POLITECNICO DI TORINO

Facoltà di Ingegneria Informatica

Corso di Laurea in Computer Engineering

Tesi di Laurea Magistrale

Historical Data Analysis and Item Response Theory for Calibration of Question Items

Supervisor:
prof. Paolo Garza

Matteo Fiore

Company Supervision
Research and Development Department
Ing. Vidar Sveen

Anno accademico 2017-2018

Contents

Chapter 1

Introduction

1.1 Problem specification

With the growing number of exams and tests a person of our modern society has to take, it is increasingly important that the tests are valid, reliable and don't take too much time. So far examination of pupils have been done using Classical Test Theory (CTT), focusing on the final score of the assessment. The main problem related to this is the inevitable impact of test difficulty on the single student: the test difficulty influences directly the final grade as a function of the ability. Extracting the ability of the pupil instead of only the score for the assessment, would help to improve evaluations, moving the focus from assessment's score to students' ability.

Technologies such as Item Response Theory (IRT) allow to extract those kind of information that would enhance both teacher and student experience and learning. The focus of this project will be to investigate the classical models for student evaluation, understanding limitations and exploiting the usable features. In addition a first solution to assess the potential of IRT will be proposed, while investigating the products already present on the market.

The thesis is organized in the following way: a small introduction to the company is done followed by an overview of the products available for IRT evaluations and technologies used during the project. In chapter 3 the preliminary work is described, analyzing the initial steps in the problem's study and the first difficulties encountered. In chapter 4 a first set of analysis is reported, followed by a deeper overview of IRT and the first implementation of the engine for the evaluations. In chapter 6 the CTT implementation is introduced, with an overview of the Inspera project already in place and the changes made. Chapter 7 treats the implementation of the implemented IRT product in the cloud, analyzing the different choices possible to make that happen. Finally, chapter 8 reports some more experiments done in relation to the relationship among CTT and IRT.

1.2 The Company

Inspera is the leading European provider of high-stakes digital examination solutions. We offer end-to-end solutions for governments, awarding bodies, higher education institutions as well as municipalities/schools. On a mission to re-invent educational assessment for the 21st century, we offer the most innovative, reliable and secure service as well as the necessary expertise to digitise large-scale examinations. Inspera Assessment, our cloud-based platform, is tailored for open- and closed- book summative and formative exams, and for diagnostics tests. Designed to meet all standards and regulatory requirements for universal access, our solution is a first choice to serve first-time learners across all levels of education, nationality, and needs. In total, over 6 million digital tests and examinations have been conducted by Inspera since 2008. Inspera Assessment has been implemented at more than 50 education institutions that use e-assessment, via both institutional devices and BYOD. At Inspera, our goal is to support digital assessment of learning and for learning. And it is our mission to provide the best solutions available in the industry, by delivering a robust, reliable and secure solution. As part of that endeavour, we have a robust R&D agenda, and continuously develop support for new assessment methods and tools.[1]

[1]The information in this section largely derives from material that will be hosted on the Inspera website in the near future.

Chapter 2

Related Word and Technologies

2.1 Introduction to IRT

Item Response Theory (IRT) is a method used to and evaluate Interactive Assessment (IA) and items, to calculate their difficulty and to extract useful information ragarding exam and examinee.

The main concept behind IRT is that score is not a fundamental measure of the knowledge of the student: as Dr. Frederic Lord pointed out, examinee observed scores and true scores are not synonymous with ability scores. Ability scores are a more fundamental measure because they are test independent while observed scores and true scores are not. The ability of each student is independent from the final result obtained in an IA. At the moment of the evaluation the student has a knowledge independent from the difficulty of the test, while the final result is strictly correlated with the IA proposed. That is why lately Classical Test Theory (CTT) has been a less common choice in IA evaluations. Nevertheless, CTT has been implemented, because still provides useful insights and represents an important part of the psychometric heritage.

More theoretical information will be reported in section 5.1.

Item Response Function

The Item Response Function (IRF) is the function representing an item in IRT. The IRF describes the probability of a student with a given ability level θ to assess correctly an item with a given difficulty β. In addition, the IRF identify difficulty and discrimination of a question. The difficulty is identified by the point in which the student has half the probability to assess the item correctly. The discrimination is the value of the first derivative in that point: given two students with a close ability level, how well can the question discriminate among those?

In figure 2.1 there are represented different types of IRF for dichotomous items: the orange line identifies the discrimination while the blue line identifies the difficulty of a certain question. Different items differ from each other in difficulty and discrimination.

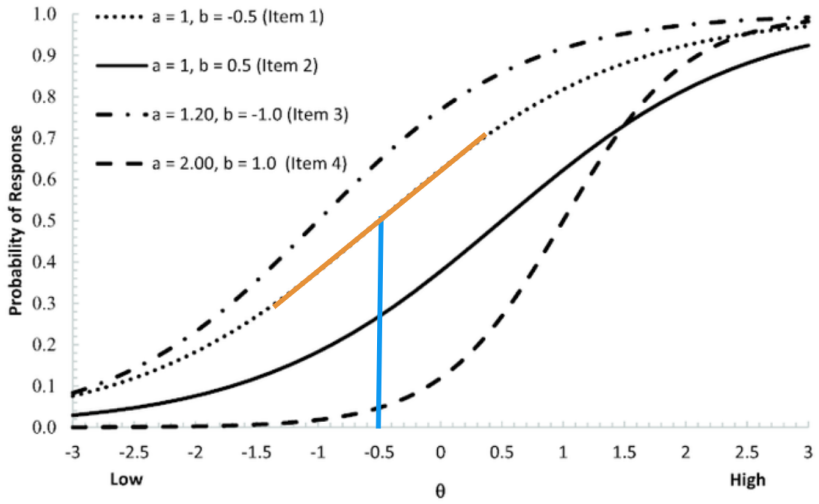

Figure 2.1. Example IRF of a dichotomous question

The ability of a student is on the same scale of the difficulty of the item: a student with an ability greater than the difficulty of the question has a probability greater than 50 percent to assess the item correctly. The opposite situation when ability lower than the difficulty.

In figure 2.2 it is showed an IRF for a polytomous item. A polytomous item has different scores, and for each of the achievable score a category is created. P1 represents the probability that a student gets the score one, P2 of getting score two and so on. The intersection among the probability function of each level identifies a boundary. In case of figure 2.2 there are 5 different level, and 4 four different boundaries.

Differences among IRT and CTT

There are some key differences among IRT and CTT. The more significant one is the following: unbiased estimate of items property may be obtained from unrepresentative samples. In CTT in order to evaluate item's parameter, it is necessary to have a representative sample of the students. For example, the difficulty is calculated as average normalized score. Having a sample of good students and a sample of bad students will result in a different evaluation of the question's difficulty based on the

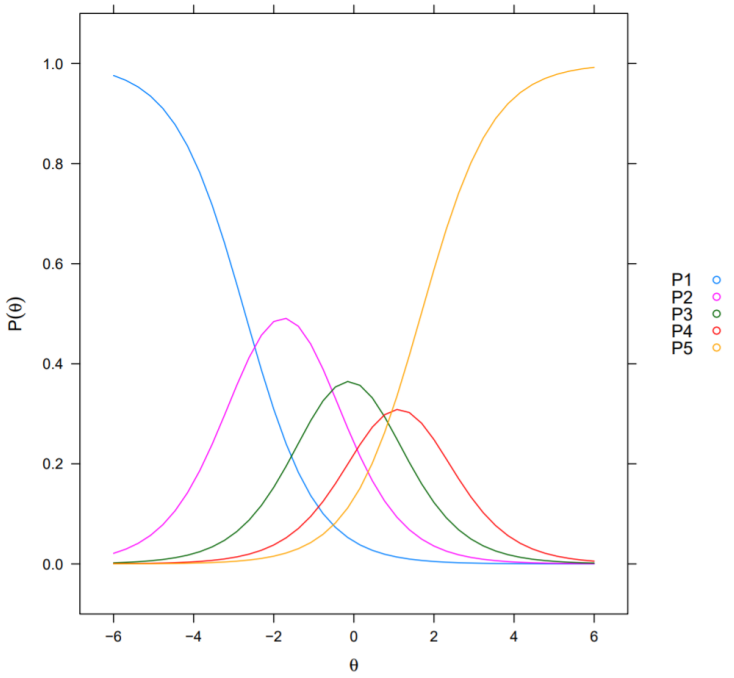

Figure 2.2. Example IRF of a polytomous question

set of students being tested. In IRT this is not always the case.

In chapter 8 an experiment is reported to prove that IRT is sample independent.

Anchoring

The concept of anchor is very important in IRT: an anchor is a question that is used in two different test, and therefor link them.

If two tests have enough common items, it is possible to use those to equate the two assessments. Equating means putting on the same scale the parameters for items in both tests. After the items are on the same scale it is possible to compare them, to evaluate the ability of the students and compare those, to understand the variation of the difficulty among the tests.

2.2 IRT Softwares

IRT has been used in different product, allowing to obtain results and insights on tests and pupils. In the following sections an overview of some software used for IRT will be done.

JMETRIK

JMetrik is a free psychometric software developed by J. Patrick Meyer. Its origins go back to 2006, and it has been released for the first time in 2009. JMetrik is based on a graphical interface, that allows the user to easily navigate the software and perform the necessary evaluations. More information about the product can be found at the link in the appendix.

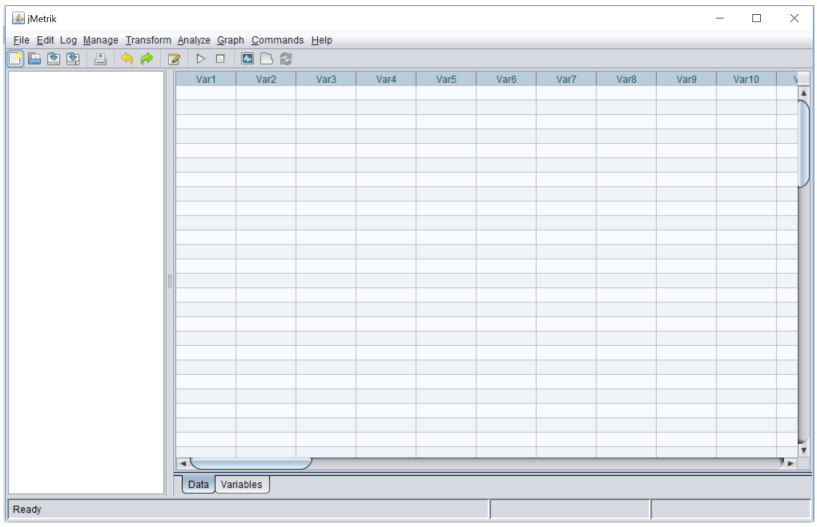

Figure 2.3. Example of JMetrik graphical interface

Pro

JMetrik is a free software with a good offer of psychometric analysis. In addition is written in Java, and this makes it portable on different platforms. It is based on a database model instead of a file-based model, and that allows to have computations and results in the same place, with easier managing of projects. Finally it provides an R package that allows the interaction among R and the product.

Cons

The database system based requires some work to import the data in the right format. In addition it does not provide any command line interfaces, that would have been very useful during the automation part.

IRTPRO

IRTPRO is an new application for item calibration and test scoring using IRT.

Pro

IRTPRO provides a way to run existing .irtpro command file from command line. This features would have been really useful for the application, but unluckily impossible to use for the reason described below.

Cons

It was impossible to install the software, due to an error during the installation. In addition this software is not free, and requires activation for full usage.

Xcalibre

Developed by Assessment Systems Corporation, "Xcalibre is a Window application designed to perform estimation of item response theory (IRT) item parameters with user-friendly reports. The purpose of these reports is to help testing programs evaluate the quality of test items by examining their psychometric characteristics, as well as provide the IRT parameters necessary for scoring examinees with IRT, especially with computerized adaptive testing (CAT)" (Guyer 2014).

Pro

Xcalibre user interface is really intuitive to use: with a quite easy structure of the input files allows the user to exploit the potentialities of IRT without too much work beforehand. The support of the software is very good, an exhaustive manual is included and contains also some theoretical knowledge useful to understanding the undergoing evaluations. It produces a report containing all the information needed for the final interpretation of the results, as well as csv files that can be re-used and re-elaborated.

Cons

The license is definitely something to take into account. In addition there is no way of running the software from command line: there is the possibility of using it with different question sets without every time re-loading results and items' specifications.

For more information the website of the product is reported in the appendix.

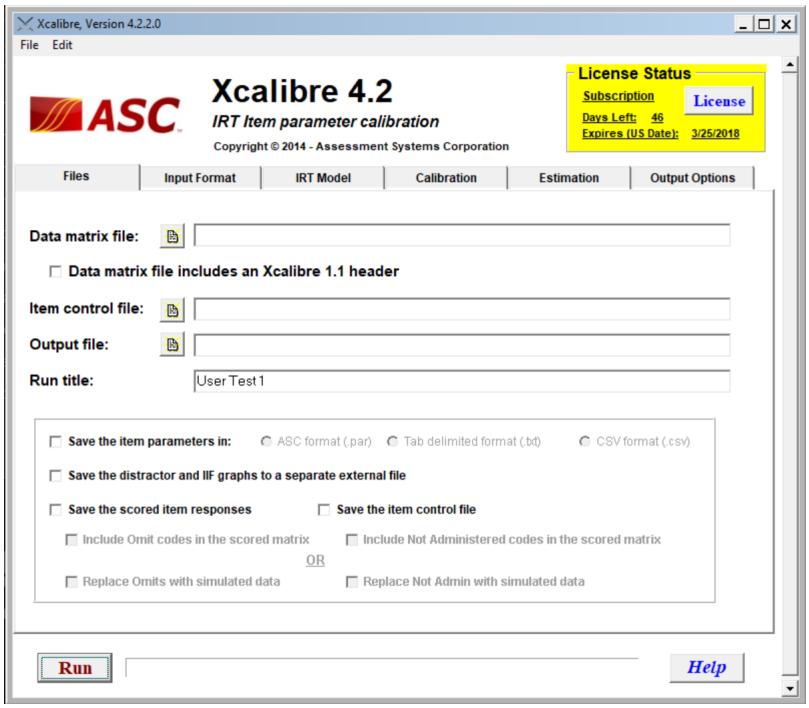

Figure 2.4. Example of Xcalibre graphical interface

2.3 Storage systems

Working with data implies storing them in the right infrastructure. Depending on the usage, the type of queries that are planned to be performed, the frequency of the interaction with these structures and the amount of data to be retrieved, different solutions can be used. An overview of the two technologies for data storage that have been used, as well as differences among them will be given.

Oracle ER database

A relational database is a storage system organized in tables: each table (or relation) contains a set of rows identified by an attribute. Attributes are unique and some of them can be used to relate to different tables containing different information. An entity-relationship database relies on transactions to update the entire system. Transactions are operation that follow the ACID principles

- *atomicity*: transactions are indivisible unit of work: all (or none) the operations contained in the transaction must be executed and the database can not remain in an intermediate state

- *consistency*: a transaction should take the database from a point of consistency (where everything in the database is correct), to another point of consistency. The final point of consistency can coincide with the initial state if abortion was performed
- *isolation*: execution of a transaction is not dependent on transactions that are simultaneously being executed. In addition the effects of a transaction are visible only when the transaction terminates
- *durability*: when a transaction terminated, its results are made permanent. The database management system provides methods of recovering to the correct state of the database if a failure happens

Relational databases are not structures to scale and to accommodate dynamic and not heavily structured data models.

Elasticsearch

Elasticsearch is a schema-free, JSON based document store. It is distributed and horizontally scalable, built on top of apache Lucene and allows real time data search and analysis. Elasticsearch provides features like full text search, fuzzy matching, proximity queries and multi-fields matches. The system allows to execute analytic queries as well as search queries in a very fast way. The reason of the speed of elasticsearch is to be found in its "memory-loading" paradigm: the information contained in the documents needs to be accessed in a very fast way and multiple times in different order. This random access pattern makes necessary to load the information in memory before the usage. A big problem that derives from it is an overloading of the memory: if too many big queries tries to load information in memory, it will be necessary to free memory and this will take some time. Freeing the memory would make processes to wait for its completion, resulting in the node not responding to the master anymore and being removed from the cluster. This is a problem called memory pressure, and solved using circuit breaker as a solution.

Memory pressure is only one of the problems related to elasticsearch: there are different problems related to resilience that make elasticsearch not optimal as primary storage system. Resilience is the concept of recovering from failures in the ideal way, and returning a result to the user that is consistent with the updates to the various documents done in precedence. Different issues can be linked to resiliency, that can be categorized in four different categories: cluster and node failures, both being either slow or fast. Independently from cluster or node failures, the fast ones are the more easy to address. Elasticsearch is doing a great job to address problems related to resiliency, but a careful choice should be made based on the type of system object of implementation. A link to more issues related to resiliency in elasticsearch

can be found in the appendix.

The internal structure of elasticsearch is organized in nodes. There are different types of nodes (Elasticsearch Reference - Node 2018):

- *master eligible node*: is a data node that can be elected as a master
- *data node*: data nodes hold data and perform data related operations such as CRUD, search, and aggregations
- *ingest nodes*: this type of node is used to pre-process documents before the actual document indexing happens

In opposition to relational databases, non relational databases supports CAP:

- *consistency*: every read receives either the most recent version of the data or an error
- *availability*: every request receives an answer, and the answer is not supposed to be an error. The version of the data returned is not guarantee to be the most recent
- *partitioning*: the cluster continues to work even if there is a partition (or interruption in the communication) among two nodes

Chapter 3

Preliminary work

The realization of the final product required some work to be done in advance, in order to understand the possible different problems, the value of the results obtained and how to exploit them. The initial study was performed in a different environment from the one that would have been used for the final implementation. Because of sensible and relevant data, any type of corruption or accidental deletion would have been contained in the testing and researching environment without any impact on the rest of the system used by costumers.

3.1 From ER DB to ES

Infrastructure plays a very important role when it comes to implement a software solution, and defining the way different modules interact represent an important step in the product specification. The information needed for the evaluations and the initial study were stored in an entity-relationship (ER) database, organized in a way to hold all the data regarding students, assessments, items, answers and all the other information that can be obtained while running online examinations.

