Open Tests: Harvard Measurement Lab

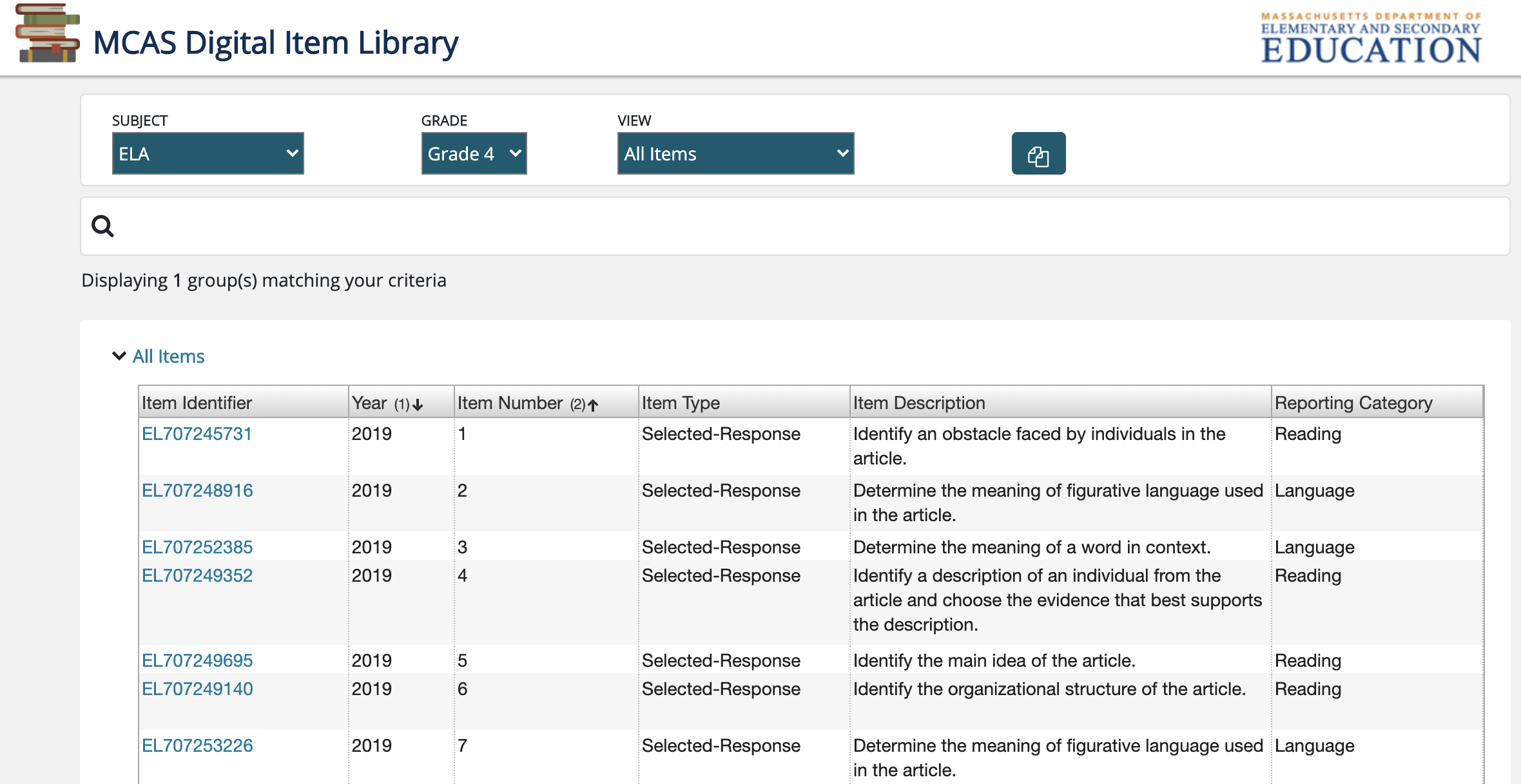
Emma Dwight

7/17/2020

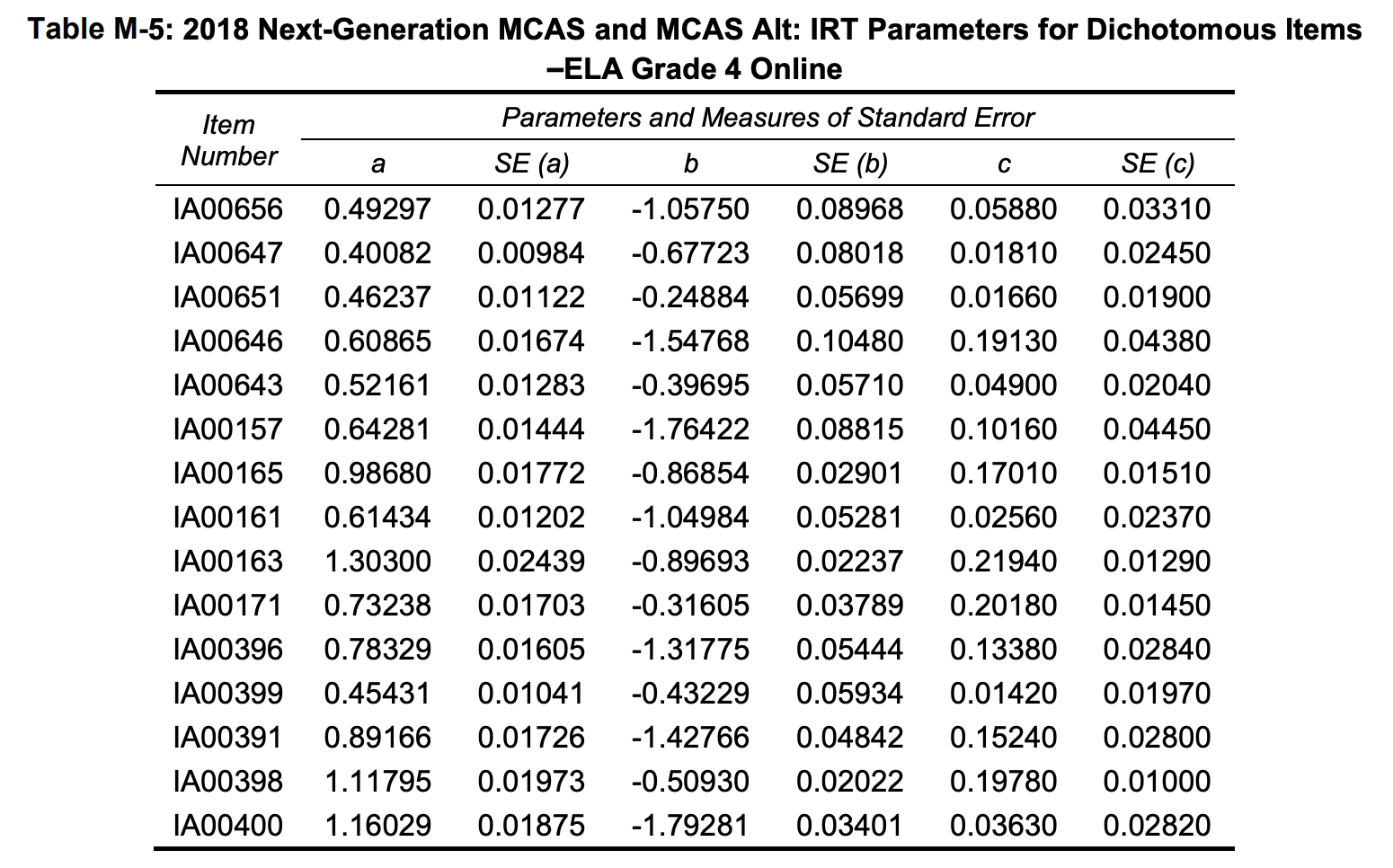
In this tutorial we show how schools, districts, and states can create and score a test comprised of publicly available questions from state websites. As our example, we use MCAS 4th grade ELA and released questions from 2018.

