Balance in test results

Ways higher education students reach test scores within Utrecht University

by

Wiebe de Vries
5713595
w.r.vries@students.uu.nl

Fourteenth of March 2023

A document submitted in partial fulfillment of the requirements for the degree of
Master of Science in Artificial Intelligence
at
Utrecht University
Supervised by dr. Matthieu Brinkhuis, Remko Nolten M.Sc. , dr. Sergey Sosnovsky

Abstract

In this thesis, we have analysed multiple-choice and text questions within faculties using deviation profiles. Three methods were used to calculate these deviation profiles, a naive percentage-based, an IRT-based approach and a proportional IRT-based approach. The results of these analyses were contradictory. Further research is needed to examine whether examinees score better on multiple-choice or text questions. We also analysed differential item functioning for two groups based on their preferred language. The result of this analysis was that within some faculties, there seems to be DIF present within tests. Further research is needed to verify whether this is true for the entire test set. We also discuss future works based on this thesis and the test set used.

Acknowledgements

I like to take a moment to acknowledge some people that have supported me while completing this thesis. Firstly, I would like to thank my supervisor Matthieu Brinkhuis, for his support during the long time it took me to finish this thesis. The feedback was always constructive, and motivation was kept high because of our interesting discussions. Secondly, I would like to thank my supervisor at Paragin, Remko Nolten, for making it possible to do such a cool project at his place of work, Paragin. I would also like to thank the wonderful people at Paragin for the internship opportunity.

Contents

1 Introduction

During the last few years, digital testing has seen a steady increase in usage within the education system (Roode, 2020). This means that the possible effects of digital testing on a student also increase. As a result, studying digital testing has become ever more interesting. A number of phenomena, such as response times, are now easier to obtain. Therefore, new avenues of study have opened up, such as comparing these response times. Testing has several types available for questions, such as multiple choice, open, text, match and order (Vromant, 2022). In digital testing, questions also have a type attached to them; in the future, we will refer to this phenomenon as strict typing. As a result of strict typing, it is possible to research these strict types. Not only do these types of research become available, but they also become scalable, enabling us to perform studies on item wide scale.
One such type is balance is given as the distribution of the achieved score over the different parts of tests. So for a maths test, we can think in terms of subdomains, such as geometry and algebra. The balance between these two domains can be given as the ratio of the achieved score achieved between categories (Haberman & Sinharay, 2009). The focus of this study, however, is to apply the methods learned from balance research to different attributes of questions. This is interesting because, in contrast to categorisation, typing is strict. Because balance analysis is only as strong as the basis for the split, strict typing might yield a clear split.
When calculating balance, we can go about it in several ways, some being item model-based as shown in section 2.5 and some person model based as shown in section 2.5.7. A third option, a percentage-based approach, calculates an expected percentage score for each part of the test for each total score. When the observed percentage differs from this expected percentage score, an imbalance occurs (Verhelst, 2017). The model-based approach works by assigning a difficulty score to each question, which can then be used for a more fair assessment of how many points should be granted for that question. This approach uses a so-called item response model, which is based on item response theory. Using these difficulties, we can then calculate the balance conditional to the difficulty. A secondary aim of this study will be to find examinee groups that get a different balance on questions. A grouping factor is, for instance, the field of study. In the following section, the research questions are laid out.

1.1 Research questions

The aim of this study is to find out how students achieve certain test scores. Do certain types of questions yield more points than others? What are the factors that influence the performance of students on certain questions? The main research question of this study is, therefore:

1. *What can we learn from sub-scores based on item types in higher education high stake testing?*

In order to answer this question, several sub-questions have been established:

2. *What types of sub-scores can we identify?*

Using the types as presented above, what types of questions are actually used? What sub-scores can we construct out of these types?

3. *What methods are available in the literature for sub-score analysis?*

We will view several sub-score analysis methods, both within sub-scores and between sub-scores.

4. *What can we learn from different test types?*

Test types refers to either practice exams, practice or high stake exams.

5. *What can we learn from different item types?*

Several item types are available in digital tests. Text questions and multiple choice are the most common types used.

6. *What can we learn from different examinee groups?*

Different examinee groups can refer to what faculty students belong to, their preferred language or their gender.

2 Literature Review

This section will present the theory that is related to the research questions. Firstly we will start by introducing how to create what is known as sub-scores, which is the notion of picking fractions of a test. Firstly we will cover sub-scores and how to make them. After this, we will cover balance, which is a way of viewing sub-scores for specific groups. There are several approaches to calculating balance. There are several approaches to calculating balance, one of which is item response modelling, which is why we will discuss it next. Another way of approaching balance is not by starting at items but at the persons completing the items. When items perform differently, this is called differential item functioning. Lastly, we will cover differential item functioning (DIF) between different participant groups.

2.1 sub-scores

Tests in higher education are often divided into sections based on categories. When scores are assigned to parts of tests, they are called sub-scores. These sub-scores hold value in two ways (Haberman, 2005). Firstly, they give the examinee insight into its performance on different test parts. Secondly, they give the institution a view of its performance in different sections. Feedback on staff or study curriculum can be constructed using these sub-scores (Haladyna & Kramer, 2004). A few factors have to be considered to justify using these sub-scores to influence decisions. Haberman (2005) argues in favour of reporting sub-scores only if the sub-score can be shown to have a more accurate representation of the measured construct than the total score. A sub-score is only useful when it gives more information on the ability of, for example, a sub-domain. An example would be a maths test with algebra and geometry questions. The total score would indicate the mathematical ability of the examinee. The sub-scores for algebra and geometry can give more fine-grained insight into the algebraic or geometric ability.

The validity of taking a sub-score of a test is studied by Haberman and Sinharay (2009). In their paper, Haberman and Sinharay (2009) outline two possible uses for taking sub-scores—that of the test subject and that of the institution behind the test. Similar analyses were applied to both cases. The examinee-level analysis was done following Haberman (2005). A similar approach was used to determine the validity of usage for institutions. These analyses were applied to two types of teacher's aide's tests. The result of the analysis was that reporting examinee means on sub-scores on a test-wide level does not appear to be justified for any realistic institution size. Which is in contrast to mean total scores, which can have value for the institution, starting from around 30 participants per test.

2.2 Balance

One way of studying identified sub-scores is through the concept of balance. Balance is the distribution of an examinee's score over the different parts of the test. So if we take the mathematical test example from above, the ratio of points gained on the geometry part is a way of depicting balance. Balance is interesting because it gives information about performance between subsets of exam items (Verhelst, 2017).
A way of calculating balance would be, adding up all points per sub-score and dividing them over the total achievable score. Because distributions of questions over sub-scores are not always equal, using this approach might result in a skewed balance. A way of counteracting this would be to weigh the sub-scores over their part of the total score, thus mitigating the unequal size problem.
In addition, some items are more difficult to answer correctly than others (Thompson, 2022c). What happens if one sub-score yields significantly more difficult items?
A response to this question has been given as item response theory (Thompson, 2022c). The idea is that when we assign a difficulty score to a question and an ability score to a respondent, we can predict the respondents chance of a correct response. Models that follow this general principle are called item response models. In their paper, Verhelst (2017) includes such models and illustrates a way of using profile analysis to calculate the balance between domain-specific questions and multiple-choice and constructed response.

2.2.1 Profile analysis

Profiles

To achieve a score, many routes can be taken. Consider, for example, a test with 7 questions worth 1 point. There is only one way to achieve a maximum score, by correctly answering 7 questions. When we calculate ways to achieve a score of 1, we see that already 7 options are available. For the scores in the middle of the spectrum, many different routes lead to the same score. When the number of questions increases, so does the number of routes one can take to achieve a specific score. Each way one can come to a score is called a profile (Verhelst, 2012).

An example of profile analysis is given as follows. A test is given for which all questions belong to one of two mutually exclusive categories. Furthermore, the maximum score is given as the number of questions on the test. We can then compute a score for each given category. When an examinee completes our test the result is called an observed profile, given by a sequence of corrects and incorrects. The sequence that we would expect when we look at the average sequence for the score our examinee got is called the expected profile. The difference between these two profiles is called the deviation profile. When we sum the deviation profile, we get 0, because the sum of the expected profile and the observed profile is the same (Verhelst, 2012).
The traditional approach of profile analysis focuses on the observed profiles. The aim is often to use these observed profiles versus the expected profiles to depict meaningful information between two categories of test (Verhelst, 2012). This leads to a focus on performance on sub-scores (Verhelst, 2017). As discussed in sub-score section 2.1, this is often not worth reporting on the institutional level (Haberman & Sinharay, 2009). Verhelst (2017) instead focuses on the deviation profiles and its meaning within the measurement model. The deviation profile is used to depict the balance between categories, larger deviations illustrate skewed balance and vice versa. When

deviation profiles are aggregated one can compute averages over respondents, even when differing items were answered. For example, when a giant test bank is used, we can compute a balance for the whole bank. This will only hold as far as the categorisation holds. A division between multiple-choice (MC) and constructed response (CR) is a clearly defined categorisation. We need the model parameters to analyse these profiles, that is, the differing parameters constructed from the item response models. We will go into these parameters in the next section. To know these parameters, we need a test set large enough to where we can substitute the estimates for the real parameters because the standard error is considered negligible; for an example, see Verhelst (2017). Once we have these deviation profiles, we must compute expected scores. To rule out these deviation profiles being a product of mere chance, we also need a way of evaluating the likelihood of the deviation profiles occurring (Verhelst, 2017). The central limit theorem is used (Kwak & Kim, 2017) for this purpose.

Expected score calculation

Verhelst (2017) are calculating the deviation profile for the TIMSS 2011 data (Martin et al., 2012). The TIMSS 2011 data is a mixture of binary and polytomous data. Verhelst (2017)'s technical overview highlights the journey from a classical Rasch model to a model suitable for both binary and polytomous data (Rasch, 1993). This paper will touch upon some of these models in the IRT part of this literature review. For now, it is interesting to see that Verhelst (2017) mixes several types of item response models to create a model suitable to the data type. They use this model to calculate the expected response (or score) for each score achieved. These are then subtracted from the observed score to yield an average deviation profile for the groups analysed.

2.2.2 Balance and item types

To use balance analyses for comparing performance on different parts of tests, we first need to decide how to split tests. Which can be done by splitting tests based on the characteristics of items. Examples of these types of splits are multiple-choice or constructed response (open-ended) and little or lots of time spent. This is why Verhelst (2017) is carefully examined; they touch upon this subject with their MC and CR comparison.

2.3 Mean-based sub-score comparisons

When sub-scores have been identified, we can use them in several ways. One is researching whether these sub-scores hold more insights than the total score (Haberman, 2005; Leading & Monaghan, 2006). Another is to compare the sub-scores with each other (de la Torre et al., 2011; Puhan et al., 2008). While both routes offer interesting insights, we are interested in the balance aspect. Because we have access to a set of items in which the questions are strongly typed. Which in turn creates very strictly cut sub-scores. Thus we will, in this section, examine the comparison of sub-scores and not their added insights.

Firstly we will look at several methods that have been identified to compare sub-scores. These methods can be sorted into two camps, the classical test theory (CTT) camp and the item response theory (IRT) camp. For the CTT camp we examined Puhan et al. (2008), they argue that propor-

tional reduction of mean squared errors is a good approach for interpreting sub-score differences and comparing them with the total score.

The paper by de la Torre et al. (2011) is our starting point for IRT-based approaches. They have identified four IRT sub-scoring methods: multidimensional scoring, augmented scoring, higher-order item response model scoring, and objective performance index scoring. They found that the first three gave similar results, while the latter resulted in poorer results on both ability estimates as proportion correct scores. In section 2.5, we will examine several of these IRT techniques.

2.4 Classical test theory

Classical test theory (CTT) and item response theory (IRT) are two different approaches to analysing test data see T. Bechger et al. (2003) for a combined approach. Classical test theory is widely used to evaluate tests within most education systems. A test is made up of questions, and each question yields a score. These scores are then combined to achieve a test score. This is a straightforward way of calculating a mark for a test. A few problems can be established immediately. Samples change: when a test is taken the results are only applicable within that sample (Verhelst, 2004). This also means that any statistics done on this set will not automatically be applicable to a second set of the same test. Another problem is the lack of inherent guessing handling (Verhelst, 2004). For multiple-choice tests, this leads to skewed results. And lastly, not all questions are of equal difficulty. Resulting in a skewed reward for achieving correct on any question. A solution for some of CTT's problems is given in the form of item response theory (Thompson, 2022c).

2.5 Item response models

Item response theory starts with the notion that each item has a difficulty parameter. Depending on the type of model used, multiple parameters can be incorporated. In the section below, we will start by looking at a simple form of an IRT model, the Rasch model (Rasch, 1993). After discussing this model, we will gradually move on to more elaborate item response models.

2.5.1 Rasch model

$$P_i\left(u_{ij}=1\middle|\theta_j,b_i\right)=\frac{\exp(\theta_j-b_i)}{1+\exp(\theta_j-b_i)} \tag{2.1}$$

Equation 2.1: Equation of the dichotomous Rasch model (Rasch, 1993)

The Rasch model is an early model in modern item response theory. Its core idea is: placing items and persons on the same scale. It has also been used to construct new ways of looking at tests, such as test information, conditional standard error and maximum likelihood estimating. In this section we will take a brief look at the makeup of the Rasch model.
The Rasch model is given as the equation 2.1 for a single test item; an example function is depicted in Figure 2.1. This exponential function denotes the probability (P_i) of a correct response ($U_{ij}=1$) for a single item (i). It uses the trait level(θ) for the participant (j) and the difficulty

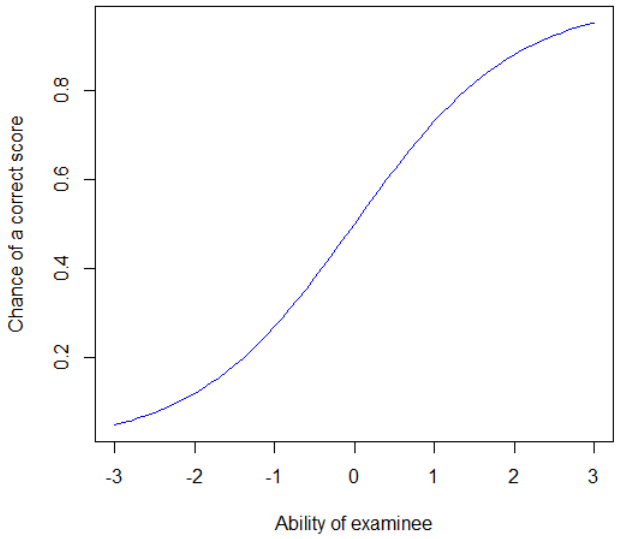

Figure 2.1: Example item response function for a single question

of an item (b_i) to calculate this. Note that the difficulty of an item thus influences where on the x-axis the sigmoid function will intersect; it moves the graph right when the difficulty increases. In case $(\theta_j - b_i)$ equals 0, a participant's trait level is the same as the difficulty of an item. In this situation $exp(\theta_j - b_i)$ results in 1 leading to $P_i = 0.5$. When a person's latent trait is higher than the difficulty, the resulting probability is greater than 0.5 and vice versa. This idea is central to the Rasch model and subsequent extensions (Weiss, 2022).
The model assumes difficulty as the sole factor influencing the probability of achieving a correct score for a given ability. This is in contrast to other IRT models, which include extensions like discrimination parameters or guessing score parameters. A positive side-effect of this is simplicity. A participant can be scored based on their number correct or observed score without considering which questions they got right exactly, their observed profile. An observed profile is a profile of what points were achieved on what questions. This also means that the number of possible responses is always the number of test items $+1$, which uses less information about the test than is available (B. Wright & Tennant, 1996). Thus the following models discussed extend upon the Rasch model to include the abovementioned problems.

2.5.2 TWO-PARAMETER LOGISTIC MODEL

$$P_i(X_i = 1|\,\theta_j) = \frac{\exp[a_i(\theta_j - b_i)]}{1 + \exp[a_i(\theta_j - b_i)]} \tag{2.2}$$

Equation 2.2: Equation of the two-parameter logistic model

The two-parameter logistic model, as given in equation 2.2 and the three-parameter logistic model, are extensions of the Rasch model. The extension for the two-parameter model is given in the form of the a_i parameter. This parameter is the discrimination parameter. It determines how well the item discriminates between high and low latent train levels. It does so by affecting the slope of the item response function, with a steeper slope being more discriminating and vice versa. An example of a test that could use the two-parameter logistic model would be a personality test because its only goal is placing a person on a scale for a personality trait. Thus using a discrimination parameter to discern between high and low trait discriminating questions.
The Rasch-type models are able to deal with multiple-choice and binary questions. Many educational tests assign scores to items based on performance. Therefore we will need a model that can handle the input of a score for an item, which should be able to be polytomous. We will discuss some polytomous IRT models below.

