2PL Demo

Adam Kurth

2024-01-19

Understanding MIRT

```r
# Simulate data with Monte Carlo experiments
# person trait level, e.g., math skill
theta <- rnorm(n = 1000, mean = 0, sd = 1)

# Item parameters (a, b, c) for 4 items
a <- c(0.5, 1, 1.5, 2)
b <- c(0, -1, 1, 0)
c <- c(0.2, 0.15, 0.25, 0.2)

n.persons <- length(theta)
n.items <- length(a)
response.data <- matrix(NA, n.persons, n.items)

for (i in 1:n.persons) {
    for (j in 1:n.items) {
        p <- c[j] + (1 - c[j]) / (1 + exp(-(a[j] * theta[i] - b[j]))) #2PL model
        u <- runif(n = 1, min = 0, max = 1)
        if (u < p) {
            response.data[i, j] <- 1
        } else {
            response.data[i, j] <- 0
        }
    }
}
colnames(response.data) <- c("I1", "I2", "I3", "I4")
```

2PL Model

- a, b, c: item parameters.
- θ, `theta`: personal trait level.
- p: probability of positive response (gets answer right)
- u: random number between 0 and 1, (uniformly distributed)
- if $u < p$ then `response.data = 1`: individual gets answer correct.
- if $u > p$ then `response.data = 0`: individual gets answer wrong.

```r
# Calibrate item parameters using the mirt package, default settings ------
# record time to convergence and estimation accuracy ---------------------
start.time = Sys.time() #start time
mirt.out <- mirt::mirt(data = response.data, model = 1, itemtype = "2PL", storeEMhistory=TRUE) # 2PL mo
```

```
## Iteration: 1, Log-Lik: -2558.199, Max-Change: 0.11112Iteration: 2, Log-Lik: -2553.794, Max-Change: 0
```

```r
end.time = Sys.time() # end time
time.to.conv = end.time - start.time
```

- mirt.out <- mirt::mirt() function fits a maximum likelihood (posterier) factor analysis model to the data (dichotomus/polytomous)

Output of `mirt.out`:

- 'M-step' = indicates that the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm used for optimization in M-step if the EM algorithm.
- 'EM acceleration' = indicates that ramsay's acceleration method is used to speed up the convergence of the EM algorithm.
- 'Number of rectangular quadrature' = the number of quadrature points used in the numerical integration of the likelihood function.
- 'Latent density type' = the distribution of the latent trait (F1 in this case)
- 'log-likelihood' = how well the model fits the data
- 'Estimated parameters' = number of estimated parameters
- 'AIC' = Akaike information criterion (measure of model fit, takes into number of parameters)
- BIC/SABIC = Bayesian information criterion/Sample-size Adjusted BIC (measure of model fit, takes into number of parameters)
- `G2 (3) = 1.1, p = 0.7772` : results of the likelihood ratio test comparing the fitted model against saturated model
- `RMSEA = 0, CFI = NaN, TLI = NaN`: additional measures of model fit, the Root Mean Square Error of Approximation (RMSEA), the Comparative Fit Index (CFI), and the Tucker-Lewis Index (TLI)

```r
# See the estimated item parameters
mirt::coef(mirt.out, simplify = T, IRTparts = T)
```

```
## $items
##       a1     d g u
## I1 0.434 0.454 0 1
## I2 1.213 1.272 0 1
## I3 0.764 0.014 0 1
## I4 1.613 0.753 0 1
##
## $means
## F1
##  0
```

```
##
## $cov
##    F1
## F1  1
```

- `coef()` function extracts the estimated parameters from the mirt analysis
- `simplify=T` simplifies the output if possible (into vector or matrix), otherwise a list
- `IRTparts = T` specifies the output should be a IRT parameter matrix (T = discrimination, F = original parameterization)

OUTPUT:

- `$items`: df with estimated item parameters, row = items, col = parameters (a1 = discrimination, d = difficulty, g = guessing, u = upper asymptote)
- `$means`: vector that contains the estimated mean of the latent trait (F1 in this case with a mean of 0)
- `$cov` = matrix that contains the estimated covariance matrix of the latent trait (F1 in this case with a variance of 1)
- Note: only trait F1 in this case so $cov is a 1x1 matrix.

What are the parameters?