# Ingredients:

1. Released test questions (items) to create the test, like these:

 <https://mcas.digitalitemlibrary.com/home?subject=ELA&grades=Grade%204&view=ALL>

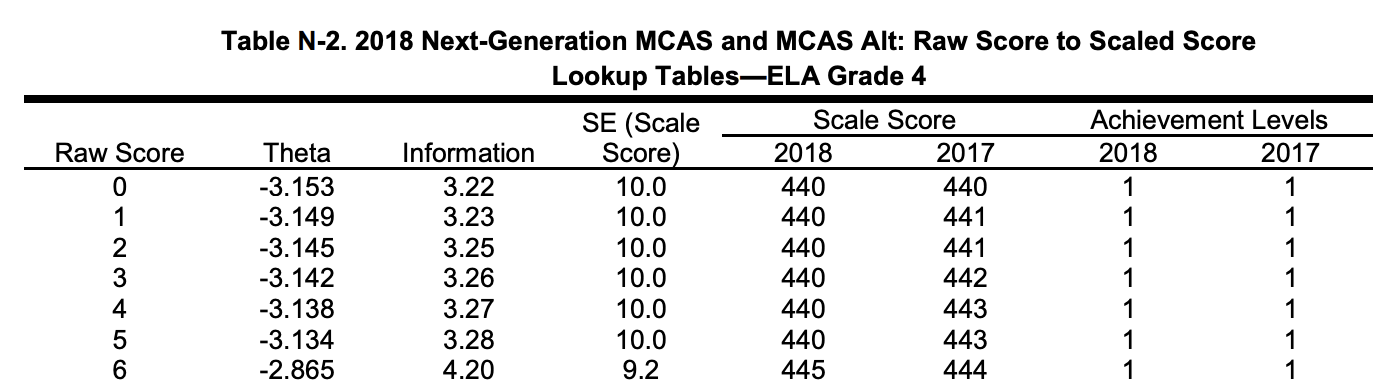
1. Released item IRT parameter estimates, like table M5 here:



Source: <http://www.mcasservicecenter.com/documents/MA/Technical%20Report/2018/NextGen/Appendix%20M%20-%20Plots%20and%20IRT%20Parameters.pdf>

<http://www.mcasservicecenter.com/documents/MA/Technical%20Report/2018/NextGen/Appendix%20M%20-%20Plots%20and%20IRT%20Parameters.pdf>

1. Common item numbers/identifiers that relate each question (item) to its parameters: currently missing for MCAS
2. Student responses for each of the test questions, graded as correct/incorrect: we simulate data below
3. A table that converts theta scores to scale scores (and possibly also achievement levels), like table N2, here (shortened):

 <http://www.mcasservicecenter.com/documents/MA/Technical%20Report/2018/NextGen/Appendix%20N%20-%20Scaled%20Score%20Distributions%20and%20Look-up%20Tables_4.17.19.pdf>

# Contents:

### Estimated Scale Scores from Sum Scores:

* Import 3PL IRT item parameters
* Import theta to scale score table
* Simulate student response data
* Estimate student thetas using sum scores
* Convert estimated thetas to scale scores
* Produce table converting sum scores to scale scores, using Test Characteristic Curve information
* Export student ability data, including estimated thetas, scale scores, and achievement levels

### Appendix 1: Simulating Imprecision for Sum Score Theta Estimates

* Invert the Test Characteristic Curve to Produce Estimates of Standard Errors
* Produce table converting sum scores to scale scores, with empirical standard errors, using simulations

### Appendix 2: Scale Score Estimation Using Full-Pattern Scoring

* Estimate student ability using full-pattern scoring, (includes standard errors)
* Export student ability data, including estimated thetas, standard errors, scale scores, and achievement levels

### Appendix 3: Diagnostics

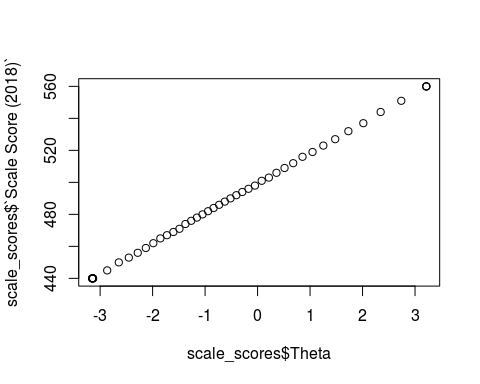
* Report Classical Test Theory statistics
* Plot Item Characteristic Curves and Test Characteristic Curve
* Plot Item Information Functions and Test Information Function

