# 1. Introduction

This paper presents a series of simulations conducted to evaluate methods to detect of item misfit in Rasch models. First, conditional item infit and outfit will be under scrutiny. Second, item infit will be compared to item-restscore (Kreiner, 2011; Mueller & Santiago, 2022). Third, a bootstrap method for item-restscore will be presented and tested.

The assessment of item fit under the Rasch model has for decades been conducted using various rule-of-thumb critical values. Müller (2020) showed how the range of critical values for conditional item infit varies with sample size. The expected average item conditional infit range was described by Müller as fairly well captured by Smith’s rule-of-thumb formula 1±2/√n (Smith et al., 1998). However, the average range does not apply for all items, since item location relative to sample location also affects model expected item fit. This means that some items within a set of items varying in location are likely to have item fit values outside Smith’s average value range while still fitting the Rasch model.

It is here proposed that by using parametric bootstrapping one can establish item fit critical cutoff values that are sample and item specific. This procedure uses the estimated item and person locations based on the available data and simulates new response data that fit the Rasch model, to determine the range of plausible item fit values for each item. The R package easyRasch (Johansson, 2024) includes a function to determine item infit and outfit cutoff values using this method and will be tested in the simulations in this paper.

It is important to note that the conditional item fit described by Müller (2020) and implemented in the iarm R package (Mueller & Santiago, 2022) should not be confused with the unconditional item fit implemented in software such as Winsteps and RUMM2030, as well as all R packages except iarm. Unconditional item fit can result in unreliable item fit in sample sizes as small as 250 with increasing likelihood of problems as sample size increases. Readers are strongly recommended to read Müller’s paper to fully understand the issues with unconditional item fit.

# 2. Methods

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

A fully reproducible manuscript with R code and data is available on GitHub: <https://github.com/pgmj/rasch_itemfit>

The simulation of response data used three steps: First, a vector of theta values (person scores on the latent variable’s logit scale) were generated using rnorm(mean = 0, sd = 1.5). Second, a set of item locations ranging from -2 to 2 logits were generated for dichotomous items, using runif(n = 20, min = -2, max = 2). Third, the theta values were used to simulate item responses for participants, using sim.xdim() from the eRm package (Mair & Hatzinger, 2007), which allows simulation of multidimensional response data. Multiple datasets with 10 000 respondents each were generated using the same item and person parameters, varying the targeting of the misfitting item(s) and number of the misfitting item(s). More details are described under the separate studies. The parametric bootstrapping procedure was implemented using random samples from the simulated datasets. Sample size variations tested are described under each study.

The general procedure for the parametric bootstrapping is as follows:

1. Estimation of item locations based on simulated item response data, using conditional maximum likelihood (CML, Mair & Hatzinger, 2007).
2. Estimation of sample theta values using weighted maximum likelihood (Warm, 1989).
3. Simulation of new response data which fit the Rasch model, using the estimated item locations and theta values.
4. Estimation of the dichotomous Rasch model for the new response data using CML.
5. Based on step 4, calculation of conditional item infit and outfit (Mueller & Santiago, 2022; Müller, 2020) and/or item-restscore metrics (Kreiner, 2011; Mueller & Santiago, 2022).

Steps three and four were iterated over, using resampling with replacement from the estimated theta values as a basis for simulating the response data in step three.

Summary statistics were created with focus on the percentage of correct detection of misfit and false positives.

A complete list of software used for the analyses is listed in #sec-addmat.

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

# 3. Study 1: Item infit and outfit

Item mean square standardized residuals are either unweighted, which is referred to as “outfit”, or information weighted, which we call “infit” (Ostini & Nering, 2006, pp. 86–87). For details on conditional item fit we refer to the previously mentioned paper by Müller (2020). Conditional item infit and outfit are expected to be near 1, with higher values indicating an item to be underfitting the Rasch model (often due to multidimensionality issues) and lower values indicating overfit.

The function RIgetfit() from the easyRasch R package is tested here. It’s source code can be accessed on GitHub, see #sec-addmat. The function offers the user a choice of the number of bootstrap iterations to use to determine the critical cutoff values for each item’s infit and outfit. Our main interest in this study is two-fold. We want to test variations in the number of iterations used in RIgetfit() and evaluate how well the critical values based on the parametric bootstrapping procedure detects misfitting items. Additionally, a comparison between infit and outfit statistics in terms of detection rate and false positive rate will be conducted.

