fcirt: Forced Choice in Item Response Theory

Naidan Tu

Contents

Overview

The fcirt package was developed to estimate forced choice models using Bayesian method. Specifically, the Multi-Unidimensional Pairwise Preference (MUPP) model is estimated using the R package rstan that utilizes the Hamiltonian Monte Carlo sampling algorithm. Below are some important features of the fcirt package:

1. Item and test information calculated using either quadrature points or estimated person parameters can be obtained using the function information( ).
2. Missing data are automatically dealt with in a way similar to how full information maximum likelihood handles missing data.
3. Dimensions are allowed to correlate and the correlations are estimated.
4. Statements are allowed to appear multiple times in different items by specifying the required pairmap argument in the function fcirt().

5. Four functions (i.e., fcirt( ), extract( ), information( ), and bayesplot( )) are provided for model estimation, results extraction, item and test information computation, and Bayesian diagnostic plottings, respectively.

Tutorial

Step 1: Input data

A randomly generated dataset is used as an example in this tutorial. The first input (**fcirt.Data**) is a dataset including responses from 5 respondents answering a 4 forced choice items (pairs) measuring 2 traits. If the first statement is preferred, the data should be coded as 1, otherwise it should be coded as 2. Note that data is stored in a wide format.

ID	item 1	item 2	item 3	item 4
1	1	1	2	1
2	2	1	1	2
3	2	1	1	2
4	1	NA	2	NA
5	1	1	1	2

The second input (**pairmap**) is a two-column matrix: the first column is the statement number for statement s; the second column is the statement number for statement t.

item	statement 1	statement 2
1	1	2
2	3	4
3	5	6
4	7	8

The next part of the input is a column vector mapping each statement to each trait. For example, c(1, 1, 1, 2, 2, 2) means that the first 3 statements measure trait 1 and the last 3 statements measure trait 2.

row	statement 1	statement 2	statement 3	statement 4	statement 5	statement 6	statement 7	statement 8
1	1	2	1	2	1	2	2	1

The last part of the input is a three-column matrix containing initial values for the three statement parameters (alpha, delta, tau) respectively. If using the direct MUPP estimation approach, 1 and -1 for alphas and taus are recommended and -1 or 1 for deltas are recommended depending on the signs of the statements. If using the two-step estimation approach, pre-estimated statement parameters are used as the initial values. The R package **bmggum** (Tu et al., 2021) can be used to estimate statement parameters for the two-step approach.

Statement	Alpha	Delta	Tau
1	1	0	-1
2	1	1	-1
3	1	1	-1
4	1	0	-1
5	1	1	-1
6	1	1	-1
7	1	0	-1
8	1	0	-1

Step 2: Estimate using the function fcirt()

```
# Fit the MUPP model
#>mod <- fcirt(fcirt.Data=fcirt.Data, pairmap=pairmap, ind=ind, ParInits=ParInits, iter=100)
#>mod
```

The function fcirt() implements full Bayesian estimation of MUPP using rstan. The returned object stores information including the (1)stanfit object, (2)estimated statement parameters, (3)estimated person parameters, (4)estimated correlations among dimensions, (5)response data, (6)the input initial values, (7)the input pairmap, and (8)the input row vector mapping each item to each trait. Below are a list of other arguments it contains, the default of which can be manually replaced:

- **model**. Models fitted. They can be "MUPP". The default is MUPP (Multi-Unidimensional Pairwise Preference) model.
- **iter**. The number of iterations. The default is 3000. See documentation for rstan for more details.
- **chains**. The number of chains. The default value is 3. See documentation for rstan for more details.

- **warmup**. The number of warmups to discard. The default value is the first half of the iterations. See documentation for rstan for more details.
- **adapt_delta**. Target average proposal acceptance probability during Stan's adaptation period. The default value is 0.90. See documentation for rstan for more details.
- **max_treedepth**. Cap on the depth of the trees evaluated during each iteration. The default value is 15. See documentation for rstan for more details.
- **init**. Initial values for estimated parameters. The default is random initial values. See documentation for rstan for more details.
- **thin**. Thinning. The default value is 1. See documentation for rstan for more details.
- **core**. The number of computer cores used for parallel computing. The default value is 2. Users can use the function **detectCores()**in the package **parallel** to detect the number of cores of their pc/laptop. Usually, users just need to set this number equal to the number of **chains**. In the case of many chains, we recommend users to leave at least one core unoccupied to avoid R crash.
- **ma**. Mean of the prior distribution for alphas, which follows a lognormal distribution. The default value is 0.
- **va**. Standard deviation of the prior distribution for alphas. The default value is 0.5.
- **md**. Mean of the prior distribution for neutral deltas, which follows a normal distribution. The default value is 0.
- **vd**. Standard deviation of the prior distribution for deltas. The default value is 1.
- **mt**. Means of the prior distributions for taus, which follows a normal distribution. The default values is 0.
- **vt**. Standard deviation of the prior distribution for taus. The default value is 2.

Step 3: Extract the estimated results using the function extract()

```
# Extract theta estimates
#>theta <- extract(x=mod, pars='theta')
# Turn theta estimates into p*trait matrix where p equals sample size and trait equals the number of la
#>theta <- theta[,1]
# nrow=trait
#>theta <- matrix(theta, nrow=2)
#>theta <- t(theta)
# theta estimates in p*trait matrix format
#>theta


# Extract tau estimates
#>tau <- extract(x=mod, pars='tau')
#>tau <- tau[,1]
#>tau
```

The function extract() extracts fcirt estimation results.

- **pars**. Names of extracted parameters. They can be "theta" (person trait estimates), "alpha" (statement discrimination parameters), "delta" (statement location parameters), "tau" (statement threshold parameters), data" (fcirt.Data), "fit" (the stanfit object), "dimension" (the input column vector mapping each statement to each trait), "cor" (correlations among dimensions), "pairmap" (A two-column data matrix: the first column is the statement number for statement s; the second column is the statement number for statement t), and "ParInits" (A three-column matrix containing initial values for the three statement parameters).

Step 4: Plotting using the function bayesplot()

```
# Obtain density plots for all alphas.
#>bayesplot(x=mod, pars='alpha', plot='density', inc_warmup=FALSE)
```

```
# Obtain the trace plots for all alphas.
#>bayesplot(x=mod, pars='alpha', plot='trace', inc_warmup=FALSE)
```

The function bayesplot() provides plots including density plots, trace plots, and auto-correlation plots to aid model convergence diagnosis. The smoothness of density plots, the stationary status of trace plots, and low degree of auto-correlation in auto-correlation plots all indicate good convergence.

- **pars**. Names of plotted parameters. They can be "theta", "alpha", "delta", "tau", or a subset of parameters (e.g., paste0("alpha[",1:2,"]"), paste0("theta[1,",1:2,"]")).
- **plot**. Types of plots. They can be "density", "trace", or "autocorrelation".
- **inc_warmup**. Whether to include warmup iterations or not when plotting. The default is FALSE.

Step 5: Obtain item and item information using the function information()

```
# Obtain item information for item 1-3.
#>OII <- information(x=mod, approach="direct", information="item", items=1:3)
#>OII
```

```
# Obtain test information.
#>OTI <- information(x=mod, approach="direct", information="test")
#>OTI
```