What is the comparative accuracy of IRT, linear regression, and XGBoost in predicting students' performance on different versions of an exam?

Merlin T. Urbanski

Department of Psychology, University of Twente

Bachelorthesis Research Methodology, Measurement, Data-analysis, 202000379

First Supervisor: Dr. Stéphanie van den Berg

Second Supervisor: Dr. Maryam Amir Heari

3rd of July 2023

Abstract

Using multiple exam versions to avoid cheating in the form of copying and teamwork is common practice at universities and schools. To ensure fairness during the grading process in many cases the versions share common items that are then used for the harmonization of the two versions. In this thesis, a comparative analysis was conducted to assess the accuracy of three harmonization methods: Item Response Theory (IRT), Linear Regression, and XGBoost. A fictional scenario was created involving 200 students who took an exam with two versions containing 20 exclusive items and 10 shared items. Two different data sets, one simulated and the other based on real data, were used for the analysis. The results indicated that IRT performed the best in terms of accuracy for both data sets. Linear regression was found to be the second most accurate method, while XGBoost showed comparatively lower accuracy. The methods used in this study were much simpler compared to other research, so it is important to interpret the results with caution.

Introduction

During educational exams, it is not uncommon for students to engage in behaviours such as copying answers from their peers or engaging in forbidden teamwork. To prevent cheating attempts, a common practice is to create different versions of the exam. These versions typically contain the same or similar content but are arranged in different ways. This helps to ensure that each student receives a unique exam experience, reducing the likelihood of copying and collaboration among students. By implementing different exam versions, educational institutions aim to maintain the integrity and fairness of the assessment process.

However, when two or more exam versions are used, it is hard to ensure that the difficulty of those is the same, resulting in concerns about fair grading. While it's understandable that two questions on the same topic can vary in difficulty, it is also crucial to recognize that even slight variations in questions can lead to significant differences in difficulty. For instance, a study by Schurmeier et al. (2010) analysing the data of more than 6100 students showed that even minor changes in question-wording can have a significant influence on students' performances. Moreover, other studies showed that other differences like the item order (Balch, 1989) or the positioning of the response options (Hohensinn & Baghaei, 2017) influence the students' performance. Understanding the impact of these differences on difficulty highlights the importance of striking a balance between preventing cheating and ensuring a fair grading process.

Nevertheless, in most cases using two or more exam versions is the most effective method to prevent cheating during exams because other countermeasures like more supervisors or more space between the test-takers are too expensive. However, when employing multiple test versions, it is crucial to ensure fairness during the grading process. To achieve this, three different design approaches are commonly utilized in tests for harmonization processes, each with its strengths and limitations.

The first approach is called Random Groups Design and is probably the simplest and the most commonly used design (Kolen & Brennan, 2014). There are two or more versions of a test with only version-exclusive items. While the overall topic of the test and the knowledge it aims to measure is the same there are no identical questions. The differences between exam versions can vary in strength, ranging from minor changes in wording to more substantial variations. However, it is important to note that each item is exclusive to its version. Next, a spiraling process is used to assign the versions to the test takers in an almost random manner. Typically, this is done in a way that ensures that test takers seated next to each other do not have the same version, thereby minimizing the potential for cheating. To conclude, in this design, there are two or more randomly equivalent groups working on two different versions.

While this design can be very effective against cheating, it also has its limitations when it comes to assessing differences between test takers of different versions. In Random Groups Design "the difference between group level performance on the two forms is taken as a direct indication of the difference in difficulty between the forms" (Kolen & Brennan, 2014, p.13). This means that if the mean total score of one version is, for example, 10 points higher, it can only be interpreted as that this version is easier rather than that the group working on this version performed better. This issue will often lead to unfairness in the grading process because in the end only the total scores will be compared.

The second approach aims to tackle this problem while also using test versions with only exclusive items. In Single Group Design either some, or all students would work on both versions in order to determine the difficulty level of each version (Kolen & Brennan, 2014). While this design allows comparability between the two versions it is also connected to a lot of effort by the students who have to write a test multiple times. Furthermore, this approach is vulnerable to biases because working on one version before the other could either result in fatigue or some form of training effect. While there are counterbalancing approaches, this approach is rarely used in practice for the harmonization of tests. Nevertheless, in some cases or other areas of data harmonization, this approach is used more often because fatigue and competitiveness play a less important role.

The final approach is known as the Common-Item Non-equivalent Groups Design. In this design, instead of having different versions for different test-takers, both versions contain a common set of items (anchor items) that allow for comparability between them (Kolen & Brennan, 2014). Next to a scenario where all test-takers sit in the same room at the same time, this design is also used when there are multiple test dates. While there are variations of this approach where the score of the anchor items does not contribute to the total score, this variation is not suitable for this study. Thus, a Common-Item Non-equivalent Groups Design with the anchor items contributing to the total score will be used as a central approach in this study.

While it is now clear which design approach is the most suitable for a school scenario it is still unclear how exactly the versions will be made comparable. Traditionally, two widely used methods for harmonizing data that are based on anchor items are Item Response Theory (IRT) and Linear Regression. These methods have dominated the field for many years (Kolen & Brennan, 2014). More recently, a promising machine learning-based approach known as

XGBoost has gained popularity as well. This research aims to find the most suitable method, by comparing methods' accuracy in order to answer the research question: "What is the comparative accuracy of IRT, linear regression, and XGBoost in predicting students' performance on different versions of an exam?" To investigate the comparative accuracy of IRT, Linear Regression, and XGBoost in predicting students' performance on different test versions, it is important to provide a brief overview of these three methods.

Item Response Theory

The first method, Item Response Theory (IRT), is a common statistical approach used for various kinds of data harmonization. Next to the harmonization of educational tests, this includes for example the harmonization of all kinds of surveys (Zhao et al., 2022) or other applications like for instance test development and adaptive testing. (Kolen & Brennan, 2014).

To be more specific, it describes the relationship between a correct response on an item and the underlying ability of the test-taker (González & Wiberg, 2017). This is done by using a function that models the probability of a correct response based on the test-takers' ability level and item parameters like difficulty and discrimination. Thus, it assumes that test-takers have different levels of ability, the so-called theta and that items vary in difficulty and discrimination. "Much of the power of IRT results from the fact that it explicitly models examinee responses at the item level, whereas, for example, the focus of classical test models and strong true score models is on responses at the level of test scores." (Kolen & Brennan, 2014, p.171).