As explained in section 2.3, an ER database is a solid option, but not the best one in case of heavy queries used to evaluate analytics parameters, especially when based on search queries. Velocity, and ability to handle lots of requests in an efficient way were necessaries key features in the system. For these reasons elasticsearch (ES) engine was chosen as the optimal solution. ES is not a relational database, but is a search engine (or a NoSQL database management system dedicated to the search of data content). As primary storage system something able to better model the structure of the data was necessary: ES is not optimal to model relationship, or at least not as good as ER databases.

For these reasons, the basic structure necessary for the project was a combination of ER database and ES indexes.

3.1.1 The extraction

ES indexes use mapping to define the structure of the index itself. The mapping is a JSON file defining all the fields of the document, their type and the eventual hierarchies. Its design was done beforehand, knowing the data necessary for IRT computation, in order to allow an easy lookup. The representation of the index is reported in the appendix.

The data necessary for IRT computation are:

- type of question
- maximum score for the question
- number of distractors
- distractors' definition: name, ID, position
- ID of the user answering the specific question
- score achieved by the student
- information used to identify a specific item in a question set

Those information were stored inside the ES index, in form of documents, each of them representing an answer of a student for a specific item. In order to populate the index, a task was designed for the specific purpose.

The task

The main targeted goal was to move the data from the ER database to the ES index, while shaping them in the appropriate way. The first step performed was fetching the data from the database. Due to the high number of tables and the dependencies among them, an heavy query was necessary. *Heavy* refers to the cost of the query: having a too costly query would have resulted in too much workload on the database, with slowdowns and performance issue that were not supposed to manifest. In the original system, caching methodologies are taking care of having the information always available, with no overload on the database. In addition, nowadays the task is run only on the new assessments performed daily, and the risk of overloading the system with queries is much lower since not all the historical data are taken into account. In the case of the task developed in the early stages all the data present in the database were supposed to be moved, and extra care in load balancing was required.

To solve the problem of load balancing the following approach was used: for each particular exam, only a subset of students' answers were processed. The size

of the students' batch was specified a priori in the settings of the program, and changed according to necessity. To avoid processing the same students in different runs, a status was set after processing a specific candidate. This solution, other than regulating load balancing, was also really helpful for debugging: setting a batch size of only a couple of students allowed to have fast execution, resulting in the possibility to check the correctness of the results without waiting too much.

Once obtained the necessary data, the reshaping was taking place. This part consisted in creating a JSON file representing the ES document. Some information were really easy to retrieve (like user ID, or the starting time) because stored in a clear way in the database and easily accessible after queering the latter. Other information were more difficult to obtain because not present or not clearly stored.

In the following sections an overview of the most problematic issues is presented.

Question definition

Each question is defined as an XML document, with different sections containing different information (like name of the question, ID, score achievable). The most important ones were the maximum score achievable, the interactions, the order of the distractors, their ID and the correct ones. Those information were useful because:

- score: used in the assessment score calculation (explained in the section *Test score* below
- interaction's ID: each interaction has an ID that is used to identify it; extracting the ID was useful to identify the right relationship distractor-interaction
- order: particularly useful for multiple choice questions, allowed to precisely identify order of the distractors in the item definition
- distractor's ID: used to identify uniquely an answer
- correct distractor: used during the item definition in the IRT models

The main problems encountered were related to the interaction's IDs: depending on the type of question the ID could have been of different format, or even not distinguishable among different interactions (resulting in the impossibility to have a clear definition of the item itself). When not possible to obtain the information, the set of distractors for each interaction was not extracted.

User answers

Each user answer is in XML format, with different sections identifying different information on the specific answer. The performed XML parsing focused on extracting

the feedback and the given answer. The feedback could have been of different types, but the most important one was *FEEDBACK_UNASWERED*. When a student was not answering the question all the correct distractors were copied in the ES document's section dedicated to the selected ones, but with the ID set to *null*.

It was not enough to just not set the selected answers: given a set of multiple interactions with one of them not answered, would have been impossible to differentiate among which of them was the not answered one. In addition given the interaction without any specific distractors (for example drag and drop questions), distinguishing between the latter case and the not answered one would have been impossible. To solve these problems, this structure was used:

```
"selectedResponseVariable": {
"responseId": "null",
"interactionId": genericInteractionId,
"order": 0
}
```

ResponseId set to *null* in order to symbolize a not answered interaction, identified by the *interactionId*.

Test score

The achievable test score was not clearly stored in the database, and was difficult to obtain this information. After digging and asking colleagues for help, I found out a function present in the database, fetching the necessary information and automatically evaluate the total score. The function was parsing the XML of each answer for a given student attending a particular exam, extracting the score for the answer and summing up the value with the ones extracted from other answers. The final value was then stored in a newly created table and used afterwords to obtain the test score given a specific student or set of students.

Language extraction

Given the opportunity to perform Differential Item Functioning (DIF) on the set of students doing the assessment, a possible grouping could have been identified by spoken language. At the beginning of tests questions regarding the spoken language were sometimes asked: the problem was that the type of the question was not set as *language* but as a generic *multiple choice* question. In order to identify those type of items the words in the question were used: if the question's text was matching a specific pattern than the question itself was treated as a language one. This kind of solution was not the best one, because relied on regular expression matching and could have been not precise.

Anchors

As explained in the section 2.1 anchors play an important role in IRT computation, especially when the goal is to link different assessments and compare them. The extraction of anchors given a certain item was performed using a table build beforehand and containing IDs (of both question and assessment it was used in), title of the question, classroom's grade it was used in and the date of the usage. Matching the title, the classroom's grade and filtering on assessment's IDs and date it was possible to recover all the same questions used in different tests.

The naive assumption behind anchor definition was the following: a question with the same title has also the same content, hence the same difficulty. The proper way of identifying anchors would have been analyzing the content, understanding it and matching it with the content of another question; if the same than label the questions as anchors. This would have required a couple of month by itself to be implemented, so I just started with a naive solution. Even though naive, it was based on real anchor labelling as explained in section 3.3.

3.2 Run IRT analysis

Having the data in the ES index was not enough: a tool to run the IRT evaluation was necessary. The choice of the IRT tool has been pretty straight forward: after asking to a Norwegian psychometrician about suggestions and comparing different products, the third party software Xcalibre was selected. Xcalibre runs using a series of files containing the data necessary for the evaluation and gives as output different results that can be interpreted in different ways. In the following sections the preparation of the files, the output of Xcalibre and the limitation of using such software will be presented.

3.2.1 Preparation

The preparation phase consisted in the identification of the type of question and the creation of the files containing the answers for each user. The organization of the file for the various items was the following:

- identification: the ID of each item
- correct response: in case of polytomous question in this part of the file a "+' or a "-" was inserted, to differentiate between ascending and descending evaluation of the score. In case of dichotomous questions, the correct answer in was inserted

- number of different alternatives: in case of polytomous questions the maximum score of the question was reported, otherwise if dichotomous the different number of alternatives
- type: the type of the question
- additional parameters: used when the item was an anchor. The different IRT parameters for the specific question were reported

An important step in the preparation of the file containing the items' specification was the pruning for polytomous items. Supposing that a generic polytomous item had four different possible scores, and not even a student got 3 points, than it was impossible to define boundaries for all the different scores. It was necessary to reshape the question as showed in figure 3.1.

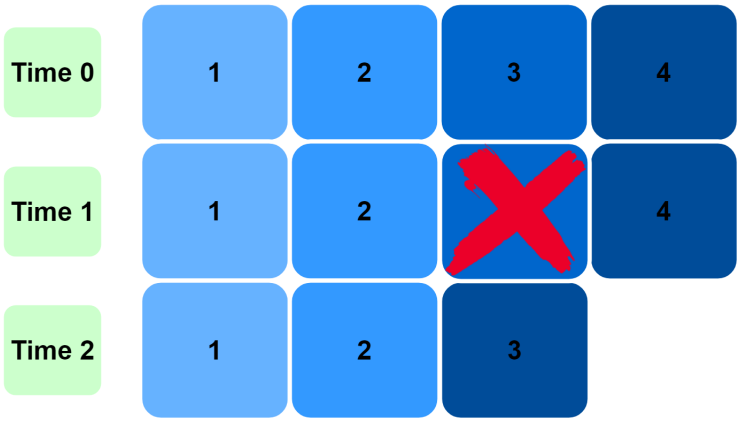

Figure 3.1. Pruning for polytomous item with 4 alternatives

The effects of pruning could have been different: either the item from polytomous became dichotomous, (remained with only two different alternatives) or it was necessary to remove the item and not perform the IRT evaluaitons on it.
A problem with pruning was that in some cases students' score was greater than the maximum score achievable for the question. A careful process of data cleaning has been necessary to detects these situations and correct them.

The preparation of the file for the students consisted in listing for all the students the score or the distractor selected, and in case of pruning re-basing the answers.

3.2.2 Output

The output consisted in ability for each student, as well as their score. In addition: difficulty, discrimination, fit and other statistics for each item were obtained. The details of those statistics are not reported here because were not calculated directly and the details can be found in the Xcalibre manual.

3.2.3 Limitations

The problem of using a software like Xcalibre in order to run IRT analysis was that there was no API available: this program requires human interaction, and when it is necessary to run several analysis for several different tests multiple times, human interaction should not happen. In order to automate the process a library for graphical interface interaction was employed. Xcalibre was remotely launched and terminated at the end of the evaluation (identified by the cpu usage falling below 5 %). One side effect was that during the run the computer was not usable for other purposes. In addition this solution was not scalable and not efficient from the point of view of the computation (starting an external program every time, waiting for responses and for the cpu usage to fall behind 5% was time consuming). In the worst case, if the cpu usage would note have fallen below 5%, detecting the termination of the analysis was actually really difficult.

A solution will be explained in section 5.3, and will consist in evaluating IRT inside the system, using a program made for the purpose eventually able to be parallelized.

3.3 Achors extraction

Anchoring means finding common question between two different assessments. In order to identify anchors the procedure described in the subsection 3.1.1: *Anchors* was used. Now the focus will be in describing the steps followed to create the table with the information about anchors. Four different approaches were used before reaching the final result.

Before going into the details it is necessary to briefly explain the structure of the system used by professors to create an exam. The object *exam* is created and put inside a folder, as well as the objects *items*. Then questions are linked to exams and the test is ready to be given to students. In addition anchors were manually labelled with a specific code: the problem was that there was no information about which assessments the anchors were linking.

First approach

As first approach a comparison among the item ids was performed: no table was created and everything was done using the already existing tables. The main problem was that after copy and pasting an item from one folder to another, the *ID* of the item changed, resulting in the impossibility for comparison.

Second approach

The following assumption was at the base of this approach: an assessment and the items that are linked to it are in the same folder. From the database it was possible to retrieve the information on the folder identification. Filtering the items using that specific folder id it was possible to retrieve all the questions of that specific interactive assessment contained in the folder itself. After repeating this process for all the tests a comparison on the name of the items was done: two questions with the same name were considered to be the same, and therefor anchors.

This approach was not working, since the scope of an assessment was not limited to its folder. It was also possible to use items that were not in the same folder as the test. In addition items in the folder could have not being used but retrieved anyway, resulting in possible linking errors.
Because of the possible errors described above this approach was not selected as the correct one.

Third approach

Each assessment had a structure definition in XML format. The XML description contained all the IDs of the items used in the test. Exploiting the XML parsing available in the database, it was possible to extract the IDs of the different items, retrieve the information necessary to populate the anchor table and perform the population.

The anchor table was a table containing for each item: the ID, the title, the assessment and the classrooms it was used in, the date of usage. Given then a question labeled as an anchor, matching title, class and date it was possible to retrieve all the items (or anchors) used in other tests.

Using this solution the problems relative to the scope of the test and the one relative to items present in the folder but not used was bypassed.

Comparison

As a comparison among the different approaches the following experiment was performed: for a set of different tests, anchor analysis was performed, and in case the

assessment contained any anchors the ID of the exam itself was returned.

Using the *second* approach 10 different assessments were categorized as *containing anchors*, while with the *third* approach 23. As a proof of the error related to items present in the folder but not used I did the difference between the 9 exams from the *second* solution and the 23 exams obtained with the *third* one. One exam was not present among the 23, meaning that a question (anchor) was present in the folder, not used, but still taken into account as valid from the *second* approach. Finally, the fact that 23 assessments are retrieved (in contrast with the 9 in the *second* approach) highlights how focusing only on the folder of the specific exam would reduce drastically the quality of the results.

Chapter 4

Analytics

The results of the IRT analysis were:

- for each item:
  - *difficulty level*
  - *discrimination level*
  - *number of students for each discriminator, before and after pruning*: this was obtained not directly from Xcalibre, but calculated outside of the third party software. The main purpose of having such information was to try to understand in a very basic way the goodness of a question analyzing the pruning. Items with a lots of pruning could have been reported as possibly not good, or with a score range too large with respect to the actual distribution of student's scores
  - *rejected items*: an item not used in the evaluation (rejected) was an item pruned too much or with not enough information to be used in the IRT evaluation. This information was definitely useful to raise a flag regarding the goodness of the question
- ability level for each student
- assessment difficulty: obtained from the *item test function*
- other information not used in the analytic part

The different evaluations of the results consisted in: distribution analysis of anchors, difficulty evolution during the years for interactive assessments, multiple choice items discriminator's analysis, time and score correlation, number of student not answering.

In the following sections an overview of the analysis will be done.

4.1 Distribution for anchors

For each item considered an anchor, and for each option, the number of student choosing the specific option in different years was examined. This type of analysis was useful to understand how good an anchor was. The more different the distributions the worse the goodness of the anchor.

In figure 4.1 we can notice how for the first item the orange and the blue distributions representing different years are more or less the same, while for the second item there is a much bigger difference. This analysis, even though seemed accurate at first was not useful: the number of students is not influencing the IRT evaluation. The only case in which this graph can be useful is when for a given choice no students are present.

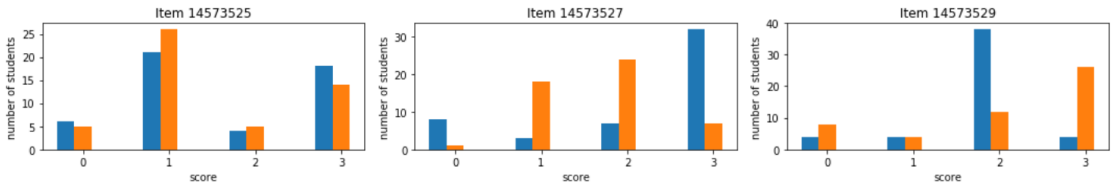

Figure 4.1. Sample of 3 anchors

4.2 Difficulty evaluation

Given a set of tests on the same subject and given to the same class, the evaluation of difficulty over the different years was performed (figure 4.2). This was possible thanks to the equation process: the presence of anchors allows to put on the same scale different tests done in different time periods. In the case reported in figure 4.2, it is possible to notice a significant difference in difficulty among some of the tests: this information could have been exploited by teachers to understand if the level of difficulty was too much (or too little) compared to the expectation, to draw a trend of different exams across multiple classrooms and investigate eventual not expected trends.

4.3 Multiple Choice discriminator analysis

If in a multiple choice item the majority of the students are choosing a particular distractor, it means that the question could be not formulated in the right way, or that the distractors are too easy (or difficult). This type of analysis was helpful to

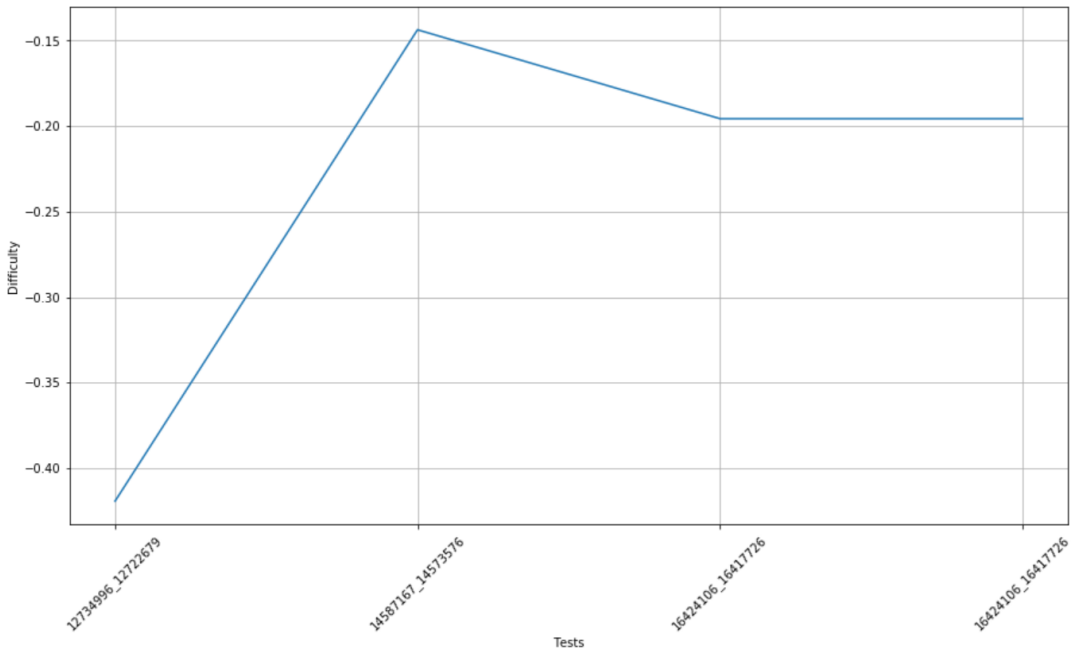

Figure 4.2. Difficulty evolution for 1 test among several years

identify problematic items and eventually modify them.

During the analysis a threshold was specified: if the difference of learners among two distractors was bigger than the threshold then the question was considered problematic. In figure 4.3 the threshold was set to 80% and the items with a uneven distribution are highlighted in red.

