Detecting item misfit in Rasch models

Magnus Johansson

2025-01-03

Psychometrics in general have long relied on rule-of-thumb cutoff values for various goodness of fit metrics. With more powerful personal computers it is both feasible and desirable to use simulation/bootstrap methods to determine appropriate cutoff values. This paper illustrates and evaluates the use of an R package for Rasch psychometrics that has implemented functions to simplify the process of identifying misfitting items using simulation based cutoff values. Through a series of simulation studies a comparison is made between information weighted conditional item fit and item-restscore using Goodman and Kruskal's gamma.

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 and Santiago 2022). Third, a bootstrap method for item-restscore will be presented and tested.

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, Schumacker, and Bush 1998). However, the average range does not apply for all items, since item location also affects model expected item fit which means that some items may have plausible item fit values outside Smith's average value range.

It is here proposed that by using parametric bootstrapping one can establish item fit critical cutoff values that are sample and item specific. The R package `easyRasch` (Johansson 2024) includes a function to determine item infit and outfit cutoff values, which 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 and 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 problems with unconditional item fit.

2 Methods

Source: Article Notebook

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) 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 and Hatzinger 2007), which allows simulation of multidimensional response data. Multiple datasets were generated using the same item and person parameters, varying the targeting and number of the misfitting item(s). More details are described under the separate studies below.

The parametric bootstrapping procedure was implemented using random samples from the 10 000 simulated responses. The sample size variations tested are described under each study.

1. Estimation of item locations based on simulated item responses using conditional maximum likelihood (CML, Mair and Hatzinger 2007).
2. Estimation of sample theta values using weighted maximum likelihood (Warm 1989)
3. Simulation of new response data, fitting 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 fit (Müller 2020; Mueller and Santiago 2022) and/or item-restscore metrics (Kreiner 2011; Mueller and 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.

Source: Article Notebook

Source: Article Notebook

Source: Article Notebook

3 Study 1

Item mean square standardized residuals are either unweighted, which is referred to as “outfit”, or information weighted, which we call “infit” (Ostini and Nering 2006, 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.

Main lines of inquiry:

1. How does the number of iterations used in RIgetfit() impact the indicated cutoff values?
2. How useful are the cutoff values in detecting misfitting items (and false positives), when using the optimal number of iterations?
3. Müller (2020) hints at outfit being less useful than infit. We will investigate this by comparing them.

20 dichotomous items are used, with one item misfitting. Item locations are the same throughout. 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.

The function `RIgetfit()` from the `easyRasch` R package is tested here. It’s source code can be accessed on GitHub. The function offers the user a choice of the number of bootstrap iterations to use to calculate the cutoff values for infit and outfit.

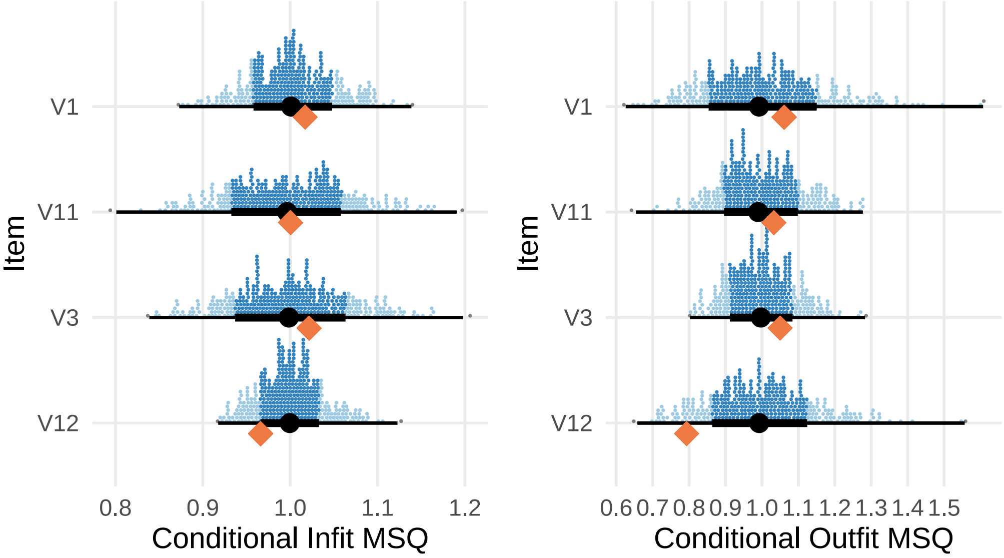
Then the `RIitemfit()` function is used to summarize the bootstrap results and also calculates the infit and outfit for each item in the observed data and highlights items with infit/outfit values outside of the cutoff values. `RIitemfit()` has a default (user modifiable) setting to slightly truncate the distribution of values using `stats::quantile()` at 0.001 and 0.999 to remove extreme values. An example is demonstrated in Table 1, using a subset of the items used in the simulations. Figure 1 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 simulations. 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

		Infit		Outfit		Outfit	
Item	InfitMSQ	thresholds	OutfitMSQ	thresholds	Infit diff	diff	Location
V1	1.017	[0.874, 1.138]	1.061	[0.631, 1.605]	no misfit	no misfit	-1.37
V11	1.000	[0.808, 1.184]	1.032	[0.666, 1.277]	no misfit	no misfit	-0.66
V3	1.022	[0.918, 1.119]	1.050	[0.668, 1.554]	no misfit	no misfit	0.46
V12	0.966	[0.841, 1.189]	0.793	[0.803, 1.28]	no misfit	0.01	1.58

Source: Article Notebook



Note: Results from 400 simulated datasets with 400 respondents. Orange diamond shaped dots indicate observed conditional item fit.