Two advantages of Item Response Theory (IRT) are its ability to measure one or multiple dimensions of a latent construct and its versatility in analysing both dichotomous items and small ordinal variables, such as a Likert scale (Kim et al., 2017). However, the flexibility of IRT models stems from their reliance on robust statistical assumptions, which are unlikely to be perfectly satisfied in real testing contexts (Kolen & Brennan, 2014). Despite this limitation, there are available methods that adjust for potential violations of these assumptions.

There are three different types of IRT models, the one- (1PLM), two- (2PLM), and three-parameter logistic model (3PLM), that differ in the number of item parameters taken into account when calculating response probabilities. The 1PLM (Rasch model), which is used in this study, uses only the difficulty parameter, while the 2PLM includes discrimination and the 3PLM adds an extra parameter that describes a probability for guessing.

Linear Regression

The second method used in this study is linear regression. Generally, a linear regression analysis “employs a model that describes the relationships between the dependent variables and the independent variables in a simplified mathematical form.” (Schneider et al., 2010, p. 776). It is a widely used statistical method in various fields of science to examine correlations between variables due to its simplicity and meaningfulness (Döring, 2018).

Linear regression models not only have a descriptive function but also allow for the estimation of dependent variable values based on observed independent variables (Schneider et al., 2010). In the context of this study, this predictive function becomes particularly useful when attempting to predict the scores of test-takers on another version. By utilizing the parameters of the linear model, predicted values can be generated to make different exam versions comparable. Similar to Item Response Theory (IRT), various approaches have already been developed that use linear regression to harmonize data (Dorans et al., 2010).

Overall, linear regression offers a versatile and widely applicable tool for investigating relationships between variables and making predictions based on observed data. Its simplicity and meaningfulness make it a popular choice in scientific research across different disciplines and allow its usage for harmonisation purposes.

Extreme Gradient Boosting (XGBoost)

The last harmonization method is called extreme gradient boosting (XGBoost). Unlike the other methods, XGBoost is a popular machine learning algorithm used for a variety of tasks related to predicting. Within past years its popularity increased drastically by winning multiple competitions (Nielsen, 2016). Generally, scalable tree boosting algorithms work by creating a sequence of decision trees, where each tree tries to correct the errors of the previous tree. The trees are trained sequentially, with each subsequent tree trying to minimize the residual errors of the previous tree (Synced, 2017).

Compared to other tree boosting algorithms, XGBoost is exceptionally good at handling large amounts of data. It is faster than other methods due to its advanced techniques to process data efficiently and it also takes advantage of parallel and distributed computing to increase the learning process even more (Chen & Guestrin, 2016). On the other hand, XGBoost is also capable to handle sparse data and instances with varying weights, which makes it applicable to a wide range of tasks. For example, it has been used in the analysis of significant statistical features related to facial behavior in human depression (Rumahorbo et al., 2023). Although there have been some attempts to use XGBoost for data harmonization, such as its application in economic studies by Piechoka-Kaluzna (2021), there is a notable lack of research on utilizing XGBoost for harmonization purposes.

Expectations

Based on prior research by Petersen et al. (1983) it can be expected that IRT and the linear regression will be almost equally accurate with slight advantages for IRT depending on the data sets that are going to be used. It will be interesting to see how XGBoost will perform compared to the other methods as there is only a limited amount of research on how it performs on harmonization yet. One might think, given its success in various competitions, that XGBoost would outperform the other methods. However, it is still uncertain whether this will hold true in the context of harmonization purposes as this is a distinct and comparatively simpler field.

Methods

R-script

A complete version of the R-script that was used for preparation and analysis is provided in Appendix.

Scenario

To be as realistic as possible, this research paper used a hypothetical scenario that could occur similarly in any university worldwide. It included 200 test-taker writing an exam containing 30 dichotomous items that could either be answered right or wrong. 100 test-takers wrote exam version A, while the other half wrote version B. Both versions consisted out of 10 shared items, the anchor items, and 20 individual items. However, the content of all questions remained unknown as it was irrelevant for this kind of harmonization and to keep the scenario as simple as possible. This scenario was represented by two different data sets, the first one was completely simulated using a Rasch-Model while the second one was retrieved from the internet and based on real data.

For the prediction and the evaluation of the prediction's accuracy this research solely used parts of the data that would have been assessable in a real-world scenario. This allowed replication of this research in real scenarios like at universities or schools. Furthermore, this means that instead of predicting how a test-taker of version A performed on version B the total scores on existing items were predicted. To specify 70 version B test-takers and their performance on the anchor items were used to train the models. The models were trained to predict the test-takers' performance on the 20 exclusive items of version B based on their performance on the anchor items. In the next step, those models were used to predict the performance of the remaining 30 test-takers. Splitting an existing data set like this is common practice when evaluating the accuracy of models, as it generates the most realistic results (Gholamy et al., 2018). The predicted scores were then compared to the participants' actual scores to evaluate the accuracy of the methods.

Rasch-Model Data Simulation

To simulate the first dataset, a Rasch model was used, which involves two main parameters: the ability level of the test-takers (theta) and the difficulty level of each item (beta). Using a function that creates normally distributed values the thetas (M = 0.4, SD = 0.7) and betas (M = 0, SD = 1) were generated randomly. Based on those parameters the probability of a correct answer for each item can be computed using the following equation:

$$probability = \frac{(\theta - \beta)^2}{(1 + (\theta - \beta)^2)}$$

In this equation theta represents the test-takers' ability level, and beta represents the difficulty level of each item. Next the seed 2001 was set to ensure the replicability of this research using the same values. Next, based on the with the equation calculated probabilities, a data set was created. It contained values of all test-takers for every question of both exam versions and no missing values for any question. The value “False” showed a wrong answer, while the value “TRUE” indicated a correct answer of the test taker. In the next step the data set was prepared to be suitable for the scenario of this study.