4.4 Time-score correlation

With this analysis it was possible to identify the students or the items that could have been problematic from the point of view of the performance and difficulty respectively. Plotting the time used on a question against the score obtained it was possible to highlight interesting patterns eventually usable by the teacher. An example is showed in figure 4.5. It is possible to notice how a student used 20 second and got a really low score. Figure 4.4 gives an interpretation of the plot showed in figure 4.5: depending on which patterns the teacher is more interested in, different part of the graph should be object of focus.

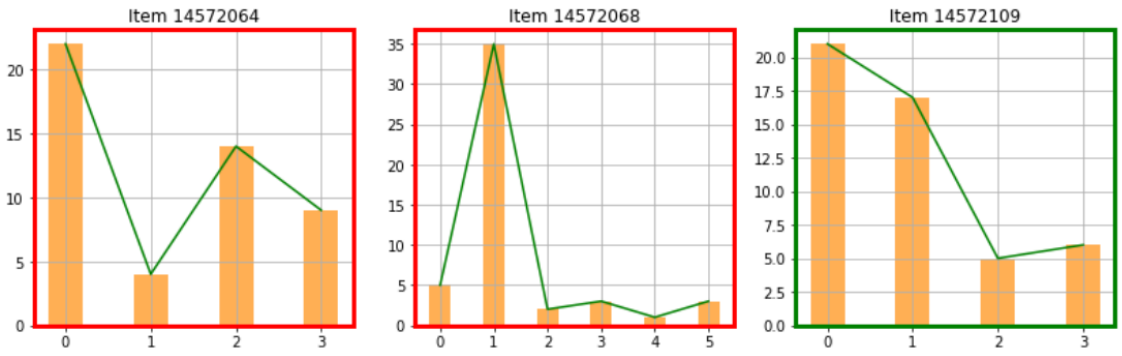

Figure 4.3. Multiple Choice distractors analysis

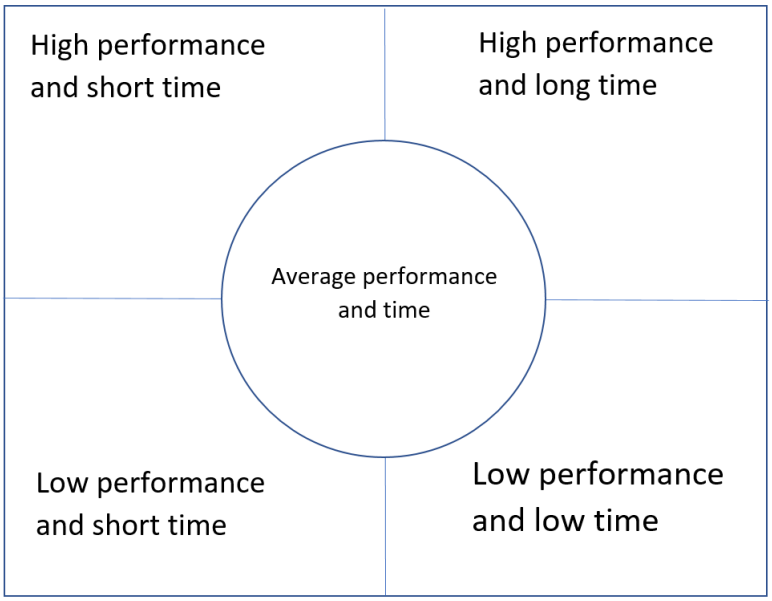

Figure 4.4. Correlation of time score explained

Number of student not answering

For a teacher knowing which items had the highest or lowest number of avoidance rate could be an interesting insight: further inspections could be done on those items, understanding why of such trend. This analysis can show a lot about the knowledge of a set of student on a specific matter: if lots of people did not answer a question on a specific topic it is probable that more work should be done to clarify the concepts.

Figure 4.6 shows an example of this analysis: given the number of students not

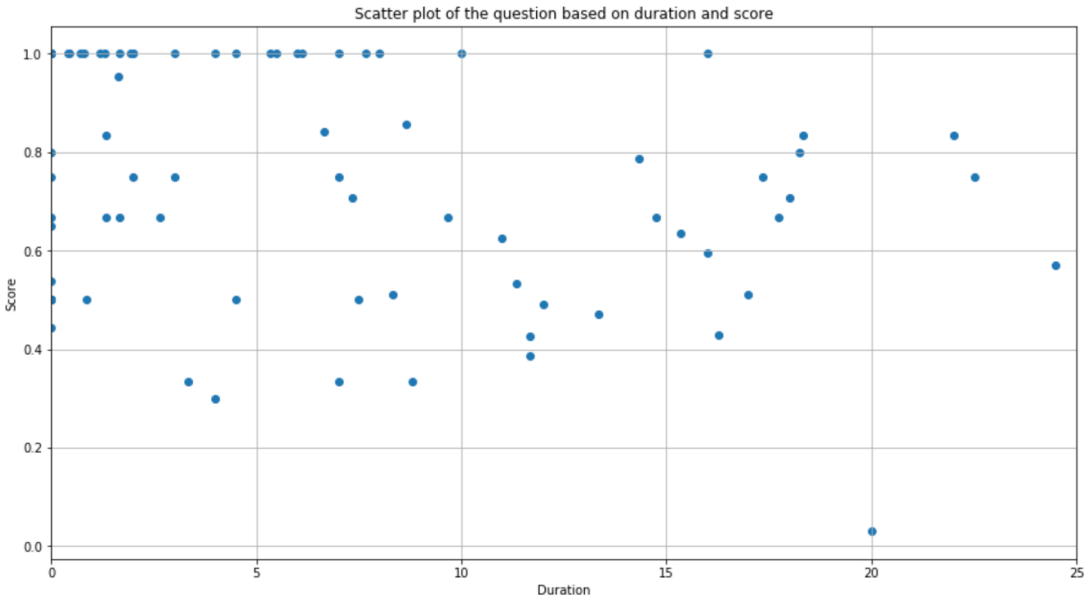

Figure 4.5. Time and score correlation

answering, items are grouped together and an histogram is drawn. As it is possible to notice, for this specific test lots of questions did receive a good response rate, but some of them had more than 15 students out of 23 not answering. Investigating those kind of patterns could be helpful: was the test too long and those questions located at the end? Or maybe the text was not well written? Or students did not understood the topic?

4.4.1 Conclusions

Those are only some of the possible analysis that can be done: more analysis will be done in other sections.

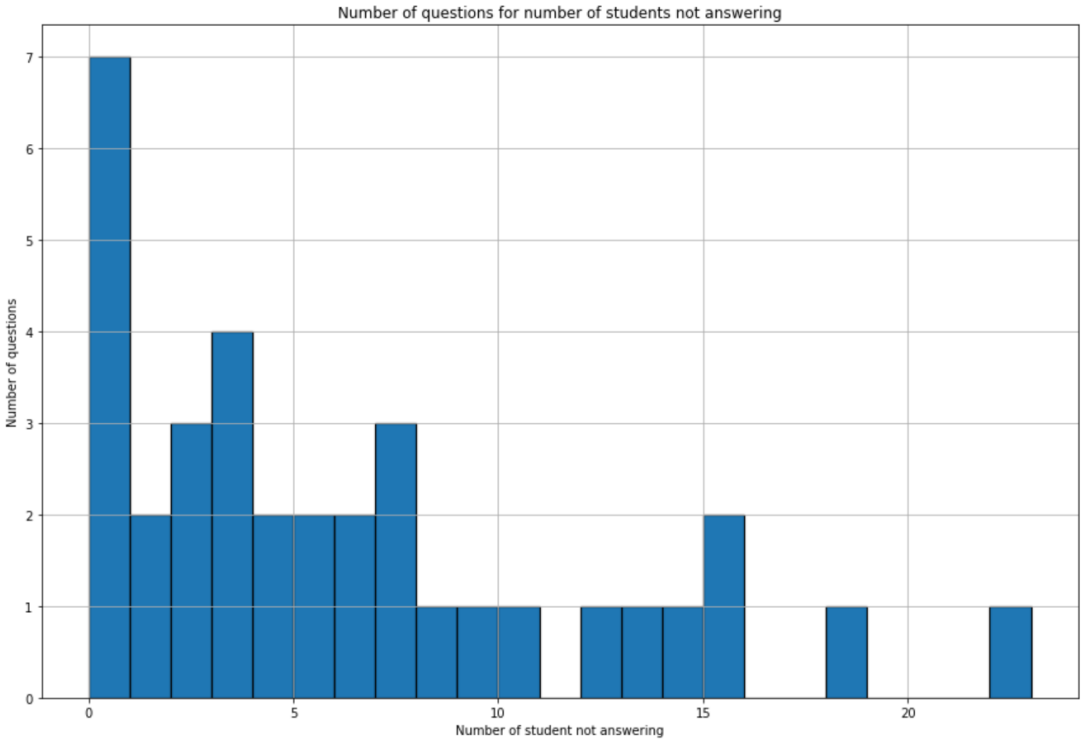

Figure 4.6. Histogram of student not answering

Chapter 5

A new IRT Solution

At this point of the work an overview on IRT with some more clear idea of the capabilities of the methodology were obtained. The following steps consisted in developing a solution for IRT evaluations able to be run on a server, interacting with other processes and exchanging information.

The following sections contain a more detailed introduction of the theory behind IRT with the motivation of the approached used, along with the introduction to the software solution implemented.

5.1 Theoretical approach to IRT

IRT evaluations are based on the Estimation Maximization (EM) procedure, that consists in two different steps, the estimation and the maximization. A shorten version of the process is reported: during the estimation part, the expected frequency and the expected sample size are calculated, as well as the likelihood. In the maximization phase the likelihood is maximized: if the improvement of the likelihood estimation is not sufficiently high, it means that convergence is reached and that the algorithm should stop. Tracking changes in the likelihood evaluation is not the only stopping criterion: as happens during optimization problems, convergence is not always assured; for this reason a maximum number of iteration is also set in order to ensure the termination of the evaluation.

After convergence it is possible to perform the Newton-Raphson procedure in order to find the final item parameter estimates.

5.2 Likelihood

There are several ways of applying maximum likelihood estimation:

- Joint Maximum Likelihood (JML)
- Marginal Maximum Likelihood (MML)
- Conditional Maximum Likelihood (CML)

A very brief explanation of what are the pros and cons of using this different methodologies will be now presented (Susan E. Embretson, Steven P. Reise 2009), followed by a more exhaustive presentation of the preferred approach.

JML

In JML estimation, unknown trait levels are handled by using provisional trait level estimates as know values. The provisional trait level estimates are improved using estimates of the item parameters, that are subsequently used to improve the trait estimates. JML has been extensively used in the early evaluation of IRT parameters, and it can be used with both 1PL and 2PL models. The main problem is that the evaluations are biased, even though correction factors can be used. Due to the bias introduction also the evaluation of standard error is not optimal. Also important is the fact that JML models are inconsistent for fixed length test: adding more students to a fixed length test does not results in a better estimation of item parameters and item fit. Finally, for a student with perfect score (all wrong or all right) it is not possible to obtain estimates.

MML

In MML (Bock and Aitkin 1981) handling of unknown trait level is done by expressing the response pattern probabilities as expectations from a population distribution. The observed data are regarded as a random sample from a population. The first implementation of this method was produced by Bock and Aitkin 1981, who developed an Estimation Maximization (EM) procedure for the estimation of parameters and traits. The EM paradigm is similar to the one used in JML, but for MML the iterations successively improve the expected frequencies for correct responses and trait level.

MML has several advantages. First of all can be applied to all models, also multidimensional ones, being efficient for long and short tests. Regarding perfect score, it is possible to obtain estimations also in these cases. Finally the MML estimation of item standard errors may be justified as good approximations of expected sampling variance of the estimates. Some disadvantages are already present: it is necessary to assume a distribution for trait level (not clear if this is a disadvantage), and could be difficult to write optimal programs to compute the estimation process.

CML

In CML trait level that are unknown are handled by expressing the response pattern probability without including the trait level parameters. This is possible if a sufficient statistic is available in the data for estimating trait level. Sufficient means that the specified statistic is more than sufficient to estimate the trait levels.

One advantage is that no assumed distribution is required for the trait levels, since the unknown person parameters are left out of the estimation equations for items. In addition standard errors for the item parameters are more justifiable as representing sampling error as compared to JML. Unfortunately CML has several disadvantages. The most salient one is that it is applied only to 1 Parameter Logistic Model (1PL), or Rasch model, so 2PL, 1PL and other models' parameters can not be estimated using CML.

Other estimation models

There are other estimation models, like the Quasi Monte Carlo EM (QMCEM) or Monte Carlo EM (MCEM). These methods were not taken into account at first, and maybe will be used for comparison in the future.

5.2.1 The approach used

I decided to use the MML for the following reasons:

- can be applied to all models and has several advantages over JML and CML as specified before
- is the same method used in the commercial software Xcalibre, and comparing results obtained with the same method was preferred in order to have a measure of the goodness of the evaluations

5.2.2 Mathematical view of MML

In this section I will try to report the mathematical procedure without going too much into the particulars. It is important to have a general understanding of the procedure and of the passages involved in the estimation.
First of all it is necessary to define the probability of assessing correctly (or not) an item, as

$$P(X_{is} = 1|\theta_s, \beta_i) = \frac{exp(\theta_s - \beta_i)}{1 + exp(\theta_s - \beta_i)} \tag{5.1}$$
$$P(X_{is} = 0|\theta_s, \beta_i) = \frac{1}{1 + exp(\theta_s - \beta_i)}$$

where i is the item, s is the student, θ is the trait level of the student and β is the difficulty level of the item i. It is possible to rewrite this as

$$P(X_{is}) = P_{is}^{X_{is}} Q_{is}^{1-X_{is}} \tag{5.2}$$

that in logarithmic form becomes

$$ln(P(X_{is})) = ln(P_{is})X_{is} + ln(Q_{is})(1 - X_{is}). \tag{5.3}$$

Equation 5.3 represents the logarithmic probability that a student will assess correctly item i.
Given a response pattern identified as

$$X_{1s}, ..., X_{ls}$$

the likelihood of the response pattern with l items for the person s, conditional to the trait level and the set of IRT item parameters $\underline{\beta}$ is

$$P(X_{1s}, X_{2s}....X_{Is}|\theta_s, \underline{\beta}) = \prod_i P_{is}^{X_{is}} Q_{is}^{1-X_{is}} = P(\underline{X_p}|\theta_s, \underline{\beta}). \tag{5.4}$$

MML models the probability of observing a response pattern in the population. A probability can be modeled for each response pattern $\underline{X_p}$ that can be observable in the population itself. The number of students having a given pattern is expressed with n_p. For a specific level of θ the probability of a response pattern can be evaluated as in equation 5.4.
Even though trait levels are not known for the students observed, it is possible to assume a priori distribution: Bock and Aitkin 1981 suggested the distribution of the trait levels can be assumed as $N(0,1)$. It is also possible to estimate the ability distribution after the first iteration: the preliminary trait levels can be used to

estimate the distribution, that is updated at every iteration. Given the population distribution, the probability of observing a given pattern $\underline{X_p}$ in a random sample of the students can be modeled as

$$P(\underline{X_p}|\beta) = \sum_{q}^{Q} P(\underline{X_p}|\theta_q, \underline{\beta})P(\theta_q). \tag{5.5}$$

Equation 5.5 involves:

- the sum over all the possible values of θ_q
- the probability of having a specific pattern $\underline{X_p}$ under θ_q
- the probability of a given θ_q in the student distribution

Since the distribution of θ is continuous - $N(0,1)$ - it is possible to transform 5.5 from a sum to an integral in the following form

$$\int_{-\infty}^{+\infty} P(\underline{X_p}|\theta_s, \underline{\beta})g(\theta)d\theta \tag{5.6}$$

where $g(\theta)$ is the probability density of θ.
It is possible to solve this integral using Gauss-Hermite quadrature, with a equal spacing among the quadrature points.

At this point it is possible to identify the data log likelihood in the following way

$$lnL(X) = \sum_{p} n_p lnP(\underline{X_p}|\underline{\beta}). \tag{5.7}$$

To conclude: given a set of θs and βs it is possible to evaluate every single response pattern likelihood as in equation 5.5, and with that result evaluate the data likelihood.

Why the data likelihood is important?

The data likelihood is important because it express the probability of having a set of response patterns given θs and βs: the highest the value of the likelihood, the most probable is to have a given set of response patterns given the student population.

What it can be used for?

Maximizing the data likelihood means that the response patterns observed are the most probable given certain θs and βs: since when an IA the only data directly observable are the response patterns, it is possible to maximize the data likelihood and obtain a measure of the trait level of each student and IRT parameters for each question in the IA.

5.2.3 Newton-Raphson Estimation

Maximum likelihood estimation can be fine tuned using Newton-Raphson estimation. This type of estimation takes into account the first and the second derivative of the function. In order to calculate the maximum (or the minimum), updates to the current value are done using the derivatives. In case of maximum search, negative first derivative means that the value is too high. On the contrary, having a first derivative that is positive means that the value is too low.

The second derivative tells us how good of a maximum we have. In the maximum of the function, the second derivative is negative: the lower it is, the better the maximum.

Some steps of the Newton-Raphson estimation methods can be run at the end of the maximum likelihood estimation, to fine tune the result.

5.3 Merlin

As explained in section 3.2.3 and section 2.2, Xcalibre represented a solid solution but with some main drawbacks that did not allowed a perfect integration of the software. It was clear from the beginning that a software with different features was needed: for this reason, after assessing the goodness of results obtained by IRT, some researches for different solution other than Xcalibre were made.

The choice made was to develop a new software based on the R package *mirt* (Chalmers 2012): this package allows to exploit the main IRT features, and supports as well multidimensional items. Before implementing a solution using the package, integrate the solution in the commercial software and provide parameters to the public, a comparison with Xcalibre was performed. It was necessary to be sure of the goodness of the results, and in case of discrepancy in the values different packages would have been used.

Mirt was not the only package taken into consideration, but was one of the most complete that I was able to find. It "allows analysis of dichotomous and polytomous response data using unidimensional and multidimensional latent trait models under the Item Response Theory paradigm" (Chalmers 2012), and implements (as well as MML) other estimations methods.

