Alma Mater Studiorum

UniversitÃ di Bologna

dottorato di ricerca in
scienze statistiche

ciclo xxxii

Settore concorsuale: 13/D1
Settore scientifico disciplinare: SECS-S/01

Dealing with uncertainty in test assembly

Presentata da:
Giada Spaccapanico Proietti

Coordinatore:
Prof. Alessandra Luati

Supervisore:
Prof. Stefania Mignani
Co-supervisori:
Prof. Mariagiulia Matteucci
Prof. Bernard P. Veldkamp

15/10/2019

Preface

The recent development of computer technologies enabled test institutes to improve their process of item selection for test administration by means of automated test assembly (ATA). A general framework for ATA consists in adopting mixed-integer programming models which are commonly intended to be handled by commercial solvers. Those softwares, notwithstanding their success in handling most of the basic ATA problems, are not always able to find solutions for highly constrained and large-sized instances. Moreover, all parameters are assumed to be fixed and known, an hypothesis that is not true for estimates of item response theory (IRT) parameters which are key elements in test assembly. These restrictions motivated us to find an alternative way to specify and solve ATA models. First, we suggest an application of the chance-constrained (CC) approach (**Charnes1959**) to ATA problems which allows to maximize the α-quantile (usually smaller than 0.05) of the sampling distribution function of the test information function (TIF) obtained by bootstrapping the calibration process. Secondly, for solving the ATA models, CC or not, we adapted a stochastic meta-heuristic called simulated annealing (SA) proposed by **goffe1996simann**. This technique can handle large-scale models and non-linear functions. A Lagrangian relaxation formulation helps to find the most feasible/optimal solution and, thanks to the SA, more than one neighbourhood of the space is explored avoiding to being trapped in a local optimum. Several simulations on ATA problems are performed and the solutions are compared to CPLEX 12.8.0 Optimizer, a benchmark solver in the linear programming field. The algorithms are coded in the open-source framework Julia. A package written in Julia is released for solving the assembly problems described in this dissertation.

Acknowledgements

I would like to express my deep gratitude to Professor Stefania Mignani and Professor Mariagiulia Matteucci, my research supervisor and co-supervisor and most of all, mentors, for their patient guidance, enthusiastic encouragement and positive critiques of this research work. The occasion to participate, in the past three years, at the INVALSI meetings for the test assembly of Italian standardized assessment project and at several international conferences of testing commissions has really increased my knowledge about practical issues and opportunities in the field of psychometric research. This would not be possible if they hadn't integrated me in their team.

Grateful thanks go to Professor Bernard Veldkamp, external co-supervisor of this thesis, for his high experienced advice and assistance in my short PhD visiting at Twente Universiteit in October 2018 which has greatly shed light on all my dilemmas. Acknowledgements are also extended to Dr. Angela Verschoor, from CITO (NL), for her priceless suggestions for the development of the optimization algorithms and for spending time with me talking and discussing about geeky and technical topics in the amazing Roman autumnal atmosphere.

Finally, I wish to thank my collegues and all the staff at the department of statistical sciences of the University of Bologna for their always friendly and welcoming hospitality.

This thesis is dedicated to my beloved Alex, to my always supportive and loyal friends and to my parents who made this adventure begin.

Contents

List of Figures

List of Tables

1 Introduction

In the field of educational measurement, a test is a collection of items developed to measure students' abilities. In order to make different measurements comparable, tests should be designed and developed providing evidence of fairness, reliability and validity (**AERA2014**). These concepts have evolved together with the development of new test theories but they all agree that testing environment and procedures must be controlled in such a way differences among testing conditions do not influence the scores. In this framework, the test assembly process plays a fondamental role because, by selecting items from an item pool to build test forms conform to content and psychometric specifications, it allows to control the entire protocol of the test production, from the item construction, because the features of the pool depend on the requirements on the final tests, to the final test building: the core of test assembly.

Large testing programs, have better access to resources like sophisticated item banking systems, opening the possibility to improve their test assembly process by means of automated test assembly (ATA). ATA has several advantages over manual test assembly. First of all, the test specifications should be defined rigorously, early on reducing the need to repeat some phases of the test development. More importantly, ATA is the only way to find optimal or near-optimal combinations of items starting from large item banks, for which manual assembly is not feasible due to the large number of possible solutions. As a consequence, ATA is fundamental to make measurements comparable while reducing operational costs.

In practice the ATA models are not always easy to solve because they involve a very large number of decision variables and constraints. Moreover, with the standard form suggested in **VanDerLinden2005** it's not possible to define any part of the model in a non-linear manner unless one resorts to approximations.

Another limit of classic optimization models, until now applied for ATA, is that they consider each variable as fixed or known. This may be valid for prices, quantities but not for estimates, key elements in ATA models. Item response theory (IRT) item parameters and subsequently the item information function (IIF) are an example.

1.1 Main contributions and outline of the dissertation

The restrictions mentioned in the last paragraph motivated us to find an alternative way to specify and solve ATA models. For the first point we suggested a test assembly model based on the *chance-constrained* (CC) approach (**charnes1958cost**) which allows to maximize the α-quantile of the sampling distribution of the test information function (TIF). The proposed model is an extension of the classical MAXIMIN ATA model (**VanDerLinden2005**, p.69-70) in which the minimum TIF among all the tests is maximized. The sampling distribution of the TIF is obtained by boostrapping the calibration process (**efron1993**; **shao2012**). In this way we ensure that, independently on the situation in which the calibration has been made, we have an high probability to have a certain error (possibly low) in the ability estimation. For solving the ATA models, chance-constrained or not, we developed an algorithm based on a stochastic heuristic called *simulated annealing* (SA) proposed by **goffe1996simann**. This technique can handle large-scale models and non-linear functions. Thanks to the Lagrange relaxation and to a random variable that accepts or rejects an inferior solution, the interesting properties of our algorithm are essentially two: it can always find the most feasible set of solutions in a limited time interval and it tries to avoid local optima seeking for the globally optimal set of test forms.

The proposed chance-constrained model, as already mentioned, is based on the empirical distribution of the TIFs of the assembled tests. This element is constructed starting from the calibration process, that is the procedure in which the IRT item parameters, such as discrimination and easiness, are estimated. More specifically, a bootstrap is performed resampling the response data and generating a sample of estimates for each item parameter. At the end of this phase, the item information function (IIF) for all the items in the pool, at a predefined ability

point, is computed using the bootstrap samples. These quantities are then used in the chance-constrained model to compute the α-quantiles of the TIFs and the model is optimized by looking for the best combination of items which produces the tests with the highest quantiles. The use of the bootstrap in the calibration is not only propaedeutic for our test assembly model but it is also a method to retrieve additional information on the variability of the item parameters without affecting their native underlying interdependence[1].

The algorithms described in this work are coded in Julia (**bezanson2017julia**), this choice has been made to achieve the best performance in numerical analysis and for the availability of a lot of valuable and customizable packages for optimization. Two packages have been developed and are available upon request[2]. The first `IRTCalibration` allows to calibrate an item pool following a unidimensional latent regression model with the one-parameter logistic or two-parameter logistic models, respectively 1PL or 2PL estimated on dichotomous response data. `IRTCalibration` uses the techniques described in Chapter 3. Futhermore, it provides the bootstrap tool to derive the sampling distribution functions of the item parameters. The second package, `ATA.jl`, implements the algorithms explained in Chapter 4 to assemble tests optimizing the classical MAXIMIN model (2.14) and the proposed MAXIMIN chance-constrained model (4.7).

Since, nowadays, several test assembly models are either based on classical test theory (CTT) or IRT, the dissertation starts, in Chapter 2.1, with a brief excursus on the available test theories and continues with a introduction to the automated test assembly models. The latter topic is enriched describing the model relaxation using the Lagrange multipliers. It follows, in Chapter 3, a software benchmarking, in which `Julia` is proposed as a programming language for calibrating of IRT item parameters following the 1PL and 2PL unidimensional latent variable models. In this setting the empirical histogram method by **woods2007** is revised and a cubic-spline method is proposed to interpolate and extrapolate the observed distribution of the ability at predefined knots. A simulation study to compare estimation accuracy and computational performance of our algorithm

[1] We do not sample independently from empirical distribution of each item parameter, instead, we resample the data following the bootstrap principle and we let the calibration algorithm produce directly the TIFs samples.This procedure should preserve the natural relationships between the item parameters.

[2] `https://github.com/giadasp/IRTCalibration.jl`, `https://github.com/giadasp/ATA.jl`

with respect to the `R` package `mirt` has been conducted. Moreover, both a parametric and a non-parametric bootstrap on the calibration is provided within the software and taking the *standard setup* defined in 3.2.1 we illustrate the results of applying this algorithms on the item parameter calibration by means of a simulation. In Chapter 4, the CC test assembly model is defined and a specific empirical version is proposed which allows to assemble parallel test forms in terms of percentiles of the TIFs. To solve the model the SA meta-heuristic is used. At last, the performance of the heuristic is tested in optimizing both the CC and MAXIMIN models on a simulated item pool always following the already mentioned *standard setup* 3.2.1. Conclusions and further research ideas are provided in Chapter 5. Supplementary tables and figure, which cannot be displayed floating in the body of text are reported at the end of the essay, in Appendix **??**.

Contents

2 Test theories and automated test assembly models

Educational assessment addresses the problem of measuring the ability of students by modelling their responses to a test under a statistical perspective. Such test is a set of items selected from an item bank. A key aspect is that, as with any other measurement instrument, different measurements of the same ability should be comparable. In order to meet this requirement, tests should be standardized. A test is considered to be standardized when the test procedures are fixed in such a way that differences among testing conditions, times, and places, do not inflence the scores (**verschoor2007genetic**). The accuracy of measurement (test reliability), and the degree to which a test actually measures the ability it is supposed to measure (test validity) are fundamental issues in the development of standardized tests and in the production of a fair scoring system. As the test takers do not necessarily get the same test, it is fundamental that the conditions of reliability and validity are fulfilled through a systematic approach to test development, in which the assembly procedures play a very important role.

According to **downing2006twelve**, several phases in such an approach can be distinguished: project plan, content definition, test specifications, item development, item banking, pretesting, item bank calibration, test assembly, test production, test administration and reporting results. Several methods are used for test assembly nowadays. A common practice is selection by hand, usually after item analysis either based on classical test theory (CTT) or item response theory (IRT). Larger testing programs, however, have better access to resources like sophisticated item banking systems, opening the possibility to improve their test assembly process by means of automated test assembly (ATA). ATA has several advantages over manual test assembly. First of all, the test specifications

should be defined rigorously early on reducing the need to repeat some phases of the test development. More importantly, ATA is the only way to find optimal or near-optimal solutions starting from large item banks, for which manual assembly is not feasible due to the large number of possible combinations of items. As a consequence, ATA is fundamental to make measurements comparable while reducing operational costs.

The development of computer technologies enabled national test institutes to introduce the use of computers in educational testing. In 2018, the same INVALSI adopted computer based testing (CBT) for grades 8 and 10 instead of the traditional paper and pencil (P&P) testing. This chapter discusses the basics of ATA and the specific characteristics of the assembly procedures used in classical settings for automated test assembly. In Section 1, the key theoretical issues of CTT and IRT, which represent the fundamentals of ATA, are briefly introduced. In Section 2, the main features of the optimization models used in ATA are presented together with an in-depth description of the Lagrange relaxation (**fisher1981lagrangian**) applied to the mentioned ATA models.

2.1 Test Theories

In educational and psychological measurement, the process of test development follows specific steps (see, e.g., **downing2006twelve**) which are guided from strong methodological test theories: classical test theory (CTT) and item response theory (IRT), respectively described in **lord1952theory** and **Hamb85**; **Hamb91**. The statistical framework of test theories was instead introduced in **lord1968statistical**. The major focus of CTT is on test-level information, while IRT primarily focuses on the item-level information. Especially IRT provides a good framework for automated test assembly methods because it is possible to compute the item Fisher information, a key object used to build optimal test in terms of accuracy of ability estimation.

2.1.1 Classical test theory

The fundamental assumption of CTT is that the true score of a person on a measurement, which is an unobservable variable, is the expected value of the observed

raw score, i.e., the expected number-correct score over an infinite number of independent test administrations (**novick1966axioms**; **lord1968statistical**). Given a person n, their observed score X_n is defined as the sum of the true component and a random error component, as follows

$$X_n = \tau_n + \epsilon_n, \tag{2.1}$$

where τ_n is the true score and ϵ_n is the normally distributed error term with expected value equal to zero and constant variance. Measurement errors are assumed to be uncorrelated over repeated administrations.

Within CTT, the test development process is based on checking the test validity and test reliability. In particular, the fundamental concept of reliability is concerned with the internal consistency of the test, i.e., the degree to which all item scores in a test correlate positively with one another. Given an item bank with items indexed by $i = 1, \ldots, I$ and let $\mathcal{I}_t$ indicate the set of items taken in test t, the most popular index used to assess test reliability is the Cronbach's α **cronbach1951coefficient**, which is defined as

$$\alpha = \frac{|\mathcal{I}_t|}{1 - |\mathcal{I}_t|} \left(1 - \frac{\sum_{i \in \mathcal{I}_t} \sigma_i^2}{\sigma^2}\right), \tag{2.2}$$

where $|\mathcal{I}_t|$ is the cardinality, i.e. the length of test t, σ^2 is the variance of the total test score and σ_i^2 is the variance of item i. The closer α to 1, the higher the test reliability. Unfortunately, this function is non-linear with respect to the items and hence is not actively used in test assembly models which are usually linear. However, it can be employed to assess the consistency of the assembly results.

Two other item properties play an essential role in CTT: item difficulty and item discrimination. For dichotomously scored items, the difficulty of an item i, in the item bank, is defined as the expected score given by a randomly selected examinee from the population of interest and is usually denoted by π_i or p-value p_i. The item discrimination is operationalized as the point-biserial correlation between the item score and the raw observed test score and denoted as ρ_{it} for item i and test t.

The main limitations of CTT are the test-dependent score, the existence of a single standard error of measurement for the population, and the focus at test

level. These drawbacks are overcome by using IRT.

2.1.2 Item response theory

An exhaustive introduction about IRT models can be found in **lord1968statistical**; **Hamb91**. Here, we are going to discuss IRT models with the only intention to supply their fundamental ideas and statistical notation needed to understand the subsequent sections. Very briefly, an IRT model expresses the relation between the observable variables (the item responses) and the unobservable, latent ability through a probabilistic model. It is assumed that the performance of an examinee can be explained through his/her latent ability(ies). The most popular IRT models are based on the unidimensionality assumption, i.e., the existence of a single latent ability independent between test-takers and the correct modelling of the phenomenon. However, multidimensional IRT models have been developed as well. A second assumption is local independence, which means that the item responses are statistically independent, conditional to the specification of the correct dimensionality (a single ability or a set of abilities).

A unidimensional IRT model for dichotomous items (e.g., correct-incorrect) expresses the probability of endorsing an item as a function of the underlying ability and a set of item parameters representing the item properties through a S-shaped curve called item characteristic curve (ICC). This function is non linear in the ability, and it is monotone increasing because the idea behind these models is that the higher a person is located on the latent trait the higher is the probability she/he will give a correct answer. Different IRT models are characterized by the item response type, the number of item parameters, the latent ability structure, and the functional form.

The Rasch model, also known as the one-parameter logistic (1PL) model, has an ICC expressed by the following formula

$$P_i(\theta) = \frac{\exp(b_i + \theta)}{1 + \exp(b_i + \theta)} \tag{2.3}$$

where $P_i(\theta)$ is the probability of a correct answer to item i for an examinee of ability level $\theta \in (-\infty, \infty)$, and the parameter $b_i \in (-\infty, \infty)$ represents the easyness of the item i. We opted for the parametrization proper of the regression

models because it simplifies the computations (addends in the exponential have positive signs).

An example of ICCs for the Rasch model is shown in Figure 2.1. Note that the curves differ only by their location on the ability scale: the easiest item is C1 while the most difficult one is the C4. In fact, the Rasch model assumes that the easyness parameter is the only item characteristic that influences the examinee's performance.

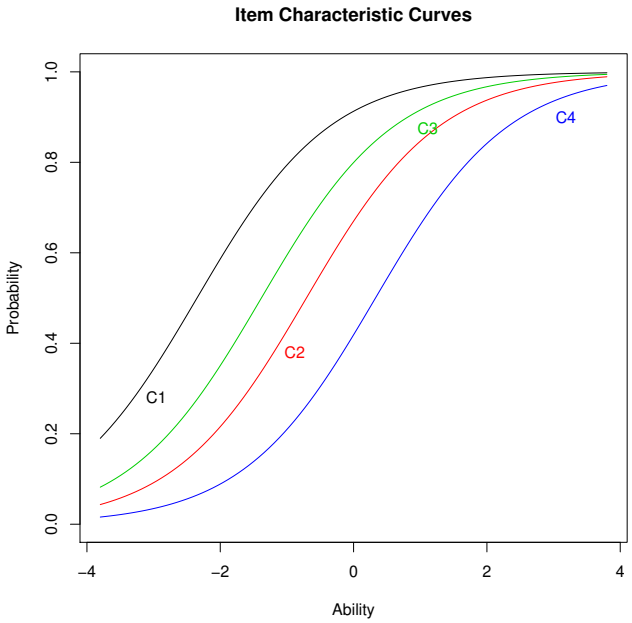


FIGURE 2.1: ICCs of 4 items C1-C4 with different difficulties according to the Rasch model.

IRT models allow the simultaneous estimation of the item parameters and the examinees' abilities. The *calibration* process usually involves the estimation of item parameters from pre-test response data. The *scoring* phase deals with the estimation of the ability scores of the candidates. Once the item parameters have been estimated, it is possible to understand how precise the test is at various ranges of the latent ability by using the test information function (TIF), which is defined as the sum of the item Fisher information for all the items in the test. In fact, under the maximum likelihood (ML) scoring, the Fisher information is asymptotically equal to the inverse of the variance of the ML estimator as follow,

$$I(\theta) = \frac{1}{\mathrm{Var}(\hat{\theta}|\theta)}. \tag{2.4}$$

The TIF has a very favourable property that is the additivity (and hence linearity) over the items of a test. Given a test with k items, the TIF is equal to

$$I(\theta) = \sum_{i=1}^{k} I_i(\theta), \tag{2.5}$$

where $I_i(\theta)$ is the item information function (IIF) for item i. Expressions for the IIFs can be easily derived within the framework of IRT.

For example, for the Rasch model, the IIF of item i is equal to

$$I_i(\theta) = P_i(\theta)(1 - P_i(\theta)) = \frac{\exp^{(b_i+\theta)}}{[1 + \exp^{(b_i+\theta)}]^2}. \tag{2.6}$$

An example of IIFs for the Rasch model is shown in Figure 2.2. The items are maximally informative (information equal to 0.25) at the ability level corresponding to the easyness parameter.

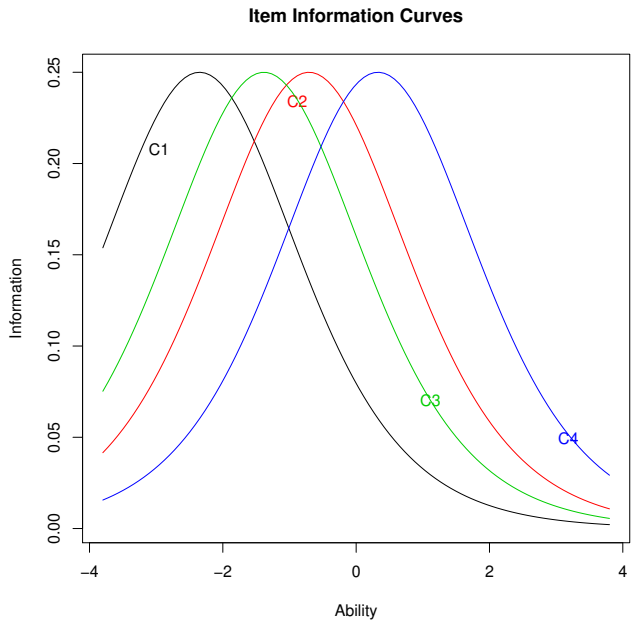


FIGURE 2.2: IIFs of 4 items C1-C4 according to the Rasch model.

Figure 2.3 shows a test information function for a test with 10 items. The

Fisher information function is a critical element in test assembly because of its linearity and its easy interpretation. Tests can be assembled merely through the selection of appropriate items out of an item bank, one way to do so is to use mathematical programming techniques like 0-1 linear programming (LP) or mixed integer programming (MIP) models. Using these approaches the tests can be built by, for instance, maximizing the TIF at predefined θ points (MAXIMIN in this dissertation), or matching it with known optimal values (MINIMAX) with linear restrictions on the values of items properties.

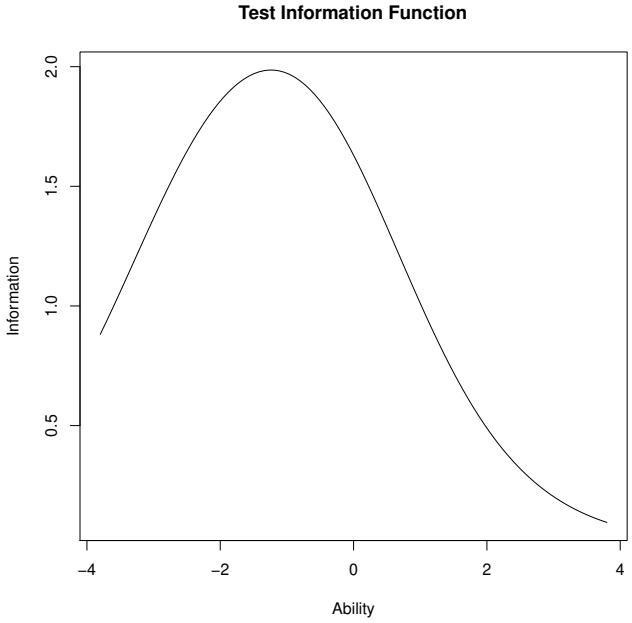


FIGURE 2.3: An example of test information function (TIF).

The 2-parameters logistic model (2PL) (**birnbaum1958**) is a generalization of the Rasch model. Like the Rasch model, the 2PL assumes that the probability of a correct response to an item i depends on the difference between the respondent's trait level θ and the easiness of the item b_i (note that, since b_i has positive sign in our parametrization, it is interpreted as easiness). Also, the 2PL postulates that, for every item, the association between this difference and the response probability depends on an additional item discrimination parameter: a_i. The ICC of the 2PL IRT model is given by:

$$P_i(\theta) = \frac{\exp(b_i + a_i\theta)}{1 + \exp(b_i + a_i\theta)} \tag{2.7}$$

The probability of a positive response (e.g., solving an item correctly) is strictly monotonic, increasing in θ and b_i. The item discrimination parameter characterizes how fast the probability of endorsing the item approaches 1.00 with increasing trait level when compared to other items. In other words, the model accounts for the possibility that responses to different items are not equally strongly related to the latent trait. The discrimination parameter describes how well a particular item discriminates between examinees with different trait levels compared to other items on the test.

We decided to focus on the 2PL model because it is widely used in practical applications, **desimone2015item**

2.2 Automated Test Assembly

In the late 1970s, the transition from paper-and-pencil to computer-based tests has begun in the United States, increasing the efficiency and the accuracy of the assessment tools. First of all, the administration of tests by computers streamlined the process of data collection and recording, allowing to have scores immediately available and free of data entry errors. Secondly, skills and variables that could not be assessed or measured by paper-and-pencil tests like higher-level thinking skills, complex problem-solving and response times now are quickly recorded and evaluated thanks to the computers. The growing number of credentialing exams for allowing the practice of a profession, admission tests for granting the access to the universities and standardized national tests for comparing abilities among different settings increased the importance of the use of test scores and hence the content and statistical features of the tests became crucial for the test validity and reliability.

Automated test assembly was born in this framework where fulfilling several specific requirements on the tests such as reducing the length of the test, maximizing the precision of the ability estimates, building tests with the same difficulty level and moreover, were needed. In practice by ATA models, a test developer can impose any content and statistical criteria, from here called constraints, by specifying them in the form of linear (in)equalities. Also, a test developer could choose an objective function to serve as the goal for test assembly. Therefore, the computer can find a set of test items that optimally meet these specifications.

It is clear that test assembly is at the heart of the test development process but the automatically produced test is only a first draft of tests that could then be reviewed and re-examined by committees.

At the time when the test is assembled, the input is required from three other processes: test specification, item construction and test data analysis. A good test needs good specifications, good items and good data.

2.2.1 The Item Bank

Once the calibration has been done, the items and their estimated and structural properties are stored in the *item bank* (or item pool). Afterwards, we can move on to the test assembly procedure in which the items will be selected depending on those distinctive characteristics.

Table **??** shows an example of the structure of an item bank with I items, where the items are displayed by row while in the columns we find the items features, from left to right: the identifier (*ID*), the IRT difficulty parameter (b) together with its standard error (b_{se}), CTT difficulty (*p-value*), content attributes (*TYPE*, *PROCESS*, *DOMAIN*), and relational attributes that specifies if the item belongs or not to a specific set (*FRIEND SET 1*, *FRIEND SET 2*, *ENEMY SET 1*, *ENEMY SET 2*).

Examinees can get different sets of items because, thanks to the calibration via IRT, the items are set on the same scale and the examinees' scores can be compared.

2.2.2 Types of Assembly Models

Given an item bank containing a sufficient number of calibrated items, is it possible to assemble one or more test forms which features can be similar or diverge. When we need to assemble only one test form, we are speaking about *single test assembly* models while if the tests are more than one the models are for *multiple test assembly*. In particular, if the obtained ones are similar under their psychometric characteristics, they are called *parallel* [1]. In this work, we will focus on the latter category of automated test assembly models.

[1] Other details in Section 2.2.4.

To solve this type of problems, in the last decades, three modes of automated test assembly became prevalent: sequential, simultaneous single or multiple test assembly, and adaptive tests. By the first two, test forms are entirely built before the administration of the test while in the adaptive framework, each test form is assembled during the testing procedure.

Sequential test Assembly

The straightforward technique for assembling single or multiple test forms is to populate the forms with items in a sequential way. This is being done by selecting items or groups of items and removing them from the pool; then the model is adapted to the new pool to try to fit the next form. In the case of assembling parallel forms, this method has two serious disadvantages. First, if the forms are assembled one after the other, the value of the objective function for the solution of the model tends to decrease due to the fact that items with the best values will be selected first. As a consequence, the forms cannot be parallel. The second disadvantage of this approach is the possible infeasibility of later models in the sequence. On the other hand, by this type of model, an extensive range of functions can be optimized, e.g., the linear structure of the problem can be relaxed. Most of these problems nowadays are solved using ad hoc greedy and heuristic techniques.

Simultaneous test assembly

In the sequential approach, an incremental number of different models must be optimized in order to obtained multiple forms causing the disadvantages cited in the last paragraph. Those can be overcome by applying simultaneous test assembly models in which the solution, and hence, the test forms, is obtained by solving only one model. They are usually represented by 0-1 integer programming problems and solved by linear programming techniques. The model can be reformulated as a mathematical optimization problem using decision variables. Decision variables are defined such that the solution of the optimization problem (i.e., the set of values for which the objective function is optimal and all constraints are satisfied) identifies the best decision that can be made. The decision variables $\{x\}_{it}$ are binary (0-1) since they represent the inclusion (1) or exclusion

(0) of the item i in the test t. In the last decades, the algorithms needed to solve such optimization problems were improved, and nowadays, powerful implementations of them are available as commercial or open-source software.

Adaptive tests

In the adaptive approach, one item is optimally picked from the pool at a time. The person's ability estimate is updated during the test, and the next item is chosen to be maximally informative at the last ability update. The test is then tailored to the candidate. Because ability estimates converge to the person's true ability level, the item selection improves during the test and the ideal test with maximum information at the person's true ability level is approached. In the early 1990s, computerized adaptive testing (CAT) was implemented in large-scale testing programs, and nowadays large numbers of subjects are tested worldwide using this type of test assembly. One of the most important benefits of this method is that it's more efficient, i.e., it is possible to get more precise ability estimates with fewer items than the previous standard approaches, but at the cost that it's quite impossible to ensure the congruence between tests since all the test takers will get a different set of items from the pool.

The description of different approaches to assemble tests is not finished here. Because of its advantages, the approach dealt with in this report is the simultaneous tests assembly mode. In Section 2.2.3 and Section 2.2.4 we introduce the reader to the standard test assembly models used to build first, a single test form and secondly, as a generalization, multiple test forms.

2.2.3 Single Test Assembly

As already discussed, a new testing program starts with formulating the set of specifications for the test to be met, that here we call *desiderata*. Sometimes they are verbally expressed as a set of learning objectives or a list of dos and don'ts for the test developer, but they can also be well-structured in tables specifying how many items should have certain content or, even better, the distribution of items according to their specific characteristics.

1.	Average p-value of the test between .40 and .60
2.	Number of items on applications equal to 24
3.	Reliability of the test should be as high as possible
4.	Items 73 and 100 never in the same test
5.	Test information function as close as possible to the target
6.	Items 33, 45 and 12 must be in the same test

TABLE 2.1: Example of desiderata.

Once the desiderata have been collected, they must be translated into a standardized language used in test assembly problems. The standard form of these problems is composed of an *objective function* to be optimized subject to several *constraints*, where the latter define a possibly feasible set of tests for a given item bank, and the former expresses our preferences for the tests in this feasible set. If the specifications have been formulated in a simple, concise, but complete way, it is possible to determine whether they are objectives or constraints. These requirements are crucial for a correct translation of the desiderata into the standard language for test assembly problems: disambiguations and possible complications may arise in test design if these principles are not satisfied. An example of verbal test specifications is described in the Table 2.1, which is partially extracted from **VDL2005**.

The specifications in Table 2.1 may represent either objectives or constraints; for example, points 1,2,4 are constraints while 3 and 5 are objectives. It is clear now that the objectives involve maximize or minimize some attributes, such as minimize the gap between the test information function at some θ point to its target (5) or maximize the reliability of the test (3) and on the other hand constraints impose a bound on an attribute of the test or of the items, such as limiting the difficulty of the test (1), fixing the number of items having specific characteristics (2) or considering enemy sets (4) or also item (friend) sets (6). An exhaustive classification of specifications together with examples of not well-expressed desiderata can be found at pages 34-39 of **VDL2005**.

optimize	*Objective function*
subject to	
	Constraint 1
	Constraint 2
	⋮
	Constraint N

TABLE 2.2: Standard form of a test assembly problem.

Standard form of a test assembly problem

If a set of desiderata is well specified, they can always be represented in the standard form of Table 2.2.

Only one objective can be optimized at a time; if we have more than one function to optimize some tricks can be applied to transform the objectives into constraints. On the other hand, there is no upper limit for the number of constraints, provided our solver can handle the problem. If, at least one combination of items that meets all the constraints do exist, the set of these combinations is called *feasible set*; if this set is empty, we say that the model in *infeasible*. The subset of tests in the feasible set that optimizes the objective function is called *optimal feasible solution*.

Decision variables

Modelling a test assembly problem does not imply only defining objectives and constraints, but we also need the so-called *decision variables* which represent the possible combination of items that compose the test form in a mathematical formulation. If we need to assemble a single test from a pool of I items, we can choose as decision variables I binary variables in the form:

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is in the test,} \\ 0 & \text{otherwise.} \end{cases}$$

We can also write the x_i in an algebraic way using the vector x of length I. Therefore, we have 2^I possible combinations of values the vector x can take.

These combinations decrease in number if we add constraints to the model. In the following Section, we will also use non binary variables like integer o continuous, which are often useful to reduce the number of objectives in the problem. Once the decision variables have been identified, the process of modelling a test assembly problem goes through the steps of modelling the constraints and objectives and solving the model looking for an optimal solution. In the following Section, we will present the basic formulations for some common types of constraints and objectives. The last step consists of solving the model by using a computer program which implements some mathematical or numerical algorithms.

The model for assembling a single test

The models presented in this Section are based mainly on item response theory attributes. Here, we will make an almost exhaustive list of categories of constraints that can be added to a model for single test assembly together with possible objective functions.

The standard model for the assembly of a single test with a generic quantitative objective from a pool of I items indexed by $i = 1, \ldots, I$ is

$$\text{optimize} \sum_{i=1}^{I} q_i x_i \quad \text{(objective)} \tag{2.8a}$$

subject to

$$n_c^{\min} \leq \sum_{i \in V_c} x_i \leq n_c^{\max}, \ \forall c \quad \text{(categorical constraints)} \tag{2.8b}$$

$$b_q^{\min} \leq \sum_{i=1}^{I} q_i x_i \leq b_q^{\max}, \quad \text{(quantitative constraints)} \tag{2.8c}$$

$$n^{\min} \leq \sum_{i=1}^{I} x_i \leq n^{\max}, \quad \text{(test length)} \tag{2.8d}$$

$$\sum_{i \in V_e} x_i \leq 1, \quad \forall e \quad \text{(enemy sets).} \tag{2.8e}$$

Then, definition of variables

$$x_i \in \{0, 1\}, \ \forall i \quad \text{(decision variables).}$$

where the sums in the constraints may be bounded only on one side, in which case we only need one of the two inequalities.

For a qualitative variable, we use V_c to represent the set of indices of the variable of the items in subset c. Allowing the test to have several items with specific qualitative attribute represented by V_c between $n_c^{\min}$ and $n_c^{\max}$ means adding a *categorical constraint* to the model. An example of a categorical constraint is "the test must contain at least 10 items in problem-solving", where the set of items in the pool that has the attribute "problem solving" is represented by $V_{\text{problem solving}}$ and $n^{\max}_{\text{problem solving}} = 10$.

On the other hand, if we want to bound the value of a quantitative variable (i.e. that can take numerical values) addressed by the symbol q between the values $b_q^{\min}$ and $b_q^{\max}$ we are constraining the model adding a *quantitative constraint* where q_i is the value that the item i takes for that variable. Suppose we want to assemble a test with the following criterion "the maximum of words must be 300", q_i will serve as the number of words displayed by the item i and $b_q^{\max} = 300$.

The interpretation of the *test length* constraint is straightforward as it is a special case of a quantitative constraint where $q_i = 1$ for each $i = 1, \ldots, I$.

If there are *enemy sets* in our pool, then the items in one of these sets, called V_e, cannot be picked together, e.g., the desideratum "items 23 and 46 cannot be in the same test" means that items 23 and 46 are enemies and both are in the enemy set V_e. Therefore, the sum of the corresponding decision variables x_{23} and x_{46} cannot be more than 1.