20 dichotomous items are used, with one item misfitting. Item locations are the same throughout all studies unless otherwise noted. The location of the misfitting item relative the to the sample theta mean was selected to be approximately 0, -1, and -2 logits. Three separate datasets were generated with these variations, each with 10 000 simulated respondents. One dataset with all three misfitting items was also generated, using the same sample size.

Then the RIitemfit() function is used to summarize the bootstrap results and also calculates the infit and outfit for each item in the observed data and highlights items with infit/outfit values outside of the cutoff values. RIitemfit() has a default (user modifiable) setting to slightly truncate the distribution of values using stats::quantile() at 0.001 and 0.999 to remove extreme values. An example is demonstrated in [Table 1](#tbl-itemfit1), using a subset of the items used in the simulations. [Figure 1](#fig-itemfit1) provides a visualization of the distribution of bootstrapped infit and outfit values, together with the infit/outfit values from the observed data illustrated using an orange diamond shape. Note the variation between items in plausible values of infit and outfit based on the bootstrap, and that Smith’s rule-of-thumb regarding infit (1±2/√n) would be 0.9-1.1 for a sample size of 400.

This study was rather computationally demanding since each simulation run entailed 100-400 underlying bootstrap iterations. The sample sizes used were 150, 250, 500, and 1000. The number of iterations to determine cutoff values were 100, 200, and 400. Sample size and iteration conditions were fully crossed with each other and the three different targeting variations of the one misfitting item, resulting in 4*3*3 = 36 conditions. Each combination used 200 simulation runs. The simulations took about 12 hours to run on a Macbook Pro Max M1 using 9 CPU cores.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1: Conditional item fit with simulation based cutoff values   | Item | InfitMSQ | Infit thresholds | OutfitMSQ | Outfit thresholds | Infit diff | Outfit diff | Location | | --- | --- | --- | --- | --- | --- | --- | --- | | V1 | 1.017 | [0.828, 1.123] | 1.061 | [0.57, 1.507] | no misfit | no misfit | -1.37 | | V11 | 1.000 | [0.793, 1.188] | 1.032 | [0.752, 1.315] | no misfit | no misfit | -0.66 | | V3 | 1.022 | [0.908, 1.114] | 1.050 | [0.63, 1.641] | no misfit | no misfit | 0.46 | | V12 | 0.966 | [0.809, 1.151] | 0.793 | [0.739, 1.206] | no misfit | no misfit | 1.58 | |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 1: Distribution of simulation based item fit and estimated item fit from observed data |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

## 3.1 Results

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Figures show the percent of simulation runs that have identified an item as misfitting. Items with more than 5% are colored in light red. A number representing the detection rate is shown adjacent to the bar representing the misfitting item. The figure grid columns are labelled with the number of iterations used by RIgetfit() to determine cutoff values, and grid rows are labelled with the sample size.

### 3.1.1 Infit

|  |
| --- |
| Figure 2: Conditional infit detection rate (misfit item at 0 logits) |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

[Figure 2](#fig-ifb0) shows the detection rate when the misfitting item is located at the sample mean. Detection rate is highest for the condition with 100 iterations with sample size 100 and 250, but it also shows higher levels of false positives when sample size increases to 500 or more.

|  |
| --- |
| Figure 3: Conditional infit detection rate (misfit item at -1 logits) |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

When the misfitting item is offset in targeting by -1 logits compared to the sample mean (see [Figure 3](#fig-ifb1)), the smallest sample size has less power to detect misfit compared to the on-target misfitting item. There are lower rates of false positives across all sample sizes and iterations.

|  |
| --- |
| Figure 4: Conditional infit detection rate (misfit item at -2 logits) |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Finally, when the misfitting item is located at -2 logits compared to the sample mean (see [Figure 4](#fig-ifb2)), we see a stronger reduction in power for sample sizes 150 and 250. No false positives are identified.