The option of developing the product starting from scratch was also taken into consideration following the theory described in Bock and Aitkin 1981. The main problem with this approach was the timing: developing this kind of solution, testing it, and optimizing it would have required way too much time. For this reason *mirt* was chosen.

5.3.1 First version

The first version was developed in order to have a first evaluation and start comparing the results against Xcalibre in order to evaluate the performances. The flowchart in figure 5.1 summarizes the logical process followed.

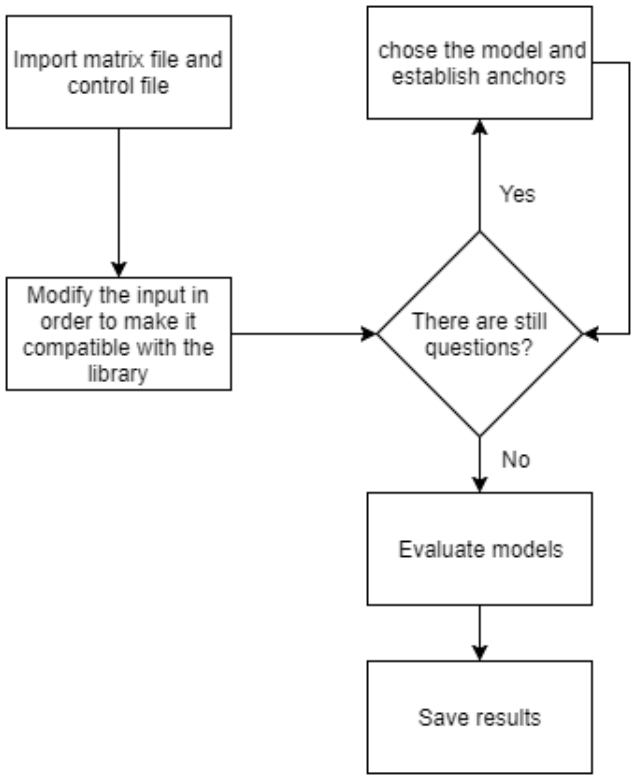

Figure 5.1. Merlin workflow

Import matrix and control file

The file imported were in the same format of the one used by Xcalibre, with the only difference of headings in the csv file. The changes were done to make it more easy to import data as data frames in R. The choice of using the same format was done for a compatibility purpose: in this way the same python code used to run Xcalibre was used to run Merlin, without any necessary big changes. The fields contained in the files were the one described in the subsection 3.2.1.

Input modification

After loading the files in R, some changes were necessary in order to obtain the desired format. Using data frames was a huge advantage, because allowed to manipulate data in an easy and user friendly way. Moreover, the format used by the *mirt* required some minor changes, like setting unanswered items to *NA*.

Model and anchors

For each item the model was chosen based on the type of the item itself (polytomous or dichotomous). It was possible to retrieve this information analyzing the *number of different alternatives* field in the control file: 2 alternatives meant dichotomous while more than 2 polytomous. In addition if the item was marked as an anchors the values for discriminator and difficulty level(s) were set. In order to set the parameters, the structure of the model was modified and saved in variables given to the model estimation function. Specifying to not use the anchors in the *ML* process, since already estimated, was also necessary.

Model evaluation results

After the definition of the models and the evaluation for each item the following results were obtained:

- discriminator and difficulty level(s) for each item
- trait estimation for each student
- items fit

5.3.2 Second version

In the second version, in addition to some changes in the structure of the program (a bit more function oriented) the following functionalities were added:

- report containing a summary of the Item Response Function (IRF) for each item
- data representing the Test Response Function (TRF)
- comparison of the Test Response Function (TRF) of different tests
- comparison of the students' level
- distractors and score analysis

Report with IRFs

As explained in section 2.1, the IRF is a function that gives the probability for a given user to assess correctly the item. Students with lower ability has a lower probability of answering correctly, while having a greater ability increases the chances of doing so.

The *mirt* package provides a function to evaluate the IRF for each item: using the standard methods provided, it was possible to plot a summary of the IRFs or each one of them singularly. An example of report is showed below in figure 5.2.

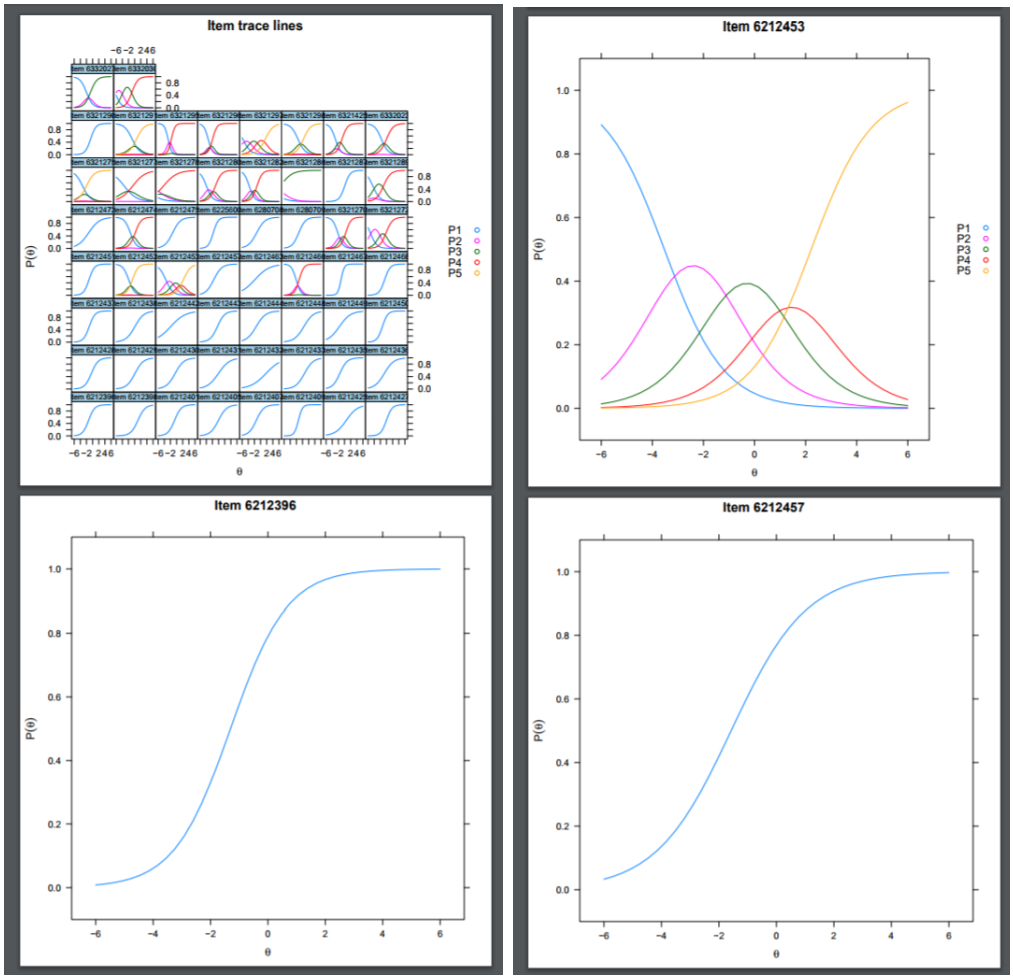

Figure 5.2. Example of report, first two pages

TRF

The Test Response Function (TRF) is the response function of the test: can be seen as the IRF for the whole IA.

In order to obtain one TRF and to compare it with TRFs of different IAs, a couple of changes in the *mirt* library were done. The behaviour of the commands to print the TRFs plots was to directly show the graphs, without returning the single points of the curve. For this reasons, storing the points of different TRF and then plotting them against each other was impossible. In order to solve the problem modification to the function used to plot the TRF were done, making it return the evaluation of the points instead of the plot. The data frame containing the necessary information to plot the TRF was then saved as a csv file and used for the comparison. The choice of saving it in a csv file came from the necessity to have data available among different runs of the software: another solution would have been storing data in the cache.

In figure 5.3 it is possible to observe a comparison of the TRF of three different tests:

- high theta levels: the three different tests does not differ that much; it is possible to start noticing a difference moving towards lower values of thetas
- low theta levels: the difference here is higher; depending on the IA students can have a different normalized score

TRF is a measure of the difficulty of the assessment: depending on the level of theta when the normalized expected score is equal to 0.5, we can have more or less difficult test. A more difficult test will have a middle point moved towards +6, while a less difficult test will have a middle point moved towards -6 (supposing the the range of difficulties goes from -6 to +6).

There are different ways of obtaining a TRF:

- expected score: $\sum_i^N \sum_j^n j_i P(X = j_i)|\theta)$
- expected numner of correct assessed items: $\sum_i^N \sum_j^n P(X = j_i|\theta)$

with:

- N: number of items in the IA
- $P(X = j_i|\theta)$: probability of obtaining the specific score for item i given a specific θ
- j_i: score for item i

The two ways of obtaining TRF are essentially the same once normalized, even though when not normalized they express a different quantity.

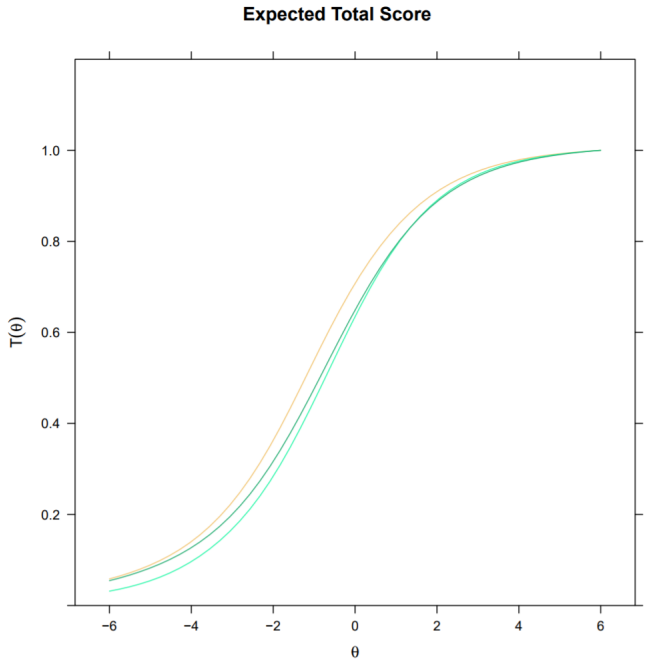

Figure 5.3. TRF example

Students' ability

Regarding the students' ability, a couple of function in the library allowed to extract the ability for each student: after saving these data, it was possible to exploit it and use it for comparison. The two graphs in figure 5.4 shows the distribution of students' ability for two different exams. Using the histogram it is possible to notice in a more clear way how the blue distribution has a mean slightly higher than the orange distribution. Overall the two ability distributions (measured with different tests that had been equated) coincides.

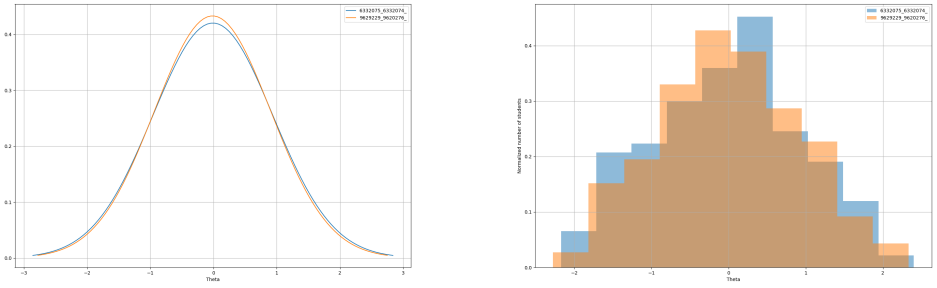

Figure 5.4. Student abilities comparison

Distractors and score analysis

The concept of analyzing the goodness of a question is being approached different times. Focusing on distractor analysis I was trying to understand if it was possible to evaluate the goodness of a question analyzing the result obtained. The starting point was a definition of *good question*: given an item with several distractors, the item is considered good if for each student's ability level a distractor can be identified to address specifically that level. In other words, to obtain a correct evaluaiton of pupils, for each level of ability a distractor should be present. This assumption was oversimplified, but was giving a solid starting point for the analysis. It is very difficult to create a distractor for each student's level, and in addition randomness and external factor should be also taken into account. The assumption behind this definition was the following: if an item contains one distractor for each level of ability, then each student can obtain the specific score related to his ability level.

In order to perform the analysis, the range of abilities was divided in intervals, and for each interval the probability of selecting a distractor over the others was evaluated.

In figure 5.5 there is an example for a polytomous item: for each score the probability of having that specific score varies depending on the ability level. In this example 15 bins are used, with the blue bars indicating the number of students in that bin (normalized with respect to the bin having the maximum number of students).
Is is possible to notice different problematic:

- for all the ability levels seems like that was more probable to have the maximum score (2 in this case)
- the result was biased: depending on the number of student in each bin, it is possible that the probability evaluated would not represent the real one. This could happen when there are too few students in the bin, some of them guess correctly the right answer and suddenly the probability of getting the maximum score goes up unexpectedly.
- it is almost impossible to write distractors in a way that each one of them is addressed specifically to a set of student with a given ability: there are too many factors involved in the selection of a distractor and I am not sure it is possible to do this kind of analysis (at least not without a sufficient big amount of students)

Given this final considerations, the analysis was not successful and evaluating the goodness of an item in this way was inconclusive. A different type of analysis was then proposed only for dichotomous items: estimating the average ability of the

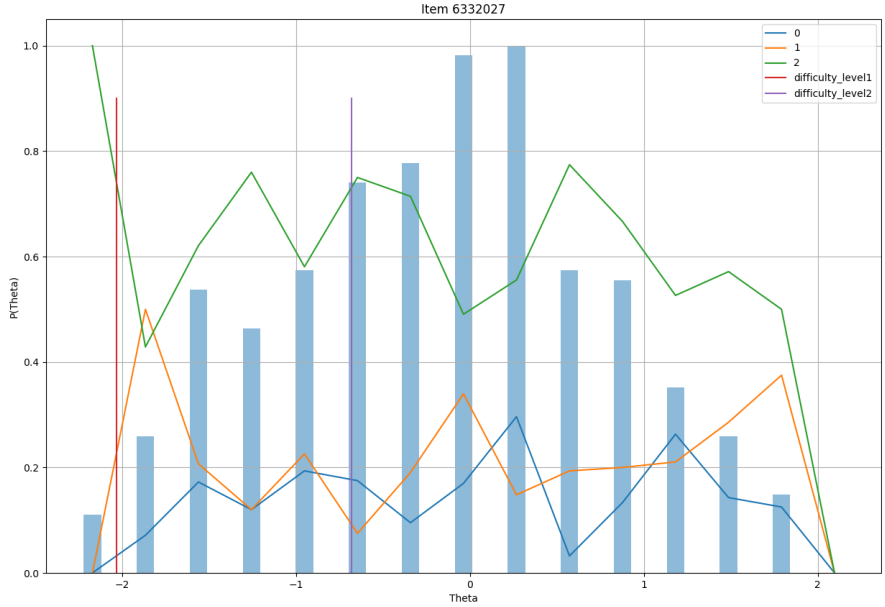

Figure 5.5. Distractor analysis

students choosing each distractor.
This kind of analysis can be useful to understand if the definition of the distractors is wrong: if for one of the wrong options the average ability is higher than the one of the correct option, means that good students are usually choosing a wrong answer. This can be an indicator of an item that is written in a wrong way, or maybe in an unclear way.

5.3.3 Third version

The third version of the solution was mainly a refactoring of the second one. The biggest difference was in trying to divide the different features implemented, organize them in functinos and put the functions in different files. It was more or less like creating classes with specific purposes and combining them in a common method afterwards.
In addition a script to emulate student answers was written. The following steps were followed in order to obtain the simulated user answers:

- generating distribution of theta (student's ability) in a range from -6 to +6
- generate a normal distribution for the discrimination

- generate a normal distribution for the difficulties
- generate patterns for each student in order to approximate the distributions of discrimination and difficulties

The results obtained in the simulation have been used as a data set for comparison of different software (Merlin and Xcalibre): it was necessary to have a set of student big enough to evaluate with precision the IRT parameters.

5.4 Comparison with Xcalibre

In this section a comparison of the results obtained with *mirt* library and the results obtained with the commercial product Xcalibre is done. Different aspects were taken into account:

- χ^2
- comparison of discrimination and difficulty for each item
- comparison of theta (ability) estimations
- comparison of the time needed for the estimation

The comparison where done using a proprietary data set and an emulated one. The reason an emulated data set was used it was the following: in IRT a significant number of students should assess the item in order to have a good estimation: the proprietary data set was not big enough, and in order to have a better evaluation of the result a bigger data set was emulated (section 5.3.3).

Chi-Square Comparison

The evaluation of Xcalibre and Merlin using χ^2 has been done using the proprietary data set, containing around 400 students and 50 different questions. The result of the comparison is shown in figure 5.6.