Item sets

Tests with sets of items organized around common stimuli (known as item sets or friend sets) are popular because of the efficiency of their format. By combining more than one item with the same stimulus, we can ask questions using more complex stimuli, such as reading passages, descriptions of cases, or problems with data in a set of tables or graphs, without having to sacrifice too many items for the test to meet the time limit. However, the presence of such sets in the item pool complicates the process of assembling the test. In particular, if the items are grouped in S item sets, or stimuli, indexed by s, there are three ways to deal with this problem (**VDL2005**).

- The *power set method* that consists in summarizing all the quantitative and qualitative attributes by summing or averaging them taking as groups the item sets; if we only need to constraint the test at the set/stimuli level the problem decreases in size since you don't have I decision variables anymore but just S. This method is not efficient if you have variables that cannot be summarized, such as standard errors and if you want to keep some constraints at item level since you need to consider again the original decision variables x_i increasing the size of the model.

- The *pivot-item method* in which each set is represented by a pivot decision variable $x_{i_s^*}$ arbitrarily chosen. Therefore, we can use the variable for a pivot item as a carrier of the attributes of both its stimulus and item set, and we can drop the stimulus variables in the model.

- The *two-stage method* that is a sequential approach with two phases, in the first, stimuli are selected and secondly, the model chooses the items with those stimuli.

Given the earlier warnings about sequential approaches we do not suggest the two-stage method and since all the variables in our item bank are summarizable (e.g., the item information function is additive, so it is possible to sum it for all the items in the stimulus) we prefer to adopt the power set method which helps to decrease the size of the problem.

The power-set formulation needs a phase of preprocessing of the item bank. Instead, for the formal representation of the model, we refer to **VDL2005**.

Single test MINIMAX

Once we defined all the constraints, and checked that the model is feasible, we need to choose an objective to optimize. A first option is to choose absolute targets for the TIF, targets are the values that a goal TIF assumes on a fixed number of θ points along the θ scale, these values must be chosen by test specialists who knows how much precision is required to estimate the abilities of the students at each ability level. That is the reason why absolute targets are used almost exclusively when tests are assembled to be parallel with respect to a known reference test. Formalizing this requirement in the standard form of test assembly models

will produce a multi-objective test assembly problem that must be reformulated using the MINIMAX approach explained below.

In particular, with the following addition to the model (2.8) we ask that the TIF of the resulting test approximates with minimum negative and positive deviations (i.e., with the highest precision) the chosen goal TIF in a finite set of points V on the θ scale, which we denote as T_k with $k \in V$:

$$\text{minimize} \quad y \quad \text{(objective)} \tag{2.9a}$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_k) x_i \leq T_k + y, \, \forall k \in V \tag{2.9b}$$

$$\sum_{i=1}^{I} I_i(\theta_k) x_i \geq T_k - y, \, \forall k \in V \tag{2.9c}$$

$$y \geq 0.$$

More θ points we choose more the TIF of the assembled test will meet the auspicable one, usually 3-5 points are enough to have a good approximation.

Single test MAXIMIN

If a reference test or absolute targets are not available, an alternative approach is trying to achieve the best predictive validity for the test, that is maximizing the TIF in some chosen theta points. This goal can be met not only setting the location of the peaks of the TIF but also defining its shape in the entire θ scale, which is to say imposing relative targets.

So, denoting with R_k the relative targets for each θ_k with k in the chosen set V of ability points in which we want to control the shape of the TIF, we must fix one of the relative targets to a value (e.g. $R_1 = 1$) and all the other target values must be adjusted correspondingly trying to reproduce the wanted shape.

Like the absolute target model, this method leads to having more than one objective function and also, in this case, it is possible to rely on a simplifyed

approach called MAXIMIN. The model can be formalized as follows:

$$\text{maximize} \quad y \quad \text{(objective)} \tag{2.10a}$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_k) x_i \geq y R_k, \ \forall t, k \in V \tag{2.10b}$$

$$y \geq 0.$$

Also here, the practice suggests that the minimum number of θ points in which maximizin the TIF must be 3 or 5.

2.2.4 Multiple Simultaneous Test Assembly

In order to discourage the phenomenon of cheating, it is necessary to administer different items to the test takers. This can be achieved building more than one test form that contains different items preserving some overall mutual psychometric features, such as the same test difficulty or same content structure such as the same percentage of items of a certain stimulus. These tests are called *parallel* (or interchangeable) and the procedure aimed to perform this task is called "multiple test assembly". In particular, test forms are defined to be *weakly parallel* if their information functions are identical **same77**. On the other hand, test forms are *strongly parallel* if they have the same test length and the exact test characteristic function **Lord80**.

As a consequence, we typically have problems with more objectives than those presented in Section 2.2.3 for a single test assembly. For example, if we assemble T tests and each test has to meet a target for its information function at K ability points (the same points for each test t, with $t = 1, \ldots, T$), the problem has at least $T \times K$ objectives. However, these sizeable multi-objective test assembly problems can be solved using a direct generalization of the approaches for single test assembly. In the following, we will present a general model for simultaneous assembly of a set of tests (that is, as a solution to a single model) that produces an optimal solution. This type of assembly requires a reorganization of the problem using a different version of the decision variables. These variables have double

indices[2], one for the items in the pool and the other for the test forms. For the current problem, the variables become

$$x_{it} = \begin{cases} 1, & \text{if item } i \text{ is assigned to test } t \\ 0, & \text{otherwise} \end{cases}$$

for all i and t. Like before, we need to complement these variables with a set of constraints that keep their values consistent. The adapted single test assembly model is presented here followed by constraints that arise only in the case of multiple test assembly.

The model for assembling multiple tests

Using the above decision variables, any model for a single test can be reformulated as a model for multiple tests. To illustrate this statement, we reformulate the standard model for a single test in (2.8a)-(2.8e). The model is

$$\text{optimize} \sum_{t=1}^{T} \sum_{i=1}^{I} q_{it} x_{it} \quad \text{(objective)} \tag{2.11a}$$

subject to

$$n_{ct}^{\min} \leq \sum_{i \in V_c} x_{it} \leq n_{ct}^{\max}, \ \forall c, t \quad \text{(categorical constraints)} \tag{2.11b}$$

$$b_{qt}^{\min} \leq \sum_{i=1}^{I} q_i x_{it} \leq b_{qt}^{\max}, \quad \forall t \quad \text{(quantitative constraints)} \tag{2.11c}$$

$$n_{t}^{\min} \leq \sum_{i=1}^{I} x_{it} \leq n_{t}^{\max}, \quad \forall t \quad \text{(test length)} \tag{2.11d}$$

$$\sum_{i \in V_e} x_{it} \leq 1, \quad \forall t, e \quad \text{(enemy sets)} \tag{2.11e}$$

Then, the definition of variables

$$x_{it} \in \{0, 1\}, \ \forall i, t \quad \text{(decision variables)}$$

[2]We use a matrix representation only for a better visual idea, actually they are still vectors but of bigger size.

The changes in (2.11) relative to the original model in (2.8a)-(2.8e) are:

1. the replacement of the variables x_i by x_{it};
2. the extension of the objective function to the case of T tests;
3. the indexing of the bounds in the constraints by t to enable to assemble tests with different specifications.

The generalization of the objective function in (2.11a) is simple and consists of taking an (unweighted) sum over the tests.

Item sets

The power-set method presented in 2.2.3 can be easily generalized to the case of multiple tests assembly, for the sake of brevity we prefer to skip the details and still refer to the van der Linden book **VDL2005**, Section 7.2.

Item use

If we want to control the minimum or the maximum number, respectively $n_i^{\min}$ and $n_i^{\max}$, of tests in which an item i can be assigned, we have to consider the following constraints:

$$n_i^{\min} \leq \sum_{t=1}^{T} x_{it} \leq n_i^{\max}, \ \forall i, \quad \text{(item use)} \tag{2.11f}$$

Test overlap

Since we are creating more than one test form we are not only concerning the properties of each form singularly but also the relationship between them, one of these is the *test overlap*, that is the number of items two forms share. Sometimes, it is not important to control for this specification, especially if an item use constraint has been fixed. However, if we do not want any overlap between all the pairs, as an example, we have to add these constraints to (2.11)

$$\sum_{t=1}^{T} x_{it} \leq 1, \ \forall i, \quad \text{(no overlap)} \tag{2.11g}$$

The latter is not enough if the test developer wants to keep the same but not a null level of overlap between all the possible pairs of tests. In this case, the model has to be modified not only adding new constraints but also a substantial number of variables (luckily binary). In particular, controlling for test overlap means adding quadratic constraints of the form:

$$o_{tt'}^{\min} \leq \sum_{i=1}^{I} x_{it} x_{it'} \leq o_{tt'}^{\max}, \ \forall t \neq t', \quad \text{(fixed overlap NON LINEAR)}$$

Those types of constraints must be linearized in order to use the standard LP solvers, so they must be replaced by the following variables

$$z_{itt'} = \begin{cases} 1 & \text{if item } i \text{ is both in test } t \text{ and test } t' \text{ (i.e. } x_{it} = x_{it'} = 1) \\ 0 & \text{otherwise.} \end{cases}$$

This modification increases the size of the model of $I * \binom{T}{2}$ binary variables. Together with the new variables, the constraints must be replaced by

$$o_{tt'}^{\min} \leq \sum_{i=1}^{I} z_{itt'} \leq o_{tt'}^{\max}, \ \forall t \neq t', \quad \text{(fixed overlap LINEAR)} \tag{2.11h}$$

Integrality constraints:

$$z_{iit'} \geq x_{it} + x_{it'} - 1 \ \forall i, t \neq t' \tag{2.11i}$$

$$2z_{iit'} \leq x_{it} + x_{it'} \quad \forall i, t \neq t' \tag{2.11j}$$

$$z_{itt'} \in \{0, 1\}, \quad \forall i, t \neq t'$$

The last two constraints are necessary to keep the values of $z_{itt'}$ consistent with their definitions.

Multiple MINIMAX

An example of an objective for multiple test assembly is the generalization of the MINIMAX principle presented in 2.2.3, formally

$$\text{minimize} \quad y \quad \text{(objective)} \tag{2.13a}$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_{kt}) x_{it} \leq T_{kt} + w_t y, \ \forall t, k \in V_t \tag{2.13b}$$

$$\sum_{i=1}^{I} I_i(\theta_{kt}) x_{it} \geq T_{kt} - w_t y, \ \forall t, k \in V_t \tag{2.13c}$$

$$y \geq 0$$

where the target values T_{kt} are indexed by t to allow us to set different targets for different tests. Besides, a different set of values V_t for each test can be given; we can specify the target values at a different set of θ values for each test. Finally, we have added weights w_t to have the option of weighting deviations from target values differently for several tests. If our goal is to assemble parallel tests, the targets and weights will be equal for all t.

Multiple MAXIMIN

The approach used in Section 2.2.3 may be generalized to the case of multiple test assembly with this mix of objective and constraints

$$\text{maximize} \ \ y \quad \text{(objective)} \tag{2.14a}$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_{kt}) x_{it} \geq y R_{kt}, \ \forall t, k \in V_t \tag{2.14b}$$

$$y \geq 0,$$

where the R_{kt} may be chosen equal between tests, i.e. $R_{kt} = R_{k't'}$ with $t \neq t'$ and $\forall k = k'$ ensuring the parallelism.

2.2.5 Lagrange relaxation

In the previous sections, we introduced the classical linear models for test assembly formalized in the book **VDL2005**. In the chapeter 4.3 of this book, an

approach to approximate those models by means of Lagrange multipliers is briefly mentioned, that is a *Lagrange relaxation* of the test assembly model (2.11). This method is beneficial in case of infeasibility, a situation very common in practice when there is a large number of constraints or optimization variables. For example, when a large-scale test assembly model of the type (2.11) is given to a MILP solver, it may happen that a set of tests that meets all the constraints cannot be found, even if the user imposed time limit is very long. This is a very big issue for the test assembler because, usually, the solver just says that the problem in infeasibile without returning any diagnostic, so he/she does not know which constraints made the problem infeasible and he/she does not have any assembled test in his/her hands, not even a barely good one.

The Lagrange relaxation (**fisher1981lagrangian**) overcomes this issue by including in the objective function the absolute deviation, called *violation*, of the incumbent from the constraints. In this way, we try to solve a simplified version of the problem, and we obtain an upper bound for the optimal solution of the initial problem. Even if the problem is highly infeasible, the solver returns the most feasible combination of variables that maximizes the modified objective function. Thus if we have the most general optimization model:

$$\text{maximize } f(x) \quad \text{(objective)} \tag{2.15a}$$

subject to

$$g_m(x) \leq 0, \forall m \quad \text{(constraints)} \tag{2.15b}$$

$$x \in \mathbb{R}^d, \quad \text{(decision variables)}$$

Its Lagrange relaxation will be:

$$\text{maximize } f(x) - \lambda \sum_m g_m(x) \quad \text{(objective)} \tag{2.16a}$$

$$x \in \mathbb{R}^d, \quad \text{(decision variables)}$$

where $f()$ ad $g()$ are generic functions from $\mathbb{R}^d$ to $\mathbb{R}$ and λ is a constant called the Lagrange multiplier, it has the role to weight the violations of the constraint

in the new objective function. We opt for a modification of the previous model to allow the violations to interfere in the optimization only when the constraints are not met. The (2.16) will become:

$$\text{maximize } f(x) - \lambda \sum_m \max\left[0, g_m(x)\right] \quad \text{(objective)} \tag{2.17a}$$

$$x \in \mathbb{R}^d, \quad \text{(decision variables)}$$

In the field of neural networks, $\max\left[0, g_m(x)\right]$ is called *RELu* activation function and, in machine learning, it is mentioned as *hinge loss* and it is used for training classifiers. In this context this function serves a different purpose but we will continue to use this names as a reference.

In the case of test assembly, the relaxation of the classical model (2.11) is still linear. We summarise this approach by an example of the Lagrange relaxation applied to a single test assembly model of the form (2.8) with an objective function to maximize. Suppose we have two set of indices $m_u \in M_u$ and $m_l \in M_l$ for upper and lower bound constraints and $m \in M_u \cup M_l$. we can rewrite the relaxed version of the model (2.8) as follows:

$$\text{maximize } \beta \sum_{i=1}^{I} q_i x_i - (1-\beta) \sum_{m=1}^{m_u+m_l} z_m \quad \text{(objective)} \tag{2.18a}$$

subject to

$$\sum_{i=1}^{I} c_{im_u} x_i - ub_{m_u} \leq z_{m_u}, \forall m_u \quad \text{(generic constraints with upper bound)} \tag{2.18b}$$

$$-\sum_{i=1}^{I} c_{im_l} x_i + lb_{m_l} \leq z_{m_l}, \quad \forall m_l \quad \text{(generic constraints with lower bound)} \tag{2.18c}$$

$$x_i \in \{0,1\}, \ \forall i \quad \text{(decision variables)}$$

$$z_m \in \mathbb{R}^+, \ \forall m \quad \text{(violations)}$$

Notice that any of the linear constraints showed in the previous subsections

can be represented as a generic constraint either of the form (2.18b) or (2.18c). The definition of z_m as a positive real number let the solver look for solutions that makes the sum of z_m go to zero and hence satisfy all the constraints.

To help the test assembler to control the trade-off between optimality and feasibility of the final solution, we let them to choose the Lagrange multiplier $(1 - \beta)$, where $0 \leq \beta \leq 1$. For example, if the assembler prefers a solution more accurate in the ability estimation sacrificing the fulfillment of tougher constraints they might choose a $\beta > 0.5$ Viceversa if they need to respect all the requirements in the desiderata they will select a lower β. For a balanced solution, the β must be equal to 0.5. In chapter 4.2 we will refer to the model without Lagrange relaxation (2.15) as the *strict* model, instead, the model of type (2.16) will be called *relaxed* model.

3 IRT item parameter calibration in Julia

Item response theory (IRT) models, already introduced in Chapter 2.1, are a class of statistical models which are intended to describe the response behaviors of individuals to a set of questions which have a discrete outcome. The item responses are observable manifestation of underlying traits or constructs not directly measured which are called *latent* variables. In the framework of latent class analysis IRT models hyphotesize the latent variable as continuous, instead, the observed variables are patterns of discrete-valued responses, in particular they can be dichotomous (correct/incorrect) or polythomous. Common assumptions in IRT are the unidimensionality of the latent trait is since it's considered to be enough to describe the individuals propensity to endorse the items in the survey and conditional indipendence of the probability of a correct response given a certain level of ability. Moreover, IRT models can be seen as a specific type of fixed-effect or random-effect model (**fox2006fixed**) or, changing the parametrization, they become a particular case of latent regression models (**von2010stochastic**).

Once the model is set, the parameters and the latent variables must be estimated. For example in the 2-parameter logistic (2PL) model (**BockMislevy1982**), easiness and discrimination power of the items must be quantified. Several algorithms to retrieve the estimates exist, most of them are based on the expectation maximization (EM) paradigm and differ on the way in which the expectation is approximated and in how the optimization model is solved, some examples are the joint maximum likelihood (JML)(**lord1968statistical**), the marginal maximum likelihood (MML) (**drasgow1989evaluation**), the stochastic EM algorithm (**fox2003stochastic**) and the Bayesian methods which adopt monte carlo

Markov chain (MCMC) or Gibbs samplers technique to approximate and maximize the posterior of the latent variable (**matteucci2012prior**).

In this work the unidimensional model, dichotoumous response data and MML estimation method will be taken into account. The latter is widely recognized to outperform the JML and MCMC in terms of consistency of the estimates (**andersen1977sufficient**; **neyman1948consistent**) and computational time (**patz1999applications**) respectively. Unlike the joint maximum likelihood estimation technique, which treat each of the responses as separate observational units, the MML estimation treats only the individuals as such. To accomplish this the MML technique assumes that the latent traits are random effects sampled from some continuous distribution, usually discretized in order to compute the expected value of the likelihood in the first step of the EM algorithm, called *E-step*. In this context, the latent regression representation allows to perform faster computations of the involved functions since some arrays are computed only once and reused in all the iterations. Another advantage is the use of simple linear algebra transformation such as matrix/matrix or matrix/vector multiplications which are extremely fast in modern programming languages[1].

Moreover, since we chose to work in the `Julia` environment due to its suitability for numerical and computational tasks, we also developed a calibration algorithm for the case of the unidimensional 2PL already introduced in 2. The cubic-spline interpolation-extrapolation method (**birkhoff1960smooth**) is used to recompute the masses of the rescaled observed ability distribution at the predefined theta points. The details of the applied methodology are provided in Section 3.1 together with a software benchmarking which compare the performance of our algorithm to the `R mirt` package.

Finally, a full reading of this chapter will suggest to both practitioners and scholars a framework for building a pre-test design without knowing the item parameters and for doing a simulation study for testing the performance of an estimation method. The framework is described in Section 3.2.1. All the code is available at `https://github.com/giadasp/Calibration`.

[1] The `BLAS` routines are an example of very fast implementation of linear algebra operations. See `http://www.netlib.org/blas/`

3.1 MML Estimation

Test assembly models might be based on CTT or IRT item parameter estimates. In order to compute these values, a test assembler needs to perform what is called, a *calibration* of the items. Here, the objective is achieved using the Expectation-Maximization (EM) paradigm in which the M-step is done by maximizing the likelihood marginalized with respect to the distribution of the initialized abilities. The model is described in **bock1981marginal**. We do not use the approach based on response patterns because we allow an unbalanced design of the pre-test after which the calibration takes place.

The algorithm requires just to input the responses of the test-takers, and since we allow for an unbalanced design, the data frame can contain missings. All the others input have a default value which can be modified, in particular, the user can define[2]: the number quadrature points, K, in which approximate the integral of the ability distribution, the bounds for the ability distribution or starting knots and masses, starting points and bounds for the item parameters. The user can also define the features of the external EM algorithm and of the internal M-step, here called respectively `external and internal optimizations`, such as the stopping criteria.

In details, our EM algorithm is composed of an iterative scheme. At the beginning of each iteration, the classical E-step and M-step are resolved; there follows a phase of readjusting and rescaling of the masses of the latent distribution. When at least one stopping criterion is met, the abilities of the test-takers are estimated. Before comprehensively outlining the mentioned stages we introduce the notation used in the next paragraphs. Given a set of N respondents and I items in a pool, we want to estimate a set of vectors $\boldsymbol{\xi}_i$ of length $nPar$ of item parameters from a matrix $\mathbf{U} = \{u\}_{i,n}$ of dichotomous responses, i.e.

$$u_{i,n} = \begin{cases} 1, & \text{if the test taker } n \text{ answered correctly to the question } i \\ 0, & \text{if the test taker } n \text{ answered incorrectly to the question } i \\ missing, & \text{if the item } i \text{ hasn't been administered to the test taker } n, \end{cases}$$

[2]Default values can be found in the file `structs.jl` in the Github page

where $i = 1, \ldots, I$ and $n = 1, \ldots, N$. For example if the chosen model is the 2PL we will have $nPar = 2$, instead for the 1PL model $nPar = 1$. For simplicity we will also define the vectors $\mathbf{u}_i$ and $\mathbf{u}_n$ rispectively as the N rows and the I columns of the matrix $\mathbf{U}$. The density of the latent variable θ, the prior in a Bayesian perspective, is denoted by $p(\theta|\boldsymbol{\tau})$ where the vector $\boldsymbol{\tau}$ contains the parameters which characterize its probability distribution. In order to compute the expectation of the complete log-likelihood which consists of a complex integral, a discretization of the distribution is performed at K knots in the domain of θ, namely the X_ks for $k = 1, \ldots, K$. Since, most of the times, the latent density is not known, for each of these points the mass W_k is initialized to the value of an arbitrarily reasonable probability density (the observed empirical distribution of a previous test administration may be used) and readjusted in the calibration process to match the distribution of the ability in the population under examination.

3.1.1 E-step

In practice, in the E-step the expected value $E_{\theta|u}\left[l(\boldsymbol{u}, \theta|\boldsymbol{\xi})\right]$ of the complete data log-likelihood is computed by summing the log-likelihoods of the I items weighted by the posterior distribution $p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \boldsymbol{\xi})$ of the latent variable:

$$
\begin{aligned}
E_{\theta|u}\left[l(\boldsymbol{u}, \theta|\boldsymbol{\xi})\right] &= \sum_{i=1}^{I} \sum_{n \in n_i} \int_{\theta} \left[l(u_{i,n}|\theta, \boldsymbol{\xi}_i) + \log(p(\theta|\boldsymbol{\tau}))\right] p(\theta|\boldsymbol{u}, \boldsymbol{\tau}, \boldsymbol{\xi}) d\theta \\
&\approx \sum_{i=1}^{I} \sum_{n \in n_i} \sum_{k=1}^{K} p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \boldsymbol{\xi}) \left[l(u_{i,n}|X_k, \boldsymbol{\xi}_i) + \log(p(X_k|\boldsymbol{\tau}))\right] \\
&\approx \sum_{i=1}^{I} \sum_{n \in n_i} \sum_{k=1}^{K} p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \boldsymbol{\xi}) l(u_{i,n}|X_k, \boldsymbol{\xi}_i) \\
&+ p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \boldsymbol{\xi}) \log(p(X_k|\boldsymbol{\tau})) \\
&= \sum_{i=1}^{I} \sum_{n \in n_i} \sum_{k=1}^{K} Q(u_{i,n}|X_k) + \phi(X_k|\boldsymbol{\tau}),
\end{aligned}
\tag{3.1}
$$

where

$$\begin{aligned} p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \boldsymbol{\xi}) &= \frac{W_k \left[\prod_{i\in i_n} L(u_{i,n}|X_k, \boldsymbol{\xi}_i)\right]}{\sum_j W_j \ \left[\prod_{i\in i_n} L(u_{i,n}|X_j, \boldsymbol{\xi}_i)\right]} \\ &= \frac{W_k \ \exp\left[\sum_{i\in i_n} l(u_{i,n}|X_k, \boldsymbol{\xi}_i)\right]}{\sum_j W_j \ \exp\left[\sum_{i\in i_n} l(u_{i,n}|X_j, \boldsymbol{\xi}_i)\right]}, \\ l(u_{i,n}|X_k, \boldsymbol{\xi}_i) &= u_{i,n}\Psi(\boldsymbol{\xi}_i|X_k) - \log(1 + \exp(\Psi(\boldsymbol{\xi}_i|X_k))) \end{aligned} \tag{3.2}$$

and $\Psi(\boldsymbol{\xi}_i|X_k)$ is the linear function proper of the latent regression models. For example, in case of the 2PL model, $\Psi(\boldsymbol{\xi}_i|X) = b_i + a_i X$.

To simplify the notation we will write $L_n(X_k, \boldsymbol{\xi})$ in place of $\exp\left[\sum_{i\in i_n} l(u_{i,n}|X_k, \boldsymbol{\xi}_i)\right]$ and $l_n(X_k, \boldsymbol{\xi}_i)$ for its logarithmic transformation.

3.1.2 M-step

Noticing that the function (3.2) is separable with respect to the I items, the M-step is performed individually for each item i. In the s-th iteration the estimates of the item parameters of item i are obtained by maximizing the just defined expected log-likelihood:

$$\begin{aligned} \hat{\boldsymbol{\xi}}_i^{(s)} &= \arg\max_{\boldsymbol{\xi}_i} \quad E_{\theta|u}\left[l(\boldsymbol{u}, \theta|\boldsymbol{\xi})\right]_i \\ &\approx \sum_{k=1}^{K} \sum_{n\in n_i} Q(u_{i,n}|X_k) \\ &\approx \sum_{k=1}^{K} \sum_{n\in n_i} p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \hat{\boldsymbol{\xi}}^{(s-1)}) l(u_{i,n}|X_k, \boldsymbol{\xi}_i) \end{aligned} \tag{3.3}$$

The maximization is usually performed numerically by using non-linear solvers. In `Julia` several high-level interfaces are available to communicate with a multitude of solvers, from commercial, such as Artelis Knitro or Fico Express, to open-source, like NLopt and Ipopt [3].

[3] http://www.juliaopt.org/JuMP.jl/v0.20.0/installation/

3.1.3 The rescale of the latent distribution

The posterior distribution depends on the item parameters estimated at the prevoous step $s-1$ and to the masses W_k. The latters can be constant of may be adapted at each iteration of the EM algorithm. Since the model is identified only if the metric of the latent probability distribution is fixed (for example having mean zero and standard deviation one), the adaptation of the masses is a very delicate phase very subject to unstable results. Usually, the starting points for the masses come from the probability density function of the standardized Normal, but they can be arbitrarily initialized. After the first iterations, the W_k may be adapted by using the approach introduced by **mislevy1984estimating**:

$$\sum_{n \in n_i} \log \sum_{k=1}^{K} L_n(X_k, \boldsymbol{\xi}) W_k \tag{3.4}$$

and adding a Lagrange multiplier to constrain the sum of the masses to be equal to 1. Practically, the optimal weights are analytically computed as the average of the posterior distribution among the N respondents:

$$W_k^{(s)} = 1/N \sum_{n=1}^{N} p(X_k | \boldsymbol{u}_n, \boldsymbol{\tau}, \hat{\boldsymbol{\xi}}^{(s)}) \tag{3.5}$$

After the masses have been adjusted, it is necessary to rescale the quadrature points in order to match a predefined metric which consists of: mean, μ_θ, and standard deviation, σ_θ. At the iteration s, the rescale is performed by generating new knots $\boldsymbol{X}^{*(s)} = \{X_1^{*(s)}, \ldots, X_K^{*(s)}\}$ by linearly transform the original quadrature points $\boldsymbol{X}^{(s)} = \{X_1^{(s)}, \ldots, X_K^{(s)}\}$ as follows,

$$X_k^{*(s)} = \frac{(X_k^{(s)} - (\boldsymbol{W}^{(s)'} \boldsymbol{X}^{(s)} - \mu_\theta))\sigma_\theta}{\sqrt{\boldsymbol{W}^{(s)'}(\boldsymbol{X}^{(s)} \odot \boldsymbol{X}^{(s)})}}, \quad k = 1, \ldots, K; \tag{3.6}$$

where $\boldsymbol{X} \odot \boldsymbol{X}$ is simply a vector containing the squares of the elements of $\boldsymbol{X}$. $\boldsymbol{W}^{(s)}$ comes from the EM algorithm as the optimum of (3.4). Using the new quadrature points in the next iterations is not advisable because they can alter the convergence of the algorithm and they can gradually translate outside a reasonable interval of variation of the ability. To overcome this issue the `R` package

`mirt` adopts the interpolation/extrapolation approach introduced by **woods2007** that linearly estimates new weights $\boldsymbol{W}^{*(s)}$ on the original quadrature points $\boldsymbol{X}^{(s)}$.

In this work, the spline interpolation is considered (see **de1978practical** for a gentle introduction and **meijering2002chronology** for a historical traceback) add applied to calibration problems by using the `Julia` package `Interpolations.jl`[4]. In particular, by this package, it is possible to choose between linear, quadratic, and cubic interpolation; on-grid and on cell interpolation objects and flat, line, free, periodic and reflect boundary conditions. After several trials, the **cubic-spline,** with on-grid interpolation and line boundary condition is selected because it had the combination which produced lower RMSEs of the estimates.

This step is performed only in the first iterations in order to avoid an undesired behaviour of the spline and a lack of convergence of the EM procedure, in particular the masses are adapted and rescaled in iterations $3, 6$ and 9.

3.1.4 Ability estimation

Once the $\hat{\boldsymbol{\xi}}_i^{(s)}$ have been obtained and the weights $\boldsymbol{W}^{(s)}$ have been updated, if no prior has been chosen, the $\hat{\theta}_n^{(s)}$ can be estimated by using the expected a posteriori (EAP, **BockMislevy1982**) or maximum a posteriori (MAP) method as follows:

$$\hat{\theta}_n^{(s)}{}_{EAP} = \sum_k p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \hat{\boldsymbol{\xi}}^{(s)})X_k,$$

$$\hat{\theta}_n^{(s)}{}_{MAP} = \{X_k : k = \arg\max_k p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \hat{\boldsymbol{\xi}}^{(s)})\}.$$

However, if a prior $p(\theta|\boldsymbol{\tau})$ is selected, the latter is transformed into:

$$\begin{aligned}\hat{\theta}_n^{(s)}{}_{MAP} &= \arg\max_\theta p(\theta|\boldsymbol{u}_n, \boldsymbol{\tau}, \hat{\boldsymbol{\xi}}^{(s)}) \\ &= \arg\max_\theta l(u_{i,n}|\theta, \boldsymbol{\xi}_i) + \log(p(\theta|\boldsymbol{\tau})).\end{aligned}$$

[4] https://github.com/JuliaMath/Interpolations.jl

The algorithm

We optimize the marginal likelihood by using the `NLopt.jl` package based on the `NLopt` suite, in particular, we use the `SLSQP` algorithm, which is fast and stable. The implemented EM sub-routine is described in the following pseudo-algorithm:

Algorithm 1 EM

Initialize $\boldsymbol{\xi}, \boldsymbol{\theta}$ to arbitrary values (e.g. $b_i^{(o)} = 0.0$, for all i)
Initialize $\boldsymbol{W}$ by the Normal$(0, \sigma_\theta)$ density.
s=1
while none of the stopping criterion is satisfied **do**
  E-step: Compute (3.2) $\rightarrow p(X_k|\boldsymbol{u}_n, \boldsymbol{\tau}, \boldsymbol{\xi}), \quad \forall n, k$
  M-Step: Maximise (3.3) by `Nlopt` $\forall i \rightarrow \hat{\boldsymbol{\xi}}^{(s)}$
  if (s==9 or s==12 or s==15) **then**
    Update $\boldsymbol{W}$: Compute (3.5) $\rightarrow \boldsymbol{W}^{*(s)}$
    Rescale $\boldsymbol{X}$: Compute (3.6) $\rightarrow \boldsymbol{X}^{*(s)}$
  end if
  Interpolate: Cubic-spline of $\boldsymbol{W}^{*(s)}$ on $\boldsymbol{X} \rightarrow \boldsymbol{W}^{(s)}$
  s+=1
end while

3.1.5 Bootstrap

The strength of IRT models, theoretically, is to have invariant item parameters across samples of examinees from the same population. Practically, invariance is hard to be guaranteed under the calibration process. Several factors may contribute to obtaining considerably different item parameter estimates from the same set of items under different conditions. Such factors include the positioning of items in the test, different populations, different points of time. In **tsutakawa1990effect** is showed that using estimates of item parameters instead of their actual values could lead to biases in the following inference about the students' ability. Also, **veldkamp2013application** pointed out that an item selection based on maximum information would capitalize on positive estimation errors if the uncertainty in the estimates of discrimination parameters is not taken into account.

Although existing methods for handling uncertainty in item parameters provide a variety of tools, most of them belong to Bayesian applications, which need

to know the prior distribution of item parameters and abilities. An alternative approach that simulates the calibration under different conditions and that does not ask to assume any probability distribution is the *bootstrap* (**efron1993**). In particular, we performed the calibration in a large number of resamples of the response data. The aim is to fully characterize the uncertainty related to each item parameter estimates by exploring its empirical distribution function (edf).

Given an item bank of items and a sample of students, from here called *full sample*, after the items have been administered, we have a matrix of dichotomous responses. For each item, we want to estimate a vector $\hat{\xi}_i$ of IRT parameters which may contain a different number of parameters in case we have a different model such as the 1PL, the 2PL or the 3PL. We first perform an *overall calibration* (on the full sample of students) by following the marginal maximum likelihood estimation method (MMLE) **bock1981marginal**, already presented previously. Once the overall estimates of item parameters and abilities of the students are obtained, we proceed with resampling eith replacement $N^* = N$ rows of the response matrix, R times, and, in each of these replications we reestimate the IRT item parameters. In this way we have R samples of each item parameter.

Depending on the way the responses are resampled we can distinguish two different algorithm: the parametric and the non-parametric schemes. They are described analitically in the next paragraphs.