Real Data

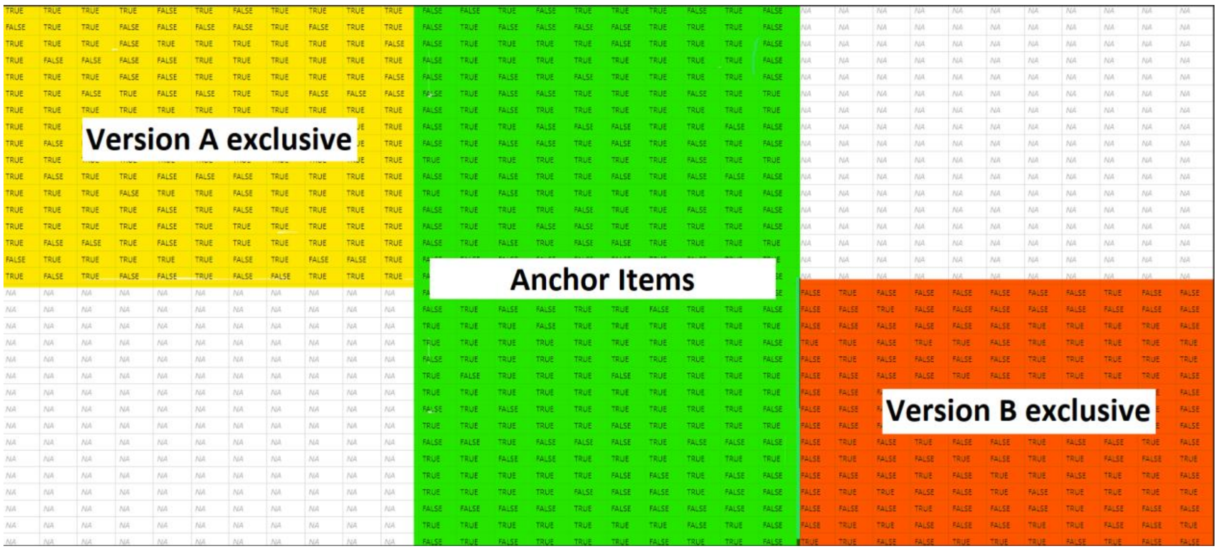
The second data set that was used in this study was, unlike the first data set, not simulated but based on real data. It includes information about a test that is administered twice a year for admission to universities and colleges and can be downloaded for free on the website of the book “Applying Test Equating Methods” by González & Wiberg (2017). In its original form it contains 10.000 observations on 160 multiple choice questions that are coded in zeros and ones. However, for this study only the first 200 observations and 50 items were used to ensure equality in our scenario. Like the simulated data set, this real data set was now adjusted to meet the requirements of this study's scenario.

Data Preparation

Before, the data sets were adjusted to fit the scenario the participants order of the real data set was randomized by using the sample command. This step was necessary to avoid any order related biases, such as the possibility of smarter test-takers finishing first. Because the data of the simulated data set cannot be influenced by such biases this step was not applied.

Figure 1

Visualization of the Data Sets

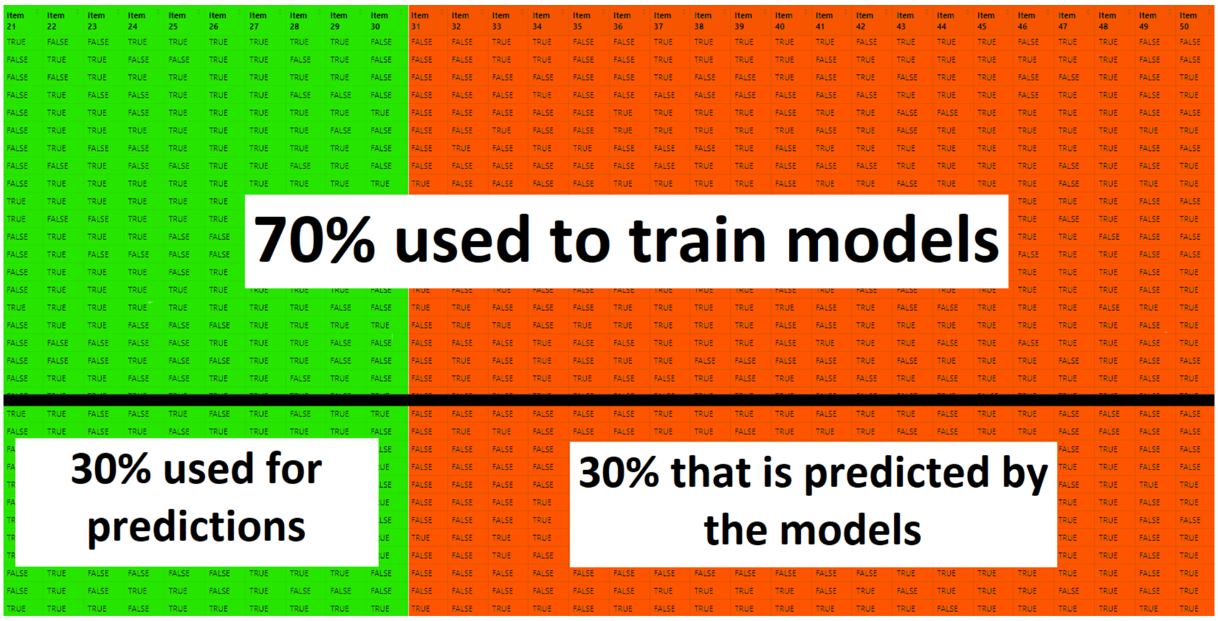


In the next step, missing values were created in both data sets to eliminate data that would be unavailable in a real scenario. It was defined that items 1-20 are exclusive to version A and items 31-50 are exclusive to version B while the items 21-30 are anchor items that appear in both versions. Assuming that the first 100 test-takers only worked on version A, their scores on the version B exclusive items (31-50) were removed from the data set. Similarly, all scores of the participants 101-200 on items 1-20 were changed to missing values in the data set. Visualized in Figure 1 the now created data set contains the shared anchor items in the middle while the version A exclusive items are on the left, and the version B exclusive items are on the right side. The upper and lower halves of Figure 1 represent the performance of test-takers for version A and version B, respectively, while the white spots in the top right and bottom left corner show missing values. Lastly, to compare the characteristics of the data sets, the variance and mean of the total scores were examined. Additionally, the Cronbach's Alpha coefficient was calculated and compared.

Prediction and Evaluation

Figure 2

Visualization of how the data is used (version B test-takers).