The blue line was computed doing the difference among χ^2 of Xcalibre and Merlin. The green line is the difference in degrees of freedom, while the orange line is the mean of the χ^2.
In order to have some kind of meaning different assumption should be made: a χ^2 difference test is meaningful only if the models in question are nested models, i. e. one of the models could be obtained simply by fixing/eliminating parameters in the other model. Focusing on Xcalibre and Merlin:

- both of them were using the same models for the items: 2PL (Birnbaum 1968) and Samejima's graded response model (Samejima 1969)

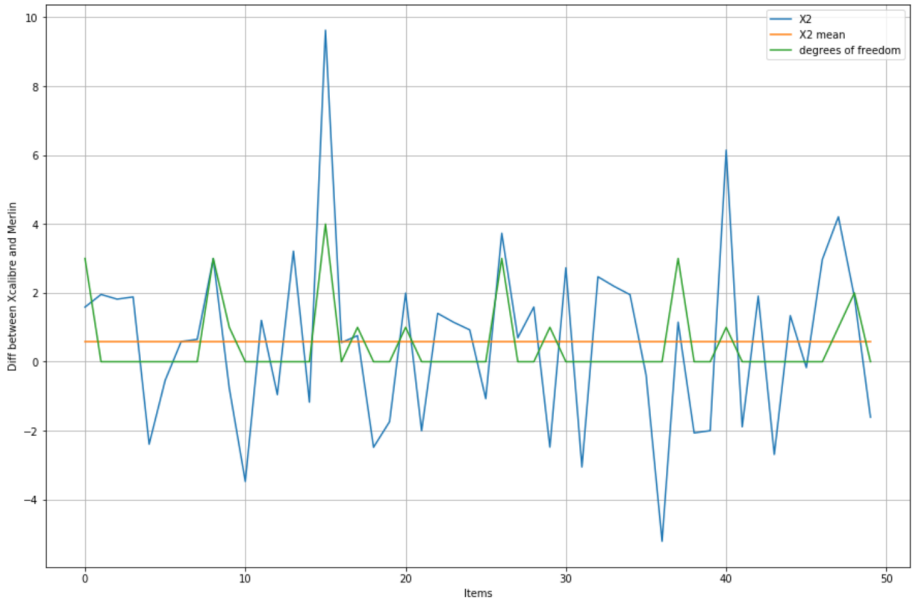

Figure 5.6. χ^2 comparison

- both of them were using Bock and Aitkin 1981 methods in order to obtain an estimation of the parameters
- some differences were present in the two approaches, for example the initialization of the parameters using proportion-correct and point biserial correlation

Given the facts described above, I thought that Merlin and Xcalibre could have been seen as nested models.
Looking at the graph (in particular looking to the average) is clear that the χ^2 parameter is overall smaller for Merlin, and that implies a better fit of the model over the data.

Discrimination and Difficulty comparison

In order to understand the goodness of the result obtained, a comparison of discrimination and difficulty levels was done. The values calculated in this section are relatives to the emulated data set. An explanation of how the data set was emulated can be found in section 5.3.3. The emulated data set consisted in 2000 students, 20 polytomous questions with 5 distractors each. The distribution of difficulty levels was a normal one with mean 0.3 and variance 1. The distribution of levels was a normal one with mean 0.2 and variance 0.3.
The reasons why I decided to show the comparison using the emulated data set are the following:

- bigger set of students
- possibility to compare against the true distribution for discrimination and difficulty levels
- it was possible to have analysis regarding the error that were not performed on the proprietary data set, since no true value was present

Figure 5.7 represents the box plots with the percentage of error between the original *b*s (difficulties) and the ones obtained using Merlin, calculated as $|original - merlin|/|original| * 100$.

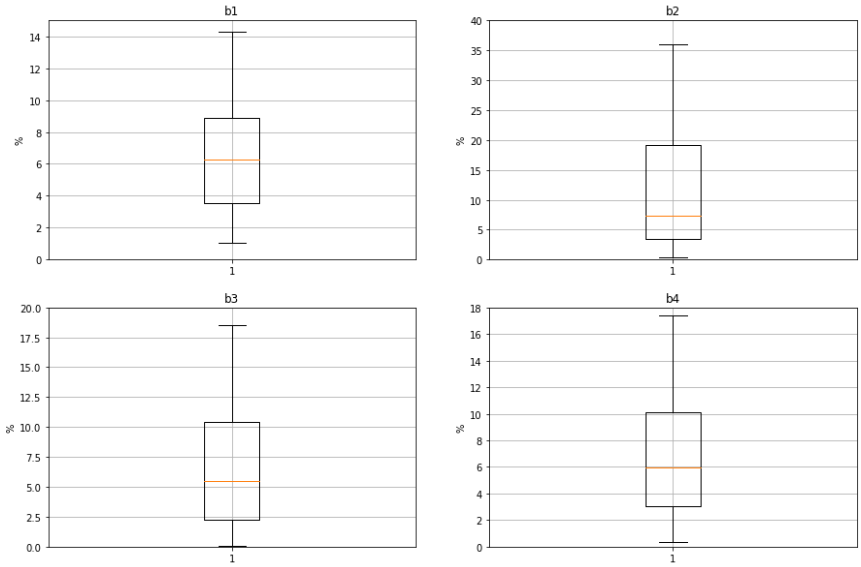

Figure 5.7. errors on b using Merlin

Figure 5.8 represents the box plots with the percentage of error between the original *b*s and the ones obtained using Xcalibre, calculated as $|original - xcalibre|/|original| * 100$.

Figure 5.9 shows the distribution of the errors showed before. The blue line is Merlin, while the orange one is Xcalibre. It is possible to notice how for Merlin the average error is usually smaller compared to one obtained using Xcalibre. Focusing on the variance (reported in the title of each image in figure 5.9 for both Merlin –M– and Xcalibre –X–), it is possible to imply that there are more evaluations close to the true value, and so with less error.

Figure 5.10 shows the estimated values for each of the 20 simulated items for both Merlin and Xcalibre. The results are quite close to the true value for both of the programs, with the difference that Merlin needed less time and less iteration in order to converge and find the estimates of discriminators and difficulties.

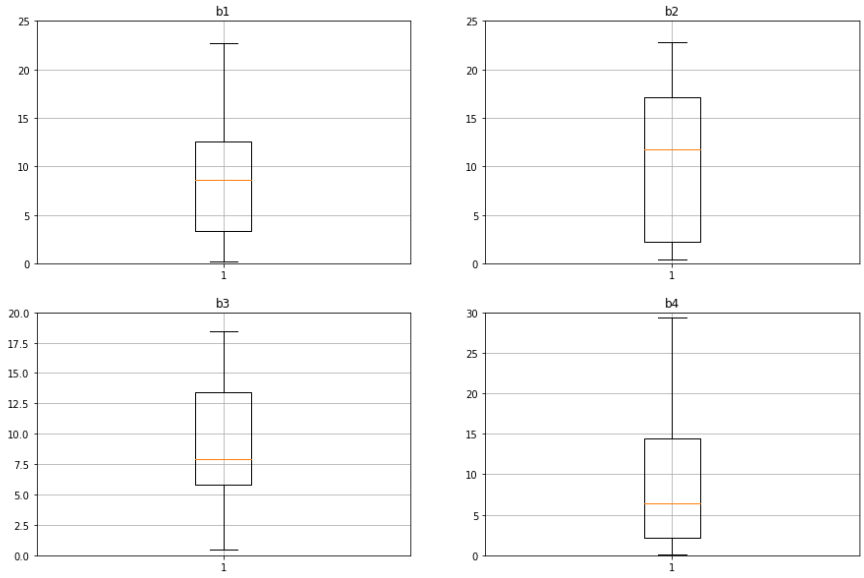

Figure 5.8. errors on b using Xcalibre

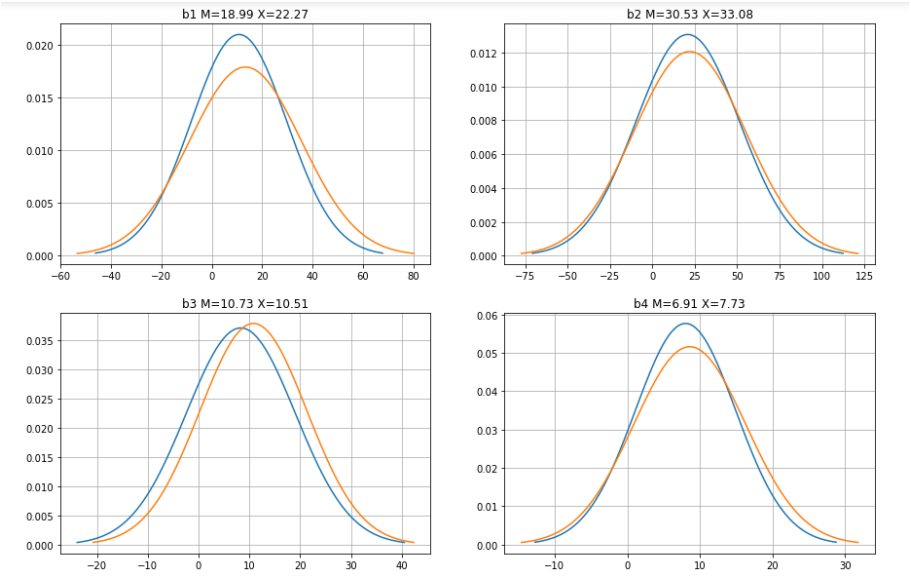

Figure 5.9. distribution of errors on b. Blue: Merlin, Orange: Xcalibre

Time comparison

Comparison on the results is not the only evaluation that can be done: also timing was an important factor. Since we were planning on running the evaluations in the cloud and for a lot of users, having something fast and parallelizable was preferred.

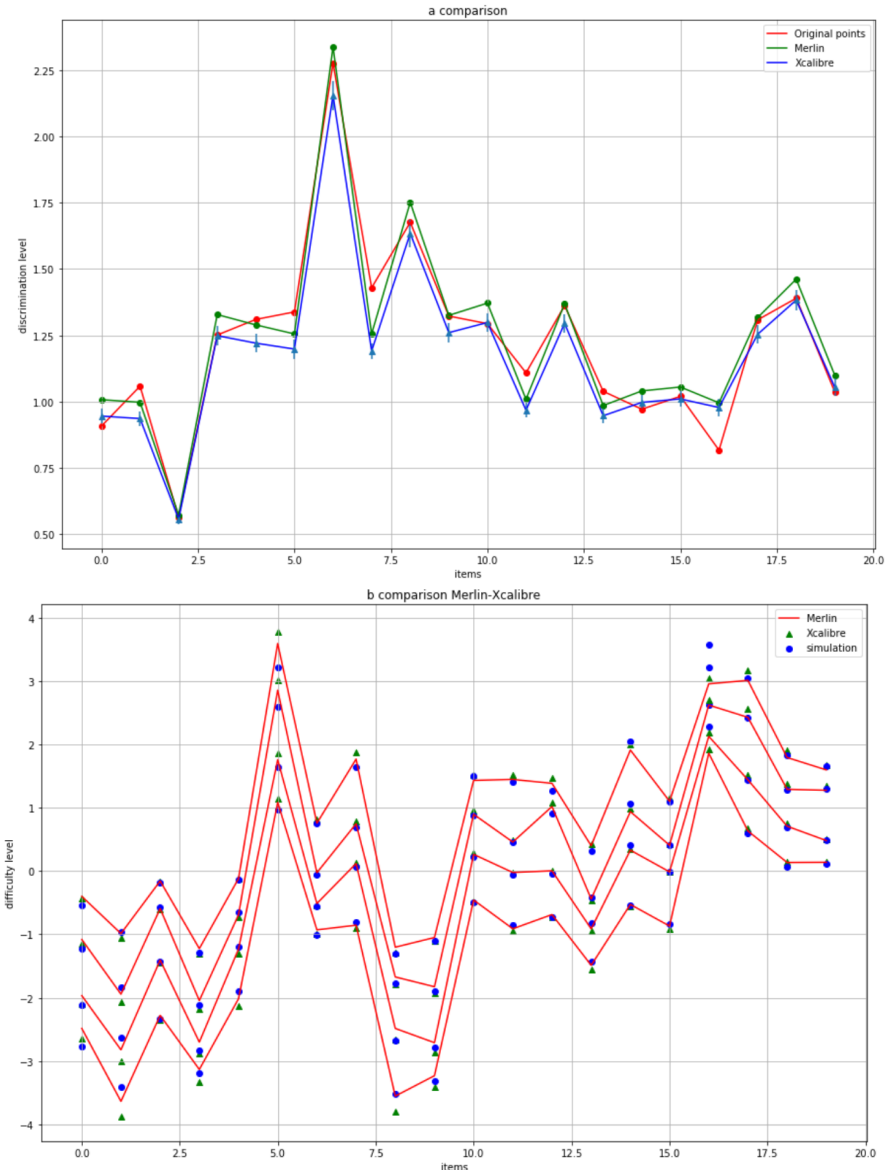

Figure 5.10. a and b comparison

The time evaluation was done in the following way:

- Xcalibre: the software was launched using a library able to interact with the graphical interface. The evaluation was started at the beginning of the graphical interaction and stopped when terminating the evaluation with Xcalibre
- Merlin: after the creation of a new process, the execution of Merlin was delegated to this process that would have returned the control to the main program

at the end of the evaluation. The time measuring was started when the new process was spawned and terminated with its termination

In table 5.1 there are the settings used for running both Merlin and Xcalibre. The question marks are due to the fact that in the Xcalibre documentation or in the software settings was not possible to find any information about the particular setting. In table 5.2 there are the results of the time evaluation: Xcalibre is slower, also due to the fact that interactions with the graphical interface make it more time consuming. The difference in the iterations of the EM process is not that high.

Table 5.1. General settings

Software	Method	optimizer	priors	accelerate	TOL	quadPoints
Merlin	EM	BFGS	NA	Ramsay	0.001	61
Xcalibre	EM	?	CTT	?	0.001	40

Table 5.2. Time comparison

Software	Time (s)	Iterations
Xcalibre	18.52	28
Merlin	3.53	19

Chapter 6

First evaluations in the platform

The preliminary work to understand the feasibility of the project was done, but before implementing the final solution some additional analysis were done. I started working with Classical Test Theory (CTT) in order to learn the system organization of the classes and give some more insight about exams using standard evaluations.

Traditional reports and the data reported in them allow teachers to have a clear evaluation of the students and to grade them based on their performance on a specific test. Reports are very useful for both teachers and students: while the teacher can see what are the questions that were more troublesome for the students and understand what are the topics to focus more on, the students can see the errors, the corrections and eventual explanations to these corrections.

Using the system was always possible to evaluate learners level but not the quality of a test or of a question. Trying to achieve evaluation of items and not only student was the goal.

6.1 A theoretical approach to CTT and statistics

6.1.1 Observed and True Score

The observed score is the score that is obtained after an evaluation of an assessment, and it is composed by true score and an error. The observable score can be written as $X = T + E$, where T is the true score and E is the error. It is not possible to observe the true score of a student directly. CTT is still used nowadays but it is not possible to compare the results obtained across tests unless some major constraints are matched. To allow comparison two test must be parallel: parallel tests have the same difficulty and they are specifically designed to evaluate the same skills. It is not easy to have parallel tests, and precise question and test planning should be done beforehand to achieve the result.

6.1.2 Statistics

The evaluations presented to the end user were the following:

- **Order**: the position of the question in the test
- **Exposed**: the number of students, among those who submitted the test, who opened the question. Equals to the sum of "Attempted" and "Omitted"
- **Attempted**: the number of students who answered the question
- **Avg. time**: the average number of seconds spent on the question among the students who answered the question
- **Omitted**: the number of students who opened the question without answering it
- **Not exposed**: the number of students who did not open the question
- **Correct**: the number of students who answered the question correctly
- **Average score**: the average score on the question among the students who submitted this test
- **Max score**: the maximum score on the question
- **P-value**: the P-value is the normalized average score on a question. This means that the maximum value of the P-value is 1, which happens if all the candidates answer the question correctly. It is worth noticing that the P-value can differ from the average score, even if the question has a maximum score of 1, because the P-value only takes into account the number of candidates that were exposed to the question
- **Correlation**: between -1 and 1. The correlation is the extent to which the question score correlates to the total score. Negative or very low positive values may indicate that the question does not discriminate (differ) well between students of high and low ability
- **Corrected correlation**: as the correlation, but the score of the item analyzed is subtracted from the total score of the test

The evaluations strictly based on CTT were *p-value*, *correlation* and *corrected correlation*.

6.1.3 P-value

The p-value, or also proportion correct value, is used to describe the *difficulty* of the item taken into account. As the description suggests, the p-value can be calculated using the proportion of students answering correctly to a question. The problem that was encountered was that the correctness of an answer was not always straight forward: if a student had a score of 5 out of 7, should have the answer been considered correct or not? In order to solve this minor problem, the average score was used, and then normalized using the maximum score for that specific question.

There are different choices regarding the evaluation of the average score:

- on the students that saw the question
- on the totality of the students that took the test

The decision to calculate the average score for the normalization using only the students that saw the question was taken: there was no information regarding the students that did not see it. Assuming a wrong answer (score equal to zero) would have impact the average score in a negative way: it could have been that a specific student not exposed to a question would have answered it correctly in case of exposure. The calculated p-value was the following:

$$\frac{\sum_i score_i}{exposure} \tag{6.1}$$

where *exposure* is the number of students seeing the question. Seeing the question means a students spending more than 0 seconds on that specific item.

6.1.4 Correlation

The correlation shows the relationship among score of the question and total score of the test. It could be seen as the capability of the item to discriminate between high and low performance students. A low correlation means that the impact of a correct or incorrect answer on the total score is not relevant: as a consequence discriminating between good and bad students using this question would be impossible. The opposite with an item with high correlation.

The correlation was initially obtained in different ways depending on the type of item analyzed:

- dichotomous: point-biserial correlation

$$cor^{(j)} = \frac{M_1 - M}{\sqrt{v}} \sqrt{\frac{p_j}{1 - p_j}} \tag{6.2}$$

where M_1 is the average test score of the students answering correclty, M is the average test score of all the students, v is the variance of the test score for all the students, p_j is the average question score

- polytomous: pearson correlation

$$cor^{(j)} = \frac{\sum_i (y_{ij} - \overline{y}_{.j})(y_{i.} - \overline{y}_{..})}{n\sqrt{v_{..}v_{.j}}} \tag{6.3}$$

  - n: number of students
  - y_{ij}: result for student i to item j
  - $\overline{y}_{.j}$: mean of the total scores for item j
  - $y_{i.}$: total score for student i
  - $\overline{y}_{..}$: mean of the total scores of all the students
  - $v_{.j}$: variance for item j
  - $v_{..}$: variance for all students/items

Since the point-biserial correlation is a simplified version of the pearson correlation for dichotomous items, the decision to use the pearson correlation for both polytomous and dichotomous items was made: in order to verify the equality of the different evaluations some investigation was done using excel sheets.