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

Source: Article Notebook

Source: Article Notebook

Source: Article Notebook

3.1 Results

Source: Article Notebook

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

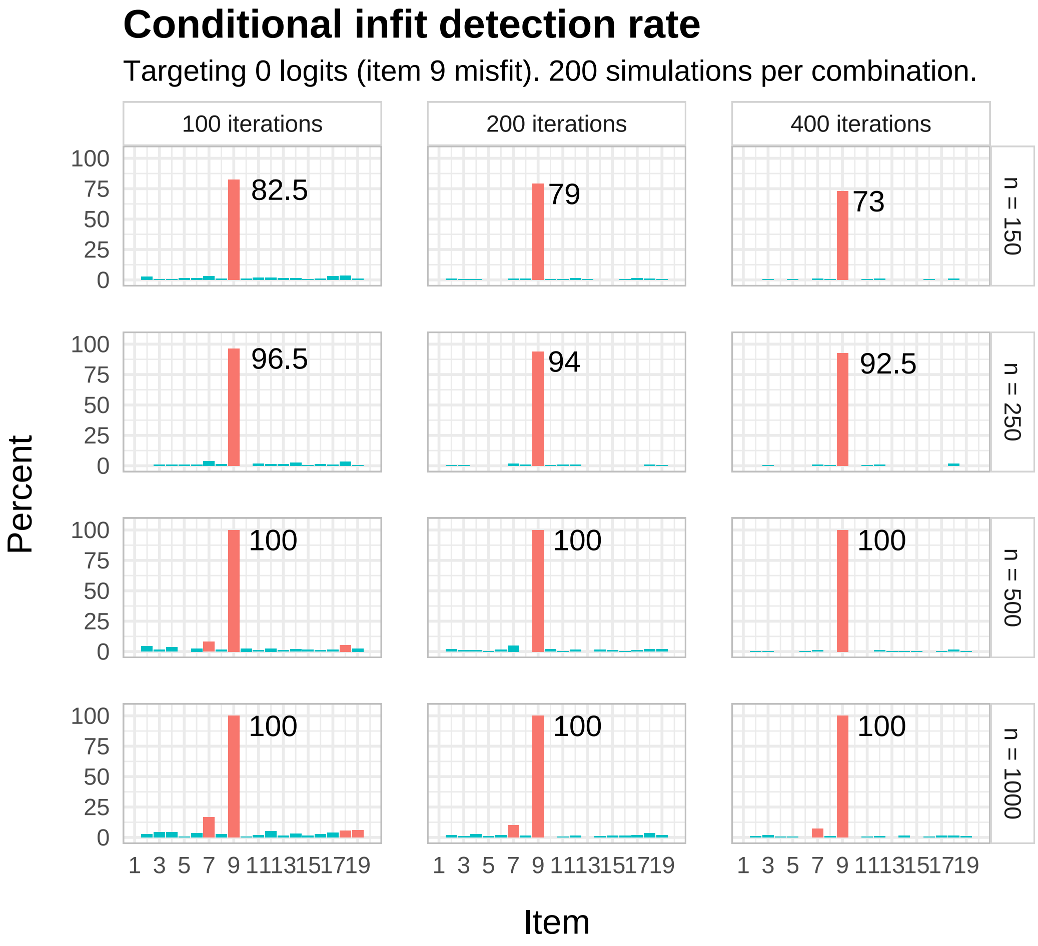


Figure 2

Source: Article Notebook

Figure 2 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.

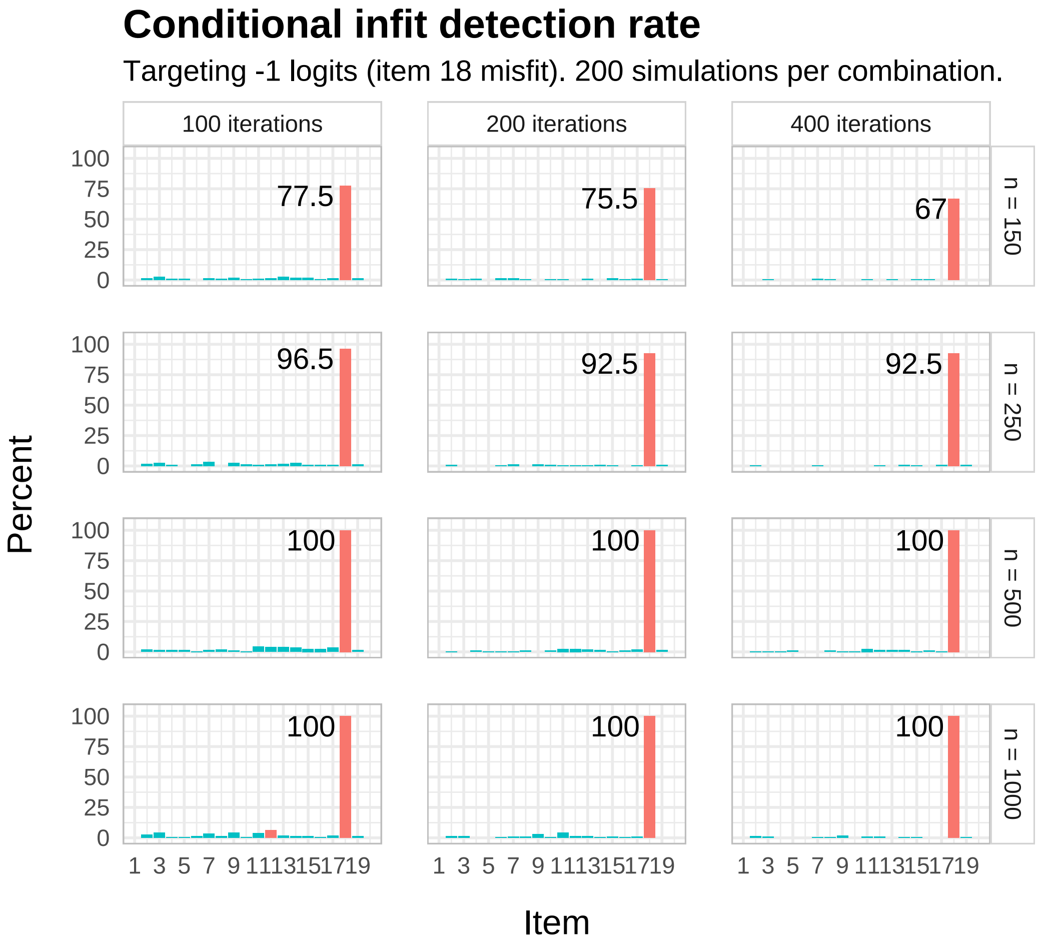


Figure 3

Source: Article Notebook

When the misfitting item is offset in targeting by -1 logits compared to the sample mean (see Figure 3), 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.

