#Remove the rows with grade_goal == 0
df[is.na(df)] <- 0
colSums(df==0)
df <- df[df$grade_goal!=0, ]

#Checking the number of 0 in the dataframe
colSums(df == 0)
dim(df)


#grade_goal, self_efficacy, prior_math_grade and the response variables
predictors = c(1, 7, 19)
outcome_vars = c(8:15)

#Data set used is called data_bayes
Y = df[,outcome_vars]
intercept <- rep(1, nrow(df)) #vector of ones
data_bayes <- cbind(intercept, df[, predictors])
data_bayes <- data.matrix(data_bayes)

#Question chosen for this model: 'why_quiz'
q_no = 4
cat_no <- max(Y[,q_no])
m = as.character(cat_no)

######HEATMAP########
heatmap_data <- as.data.frame(sapply(df[, c(1:7, 16:19)], as.numeric))
cors <- round(cor(heatmap_data, method = "pearson" ), 2)
cors
heatmap(cors, Rowv = NA, Colv = NA, margins = c(12, 12))
```

```r
melt_data <- melt(cors)
ggp <- ggplot(melt_data, aes(X1, X2)) +
  ggtitle("Heat map of correlations") +
  geom_tile(aes(fill = value)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        plot.title = element_text(hjust = 0.5)) +
  scale_fill_gradient2(low = "white", high = "darkblue")
ggp


#Scatterplots for the pairs that have a higher correlation than expected
p1 <- ggplot(heatmap_data, aes(x = SE_mean, y = grade_expected)) +
  geom_point(position = "jitter")
p2 <- ggplot(heatmap_data, aes(x = grade_expected, y = prior_math_grade)) +
  geom_point(position = "jitter")
p3 <- ggplot(heatmap_data, aes(x = grade_expected, y = SE4)) +
  geom_point(position = "jitter")
p4 <- ggplot(heatmap_data, aes(x = grade_goal, y = prior_math_grade)) +
  geom_point(position = "jitter")

plot_all_high <- (p1 + p2) / (p3 + p4) +
  plot_annotation(title = 'Scatterplots of variables with high correlation') &
  theme(plot.title = element_text(hjust = 0.5))

plot_all_high


#Scatterplots for the pairs that have a low correlation
p11 <- ggplot(heatmap_data, aes(x = grade_lowest_satisfied, y = SE5)) +
  geom_point(position = "jitter")
p22 <- ggplot(heatmap_data, aes(x = grade_aim, y = SE2)) +
  geom_point(position = "jitter")
p33 <- ggplot(heatmap_data, aes(x = prior_math_grade, y = SE5)) +
  geom_point(position = "jitter")
p44 <- ggplot(heatmap_data, aes(x = grade_lowest_satisfied, y = SE2)) +
  geom_point(position = "jitter")

plot_all_low <- (p11 + p22) / (p33 + p44) +
  plot_annotation(title = 'Scatterplots of variables with low correlation') &
  theme(plot.title = element_text(hjust = 0.5))

plot_all_low


#######GENERATE OWN UNIFORM DATA########
n=300
X_1 <- runif(n, min = 1, max = 10) #prior_math_grade
X_2 <- runif(n, min = 1, max =5) #SE_mean
X_3 <- runif(n, min =1, max =10) #grade_goal
intcpt <- rep(1, n)
gen_data <- cbind(intcpt, X_1, X_2, X_3)
y_gen <- sample(1:cat_no,n, TRUE) #why_quiz, there are 4 categories
gen.mnl = UPG(y = y_gen, X = gen_data, model = "mnl", baseline = "1", draws=10000, burnin
    =2000, A0=4, B0=4 )
summary(gen.mnl)
plot(gen.mnl)
predict(gen.mnl)
coef(gen.mnl)$`Posterior Mean`


#######UPG PACKAGE########
#Question  why_quiz
pred <- UPG(y=Y[, q_no], X=data_bayes, model='mnl', baseline="1", draws=10000, burnin=2000, A0
    =4, B0=4)
summary(pred)
posterior <- pred$posterior$beta