As explained in the scenario section, this research focused only on version B participants in the bottom half of Figure 1. Figure 2 represents a simplified version of this data without missing values and more accurate ratios between anchor- and version-exclusive items. The goal of each method was to predict the total score on version-exclusive items of the last 30% of the test-takers in the bottom right corner. The models were trained by the data of the first 70% of the participants in the upper half of Figure 2. The trained models were then applied to the data in the bottom left corner of Figure 2 to create predictions for the version-exclusive items in the bottom right corner. Since the exact process varied slightly for each method, further details will be provided in the following sections dedicated to each specific method. It is important to note that the application of each method was identical for both data sets.

Because all methods aimed to predict the same total score a variable was created to assess the accuracy of each method. The variable contains the true score on the version-exclusive items of the last 30 test-takers (orange, bottom right of Figure 2) and was compared to the predictions of each method using the RMSE function of the “caret” package. RMSE stands for root mean squared error and is next to other methods like the mean absolute error a widely used measure to assess the accuracy of models (Hodson, 2022). In this study, the function first found the differences between the predicted and the observed values before the differences were squared to ensure that, they are positive. Finally, the values were added up and divided by the number of observations, in this case, 30, to obtain some form of average error.

Harmonization Methods

IRT

When applying IRT as a harmonization method, there are two main assumptions that must be fulfilled (Kolen & Brennan, 2014). The unidimensionality assumption assumes that the test measures only a single latent variable, in the case of this study knowledge about the topic of the test. The local independence assumption assumes that the test-takers' responses are statistically independent and not influencing the performance on other items, such as in follow-up questions. However, in this study, those assumptions could be considered less relevant because due to the simulated nature of this study there was full control over the dimensions and independence of the simulated items, ensuring that they align with the underlying assumptions.

In contrast to the other methods, the IRT model was not trained using only 70% of the data. This is because the nature of IRT allows for utilizing more data during model training. Instead, a Rasch model was created using the "mirt" package, taking into account the performance of version A participants on the anchor items, as well as the performance of version B participants on both the anchor items and the version B exclusive items. This corresponds to the areas highlighted in green and orange in Figure 1. Based on that model a vector was created containing the theta scores that reflected the knowledge scores of each participant. However, because this vector consisted of values around zero and no total scores, it was made comparable next. The first step involved cantering the vector using the "scale" function, which subtracts the mean and scales the data by dividing it by the standard deviation. Then, all values were multiplied by the standard deviation of the total scores and added to the mean of the true scores. The output of this process was predicted total scores on the version-exclusive items that were now compared to the true scores using the RMSE.

Linear Regression

Before starting with the harmonization process of linear regression, three variables needed to be created. The first two variables that were used to train the linear model were based on the first 70% of test-takers. One variable consisted of the test-takers' total scores on the anchor items (green, top left of Figure 2) while the other variable consisted of the total scores on the corresponding version exclusive items (orange, top right of Figure 2). Those two variables are then merged into one data frame, called "traindata", that will be used for the creation of a linear model. The last variable was used for the actual prediction process and consisted of the remaining 30% of test-takers' total scores on the anchor items (green, bottom left corner of Figure 2). This variable was then implemented into a data frame called "testdata".

In the actual harmonization process, the first to variables inside the data frame were now used to train a linear model to predict the test-takers' total score on version exclusive items based on their total scores on the anchor items. In the next step, the trained linear model was applied to the "testdata" data frame using the predict function from the stats package, resulting in the generation of predicted data. Finally, the created data was now compared to the true scores using the RMSE function.

XGBoost

Unlike the other methods, XGBoost does not have any existing assumptions that need to be met because there is limited literature available for this particular method. Before the harmonization process, the "xgboost" package needed to be installed in R and the data sets were prepared. Instead of creating variables, this time two matrices were created that consist of the test-takers' performance on the anchor items. The first matrix consists of the data of the first 70% of test-takers (green, top left of Figure 2), while the second matrix includes the data of the remaining 30% of test-takers (green, bottom left corner of Figure 2). It is important to note that the matrices contain information about the performance on each item and not only total score like the variables used for linear regression. However, the previously created variable containing the total scores of 70% of the test-takers on the version-exclusive items (orange, top right of Figure 2) was used again.

After finishing the preparation process, a XGBoost model was trained to predict the total scores of the version exclusive items based on the test-takers' performance on the anchor items. In this process the matrix containing information about the 70% of participants and the associated vector were used while the tree limit was set to 100. Next, the created model was used together with the matrix containing information about the 30% of the test-takers to create predicted values. For this, the "predict" function of the "stats" package was used, while the tree limit again was set to 100. The created values were then compared to the true scores.

Results

Table 1

Characteristics of Data Sets' total scores on Version B

	Simulated Data	Real Data
Mean	17.03	14.03
Variance	21.32	22.51
Cronbach's Alpha	0.83	0.82

The characteristics of both data sets (Table 1) indicated that there are no big differences that could influence the harmonization process. Compared to the simulated data set, the real data had a lower mean and a slightly higher variance, showing that in total the questions were more difficult. Given that both data sets exhibited similar Cronbach's alpha values, it could be inferred that their internal consistencies are comparable. However, those results needed to be treated carefully because the values indicate only that there was a correlation between some items. The existence of substantial differences between the items is still possible.

Table 2

Accuracy of Harmonization Methods

Harmonization Method	Root Mean Squared Error	
	Rasch Model Data	Real Data
Item Response Theory	1.16	1.45
Linear Regression	2.84	3.37
XGBoost	3.39	3.72

The comparison of the root mean squared error (RMSE) in Table 2 indicated that all methods are more accurate when applied on the Rasch Model Data (RMD) rather than the Real Data (RD). For both data sets the IRT was the most accurate (RMD: 1.16, RD: 1.45), while linear regression was the second most accurate (RMD: 2.84, RD: 3.37) and XGBoost the least accurate method (RMD: 3.39, RD: 3.72). It is notable that for both data sets IRT is more than two times as accurate than linear regression, the second most accurate method.

When analysing the RMSE differences between the datasets, it was observed that these differences remained relatively consistent ranging from .29 to .53. The difference in accuracy between the datasets was found to be similar for both IRT (d = .29) and XGBoost (d = .33). However, in the case of linear regression, the difference in root mean square error (RMSE) was .53 and thus slightly larger.