-discrimination parameter (a) and difficulty parameter (b), and guessing parameter (c):

- (`a`) describes how well an item (e.g. question on test) can differentiate between individuals with different levels of the underlying trait that the test is measuring
- (`b`) describes the level of the underlying trait that is required to have a 50% chance of answering the item correctly other parameters:
- (`c`) guessing parameter (measures prob. that a person with a very low level of underlying trait will respond positively/ aka. get the question right)
- (`u`) upper asymptote parameter (measures the maximum probability that a individual will respond positively(get question right), regardless of underlying trait)

– Note: higher the upper asymptote, the easier the item is.

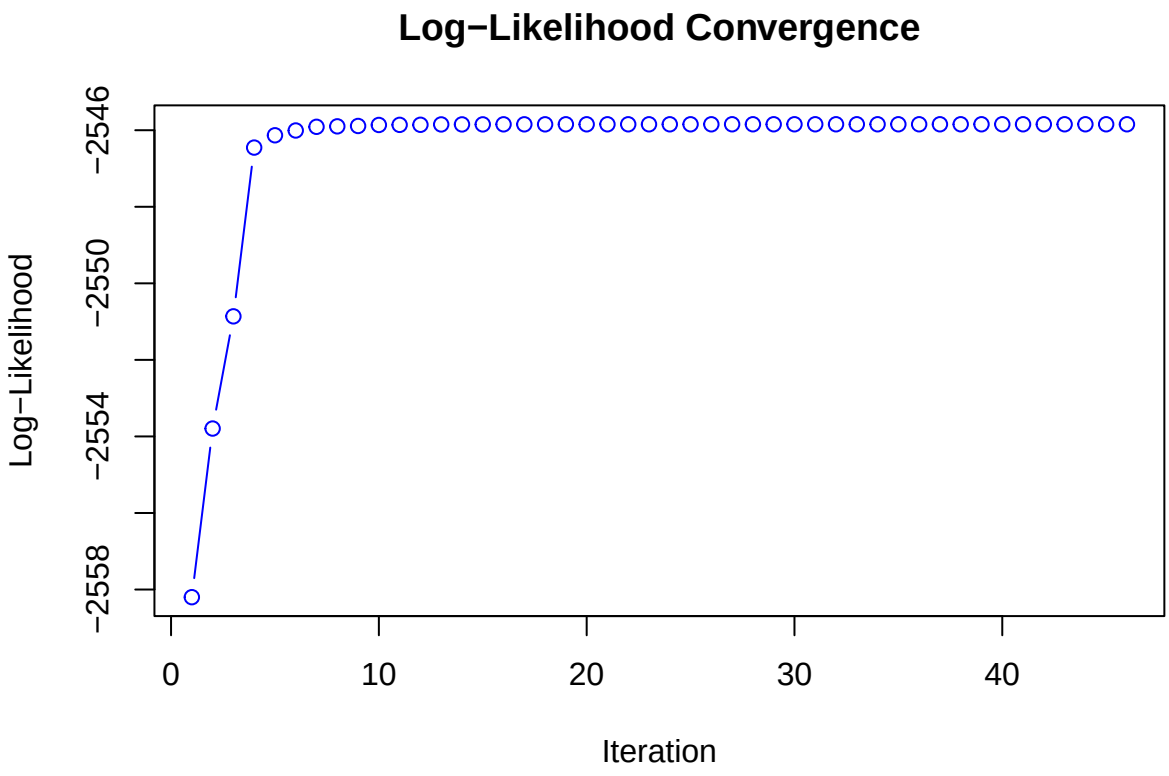
```r
# Calculate estimation accuracy and time to convergence
a.est <- mirt::coef(mirt.out, simplify = T, IRTparts = T)$items[, 1]
RMSE.a <- sqrt(mean((a - a.est)^2))

b.est <- mirt::coef(mirt.out, simplify = T, IRTparts = T)$items[, 2]
RMSE.b <- sqrt(mean((b - b.est)^2))

c.est <- mirt::coef(mirt.out, simplify = T, IRTparts = T)$items[, 3]
RMSE.c <- sqrt(mean((c - c.est)^2))

# Store RMSE values and time to convergence in dataframe
df <- data.frame("a.discrim" = a, "b.difficulty" = b, "c.guessing" = c,
```

```
                "a.est" = a.est, "b.est" = b.est, "c.est" = c.est,
                "RMSE.a" = RMSE.a, "RMSE.b" = RMSE.b, "RMSE.c" = RMSE.c)

df$time.to.conv <- time.to.conv
```

