The Journal of Open Source Software

FedIRT: An R package and shiny app for estimating federated item response theory models

Biying Zhou[1] and **Feng Ji**[1¶]

1 Department of Applied Psychology & Human Development, University of Toronto, Toronto, Canada ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 19 September 2024
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

DRAFT

Summary

We developed an R package, FedIRT, to estimate item response theory (IRT) models—including 1PL, 2PL, and graded response models—with additional privacy features. This package enables parameter estimation in a distributed manner without compromising accuracy, leveraging recent advances in federated learning. Numerical experiments demonstrate that federated IRT estimation achieves statistical performance comparable to mainstream IRT packages in R, with the added benefits of privacy preservation and minimal communication costs. The R package also includes a user-friendly Shiny app that allows clients (e.g., individual schools) and servers (e.g., school boards) to easily apply our proposed method.

Statement of Need

IRT (Embretson & Reise, 2013) is a statistical modeling framework grounded in modern test theory, frequently used in the educational, social, and behavioral sciences to measure latent constructs through multivariate human responses. Traditional IRT estimation mandates the centralization of all individual raw response data in one location, which potentially compromises the privacy of the data and participants (Lemons, 2014).

Federated learning has emerged as a field addressing data privacy issues and techniques for parameter estimation in a decentralized, distributed manner. However, there is currently no package available in psychometrics, especially in the context of IRT, that integrates federated learning with IRT model estimation.

Popular IRT packages in R, such as `mirt` (Chalmers, 2012) and `ltm` (Rizopoulos, 2007), require storing and computing all data in a single location, which can potentially lead to violations of privacy policies when dealing with highly sensitive data (e.g., high-stakes student assessment data).

Therefore, we have developed a specialized R package, FedIRT, which integrates federated learning with IRT and includes an accompanying Shiny app designed to address real-world implementation challenges and reduce the burden of learning R programming for users. This app implements the method in a user-friendly and accessible manner.

Method

Here we briefly introduce the key idea behind integrating federated learning with IRT. For technical details, please refer to our methodological discussions on Federated IRT (Zhou & Ji, 2023, 2024, In submission).

Model formulation

The two-parameter logistic (2PL) IRT model is often considered the most popular IRT model in practice. In the 2PL model, the response of person i to item j is binary ($X_{ij} \in 0, 1$), and the probability that person i answers item j correctly, given discrimination parameter α_j and difficulty parameter β_j, is given by:

$$P(X_{ij} = 1|\theta_i) = \frac{e^{\alpha_j(\theta_i - \beta_j)}}{1 + e^{\alpha_j(\theta_i - \beta_j)}}$$

To make our package available for polytomous response, we also developed a federated learning estimation algorithm for the Generalized Partial Credit Model (GPCM) in which the probability of a person with the ability θ_i obtaining x scores in item j is:

$$P^{\text{GPCM}}(X_{ij} = x|\theta_i) = \frac{e^{\sum\limits_{h=1}^{x} \alpha_j(\theta_i - \beta_{jh})}}{\sum\limits_{c=0}^{m_j} e^{\sum\limits_{h=1}^{c} \alpha_j(\theta_i - \beta_{jh})}}$$

In this function, β_{jh} is the difficulty of scoring level h for item j, and for each item j, all difficulty levels have the same discrimination α_j. m_j is the maximum score of item j.

Model estimation

In both the 2PL and GPCM models, we often assume that ability follows a standard normal distribution, allowing us to apply marginal maximum likelihood estimation (MMLE).

We use a combination of traditional MMLE with federated average (FedAvg) and federated stochastic gradient descent (FedSGD) (McMahan et al., 2017). In our case, the log-likelihood and partial gradients are sent from the clients to the server. The server then uses FedSGD to update the item parameters and sends them back to the clients.

Taking the 2PL model as an example, the marginal log-likelihood function l for each school k can be approximated using Gaussian-Hermite quadrature with q equally-spaced levels. Let $V(n)$ be the ability value at level n, and $A(n)$ be the weight at level n.

$$l_k \approx \sum_{i=1}^{N_k} \sum_{j=1}^{J} X_{ijk} \times \log[\sum_{n=1}^{q} P_j(V(n))A(n)] + (1 - X_{ijk}) \times \log[\sum_{n=1}^{q} Q_j(V(n))A(n)]$$

By applying FedAvg, the server collects the log-likelihood values from all k schools and then sums up all the likelihood values to get the overall log-likelihood value: $l = \sum\limits_{k=1}^{K} l_k$.

