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: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

We developed an R package FedIRT, to estimate traditional IRT models, including 2PL and the graded response models with additional privacy, allowing parameter estimation in a distributed manner without compromising estimation accuracy. Numerical experiments demonstrate that Federated IRT estimation achieves comparable statistical performance to mainstream IRT packages in R, with the 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 apply our proposed method in a user-friendly manner.

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, thereby potentially compromising 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.

Mainstream 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 assessments).

We have therefore developed a specialized R package, `FedIRT`, to integrate federated learning with IRT. We have also developed an accompanying Shiny app to recognize real-world challenges and aim to reduce the burden of learning R programming for applying this package. 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 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 2PL, the response by person i for item j is often binary: $X_{ij} \in \{0, 1\}$, and the probability of person i answering item j with discrimination α_j and difficulty β_j correctly:

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

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 2PL and GPCM, often we assume the ability follows a standard normal distribution, thus we can apply 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. Then, the server uses FedSGD to update the item parameters and send them back to clients. By iterations, the model converges and displays the estimates on the interface.

Taking the 2PL model as an example, which has a marginal log-likelihood function l for each school k that can be approximated using Gaussian-Hermite quadrature with q (by default, $q = 21$) equally-spaced levels, and let $V(n)$ to be the ability value of level n, and $A(n)$ is the weight of 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). It is more robust when there are outliers in input data.

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

Comparison with existing packages

We showcase that our package could generate the same result as traditional IRT packages, for example, `mirt` (Chalmers, 2012). Take 2PL as an example, we use a synthesized dataset with 160 students and 10 items. %For traditional packages, the whole dataset is used. For our package, the dataset was separated into two parts, which contain 81 and 79 students.

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.