### 3.1.2 Outfit

|  |
| --- |
| Figure 5: Conditional outfit detection rate (misfit item at 0 logits) |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 6: Conditional outfit detection rate (misfit item at -1 logits) |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 7: Conditional outfit detection rate (misfit item at -2 logits) |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

As shown in [Figure 5](#fig-ifb0out), [Figure 6](#fig-ifb1out), and [Figure 7](#fig-ifb2out), outfit is performing worse than infit across the board.

### 3.1.3 Comments

Based on these simulation, it is highly recommended to use infit over outfit in assessing item fit. The performance of outfit calls to question whether it is useful at all for detecting item misfit.

Regarding infit and the use of parametric bootstrapping with the function RIgetfit(), it looks like 100 iterations are to recommend to determine cutoff values when the sample size is 250 or lower, while 200 or 400 iterations reduce the risk for false positives at sample sizes of 500 or larger. False positives are found at sample sizes 500 and 1000 only. The risk for false positives is notably higher when the misfitting item is located at the sample mean compared to when the misfitting item is off-target by -1 logits or more.

# 4. Study 2: Item-restscore

Item-restscore is a metric that compares an expected correlation with the observed correlation, using Goodman and Kruskal’s (Goodman & Kruskal, 1954; Kreiner, 2011). Lower observed values than expected indicates than an item is underfit to the Rasch model, while higher values indicate overfit. The item-restscore function used in this simulation is from the iarm package (Mueller & Santiago, 2022) and outputs Benjamini-Hochberg corrected *p*-values (Benjamini & Hochberg, 1995), which are used to determine whether the differences between the observed and expected values are statistically significant (using *p* < .05 as critical value) for each item.

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

## 4.1 Results

|  |
| --- |
| Figure 8: Item-restscore detection rate across targeting and sample size |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

This simulation includes an additional condition with 100 respondents, which results in significantly lower detection rates compared to n = 150. Compared to infit at 250 respondents, item-restscore has detection rates of 95.2%, 90.9%, and 62.4% for targeting 0, -1, and -2, while infit has 96.5%, 96.5%, and 71%. For sample size 500 and 1000, detection rate is similar, including the increased tendency for false positives at n = 1000. The false positive rate is lower for item-restscore than infit for sample sizes below 1000.

# 5. Study 3: Comparing infit and item-restscore

We will now compare the performance of infit and item-restscore when all three items are misfitting at the same time. This simulation will also include a condition with 2000 respondents, to examine if the false positive rate increases with more respondents. For infit, we will only use 200 iterations with RIgetfit() since that condition seemed to strike a balance between detection rate and false positives. Outfit is also included to see if it performs as bad as with one misfitting item.

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

### 5.0.1 Results

|  |
| --- |
| Figure 9: Conditional outfit detection rate with three misfitting items |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 10: Conditional infit detection rate with three misfitting items |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Looking at the performance of infit with three misfitting items ([Figure 10](#fig-ifb3)), we can see that the detection rate is markedly worse for item 13 (targeting -2 logits) in sample sizes 500 and below, compared to when single items were misfitting. The false positive rate has increased for sample size of 1000 and we can see it increase strongly at n = 2000. Outfit ([Figure 9](#fig-ifb3out)) again performs worse than infit.

|  |
| --- |
| Figure 11 |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 12: Detection rate for item-restscore compared to infit |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Item-restscore (see [Figure 11](#fig-itemrestscore2)) shows comparable detection rate to infit and higher levels of false positives. A comparison is made between the two in [Figure 12](#fig-comp1), where item-restscore is performing better than infit at detecting the -2 logits off-target item at n = 250, and better across all items for n = 500 and n = 1000. Infit performs better for samples n = 150 and n = 250 (except the item with location -2 logits).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 2: Item-restscore summary results across all sample sizes   | Item | Type of misfit | Percent | | --- | --- | --- | | 9 | underfit | 85.28 | | 18 | underfit | 78.96 | | 13 | underfit | 63.80 | | 2 | overfit | 20.80 | | 6 | overfit | 19.00 | | 20 | overfit | 15.48 | | 8 | overfit | 15.28 | | 10 | overfit | 14.60 | | 11 | overfit | 13.04 | | 7 | overfit | 12.52 | | 15 | overfit | 12.52 | | 1 | overfit | 11.96 | | 5 | overfit | 11.92 | | 16 | overfit | 11.12 | | 3 | overfit | 8.96 | | 14 | overfit | 7.28 | | 4 | overfit | 7.24 | | 12 | overfit | 5.96 | | 17 | overfit | 5.16 | | 19 | overfit | 1.24 | | 12 | underfit | 0.08 | |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Reviewing the type of misfit identified by item-restscore (see [Table 2](#tbl-overunder)), the false positives are all overfitting the Rasch model, except for two instances (out of 2500) indicating underfit for item 12. Items 9, 13, and 18, that were simulated to be misfitting due to loading on a separate dimension, are as expected showing underfit to the Rasch model.