Non-parametric bootstrap

Algorithm 2 Non-parametric bootstrapped calibration

Choose a large number of repetitions R, the subsample size $N^* = N$ and an ability point θ_0 in which maximize the TIFs.
for $r = 1 : R$ **do**
Sample with replacement N^* rows of the responses matrix assuming that each row has a $1/N$ probability to be drawn.
Calibrate the items in the subsample. Get $\hat{\xi}_r = \{\hat{\xi}_{1r}, \ldots, \hat{\xi}_{Ir}\}$.
Compute the IIF for each item for a chosen set of ability points θ_0 using $\hat{\xi}_{ir}$.
Store the vectors $\hat{\text{IIF}}(\theta_0)_r = \left\{\hat{\text{IIF}}(\theta_0)_{1r}, \ldots, \hat{\text{IIF}}(\theta_0)_{Ir}\right\}$
end for

Parametric bootstrap

Algorithm 3 Parametric bootstrapped calibration

Choose a large number of repetitions R, the subsample size $N^* = N$ and an ability point θ_0 in which maximize the TIFs.
Estimate $\hat{\boldsymbol{\xi}}$, $\hat{\boldsymbol{\theta}}$ and $\hat{\mathbf{W}}$ on the full sample.
Discretize the distribution of the ability by dividing its continuum in K bins and approximate the probability of sampling the n-th examinee by its relative frequency $\hat{p}_n$ of his $\hat{\theta}_n$ assuming that, in each bin, the students (and their abilities) are uniformely distributed.
for $r = 1 : R$ **do**
Resample with replacement from the discretized distribution, N^* rows from the responses matrix.
Calibrate the items in the subsample taking $\hat{\mathbf{W}}$ as fixed and equal to its overall estimate. Get $\hat{\xi}_r = \{\hat{\xi}_{1r}, \ldots, \hat{\xi}_{Ir}\}$.
Compute the IIF for each item for a chosen set of ability points θ_0 using $\hat{\xi}_{ir}$.
Store the vectors $\hat{\text{IIF}}(\theta_0)_r = \left\{\hat{\text{IIF}}(\theta_0)_{1r}, \ldots, \hat{\text{IIF}}(\theta_0)_{Ir}\right\}$
end for

3.2 Simulation study

In order to show the suitability of `Julia` as a programming language for calibration purposes we present a benchmark analysis between our application and other open-source frameworks, such as the `R` programming language. In particular, we focused on the `mirtR` package (**R**; **JSSv048i06**) because we believe it is the most reliable and fast open-source software for this task. Other open-source options available for items calibration were: the `R` package `ltm` (**JSSv017i05**) and `stan` (**JSSv076i01**) from a Bayesian perspective. The first, as far as we know, doesn't handle the missings coming from an unbalanced pre-test design, the second produced unstable results. Both were very slow compared to `mirt`.

All the tasks are performed using `Julia 1.2.0` and `R 3.6.1` and working on a desktop computer with the following features: Windows 10, Intel-core i5-4670 CPU and 16GB of ram.

Specifically, the comparison is made between `Julia` and two versions of the algorithm implemented in `mirt`. About `mirt`, the first version uses the default of the argument `dentype`, which rules the type of density that is used for the latent

trait parameters, i.e., the Gaussian density. The second version, named here `mirt EHW`, estimates the latent distribution with the empirical histogram and interpolation extrapolation method described in **woods2007**. The unidimensional 2PL model has been chosen for the simulation together with dichotomous observed response variables. Through the analysis of the simulation results, first, the accuracy of estimates, in terms of BIAS and RMSE, and computational performance are evaluated for the three softwares in all the cases defined in 3.2. Secondly, a non-parametric and parametric bootstrap is applied on the calibration process for the standard setup in order to characterize the uncertainty related to the estimates and to illustrate their sampling distributions.

3.2.1 Simulation settings (Framework for pre-testing)

Several simulations are performed. Each design is driven by a different combination of variables that may recreate real pre-test situations. The aim of this section is not only to describe the process in which we benchmarked the mentioned algorithms but also to give to potential users an efficient framework for developing a test assembly strategy, from the assembly of the pre-tests to the calibration of item parameters. In practice, we chose to fix the length of the test at 50 items, $I = 250$, and we varied the other variables as follows:

TABLE 3.1: Simulated distributions for item parameters and ability

b	Normal$(0, 1)$, Uniform$(-4, 4)$
a	LogNormal$(0, 0.5)$, Uniform$(0.001, 4)$
θ	Normal$(0, 1)$, LogNormal$(0, 0.25)$
N	600, 3000, 6000

The distributions of the item parameters and the ability have been chosen according to the literature (**glas2005modeling**; **glas2005testing**; **ban2002data**). We decided to include a case plausible in the real world, that is the case 2b where the abilities of the test-takers will be drawn from a LogNormal(0,0.5) and then changed of sign in order to increase the probability of having lower high-performing examinees.

Next table details all the examined cases:

TABLE 3.2: Simulation settings

#case	b	a	θ	N
1a	$N(0,1)$	LogN$(0,0.25)$	$N(0,1)$	3000
1b	$N(0,1)$	LogN$(0,0.25)$	$N(0,1)$	600
1c	$N(0,1)$	LogN$(0,0.25)$	$N(0,1)$	6000
2a	$N(0,1)$	LogN$(0,0.25)$	LogN$(0,0.5)$	3000
2b	$N(0,1)$	LogN$(0,0.25)$	$-$LogN$(0,0.5)$[5]	3000
3	$N(0,1)$	U$(0.001,4)$	$N(0,1)$	3000
4	U$(-4,4)$	LogN$(0,0.5)$	$N(0,1)$	3000

The case n. 1a will be deeply explored since it is a standard-setting for calibration. We will call it the *standard setup*.

Each step of the framework for the simulation study for the MML estimation is comprehensively described herewith:

1. A combination of variables from Table 3.1 and a metric are chosen. We decide to fix μ_θ to 0 and σ_θ to 1.

2. True values for $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$ are sampled from the respective distributions. The simulated pool $\boldsymbol{\xi}^\dagger$ and abilities $\boldsymbol{\theta}^\dagger$ are linearly rescaled to fit the predefined metric. These quantities represent the true values in the benchmarking phase.

3. To assemble the pre-test form, assuming no information is available about the IRT item parameters, we assign to each item an approximated easiness level, from 1 to 3. This information can be given by the experts of the domain of the item under examination. In this simulation the levels are approximated by cutting the sampled b_is by the 25th, 50th and 75th percentiles of the true distribution of the easiness.

4. $T = 6$ tests are assembled with the following features: 10 anchor items, length = 50, fixed difficulty (easiness) composition (25%, 50%, and 25%) approximated by the three levels previously assigned. The resulting pre-test design will be unbalanced. This means that the test forms are partially

overlapped (presence of anchor items) and hence the response data will contain missings. To help understanding the structure of the design we define two types of design matrices: an *items* $\times$ *tests* design matrix and a *tests* $\times$ *examinees* design matrix. The first assigns the items from the pool to the tests; this is the result of the pre-test assembly. The second assigns the test forms to the test-takers. The combination of these two designs generates the *items* $\times$ *examinees* design matrix. All these further configurations can be represented by a 0-1 matrix, respectively of sizes $I \times T$, $T \times N$ and $I \times N$. See figures A.1, A.2 and A.3 for a visual explanation.

5. For each of the $S = 100$ simulations, we use the *items* $\times$ *examinees* design to generate the responses. One vector, i_n, for each test-taker and one vector, n_i, for each item are produced. They represent, in the same order, the indices where the columns and the rows of the *items* $\times$ *examinees* design matrix are equal to 1. For example, if the examinee n takes the items 1, 3, 40 and 52, their vector i_n will be equal to $\{1, 3, 40, 52\}$. On the other side, if the item i is given to the examinees 100, 2540 and 351 its vector n_i will be equal to $\{100, 2540, 351\}$. These vectors are intensively used in the algorithm to filter the response matrix $\mathbf{U}$. The data matrices $\mathbf{U}_s$ will be generated by sampling, for each student n, a correct response to the item i using a Bernoulli with a probability equal to

$$P_{i,n} = \frac{1}{1 + \exp -(\Psi(\boldsymbol{\xi}_i^{\dagger} | \theta_n^{\dagger}))}$$

6. The calibration process is run for each of the S simulations $\mathbf{U}_s$, for $s = 1, \ldots, S$ by the EM algorithm 1 and by the `mirt` package using the following settings:

   - Bounds for a and b are $\{0, 5\}$ and $\{-6, 6\}$[6].
   - Bounds for θ are $\{-6, 6\}$. The quadrature is done on $K = 61$ knots.
   - Tolerance for the item parameters is $1e^{-4}$. Tolerance for the likelihood is $1e^{-8}$.
   - The maximum number of iteration is set to 500 and the time limit to 1000 seconds.

[6] `mirt` default solver which allows box bounds is `minlp`

7. The elapsed computational time and the RMSEs and BIASs for the estimated item parameters $\hat{\boldsymbol{\xi}}_s$ and individual abilities $\hat{\theta}_n$ across the S simulations of response data are compared.

   Let $x_j^\dagger$ be the true value of the variable x_j where x can be an item parameter or the ability and the index j may indicate a particular item or the test-taker Moreover, let $\hat{x}_{js}$ the estimates of the same variable in the s-th simulation, we define the RMSE and BIAS of the object i as:

$$RMSE_j = \sqrt{\frac{1}{S}\sum_{s=1}^{S}\left(\hat{x}_{js} - x_j^\dagger\right)^2} \tag{3.7}$$

$$BIAS_j = \frac{1}{S}\sum_{s=1}^{S}\left(\hat{x}_{js} - x_j^\dagger\right) \tag{3.8}$$

3.2.2 Results

In the next sections, we illustrate the results of the calibration for the six cases under analysis calibrated using the three mentioned softwares `Julia` *(jl)*, `mirt` with default settings (*mirt*) and `mirt` with the Woods' empirical histogram (*mirtEHW*).

First, the estimation accuracy of the algorithms is visually compared by analyzing the boxplotsof the RMSEs and BIASs, Figures A.4 and, A.8, for the estimates of item parameters and abilities. In order to understand if the algorithms have a different behaviour at specific intervals of the domains we examinate also the scatter plots of the RMSEs and BIASs with respect to the true values, accordingly in Figures A.6 and A.7. For the sake of brevity, in this chapter, only the complete results for the *standard setup*, i.e. case 1a, are presented. On the other hand, the overall precision of each algorithms in all the cases is summarized in Tables 3.3 and 3.4, where the RMSEs and BIASs are averaged across the item pool, for the item parameters, and across test-takers, for the abilities. For the plots of the other cases (from 1b to 4) please refer to Appendix A. Together with the precision of estimates, we intend to evaluate the computational performance of our implementation by comparing the elapsed time and the number of iterations observed for calibrating the items for all the algorithms. Secondly, the estimated $\hat{\mathbf{W}}^{(s)}$, where $s = 1, \ldots, S$, are plotted against the true $\mathbf{W}$ obtained

by drawing 1.000.000 samples from the true distribution and computing the relative frequencies in the K chosen $\mathbf{X}$ points. In the last section, some examples of sampling distribution of the item parameters obtained by bootstrapping the calibration is reported.

Summary of estimation accuracy

In the following tables the RMSEs and BIASs averaged across the item pool for discrimination and easiness parameters and across the test-takers for the latent variable are presented for all the cases under analysis. The lowest RMSEs and the BIASs most proximal to zero, i.e., the best achieved, are formatted in bold font. In the case of Normality of the latent trait (cases 1a, 1b and 1c) we can notice that the precision of the algorithms is almost the same for all the estimates except the `mirt`$_{\text{EHW}}$ which does not behave well in case of few observation, that is the case 1b in which $N = 600$. On the other hand, when it comes to have a non-Normal latent trait (case 2a and 2b) our algorithm has the best performance notably for the discrimination parameter in which the RMSEs are 0.01 lower than the mirt package. No remarkable differences are observed in cases 3 and 4 where the distributions of the item parameters are different from the usual setting. A substantial discrepancy is observed analysing the BIASs where our approach has a better overall performance.

TABLE 3.3: RMSEs averaged across the item pool ($\hat{a}$ and $\hat{b}$) and test-takers ($\hat{\theta}$)

	$\hat{b}$			$\hat{a}$			$\hat{\theta}$		
case	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$
1a	**0.114**	**0.114**	**0.114**	**0.134**	0.135	0.135	**0.311**	**0.311**	**0.311**
1b	0.264	**0.263**	0.877	0.339	**0.336**	0.341	0.320	**0.317**	0.799
1c	**0.779**	**0.779**	0.780	**0.094**	**0.094**	**0.094**	**0.310**	**0.310**	**0.310**
2a	**0.111**	0.117	**0.111**	**0.152**	0.166	0.154	**0.294**	0.333	0.315
2b	**0.114**	0.119	6.076	**0.149**	0.159	0.232	**0.302**	0.331	5.278
3	**0.138**	**0.138**	**0.138**	**0.214**	0.221	0.227	0.221	**0.220**	**0.220**
4	**0.188**	0.189	0.190	**0.186**	**0.186**	0.187	**0.372**	**0.372**	**0.372**

TABLE 3.4: BIASs averaged across the item pool ($\hat{a}$ and $\hat{b}$) and test-takers ($\hat{\theta}$)

	$\hat{b}$			$\hat{a}$			$\hat{\theta}$		
case	jl	mirt	mirtEHW	jl	mirt	mirtEHW	jl	mirt	mirtEHW
1a	**0.002**	0.004	**0.002**	**0.009**	0.011	0.010	2.8×10^{-4}	-0.002	$\mathbf{2.0 \times 10^{-4}}$
1b	**0.005**	0.006	0.120	0.060	**0.059**	0.642	**-0.002**	-0.004	-0.105
1c	**0.001**	0.002	**0.001**	0.005	**0.004**	0.005	-1.6×10^{-4}	-1.1×10^{-3}	$\mathbf{1.5 \times 10^{-4}}$
2a	**0.004**	-0.042	0.005	**0.005**	-0.091	0.118	$\mathbf{4.8 \times 10^{-4}}$	-0.002	-0.028
2b	**-0.003**	0.040	6.073	**-0.009**	-0.072	0.006	$\mathbf{9 \times 10^{-4}}$	-0.002	-5.268
3	**0.001**	0.010	**0.001**	**-0.013**	0.033	0.044	$\mathbf{3.9 \times 10^{-4}}$	-0.003	9.2×10^{-4}
4	**0.001**	0.006	**0.001**	**0.013**	**0.013**	0.014	$\mathbf{-7.1 \times 10^{-5}}$	-1.3×10^{-3}	2.2×10^{-4}

The *standard setup*

Cubic-spline interpolation/extrapolation

Computational performance

To compare the performance of the three considered algorithms, we evaluate their CPU times, and the iteration counts averaged across the S simulations. The results for the cases under analysis are summarized in the following table:

TABLE 3.5: Computational power summary

	average CPU time			average iterations count		
	jl	mirt	mirtEHW	jl	mirt	mirtEHW
1a	**5.75**	11.90	17.27	88.88	55.12	59.99
1b	**2.55**	9.137	16.75	109.54	51.31	69.12
1c	**10.58**	14.69	20.50	92.73	56.42	59.31
2a	**7.75**	12.32	21.39	105.85	47.09	65.43
2b	**5.77**	9.47	97.42	97.35	49.87	169.73
3	**12.81**	24.81	37.54	222.28	140.44	151.14
4	**3.59**	12.98	16.45	58.17	43.86	53.61

For the standard setup `Julia` had a better time performance. `Julia` is always

faster than the `mirt EHW` algorithm. Furthermore, we think that our code is further optimizable by introducing more parallel or distributed computing `Julia`'s features[7].

Sampling distribution of IRT item parameters

[7] See https://docs.julialang.org/en/v1/manual/parallel-computing/index.html

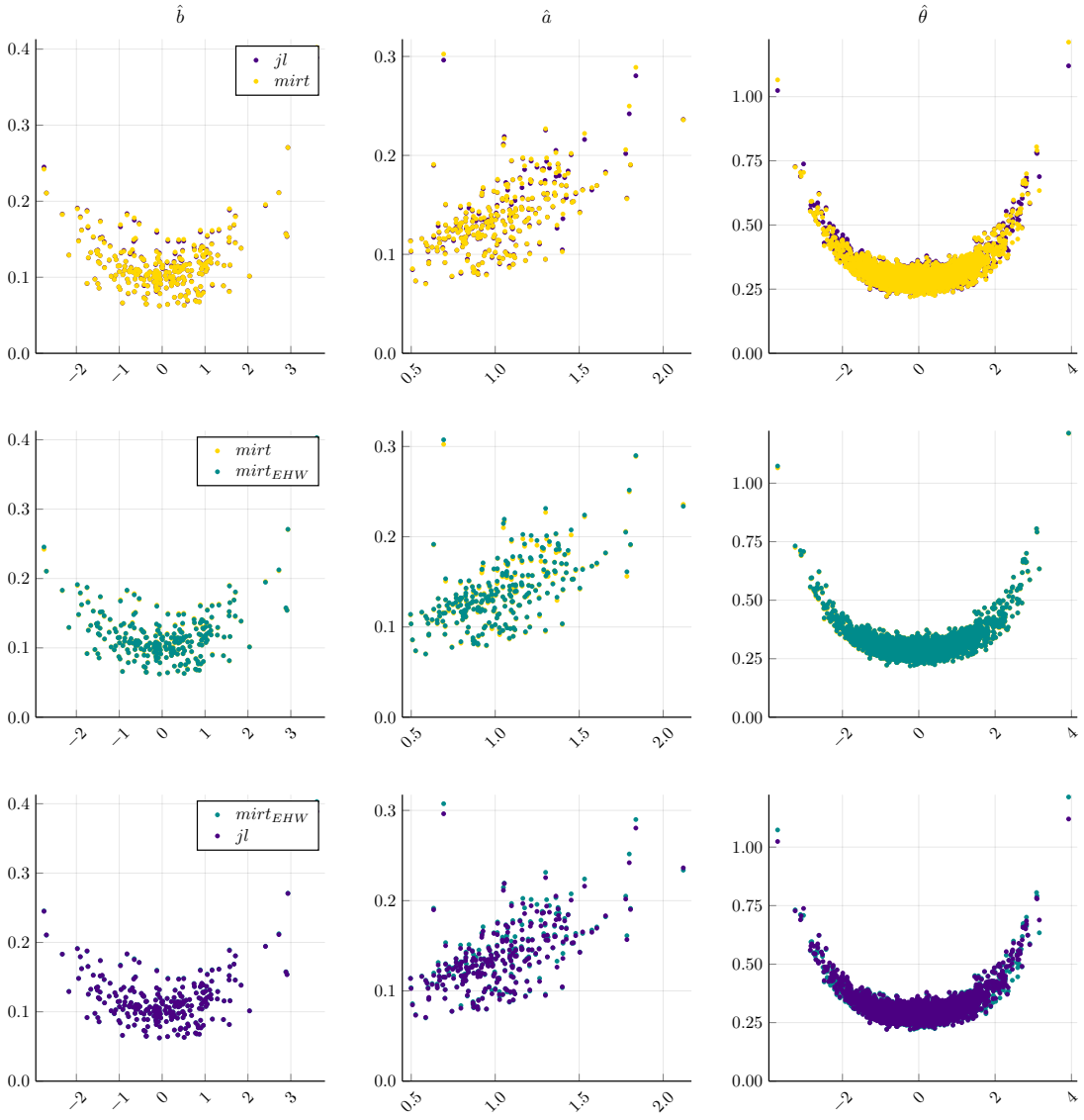


FIGURE 3.1: Case 1a - Scatter plots of RMSEs.

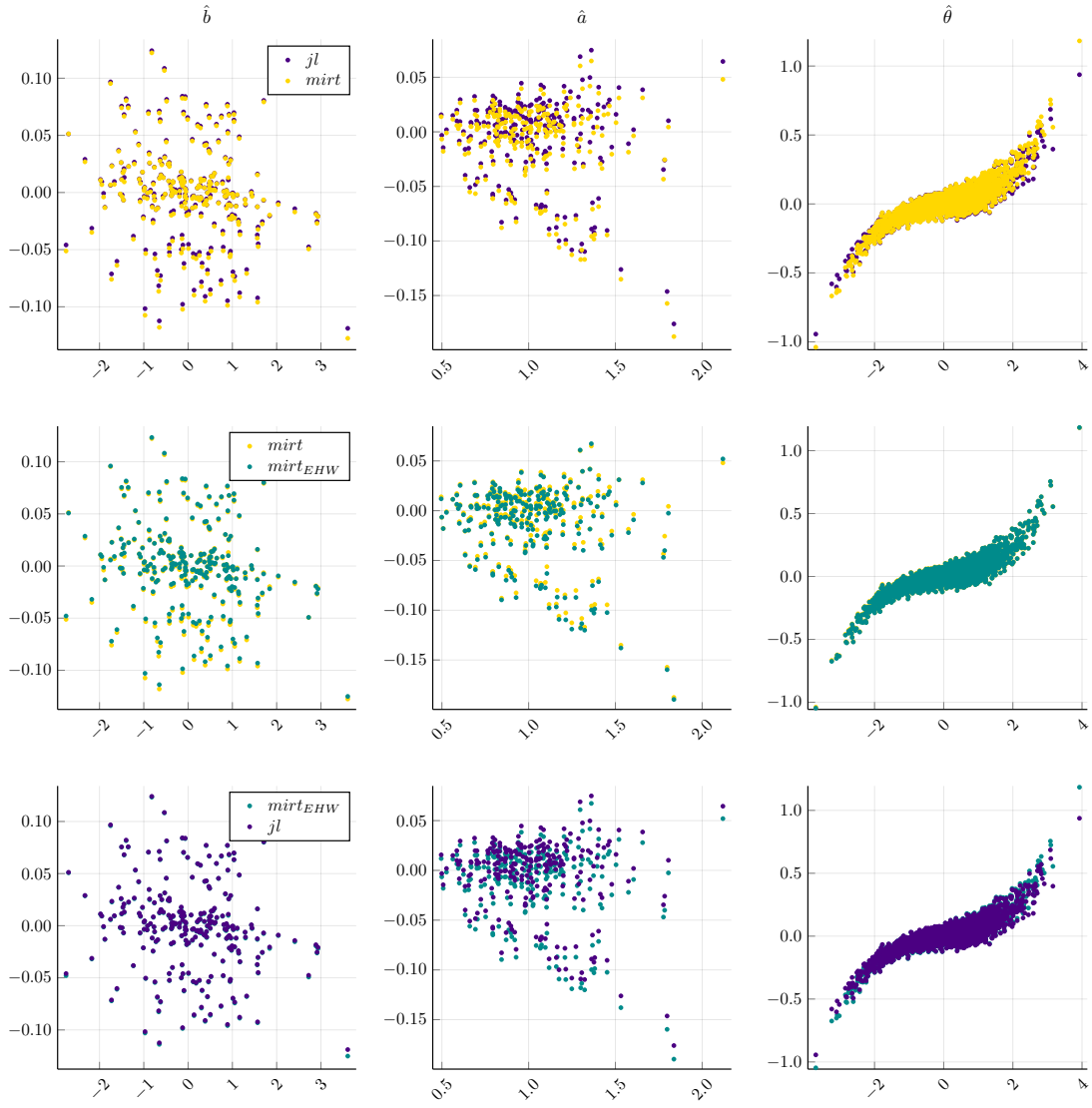
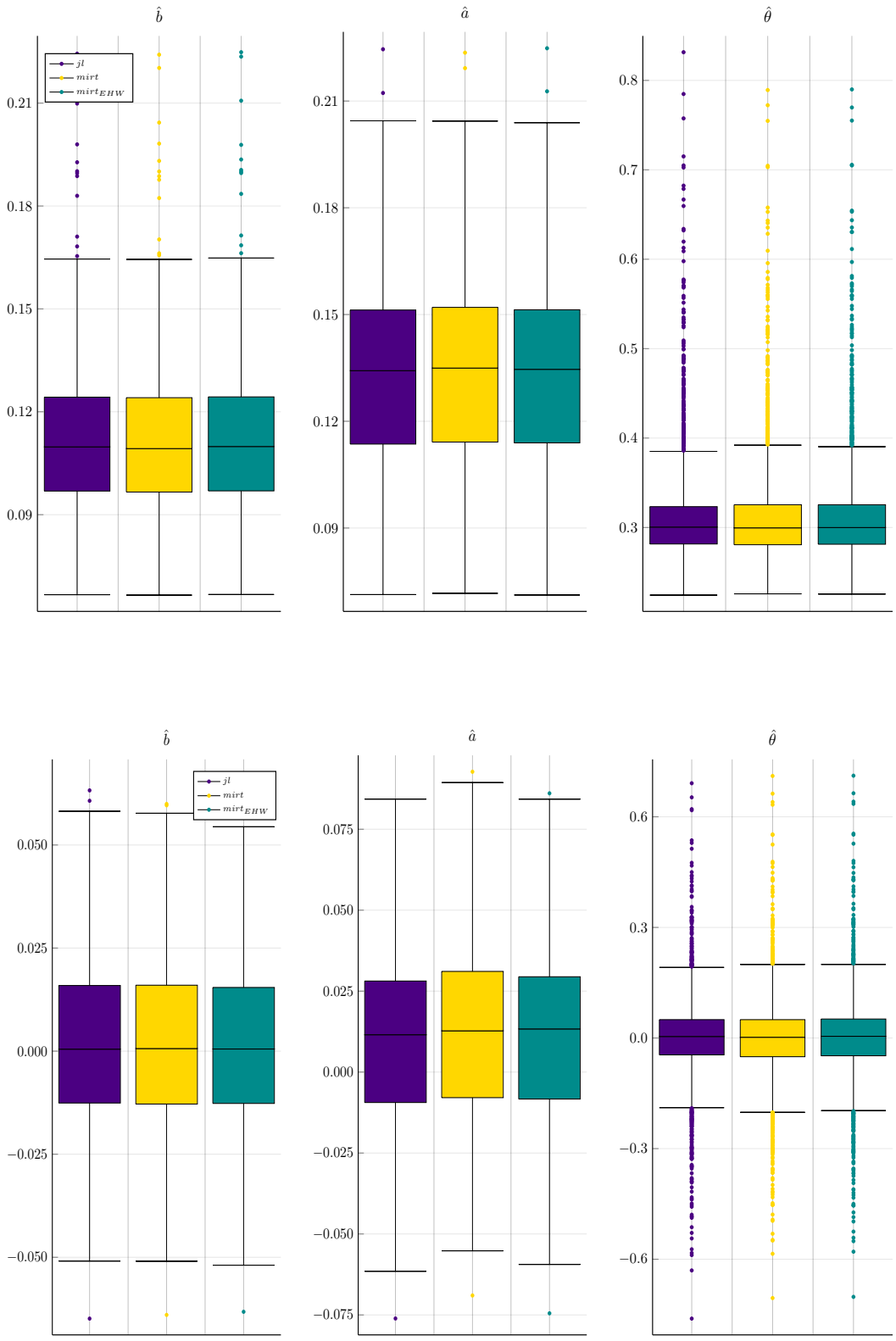


FIGURE 3.2: Case 1a - Scatter plots of BIASs


$\hat{b}$
$\hat{a}$
$\hat{\theta}$
jl
mirt
$mirt_{EHW}$
0.21
0.18
0.15
0.12
0.09
0.8
0.7
0.6
0.5
0.4
0.3
0.050
0.025
0.000
−0.025
−0.050
0.075
−0.075
0.0
−0.3
−0.6

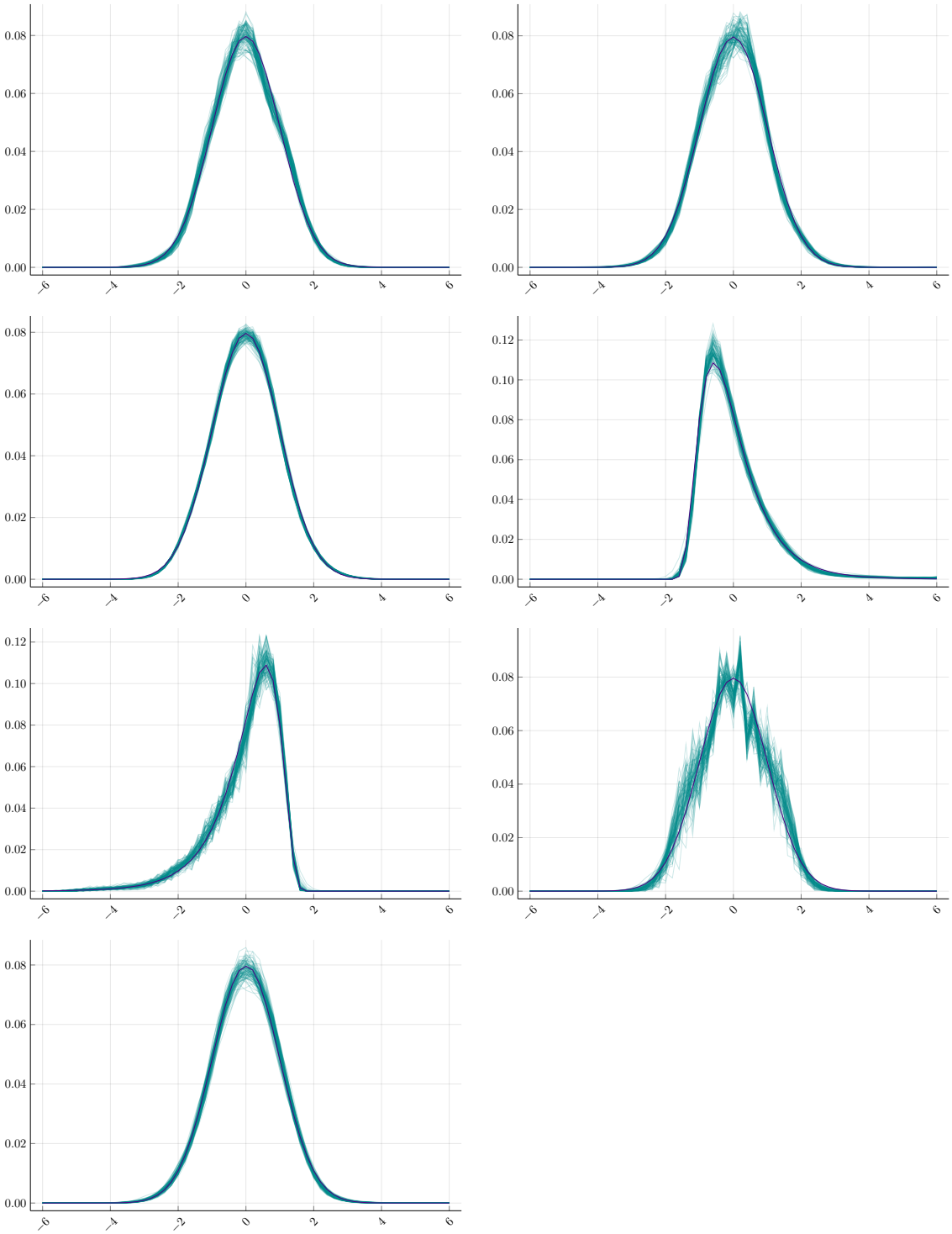


FIGURE 3.3: Retrieval of the latent probability distribution by cubic-spline

4 Chance-constrained test assembly

The test information function (TIF) is a key object both in the IRT and in the test assembly framework. Most of the automated test assembly models (ATA) are based on this quantity that usually appears in the objective function, being the goal for the optimization model. As explained in Section 2 the IIFs are considered as given values. This approach may lead to several issues such as infeasibility of the MINIMAX or MAXIMIN model, e.g. if it is not possible to find T parallel tests that have TIFs inside a fixed interval around the targets. Another issue is the incorrect interpretation of the assembly results. For example, if the calibration algorithm had produced wrong estimates for the item parameters and hence the item information functions are not accurate enough, the TIF of the assembled test might be overestimated. Regarding the latter issue, a good test assembly model would consider the variation of the item parameter estimates in order to build test forms in a conservative fashion, i.e., it would produce tests with a maximum plausible lower bound of the TIF.

There is a need for better treatment of this problem in test assembly models. My attempt in this dissertation is to incorporate uncertainty in the optimization models most seen in practice and in literature for simultaneous multiple test assembly using the modern techniques of the stochastic programming framework. Chance-constraints (or probabilistic constraints) are a natural solution to the mentioned problems. They are among the first extensions proposed in the stochastic programming framework to deal with constraints where some of the coefficients are uncertain (**charnes1963deterministic**; **krokhmal2002portfolio**). In particular, by adjusting a conservative parameter α, also called *risk level*, it is possible to modulate the level of fulfilment of some probabilized constraints

enabling the user to relax or to tighten the feasibility of the problem. Narrowing our focus on the MAXIMIN test assembly model introduced in 2.2, a percentile optimization model would maximize a reasonable lower bound of the TIF, its α-quantile, approximated by the $\lceil \alpha R \rceil$-th ranked value of the TIF computed on the R bootstrap replications of the estimates of item parameters.

An introduction to the idea of chance-constrained modeling is provided in the first Section together with a brief literature review of the issues and of the existing methods to solve this type of problems. Subsequently, a chance-constrained version of the MAXIMIN test assembly model is proposed and, since this novel model cannot be approximated by a linear formulation, a heuristic based on simulated annealing (**goffe1996simann**) has been developed. This technique can handle large-scale models and non-linear functions. A Lagrangian relaxation formulation helps to find the most feasible/optimal solution and, thanks to a random variable, more than one neighborhood of the space is explored avoiding to being trapped in a local optimum. Moreover, the proposed heuristic can solve a wide class of optimization problems characterized by having binary optimization variables and a separable objective function. Furthermore, the details of the results of the retrieval of the empirical distribution function of the TIF are provided in Section 4.2.1.

Several simulations of ATA problems are performed and the solutions are compared to CPLEX 12.8.0 Optimizer, a benchmark solver in the linear programming field. In particular, since our heuristic is able also to solve the classical ATA models we compare the results of the optimization in both the framework: exact and chance-constrained. The described algorithms are coded in the open-source framework Julia. A package written in Julia has been released [1].

4.1 Chance-constrained modelling

In the past five decades, the developments in the theory of choice under risk in financial applications, e.g. portfolio optimization where the prices of instruments are random variables (see **rockafellar2000optimization**; **rockafellar2001uryasev**), followed the expected mean-variance approach (**chen1973quadratic**; **freund1956introduction**;

[1] http:\github.com\giadasp\ATA.jl

scott1972practical)).In risk management and in reliability applications, the decision maker must select a combination of assets for building a portfolio by maximizing their utility function. The latter is defined in terms of the expected mean and variance of the returns or of the prices of the instruments which are uncertain coefficients in the linear objective or constraints of the optimization model. More recently, instead, the regulations for finance businesses require to reformulate the problem in terms of percentiles of loss distributions. These requirements gave rise to the theory of *chance-constraints*, also called probabilistic constraints, originally proposed by **Charnes1959**.

