Detecting item misfit in Rasch models

Magnus Johansson

2025-01-06

Psychometrics in general have long relied on rule-of-thumb critical values for various goodness of fit metrics. With more powerful personal computers it is both feasible and desirable to use simulation methods to determine appropriate critical 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 determining simulation based cutoff values. Through a series of simulation studies a comparison is made between information weighted conditional item fit ("infit") and item-restscore correlations using Goodman and Kruskal's γ. Results indicate the limitations of small samples (n $<$ 500) in correctly detecting item misfit, especially when several items are misfit. Item outfit shows very low performance and should not be used. Conditional infit with simulation based cutoffs performs slightly better than item-restscore with samples below n = 500. Both methods have strongly increased rates of false positives with large samples (n >= 1000). Large samples should use non-parametric bootstrap of subsamples with item-restscore to avoid type-1 errors. Finally, the importance of iterative analyses is emphasized since a situation where several items show underfit will induce seemingly overfit items. Underfit item should be removed one at a time, and a re-analysis conducted for each step to avoid erroneously removing items.

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.

The assessment of item fit under the Rasch model has for decades been conducted using various more or less arbitrary 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 $1 \pm 2/\sqrt{n}$ (Smith, Schumacker, and Bush 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.

Similar developments have recently taken place in the related field of confirmatory factor analysis. McNeish and Wolf (2024) have created an R package called `dynamic` that uses simulation to determine appropriate critical values for commonly used model fit metrics, both for ordinal data and interval data.

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 issues 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'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 and 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 and 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 (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.

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

Source: Article Notebook

Source: Article Notebook

Source: Article Notebook

Source: Article Notebook

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 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.

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, 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 ($1 \pm 2/\sqrt{n}$) 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 $4 \cancel{3} 3 = 36$ 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	Infit MSQ	Infit thresholds	Outfit MSQ	Outfit thresholds	Infit diff	Outfit 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

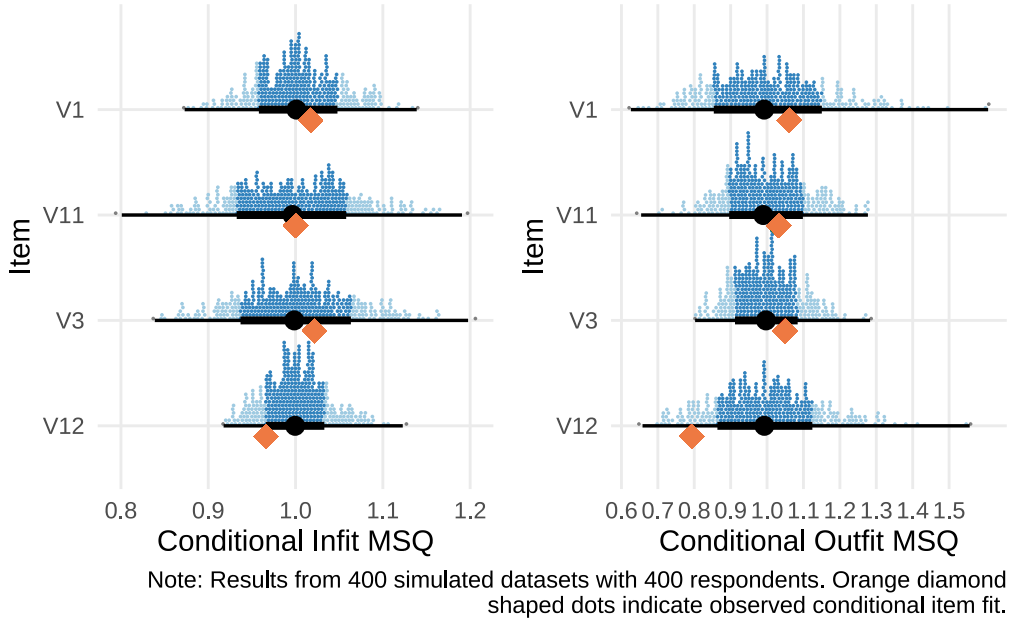

Figure 1: Distribution of simulation based item fit and estimated item fit from observed data

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

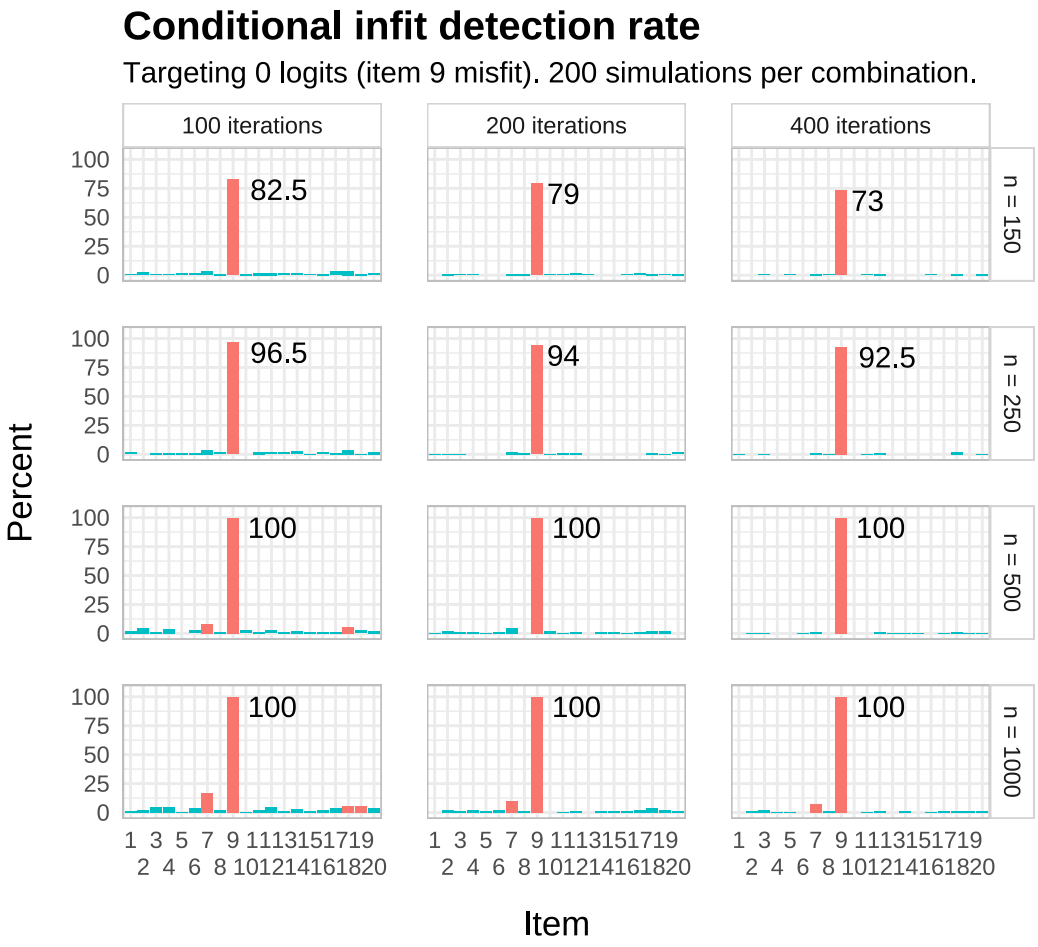

Figure 2: Conditional infit detection rate (misfit item at 0 logits)

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 150 and 250, but it also shows higher levels of false positives when sample size increases to 500 or more.

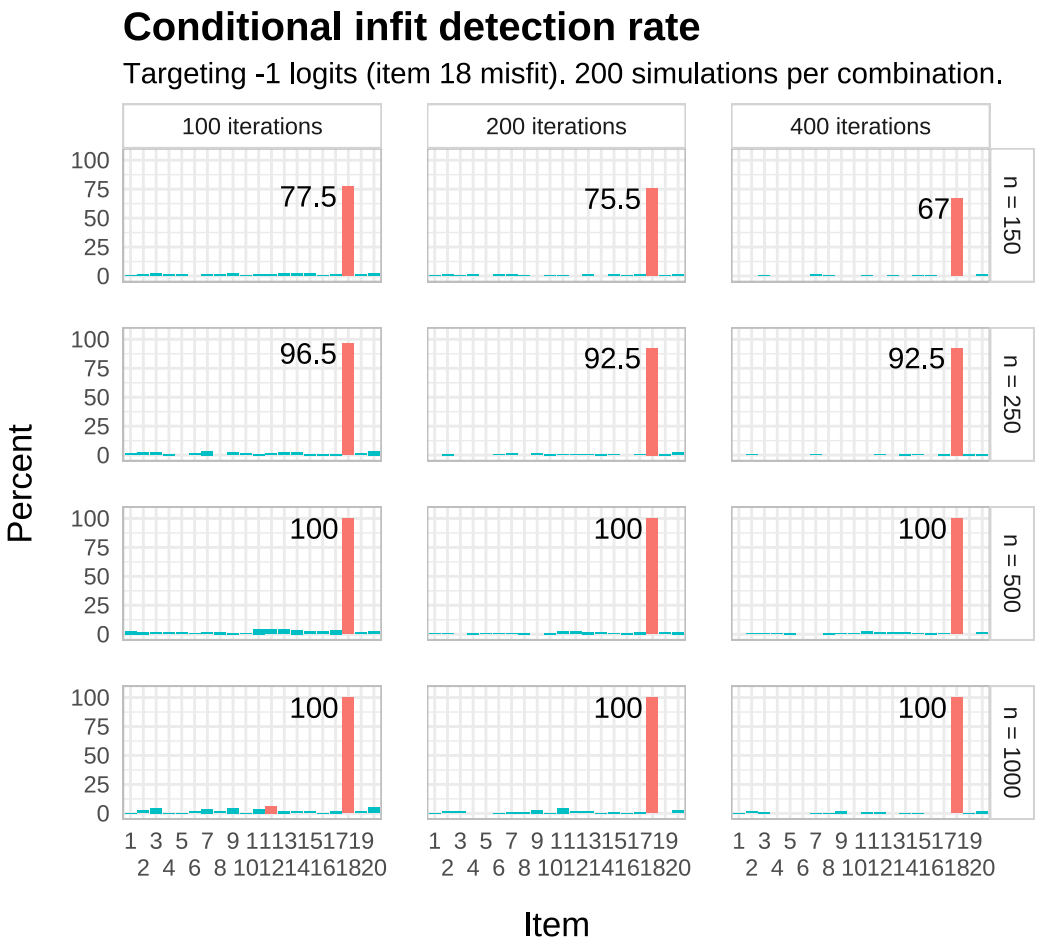

Figure 3: Conditional infit detection rate (misfit item at -1 logits)

Source: Article Notebook

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.