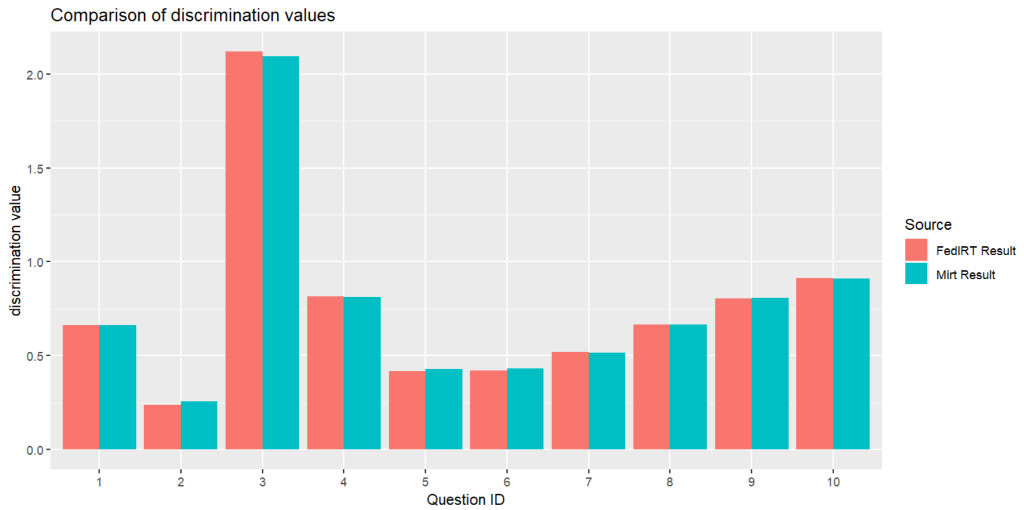

Figure 1: Discrimination parameter estimates comparison

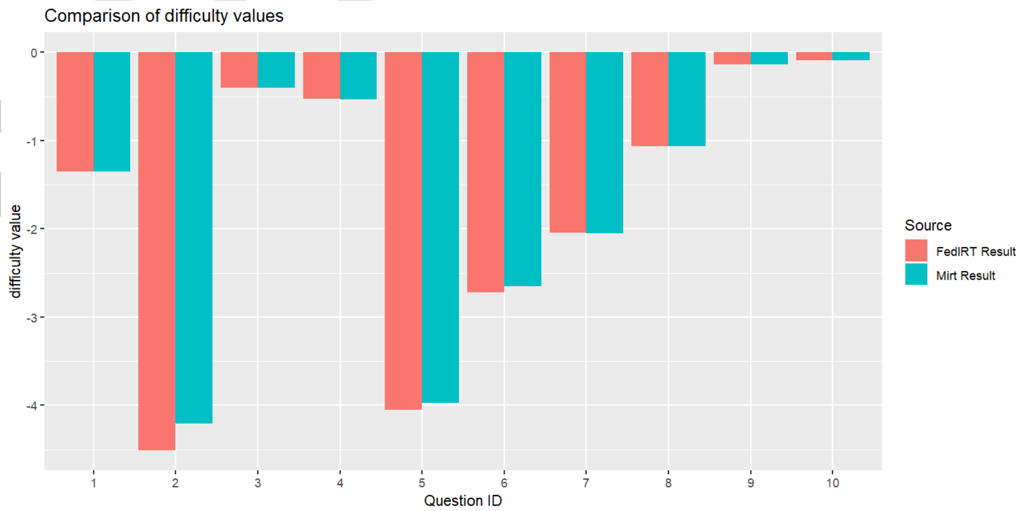

Figure 2: Difficulty parameter estimates comparison

Availability

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

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

Sample of the integrated function

We provide a function `fedirt` in the package, and the detailed usage of the function is shown in the user manual. We demonstrate a sample here.

Suppose we have a dataset called `dataset.csv`, and the head of this dataset is shown below:

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 split the dataset by different sites. The index of each site is indicated in the column `site`.

```r
# split the dataset by sites
data <- read.csv("dataset.csv", header = TRUE)
data_list <- split(data[, -1], data$site)
```

Then, we change every sites' data into a list of matrices in `R`.

```r
# change the list into matrices
inputdata <- lapply(data_list, as.matrix)
```

Then, we call the function `FedIRT::fedirt()` to get the result. It returns a list of item discriminations, item difficulties, and each sites' effect and each students' abilities. Call `summary()` to see the returned list.

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

At last, print the results or use the parameters for further analysis.

```r
print(result$a)
print(result$b)
```

In summary, we read a dataset and split it into different sites. Note that the dataset should indicate different sites. Then call the function `fedirt` with corresponding arguments. At last, print the results in need or use the part of results needed.

Sample of the Shiny App

To provide wider access for practitioners, 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()`) with the interface shown below:

