Package: FedIRT (via r-universe)

October 29, 2024

Type Package

Title Federated Item Response Theory Models

Version 1.1.0

Description Integrate Item Response Theory (IRT) and Federated Learning to estimate traditional IRT models, including the 2-Parameter Logistic (2PL) and the Graded Response Models, with enhanced privacy. It allows for the estimation in a distributed manner without compromising accuracy. A user-friendly 'shiny' application is included.

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports purrr, pracma, shiny, httr, callr, DT, ggplot2, shinyjs,

Encoding UTF-8

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

NeedsCompilation no

Author Biying Zhou [cre], Feng Ji [aut]

Maintainer Biying Zhou <zby.zhou@mail.utoronto.ca>

Date/Publication 2024-09-28 19:10:02 UTC

Repository https://biyingzhou.r-universe.dev

RemoteUrl https://github.com/cran/FedIRT

RemoteRef HEAD

RemoteSha 38abff33cc99a93c33d799d074aefe4474c4ed91

Contents

example_data_2PL . . . 2
example_data_2PL_1 . . . 3
example_data_2PL_2 . . . 3
example_data_graded . . . 4
example_data_graded_and_binary . . . 5
fedirt . . . 5
fedirt_2PL . . . 6
fedirt_file . . . 8
fedirt_gpcm . . . 9
g_logL . . . 10
g_logL_entry . . . 11
g_logL_gpcm . . . 12
logL . . . 13
logL_entry . . . 14
logL_gpcm . . . 14
mem . . . 15
personfit . . . 16
personscore . . . 16
runclient . . . 17
runserver . . . 18
SE . . . 19

Index **20**

`example_data_2PL` *Binary Response Dataset for Federated 2PL Model*

Description

A synthetic dataset composed of responses from 160 students to 10 items, with all responses binary.

Usage

```
example_data_2PL
```

Format

A dataframe with 160 rows and 10 columns. Each row corresponds to a student's set of responses, and each column represents the responses of all students to a particular item.

Details

A response of 1 indicates a correct answer, and 0 represents an incorrect answer.

Source

The data are synthetically generated.

Examples

```
data(example_data_2PL)
```

`example_data_2PL_1` *Binary Response Dataset for Federated 2PL Model*

Description

A synthetic dataset composed of responses from 81 students to 10 items, with all responses binary. It is the first part of "example_data_2PL". It is used to test the correctness of federated 2PL model. A response of 1 indicates a correct answer, and 0 represents an incorrect answer.

Usage

```
example_data_2PL_1
```

Format

A dataframe with 81 rows and 10 columns. Each row corresponds to a student's set of responses, and each column represents the responses of all students to a particular item.

Source

The data are synthetically generated.

Examples

```
data(example_data_2PL_1)
```

`example_data_2PL_2` *Binary Response Dataset for Federated 2PL Model*

Description

A synthetic dataset composed of responses from 79 students to 10 items, with all responses binary. It is the second part of "example_data_2PL". It is used to test the correctness of federated 2PL model. A response of 1 indicates a correct answer, and 0 represents an incorrect answer.

Usage

```
example_data_2PL_2
```

Format

A dataframe with 79 rows and 10 columns. Each row corresponds to a student's set of responses, and each column represents the responses of all students to a particular item.

Source

The data are synthetically generated.

Examples

```
data(example_data_2PL_2)
```

`example_data_graded` *Graded Response Dataset for Federated Graded Model*

Description

A synthetic dataset containing responses of 100 students across 10 items, with all items a graded response (0-3 points). This kind of dataset is typically used for testing and validating federated graded response models.

Usage

```
example_data_graded
```

Format

A data frame with 100 rows and 10 columns. Each row represents the responses of a single student, and each column represents one of the 10 graded response items.

Details

The dataset is particularly suitable for testing federated graded model.

Source

The data have been created for demonstration purposes.

Examples

```
data(example_data_graded)
```

example_data_graded_and_binary *Graded Response Dataset for Federated Graded Model*

Description

A synthetic dataset containing 160 students and 8 items, with the first item graded response (0-3 points) and other items binary response. It could be used in the federated graded model testing.