2.5.3 Three-parameter logistic model

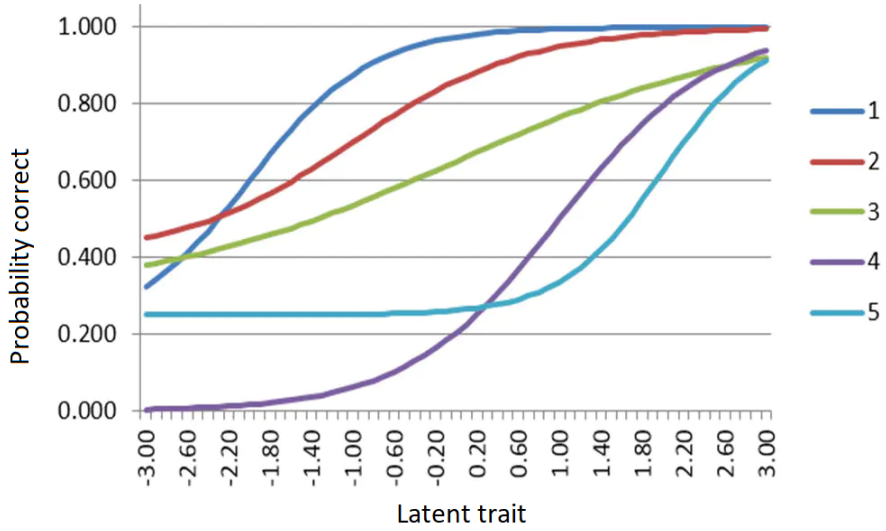

Figure 2.2: Example three-parameter logistic model for five items with different parameter values (Thompson, 2022c)

$$P_i(X_i = 1|\,\theta_j) = c_i + (1 - c_i)\frac{\exp[a_i(\theta_j - b_i)]}{1 + \exp[a_i(\theta_j - b_i)]} \tag{2.3}$$

Equation 2.3: Equation of the three-parameter logistic model

In formula 2.3, we see an extension to the Rasch, and two-parameter model introduced earlier, called the three-parameter logistic model (Birnbaum, 1968). The extra parameter is the c_i. With a_i still being the discrimination parameter determining how well the item discriminates between high and low latent trait levels. We have now seen the two factors influencing the slope and the

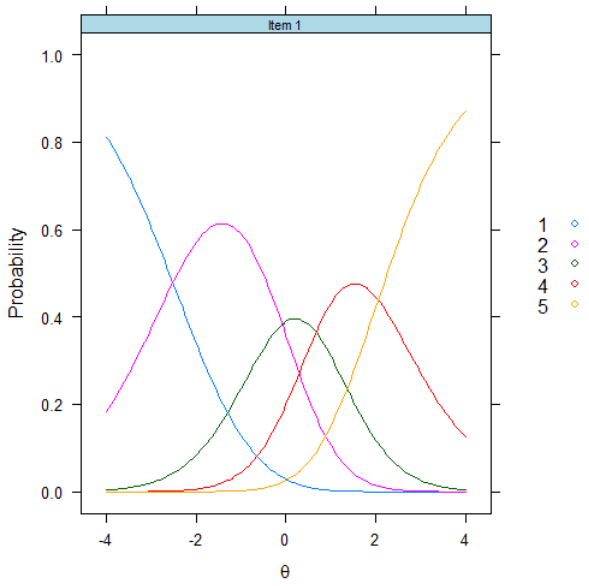

Figure 2.3: Example generalised partial credit model for a single item

place on the x-axis of this function. This is demonstrated by comparing items 2 (red) and 3 (green) in Figure 2.2. Blue has a steeper slope, thus it discriminates more.
The guessing parameter given as c_i influences the horizontal asymptote. The guessing parameter influences where the function starts for the lowest latent trait possible. Let's take a multiple-choice item with five options. When guessing randomly, a participant answering this item will have a 20% chance of guessing the correct item. Therefore, the guessing parameter aims to take this minimal expected reward into account (Loe, 2021). In Figure 2.2, each curve denotes an item. Most items seem to have some guessing correction because the asymptote is assumably not at probability 0.000. Item 4, the purple item, does not have a guessing parameter defined.

2.5.4 Generalised partial credit model

$$P_{ix} = \frac{\exp[\sum_{j=0}^{x} a_i(\theta - g_{ij})]}{\sum_{r=0}^{m} \exp[\sum_{j=0}^{r} a_i(\theta - g_{ij})]} \tag{2.4}$$

Equation 2.4: Equation of the generalised partial credit model

In equation 2.4, the generalised partial credit model is given. This model is used to fit polytomous items. Polytomous items are items on which a score can be achieved between correct and incorrect. We will briefly examine the equation below.

- m is the max score
- x is the student's score on the item
- i is the index for a given item
- r is an index used by the summation
- a_i is the discrimination parameter
- θ is the latent trait parameter or ability
- g_{ij} is the boundary parameter used for step j on item i
- r is used by the summation

The GPCM can be used to evaluate questions that can yield multiple scores. It is mostly used within educational testing alongside 3pl or 2pl model. The GPCM model can then be used on the polytomous questions, while the 2pl or 3pl model can be used on the dichotomous questions. a_i and θ are back and have the same effect as earlier models. However, note that the discrimination parameter a_i is a unique item discriminator instead of a unitary one. This means that this discrimination parameter only applies to this item and can not be used to scale the discrimination with other items (Hao, 2022). The difference, however, is in the summations. To grasp how the g_{ij} parameter works, we will look at an item to which GPCM can be applied. We have chosen an item for which 4 points can be achieved. This means there are five possible scores, 0,1,2,3 and 4. If we plot these scores as a function of theta, we could get something like plot 2.3, in which option 1 is a score of 0 and option 5 is a score of 4. In Figure 2.3 we see five curves which denote the chance of achieving that score for a given θ. There are a few intersections within this graph; these are the boundary parameters g_{ij}, where the likelihood for a score given a latent trait shifts from one score to another. In equation 2.4, they are given as g_{ij}, the boundary parameter for step j on item i. For a GPCM function to be constructed, a graph like figure 2.3 is a visualisation. There are always $m - 1$ boundaries (Loe, 2021; Thompson, 2022b; Weeks, 2010).

2.5.5 Extended nominal response model

The extended nominal response model is an extension to the original nominal response model as proposed by Darrell Bock (1972). It was extended by G. Maris et al. (2015) and specifically tailored to suit the Dexter R package by Koops et al. (2020). The original Nominal response model is used to model the situation in which a question has unordered responses, so there is no best or worst answer. An example of such a question would be: "What is your favourite colour?" The answers to the question are of categorical nature.
The nominal response model was extended to also encompass the marginal Rasch and GPC model by G. Maris et al. (2015). This was done by taking the desirable traits from the nominal response model and combining them with characteristics from the marginal Rasch and GPC model. The marginal Rasch model used was taken from Cressie and Holland (1983). G. Maris et al. (2015) use a Gibbs sampler to estimate the extended nominal response model (ENORM) (Geman & Geman, 1984).

Koops et al. (2020) extends upon this approach by ensuring that their Markov Chain - Monte Carlo method is suited for large-scale calibrations of test administrations. The version that they use is implemented in the Dexter R-package under the name fit_enorm (Maris & Bechger, 2023).

2.5.6 Minimum sample sizes

The models described above are well-defined and yield good approximations of parameters in theory (Lord, 2012). In practice, a minimal sample size of participants is recommended for each of them to satisfy statistical significance (Lord, 2012). In this section, we will take a look at the minimum sample sizes needed to fit the models above.
With regard to sample sizes used, opinions differ. Some sources suggest a minimum of 150 (Sahin & Anil, 2017) or 200 (Wright & Stone, 1979) participants for the Rasch model. Other sources recommend at least 400 candidates (Harris & Crouse, 1993; Kolen & Brennan, 2004). O'Neill et al. (2020), however, found that while less precision and item instability occur when smaller sample sizes are used, this has minimal effect on person ability estimates for the Rasch model. For the 2pl and 3pl models, estimates vary from 250 to 1000 participants. The GPCM is a more complex model, and thus most estimates vary between 500 and 1000 participants (O'Neill et al., 2020). Concerning the number of items within a test, Linacre (1994) and B. Wright and Tennant (1996) state that the sample size should be at least 30 for the GPCM. These sample sizes mean that, in practice, these IRT models are hard to apply in the context of exams in higher education.

2.5.7 Differential item functioning

In this section, we will briefly address differential item functioning (DIF) due to its relation to the balance type analysis we aim to do in this paper (Roju et al., 1995).

Differential item functioning is a technique to assess whether an item is biased against a certain group of examinees. Often the groups are predetermined, such as gender or race. A straightforward approach would be, do members of this group perform worse on this item? When the answer is yes, the item is biased (Sehmitt & Dorans, 1990).
A frequently used DIF analysis technique is the Mantel-Haenszel analysis (Dorans & Holland, 1992). This technique uses the raw number correct score to depict latent trait. This is then used to depict the percent correct per ability for both the regular group as the (maybe) biased against group. See figure 2.4 for an example. When we see differences between the lines that form, we can see the differing performance of both groups and whether there is DIF present (Thompson, 2022a).
However, problems arise when there are ability or trait differences between the biased against group and the regulars. These problems being, that comparison based on raw performance does not yield a fair comparison. Both groups tend to perform differently overall. This is why we need a separate technique to determine bias. We need to analyse an examinee's performance conditional to their ability. T. Bechger and Maris (2015) argue that difficulty should not be based on observing correct scores on individual items. Because Lord (2012) has proven that relative item difficulties are equal across groups, T. Bechger and Maris (2015) argue that using relative difficulties yields more accurate comparisons and thus a more accurate DIF analysis. Their test is closely related to

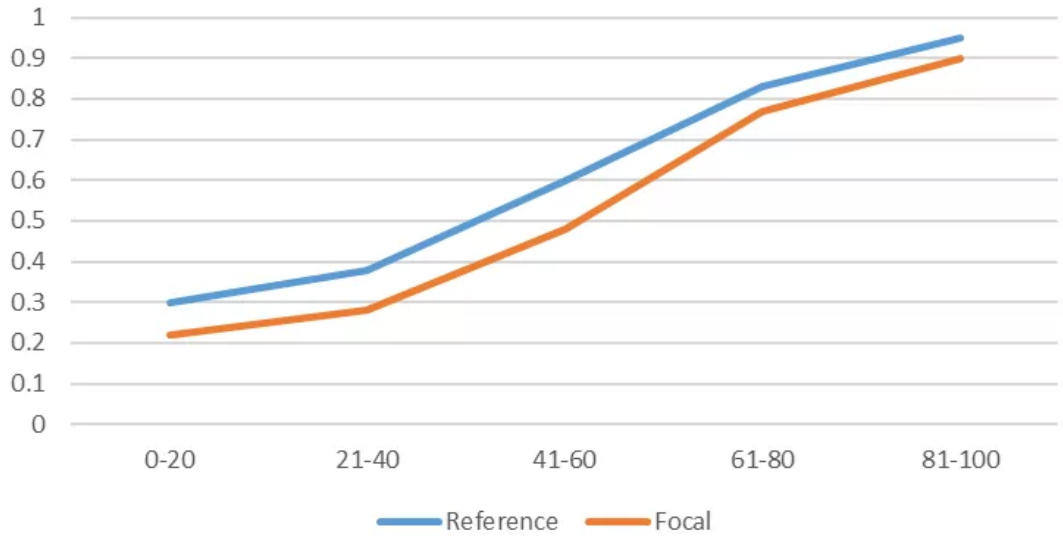

Figure 2.4: Example Mantel-Haenszel analysis as given by Thompson (2022a), the y-axis is the probability correct for an item that presumably has DIF, while the x-axis is the total score achieved, a substitute for ability

Lord (2012), with a different interpretation. For a visualisation, consult T. Bechger (2022). T. Bechger (2022) DIF test was implemented in the R-package Maris and Bechger, 2023.

DIF is a measurement of differences in item performance. It measures whether items perform differently in differing groups. It does so while also incorporating the skill of a group into the mix. So DIF's relation to balance is that it's an inverse perspective. Balance is a person-to-item type relationship, whereas DIF is an item-to-person type relation. This means that the outcome of both types of analysis can be pretty similar, even though the way of measurement is very different.

3 Data Exploration

In this section, the data as examined will be examined from differing angles. All plots in this section have been created using H. Wickham (2016). For a full overview of how the plots were created view the R code in appendix 1.3.

3.1 Source

Utrecht University uses a system provided by Paragin (2021). The system is called RemindoToets and is used by various educational institutes in the Netherlands. Students use this system to fill out high-stakes tests. Their results are then saved to a central database that Paragin is maintaining, but is the property of Utrecht University. It holds the entirety of the digital testing data that the university has gathered. The university has used these means of digital testing since 2015. Our export was taken on the 10th of November, 2022. This means that we have roughly 7 years of data at our disposal. The database was stored in a MariaDB database structure. The total size of the database is 56 GB. Below you will find a summary of the data available. Importing the database into R was done using the DBI, and RMySQL packages (M. Wickham, 2022).

3.1.1 RemindoToets

The tests that are taken within this environment have a lot of variances. Tests can be of a certain length. Tests can have a random order of questions. Tests can be split into sections, which can then have a random order of questions within them. The questions themselves can have a random order of answering options, so one candidate might get ABC as options, whereas another candidate sees BAC. There are a lot of different question types possible within RemindoToets. Within a question, questions of multiple types can be asked. For example, an inline choice followed by a multiple-choice can be found in the database within one question. See 3.1 for an example of such a question. The same question can be asked in several ways. For example, the multiple-choice option. There is also the inline choice option; when a single inline choice is used, the same effect as with a multiple-choice option can be achieved. See figures 3.2 and 3.3 for an example of the same question in differing formats [1].

[1] Occurrences like this mean that navigating this database requires intimate knowledge of the contents. Luckily the RemindoToets team has been very helpful.

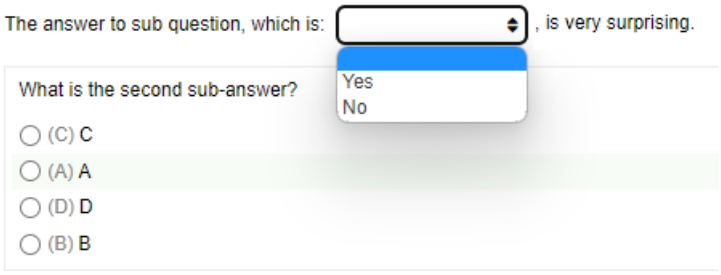

Figure 3.1: Multiple question elements

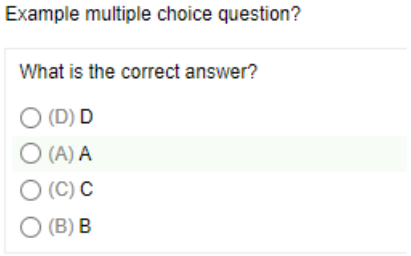

Figure 3.2: Multiple-choice question example

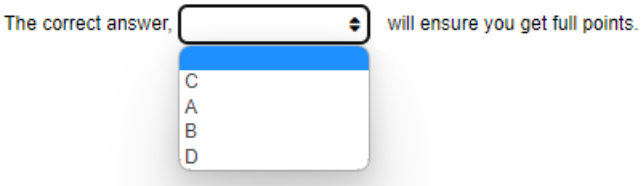

Figure 3.3: Inline question example

Feature name	Recipe specific	Question specific	Answer specific
Recipe ID	X		
Question sequence		X	
Question ID		X	
Section ID		X	
Guess score		X	
Maximum score		X	
Weight		X	
NeedsManualCheck		X	
QTI		X	
Metadata		X	
Questiontype		X	
Answer ID			X
Result ID			X
User ID			X
Date/time			X
Duration			X
Score			X
Fullscore			X
Dont know			X

Table 3.1: Domain specification per feature

3.2 Contents

3.2.1 Features

The data has been transformed into a row-wise format. Each given answer has several features. These features are: Question sequence, Recipe ID, Answer ID, Result ID, Question ID, Section ID, User ID, Date/time stamp last change, Duration, Completion status, Guess score, Score, Maximum score, Fullscore, Weight, NeedsManualCheck, DontKnow, QTI XML, Metadata and Questiontype. In table 3.1, we see the specific domain a feature is a part of. For a full explanation of each feature check the appendix in section 1.1.

3.2.2 Feature statistics

In table 3.2, we see the numeric values from the list above laid out in a summary table. The year and month is an extract from the date/time stamp. Furthermore, we see some outliers in the Guess score and Duration columns. Tests were taken over a period of roughly 7 years.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max	NAs
Duration	0.00	27.00	58.00	151.90	126.00	1205550.00	4771
Guess score	0.00	0.25	0.25	0.34	0.25	44.25	0
Score	0.00	0.00	1.00	1.33	1.00	200.00	0
Maximum score	0.00	1.00	1.00	2.09	1.00	200.00	0
fullscore (y/n)	0.00	0.00	1.00	0.62	1.00	1.00	0
Weight	0.00	1.00	1.00	1.08	1.00	40.00	0
Totalscore	0.01	25.00	40.00	55.95	54.00	1015.00	0
Year	2015	2019	2020	2020	2021	2022	0
Month	1.00	3.00	6.00	6.83	10.00	12.00	0

Table 3.2: Summary of some interesting numerical values from our set

3.2.3 Type distribution

In these 7 years, 16.5 million questions have been answered in over 9000 tests. These questions belong to one of 12 types. See figure 3.4 for the distribution of the questions over these types. Questions were filtered to only include the types of questions that we want, namely multiple-choice, extended text and inline choice. Inline choice and multiple-choice were put together under the term: choice. Answers were only used if they did not have multiple question elements within a single question. After this filtering, 12.6 million questions still remained, which is 79% of the original data. These questions belonged to either choice or extended text. See figure 3.5 for the distribution over these two categories.

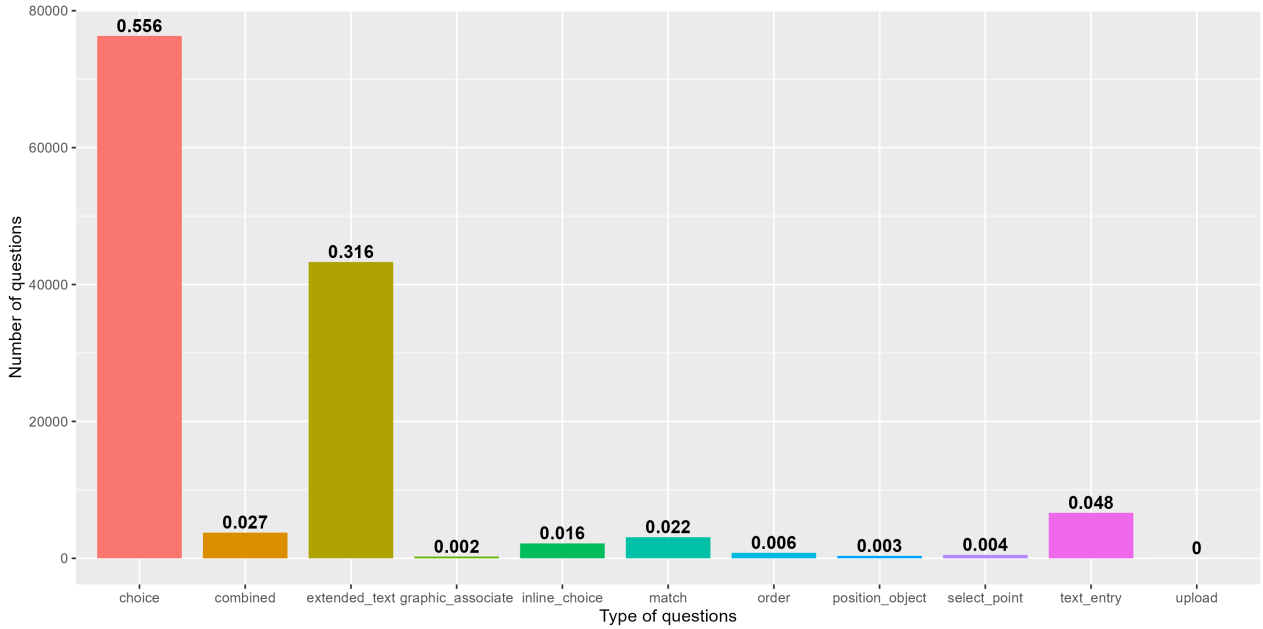

Figure 3.4: Type distribution of all questions - see 3.2.3.

