Designing Software to Unify Person-Fit Assessment

Phillip Isaac Pfleger
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Education Commons

BYU ScholarsArchive Citation
Pfleger, Phillip Isaac, "Designing Software to Unify Person-Fit Assessment" (2020). Theses and Dissertations. 8776.
https://scholarsarchive.byu.edu/etd/8776

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.
Designing Software to Unify Person-Fit Assessment

Phillip Isaac Pfleger

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Richard Sudweeks, Chair
Ross Larsen
Joseph Olsen
Peter Rich

Educational Inquiry, Measurement, and Evaluation Brigham
Young University

Copyright © 2020 Phillip Isaac Pfleger
All Rights Reserved
ABSTRACT

Designing Software to Unify Person-Fit Assessment

Phillip Isaac Pfleger
Educational Inquiry, Measurement, and Evaluation, BYU
Doctor of Philosophy

Item-response theory (IRT) assumes that the model fits the data. One commonly overlooked aspect of model-fit assessment is an examination of person fit, or person-fit assessment (PFA). One reason that PFA lacks popularity among psychometricians is that comprehensive software is not present. This dissertation outlines the development and testing of a new software package, called wizirt, that will begin to meet this need. This software package provides a wide gamut of tools to the user but is currently limited to unidimensional, dichotomous, and parametric models. The wizirt package is built in the open source language R, where it combines the capabilities of a number of other R packages under a single syntax. In addition to the wizirt package, I have created a number of resources to help users learn to use the package. This includes support for individuals who have never used R before, as well as more experienced R users.

Keywords: item response theory, computer software, test validity, statistical analysis
ACKNOWLEDGMENTS

I would like to acknowledge and thank my chair, Richard Sudweeks, for inspiring the idea that became my dissertation and for the excellent feedback that helped me along. I would like to thank my committee for their enthusiasm and support.

All those who were involved in beta testing, Kerong, Jonathan, Rebecca, and Chris, thank you. Your feedback was key to the improvement of wizirt.

Thank you to Jerry and Jeanie Dunn for lending me a room for a personal office; it was a critical support while writing this dissertation. Thank you to my parents and siblings for making this path possible.

Thank you to my wife, Brenna, and my children for making this long journey with me and sacrificing so much to help me through. And of course, thanks to God for helping me rise to the challenge.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Title Page</th>
<th>ii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>Chapter 1: Background</td>
<td>1</td>
</tr>
<tr>
<td>Statement of the Problem</td>
<td>1</td>
</tr>
<tr>
<td>Statement of Purpose</td>
<td>2</td>
</tr>
<tr>
<td>A Note on Style</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 2: Review of Literature</td>
<td>4</td>
</tr>
<tr>
<td>Person-Fit Assessment</td>
<td>4</td>
</tr>
<tr>
<td>Detection</td>
<td>4</td>
</tr>
<tr>
<td>Person Response Function</td>
<td>9</td>
</tr>
<tr>
<td>Quantitative Modeling</td>
<td>10</td>
</tr>
<tr>
<td>Available PFA Software</td>
<td>12</td>
</tr>
<tr>
<td>Chapter 3: Method</td>
<td>14</td>
</tr>
<tr>
<td>A More Integrated Suite</td>
<td>16</td>
</tr>
<tr>
<td>Comprehensive PFA</td>
<td>17</td>
</tr>
<tr>
<td>Holistic IRT Perspective</td>
<td>18</td>
</tr>
<tr>
<td>Programming Strategy</td>
<td>19</td>
</tr>
<tr>
<td>For R Novices</td>
<td>19</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1  PFA Detection Statistics and Their Adaptations .......................................................... 5
Table 2  Available PFA Statistics Among Some Common IRT Software .................................... 13
Table 3  Example of a Technical Summary Produced by wizirt ................................................. 35
Table 4  Example of a Likelihood Ratio Test for Comparing Models in wizirt .............................. 36
Table 5  Example of a Table Reporting Unidimensionality Statistics ......................................... 38
Table 6  Example of an Output from irt_item_fit() ..................................................................... 52
Table 7  Example of Person-Level Statistics, Response Patterns by Difficulty ............................ 58
Table 8  Example of Persons-Level Statistics, Selected Items in Response Patterns .................... 58
Table D1  Comparison of wizirt and Sinharay (2017) ................................................................. 101
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>The wizrt Hex Logo</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2</td>
<td>A Screenshot of the Model Overview Section of the irt_report() Output</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3</td>
<td>A Screenshot of the Items Section of the irt_report() Output</td>
<td>22</td>
</tr>
<tr>
<td>Figure 4</td>
<td>A Screenshot of the Persons Section of the irt_report() Output</td>
<td>23</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Summarized Workflow Diagram for the wizrt Package</td>
<td>24</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Demonstration of the Standardized Translation Objects</td>
<td>28</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Plot for Diagnosing Local Dependence</td>
<td>39</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Test Information Function</td>
<td>40</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Examinee Responses by Ability</td>
<td>41</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Item Characteristic Curves</td>
<td>42</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Item Information Functions</td>
<td>43</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Item Residual Plots</td>
<td>44</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Standardized Residual Plot</td>
<td>45</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Item Characteristic Plot with Residuals</td>
<td>46</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Juxtaposed Item Characteristic Curves with Residuals</td>
<td>47</td>
</tr>
<tr>
<td>Figure 16</td>
<td>All Item Characteristic Curves Superimposed</td>
<td>48</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Selected Item Characteristic Curves Superimposed</td>
<td>49</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Flagged Item Characteristic Curves Superimposed</td>
<td>50</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Summed Information Function from Selected Items</td>
<td>51</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Person Response Functions for Persons 1-25</td>
<td>53</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Person Response Functions for Persons 26-50</td>
<td>54</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Person Response Functions for Persons 51-75</td>
<td>55</td>
</tr>
</tbody>
</table>
CHAPTER 1

Background

Any statistical method rests on certain assumptions. Item response theory (IRT) is a statistical theory used to estimate an individual’s abilities, attitudes, or other latent characteristics based on mathematically described relationships between the items on a test and the characteristics of interest. A general model common in IRT is Birnbaum’s three-parameter-logistic model for dichotomous items (De Ayala, 2009):

\[
p(x_j = 1|\theta, \alpha_j, \delta_j, \chi_j) = \chi_j + (1 - \chi_j) \frac{e^{\alpha_j(\theta - \delta_j)}}{1 + e^{\alpha_j(\theta - \delta_j)}}
\]  

(1)

In this model, the probability of a person answering a test item correct is a function of their ability (\( \theta \)), the discriminating power of the item (\( \alpha_j \)), the difficulty of the item (\( \delta_j \)), and a lower-bound on the probability called the guessing parameter (\( \chi_j \)). Unidimensional IRT assumes unidimensionality, local dependence, and model-data fit (De Ayala, 2018). Readers are likely familiar with various methods for evaluating model-data fit in item-response theory including absolute fit, relative fit, item-level fit, and person-level fit (Rupp, 2013). Person-fit assessment (PFA) is a useful tool for detecting and diagnosing response patterns of individual examinees inconsistent with the model (Meijer & Sijtsma, 1995, 2001).

Statement of the Problem

PFA involves using various measures and statistics to identify persons whose response patterns are considered aberrant and to diagnose the nature and degree of aberrance (i.e., spuriously high, spuriously low, or spuriously mixed). Spuriously high responses may be due to (a) cheating, (b) guessing, or (c) prior exposure to the assessment; while reasons for spuriously low responses might include (a) carelessness, (b) test anxiety, or (c) slow starting; and spuriously
mixed responses might be caused by (a) language barriers or (b) lack of effort or motivation in testing (Rupp, 2013).

Poorly fitting response patterns are problematic because they provide misinformation to test users about an examinee’s trait levels (Walker, 2017). While PFA is increasingly recommended by (a) researchers (De Ayala, 2018), (b) professional organizations, (c) testing companies (Tendeiro & Meijer, 2014), and (d) governments (Olson & Fremer, 2013), it is currently underused and lacks popularity among practitioners at large (Meijer et al., 2016).

Rupp (2013) suggested that PFA would not gain in popularity until more holistic software became available.

Even though powerful commercial packages exist that estimate a wide range of parametric models…, practitioners in the area of IRT have long yearned for more comprehensive data-analytic suites that integrate advanced graphical tools, a wide variety of relative, absolute, item- and person-fit statistics, as well as routines for estimating unidimensional and multidimensional IRT models. … If person fit analyses are to become more widely used by practitioners, more integrated software suites are indeed desirable to make this process easier. (pp. 26-27)

Statement of Purpose

The purpose of this dissertation is to develop an integrated software product that permits users to conduct a broad range of analyses including unidimensional, parametric, and dichotomous models and satisfies the following criteria:

- Accurate: A software for statistical estimation should produce accurate results.
- Informative: An integrated software product should provide all of the necessary information.
• Useful: The information and estimates should be easily accessible so that the software can fit easily into the user’s workflow.

• Aesthetic: A software product should adhere to a style guide and produce output that is attractive and facilitates both understanding and use.

A Note on Style

This dissertation was a challenge to write up because my project is a blend of software development and academic research. It was difficult to find the balance between the casual, pragmatic approach used in programming and the formulaic approach of academics. One place that I struggled to balance these two was in the presentation of code. As far as I am aware, there is no standard for including code in APA format. So, I found a compromise.

Throughout this dissertation, code will follow the same formatting as equations. Specifically, each statement will be centered with a right aligned numeric label enclosed in parentheses. In the text, phrases that represent functions are followed by empty parentheses. This is a common designation of a function in R programming.

A second compromise was made for the placing of the code that created plots. In an academic paper, a large plot is placed on its own page without any text. In this dissertation the code that created a plot is typically included on the same page as the plot, regardless of the size of the plot. This was done to increase the usefulness of this document as a reference for programmers, though it is a departure from typical APA formatting.
CHAPTER 2

Review of Literature

It may be that PFA has not grown in popularity because of the general disagreement as to how best to approach it (Karabatsos, 2003). However, in recent years there has been a growing acceptance of the procedure suggested by Rupp (2013; Tendeiro & Meijer, 2014). Rupp’s framework for PFA includes five steps: (a) statistical detection using local and global fit measures (either parametric or nonparametric); (b) numerical tabulation, or summarization of the incidence of each type of aberrant response pattern; (c) graphical exploration such as person response functions (PRFs); (d) quantitative explanation using additional modeling; and (e) qualitative explanation using such evidence as think-aloud procedures. While this framework is gaining traction, the practice of PFA has not yet caught up to the literature.

Person-Fit Assessment

Detection

Traditionally, PFA has focused primarily on the detection of individuals whose response patterns are considered aberrant. Thus, a wide range of detection statistics have been created. For unidimensional models with dichotomous data these detection indices can be broken into four families of measures (see Table 1) including (a) nonparametric statistics (Karabatsos, 2003; Meijer & Sijtsma, 2001), (b) Cumulative Sum (CUSUM) statistics (Armstrong & Shi, 2009a; 2009b), (c) traditionally used parametric statistics (Rupp, 2013; Smith, 1986), and (d) Bayesian statistics (Glas & Meijer, 2003; Sinharay, 2018).