Usage

```
example_data_graded_and_binary
```

Format

A dataframe with 160 rows and 8 columns. Each row denotes a student's responding status, and each column denotes the responding status for all students in one item.

Source

The data were generated synthetically for illustrative purposes.

Examples

```
data(example_data_graded_and_binary)
```

fedirt *Federated IRT model*

Description

This function combines all types of algorithm of federated IRT models. It inputs a dataset and return the estimated IRT parameters.

Usage

```
fedirt(inputdata, model_name = "2PL", school_effect = FALSE, federated = "Avg")
```

Arguments

inputdata	A list of all responding matrices.
model_name	The name of the model you want to use. Can be "1PL" "2PL" or "graded". "1PL" refers to Rasch Model, "2PL" refers to two-parameter logistic model, "graded" refers to graded model.
school_effect	A bool parameter, TRUE refers to considering the school effect as a fixed effect. Default is FALSE.
federated	The federated learning method. Default is "Avg", meaning using Federated Average. Can also be "Med", meaning Federated Median.

Details

Input is a list of responding matrices from each school, every responding matrix is one site's data.

Value

Corresponding model result as a list.

Examples

```
## Not run:
# turn input data to a list
inputdata = list(as.matrix(example_data_2PL))
# Call fedirt() function, and use 2PL model with school effect as a fixed effect
fedresult = fedirt(inputdata, model_name = "2PL",school_effect = TRUE)

# turn input data to a list
inputdata = list(as.matrix(example_data_2PL_1), as.matrix(example_data_2PL_2))
# Call fedirt() function, and use graded model
fedresult = fedirt(inputdata, model_name = "graded")

## End(Not run)
```

---

fedirt_2PL *Federated 2PL model*

Description

This function implements a federated learning approach to estimate the parameters of the 2PL IRT model. It allows for collaborative estimation across multiple datasets, while maintaining the privacy of each individual data source. The federated 2PL model is particularly useful in contexts where data sharing might be limited due to privacy concerns or logistical constraints.

Usage

```
fedirt_2PL(J, logL_entry, g_logL_entry)
```

Arguments

J	An integer indicating the number of items in the IRT model across all sites. This number should be consistent for all response matrices provided.
logL_entry	A function that calculates the sum of log-likelihoods for the response matrices across all sites. This function is crucial for evaluating the fit of the model at each iteration.
g_logL_entry	A function that computes the aggregated gradient of the log-likelihood across all participating entities.

Details

The algorithm leverages federated learning techniques to estimate shared item parameters and individual ability levels without requiring the raw data to be combined into a single dataset. The estimation procedure is composed of several steps, including initialization, local computations at each data source, communication of summary statistics to a central server, and global parameter updates. This cycle is repeated until convergence criteria are met and the global parameters stabilize.

Regarding the input parameters, 'J' is the number of items across all sites, which should be consistent and known in advance. The 'logL_entry' parameter should be a function that computes the log-likelihood of the observed responses given the current model parameters. Likewise, 'g_logL_entry' is expected to be a function that computes the gradient of the log-likelihood with respect to the model parameters to inform the optimization process during parameter estimation.

Value

A list containing the following components from the federated 2PL model estimation:

- par: Numeric vector of model's fitted parameters including item discrimination (a) and item difficulty (b) parameters.
- value: The optimization objective function's value at the found solution, typically the log-likelihood.
- counts: Named integer vector with counts of function evaluations and gradient evaluations during optimization.
- convergence: Integer code indicating the optimization's convergence status (0 indicates successful convergence).
- message: Message from optimizer about optimization process, NULL if no message is available.
- loglik: The calculated log-likelihood of the fitted model, identical to the 'value' element when the objective function is log-likelihood.
- a: Numeric vector of estimated item discrimination parameters.
- b: Numeric vector of estimated item difficulty parameters.
- person: List containing person-related estimates with elements:
  - a: Vector of discrimination parameters (same as top-level 'a').
  - b: Vector of difficulty parameters (same as top-level 'b').
  - ability: List of numeric vectors with person abilities per site.

- site: Numeric vector of abilities or locations specific to each site.
- person: List of numeric vectors of person abilities minus site ability.