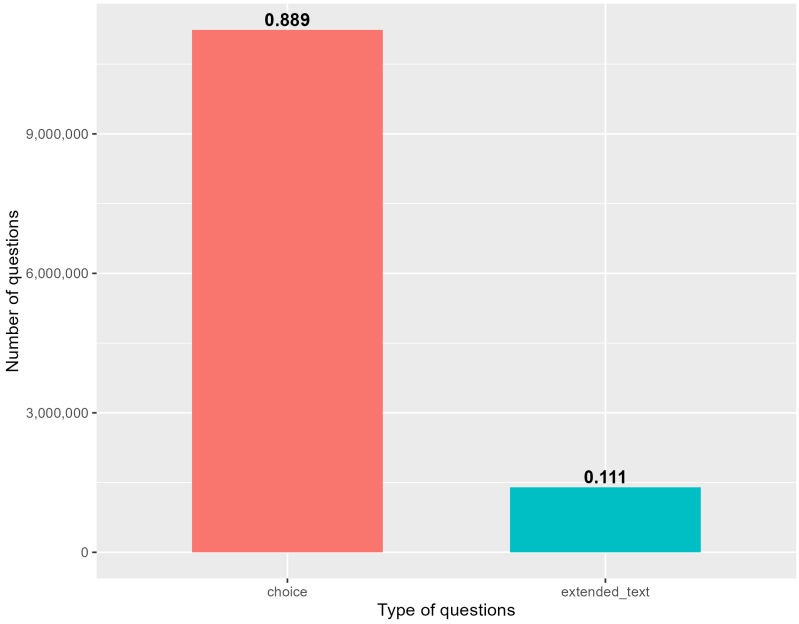

Figure 3.5: Distribution between multiple-choice and extended text, with eligible inline choice incorporated into multiple-choice - see 3.2.3.

3.2.4 Stats per year and month

In figure 3.6, the number of tests per year is laid out. The drop in 2022 is due to our export being taken in November, which means our data for 2022 is incomplete. In figure 3.7, the distribution over the months of the year can be seen. There is a clear dip within the summer months. In figure 3.8, we see the mean normalised scores achieved on questions in each month. Normalisation was done based on the maximum score possible for each question. In figure 3.9, we see the mean normalised scores achieved on questions in each year.

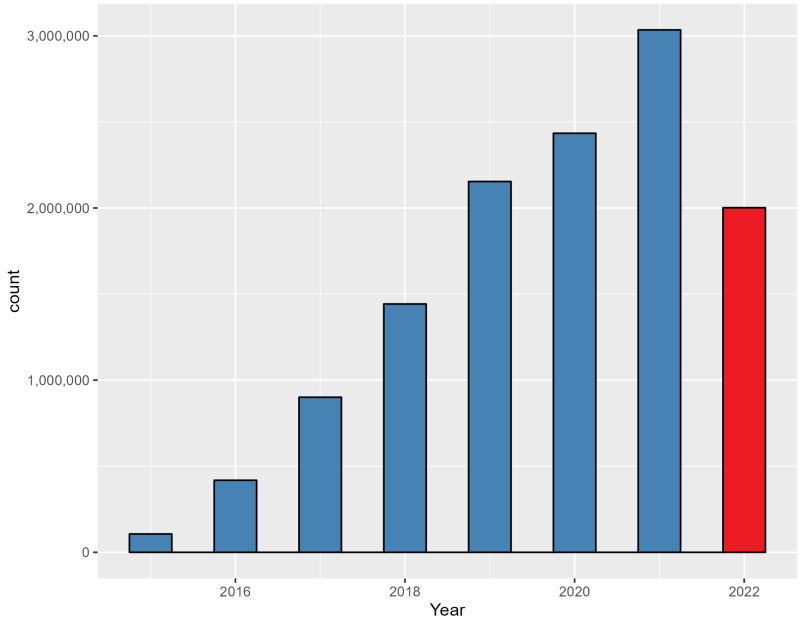

Figure 3.6: Distribution of tests over the years - see 3.2.4.

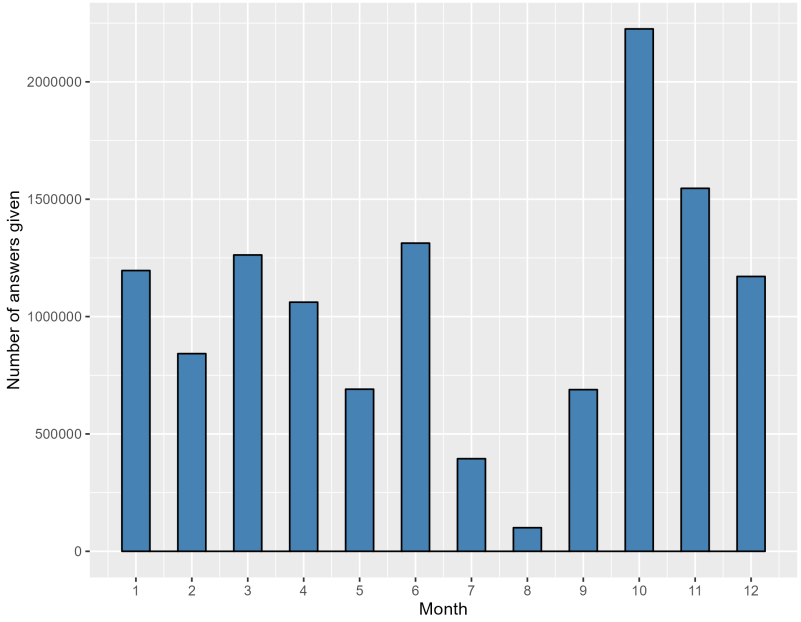

Figure 3.7: Distribution of tests over the months - see 3.2.4.

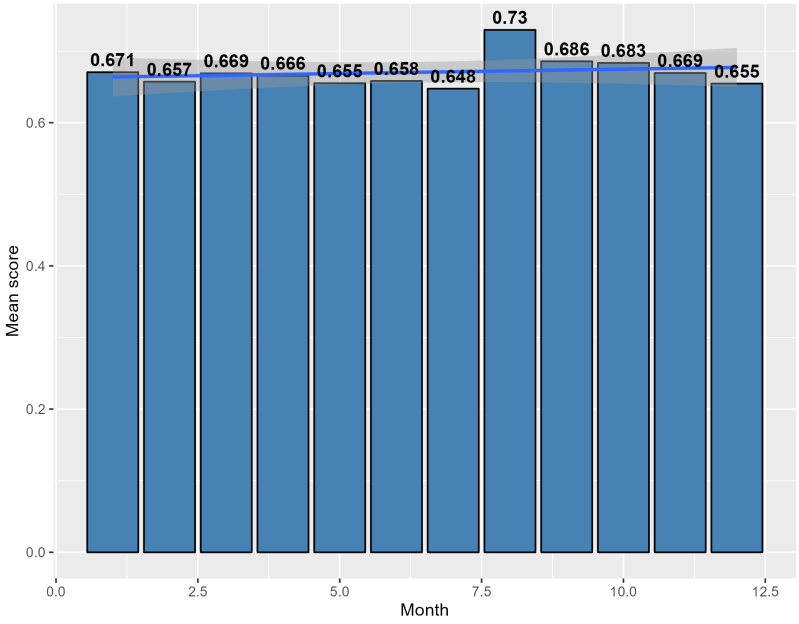

Figure 3.8: Mean normalised score per month - see 3.2.4.

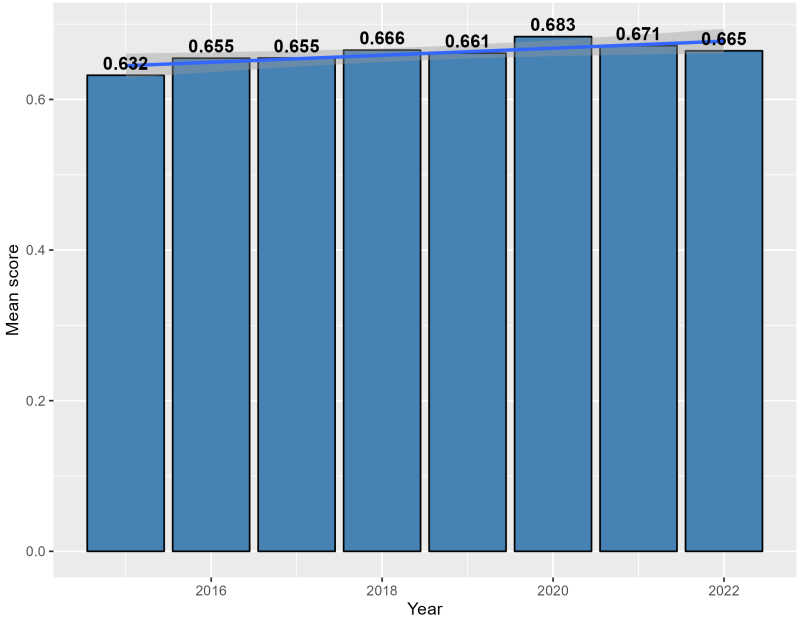

Figure 3.9: Mean normalised score per year - see 3.2.4.

3.2.5 Multiple-choice and extended text distributions

In figure 3.10 the density of the score given a question type is laid out. In figure 3.11 we see the cumulative distribution of scores using an ECDF plot (DeJesus, 2021).

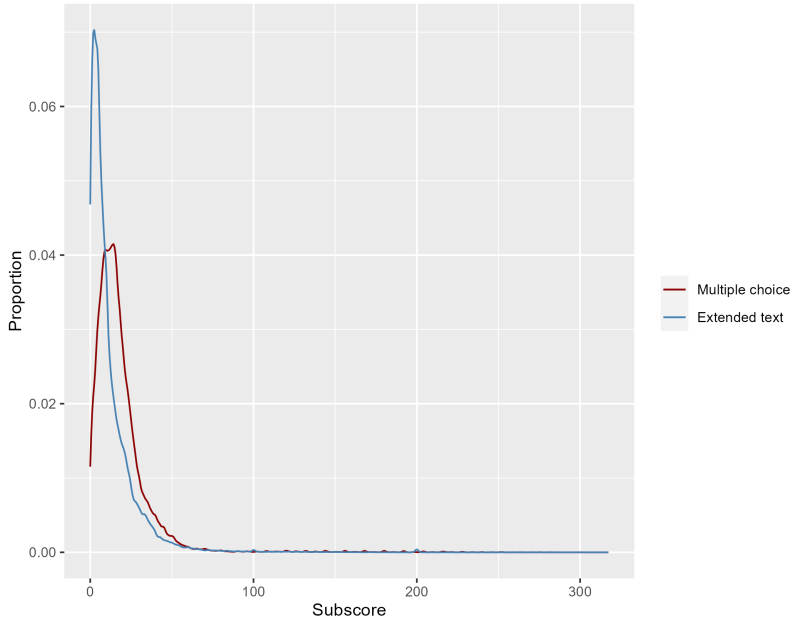

Figure 3.10: Density plot of types - see 3.2.5.

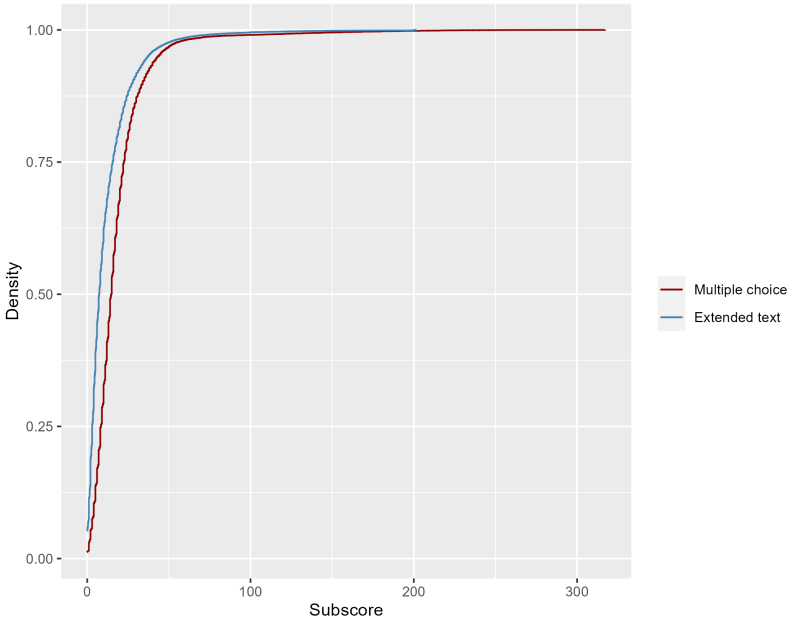

Figure 3.11: ECDF plot of types - see 3.2.5.

Score buildups

In figure 3.12, multiple-choice and extended text scores are depicted. Every dot represents a test, with an extended test score and a multiple-choice score. Note that the scores achievable are assigned by the test creator. This means that people can assign high scores to single questions, see table 3.2 at the maximum score for the distribution of these values. Next to the main plot in figure 3.12 are density plots of the frequencies. In figure 3.13 and 3.14, we see the same plot as in figure 3.12, except it is zoomed in to where the most activity happens. In figures 3.15, 3.16, and 3.17, we see the same data, but in a hexbin format. This is a heatmap depiction of figures 3.12, 3.13 and 3.14.

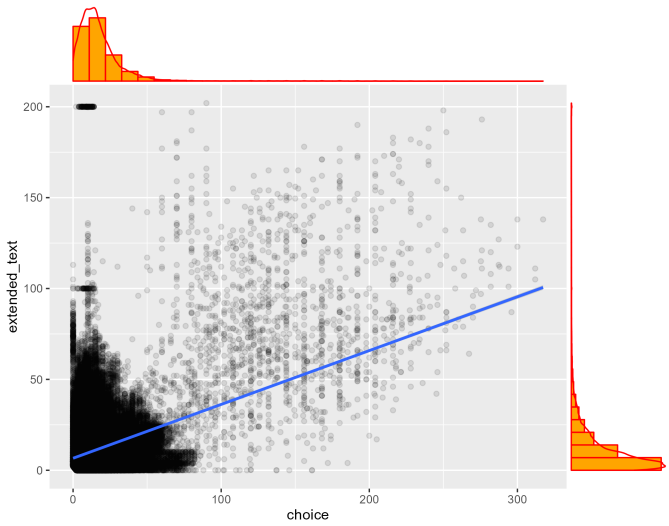

Figure 3.12: Scatterplot of score distribution of all tests - see 3.2.5

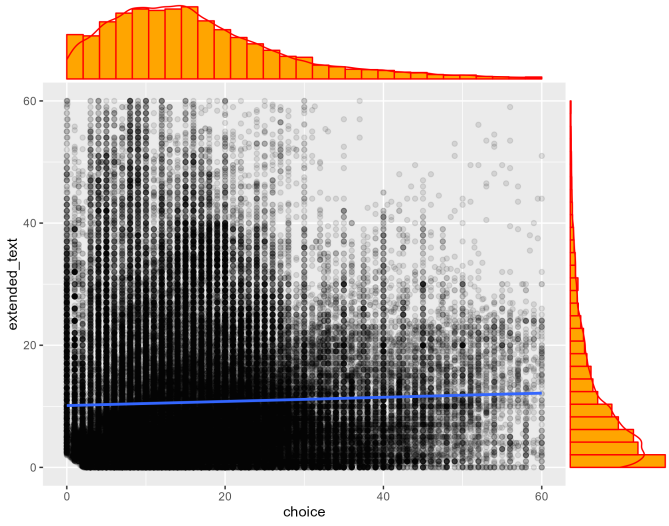

Figure 3.13: Scatterplot of score distribution of all tests limited to score 60 - see 3.2.5

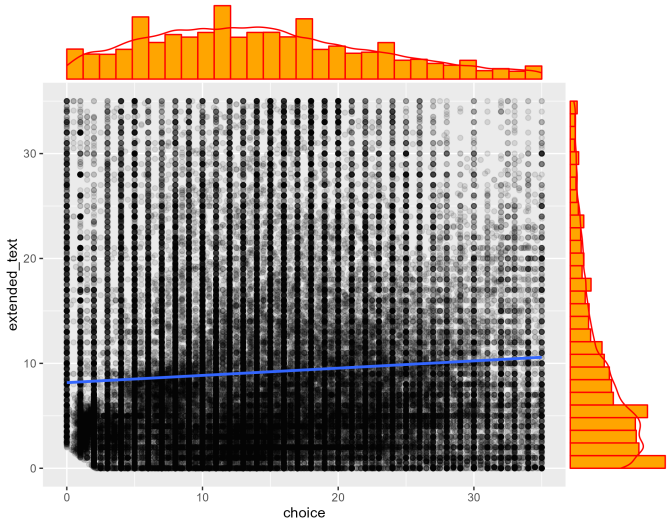

Figure 3.14: Scatterplot of score distribution of all tests limited to score 30 - see 3.2.5