- Plots: – Log-likelihood – Item characteristic curves (ICC) – Test information function (TIF) – Person-item map – Factor loadings – Residual analysis

Log-Likelihood

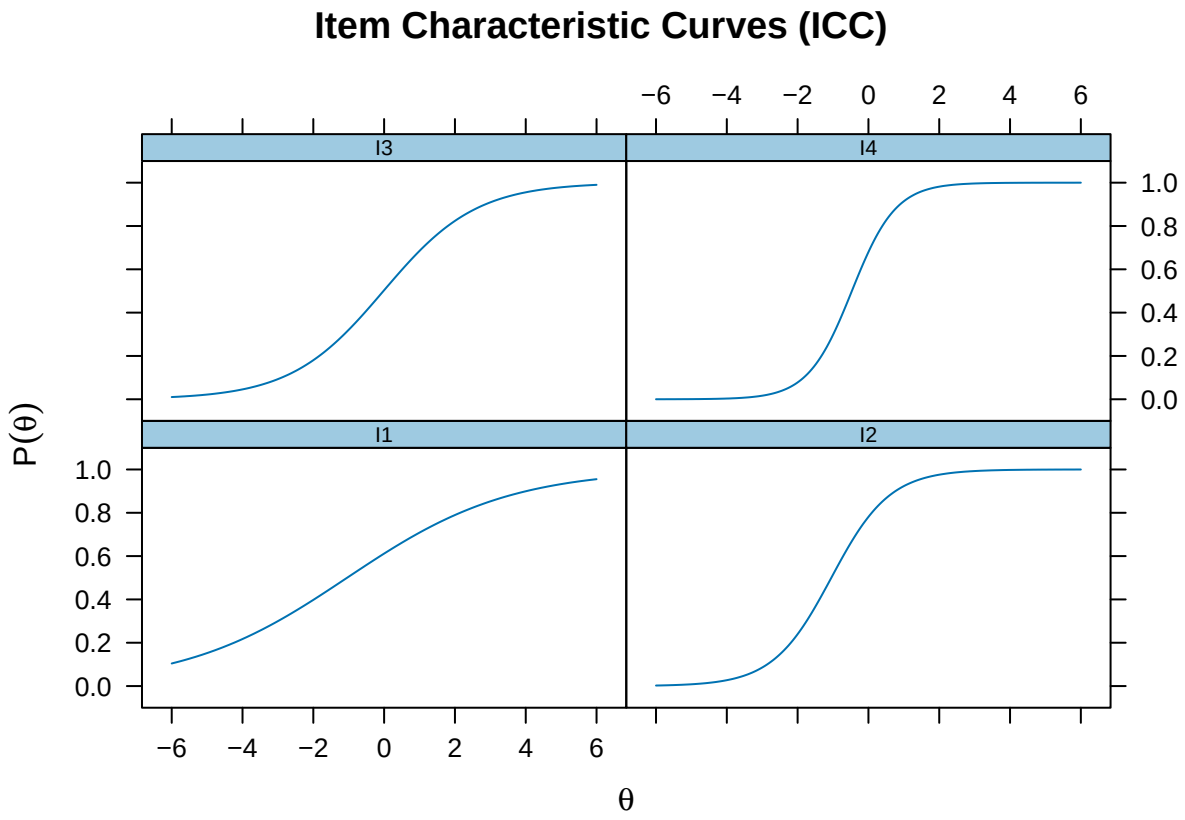
```
ll.history <- extract.mirt(mirt.out, 'LLhistory')

plot(ll.history, type="b", col='blue', ylim = range(ll.history),
    xlab = "Iteration", ylab = "Log-Likelihood", main = "Log-Likelihood Convergence")
```