#Find the predictions and calculate the accuracy
predict(pred)
correct <- rep(0, nrow(predict(pred)$Q2.5))
low <- predict(pred)$Q2.5
low <- cbind(low[, cat_no], low[,1:(cat_no-1)])
high <- predict(pred)$Q97.5
high <- cbind(high[,cat_no], high[, 1:(cat_no-1)])
mean <- predict(pred)$`Posterior mean`
mean <- cbind(mean[,cat_no], mean[,1:(cat_no-1)], Y[,q_no], correct)

summary(pred)
plot(pred)
for (i in 1:nrow(mean)){
  loc <- mean[i,cat_no+1]
  if (max(mean[i,1:cat_no]) == mean[i, loc]){
    mean[i,ncol(mean)] <- 1
  }
}
accuracy <- sum(mean[,ncol(mean)])/(nrow(mean))
accuracy


#Plot posterior estimates
plot(pred)

#######CREDIBLE INTERVAL PLOT########
#Consider the first 15 students and reformat to get data frame cred_full
cred_ints <- as.data.frame(mean[1:15, ])
cred_ints$correct <- cred_ints[, cat_no+2]
cred_ints <- cred_ints[cred_ints$correct != 1 ,]
cred_mean <- data.frame(x = unlist(as.data.frame(t(cred_ints[,1:cat_no]))))

indices <- as.integer(rownames(cred_ints))
cred_high <- data.frame(x = unlist(as.data.frame(t(high[indices, ]))))
cred_low <- data.frame(x = unlist(as.data.frame(t(low[indices, ]))))
cred_high

indices_long <- rep(indices, each = cat_no)
cats <- as.character(rep(1:cat_no, times = nrow(cred_ints)))

cred_full <- data.frame(student = indices_long, Probability = cred_mean$x,
                        upper = cred_high$x, lower = cred_low$x, Category = cats)

#Using the reformatted data frame the credible interval plot can be found
ggplot(data = cred_full, aes(x = Probability, y = as.factor(student))) +
  geom_errorbarh(aes(xmin = lower, xmax = upper, color = Category), height = 0.3) +
  ggtitle("Probabilities with their credible intervals for each category") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_pointrange(aes(x = Probability, xmin = lower, xmax = upper, color = Category), size =
    0.3) +
  scale_y_discrete(name = "Student")


#######TRAINING AND TEST SET########
#Generate random numbers for the split into training and test set
set.seed(1)
samplee <- sample(1:nrow(df), 28)
samplee
X_training <- data_bayes[-samplee, ]
X_test <- data_bayes[samplee, ]
Y_training <- Y[-samplee,q_no]
Y_test <- Y[samplee, q_no]

#Train the model using the training set
train.mnl <- UPG(y = Y_training, X = X_training, baseline = "1",
                 model = "mnl", A0 = 4, B0 =4, draws= 10000, burnin = 2000)

#Save the regression coefficients
betas <- coef(train.mnl)$`Posterior Mean`
betas <- cbind(betas[,cat_no], betas[,1:(cat_no-1)])
b_x <-  X_test %*% betas

#Save posterior estimates for the 0.025 and 0.975-quantiles
beta_high <- cbind(coef(train.mnl)$`Q97.5`[,cat_no], coef(train.mnl)$`Q97.5`[, (1:cat_no-1)])
beta_high <- X_test %*% beta_high

beta_low <- cbind(coef(train.mnl)$`Q2.5`[,cat_no], coef(train.mnl)$`Q2.5`[, 1:(cat_no-1)])
beta_low <- X_test %*% beta_low

```

```r
#Calculate the probabilities for the test set using estimated posterior
pi = matrix(0, nrow = nrow(X_test), ncol = cat_no)
pi_high = matrix(0, nrow = nrow(X_test), ncol = cat_no)
pi_low = matrix(0, nrow = nrow(X_test), ncol = cat_no)

for (i in 1:nrow(X_test)){ #i is de student
  for (j in 2:cat_no){ #j is de categorie
    pi[i, j] <- exp(b_x[i,j]) /(1+sum(exp(b_x[i, 2:cat_no])))
    pi_high[i,j] <- exp(beta_high[i,j])/(1+sum(exp(beta_high[i,2:cat_no])))
    pi_low[i,j] <- exp(beta_low[i,j])/(1+sum(exp(beta_low[i,2:cat_no])))
  }
  pi[i, 1] <- 1-sum(pi[i,])
  pi_high[i, 1] <- 1-sum(pi_high[i, ])
  pi_low[i, 1] <- 1-sum(pi_low[i, ])
}
zeroes <- rep(0, nrow(pi))
pi_res <- cbind(pi, Y_test, zeroes)