Smith (1986) provided a brief but insightful history of nonparametric person-fit measures. In his summary, Smith identified four themes in aberrancy measurement research: (a) weighting certain patterns of scores if they are expected to be invalid or removing the individuals
Table 1

PFA Detection Statistics and Their Adaptations

<table>
<thead>
<tr>
<th>PFA Detection</th>
<th>Reference</th>
<th>Variants</th>
<th>Model</th>
<th>Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Guttman, 1944, 1950</td>
<td>Gp, GpN</td>
<td>Nonparametric</td>
<td>Guttman</td>
</tr>
<tr>
<td>A, D, E</td>
<td>Kane &amp; Brennan, 1980</td>
<td></td>
<td>Nonparametric</td>
<td>Guttman</td>
</tr>
<tr>
<td>U3</td>
<td>Van Der Flier, 1982</td>
<td>U3, ZU3, U3p</td>
<td>Nonparametric</td>
<td>Guttman</td>
</tr>
<tr>
<td>C*</td>
<td>Harnisch &amp; Linn, 1981; Sato, 1975</td>
<td>C</td>
<td>Nonparametric</td>
<td>Caution Index</td>
</tr>
<tr>
<td>NCI</td>
<td>Tatsuoka &amp; Tatsuoka, 1983</td>
<td>U1</td>
<td>Nonparametric</td>
<td>Caution Index</td>
</tr>
<tr>
<td>ICI</td>
<td>Tatsuoka &amp; Tatsuoka, 1982, 1983S</td>
<td></td>
<td>Nonparametric</td>
<td>Caution Index</td>
</tr>
<tr>
<td>ECI</td>
<td>Harnisch &amp; Tatsuoka, 1983; Tatsuoka, 1984</td>
<td>ECI1, ..., ECI6, ECI1Z, 2Z, 4Z, 5Z</td>
<td>Parametric</td>
<td>Caution Index</td>
</tr>
<tr>
<td>rPBS</td>
<td>Donlon &amp; Fischer, 1968</td>
<td>rPBS</td>
<td>Nonparametric</td>
<td>Correlation</td>
</tr>
<tr>
<td>H^T</td>
<td>Sijtsma, 1986; Sijtsma &amp; Meijer, 1992</td>
<td></td>
<td>Nonparametric</td>
<td>Correlation</td>
</tr>
<tr>
<td>lZ</td>
<td>Drasgow, Levine, &amp; McLaughlin, 1991; Levine &amp; Rubin, 1979</td>
<td>L0, Lco, lcz, lZ*</td>
<td>Parametric</td>
<td>Likelihood</td>
</tr>
<tr>
<td>M</td>
<td>Molenaar &amp; Hoijtink, 1990</td>
<td>M-</td>
<td>Parametric</td>
<td>Likelihood</td>
</tr>
<tr>
<td>Likelihood Ratio</td>
<td>Levine &amp; Drasgow, 1988</td>
<td></td>
<td>Parametric</td>
<td>Likelihood</td>
</tr>
<tr>
<td>UW</td>
<td>Smith, 1985</td>
<td></td>
<td>Rasch</td>
<td>Residual</td>
</tr>
<tr>
<td>UB</td>
<td>Smith, 1985</td>
<td></td>
<td>Rasch</td>
<td>Residual</td>
</tr>
<tr>
<td>U</td>
<td>Wright &amp; Stone, 1979; Wright &amp; Masters, 1982</td>
<td>ZU</td>
<td>Rasch</td>
<td>Residual</td>
</tr>
<tr>
<td>W</td>
<td>Wright &amp; Stone, 1979; Wright &amp; Masters, 1982</td>
<td>ZW</td>
<td>Rasch</td>
<td>Residual</td>
</tr>
</tbody>
</table>
with extreme patterns (Cronbach, 1946, 1950); (b) person response curves as introduced by Weiss (1973) and developed by Lumsden (1977); (c) caution indices based on Guttman scaling (1944; Harnisch & Linn, 1981; Sato, 1975); and (d) the work of Fowler (1954) and Donlon and Fischer (1968) using point biserial correlations.

Of the nonparametric measures discussed by Smith (1986), those based on Guttman scales are the most frequently used. These Guttman-based measures share the same logic that individuals who incorrectly answer an easy item should not subsequently answer a more difficult item correctly. Any discrepancy is considered an error (Smith, 1986). This logic has led to the birth of fit measures such as G (Guttman, 1944, 1950), U3 (Van Der Flier, 1982), and A, D, and E (Kane & Brennan, 1980). Caution indices are another extension of Guttman errors. These include the popular parametric ECI statistics from Tatsuoka (1984), as well as NCI (Tatsuoka & Tatsuoka, 1983), ICI (Tatsuoka & Tatsuoka, 1982, 1983), C (Sato, 1975), and C* (Harnisch & Linn, 1981). Of the nonparametric measures, the most frequently praised in the research is the correlation-based $H_T^T$ (Sijtsma, 1986). $H_T^T$ (more formally written as $H_T^T$) is calculated using the following equation:

$$H_T^T = \frac{\sum_{j \neq i} \sigma_{ij}}{\sum_{j \neq i} \sigma_{ij}^{max}},$$

where

$$\sigma_{ij} = \beta_{ij} - \beta_i \beta_j$$

and

$$\sigma_{ij}^{max} = \beta_i (1 - \beta_j),$$

and $\beta_i$ and $\beta_j$ equal the proportion of items correctly answered by examinee $i$ and $j$ respectively, and $\beta_i \beta_j$ is the proportion of items that both examinee $i$ and $j$ answered correctly.
Smith (1986) criticized the four branches of nonparametric statistics for being sample
dependent and claimed that parametric fit measures based on Rasch analysis were more desirable
because they were not sample dependent. Thus began a long-standing competition between
parametric (model-based) and nonparametric (model-free or group-based) fit measures.

Prior to one of two key reviews written by Meijer and Sijtsma (1995) parametric
measures were consequently disproportionately represented in the literature. However, Meijer
and Sijtsma pointed out that when a model is inappropriate to use, nonparametric fit measures
are the only resource researchers have to diagnose problems of person misfit. They implied that
the best PFA measure to use is the one that best fits the model.

In a second key review, Meijer and Sijtsma (2001) provided more guidance on when to
use what kind of fit measure; much of that advice came from the type of model that was
estimated. For example, they suggested M when a Rasch model has been run because the critical
values are more accurate. They also suggested the ZU3 statistic in a nonparametric context but
were not positive about its usage. No fit measure was suggested in the context of 2PL or 3PL
models. They ended their review echoing Smith’s (1985) four suggestions for handling misfitting
measures: (a) report several measures, (b) remove invalid items and re-estimate theta, (c) retest
the misfitting individual, and (d) determine whether the impact on theta is large enough to be a
cause of concern. Overall, Meijer and Sijtsma’s (2001) review was balanced in terms of the
nonparametric versus parametric debate, maintaining the idea that the fit measure used depends
on the model used.

This sentiment was not shared by Karabatsos (2003). In a large simulation study
comparing 36 person-fit measures, the largest and most comprehensive simulation on the topic to
date, Karabatsos found that nonparametric fit measures outperformed parametric fit measures
(including the frequently used $l_z^*$ statistic). Specifically, the $H^T$ measure performed the best. After reviewing previous simulation studies, Rupp (2013) asserted that simulation studies like Karabatsos's, while necessary and good, were not universally applicable. Karabatsos’s conclusions were only valid when conditions were similar to those used in the simulation, which were quite restrictive. Sinharay (2017) reiterated Rupp’s declaration, adding his own simulation study showing that $l_z$ was just as good under other circumstances. This back and forth between parametric and nonparametric was a common theme in the literature, though the most widely used person-fit statistics are the traditional parametric measures.

Smith (1986) claimed that parametric measures were advantageous over the group-dependent measures because they relied on item-statistics that were independent of samples. Another cited advantage of the parametric fit measures is the ability to calculate a p-value and to therefore obtain Type I and Type II error rates (Snijders, 2001). This has been one of the most frequently discussed topics within parametric research (Sinharay, 2016). For example, the popular $l_z$ statistic was first proposed by Drasgow et al. (1991). However, the distribution of the $l_z$ statistic was simply assumed to be normal. Snijders (2001) later showed that the assumption was false but provided a correction that could be applied to the $l_z$ statistic and many other measures within the same family. These corrected statistics are usually designated with an asterisk, such as the $l_z^*$. This measure was further improved by others, most notably Sinharay (2018) who extended the measure by adding Bayesian posterior predictive model checking to increase power.

There are many parametric statistics with many versions (Meijer & Sijtsma, 2001). It should be noted that within these measures there are methods dedicated specifically to Rasch models and other measures designed for more general models, such as the two-parameter logistic
model and the three-parameter logistic model. Infit and outfit are examples of Rasch specific
person-fit measures, while lz is a method belonging to the two- and three-parameter logistic
models.

The most consistent limitation of parametric measures is that their distribution depends
on the test; therefore, cutoffs are never the same (Sinha, 2018). This is exactly where
Bayesian methods shine. According to the work by Sinha (2018), Bayesian methods are really
more extensions of older methods than totally new measures. Usually, a traditional parametric fit
measure will be used, and a Bayesian posterior predictive p-value will be done post hoc to get an
appropriate p-value. While they can be used to get more accurate p-values, Bayesian methods are
very computationally intensive. Sinha said that Bayesian methods are usually adapted
versions of the other mentioned statistics, such as computing a posterior predictive p-value for lz.
or a modified version of the response time models (Glas & Meijer, 2003). While these
adaptations are usually computationally expensive, they may be more accurate and also provide
more easily digestible interpretations (Sinha, 2018).

CUSUM methods come from the field of mechanical operations (Armstrong & Shi,
2009a). In that field, they inform operators when a machine needs repairs. In the world of testing,
these measures help researchers identify aberrant responses with a likelihood ratio test

**Person Response Function**

Research on PFA has not only developed detection measures. Graphical measures have
also been developed. In particular, the person response function (PRF; Nering & Meijer, 1998;
Sijtsma & Meijer, 2001). The PRF, which has its roots in the subject characteristic curve
introduced by Weiss (1973), is a graphical measure describing the probability of an examinee
answering correctly depending on the difficulty of the item. It is often created by ordering the items on a test from easiest to most difficult and then plotting an individual examinee’s probability for answering correctly (Walker et al., 2018). The PRF was expanded by Lumsden (1977) to create a person-reliability statistic. This person-reliability statistic was calculated as the slope of the PRF.

Some of the greatest advantages of the PRF are that it can be used to detect local and global misfit, and it can even help diagnose reasons for misfit (spuriously low, spuriously high, or spuriously mixed scores; Walker et al., 2018). It is also very flexible and can be used in a nonparametric context (Smith, 1986). The main disadvantage of the PRF is that it is used for one respondent at a time (Sijtsma & Meijer, 2001). This has led it to be used mainly after a reason for misfit has been detected with another statistic (Nering & Meijer, 1998). However, there is a statistic for whether the observed PRF is significantly different from the expected PRF. The PRF is viewed as such a useful tool for identifying and understanding misfitting response patterns that Rupp (2013) recommended every analysis of person fit include it.

While PRF is a popular graphical method, CUSUM methods come with their own diagram that fills a similar purpose. For examples of usage see Armstrong and Shi (2009b).

Quantitative Modeling

While there are many of these detection-oriented statistics, this traditional approach falls short of the current research on the topic. Newer approaches to PFA include modeling reasons for person misfit. Four different methods of modeling show promise in this area: (a) response time modeling, (b) multilevel modeling, (c) person-reliability modeling, and (d) machine learning modeling. The multilevel modeling method first requires that an IRT model be run, including the calculation of person-fit measures. Then a multilevel model is run. This allows for
the inclusion of other explanatory variables. The model by Conijn et al. (2011) is recommended because older models violate important assumptions.

Response time models (Fox & Marianti, 2017; Marianti et al., 2014) consider the amount of time an individual took to answer each question. This is very helpful in diagnosing speededness and various forms of cheating. There are additional fit measures applicable in response time modeling (Sinha ray, 2018). The model by Fox and Marianti (2017) is recommended for this type of modeling.

Reise (2000) developed a model that involved two steps. In the first step, the researcher estimates an IRT model. Then, using multilevel logistic regression, the researcher predicts the probability that each person would get each item correct or incorrect, including predictors if possible. The slopes from this model are then used to detect person misfit (Walker & Engelhard, 2015). Conijn et al. (2011) showed that this model did not meet the assumptions that it was founded on, though it has been used in research. They proposed a new model that involves using either multiple observations or estimating person-fit measures on subsets of data and then running a multilevel model on the repeated observations.

The most recent addition to the world of PFA comes from the big data movement. These models tend to use nonparametric models that include additional information about students to predict model misfit. Man et al. (2019) applied principles of data science to identify cheaters on an exam. They found that they were able to identify cheaters more often than through other measures. Person-reliability models are another option that directly model a new person statistic, the person-reliability (Ferrando, 2015).

Rupp’s approach to PFA combines statistical detection, graphical exploration, and modeling into one comprehensive activity. While Rupp’s framework is the most comprehensive
approach so far, and while it has the potential to unify the field of PFA and consensus has been growing around it, the software for conducting it in this manner is severely limited.

Available PFA Software

Meijer et al. (2016) suggest that the reason PFA is underused is because of the lack of available software. The same reason could explain why Rupp’s framework has not been implemented more often. Some of the most popular IRT software do not even produce PFA statistics (e.g., flexMIRT, IRTPRO, BILOG; see Table 2). Those that produce PFA measures usually only produce a small number of global detection indices (e.g., Winsteps, Acer Conquest) and neglect the other aspects of Rupp’s framework for PFA, though some will generate person response functions (i.e., PerFit, WPerFit).

Winsteps (Linacre, 2020) is a notable software for PFA. While it produces only detection indices, Winsteps also generates a number of tables summarizing the person-fit information, thus excelling at the second step of Rupp’s framework, tabulation.