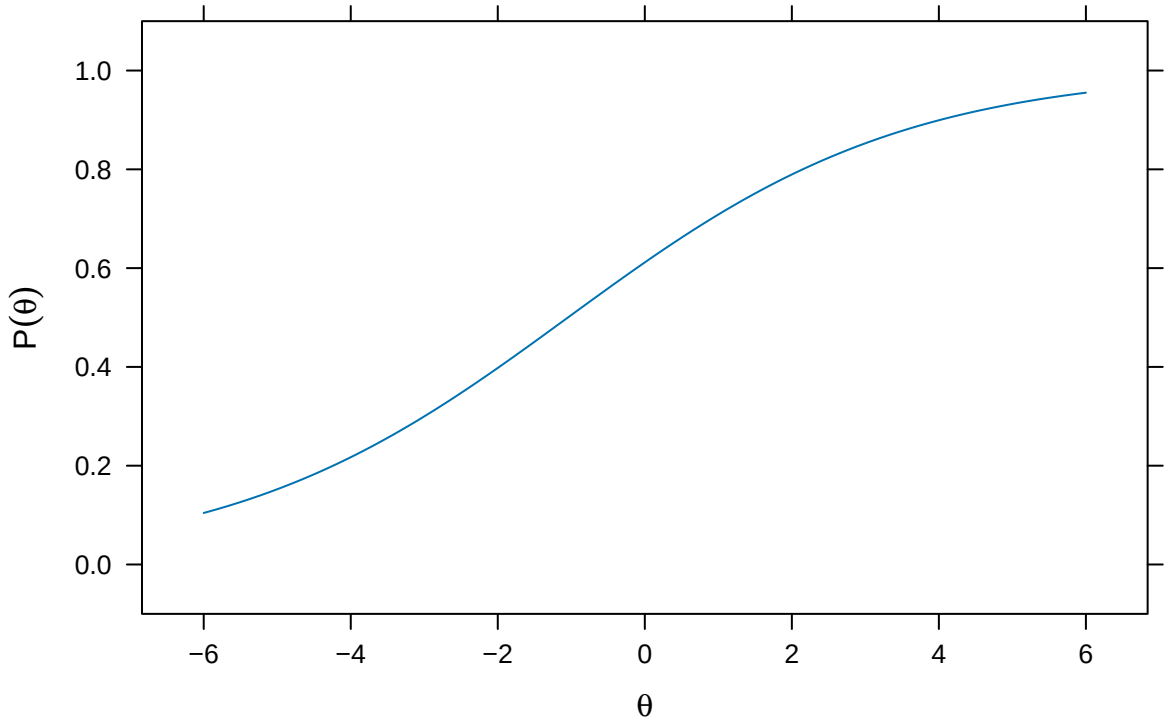
Test Information Function (TIF)

- *Test Information Function (TIF)* provides information about the reliability of a test at different levels of a trait being measured. It is a function of the item parameters and indicates where on the trait scale the test is most informative.

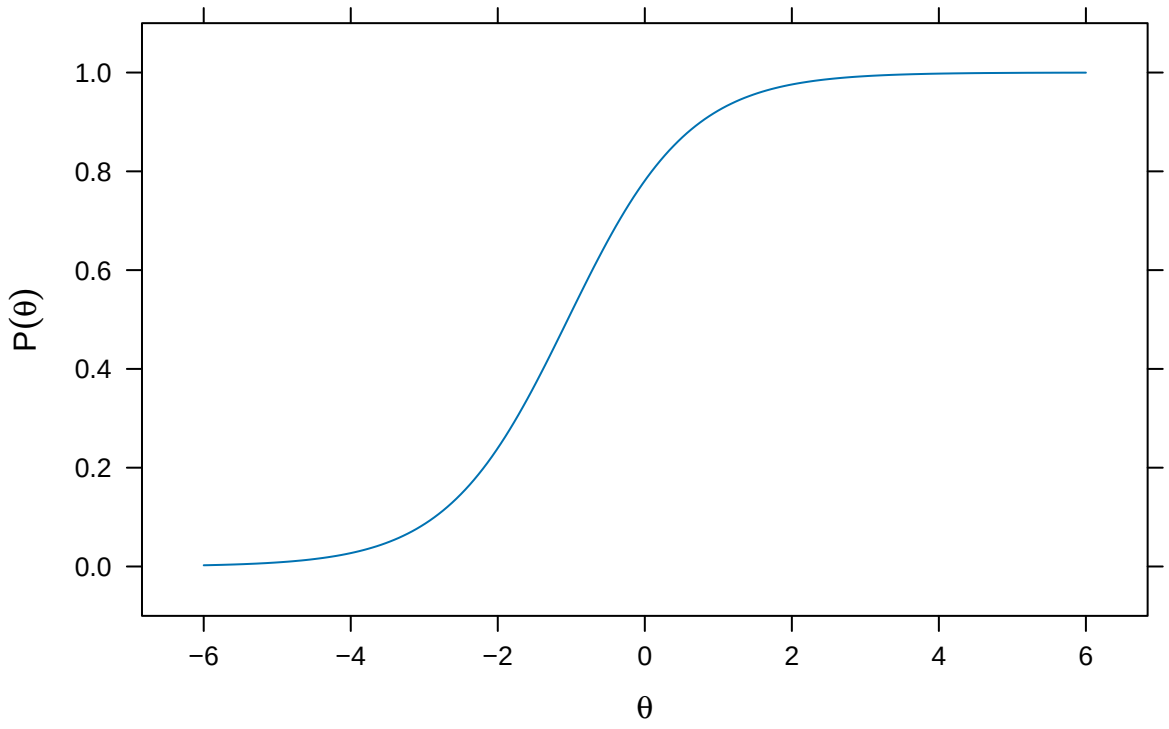
```r
# Plot item characteristic curves (ICC)
plot(mirt.out, type = "trace", auto.key =  list(columns = 4, legend = TRUE),
     main = "Item Characteristic Curves (ICC)")
```