The server collects a log-likelihood value l_k and all derivatives $\frac{l_k}{\partial \alpha_j}$ and $\frac{l_k}{\partial \beta_j}$ from all clients, then observe that $\frac{\partial l}{\partial \alpha_j} = \sum\limits_{k=1}^{K} \frac{l_k}{\partial \alpha_j}$ and $\frac{\partial l}{\partial \beta_j} = \sum\limits_{k=1}^{K} \frac{l_k}{\partial \beta_j}$ by FedSGD, the server sums up all log-likelihood values and derivative values.

Also, we provided an alternative solution, Federated Median, which uses the median of the likelihood values to replace the sum of likelihood values in Fed-MLE (Liu et al., 2020), with additional robustness to handle outliers in input data.

With estimates of α_j and β_j in 2PL or β_{jh} in GPCM, we can obtain empirical Bayesian estimates of students' ability (Bock & Aitkin, 1981).

DRAFT

Comparison with existing packages

We demonstrate that our package generates comparable results to established IRT packages, such as mirt (Chalmers, 2012).

Figure 1 and Figure 2 show the comparison of the discrimination and difficulty parameters between `mirt` and `FedIRT` based on `example_data_2PL` in our package.

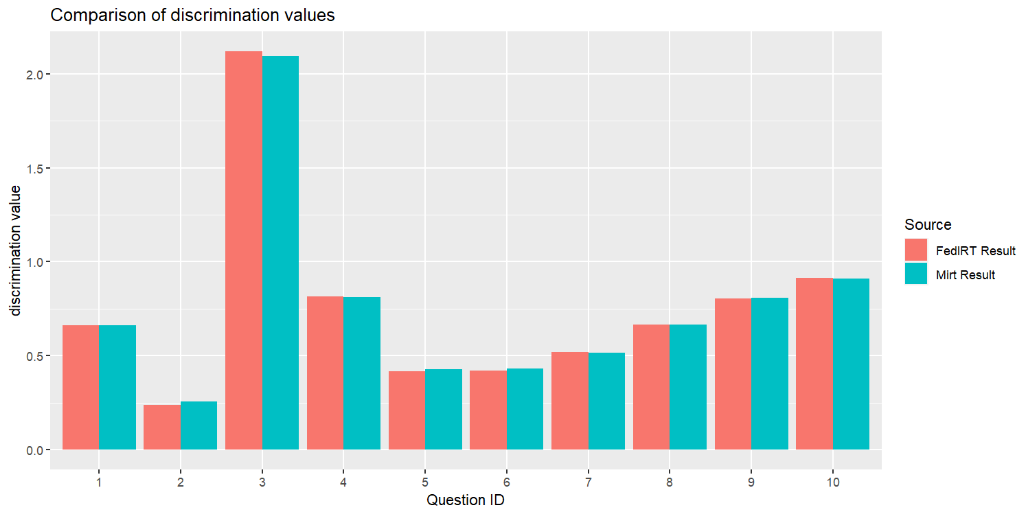

Figure 1: Discrimination parameter estimates comparison

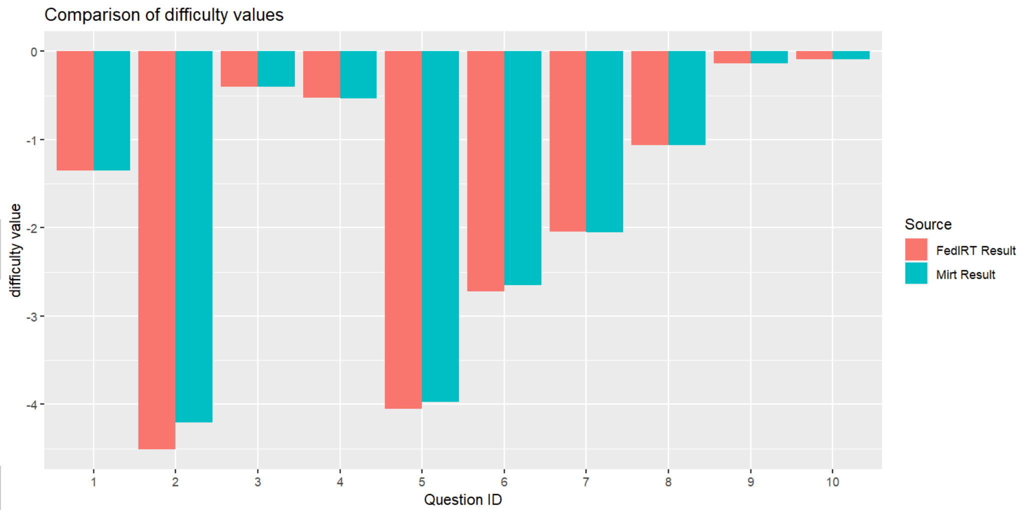

Figure 2: Difficulty parameter estimates comparison

Availability

The R package `FedIRT` is publicly available on Github. It could be installed and run by using the following commands:

```
devtools::install_github("Feng-Ji-Lab/FedIRT")
library(FedIRT)
```