Currently, the most comprehensive PFA software is the PerFit package in R. PerFit calculates many of the most researched PFA measures, including some parametric and nonparametric ones (Tendeiro et al., 2016). It also generates person response functions and calculates IRT item statistics. However, even PerFit fails to detect local person-misfit, provide tables of aberrant patterns, or facilitate further modeling of reasons for person misfit, all of which are essential aspects of Rupp’s framework.
<table>
<thead>
<tr>
<th>Software</th>
<th>Estimates Person Abilities</th>
<th>Global Detection Measures</th>
<th>PRFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winsteps</td>
<td>Yes</td>
<td>infit, outfit</td>
<td>No</td>
</tr>
<tr>
<td>flexMIRT</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IRTPRO</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BILOG</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Acer Conquest</td>
<td>Yes</td>
<td>infit, outfit, case fit</td>
<td>No</td>
</tr>
<tr>
<td>WPerFit</td>
<td>No</td>
<td>Lz, ECI4z, Trabin and Weiss chi-square</td>
<td>Yes</td>
</tr>
<tr>
<td>PerFit(^a)</td>
<td>Yes</td>
<td>Lz, Lz(^*), rpbis, caution index, Guttman errors, agreement, disagreement, dependability, U3, ZU3, NCI, H(^T)</td>
<td>Yes</td>
</tr>
<tr>
<td>mirt(^a)</td>
<td>Yes</td>
<td>infit, outfit, Zh</td>
<td>No</td>
</tr>
<tr>
<td>ltm(^a)</td>
<td>Yes</td>
<td>L0, Lz</td>
<td>No</td>
</tr>
<tr>
<td>sirt(^a)</td>
<td>Yes</td>
<td>caution index, dependability index, ECI 1-6, L0, Lz, outfit, infit, rpbis, U3</td>
<td>No</td>
</tr>
<tr>
<td>eRm(^a)</td>
<td>Yes</td>
<td>infit, outfit</td>
<td>No</td>
</tr>
</tbody>
</table>
CHAPTER 3

Method

One frustration that I have had with IRT software in the past is that I must use multiple software to accomplish what I want to do. For example, there are many packages in the R programming language (R Core Team, 2020) that perform aspects of IRT. Unfortunately, these packages have different goals and sometimes vastly different syntaxes. Consider this simple example of PFA. In this example, I import my data using the function base::read.csv(). The notation base::read.csv indicates that read.csv() is a function that accompanies the base packages in R.

\[
\text{responses} \leftarrow \text{read.csv("path/to/data.csv")}
\]

Then I estimate an IRT model using mirt::mirt():

\[
\text{my_model} \leftarrow \text{mirt(data = responses, model = 1, itemtype = "2PL")}
\]

Then I get the lz statistic using PerFit::lz() and I get a cutoff using PerFit::cutoff():

\[
\text{a} \leftarrow -\text{lz(responses, IRT.PModel = 2PL)}
\]

\[
\text{plot(a, cutoff.obj = cutoff(a))}
\]

There is no method to calculate local fit in R, but I can display aberrant responses:

\[
\text{responses% > %dplyr::filter(cutoff(a))% > %dplyr::distinct( )}
\]

Person response functions can be created using the PerFit::PRFplot():

\[
\text{PRFplot(responses[which(cutoff(a)), ])]}
\]

To model the detection statistic (Conijn et al., 2011) we use the function lme4::glm(). This assumes that another data set called person_details has been created that contains the calculated statistics and the data that will be used to model the measures:

\[
\text{glm(lz~view ime, data = person_details)}
\]
In this example I used five different packages, each with a different philosophy behind its design. I would still need to extract the information from models and compile it into a report, either putting it into a Rmarkdown document or pasting it into a separate document such as Word. This additional work provides a significant hurdle that is superfluous from the analysis.

My intent with wizirt was to develop a software package that can glue all of these disparate pieces together. This makes R the ideal programming language within which to build wizirt. R is a powerful and popular open source\textsuperscript{2} software. Not only is R free, but most of the required functions for conducting IRT and PFA (with the exception of local detection indices) have previously been written by others. My work will be to bring these various functions together into one single R package.

The software product that I designed for my dissertation is called wizirt (Figure 1). The wizirt software is intended to (a) provide a more holistic approach to PFA, (b) be capable of being extended by other users, (c) be easy to use, and (d) be of professional quality. In this section I discuss the design of the software package, including the learning resources that will help a wider audience use wizirt and the criteria used to determine success. In the next section I will demonstrate the use of the wizirt package, show how it meets the evaluative criteria specified, and describe the learning resources.
A More Integrated Suite

After noting that it is common for researchers and practitioners to use as many as five software products to perform PFA, Rupp (2013) called for the development of new PFA software:

Even though powerful commercial packages exist that estimate a wide range of parametric models, . . . practitioners in the area of IRT have long yearned for more comprehensive data-analytic suites that integrate advanced graphical tools, a wide variety of relative, absolute, item- and person-fit statistics, as well as routines for estimating unidimensional and multidimensional IRT models. . . . If person fit analyses are to become more widely used by practitioners, more integrated software suites are indeed desirable to make this process easier. (pp. 26-27)
Wizirt is intended to be this “more integrated software [suite],” at least for unidimensional, parametric models using dichotomous data at first. The wizirt package is an integrated suite in three ways: (a) it approaches PFA with Rupp’s comprehensive framework, (b) it considers IRT holistically, and (c) it unites other powerful software packages.

**Comprehensive PFA**

Rupp’s framework plays a key role in the wizirt software. For global detection, the PerFit (Tendeiro et al., 2016) package is utilized. The PerFit package calculates the most popular person-fit measures for IRT, including many nonparametric ones. Some of the most popular are the $H^T$ and $Iz^*$ statistics. The wizirt package uses PerFit to estimate these person-fit measures as well as $Iz$, $NCI$, $E$, $D$, $A$, $ZU3$, $U3$, $C^*$, $C$, and $G$. These measures are available for all models, and wizirt uses PerFit to calculate cutoffs at a given alpha level for each statistic requested. Rasch models are allowed additional measures through the sirt package, though no cutoffs are available.

Currently, no software calculates local person fit. Initially, I intended to include either the ICI by Tatsuoka and Tatsuoka (1983) or another easily calculated local detection statistic. While trying to apply the ICI, I felt that it did not fit the overall goal of wizirt. The purpose of a local fit statistic in the workflow of wizirt is to help the user identify where aberrant persons misfit the model. The ICI statistic is instead used to identify when persons reach a learning plateau, and it is intended to be used across multiple parallel tests (Tatsuoka & Tatsuoka, 1982, 1983). This makes it impractical to use in wizirt because parallel forms will not always be available. I was not able to discover an alternative and thus did not include one for this step. Users looking for local misfit information might find success using the person response functions.
It is easy to print a table of person-level statistics in wizirt. This includes (a) whatever person-fit statistics an individual requests, (b) the cutoffs for those statistics, and (c) whether those individuals are flagged as misfitting based on the fit measures requested. There is an option to include response patterns. They can be sorted by difficulty, by their position in the data, or by any order specified by the user.

The PerFit package is combined with the ggplot2 package to create nonparametric person response functions for all individuals. Those who are flagged as aberrant are highlighted pink in the plot. Methods are available to superimpose these plots, to focus on a specified number of examinees, and to break the examinees into groups so there are fewer examinees on a page. Because these plots, and all other plots created by wizirt, are made with ggplot2, they are easily customizable.

Rupp’s quantitative modeling step is done following the multilevel modeling method suggested by Conijn et al. (2011) using the blme (Chung et al., 2013) package in R. The data are randomly divided into a certain number of bins (specified by the bins argument). Then person-fit statistics are calculated on each of the bins, and a multilevel model is run.

**Holistic IRT Perspective**

When designing wizirt, I used Rupp’s framework and the principles described in Chapter 11: *Item Response Theory and Rasch Modeling* by R. J. De Ayala in *The Reviewers Guide to Quantitative Methods for the Social Sciences* (2018, pp. 145-162; Appendix A) to create a checklist (Appendix B). While the checklist I created includes a list of the resources I planned to develop and the criteria for evaluating my success, the bulk of it focuses on creating a comprehensive IRT software. This checklist was heavily referenced throughout the design of the functions in wizirt.
There are four main functions in wizirt. The `wizirt()` function is used to estimate models with mirt (Chalmers, 2012), ltm (Rizopoulos, 2006), and eRm (Mair et al., 2020) doing the actual estimation. The `irt_assume()` function is used to check the assumption for a model, `irt_item_fit()` gets PFA statistics, and `irt_item_fit()` calculates item fit statistics. Users extract information from objects created with these functions by employing print and plot methods.

The examples in the Results chapter of this dissertation demonstrate more fully what I mean by “holistic IRT.” Ultimately, I envisioned wizirt to be comprehensive by designing it to satisfy all the desired characteristics Rupp and De Ayala proposed.

**Programming Strategy**

My overarching purpose behind building wizirt was to make PFA more easily available. By choosing R as the platform for my package, I am limiting the number of people who will ever use the package since R has an intimidating, steep learning curve.

To overcome this, I followed the pattern of the `bibliometrix` package (Aria & Cuccurullo, 2017) in providing tiered support; the `bibliometrix` package provides an easier interface for individuals who are new to R and more flexibility through functions for more advanced users. I likewise have developed a function for R novices and a more flexible set of functions for the experienced user.

**For R Novices**

To help ease the new user into R, I created the `irt_report()` function. This is a very useful function that runs almost all of the other functions found in wizirt. This function can be used to create a general report based on the data given to it. The user of this function only has to use a single line of code, and then wizirt has a dataset to examine.

\[
\text{irt_report}(\text{data} = \text{responses}, \text{item_type} = \text{"Rasch"})
\]  \hspace{1cm} (11)
Behind the scenes, this function (a) runs every other function in wizirt to estimate the desired model, (b) gathers information about the model typical to software such as IRTPRO (Cai et al., 2011), (c) generates the information required for PFA, and then (d) outputs a general report in the html format. Screenshots of this report are included in Figures 2-4. It is organized according to De Ayala’s (2018) checklist, though it also includes Rupp’s (2013) framework.

The results shown in these figures are all that the users see. Behind the scenes, wizirt converts the call they made to irt_report() and then translates it into code run for various other packages, as seen in Figure 5. Figure 5 shows that the user inputs a small amount of code, which is then translated into the appropriate syntax for the estimator. The output from this is converted to a standard object, which is then saved. The wizirt package then continues to break down other aspects of the syntax, translating standardized objects into the information needed for various function calls and then standardizing the output. This standardization is the keystone of my programming strategy. Ultimately, irt_report() calls every function in wizirt and then compiles the various tables and plots associated with the objects returned from these functions in a single, stylized report.

*For Experienced R Users*

Individuals who are more comfortable in R will find extended functionality and flexibility in the four functions I mentioned previously, wizirt(), irt_assume(), irt_item_fit(), and irt_person_fit(). The irt_model_pfa() is a fifth function that is also useful.
**Figure 2**

*A Screenshot of the Model Overview Section of the `irt_report()` Output*

---

### My Report

This report was generated using the practice data from the `wizirt` package.

### Model Overview

#### Technical Information

A 1 factor Rasch model was estimated using the function `mirt` from the package `mirt` (v 1.32.7). Estimation has converged using the EM method after 30 iterations with a convergence criteria $10^{-5}$.

#### Data Summary

Model built on a data set with 75 examinees ($\bar{\theta} = 0, \sigma_\theta = 0.39$) and 25 items ($\bar{\delta} = -1.19, \sigma_\delta = 0.24$).

#### Missing Data Summary

No data were missing from this data set.

### Assumptions

The data was tested for unidimensionality, relative fit, local/conditional dependence.

**Table 1**

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Value</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETECT</td>
<td>-2.62</td>
<td>Essential Unidimensionality</td>
</tr>
<tr>
<td>ASSI</td>
<td>-0.29</td>
<td>Essential Unidimensionality</td>
</tr>
<tr>
<td>RATIO</td>
<td>-0.56</td>
<td>Essential Unidimensionality</td>
</tr>
<tr>
<td>MADCDV100</td>
<td>1.69</td>
<td></td>
</tr>
</tbody>
</table>
Figure 3

A Screenshot of the Items Section of the irt_report() Output
Figure 4

A Screenshot of the Persons Section of the irt_report() Output
Figure 5

Summarized Workflow Diagram for the wizirt Package
These functions represent my vision of a new, unified syntax for IRT in R. This syntax was originally intended to follow very closely along the lines of the parsnip package (Kuhn & Vaughan, 2020), and wizirt is heavily influenced by parsnip and the tidyverse (Wickham et al., 2019).

This influence is because the parsnip package serves to unify many packages in the realm of machine learning. It provides a single, unified syntax to various machine learning models. For example, here is the same random forest model run in three different packages:

randomForest(y~., data = ., mtry = 10, ntree = 2000, importance = TRUE)  (12)
ranger(y~., data = dat, mtry = 10, num. trees = 2000, importance = impurity)  (13)
ml_random_forest(dat, intercept = FALSE, response = y,  
features = names(dat)[names(dat)! = y],  
col.sample.rate = 10, num. trees = 2000)  (14)

In this example, the differences are minimal, but they can still be frustrating. In other examples the differences in syntax are more extreme. Parsnip takes these packages and unifies the syntax so that it is now:

rand_forest(mtry = 10, trees = 2000) %>%
set_engine(spark) %>%
set_mode(regression) %>%
fit(mpg~., data = mtcars)  (15)

The vertical stylization of this code is intended for readability, a defining characteristic of the tidyverse that makes R more approachable for new users. While there is more code in the parsnip syntax, it is clear what exactly is being done. In the first line, R is told that it will be running a random forest model. In the next line, the package that will serve as the engine (“spark”) is named. The specific type of random forest model is specified as regression, then the model is run with an equation specifying the dependent variable (mpg), predictors (The “.” means all other
variables in the data set), and the data that it comes from ("mtcars"). Each function has a single purpose and is optimized for clarity.