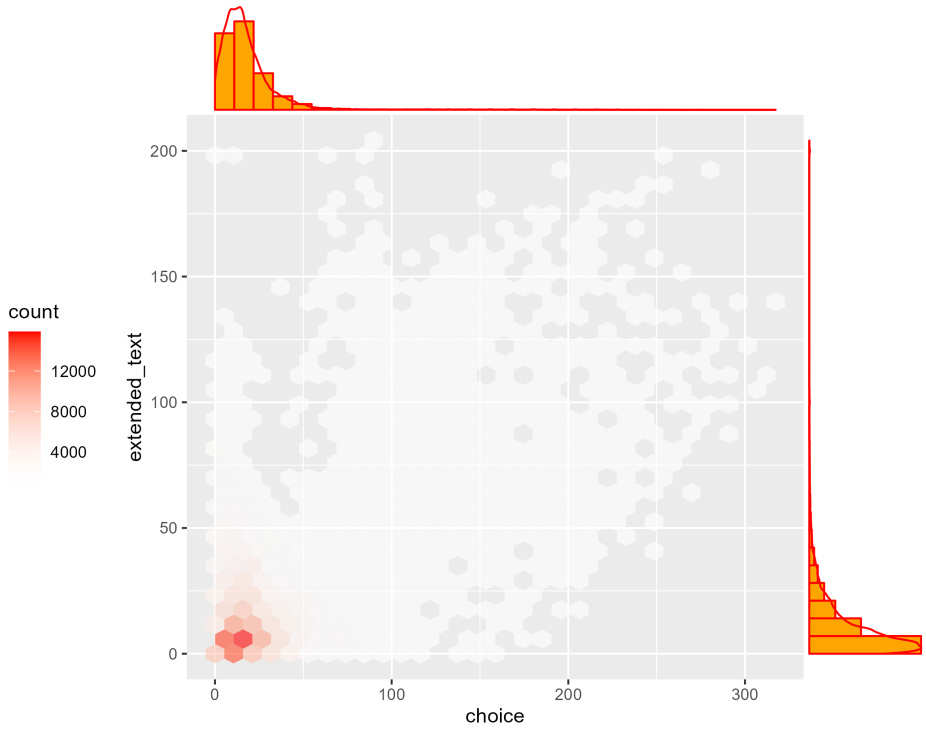

Figure 3.15: Hexbin of score distribution of all tests - see 3.2.5

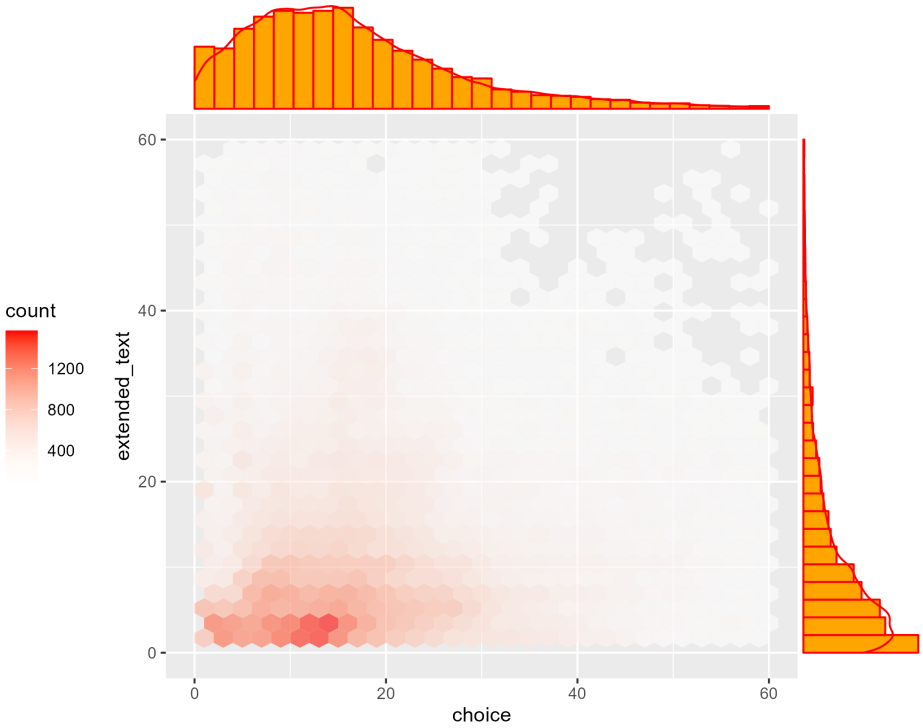

Figure 3.16: Hexbin of score distribution of all tests limited to score 60 - see 3.2.5

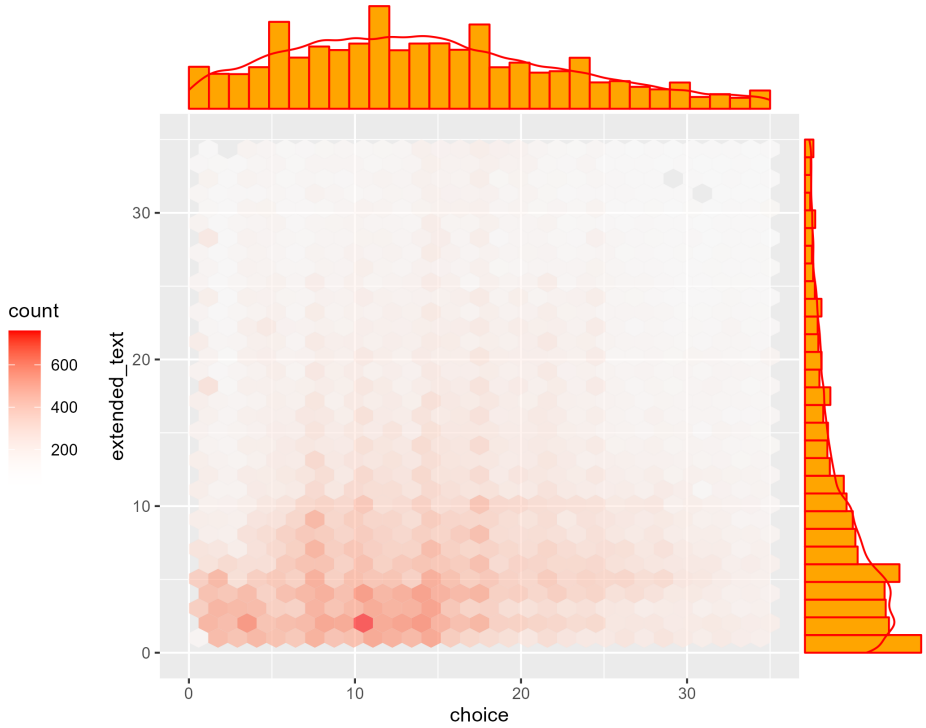

Figure 3.17: Hexbin of score distribution of all tests limited to score 35 - see 3.2.5

Normalized versions

In figure 3.18 and 3.19, the data has been normalised based on the max score achievable per type. So for each test from a student, we take its multiple-choice score and divide it by the maximum achievable score for multiple-choice. The same is done for the extended text type. The result is a graph with a maximum of 1 for both axes. A normalised scatterplot was also made with exam types expressed. See Figure 3.20 for this plot. Within this plot we have three lines, indicating the regression lines for each type of exam.

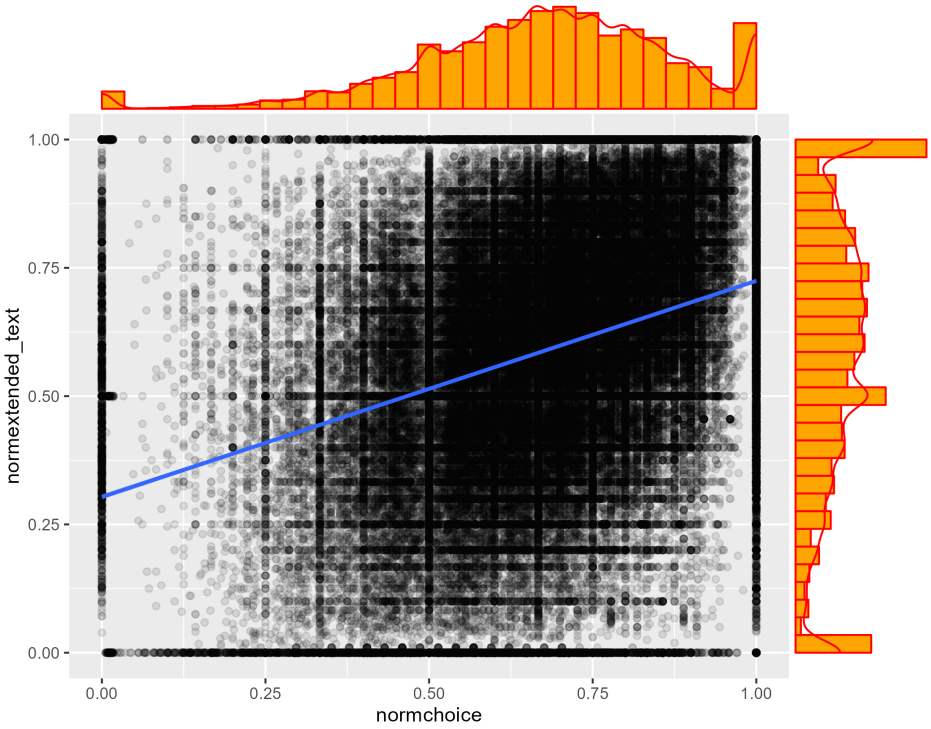

Figure 3.18: Scatterplot of tests normalised by max score achievable per type

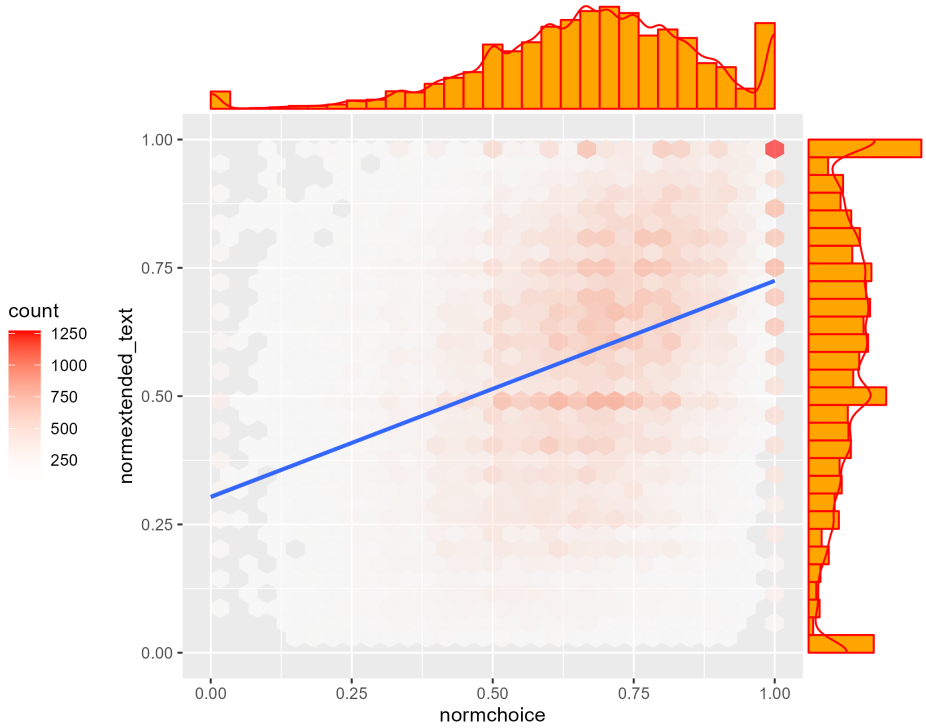

Figure 3.19: Hexbin of tests normalised by max score achievable per type

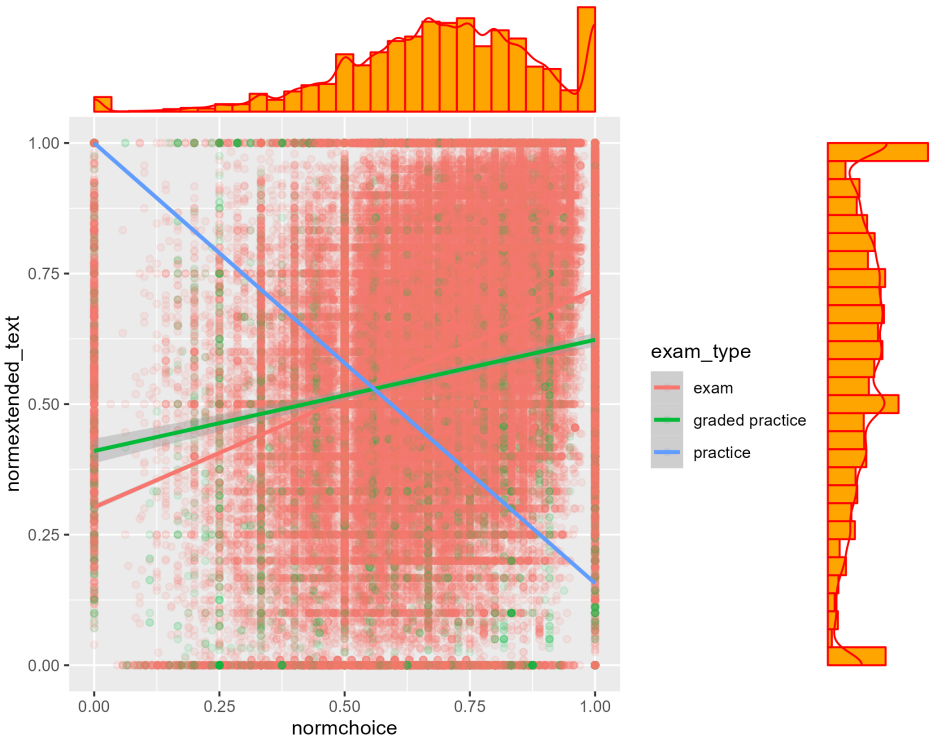

Figure 3.20: Scatterplot of tests normalised by max score achievable per test type

4 Methods

This chapter will discuss the practices used during the analysis part of this thesis. Firstly a brief overview of the data cleanup and collection practices will be laid out. Following this, the analysis part of this paper will be discussed.

4.1 Data preparation

4.1.1 Data collection

To access the data, a MySQL dump was provided by Paragin. This dump consisted of their entire database for the Utrecht University test bank. It was anonymized in advance so as not to include any names. We will not go deeply into the shape of this database, as this is out of this project's scope. It will suffice to say that this database was optimized for MySQL speed and not for ease of access. The MySQL table containing all these answers with relevant parameters was created to extract all of the answers. See appendix 1.1 for an overview of what a single row for a single answer looks like. This table was imported into R using RmySQL, see the appendix for the SQL and R code (Ooms & James, 2021). The tables imported consisted of roughly 16 million rows. In the next section, we will examine how the data was cleaned.

4.1.2 Data cleanup

Before we loaded the data into R, we already applied some cleanup inside the query. In the query's WHERE clause, we cleaned the data in several ways: filtering incomplete answers, filtering out tests with sectioning, and filtering out questions that consist of multiple elements as described in Figure 3.1.

Once we used RMySQL to import the data, a few more cleaning steps were taken. The study code feature was split up only to include the faculty code, and then anonymized to numbers. There was also a response variable created, which is the response answer for multiple-choice, but the weighted score for text-based answers. In the analysis section, we will explain why this was necessary.

4.2 The analysis

In this section, we will explain our naive percentage-based method, the Verhelst (2017) based method and our own IRT-based proportional deviation method. Lastly, the differential item functioning analysis based on T. Bechger and Maris (2015) will be discussed.

Variable	Calculated result	Calculation
choice score	14	
text score	20	
total score	34	
max choice score	20	
max text score	32	
max total score	52	
proportion of choice achieved	0.41	choice score / total score
proportion of text achieved	0.59	text score / total score
proportion of choice test	0.38	max choice score / max total score
proportion of text test	0.62	max text score / max total score
percentage based deviation choice	0.03	proportion of choice achieved - proportion of choice test
percentage based deviation text	-0.03	proportion of text achieved - proportion of text test

Table 4.1: Example sub-score proportional deviation score

Research question 4: *"What can we learn from different test types?"* was analysed using the balance method, with different test types being tests that are from a certain university faculty. It was also analysed using the regression lines in Figure 3.20.

Research question 5: *"What can we learn from different item types?"* was analysed using the balance analysis and mean score analyses.

To analyse research question 6: *"What can we learn from different examinee groups?"* we added a grouping variable to examinees and analysed different faculties for DIF within their tests based on this grouping variable.

We believe that in order to study two sets, we must determine the statistical significance of the difference between them. As such, we have also performed an independent samples t-test on each faculty. We compared the distributions of item scores of the two differing types, choice and text. We used the Welch T-test specifically because these two sets are of unequal size and of unequal distribution (Delacre et al., 2017).

4.2.1 Percentage based

Firstly we will examine the mean scores for both types per faculty. After this, we will do the actual percentage-based analysis. An example test percentage-based deviation profile calculation is given in Table 4.1. The result is a deviation profile, where a positive profile means a higher proportion correct than the proportion of that type in the test. These profiles were calculated for all tests. These profiles were grouped by faculties, and the mean per faculty was calculated using dplyr's mean function (H. Wickham & François, 2022).

row_number	1	2	3	4	5	6	7	8
booklet_id	1	1	1	1	1	1	1	1
booklet_score	0	1	2	3	4	5	6	7
expected_choice_score	0.00	0.00	0.00	0.00	1.93	1.89	3.83	3.74
expected_text_score	0.00	1.00	2.00	3.00	2.07	3.11	2.17	3.26

Table 4.2: Example expected score calculations

4.2.2 IRT-based

Score based deviation profiles

The first IRT-based method was done following Verhelst (2017) deviation profile calculations. The general steps taken were:

1. Fit an item response model for each test.
2. Calculate expected domain scores for each test.
3. Calculate observed domain scores for each test result, which is when a single student takes a certain test.
4. Calculate the deviation profile per type for each test result.
5. Calculate the mean deviation profiles per faculty.

To calculate the item response model, we use the R package Dexter (Maris & Bechger, 2023). Because Dexter uses keys, we cannot use text-based responses directly. Instead, we create a rule set for each item based on the scores possible. If, for instance, possible scores on a text item were 1,2,3.5 and 6, we would give each answer a score within this range and an equal interaction answer to the score. This way, we are able to input our text-based answer scores into dexter for analysis.

The model used was the ENORM model as implemented in (Maris & Bechger, 2023) and described in 2.5.5. Using the ENORM model, we estimate the expected sub-scores for each test in the database. The fitting of this model on all tests is not without hurdles. There are several types of errors that occur when training on this many tests. The most common error was caused by a uniform response to a question. This meant that for a single item in a test, all responses were equal. The ENORM model can not be fit when this is the case. We ignored these tests in our analysis and ended up fitting models for about 3000 tests.

When the fitting of a model was completed, it was used to estimate expected domain scores. These were given in the shape of table 4.2.

Afterwards, we add the domain scores to the table and use these to calculate the deviation profiles. This calculation was done by subtracting all expected scores from the actual score. Table 4.3 shows an example resulting table. Note that the deviation scores for choice and text are each

ResultID	1	1	1	1	1	1	1
booklet_id	1	1	1	1	1	1	1
booklet_score	4.00	3.00	3.00	4.00	3.00	3.00	3.00
text_score	0.00	0.00	0.00	0.00	0.00	0.00	0.00
choice_score	4.00	3.00	3.00	4.00	3.00	3.00	3.00
expected_choice_score	3.97	2.99	2.99	3.97	2.99	2.99	2.99
expected_text_score	0.03	0.01	0.01	0.03	0.01	0.01	0.01
deviation_choice	0.03	0.01	0.01	0.03	0.01	0.01	0.01
deviation_text	-0.03	-0.01	-0.01	-0.03	-0.01	-0.01	-0.01
faculty	4	4	4	4	4	4	4

Table 4.3: Example deviation profile table

other's opposite sign. This is because when the actual scores differ for an observed score, this difference has to come from somewhere, in this case, the other item type.

Finally, we calculate mean deviation profiles per faculty, which can then be reported in a diverging bar chart.

Proportional based deviation

The start of this analysis is laid out exactly the same as within section 4.2.2. We diverge from this method when calculating the deviation profiles. Instead of subtracting the expected_choice_score from the choice_score, we calculate proportions for both and subtract those, see Table 4.4. The result is a proportional deviation profile

A problem that this analysis solves is that it scales the deviation profiles in proportion to the tests. If a test has higher scores on average, then the deviation profiles based on the score will reflect this difference. When a multiple-choice question is answered the question is often either right or wrong, resulting in either 0 or more than 0 points. When the scores per question then differ between tests, a right or wrong answer on two different tests can have different influences on their overall deviation profiles depending on the scores assigned to each question. This is why this way of analysing also allows us to make a better comparison between tests, whereas the Verhelst (2017) method is more suitable for single tests.