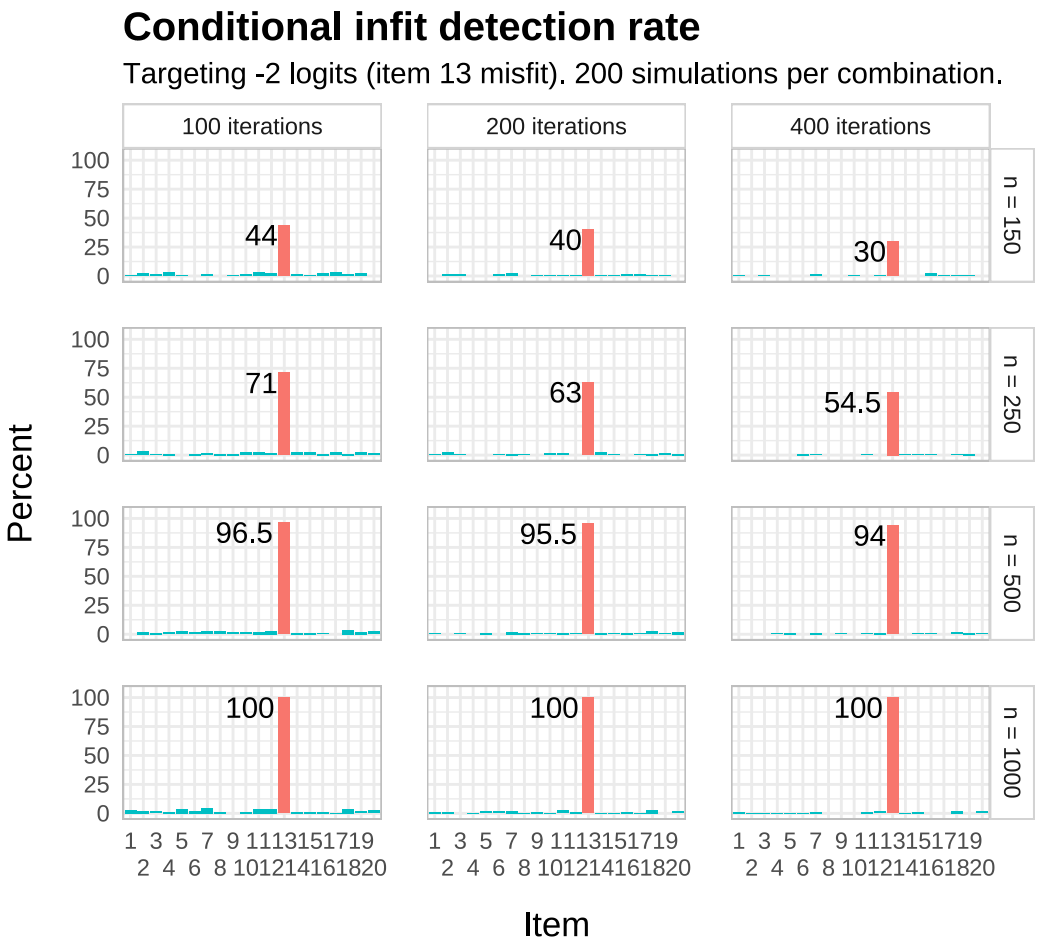

Figure 4: Conditional infit detection rate (misfit item at -2 logits)

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

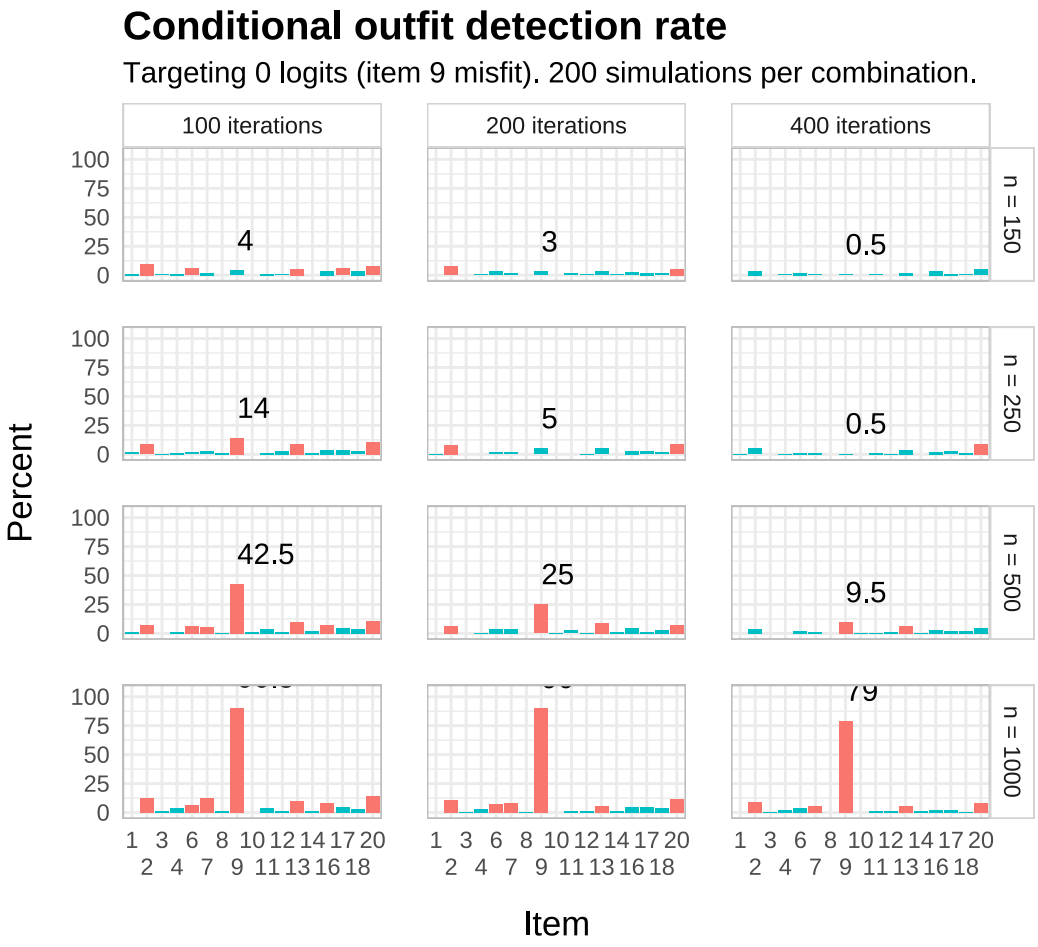

Figure 5: Conditional outfit detection rate (misfit item at 0 logits)

Source: Article Notebook

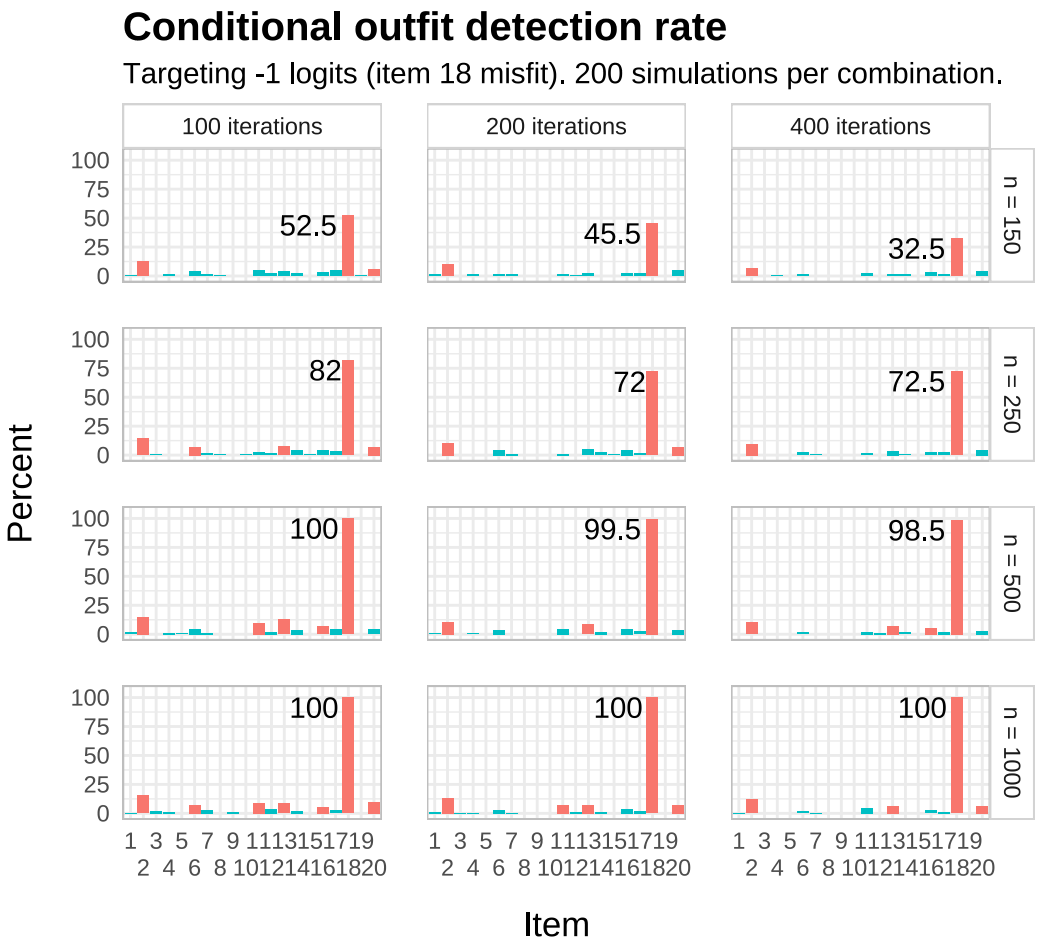

Figure 6: Conditional outfit detection rate (misfit item at -1 logits)