Discussion

Examining the accuracy of the three selected methods in predicting test-takers' performance on different versions of an exam, the results of this study indicate that IRT stands out as the most accurate method. The second most accurate method, linear regression, was only half as accurate as IRT while still outperforming the new XGBoost approach.

IRT

As expected, when examining the outcomes of various harmonization attempts, the Item Response Theory (IRT) method exhibited overall the highest accuracy. Moreover, it was not surprising that IRT was more accurate when applied to the data set that was simulated based on a Rasch model. However, it is important to note that in the present scenario, the test-takers' ability levels (thetas) were calculated based on more data compared to the other methods. Because the model had more information than the other models it is thus not surprising that its accuracy is higher.

Nevertheless, the fact that IRT is the most accurate and consistent for both data sets goes hand in hand with the findings by Li et al. (2012) that indicated a high and consistent accuracy across multiple variations of IRT equating methods. However, it is important to note that compared to the usage of IRT in this study, it is more common to use more advanced IRT approaches. For instance, the approaches used by Li et. Al. (2012) used all available data of the data set (in this study that would correspond to all coloured areas in Figure 1) instead of using only information from the anchor items and version B exclusive items. Furthermore, there are even more advanced harmonization methods that simulate missing values by using the test-takers' ability- and the items' difficulty score in order to generate more training data (van den Berg et al., 2014). However, while utilizing them in this study would have been possible, it would at the same time further reduce the comparability to the other methods as they were trained with less information.

Linear Regression

The fact that linear regression performed worse than IRT in the given scenario can be explained by the amount of information that was used to train the model. Unlike IRT, the linear regression model used only the total scores of the anchor items and the exclusive items from 70 percent of the data for training. This limited amount of information used for training is significantly less than what was employed for the IRT model and even less than what was used for training the XGBoost model, which took into account the individual performance on each anchor item. Hence, it is understandable why linear regression performed worse than IRT. However, the reasons behind its better performance compared to XGBoost will be further investigated in the XGBoost section.

In order to enhance the performance of linear regression, more advanced methods can be used that incorporate more information than just the total scores of anchor- and version-exclusive items. One such method is the Tucker method (Kolen & Brennan, 2014), which takes into account all the available information in the dataset. When comparing the accuracy of the Tucker method to other IRT approaches, research conducted by Yang and Houang (1996) revealed only a marginal difference in accuracy. These results suggest that by refining the linear regression approach, its accuracy has the potential to be similar with that of IRT.

Finally, a study by Topczewski et al. (2013) showed that the Tucker method is notably less accurate when there are significant differences between groups. This finding goes hand in hand with the observed differences between the data sets in this study. However, the comparison between the study by Topczewski et al. (2013) and this study needs to be treated carefully as there are substantial differences between the way linear regression was used in this study and the Tucker method.

XGBoost

Considering the results, the performance of XGBoost might appear somewhat disappointing compared to its reputation for outperforming other algorithms in competitions. Surprisingly, in this study, it performed worse than other methods on both data sets. But why did XGBoost fail to perform better than the other methods in this scenario? One possible explanation could be overfitting, a phenomenon described by Dietterich (1995, p. 326-327) as follows: "Overfitting occurs when a learning algorithm fails to generalize from the training data to unseen test data. Instead, the algorithm learns or memorizes the training examples and their correct responses." In other words, overfitting happens when a machine learning model becomes too complex, capturing noise or irrelevant patterns from the training data, thereby leading to poor performance on new data.

In this study, XGBoost, a highly complex machine learning algorithm, was utilized to predict relatively simple data based on a small sample. Given that XGBoost is known to be prone to overfitting (Ellis, 2022), it is likely that this phenomenon occurred in the context of this study. To tackle this problem, there are several approaches that can help reduce overfitting. These approaches involve adjusting the parameters related to the trees, learning rate, and number of features, all aimed at reducing the complexity of the method (Ellis, 2022). Another approach that aims at making the training data more complex by increasing the number of features is called feature engineering. It describes the process of process of transforming or creating new features from existing data to improve the performance and accuracy of machine learning models (Dong & Liu, 2018). By implementing these strategies, overfitting can be alleviated, leading to improved accuracy in the predictions made by XGBoost.

Limitations

This study has several limitations due to its rather simple design. The first limitation is that only one prediction was made per harmonization method. This limits the replicability of the results and consequently affects the representativeness of the findings. Especially, when working with small sample sizes it is generally important to do multiple tries due to the comparatively big influence of potential outliers. Due to the fact that other related studies work with multiple tries (Livingston et al., 1990, Li et al., 2012, Yang & Houang, 1996), the results of this study need to be treated rather carefully.

Moreover, it is worth noting that the majority of comparable studies focus on larger and more complex datasets, often comprising thousands of observations (Yang & Houang, 1996; Livingston et al., 1990). In contrast, the current scenario has a limited number of observations, which affects the significance of the results. While repeating the experiment multiple times could alleviate this problem, having a larger number of predictions available for evaluation would also provide a clearer indication of the accuracy of a harmonization method.

Another limitation is related to the rather small heterogeneity of the datasets. Despite one being simulated and the other retrieved from the internet, they share similar characteristics, such as comparable variance. Although some differences were observed between the datasets, it is important to consider that more significant variations may arise when using datasets with stronger varying characteristics. However, in this study it could have been advantageous that the data sets shared similar characteristics as the RMSE between the data sets was compared. Comparing the RMSE between data sets with different characteristics can be problematic as the RMSE is influenced by the variance of the data sets.

The main challenge of this research lies in comparing the different methods, as they use varying amounts of information for training. In this study, the three methods IRT, linear regression, and XGBoost were compared. IRT utilizes data from participants in version A and information about every item, while the other methods only use the 70% training data from participants in version B. Moreover, there are notable differences even between linear regression and XGBoost, as linear regression is trained solely with total scores, whereas XGBoost incorporates item-specific information to some extent. Generally, it can be expected that methods using more information would show greater accuracy. However, an exception occurred in this study where the occurrence of overfitting prevented XGBoost from outperforming linear regression, despite using more information. In summary, comparing methods becomes challenging when they rely on different amounts of information.