Variable	Calculated result	Calculation
choice score	14	
text score	20	
total score	34	
expected choice score	16	
expected text score	18	
proportion of choice achieved	0.41	choice score / total score
proportion of text achieved	0.59	text score / total score
proportion of expected choice	0.47	expected choice score / max total score
proportion of expected text	0.53	expected text score / max total score
percentage based deviation choice	-0.06	proportion of choice achieved - proportion of expected choice
percentage based deviation text	0.06	proportion of text achieved - proportion of expected text

Table 4.4: Example proportional deviation profile calculation

4.2.3 Differential item analysis

In order to answer research question 6: *"What can we learn from different examinee groups?"* - see section 1.1, we perform a differential item analysis, and we follow the same steps as shown in section 4.2.2. We have added the language feature to each student. This feature is the language that they have selected as their preferred language in Remindo. Instead of training an ENORM we use the DIF function as available in Dexter on this language feature (Maris & Bechger, 2023). This analysis is based upon the paper by T. Bechger and Maris (2015). Unfortunately Dexter is not able to analyse all tests for several reasons such as uniform responses and not enough score variation. The result of this analysis is a Chi-square value, degrees of freedom value and probability value. The probabilities used were then analysed using boxplots for each faculty.
When testing for DIF through multiple statistical tests. One should expect to have some false discoveries, discovering DIF where there is none. Kim and Oshima (2013) studied this effect and concluded that for some DIF methods, the Holm method was beneficial to the analysis (Holm, 1979). We will also report p values after the Holm correction.

5 Results

5.1 Balance

5.1.1 Percentage based

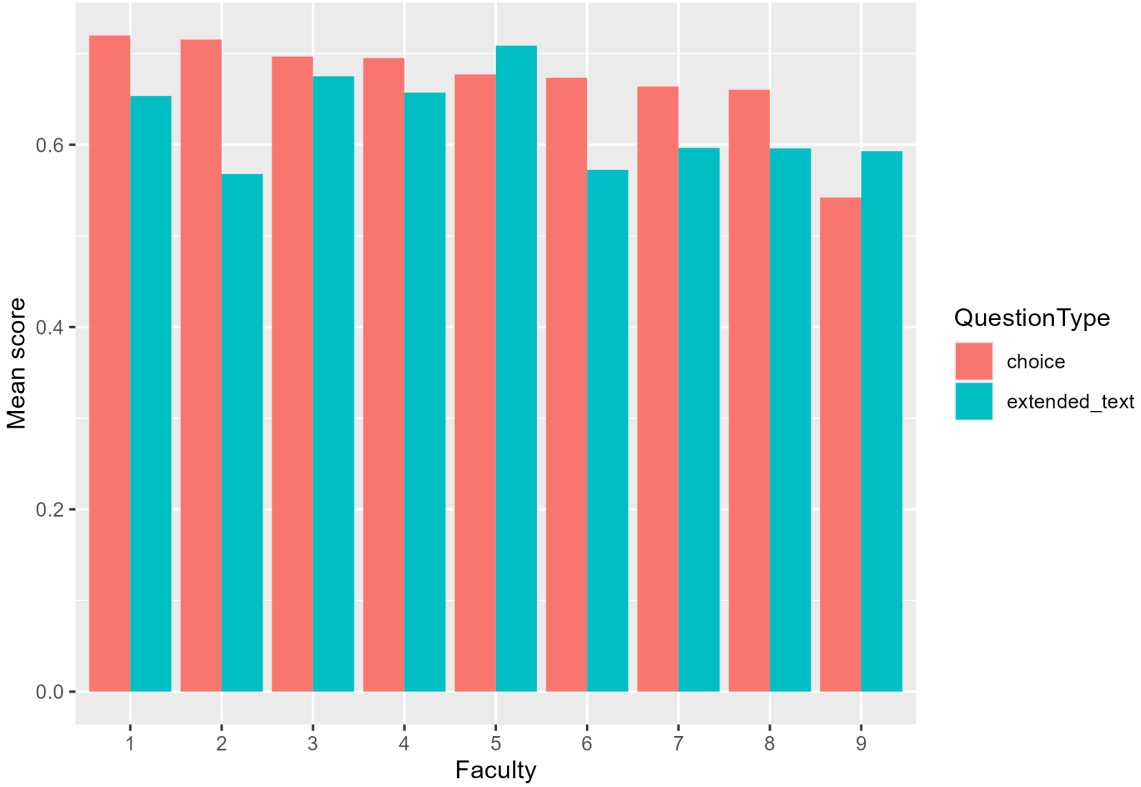

Figure 5.1: Mean faculty scaled scores

In Figure 5.1, the mean score per faculty is displayed for multiple-choice and text formats. It is ordered by the highest mean choice score. The lowest is faculty 9, while the highest mean choice score is faculty 1.

In Figure 5.2, the mean deviation score per faculty is displayed for multiple-choice and text questions. It is ordered by the highest text score. Faculty 9 has the highest mean choice deviation, while faculty 8 has the lowest mean choice deviation. All faculties score better proportionally on

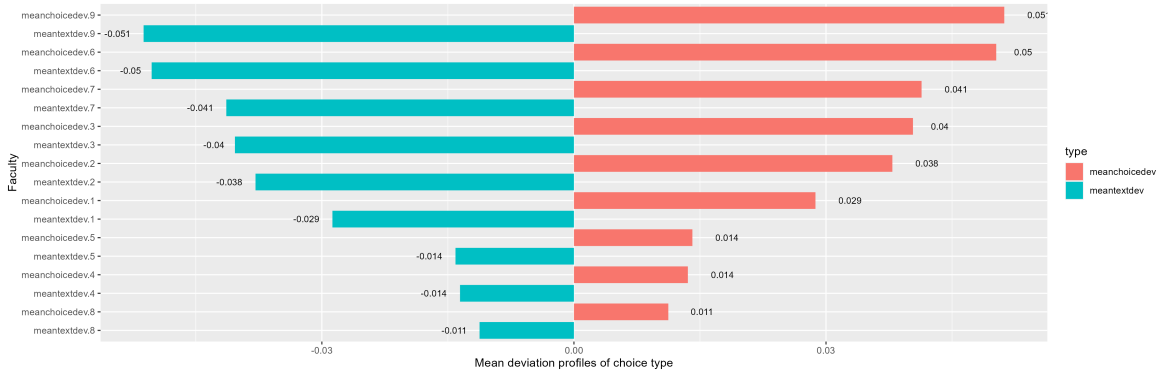

Figure 5.2: Mean percentage based deviation of scores

multiple-choice, according to this analysis. When we take faculty 9 for example the 0.051 means that 5.1 percent more score was earned on the multiple-choice part of the test than the sub-score distribution over text and multiple-choice would suggest.

5.1.2 IRT-based

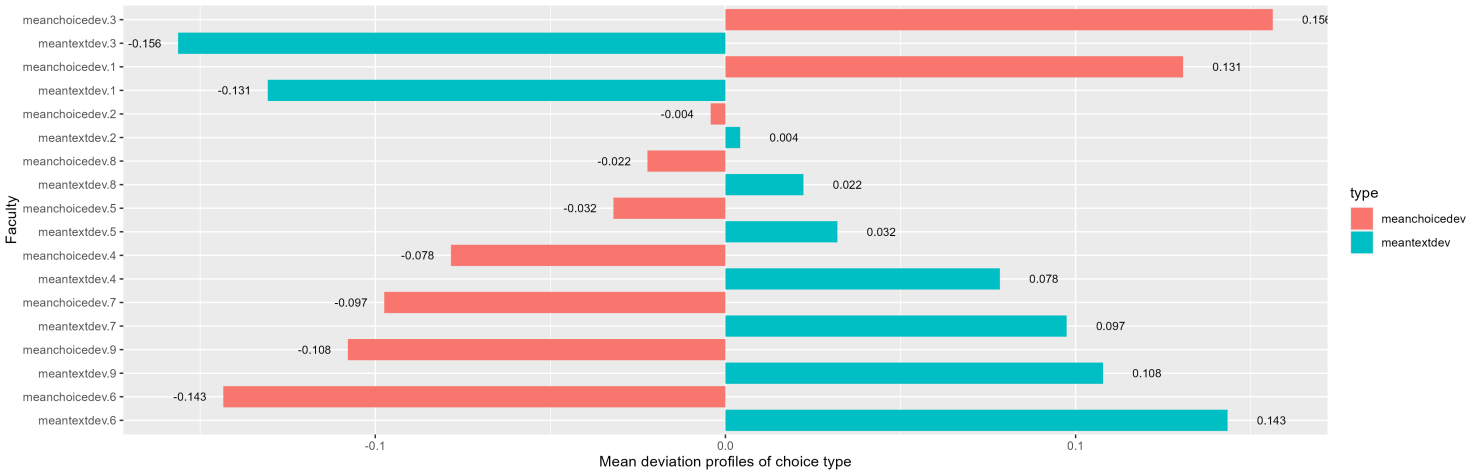

Figure 5.3: Mean deviation scores per faculty

In Figure 5.3, the deviation scores are depicted. The data is presented in the same manner as Verhelst (2017). Note that the profiles switch from being text favoured to choice at faculty 3 and 1. The faculty with the most text-heavy deviation profile is faculty 6, while this is 3 for choice. The result of 0.156 for faculty 3 means that on average within this faculty examinees tend to gain 0.156 higher multiple-choice scores on their tests than the ENORM model estimates, meaning that the balance is tilted towards the multiple-choice type questions according to this analysis.

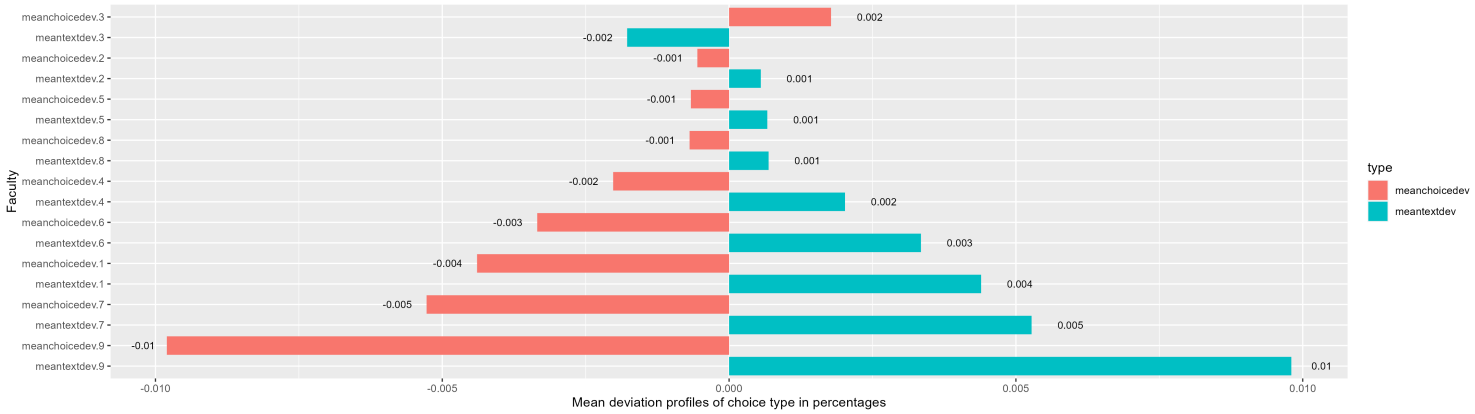

Figure 5.4: Mean proportional deviation scores per faculty

5.1.3 Proportional IRT-based

In Figure 5.4, the proportional deviation scores are depicted. Note that the profiles switch from being text favoured to choice at faculty 2 and 3. The faculty with the most text-heavy deviation profile is faculty 9, while this is 3 for choice. The result of 0.002 for faculty 3 means that on average within this faculty examinees tend to gain 0.2% higher multiple-choice scores on their tests than the ENORM model estimates, meaning that the balance is tilted towards the multiple-choice type questions according to this analysis.

5.1.4 Comparison between methods

In Figure 5.1, the results from the percentage, IRT-based and proportional IRT-based methods are shown. The faculty names with an * after them are the faculties for which results were equal. An interesting observation is that between the two IRT-based methods, results are pretty similar, with the exception of faculty 1, which switches between being choice heavy in the regular IRT-based approach, to text-heavy within the proportional IRT-based approach.

Faculty	Percentage	IRT	Proportional IRT	Welch t-test
9	choice	text	text	0.33
8	choice	text	text	0.00
7	choice	text	text	0.00
6	choice	text	text	0.00
5	choice	text	text	0.02
4	choice	text	text	0.00
3*	choice	choice	choice	0.00
2	choice	text	text	0.00
1	choice	choice	text	0.00

Table 5.1: Comparison between percentage, IRT-based and proportional IRT-based methods

A T-test was performed on scaled score values per faculty for the two different question types, choice and text. They were tested using a Welch T-test. The Welch T-test was used because this concerns two sets of unequal size and of unequal variance (Zach, 2020).

5.2 Differential item functioning

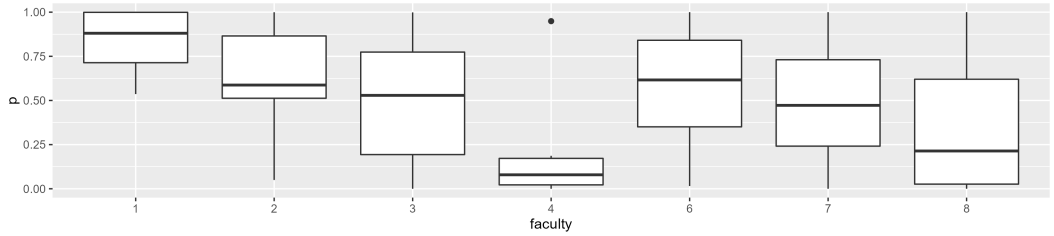

Figure 5.5: Differential item functioning per faculty based on language grouping

In Figure 5.5, boxplots are shown for the differential item functioning analysis. For this analysis, 359 tests, or booklets, were analysed for DIF based on the language feature. The distributions per faculty can be seen in Table 5.2. Of the 359 tests, 61 had a p-score lower than 0.05 before Holm correction. After applying Holm correction on the entire set of 359 tests, we find that 18 tests have a value below 0.05 and thus contain DIF. Frequency of DIF tests per faculty can be seen in Table 5.2. Most notable is faculty 8, with 14 tests containing significant DIF between the language feature.

Faculty	Amount of tests analysed	Amount of tests with DIF	Amount of tests with DIF after holm correction	Percentage tests containing DIF
1	5	-	-	-
2	13	1	-	-
3	33	4	1	0.03
4	9	4	1	0.11
6	70	5	-	-
7	93	7	2	0.02
8	136	40	14	0.10

Table 5.2: Frequencies of tests per faculty containing DIF

6 Discussion

The aim of this research was to learn something from sub-scores achieved on tests in higher education. The central question was:

1. *What can we learn from sub-scores based on item types in higher education high stake testing?*

To answer this question, several sub-questions have been examined. We will examine these questions below.

2. *What types of sub-scores can we identify?*

To examine this question, we did both a literature search and a data exploration. We found that within the literature, most sub-scoring papers tend to examine domains such as math or reading. We also found the paper by Verhelst (2017), doing sub-scoring research based on question types rather than domains. In summary, we identified two ways of sub-scoring, question type-based and domain type-based.

3. *What methods are available in the literature for sub-score analysis?*

We found several ways of analysing sub-scores. Haberman (2005) analyses sub-scores by using classical test theory to examine whether a sub-score holds extra value over the total score. Haberman and Sinharay (2009) examine the same question using mean square error approaches. Approaches also examined are direct reports of subscores, estimates of subscores based on total score, combined estimates based on subscores and total scores, and residual analysis of subscores. Verhelst (2012) uses IRT to analyse three different categories within the PISA dataset (Mulford, 2002). They construct deviation profiles, which are the difference between observed and estimated scores on a sub-score. In Verhelst (2017), the same technique was used to compare countries within the TIMSS dataset (Mullis et al., 2012). They also analyse different item types within this paper, comparing multiple-choice and constructed responses. In summary, we find a classical test theory approach, several mean square error approaches and an item response theory approach to analysing sub-scores.

4. *What can we learn from different test types?*

In Figure 3.20 we see the distribution between different exam types. This plot seems to suggest that when the total score achieved is higher, students score better on exam text questions than on graded practice text questions. For multiple-choice questions for instance, when the total score is lower, students score better on graded practices than on exams. This plot just scratches the surface of this research question and there is a lot more to discover about different test types. Are there different score distributions? Do people on average score higher on a graded practice?
We also analysed different test types by grouping them based on their faculty, below we will discuss our findings.

5. *What can we learn from different item types?*

Figure 3.4 shows us the distribution of questions over the available item types in Remindo. The most prevalent types are multiple-choice and extended text. Together they make up over 85% of all questions. To study these types we did two analyses. A naive percentage-based deviation profile calculation and two deviation profile calculations based on an IRT model. A Welch t-test was applied to the set of multiple-choice and text answers for each faculty. The t-test indicates that for all faculties except for faculty 9, the differences observed between the scores of type multiple-choice and text are not due to chance.

For the naive percentage-based deviation profile we calculated the differences between the two proportions. The first is the proportion between the achieved type score and the achieved total score. The second proportion is the proportion between the test type score and the test total score. The results indicate that for all tested faculties, multiple-choice yields higher scores than text. This analysis was done on the entire set that was available after the filtering process shown in the data cleanup section, see Section 4.1.2. While results indicate that higher scores are achieved on multiple-choice, this analysis fails to address the problem caused by guessing correction. When we look at Figure 3.19, we see that on the left side, the graph becomes scarce. This is probably, for a large part, caused by guessing. On a Multiple-choice question, an examinee has a chance of achieving a score higher than 0 when answering randomly. This means that probably, multiple-choice normalised scores will be on average higher than their text counterparts, which is what the result of our percentage-based analysis seems to suggest. More research is needed to examine the effects of guessing on this naive approach. Another problem with this approach is item difficulty. Some items are harder than others but yield the same score. This occurs when one of the sub-scores was overall harder than the other, making scoring higher on that sub-score more difficult and thus lowering the mean score.

The IRT-based deviation profile was constructed using the ENORM model as presented in Koops et al. (2020). This model was used to estimate sub-scores for both text and choice-type questions. The resulting expected scores were then compared to the actual scores of examinees. The mean deviation profiles were then used to calculate the means per faculty, see Figure 5.3. Results indicate that for faculties 1 and 3, more score is achieved on multiple-choice, while this trend is reversed for the rest. This approach takes the difficulty of items as well as guessing correction into account. A problem with this approach is that when the scales for tests differ, meaning one test has a higher total score, these tests are hard to compare. In order to combat this, we performed a similar test using proportions, which is discussed below. Another shortcoming of our analysis is the set used. Not all tests lend themselves to fitting within the ENORM model. Items that had equal response patterns over all examinees could not be fit within the ENORM, and thus we had to remove the test. As a result, we only analysed less than half of the entire test set. Results could be different if we analysed the entire set. More research is needed into whether simply removing these equal response questions will be an adequate solution. Examining whether the current result is similar to results for the entire set could also be interesting.