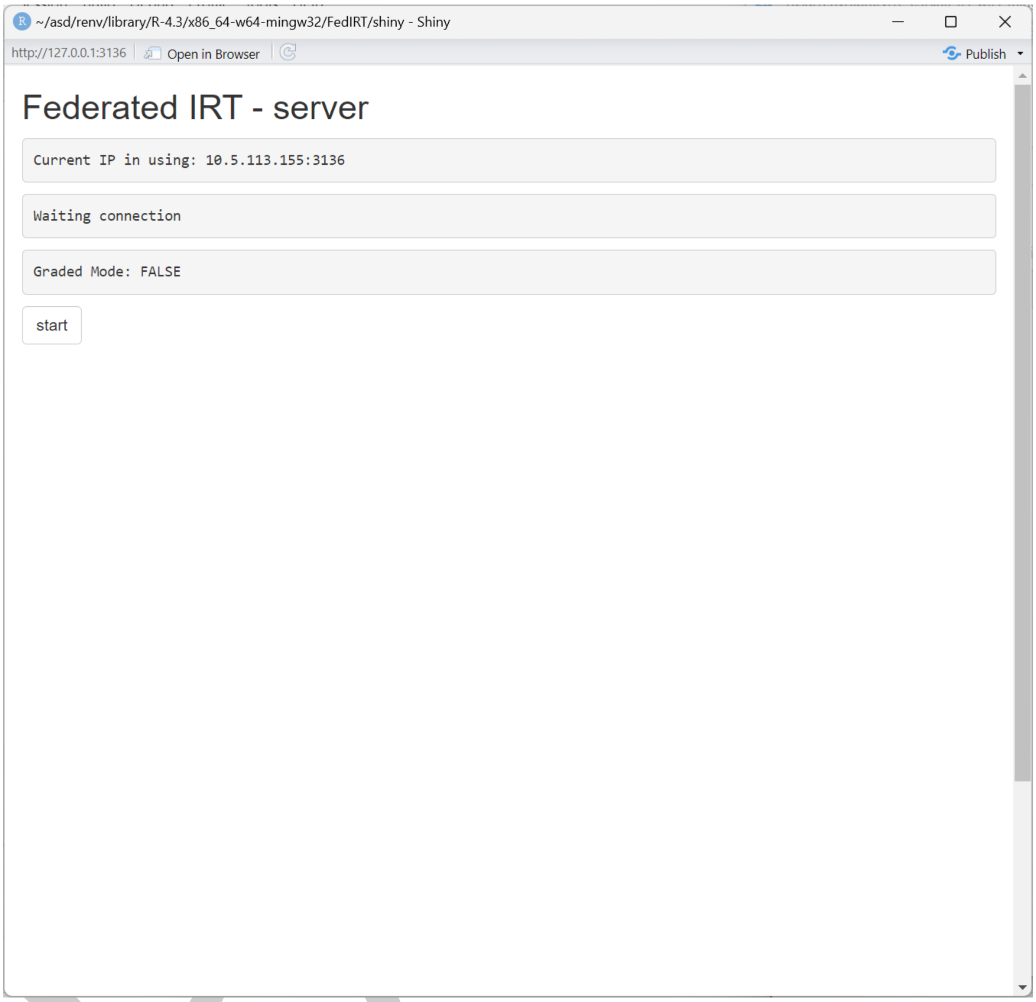

Figure 3: The initial server interface.

Then, the client-end Shiny app can be initialized (`runclient()`).

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

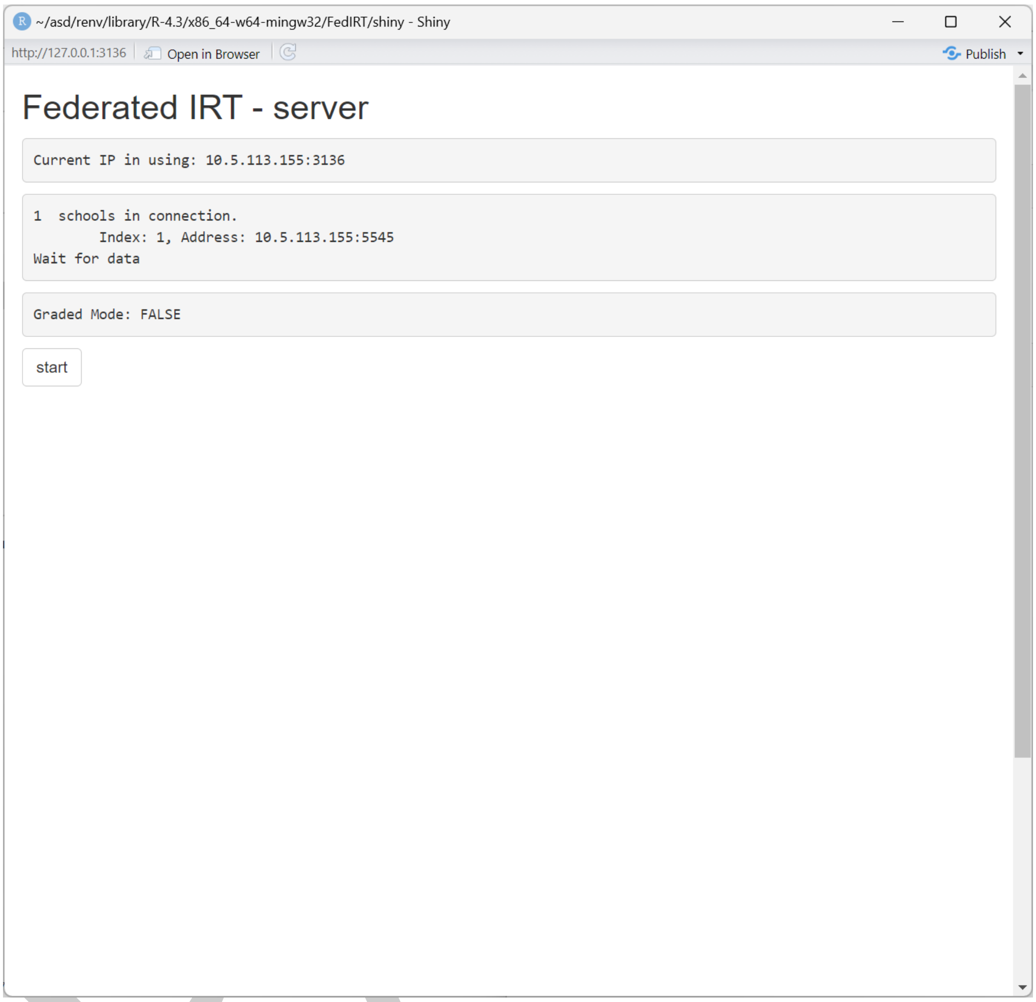

Figure 4: Server interface when one school is connected.