The probabilistic constraints present coefficients which are assumed to be randomly distributed and they are subject to some predetermined threshold α of the constraints fulfilment. Modifying α it is possible to relax or tighten some constraints modulating the level of conservativeness of the model. The standard form of a mixed-integer optimization problem can be represented by

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0 \quad j = 1, \ldots, J \end{aligned} \tag{4.1}$$

$$\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$$

where $f(\cdot)$ is the objective function to be optimized, $\mathbf{x}$ is the vector of p integer and $n-p$ continuous optimization variables. Both $f(\cdot)$ and $g(\cdot)$ are scalar functions.
The optimization domain is $D = \text{dom}(f) \cap \bigcap_{j=1}^{J} \text{dom}(g_j)$ and the set
$\mathbf{X} = \{\mathbf{x} : \mathbf{x} \in D, g_j(\mathbf{x}) \leq 0 \ \forall j\}$ is called *feasible set*, i.e. a solution $\mathbf{x}$ is feasible if it is in the optimization domain and satisfies the constraints. Starting from (4.1), a chance-constraints reformulation will add the following set of constraints:

$$\mathbb{P}\left[g_k(\mathbf{x}, \boldsymbol{\xi}) \leq 0\right] \geq 1 - \alpha \quad k = 1, \ldots, K \tag{4.2}$$

where $\boldsymbol{\xi}$ is a vector of random variables. This formulation seeks a decision vector $\mathbf{x}$ that minimizes the function $f(\mathbf{x})$ while satisfying the chance constraints $g_k(\mathbf{x}, \boldsymbol{\xi}) \leq 0$ with probability at least $1 - \alpha$. Such constraints imply having a

function to compute (or better approximate) the probability and a solver which can deal with that function. Whenever a MAXIMIN principle is applied, they can be seen as *percentile optimization* problems (**krokhmal2002portfolio**) because the probability in (4.2) is replaced by the α-th percentile of the distribution function of $g_k(\mathbf{x}, \boldsymbol{\xi})$ and these percentiles must be maximized.

Despite the old age, chance-constrained models are, still hard to be solved. An issue is the general non-convexity of the probabilistic constraints. Even if the original deterministic constraints[2] were convex the respective chance-constraints may be non-convex. In general they are usually untractable (see **Nemirovski**) because even if they are convex the quantiles of the random variables are difficult or impossible to compute. Examples of approximations of chance-constraints are the linearization method called sample average approximation (**Ahmed2008**) and the case when the random variables follow a known multivariate distribution with known mean and variance. For the first case, a big-M approach is needed to deal with the indicator function bringing numerical instability in the optimization. The second approach instead imposes strong distributional assumptions (see **Kataria** for a list of distributional assumptions) and, since they are based on the Chebishev inequality they require a modest number of elements in the summations to achieve the convergence, they need also a solver which can deal with second-order conic constraints, the most difficult type of convex functions to be optimized. All the mentioned formulations increase exponentially the number of optimization variables, thus they are not suitable for large-scaled models.

Other approaches rely on discretization of the random variable and hence the model is optimized in all possible scenarios (i.e. realizations of the random variables) thus they do not fit to problems with a large number of random variables because all the patterns must be considered (**margellos2014road**; **wang2011chance**; **tarim2006**). In finance, such models are called VaR (value at risk) and they are usually characterized by non-concavity and hence computational intractability except in certain cases where returns are known to have an elliptical distribution, see for example **vehvi2003** or **mcneil2005**.

Another question is the domain of optimization. Usually, stochastic optimization models are addressed in the case of continuous optimization variables while mixed-integer problems are still neglected because of their greater complexity.

[2] $g_k(\mathbf{x}, \boldsymbol{\xi})$ where $\boldsymbol{\xi}$ is not random.

Given a lack in optimization techniques which can handle such problems, we first use a Monte-Carlo approach to approximate the quantiles in a percentile optimization perspective and, in Section 4.2.2, we propose an heuristic to solve the chance-constrained test assembly model defined previously. In the last Section a simulation study is conducted to show the practical and computational advantage of our approach in the test assembly research field.

4.2 Chance-constrained test assembly model

In the context of test assembly the optimization models used for selecting the items does not consider the inaccuracy of the estimates of item parameters (**VDL2005**). However, estimates are never exact. Thus, ignoring the potential imprecision can lead to wrong conclusions and misinterpretations of the results such as overestimation of the information function and hence of the accuracy of the test in ability estimation. Some attempts to include uncertainty in the test assembly models have been done by **veldkamp2013application** and **veldkamp2013uncertainties** who developed and applied the robust model introduced in **bertsimas2003robust**. The mentioned automated test assembly model considers the standard error of the estimates and a protection level Γ that indicates how many items in the model are assumed to be changed in order to affect the solution. It treats the uncertainty in a deterministic way and, given Γ, it adjust the solution adopting the most conservative approach, because standard errors are the maximum expression of uncertainty of the estimates.

In contrast, if we consider the MAXIMIN model (2.14) its chance-constrained equivalent would replace the constraints (2.14b) involved in the maximization of the TIF by

$$\mathbb{P}\left[\sum_{i=1}^{I} I_i(\theta_{k_t}) x_{it} \geq y\right] \geq 1-\alpha, \ \forall t, k_t, \tag{4.3}$$

where $t = 1, \ldots, T$ are the test to be assembled and θ_{k_t} are the ability points in which the TIF of the test t must be maximized. We decided to ignore the weights to simplify the notation but the extension to the weighted case is straightforward. We will call the model (4.3) *chance-constrained MAXIMIN*, or CCMAXIMIN. The key element of this model is, again, the information function which is assumed

to be random. This assumption arises, as already explained, by the necessity of taking into account the uncertainty of the item parameter estimates, of which the item information function is a statistic (see (2.6) for an example).

The CCMAXIMIN model allows to maximize the expected precision of the assembled tests in estimating the latent trait of the test-takers at pre-determined ability points with a high confidence level if the α is chosen to be next to zero. In terms of probability we can say that the constraints in (2.14b) must be fulfilled with a probability at least $1 - \alpha$. Adjusting the confidence level it is possible to relax or tighten the fulfilment of the chance-constraints setting a specific conservative attitude, i.e. a small α means an high level of conservatism, on the contrary a big α means an almost relaxation of the constraints. This is the novelty of the CCMAXIMIN model with respect to the robust model proposed in **veldkamp2013application**; **veldkamp2013uncertainties** which, instead, perform a worst-case optimization.

Once the chance-constraints have been defined, a way to evaluate the probability in (4.3) must be found in order to quantify the feasibility of a solution. To solve this problem, some methods rely on assumptions on the probability distribution of $\boldsymbol{\xi}$, such as the multivariate normal (**kim1990deterministic**). Others try to approximate the probability using samples of the random variable obtained by a Monte Carlo simulation (**Ahmed2008**) which is a specific case of a scenario generation where all the scenarios have the same probability of occurrence. We decided to use the Monte Carlo method because of its flexibility and adaptability to our problem.

In particular, our random variable is the TIF of a test form, that is a statistic on some estimates which are uncertain. There are different ways to sample from the distribution function of this random variable: given the standard errors of the estimates, the samples can be uniformily drawn from their confidence intervals[3]; otherwise, if a Bayesian estimation is carried on, the last samples in the Markov chain can be used. In this dissertation the samples are picked by bootstrapping the estimation process and the empirical distribution function of the statistic is obtained. The bootstrap (**efron1993**) is a very powerful algorithm to extract information about the distribution of some estimate, provided that the method of resampling is accurate enough to reproduce the underlying data generation

[3] as in the robust model (**veldkamp2013application**)

process. The details of the retrieval of the empirical distribution function of the TIF are reported in the following section.

4.2.1 Empirical measure of the TIF

Conventionally, the estimates of IRT item parameters are considered as known values in test assembly models (**VDL2005**). Test assembly models ignore the uncertainty related to the calibrated items and thus they often yield an overstated measurement accuracy of the assembled tests in terms of their TIFs. A standard approach to extract the uncertainty related to the estimates of the item parameters would be first, sampling a high number of plausible values of the item parameters $\boldsymbol{\xi}_i$ in the confidence intervals built using the standard errors of the estimates and, secondly, computing the related IIFs at target θ points. This may be an optimal starting point to assemble robust tests, (**veldkamp2013uncertainties**; **veldkamp2013application**) but it has its own downsides because a uniform interval of plausible values is assumed. Another attempt to account for the influence of sampling error in the Bayesian framework has been made by **Yang2012** who proposed a multiple-imputation approach with the aim to better measure the latent variable of a respondent.

In **matteucci2012prior** it has been shown that the behavior of the estimates of item parameters usually follows joint densities not equal to the product of their marginals, suggesting the presence of an underlying dependence structure. This observation motivated the search for a new technique to recreate the distribution function of the IIFs and hence iof the TIF. Our solution is based on bootstrapping the calibration process (see **efron1993** for a gentle introduction to the bootstrap), in particular, the observed vectors of responses (one vector for each test-taker) are resampled with replacement R times and the item parameters are re-estimated for each sample. In this way, it is possible to preserve the natural relationship between the items and, given the ability targets, it is possible to compute their IIFs. After that, given a set of items, we can build a test form and compute its TIF for each of the R replications. The resulting sample constitutes the *empirical distribution function* of the TIF.

More formally, let $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_R$ be an independent identically distributed (iid)

sample of R realizations of a I-dimensional random vector $\boldsymbol{\xi}$, its respective empirical measure is

$$\hat{F}_R := R^{-1} \sum_{r=1}^{R} \Delta\boldsymbol{\xi}_r,$$

where $\Delta\boldsymbol{\xi}_r$ denotes the mass at point $\boldsymbol{\xi}_r$[4]. Hence $\hat{F}_R$ is a discrete measure assigning probability $1/R$ to each sample. In this way we can approximate the probability in the left-hand side of (4.2) by replacing the true cumulative distribution function of $\boldsymbol{\xi}$ by $\hat{F}_R$. Let $\mathbf{1}_{(-\infty,0]}\{x\} : \mathbb{R} \to \mathbb{R}$ be the indicator function of x in the interval $(-\infty, 0]$, i.e.,

$$\mathbf{1}_{(-\infty,0]}\{x\} = \begin{cases} 0, & \text{if } x > 0 \\ 1, & \text{if } x \leq 0, \end{cases}$$

Thus, given a specific chance-constraint k, a known set of optimization variables $\mathbf{x}$ and a sample $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_R$ of our random vector, we can rewrite

$$\begin{aligned} \mathbb{P}\left[g_k(\mathbf{x}, \boldsymbol{\xi}) \leq 0\right] =& \mathbb{E}_F\left[\mathbf{1}_{(-\infty,0]}\{g_k(\mathbf{x}, \boldsymbol{\xi})\}\right] && (4.4) \\ \approx& \mathbb{E}_{\hat{F}_R}\left[\mathbf{1}_{(-\infty,0]}\{g_k(\mathbf{x}, \boldsymbol{\xi})\}\right] && (4.5) \\ =& \frac{1}{R}\sum_{r=1}^{R} \mathbf{1}_{(-\infty,0]}\{g_k(\mathbf{x}, \boldsymbol{\xi}_r)\}. && (4.6) \end{aligned}$$

That is, the chance-constraint is evaluated by the proportion of realizations with $g_k(\mathbf{x}, \boldsymbol{\xi}) \leq 0$ in the iid sample.

Adopting the same principle to the left-hand side of the chance-constraints in (4.3), the CCMAXIMIN model can be approximated by:

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & -y && (4.7) \\ \text{subject to} \quad & \frac{1}{R}\sum_{r=1}^{R} \mathbf{1}_{[y,\infty)}\{\mathbf{I}_r(\theta_{k_t})'\mathbf{x}_t\} \geq 1-\alpha, \quad \forall t, k_t, \\ & g_j(\mathbf{x}_t) \leq 0 \qquad\qquad\qquad\qquad \forall j, t, \\ & \mathbf{x}_t \in \{0,1\}^I, y \in \mathbb{R}^+, \end{aligned}$$

where $\mathbf{I}_r(\theta_{k_t})$ is the vector of the I item information functions at pre-defined θ_{k_t}

[4] $\Delta\boldsymbol{\xi}_r(A) = 1$ when $\boldsymbol{\xi}_r \in A$

points computed from the estimates of the item parameters in the r-th bootstrap replication.

The model (4.7) is clearly non-convex because of the chance-constraints (see, **rockafellar2000optimization**; **rockafellar2001uryasev** for the demonstration) and most of the commercial solver doesn't deal with indicator functions, to overcome this issues we solved the previous model by an heuristic described in the next Section.

To have an idea of the empirical distribution function of the information function of an item with true values of discrimination and easiness parameters, $a = 1$ and $b = 0$, estimated by bootstrapping the calibration of a 2-parameter logistic model following the algorithm described in 3, the histograms for several values of the latent trait are reported herewith:

4.2.2 Solving the CCMAXIMIN model

Since the model (4.7) is not practically solvable by commercial solvers, we devoloped a heuristic based on the *simulated annealing* approach. It is a flexible and simple numerical procedure that can be used to find an optimal solution for a model of arbitrary complexity which seeks the minimal of a function, $f(\mathbf{x})$, called *loss*. The loss function serves as a distilled form of the greater problem and it depends on the values of some fixed coefficients and optimization variables $\mathbf{x}$, a vector d dimensional. Changing the value of the objective variables the returned loss function will increase, decrease or remain constant telling if the variation is useful or not to reach a minimum (preferrably global). Once the loss function is determined and it is evaluable for each value of the optimization variables the simulated annealing algorithm can be applied leading to the best configuration of $\mathbf{x}$ which minimizes the loss. In practice, an inital value of $\mathbf{x}$, namely $\mathbf{x}_0$, is chosen and a forward pass to evaluate $f(\mathbf{x}_0)$ is performed. At each successive step, $s > 0$, the current $\mathbf{x}_s$ is a perturbation of $\mathbf{x}_{s-1}$ in the sense that one of more elements are changed in order to explore another neighbourhood of the solution space.

The movement from a neighbourhood to another will be called *journey* and how it is performed depends on the problem under inspection and on how far we want to travel from the last accepted solution, called incumbent. If the loss for

the perturbed $\mathbf{x}_s$ is more or equally optimal than the previous, then $\mathbf{x}_s$ is accepted as a basis for the next iterations. However, if the solution is less optimal (e.g. the loss increase), the choice of whether discard or keep it depends on the value of a sample of a random variable. The random variable is built considering the amount of variation of the loss function induced by the journey and the state of the cooling schedule defined by the temperature $T(s)$ deterministically determined. Here the Metropolis Hastings algorithm appears and plays an important role in defining the convergence properties of the heuristic. The details are provided in the next paragraph.

Simulated Annealing

The principle of cooling schedule comes from the language used to describe mechanical processes of metal annealing, which involves heating a metallic object to a very high temperature and gradually cooling it. By letting the metal cool down, the particles (the optimization variables) arrange themselves into the lowest possible energy state (evaluated loss function). The atoms are allowed to move to further areas of the space (neighbourhoods) at hot temperatures than at low temperatures, this avoid to be stuck in local minima in the first phases of the annealing. After the minimum temperature has been reached a reannealing can be performed to explore other areas of the space. This allows to have an arbitrary number of non-unique solutions to compare and select. The system temperature, $T(s)$, is a non-increasing function with respect to the iteration count. It has been proved that, if an infinite number of iterations is made, the algorithm will reach the global minimum (**belisle1992**). Since the infinite assumption cannot be fulfilled in practice, a reasonable number of iterations is chosen, usually depending on the maximum allowed computational time.

The random variable which rules the acceptance/rejection step comes from the normalized Boltzmann factor

$$\mathbb{P}[E] = \frac{1}{z(T)} e^{\frac{-E}{kT}} \tag{4.8}$$

which determines the probability of observing a particular energy E given a termperature T, a normalizing factor $z(T)$ and a Boltzmann constant k. In

practice, if at the iteration s we observe $f(\mathbf{x}_s)$ and this is higher than $f(\mathbf{x}_{s-1})$ the probability of keeping $\mathbf{x}_s$ is equal to the probability of the variation, Δf_s, in the loss function:

$$\mathbb{P}\left[\Delta f_s\right] = \frac{e^{\frac{-f(\mathbf{x}_s)}{kT(s)}}}{e^{\frac{-f(\mathbf{x}_{s-1})}{kT(s)}}} = e^{\frac{-\Delta f(\mathbf{x}_s)}{kT(s)}}. \tag{4.9}$$

If the variation in energy is large the probability that the parameters will be kept is low, while for a small variation they might be accepted. In this way, the algorithm allows to escape from local minima, increasing the chance that the global minimum will be found. The actual choice is made by comparing the value given in (4.9) to a random variate from the uniform distribution. If the random value is smaller, the parameters are kept. As time goes on and the system temperature drops, however, the probability of keeping the state approaches zero, even for small changes in energy.

Heuristic

Adopting the simulated annealing algorithm it is possible to solve all the test assembly models which take the form (2.18). The heuristic we developed is inspired by the work of **Stocking1993** because the constraints in the optimization model are treated as part of the loss function using the hinge function and more in general, through the Lagrange relaxation, two main concepts introduced in 2.2.5. The algorithm is based on the separation of the problem, in particular we differentiate the T vectors $\mathbf{x}_1, \ldots, \mathbf{x}_T$ of I binary variables, each vector $\mathbf{x}_t$ corresponds to a test assembly sub-problem for the test form t. Also the T matrices and vectors involved in the linear constraints and in the objective function are kept separated. Along the iterations each of the test is evaluated separatedly in terms of optimality and feasibility. This separation allows to speed up the algorithm since all the algebraic operations are made on smaller objects. The only constraints which are not separable are the overlap 2.2.4 and the item use 2.2.4 which are evaluated on the full-length vector of optimization variables.

The simulated annealing has the disadvantage that is hardly able to find the feasible space of a problem, this is why we decided to start our heuristic by a *fill up* sequential phase in which the worst performing test, both in terms of

optimality and feasibility[5], is "filled up" with the best item available in the item pool. After the item has been assigned, the process is repeated until all the tests have reached their maximum length, i.e. they are all "filled-up".

Once the first step is performed, luckily we have at least a feasible solution to process with the simulated annealing principle. In details, the first W worst tests $\mathbf{x}_1, \dots, \mathbf{x}_W$ are taken and a fixed number of items V, already taken in these tests, is sampled, namely $\mathbf{x}_{1,1}, \dots, \mathbf{x}_{1,V}, \dots, \mathbf{x}_{W,1}, \dots, \mathbf{x}_{W,V}$. These sampled items are first, removed and secondly, switched with all the other available items in the pool. The test resulting from the removal and the switch is accepted with a chance equal to (4.9). After the sampling phase, the performance of the tests is again evaluated and if the termination criteria have not been met, tests and items are sampled again. When a certain convergence in the objective is attained we say that a neighbourhood of the space has been explored. The user can decide how far he/she wants to go from the most recent solution and hence how many neighbourhoods he/she wants to explore. If the *journey* is not completed, the last solution is substantially perturbed and the heuristic performs again the *fill up* and sampling steps.

The result of the heuristic is a set of solution of length H which is the number of neighbourhoods explored. It is also possible to decide how many of these areas must be evaluated just in terms of feasibility, H_f, and how many in terms of optimality, H_o, i.e. $H = H_f + H_o$. In this way the test assembler has a wider choice of optimally assembled tests in terms of other features not considered in the assembly model, such as content validity.

The hyperparameters (i.e. parameters chosen by the user) in the algorithms are several, the following list summarize all the customizable features:

- **Lagrange relaxation:**
  - $\beta \in [0, 1]$, as in (2.18), it serves as a balancing between optimality and feasibility. A β approaching one puts more emphasis on the optimality of the solution. Viceversa an almost zero β takes into account only the feasibility of the solution.
- **simulated annealing:**

[5] In practice we allow the user to decide if the fill up phase must be done by considering only the feasibility of the problem or adding the optimality evaluation.

  - F_f: number of feasible *fill-up* phases;
  - t_0: starting temperature;
  - `geom_temp`: the factor by which the temperature is geometrically decreased at each iteration s, i.e. $t_s = t_{s-1}/\texttt{geom\_temp}$;
  - W: number of worst performing tests to sample;
  - V: number of already selected items in each of the W tests to sample.

- **termination criteria:**
  - H_f: number of feasible neighbourhoods;
  - H_o: number of optimal neighbourhoods;
  - `rel_tol`: relative tolerance of the optimziation function for determing the convergence of the algorithm;
  - `max_time`: maximum elapsed CPU time, the algorithm stops if, at convergence, the actual elapsed CPU time is higher than `max_time`;

Most of the hyperparameters, apart from β and t_0, are positively correlated with the chance to find the global optimum, but obviously they are negatively correlated with the elapsed time. Consequently, more we increase F_f, W, V, H_f and H_o more it is likely to find the global optimal tests at the cost of a large computional time.

4.3 Simulation study

The performance and benefits of the chance-constrained test assembly model (4.7) are investigated through a simulation by implementing our heuristic 4.2.2 in `Julia` to optimize it. This setting allow us to evaluate the effects of using probabilistic methods in the field of ATA models in terms of conservatism of the test solution. In particular this achievement is assessed by comparing the quantile of TIFs obtained by our model and the classical one solved by `CPLEX`. On the other hand, in order to show the computational and pratical power of our solver, also the classical linear ATA MAXIMIN model (2.10) is solved through our heuristic and the overall estimated TIFs of the assembled tests are compared with `CPLEX` solutions.

The data needed for assembling the chance-constrained tests consists of the sample of the IIFs computed at the predetermined ability points, θ_{k_t}, of each item in the pool, namely the $\mathbf{I}_r(\theta_{k_t})$, for $r = 1 \ldots, R$. These quantities are obtained by bootstrapping the calibration process, the procedure is described in Section 3.1.5. In particular the parametric approach is arbitrarily used in this simulation. As a golden rule, the sample which better represents the distribution function of the random variables in the test assembly model should be used. For the classical model a calibrated item pool is needed. A 2-parameter logistic IRT model is assumed and for the other settings the *standard setup* described in Section 3.2.1 has been chosen.

Tests specifications

After the calibration, the models (2.10) and (4.7) are solved using our heuristic under different specifications, such as the number of test forms and the confidence level, α. The assembly is performed in a parallel framework, i.e. all the tests must meet the same constraints. Two fictiuos categorical variables, *content_A* and *content_B*, with three possible values each, are simulated to constrain the test to have a certain content validity. The complete set of specifications are summarized in the following table:

TABLE 4.1: Test specifications

Number of tests	{10, 20, 25}
Test length	{38, 40}
content_A	[6, 10], [9, 12], [18, 25][6]
content_B	[9, 12], [15, 19], [9, 12]
Maximum overlap between tests	11
α	{0.05, 0.01}

Different combinations of these specifications create 12 cases to be investigated.

[6]This specification requires that each test must have from 6 to 10 items having the first value of the variable content_A, from 9 to 12 items having the second value etc...

4.3.1 Results

Classic MAXIMIN parallel test assembly model

TABLE 4.2: $\min_t [TIF_t(0)]$(infeasibility)

	model		MAXIMIN strict	MAXIMIN LR	
case	T	Item use max	CPLEX	CPLEX	ATAheur
1	10	4	14.863	14.992(0.0045)	**15.350**(0)
2	10	2	**11.318**	11.317 (0)	11.255(0)
3	20	4	11.018	11.237(0)	**11.244**(0)
4	25	4	No sol.	6.883(131.27)	**9.309**(1e-4)

CCMAXIMIN parallel test assembly model

TABLE 4.3: $\min_t [Q(TIF_t(0), 0.05)]$(infeasibility)

model			MAXIMIN strict	MAXIMIN LR	MAXIMIN CC
case	T	Item use max	CPLEX	CPLEX	ATAheur
5	10	4	14.370	14.553(0.0045)	**14.862**(0)
6	10	2	10.837	10.808(0)	**11.034**(0)
7	20	4	10.652	10.685(0)	**10.970**(0)
8	25	4	No sol.	6.639(131.27)	**9.394**(1e-3)

TABLE 4.4: $\min_t [Q(TIF_t(0), 0.01)]$(infeasibility)

model			MAXIMIN strict	MAXIMIN LR	MAXIMIN CC
case	T	Item use max	CPLEX	CPLEX	ATAheur
9	10	4	13.892	14.004(0.0045)	**14.703**(0)
10	10	2	10.567	10.402(0)	**10.688**(0)
11	20	4	10.288	10.336(0)	**10.664**(0)
12	25	4	No sol.	6.332(131.27)	**8.715**(0)

5 Conclusions and further research

In Chapter 3 a framework for deriving a pre-test design and for doing simulation studies on IRT models estimation methods have been described together with the cubic-spline method for interpolation and extrapolation of the masses used in the quadrature to the starting knots. Moreover, the results of the simulation showed that `Julia` is a programming framework with a potential to be exploited by statisticians interested in optimization, of which parameters estimation involving likelihood maximization is a special case. In particular the package `JuMP` helps the user to interface itself to the large number of solvers available on the market, both commercial (e.g. `CPLEX` and `Gurobi`) and open-source (e.g. `NLopt` and `Ipopt`).

We want to point out that the software we used as a benchmark, that is the `R` package `mirt` offers a multitude of options for latent regression models such as multidimensional models and different EM algorithms (such as stochastic EM and Monte Carlo EM), we suggest to refer to the documentation of the package for further details. Also, their output is very rich; it produces standard errors and other model diagnostics. Our code, instead, provides only the estimates of the latent variable, the final weights of the latent distribution, the calibrated IRT item parameter and their bootstrapped standard errors, since it is out of the scope of this work to inspect model diagnostics and other statistics.

The analysis of the empirical distribution functions of the item parameters showed that the bootstrap is a powerful, prior free, tool to inspect the uncertainty of the item parameters because is able to capture the full characterization of the variability of the items due to the sample error. We believe that a better specification of the parametric scheme would improve also the accuracy of the parameters estimation.

Furthermore, looking at the number of iterations for each case under analysis

we can say that the cubic-spline method for extrapolating the rescaled masses on the original knots of the ability distribution is a valid alternative to Wood's empirical histogram in terms of convergence and accuracy of estimates, mostly in the non-normal case. Finally, the results showed that `Julia` could compete with `R` in terms of computational performance.

A Tables and Figures

A.1 Test theories and ATA

Table A.1: Example of item bank.

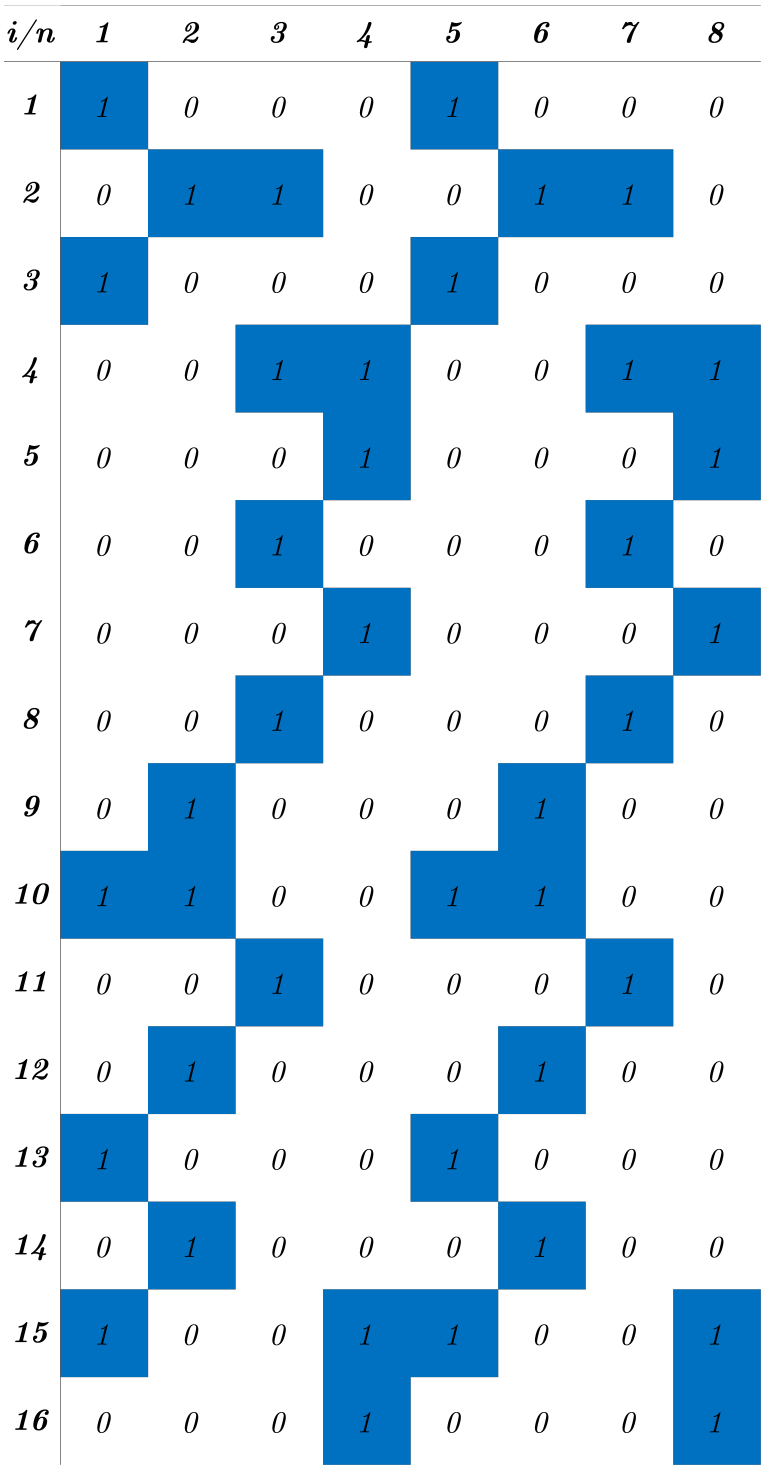
i	*ID*	b	b_{se}	*ES*	*FORMAT*	*PROCESS*	*DOMAIN*	ITEM SET 1	ITEM SET 2	ENEMY SET 1	ENEMY SET 2	ENEMY SET 3
1	M02KL	-2.0	1.02	0.6	Matching	Problem solving	Numbers	1	0	1	0	0
2	M35KL	-1.5	0.12	0.3	Multiple-choice	Knowing	Space and figures	1	0	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$I-2$	M03PF	1.2	0.05	0.4	Open-ended	Knowing	Numbers	0	1	1	0	1
$I-1$	M08PF	0.06	0.98	0.35	Multiple-choice	Knowing	Numbers	0	1	1	0	1
I	M10ML	0.75	0.4	0.12	Multiple-choice	Knowing	Space and figures	0	0	1	0	1

i/t	*1*	*2*	*3*	*4*
1	*1*	*0*	*0*	*0*
2	*0*	*1*	*1*	*0*
3	*1*	*0*	*0*	*0*
4	*0*	*0*	*1*	*1*
5	*0*	*0*	*0*	*1*
6	*0*	*0*	*1*	*0*
7	*0*	*0*	*0*	*1*
8	*0*	*0*	*1*	*0*
9	*0*	*1*	*0*	*0*
10	*1*	*1*	*0*	*0*
11	*0*	*0*	*1*	*0*
12	*0*	*1*	*0*	*0*
13	*1*	*0*	*0*	*0*
14	*0*	*1*	*0*	*0*
15	*1*	*0*	*0*	*1*
16	*0*	*0*	*0*	*1*

FIGURE A.1: An example of unbalanced *items* × *tests* design, for $T = 4$ tests with lenght equal to 5 and 2 anchor items (overlap), and $I = 16$ items from the pool.

t/n	1	2	3	4	5	6	7	8
1	1	0	0	0	1	0	0	0
2	0	1	0	0	0	1	0	0
3	0	0	1	0	0	0	1	0
4	0	0	0	1	0	0	0	1

FIGURE A.2: An example of *tests* × *examinees* design, for $T = 4$ tests, and $N = 8$ test takers.



i/n	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
1	1	0	0	0	1	0	0	0
2	0	1	1	0	0	1	1	0
3	1	0	0	0	1	0	0	0
4	0	0	1	1	0	0	1	1
5	0	0	0	1	0	0	0	1
6	0	0	1	0	0	0	1	0
7	0	0	0	1	0	0	0	1
8	0	0	1	0	0	0	1	0
9	0	1	0	0	0	1	0	0
10	1	1	0	0	1	1	0	0
11	0	0	1	0	0	0	1	0
12	0	1	0	0	0	1	0	0
13	1	0	0	0	1	0	0	0
14	0	1	0	0	0	1	0	0
15	1	0	0	1	1	0	0	1
16	0	0	0	1	0	0	0	1

FIGURE A.3: An example of *items* × *examinees* design, for $I = 16$ items, and $N = 8$ test takers.