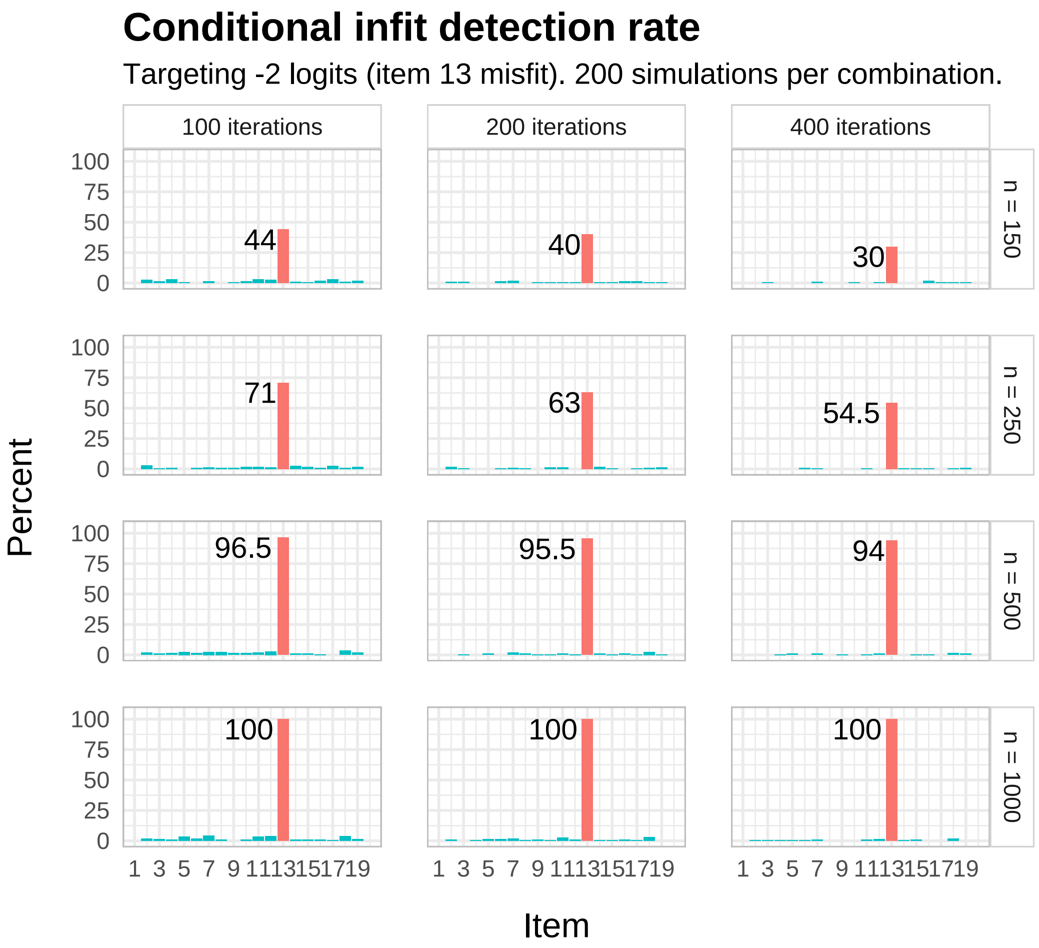


Figure 4

Source: Article Notebook

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

3.1.2 Outfit

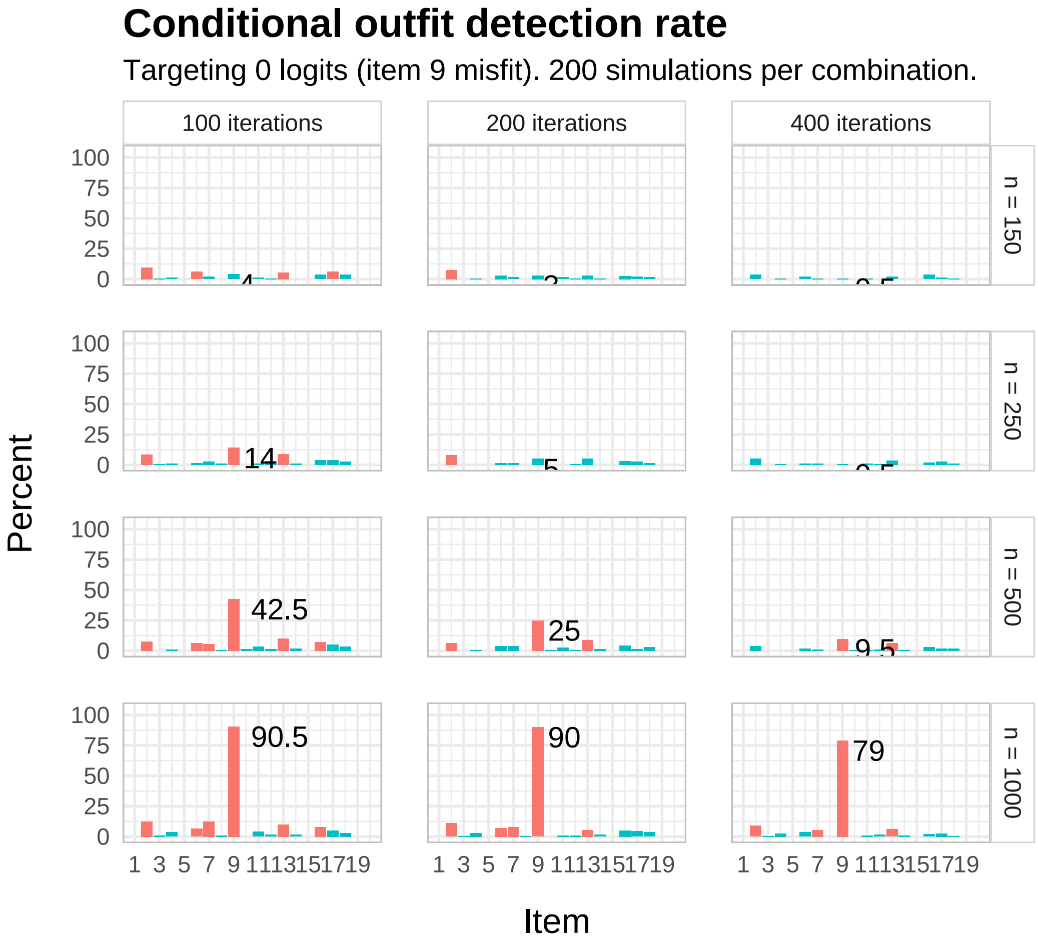


Figure 5

Source: Article Notebook