# 6. Study 4: Bootstrapped item-restscore

For our final set of simulations, we will use a non-parametric bootstrap procedure with item-restscore. The difference from the parametric bootstrap is that the non-parametric bootstrap samples with replacement directly from the observed response data. First, based on the above problematic sample size of 2000 when three items are misfitting, we will use the bootstrap function to sample with replacement using n = 800 and 250 bootstrap samples. The function RIbootRestscore() from the easyRasch package will be used.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 3: Example output from RIbootRestscore()   | Item | Item-restscore result | Percent of iterations | | --- | --- | --- | | V18 | underfit | 100.0 | | V9 | underfit | 100.0 | | V13 | underfit | 98.8 | | V14 | overfit | 45.2 | | V20 | overfit | 41.2 | | V11 | overfit | 34.0 | | V1 | overfit | 32.0 | | V6 | overfit | 25.6 | | V5 | overfit | 24.8 | | V3 | overfit | 19.6 | | V15 | overfit | 17.6 | | V2 | overfit | 14.8 | | V12 | overfit | 10.8 | | V7 | overfit | 10.4 | | V8 | overfit | 9.2 | | V16 | overfit | 8.0 | | V17 | overfit | 3.6 | | V10 | overfit | 2.8 | | V19 | overfit | 2.4 | | V4 | underfit | 0.8 | | V10 | underfit | 0.4 | |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

RIbootRestscore() is demonstrated using a single sample in [Table 3](#tbl-bootir), where the table is sorted on Percent of iterations. The runtime was around 10-12 seconds using 8 CPU cores on a Macbook Pro M1 Max. In our simulation, we will repeat this procedure 500 times and report the average and standard deviation for the percent indicating misfit for each item.

Second, we will also apply the bootstrapped item-restscore method to sample sizes 150 and 250, using the complete sample for the same bootstrap procedure to see if this produces more useful information than previously tested strategies for identifying misfitting items.

## 6.1 Results

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 13: Item-restscore bootstrap results |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | (a) Misfitting items   | Item | Median | MAD | Mean | SD | Percentile .05 | | --- | --- | --- | --- | --- | --- | | V9 | 100.0 | 0.0 | 100.0 | 0.1 | 100.0 | | V18 | 100.0 | 0.0 | 99.8 | 0.9 | 99.2 | | V13 | 95.6 | 4.7 | 93.0 | 8.0 | 76.4 | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | (b) False positives   | Item | Median | MAD | Mean | SD | Percentile .95 | | --- | --- | --- | --- | --- | --- | | V2 | 20.0 | 15.4 | 24.3 | 16.7 | 58.8 | | V6 | 18.4 | 14.2 | 21.3 | 14.9 | 52.8 | | V8 | 14.4 | 12.5 | 19.7 | 15.6 | 51.6 | | V11 | 14.0 | 11.9 | 17.6 | 13.9 | 47.7 | | V15 | 13.2 | 10.1 | 16.6 | 13.4 | 45.2 | | V20 | 16.4 | 13.6 | 19.5 | 13.9 | 45.2 | | V5 | 12.0 | 10.1 | 15.7 | 12.6 | 44.4 | | V10 | 13.2 | 11.3 | 16.4 | 13.0 | 42.4 | | V16 | 11.4 | 10.4 | 15.2 | 13.0 | 42.0 | | V7 | 13.2 | 11.9 | 16.7 | 12.6 | 41.6 | | V1 | 12.0 | 10.1 | 15.5 | 12.8 | 41.3 | | V3 | 10.8 | 9.5 | 13.6 | 11.2 | 36.1 | | V14 | 7.2 | 6.5 | 10.4 | 10.1 | 30.7 | | V12 | 7.6 | 7.7 | 10.4 | 9.5 | 29.7 | | V4 | 7.2 | 6.5 | 10.2 | 9.5 | 29.3 | | V17 | 6.4 | 6.5 | 8.4 | 8.0 | 24.4 | | V19 | 2.4 | 2.4 | 3.9 | 4.6 | 13.6 | | |