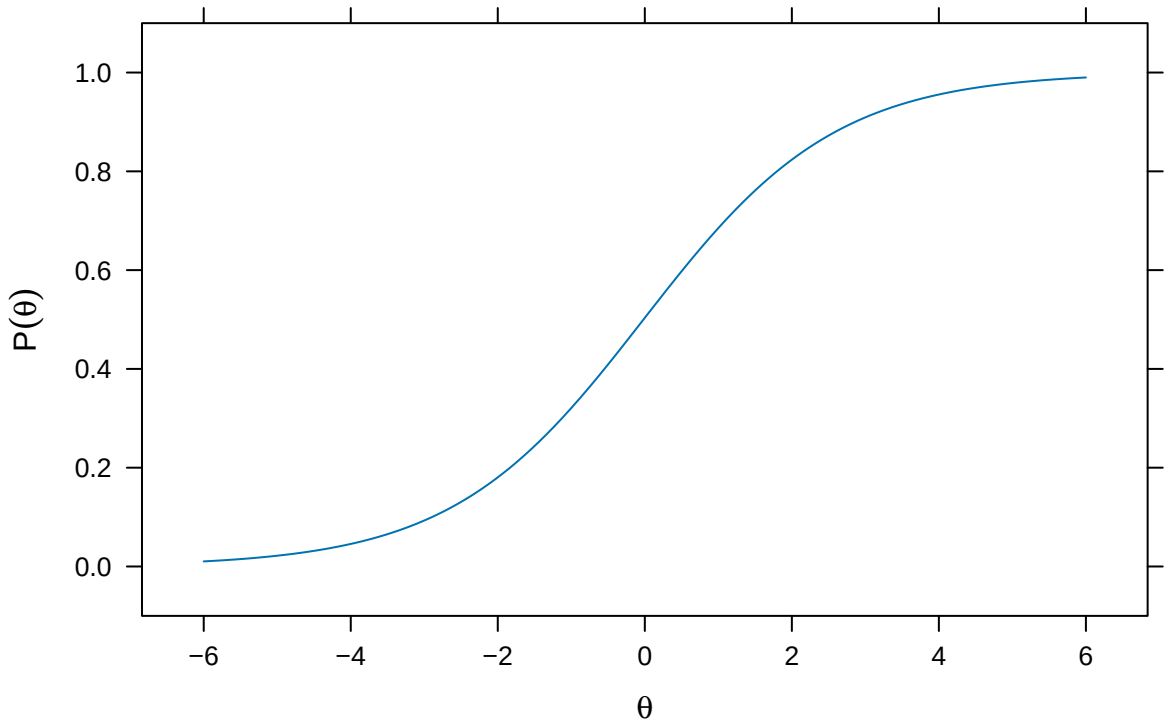

```r
itemplot(mirt.out, item = 1, type = "trace", auto.key =  list(columns = 4, legend = TRUE),
         main = "Item Characteristic Curves (ICC) for Item 1")
```