#Calculate accuracy of the predicted test set
for (i in 1:nrow(pi_res)){
  y = as.numeric(pi_res[i,cat_no+1])
  if (max(pi_res[i, 1:cat_no]) == pi_res[i,y]){
    pi_res[i, cat_no+2] <- 1
  }
}
accuracy_tt <- sum(pi_res[,cat_no+2])/nrow(pi_res)
accuracy_tt

#Plot posterior estimates
plot(train.mnl)



#######CREDIBLE INTERVAL PLOT, TRAINING TEST########
#Consider the first 15 students and reformat to get data frame cred_full_tt
cred_ints_tt <- as.data.frame(pi_res[1:15, ])
cred_ints_tt$correct <- cred_ints_tt[, cat_no+2]
cred_ints_tt <- cred_ints_tt[cred_ints_tt$correct != 1 ,]
cred_mean_tt <- data.frame(x = unlist(as.data.frame(t(cred_ints_tt[,1:cat_no]))))

indices <- as.integer(rownames(cred_ints_tt))
cred_high_tt <- data.frame(x = unlist(as.data.frame(t(pi_high[indices, ]))))
cred_low_tt <- data.frame(x = unlist(as.data.frame(t(pi_low[indices, ]))))
#cred_high_tt

indices_long_tt <- rep(indices, each = cat_no)
cats_tt <- as.character(rep(1:cat_no, times = nrow(cred_ints_tt)))

cred_full_tt <- data.frame(student = indices_long_tt, Probability = cred_mean_tt$x,
                        upper = cred_high_tt$x, lower = cred_low_tt$x, Category = cats_tt)

#Using the reformatted data frame the credible interval plot can be found
ggplot(data = cred_full_tt, aes(x = Probability, y = as.factor(student))) +
  geom_errorbarh(aes(xmin = lower, xmax = upper, color = Category), height = 0.3) +
  ggtitle("Probabilities with their credible intervals for each category") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_pointrange(aes(x = Probability, xmin = lower, xmax = upper, color = Category), size =
    0.3) +
  scale_y_discrete(name = "Student")




#######K FOLD CROSS VALIDATION########
#Fix number of folds
number_of_folds <- 10
quality <- rep(0, number_of_folds)
val_data <- cbind(Y[,q_no], data_bayes)

#Split the data in 10 equal sets
folds <- createFolds(Y[,q_no], k = number_of_folds, list = TRUE)

#For each fold calculate the posterior estimates and validate using the test set
```

```
for (foldnr in 1:number_of_folds){
  training.Y <- val_data[-folds[[foldnr]], 1]
  training.X <- val_data[-folds[[foldnr]], 2:(1+cat_no)]
  test.X <- val_data[folds[[foldnr]], 2:(1+cat_no)]
  test.Y <- val_data[folds[[foldnr]], 1]
  highest_pi <- rep(0, nrow(test.X))
  pi <- matrix(0, ncol = cat_no, nrow = nrow(test.X))
  trainmod.mnl <- UPG(y = training.Y, X = training.X,
                      baseline = "1", model = "mnl", A0 =4, B0 =4)
  result_betas <- cbind(coef(trainmod.mnl)$`Posterior Mean`[,cat_no],
                        coef(trainmod.mnl)$`Posterior Mean`[,1:(cat_no-1)])
  beta_x <- test.X %*% result_betas
  for (student in 1:nrow(test.X)){
    denominator <- sum(exp(beta_x[student,]))
    for (category in 2:cat_no) {
      pi[student, category] <- exp(beta_x[student, category])/denominator
    }
    pi[student, 1] <- 1 - sum(pi[student, ])
    pi <- as.data.frame(pi)

    #The maximal probability is saved in highest_pi[student]
    highest_pi[student] <- as.numeric(substring(colnames(pi)[max.col(pi[student,], ties.method
     = "first")],2))

    #This is then compared to the actual choice
    if (highest_pi[student] == test.Y[student]){
      #Quality tracks the accurately predicted number of students
      quality[foldnr] = quality[foldnr] + 1
    }
  }
  #Finally divide this by the number of students in the test set to get the accuracy for the
    given fold
  quality[foldnr] <- quality[foldnr]/nrow(test.X)