Example of the integrated function

We provide a function `fedirt_file()` in the package, and the detailed usage of the function is shown in the user manual. We demonstrate an example here.

Suppose we have a dataset called `dataset.csv`, and the head of this dataset is shown below. There should be one column indicating the school, for example, "site" here. Each other column indicates an item, and each row represents an answering status.

site	X1	X2	X3	X4	X5
10	1	0	0	0	0
7	0	0	1	0	0
9	0	0	1	1	1
1	1	0	1	1	1
2	1	0	0	0	0

First, we need to read the dataset.

```r
# read dataset
data <- read.csv("dataset.csv", header = TRUE)
```

Then, we call the function `FedIRT::fedirt_file()` to obtain the result. It returns a list of parameter estimates for item discriminations, item difficulties, and each sites' effect and each students' abilities.

```r
# call the fedirt_file function
result <- fedirt_file(data, model_name = "2PL")
```

Finally, we can extract the results or use the parameter estimates for further analysis.

```r
result$a
result$b
```

Apart from using the results for further analysis, we can also use `summary()` to generate a snapshot of the result. Here is an example below.

```r
summary(result)
```

Then, the result will be printed in the console as follows:

```
Summary of FedIRT Results:


Counts:
function gradient
     735      249

Convergence Status (convergence):
Converged

Log Likelihood (loglik):
[1] -7068.258

Difficulty Parameters (b):
 [1] -185.88151839    0.99524035    0.92927254   ...

Discrimination Parameters (a):
 [1]  0.0028497700  0.8440140746 -0.1190176844 ...

```

DRAFT

```
Ability Estimates:
School 1:
 [1] -1.127097195 -0.922572829 -0.993953038  ...
School 2:
 [1] -1.41454573  1.78068772  1.87469389 ...
...

End of Summary
```

Example of the personscore function

We provide a function personscore in the package to obtain ability estimates. The detailed usage of the function is shown in the user manual. We demonstrate an example here.

```
personscoreResult = personscore(result)
summary(personscoreResult)
```

Summary of the person score is shown below.

```
Summary of FedIRT Person Score Results:

Ability Estimates:
School 1:
 [1] -1.127097195 -0.922572829 -0.993953038  ...
School 2:
 [1] -1.41454573  1.78068772  1.87469389 ...
...

End of Summary
```

Example of the personfit function

We provide a function personfit in the package. The detailed usage of the function is shown in the user manual. We demonstrate an example here.

```
personfitResult = personfit(result)
summary(personfitResult)
```

After getting the result, use personfit function to get the person score result from result by personfit(result).

```
Summary of FedIRT Person Fit Results:

Fit Estimates:
School 1:
              Lz           Zh       Infit    Outfit
4    0.7584470759  0.923163304 0.002323484 0.1482672
16  -0.7562447025 -1.131668935 0.005457117 0.1799583
27   0.3417488360  0.357870094 0.005966933 0.1734402
33  -0.9244005411 -1.359789298 0.179834037 0.2266634
...
School 2:
            Lz           Zh        Infit    Outfit
5   -0.90114567 -1.175767350 0.0009824580 0.1535794
8   -1.47957351 -1.888763364 0.1491518127 0.2255230
18  -0.13292541 -0.228824721 0.1104556086 0.2007658
19  -0.17257549 -0.277699184 0.0075031313 0.1350857
...
```

DRAFT

Standard error (SE) calculation

To obtain SE, we can call the `SE()` function and input a `fedirt` object to display standard errors of item parameter estimates.

```r
SE(result)
```

Below is the result of SE.

```
$a
 [1] 0.0041815497 0.1638884452 0.1204696925 ...
$b
 [1] 272.43863961   0.20737386   1.25896302 ...
```

Example of the Shiny App

To provide wider access for practitioners in real-world applications, we include the Shiny user interface in our package. A detailed manual was provided in the package. Taking the 2PL as an example, we illustrate how to use the Shiny app below.