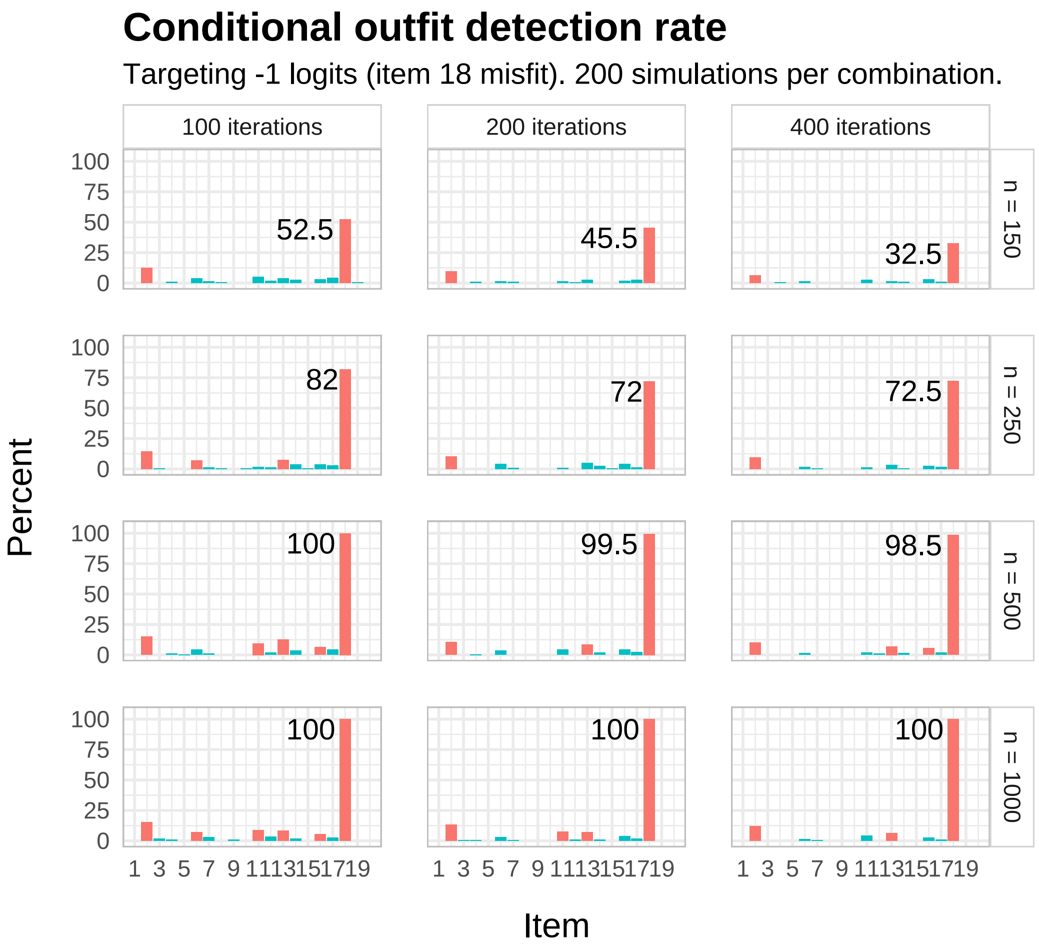


Figure 6

Source: Article Notebook

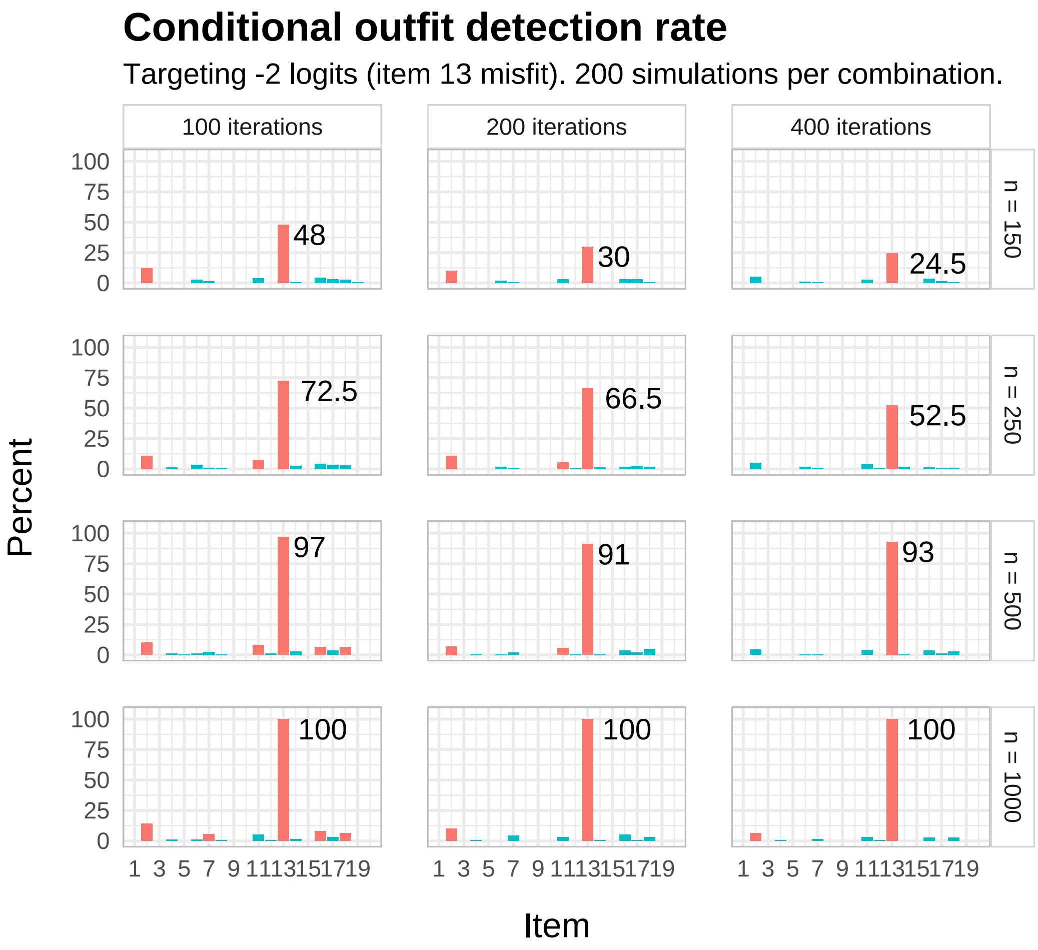


Figure 7

Source: Article Notebook

As shown in Figure 5, Figure 6, and Figure 7, outfit is performing much worse than infit across the board.