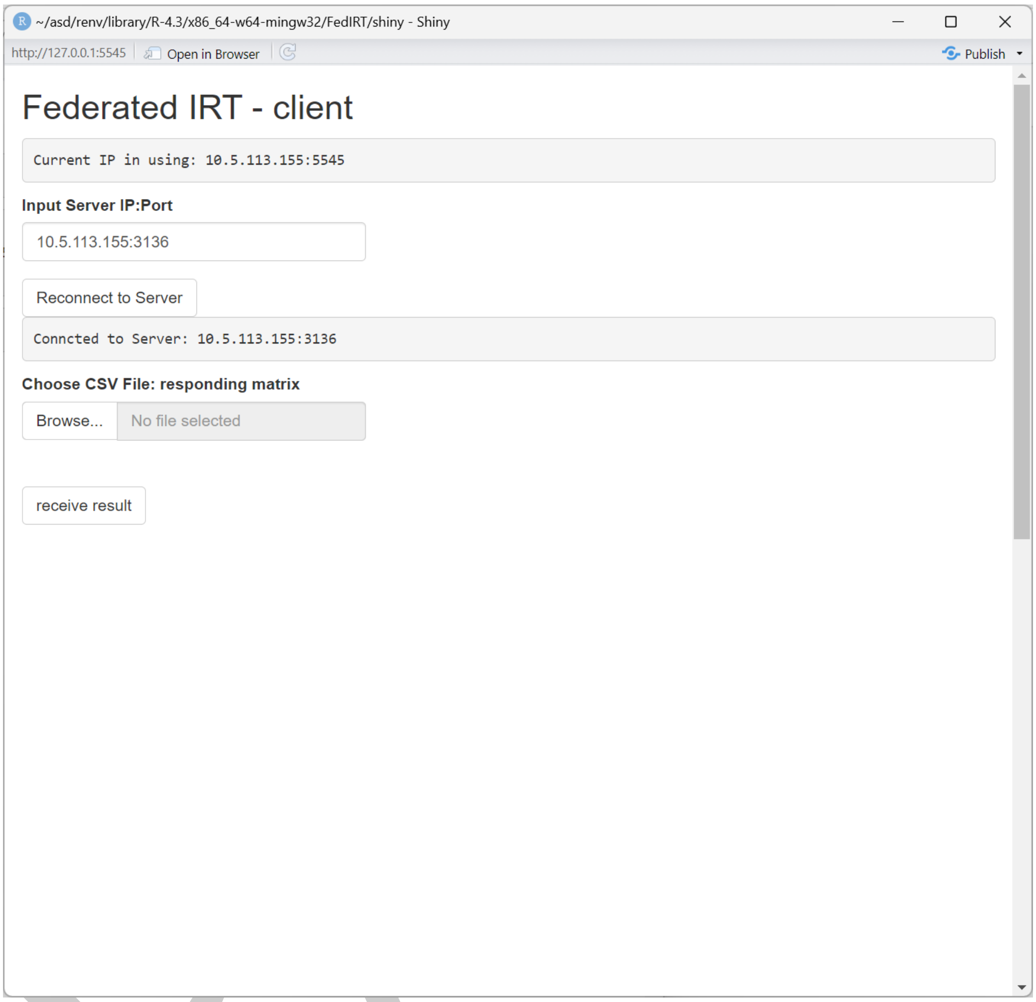

Figure 5: Client interface when connected to server.

Then, the client should choose a file to upload to the local Shiny app to do local calculations, without sending it to the server. The file should be a csv file, with either binary or graded response, and all clients should share the same number of items, and the same maximum score in each item (if the answers are polytomous), otherwise, there will be an error message suggesting to check the datasets of all clients.