Source: Article Notebook

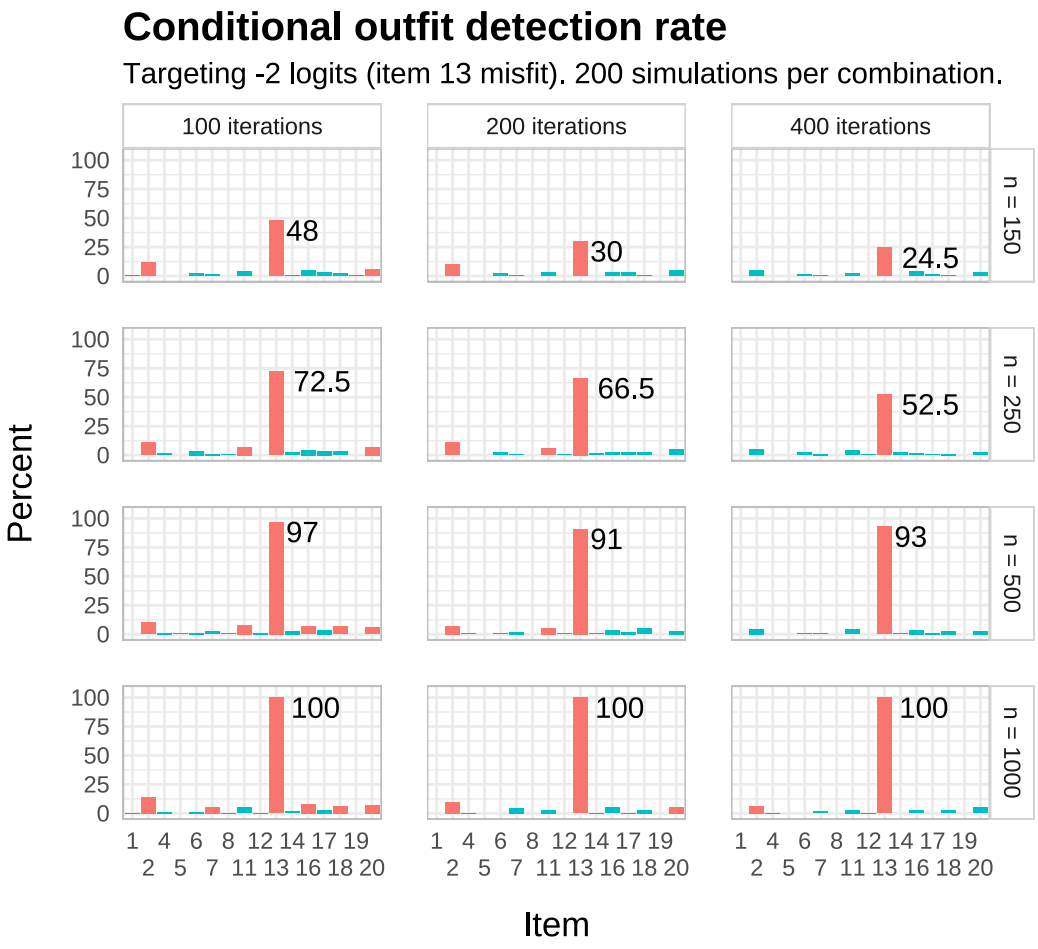

Figure 7: Conditional outfit detection rate (misfit item at -2 logits)

Source: Article Notebook

As shown in Figure 5, Figure 6, and Figure 7, 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 γ (Goodman and 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 and Santiago 2022) and outputs Benjamini-Hochberg corrected p-values (Benjamini and 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

Source: Article Notebook

4.1 Results

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

Source: Article Notebook

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 use 100 iterations with `RIgetfit()` for n $<$ 500, and 200 for n $>=$ 500, since this produced the best results in Study 1. Outfit is also included to see if it performs as bad as with one misfitting item.