At first, I developed a similar layered syntax:

\[
\text{irt}(\text{rownames = ids, item_type = "Rasch"}) \% > \%
\text{set_engine("mirt")} \% > \%
\text{fit_wizirt(data)}
\]

In this example, I declare that I am running a Rasch model using the package mirt for estimation. Lastly, the fit_wizirt line executes the code and points R to a data set called "responses" that the researcher would have previously imported into R.

During usability testing, I realized that this syntax was too complicated to be comfortable for most users, but I wanted to retain the layered syntax in case it worked better with the more complicated models I plan to add in the future. I ultimately decided to listen to my testers, and I created an easier function for users. Now a user can estimate a wizirt model using a simple syntax:

\[
\text{wizirt(data = responses, item_type = "Rasch", engine = "mirt")}
\]

The beauty of this is that a researcher does not need to know a half dozen different syntaxes to accomplish their purpose. Instead, wizirt writes the necessary syntax and runs the other packages for them. Thus, researchers can change the engine to be “mirt,” “ltm,” or “eRm” and run the needed model with minimal changes in code.

**Made to be Extended**

I designed wizirt so that other developers could add more functionality in the future, such as local person-fit or response-time models. To achieve this, I chose to host the code on GitHub, a service where code is publicly available, and anyone can suggest changes. This way, others can take the code and expand it while I referee their suggestions and potentially add their work back
into the main package. More than making wizirt open source, I designed it to be expandable. In Figure 5, I demonstrated that a key aspect of my strategy involved standardizing output. Figure 6 narrows the focus to make this design more explicit.

At each stage of the estimation process, the functions require variously shaped input and likewise return differently shaped output. This is represented in Figure 6 with the various shapes indicating input and output. The hourglass shape represents the standardized outputs that I will force the data into at each stage. By breaking the code into standardized pieces, I have limited the amount of work that future contributors will have to do. These contributors will simply have to connect their addition to the standardized translation objects to have their work seamlessly integrate.

Learning Resources

While developing wizirt, I also developed several learning resources: (a) a GitHub-hosted website walking through the use of the package, (b) several package vignettes, (c) a tidyverse style cheat sheet, and (d) an article walking through a replication of Sinharay (2017). All of these resources, except the replication of Sinharay, are stored online in the GitHub repository for wizirt (https://github.com/Pflegermeister/wizirt).

Evaluation Criteria

When evaluating wizirt, I am particularly interested in (a) the accuracy of the results, (b) the informative value of the output, (c) the usefulness of the functions, and (d) the aesthetic of the overall package. The most important among these is accuracy, followed by informative value, usefulness, and aesthetic, in that order.
Figure 6

*Demonstration of the Standardized Translation Objects*

Parameter Estimation

- mirt
- eRm
- ltm

Standardized Translation Object

Additional Statistics

- Assumptions
- PFA
- Else
Accuracy

Because I am not writing the functions that do the estimation, accuracy is more about passing information back and forth through functions appropriately. To make sure that wizirt is accurate, I compared it to several other known sources of estimation: (a) non-R software, (b) other R packages, (c) results given in the literature, and (d) an example in De Ayala (2009).

A common goal of researchers moving into R for the first time is determining how the estimates from R compare to IRTPRO and Winsteps. To test for accuracy, I did this myself. I ran a model in wizirt, then I ran it in Winsteps and IRTPRO, making changes to the command files for the two other programs to see what was required to get matching estimates. My goal was to have the estimates be within .01 of each other.

To check my accuracy with other R packages, I had wizirt export the raw estimated objects as well as the standardized objects. Then I explicitly called the functions I intended to call behind the scenes on those raw objects and compared that to what was exported from wizirt. Because I was using the same code, I expected that the results would be identical.

The replication was another source for checking the accuracy of wizirt. My original plan was to do a small-scale replication of Karabatsos’ (2003) landmark simulation. However, further exploration revealed an article by Sinharay (2017) that performed an enhanced version of the study, then applied a similar thought process to real data. I was able to acquire the real data used by Sinharay and the code that he used to perform his second analysis. The replication was then as simple as copying what he did but using wizirt syntax. Like with the comparison to IRTPRO and Winsteps, my goal was to replicate the same values to a .01 difference.
Lastly, I loosely followed the example found in chapters two-6 of De Ayala (2009). I was particularly interested in making sure that my plots looked like the plots found therein. This was very helpful since I had the data that De Ayala used for his example.

**Informative Value**

When I designed wizirt, I intended it to be high in informative value. I defined this to mean that wizirt will provide the needed statistics and graphics for the psychometrician to adequately perform any task related to parametric, unidimensional, dichotomous IRT models as defined by De Ayala’s (2018; Appendix A) checklist for reviewers as a guide for the development of IRT and by Rupp’s (2013) framework for PFA. To ensure that every element was present, I created the checklist found in Appendix B.

**Usefulness**

Having the right information is a key component of wizirt’s design. But information does no good if it cannot be easily accessed. With this in mind, I chose usefulness as one of the criteria for evaluating wizirt. Usefulness here means that users can access the information they need and want. I envisioned this as being able to conduct a complete IRT analysis in relatively few lines of code.

The development of wizirt followed the principles of design theory (Interaction Design Foundation, n.d.). That meant many iterations and prototypes, as well as meeting with and testing by end users. Throughout the development of wizirt, I had prototypes evaluated in three rounds. In the first round, I had three individuals try out the product. In the second round, I gave a demonstration to three individuals. In the third round, I asked three individuals to try out the package and to view an example report.
Aesthetic

One aspect of software that is important is the aesthetic of it. I designed my learning resources and the output from the wizirt functions using the four basic principles of design taught by Williams (2015). These include contrast, repetition, alignment, and proximity. In all of my products, design principles were used to direct the researcher’s attention, encourage efficient dissemination of accurate information, and prevent the user from being lost in a sea of information.

These same principles also influenced the design of the input to wizirt. I patterned the syntax after the R Style Guide for the tidyverse (Wickham, n.d.), adhered to the principles found in Advanced R, and used tools in the devtools (Wickham et al., 2020) package. My standards for aesthetic value involved being able to identify the use of the principles of design in the package and following the study guide.
CHAPTER 4

Results

The vision driving this dissertation is the hope that I can provide a way for PFA to become more widely used. To accomplish this, I have set out to design and create a new software for IRT. This software, wizirt, is a more holistic approach to IRT because it not only facilitates every aspect of De Ayala’s (2018) checklist, but it also incorporates Rupp’s (2013) framework for PFA. In the previous section, I discussed the development of the wizirt package. In this section, I describe the application and use of the wizirt package, including the accompanying learning materials. I also show that the package meets my criteria for evaluation.

Overview of wizirt

Before users can use the functions found in wizirt, they must install the package from GitHub. This is done using the function install_github from the package devtools. The package is stored in a repository called “Pflegermeister/wizirt”:

\[
\text{devtools :: install_github("Pflegermeister/wizirt")}
\]

(18)

Installation is a one-time operation (or as often as the user wants to update the package). Loading the wizirt package must be done any time a user wants to use the functions it contains:

\[
\text{library(wizirt)}
\]

(19)

Data must be a person by questions data frame or matrix, where the values of the data are dichotomous right/wrong, 1/0. A practice data set is available in wizirt that can be called up using the data() function.

\[
\text{data("responses")}
\]

(20)

The wizirt package includes a very useful function that runs almost all of the other functions found in wizirt. This is irt_report(). It can be used to create a general report based on
the data given to it. The irt_report() function has many arguments that are based on the functions in wizirt (since irt_report() combines them). These arguments include the data, report_type (only html is available right now), title, author, note, engine, and rownames. Other arguments can be found in the documentation (?irt_report).

A Function for New R Users

The wizirt package also includes another function intended to ease the user into R programming: irt_report(). This function requires at a minimum data to be input (other functions can be seen in the package documentation). It then calls the majority of functions in the wizirt package in a sensible order and arranges the information in an html report. An example of this report can be found at https://isaacpfleger.com/resources/dissertation/example_report.html. Screenshots of the report were presented in the Methods section (Figure 2-Figure 4).

\[
\text{irt_report}(\text{data = data}, \text{title = "My Report"}, \text{author = "Isaac"})
\] (21)

Functions for More Flexibility

The irt_report() function is designed for convenience and to gradually break in the new R user. For those who are more familiar with R, there are a series of functions that provide greater access to the capabilities of wizirt. The simplest wizirt model can be specified simply by passing data to the function wizirt().

\[
\text{mod} \leftarrow \text{wizirt}(\text{data = "responses"})
\] (22)

Some users would be interested to know how fast wizirt is. The estimation time can be extracted from the returned object:

\[
\text{mod$elapsed}
\] (23)

For reference, it takes the wizirt() function around 14 seconds to run a Rasch model with 1644 examinees and 170 items. It takes around 25 seconds to run a two-parameter logistic model
on the same data. This is without calculating the absolute fit statistics, which can add a considerable amount of time to estimation. The absolute fit measures are called for with the abs_fit argument. Passing “abs_fit = FALSE” to wizirt will turn off the estimation of the absolute fit measures.

By default, the wizirt function runs a Rasch model using the mirt package. However, a person can change the model with the item_type argument. Rasch, 1PL, 2PL, and 3PL are available for item_type. Likewise, a person can change the package doing the estimation with the engine argument. Possible engines include mirt, ltm, and eRm. The extensibility of wizirt allows other potential packages as engines to be added in future iterations.

\[
\beta_{j} \beta_{i} \beta (j, w_{i}, z_{i}, r_{t} = "tech") (25)
\]

To access the information inside of wizirt, use the print method with the type argument. Options for printing include descriptive statistics (‘desc’), technical information (‘tech’), missing data information from the perspective of the items (‘na_items’), missing data from the perspective of the persons (‘na_persons’), and the estimates for person and item parameters (‘person’ and ‘item’ respectively). For space reasons, I will not show each of these here, but Table 3 is an example of the output from a print function using the type = “tech” argument.

\[
\text{print}(\text{mod}, \text{type} = "tech") (25)
\]
**Table 3**

*Example of a Technical Summary Produced by wizirt*

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>package</td>
<td>mirt</td>
</tr>
<tr>
<td>function</td>
<td>mirt</td>
</tr>
<tr>
<td>version</td>
<td>1.32.7</td>
</tr>
<tr>
<td>call</td>
<td>mirt::mirt(data = data, model = 1, itemtype = &quot;Rasch&quot;, SE = T, TOL = 1e-05, verbose = F)</td>
</tr>
<tr>
<td>factors</td>
<td>1</td>
</tr>
<tr>
<td>item type</td>
<td>Rasch</td>
</tr>
<tr>
<td>converged</td>
<td>TRUE</td>
</tr>
<tr>
<td>method</td>
<td>EM</td>
</tr>
<tr>
<td>log-likelihood</td>
<td>-1025.62</td>
</tr>
<tr>
<td>criteria</td>
<td>1e-05</td>
</tr>
<tr>
<td>iterations</td>
<td>30</td>
</tr>
</tbody>
</table>

To get the other information, switch 'tech' with any of the other options ('desc,' 'na_items,' 'na_persons,' 'item,' 'person'). After estimating two models as we have done here, it is possible to compare them using the anova() function as shown in Table 4. However, you should make sure that the engines are the same in order to ensure comparability. They use different methods for optimization, so the log-likelihoods may not be comparable.

\[
\text{anova}(\text{mod2}, \text{mod}) (26)
\]

The is table contains the information for a likelihood ratio test. Here the X2 represents \( \chi^2 \). The function call for each model is depicted to differentiate models and some comparative fit statistics are included (AIC, AICc, SABIC, HQ, BIC, and log-likelihood).
Table 4

Example of a Likelihood Ratio Test for Comparing Models in wizirt

<table>
<thead>
<tr>
<th>AIC</th>
<th>AICc</th>
<th>SABIC</th>
<th>HQ</th>
<th>BIC</th>
<th>logLik</th>
<th>X2</th>
<th>df</th>
<th>p</th>
<th>call</th>
</tr>
</thead>
<tbody>
<tr>
<td>2101.240</td>
<td>2127.771</td>
<td>2080.384</td>
<td>2124.374</td>
<td>2159.177</td>
<td>-1025.62</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>mirt::mirt(data = data, model = 1, itemtype = &quot;Rasch&quot;, SE = T, TOL = 1e-05, verbose = F)</td>
</tr>
<tr>
<td>2118.753</td>
<td>2331.253</td>
<td>2077.040</td>
<td>2165.020</td>
<td>2234.627</td>
<td>-1009.38</td>
<td>32.48739</td>
<td>25</td>
<td>0.1443</td>
<td>mirt::mirt(data = data, model = 1, itemtype = &quot;2PL&quot;, SE = T, TOL = 1e-05, verbose = F)</td>
</tr>
</tbody>
</table>
Checking Assumptions

After estimation, it is possible to test the assumptions of the model, (a) unidimensionality, (b) local dependence, and (c) fit to the functional form. A single function is used to gather the necessary information to address these measures, irt_assume().

\[ assumptions \leftarrow \text{\texttt{irt\_assume}(mod)} \] (27)

The information needed to address the assumptions of unidimensional IRT is extracted using the print function. This time the type arguments are “unid” for unidimensionality, “ld” for local dependence, “rel” for relative fit, and “abs” for absolute fit. Relative and absolute fit are used to address the fit to the functional form assumption. Absolute fit is currently only available when the mirt engine is used.