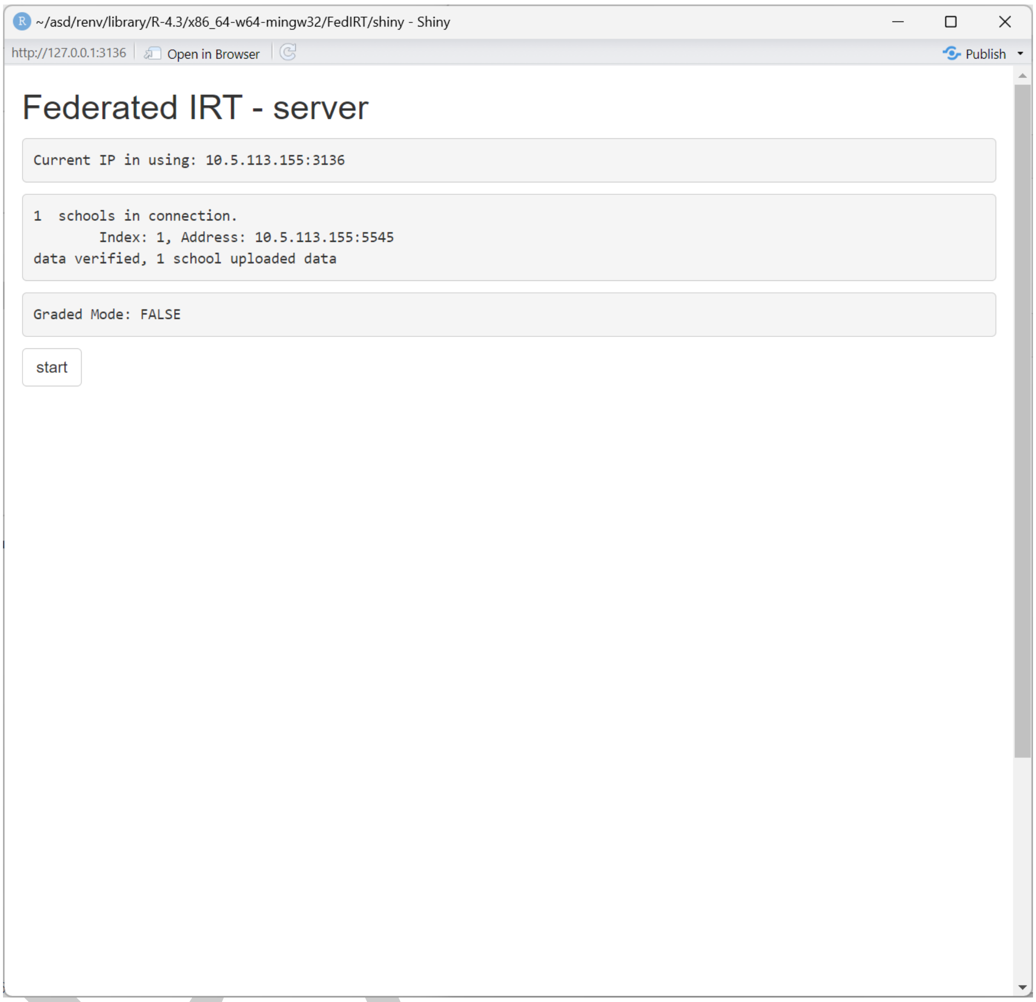

Figure 6: Server interface when one school uploaded dataset.