Source: Article Notebook

Source: Article Notebook

Source: Article Notebook

5.0.1 Results

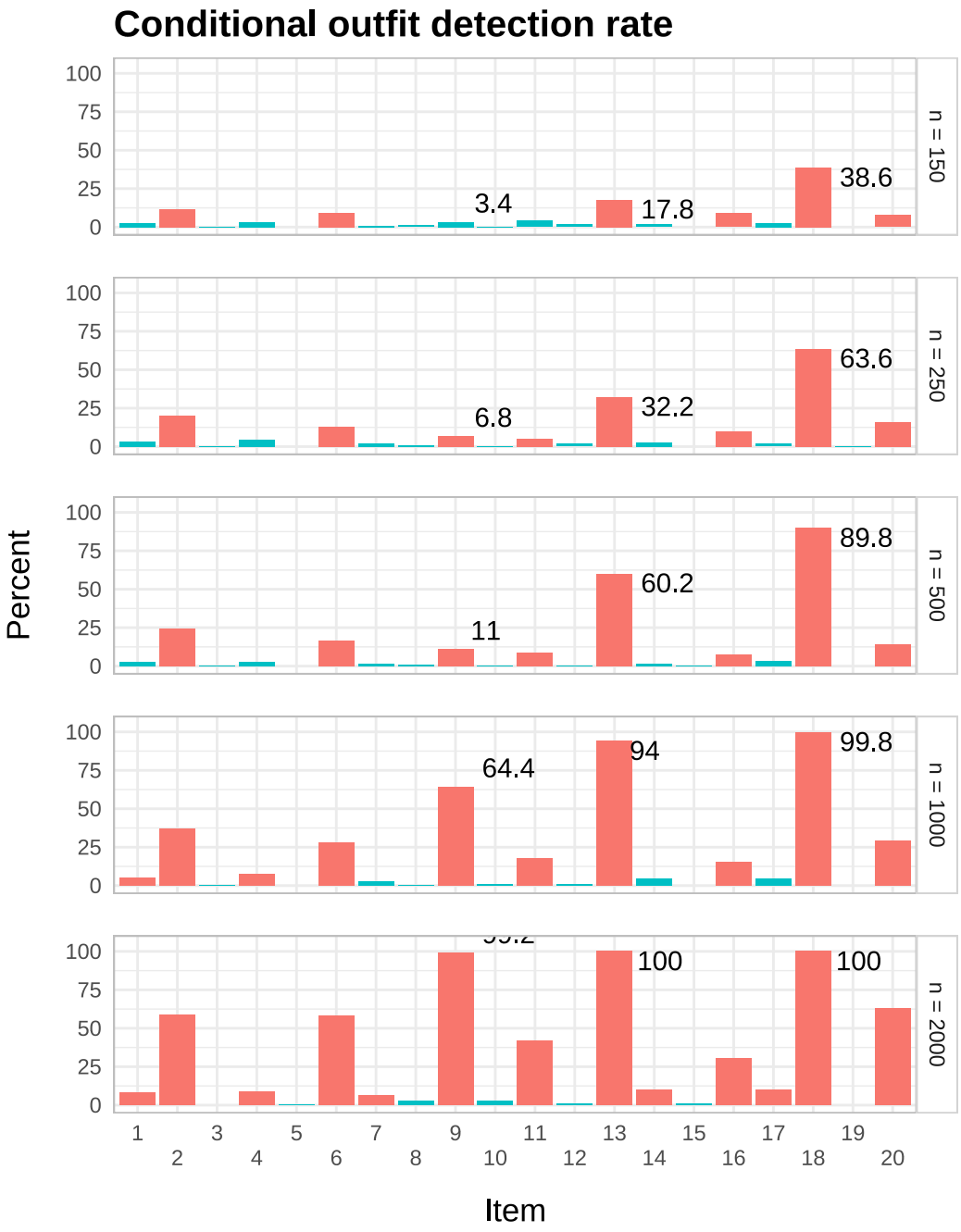

Figure 9: Conditional outfit detection rate with three misfitting items

Source: Article Notebook

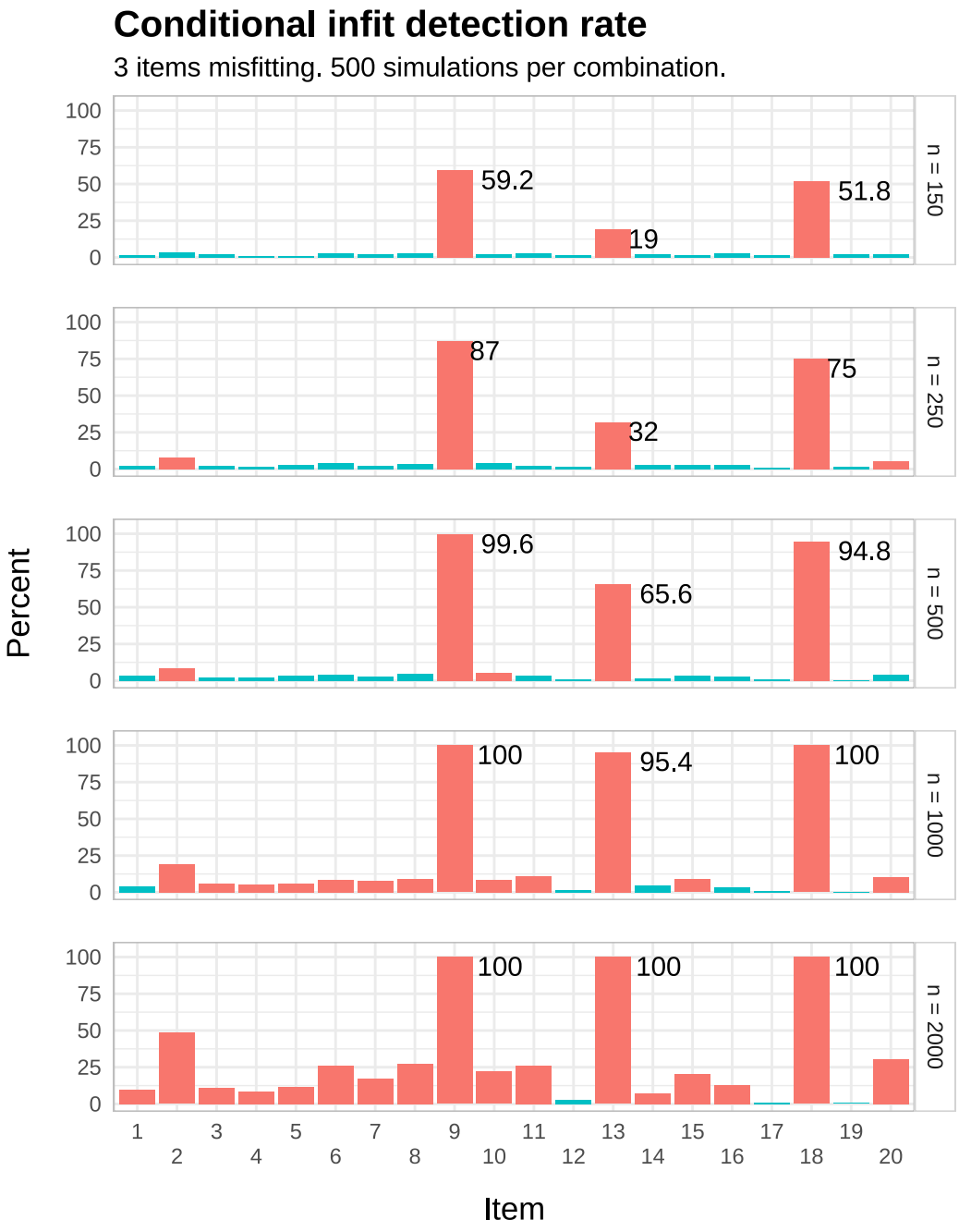

Figure 10: Conditional infit detection rate with three misfitting items

Source: Article Notebook

Looking at the performance of infit with three misfitting items (Figure 10), 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) again performs worse than infit.

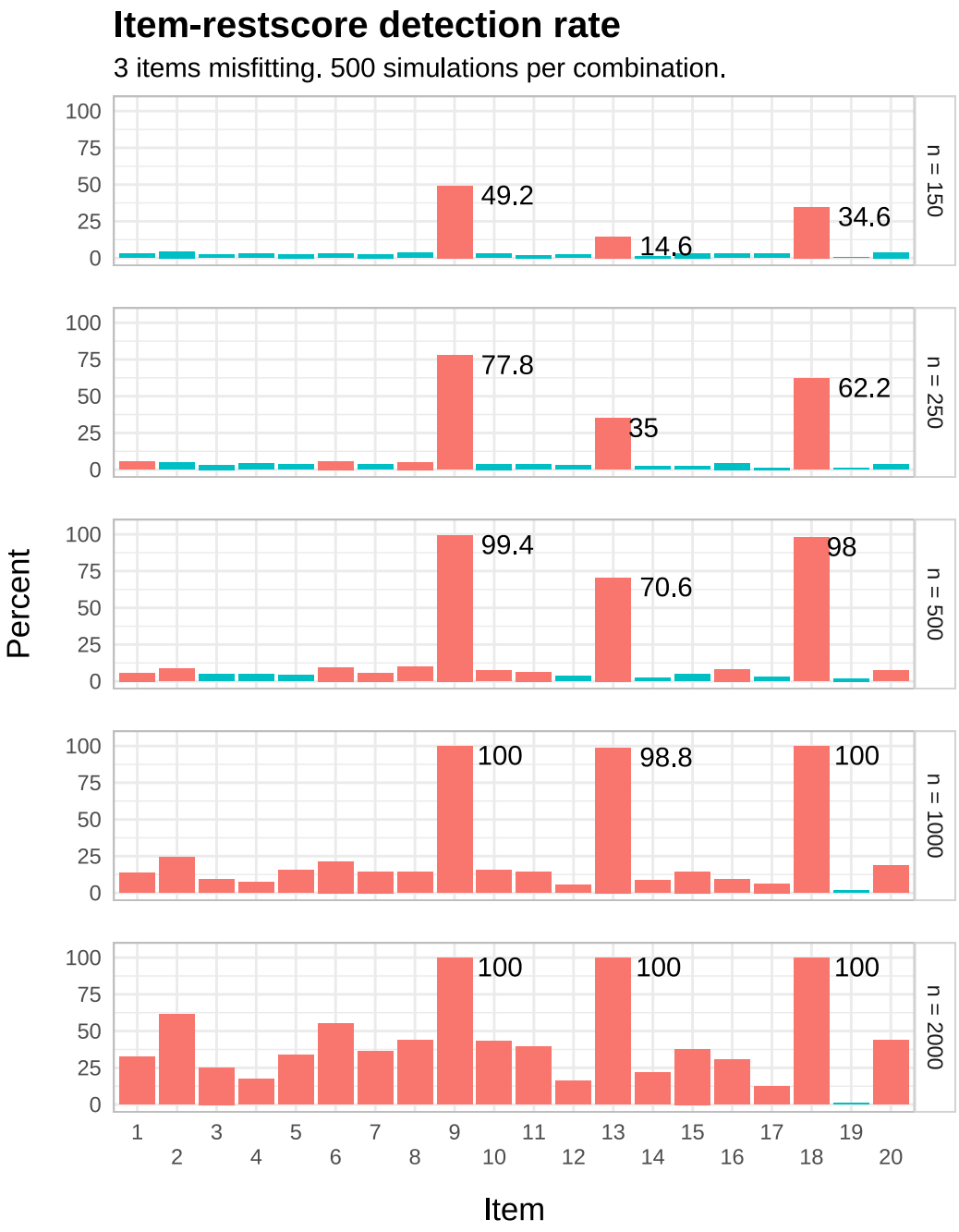

Figure 11

Source: Article Notebook

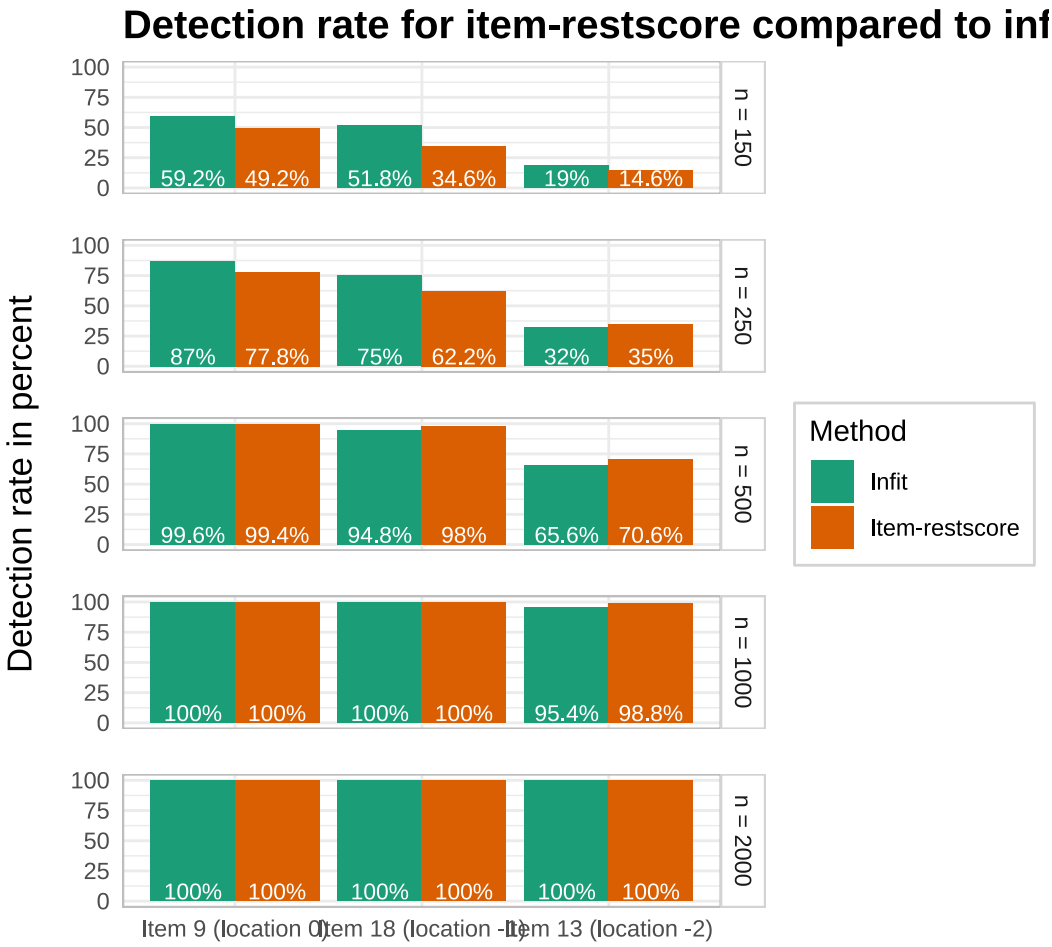

Figure 12: Detection rate for item-restscore compared to infit

Source: Article Notebook

Item-restscore (see Figure 11) shows comparable detection rate to infit and higher levels of false positives. A comparison is made between the two in Figure 12, 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

Table 2: Item-restscore summary results across all sample sizes

Item	Type of misfit	Percent
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

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 (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
V9	underfit	100.0
V18	underfit	99.6
V13	underfit	96.8
V2	overfit	67.2
V10	overfit	36.8
V6	overfit	35.2
V7	overfit	23.6
V5	overfit	19.6
V8	overfit	18.0
V15	overfit	17.2
V4	overfit	13.2
V16	overfit	9.6
V11	overfit	8.4
V20	overfit	7.6
V1	overfit	6.4
V3	overfit	4.4
V17	overfit	3.6
V14	overfit	1.6
V19	overfit	1.2
V12	overfit	0.4
V3	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