`fedirt_file` *Federated IRT model*

Description

This function combines all types of algorithm of federated IRT models. It inputs a dataframe and return the estimated IRT parameters.

Usage

```
fedirt_file(
  inputdata,
  model_name = "2PL",
  school_effect = FALSE,
  federated = "Avg",
  colname = "site"
)
```

Arguments

inputdata	A dataframe.
model_name	The name of the model you want to use. Can be "1PL" "2PL" or "graded". "1PL" refers to Rasch Model, "2PL" refers to two-parameter logistic model, "graded" refers to graded model.
school_effect	A bool parameter, TRUE refers to considering the school effect as a fixed effect. Default is FALSE.
federated	The federated learning method. Default is "Avg", meaning using Federated Average. Can also be "Med", meaning Federated Median.
colname	Column name indicating the school.

Details

Input is a dataframe from each school with a column indicating the school name.

Value

Corresponding model result as a list.

Examples

```
## Not run:
data <- read.csv("dataset.csv", header = TRUE)
fedresult <- fedirt_file(data, model_name = "2PL")

## End(Not run)
```

`fedirt_gpcm` *Federated Graded Response Model Estimation Function*

Description

Implements a federated learning approach for the estimation of the graded response model parameters, enabling collaborative parameter estimation across distributed datasets while ensuring individual data source privacy.

Usage

```
fedirt_gpcm(J, M, logL_entry, g_logL_entry)
```

Arguments

`J`	An integer indicating the number of items in the IRT model across all sites. This number should be consistent for all response matrices provided.
`M`	An integer vector indicating the maximum level (number of categories minus one) for each item across all sites, which determines the total number of step difficulties to estimate for the graded response model.
`logL_entry`	A function that calculates the sum of log-likelihoods for the response matrices across all sites. This function is crucial for evaluating the fit of the model at each iteration.
`g_logL_entry`	A function that computes the aggregated gradient of the log-likelihood across all participating entities.

Details

The function adopts a federated learning framework to perform estimation of item step difficulties and individual ability levels in an IRT graded response model without needing to pool the data into one centralized dataset. The estimator follows an iterative optimization procedure consisisting of local computations, information sharing with a central aggregator, and updating of the global parameters.

Value

A list containing the following components from the federated graded model estimation:

- `par`: Numeric vector of model's fitted parameters including item discrimination (a) and item difficulty (b) parameters.
- `value`: The optimization objective function's value at the found solution, typically the log-likelihood.
- `counts`: Named integer vector with counts of function evaluations and gradient evaluations during optimization.
- `convergence`: Integer code indicating the optimization's convergence status (0 indicates successful convergence).

- `message`: Message from optimizer about optimization process, NULL if no message is available.
- `loglik`: The calculated log-likelihood of the fitted model, identical to the 'value' element when the objective function is log-likelihood.
- `a`: Numeric vector of estimated item discrimination parameters.
- `b`: Numeric vector of estimated item difficulty parameters.

References

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176. doi:10.1177/014662169201600206

`g_logL` *Gradient of Log-Likelihood for the federated 2PL Model*

Description

Calculates the gradients of the log-likelihood function with respect to the item discrimination (a) and difficulty (b) parameters for the Two-Parameter Logistic (2PL) Item Response Theory (IRT) model. This computation is vital for optimizing the item parameters via gradient-based optimization algorithms.

Usage

```
g_logL(a, b, data, q = 21, lower_bound = -3, upper_bound = 3)
```

Arguments

`a`	Numeric vector of item discrimination parameters in the 2PL model.
`b`	Numeric vector of item difficulty parameters in the 2PL model.
`data`	The matrix of observed item responses, with individuals in rows and items in columns.
`q`	The number of Gaussian quadrature points for numerical integration (default is 21).
`lower_bound`	The lower bound for Gaussian quadrature integration (default is -3).
`upper_bound`	The upper bound for Gaussian quadrature integration (default is 3).

Details

The function approximates the partial derivatives by utilizing Gaussian quadrature for numerical integration. Memoization techniques are used to cache intermediate results, which is crucial for efficient computation because it avoids redundant calculations. This can significantly speed up iterative algorithms, particularly in the context of large datasets.