The proportional IRT-based method was constructed following the same first steps as the regular IRT-based method. The difference between the two methods was that the expected scores were turned into proportions of the max achievable score. The achieved scores were also turned into proportions of the max achievable score. The expected proportional score was then subtracted from the achieved proportional score, to obtain the proportional deviation profile. The mean proportional deviation profiles were then used to calculate the means for each faculty, see Figure 5.4. Results indicate that only for faculty 3 more points are achieved than expected on multiple-choice, while this trend is reversed for the rest. For this analysis, the same shortcomings apply as described within the section above.

In summary, the methods give contradictory results. The naive percentage-based approach is based solely on the proportion of scores within tests to calculate the estimated score. The regular IRT-based method uses more information available within a test to calculate the estimated score and the proportional IRT-based approach tries to enhance the IRT-based approach to be suitable for aggregating between tests. Furthermore, we can not directly compare the regular IRT-based method to the naive percentage-based method, because the regular IRT-based method yields score deviation and not proportion. The naive percentage-based method was also performed on a bigger data set than the IRT-based methods.

6. *What can we learn from different examinee groups?*

We examined two groups of examinees. One group self-selected English as their preferred system language within the testing software while the other group selected Dutch. These groups were examined per faculty using the DIF analysis function within the R package Dexter (T. Bechger & Maris, 2015; Maris & Bechger, 2023). The results were further analysed using Holm correction (Holm, 1979). We found four faculties with tests containing DIF.
Dexter was unable to calculate DIF on tests for several reasons. As a result, we only analysed 359 tests. This means that our analysis does not incorporate a large part of the tests available. Further research is needed to examine DIF on all tests. Analysis of the entire set could result in a different outcome. Furthermore, research is needed to find out how the language selection is performed. What is the default option? Can a user change this language on the go? These aspects have to be examined to understand the context of the language grouping variable.
A missed opportunity is the grouping variables. Unfortunately for this analysis, there were not many background variables available within this set. Interesting background variables, such as gender, country of origin, mother tongue and age were not available.

6.0.1 In conclusion

To answer our main research question we have analysed multiple-choice and text questions per faculty using deviation profiles. To calculate these deviation profiles three methods were used, a naive percentage-based approach, a regular IRT-based approach and a proportional IRT-based approach. The results of these analyses were contradictory, but they seem to indicate that there is a difference in balance between faculties. Further research is needed to examine whether examinees score better on multiple-choice or text questions. We also analysed differential item functioning for two groups based on their preferred language. The result of this analysis was that within some

faculties, there seems to be DIF present within tests. Further research is needed to verify that this is true for the entire test set.

6.0.2 Future works

For future research into this dataset, we would like to recommend some more attention be given to the filtering of equal response questions. By filtering these types of questions, we think the size of the to-be-analysed set can be increased. By doing so the results of this study can be made more reliable. Another option for resolving the equal response patterns would be to change the expected profile estimation. Expected profiles were able to be generated for all tests using the naive-percentage method. By changing which item response model we use to calculate estimated scores, we could enable these calculations for equal response-containing tests.

Adding background variables could result in interesting findings as well. By adding a mother tongue and/or country of origin variable, cultural differences could be examined on different types. Gender is another popularly examined background variable and would fit nicely within our analysis. More involved background variable examples would be risk averseness. This could be used to see whether risk-averse people tend to score better on multiple-choice. By doing surveys before a test is administered other phenomena could be studied, such as the impacts of nervousness on test scores, and types of questions.

Time could be used as a third correction parameter, do we see that certain faculties yield different scores over the years? Do they yield different scores throughout the year? Maybe we even see that on average, students score better between 12:00 and 15:00. An advantage of examining time is that there is already data available within our data set to analyse this, in the form of a datestamp per question. In Figure 3.7 we have already shown the differences in scores between months of the year, to briefly touch upon this subject.

Lastly, an interesting comparison that could be made is between educational institutions. Do certain institutes yield higher mean scores on average? Do similar faculties within different institutions yield similar scores on both item types? Do the same trends hold for high school and higher education? Do the trends differ between types of higher education, such as universities of applied sciences and regular universities? Do we see different trends in the more technically focused universities? Paragin has data on most of these types of institutions, thus enabling this research from a practical perspective.

Bibliography

Bechger, T. (2022). https://mran.microsoft.com/snapshot/2023-01-07/web/packages/dexter/vignettes/DIF_vignette.html

Bechger, T., & Maris, G. (2015). A statistical test for differential item pair functioning. *Psychometrika*, *80*(2), 317–340.

Bechger, T., Maris, G., Beguin, A., & Verstralen, H. (2003). Combining classical test theory and item response theory. *Measurement and Research Department Reports*, (2003-4).

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. *Psychometrika*, *33*(4), 427–447.

Cressie, N., & Holland, P. (1983). Characterizing the manifest probabilities of latent trait models. *Psychometrika*, *48*, 129–141.

Dahl, D. B., & Scott, D. (2019). *Xtable: export tables to latex or html* [R package version 1.8-4]. http://xtable.r-forge.r-project.org/

Darrell Bock, R. (1972). Estimating item parameters and latent ability when responses are scored in two or more nominal categories. *Psychometrika*, *37*(1), 29–51.

de la Torre, J., Song, H., & Hong, Y. (2011). A comparison of four methods of irt subscoring. *Applied Psychological Measurement*, *35*(4), 296–316.

DeJesus, J. (2021). What, why, and how to read empirical cdf. https://towardsdatascience.com/what-why-and-how-to-read-empirical-cdf-123e2b922480

Delacre, M., Lakens, D., & Leys, C. (2017). Why psychologists should by default use welch's t-test instead of student's t-test. *International Review of Social Psychology*, *30*(1).

Dorans, N., & Holland, P. (1992). Dif detection and description: mantel-haenszel and standardization 1, 2. *ETS Research Report Series*, *1992*(1), i–40.

Geman, S., & Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 721–741.

Haberman. (2005). When can subscores have value? *ETS Research Report Series*, *2005*(1), i–15.

Haberman & Sinharay. (2009). Reporting subscores for institutions. *British Journal of Mathematical and Statistical Psychology*, *62*(1), 79–95.

Haladyna, T., & Kramer, G. (2004). The validity of subscores for a credentialing test. *Evaluation & the health professions*, *27*(4), 349–368.

Hao, H. (2022). Intro to item response modeling in r. https://hanhao23.github.io/project/irttutorial/irt-tutorial-in-r-with-mirt-package/

Harris, D., & Crouse, J. (1993). A study of criteria used in equating. *Applied Measurement in Education*, *6*(3), 195–240.

Henry, L., & Wickham, H. (2022a). *Purrr: functional programming tools* [R package version 0.3.5]. https://CRAN.R-project.org/package=purrr

Henry, L., & Wickham, H. (2022b). *Rlang: functions for base types and core r and tidyverse features* [R package version 1.0.6]. https://CRAN.R-project.org/package=rlang

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, 65–70.

Kim, J., & Oshima, T. (2013). Effect of multiple testing adjustment in differential item functioning detection. *Educational and Psychological Measurement*, *73*(3), 458–470.

Kolen, M., & Brennan, R. (2004). Test equating, scaling, and linking.

Koops, J., Bechger, T., & Maris, G. (2020). Bayesian inference for multistage and other incomplete designs.

Kwak, S. G., & Kim, J. H. (2017). Central limit theorem: the cornerstone of modern statistics. *Korean journal of anesthesiology*, *70*(2), 144–156.

Leading, L. L., & Monaghan, W. (2006). The facts about subscores. *Princeton, NJ: Educational Testing Services*.

Linacre, J. (1994). Sample size and item calibration stability. *Rasch Mes Trans.*, *7*, 328.

Loe, A. (2021). https://aidenloe.github.io/introToIRT.html

Lord, F. M. (2012). *Applications of item response theory to practical testing problems*. Routledge.

Maris & Bechger. (2023). *Dexter: data management and analysis of tests* [R package version 1.2.3]. https://dexter-psychometrics.github.io/dexter/

Maris, G., Bechger, T., & Martin, E. S. (2015). A gibbs sampler for the (extended) marginal rasch model. *psychometrika*, *80*, 859–879.

Martin, M., Mullis, I. V., Foy, P., & Stanco, G. M. (2012). *Timss 2011 international results in science.* ERIC.

Mulford, B. (2002). Sorting the wheat from the chaff: knowledge and skills for life: first results from oecd's pisa 2000. *European Journal of Education*, *37*(2), 211–221.

Müller, K., & Wickham, H. (2022). *Tibble: simple data frames* [R package version 3.1.8]. https://CRAN.R-project.org/package=tibble

Mullis, I. V., Martin, M. O., Foy, P., & Arora, A. (2012). *Timss 2011 international results in mathematics.* ERIC.

O'Neill, T., Gregg, J., & Peabody, M. (2020). Effect of sample size on common item equating using the dichotomous rasch model. *Applied Measurement in Education*, *33*(1), 10–23.

Ooms, J., & James, D. (2021). *Rmysql: database interface and mysql driver for r* [R package version 0.10.23]. https://CRAN.R-project.org/package=RMySQL

Paragin, R. (2021). De makers van remindotoets, remindocontent, mijnportfolio en remindoevc. https://www.paragin.nl/

Puhan, G., Sinharay, S., Haberman, S., & Larkin, K. (2008). Comparison of subscores based on classical test theory methods. *ETS Research Report Series*, *2008*(2), i–23.

qti. (2022). *Qti* [https://www.imsglobal.org/activity/qtiapip]. https://www.imsglobal.org/activity/qtiapip

R Core Team. (2022). *R: a language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Rasch, G. (1993). *Probabilistic models for some intelligence and attainment tests.* ERIC.

Roju, N., Van der Linden, W., & Fleer, P. (1995). Irt-based internal measures of differential functioning of items and tests. *Applied Psychological Measurement*, *19*(4), 353–368.

Roode, M. (2020). Digital assessment during the coronavirus: what our data shows about how higher education institutions handled the pandemic. https://www.uniwise.co.uk/news/digital-assessment-during-the-coronavirus-what-our-data-shows-about-how-higher-education-institutions-handled-the-pandemic

Sahin, A., & Anil, D. (2017). The effects of test length and sample size on item parameters in item response theory.

Sehmitt, A. P., & Dorans, N. J. (1990). Differential item functioning for minority examinees on the sat. *Journal of Educational Measurement*, *27*(1), 67–81.

Thompson, N. (2022a). Differential item functioning (dif): definition and examples. https://assess.com/differential-item-functioning/

Thompson, N. (2022b). What is the generalized credit model (gpcm)? https://assess.com/what-is-the-generalized-partial-credit-model/

Thompson, N. (2022c). Item response theory (irt) - introduction, benefits, models: asc. https://assess.com/what-is-item-response-theory/

Verhelst, N. (2004). Classical test theory. *Council of Europe, Reference Supplement to the Manual for Relating Language Examinations to the Common European Framework of Reference for Languages: Learning, Teaching, Assessment (Section C). Strasbourg: Council of Europe.(download from http://www. coe. int/t/dg4/linguistic/manuel1_en. asp*.

Verhelst, N. (2012). Profile analysis: a closer look at the pisa 2000 reading data. *Scandinavian Journal of Educational Research*, *56*(3), 315–332.

Verhelst, N. (2017). Balance: a neglected aspect of reporting test results. In *Cognitive abilities and educational outcomes* (pp. 273–293). Springer.

Vromant, K. (2022). More than multiple choice: all question types at a glance. https://assessmentq.com/blogpost-more-than-multiple-choice-all-question-types-for-digital-exams-at-a-glance-2022/?lang=en

Weeks, J. (2010). plink: an R package for linking mixed-format tests using irt-based methods. *Journal of Statistical Software*, *35*(12), 1–33. http://www.jstatsoft.org/v35/i12/

Weiss, D. (2022). The rasch model. https://assess.com/the-rasch-model/

Wickham, H. (2016). *Ggplot2: elegant graphics for data analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org

Wickham, H. (2022a). *Forcats: tools for working with categorical variables (factors)* [R package version 0.5.2]. https://CRAN.R-project.org/package=forcats

Wickham, H. (2022b). *Plyr: tools for splitting, applying and combining data* [R package version 1.8.7]. https://CRAN.R-project.org/package=plyr

Wickham, H. (2022c). *Stringr: simple, consistent wrappers for common string operations* [R package version 1.4.1]. https://CRAN.R-project.org/package=stringr

Wickham, H. (2022d). *Tidyverse: easily install and load the tidyverse* [R package version 1.3.2]. https://CRAN.R-project.org/package=tidyverse

Wickham, H., & Bryan, J. (2022). *Usethis: automate package and project setup* [R package version 2.1.6]. https://CRAN.R-project.org/package=usethis

Wickham, H., & Chang, W. (2022). *Ggplot2: create elegant data visualisations using the grammar of graphics* [R package version 3.3.6]. https://CRAN.R-project.org/package=ggplot2

Wickham, H., & François, R. (2022). *Dplyr: a grammar of data manipulation* [R package version 1.0.10]. https://CRAN.R-project.org/package=dplyr

Wickham, H., & Girlich, M. (2022). *Tidyr: tidy messy data* [R package version 1.2.1]. https://CRAN.R-project.org/package=tidyr

Wickham, H., & Hester, J. (2022). *Devtools: tools to make developing r packages easier* [R package version 2.4.5]. https://CRAN.R-project.org/package=devtools

Wickham, H., Hester, J., & Bryan, J. (2022). *Readr: read rectangular text data* [R package version 2.1.3]. https://CRAN.R-project.org/package=readr

Wickham, H., & Müller, K. (2022). *Dbi: r database interface* [R package version 1.1.3]. https://CRAN.R-project.org/package=DBI

Wickham, M. (2022). *Dbi: r database interface* [https://dbi.r-dbi.org, https://github.com/r-dbi/DBI].

Wright, B., & Tennant, A. (1996). Sample size again. *Rasch Measurement Transactions*, *9*(4), 468.

Wright & Stone. (1979). Best test design.

Zach. (2020). Welch's t-test: when to use it + examples. https://www.statology.org/welchs-t-test/

On the relevance of this thesis to the field of AI

This thesis is relevant to the field of artificial intelligence because it highlights the problems that come with latent variable estimation. The way in which this thesis tackles the question of balance is one of interest to the field of AI. Balance is an evaluation of fairness in a sense. Are these differing types of questions fair? Does one of these question types yield a performance bias against certain groups? Fairness is not a clear-cut concept. As such, there are also many ways of investigating whether something is fair. In this thesis, we found that there is differential item functioning based on what language preference students picked. Which means that certain items performed differently between these groups.

In this current thesis, the balance research was done on a set of educational tests. In future work, we might need to apply similar techniques to different decisions of machine learning algorithms. This is where IRT provides a solution; it formulates the ability of an individual not by raw performance but by relative ability. By doing so, it aims to give a more fair and accurate estimation of ability. This thesis shows a way of examining latent traits using IRT. Therefore I am convinced that we will see IRT-based methods seep into the fairness measurements as used by machine learning experts.

Appendix

1 Data exploration

1.1 Feature explanations

1. *Question sequence*

The sequence that a question got within the test. These sequences can be the same for every test of that type, or, as discussed earlier, differ from one another.

2. *Recipe ID*

An identifier that is unique to a single test. An answer ID is an identifier unique to a specific answer for a specific candidate on a specific question. A recipe is a mould used to create a test out of an item bank

3. *Answer ID*

An identifier unique to a specific answer for a specific candidate on a specific question.

4. *Result ID*

An identifier unique to all answers given by a specific student on a specific test.

5. *Question ID*

An identifier that is unique to a question (version).

6. *Section ID*

An identifier used to specify the section that a question belonged to, the default being NULL, for no sectioning.

7. *User ID*

An identifier unique to a single user.

8. *Date/time stamp last change*

The last changed is used to see when the last action concerning the question was performed. When a student fills out a question and returns to it after 30 minutes to update it, this new time is the date/time saved.

9. *Duration*

A measurement in seconds of the amount of time an examinee has spent on the question.

10. *Completion status*

A status type given to every answer. The status has several options, being: completed, not_attempted, incomplete and unknown. For this dataset we have filtered out all types of completion exept for completed.

11. *Guess score*

The score that a candidate would get were a candidate to guess the answer. This is automatically calculated by RemindoToets.

12. *Score*

The achieved score that a candidate has achieved.

13. *Maximum score*

The maximum score a candidate could have achieved.

14. *Fullscore*

A boolean value depicting whether the candidate achieved the maximum score for this question.

15. *Weight*

The weight that the test designer has given to the question.

16. *NeedsManualCheck*

A boolean value that is true when the score can or should not be calculated by RemindoToets.

17. *DontKnow*

The DontKnow feature is an optional feature for questions where it is possible for the candidate to indicate that a candidate does not know the answer.

18. *QTI XML*

A value that is a description of the question in a QTI format qti, 2022. This is a standard for saving test questions in.

19. *Metadata*

The metadata is a JSON version of everything that is needed to show the question to a candidate within the RemindoToets environment.

20. *Questiontype*

Finally the question type is one of eleven options: choice, combined, extended_text, graphic_associate, inline choice, match, order, position_object,select_point, text_entry and upload. For our analysis we have flattened this list of eleven into three types. Multiple choice, text and other. In the Multiple choice type we find the choice type combined with the inline choice with only one inline choice box within a question. Within the text type we find exclusively extended_text type. The other type is populated using the rest of the types. In our analyses the other type questions are not used for calculating totalscores.