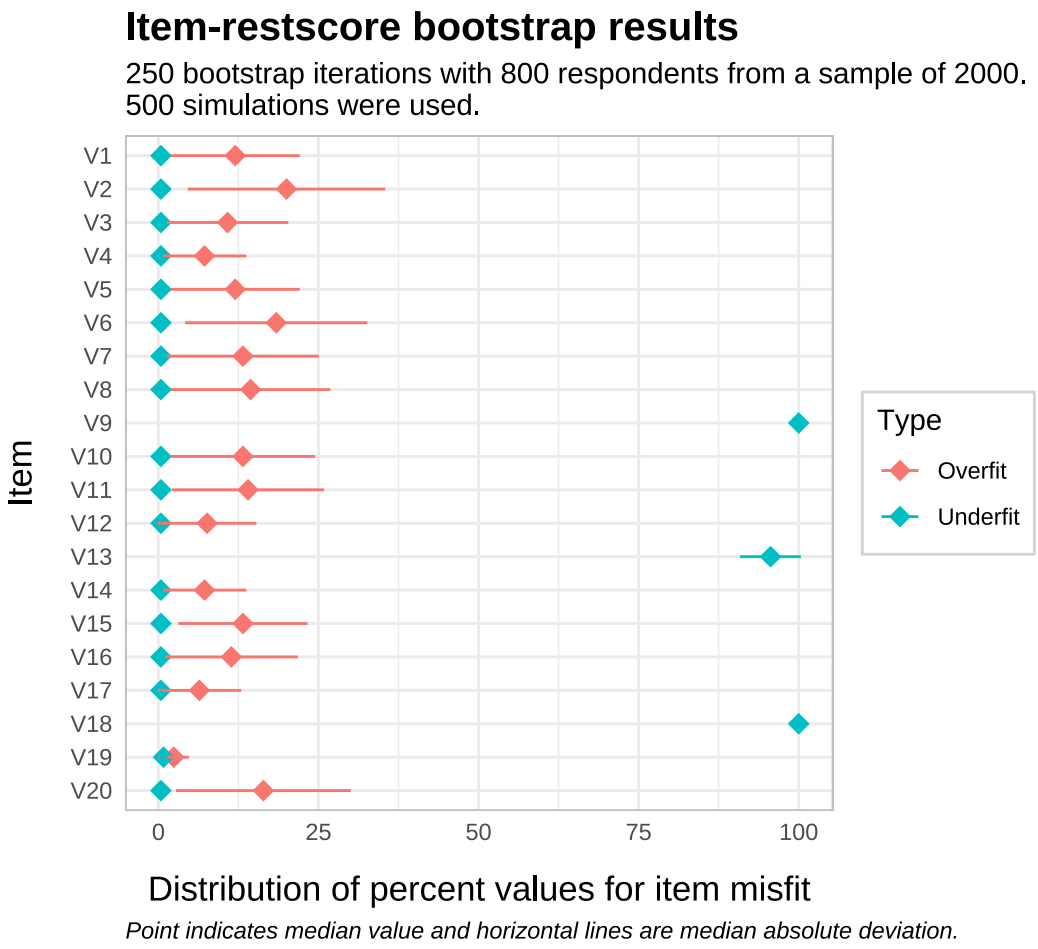

Figure 13: Item-restscore bootstrap results

Source: Article Notebook

Figure 13 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 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 bootstrap iterations for item-restscore in each simulation.

Table 4: Summary statistics for item-restscore bootstrap simulation

(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

Source: Article Notebook

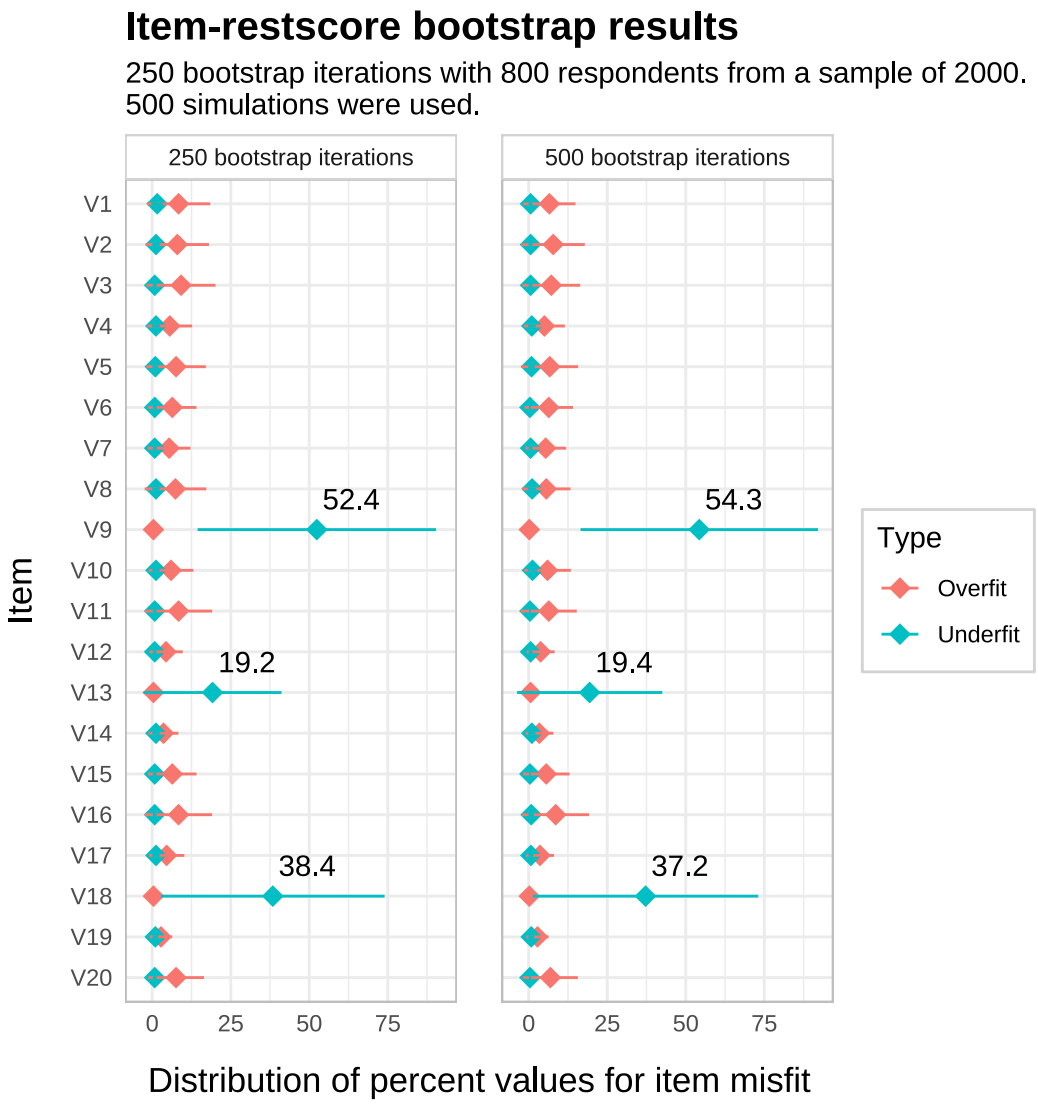

Figure 14

Source: Article Notebook

Item-restscore bootstrapping improves slightly on the single instance of item-restscore for the n = 150 condition (see Figure 14). When comparing to the previous results in Figure 11, 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). Compared to the results using infit (Figure 10), with detection rates of 59.2%, 19%, and 51.8%, item-restscore inferior also when bootstrapped.