6.1.5 Corrected correlation

As said in Guilford, J. P. 1954 "when an item is correlated with the total score of which it is a part, the value of the correlation tends to be inflated. The shorter the test, the greater this inflation is likely to be. Even if all the items correlated actually zero with what the total score measures, and if all item variances were equal, each item would correlate to the extent of $\frac{1}{\sqrt{n}}$, where n is the number of items. Under these conditions, with 25 items the correlation would be .20 for all items. This is slightly above the .01 confidence level for a sample of good size, and it might therefore be taken as indicating a significant correlation if one were net aware of the possible spurious origin". For this reason the necessity of evaluating a corrected correlation.

The corrected correlation was evaluated as the normal one, with the difference that when calculating $y_{i.}$ (total score for student i), $\overline{y}_{..}$ (mean of the total scores of all the students) and $v_{..}$ (variance for all students/items) the score of the item was subtracted from the total score of each student.

6.1.6 Limitation of the statistics

The statistics calculated were only available in the context of a test for the following reasons:

- position in the test: depending on the position in the test, the item could be reached or not; having statistics across multiple tests where the position could play an important role in the definition of those statistics was not correct

- length of the test: depending on the length of the test the item was used in, students could have different performances depending not on the question itself. A student with a shorter test could be less tired with respect to a student with a longer test when answering a specific question

- classroom in which the question was used: it was possible that the question was given to students of different classes (for example fourth and fifth grade). Grouping students with clear different skill levels in order to evaluate a common statistic was not relevant in this case

A lot of discussions were done before starting with the evaluation, and deciding to operate on tests instead of on the entire population had as a result an increment of the different possible problematic. First of all the groupings on the data and the evaluation of the parameters necessary for the analytic evaluations: generating queries to obtain such results in a scalable way and without overloading the server was not a trivial task. In addition defining a structure to show those values in a user friendly and intuitive way was another problem that took different iterations before having a final implementation.

6.2 System

This section gives an overview of the system in which the new statistics are shown. The system allows to:

- create items: it is possible to create different types of items that will be than used in a questions set

- create quesiton sets: each question set is associated to one or more items; each one of them can be used in one or more question sets

- create a test: a test is associated to one question set, and a specific question set can be used in one or more tests

After the creation of a test, the teacher can decide to open the test for a given period of time (to make it available to students), and grade it afterwards. During the opening period students can log in, start the test, answer the questions and submit the answers (an example of question and submission can be seen in figure 6.1).

The answers will be used for grading, that can be done manually or automatically. Automatic grading is done by the system, and the teacher can either review the grades or accept them as they are; manual grading is done entirely by the teacher. In order to grade automatically the test for each student, grading rules are defined for each question during their creation.

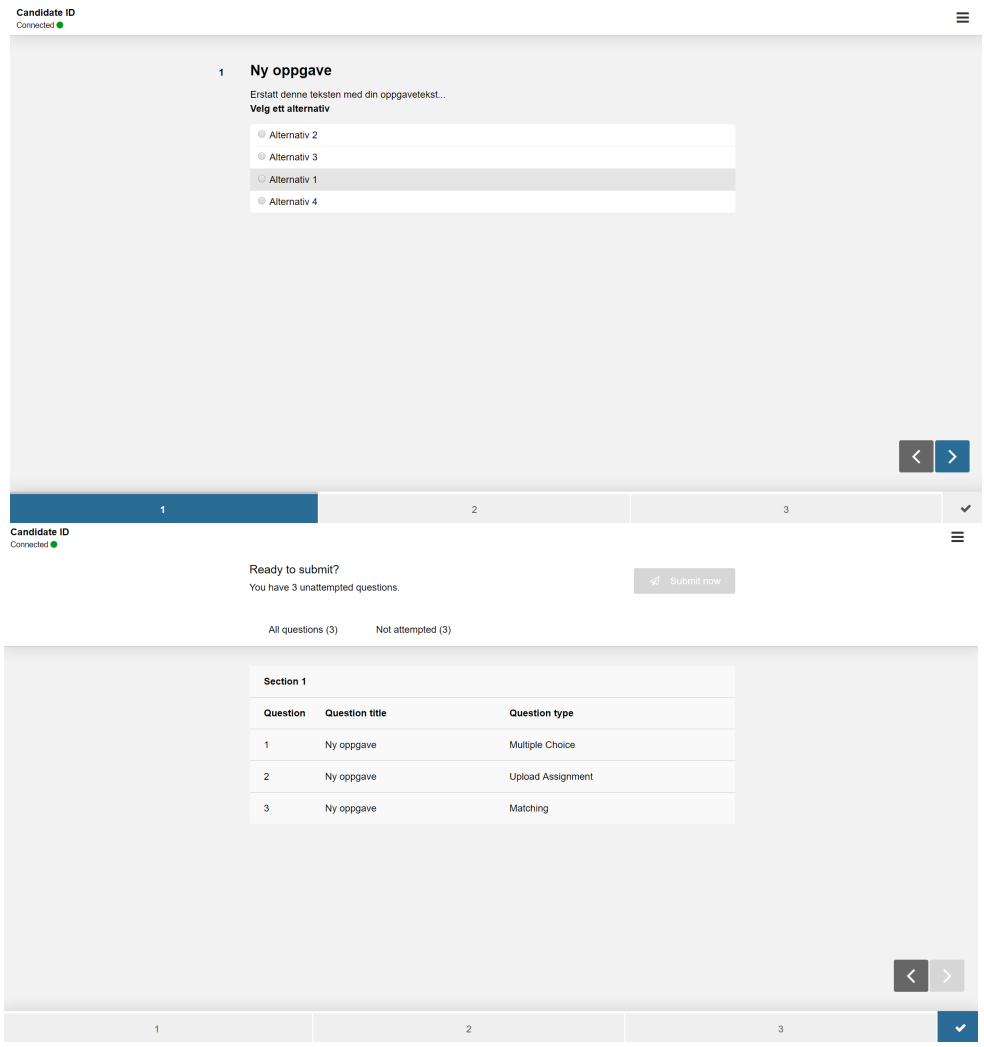

Figure 6.1. Example of a question and submission

As showed in figure 6.2, after the submission the answers were stored in the database. The storing procedure was carried by a task taking the xml representation of the items and storing it in the database. The analytics task was then fetching raw data from the database, elaborate them and store them in the item bank.

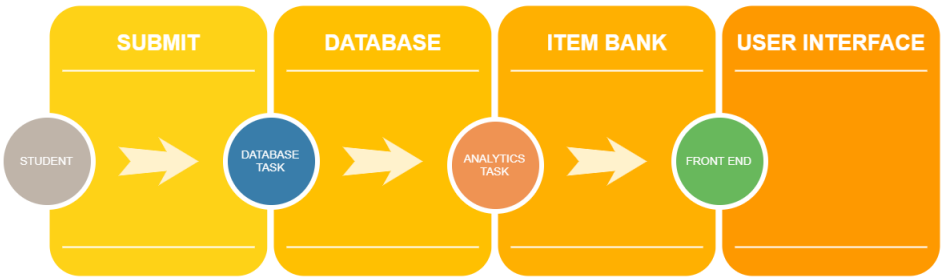

Figure 6.2. Item bank workflow

The item bank was composed of different ES indexes containing different documents with different useful information. The relational database was not the right infrastructure to use in the evaluation of the analytic parameters for the following reasons:

- something fast and able to process lots of data in a small amount of time was needed: different users will ask for different type of analysis at the same time. Being able to satisfy them all without latency was a key feature, especially when the analysis requires text search
- the ability of doing heavy calculations without overloading the servers was needed: lots of evaluations are done directly in the ES query; the possibility of delegating the calculations to the cluster instead of doing them directly on the server was one of key choices in the decision
- ability of re-indexing and changing easily the structure of the storing infrastructure was needed: while experimenting, changes in the data representation and in the type of information needed was performed quite often; using ES allowed to easily change mapping, re-index and play around with data without modifying heavily code and without loosing lots of time.

In addition it is necessary to point out that an infrastructure able to process a lot of reads was necessary. ES is a system that supports lots of read operations and for this specific part of the project something with this features was needed. Furthermore overloading the relational database was not an option: the database must remain as free as possible, and queried the less possible. In order to achieve

that a layer of cache is built around the database and ES indexes used for fast and frequent lookups.

After storing the evaluation of the analytic parameters in the item bank it was possible to see them in the user interface, populated with the same evaluations present in the cluster. An example of this view is showed in figure 6.3.

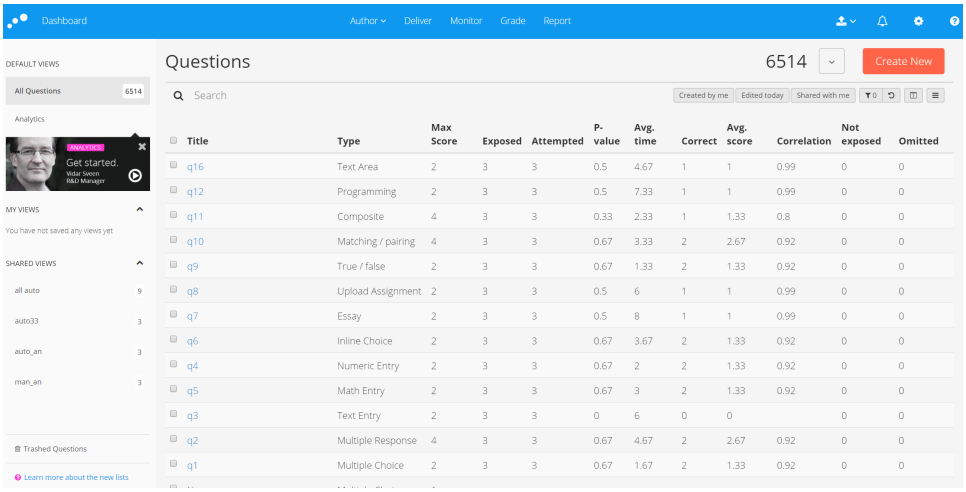

Figure 6.3. User interface with analytics

6.3 Project Structure

The role of the implemented task was to fetch data from the elasticseach index, containing the data taken from the database, and from the database itself, elaborate them and store the evaluations in the item bank. The task, from now on called *AnalyticsDerivedValuesTask*, and all the other classes used to obtain the wanted results, went through a process of refinement that lasted for different iterations. As described briefly before, finding the right structure was not easy: while the already present structure was more than enough for the initial evaluation, it became very difficult to maintain while adding the new types of statistics. The final structure is the one showed in figure 6.4. In the following section a brief explanation of the classes is done.

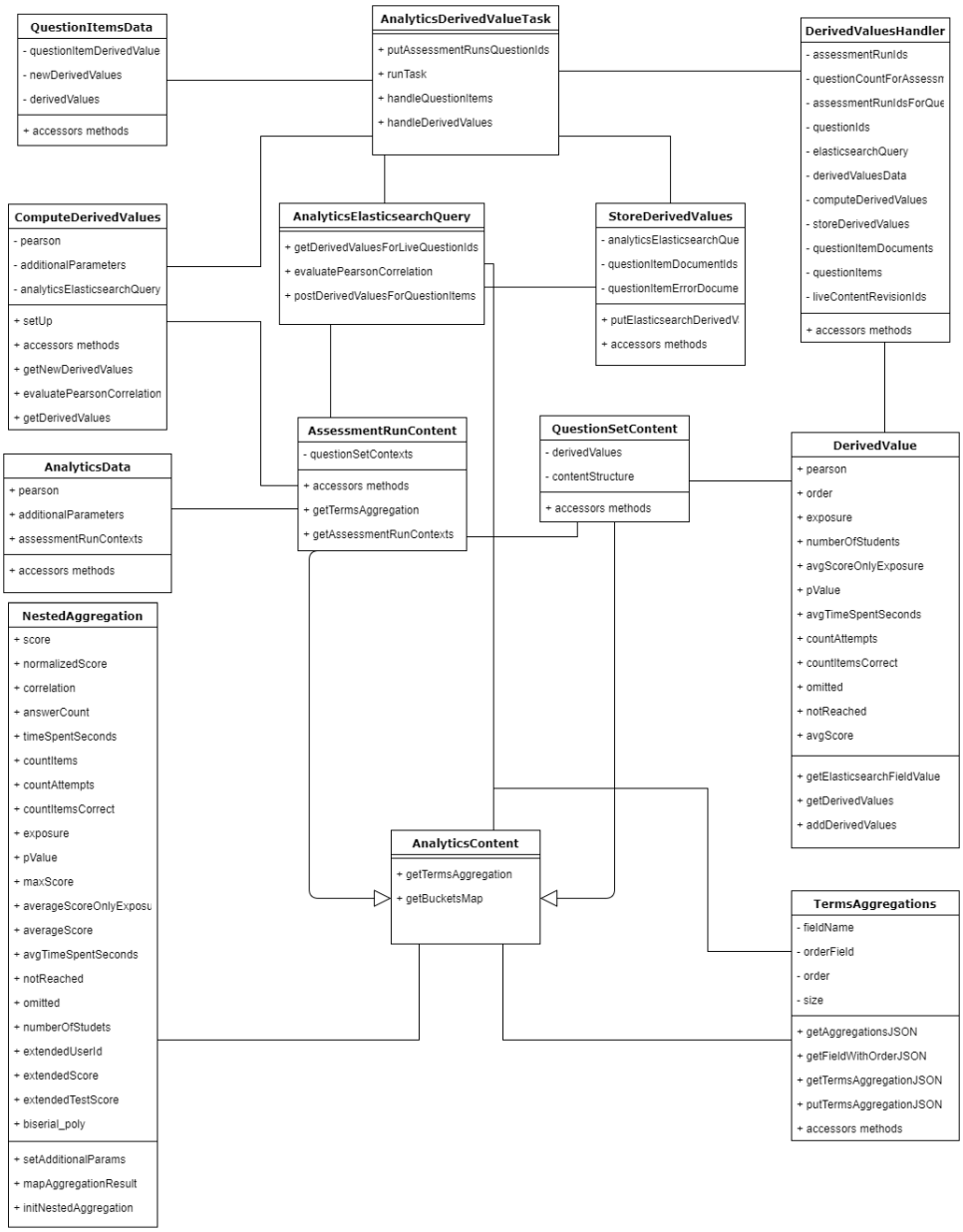

Figure 6.4. Class diagram

- **AnalyticsDerivedValuesTask**: this was the main class, the one executing the task. Its main role was setting up the environment in order to allow the evaluation of the analytic values
- **QuestionItemData**: it was used to store the new and the old analytic values;

the old values were the ones that were already calculated, while the new values were the newly calculated. New values could overwrite the old ones

- **DerivedValuesHandler**: it was mainly used as a container to pass all the different parameters from a class to another one
- **StoreDerivedValues**: it was used to store in the ES database the derived values. The storing procedure was the following: recreate (or create in case was not present) the document in the ES index, use the same *_id* (if document already present) and post the changes: if the *_id* was of a document already present in the index, the document was overridden
- **AnalyticsElasticsearchQuery**: it was used to create the ES queries; it allowed to insert filters and create the query that was needed based on the case
- **ComputeDerivedValues**: it was used to start the actual computation of the parameters. Initially a set up was performed, calculating the additional parameters that were needed (for example the ones for the evaluation of the Pearson correlation explained in 6.1.4), followed by all the other evaluations
- **AnalyticsContext**: it was an abstract class containing the basic methods for the other two *context* extending *AnalyticsContext*. In particular the method *getTermsAggregation* allowed to insert custom aggregation in the ES query
- **AssessmentRunContext**: it was used to conceptualize the context of an assessment run. Each assessment run (or test) contains one question set (as explained in 6.2); since the evaluated parameters were for each item in each test, identifying the context of a test was foundamental
- **QeustionSetContext**: this class was not useful at the beginning, since a test only contained one question set; this means that for each *AssessmentRunContext* there was only one *QuestionSetContext*. This class will be more useful in the future when more question set could be associated to one assessment run
- **AnalyticsData**: it was used as a container for the Pearson parameters and the additional parameters used in the evaluations and all the *AssessmentRunContext*s. The function of this class is similar to *DerivedValuesHandler*
- **DerivedValues**: enum class that was used as a container for all the different evaluated values. Each named value represented a type of evaluation, and defineed which method to use to obtain the value as well as its the name. Methods allowed to extract the value from an object containing different results, and if additional operations were necessary on the parameter, those were performed before returning it

- **TermsAggregation**: it was used in the creation of aggregations for the ES queries
- **NestedAggregations**:it was a enum class, like *DerivedValues*; it was used to define how the aggregations for the different parameters were composed, in particular the composition of the scripts. ES allows to use scripts inside aggregation fields: those script can use document fields and perform different actions depending on their function

6.4 Contributions

My contributions consisted in the implementation of the code evaluating CTT parameters. In the following sections an overview of the different evaluated parameters that were more difficult to obtain will be done.