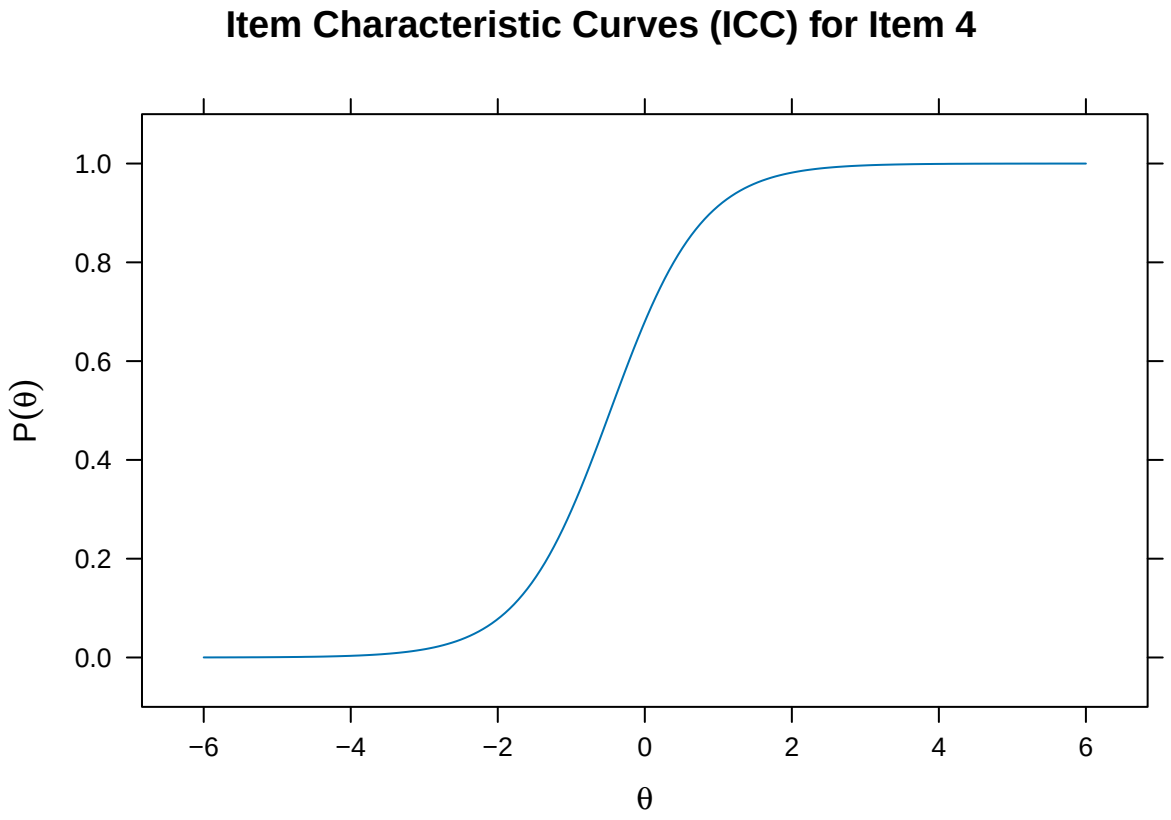

```
itemplot(mirt.out, item = 2, type = "trace", auto.key =  list(columns = 4, legend = TRUE),
         main = "Item Characteristic Curves (ICC) for Item 2")
```




```
itemplot(mirt.out, item = 3, type = "trace", auto.key =  list(columns = 4, legend = TRUE),
         main = "Item Characteristic Curves (ICC) for Item 3")
```




```
itemplot(mirt.out, item = 4, type = "trace", auto.key =  list(columns = 4, legend = TRUE),
         main = "Item Characteristic Curves (ICC) for Item 4")
```