In the first step, the server end (e.g., test administer, school board) can be launched by running the Shiny app `runserver()` and the client-end Shiny app can be initialized with `runclient()` with the interface shown below:

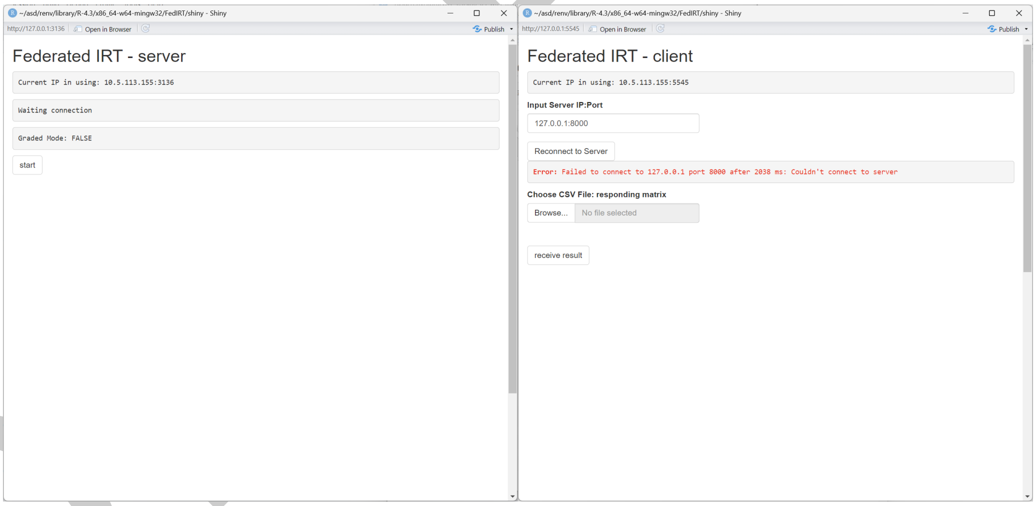

Figure 3: The initial server and client interface.

When the client first launches, it will automatically connect to the localhost port `8000` by default.

If the server is deployed on another computer, type the server's IP address and port (which will be displayed on the server's interface), then click "Reconnect". The screenshots of the user interface are shown below.

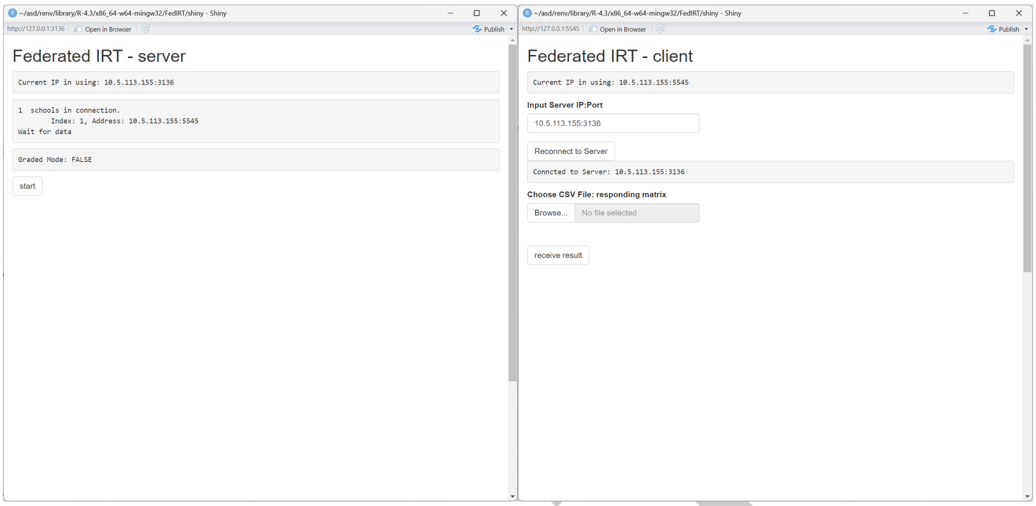

Figure 4: Server and client interface when one school is connected.

Then, the client should choose a file to upload to the local Shiny app to perform local calculations, without sending it to the server. The file should be a CSV file with either binary or graded responses. All clients should share the same number of items and the same maximum score for each item (if the responses are polytomous); otherwise, an error message will suggest checking the datasets of all clients.