One possible solution to solve this issue would be to use the most accurate approach known for each method, which likely involves using the maximum available information. In doing so, researchers can argue that each method is trained with as much data as it allows, thereby reducing the impact of differing information inputs. However, in this study, using more advanced methods was not possible due to the researchers' limited knowledge in the field of data science.

Suggestions for Future Research

To address the limitations, future research should consider changing essential aspects of the study design. Rather than predicting and evaluating accuracy once per method, researchers should repeat the process multiple times and calculate the average accuracy of the attempts. When using this approach, it is easier to avoid biases that may arise when working with small sample sizes. Additionally, using larger and more diverse datasets for harmonization would increase the significance of the study and lead to a clearer understanding of which harmonization method is most effective for specific types of data. Following, if there are bigger differences between the data sets, the normalized root mean squared error (NRMSE) should be used rather than the RMSE. Finally, by striving for the maximum accuracy of each method the problem regarding their comparability could be solved.

References

Balch, W. R. (1989). Item Order Affects Performance on Multiple-Choice Exams. *Teaching of Psychology*, *16*(2), 75–77. https://doi.org/10.1207/s15328023top1602_9

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 785–794. https://doi.org/10.1145/2939672.2939785

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, *27*(3), 326–327. https://doi.org/10.1145/212094.212114

Dong, G., & Liu, H. (2018). *Feature Engineering for Machine Learning and Data Analytics*. Crc Press.

Dorans, N. J., Moses, T. P., & Eignor, D. R. (2010). Principles and Practices of Test Score Equating. *ETS Research Report Series*, *2010*(2), i–41. https://doi.org/10.1002/j.2333-8504.2010.tb02236.x

Döring, M. (2018, December). Supervised Learning: Model Popularity from Past to Present. *KDnuggets*. https://www.kdnuggets.com/2018/12/supervised-learning-model-popularity-from-past-present.html

Ellis, C. (2022, August 26). *XGBoost overfitting*. Crunching the Data. https://crunchingthedata.com/xgboost-overfitting/

Gholamy, A., Kreinovich, V., & Kosheleva, O. (2018). Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation. *Departmental Technical Reports (CS)*, *1209*. https://scholarworks.utep.edu/cs_techrep/1209?utm_source=scholarworks.utep.edu%2Fcs_techrep%2F1209&utm_medium=PDF&utm_campaign=PDFCoverPages

González, J., & Wiberg, M. (2017). *Applying Test Equating Methods*. Springer.

Hodson, T. O. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not. *Geoscientific Model Development*, *15*(14), 5481–5487. https://doi.org/10.5194/gmd-15-5481-2022

Hohensinn, C., & Baghaei, P. (2017). Does the position of response options in multiple-choice tests matter? *Psicológica*, *38*(1), 93–109. https://www.redalyc.org/articulo.oa?id=16949050004

Kim, S.-H., Kwak, M., Bian, M., Feldberg, Z., Henry, T., Lee, J.-Y., İbrahim Burak Ölmez, Shen, Y., Tan, Y., Tanaka, V. T., Wang, J., Xu, J., & Cohen, A. S. (2017). A Taxonomy of Item Response Models in Psychometrika. *Quantitative Psychology*, 13–23. https://doi.org/10.1007/978-3-030-01310-3_2

Kolen, M. J., & Brennan, R. L. (2014). *Test Equating, Scaling, and Linking*. Springer Science & Business Media.

Li, D., Jiang, Y., & von Davier, A. A. (2012). The Accuracy and Consistency of a Series of IRT True Score Equatings. *Journal of Educational Measurement*, *49*(2), 167–189. https://doi.org/10.1111/j.1745-3984.2012.00167.x

Livingston, S. A., Dorans, N. J., & Wright, N. K. (1990). What Combination of Sampling and Equating Methods Works Best? *Applied Measurement in Education*, *3*(1), 73–95. https://doi.org/10.1207/s15324818ame0301_6

Nielsen, D. (2016). *Tree Boosting With XGBoost - Why Does XGBoost Win "Every" Machine Learning Competition?* [Master Thesis]. http://hdl.handle.net/11250/2433761

Otto, S. A. (2019, January 7). How to normalize the RMSE . *Marinedatascience*. https://www.marinedatascience.co/blog/2019/01/07/normalizing-the-rmse/

Petersen, N. J., Cook, L. S., & Stocking, M. L. (1983). Irt Versus Conventional Equating Methods: A Comparative Study of Scale Stability. *Journal of Educational Statistics*, *8*(2), 137–156. https://doi.org/10.3102/10769986008002137

Piechoka-Kaluzna, A. (2021). Measuring The Quality of Financial Statements After the Conversion To IFRS, Case of Poland. *Journal of Financial Studies and Research*, 1–26. https://doi.org/10.5171/2021.119430

Rumahorbo, B. N., Nanggala, K., Elwirehardja, G. N., & Pardamean, B. (2023). Analyzing important statistical features from facial behavior in human depression using XGBoost. *Commun. Math. Biol. Neurosci*, *35*(1). http://dx.doi.org/10.28919/cmbn/7916

Schneider, A., Hommel, G., & Blettner, M. (2010). Linear Regression Analysis . *Deutsches Ärzteblatt*, *107*(44), 776–782. PubMed Central. https://doi.org/10.3238%2Farztebl.2010.0776

Schurmeier, K. D., Atwood, C. H., Shepler, C. G., & Lautenschlager, G. J. (2010). Using Item Response Theory To Assess Changes in Student Performance Based on Changes in Question Wording. *Journal of Chemical Education*, *87*(11), 1268–1272. https://doi.org/10.1021/ed100422c

Synced. (2017, October 22). *Tree Boosting With XGBoost – Why Does XGBoost Win "Every" Machine Learning Competition? | Synced*. Syncedreview.com. https://syncedreview.com/2017/10/22/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition/

Topczewski, A., Cui, Z., Woodruff, D. P., Chen, H., & Fang, Y. (2013). A Comparison of Four Linear Equating Methods for the Common-Item Nonequivalent Groups Design Using Simulation Methods. *ACT Research Report Series*. https://www.act.org/content/act/en/research/pdfs/a-comparison-of-fourlinearequatingmethodsforthecommon-itemnonequ.html

van den Berg, S. M., de Moor, M. H., M, Mcgue, M., Pettersson, E., Terracciano, A., Verweij, K. J., H, Amin, N., Derringer, J., Esko, T., van Grootheest, G., Hansell, N. K., Huffman, J., Konte, B., Lahti, J., Luciano, M., Matteson, L. K., Viktorin, A., &