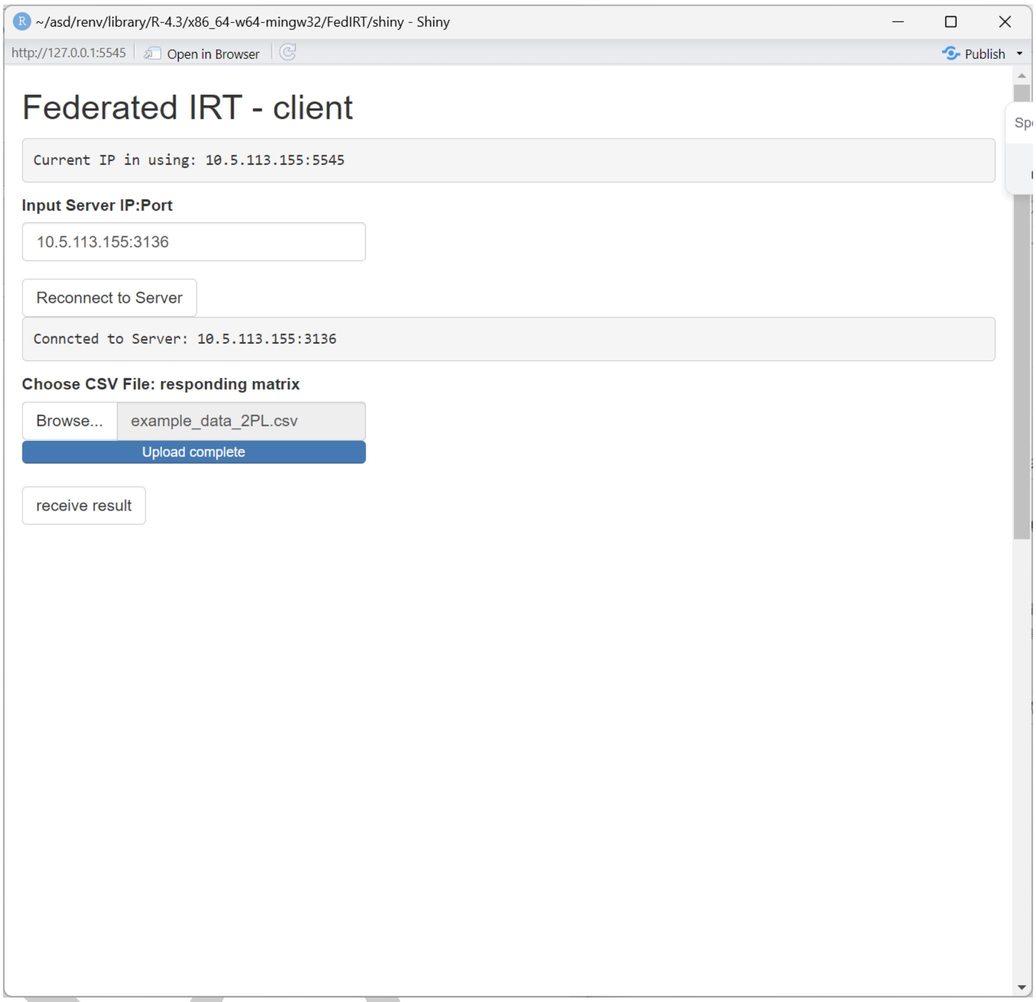

Figure 7: Client interface when a dataset is uploaded successfully.

After all the clients upload their data, the server should click "start" to begin the federated estimates process and after the model converges, the client should click "receive result". The server will display all item parameters and the client will display all item parameters and individual ability estimates.