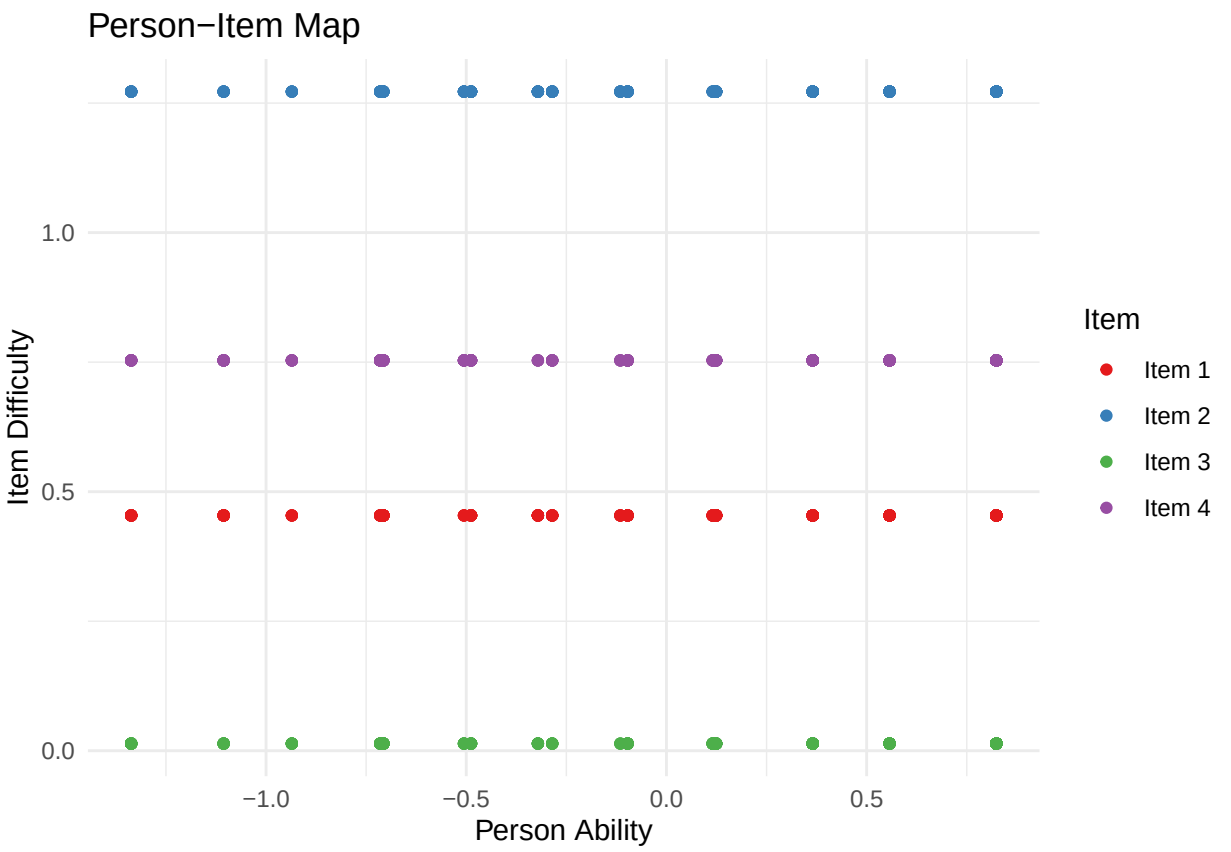


Person-Item Map

- *Person-Item Map* usually represents the distribution of person abilities in relation to item difficulties. It helps in understanding how well the items on a test are targeted to the ability of the test takers.

```
# Plot person-item map

# retrieving coef()$items difficutly estimates
item.difficulties <- coef(mirt.out, simplify = TRUE)$items[, 2]
item.difficulties <- data.frame(Item = paste("Item", 1:4), Difficulty = item.difficulties)

# retrieving f-scores (person ability estimates)
person.abilities <- fscores(mirt.out)
person.abilities <- data.frame(Ability = person.abilities)

item.difficulties.long <- item.difficulties[rep(seq_len(nrow(item.difficulties)), each = nrow(person.abi
item.difficulties.long <- item.difficulties[rep(1:nrow(item.difficulties), each = length(person.abilitie

# difficulties with person abilities
plot_data <- data.frame(
    Item = item.difficulties.long$Item,
    Difficulty = item.difficulties.long$Difficulty,
    Ability = person.abilities
  )
```