6.4.1 Pearson correlation

The pearson correlation was calculated in two different steps:

- evaluation of the initial values, like (refer to the formula in 6.1.4):
  - n: number of students
  - $\overline{y}_{.j}$: mean of the total scores for item j
  - $\overline{y}_{..}$: mean of the total scores of all the students
  - $v_{.j}$: variance for item j
  - $v_{..}$: variance for all students/items
- evaluation of the pearson correlation itself

The most difficult part was to extract the information using the least interaction with the ES index possible. In the first step described above, an ES query was extracting the values. Since each value was referring to a specific item in a test, the query was structured in this way:

```
{
  "aggregations": {
    "1654044": {
      "aggregations": {
        "questionSetContext": {
          "terms": {
```

```
        "field": "questionSetContent"
      },
      "aggregations": {
        "extended": {
          "terms": {
            "field": "questionContent"
          },
          "aggregations": {
            "extendedScore": {
              "extended_stats": {
                "field": "score"
              }
            },
            "avgScoreCorrect": {
              "avg": {
                "script": "doc['score'].value ==
            doc['maxScoreForQuestion'].value
            ? doc['testScore'].value : null"
              }
            }
          }
        },
        "extendedUserId": {
          "extended_stats": {
            "field": "userId"
          }
        },
        "extendedTestScore": {
          "extended_stats": {
            "field": "testScore"
          }
        }
      }
    }
  },
  "filter": {
    "bool": {
      "must": [
        {
          "terms": {
            "assessmentRunId": [
```

```
                    1654044
                ]
              }
            }
          ]
        }
      }
    },
    "size": 0
  }
}
```

The aggregation *extended_stats* allowed to obtain stats like average, mean, variance, standard deviation, min and max. The result was a set of buckets containing assessment runs, question sets and questions. Iterating over each assessment run, question set and questions, the query evaluating the pearson correlation was created. The structure of the query was the following:

```
{
  "aggregations": {
    "2244228_2244227_2244226": {
      "aggregations": {
        "biserial_poly": {
          "sum": {
            "script": pearsonScript,
            "params": {
              "varianceOverStudentsForQuestion": 0,
              "meanScoreOverQuestionsStudents": 2,
              "meanScoreOverStudentAnsweringCorreclty": 0,
              "meanScoreOverStudentsForQuestion": 1,
              "numberOfStudents": 1
            }
          }
        },
        "maxScore": {
          "max": {
            "script": " (doc['maxScoreForQuestion'].value > 0
            ?doc['maxScoreForQuestion'].value : 0)"
          }
        }
      },
```

```
      "filter": {
        "bool": {
          "must": [
            {
              "terms": {
                "assessmentRunId": [
                  2244228
                ]
              }
            },
            {
              "terms": {
                "questionSetContentRevisionId": [
                  2244227
                ]
              }
            },
            {
              "terms": {
                "questionContentRevisionId": [
                  2244226
                ]
              }
            }
          ]
        }
      }
    }
  },
  "size": 0
}
```

Where *personScript* was:

```
"varianceOverStudentsForQuestion > 0 &&
numberOfStudents > 0 ? ((doc['score'].value -
meanScoreOverStudentsForQuestion) * (doc['testScore'].value -
meanScoreOverQuestionsStudents ))/
(varianceOverStudentsForQuestion * numberOfStudents ) : null"
```

The filters on *assessmentRunId*, *questionSetContentRevisionId* and *questionContentRevisionId* allowed to recreate the sum in equation 6.3. The name of the aggregation composed by the ids of *assessmentRunId*, *questionSetContentRevisionId* and

questionContentRevisionId, was used to have a one to one relation among assessment run and item id.

A note on the maximum score: it was extracted in the query evaluating the pearson correlation because the correlation was evaluated before the other values, and since the maximum score for the item was used in the evaluation of the *pValue*, having it in advance was significantly decreasing the complexity.

Before elaborating the final solution the following steps were done:

- creation of a query to understand if an item was dichotomous or polytomous: initially I did some trial using excel sheets on the formulas reported in 6.1.4, and the results for dichotomous items were different when using equation 6.2 or 6.3. So in order to use the right query for the right type of item, understanding the type was a fundamental step. To do that the score of each student answering the question was evaluated: if students obtained only zero or the maximum score, then the item was considered dichotomous, otherwise polytomous

- validation of the results: a new tool for the manual evaluation was discovered, and the results obtained checked against the one obtained from the query. Due to inconsistency, changes in the query were made in order to obtain the correct result: the errors that were present were due to distraction and incorrect formulation

- single query for both polytomous and dichotomous: after fixing the errors and the confirmation of the same result using equation 6.2 or 6.3, the query to obtain the type of the item was deleted

- addition of the aggregation to obtain maximum score: as reported before, the maximum score was added to obtain values before their actual use, and to simplify the flow

6.4.2 PValue and other parameters

The other parameters were not too difficult to evaluate but still some unexpected behaviours were sometimes observed.

PValue

The pValue was in some cases greater than one. The initial issue was the score of a student being greater than the maximum score for a given item: I discovered that the stored score was not correct in some cases and the reason was the evaluation of

the manual score. As described in section 6.2, a teacher can manually set the score of an answer; in addition there could be multiple teachers scoring the same student (in order to avoid partial evaluation and obtain impartiality). Errors in retrieving the scores and combining them together were found and fixed accordingly to the specifications.

Another issue was related to the score being set for students that did not answer the item: having a score different from zero when not answered mean that the average score calculated on the students exposed to the question was higher than the actual value, resulting then in a pValue greater than one. To fix this issue, an approach was to use the feedback contained in the xml representation of the student answer: if the feedback was *NOT_ANSWERED* and the score was different from zero, then the score was set back to zero.

Another issue was the value of *exposed* (see section 6.1 for definition) not being correctly evaluated.

Exposed

The number of students exposed to the question was evaluated in the following way (ES aggregation reported):

```
"exposed": {
        "sum": {
        "script": "(doc['durationSeconds'].value > 0 ? 1 : 0)"
        }
}
```

The problem was that *durationSeconds* was not updated correctly: some times (especially when the time spent on an item was less than three seconds) the value was not stored correctly. In order to fix this, changes in the code were done by other developers in charge of this part, but a error still remained on the exams evaluated before the fix. Because of this the equation $Submission = Attempts + Omitted + NotExposed$ was not satisfied. To correct this the following code was used:

```
if (exposure < countAttempts + omitted){
        exposure = countAttempts + omitted;
}
if (numberOfStudents != exposure + notReached){
        notReached = numberOfStudents - exposure;
}
```

Order

Extracting the order was not dependent on the ES query. Given the id of the item and the id of the test, the xml representation of the test was retrieved and then the order for the specific item was obtained.

The most difficult parts in this process have been:

- the study of the infrastructure: the code was initially written by another employee, and learning the structure at the beginning was not easy
- understanding and applying the theory: understanding it and applying the theory in the correct way was not an easy task; initially the evaluations were all wrong because the granularity of the aggregations was completely wrong. Deciding what to evaluate, evaluating it in the correct way and checking the result without committing even more errors was not straight forward and took more than one iteration to get it done properly
- design of the item bank: defining a final structure for the item bank was not a straightforward part too; it was not easy to create a final structure at the first try, also because of the always changing requirements
- restructuring the entire project: the initial structure of the classes and the interaction among them was not optimal, and towards the end a restructuring of the project was necessary; thinking of a new structure was not an easy task, but also with the help of other colleagues we figure out the best way to conceptualize all the different ideas

6.5 GUI

The graphical interface had the purpose of showing the new analytic evaluations in a user friendly way. Since the initial idea was to show for each item the tests and the question sets it was used in. The problem with this set up was filtering and showing values in a specific context: selecting a specific assessment run resulted in hiding the other assessment runs, preventing the user from selecting them in a second moment. To obtain a more correct and usable design, we decided to show only the context of a test, and not assessment run and question set: a test was an assessment run, but since only one question set for now was associated to the assessment run, filtering on question set or assessment run was the same thing.

Another improvement was the collapsible section to show values when the question was associated to multiple tests. In addition it was possible to filter on them, order them and decide which one to show and in which sequence.

6.6 CTT for Question Sets

CTT can be used also within the context of a question set. Alongside this new analysis, different evaluation reported below were done:

- **number of questions**: the number of items used in the question set
- **number of submissions**: the number of submissions for a specific test using the question set
- **average score**: the average score of the students taking a specific test where the question set was used
- **average score normalized**: as the average score but normalized
- **score first quartile**: given the population of the students, the maximum score of the first 25 percent of the population
- **score third quartile**: given the population of the students, the maximum score of the first 75 percent of the population
- **score median**: given the population of the students, the maximum score of the first 50 percent of the population
- **max score reachable**: the maximum score for the question set
- **max score achieved**: the maximum score achieved in the context of a test where the question set was used
- **min score achieved**: the minimum score achieved in the context of a test where the question set was used
- **standard deviation of the score**: the standard deviation of the scores achieved by the students in the context of a specific test using the question set
- **average time spent on the test**: the average time spent on the test by the students

- **percentage of not exposed**: evaluated as the sum of not exposed on each items compared to the total number of items in the question set multiplied by the number of submissions
- **Cronback's alpha**: it is a coefficient that is used to estimate a measure of the reliability of a test

The principle followed for these calculations was the same used for the items' parameters: an ES query with different aggregations that allowed to obtain the wanted results. The query contained a filter for each question set, and for each of them aggregations on the assessment runs were done. In this way, given the context of a question set used in different assessment runs, for each assessment run the analytic values were calculated.

Second & third quartile, median

Two different approaches were used. The first approach consisted in the retrieval of all the student answers, organize them properly and extract the maximum score for the 25 percent of the population, 50 percent and 75 percent. This approach was easy and performing well, but in case of great number of students, fetching all the results was resulting in too much network traffic given the task performed.

The second and actual solution was to use the possibility given by ES to select the subset of results to return: after ordering the user answers by achieved test score, using *from* and *size* allowed to select just one answer. The *from* field was set using $number\_of\_students * number\_of\_questions * ratio$ where ratio was 0.25, 0.50 or 0.75, while *size* was set to 1. The necessity of multiplying the *number_of_students* by the *number_of_questions* derived from the fact that each students appeared *number_of_questions* times in the ES index (remember that the index contained for each user all the answers to the submitted tests).

Cronback's alpha

The Cronback's alpha is a coefficient used to estimate a measure of the reliability of a test. It was evaluated in the following way

$$\frac{K}{K-1}(1 - \frac{\sum_{i=1}^{K} \sigma_{Y_i}^2}{\sigma_X^2}) \tag{6.4}$$

where K is the total number of items, $\sigma_{Y_i}^2$ is the variance for item Y_i and σ_X^2 is the variance on the total score. In order to obtain the arguments of equation 6.4 a query with aggregations on items and users was done. Using then *extended_stats*, in both contexts, $\sigma_{Y_i}^2$ was obtained from item's context while σ_X^2 from student's context.

Chapter 7

IRT in the platform

After having the CTT evaluaiton in place it was time to move the IRT evaluations in the platform. As described in section 5.3, a solution to evaluate IRT parameters was developed: in the following sections the approach followed to implement the code in the platform will be explained. In figure 7.1 the followed workflow is reported.

7.1 Question retrieval

To retrieve the single instances of each item a query fetching all the answers was performed. The ES query was filtering for each question set and retrieving all the answer for each question. The single instances of the item were obtained exploiting lambda expressions to remap the result obtained from ES. The main problem was to return large amount of documents representing the answers: ES returns at most 10000 documents. The reason is that pagination (Elasticsearch Reference - Pagination 2018) is expensive, especially because of the ordering of the documents. In order to avoid this problem and retrieve great amount of documents, *scroll* (Elasticsearch Reference - Scroll 2018) was implemented. The advantage of using scroll is that there is no ordering of the documents, and this allows to retrieve greater amount of documents without overloading the cluster with work.

Once the question definition was finished, the identification of the question type was done: based on the number of different score options, each question was associated with either polytomous or dichotomous type.

7.2 Answer manipulation

For each question the set of answers was evaluated: for each student the chosen answer (in case of dichotomous item) or the achieved score (in case of polytomous items) was set. A pruning phase was then performed, as described in section 3.2.1.

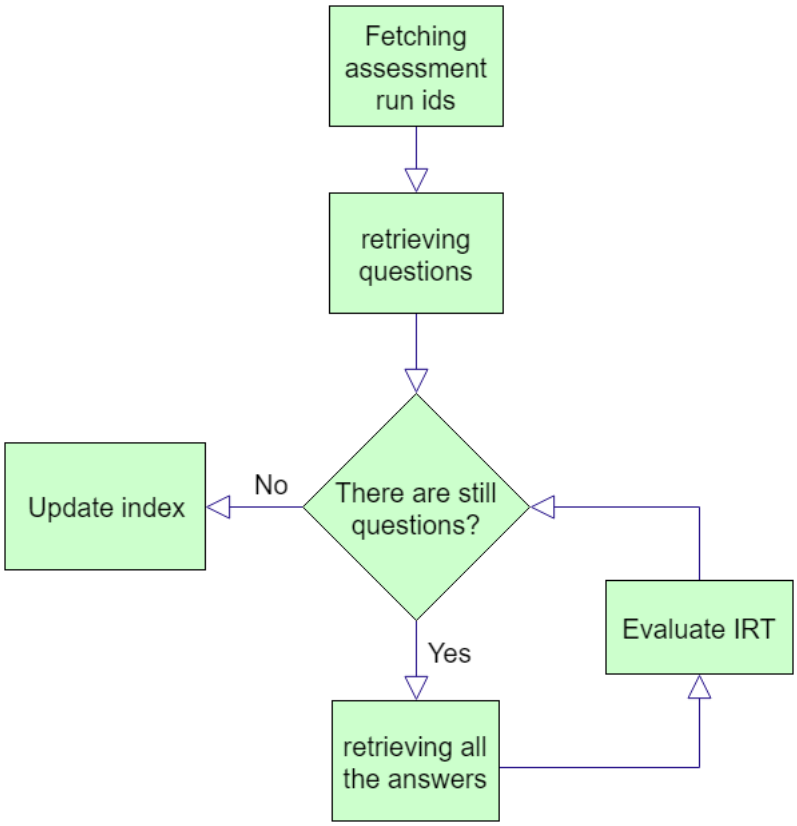

Figure 7.1. IRT workflow

The pruning phase was obtained using two lambda functions associated to two stream operations. Once the pruning phase was completed, more question answers were analyzed until no more items were present.

7.3 IRT evaluation

In order to run the IRT evaluations on the cleaned and prepared data, the command in R language were used. In the following section different approaches to achieve the result will be presented. The result of the IRT evaluations was a set of discrimination and difficulties for each items and theta values for each student.

7.4 R in Java

The Item Response Theory (IRT) application written in R should have been introduced in the new R code in charge of calculating the values to be inserted in the item bank. Different solution are present to use R code inside a Java project. In the following sections an overview of the different methodologies will be proposed.

7.4.1 RJava

"*RJava is a simple R-to-Java interface. It is comparable to the .C/.Call C interface. RJava provides a low-level bridge between R and Java (via JNI). It allows to create objects, call methods and access fields of Java objects from R*" (RJava Documentation 2016). RJava can be used as an API, there is no server-like interaction. The R command were stored as strings and passed to the *eval* method of the rJava package.

The possibility of using such a library like rJava, with the opportunity to run R commands was not possible because of stack overflow errors occouring when loading libraries from R. In order to use *mirt*, other libraries like *lattice* and *stats4* were necessary. When loading *stats4* due to stack overflow errors the program was crashing.

Given the failure in loading fundamental libraries, it was impossible to use rJava.

7.4.2 Rserve

"*Rserve is a TCP/IP server which allows other programs to use facilities of R from various languages without the need to initialize R or link against R library*" (Rserve Documentation 2015). There are different advantages in using Rserve:

- fast: there is no need to initialize any instance of R form the java code
- client independence: the clients are not linked to R. For this reason whatever client can connect to the server and use R independently
- security: encrypted user password is supported by Rserve
- file transfer: it is possible to transfer files from client to server and back. This is very useful when images and plot from R should be visualized in the client
- persistent: each connection has its own work-space and its working directory. The initialized objects are not destroyed until the session is active

This system of using R from Java was pretty convenient. The main problem was the TCP/IP connection on the server: our servers have a limited number of connection that can be established, and creating a connection dedicated only to R was not possible. For this reason Rserve was not possible to be used.

7.4.3 Renjin

"*Renjin is an interpreter for the R programming language for statistical computing written in Java much like JRuby and Jython are for the Ruby and Python programming languages*" (Renjin Documentation 2018).

Renjin was chose as a solution for the following reasons:

- objects sharing: using Rserve it is possible to pass object instances from R to Java and from Java to R. Evaluating IRT parameters had as result the usage of S4 object oriented system in R. The possibility to access S4 from Java was a great advantage

- error handling: errors generated in R could be caught in Java. This feature was really important, especially when it came to *log* errors and misbehaviour

After starting with the implementation though, major problem raised. Renjin has a lot of dependencies: it is necessary to install the R packages that are to be used, and also to install the Java packages necessary for Renjin to run. Managing Java dependencies was an issue: in Inspera there is no package management software and trying to match all the dependencies of the platform with the ones of Renjin was too much to be handled manually. We did not want to have any duplicate package (even worse if with different versions), and setting up a software solution for package management was not an option due to time issues.

7.4.4 FastR

FastR is an implementation of the R Language in Java atop Truffle, a framework for building self-optimizing AST interpreters (FastR Documentation 2017). FastR is a sultion build from Oracle, but it was not possible to test it since there is no implementation for Windows yet.

7.4.5 Plumber

An R package that converts your existing R code to a web API using a handful of special one-line comments (Plumber Documentation 2017). Plumber is very easy to use, and pretty straightforward. This was selected as final solution to use R in Java: no actual R code was called, only a connection through a specific port in *localhost* and it was possible to call the same functions that were used in the very first implementation of Merlin. Moreover, the messages retrieved via API allowed to catch also eventual errors, and perform some error handling. Error handling was not as precise as in Renjin, but good enough to achieve the wanted results.

Some changes were made though: the preparation of the data was done using Java code, and not in R as in the first version of Merlin. The pruning done in the python code, was now done in the Java implementation.

In addition a script in R starting end ending the service was written.

Chapter 8

IRT sample independent

It is important to distinguish among *population* and *sample* independence. Population independence means that changing population of students, the estimation of the IRT parameters does not differ based on the different populations. This is not achievable in IRT: being model based ties IRT to fit a specific model represented by the population it is evaluated against. If the population changes, also the parameters evaluated do. Sample independence means that the estimation of the parameters do not vary when oversampling from the population the model is fitting on.

Independence in item calibration

In item calibration sample independence acquires different meanings depending on the situation analyzed. Supposing that the item is already calibrated, that means it is fitting the population and the item parameters are estimated. The trait level estimate of a given student s belonging to a subsample S will not depend on the distribution of S. The immediate consequence is that, regardless of the sample distribution (majority of good or bad students), the ability estimates are not going to be biased. If the item is not calibrated the usual scenario is the following: subsample *S1* is used to evaluate item parameters, than another subsample *S2* that contains also *S1* is used again for the evaluation. In this case the estimations are not sample independent: we need enough samples to approximate the population and achieve model fit. Afterwards sample independence for students estimates can be applied again.

In order to understand when an item is calibrated, it is enough to define an error among the different estimation of the parameters: when the drift is small enough it means that the item is calibrated on the population.

This is not different from CTT: when having parameters estimated from the entire population, than the level of the student evaluated as a function of the difficulty of the items is not dependent of the subsample the pupil belongs to.