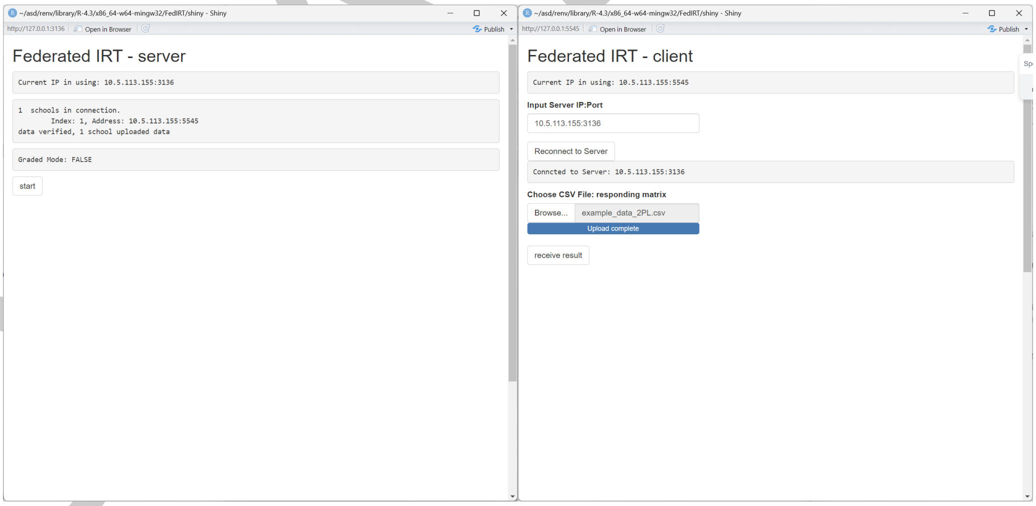

Figure 5: Server interface when one school uploaded dataset and client interface when a dataset is uploaded successfully.

After all the clients upload their data, the server should click "Start" to begin the federated estimation process. After the model converges, the clients should click "Receive Result". The server will display all item parameters, and the clients will display all item parameters and individual ability estimates.

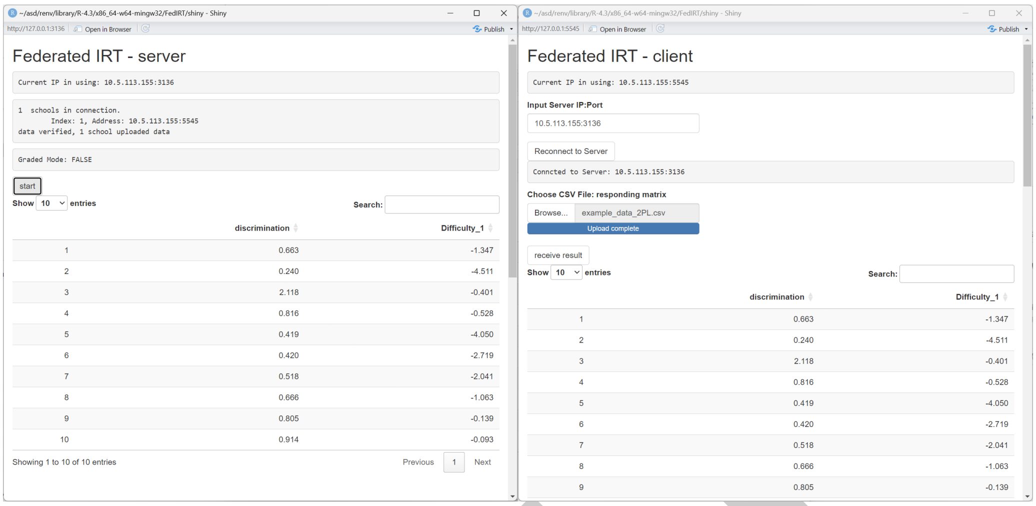

Figure 6: Server interface when estimation is completed and client interface when the results received.

The clients will also display bar plots of the ability estimates.