A.2 Items calibration in Julia

A.2.1 Case 1a (Standard setup)

TABLE A.2: Case 1a - RMSE and BIAS of item parameters averaged across the simulations, part 1/5

		b RMSE			b BIAS				a RMSE			a BIAS		
i	$b^{\dagger}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$	$a^{\dagger}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$
1	-1.942	0.09	0.089	0.09	-0.018	-0.014	-0.018	0.893	0.111	0.112	0.111	-0.015	-0.015	-0.014
2	-0.704	0.095	0.094	0.095	-0.022	-0.019	-0.022	0.805	0.117	0.117	0.117	-0.017	-0.014	-0.017
3	-0.842	0.084	0.084	0.084	-0.02	-0.017	-0.02	0.963	0.098	0.099	0.098	-0.023	-0.02	-0.023
4	-0.502	0.09	0.09	0.09	0.044	0.045	0.045	0.838	0.071	0.072	0.071	0.003	0.003	0.004
5	1.104	0.1	0.098	0.1	-0.039	-0.036	-0.039	1.12	0.104	0.104	0.104	0.006	0.008	0.007
6	-0.625	0.077	0.075	0.077	-0.027	-0.023	-0.026	1.297	0.106	0.105	0.106	-0.043	-0.037	-0.042
7	-0.39	0.073	0.072	0.073	-0.015	-0.011	-0.015	1.035	0.09	0.09	0.09	-0.032	-0.028	-0.031
8	0.077	0.087	0.086	0.087	-0.03	-0.026	-0.03	1.457	0.116	0.115	0.116	-0.038	-0.03	-0.037
9	-1.368	0.093	0.092	0.093	-0.019	-0.015	-0.019	1.059	0.105	0.106	0.104	-0.015	-0.012	-0.014
10	-0.437	0.086	0.084	0.086	-0.041	-0.037	-0.041	1.031	0.101	0.102	0.102	0.006	0.01	0.007
11	-1.4	0.1	0.099	0.1	-0.03	-0.026	-0.03	1.094	0.101	0.101	0.101	-0.018	-0.014	-0.016
12	-0.617	0.087	0.086	0.087	-0.033	-0.03	-0.033	0.903	0.085	0.085	0.085	-0.022	-0.019	-0.022
13	-0.068	0.078	0.077	0.078	-0.016	-0.014	-0.016	0.55	0.084	0.084	0.084	0.012	0.013	0.012
14	1.432	0.101	0.1	0.101	-0.039	-0.036	-0.039	1.033	0.106	0.105	0.106	-0.024	-0.022	-0.022
15	-0.379	0.08	0.079	0.08	-0.02	-0.017	-0.02	0.893	0.096	0.097	0.097	-0.009	-0.005	-0.008
16	-1.547	0.104	0.103	0.103	-0.026	-0.021	-0.026	1.402	0.136	0.134	0.134	-0.062	-0.055	-0.059
17	-0.628	0.09	0.088	0.091	-0.037	-0.034	-0.037	1.086	0.099	0.101	0.099	0.024	0.029	0.026
18	-0.023	0.085	0.084	0.085	-0.025	-0.022	-0.026	1.162	0.099	0.101	0.1	0.002	0.007	0.003
19	0.051	0.073	0.072	0.073	-0.029	-0.027	-0.029	0.811	0.088	0.088	0.088	-0.03	-0.027	-0.029
20	0.906	0.07	0.069	0.07	-0.019	-0.017	-0.019	0.701	0.086	0.086	0.087	0.002	0.003	0.003
21	-2.054	0.13	0.13	0.13	0.015	0.019	0.015	1.057	0.114	0.114	0.114	0.009	0.007	0.011
22	1.0	0.092	0.092	0.092	0.014	0.016	0.015	1.225	0.109	0.11	0.11	0.004	0.005	0.006
23	0.963	0.113	0.112	0.113	-0.028	-0.025	-0.028	0.978	0.119	0.119	0.119	0.019	0.021	0.02
24	0.013	0.07	0.07	0.07	0.017	0.019	0.018	1.323	0.107	0.109	0.108	0.023	0.028	0.025
25	1.543	0.12	0.119	0.12	-0.038	-0.035	-0.037	1.514	0.137	0.136	0.136	-0.027	-0.025	-0.026
26	0.697	0.081	0.081	0.081	0.046	0.047	0.047	1.177	0.101	0.101	0.103	0.012	0.012	0.014
27	-0.456	0.131	0.13	0.131	-0.033	-0.029	-0.033	1.24	0.147	0.149	0.148	0.027	0.034	0.029
28	-1.062	0.121	0.121	0.121	-0.006	-0.002	-0.005	1.152	0.133	0.134	0.134	-0.016	-0.013	-0.015
29	2.575	0.155	0.155	0.155	-0.017	-0.015	-0.017	1.079	0.147	0.146	0.148	0.023	0.023	0.025
30	-0.251	0.081	0.081	0.081	0.011	0.013	0.012	1.222	0.098	0.099	0.099	0.007	0.011	0.008
31	1.672	0.091	0.092	0.091	0.042	0.042	0.042	0.938	0.089	0.089	0.09	0.03	0.029	0.032
32	-1.746	0.124	0.123	0.124	-0.03	-0.025	-0.03	1.244	0.117	0.117	0.117	-0.028	-0.025	-0.027
33	-0.363	0.076	0.077	0.077	0.03	0.03	0.03	0.934	0.093	0.093	0.093	0.006	0.007	0.008
34	0.509	0.067	0.067	0.067	-0.005	-0.003	-0.005	0.826	0.089	0.089	0.088	-0.014	-0.011	-0.013
35	-0.691	0.11	0.109	0.11	-0.022	-0.019	-0.022	1.104	0.146	0.148	0.146	-0.003	0.002	-0.001
36	-0.517	0.088	0.086	0.088	-0.042	-0.038	-0.042	1.169	0.1	0.103	0.1	0.028	0.034	0.03
37	1.565	0.111	0.112	0.111	0.024	0.025	0.025	1.176	0.112	0.111	0.113	0.027	0.027	0.029
38	-0.464	0.109	0.109	0.109	0.016	0.018	0.017	1.203	0.143	0.145	0.144	0.036	0.039	0.038
39	0.396	0.117	0.115	0.117	-0.041	-0.037	-0.04	1.155	0.139	0.139	0.14	-0.024	-0.018	-0.022
40	-0.291	0.138	0.138	0.138	0.09	0.09	0.091	1.453	0.177	0.178	0.178	0.041	0.046	0.043
41	-0.24	0.115	0.114	0.115	-0.034	-0.03	-0.034	1.413	0.149	0.148	0.149	-0.038	-0.029	-0.037
42	0.749	0.092	0.091	0.092	-0.017	-0.014	-0.017	1.306	0.115	0.115	0.114	-0.022	-0.018	-0.021
43	-1.447	0.131	0.129	0.131	-0.042	-0.039	-0.042	0.941	0.171	0.171	0.171	-0.044	-0.04	-0.043
44	-0.835	0.103	0.103	0.103	-0.025	-0.023	-0.025	0.654	0.115	0.115	0.115	-0.009	-0.008	-0.008
45	1.481	0.121	0.121	0.121	0.021	0.023	0.021	0.817	0.141	0.141	0.141	0.004	0.005	0.004
46	2.0	0.123	0.122	0.123	-0.017	-0.015	-0.017	1.026	0.122	0.122	0.123	0.006	0.006	0.008
47	-0.488	0.082	0.081	0.082	0.004	0.006	0.004	0.865	0.101	0.102	0.102	0.014	0.016	0.015
48	1.146	0.086	0.086	0.086	0.018	0.02	0.019	0.937	0.093	0.093	0.092	-0.011	-0.01	-0.009
49	-0.26	0.111	0.11	0.111	-0.026	-0.023	-0.026	1.044	0.113	0.115	0.113	0.003	0.007	0.005
50	-0.016	0.09	0.091	0.09	0.002	0.004	0.002	0.737	0.109	0.11	0.11	0.01	0.012	0.011

TABLE A.3: Case 1a - RMSE and BIAS of item parameters averaged across the simulations, part 2/5

		b RMSE			b BIAS				a RMSE			a BIAS		
i	$b^{\dagger}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$	$a^{\dagger}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$
51	-2.564	0.19	0.189	0.19	0.025	0.028	0.026	0.794	0.163	0.159	0.164	0.029	0.023	0.029
52	-1.078	0.122	0.122	0.122	0.039	0.04	0.04	0.863	0.122	0.122	0.123	0.035	0.034	0.036
53	-1.115	0.124	0.125	0.125	0.062	0.063	0.063	1.067	0.14	0.14	0.141	0.01	0.011	0.012
54	-0.156	0.131	0.131	0.132	0.075	0.075	0.076	1.078	0.138	0.139	0.138	0.048	0.049	0.049
55	1.776	0.165	0.164	0.165	0.081	0.081	0.082	0.945	0.162	0.162	0.163	0.031	0.029	0.032
56	0.452	0.089	0.09	0.089	0.053	0.053	0.053	1.22	0.084	0.086	0.085	0.037	0.039	0.039
57	-0.757	0.146	0.146	0.147	0.086	0.087	0.087	1.154	0.148	0.15	0.148	0.03	0.032	0.032
58	-0.763	0.128	0.127	0.128	-0.015	-0.011	-0.015	1.042	0.137	0.138	0.138	0.027	0.031	0.028
59	1.02	0.104	0.103	0.104	-0.018	-0.015	-0.018	0.853	0.127	0.128	0.127	0.034	0.036	0.035
60	-0.4	0.101	0.1	0.101	-0.024	-0.022	-0.025	0.711	0.104	0.105	0.104	0.022	0.024	0.022
61	0.625	0.107	0.107	0.107	-0.008	-0.005	-0.009	1.185	0.145	0.147	0.145	0.029	0.033	0.031
62	-0.325	0.098	0.097	0.098	-0.021	-0.019	-0.021	0.649	0.109	0.11	0.11	0.035	0.037	0.036
63	-1.807	0.157	0.156	0.157	-0.048	-0.044	-0.048	0.796	0.164	0.164	0.166	0.037	0.036	0.037
64	-0.4	0.106	0.106	0.106	-0.013	-0.009	-0.013	1.081	0.134	0.136	0.133	0.027	0.031	0.028
65	0.824	0.111	0.11	0.11	-0.02	-0.017	-0.021	1.228	0.157	0.158	0.158	0.04	0.044	0.041
66	0.577	0.112	0.111	0.112	-0.027	-0.024	-0.028	1.157	0.17	0.172	0.172	0.05	0.053	0.051
67	-0.896	0.117	0.117	0.117	0.057	0.058	0.057	0.942	0.134	0.134	0.135	0.027	0.027	0.029
68	-0.425	0.144	0.144	0.144	0.083	0.084	0.084	1.172	0.161	0.163	0.16	0.03	0.033	0.032
69	-1.95	0.171	0.17	0.171	0.041	0.043	0.042	0.882	0.16	0.159	0.161	0.04	0.036	0.042
70	0.815	0.151	0.152	0.152	0.075	0.075	0.076	1.023	0.136	0.136	0.136	0.024	0.024	0.025
71	0.16	0.108	0.108	0.109	-0.034	-0.031	-0.034	1.047	0.135	0.136	0.135	0.013	0.017	0.015
72	0.164	0.092	0.092	0.092	-0.012	-0.009	-0.011	0.759	0.104	0.105	0.105	-0.008	-0.006	-0.007
73	-0.651	0.101	0.101	0.101	-0.019	-0.017	-0.019	0.736	0.105	0.106	0.105	0.011	0.013	0.011
74	0.042	0.1	0.1	0.101	-0.009	-0.006	-0.009	0.851	0.133	0.133	0.133	-0.029	-0.025	-0.028
75	0.653	0.108	0.108	0.108	-0.026	-0.023	-0.025	0.879	0.122	0.121	0.122	-0.028	-0.025	-0.027
76	-0.765	0.087	0.086	0.087	-0.028	-0.027	-0.028	0.408	0.122	0.122	0.122	-0.002	-0.001	-0.002
77	0.729	0.106	0.106	0.106	0.005	0.007	0.004	0.787	0.126	0.127	0.126	0.029	0.031	0.03
78	-2.147	0.146	0.144	0.145	-0.033	-0.028	-0.032	1.208	0.173	0.172	0.174	0.048	0.044	0.051
79	-0.702	0.109	0.108	0.109	-0.036	-0.033	-0.037	0.979	0.12	0.122	0.12	0.029	0.033	0.03
80	-1.629	0.139	0.138	0.139	-0.042	-0.039	-0.042	0.65	0.138	0.139	0.138	-0.01	-0.009	-0.01
81	-0.408	0.118	0.116	0.118	-0.037	-0.034	-0.038	1.251	0.155	0.158	0.154	0.022	0.029	0.023
82	-0.948	0.114	0.113	0.115	-0.017	-0.013	-0.017	0.971	0.144	0.146	0.144	0.01	0.014	0.011
83	0.62	0.106	0.106	0.106	0.032	0.033	0.033	0.741	0.123	0.124	0.125	0.027	0.028	0.029
84	1.187	0.116	0.115	0.116	-0.008	-0.005	-0.007	0.92	0.154	0.155	0.154	-0.004	-0.002	-0.002
85	-2.191	0.21	0.204	0.211	-0.041	-0.033	-0.042	1.416	0.2	0.2	0.2	0.013	0.013	0.014
86	0.467	0.11	0.11	0.111	-0.018	-0.015	-0.018	0.997	0.119	0.12	0.12	0.015	0.018	0.016
87	0.865	0.117	0.117	0.117	-0.005	-0.002	-0.005	1.022	0.151	0.153	0.152	0.053	0.056	0.054
88	-0.745	0.108	0.108	0.108	-0.003	0.0	-0.003	1.308	0.148	0.15	0.147	-0.001	0.005	0.0
89	1.071	0.125	0.125	0.125	0.028	0.028	0.028	1.029	0.144	0.144	0.145	0.007	0.007	0.01
90	0.124	0.113	0.113	0.113	-0.035	-0.032	-0.035	0.985	0.123	0.124	0.123	-0.0	0.003	0.001
91	0.695	0.115	0.115	0.115	0.021	0.022	0.021	0.971	0.125	0.125	0.125	0.006	0.007	0.008
92	0.516	0.108	0.108	0.108	0.023	0.026	0.023	1.152	0.152	0.153	0.152	-0.022	-0.018	-0.021
93	-0.088	0.087	0.088	0.087	0.01	0.012	0.01	0.955	0.121	0.121	0.12	-0.029	-0.027	-0.028
94	0.544	0.114	0.115	0.115	0.047	0.047	0.047	0.665	0.115	0.115	0.115	0.024	0.024	0.025
95	-1.625	0.146	0.144	0.146	-0.039	-0.035	-0.039	0.913	0.159	0.159	0.16	-0.003	-0.003	-0.001
96	0.505	0.106	0.106	0.107	-0.017	-0.013	-0.018	1.348	0.163	0.164	0.164	0.04	0.046	0.041
97	0.844	0.122	0.121	0.122	-0.023	-0.02	-0.023	0.821	0.129	0.13	0.13	0.034	0.036	0.035
98	1.42	0.153	0.153	0.153	0.064	0.064	0.065	0.977	0.131	0.13	0.131	-0.006	-0.007	-0.005
99	-0.409	0.121	0.12	0.122	-0.026	-0.022	-0.026	1.187	0.158	0.161	0.159	0.052	0.058	0.053
100	-0.525	0.117	0.116	0.117	-0.035	-0.032	-0.036	1.054	0.141	0.144	0.142	0.034	0.039	0.036

TABLE A.4: Case 1a - RMSE and BIAS of item parameters averaged across the simulations, part 3/5

		b RMSE			b BIAS				a RMSE			a BIAS		
i	$b^{\dagger}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$	$a^{\dagger}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$
101	0.648	0.108	0.108	0.108	-0.025	-0.021	-0.025	1.574	0.192	0.193	0.192	0.015	0.022	0.017
102	0.342	0.111	0.11	0.111	-0.029	-0.026	-0.029	1.019	0.138	0.138	0.138	-0.032	-0.028	-0.032
103	-1.121	0.163	0.163	0.163	0.087	0.088	0.087	1.363	0.174	0.178	0.175	0.06	0.064	0.062
104	-0.354	0.11	0.109	0.11	-0.018	-0.014	-0.018	1.083	0.148	0.151	0.149	0.008	0.012	0.009
105	0.302	0.075	0.075	0.075	-0.007	-0.004	-0.006	0.851	0.086	0.086	0.086	-0.017	-0.014	-0.016
106	-0.535	0.119	0.117	0.12	-0.027	-0.023	-0.028	1.236	0.161	0.165	0.161	0.036	0.042	0.037
107	0.156	0.09	0.089	0.09	-0.033	-0.031	-0.033	0.646	0.108	0.108	0.108	-0.021	-0.019	-0.021
108	1.033	0.111	0.111	0.111	-0.008	-0.006	-0.008	0.753	0.144	0.145	0.145	0.037	0.039	0.038
109	-1.446	0.122	0.121	0.122	-0.016	-0.015	-0.016	0.746	0.137	0.136	0.138	0.039	0.038	0.041
110	-2.571	0.223	0.22	0.224	-0.024	-0.017	-0.024	1.02	0.225	0.224	0.225	-0.024	-0.028	-0.024
111	0.627	0.095	0.094	0.095	-0.014	-0.013	-0.014	0.624	0.106	0.107	0.107	0.005	0.006	0.006
112	0.855	0.183	0.182	0.184	0.116	0.116	0.118	1.758	0.204	0.204	0.204	0.014	0.016	0.016
113	-1.154	0.168	0.166	0.169	-0.066	-0.061	-0.067	1.482	0.212	0.219	0.213	0.083	0.093	0.084
114	1.02	0.104	0.104	0.104	-0.001	0.001	-0.001	0.593	0.125	0.124	0.125	-0.013	-0.012	-0.013
115	0.851	0.106	0.106	0.106	-0.012	-0.01	-0.012	0.933	0.146	0.147	0.147	0.023	0.026	0.025
116	0.88	0.127	0.127	0.127	0.008	0.01	0.008	0.839	0.146	0.146	0.146	-0.015	-0.014	-0.014
117	-0.924	0.136	0.134	0.135	-0.048	-0.044	-0.048	1.098	0.164	0.166	0.164	0.049	0.053	0.05
118	1.09	0.126	0.125	0.126	-0.032	-0.029	-0.032	1.213	0.135	0.135	0.135	-0.018	-0.015	-0.017
119	-0.833	0.102	0.101	0.102	-0.014	-0.011	-0.014	1.02	0.145	0.145	0.145	-0.047	-0.042	-0.046
120	0.48	0.091	0.091	0.091	-0.021	-0.018	-0.021	0.707	0.108	0.108	0.108	-0.028	-0.026	-0.027
121	-0.074	0.096	0.096	0.096	0.013	0.014	0.014	0.795	0.103	0.103	0.104	0.017	0.017	0.019
122	-0.808	0.124	0.124	0.124	0.029	0.031	0.03	0.994	0.146	0.147	0.147	0.013	0.015	0.015
123	-1.243	0.136	0.136	0.136	-0.025	-0.021	-0.026	1.103	0.142	0.144	0.142	0.031	0.034	0.031
124	-0.524	0.116	0.116	0.116	0.003	0.005	0.004	1.309	0.156	0.159	0.157	0.047	0.051	0.049
125	0.002	0.092	0.092	0.092	-0.013	-0.01	-0.013	0.893	0.125	0.126	0.125	0.013	0.015	0.014
126	-0.355	0.097	0.097	0.097	0.013	0.014	0.014	0.705	0.114	0.114	0.114	0.005	0.005	0.007
127	-0.575	0.1	0.1	0.1	-0.001	0.001	-0.001	0.672	0.106	0.107	0.106	-0.007	-0.006	-0.006
128	1.402	0.144	0.145	0.145	0.02	0.022	0.02	1.021	0.154	0.154	0.154	0.017	0.018	0.019
129	-0.337	0.109	0.108	0.109	-0.04	-0.036	-0.04	0.976	0.152	0.155	0.153	0.049	0.053	0.05
130	0.495	0.121	0.121	0.121	-0.028	-0.024	-0.027	1.384	0.149	0.151	0.149	0.017	0.023	0.019
131	-0.293	0.094	0.093	0.094	-0.014	-0.012	-0.014	0.689	0.115	0.115	0.115	-0.037	-0.035	-0.036
132	0.841	0.105	0.105	0.105	-0.017	-0.014	-0.016	0.886	0.148	0.149	0.148	0.018	0.02	0.02
133	-1.18	0.113	0.112	0.113	-0.032	-0.03	-0.032	0.676	0.119	0.119	0.12	0.005	0.007	0.007
134	1.117	0.11	0.11	0.11	0.026	0.026	0.026	1.027	0.157	0.157	0.158	0.047	0.047	0.049
135	-0.275	0.102	0.101	0.102	-0.023	-0.02	-0.023	0.922	0.133	0.133	0.133	-0.037	-0.033	-0.036
136	-0.629	0.104	0.104	0.105	0.001	0.002	0.001	0.699	0.122	0.122	0.122	0.016	0.016	0.018
137	1.167	0.128	0.126	0.128	-0.053	-0.049	-0.053	1.403	0.169	0.169	0.169	-0.048	-0.044	-0.047
138	0.589	0.107	0.107	0.107	0.024	0.025	0.024	0.868	0.144	0.144	0.145	0.02	0.021	0.022
139	-0.431	0.095	0.096	0.095	-0.002	0.001	-0.002	0.869	0.131	0.131	0.131	-0.006	-0.003	-0.005
140	-0.481	0.102	0.103	0.103	0.026	0.027	0.026	1.007	0.122	0.123	0.122	0.031	0.032	0.033
141	0.407	0.114	0.114	0.115	0.063	0.064	0.064	0.908	0.124	0.124	0.124	0.022	0.022	0.023
142	1.198	0.091	0.091	0.091	0.028	0.029	0.028	0.902	0.105	0.105	0.106	0.031	0.031	0.032
143	1.97	0.19	0.19	0.191	-0.021	-0.018	-0.021	1.256	0.175	0.175	0.176	0.036	0.037	0.037
144	-1.514	0.136	0.134	0.136	-0.034	-0.029	-0.034	1.354	0.189	0.188	0.188	-0.076	-0.069	-0.074
145	-0.594	0.103	0.103	0.103	0.012	0.013	0.012	1.069	0.131	0.132	0.132	0.014	0.015	0.016
146	0.685	0.119	0.119	0.119	0.037	0.038	0.037	1.06	0.14	0.141	0.141	0.046	0.047	0.048
147	1.502	0.14	0.14	0.14	-0.001	0.002	-0.001	1.253	0.183	0.183	0.184	-0.021	-0.019	-0.02
148	0.268	0.089	0.088	0.089	-0.029	-0.027	-0.029	0.825	0.116	0.116	0.117	-0.009	-0.007	-0.008
149	-0.742	0.107	0.107	0.107	-0.002	-0.0	-0.001	1.367	0.157	0.16	0.157	0.03	0.034	0.032
150	0.955	0.117	0.115	0.117	-0.018	-0.015	-0.018	1.047	0.136	0.137	0.136	0.036	0.039	0.038

TABLE A.5: Case 1a - RMSE and BIAS of item parameters averaged across the simulations, part 4/5

		b RMSE			b BIAS				a RMSE			a BIAS		
i	$b^{\dagger}$	jl	mirt	mirtEHW	jl	mirt	mirtEHW	$a^{\dagger}$	jl	mirt	mirtEHW	jl	mirt	mirtEHW
151	-0.769	0.116	0.115	0.116	-0.024	-0.022	-0.025	0.648	0.128	0.128	0.128	0.017	0.017	0.017
152	0.702	0.106	0.105	0.107	-0.031	-0.027	-0.031	1.295	0.15	0.15	0.15	-0.036	-0.03	-0.035
153	-0.216	0.1	0.099	0.1	-0.021	-0.019	-0.022	0.655	0.121	0.121	0.12	0.016	0.017	0.017
154	-0.975	0.114	0.112	0.114	-0.026	-0.023	-0.027	0.988	0.153	0.155	0.154	0.047	0.05	0.048
155	-1.306	0.123	0.123	0.123	-0.003	0.0	-0.003	1.019	0.146	0.148	0.147	-0.019	-0.017	-0.018
156	-1.22	0.131	0.131	0.131	0.023	0.025	0.023	1.135	0.167	0.168	0.168	0.024	0.024	0.026
157	0.777	0.14	0.141	0.141	0.084	0.084	0.085	1.316	0.159	0.159	0.16	0.002	0.003	0.003
158	-0.375	0.111	0.111	0.111	-0.02	-0.017	-0.02	1.72	0.185	0.188	0.186	-0.012	-0.0	-0.012
159	-0.077	0.124	0.122	0.124	-0.034	-0.03	-0.034	1.645	0.165	0.163	0.166	-0.046	-0.035	-0.045
160	0.581	0.097	0.097	0.097	0.01	0.01	0.01	0.612	0.112	0.112	0.112	0.018	0.018	0.02
161	-1.697	0.149	0.148	0.148	0.014	0.018	0.014	0.934	0.159	0.158	0.16	-0.035	-0.035	-0.034
162	0.255	0.119	0.12	0.12	0.041	0.042	0.042	1.354	0.165	0.166	0.166	0.047	0.049	0.049
163	2.141	0.198	0.198	0.198	0.04	0.042	0.041	1.444	0.203	0.202	0.203	0.003	0.002	0.005
164	-0.082	0.105	0.105	0.105	-0.012	-0.009	-0.012	1.015	0.118	0.12	0.119	0.004	0.008	0.006
165	0.53	0.087	0.087	0.087	-0.02	-0.017	-0.019	0.743	0.122	0.122	0.121	-0.027	-0.025	-0.026
166	-0.082	0.109	0.109	0.109	-0.003	0.001	-0.003	1.109	0.155	0.155	0.154	-0.04	-0.034	-0.039
167	-0.053	0.106	0.105	0.106	-0.015	-0.012	-0.015	1.145	0.136	0.137	0.136	-0.002	0.002	-0.001
168	-0.401	0.104	0.104	0.103	-0.013	-0.011	-0.013	0.717	0.118	0.119	0.118	0.006	0.008	0.007
169	-0.994	0.12	0.119	0.12	-0.039	-0.036	-0.039	1.088	0.142	0.144	0.142	0.004	0.008	0.005
170	0.656	0.117	0.117	0.118	0.069	0.069	0.07	1.05	0.133	0.133	0.133	0.039	0.04	0.04
171	-1.592	0.143	0.142	0.143	0.017	0.02	0.018	1.11	0.144	0.144	0.144	0.025	0.024	0.027
172	0.655	0.117	0.117	0.117	0.019	0.02	0.02	0.894	0.137	0.137	0.138	0.027	0.027	0.029
173	2.449	0.193	0.193	0.194	0.027	0.029	0.028	0.995	0.188	0.186	0.189	0.011	0.01	0.014
174	-0.666	0.089	0.09	0.09	0.013	0.016	0.014	0.787	0.123	0.123	0.123	-0.003	-0.002	-0.003
175	-0.746	0.115	0.114	0.115	-0.035	-0.031	-0.035	1.003	0.138	0.139	0.138	0.025	0.028	0.026
176	-0.236	0.093	0.093	0.093	-0.002	0.0	-0.002	0.587	0.106	0.106	0.106	0.002	0.003	0.003
177	0.59	0.096	0.095	0.096	-0.026	-0.023	-0.026	0.812	0.109	0.109	0.109	-0.022	-0.019	-0.021
178	0.102	0.123	0.121	0.123	-0.043	-0.039	-0.043	1.815	0.201	0.202	0.201	-0.007	0.004	-0.006
179	0.992	0.097	0.098	0.097	0.046	0.047	0.047	1.06	0.096	0.097	0.097	0.018	0.018	0.02
180	0.107	0.099	0.098	0.099	-0.031	-0.027	-0.031	1.188	0.142	0.144	0.142	0.024	0.029	0.025
181	0.767	0.123	0.124	0.124	0.035	0.036	0.036	1.284	0.177	0.177	0.178	0.048	0.049	0.051
182	0.376	0.108	0.107	0.107	-0.013	-0.011	-0.013	0.876	0.128	0.128	0.128	-0.011	-0.009	-0.01
183	-0.357	0.112	0.112	0.112	0.016	0.017	0.017	1.488	0.164	0.167	0.165	0.06	0.066	0.063
184	0.109	0.102	0.101	0.102	-0.013	-0.009	-0.013	0.952	0.127	0.128	0.128	0.024	0.028	0.025
185	1.775	0.125	0.124	0.125	-0.021	-0.018	-0.02	1.325	0.131	0.13	0.131	-0.017	-0.016	-0.016
186	1.444	0.097	0.096	0.097	0.003	0.004	0.003	0.844	0.084	0.083	0.084	0.001	0.001	0.003
187	-1.223	0.129	0.127	0.129	-0.04	-0.036	-0.039	1.023	0.119	0.119	0.119	-0.007	-0.004	-0.005
188	-0.689	0.079	0.079	0.079	-0.021	-0.018	-0.021	0.882	0.095	0.096	0.095	0.01	0.013	0.011
189	-0.165	0.096	0.095	0.096	-0.017	-0.014	-0.017	0.765	0.132	0.133	0.132	-0.03	-0.027	-0.029
190	-0.59	0.128	0.128	0.128	0.058	0.058	0.058	0.847	0.125	0.125	0.125	0.014	0.015	0.015
191	0.649	0.111	0.111	0.111	0.007	0.008	0.008	1.115	0.129	0.128	0.129	-0.005	-0.004	-0.003
192	0.294	0.089	0.089	0.089	0.031	0.031	0.031	0.595	0.111	0.111	0.112	0.014	0.014	0.014
193	0.372	0.097	0.097	0.097	-0.023	-0.019	-0.023	1.345	0.152	0.154	0.151	-0.007	-0.0	-0.006
194	-0.392	0.097	0.097	0.098	-0.023	-0.021	-0.023	0.793	0.119	0.12	0.119	0.01	0.012	0.011
195	-1.46	0.089	0.089	0.089	0.003	0.005	0.003	0.874	0.093	0.093	0.093	0.008	0.008	0.01
196	0.477	0.107	0.107	0.108	0.024	0.025	0.024	1.136	0.151	0.152	0.151	0.034	0.035	0.036
197	-0.815	0.096	0.096	0.096	0.004	0.006	0.004	0.786	0.105	0.105	0.105	-0.014	-0.012	-0.014
198	0.434	0.106	0.105	0.106	-0.01	-0.008	-0.01	0.786	0.113	0.114	0.113	0.015	0.017	0.017
199	-0.241	0.088	0.085	0.088	-0.05	-0.046	-0.05	1.116	0.098	0.098	0.097	-0.02	-0.014	-0.019
200	0.149	0.1	0.1	0.1	-0.01	-0.007	-0.01	0.871	0.119	0.12	0.118	-0.026	-0.023	-0.026

TABLE A.6: Case 1a - RMSE and BIAS of item parameters averaged across the simulations, part 5/5

		b RMSE			b BIAS				a RMSE			a BIAS		
i	$b^{\dagger}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$	$a^{\dagger}$	jl	mirt	mirt$_{\text{EHW}}$	jl	mirt	mirt$_{\text{EHW}}$
201	0.537	0.097	0.097	0.098	0.059	0.06	0.06	1.242	0.114	0.113	0.114	0.017	0.018	0.019
202	-0.069	0.121	0.119	0.121	-0.046	-0.041	-0.045	1.715	0.176	0.179	0.177	0.027	0.038	0.029
203	-0.476	0.103	0.102	0.103	-0.027	-0.024	-0.028	0.885	0.137	0.14	0.138	0.041	0.044	0.042
204	-1.04	0.085	0.085	0.086	-0.015	-0.013	-0.015	0.759	0.086	0.086	0.087	-0.029	-0.027	-0.028
205	0.479	0.106	0.106	0.105	0.018	0.02	0.018	0.823	0.132	0.134	0.132	0.025	0.026	0.026
206	-0.165	0.101	0.101	0.101	0.02	0.021	0.02	1.177	0.144	0.146	0.146	0.036	0.038	0.038
207	-0.848	0.122	0.121	0.122	-0.045	-0.041	-0.045	1.065	0.148	0.151	0.149	0.04	0.044	0.041
208	0.333	0.105	0.105	0.105	0.0	0.003	0.001	1.267	0.152	0.153	0.152	-0.027	-0.023	-0.026
209	-0.807	0.13	0.129	0.13	-0.032	-0.028	-0.032	1.402	0.16	0.161	0.159	-0.006	0.001	-0.005
210	0.048	0.097	0.097	0.098	-0.022	-0.018	-0.022	1.165	0.134	0.137	0.135	0.04	0.046	0.042
211	1.504	0.142	0.142	0.142	0.027	0.029	0.028	1.036	0.147	0.147	0.147	0.018	0.018	0.019
212	0.85	0.111	0.11	0.111	-0.021	-0.018	-0.021	0.879	0.133	0.133	0.133	-0.019	-0.016	-0.018
213	-0.392	0.099	0.099	0.1	-0.015	-0.012	-0.015	0.84	0.136	0.137	0.137	-0.03	-0.027	-0.029
214	-0.018	0.097	0.096	0.097	0.005	0.008	0.006	0.887	0.138	0.139	0.138	-0.025	-0.022	-0.024
215	0.842	0.104	0.104	0.104	-0.012	-0.009	-0.012	0.739	0.129	0.129	0.129	-0.016	-0.013	-0.015
216	-0.591	0.101	0.099	0.1	-0.032	-0.029	-0.032	0.847	0.133	0.134	0.133	0.018	0.02	0.02
217	-0.347	0.12	0.118	0.12	-0.05	-0.045	-0.05	1.612	0.177	0.175	0.176	-0.057	-0.045	-0.056
218	0.107	0.088	0.088	0.088	-0.0	0.002	-0.0	0.785	0.098	0.098	0.098	-0.001	0.0	-0.001
219	0.236	0.119	0.12	0.12	0.02	0.021	0.021	1.385	0.15	0.151	0.15	0.002	0.005	0.004
220	-0.377	0.096	0.096	0.096	0.034	0.035	0.035	0.69	0.107	0.108	0.107	0.012	0.012	0.013
221	0.971	0.098	0.098	0.098	-0.013	-0.01	-0.013	0.842	0.118	0.117	0.117	-0.046	-0.044	-0.045
222	-0.592	0.112	0.11	0.112	-0.036	-0.032	-0.036	1.286	0.144	0.147	0.145	0.02	0.026	0.022
223	1.279	0.124	0.124	0.124	0.013	0.014	0.013	0.813	0.139	0.139	0.14	0.019	0.019	0.021
224	0.022	0.088	0.087	0.088	-0.018	-0.016	-0.019	0.748	0.131	0.132	0.132	0.027	0.03	0.028
225	-2.236	0.159	0.158	0.16	-0.003	0.001	-0.003	0.731	0.16	0.16	0.161	-0.009	-0.01	-0.008
226	1.128	0.115	0.115	0.115	-0.014	-0.011	-0.013	1.12	0.151	0.151	0.151	0.007	0.01	0.009
227	-1.086	0.134	0.135	0.135	0.075	0.076	0.076	1.115	0.151	0.152	0.151	0.043	0.043	0.044
228	-1.638	0.135	0.135	0.135	-0.008	-0.007	-0.008	0.624	0.15	0.15	0.15	0.008	0.007	0.01
229	1.462	0.189	0.188	0.19	0.095	0.095	0.097	1.518	0.199	0.199	0.2	0.018	0.017	0.02
230	0.981	0.151	0.151	0.152	0.088	0.089	0.089	1.099	0.162	0.162	0.162	0.013	0.013	0.015
231	-0.937	0.136	0.137	0.137	0.072	0.072	0.072	0.862	0.133	0.133	0.133	-0.007	-0.007	-0.006
232	1.23	0.129	0.129	0.129	0.059	0.059	0.06	0.597	0.149	0.149	0.15	0.035	0.035	0.035
233	0.364	0.096	0.096	0.096	0.053	0.053	0.053	0.669	0.103	0.103	0.103	0.018	0.017	0.019
234	-0.501	0.078	0.078	0.078	0.043	0.044	0.044	0.911	0.096	0.097	0.097	0.022	0.023	0.023
235	0.382	0.115	0.115	0.115	0.07	0.07	0.07	0.852	0.108	0.109	0.109	0.011	0.012	0.012
236	0.321	0.087	0.088	0.087	0.035	0.036	0.036	0.962	0.097	0.097	0.098	0.012	0.013	0.014
237	-0.181	0.129	0.13	0.13	0.066	0.066	0.067	0.974	0.152	0.152	0.152	0.053	0.054	0.054
238	-0.551	0.126	0.126	0.126	0.082	0.082	0.083	1.462	0.183	0.187	0.185	0.069	0.074	0.071
239	0.566	0.165	0.166	0.166	0.111	0.111	0.112	1.198	0.152	0.153	0.154	0.044	0.045	0.046
240	1.567	0.148	0.147	0.148	0.065	0.065	0.066	0.901	0.153	0.153	0.154	0.013	0.013	0.015
241	2.517	0.224	0.224	0.225	0.068	0.068	0.069	0.809	0.183	0.182	0.184	0.019	0.019	0.021
242	-0.035	0.115	0.116	0.115	0.071	0.071	0.072	1.032	0.135	0.136	0.136	0.041	0.043	0.042
243	0.98	0.127	0.126	0.129	-0.04	-0.037	-0.041	1.461	0.181	0.184	0.183	0.084	0.089	0.086
244	-0.606	0.135	0.135	0.135	0.067	0.068	0.068	1.227	0.153	0.155	0.154	0.031	0.034	0.033
245	1.046	0.156	0.157	0.157	0.096	0.096	0.097	1.289	0.163	0.163	0.164	0.027	0.027	0.028
246	0.018	0.113	0.113	0.114	0.066	0.067	0.067	1.06	0.143	0.144	0.144	0.041	0.043	0.042
247	-1.128	0.13	0.131	0.131	0.053	0.054	0.054	1.127	0.145	0.146	0.145	0.008	0.009	0.01
248	-0.873	0.103	0.103	0.102	0.013	0.013	0.013	0.562	0.116	0.115	0.116	0.007	0.007	0.008
249	-0.903	0.135	0.135	0.135	0.068	0.069	0.069	0.919	0.134	0.134	0.135	0.037	0.038	0.039
250	-0.796	0.083	0.084	0.083	0.038	0.039	0.039	1.015	0.112	0.113	0.114	0.052	0.053	0.054

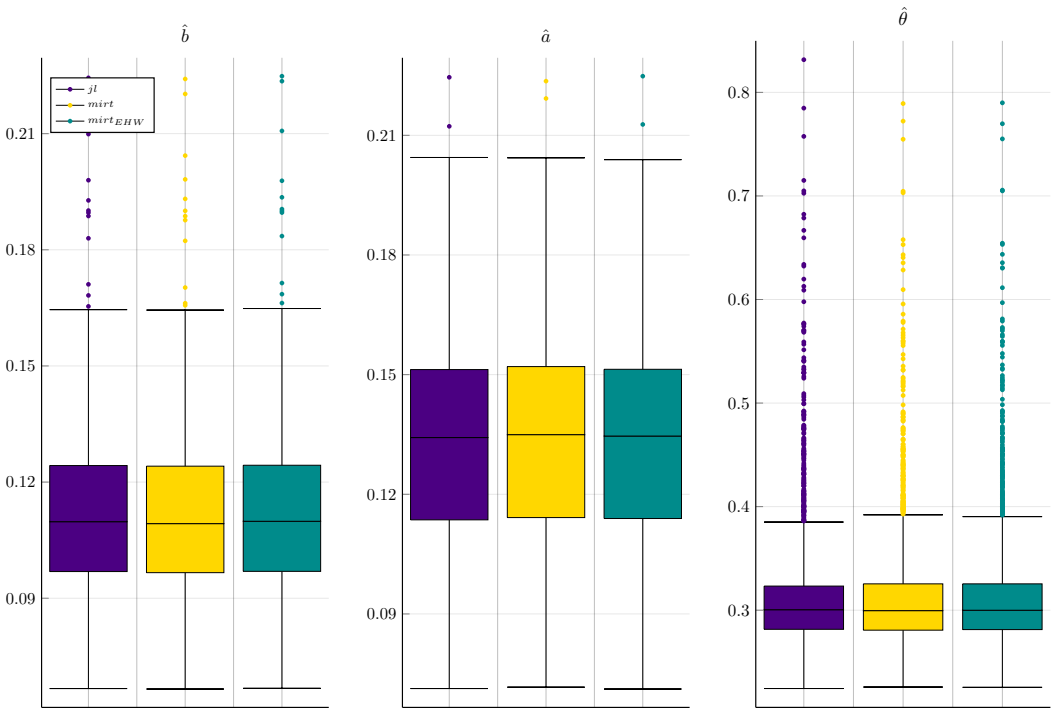


FIGURE A.4: Case 1a - Boxplots of RMSEs.