Essential unidimensionality is checked using the sirt function conf.detect. The evaluations based on the cutoffs are included in the output (Table 5), but more technical information is available in the help documentation for sirt::conf.detect().

\[ \text{\texttt{print}(assumptions, type = "unid")} \] (28)

Local dependence uses a combination of the standardized LD statistic as calculated by mirt::residuals-method(). The p-value is also included. The correlations (ccov) are from the ltm package with its ltm::rcor.test() function. The statistics are calculated for each pair of items, and the pairs are arranged by decreasing values of the standardized LD.
Table 5

Example of a Table Reporting Unidimensionality Statistics

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Value</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETECT</td>
<td>-0.62</td>
<td>Essential Unidimensionality</td>
</tr>
<tr>
<td>ASSI</td>
<td>-0.29</td>
<td>Essential Unidimensionality</td>
</tr>
<tr>
<td>RATIO</td>
<td>-0.36</td>
<td>Essential Unidimensionality</td>
</tr>
<tr>
<td>MADCOV100</td>
<td>1.69</td>
<td></td>
</tr>
<tr>
<td>MCOV100</td>
<td>-0.62</td>
<td></td>
</tr>
</tbody>
</table>

There is a plot method for assessing local dependence as well (Figure 7). To conserve space, I will forgo the print method and only display the plot method.
Notice that the plot method is run on the original mod object and not the assumptions object. The type options that relate to local dependence are “ld” and “ld_pairs.” The type “ld” shows boxplots of the standardized LD for each item. The “ld_pairs” shows which items have been flagged as significant by the p-value associated with the item pairs ($\alpha = .05$).

In addition to printing the assumption object with type = “rel” or type = “abs”, model fit to the functional form can be tested at the item and person level. The topic of testing for model fit will be brought up again when discussing persons and items, though we leave it for a moment and turn toward plots.
There are a handful of plots that summarize information about the test generally. These are all accessed using the plot() function and the type argument. The type options that are related to summarizing the test are "tinfo," "SE," "tinfo SE," "theta SE," "theta," "diff," and "theta_diff." Figure 8 shows the plot with type = "tinfo." Some of these plots are included on their own page. When a plot is included on its own page, the code that created the plot is included above the plot as a reference. For example, the syntax in function 30 generates the plot shown in Figure 8 and the syntax in functions 31 and 32 generate Figure 9 and Figure 10 respectively.

\[
\text{plot}(\text{mod2, type = "tinfo")}
\]

**Figure 8**

*Test Information Function*
**Item-Level Statistics**

Previously, we saw that it is possible to get item-level statistics using the print function and setting type = "item." But there is more item-level information available through plot() and through irt_item_fit().

There are more plot methods that relate to the items than there are for the test generally or for person-level statistics. These are obtained using plot() in combination with one or more types. The types can be "obs" (Figure 9), "trace" (Figure 10), "resid" (Figure 11), "stand" (Figure 12), or "info" (Figure 13).

\[
\begin{align*}
\text{plot(mod2, type = "obs", item = 1)} \\
\end{align*}
\]

(31)

**Figure 9**

*Examinee Responses by Ability*
$plot(mod2, type = \text{trace}, item = 1)$

Figure 10

*Item Characteristic Curves*
\textit{Item Information Functions}

\begin{center}
\begin{figure}
\centering
\includegraphics[width=\textwidth]{item_information_functions.png}
\caption{Item Information Functions}
\end{figure}
\end{center}

\texttt{plot(mod2, type = "info", item = 1)}
plot(mod2, type = resid, item = 1)
\[ \text{plot}(\text{mod2}, \text{type} = \text{stand}, \text{item} = 1) \]  

Figure 13

*Standardized Residual Plot*

![Standardized Residual Plot](image)

It is possible to include multiple plot types in the plot argument. Any combination of these plot types can be included in the type argument, and they will be overlayed. For example, type = "resid trace" (Figure 14) can be used to help diagnose item-level misfit. The quads argument can be used to increase the number of quadratures (or bins) that are used to create the residuals. If more than one item is passed to the "items" argument, these will be juxtaposed in separate plots (Figure 15).
\texttt{plot(mod2, type = "resid trace", quads = 15, items = 1)}

Figure 14

\textit{Item Characteristic Plot with Residuals}
Users may prefer to see all of the items in a single plot with the items superimposed on each other. This can be accomplished by passing FALSE to the facets argument (Figure 16).
\begin{equation}
\text{plot}(\text{mod2, type = "trace", facets = FALSE})
\end{equation}

Figure 16

All Item Characteristic Curves Superimposed

Perhaps there are many items in the data set, and the plots come out squished. Users can pass a selection of items to the "items" argument. This can be either as the numeric positions of the items in the data set or as the names of the items as found in the data (Figure 17).
Combining with other code, we can subset the items based on some characteristic. For example, maybe we only want to plot the items labeled “A”, “E”, “I”, “O”, and “U”. The which() function is used to give us the position of the TRUE values in a vector of logical TRUE and FALSE values. We use this to get the positions of the items that have names in our chosen subset (Figure 18).
\[
\text{selection} < - \text{which(}\text{names(mod2$fit$data)} \text{) in} \{\text{"A","E","I","O","U"}\})
\]

\[
\text{plot(mod2,type = "trace", facets = FALSE, items = selection)}
\]

Figure 18

\textit{Flagged Item Characteristic Curves Superimposed}

This process of selecting items applies to all plots. For example, we could obtain the summed information function of the selected subsample (Figure 19).
While the plots focus on information from the estimated model, it is also possible to gather statistics related to item-level fit. This is done using the irt_item_fit() function, which includes the argument stats. The stats argument refers to item-level fit statistics. These are calculated using mirt::itemfit(). Options available include ‘Zh,’ ‘X2,’ ‘G2,’ or ‘infit.’ Note that this is not a complete selection of what is available in mirt. Other item fit statistics will become available at a future date. More information on the calculation of these measures is available in the documentation for mirt::itemfit(). The print method is used to get the output from irt_item_fit (Table 6).
\[
\text{if} \ a < -\text{irt\_item\_fit}(\text{mod, stats = "Zh"})
\]

\[
\text{print} (ifa) \% > \%\text{slice} (1:5)
\]

Table 6

Example of an Output from \text{irt\_item\_fit()}

<table>
<thead>
<tr>
<th>item</th>
<th>difficulty</th>
<th>discrimination</th>
<th>guessing</th>
<th>Zh</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-0.74</td>
<td>1</td>
<td>0</td>
<td>6.21</td>
</tr>
<tr>
<td>B</td>
<td>-1.07</td>
<td>1</td>
<td>0</td>
<td>6.02</td>
</tr>
<tr>
<td>C</td>
<td>-1.38</td>
<td>1</td>
<td>0</td>
<td>5.87</td>
</tr>
<tr>
<td>D</td>
<td>-0.87</td>
<td>1</td>
<td>0</td>
<td>6.13</td>
</tr>
<tr>
<td>E</td>
<td>-1.00</td>
<td>1</td>
<td>0</td>
<td>6.06</td>
</tr>
</tbody>
</table>

Person-Level Statistics

Working with person-level statistics is very similar to working with item-level statistics. For example, there is an \text{irt\_person\_fit()} function that provides a number of person-fit statistics. These are obtained from the PerFit package. Options for the stats argument in \text{irt\_person\_fit()} are \text{"lz", "lzstar", "NCI", "E.KB", "D.KB", "A.KB", "H^T", "ZU3", "U3", "Cstar", "C.Sato", or "G."} The default is \text{"H^T."} Cutoffs are created from \text{PerFit::cutoff()}. Users are directed to the documentation of PerFit for further information regarding these statistics or the way cutoffs are created.

Additional statistics are available for Rasch models from \text{sirt::personfit.stat()}, though cutoffs are not created for these. These extra stats are \text{"caution," "depend," "ECI1," "ECI2," "ECI3," "ECI4," "ECI5," "ECI6," "l0," "infit," "rpbis," and "rpbis.itemdiff."} The stat option \text{"infit"} will calculate both infit and outfit.
There is less available to person statistics in terms of plots. Right now, the only plot is the nonparametric persons response function, which is produced by setting the type argument to 'np_prf' in the plot function. Because there are frequently too many persons to be included in a single plot and to still have it be visible, a `plot_wrap()` function is included that works with item and person plots (Figure 20-22).

\[
\text{plot_wrap}(\text{mod2, type} = "np\_prf", \text{persons\_per} = 30, pfa = pfa) \quad (46)
\]  

**Figure 20**  

*Person Response Functions for Persons 1-25*
Figure 21

*Person Response Functions for Persons 26-50*
Figure 22

Person Response Functions for Persons 51-75

[Graph showing person response functions for ages 51 to 75, with abscissa labeled 'Item Difficulty' and ordinate labeled 'Person Response Functions'.]
Alternatively, all persons can be plotted together by turning facets to FALSE (Figure 23).

\[
\text{plot}(\text{mod2, type = "np_prf"}, \text{facets = FALSE, pfa = pfa})
\]

(47)

**Figure 23**

*All Person Response Functions Superimposed*

Individuals may be interested in plotting only those persons who are flagged as aberrant. This can be done with the persons argument, the same way the which function was used in conjunction with the items argument to plot only items with negative discrimination (Figure 24).

\[
\text{selection} \leftarrow \text{which(pfa$person\_estimates$flagged})
\]

(48)

\[
\text{plot}(\text{mod2, type = "np_prf"}, \text{facets = FALSE, pfa = pfa, persons = selection})
\]

(49)
Figure 24

Selected Person Response Functions Superimposed

Person fit in wizirt does not end there. The wizirt package was conceptualized as a result of the realization that software for person-fit assessment (PFA) are inconveniently separated from software for estimation. With this in mind, the first four steps of Rupp’s (2013) framework for PFA are facilitated by wizirt. Global detection and graphical exploration have been demonstrated in the previous pages. The current version of wizirt does not include statistics for local misfit detection, but it does identify each unique response pattern and calculate the frequency of each.
To show the person-fit statistics, ability estimates, and response patterns, use the `print()` function with the `patterns = TRUE` argument. The argument `item_order = "by_diff"` is also useful. The default is to print the items in the order they appear in the dataset (Table 7).

\[
\text{print}(pfa, \text{patterns} = T, \text{item_order} = "by\_diff") \% > \% \text{slice}(1:3)
\]

Table 7

Example of Person-Level Statistics, Response Patterns by Difficulty

<table>
<thead>
<tr>
<th>ability</th>
<th>std_err</th>
<th>ids</th>
<th>Ht</th>
<th>Ht_cut</th>
<th>flagged</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.02</td>
<td>0.53</td>
<td>1</td>
<td>0.0044</td>
<td>-0.044</td>
<td>FALSE</td>
<td>11111111110111001100110111</td>
</tr>
<tr>
<td>0.66</td>
<td>0.64</td>
<td>2</td>
<td>0.0174</td>
<td>-0.044</td>
<td>FALSE</td>
<td>1110111111111111111101110</td>
</tr>
<tr>
<td>-0.20</td>
<td>0.51</td>
<td>3</td>
<td>0.0040</td>
<td>-0.044</td>
<td>FALSE</td>
<td>01110011111100110011110011</td>
</tr>
</tbody>
</table>

Alternatively, `item_order` can be a character or numeric vector of item names or positions. This can be less than the total number of items for a subset pattern (Table 8).

\[
\text{print}(pfa, \text{patterns} = T, \text{item_order} = 1:5)
\]

Table 8

Example of Persons-Level Statistics, Selected Items in Response Patterns

<table>
<thead>
<tr>
<th>ability</th>
<th>std_err</th>
<th>ids</th>
<th>Ht</th>
<th>Ht_cut</th>
<th>flagged</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.02</td>
<td>0.53</td>
<td>1</td>
<td>0.0044</td>
<td>-0.044</td>
<td>FALSE</td>
<td>10110</td>
</tr>
<tr>
<td>0.66</td>
<td>0.64</td>
<td>2</td>
<td>0.0174</td>
<td>-0.044</td>
<td>FALSE</td>
<td>11111</td>
</tr>
<tr>
<td>-0.20</td>
<td>0.51</td>
<td>3</td>
<td>0.0040</td>
<td>-0.044</td>
<td>FALSE</td>
<td>01010</td>
</tr>
</tbody>
</table>

Modeling reasons for person fit can also be done using the `irt_model_pfa()` function. This command uses the Conijn et al. (2011) multilevel method. The data are randomly divided into a
certain number of bins (specified by the bins argument). Then person-fit statistics are calculated on each of the bins, and a multilevel model is run. Here a list or data frame of predictors can be included. This data does not have any predictors, so I made a fake one to show how it works.

\[
\text{age} < - \text{round}(\text{runif}({\text{nrow}}(\text{pfa}\$\text{person_estimates}), 17, 25))
\]

\[
pfa\_mlm < - \text{irt}\_\text{model}\_\text{pfa}(	ext{mod2, pfa = pfa, predictors = list(age = age)})
\]

The ICC is a key statistic with this model. A user can get it simply by indexing the resultant object.

\[
pfa\_mlm\$\text{icc}
\]

The other object within pfa\_mlm is the multilevel model object returned from blme. To learn more about this object, check out the documentation for blme::blmer().

\[
\text{summary(pfa\_mlm}\$\text{models}\$\text{Ht})
\]

Learning Resources

GitHub Website

The most helpful single resource for those who wish to learn wizirt is the GitHub website I built for that purpose (https://pflegermeister.github.io/wizirt/). The main page of the website includes a brief introduction to the package (Figure 25). This introduction guides the user through installing the package and then checking to make sure it works by creating a simple report from the data native to wizirt.