  #Save the best model
  if (quality[foldnr] >= max(quality)){
    best_pred = pi
    result <- trainmod.mnl
  }
}

quality
best_pred
mean(quality)

#Plot posterior estimates of the model with highest accuracy
plot(result)

#######WRITING TO EXCEL FILE########
#Save some results in an excel file

data_excel <- list('all_data_mean' = mean, 'all_data_low' = low,
                   'all_data_high' = high, 'training_test' = pi_res, 'quality_kfolds' =
    quality, "highest_pi" = best_pred)
openxlsx::write.xlsx(data_excel, file = "Results_pi.xlsx")
```

Appendix D

Code for the BRMS package

```r
#######PACKAGES########
library("readxl")
library("xlsx")
library("UPG")
library("ggplot2")
library("PerformanceAnalytics")
library("MASS")
library("geometry")
library("openxlsx")
library("data.table")
library("rstan")
library("brms")
library("caret")
library("fGarch")
library("timeSeries")
library("loo")
library("future")

#######PREPROCESSING########
full_data <- read_xlsx("C:\\Users\\mahii\\OneDrive\\Documenten\\BEP\\data_BEP.xlsx", 1)
full_data <-data.frame(full_data)
#Removing the variables that are non-numerical and only keeping the variables of interest

df <- full_data[c(16, 45:49, 51:59, 61:63)]
grade_goal = round((df$grade_aim + df$grade_expected + df$grade_lowest_satisfied)/3, 2)
df <- cbind(df, grade_goal)

#Data cleaning
#Remove the rows with grade_goal == 0
df[is.na(df)] <- 0
colSums(df==0)
df <- df[df$grade_goal!=0, ]

#Checking the number of 0 in the dataframe
colSums(df == 0)
dim(df)


#grade_goal, self_efficacy, prior_math_grade and the response variables
predictors = c(1, 7, 19)
outcome_vars = c(8:15)

#Data set used is called data_bayes
Y = df[,outcome_vars]
intercept <- rep(1, nrow(df)) #vector of ones
data_bayes <- cbind(intercept, df[, predictors])
data_bayes <- data.matrix(data_bayes)

#Question chosen for this model: 'why_quiz'
q_no = 4
cat_no <- max(Y[,q_no])
m = as.character(cat_no)


```

```r

######BRMS DATA PREP########
##Restart R and then run install.packages(c("StanHeaders","rstan"),type="source") to reinstall
     packages from the source
y_m = matrix(0, nrow = nrow(data_bayes), ncol = cat_no)
for (i in 1:nrow(data_bayes)){
  ind <- as.numeric(Y[i,q_no])
  y_m[i, ind] <- 1
}

val_data <- data.frame(intercept = data_bayes[,1], x1 = data_bayes[,2],
                       x2 = data_bayes[,3], x3 = data_bayes[, 4])
val_data$size <- with(val_data, rowSums(y_m[,1:cat_no]))
val_data$y <- with(val_data, cbind(y_m[,1:cat_no]))



######BRMS MODEL WITH DEFAULT PRIOR########
#Train the model
model.brm <- brm(formula = y | trials(size) ~  intercept + x1 + x2 + x3,
             data = val_data, family = multinomial(), iter = 2000, cores = 4, chains = 4)

#Find predicted probabilities
brm.fit <- fitted(model.brm)
brm.fit

prior_summary(model.brm)
correct <- rep(0, length(brm.fit[,1,1]))

#Save predictions including the 0.025 and 0.975-quantiles for the plot
pred_default <- cbind(brm.fit[,1,1:cat_no], Y[,q_no], correct) #mean
pred_default_low <- brm.fit[,3, 1:cat_no]
pred_default_high <- brm.fit[,4,1:cat_no]

#Calculate the accuracy
for (i in 1:nrow(pred_default)){
  loc <- pred_default[i,cat_no+1]
  if (max(pred_default[i,1:cat_no]) == pred_default[i, loc]){
    pred_default[i,cat_no+2] <- 1
  }
}
accuracy_default <- sum(pred_default[,(cat_no+2)])/(nrow(pred_default))
accuracy_default