Table 4: Summary statistics for item-restscore bootstrap simulation

[Figure 13](#fig-irb0all) shows that there is variation in false positive rate and it is nearly always indicating overfit, while the misfitting items are only indicated as underfit. The summary statistics in [Table 4](#tbl-irb0mis) show that there can be quite a bit of variation for false positives, but the clear majority of results are below 50%. 3 items have 95th percentile values above 50, with the highest at 58.8.

## 6.2 Small sample (n = 150)

We will use 200 simulations to check the performance of the bootstrapped item-restscore function for sample size 150. As an additional experimental condition, we will use both 250 and 500 bootstraps for item-restscore in each simulation.

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 14 |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Item-restscore bootstrapping improves slightly on the single instance of item-restscore for the n = 150 condition (see [Figure 14](#fig-irboot150)). When comparing to the previous results in [Figure 11](#fig-itemrestscore2), where the detection rate for the same sample size were at 49.2%, 14.6%, and 34.6% (for items 9, 13, and 18 respectively), the corresponding median values from the bootstrapped item-restscore with 250 iterations were 52.4%, 19.2%, and 38.4%. Using 500 bootstrap iterations did not result in relevant improvements over 250 iterations (see [Table 5](#tbl-irb150mis)).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 5: Summary statistics for item-restscore bootstrap simulation (n = 150)   | Bootstrap iterations | Item | Median | MAD | Mean | SD | | --- | --- | --- | --- | --- | --- | |  | V9 | 52.4 | 38.0 | 51.4 | 29.2 | | 250 | V18 | 38.4 | 35.6 | 42.0 | 27.8 | |  | V13 | 19.2 | 21.9 | 27.3 | 25.0 | |  | V9 | 54.3 | 37.8 | 51.0 | 29.5 | | 500 | V18 | 37.2 | 35.9 | 41.4 | 27.9 | |  | V13 | 19.4 | 23.1 | 27.2 | 24.7 | |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

# 7. Study 5: Varying number of items

When doing simulation studies there is always a balance to strike between trying to evaluate many scenarios and not having too high complexity. We have been keeping several things constant, such as item locations and number of items, which makes interpretation easier but may limit the applicability of the results. For our final simulation, we will vary the number of items and the number of misfitting items. First, 40 dichotomous items will be used, adding 20 new item locations to the previously used set, with the same three items misfitting (items 9, 13, and 18). Second, items 1-10 out of the initial 20 items will be used, which means only item 9 will be misfit. We’ll again be using sample sizes of 150, 250, 500, and 1000.

Item-restscore and item infit will be compared. The latter will use 100 bootstrap iterations to determine critical values for sample sizes 150 and 250, and 200 bootstrap iterations for n >= 500.

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

## 7.1 Results 40 items

|  |
| --- |
| Figure 15 |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 16 |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Infit performs better when sample size is 150 or 250 (see [Figure 15](#fig-ifb40)), while performance is slightly better for item-restscore for n >= 500 in terms of lower rates of false positives (see [Figure 16](#fig-itemrestscore40)).

## 7.2 Results 10 items

|  |
| --- |
| Figure 17 |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

|  |
| --- |
| Figure 18 |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Compared to when we had 20 items, … create a summary table! And run more simulations tonight.

# 8. Discussion

## 8.1 Limitations

Number of items could be varied more. However, the results from Müller (2020), which use 10, 15, and 20 items, indicates small differences in critical value ranges. But this might not have implications for detection rate of misfitting items (we need the 40 items simulation and maybe 10 also?). Partial credit model for polytomous data would have been nice to also test. Although results regarding detection rate should generalize from RM to PCM, maybe the sample size in relation to number of items does not easily translate from the dichotomous case?