Several other pages appear on the site, including a Reference page, where all functions in the package are listed with explanations, details, and examples. A Changelog presents a list of changes to the software over time. As of now, this page simply has a list of the package goals at the start of its development. A third page, called articles, includes a handful of short vignettes using the package with different goals in mind.
Package Vignettes

To help new users learn wizirt, I built three vignettes; the first demonstrating all of the functions in the wizirt package, a second based loosely on an example from De Ayala (2009), and a third walking through the use of Rupp’s (2013) framework for detecting and explaining person misfit. For space, I will not repeat all three vignettes here, but they can be found on the GitHub website. The Overview of wizirt section of this paper is based on the overview vignette. The other two can also found in Appendix C.

Cheat Sheet

Cheat sheets are one of the most popular methods of communicating R programming knowledge. Even advanced users reference cheat sheets in their work. A cheat sheet is most often a single sheet of paper printed on both sides with the (a) various package functions grouped, (b) key arguments described, and (c) operations depicted. The cheat sheet I developed is barer than some others as wizirt is a young package and will likely evolve as more functions
are added. However, it still serves the purpose of a cheat sheet as part reference and part gallery as illustrated in Figure 26 and Figure 27.

**In-R Reference Pages**

Another common tool used to help R programmers is the question mark. By putting a question mark in front of a function name (e.g., `?print`), a reference page is displayed. This reference page describes the function, lists the arguments to the function, defines those arguments and their acceptable values, lists the output from the function and explains it, and provides examples of the use of the function.

While building wizirt, I also developed these reference pages. These can be found on the GitHub website for the package, or they can be shown in R by calling `"?"` and then the name of the function (i.e., `?irt_assume`). To save space, I will not include these reference pages here, however I do include a screenshot of the help documentation for the wizirt() function (Figure 28).

**Accuracy Checks**

To ensure that my package was making appropriate calculations, I wanted to make sure that I could replicate what was found in commercial software (i.e., IRTPRO and Winsteps) and in the literature.

**Comparison to IRTPRO.** The comparison to IRTPRO for the Rasch model was straightforward. I decreased the convergence criteria in wizirt using the "tol" argument, thus requiring more accuracy from the algorithm. It should be noted that tol does not work except with the mirt engine.
Figure 26

Cheat Sheet for wizirt Front Page

Item response theory with wizirt: CHEAT SHEET

devtools::install_github("Pflegermelster/wizirt")

### Item-fit Assessment

- **Item-fit analysis**
  ```r
  int_item_fit(model, type = "type")
  model = an object from fit, wizirt
  type = One of "Y", "Y'", "C", or "left"
  ```

### Modeling

- **DEFINITION**
  ```r
  item_type = [type] %n+1
  item_type = "Rasch", "3PL", "2PL", "1PL"
  ```
- **SELECT ENGINE**
  ```r
  set_engine([eng], [type])
  eng = R, Rstudio, SAS, or SPSS
  type = item or person
  ```
- **FIT**
  ```r
  fit_wizirt(data)
  data = Full data of responses
  ```

### Person-fit Assessment

- **Person-fit Analysis**
  ```r
  int_person_fit(model, type = [type])
  model = an object from fit, wizirt
  type = One of "Y", "Y'", "C", or "left"
  ```

### Assumptions

- **assumptions**
  ```r
  assumptions <- int_assumptions(model)
  print(assumptions)
  print(assumptions, type = "rel")
  print(assumptions, type = "abs")
  print(assumptions, type = "uniq")
  print(assumptions, type = "int")
  ```

### wizirt

- **model**
  ```r
  model <- wizirt(data, item_type = "type", engine = "wizirt"
  ```

- **Explaining Misfit**
  ```r
  pfa_nlm <- int_model_pfa(model, pfa = pfa, predictors = predictors)
  pfa_nlm_summary
  summary(pfa_nlm(models))
  ```

### General Summary

- **Generate an overview of an item using all of the defaults of wizirt**
  ```r
  int_report(data, title, author, meta)
  ```

- **Title**
  ```r
  data = File name of dichotomous response data
  ```

- **Author**
  ```r
  author = String to put under the title
  ```

- **Note**
  ```r
  meta = A statement to be included at the top of the report
  ```

### References

wizirt relies heavily on other packages for all calculations. The following packages are used behind the scenes for estimation:

- mirt
- mix
- ltm
- eRm
- PoF
Figure 27

Cheat Sheet for wizirt Back Page
Figure 28

Screenshot of wiizirt::wiizirt() Help File

Estimate an IRT model

Description
Estimate an IRT model using various engines

Usage
wiizirt(
  data,
  rownames = NULL,
  item_type = "Rasch",
  engine = "mirt",
  tol = 1e-05
)

Arguments
data An Person x items matrix or dataframe of dichotomous response values (e.g. correct/incorrect). Rows are persons and columns are items, one row per person, one column per item. No other information allowed.
rownames Optional unique row IDs for the data (i.e. examinee IDs). If omitted, uses 1:nrow(data).
item_type Character. Must be one of "Rasch", "1PL", "2PL" or "3PL".
engine Character. Currently supported engines are "mirt" and "ltm" for Rasch, 1PL, 2PL, and 3PL models. "eRm" is supported for Rasch models only.
tol Numeric. Convergence criterion. Currently only implemented when engine is mirt.

Value
Returns a list of class wiizirt. spec is a list of information for the parsnip backend. printing spec prints a summary of the model run. elapsed contains the time it took the model to run. fit contains the model information:

- data is the data passed to the model
- model contains model fit information, including:
  - engine a list with values pkg (the package used for estimation), ver (pkg version), func (function used from pkg), and call (call made to pkg)
  - n_factors the number of factors estimated
  - item_type the item type passed to wiizirt (Rasch, 1PL, 2PL, or 3PL).
  - estimation a list with information related to convergence. convergence a T/F value of whether the model converged, method the estimation method, criteria the convergence criteria, iterations the number of iterations it took for the model to converge, log_lik the loglikelihood at convergence, abs_fit the absolute fit of the model, df the number of parameters estimated.
  - parameters a list of estimated parameters. coefficients is a data frame of estimated item-statistics and persons is a data frame of estimated person statistics.
- original_object is the object returned from the engine.

Examples
data("responses")
my_model <- wiizirt(data = data, item_type = "2PL", tol = 1e-4, engine = "mirt")
print(my_model, type = "text")
print(my_model, type = "descriptive")
print(my_model, type = "item")
print(my_model, type = "person")
print(my_model, type = "na_item")
anova(my_model)

[Package wiizirt version 0.0.30]
To match wizirt’s 2PL (with the mirt engine), in IRTPRO I ran a 2PL model, but I constrained all of the slopes to be 1 and fixed the mean of the estimates to be zero. Then IRTPRO and wizirt produced identical estimates.

The data, control file, and R script are available on GitHub at https://github.com/Pflegermeister/wizirt-accuracy-checks. Note that the data for running it in R are found natively in the wizirt package. To get the same output from IRTPRO as wizirt for the 1PL model, I had to constrain the covariances to be 1 and the mean to be 0. The same worked for the 2PL model, though without making the convergence criteria more stringent some estimates were off by up to 0.02. This did not affect the ranking of the items in terms of difficulty or discrimination. The differences in item estimates disappeared when the convergence criteria were made stricter (i.e., 1e-8), though there were minimal differences in estimates of person abilities on the extremes.

The wizirt model did not converge for the 3PL model with this data set. IRTPRO included priors to encourage convergence. Priors are not yet available in wizirt. Unfortunately, wizirt was unable to replicate the 3PL model’s estimates. The inability for wizirt to converge likely stems from the inability to add Bayesian priors. Priors are a way to tell a statistical model what the output is expected to look like, and they are a very powerful tool to aid convergence.

Comparison to Winsteps. Constraining Winsteps and wizirt to produce the same statistics was more difficult, as was alluded to on the Winsteps website (https://www.winsteps.com/winman/compatibility.htm). Linacre explains that “comparing or combining measures across packages can be awkward (Linacre, n.d.).” He further describes that
a choice of origin point, scaling multiplier, and handling of extreme scores can affect the estimated value.

Instead of constraining software to get the same values, I standardized the estimates from wizirt and the estimates from Winsteps. This allowed me to compare the estimates without worrying about the mean or scale since Linacre suggests these are the main differences. Ultimately, I was able to replicate Winsteps’s parameter estimates after standardizing both estimates, though with a few differences that were very small (<.02). The correlation between the estimates was greater than .99.

**Sinha Ray Replication.** The Sinharay (2017) replication (Appendix D) was an excellent tool to check the accuracy of wizirt. There were a few differences in results between wizirt and the Sinharay study. I was able to determine that the cause of the difference was that Sinharay used a different method to calculate his theta estimates than I did. When I used the same method he did, I got the same answer. Even without using the same method for extracting theta estimates, the correlations between his results and mine were greater than .95. Because of this, I did not change the method that wizirt uses to extract theta estimates. More details on the replication can be found in Appendix D. This includes my code used to perform the analysis. However, I have not included the code Sinharay used, and the data for the replication are not publicly available.

**Informative Checks**

While considering the informative value of the wizirt package, I focused on the framework for PFA by Rupp (2013) and the checklist for IRT by De Ayala (2009). To help guide my development efforts, I created my own checklist, which contained every piece of information I needed to include in my package (Appendix B). I stuck to this checklist very closely throughout
development, it was used to help me determine what problems were important to solve right now, and what problems needed to wait until later.

Because I held tightly to this checklist, I can report that I met each of the requirements for information, except for a small number. To summarize them, absolute fit is only calculated for some models, I have not implemented an automatic way of flagging poorly performing items, and I did not include a local person-fit statistic. These deficiencies are minor because there are still ways to get the same information using wizirt.

The most notable deficiency in wizirt is that I do not include function-level information about how missing data are handled. I intend to include specific information regarding how missing data were handled. The missing data handling depends on what activity is being done. For parameter estimation, full information is always used to handle missing data. For estimating person-fit statistics, pairwise information is generally used, though in some cases, listwise might be occurring. In most, if not all, other situations, listwise deletion occurs. This information is lacking from the output itself.

Usefulness Checks

Usability Testing. There were three rounds of usability testing. The first round of testing came early in the development process. Three individuals who were familiar with both IRT and R were asked to test out the wizirt software; we then met virtually so that I could record their feedback. All three of these users were excited at the prospect of the irt_report() function. One of these, an advanced R user, indicated that while they favor the mirt package for their work, the irt_report() function is very tempting and would be the tool that would lead them to switch.
These early users did not favor the layered, parsnip-style syntax, so I created the wizirt() function. I was pleased with the simplicity of the single function syntax. I think this was one of the more valuable suggestions I received.

The second round of testing came near the end of development. In this round of testing, I gave three individuals a virtual tour of wizirt’s capabilities. Again, I recorded the meeting to be able to review their feedback. One of these three had also been one of those to test wizirt in the first round and was competent in both R and IRT. The other two testers were familiar with IRT but were less comfortable with R, though they had been exposed to it before. Users were once again pleased by the irt_report() function and the offerings of the wizirt package. They were eager to see the package extended for differential item functioning and for multidimensional IRT. Their strongest point of constructive feedback was that the output was sometimes unclear. For example, one user confessed that they were unfamiliar with some of the statistics used and found some columns with statistics names as labels confusing.

The lack of clarity has proved to be a weakness in these early stages of the wizirt software, as was seen in the third and final stage of testing. Two of the individuals who were asked to use wizirt in the beginning were asked to review a sample of the output; their main point of feedback was that some of the acronyms and terms included in the output were unfamiliar. This made it difficult for them to interpret some of them. However, this criticism should not be taken to suggest these users were dissatisfied. In fact, the users were strongly in favor of the new package. All of the users were excited by the package, and one user went as far as to say that they were amazed.