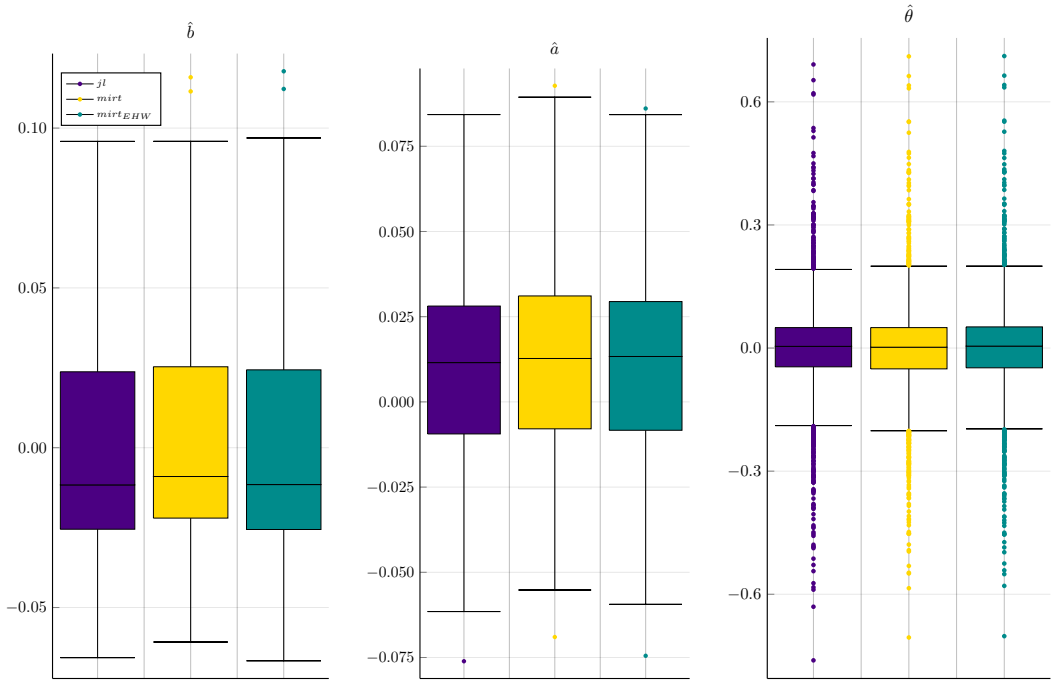


FIGURE A.5: Case 1a - Boxplots of BIASs.

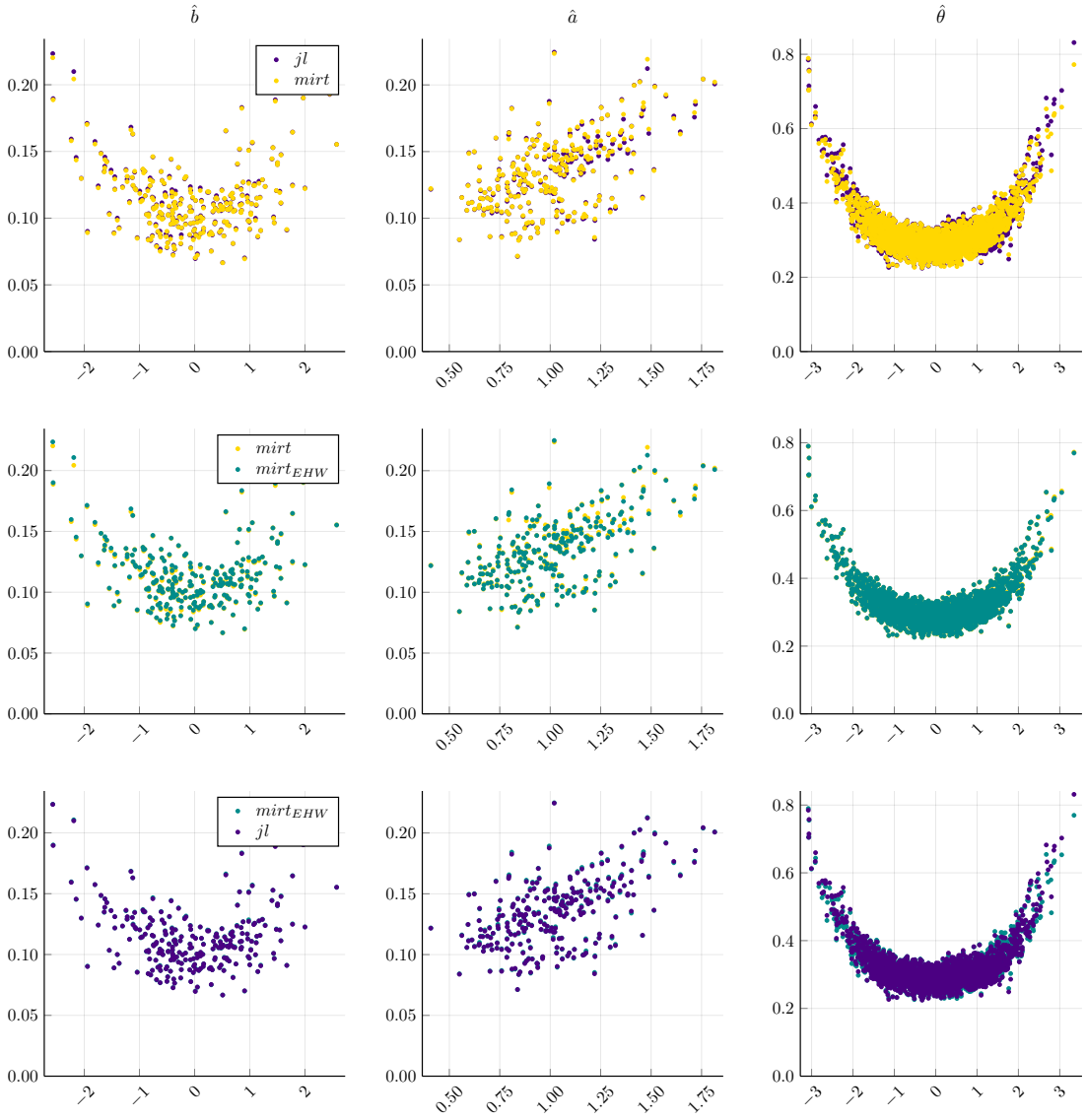


FIGURE A.6: Case 1a - Scatter plots of RMSEs.

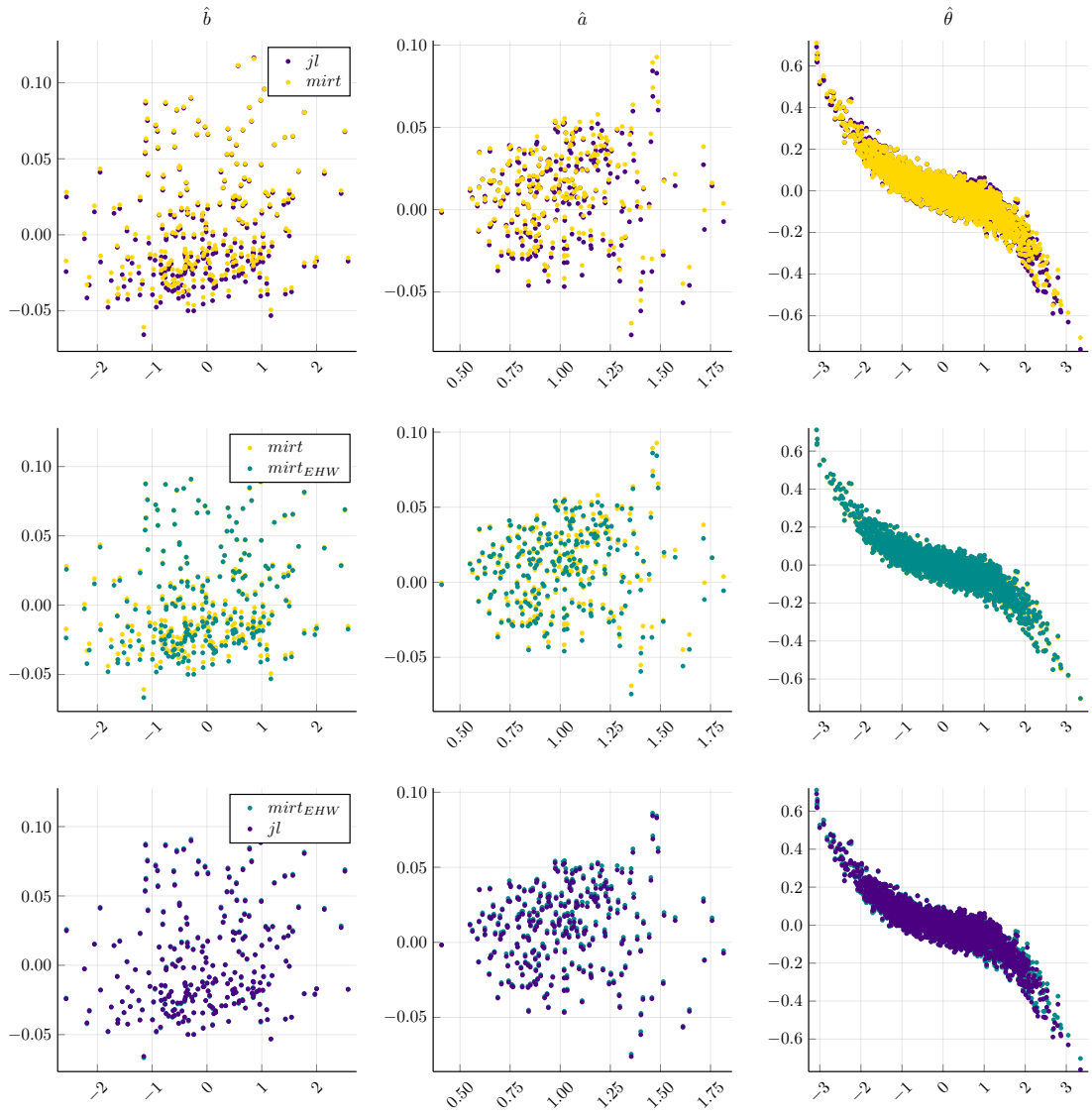


FIGURE A.7: Case 1a - Scatter plots of BIASs

A.2.2 Case 1b

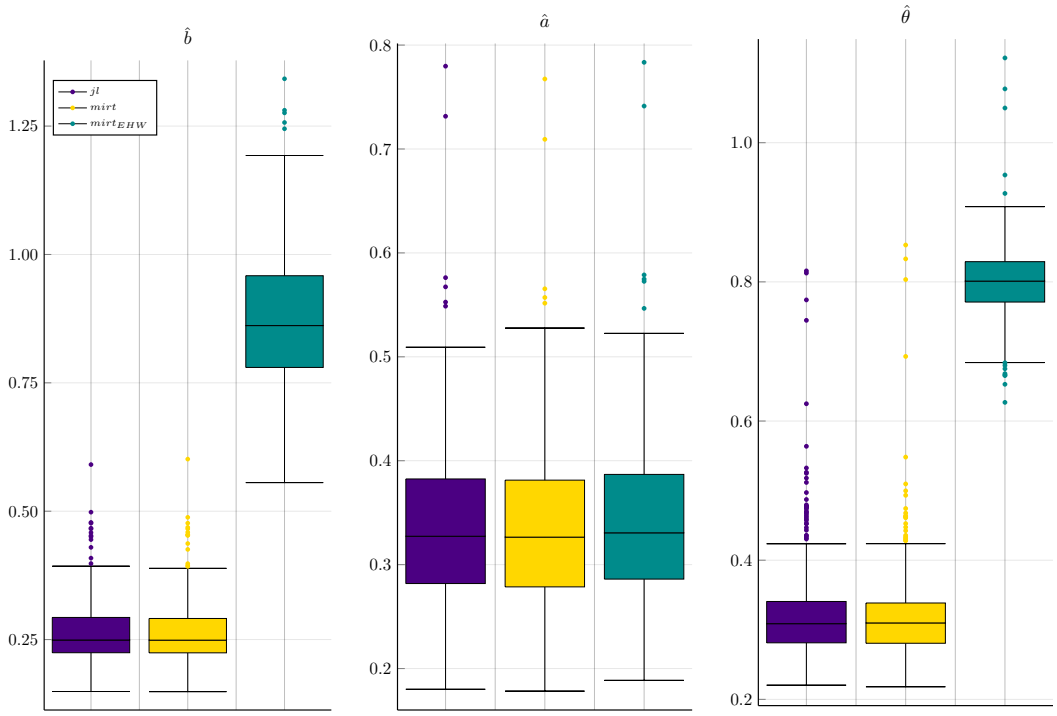


FIGURE A.8: Case 1b - Boxplots of RMSEs.

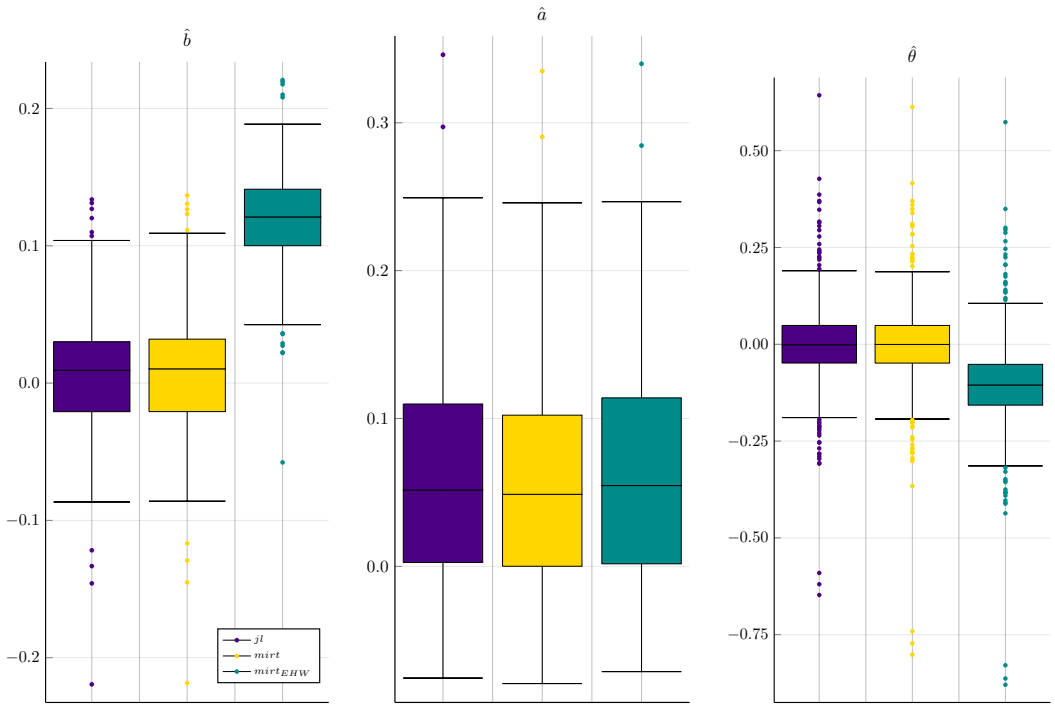


FIGURE A.9: Case 1b - Boxplots of BIASs.

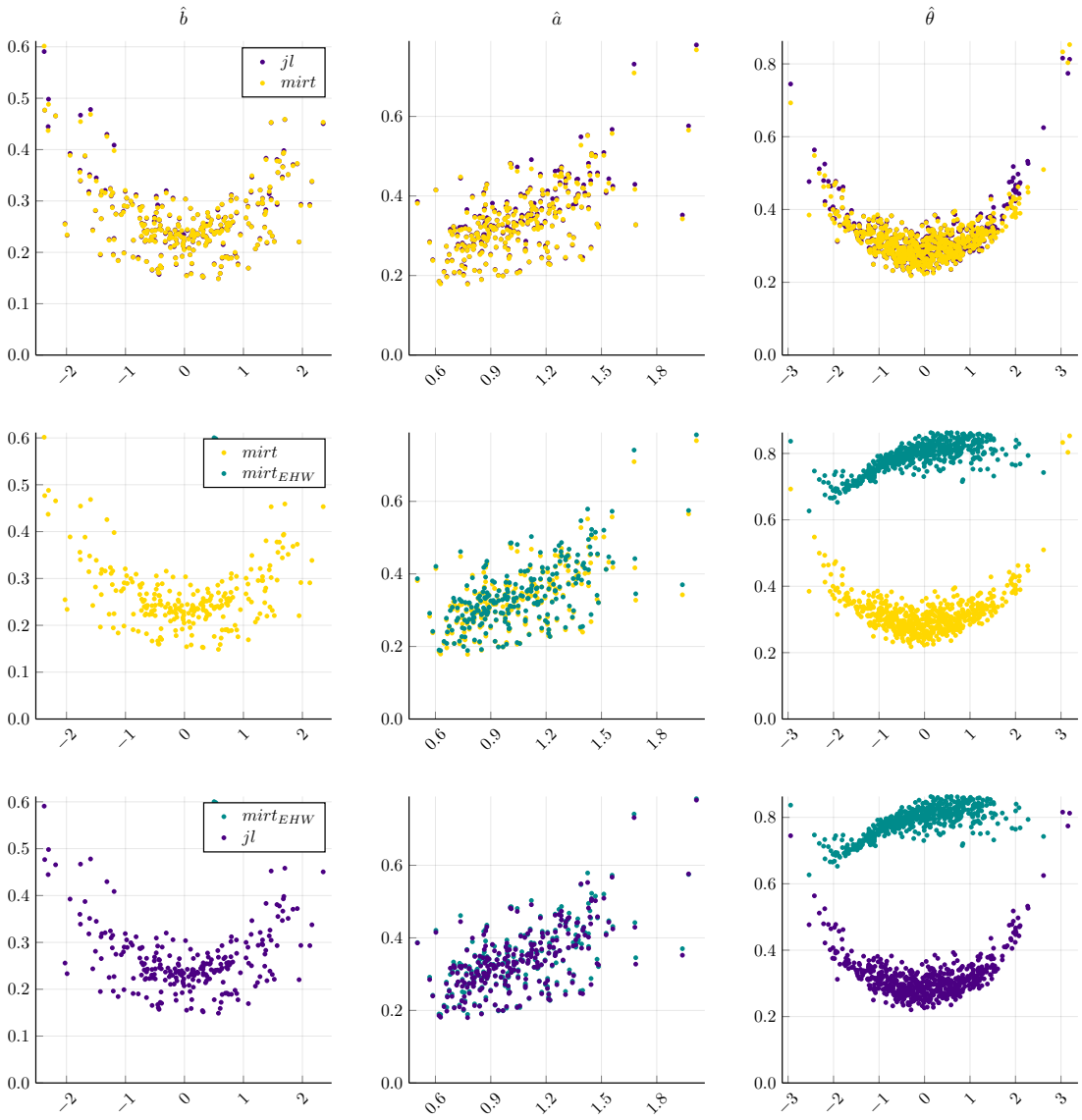


FIGURE A.10: Case 1b - Scatter plots of RMSEs.

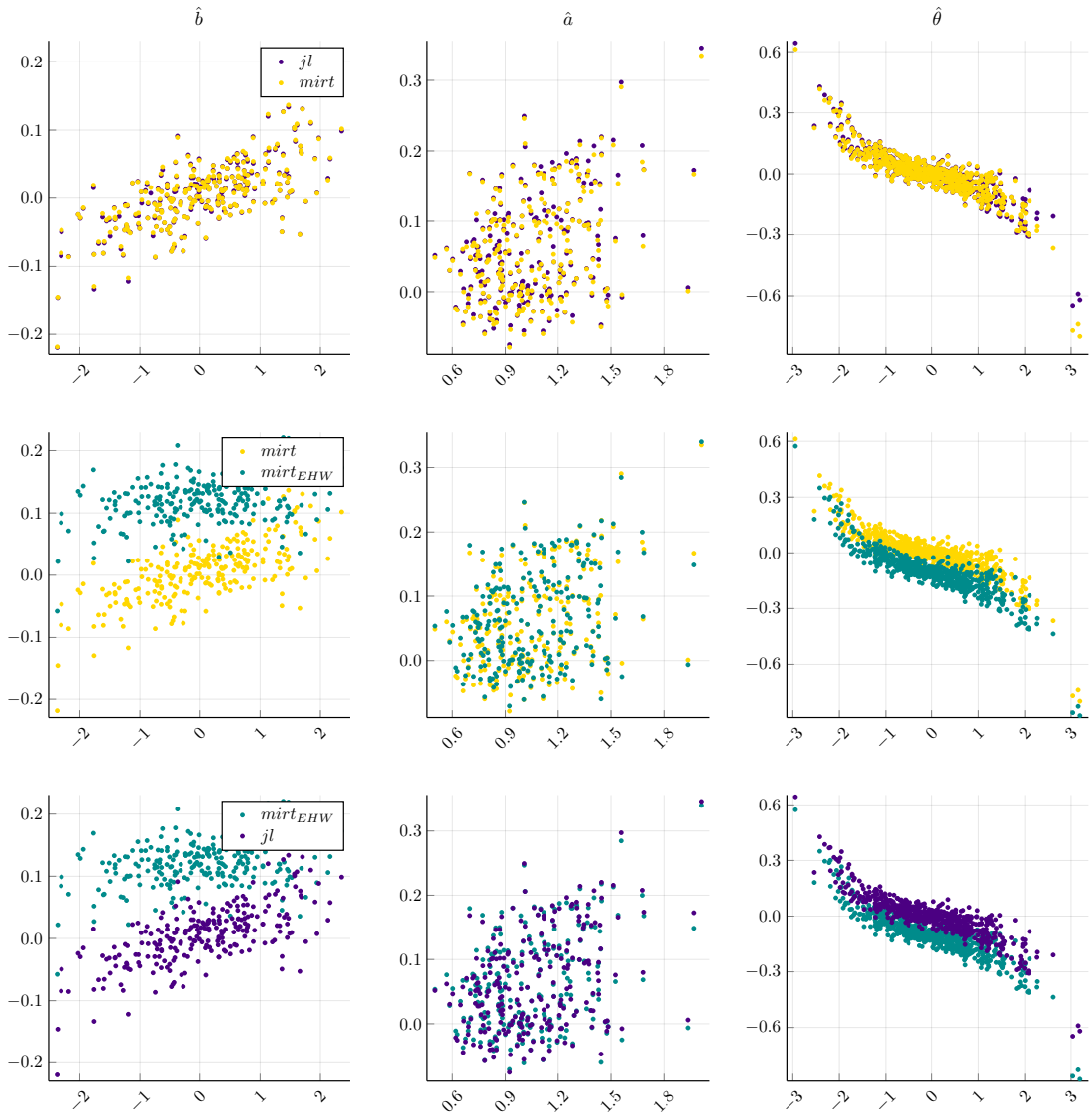


FIGURE A.11: Case 1b - Scatter plots of BIASs

A.2.3 Case 1c

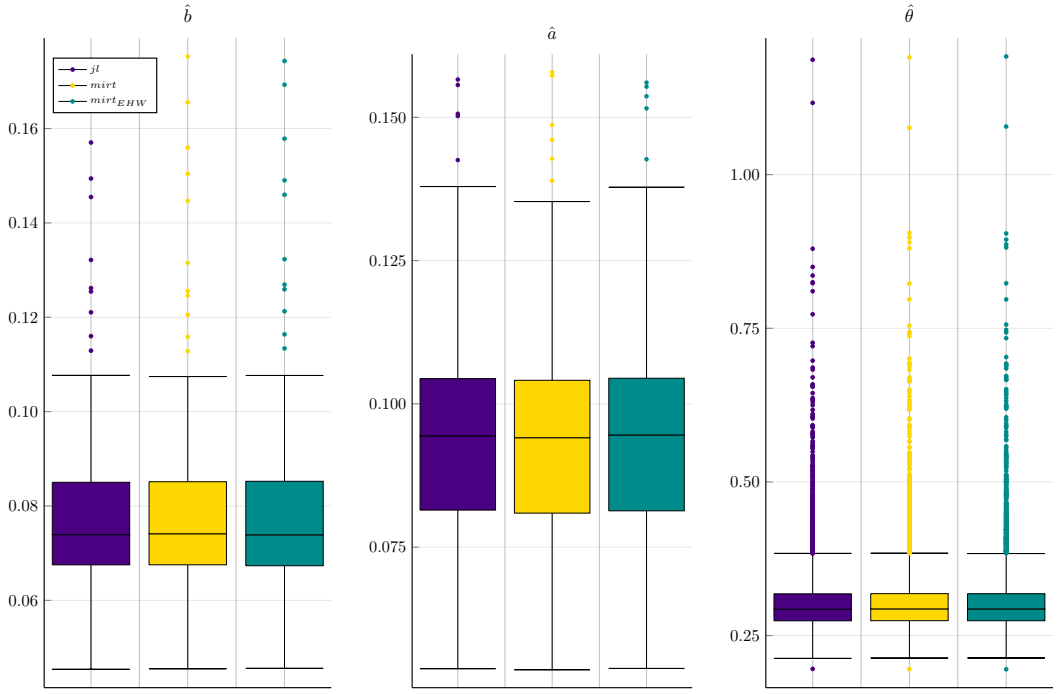


FIGURE A.12: Case 1c - Boxplots of RMSEs.

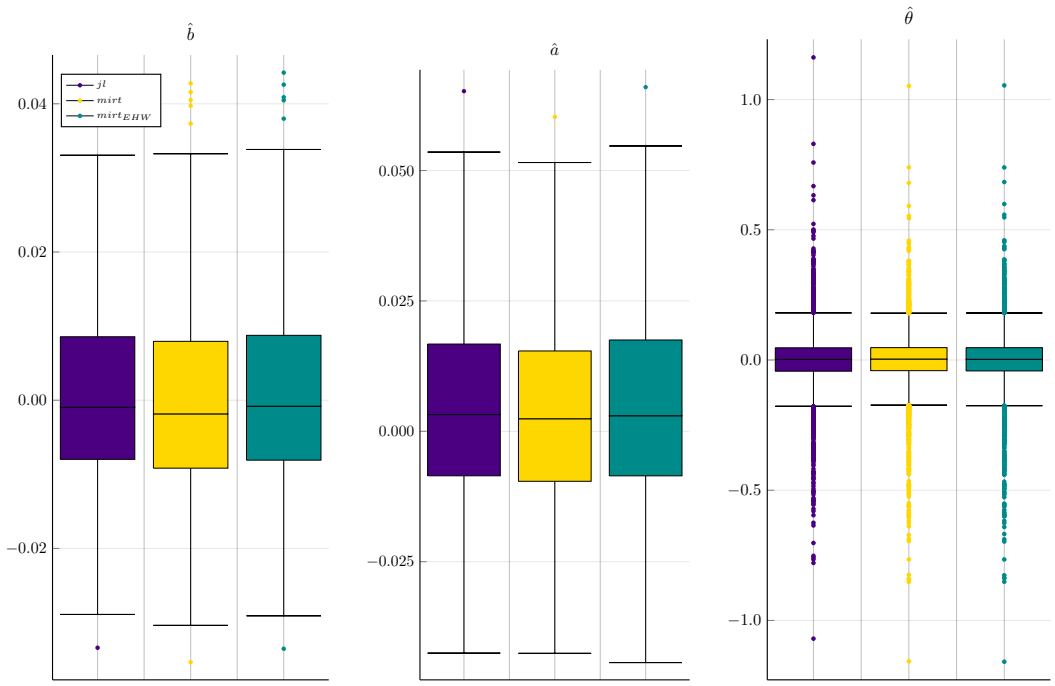


FIGURE A.13: Case 1c - Boxplots of BIASs.

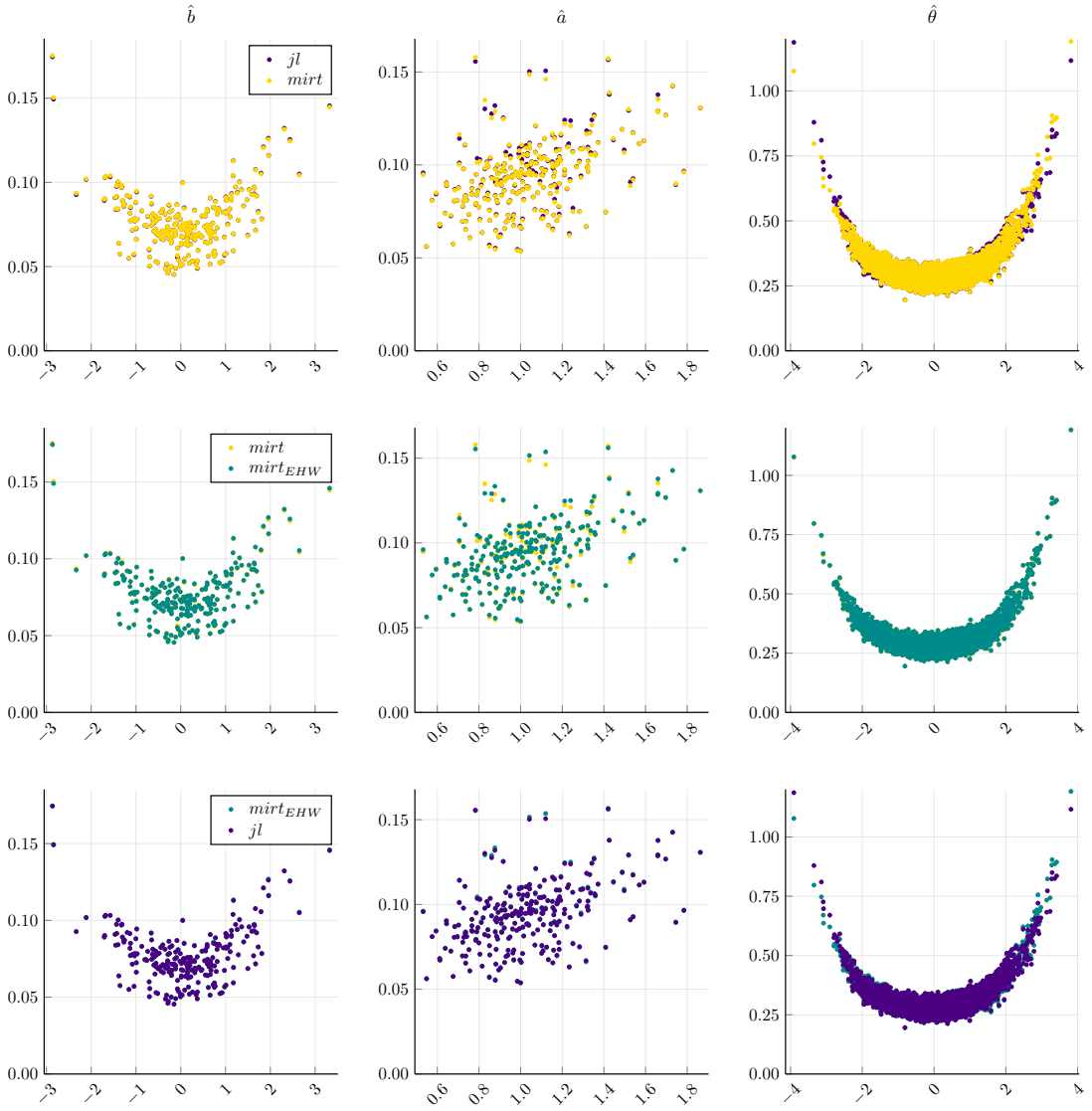


FIGURE A.14: Case 1c - Scatter plots of RMSEs.