#############CREDIBLE INTERVALS PLOT#############
#Consider the first 15 students and reformat to get data frame cred_full
cred <- as.data.frame(pred_default[1:15, ])
cred$correct <- cred[, cat_no+2]

#Save the ones that are not correctly predicted
cred <- cred[cred$correct != 1 ,]
cred_mean <- data.frame(x = unlist(as.data.frame(t(cred[,1:cat_no]))))

indices <- as.integer(rownames(cred))
cred_high <- data.frame(x = unlist(as.data.frame(t(pred_default_high[indices, ]))))
cred_low <- data.frame(x = unlist(as.data.frame(t(pred_default_low[indices, ]))))
cred_high

indices_long <- rep(indices, each = cat_no)
cats <- as.character(rep(1:cat_no, times = nrow(cred)))

cred_full <- data.frame(student = indices_long, Probability = cred_mean$x,
                        upper = cred_high$x, lower = cred_low$x, Category = cats)

#Using the reformatted data frame the credible interval plot can be found
ggplot(data = cred_full, aes(x = Probability, y = as.factor(student))) +
  geom_errorbarh(aes(xmin = lower, xmax = upper, color = Category), height = 0.3) +
  ggtitle("Probabilities with their credible intervals for each category") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_pointrange(aes(x = Probability, xmin = lower, xmax = upper, color = Category), size =
    0.3) +
  scale_y_discrete(name = "Student")

```

```


#############K FOLDS CROSS VALIDATION USING KFOLD()############
#K folds cross validation using paralellization
plan(multicore, workers = 4)
kfold_res <- kfold(model.brm, K = 10, save_fits = TRUE, chains = 4, cores = 4)
kfold_res


############LEAVE ONE OUT CROSS VALIDATION############
#leave one out cross validation which is K-fold cross-validation with K=n
loo(model.brm)


############CAUCHY PRIOR############
cauchy_prior <- c(prior("cauchy(0, 2.5)", class = "b"), prior("cauchy(0,2.5)", class = "
    Intercept"))
cauchy.mod <- brm(formula = y | trials(size) ~ intercept + x1 + x2 + x3,
                  family = multinomial(), iter = 2000, data = val_data, prior = cauchy_prior,
                  chains = 4, cores = 4)

#Find predicted probabilities
cauchy.fit <- fitted(cauchy.mod)
correct <- rep(0, length(cauchy.fit[,1,1]))

pred_cauchy <- cbind(cauchy.fit[,1,1:cat_no], Y[,q_no], correct)


for (i in 1:nrow(pred_cauchy)){
  loc <- pred_cauchy[i,cat_no+1]
  if (max(pred_cauchy[i,1:cat_no]) == pred_cauchy[i, loc]){
    pred_cauchy[i,(cat_no+2)] <- 1
  }
}
accuracy_cauchy <- sum(pred_cauchy[,cat_no+2])/(nrow(pred_cauchy))
accuracy_cauchy
accuracy_default

#leave one out cross validation cauchy model
loo(cauchy.mod)



############CREDIBLE INTERVALS PLOT CAUCHY############
#Consider the first 15 students and reformat to get data frame cred_full_c
credc <- as.data.frame(pred_cauchy[1:15, ])
credc$correct <- credc[, cat_no+2]

#Save the ones that are not correctly predicted
credc <- credc[credc$correct != 1 ,]
credc_mean <- data.frame(x = unlist(as.data.frame(t(credc[,1:cat_no]))))

#Save predictions of the 0.025 and 0.975-quantiles for the plot
predc_default_high <- cauchy.fit[, 4, 1:cat_no]
predc_default_low <- cauchy.fit[, 3, 1:cat_no]
indices_c <- as.integer(rownames(credc))
credc_high <- data.frame(x = unlist(as.data.frame(t(predc_default_high[indices_c, ]))))
credc_low <- data.frame(x = unlist(as.data.frame(t(predc_default_low[indices_c, ]))))
credc_high

indices_long_c <- rep(indices_c, each = cat_no)
cats_c <- as.character(rep(1:cat_no, times = nrow(credc)))

cred_full_c <- data.frame(student = indices_long_c, Probability = credc_mean$x,
                          upper = credc_high$x, lower = credc_low$x, Category = cats_c)