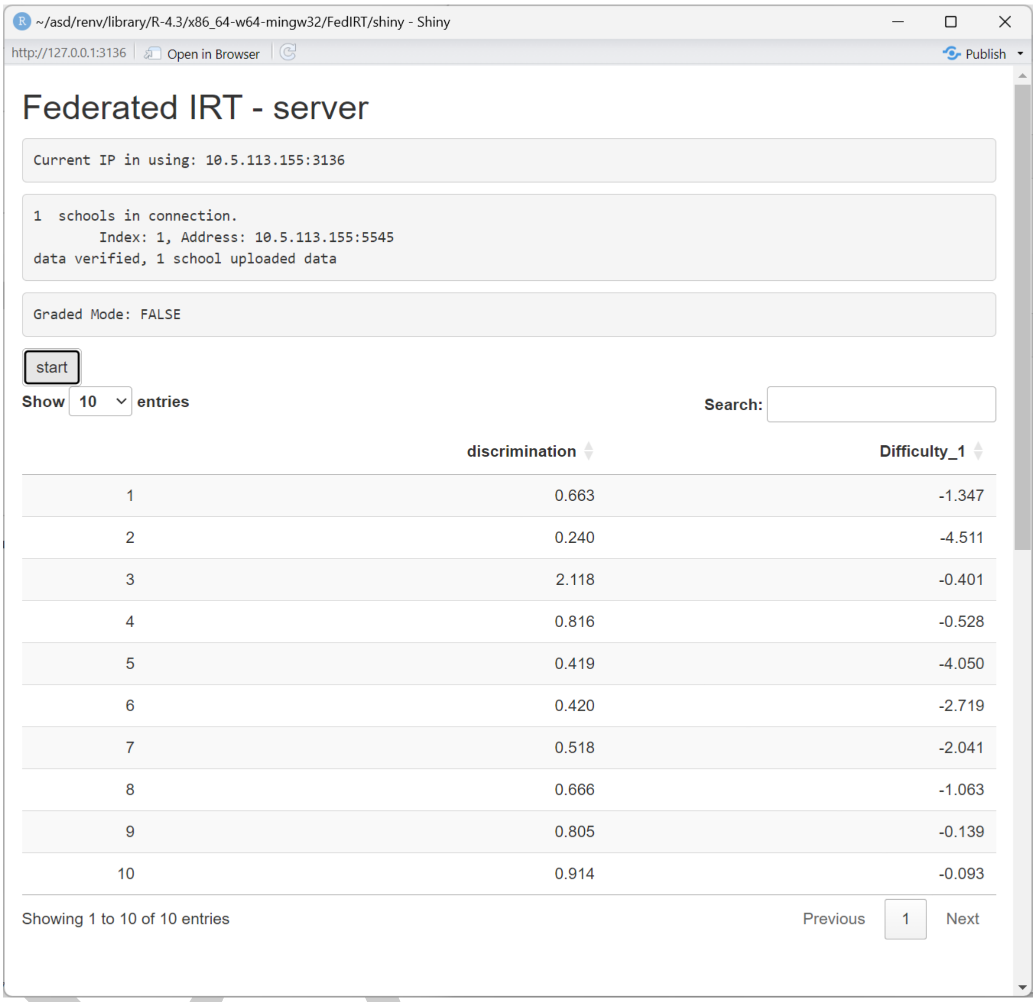

Figure 8: Server interface when estimation is completed.