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

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

Source: Article Notebook

Source: Article Notebook

7.1 Results 40 items

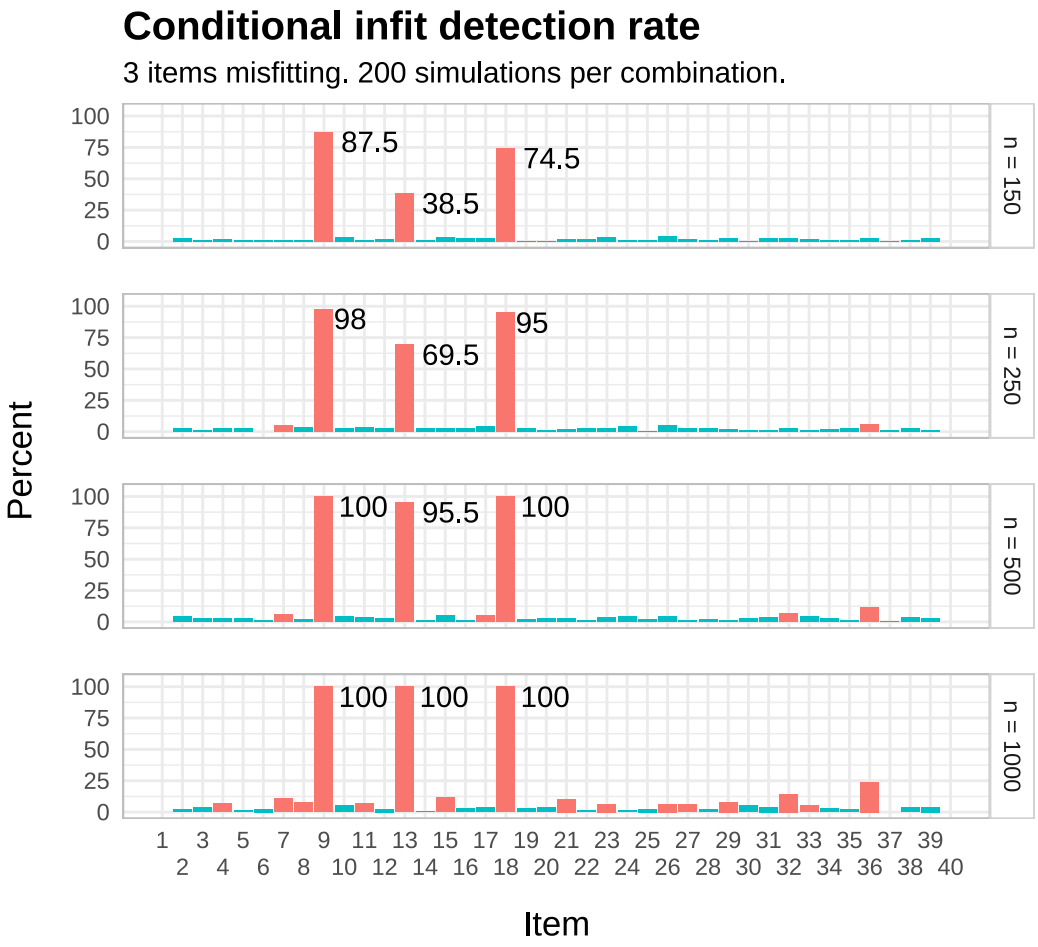

Figure 15

Source: Article Notebook

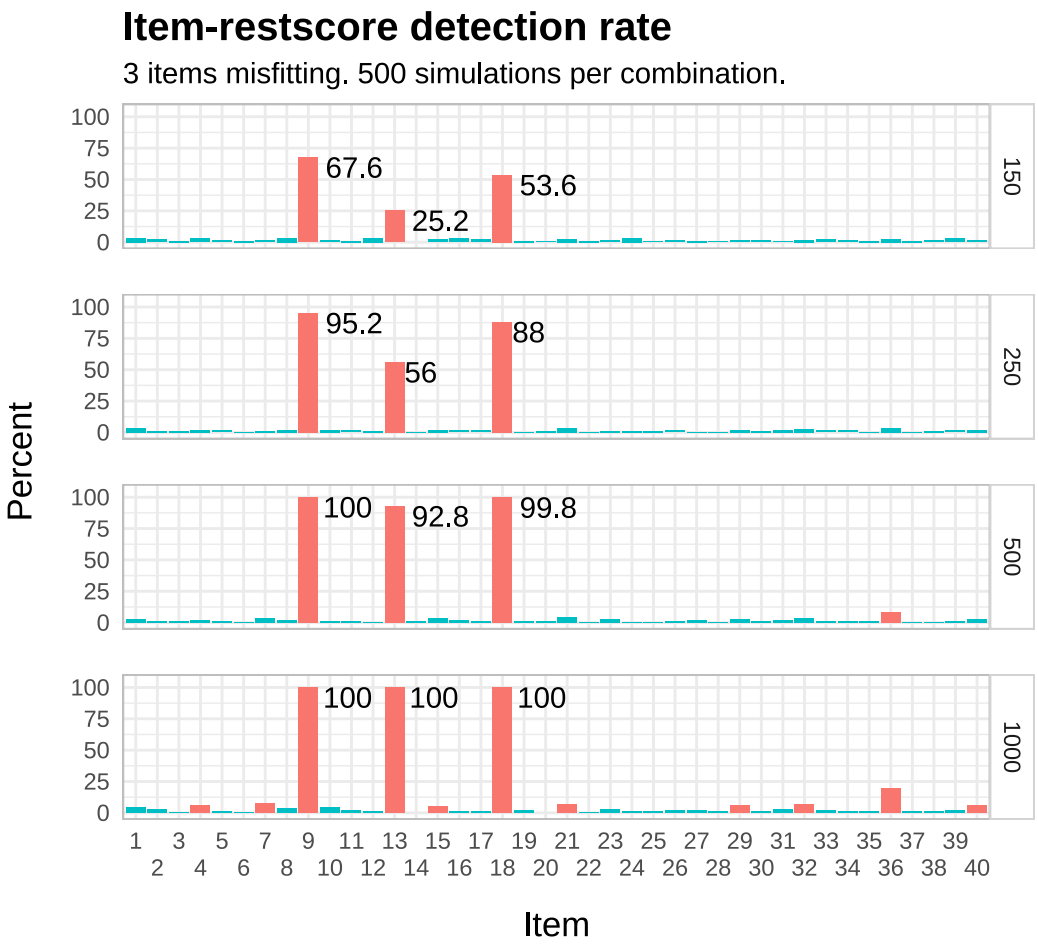

Figure 16

Source: Article Notebook

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

7.2 Results 10 items

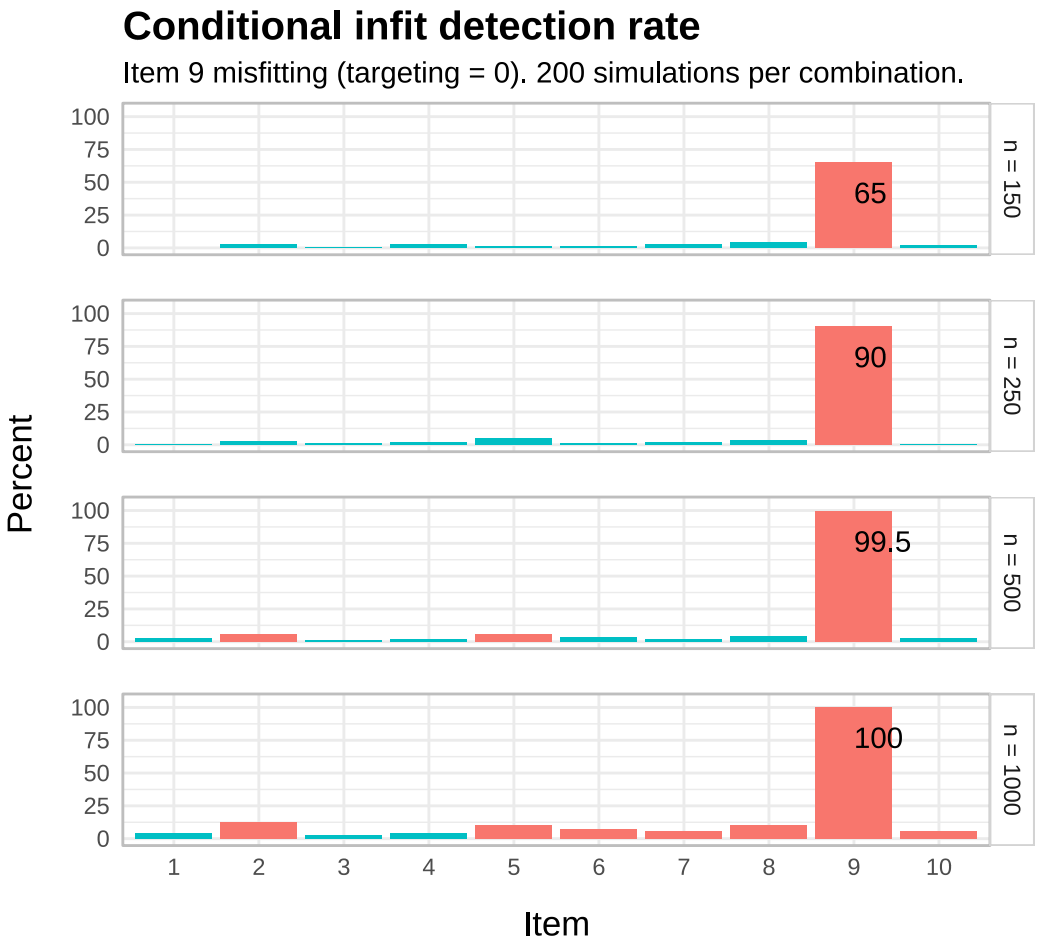

Figure 17

Source: Article Notebook

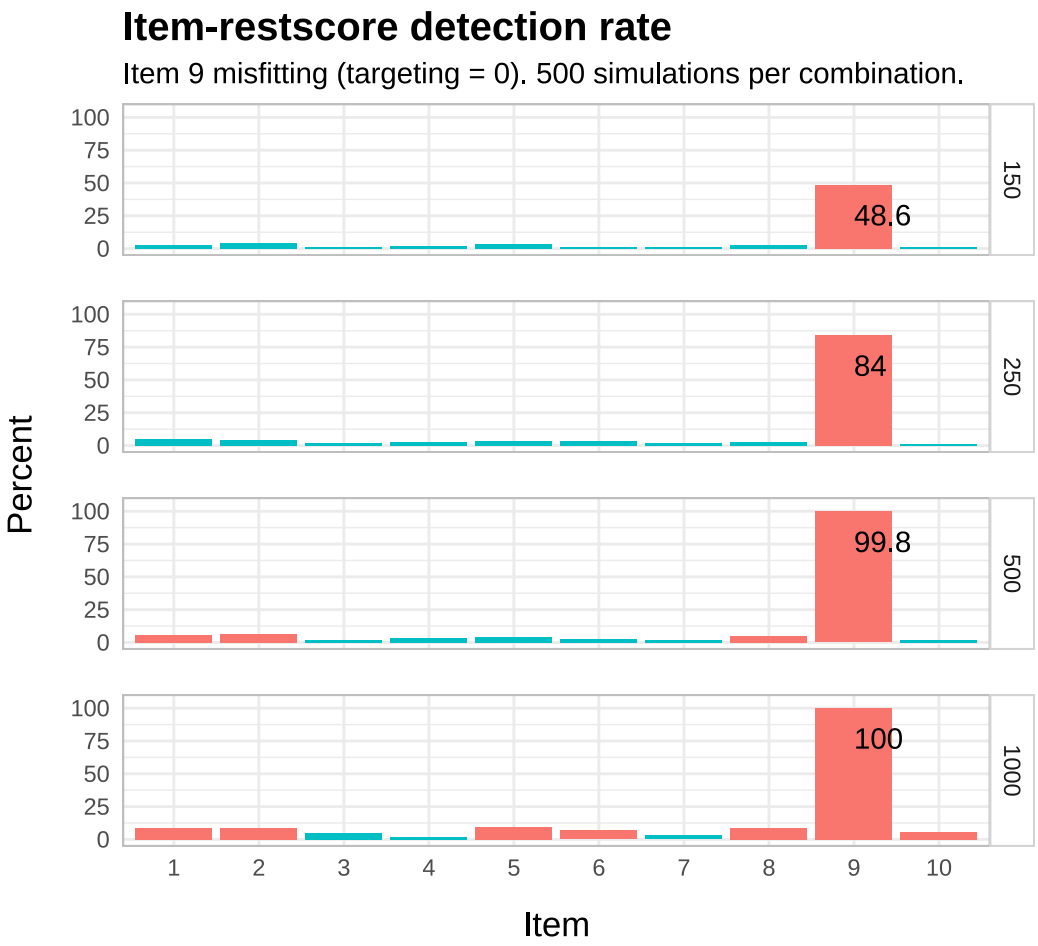

Figure 18

Source: Article Notebook

7.3 Summary figure

Source: Article Notebook

Source: Article Notebook

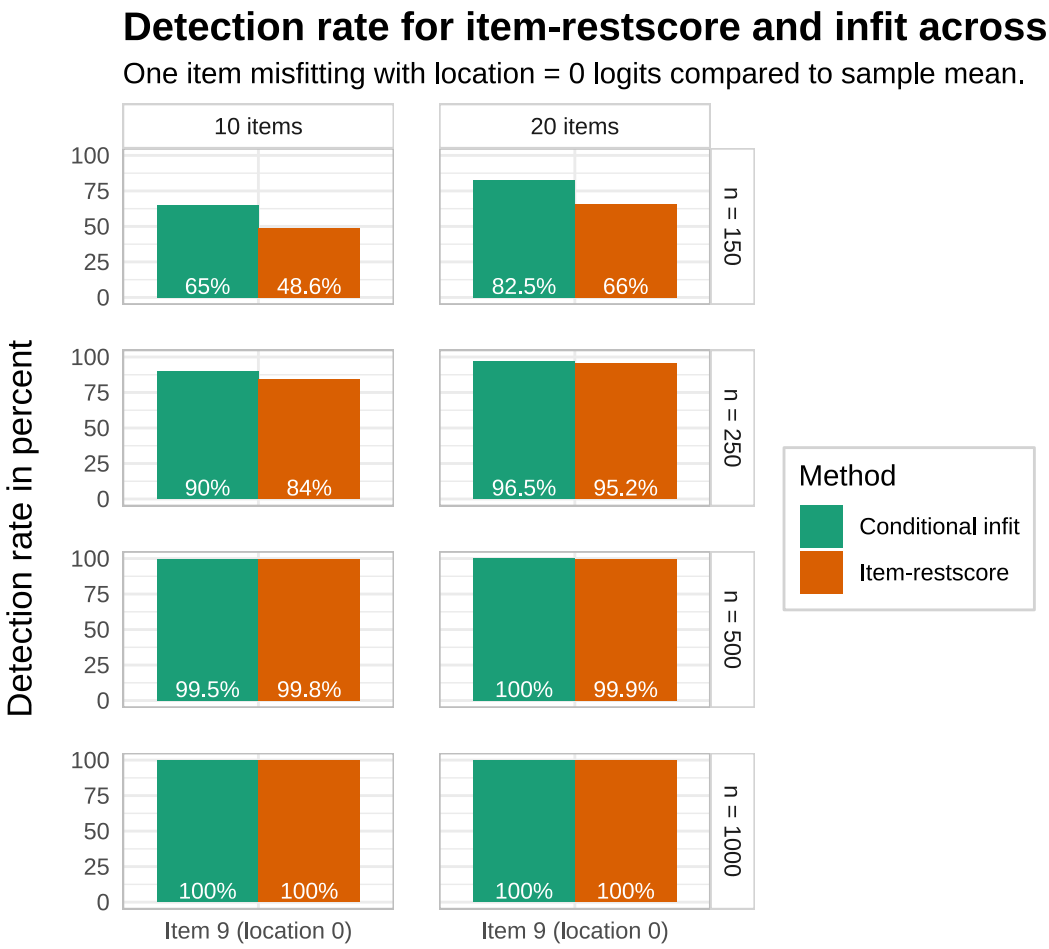

Figure 19: Detection rate for item-restscore and infit for 10 items

Source: Article Notebook

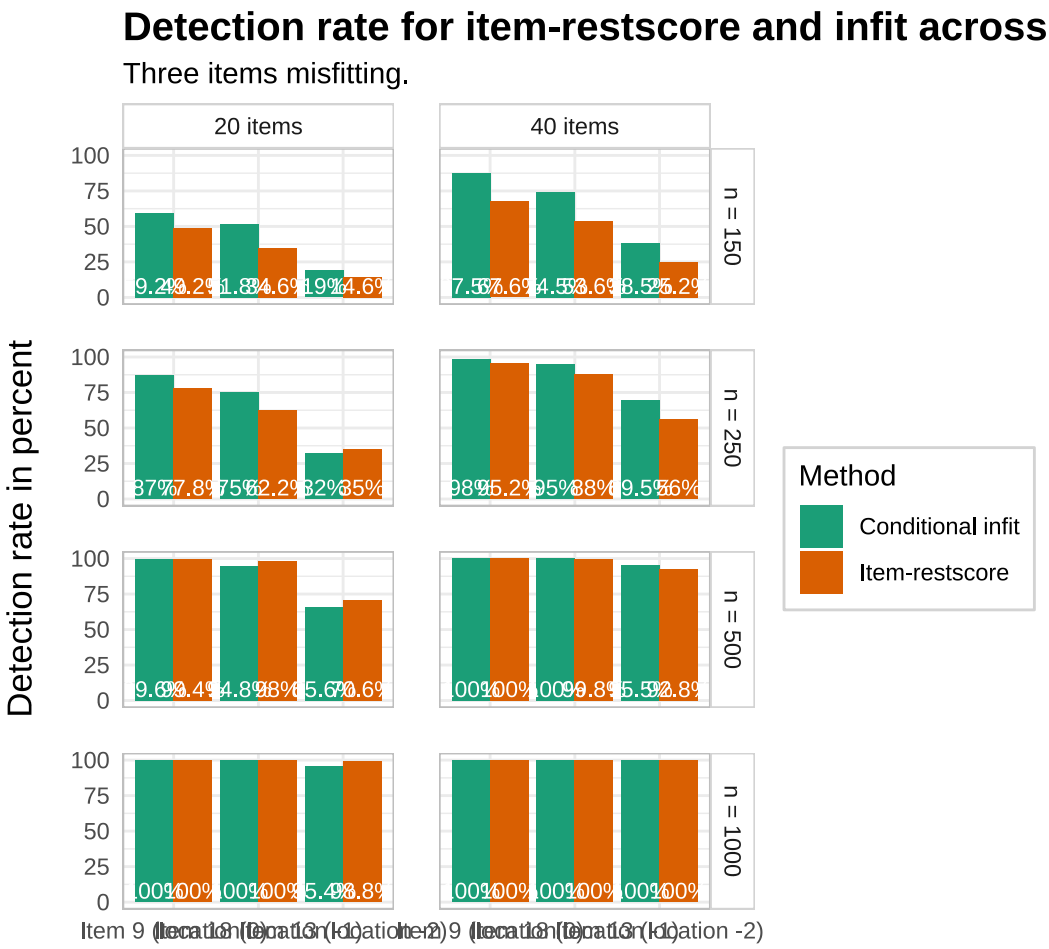

Figure 20: Detection rate for item-restscore and infit for 20 and 40 items