Another user was asked to test out the package and helped identify a bug that prevented installation. This was quickly remedied.
Other Evidence of Usefulness. Usefulness has been front and center in the development of wizirt. The decision to let the wizirt() function overshadow the layered, parsnip-style syntax of the irt() function was to increase the experience for wizirt users. A second effort I made in this regard is constraining the output to primarily consist of tibbles. Tibbles are a special form of data frame in R that are aesthetically pleasing and easier to work with than some other objects returned in R. This increases the ability of a package to merge into existing workflows by giving the user a consistent form.

It is my personal evaluation that the package is useful, but I recognize that there are some limitations to this usefulness. The biggest issue is the way information is extracted. Currently, what most information users would be interested in is extracted from a wizirt object using the print() method. I believe this to be slightly inconsistent with R conventions. This may disrupt the user's workflow and potentially even confuse the user.

Aesthetic Checks

In designing wizirt, I said I would follow the four core principles of design, (a) repetition, (b) alignment, (c) contrast, and (d) proximity. Proximity and repetition are seen in the common structure that holds across all my products. I grouped all information into three categories: information related to the test, information related to items, and information related to persons. This structure informs everything: the way wizirt is intended to be used, how the cheat sheet is designed, and the format of the outputted reports.

Repetition also plays a role in the decision to have all output be in either list or tibble format. Lists are the most common format for outputting information in R. They are only used in wizirt for returning information from estimation functions, like wizirt and irt_person_fit(). Within those lists, all the data are stored in tibble form. By constraining the output to tibble
format I am reducing the user’s cognitive load, who only needs to know one way of interacting with wizirt objects. This also encourages wizirt objects to be part of a larger workflow.

Contrast is used primarily to enhance the attractiveness of the report and to make important information clearer. For example, it is used with the local dependence plots to highlight item pairs tagged as potentially being dependent (Figure 7). The flagged pairs show a deep crimson, while the pairs that have not been flagged are white. This helps the user to identify violating items immediately. The same holds true in the associated local dependence table. Likewise, persons flagged as aberrant are also identified with a shade of pink in the person response function plot. This contrasts with the blue of the other plots, immediately drawing the user’s eye.

Alignment was primarily used when giving user’s the opportunity to overlay plots. Beyond that, alignment is used in the layout of the summary report generated by irt_report(). All headers share the same left alignment. Numbers are aligned left in tables, while text is aligned right. The table of context is aligned left in a separate section with a natural interface for navigation. In all, I am pleased with the way the summary report looks and with the plethora of praise the report received from testers. This was generally seen as the most appealing aspect of the package.
CHAPTER 5

Discussion

The purpose of this dissertation was to develop, evaluate, and systematically revise and improve a comprehensive, integrated R package that can be used to conduct person-fit analyses in IRT. As a comprehensive tool, wizirt approaches IRT in a way that other software do not. For example, it explicitly provides functions for exploring the assumptions of IRT, with an extra focus on person-fit assessment. No other IRT software incorporates PFA in the same way that wizirt does. This enables users to determine the validity of the results from their models.

This new package greatly simplifies some aspects of conducting an IRT analysis in R, and it significantly lessens the learning curve for those new to R. This package effectively unites powerful functions from several packages into a sensible workflow and provides users with the information necessary to perform rigorous psychometric research on unidimensional, parametric IRT models with dichotomous data. However, as a package that has only lived through a few months of development, wizirt still has several limitations, and there are features that I would like to add in the future.

One area in which I failed to meet my goals was incorporating of a local person misfit statistic, which was called for by Rupp (2013). My original intent was to include the ICI, the only local fit statistic I could easily identify. However, the ICI aims to identify decreasing learning in students and requires parallel forms on a test. This made it less desirable and less practical in wizirt. It brings up a point worth discussion. Rupp suggested that local fit statistics should be implemented, but there does not seem to be a robust literature on that topic at this time. Perhaps this warrants further investigation. It may be that the information that would be provided by local detection statistics might be found more easily in person response functions.
Limitations

The most frequent criticism of wizirt given both formally and informally is the interpretability of its output. There are issues on some plots that obscure information. For instance, some of the plots are zoomed out too far. In others, the axes are squashed and difficult to read. In one, colors separate item versus person parameters, but the colors are not easily distinguishable. I have also failed to test the plots for accessibility for colorblind individuals.

Beyond the plots, the output itself is sometimes not clear. Most of the users were not familiar with PFA, and the person-fit statistics reported were unfamiliar and therefore seemed to them to be without context. The multilevel aspect of PFA in wizirt was similarly confusing to users. This goes contrary to my vision of wizirt as a software that will put PFA in the hands of more researchers. Future versions of wizirt will need to have better explanations of the statistics provided, and the plots will need to be refined.

At this point in the design of wizirt, there are some things with which I am not content. The accessibility of convergence information is one concern of mine. As of right now, users can access a dichotomous variable that says whether or not the wizirt model converged. Users are not warned if their model did not converge, except by whatever engine was chosen. I wish that the information about convergence issues was more standardized and robust. I would also like to test this with additional data and more test cases.

Another aspect of wizirt that I am not content with is the data interface, which I think will feel clunky and restrictive as more complicated models are added to the data. Currently, users are asked to include response data, row names, and predictors separately. I would like to make it possible to introduce these in a single, long-form data set.
I am also certain that I have relied too heavily and inappropriately on the print method to deliver information from wizirt functions. These print methods are currently experiencing a bug where they do not auto print like other print methods in R, which means they must be called explicitly. This feels awkward to me, and I would like to see this more in line with other R packages.

The final aspect that I am not content with is an issue with the degrees of freedom. Currently, degrees of freedom are off by one with mirt models. This is one area where I did my own calculations, because I had difficulty taking models generated by other packages and feeding them into some functions in the mirt package. Unfortunately, my calculations are not dynamic with the engine used (which have different constraints). This is an issue with which I am still wrestling. I am looking for a solution that will not add to the workload of other developers who may want to add other engines in the future.

Future Work

My intent in building the wizirt package has been to build a software suite that would make PFA more easily accessible to the researcher. Over the course of this project, I have wondered if PFA is limited for more reasons than simply a lack of software. The statistics that I have used are not powerful enough to lead a researcher to say with too much confidence that they had identified a person who did not fit. For example, review the replication of the work by Sinharay (2017; Appendix D). The $H^T$ statistic, which is praised in the literature, only detected known cheaters 27% of the time at its best, while also incorrectly identifying 8% of non-cheaters. While this simulation does not investigate other forms of person misfit, I think the point is still valid. Simple global detection statistics based on response patterns are not enough.
I do not think that this means PFA is an unworthy endeavor. I do think that this shows the wisdom of Rupp’s proposed framework. Detection is only the first step in a bigger process. I think that wizirt will give researchers the tools they need to explore PFA further and to develop better fit statistics. Additionally, I hope to provide users with a greater variety of more modern statistics. A new statistic for log-normal response time models that has been developed by Sinharay (2018) is one that I would like to implement in the future. I would encourage those who want to add to wizirt to contribute by strengthening the documentation around fit statistics, or even by adding more statistics, though there are many areas in which wizirt could grow.

I would like to describe how others might contribute to this package more fully, so others can add more. Beyond this, I want to make sure that it is impossible for someone to use wizirt without knowing what packages are doing the estimation.

One problem I have noticed is that the different estimation packages put the parameters on their own scale. If these are standardized, they are nearly identical. I would like to include an option that puts all the parameter estimates on the same scale. Perhaps I could scale the item parameters and then estimate the person parameters in a second step using the newly standardized coefficients.

In the future, I would also like to add additional report types, more engines, and more complicated model types and model specifications (e.g., polytomous, multidimensional, response times, computer adaptive, constraints, priors).

**Conclusion**

In the end, the wizirt package successfully does what I set out to do with it. With wizirt, users can easily incorporate PFA into their IRT analysis. Thus, wizirt has the potential to increase the widespread use of person-fit measures in practice, and their continued development
in research. It has the added benefit of serving as a starting point for individuals who want to learn to use R but are otherwise intimidated, because it provides the `irt_report()` function. This function allows the new R user to do more than they could otherwise. The function also provides all of the code used to create the report, so users can see what was done and learn from it. Even the more advanced `wizirt()` function retains the call it makes to the engine to run the model requested so users can learn from `wizirt` in that way.

It is my sincere desire that this package will continue to be improved, both by myself and by other individuals who want to see a comprehensive, unified interface for IRT that is easily accessible and well-documented.
REFERENCES

Information from articles with an * is exclusively included in Table 1.


https://doi.org/10.1111/jedm.12143


Kuhn, M., & Vaughan, D. (2020). *Parsnip* (Version 0.1.3) [Computer Software].
https://CRAN.R-project.org/package=parsnip


https://www.winsteps.com/


https://doi.org/10.1111/jedm.12208


Mair, P., Hatzinger, R., & Maier, M. J. (2020). *eRm: Extended Rasch Modeling* (Version 1.0-1) [Computer Software]. https://cran.r-project.org/package=eRm

https://doi.org/10.1207/s15324818ame0803_5
https://doi.org/10.1177/01466210122031957

https://doi.org/10.1177/1073191115577800


https://doi.org/10.1177/0146621616649963


https://doi.org/10.1080/08957347.2017.1353990


https://doi.org/10.1080/15366367.2018.1437308


## APPENDIX A

### Table 11.1 from De Ayala (2018)

<table>
<thead>
<tr>
<th>Desideratum</th>
<th>Manuscript Section(s)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The construct of interest is defined.</td>
<td>I</td>
</tr>
<tr>
<td>2. IRT is justified as the appropriate measurement approach (e.g., continuous latent variable vs. categorical latent variable).</td>
<td>I</td>
</tr>
<tr>
<td>3. The specific model(s), with description and justification, are provided.</td>
<td>I, M</td>
</tr>
<tr>
<td>4. The response data are fully described, including sampling, sample size(s), demographics, and testing environment (if appropriate).</td>
<td>M</td>
</tr>
<tr>
<td>5. All instruments are fully described, including length, response format, and validity evidence (if appropriate).</td>
<td>M</td>
</tr>
<tr>
<td>6. Software and estimation approach(es) are fully specified.</td>
<td>M</td>
</tr>
<tr>
<td>7. Estimation problems are documented, as are details as to how they were addressed.</td>
<td>R, D</td>
</tr>
<tr>
<td>8. A complete description is provided of how missing data were addressed.</td>
<td>R, D</td>
</tr>
</tbody>
</table>

**Item-focused studies** (linking, item bank construction, instrument construction).

| 9. Details regarding model fit analysis are provided, including those related to dimensionality, fit statistics, invariance, and model selection (if appropriate). | M, R, L |
| 10. Details regarding item fit analysis are provided, including those related to conditional independence, functional form, fit statistics, invariance, predicted vs. observed item response functions, and handling of misfitting items. | R, D     |
| 11. Instrument calibration results are presented (item parameter estimates and/or summary statistics, total information function). | R, D     |

**Person-focused studies** (CAT, diagnosis, equating, vertical scaling).

| 12. Person fit analysis results are presented, including fit statistics and appropriateness measurement. | R, D     |
| 13. Person location estimate results are described, including relevant standard errors. | R, D     |
| 14. Methods of equating scores on different metrics are described in detail. | M, R     |

* I = Introduction, M = Methods, R = Results, D = Discussion.
APPENDIX B

Checklist for wizirt

The wizirt package represents the work of my dissertation. As such I have agreed to do the following for my dissertation:

- Provide general model information
  - model call
  - description of model run
  - software and version
  - estimator
  - convergence status, criteria, values
  - estimation issues
  - response data described (sample size, demographics)
  - descriptions of items and test
  - missing data summary
  - description of missing data handling

- Provide model-fit information
  - unidimensionality (DETECT)
  - absolute fit
  - relative fit
  - ANOVA method for comparing models

- Provide item information
  - parameter estimates
  - summary statistics
- Provide item-fit information
  - conditional dependence
  - functional form
  - fit statistics
  - predicted vs observed item response functions
  - misfitting items flagged
- Provide person information
  - person location estimates (including SE)
- Provide person-fit information
  - global detection ($H^T$, $L_z$, infit and outfit, with cutoffs where applicable)
  - local detection (ICI or other)
  - tabulation or presentation of response patterns (Winsteps tables as guides)
  - PRF (nonparametric, parametric may follow later)
  - MLM
- Two levels of accessibility
  - comprehensive report for beginners
  - flexible functions for advanced users
- Learning Resources
  - GitHub website
  - CRAN-style reference page
  - tidyverse-style cheat sheet
  - Karabatsos’s simulation replication article
• Quality Assurance
  o Accuracy check
    ▪ match IRTPRO, Winsteps
    ▪ replicate Karabatsos
  o Informative check (meet all the checks above two levels…)
  o Usefulness check
    ▪ usability testing
    ▪ call it from python on separate laptop
  o Aesthetic check
    ▪ repetition
    ▪ contrast
    ▪ alignment
    ▪ proximity
APPENDIX C

Vignettes

These vignettes have been modified to conserve space. Tables that are found in the online version of the vignette are not found here. These tables are largely redundant, as similar output can be seen in the Overview of wizirt section. Additionally, these vignettes do not follow the same formatting as the rest of the paper. They more closely match the formatting of the GitHub webpage from which they are taken.