Wouda, J. (2014). Harmonization of Neuroticism and Extraversion phenotypes across inventories and cohorts in the Genetics of Personality Consortium: an application of Item Response Theory. *ProQuest*, *44*(7), 295–313. https://doi.org/10.1007/s10519-014-9654-x

Yang, W.-L., & Houang, R. T. (1996). The Effect of Anchor Length and Equating Method on the Accuracy of Test Equating: Comparisons of Linear and IRT-Based Equating Using an Anchor-Item Design. *Non-Journal*. https://eric-ed-gov.ezproxy2.utwente.nl/?id=ED401308

Zhao, X., Coxe, S., Sibley, M. H., Zulauf-McCurdy, C., & Pettit, J. W. (2022). Harmonizing Depression Measures Across Studies: a Tutorial for Data Harmonization. *Prevention Science*. https://doi.org/10.1007/s11121-022-01381-5

Appendix

Versions of used packages

dplyr version is 1.1.2

readr version is 2.1.4

tidyverse version is 2.0.0

mirt version is 1.39

kableExtra version is 1.3.4

Metrics version is 0.1.4

factoextra version is 1.0.7

FNN version is 1.1.3.2

lsa version is 0.73.3

modelr version is 0.1.11

xgboost version is 1.7.5.1

caret version is 6.0.94

psych version is 2.3.3

ltm version is 1.2.0

Used R-script

```
# 1 Load packages

library(dplyr)

library(readr)

library(tidyverse)

library(mirt)

library(kableExtra)

library(Metrics)

library(factoextra)
library(FNN)

library(lsa)

library(modelr)

library(xgboost)

library(caret)

library(psych)

library(ltm)

# 1.2 Check Versions of packages


packages <- c("dplyr", "readr", "tidyverse", "mirt", "kableExtra",

              "Metrics", "factoextra", "FNN", "lsa", "modelr", "xgboost", "caret", "psych", "ltm")


for (package in packages) {

  package_version <- packageVersion(package)

  cat(sprintf("%s version is %s\n", package, package_version))

}


# 2 Simulating Rash model data


Data_Rasch <- function(Nitems,Npersons,Nanchors) {


  # function to compute Rasch model probabilities

  compute_p <- function(theta, beta){

    exp(theta-beta)/(1+exp(theta-beta))

  }
```

```
# simulate thetas for populations P and Q,
# based on Benton T, 2017
theta <- c(rnorm(Npersons, 0, 0.7), rnorm(Npersons, 0.4, 0.7)) %>%
  matrix(Npersons*2, 2*Nitems+Nanchors, byrow = FALSE)
beta <- c(rnorm(Nitems, -1, 1), # form A
          rnorm(Nanchors, 0, 1),# anchors
          rnorm(Nitems, 0, 1)) %>%  # form B
  matrix(Npersons*2, 2*Nitems+Nanchors, byrow = TRUE)

p <- compute_p(theta, beta)


truedata <- rbernoulli(2*Npersons*(2*Nitems+Nanchors), p = p)

# create data to work with
data <- truedata

# common item design, creating missings
# middle Nanchors items are common to both tests (anchor items)
data[1:(Npersons),          (Nitems+Nanchors+1):(2*Nitems+Nanchors)] <- NA
data[(Npersons+1):(2*Npersons),  1:Nitems] <- NA
```

```
  # names of the items
  colnames(data) <- paste0("Item ", seq(1, ncol(data)))
  colnames(truedata) <- paste0("Item ", seq(1, ncol(data)))
  anchor_names <- colnames(data)[(Nitems+1):(Nitems+Nanchors)]

  data_list<- list("truedata" = truedata, "data" = data)
  return(data_list)
}
# 2.1 Data Preparation of Rasch model data

# Simulation parameters for data simulation
# based on Benton T, 2017
Nitems <- 20  # number of items per test (form)
Npersons <- 100 # number of persons per form
Nanchors <- 10 # number of anchor items (extra items)

#Data Generation
set.seed(2001)

#Rasch model
All_data <- Data_Rasch(Nitems,Npersons,Nanchors)

data <- All_data$data
```

```
truedata <- All_data$truedata

# names of the items

colnames(data) <- paste0("Item ", seq(1, ncol(data)))

colnames(truedata) <- paste0("Item ", seq(1, ncol(data)))

anchor_names <- colnames(data)[(Nitems+1):(Nitems+Nanchors)]


# 2.2 Computing test scores for Rasch model data


testA <- data[1:(Npersons),          1:(Nitems+Nanchors)]

testB <- data[(Npersons+1):(2*Npersons), (Nitems+1):(2*Nitems+Nanchors)]

S_Anchors <- apply(data[,anchor_names], 1, sum) # sum score on anchor items

S_A <- apply(testA, 1, sum) # sum score test A

S_B <- apply(testB, 1, sum) # sum score test B





# 3 Importing and preparing second (ADM)Data set


#download it from the internet

load(url("http://www.mat.uc.cl/~jorge.gonzalez/EquatingRbook/ADM1.Rda"))


#rename the dataset

truedata2 <- ADM1

rm(ADM1)  # remove original dataset to avoid confusion
```

```
#select relevant part

truedata2 <- truedata2[1:200, 1:50]


#create missings to get a dataset that contains only data that would be available in a real
scenario

data2 <- truedata2

data2[1:100, 31:50] <- NA

data2[101:200, 1:20] <- NA


#rearrange data

data2 <- data2[c(1:100, sample(101:200)), ]


# 4 Create necessary variables for prediction models


# 4.1 for Rasch Model Data


# sum scores on the anchor items

S_Anchors <- rowSums(truedata[, 21:30])


#to evaluate accuracy the total scores on b exclusive items by the 30 percent we want to
predict

ts_on_B_by_B30 <- rowSums(truedata[171:200, 31:50]) # observed sum scores
```

```
# 4.2 for Data 2 (ADM)

#sum scores on the anchor items
S_Anchors2 <- rowSums(data2[, 21:30])

#to evaluate accuracy the total scores on b exclusive items by the 30 percent we want to
predict
ts_on_B_by_B302 <- rowSums(data2[171:200, 31:50])

# 5 Item Response Theory

# 5.1 Creating model and evaluate Accuracy for Rasch model Data

#Create a Rasch model for the 30%  we want to predict (possiible for all students but only