#Using the reformatted data frame the credible interval plot can be found
ggplot(data = cred_full_c, aes(x = Probability, y = as.factor(student))) +
  geom_errorbarh(aes(xmin = lower, xmax = upper, color = Category), height = 0.3) +
  ggtitle("Probabilities with their credible intervals for each category") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_pointrange(aes(x = Probability, xmin = lower, xmax = upper, color = Category), size =
    0.3) +
```

```r
  scale_y_discrete(name = "Student")




#############GAUSSIAN PRIOR#############
normal_prior <- c(prior("normal(0,4)", class = "b"), prior("normal(0,4)", class = "Intercept")
    )
mod.norm <- brm(formula = y | trials(size) ~ intercept + x1 + x2 + x3,
                 family = multinomial(), iter = 2000, data = val_data, prior = normal_prior,
                chains = 4, cores = 4)

#Find predicted probabilities
normal.fit <- fitted(mod.norm)

correct <- rep(0, length(normal.fit[,1,1]))
pred_normal <- cbind(normal.fit[,1,1], normal.fit[,1,2], normal.fit[,1,3], normal.fit[,1,4], Y
    [,4], correct)

#Calculate the accuracy
for (i in 1:nrow(pred_normal)){
  loc <- pred_normal[i,cat_no+1]
  if (max(pred_normal[i,1:cat_no]) == pred_normal[i, loc]){
    pred_normal[i,cat_no+2] <- 1
  }
}
accuracy_normal <- sum(pred_normal[,cat_no+2])/(nrow(pred_normal))
accuracy_normal
accuracy_cauchy
accuracy_default

#leave one out cross validation gaussian model
loo(mod.norm)


#############CREDIBLE INTERVALS PLOT GAUSSIAN############
#Consider the first 15 students and reformat to get data frame cred_full_g
credg <- as.data.frame(pred_normal[1:15, ])
credg$correct <- credg[, cat_no+2]

#Save the ones that are not correctly predicted
credg <- credg[credg$correct != 1 ,]
credg_mean <- data.frame(x = unlist(as.data.frame(t(credg[,1:cat_no]))))

#Save predictions of the 0.025 and 0.975-quantiles for the plot
predg_default_high <- normal.fit[, 4, 1:cat_no]
predg_default_low <- normal.fit[, 3, 1:cat_no]
indices_g <- as.integer(rownames(credg))
credg_high <- data.frame(x = unlist(as.data.frame(t(predg_default_high[indices_g, ]))))
credg_low <- data.frame(x = unlist(as.data.frame(t(predg_default_low[indices_g, ]))))
credg_high

indices_long_g <- rep(indices_g, each = cat_no)
cats_g <- as.character(rep(1:cat_no, times = nrow(credg)))

cred_full_g <- data.frame(student = indices_long_g, Probability = credg_mean$x,
                          upper = credg_high$x, lower = credg_low$x, Category = cats_g)

#Using the reformatted data frame the credible interval plot can be found
ggplot(data = cred_full_g, aes(x = Probability, y = as.factor(student))) +
  geom_errorbarh(aes(xmin = lower, xmax = upper, color = Category), height = 0.3) +
  ggtitle("Probabilities with their credible intervals for each category") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_pointrange(aes(x = Probability, xmin = lower, xmax = upper, color = Category), size =
    0.3) +
  scale_y_discrete(name = "Student")




######WRITING TO EXCEL FILE########
#Save some results in an excel file
data_excel <- list("default_mean" = pred_default, "default_low" = pred_default_low,
                   "default_high" = pred_default_high)
openxlsx::write.xlsx(data_excel, file = "Results_brms.xlsx")



############Plotting used priors###########

x_axis <- seq(-8, 8, length = 200)
plot(x_axis, dnorm(x_axis, mean = 0, sd = 2), type = "l", col = "black", ylim = c(0, 0.4),
     main = "Different priors", xlab = "x", ylab = "Probability density")
lines(x_axis, dcauchy(x_axis, location = 0, scale = 2.5), col = "blue")
lines(x_axis, dt(x_axis, df= 3), col = "red")
legend("topright", c("Normal", "Cauchy", "Student t, df=3"), col = c("black", "blue", "red"),
    lty = c(1, 1, 1))
```