The partial gradient for each parameter is:

$$\frac{\partial l_k}{\partial \alpha_j} = \sum_{n=1}^{q} \sum_{i=1}^{N_k} (V_{ik}(n) - \beta_j)[r_{ijnk} - m_{ink} P_j(V_{ik}(n))]$$

$$\frac{\partial l_k}{\partial \beta_j} = (-\alpha_j) \sum_{n=1}^{q} \sum_{i=1}^{N_k} [r_{ijnk} - m_{ink} P_j(V_{ik}(n))]$$

Value

A list containing two elements: the gradient vector with respect to item discrimination parameters ('a') and the gradient vector with respect to item difficulty parameters ('b').

`g_logL_entry` *Aggregated Gradient of Log-Likelihood for Federated Learning*

Description

Calculates the sum of the gradients of the log-likelihood with respect to item discrimination (a) and difficulty (b) parameters across all schools participating in a federated learning process. The function `g_logL_entry` is a critical component in the gradient-based optimization process within `fedirt`.

Usage

```
g_logL_entry(ps)
```

Arguments

`ps`	A numeric vector including the model's current estimates for the item parameters, organized consecutively with discrimination parameters followed by difficulty parameters.

Details

The function aggregates the gradients computed locally at each school. The cumulative gradient is then used in the optimization algorithm to update the model parameters. Each school should implement the function `get_g_logL_from_index` which computes the gradients of log-likelihood locally. This function needs to be aligned with the federated learning framework, typically involving network communication to retrieve the gradient information.

In simplified scenarios, or during initial testing and development, users can substitute the network communication with a direct call to a local `g_logL` function that computes the gradient of log-likelihood.

Value

A matrix where the first half of rows corresponds to the aggregated gradient with respect to item discrimination parameters and the second half corresponds to the aggregated gradient with respect to item difficulty parameters.

`g_logL_gpcm` *Gradient of Log-Likelihood for the federated graded Model*

Description

Calculates the gradients of the log-likelihood function with respect to the item discrimination (a) and difficulty (b) parameters for the graded IRT model. This computation is vital for optimizing the item parameters via gradient-based optimization algorithms.

Usage

```
g_logL_gpcm(a, b, data, q = 21, lower_bound = -3, upper_bound = 3)
```

Arguments

`a`	Numeric vector of item discrimination parameters in the graded model.
`b`	Numeric vector of item difficulty parameters in the graded model.
`data`	The matrix of observed item responses, with individuals in rows and items in columns.
`q`	The number of Gaussian quadrature points for numerical integration (default is 21).
`lower_bound`	The lower bound for Gaussian quadrature integration (default is -3).
`upper_bound`	The upper bound for Gaussian quadrature integration (default is 3).

Details

The function approximates the partial derivatives by utilizing Gaussian quadrature for numerical integration. Memoization techniques are used to cache intermediate results, which is crucial for efficient computation because it avoids redundant calculations. This can significantly speed up iterative algorithms, particularly in the context of large datasets.

Value

A list containing two elements: the gradient vector with respect to item discrimination parameters ('a') and the gradient vector with respect to item difficulty parameters ('b').

logL *Log-Likelihood of the federated 2PL Model*

Description

Computes the log-likelihood of the Two-Parameter Logistic (2PL) IRT model given item parameters and response data. The computation utilizes numerical integration and is optimized through memoization for repeated evaluations.

Usage

```
logL(a, b, data, q = 21, lower_bound = -3, upper_bound = 3)
```

Arguments

a	The vector of item discrimination parameters in the 2PL model.
b	The vector of item difficulty parameters in the 2PL model.
data	The matrix of observed responses, with individuals in rows and items in columns.
q	The number of Gaussian quadrature points to use for numerical integration (default is 21). Gaussian quadrature is a numerical integration technique to approximate the integral of a function, and is particularly useful for accurate and efficient computation.
lower_bound	The lower limit for the Gaussian quadrature integration (default is -3).
upper_bound	The upper limit for the Gaussian quadrature integration (default is 3).