30% to make it as comparable to the other methods as possible as possible)
IRTmodel <- (1*data[1:200, 21:50]) %>% mirt(itemtype = "Rasch", verbose = F)

#Create Vector with test-takers' thetas (ability/knowledge score)
theta_hat <- fscores(IRTmodel)

#adjust thetas of IRT model to make it comparable to the totalscores on the Items we want to
predict
#(totalscore on version A questions by test-takers who did version B)
S_hat_IRT <- scale(theta_hat[171:200])
S_hat_IRT <- S_hat_IRT*sd(ts_on_B_by_B30) + mean(ts_on_B_by_B30)
```

```
#Check accuracy by comparing predicted values to truescores using RMSE

RMSE(S_hat_IRT, ts_on_B_by_B30)


# 5.2 Creating model and evaluate Accuracy for Data2


#Create a Rasch model for the 30%  we want to predict (possiible for all students but only
30% to make it as comparable to the other methods as possible as possible)

IRTmodel2 <- (1*data2[1:200, 21:50]) %>% mirt(itemtype = "Rasch", verbose = F)


#Create Vector with test-takers' thetas (ability/knowledge score)

theta_hat2 <- fscores(IRTmodel2)


#adjust thetas of IRT model to make it comparable to the totalscores on the Items we want to
predict

#(totalscore on version A questions by test-takers who did version B)

S_hat_IRT2 <- scale(theta_hat2[171:200])

S_hat_IRT2 <- S_hat_IRT2*sd(ts_on_B_by_B302) + mean(ts_on_B_by_B302)


#Check accuracy by comparing predicted values to truescores using RMSE

RMSE(S_hat_IRT2, ts_on_B_by_B302)




# 6 Linear Regression
# 6.1 Linear Regression for data 1

#Create Sum score of anchor items for the first 70% of version B students (used to train train the models)

S_AnchorsB70 <- S_Anchors[101:170]


#Create Sum score of anchor items for the  last 30% of version B students (used for prediction)

S_AnchorsB30 <- S_Anchors[171:200]


#create variable to train the model (total score on B by the first 70% of test takers who did B)

ts_on_B_by_B70 <- rowSums(data[101:170, 31:50])


traindata <- data.frame(ts_on_B_by_B70, Ascore = S_AnchorsB70)


lm <- lm(ts_on_B_by_B70 ~ Ascore, data = traindata)

#predict the total scores of remaining 30%


testdata <- data.frame(Ascore = S_AnchorsB30)


plm <- predict.lm(lm, testdata)


RMSE(plm,ts_on_B_by_B30)



# 6.2 Linear Regression for data 2
```

```
#Create Sum score of anchor items for the first 70% of version B students (used to train train the models)

S_AnchorsB702 <- S_Anchors2[101:170]


#Create Sum score of anchor items for the  last 30% of version B students (used for prediction)

S_AnchorsB302 <- S_Anchors2[171:200]


#create variable to train the model (total score on B by the first 70% of test takers who did B)

ts_on_B_by_B702 <- rowSums(data[101:170, 31:50])


traindata2 <- data.frame(ts_on_B_by_B702, Ascore = S_AnchorsB702)


lm2 <- lm(ts_on_B_by_B702 ~ Ascore, data = traindata2)

#predict the total scores of remaining 30%


testdata2 <- data.frame(Ascore = S_AnchorsB302)


plm2 <- predict.lm(lm2, testdata2)


RMSE(plm2,ts_on_B_by_B302)
```

```
# 7 XGBoost


# 7.1 XGBoost muliple predictors


#Create Matrix of performance of 70% of b test takers on anchor items (no total score)

anch_mat70 <- as.matrix(data[101:170, 21:30])

anch_mat70 <- apply(anch_mat70, c(1, 2), as.numeric)





#create a model that predicts the total scores on version b exclusive items based on the anchor
item performance (trained with 70% of data)

#ts_on_B_by_B70 same as in lm prediction

xgb_model <- xgboost(data = anch_mat70,

                     label = ts_on_B_by_B70, nrounds = 100,

                     objective = "reg:squarederror")


#for prediction create matrix of anchor performance of remaining 30%


anch_mat30 <- as.matrix(data[171:200, 21:30])

anch_mat30 <- apply(anch_mat30, c(1, 2), as.numeric)


#predict the total scores of remaining 30%

pxgb <- predict(xgb_model, anch_mat30, ntree_limit = 100)
```

```
#compare the prediction to the true score

RMSE(pxgb, ts_on_B_by_B30)



# 7.2 XGBoost muliple predictors


#Create Matrix of performance of 70% of b test takers on anchor items (no total score)

anch_mat702 <- as.matrix(data2[101:170, 21:30])

anch_mat702 <- apply(anch_mat702, c(1, 2), as.numeric)




#create a model that predicts the total scores on version b exclusive items based on the anchor
item performance (trained with 70% of data)

#ts_on_B_by_B702 same as in lm prediction

xgb_model2 <- xgboost(data = anch_mat702,

                      label = ts_on_B_by_B702, nrounds = 100,

                      objective = "reg:squarederror")


#for prediction create matrix of anchor performance of remaining 30%


anch_mat302 <- as.matrix(data2[171:200, 21:30])

anch_mat302 <- apply(anch_mat302, c(1, 2), as.numeric)


#predict the total scores of remaining 30%
```

```
pxgb2 <- predict(xgb_model2, anch_mat302, ntree_limit = 100)


#compare the prediction to the true score

RMSE(pxgb, ts_on_B_by_B302)




# 8 Compare characteristics of data set


S_true <- rowSums(data[101:200, 21:50])


S_true2 <- rowSums(data2[101:200, 21:50])


#data1

var(S_true)

sd(S_true)

mean(S_true)


alpha_value <- alpha(truedata)$total$raw_alpha


print(alpha_value)


#data 2

var(S_true2)

sd(S_true2)

mean(S_true2)
```

```
alpha_value2 <- alpha(truedata2)$total$raw_alpha

print(alpha_value2)
```