Vignette Based on De Ayala (2009)

This vignette demonstrates the use of the wizirt package by loosely following the example in chapters two-6 of De Ayala (2009) with particular emphasis on chapters two-4. It is important to note that wizirt does relatively few calculations on its own. Instead, wizirt provides a common syntax that is used to work with other packages. Estimation is done invisibly using packages like mirt, ltm, sirt, and PerFit. See the wizirt package documentation for information.

For this vignette we use the following packages:

\[
\text{library}(\text{wizirt})
\]

\[
\text{library}(\text{mirt})
\]

The data that De Ayala used is available in the mirt package. It is response data from a five-item math test. These responses can be loaded into R using the following code:

\[
\text{data(}"\text{deAyala}\text{")}
\]

\[
\text{responses} \leftarrow \text{expand.table}(\text{deAyala})
\]

In chapter 2, De Ayala first estimates a Rasch model. We can do this in wizirt:

\[
\text{mod1} \leftarrow \text{wizirt}(\text{responses, item_type = } "\text{Rasch}\text{", engine = } "\text{mirt}\text{")}
\]
Here we have first identified our data. The data must be a person by items matrix or data frame of dichotomous responses. The item_type argument defines whether the model is a Rasch, 1PL, 2PL, or 3PL model. The engine argument specifies what package does the parameter estimation behind the scenes. We choose mirt as the engine because it is fast. After running the model, De Ayala presents some summary statistics:

\[
\text{print}(\text{mod1}, \text{type} = \text{"desc"})
\]

Some technical information is also presented:

\[
\text{print}(\text{mod1}, \text{type} = \text{"tech"})
\]

The technical information includes information about the model specified, including the code run behind the scenes. It also provides information about convergence. In this same chapter, De Ayala mentions three assumptions for IRT:

- Unidimensionality
- Conditional or local independence
- Functional form

In wizirt, evidence for each of these assumptions can be estimated using the function irt_assume():

\[
\text{assumptions} \leftarrow \text{irt\_assume}(\text{mod1})
\]

The evidence for each assumption can be extracted from the model using the print() function in conjunction with the type argument. For example, if we want information for the unidimensionality we can write the following:

\[
\text{print}(\text{assumptions}, \text{type} = \text{"unid"})
\]
This displays the DETECT, ASSI, and RATIO statistics as calculated by sirt::conf.detect(), as well as the conclusion that can be drawn from each statistic.

Conditional independence can be displayed in a similar way:

```
print(assumptions, type = "ld")
```

This displays a table with standardized LD statistics and p-values as calculated by mirt::residuals-method() and correlations as calculated ltm::rcor.test(). This can be used to identify pairs of items that potentially violate this assumption. There is also a plot method that can be used for the same purpose. However, because all items in this data set are marked as potentially violating this assumption, the plot is not very informative here.

```
plot(mod1, type = "ld_pairs")
```
Information about fit to the functional form is available in wizirt through several absolute-, relative-, person-, and item-fit statistics. Currently, absolute-fit statistics are only available for models generated using the mirt engine. These statistics can be displayed using the type = “abs” argument for absolute-fit measures and type = “rel” for relative-fit measures.

\[
\text{print}(\text{assumptions, type } = \text{"abs")}
\]

\[
\text{print}(\text{assumptions, type } = \text{"rel")}
\]

Person-fit and item-fit statistics are available through the \texttt{irt_person_fit()} and \texttt{irt_item_fit()} respectively. Both functions offer a \texttt{stats} option. To be consistent we select \texttt{infit} and \texttt{outfit} statistics using \texttt{stats} = “infit.” The output can be displayed using the \texttt{print} function. For the person-fit statistics it is useful to include the arguments \texttt{patterns = TRUE}, and \texttt{item_order = “by_diff”} as arguments to the \texttt{print} function. In my code here I also remove individuals who have identical response patterns to make the output easier to read. I do this using \texttt{dplyr}.

\[
pfa \leftarrow \texttt{irt_person_fit(mod1, stats = \text{"infit")}
\]

\[
\text{persons } \leftarrow \texttt{print(pfa, patterns } = \text{T, item_order } = \text{"by_diff") } \%
\]

\[
> \% \text{dplyr::distinct(ability, std.err, outfit, pattern, \_keep_all } = \text{T})
\]

\[
ifa \leftarrow \texttt{irt_item_fit(mod1, stats } = \text{"infit")}
\]

\[
\text{print(ifa)}
\]
Several plots can be used to help diagnose person and item fit as well. These include person response functions:

\[
\text{plot}(\text{mod1, type = "np_prf", persons = persons$ids, pfa = pfa})
\]

Notice the plot type is "np_prf." This stands for nonparametric person response function. The persons argument is here used to present response functions for unique response patterns only. Additionally, the pfa argument is used to pass the previously estimated pfa object into the plot function so that it doesn’t have to be called by the function.

Item response functions can be called using the type = "trace" argument. Which items to plot can also be specified either by column-wise position in the data or by the item name using the items argument.
Other information can be added to these plots by adding tags to the plot type. For example, to plot the item residuals add “resid” to the plot type. To show the observed values on the item by the person abilities, add “obs” to the type. Punctuation and spaces can be added to make the plot type more readable.
De Ayala looks at plots of the item and test information. This can be done by specifying the type as "info" and "tinfo" respectively. To force all item information functions to be plotted on the same line, specify the facets argument as FALSE. The standard error of the estimate can be added to the test information plot by adding "SE" to the type.

\[ \text{plot}(\text{mod1, type = "obs trace", items = c(1,3,5), persons = persons$ids}) \]
All that has been run in this vignette applies to 1PL, 2PL, and 3PL models as well (though the infit and outfit statistics do not apply to non-Rasch models). These other models can be run using the following:

```r
mod2 <- wizirt(responses, item_type = "1PL")
mod3 <- wizirt(responses, item_type = "2PL")
```

They can then be compared using the anova() function:

```r
anova(mod1, mod2)
```
Vignette Based on Rupp (2013)

Rupp (2013) created a framework to assist researchers in identifying and handling person-misfit in item response theory (IRT). Rupp’s framework consists of five steps:

- statistical detection using local and global fit measures (either parametric or nonparametric),
- numerical tabulation, or summarization of the incidence of each type of aberrant response pattern,
- graphical exploration such as person response functions (PRFs),
- quantitative explanation using additional modeling, and
- qualitative explanation.

The wizirt package is designed to facilitate the wider spread use of this framework in Psychometric research. With this end in mind, this vignette walks through the use of wizirt for PFA as described by Rupp (2013).

First, users need to load wizirt.

```r
library(wizirt)
```

The data used in this example are not publicly available, but a practice data set is available in wizirt.

After loading wizirt, users can run a model with the wizirt function:

```r
wizirt_fit <- wizirt::wizirt(data = responses, item_type = "Rasch")
```
Here I specify my data and I designate the model as a Rasch model. The first step of Rupp’s framework asks for global and local person-fit statistics. At this time, wizirt does not calculate local person fit, but global-fit statistics can be calculated using the `irt_person_fit()` function.

```
pfa <- irt_person_fit(wizirt_fit, stats = c("Ht"))
```

These fit statistics, as well as the person abilities and the response patterns (step two) can be displayed using the `print()` function.

```
print(pfa, patterns = T, item_order = "by_diff") %>%
dplyr::mutate(dplyr::across(c(ability, std_err, Ht, Ht_cut), .fns = round, 2))
```

By specifying the item order as “by_diff” the responses patterns are now ordered by the difficulty of the item.

Rupp’s step three asks for person fit to be explored graphically. The nonparametric person response function is used for this in wizirt.

```
plot(wizirt_fit, type = "np_prf")
```

Notice that the axes on this plot are compressed and difficult to read. This happens because of the default settings within the plotting package used (ggplot2) and the dimensions of the plot in this paper. In the software a user can expand the plot so that the axis are not so cramped.
Rupp's step four instructs researchers to try to explain reasons for person misfit. This is done with the `irt_model_pfa()` function, which allows for the inclusion of predictors.

```r
mod <- irt_model_pfa(wizirt_fit, pfa, bins = 10, predictors = lapply(data[1:50, c("School", "center")], scale))
mod$sicc
summary(mod$models$Ht)
```

These first four steps should be used in conjunction with a qualitative explanation step, something that cannot be done in wizirt.
APPENDIX D

Sinhary Replication

The Karabatsos (2003) study was replicated on the same small scale I intended to replicate to demonstrate the accuracy of wizirt (Sinhary, 2017). In his replication of Karabatsos, Sinharay used only the $H^T$, $U3$, $Lz^*$, and ECI4 statistics. He found them to be comparable in his simulation study, contrary to what Karabatsos found. Sinharay then applied the statistics to real data with several examinees who are strongly believed to have cheated on the assessment. Here again the four statistics performed similarly.

I had originally intended to replicate the simulation study of Karabatsos but found Sinharay’s (2017) modified replication of Karabatsos’s study with real data with known cheaters to be more desirable for demonstrating the application of wizirt. I was particularly drawn to this sample because I was able to get access to the same data set and to Sinharay’s code for the study. This meant I could compare my results to his and know the cause of every deviation, no matter how small. This was a valuable tool as I explored wizirt for inaccuracies.

Methods

Data

The data used are the same data that were used by Sinharay (2017). They come from a single year of testing for a computer-based credentialing program. The administered exam contained 170 dichotomously scored items with 10 additional items that represented a pretest. The exam was administered to 1,644 individuals.

The data were selected because they are known to contain the responses of individuals that are believed to have cheated. These individuals were identified through a rigorous multi-step process using multiple sources of information.
**Software**

Data were analyzed using the package wizirt (0.1.1) in the open source software R (4.0.2). The wizirt package is built on a number of other packages, but the relevant packages running behind the scenes were ltm (Rizopoulos, 2006), mirt (Chalmers, 2012), and PerFit (Tendeiro et al., 2016). The analysis I was replicating likewise originally used FORTRAN, but Sinharay also had a script conducting the same study using R. In this script, Sinharay used ltm, PerFit, and another package called irtos (Partchev & Maris, 2017).

This replication was done to compare the results from the wizirt package, which is in development, to the results of Sinharay to evaluate the accuracy of the wizirt package.

**Analysis**

The analysis of Sinharay was simple. First, he calculated the $H^T$, $l_2^*$, $U_3$, ECI4, and an optimal fit measure. Then, he compared the proportions of individuals who were flagged by each statistic based on whether they had actually cheated (or are identified in the data as being strongly suspected of cheating). This same methodology was replicated here with the ltm package as the engine and the mirt package as the engine. The code combining Sinharay’s analysis and my own are found at https://github.com/Pflegermeister/wizirt-accuracy-checks.

**Results**

The coefficients were nearly identical from Sinharay’s code to wizirt using either the mirt or ltm engine. The person ability estimates were not exactly the same, though the correlations were very high. While Sinharay (2017) estimated the coefficients with the ltm package, he used the irtos package to estimate person abilities after the fact. This is why the wizirt + ltm results were not identical.
Fit measures were very similar, though the cutoffs were generated using different methods. This showed in the output. It seemed that for U3 Sinharay’s cutoffs were ideal (Table D1). This may be considered in future versions of wizirt. For H\textsuperscript{T}, the cutoff procedure employed by wizirt (PerFit::cutoff) seemed to be better at separating between aberrant and non-aberrant examinees.

**Table D1**

*Comparison of wizirt and Sinharay (2017)*

<table>
<thead>
<tr>
<th>Source</th>
<th>Examinees</th>
<th>Iz\textsuperscript{*}</th>
<th>Ht</th>
<th>U3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sinharay (2017 FORTRAN)</td>
<td>All</td>
<td>0.09</td>
<td>0.05</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>Flagged</td>
<td>0.17</td>
<td>0.19</td>
<td>0.15</td>
</tr>
<tr>
<td>Sinharay (R Code)</td>
<td>All</td>
<td>0.09</td>
<td>0.05</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>Flagged</td>
<td>0.17</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>My Code</td>
<td>All</td>
<td>0.10</td>
<td>0.08</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>Flagged</td>
<td>0.17</td>
<td>0.27</td>
<td>0.15</td>
</tr>
</tbody>
</table>

It should be noted that the values generated by Sinharay’s code were slightly different from what was reported in his article. This was expected by Sinharay, who gave me forewarning and let me know that this is because he used FORTRAN to produce the results in his 2017 article. Overall though, the results were nearly identical between the two.

**Discussion**

All of this suggests that the wizirt package is accurately calculating its coefficients, abilities, and person-fit statistics.
APPENDIX REFERENCES


https://doi.org/10.18637/jss.v074.i05