Source: Article Notebook

Figure 19 and Figure 20 summarize the findings from the two different comparisons of item numbers. Adding more items improves the detection rate substantially for both methods, particularly for smaller samples and the off-target items. 40 items compared to 20 items results in a larger improvement for infit over item-restscore for the n = 150 condition, but also the n = 250 and n = 500 conditions for the -2 logits off-target item.

8 Discussion

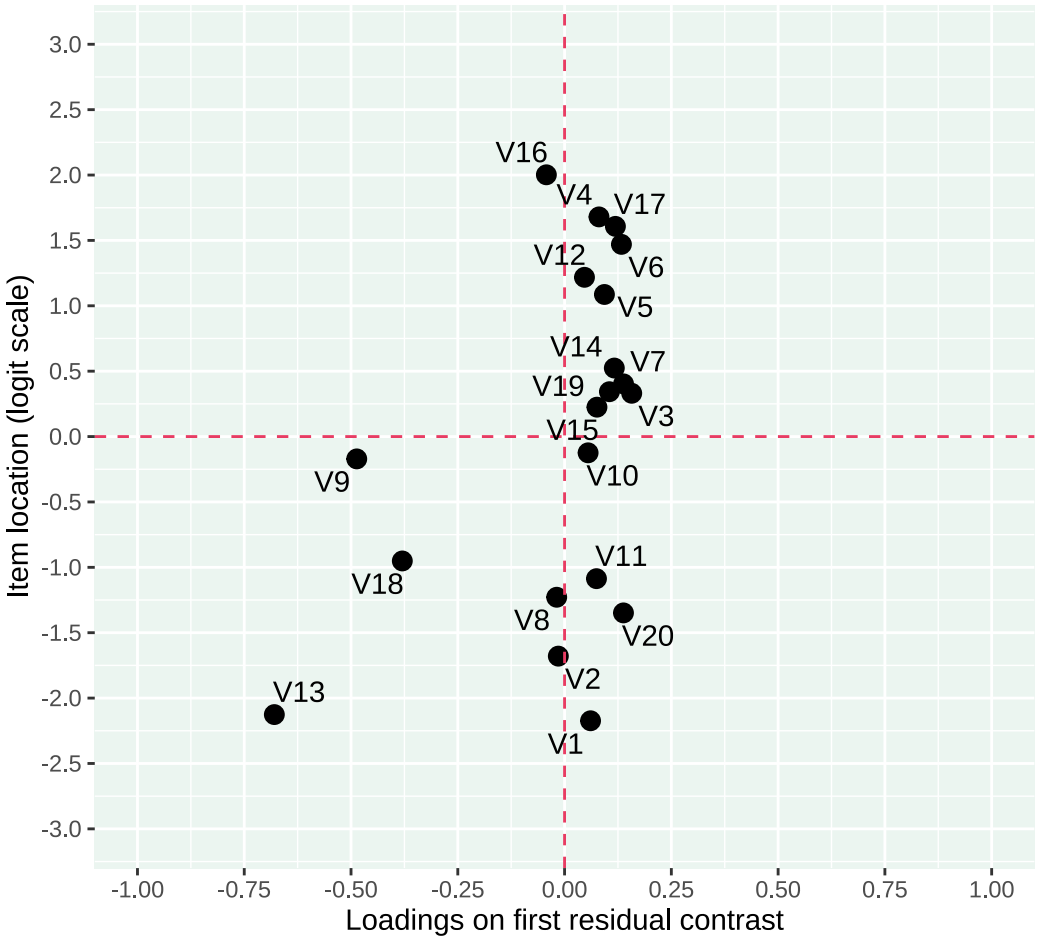

Figure 21

Source: Article Notebook

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 21 for an example) as well as Yen's Q3 residual correlations (Christensen, Makransky, and Horton 2017).

What does this look like with a small sample (Figure 22) and what is the item fit in the same small sample (Table 6 and Figure 23)?