### To do next:

* Format content/hyperlinks within Rmd/html
* Add support for polytomously scored items? (Hand-scored, GRM stuff for MCAS)
* Add support/instructions for 1PL & 2PL models
* Add code to import student responses, including student names/identifiers to attach to the thetas & scale scores

# Importing item parameters and scale scores

# Import item parameter data  
item\_parameters\_raw <- read\_excel("tablem5.xlsx")   
  
# Separate the item parameters from the standard errors  
my\_ip <- as.matrix(item\_parameters\_raw %>% dplyr::select(a, b, c))   
my\_se <- as.matrix(item\_parameters\_raw %>% dplyr::select("se(a)", "se(b)", "se(c)"))  
  
# The MCAS Technical Report tells us they use a normalizing constant of D = 1.701, other states and tests may not have a D at all in their equations  
# Set D below, either to 1 if none appears in the 2PL/3PL equations in your technical report, or to the appropriate value (probably 1.7 or 1.701)  
D <- 1.701 # Change to D = 1 if none appears in your technical report  
  
# Divide "a" parameters by D, and the se(a) parameters by D^2, to "undo" the normalizing constant that was applied, to fit what the package irtoys expects  
my\_ip[,1] <- my\_ip[,1]/D  
my\_se[,1] <- my\_se[,1]/(D^2)  
  
# Import student responses (if using real data)  
  
# Import theta -> scale score table  
scale\_scores\_raw <- read\_excel("tablen2.xlsx")  
  
# Select the relevant columns:  
scale\_scores <- scale\_scores\_raw %>% dplyr::select("Theta", "Scale Score (2018)")  
  
# Plot relationship between thetas and scale scores:  
plot(x = scale\_scores$Theta, y = scale\_scores$`Scale Score (2018)`)



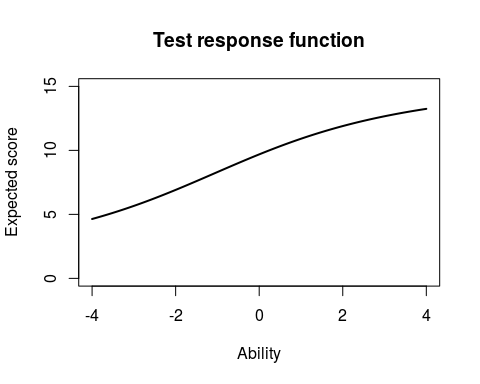
# Relationship is a straight line! Learn the model so we can make this conversion ourselves, later.  
theta\_to\_scale\_score <- lm(`Scale Score (2018)` ~ Theta, data = scale\_scores)  
summary(theta\_to\_scale\_score)

##   
## Call:  
## lm(formula = `Scale Score (2018)` ~ Theta, data = scale\_scores)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.48284 -0.21294 -0.04279 0.35221 0.49858   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 499.42554 0.04645 10751.0 <2e-16 \*\*\*  
## Theta 18.85391 0.02377 793.3 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2994 on 43 degrees of freedom  
## Multiple R-squared: 0.9999, Adjusted R-squared: 0.9999   
## F-statistic: 6.294e+05 on 1 and 43 DF, p-value: < 2.2e-16

# Scale Score Estimation Using Sum Scoring

“Invert” the TCC to get a map from sum scores to thetas:

# Plot Test Characteristic Curve (which underlies all of what follows)  
# Theta on x axis, predicted sum score on y axis  
plot(trf(my\_ip))



# Using the smallest and largest thetas provided in the scale\_scores table, make a sequence of every possible Theta between these  
thetas = seq(from = min(scale\_scores$Theta), to = max(scale\_scores$Theta), length.out = 100000)  
# Calculate the estimated sum score for each of these thetas   
all\_sum\_scores <- trf(my\_ip, x = thetas)  
  
# Simplifies the above possible sum scores (decimals) into possible whole number scores  
possible\_sum\_scores <- seq(from = ceiling(all\_sum\_scores$f[1]), to = floor(all\_sum\_scores$f[100000]), by = 1)  
  