3.1.3 Comments

Based on these simulation, it seems reasonable to recommend the use of infit in determining item fit over outfit. The performance of outfit calls to question whether it is useful at all.

Regarding infit and the use of parametric bootstrapping with `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.

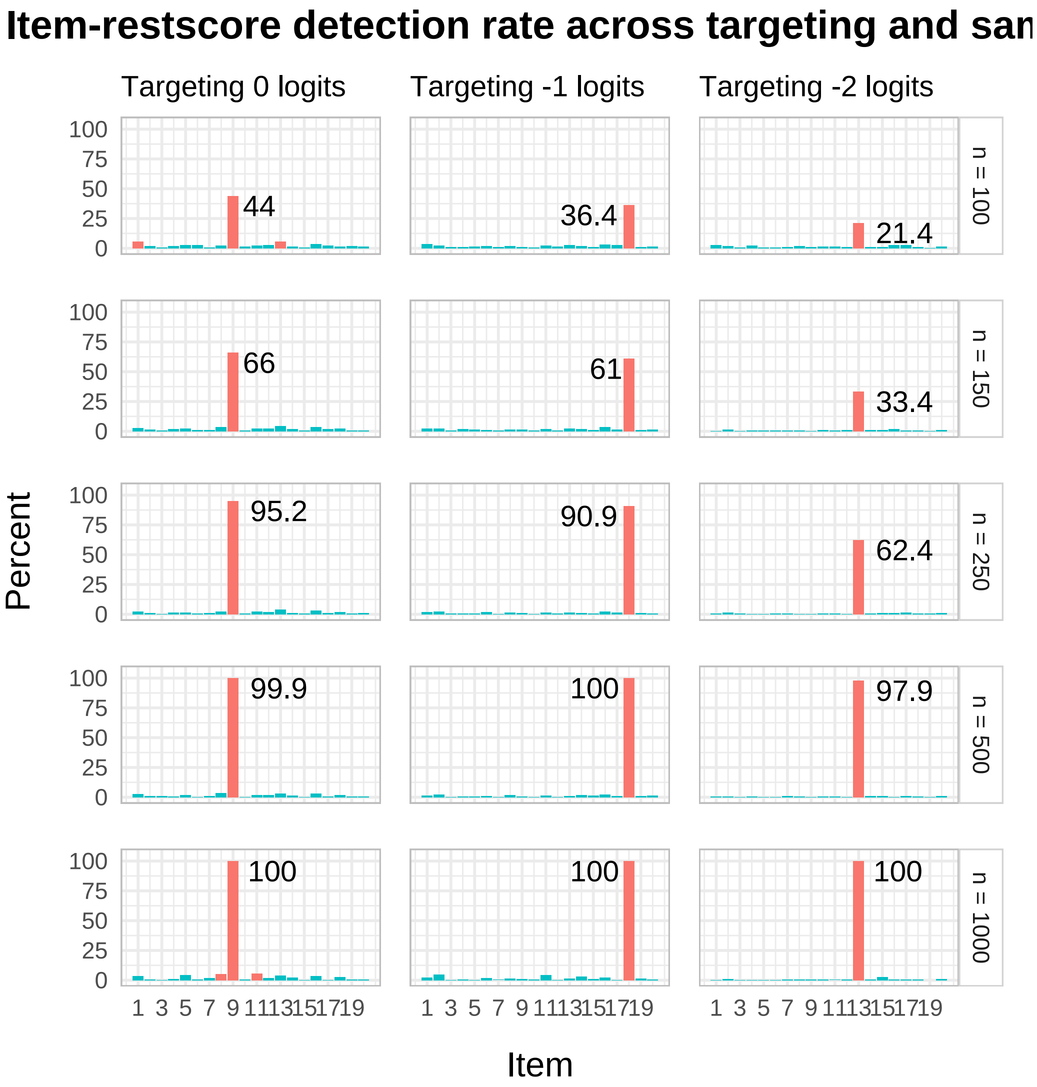
4 Study 2

Item-restscore is a metric that compares an expected correlation with the observed correlation. Lower 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 and Santiago 2022) and outputs Benjamini-Hochberg corrected *p*-values (Benjamini and Hochberg 1995), which are used in determining whether the observed value is statistically significant (using $p < .05$ as critical value) from the expected value for each item.

Source: Article Notebook

Source: Article Notebook

4.1 Results



Note: 1000 simulated datasets for each combination.

Figure 8

Source: Article Notebook

This simulation includes an additional condition with 100 respondents, which results in significantly lower detection rates than with 150 respondents. 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, performance is similar, including the increased tendency for false positives at n = 1000.

Similarly to infit, item-restscore has decreased detection rate for off-target misfitting items. The false positive rate is lower for item-restscore than infit for sample sizes below 1000.

5 Study 3

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.