Details

The function performs numerical integration over a set of quadrature points to calculate the probabilities of the observed responses under the 2PL model, considering the item discrimination (a) and difficulty (b) parameters. Memoization is used to cache computed values of the probabilities, logits, and log-likelihoods to avoid redundant calculations and speed up the process.

Value

The computed log-likelihood of the 2PL model as a single numeric value.

logL_entry *Aggregate Log-Likelihood Function for Federated Learning*

Description

Computes the sum of log-likelihoods across multiple schools in a federated learning setting. The function `logL_entry` aggregates contribution of each school's log-likelihood to the overall model. It is designed to be used within the optimization process of `fedirt`.

Usage

```
logL_entry(ps)
```

Arguments

`ps`	A parameter vector consisting of item parameters; it should include both discrimination (a) and difficulty (b) parameters.

Details

In a federated learning context, each school computes its log-likelihood locally. The `logL_entry` function is responsible for aggregating these values. Users are expected to provide an implementation for `getlogL_from_index`, which should include network requests to retrieve log-likelihoods calculated by each school, or for simplified prototyping purposes, could directly use a `logL` function to compute likelihoods locally.

Value

The sum of log-likelihoods as a single numeric value, representing the likelihood of the entire federated dataset under the current model's parameters.

logL_gpcm *Log-Likelihood of the federated graded Model*

Description

Computes the log-likelihood of the graded IRT model given item parameters and response data. The computation utilizes numerical integration and is optimized through memoization for repeated evaluations.

Usage

```
logL_gpcm(a, b, data, q = 21, lower_bound = -3, upper_bound = 3)
```

Arguments

a	The vector of item discrimination parameters in the graded model.
b	The vector of item difficulty parameters in the graded model.
data	The matrix of observed responses, with individuals in rows and items in columns.
q	The number of Gaussian quadrature points to use for numerical integration (default is 21). Gaussian quadrature is a numerical integration technique to approximate the integral of a function, and is particularly useful for accurate and efficient computation.
lower_bound	The lower limit for the Gaussian quadrature integration (default is -3).
upper_bound	The upper limit for the Gaussian quadrature integration (default is 3).

Details

The function performs numerical integration over a set of quadrature points to calculate the probabilities of the observed responses under the graded model, considering the item discrimination (a) and difficulty (b) parameters. Memoization is used to cache computed values of the probabilities, logits, and log-likelihoods to avoid redundant calculations and speed up the process.

Value

The computed log-likelihood of the graded model as a single numeric value.

---

mem *Memoization Function for Speed Optimization*

Description

A simple memoization function that stores the results of expensive function calls and reuses those results when the same inputs occur again. This technique greatly speeds up the computation of `fedirt` function by caching previously computed values.

Usage

```
mem(f)
```

Arguments

f	Function to be memd.

Value

Returns a memd version of function `f` that will cache its previously computed results for faster subsequent evaluations, especially beneficial when applied to `fedirt`.

Examples

```
# To mem a function, simply wrap it with `mem`:
mem(function(a,b){return(a+b)})
```

personfit *Federated IRT person fit*

Description

personfit calculates the Zh values, infit and outfit statistics. The returned object is a list.

Usage

```
personfit(fedresult)
```

Arguments

fedresult	fedirt result object

Details

Input is the object of fedirt class.

Value

a list of person fit in each school.

Examples

```
# turn input data to a list
inputdata = list(as.matrix(example_data_2PL))
# Call fedirt() function, and use 2PL model
fedresult = fedirt(inputdata, model_name = "2PL")
personfitResult = personfit(fedresult)
```

personscore *Federated IRT person score*

Description

This function calculates persons' ability.

Usage

```
personscore(fedresult)
```

Arguments

fedresult	fedirt result object

Details

Input is the object of fedirt class.

Value

a list of person score in each school.

Examples

```
# turn input data to a list
inputdata = list(as.matrix(example_data_2PL))
# Call fedirt() function, and use 2PL model
fedresult = fedirt(inputdata, model_name = "2PL")
personscoreResult = personscore(fedresult)
```

---

runclient *Client for Federated IRT Model Estimation*

Description

Initializes a client interface for the federated learning estimation of Item Response Theory (IRT) model parameters, connecting to a central server to participate in collaborative parameter estimation. It is essential to start the server prior to the client to ensure the client can establish a successful connection, otherwise an error will occur.