# For each possible whole number score, find the associated theta that's the best match  
matching\_theta\_scores <- rep(NA, times = length(possible\_sum\_scores))  
for(i in 1:length(possible\_sum\_scores)) {matching\_theta\_scores[i] <- all\_sum\_scores$x[which.min(abs(all\_sum\_scores$f - possible\_sum\_scores[i]))]}  
  
# Produce a table with sum scores, thetas, scale scores, and achievement levels  
matching\_theta\_scores\_df <- as.data.frame(cbind(possible\_sum\_scores, matching\_theta\_scores))  
colnames(matching\_theta\_scores\_df) <- c("SumScore", "Theta")  
matching\_theta\_scores\_df$ScaleScore <- predict(theta\_to\_scale\_score, newdata = matching\_theta\_scores\_df)  
matching\_theta\_scores\_df$AchievementLevel <- 1  
matching\_theta\_scores\_df$AchievementLevel[matching\_theta\_scores\_df$ScaleScore > 470] <- 2  
matching\_theta\_scores\_df$AchievementLevel[matching\_theta\_scores\_df$ScaleScore > 500] <- 3  
matching\_theta\_scores\_df$AchievementLevel[matching\_theta\_scores\_df$ScaleScore > 530] <- 4  
matching\_theta\_scores\_df$ScaleScore <- round(matching\_theta\_scores\_df$ScaleScore)  
  
# View table  
matching\_theta\_scores\_df

## SumScore Theta ScaleScore AchievementLevel  
## 1 6 -2.7141767 448 1  
## 2 7 -1.9365725 463 1  
## 3 8 -1.2176181 476 2  
## 4 9 -0.5080248 490 2  
## 5 10 0.2388218 504 3  
## 6 11 1.0813166 520 3  
## 7 12 2.1147260 539 4

# Export table  
write.csv(matching\_theta\_scores\_df, file = "Estimated\_Scale\_Scores\_From\_Sum\_Scores.csv", row.names = F)

# Appendix 1: Simulating Imprecision for Sum Score Theta Estimates

How do we convey a sense of uncertainty with these sorts of estimates?

Uncertainty arises from two sources: \* students with “fixed” thetas can retake the test and score slightly differently each time \* the standard errors on the item parameter estimates

Simulation 1 focuses only on the retesting, while simulation 2 include both sources of uncertainty.

## Simulation 1: Theta imprecision from “retesting”

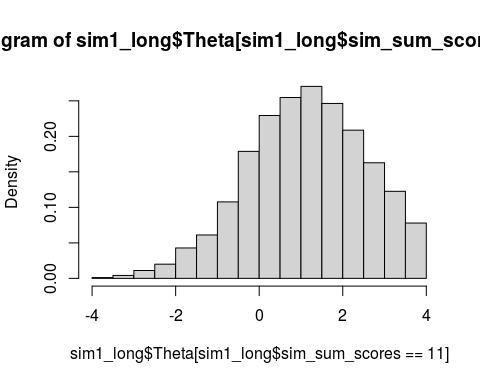
# Create a sequence of possible thetas, equally spaced  
sim\_thetas <- seq(from = -4, to = 4, length.out = 801)  
# Replicate each of the possible thetas 100 times, as if each of these students took the test 100 times  
sim\_thetas\_n <- rep(sim\_thetas, each = 100)  
# Simulate the sum scores that these students received, using the item parameters for this test (assume these item parameters are estimated perfectly)  
sim\_sum\_scores <- rowSums(sim(my\_ip, sim\_thetas\_n))  
# Combine the thetas and sum scores into a data frame  
sim1\_long <- as.data.frame(cbind(sim\_thetas\_n, sim\_sum\_scores))  
# Rename column to the name expected by theta\_to\_scale\_score conversion  
colnames(sim1\_long) <- c("Theta", "sim\_sum\_scores")  
# Add column for scale scores (from thetas)  
sim1\_long$sim\_scale\_scores <- predict(theta\_to\_scale\_score, newdata = sim1\_long)  
  