Source: Article Notebook

Source: Article Notebook

Source: Article Notebook

5.0.1 Results

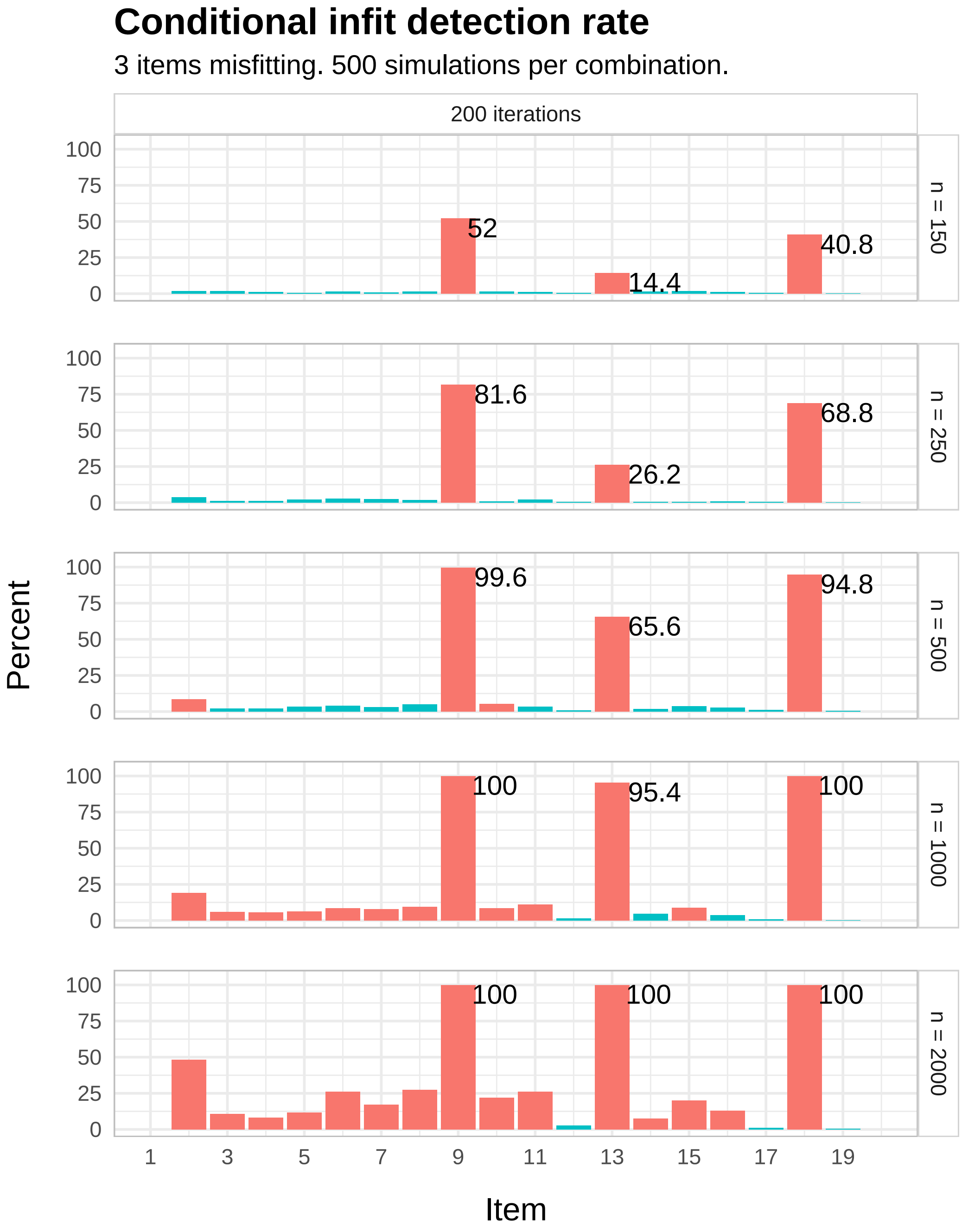


Figure 9

Source: Article Notebook

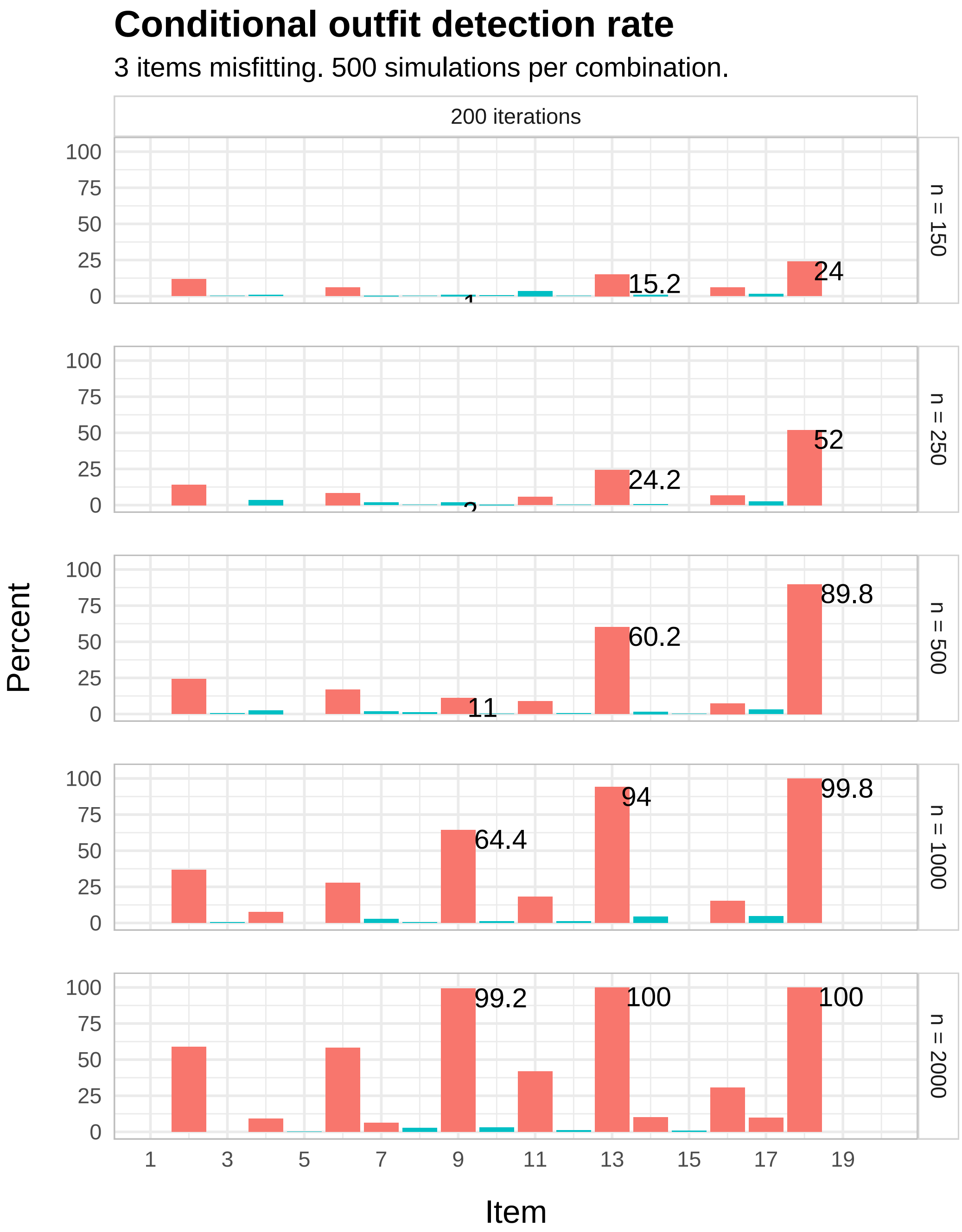


Figure 10

Source: Article Notebook

Looking at the performance of infit with three misfitting items (Figure 9), we can see that the detection rate is markedly worse for item 13 (targeting -1 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 escalate when n = 2000. Outfit (Figure 10) again performs worse than infit.