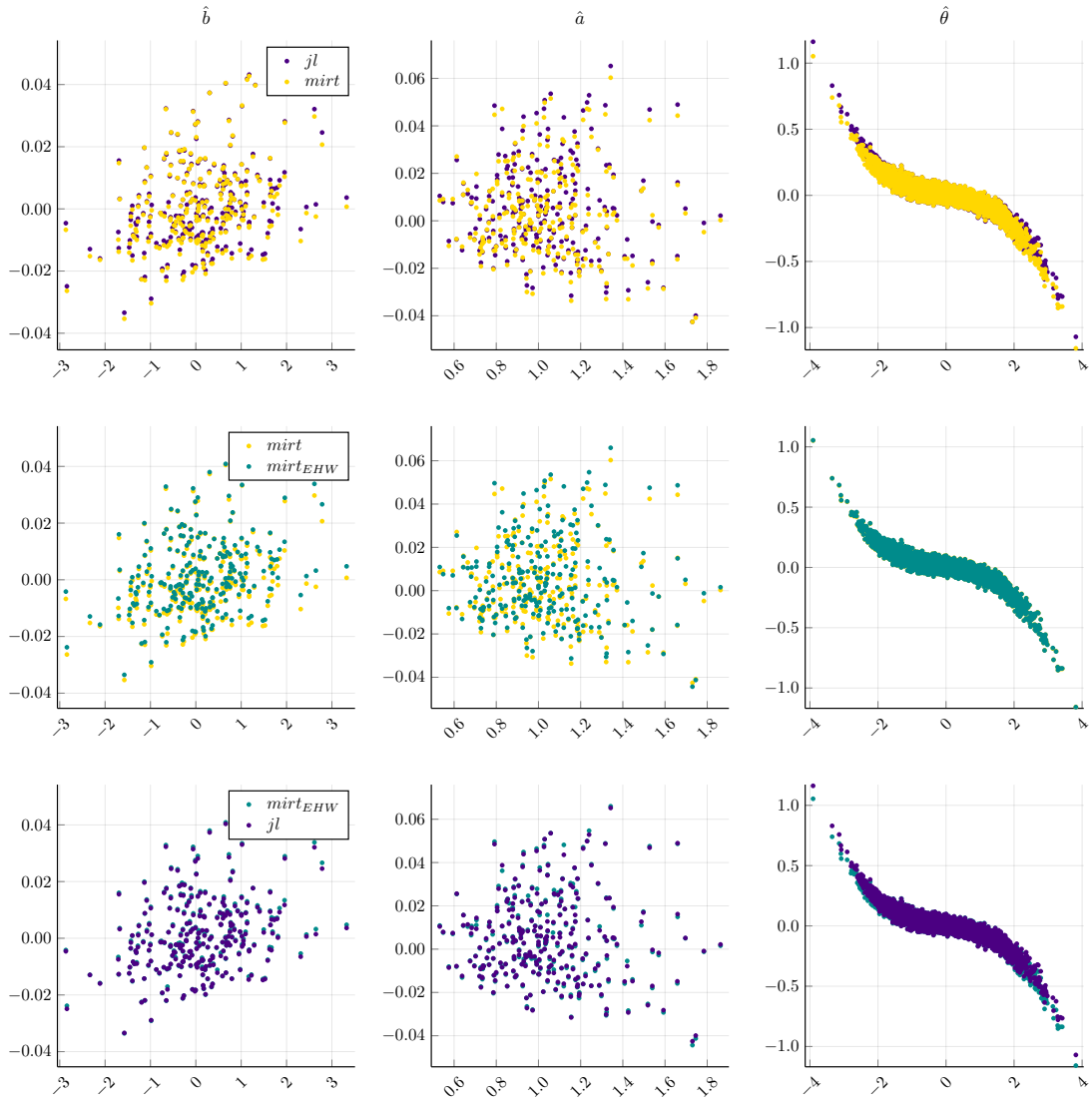


FIGURE A.15: Case 1c - Scatter plots of BIASs

A.2.4 Case 2a

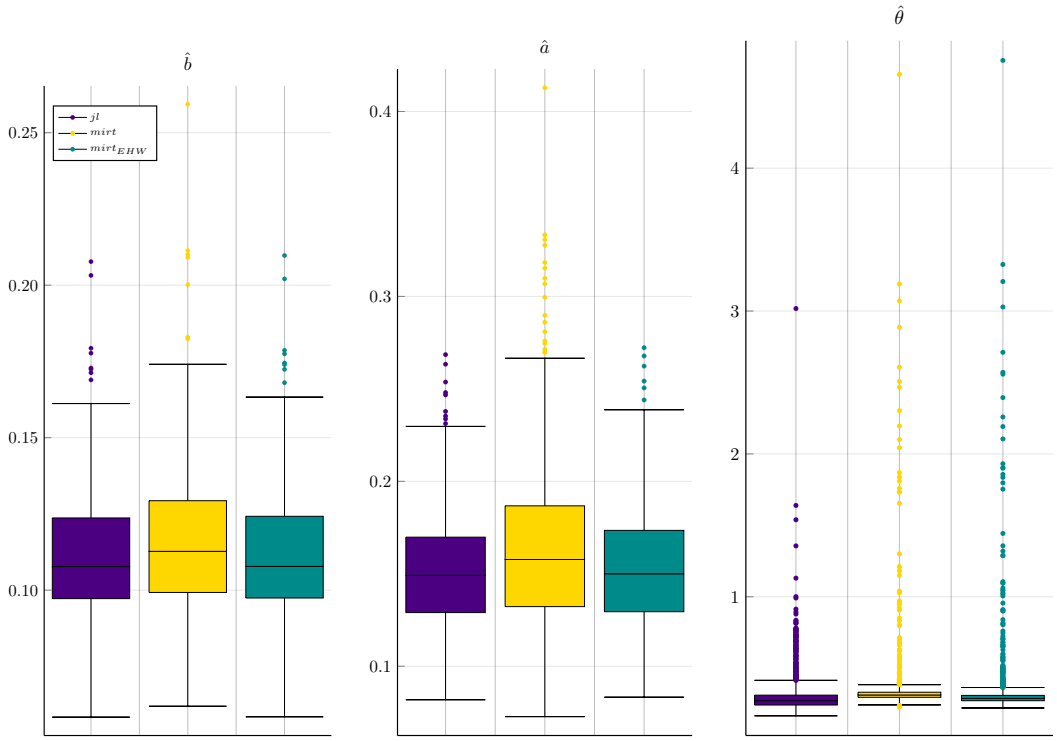


FIGURE A.16: Case 2a - Boxplots of RMSEs.

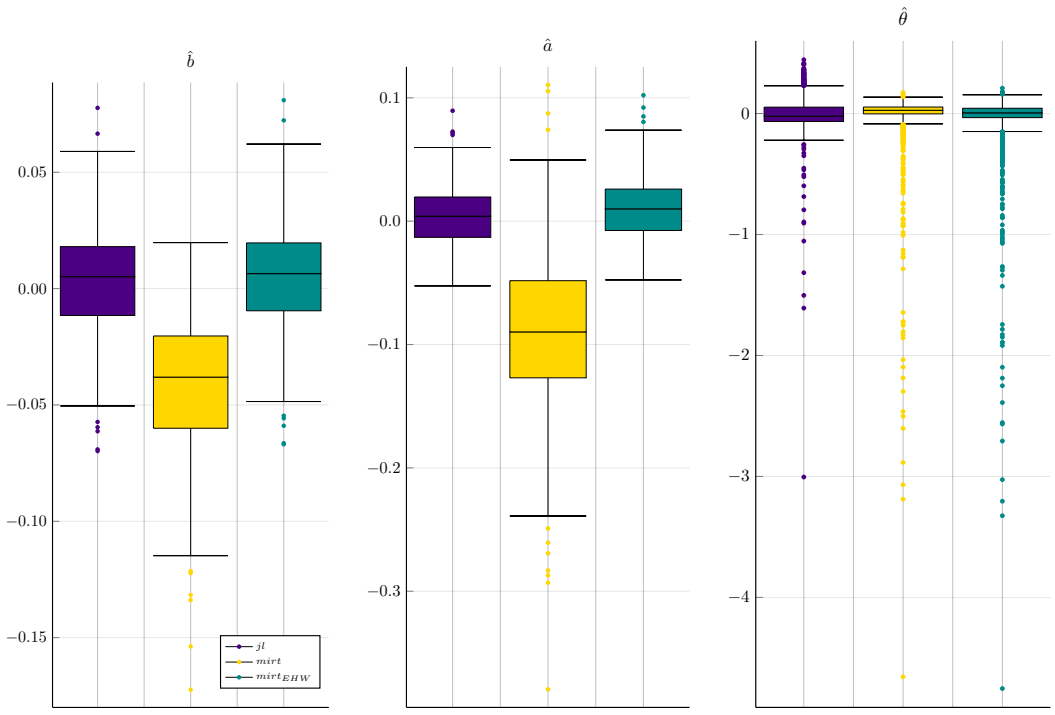


FIGURE A.17: Case 2a - Boxplots of BIASs.

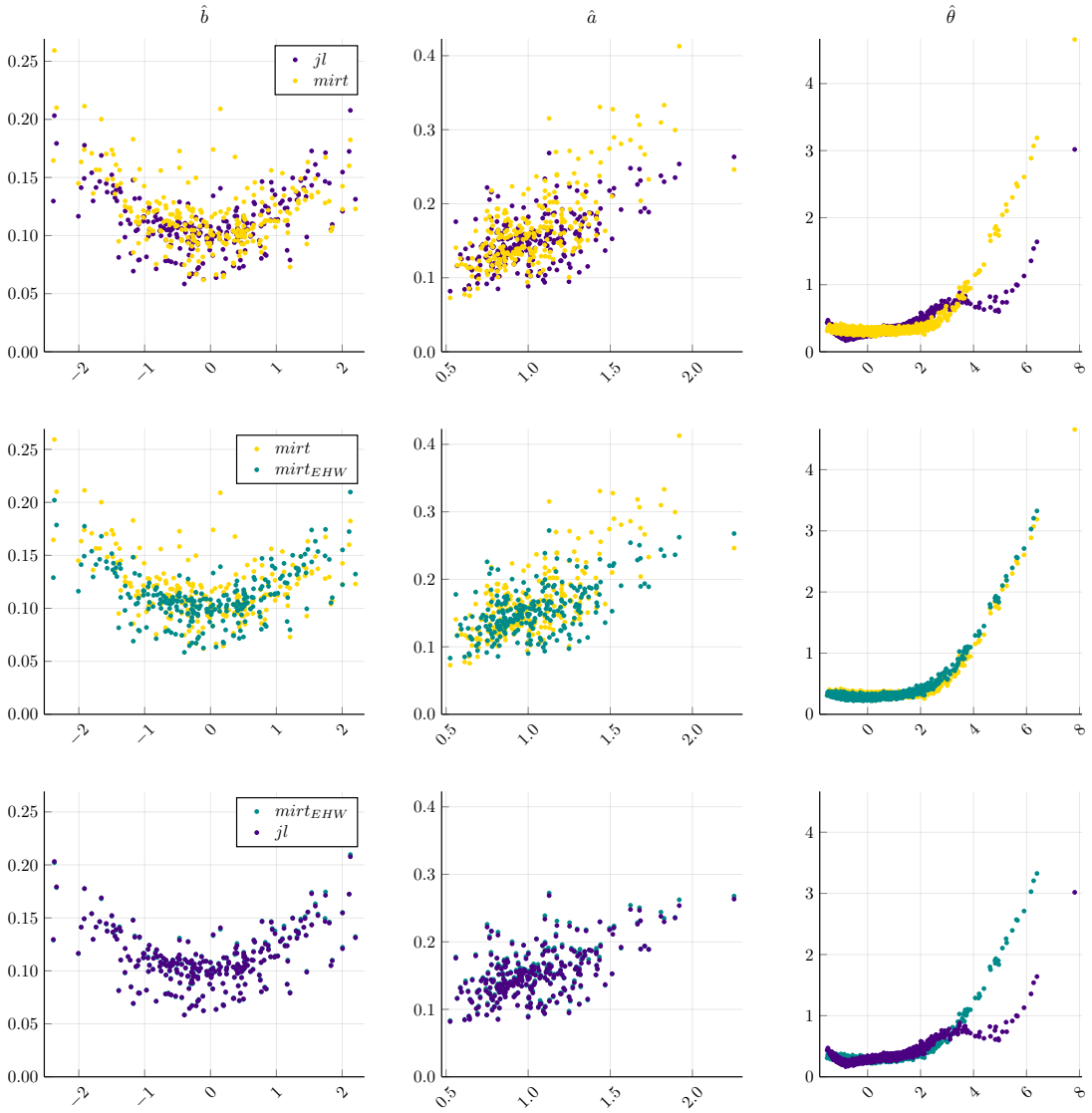


FIGURE A.18: Case 2a - Scatter plots of RMSEs.

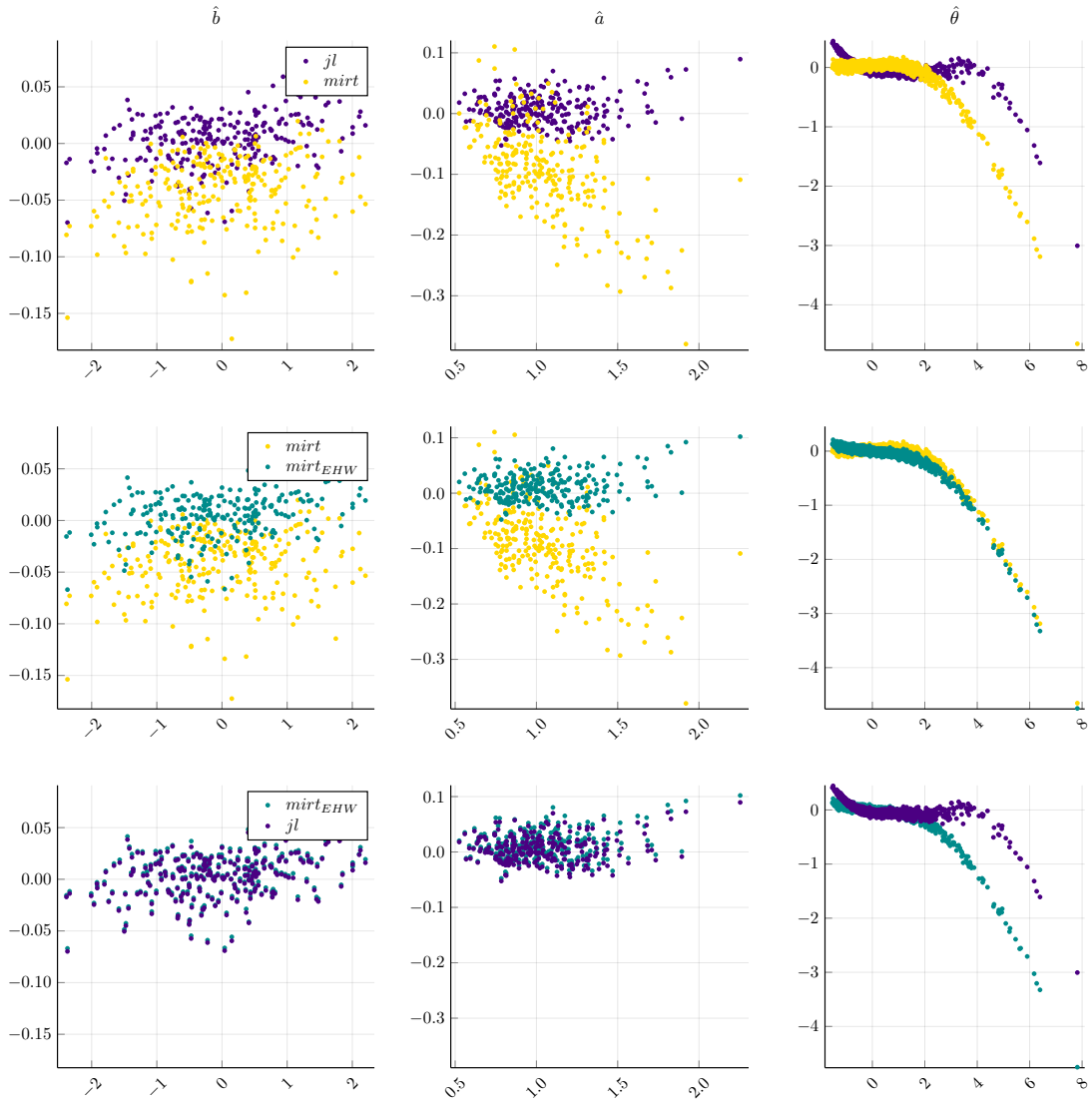


FIGURE A.19: Case 2a - Scatter plots of BIASs

A.2.5 Case 2b

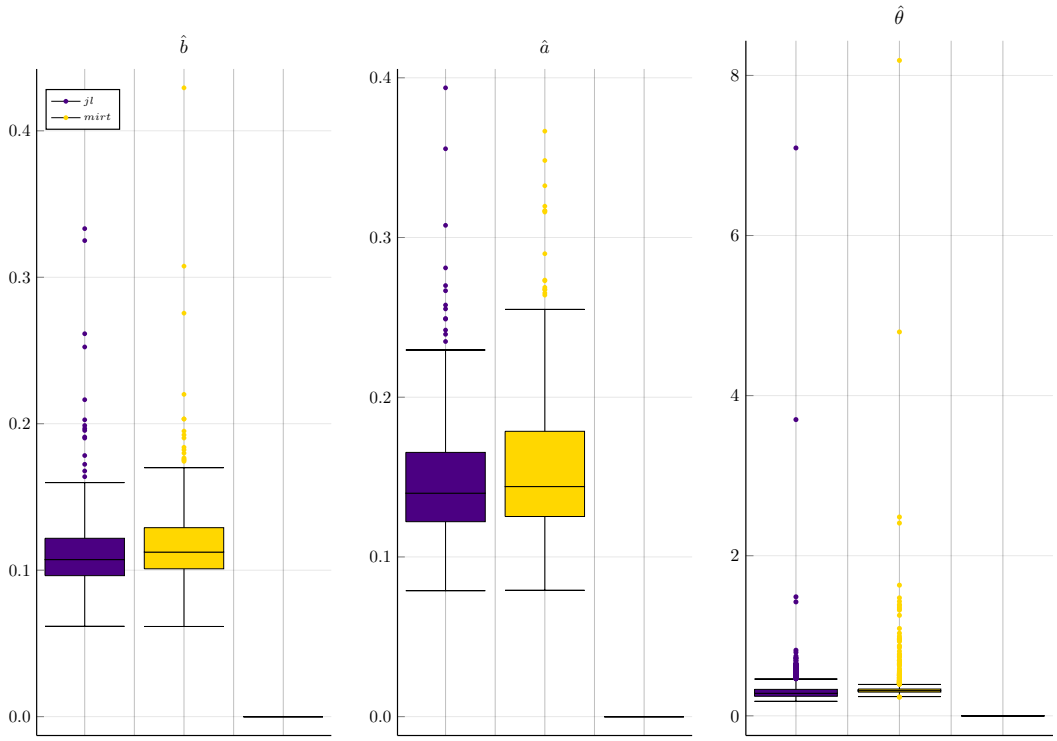


FIGURE A.20: Case 2b - Boxplots of RMSEs.

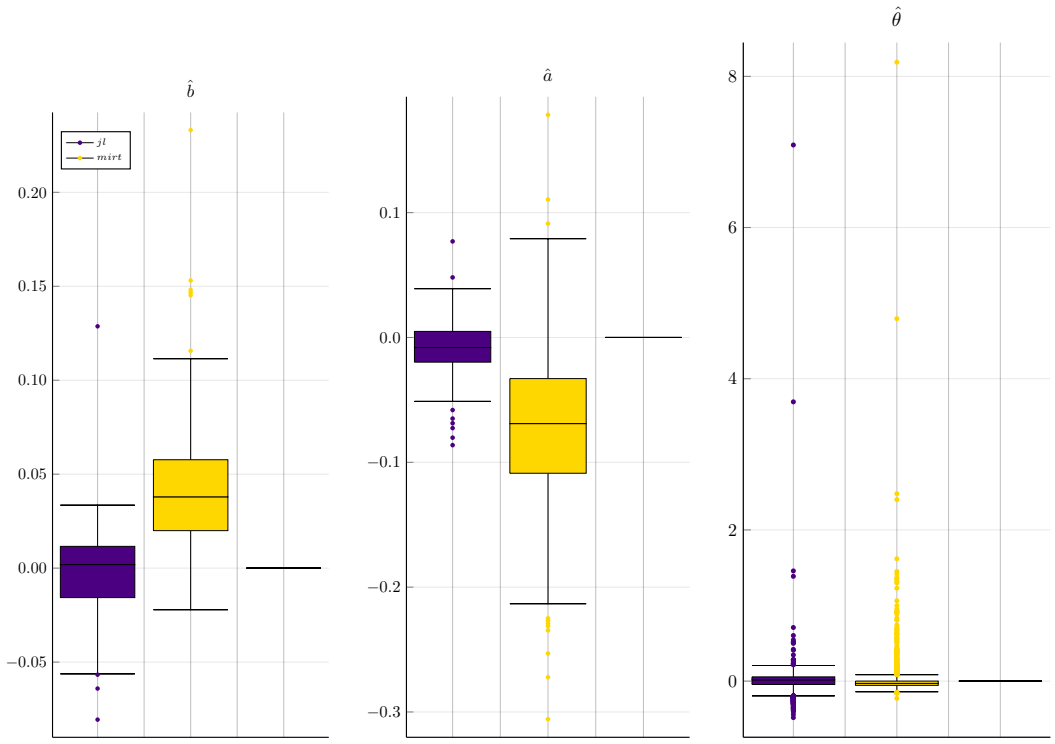


FIGURE A.21: Case 2b - Boxplots of BIASs.

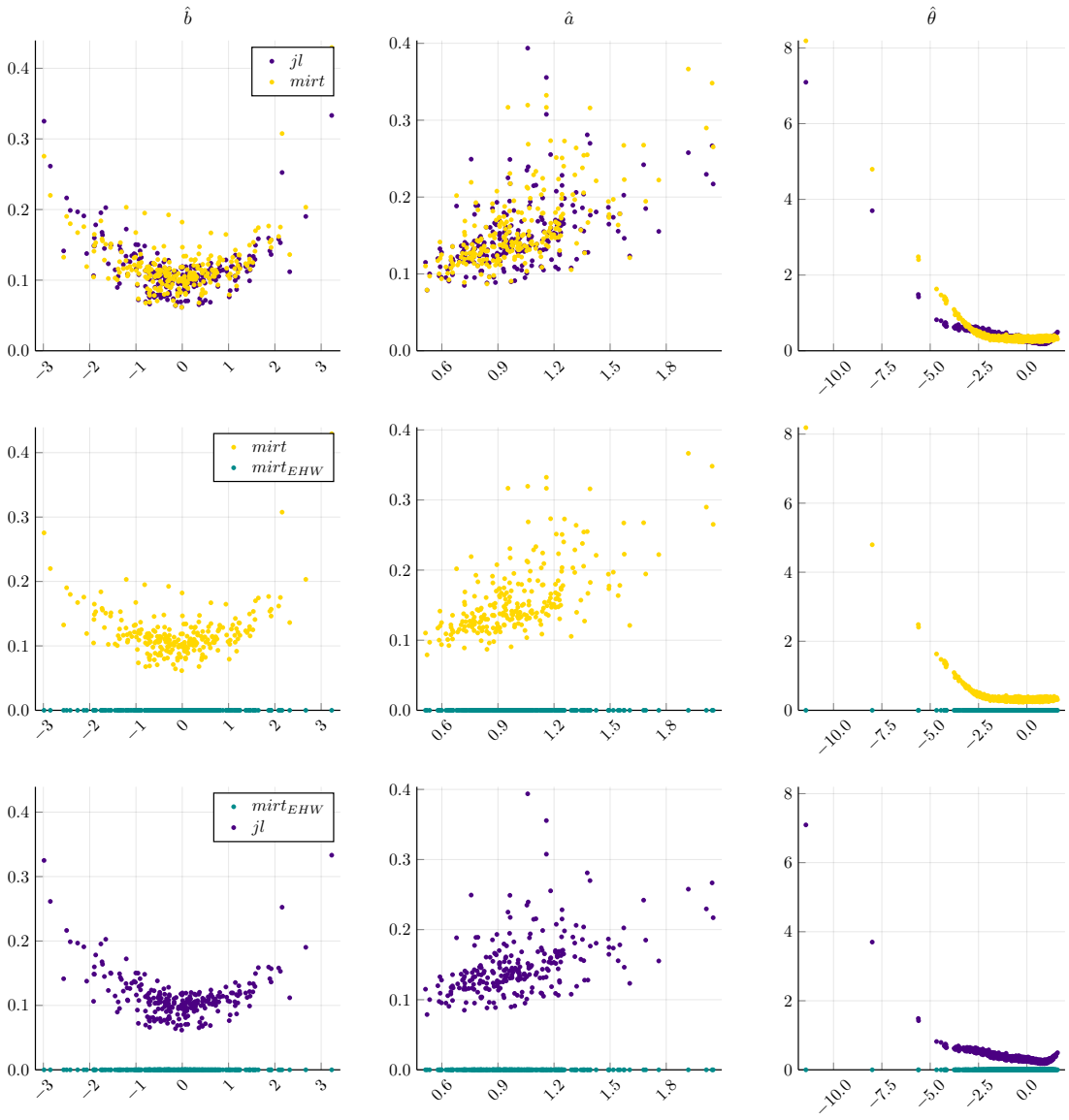


FIGURE A.22: Case 2b - Scatter plots of RMSEs.

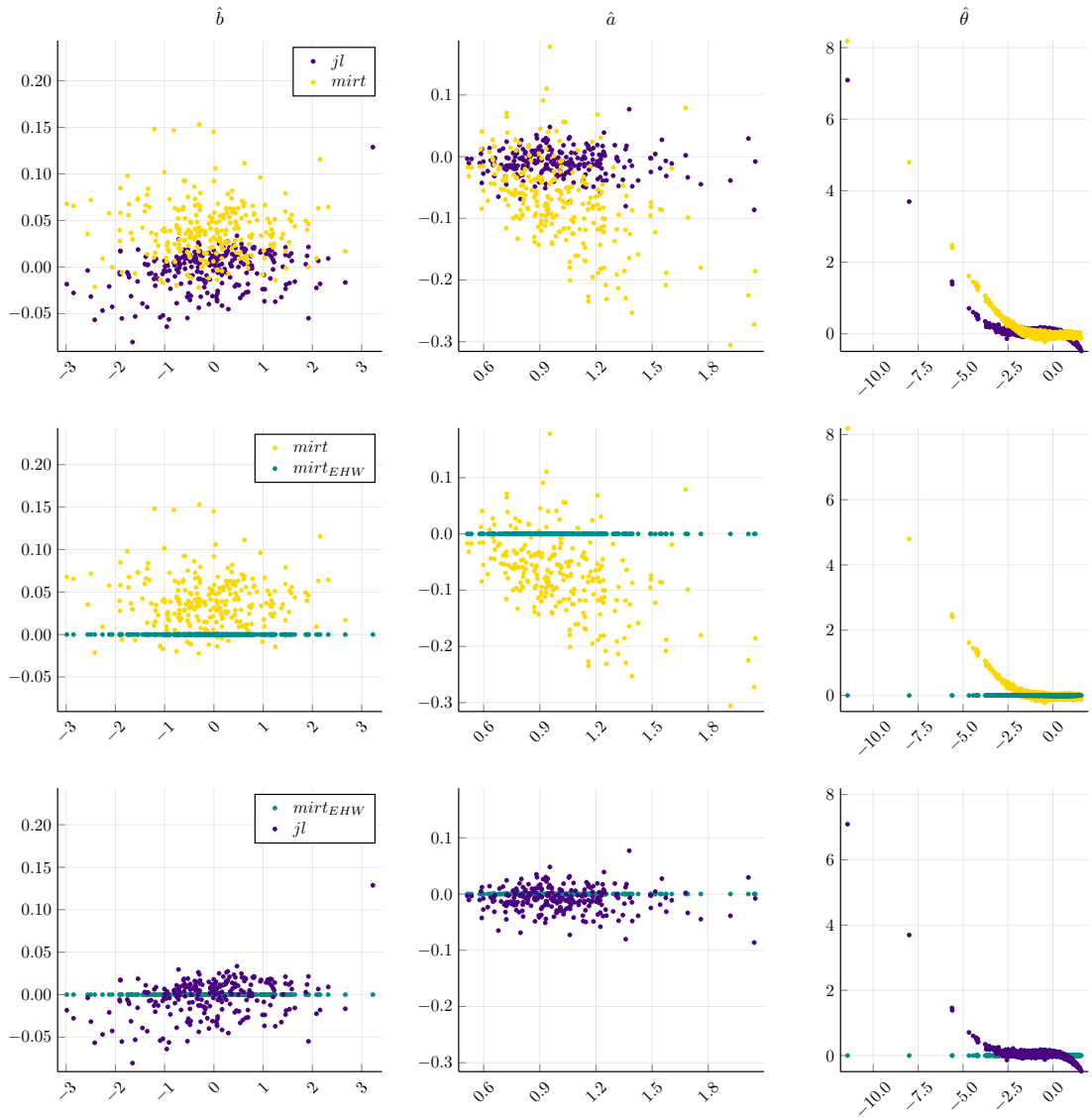


FIGURE A.23: Case 2b - Scatter plots of BIASs

A.2.6 Case 3

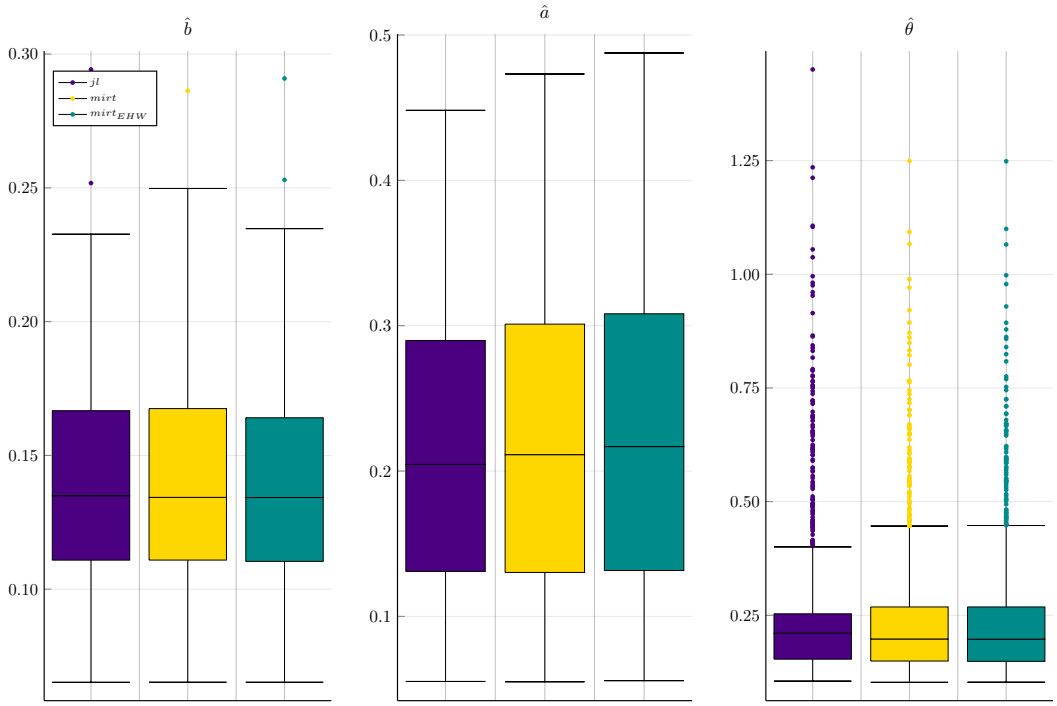


FIGURE A.24: Case 3 - Boxplots of RMSEs.

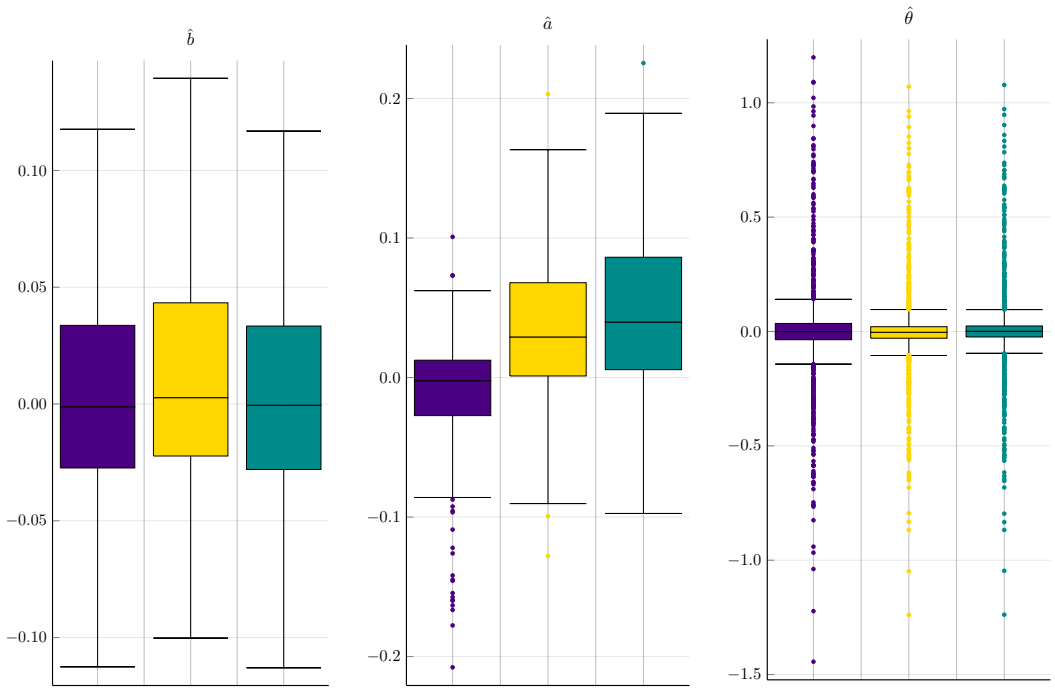


FIGURE A.25: Case 3 - Boxplots of BIASs.