Usage

```
runclient()
```

Details

The client interface, built with Shiny, provides an interactive platform that enables users to upload response matrix data in CSV format, connect to a central server, and receive the estimation results once the computation is complete. The client sends computed local statistics or partial results to the server, which then aggregates information from all clients to update the global IRT model parameters. Users can input the server's IP address and port number, reconnect if needed, and visualize the computed item and ability parameters through plots and tables displayed in the interface.

The client is capable of uploading data, processing it locally to compute log-likelihood or gradient information, and sending these details to the server based on HTTP POST requests. The client also includes functionality to handle responses from the server, either to signal the status of the connection or to receive and display results of the federated estimation process. Through this interactive client-server architecture, the federated IRT model estimation becomes a seamless process, allowing participants to contribute computational resources while preserving data privacy within their local environments.

Additional client functions include local IP retrieval for network communication, server connection initiation, response data processing, and result visualization. Interactive components built in Shiny enable a smooth user experience and real-time updates, making the client an integral part of the federated IRT model estimation framework.

Value

shows the discriminations and difficulties of each item and plot them. Also displays each students' abilities.

Note

This shiny app should be used together with server version. Run server before run client, and get the correct address from server interface to initialize the estimating process.

`runserver` *Server for Federated IRT Model Estimation*

Description

Launches a server that handles federated learning across multiple schools or institutions for the estimation of Item Response Theory (IRT) model parameters. This server facilitates communication between the central aggregator and distributed data sources, coordinating the data sharing process while maintaining privacy.

Usage

```
runserver()
```

Details

The server establishes a federated learning environment where each participating entity (school) computes parts of the model locally. The server then collects summary statistics from each entity and uses them to update the global model parameters. It features a user interface for initiating the estimation process and for displaying the results of the federated learning procedure. The user interface provides real-time information about the connected schools, data consistency checks, and the mode of the IRT model being estimated (binary or graded).

Function 'updateM' checks for consistency in the number of maximum item levels across all schools, setting a flag to indicate whether a binary or graded model should be used. Function 'check_J' ensures that all schools have a consistent number of items in their datasets. The 'ui' function serves as the user interface for the server, while 'getLocalIP' retrieves the server's IP address for connections. Finally, the 'server' function contains the logic for receiving data from schools, triggering the estimation process, and sending the results back to participating schools.

Overall, the 'runserver' function orchestrates the federated IRT model estimation process by combining local computations from schools, managing data traffic, executing the appropriate estimation function, and providing users with an interactive web interface.

The web interface is built using Shiny, allowing users to check connection statuses, start the estimation process, and view results. It supports both GET and POST HTTP methods for handling data exchange with clients. The server is designed to be flexible and can be adapted for various federated learning scenarios in the education sector.

Value

No return value, called for side effects (initiates interactive Shiny server session) and display estimates on the interface.

Note

This shiny app should be used together with client version.

SE *Federated IRT SE*

Description

Calculates Standard Error(SE) for FedIRT models.

Usage

```
SE(fedresult)
```

Arguments

`fedresult`	fedirt result object

Details

Input is the object of fedirt class.

Value

An array of standard errors for all parameters.

Examples

```
# turn input data to a list
inputdata = list(as.matrix(example_data_2PL))
# Call fedirt() function, and use 2PL model
fedresult = fedirt(inputdata, model_name = "2PL")
# get SE result
SEresult = SE(fedresult)
```

Index

∗ **datasets**
example_data_2PL, 2
example_data_2PL_1, 3
example_data_2PL_2, 3
example_data_graded, 4
example_data_graded_and_binary, 5

example_data_2PL, 2
example_data_2PL_1, 3
example_data_2PL_2, 3
example_data_graded, 4
example_data_graded_and_binary, 5

fedirt, 5
fedirt_2PL, 6
fedirt_file, 8
fedirt_gpcm, 9

g_logL, 10
g_logL_entry, 11
g_logL_gpcm, 12

logL, 13
logL_entry, 14
logL_gpcm, 14

mem, 15

personfit, 16
personscore, 16

runclient, 17
runserver, 18

SE, 19