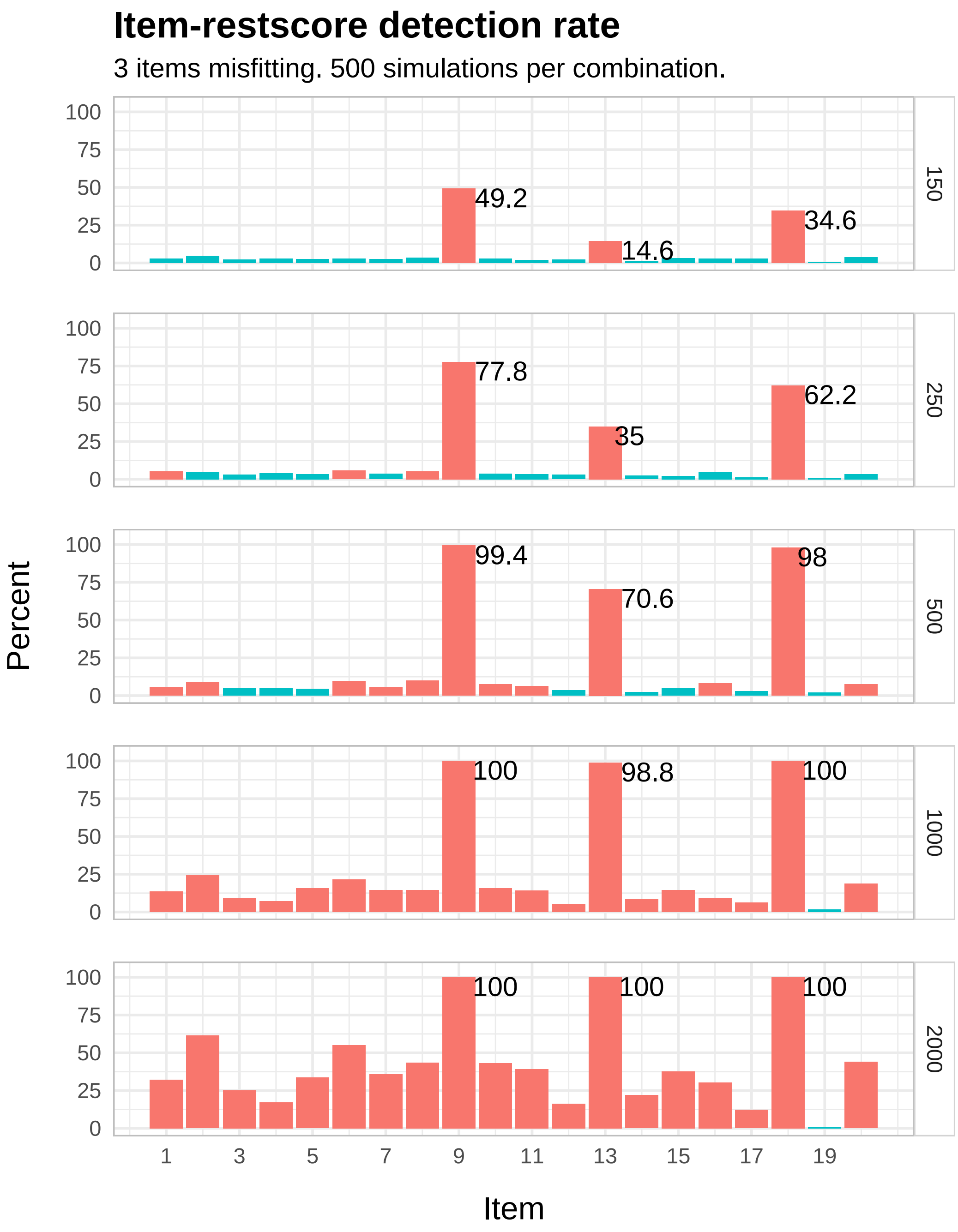


Figure 11

Source: Article Notebook

Table 2

Item	Type of misfit	n
1	overfit	299
2	overfit	520
3	overfit	224
4	overfit	181
5	overfit	298
6	overfit	475
7	overfit	313
8	overfit	382
9	underfit	2132
10	overfit	365
11	overfit	326
12	overfit	149
12	underfit	2
13	underfit	1595
14	overfit	182
15	overfit	313
16	overfit	278
17	overfit	129
18	underfit	1974
19	overfit	31
20	overfit	387

Source: Article Notebook

Item-restscore has higher detection rate than infit (see Figure 11), but also higher levels of false positives. Reviewing the type of misfit identified by item-restscore (see Table 2), the false positives are all overfitting the Rasch model, except for two instances 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

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	92.8
V2	overfit	73.6
V10	overfit	34.0
V6	overfit	31.2
V7	overfit	28.4
V8	overfit	22.4
V15	overfit	20.0
V5	overfit	19.6
V4	overfit	12.0
V16	overfit	11.6
V20	overfit	8.8
V11	overfit	8.0
V1	overfit	6.8
V17	overfit	4.8
V14	overfit	3.2
V19	overfit	1.2
V3	overfit	1.2
V12	overfit	0.4
V12	underfit	0.4
V14	underfit	0.4

Source: Article Notebook

`RIbootRestscore()` is demonstrated using a single sample in Table 3, 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