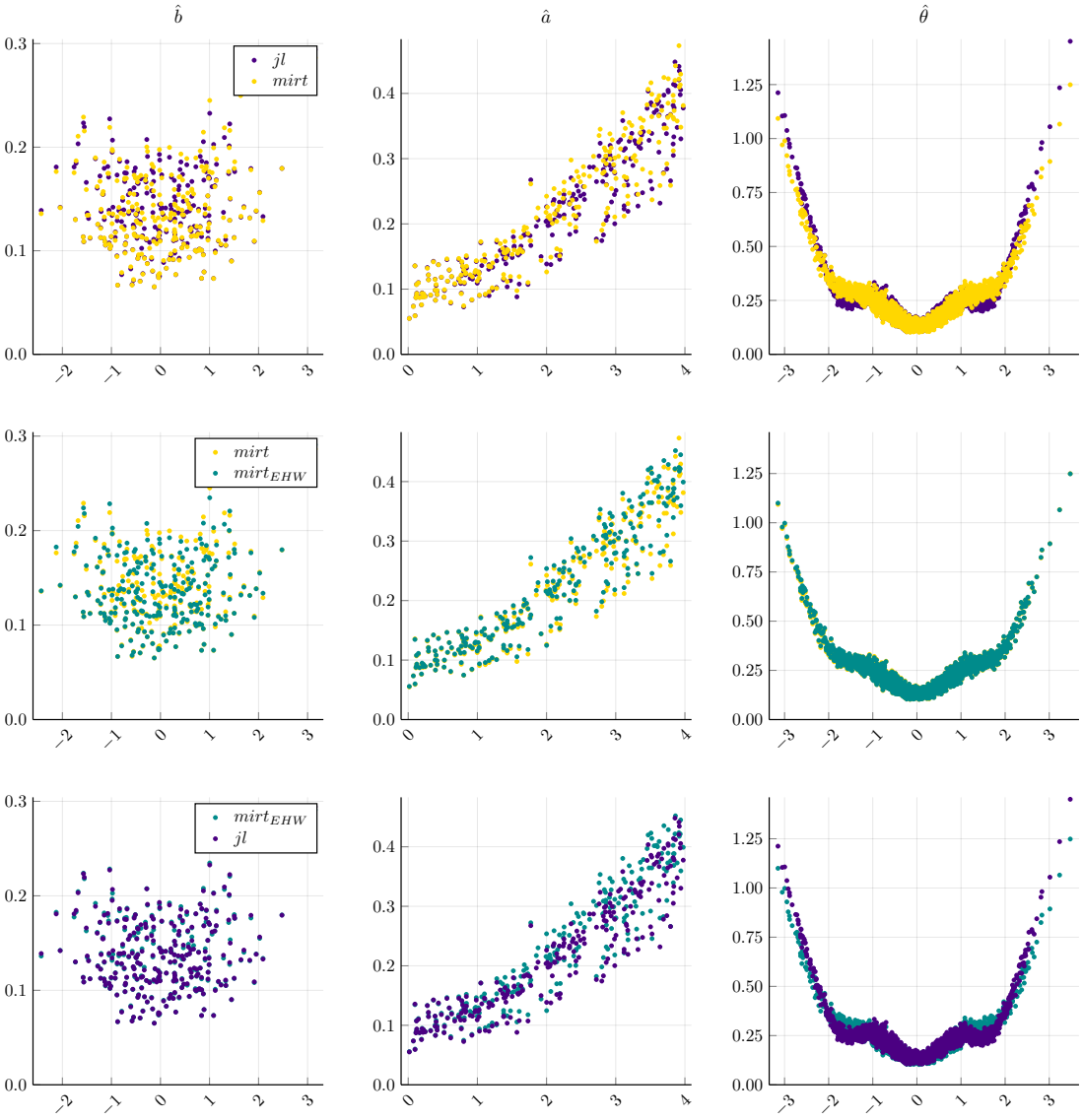


FIGURE A.26: Case 3 - Scatter plots of RMSEs.

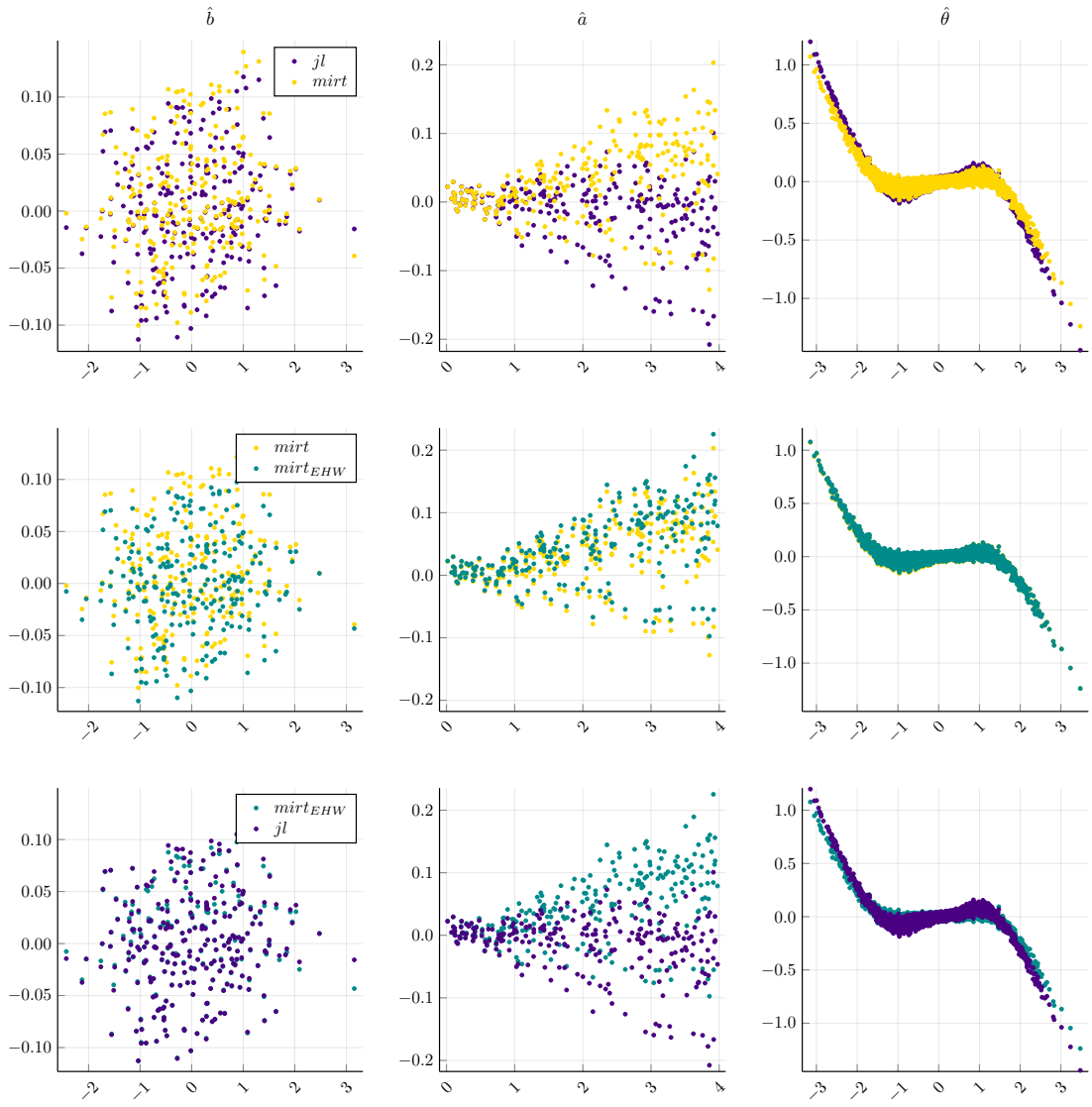


FIGURE A.27: Case 3 - Scatter plots of BIASs

A.2.7 Case 4

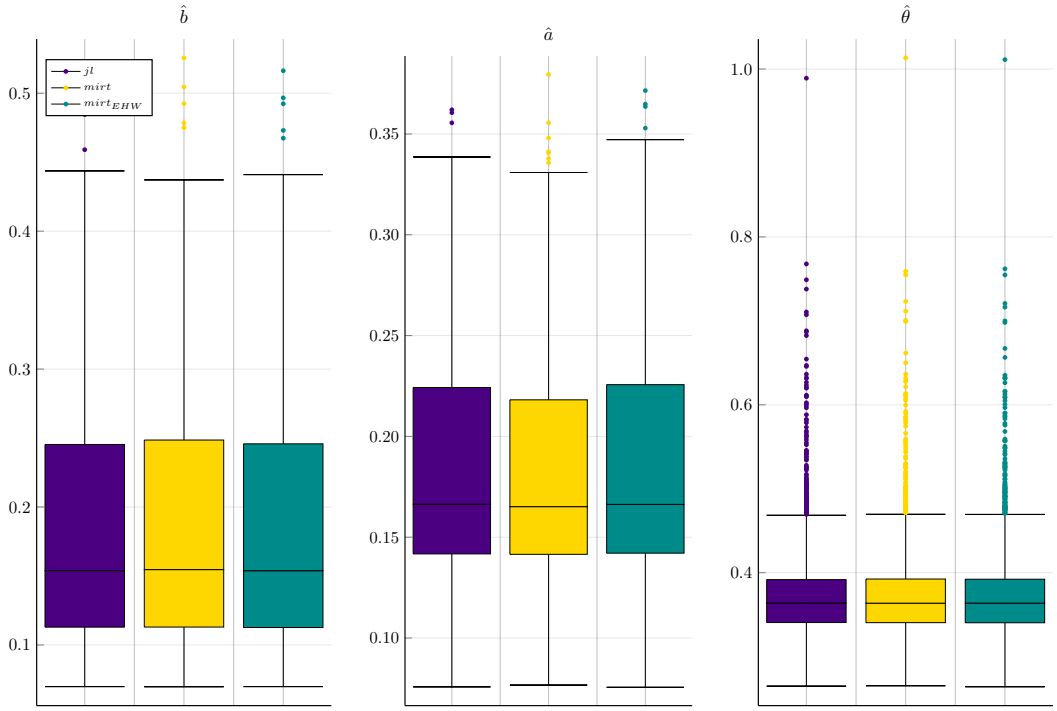


FIGURE A.28: Case 4 - Boxplots of RMSEs.

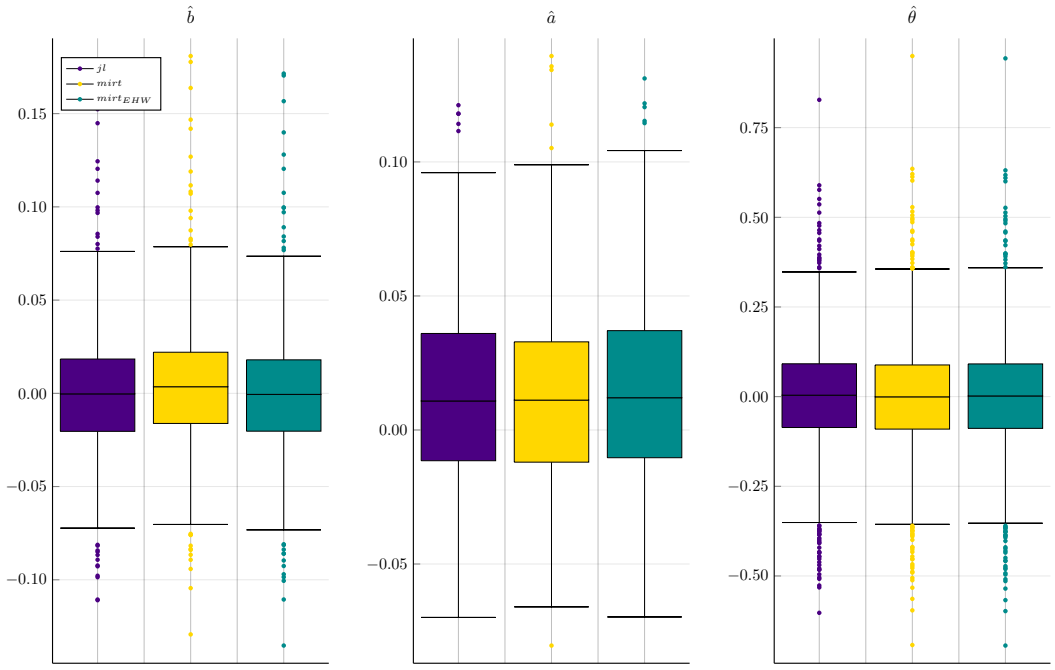


FIGURE A.29: Case 4 - Boxplots of BIASs.

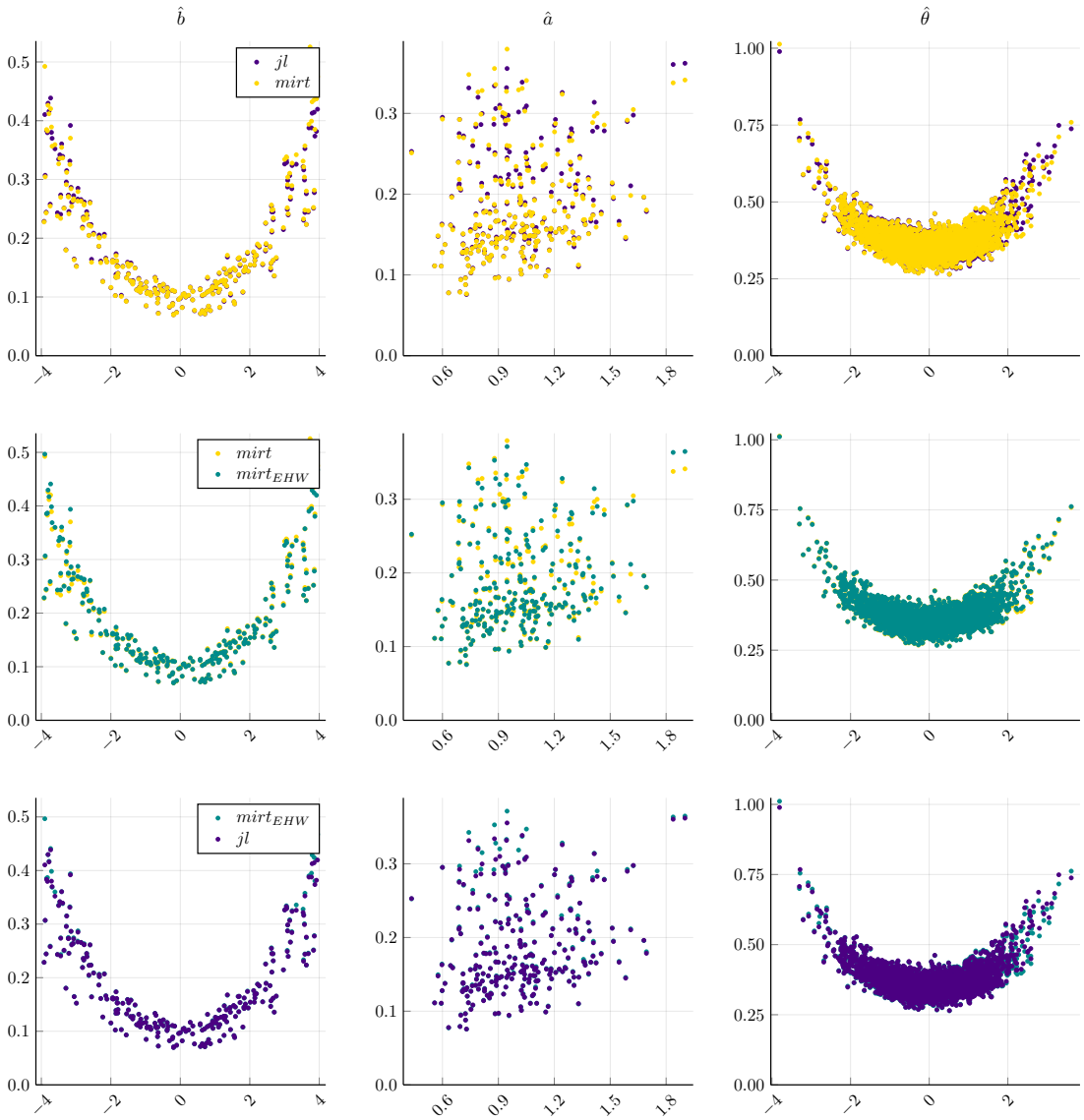


FIGURE A.30: Case 4 - Scatter plots of RMSEs.

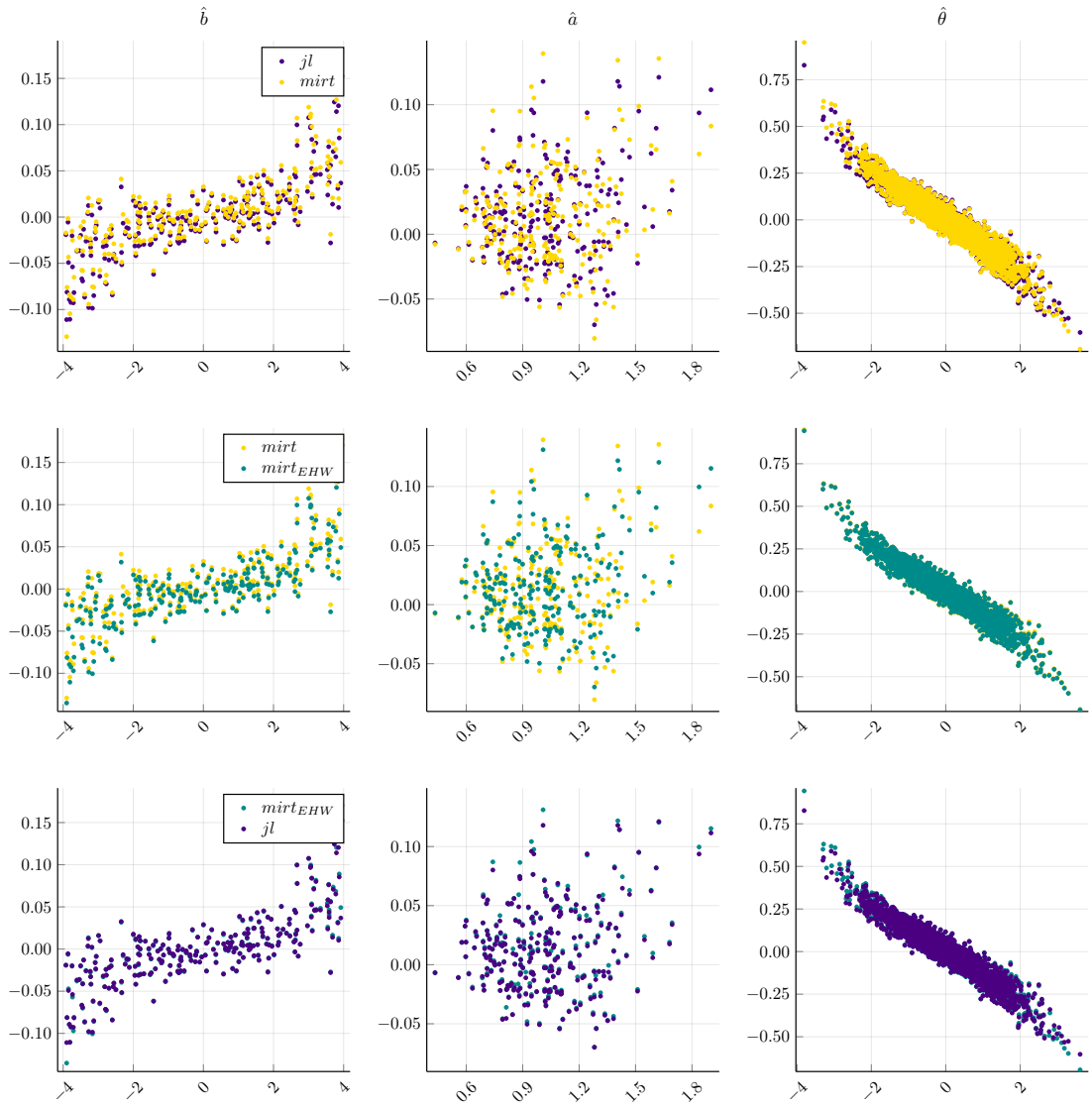


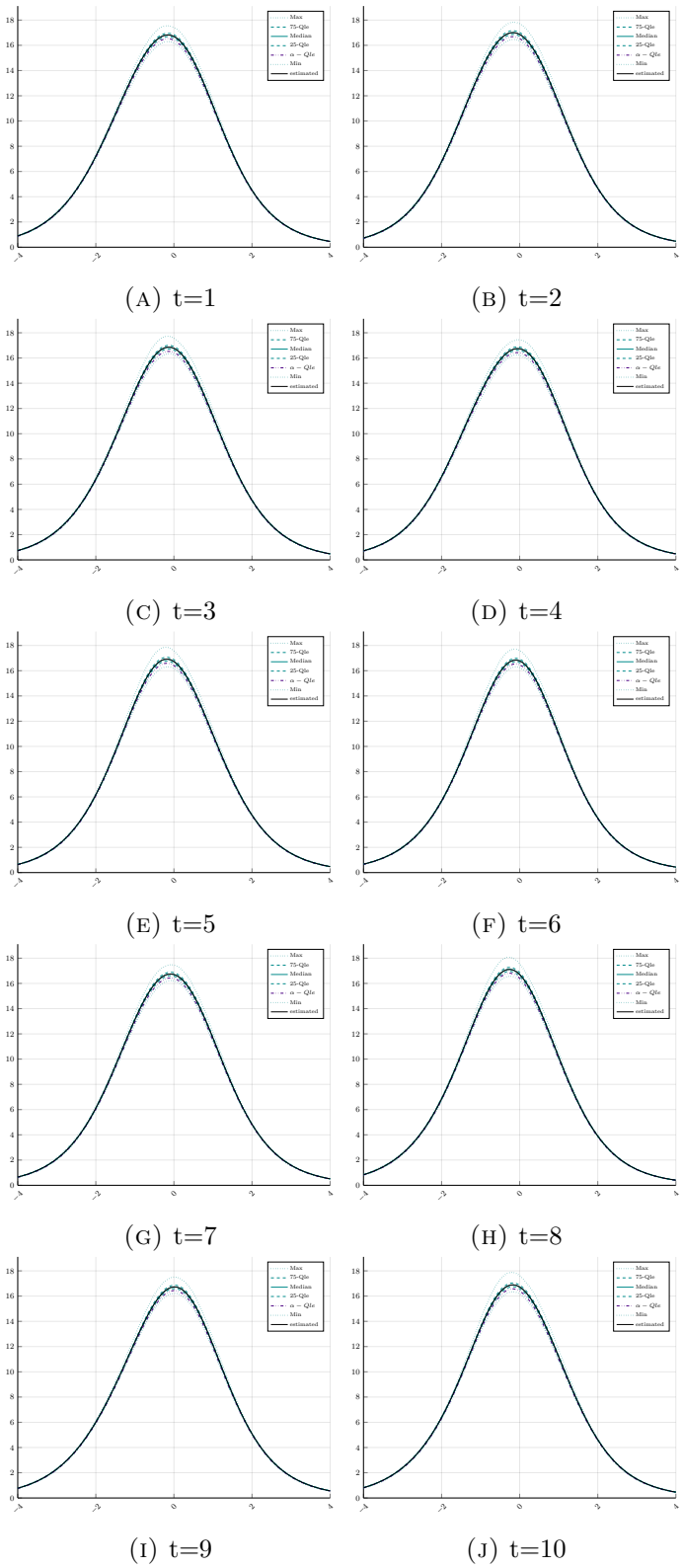
FIGURE A.31: Case 4 - Scatter plots of BIASs

A.3 Chance-constrained test assembly

A.3.1 Application on real data

TABLE A.7: Structure of assembled tests.

test	length	DOMAIN[1]	overlap[2]
1	46	{ 12;12;12;10 }	{46;10;10;11;11;10;11;10;11;11;10;10;08;11;10;11;11;10;11;11}
2	45	{ 12;12;12;09 }	{10;45;11;11;10;11;10;11;11;11;11;11;10;11;10;11;11;11;11;11}
3	45	{ 11;12;12;09 }	{10;11;45;11;11;10;11;11;11;11;11;10;10;11;10;10;09;11;10;11}
4	45	{ 12;12;12;09 }	{11;11;11;45;11;11;11;11;10;09;11;11;11;11;11;11;10;09;11;08}
5	44	{ 12;12;12;08 }	{11;10;11;11;44;11;10;11;10;11;10;09;10;11;11;11;11;10;11;11}
6	43	{ 12;12;12;07 }	{10;11;10;11;11;43;11;11;11;10;08;11;11;11;11;11;10;11;10;10}
7	45	{ 12;12;12;09 }	{11;10;11;11;10;11;45;10;11;11;11;11;08;11;11;11;10;11;11;11}
8	44	{ 12;12;12;08 }	{10;11;11;11;11;11;10;44;11;11;09;11;11;09;09;11;11;11;10;11}
9	46	{ 12;11;10;13 }	{11;11;11;10;10;11;11;11;46;10;11;10;11;11;10;10;11;09;11;11}
10	45	{ 12;11;12;10 }	{11;11;11;09;11;10;11;11;10;45;11;11;11;09;11;10;09;11;11;10}
11	44	{ 12;12;12;08 }	{10;11;11;11;10;08;11;09;11;11;44;10;10;10;07;08;11;11;08;11}
12	45	{ 12;12;12;09 }	{10;11;10;11;09;11;11;11;10;11;10;45;11;11;11;11;11;11;10;09}
13	45	{ 12;11;11;11 }	{08;10;10;11;10;11;08;11;11;11;10;11;45;11;09;11;10;11;11;11}
14	45	{ 12;12;12;09 }	{11;11;11;11;11;11;11;09;11;09;10;11;11;45;08;11;11;08;10;10}
15	45	{ 12;12;12;10 }	{10;10;10;11;11;11;11;09;10;11;07;11;09;08;46;11;11;09;11;11}
16	46	{ 12;12;12;09 }	{11;11;10;11;11;11;11;11;10;10;08;11;11;11;11;45;11;11;11;05}
17	45	{ 12;12;12;09 }	{11;11;09;10;11;10;10;11;11;09;11;11;10;11;11;11;45;11;09;11}
18	45	{ 12;12;12;09 }	{10;11;11;09;10;11;11;11;09;11;11;11;11;08;09;11;11;45;11;11}
19	45	{ 12;12;12;09 }	{11;11;10;11;11;10;11;10;11;11;08;10;11;10;11;11;09;11;45;10}
20	45	{ 12;12;12;09 }	{11;11;11;08;11;10;11;11;11;10;11;09;11;10;11;05;11;11;10;45}



(A) t=1 (B) t=2

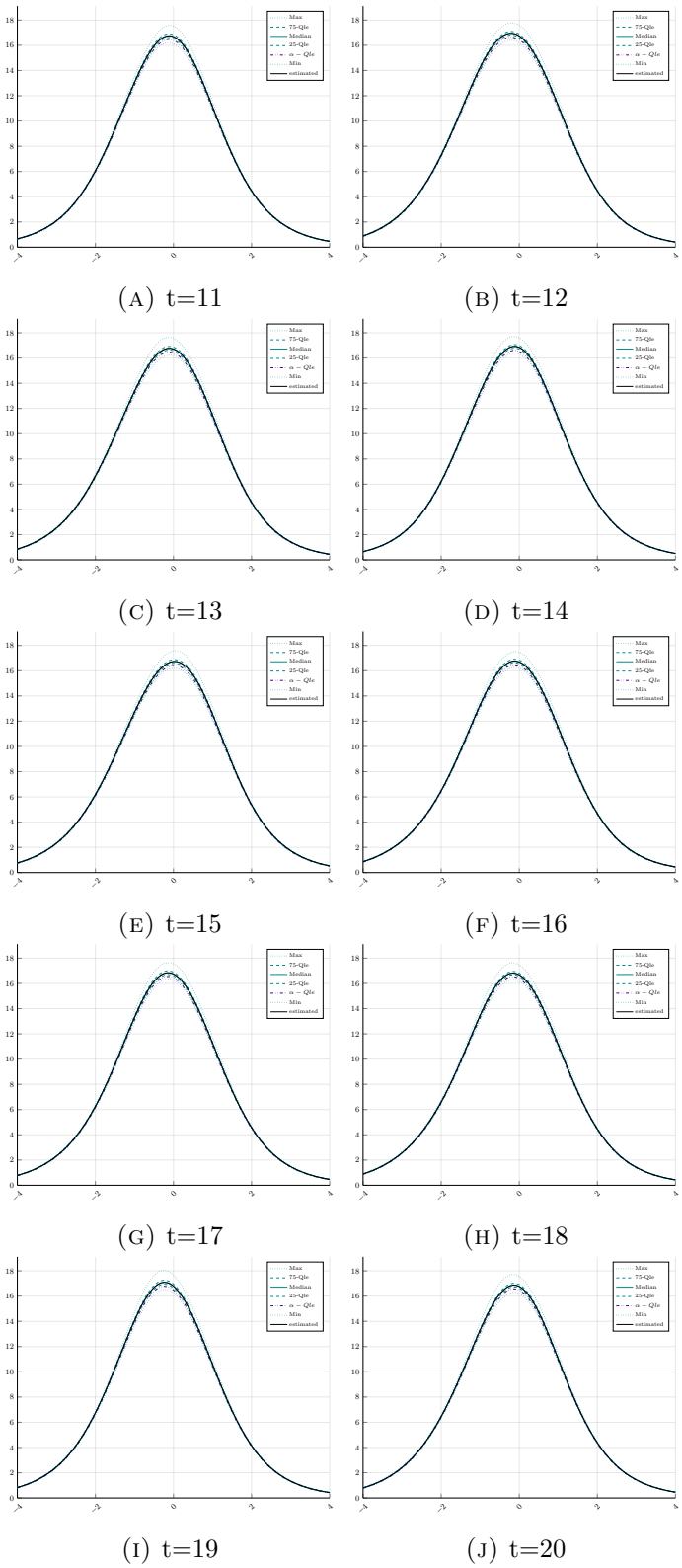
(C) t=3 (D) t=4

(E) t=5 (F) t=6

(G) t=7 (H) t=8

(I) t=9 (J) t=10

FIGURE A.32: Sampling distributions of TIFs. Tests 1-10. The horizontal axis represents the latent trait θ



(A) t=11 (B) t=12

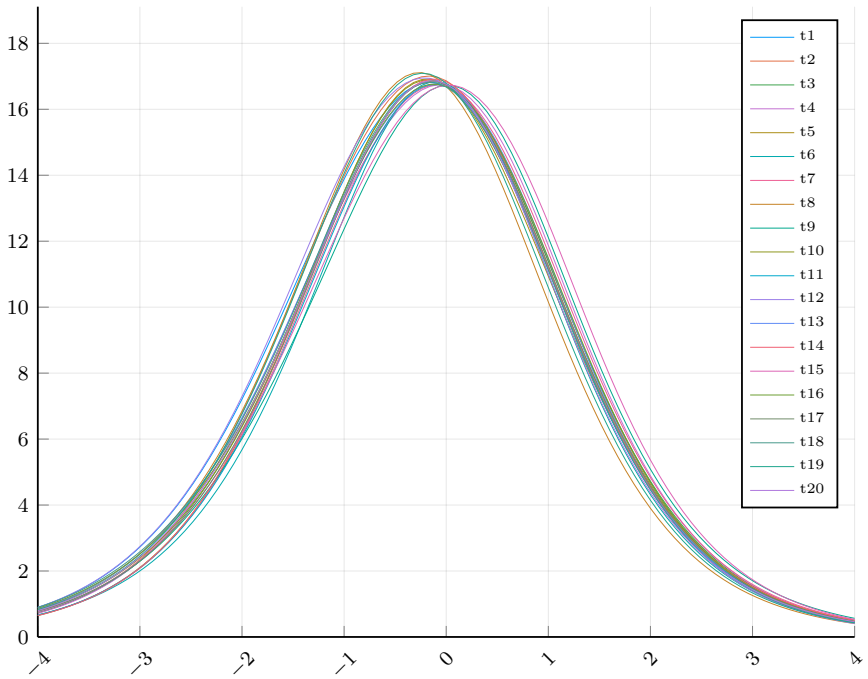
(C) t=13 (D) t=14

(E) t=15 (F) t=16

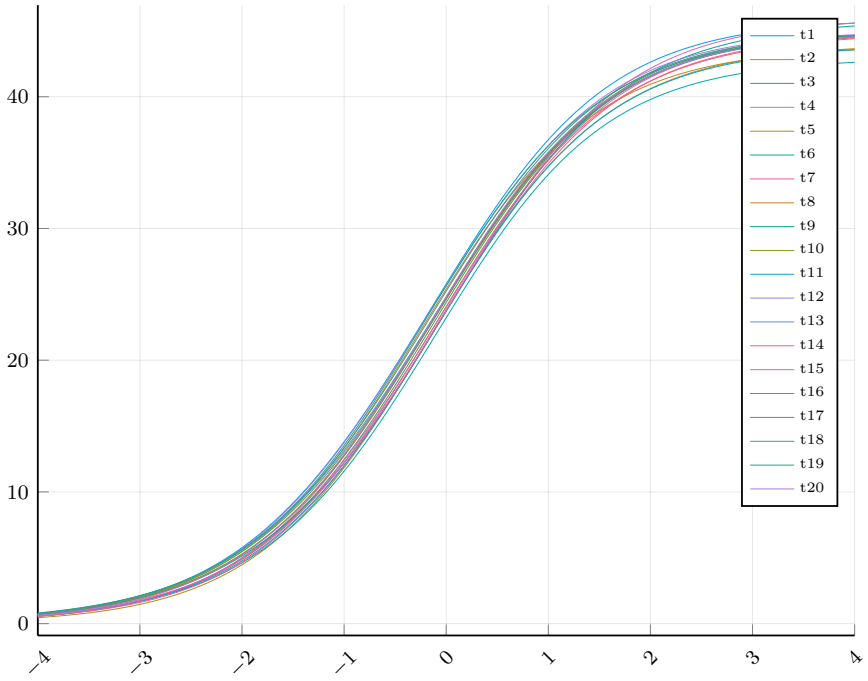
(G) t=17 (H) t=18

(I) t=19 (J) t=20

FIGURE A.33: Sampling distributions of TIFs. Tests 11-20. The horizontal axis represents the latent trait θ.



(A) TIF



(B) ICC

FIGURE A.34: TIFs and Item characteristic curves (ICCs) obtained from the full sample. The horizontal axis represents the latent trait θ.