Independence in item comparison

Lets analyze the following scenario: subset *S1* is associated with test *T1*, while subset *S2* is associated to test *T2*. *S1* and *S2* (which are disjoint) are used to evaluate item parameters. The parameters are then used in the comparison against other tests. In order to run the comparisons it is necessary to have the parameters a common scale, and this implies rescaling. Supposing that *T1* and *T2* are the same test, how the comparison can be done in the context of IRT and CTT?

How the definition of sample independence can be reformulated in order to provide a way to compare those parameters using different technologies?

In the following sections an approach and a solution to the problem is reported.

8.1 Population generation

The first step was to simulate response patterns for each student belonging to the population. A response pattern was defined as the set of answers for a specific set of questions: the synthetic test was composed of 20 polytomous items with five different achievable scores.

The response patterns were generated using the function *simdata* provided by the *mirt* library in R. The total number of students was 5000.

8.2 Subsets generation

The subsets generation was performed in two different ways. The first approach was to divide the entire population based on the 50 percent quantile of the ability. The second approach was to use random sapling without replacement from the initial population.

The random sampling procedure was done in the following way:

- evaluation of the total score: for each student the related total score was calculated based on the pattern
- mean definition: for each subset, the mean score was defined. Two groups (*high* and *low* scores) were created and the mean was specified as the distance from the population's mean score equal to two times the standard deviation of the population's score itself
- probability definition: after defining the mean, a probability for each student was defined. The closer the student score to the mean, the higher the probability of selection. The function to evaluate the probability was based on the following reasoning: given a mean of three, all the students with score three

must have the highest probability; students with score close to three (like four, five, two or one) should have a lower probability and the higher the difference between mean and score, the lowest the probability of being selected. To achieve the result the simple code was used:

```
probVector <- totalScore
probVector <- -abs(meanValue - probVector)
#add 1 because we don't want zero prob
correctProbVector <-
        probVector +
        abs(min(probVector)) + 1
if (probType == "exp"){
        correctProbVector <-
                floor(exp(correctProbVector))
}
if (probType == "square"){
        correctProbVector <-
                correctProbVector^2
}
if (probType == "plain"){
        correctProbVector <-
                correctProbVector
}
```

The probability type was used to emphasize even more the difference among the different probability values.

- sampling: based on the probability of being selected, 1000 students were sampled from the total population. After clustering them based on the total score, the size of each group was evaluated. An example is showed in table 8.1

- pattern selection: for each set the maximum number of possible patterns was retrieved. After evaluating the different number of patterns for each score, using a simple division was easy to know how many student for that specific pattern to retrieve. An example is reported in table 8.2

- student selection: based on the results obtained from the previous step, the number of students for each pattern was selected and the subset of the population created

Table 8.1. Number of students for each random group

Score	Number Of Students
0	120
1	81
2	88
3	166
4	102
5	101
6	106
...	...
13	2

Table 8.2. Number of students for each pattern

Score	Number Of Students	Number of unique patterns	Student for pattern
0	120	1	120
1	81	4	17
2	88	12	7
3	166	20	8
4	102	25	4
5	101	33	3
6	106	35	3
...	...	...	...
13	2	1	2

8.3 Parameters evaluation

In figure 8.1 the evaluations of the five thresholds for the different items are reported. The results refers to the subsets created using the first method. With the second methods the similar trends were obtained.

It is possible to notice the difference among the evaluations done with CTT and the ones done with IRT: in figure 8.2 the parameter evaluations done with the two subsamples have linear dependence and a correlation close to one for all the thresholds (0.958 for b1, 0.971 for b2, 0.967 for b3 and 0.956 for b4). In figure 8.3 it is possible to notice how there is no linear dependence and correlation is very low in comparison to IRT (0.764).

8.4 Conclusions

It is possible to reformulate the concept of sample independence as follows:

"*IRT is sample independent because, even though the parameters differ when evaluated using different student distributions, they have linear dependence. It is possible to rescale them and compare them regardless of the distribution they are evaluated from. In CTT this is not possible because there is no linear dependence, and the rescaling would end up being messy if not impossible.*"

This property is very important when comparing tests with difficulties obtained from samples not representative of the entire population, because it allows to obtain significant results using IRT.

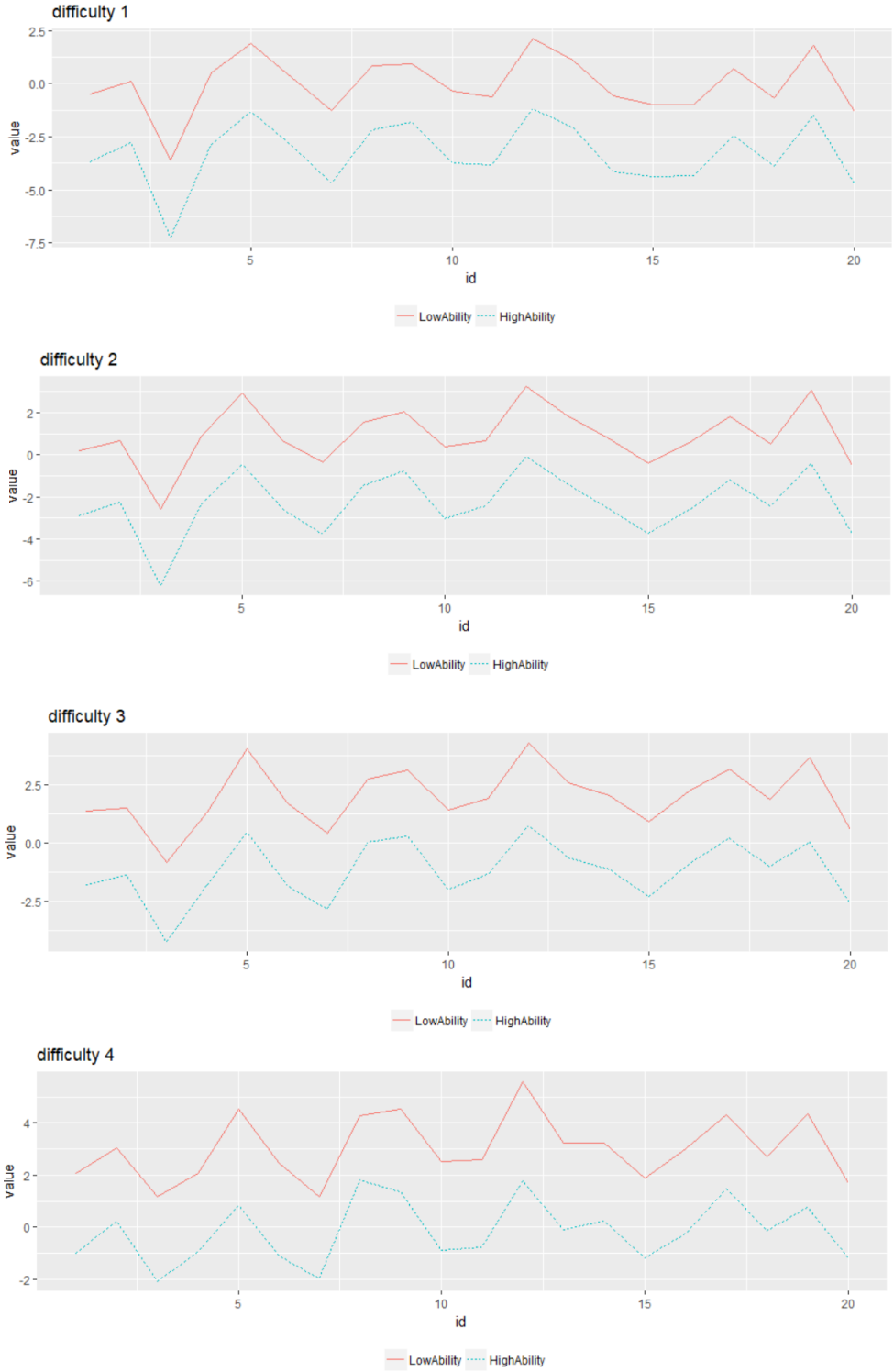

Figure 8.1. IRT grade comparison

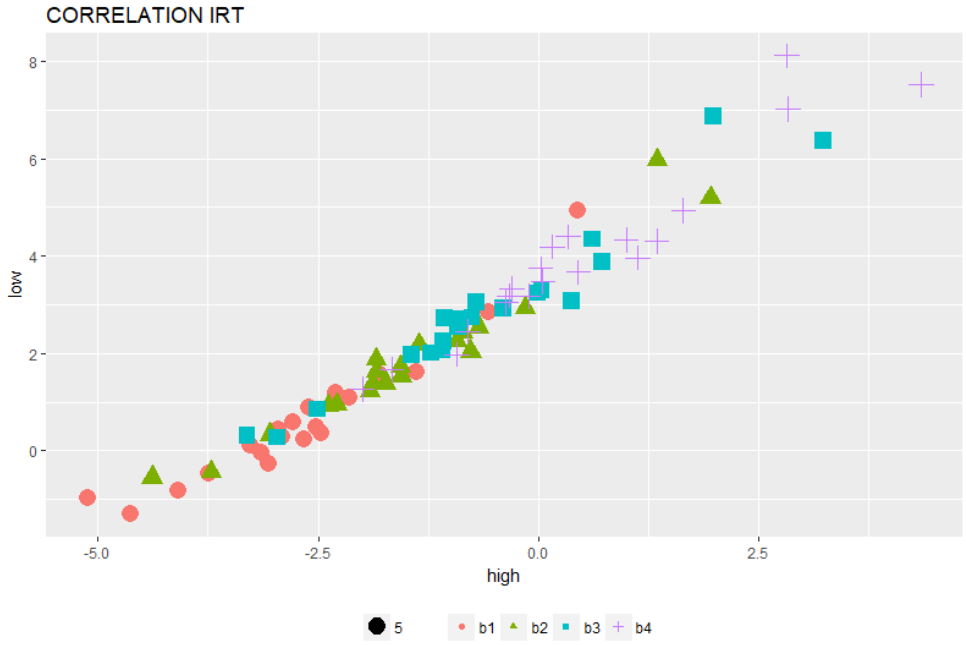

Figure 8.2. IRT correlation

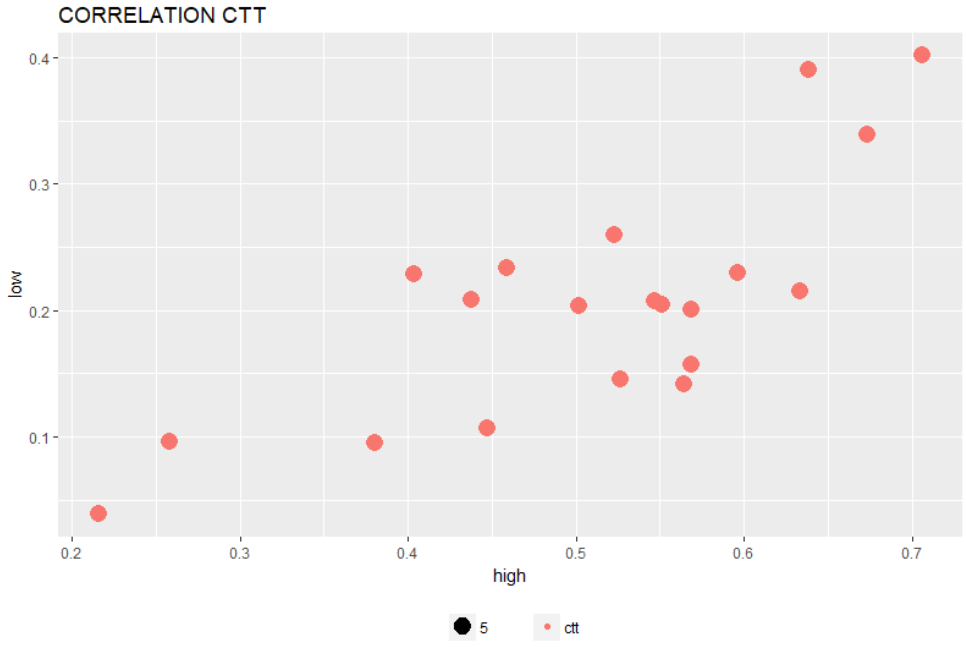

Figure 8.3. CTT correlation

Chapter 9

Summary and Further work

This project evaluated the possibility for a solution using IRT for the creation of an item bank containing parameters regarding difficulty and discrimination of questions as well as ability level of pupils. Different solutions have been evaluated and a final implementation has been proposed.

The calibrated items as well the item bank will be used in the future as a basis for further work. Me and my team will investigate more on the model fitting, how to assess unidimensionality and better modeling solutions. Furthermore, work related to item quality will be done: a platform to help teachers writing more effective and better questions will be developed. This step will require understanding of how good questions are defined, similarity among questions and how to highlight them, how to give feedback to teachers and how to help them in the construction of tests with specific characteristics tailored for different situations.

Chapter 10

Appendix

Elasticsearch resiliency

Elasticsearch resiliency status

https://www.elastic.co/guide/en/elasticsearch/resiliency/current/index.html

JMetrik

JMetrik website

https://itemanalysis.com/

Xcalibre

Xcalibre website

http://www.assess.com/xcalibre/

Elasticsearch index

- userId : Id of the student
- tenanceId : Id of the marketplace
- sex : it is the gender of the student, could be M or F (male or female), extracted from the object table
- containing the logical name: if even male, odd female
- questionContentItemId : Id of the question

- questionContentRevisionId : Id of the revision of the question
- score : the score of the student for the question (equal to automaticScore if isAutomaticallyCalculated is true)
- isAutomaticallyCalculated : false if the question is manually scored
- automaticScore : zero if isAutomaticallyScored is false
- maxScoreForQuestion : the maximum score of the question
- testScore : the score of the student for the assessmentRun
- maxTestScore : the maximum score for the assessmentRun (sum of maxScore-ForQuestion for scorable questions)
- isAutomaticallyScored : true for question that are automatically evaluated
- maxAutomaticTestScore : the maximum score for the assessmentRun (sum of maxScoreForQuestion for automatically scorable questions)
- questionSetContentItemId : Id of the assessment
- questionSetContentRevisionId : Id of the revision of the assessment
- anchors: it is possible to uniquely identify an assessment that contains an anchor to the current one. The unique identification is done using the assessmentRevisionId and assessmentId and the questionId and questionRevisionId for the question. It is possible that a question was used as an anchor in different assessments.
- class: is the class assessment was use for
- questionSectionId : index of the assessment section the question belongs to
- questionIndex : position of the question in the assessment
- sessionId : Id of the assessmentRun
- startTime: the start for the assessmentRun (can be missing)
- endTime : the time when the answer was submitted (can be missing)
- category: the category of the assessmentRun
- durationSeconds : the seconds the student spent to answer the question (0.0 by default)

- numberOfAttempts : times the answer to the question was attempted by the student (0 by default)
- responseVariables : metadata that, depending on the question type, maximally includes the answer, both correct and wrong, to each question element
- responseVariable : metadata that, depending on the question type, is included in either the correct or wrong answer to a question element
- for Multiple Choice question type:
  - interactionId : unique identifer of a question element
  - responseId : unique identifer of an alternative to the question element with interactionId
  - order : sequence number (counting from 1) of a responseId in the question element with interactionId
  - correctResponseVariables: metadata that, depending on the question type, minimally includes the correct answer to each question element correctResponseVariable : metadata that, depending on the question type, is included in the correct answer to a question element
- for Multiple Choice question type:
  - interactionId : unique identifer of a question element
  - responseId : unique identifer of a correct alternative to the question element with interactionId
  - order : the sequence number of a correct alternative responseId to the question element with interactionId

Bibliography

Birnbaum (1968). "Some latent trait models and their use in inferring an examinee's ability". In: *FM Lord, MR Novick (eds.), Statistical Theories of Mental Test Scores*.

Bock, R. Darrell and Murray Aitkin (Dec. 1981). "Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm". In: *Psychometrika* 46.4, pp. 443–459.

Chalmers, R Philip (May 2012). "mirt: A multidimensional item response theory package for the R environment". In: *Journal of Statistical Software* 48.6, pp. 1–29.

Elasticsearch Reference - Node (2018). URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html#modules-node.

Elasticsearch Reference - Pagination (2018). URL: https://www.elastic.co/guide/en/elasticsearch/guide/current/pagination.html.

Elasticsearch Reference - Scroll (2018). URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-scroll.html.

FastR Documentation (2017). URL: https://github.com/oracle/fastr.

Guilford, J. P. (1954). "Psychometric methods, 2nd edition". In: *New York: McGraw-Hill*.

Guyer R., Thompson (2014). "User's Manual for Xcalibre item response theory calibration software, version 4.2.2 and later". In: *Woodbury MN: Assessment Systems Corporation*.

Plumber Documentation (2017). URL: https://www.rplumber.io/.

Renjin Documentation (2018). URL: http://docs.renjin.org/en/latest/.

RJava Documentation (2016). URL: http://www.rforge.net/rJava.

Rserve Documentation (2015). URL: https://www.rforge.net/Rserve/.

Samejima, F. (1969). "Estimation of Latent Ability Using a Response Pattern of Graded Scores". In: *Psychometric Monograph No. 17*.

Susan E. Embretson, Steven P. Reise (2009). *Item Response Theory for Psychologists*. Psychology Press.

Glossary

anchor A question that is used in different assessment and can link them. 5, 17, 21, 34

assessment An exam. 1, 5, 11–15, 17–19, 36, 47

dichotomous Item with only two possible outcomes: right or wrong. 4, 15, 16, 34, 49, 50, 61, 67

distractor An option given an Item. 12–14, 16, 22, 23, 34, 38, 39, 41

interaction A specific question inside an Item. It is possible to have different interactions in an item. 13, 14, 87

interactive assessment It is a test given to a set of students. 87

item A question given an Interactive assessment. 3–5, 7, 11–18, 21–25, 30, 32, 34–36, 38–40, 47–51, 53, 56, 57, 61–65, 67, 68, 74, 87

polytomous Item with more than two possible outcomes. 4, 15, 16, 34, 50, 61, 67, 74

trait The specific ability that an Interactive Assessment (IA) is designed to measure. 30, 31, 34