# 9. Conclusion

These findings make a good argument for removing one item at a time when the analysis indicates misfitting items, starting with the most underfitting item. This is especially relevant for *n* >= 500 and when misfitting items are located close to the sample mean.

|  |
| --- |
| Figure 19 |

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

Assessing item fit and dimensionality should be done using multiple methods. Item fit and item-restscore should be used in parallel, while also examining residual patterns by reviewing standardized factor loadings on the first residual contrast (see [Figure 19](#fig-loadloc) for an example) as well as Yen’s Q3 residual correlations.

What does this look like with a small sample?

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)

While the simulations in this paper have used dichotomous data, all functions evaluated in this paper also work with polytomous data using the Rasch Partial Credit Model.

# References

Benjamini, Y., & Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, *57*(1), 289–300. <https://doi.org/10.1111/j.2517-6161.1995.tb02031.x>

Goodman, L. A., & Kruskal, W. H. (1954). Measures of Association for Cross Classifications. *Journal of the American Statistical Association*, *49*(268), 732–764. <https://doi.org/10.2307/2281536>

Johansson, M. (2024). *easyRasch: Psychometric analysis in r with rasch measurement theory*. <https://github.com/pgmj/easyRasch>

Kreiner, S. (2011). A Note on Item–Restscore Association in Rasch Models. *Applied Psychological Measurement*, *35*(7), 557–561. <https://doi.org/10.1177/0146621611410227>

Mair, P., & Hatzinger, R. (2007). Extended Rasch Modeling: The eRm Package for the Application of IRT Models in R. *Journal of Statistical Software*, *20*(1), 1–20. <https://doi.org/10.18637/jss.v020.i09>

Mueller, M., & Santiago, P. H. R. (2022). *Iarm: Item Analysis in Rasch Models*. <https://cran.r-project.org/web/packages/iarm/index.html>

Müller, M. (2020). Item fit statistics for Rasch analysis: Can we trust them? *Journal of Statistical Distributions and Applications*, *7*(1), 5. <https://doi.org/10.1186/s40488-020-00108-7>

Ostini, R., & Nering, M. (2006). *Polytomous Item Response Theory Models*. SAGE Publications, Inc. <https://doi.org/10.4135/9781412985413>

Smith, R. M., Schumacker, R. E., & Bush, M. J. (1998). [Using item mean squares to evaluate fit to the Rasch model](https://www.ncbi.nlm.nih.gov/pubmed/9661732). *Journal of Outcome Measurement*, *2*(1), 66–78.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, *54*(3), 427–450. <https://doi.org/10.1007/BF02294627>

# 10. Additional materials

* GitHub link for easyRasch source code: <https://github.com/pgmj/easyRasch/>
  + Most functions are defined in this file: <https://github.com/pgmj/easyRasch/blob/main/R/easyRasch.R>

## 10.1 Session info

This documents the specific R packages and versions used in this study.

R version 4.4.2 (2024-10-31)  
Platform: aarch64-apple-darwin20  
Running under: macOS Sequoia 15.2  
  
Matrix products: default  
BLAS: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib   
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib; LAPACK version 3.12.0  
  
locale:  
[1] en\_US.UTF-8/en\_US.UTF-8/en\_US.UTF-8/C/en\_US.UTF-8/en\_US.UTF-8  
  
time zone: Europe/Stockholm  
tzcode source: internal  
  
attached base packages:  
 [1] parallel grid stats4 stats graphics grDevices utils   
 [8] datasets methods base   
  
other attached packages:  
 [1] showtext\_0.9-7 showtextdb\_3.0 sysfonts\_0.8.9 arrow\_16.1.0   
 [5] easyRasch\_0.3.3 doParallel\_1.0.17 iterators\_1.0.14 furrr\_0.3.1   
 [9] future\_1.34.0 foreach\_1.5.2 janitor\_2.2.0 hexbin\_1.28.4   
[13] catR\_3.17 glue\_1.8.0 ggrepel\_0.9.6 patchwork\_1.3.0   
[17] reshape\_0.8.9 matrixStats\_1.4.1 psychotree\_0.16-1 psychotools\_0.7-4  
[21] partykit\_1.2-22 mvtnorm\_1.3-1 libcoin\_1.0-10 psych\_2.4.6.26   
[25] mirt\_1.43 lattice\_0.22-6 kableExtra\_1.4.0 formattable\_0.2.1  
[29] lubridate\_1.9.3 forcats\_1.0.0 stringr\_1.5.1 dplyr\_1.1.4   
[33] purrr\_1.0.2 readr\_2.1.5 tidyr\_1.3.1 tibble\_3.2.1   
[37] tidyverse\_2.0.0 ggdist\_3.3.2 iarm\_0.4.3 ggplot2\_3.5.1   
[41] eRm\_1.0-6   
  