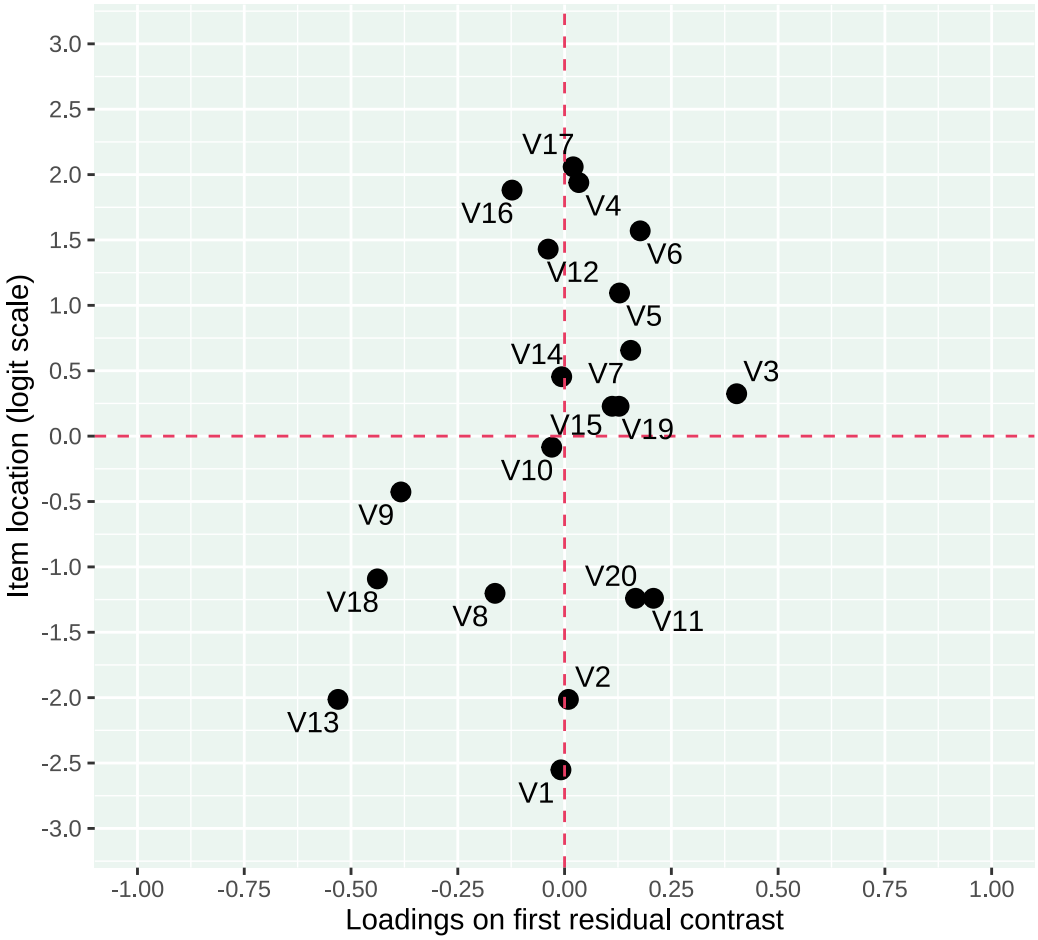

Figure 22

Source: Article Notebook

Table 6

Item	Infit MSQ	Infit thresholds	Outfit MSQ	Outfit thresholds	Infit diff	Outfit diff	Location
V1	0.996	[0.84, 1.268]	0.781	[0.445, 3.268]	no misfit	no misfit	-2.55
V2	0.905	[0.79, 1.172]	0.890	[0.7, 1.661]	no misfit	no misfit	-2.01
V3	0.937	[0.868, 1.217]	1.127	[0.708, 1.893]	no misfit	no misfit	0.32
V4	0.996	[0.805, 1.294]	0.769	[0.672, 3.2]	no misfit	no misfit	1.94
V5	0.985	[0.833, 1.181]	0.949	[0.533, 2.305]	no misfit	no misfit	1.09
V6	0.974	[0.804, 1.293]	1.059	[0.71, 1.505]	no misfit	no misfit	1.57
V7	0.999	[0.837, 1.259]	1.035	[0.777, 1.42]	no misfit	no misfit	0.66

Table 6

Item	Infit MSQ	Infit thresholds	Outfit MSQ	Outfit thresholds	Infit diff	Outfit diff	Location
V8	1.014	[0.842, 1.193]	1.023	[0.651, 2.678]	no misfit	no misfit	-1.20
V9	1.156	[0.742, 1.174]	1.224	[0.554, 2.483]	no misfit	no misfit	-0.43
V10	0.950	[0.832, 1.133]	0.887	[0.694, 2.017]	no misfit	no misfit	-0.08
V11	0.982	[0.816, 1.233]	1.027	[0.726, 1.268]	no misfit	no misfit	-1.24
V12	1.156	[0.831, 1.165]	1.342	[0.589, 2.434]	no misfit	no misfit	1.43
V13	1.055	[0.883, 1.191]	1.379	[0.705, 1.736]	no misfit	no misfit	-2.01
V14	0.981	[0.804, 1.174]	1.058	[0.737, 1.398]	no misfit	no misfit	0.45
V15	0.948	[0.759, 1.25]	0.887	[0.591, 2.128]	no misfit	no misfit	0.23
V16	0.864	[0.799, 1.217]	0.875	[0.622, 2.079]	no misfit	no misfit	1.88
V17	0.967	[0.774, 1.215]	0.791	[0.63, 3.558]	no misfit	no misfit	2.06
V18	1.143	[0.771, 1.261]	1.172	[0.664, 1.538]	no misfit	no misfit	-1.09
V19	0.887	[0.84, 1.171]	0.845	[0.714, 1.432]	no misfit	no misfit	0.23
V20	0.995	[0.867, 1.194]	0.925	[0.774, 1.334]	no misfit	no misfit	-1.24

Source: Article Notebook

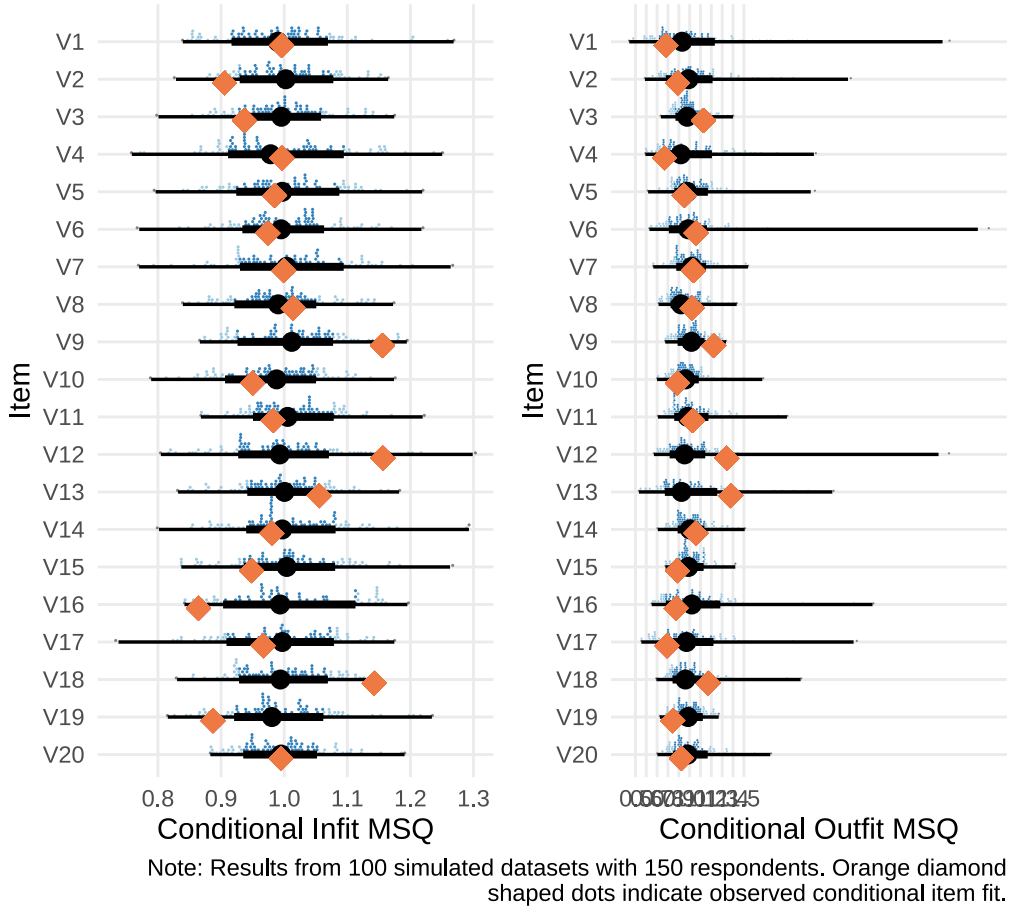

Figure 23

Source: Article Notebook

8.1 Limitations

Number of items could have been varied more and investigated further with more variation in sample sizes.

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?

iterations for the bootstrapped item-restscore - more testing could be conducted.

9 Conclusion

For sample size under 500, rely primarily on item infit with simulation based critical values, using 100 iterations with `RIgetfit()`. For sample sizes closer to 500, item-restscore is recommended, either as a single-run test, or bootstrapped. With samples larger than 500, bootstrapped item-restscore controls false positive rates well, while identifying misfitting items at high detection rates. Using 250 iterations for the bootstrapped item-restscore seems adequate, but more testing could be conducted.

Use both infit and item-restscore in your analysis process, if you have sample size below 1000.

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.

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

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.

Christensen, Karl Bang, Guido Makransky, and Mike Horton. 2017. "Critical Values for Yen's Q3: Identification of Local Dependence in the Rasch Model Using Residual Correlations." *Applied Psychological Measurement* 41 (3): 178–94. https://doi.org/10.1177/0146621616677520.

Goodman, Leo A., and William H. Kruskal. 1954. "Measures of Association for Cross Classifications." *Journal of the American Statistical Association* 49 (268): 732–64. https://doi.org/10.2307/2281536.

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.

McNeish, Daniel, and Melissa G. Wolf. 2024. "Direct Discrepancy Dynamic Fit Index Cutoffs for Arbitrary Covariance Structure Models." *Structural Equation Modeling: A Multidisciplinary Journal* 31 (5): 835–62. https://doi.org/10.1080/10705511.2024.2308005.

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.

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;  L

locale:
[1] sv_SE.UTF-8/sv_SE.UTF-8/sv_SE.UTF-8/C/sv_SE.UTF-8/sv_SE.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