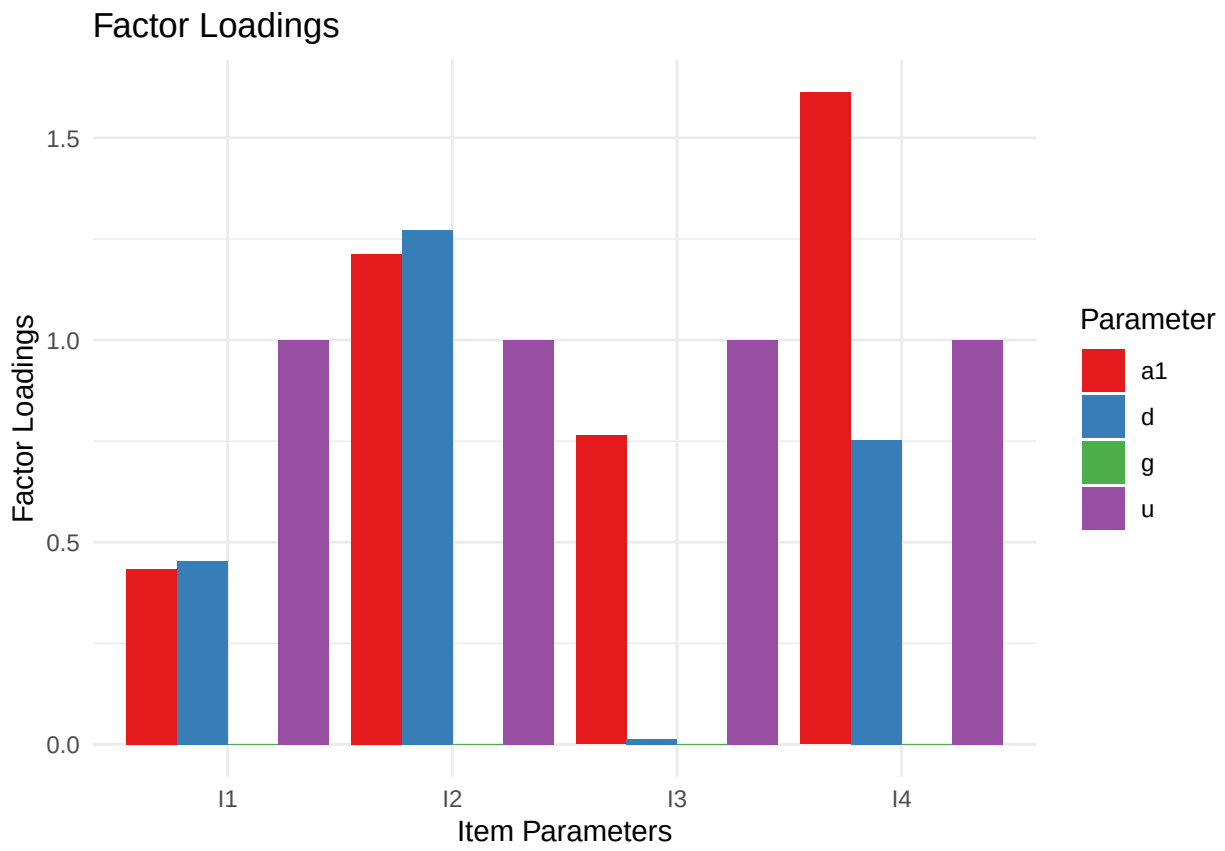
```r
ggplot(plot_data, aes(x = F1, y = Difficulty, color = Item)) +
    geom_point() +
    labs(title = "Person-Item Map", x = "Person Ability", y = "Item Difficulty") +
    theme_minimal() +
    scale_color_brewer(palette = "Set1")
```



Factor Loadings

- *Factor Loading* in multidimensional IRT, factor loadings describe the relationship (correlation) between the items and latent traits. High factor loadings indicate a strong association between an item and a particular trait.

```r
# Factor loadings
loadings <- coef(mirt.out, simplify = TRUE)$items
loadings.long <- reshape2::melt(loadings)
colnames(loadings.long) <- c("Item", "Parameter", "Value")

ggplot(loadings.long, aes(x = Item, y = Value, fill = Parameter)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Factor Loadings", x = "Item Parameters", y = "Factor Loadings") +
    theme_minimal() +
    scale_fill_brewer(palette = "Set1")
```



Plot residual analysis

- *Residual Analysis* involves examining the differences between the observed item responses and the responses predicted by the model. It helps in identifying items that do not fit well in the model, indicating potential issues with the item or assumptions.

```
# Plot residual analysis
residuals <- residuals(mirt.out)
```

```
## LD matrix (lower triangle) and standardized values.
##
## Upper triangle summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.007  -0.002   0.000   0.000   0.003   0.004
##
##       I1     I2    I3     I4
## I1    NA -0.007 0.004  0.003
## I2 0.043     NA 0.001 -0.001
## I3 0.013  0.002    NA -0.003
## I4 0.010  0.000 0.007     NA
```

```
residuals <- data.frame(residuals)
residuals_long <- reshape2::melt(residuals)
```

```
## No id variables; using all as measure variables
```

```r
colnames(residuals_long) <- c("Var1", "Residual")

ggplot(residuals_long, aes(x = Var1, y = Residual, color = abs(Residual))) +
    geom_point(alpha = 0.5) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    labs(title = "Residual Analysis", x = "Item", y = "Residual") +
    theme_minimal() +
    scale_color_gradient(low = "blue", high = "red")
```

```
## Warning: Removed 4 rows containing missing values (‘geom_point()‘).
```