loaded via a namespace (and not attached):  
 [1] splines\_4.4.2 R.oo\_1.26.0 cellranger\_1.1.0   
 [4] rpart\_4.1.23 lifecycle\_1.0.4 rprojroot\_2.0.4   
 [7] globals\_0.16.3 vroom\_1.6.5 MASS\_7.3-61   
 [10] backports\_1.5.0 magrittr\_2.0.3 vcd\_1.4-12   
 [13] Hmisc\_5.2-0 rmarkdown\_2.28 yaml\_2.3.10   
 [16] sessioninfo\_1.2.2 pbapply\_1.7-2 RColorBrewer\_1.1-3   
 [19] audio\_0.1-11 quadprog\_1.5-8 R.utils\_2.12.3   
 [22] nnet\_7.3-19 listenv\_0.9.1 testthat\_3.2.1.1   
 [25] RPushbullet\_0.3.4 vegan\_2.6-8 parallelly\_1.38.0   
 [28] svglite\_2.1.3 permute\_0.9-7 codetools\_0.2-20   
 [31] xml2\_1.3.6 tidyselect\_1.2.1 farver\_2.1.2   
 [34] base64enc\_0.1-3 jsonlite\_1.8.9 progressr\_0.14.0   
 [37] Formula\_1.2-5 survival\_3.7-0 systemfonts\_1.1.0   
 [40] tools\_4.4.2 gnm\_1.1-5 snow\_0.4-4   
 [43] Rcpp\_1.0.13-1 mnormt\_2.1.1 gridExtra\_2.3   
 [46] xfun\_0.46 here\_1.0.1 mgcv\_1.9-1   
 [49] distributional\_0.4.0 ca\_0.71.1 withr\_3.0.2   
 [52] beepr\_2.0 fastmap\_1.2.0 fansi\_1.0.6   
 [55] digest\_0.6.37 timechange\_0.3.0 R6\_2.5.1   
 [58] colorspace\_2.1-1 R.methodsS3\_1.8.2 inum\_1.0-5   
 [61] utf8\_1.2.4 generics\_0.1.3 data.table\_1.16.0   
 [64] SimDesign\_2.17.1 htmlwidgets\_1.6.4 pkgconfig\_2.0.3   
 [67] gtable\_0.3.5 lmtest\_0.9-40 brio\_1.1.5   
 [70] htmltools\_0.5.8.1 scales\_1.3.0 snakecase\_0.11.1   
 [73] knitr\_1.48 rstudioapi\_0.17.1 tzdb\_0.4.0   
 [76] checkmate\_2.3.2 nlme\_3.1-166 curl\_6.0.1   
 [79] zoo\_1.8-12 relimp\_1.0-5 vcdExtra\_0.8-5   
 [82] foreign\_0.8-87 pillar\_1.9.0 vctrs\_0.6.5   
 [85] Deriv\_4.1.3 cluster\_2.1.6 dcurver\_0.9.2   
 [88] archive\_1.1.8 GPArotation\_2024.3-1 htmlTable\_2.4.3   
 [91] evaluate\_1.0.1 cli\_3.6.3 compiler\_4.4.2   
 [94] rlang\_1.1.4 crayon\_1.5.3 future.apply\_1.11.2   
 [97] labeling\_0.4.3 plyr\_1.8.9 stringi\_1.8.4   
[100] viridisLite\_0.4.2 assertthat\_0.2.1 munsell\_0.5.1   
[103] Matrix\_1.7-1 qvcalc\_1.0.3 hms\_1.1.3   
[106] bit64\_4.0.5 bit\_4.0.5 readxl\_1.4.3

Source: [Article Notebook](https://pgmj.github.io/rasch_itemfit/index.qmd.html)