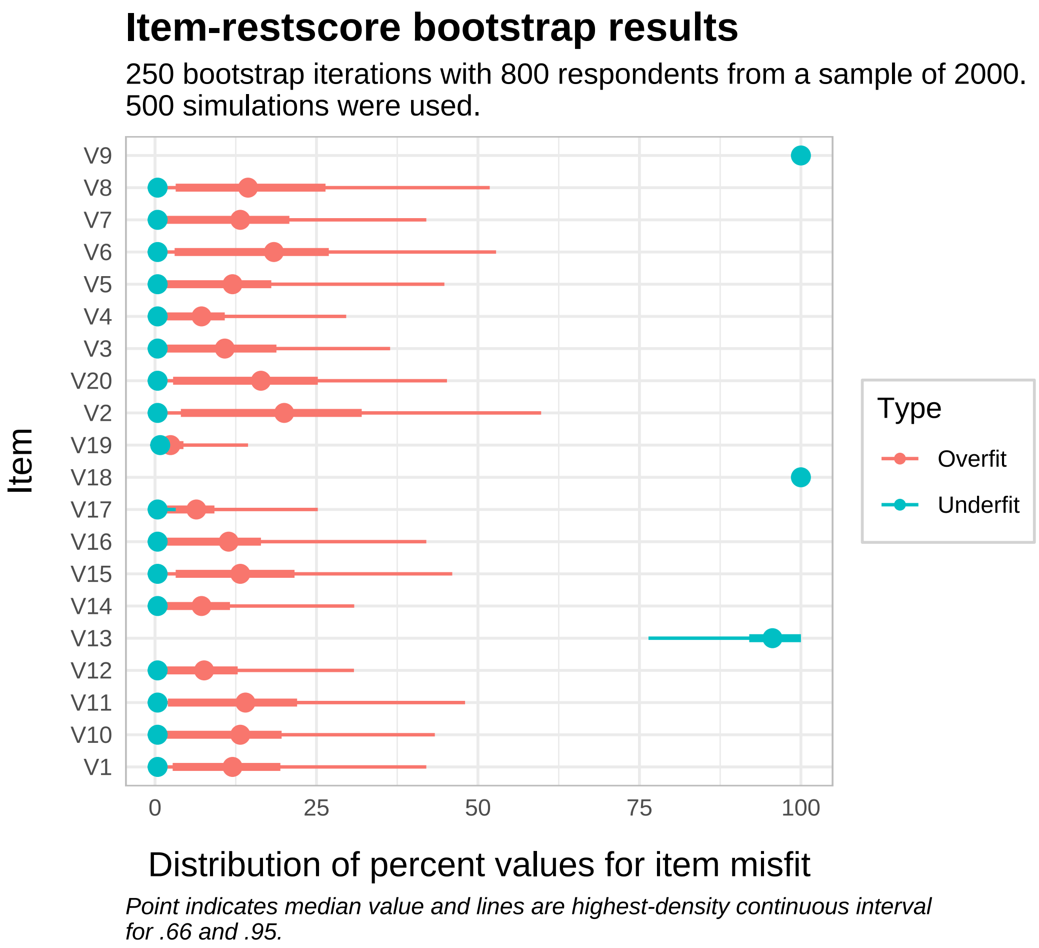


Figure 12

Source: Article Notebook

Figure 12 shows that there is variation in false positive rate, but it is nearly always indicating overfit, while the misfitting items are only indicated as underfit. The summary statistics in Table 4 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 Smaller samples

We will use 200 simulations for this part, applied to sample sizes 150 and 250. As an additional condition, we will use both 250 and 500 bootstraps for item-restscore in each simulation.

Table 4: Summary statistics for item-restscore bootstrap simulation

(a) Misfitting items

Item	Median	MAD	Mean	SD	p05	p01
V9	100.0	0.0	100.0	0.1	100.0	99.6
V18	100.0	0.0	99.8	0.9	99.2	98.0
V13	95.6	4.7	93.0	8.0	76.4	64.0

(b) False positives

Item	Median	MAD	Mean	SD	p95	p99
V2	20.0	15.4	24.3	16.7	58.8	70.0
V8	14.4	12.5	19.7	15.6	51.6	65.6
V6	18.4	14.2	21.3	14.9	52.8	64.0
V11	14.0	11.9	17.6	13.9	47.7	62.4
V15	13.2	10.1	16.6	13.4	45.2	61.7
V1	12.0	10.1	15.5	12.8	41.3	59.7
V10	13.2	11.3	16.4	13.0	42.4	58.5
V16	11.4	10.4	15.2	13.0	42.0	58.4
V20	16.4	13.6	19.5	13.9	45.2	57.2
V5	12.0	10.1	15.7	12.6	44.4	52.8
V7	13.2	11.9	16.7	12.6	41.6	51.7
V3	10.8	9.5	13.6	11.2	36.1	49.3
V14	7.2	6.5	10.4	10.1	30.7	47.0
V12	7.6	7.7	10.4	9.5	29.7	41.4
V4	7.2	6.5	10.2	9.5	29.3	40.1
V17	6.4	6.5	8.4	8.0	24.4	32.6
V19	2.4	2.4	3.9	4.6	13.6	19.5

Source: Article Notebook

6.3 Final simulation?

Run with 40 items and 3 misfitting, just 500 simulations with normal item-restscore to assess if this makes a difference. And 10 items with 1-2 misfit?

7 Discussion

7.1 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?

8 Conclusion

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

References

Benjamini, Yoav, and Yosef Hochberg. 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.

Johansson, Magnus. 2024. *easyRasch: Psychometric Analysis in r with Rasch Measurement Theory*. https://github.com/pgmj/easyRasch.

Kreiner, Svend. 2011. "A Note on Item–Restscore Association in Rasch Models." *Applied Psychological Measurement* 35 (7): 557–61. https://doi.org/10.1177/0146621611410227.

Mair, Patrick, and Reinhold Hatzinger. 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, Marianne, and Pedro Henrique Ribeiro Santiago. 2022. "Iarm: Item Analysis in Rasch Models." https://cran.r-project.org/web/packages/iarm/index.html.

Müller, Marianne. 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, Remo, and Michael Nering. 2006. *Polytomous Item Response Theory Models*. SAGE Publications, Inc. https://doi.org/10.4135/9781412985413.

Smith, R. M., R. E. Schumacker, and M. J. Bush. 1998. "Using Item Mean Squares to Evaluate Fit to the Rasch Model." *Journal of Outcome Measurement* 2 (1): 66–78.

Warm, Thomas A. 1989. "Weighted Likelihood Estimation of Ability in Item Response Theory." *Psychometrika* 54 (3): 427–50. https://doi.org/10.1007/BF02294627.