# Group the above data frame by sum scores, and summarise the different thetas that could have produced each sum score  
sim1\_thetas <- sim1\_long %>%   
 group\_by(sim\_sum\_scores) %>%   
 summarise(tibble('2.5' = round(quantile(sim\_scale\_scores, 0.025), 0),  
 '10' = round(quantile(sim\_scale\_scores, 0.1), 0),  
 mean = round(mean(sim\_scale\_scores), 0),   
 median = round(median(sim\_scale\_scores), 0),   
 se = round(sd(sim\_scale\_scores), 1),  
 '90' = round(quantile(sim\_scale\_scores, 0.9), 0),  
 '97.5' = round(quantile(sim\_scale\_scores, 0.975), 0),   
 max = round(max(sim\_scale\_scores), 0),  
 nsims = n()))  
sim1\_thetas

## # A tibble: 16 x 10  
## sim\_sum\_scores `2.5` `10` mean median se `90` `97.5` max nsims  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>  
## 1 0 424 426 433 431 7.9 444 451 451 26  
## 2 1 425 426 435 432 10.5 448 467 476 202  
## 3 2 424 426 438 435 11.8 454 466 500 784  
## 4 3 424 426 440 437 13.2 459 472 498 1862  
## 5 4 425 427 445 441 15.9 467 482 521 3446  
## 6 5 425 428 449 446 17.5 473 490 526 5012  
## 7 6 426 431 457 455 20.6 485 502 543 6077  
## 8 7 427 436 465 464 22.6 496 512 562 6709  
## 9 8 431 444 478 477 25.3 510 529 574 7054  
## 10 9 440 457 491 490 26.6 525 545 575 7473  
## 11 10 451 471 506 505 27.5 543 561 575 7854  
## 12 11 465 487 521 521 26.6 556 568 575 8597  
## 13 12 483 502 534 536 24.2 565 572 575 9534  
## 14 13 496 515 545 548 21.5 570 574 575 8401  
## 15 14 510 526 552 556 17.9 571 574 575 5298  
## 16 15 518 536 556 560 15.1 573 574 575 1771

sim1\_scale\_scores <- sim1\_long %>%   
 group\_by(sim\_sum\_scores) %>%   
 summarise(tibble('2.5' = round(quantile(sim\_scale\_scores, 0.025), 0),  
 '10' = round(quantile(sim\_scale\_scores, 0.1), 0),  
 mean = round(mean(sim\_scale\_scores), 0),   
 median = round(median(sim\_scale\_scores), 0),   
 se = round(sd(sim\_scale\_scores), 1),  
 '90' = round(quantile(sim\_scale\_scores, 0.9), 0),  
 '97.5' = round(quantile(sim\_scale\_scores, 0.975), 0),   
 max = round(max(sim\_scale\_scores), 0),  
 nsims = n()))  
sim1\_scale\_scores

## # A tibble: 16 x 10  
## sim\_sum\_scores `2.5` `10` mean median se `90` `97.5` max nsims  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>  
## 1 0 424 426 433 431 7.9 444 451 451 26  
## 2 1 425 426 435 432 10.5 448 467 476 202  
## 3 2 424 426 438 435 11.8 454 466 500 784  
## 4 3 424 426 440 437 13.2 459 472 498 1862  
## 5 4 425 427 445 441 15.9 467 482 521 3446  
## 6 5 425 428 449 446 17.5 473 490 526 5012  
## 7 6 426 431 457 455 20.6 485 502 543 6077  
## 8 7 427 436 465 464 22.6 496 512 562 6709  
## 9 8 431 444 478 477 25.3 510 529 574 7054  
## 10 9 440 457 491 490 26.6 525 545 575 7473  
## 11 10 451 471 506 505 27.5 543 561 575 7854  
## 12 11 465 487 521 521 26.6 556 568 575 8597  
## 13 12 483 502 534 536 24.2 565 572 575 9534  
## 14 13 496 515 545 548 21.5 570 574 575 8401  
## 15 14 510 526 552 556 17.9 571 574 575 5298  
## 16 15 518 536 556 560 15.1 573 574 575 1771

# Can also create a histogram for the possible thetas that produced each sum score, for example:  
hist(sim1\_long$Theta[sim1\_long$sim\_sum\_scores == 11], freq = F)