1.2 Session information

version	R version 4.2.1 (2022-06-23 ucrt)
os	Windows 10 x64 (build 22621)
system	x86_64, mingw32
ui	RStudio
language	(EN)
collate	English_Netherlands.utf8
ctype	English_Netherlands.utf8
tz	Europe/Berlin
date	2023-02-08
rstudio	2022.07.2+576 Spotted Wakerobin (desktop)
pandoc	NA

Table 1: R version information R Core Team, 2022

package	ondiskversion	loadedversion	date	source
DBI H. Wickham and Müller, 2022	1.1.3	1.1.3	2022-06-18	CRAN (R 4.2.1)
devtools H. Wickham and Hester, 2022	2.4.5	2.4.5	2022-10-11	CRAN (R 4.2.2)
dplyr H. Wickham and François, 2022	1.0.10	1.0.10	2022-09-01	CRAN (R 4.2.2)
forcats H. Wickham, 2022a	0.5.2	0.5.2	2022-08-19	CRAN (R 4.2.2)
ggplot2 H. Wickham and Chang, 2022	3.3.6	3.3.6	2022-05-03	CRAN (R 4.2.1)
plyr H. Wickham, 2022b	1.8.7	1.8.7	2022-03-24	CRAN (R 4.2.1)
purrr Henry and Wickham, 2022a	0.3.5	0.3.5	2022-10-06	CRAN (R 4.2.1)
readr H. Wickham et al., 2022	2.1.3	2.1.3	2022-10-01	CRAN (R 4.2.2)
rlang Henry and Wickham, 2022b	1.0.6	1.0.6	2022-09-24	CRAN (R 4.2.1)
RMySQL Ooms and James, 2021	0.10.23	0.10.23	2021-12-14	CRAN (R 4.2.1)
stringr H. Wickham, 2022c	1.4.1	1.4.1	2022-08-20	CRAN (R 4.2.1)
tibble Müller and Wickham, 2022	3.1.8	3.1.8	2022-07-22	CRAN (R 4.2.1)
tidyr H. Wickham and Girlich, 2022	1.2.1	1.2.1	2022-09-08	CRAN (R 4.2.1)
tidyverse H. Wickham, 2022d	1.3.2	1.3.2	2022-07-18	CRAN (R 4.2.2)
usethis H. Wickham and Bryan, 2022	2.1.6	2.1.6	2022-05-25	CRAN (R 4.2.2)
xtable Dahl and Scott, 2019	1.8.4	1.8-4	2019-04-21	CRAN (R 4.2.2)

Table 2: R package information, as used for the data exploration

1.3 R code for the plots within the data exploration

```
##################################################### DATA EXPLORATION ##########################################
library(DBI)
library(RMySQL)
library(rlang)
library(tidyverse)
library(plyr)
library(tidyr)
library(ggplot2)
library(ggExtra)
library(plotly)
library(gapminder)
library(reshape2)
library(rjson)
library(ltm)
library(rlist)
library(svMisc)
library(dexter)
library(digest)
library(knitr)
library(here)
library(xtable)
library(devtools)
library(lubridate)
```

```
library(mirt)
library(data.table)

#load password under pass var
source("somewhere I keep my passwords")

#load in the mariadb
con <- dbConnect(RMariaDB::MariaDB(), "UUtests2", password = pass, user = "root")

#load in the data from the Mysql database
vspr<- dbSendQuery(con,"SELECT
    sequence_index as 'Vraag nr.',
    item_result.recipe_id as 'Toets ID',
    item_result.id as 'Antwoord ID',
    item_result.result_id as 'Result ID',
    item_result.item_identifier as 'Vraag+versie ID',
    item_result.section_id as 'sectionID',
    item_result.user_id as 'User ID',
    datestamp as 'Datum/tijd laatste aanpassing kandidaat',
    duration as 'Hoe lang is de kandidaat bezig met vraag',
    completion_status as 'Antwoord-status', # Kan zijn: completed, not_attempted, unknown, incomplete
    CAST(JSON_VALUE(metadata, '$.guess_score') AS float) as 'Raadscore',
    score, # Som van de item_result_variable scores
    item_result.max_score as 'Maximum score', # wat is het mx aantal punten wat je kan halen
    passed as 'Heb je hem 100% correct ja/nee',
    weight as 'Gewicht',
    check_manually as 'Moet handmatig nagekeken worden',
    dontknow as 'Kandidaat heeft aangegeven het antwoord niet te weten',
    assessmentitem_cache.xml as 'QTI XML',
    assessmentitem_cache.metadata as 'Overige metadata', # zit ook vraagtypen in verborgen
    JSON_VALUE(metadata, \"$.type\") as 'typevraag',
    JSON_VALUE(metadata, \"$.qtidata.RESPONSE-1\") as 'meerineen',
    item_result.recipe_id

FROM `item_result`
    INNER JOIN assessmentitem_cache ON (item_result.item_identifier = assessmentitem_cache.item_identifier)
WHERE
 `assessmentitem_cache`.`metadata` NOT LIKE '%RESPONSE-1%' AND (JSON_VALUE(metadata, \"$.type\") IN (\"choice\", \"extend
  #AND (item_result.recipe_id  IN (14899,16378,23707,27897,28947,30069,35130,35635,36365,52103))
#GROUP BY
#        assessmentitem_cache.item_identifier

#LIMIT 10000
;                                ")

vragenenscoresperresultaat <-dbFetch(vspr)

#make a copy before we adjust anything
savedset = vragenenscoresperresultaat
```

```
# filter out statusses that do not yield accurate scores
vragenenscoresperresultaat = subset(vragenenscoresperresultaat, vragenenscoresperresultaat$`Antwoord-status` != 'un
vragenenscoresperresultaat = subset(vragenenscoresperresultaat, is.na(vragenenscoresperresultaat$sectionID))

#replace inline_choice for choice, this is correct because within the sequel query we already filter out all inline
vragenenscoresperresultaat$typevraag[vragenenscoresperresultaat$typevraag == "inline_choice"] = "choice"

#show type distribution
typedistribution = vragenenscoresperresultaat %>% dplyr::count(vragenenscoresperresultaat$typevraag)

#addtotalscorecandidate and totalscoretest for each answer for both types
vragenenscoresperresultaatplustotaal = vragenenscoresperresultaat %>% group_by(`Toets ID`, `Result ID`,  `User ID`
vragenenscoresperresultaatplustotaalchoice = vragenenscoresperresultaatplustotaal %>% group_by(`Toets ID`, `Result
vragenenscoresperresultaatplustotaaltext = vragenenscoresperresultaatplustotaal %>% group_by(`Toets ID`, `Result ID

#add the two types back together
vragenenscoresperresultaatplustotaal = rbind(vragenenscoresperresultaatplustotaalchoice, vragenenscoresperresultaat

#calculate typescores
mettype = vragenenscoresperresultaatplustotaal %>% group_by(`Toets ID`, `Result ID`,  `User ID`, `typevraag`) %>%

#drop all entries that are the same so one typescore and total score row per examinee per tests stays
mettype <- mettype %>%
  distinct(`Toets ID`, `Result ID`,  `User ID`,`typevraag`, .keep_all = TRUE)

#pivot results into one row
lijstvantype = mettype %>% pivot_wider(id_cols = `Result ID` ,names_from= typevraag, values_from = typescore)%>% dp

#filter out all of the tests that do not have two types of answers
typeABscatter = lijstvantype %>% filter(!is.na(lijstvantype$choice))%>% na.omit() %>% mutate(totalscore = choice +

#keep one row per test

selection2 = mettype %>% dplyr::select(`Result ID`, `Toets ID`, `totalscoretest`, `maxmpscore`, `maxtextscore`, `

#add pre existing variables backm such as max scores and totalscores
typeABscatter = left_join(typeABscatter, selection2, by= "Result ID")

#again filter out all tests that do not have two different types of questions
typeABscattertypeA =typeABscatter %>% filter(!is.na(maxtextscore)) %>% dplyr::select(-maxmpscore)
typeABscattertypeB =typeABscatter %>% filter(!is.na(maxmpscore)) %>% dplyr::select(`Result ID`,maxmpscore)

typeABscatter = inner_join(typeABscattertypeA, typeABscattertypeB, by = "Result ID")

#calc freq tables for both users and tests
answerspertest = vragenenscoresperresultaat %>% group_by(`Toets ID`) %>% dplyr::summarize(Freq=n())
answerspertestperuser = vragenenscoresperresultaat %>% group_by(`Toets ID`, `User ID`) %>% dplyr::summarize(Freq=n

#filter out tests with a lower score than 2 because they cant tell us much about balance
```

```
typeABscatter = typeABscatter%>% filter(totalscore >2)

#add percentages
typeABscatter = typeABscatter %>% mutate(percentagetextscore = extended_text / totalscore)
typeABscatter = typeABscatter %>% mutate(percentagempscore = choice / totalscore)
#typeABscatter = typeABscatter %>% mutate(percentageotherscore = otherscore / totalscore)
#add normalized versions relative to each scoretype
typeABscatter = typeABscatter  %>% mutate(normextended_text = extended_text/maxtextscore)
typeABscatter = typeABscatter  %>% mutate(normchoice = choice/maxmpscore)
#typeABscatter = typeABscatter %>% mutate(normotherscore = otherscore / mean(otherscore))

#add normalized versions relative to each test
typeABscatter = typeABscatter  %>% mutate(normextended_texttest = extended_text/totalscoretest)
typeABscatter = typeABscatter  %>% mutate(normchoicetest = choice/totalscoretest)
typeABscatter = round(typeABscatter, digits = 3)

#here we use a predifined saver function to save the ggplots with a date
saver = function(name, ggplotje, width = 7)
{

    # add date to snure we backup automatically on system
  ggsave(filename = paste0(format(format(Sys.time(), "%m%d_%H%M")) , name),  path = "somewhere", plot = ggplotje, width =

  # export VERSION SAVE NODATE

}

#create all scatter and hexbin plots and save them
saver("typeecdf.png",ggplot(typeABscatter) + stat_ecdf(aes(x = choice, colour = "Multiple choice"), geom="step",pad = FALS

saver("typedensity.png",ggplot(typeABscatter) + stat_density(aes(x = choice, colour = "Multiple choice"), geom="line", sho


p1 = ggExtra::ggMarginal(ggplot(typeABscatter, aes(x = choice, y = extended_text)) + geom_point(alpha = 0.1) + geom_smooth
saver("scatterrugtypeunnormnolimit.png", p1)

p2 = ggExtra::ggMarginal(ggplot(typeABscatter, aes(x = choice, y = extended_text)) + geom_point(alpha = 0.1) +ylim(0,60) +
saver("scatterrugtypeunnorm60limit.png", p2)

p3 = ggExtra::ggMarginal(ggplot(typeABscatter, aes(x = choice, y = extended_text)) + geom_point(alpha = 0.1) +ylim(0,35) +
saver("scatterrugtypeunnormzoomin.png", p3)

p4 = ggExtra::ggMarginal(ggplot(typeABscatter, aes(x = choice, y = extended_text)) + stat_binhex(alpha = 0.6) +theme(legen
saver("binhexnozoom.png",p4)

p5 = ggExtra::ggMarginal(ggplot(typeABscatter, aes(x = choice, y = extended_text))  +ylim(0,60) + xlim(0,60) + stat_binhex
saver("binhexzoomout.png",p5)

p6 = ggExtra::ggMarginal(ggplot(typeABscatter, aes(x = choice, y = extended_text))  +ylim(0,35) + xlim(0,35) + stat_binhex
```

```
saver("binhexzoomin.png",p6)

#normalized on totaltypescorepertest

p7 = ggExtra::ggMarginal(ggplot(typeABscatter, aes(x = normchoice, y = normextended_text)) + geom_point(alpha = 0.1

saver("normescatter.png", p7) #ggplot(typeABscatter, aes(x = normchoice, y = normextended_text)) + geom_point() +yl

p8 = ggExtra::ggMarginal(ggplot(typeABscatter, aes(x = normchoice, y = normextended_text))  +ylim(0,1) + xlim(0,1)

saver("normhexbin.png",p8) #ggplot(typeABscatter, aes(x = normchoice, y = normextended_text))  +ylim(0,1) + xlim(0,

saver("normbin.png",ggplot(typeABscatter, aes(x = normchoice, y = normextended_text))  +ylim(0,1) + xlim(0,1) + geo

#normalized on total score per test


#percentage of totalscore on totalscore
saver("testnormescatter.png",ggplot(typeABscatter, aes(x = normchoicetest, y = normextended_texttest)) + geom_point

saver("testnormhexbin.png",ggplot(typeABscatter, aes(x = normchoicetest, y = normextended_texttest))  +ylim(0,1) +


# #normalized
ggplot(typeABscatter, aes(x = normchoice, y = normextended_text)) + geom_point() + geom_smooth(method="lm")

ggplot(typeABscatter, aes(x = percentagempscore, y = percentagetextscore)) + geom_point() +ylim(0,1) + xlim(0,1) +


####################################plot size of types of questions per question(small) ##########################

#load in the question types and question identifiers
vragenplustype = dbSendQuery(con, "SELECT * from `vragenplustypenn`;")
vragenplustypes = dbFetch(vragenplustype)

#frequency table of types
freqtableoftypes = vragenplustypes %>% dplyr::count(QuestionType)
totaal = sum(freqtableoftypes$n)
freqtableoftypes = freqtableoftypes %>% mutate(percentage = round(n/totaal, digits = 3))

#plotting the frequency table for questions
c4 = c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K")
saver("questiontypes.png",ggplot(freqtableoftypes, aes(x=as.character(`QuestionType`), y=`n`, fill = c4, label = `
  geom_bar(stat = "identity", width = 0.8) +
  theme(legend.position="none")+
  ylab("Number of questions")+
  xlab("Type of questions")+
  geom_text(nudge_y= -.01,
```

```
            color="Black",
            size = 4,
            vjust = -0.3,
            fontface="bold"),11)

#plot size of types of questions for answers(big) -> meaning actual filled in question results
freqtableoftypes = vragenenscoresperresultaat %>% dplyr::count(typevraag)
totaal = sum(freqtableoftypes$n)
freqtableoftypes = freqtableoftypes %>% mutate(percentage = round(n/totaal, digits = 3))
c4 = c("A", "B")
saver("answertypes.png", ggplot(freqtableoftypes, aes(x=as.character(`typevraag`), y=`n`, fill = c4, label = `percentage`
  geom_bar(stat = "identity", width = 0.6) +
  #scale_fill_brewer(palette = "Set1") +
  theme(legend.position="none")+
  ylab("Number of questions")+
  xlab("Type of questions")+
  scale_y_continuous(labels = scales::comma)+
  geom_text(nudge_y= -.01,
            color="Black",
            size = 4,
            vjust = -0.3,
            fontface="bold"))

svrdyvragen = vragenenscoresperresultaatplustotaal%>% filter(as.character(Datelastchanged) != "-001-11-30 UTC")

#add as month
vragenenscoresperresultaatplustotaal = vragenenscoresperresultaatplustotaal%>% mutate(Year = as.integer(format(as.Date(`Da
vragenenscoresperresultaatplustotaal = vragenenscoresperresultaatplustotaal %>% filter(Year != -1)
print(xtable(summary(dplyr::select(vragenenscoresperresultaatplustotaal, c("Hoe lang is de kandidaat bezig met vraag","Raa

# plot histogram of occurrences per year
saver("testsperyear.png",ggplot(vragenenscoresperresultaatplustotaal, aes(x=Year)) +
  geom_histogram(binwidth=.5, colour="black", fill="steelblue") +
  scale_y_continuous(labels = scales::comma) )


library(scales)
saver("testspermonth.png",ggplot(vragenenscoresperresultaatplustotaal, aes(x=Month)) +
        geom_histogram(binwidth=.5, colour="black", fill="steelblue") +
        scale_x_continuous(n.breaks = 12) + ylab("Number of answers given"))

#score per month
vragenenscoresperresultaatplustotaal = vragenenscoresperresultaatplustotaal %>% mutate(normscore = case_when(score == 0 ~
                                                                                                              score !=0 ~sc

meanscorepermonth = vragenenscoresperresultaatplustotaal %>% group_by(Month) %>% summarise_at(vars(normscore), list(name =

saver("meanscorepermonth.png",ggplot(meanscorepermonth, aes(x=Month, y = name)) +
        geom_bar(stat = "identity", colour="black", fill="steelblue") + ylab("Mean score") + geom_smooth(method="lm"))
```

```
#score per year
meanscoreperyear = vragenenscoresperresultaatplustotaal %>% group_by(Year) %>% summarise_at(vars(normscore), list(n

saver("meanscoreperyear.png",ggplot(meanscoreperyear, aes(Year, y = name, label = round(`name`,3))) +
        geom_bar(stat = "identity", colour="black", fill="steelblue") + ylab("Mean score") + geom_smooth(method="lm
        geom_text(nudge_y= -.01,
                  color="Black",
                  size = 4,
                  vjust = -0.3,
                  fontface="bold"),11)

#create list of all tests
lijstvantoetsen <- as.data.frame(table(vragenenscoresperresultaat$`Toets ID`))
lijstvantoetsen <- as.data.frame(table(vragenenscoresperresultaat$`Toets ID`))
Answerspertestplot = ggplot(lijstvantoetsen, aes(Freq)) + geom_histogram()
print(Answerspertestplot)

#delete the max value because it was a corona questionaire and not a test
flijstvantoetsen = filter(lijstvantoetsen, `Var1` !=13375)

#ecdfplot from tests on amount of results per tests
ecdfplot = ggplot(data=flijstvantoetsen, mapping = aes(x=Freq))+stat_ecdf(geom = 'step')
```

2 Methods

2.1 MySQL code

```
CREATE TABLE filteredanswers SELECT
    sequence_index,
    item_result.recipe_id,
    item_result.id,
    item_result.result_id,
    item_result.item_identifier,
    item_result.section_id,
    item_result.user_id,
    datestamp,
    item_result.duration,
    completion_status, # Possible values: completed, not_attempted, unknown, incomplete
    CAST(JSON_VALUE(metadata, '$.guess_score') AS float) as 'GuessScore',
    item_result.score, # Sum of item result variable scores
    item_result.max_score, # Max score possible on question
    passed,
    weight,
    check_manually,
    dontknow,
    item_type,
```

```
    item_result.recipe_id,
    COALESCE(item_result_variable.value_float, item_result_variable.value_int,
    item_result_variable.value_string, item_result_variable.value_text) as 'interaction
↪   answer', #Because answers are stored in a different table, we need to gather the answers
↪   from this table, making this part of the query a bit hard to read.
    JSON_VALUE(metadata, '$.responseDeclarations.RESPONSE.cardinality') AS cardinality,
    result.recipe_type AS ExamType,
    studievakken.code,
    studievakken.name
FROM `item_result`
    INNER JOIN assessmentitem_cache ON (item_result.item_identifier =
↪   assessmentitem_cache.item_identifier AND item_type IN ('extended_text','choice',
↪   'inline_choice' ))  # IUnline choice is acceptable when there is only one inline choice
↪   element.
    LEFT JOIN item_result_variable ON (item_result_variable.item_result_id = item_result.id
↪   AND item_result_variable.type = 'response')
    LEFT JOIN result ON (item_result.recipe_id = result.recipe_id)
    LEFT JOIN studievakken ON (item_result.recipe_id = studievakken.recipe_id)
WHERE # Here, we filter out any answers which have not been completed, any answers on tests
↪   that have sectioning and any question which consists of multiple elements.
completion_status != 'unknown' AND item_result.section_id IS NULL AND JSON_VALUE(metadata,
↪   '$.responseDeclarations.RESPONSE.cardinality') != 'multiple'
;
```