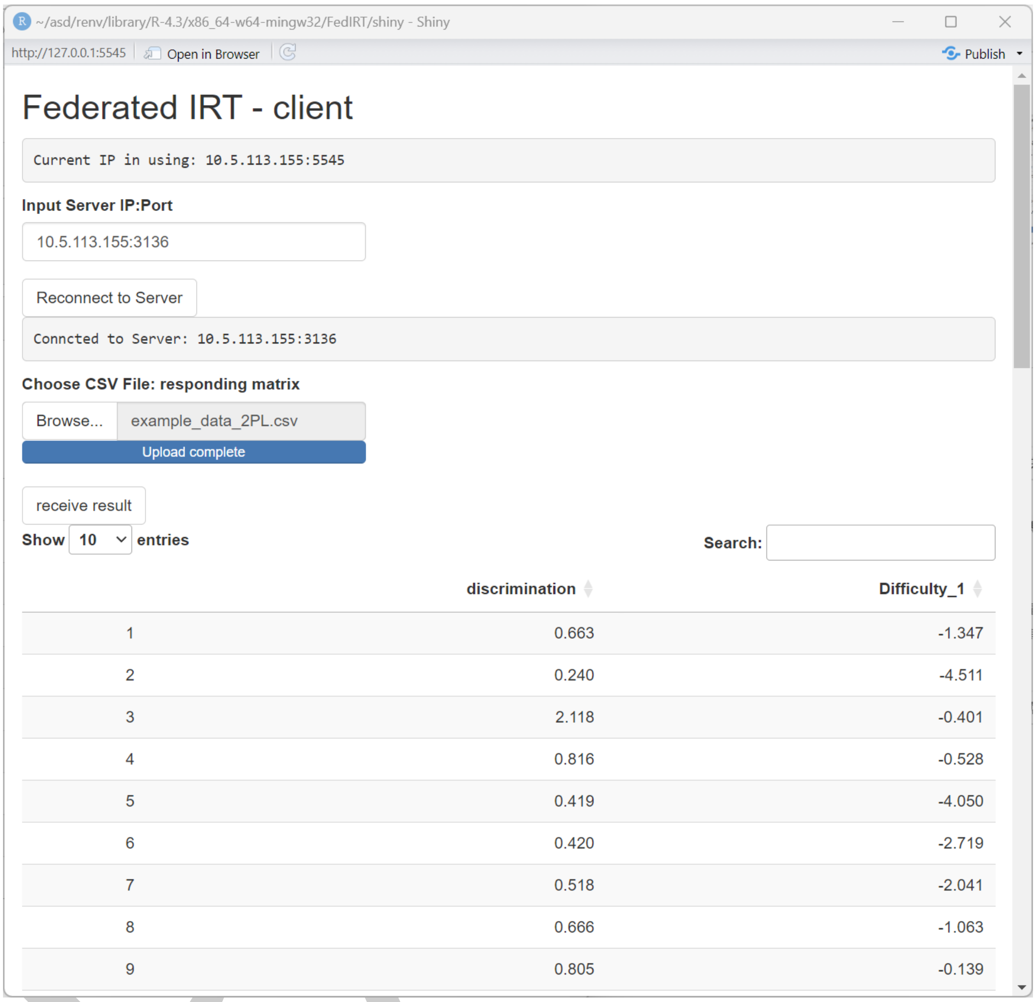

Figure 9: Client interface when the results received.

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

The Journal of Open Source Software

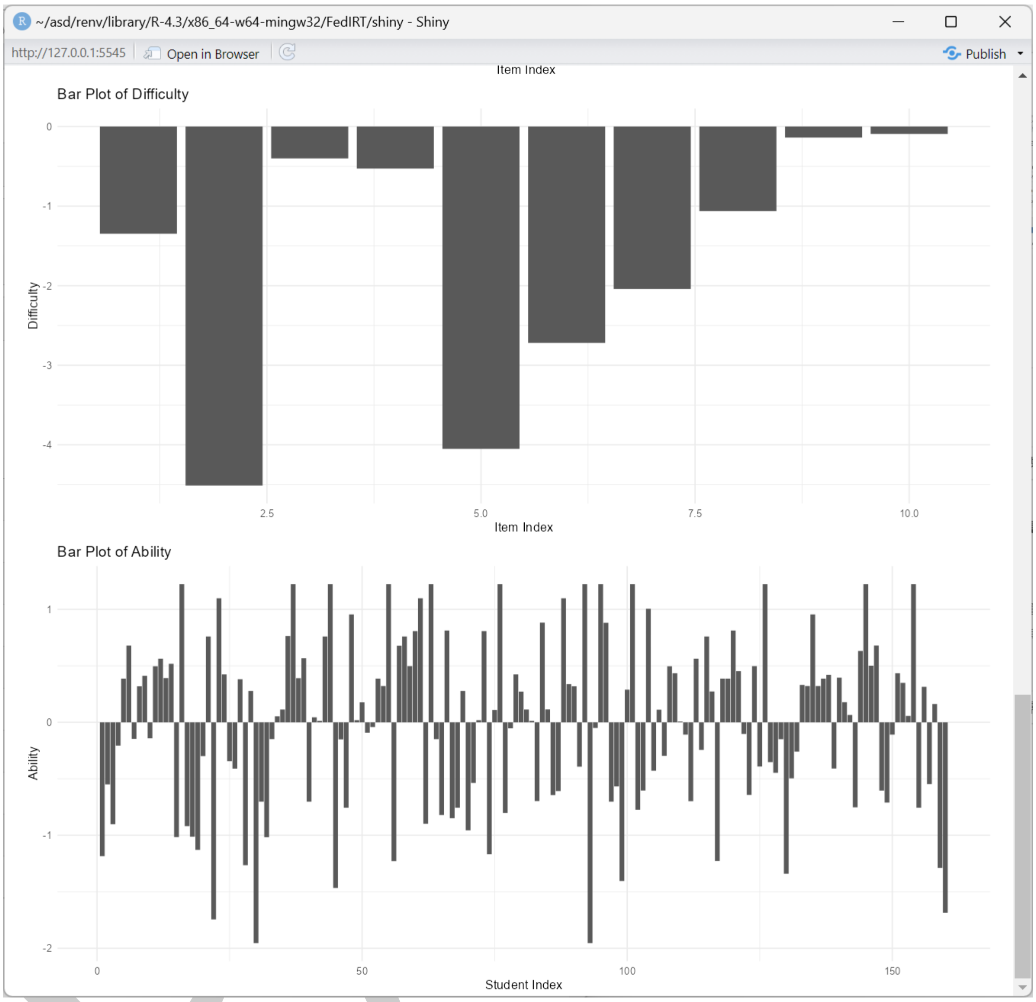

Figure 10: Client interface for displaying results.

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*.