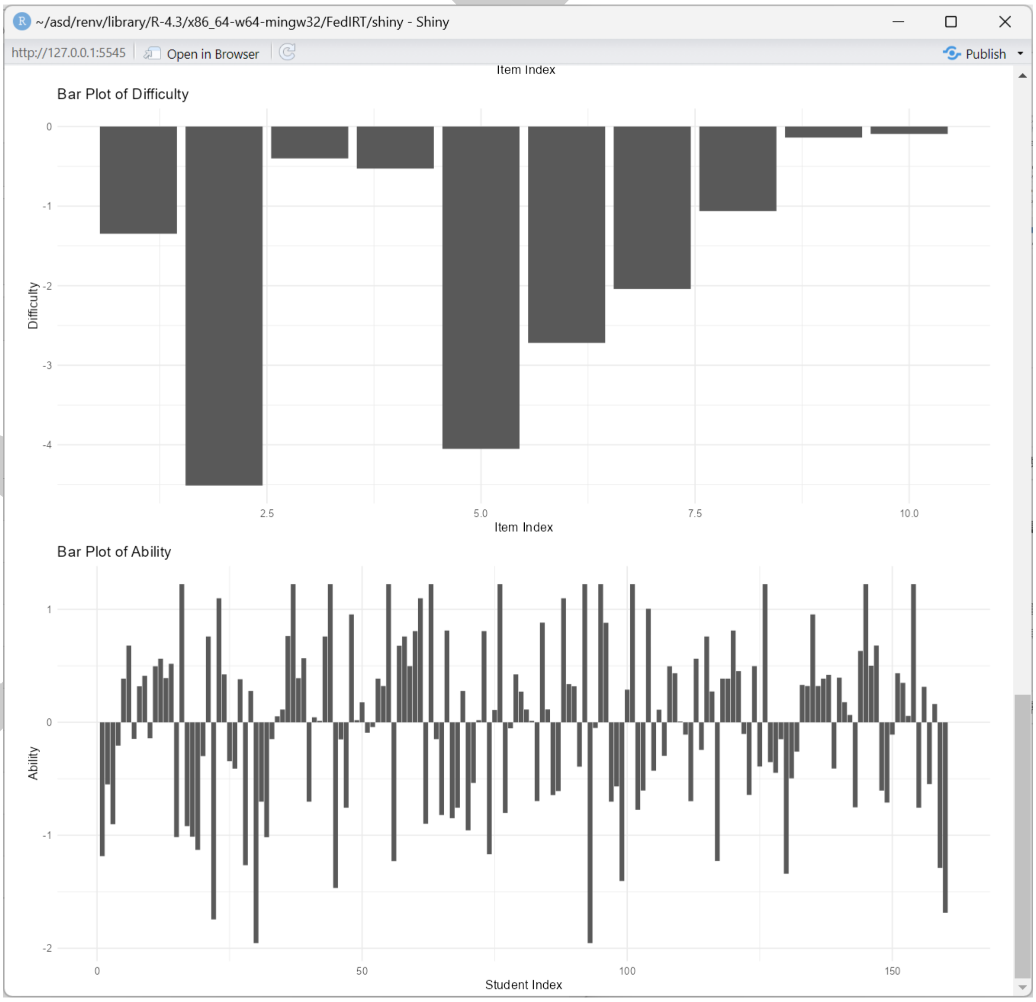

Figure 7: Client interface for displaying results.

References

Bock, R. D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika*, *46*(4), 443–459. https://doi.org/10.1007/BF02293801

Chalmers, R. P. (2012). Mirt: A multidimensional item response theory package for the r environment. *Journal of Statistical Software*, *48*, 1–29. https://doi.org/10.18637/jss.v048.i06

Embretson, S. E., & Reise, S. P. (2013). *Item response theory*. Psychology Press. https://doi.org/10.4324/9781410605269

Lemons, M. Q. (2014). *Predictive modeling of uniform differential item functioning preservation likelihoods after applying disclosure avoidance techniques to protect privacy* [PhD thesis]. Virginia Polytechnic Institute; State University.

Liu, Y., Zhang, L., Ge, N., & Li, G. (2020). A systematic literature review on federated learning: From a model quality perspective. *arXiv Preprint arXiv:2012.01973*. https://doi.org/10.48550/arXiv.2012.01973

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017). Communication-efficient learning of deep networks from decentralized data. *Artificial Intelligence and Statistics*, 1273–1282.

Rizopoulos, D. (2007). Ltm: An r package for latent variable modeling and item response analysis. *Journal of Statistical Software*, *17*, 1–25. https://doi.org/10.18637/jss.v017.i05

Zhou, B., & Ji, F. (2023). Federated psychometrics: A distributed, privacy-preserving, and efficient IRT estimation algorithm. *APHD Research Gala*.

Zhou, B., & Ji, F. (2024). Federated item response theory: A distributed, privacy-preserving, and efficient IRT estimation algorithm. *DPE MED Research Practicum Poster*.

Zhou, B., & Ji, F. (In submission). *Federated item response models*.

DRAFT