3 Results

3.1 R code for the plots within the results

```
library(DBI)
library(RMySQL)
library(rlang)
library(tidyverse)
library(plyr)
library(tidyr)
library(ggplot2)
library(ggExtra)
library(plotly)
library(gapminder)
library(reshape2)
library(rjson)
library(ltm)
library(rlist)
library(svMisc)
library(dexter)
library(digest)
library(knitr)
library(here)
library(xtable)
```

```
library(devtools)
library(lubridate)
library(mirt)
library(data.table)

#load password under pass var
source("somewhere I keep my passwords")

#load in the mariadb
con <- dbConnect(RMariaDB::MariaDB(), "UUtests2", password = pass, user = "root")

#import the saved mysql table of all question answers
getkeys = dbSendQuery(con,"SELECT * from `filteredanswers`")
keys = dbFetch(getkeys)

#import the userlanguages per userid
getuserlanguage =dbSendQuery(con, "SELECT
        user.`language`,
        user.id
from `user`")
userlanguage = dbFetch(getuserlanguage)

#filter out  only completed statusses as the rest likely wont have a score
keys2 = keys %>% filter(CompletionStatus == "completed")

#add a column containing the faculty
keys2 = keys2%>% mutate(faculty = str_split(studycode, "\\-| |\\_", simplify = TRUE)[,1])

#save a copy
keystobedistributed = keys

#add weighted scores
keys2 = keys2 %>% mutate(Wscore = Weight * score)
keys2 = rename(keys2, "item_id" = QID, "item_score" = Wscore)

#create scores as answers for all text based questions to prepare for dexter
keyswithanswers = keys2 %>%
  mutate(response = case_when(QuestionType == "choice" | QuestionType == "inline_choice" ~
  ↪  `interaction answer`,
                              QuestionType == "extended_text" ~ as.character(`item_score`)))

#filter out what we need for dexter for the rules
keyrules = keyswithanswers %>% group_by(item_id) %>%  distinct(response, .keep_all = TRUE)%>%
↪  dplyr::select(item_id, response, item_score)

#make integers from all of the scores as they were int64 which dexter did not support (I think
↪  it does now)
keyrules$item_score = as.integer(keyrules$item_score)
```

```
#filter out all itemkeys that would cause an error within dexter
keyrules = keyrules%>% group_by(item_id) %>% filter(any(item_score == 0)) %>% ungroup()
keyrules = keyrules%>% group_by(item_id) %>% filter(n()>=2) %>% ungroup()
keyrules = keyrules%>% group_by(item_id) %>% filter(n_distinct(item_score)>=2) %>% ungroup()

#start the dexter project
db = start_new_project(keyrules,":memory:")

#create  alist of all items that occur within the keys
filteredbyrulesanswers = semi_join(keyswithanswers,keyrules, by = "item_id")

#add weightedscore
filteredbyrulesanswers = filteredbyrulesanswers %>% mutate(Wscore = Weight*score)

#add language for DIF
filteredbyrulesanswers = inner_join(filteredbyrulesanswers, rename(userlanguage, UserID =
↪  "id"), by = "UserID")

#change inline choice to choice type because we filtered all inline choice that are actually
↪  multiple choice types out earlier
filteredbyrulesanswers$QuestionType[filteredbyrulesanswers$QuestionType == "inline_choice"] =
↪  "choice"

#filter distinct values
filtrba = filteredbyrulesanswers %>% dplyr::select(UserID, TestID, item_id, response) %>%
↪  rename("person_id" = UserID, "booklet_id" = TestID) %>% distinct(person_id, booklet_id,
↪  item_id, .keep_all = TRUE)

#create a design, a dataframe that has sets of items and which booklet they belong to
design_data = filtrba %>% group_by(booklet_id) %>% distinct(item_id) %>%
↪  dplyr::select(booklet_id, item_id)

#load in the repsonses into dexter
add_response_data(db, filtrba, design = design_data, auto_add_unknown_rules = TRUE)

#create a dataframe containing for every item what type of item it is
typeofquestion = keyswithanswers %>% dplyr::select(item_id, QuestionType) %>% distinct()
typeofquestion$QuestionType[typeofquestion$QuestionType == "inline_choice"] = "choice"

#add the type per question to dexter
add_item_properties(db, item_properties = typeofquestion, default_values = NULL)

#rename the languageproperties for language and create a dataframe that has for each person
↪  their preffered language
personproperties = filteredbyrulesanswers %>% dplyr::select(UserID, language) %>%
↪  rename("person_id" = UserID) %>% distinct()
personproperties$language[personproperties$language == "nl-NL"] = "dutch"
personproperties$language[personproperties$language == "en-US"] = "english"
```

```
#add these person properties to dexter
add_person_properties(db, personproperties)

#add info dfs for debugging
bookletsinfo = get_booklets(db)
itemsinfo = get_items(db)

#show how many tests there are per faculty
facultycounts = filteredbyrulesanswers %>% group_by(faculty)
↪  %>%summarise(Numberoffacultyresults = count(ResultID))


############################################## T Test ################################
#filter out what we want and split based on faculty and type
pairedttestonfacultyvstype = filteredbyrulesanswers %>% mutate(weightedpercentagescore =
↪  (score/MaxScore)) %>% split(filteredbyrulesanswers$faculty)

#create a function that takes a faculty and returns a typed welch t-test
pairttester = function(facultydf)
{
  t.test(weightedpercentagescore ~ QuestionType, data = facultydf)
}

#run for all faculties
test = lapply(pairedttestonfacultyvstype,pairttester)

#access the returned list
templist = lapply(test, function(x) x[["p.value"]] )
ttestdf =  as.data.frame(s)

#print results to a latex table
print(xtable(ttestdf, type = "latex"))

######################## MEAN SCORE ################################

#create meanscores per faculties that we will examine
meanscoresperfaculty = filteredbyrulesanswers %>% mutate(weightedpercentagescore =
↪  (score/MaxScore)) %>%
  group_by(faculty, QuestionType) %>%summarise(meanscore =
  ↪  mean(weightedpercentagescore),.groups = "keep") %>%
  filter(!(faculty %in% c("[REDACTED]")))%>% group_by(faculty, QuestionType)%>%
  ↪  arrange(desc(meanscore),.by_group=TRUE) %>% #%>% pivot_wider(names_from = QuestionType,
  ↪  values_from = meanscore)
group_by(QuestionType) %>% arrange(desc(meanscore), .by_group = TRUE) %>%ungroup() %>%
↪  mutate(rownr = as.factor(row_number()))

#set the factor so the plot will be ordered
meanscoresperfaculty = meanscoresperfaculty %>% mutate(faculties = factor(faculty, levels =
↪  faculty))
```

```
library(forcats)
meanscoresperfaculty = meanscoresperfaculty %>% mutate(name = fct_reorder(faculty,
↪  desc(rownr)))

#create plot for mean score per factulty
orderedmeanscore = ggplot(meanscoresperfaculty, aes(x = rownr, y = meanscore, fill =
↪  QuestionType)) +
  geom_bar(stat = "identity", position = "dodge") + xlab("Faculty")+ ylab("Mean score")

#save using same fucntion from data exploration
saver("orderedmeanscorenew.png", orderedmeanscore,width = 13)


######################### Proportional part#############################

#get all typescores and total scores for each result
alltypescores = filteredbyrulesanswers %>% group_by(ResultID) %>% summarise(textscore =
↪  sum(Wscore[QuestionType=="extended_text"]), choicescore =
↪  sum(Wscore[QuestionType=="choice"]), totalscore = textscore+choicescore)

#get all other background info on a test
testresultkoppel = filteredbyrulesanswers %>% dplyr::select(ResultID,TestID) %>% distinct()

#add all backgorund info to typescore and totalscores
typescoresplusinfo = inner_join(alltypescores,testresultkoppel,by ="ResultID")

#create dataframe containing all expected scores according to tests proportions
dexo = rbindlist(output)
dexo$booklet_id = as.integer(dexo$booklet_id)
dexo = pivot_wider(dexo,names_from = questiontype, values_from = expected_domain_score)%>%
↪  rename(maxchoice = "choice", maxtext = "extended_text")

#get all max scores according to DEXTER
maxscooresaccordingtodexter = dexo %>% group_by(booklet_id) %>% summarise_all(last)

#calculate max possible scores according to data
maxtotalscores = filteredbyrulesanswers %>% group_by(RecipeID, ResultID) %>%
↪  summarise(maxtotalscore = sum(MaxScore),  .groups = "keep") %>%
  ungroup()%>% group_by(RecipeID) %>% summarise(maxtotalscoreall = max(maxtotalscore))

#create typescores again
typescores = filteredbyrulesanswers %>% group_by(RecipeID, ResultID) %>% summarise(choicescore
↪  =sum(score[QuestionType == "choice"]), textscore = sum(score[QuestionType ==
↪  "extended_text"]), .groups = "keep") %>% ungroup() %>%
  rename(booklet_id = "RecipeID")

#filter out and add proportions -> for the proportional analysis
maxtypescoreswithactual = left_join(typescores, maxscooresaccordingtodexter, by= "booklet_id")
maxtypescoreswithactual = na.omit(maxtypescoreswithactual)
```

```
maxtypescoreswithactual = maxtypescoreswithactual %>% mutate(totalscore =
↪  choicescore+textscore, proportionmaxtext = maxtext/booklet_score, proportionmaxchoice =
↪  maxchoice/booklet_score, proportionchoice = choicescore/totalscore, proportiontext =
↪  textscore/totalscore, differencechoice =
↪  proportionmaxchoice-proportionchoice,differencetext = proportionmaxtext-proportiontext  )


######################################## IRT ENORM PART ########################

#create domain, which is something used to train the ENORM model
aangepastdomain = filteredbyrulesanswers %>% group_by(TestID) %>% filter(TestID == 9556) %>%
↪  rename(questiontype = "QuestionType")%>% ungroup() %>% dplyr::select(item_id,
↪  questiontype) %>% distinct()

#change choice types again ;)
aangepastdomain$questiontype[aangepastdomain$questiontype == "inline_choice"] = "choice"

#check how many types per type of question
goedeverdeling = filteredbyrulesanswers %>% group_by(TestID, QuestionType) %>%
↪  rename(questiontype = "QuestionType")%>%
  dplyr::select(TestID, item_id, questiontype) %>% summarise(countpertype = n())

#calculate scores per question type for each result
domainscores = filteredbyrulesanswers %>% group_by(TestID, ResultID) %>%
↪  dplyr::summarize(choicescore = sum(Wscore[QuestionType == "choice"]), extendedtextscore =
↪  sum(Wscore[QuestionType == "extended_text"]) ) %>% mutate(totalscore =
↪  choicescore+extendedtextscore)

#function that goes over a test and tries to calculate ENORM, if it fails returns NULL
maakprofieldingvoortest = function(bookletid)
{tryCatch(
    expr = {
    print(bookletid)
    enorm = fit_enorm(db, booklet_id == bookletid)
    #plot(enorm)
    aangepastdomain = filteredbyrulesanswers %>% group_by(TestID) %>% filter(TestID ==
    ↪  bookletid) %>% rename(questiontype = "QuestionType")%>% ungroup() %>%
    ↪  dplyr::select(item_id, questiontype) %>% distinct()
    aangepastdomain$questiontype[aangepastdomain$questiontype == "inline_choice"] = "choice"
    profielding = profile_tables(parms = enorm, domains = aangepastdomain, item_property
    ↪  ="questiontype")
    return(profielding)
},
error = function(e){
  return(NULL)
})
}

#get all unique booklet id's within dexter
```

```
h = unique(bookletsinfo$booklet_id)

#Fit an ENORM for all unique tests - takes about 4 hours for a little under 7000 tests (about
↪  10 million answers)
output = lapply(h, function(x) maakprofieldingvoortest(x))

#save and load results because we dont want to keep having to run this
save(output, file = "expectdag.Rdata")
load("expectdag.Rdata")

# create new list of all enorm results
lv = rbindlist(output)
lv$booklet_id = as.integer(lv$booklet_id)

#create a list containing all og
lv = pivot_wider(lv,names_from = questiontype, values_from = expected_domain_score ) %>%
↪  rename(expectedchoicescore = "choice", expectedtextscore = "extended_text")

#lv = lv %>% group_by(booklet_id,booklet_score) %>% summarise(expected_text_score =
↪  expected_domain_score[questiontype=="extended_text"],expected_choice_score =
↪  expected_domain_score[questiontype=="choice"] )
typescoresplusinfo = typescoresplusinfo %>% rename(booklet_score = "totalscore", booklet_id =
↪  "TestID")

#check for every test result what the expected enorm result would be
deviationprofiles = typescoresplusinfo %>% merge(lv, by = c("booklet_id", "booklet_score"))
↪  %>%
  mutate(deviationchoice = choicescore - expectedchoicescore  , deviationtext =  textscore
  ↪  -expectedtextscore )

#save the rows of deviationprofiles
save(deviationprofiles, file = "deviationprofiles.Rdata")

#get studies and testids in a table
studytable = filteredbyrulesanswers %>% dplyr::select(c("faculty","ResultID")) %>% distinct()

# add studies to all results from ENORM table
deviationprofiles = merge(deviationprofiles,studytable, by = "ResultID")
deviationproffiltered = deviationprofiles %>% na.omit()

#calculate means per faculty per type
meandevperfaculty = deviationproffiltered %>% group_by(faculty) %>% summarise(meantextdev =
↪  mean(deviationtext,na.rm=TRUE), meanchoicedev = mean(deviationchoice, na.rm=TRUE))

#pivot for plotting
longermeandevperfac = meandevperfaculty %>% arrange(desc(meantextdev)) %>%
↪  pivot_longer(!faculty, names_to = "type", values_to = "mean")
library("forcats")
```

```
#add interaction for ordering
longermeandevperfac$int = interaction(longermeandevperfac$type, longermeandevperfac$faculty)

#lock in ordering
longermeandevperfac$int = factor(longermeandevperfac$int, levels = longermeandevperfac$int)

#plot result per type
plotjedev = ggplot(longermeandevperfac, aes(x = int , y = mean, fill = type)) +
  geom_bar(stat = "identity",
           show.legend = TRUE) +
  geom_text(aes(label = round(mean, 3),
                hjust = ifelse(mean < 0, 1.5, -1),
                vjust = 0.5),
            size = 3) +
  xlab("Faculty") +
  ylab("Mean deviation profiles of choice type") +
  coord_flip()

#save plot
saver("meandeviation1103.png",plotjedev, width = 17)

############################## IRT based proportional part


#create dataframe which has deviationprofiles, maxscores and achieved scores
enormplusmax = left_join(maxscoresaccordingtodexwithbookletid, deviationproffiltered, by =
↪  "booklet_id")

#calculate the proportional deviation score based on comaprison between ENORM expected score
↪  and achieved score
proprtionalirtdev = enormplusmax %>%
  mutate(proportionactualtext = textscore/maxscore, proportionactualchoice =
  ↪  choicescore/maxscore, proportionexpectedtext = expectedtextscore/maxscore,
  ↪  proportionexpectedchoice = expectedchoicescore/maxscore,
        deviationprofilechoiceprop = proportionactualchoice -proportionexpectedchoice,
        ↪  deviationprofiletextprop = proportionactualtext - proportionexpectedtext
  )

#remove values for which no proportion oculd be calculated
proportionalwithoutna = na.omit(proprtionalirtdev)


#calculate means per faculty
meanpropdev = proportionalwithoutna %>% group_by(faculty) %>% summarise(meantextdev =
↪  mean(deviationprofiletextprop,na.rm=TRUE), meanchoicedev =
↪  mean(deviationprofilechoiceprop, na.rm=TRUE))

#pivot for plotting
```

```
longermeanpropdev = meanpropdev %>% arrange(desc(meantextdev)) %>% pivot_longer(!faculty,
↪  names_to = "type", values_to = "mean")

library("forcats")

#add interaction for ordering
longermeanpropdev$int = interaction(longermeanpropdev$type, longermeanpropdev$faculty)

#lock in ordering
longermeanpropdev$int = factor(longermeanpropdev$int, levels = longermeanpropdev$int)

#plot
plotjedev = ggplot(longermeanpropdev, aes(x = int , y = mean, fill = type)) +
  geom_bar(stat = "identity",
           show.legend = TRUE) +
  geom_text(aes(label = round(mean, 3),
                hjust = ifelse(mean < 0, 1.5, -1),
                vjust = 0.5),
            size = 3) +
  xlab("Faculty") +
  ylab("Mean deviation profiles of choice type in percentages") +
  coord_flip()

#save plot
saver("meandeviationprop.png",plotjedev, width = 17)


############################### DIF PART ################################

#function that tries to calculate DIF for a single test, if it fails returns NULL
maakdifcalcs = function(bookletid)
{tryCatch(
  expr = {
    difresult = dexter::DIF(db, person_property = "language", booklet_id==bookletid)
    chi = difresult[[1]][[1]]
    df = difresult[[1]][[2]]
    prob = difresult[[1]][[3]]
    matr = matrix(c(bookletid, chi, df,prob), nrow = 1, ncol = 4)
    colnames(matr) =  c("booklet_id","chi","df","p")
    return(matr)
  },
  error = function(e){
    return(NULL)
  })
}

#get all unique booklet ID's
h = unique(bookletsinfo$booklet_id)
```

```
#apply DIF function to calculate DIF for all tests
difs = lapply(h, function (x) maakdifcalcs(x))

#delete NULLS
diffies = difs %>% discard(is.null)

#make a nice and tify Dataframe
diffies = lapply(diffies, function (x) as.data.frame(x))
difchiresults = rbindlist(diffies)

#save results of DIF calcs --- This calculation took about 2 hours, so we do not want to lose
↪  it somehow
save(difchiresults, file = "difchiresults.Rdata")

#load results of DIF calcs
load("difchiresults.Rdata")

#add faculties to a df with Test ID' ,which is booklet id
testsandfaculties = filteredbyrulesanswers %>% summarise(TestID, faculty) %>% distinct() %>%
↪  rename(booklet_id = "TestID")
testsandfaculties$booklet_id = as.character(testsandfaculties$booklet_id)

#merge the table above to add faculties for each DIF analysed test
intdif = difchiresults
difandfaculty = merge(intdif,testsandfaculties, by = "booklet_id")
difandfaculty$p = as.numeric(difandfaculty$p)

#apply holm method to the resulting p values
difandfaculty$adjustedp = p.adjust(difandfaculty$p, method = "holm")

#create boxplots per faculty
difperfac = ggplot(difandfaculty, aes(x = faculty, y = p)) +            # Applying ggplot
↪  function
  geom_boxplot()

#save plots
saver("difperfac1103.png", difperfac, width = 11)


difperfacholm = ggplot(difandfaculty, aes(x = faculty, y = adjustedp)) +            # Applying
↪  ggplot function
  geom_boxplot()

#save plots corrected for holm
saver("difperfac1103.png", difperfacholm, width = 11)

# check how many we have in total
filtereddifandfaculty = difandfaculty %>% filter(adjustedp<0.05)
```