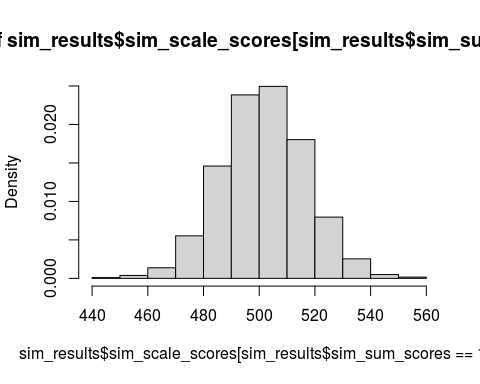
## Simulation 2: Theta imprecision arising from both “retesting” and from item parameter imprecision

Simulation 2 takes both sources of error into account to produce a range of possible scale scores for a given sum score \* each simulation will use slightly different item parameters, where each a\*, b\*, and c\* is randomly drawn from ~N(a, se(a)), ~N(b, se(b)), ~N(c, se(c)) \* each simulation will use these item parameters to assign probabilities for a simulated “student” of a particular theta getting an item correct, and then “flip a coin” to determine whether that “student” got the item correct \* we then can map fixed student thetas to sum scores, which can then be used backwards to see what range of thetas each sum score could have been produced by

n\_tests = 100  
n\_students = 100  
  
# Initialise an empty data frame  
sim\_results <- data.frame(matrix(ncol = 2, nrow = 0))  
for(i in 1:n\_tests)  
{  
 new\_ip <- cbind(rnorm(n = 15, mean = my\_ip[,1], sd = my\_se[,1]),   
 rnorm(n = 15, mean = my\_ip[,2], sd = my\_se[,2]),  
 rnorm(n = 15, mean = my\_ip[,3], sd = my\_se[,3]))  
   
 new\_thetas <- rnorm(n\_students)  
 new\_responses <- sim(new\_ip, new\_thetas)  
 new\_sums <- rowSums(new\_responses)  
 sim\_results <- rbind(sim\_results, cbind(new\_thetas, new\_sums))  
}  
  
colnames(sim\_results) <- c("Theta", "sim\_sum\_scores")  
sim\_results$sim\_scale\_scores <- predict(theta\_to\_scale\_score, newdata = sim\_results)  
colnames(sim\_results)[1] <- "sim\_thetas"  
  
sim2\_scale\_scores <- as.data.frame(sim\_results) %>%   
 group\_by(sim\_sum\_scores) %>%   
 summarise(tibble('2.5' = round(quantile(sim\_scale\_scores, 0.025), 0),  
 '10' = round(quantile(sim\_scale\_scores, 0.1), 0),  
 mean = round(mean(sim\_scale\_scores), 0),   
 median = round(median(sim\_scale\_scores), 0),   
 se = round(sd(sim\_scale\_scores), 1),  
 '90' = round(quantile(sim\_scale\_scores, 0.9), 0),  
 '97.5' = round(quantile(sim\_scale\_scores, 0.975), 0),   
 max = round(max(sim\_scale\_scores), 0),  
 nsims = n()))  
  
  
sim2\_scale\_scores

## # A tibble: 15 x 10  
## sim\_sum\_scores `2.5` `10` mean median se `90` `97.5` max nsims  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>  
## 1 1 465 467 475 475 14 483 484 485 2  
## 2 2 449 450 464 464 14.2 483 485 486 8  
## 3 3 443 447 465 463 14.9 481 492 499 30  
## 4 4 438 449 472 472 16.5 492 496 515 80  
## 5 5 450 461 479 478 15 498 507 525 242  
## 6 6 453 463 483 484 15.1 502 511 531 483  
## 7 7 456 466 486 486 15.5 505 519 534 842  
## 8 8 462 474 492 492 14.6 510 520 540 1247  
## 9 9 465 475 495 495 15.8 516 527 541 1650  
## 10 10 472 483 502 501 15.4 521 533 555 1808  
## 11 11 476 487 506 506 15.2 526 535 553 1598  
## 12 12 480 491 511 510 15.8 531 543 562 1151  
## 13 13 486 496 515 516 15.1 534 544 562 608  
## 14 14 491 502 523 521 16.9 545 556 564 214  
## 15 15 491 508 527 528 15.8 548 552 554 37

# Can even create histograms of likely scale scores given a particular sum score: (eg. 10)  
hist(sim\_results$sim\_scale\_scores[sim\_results$sim\_sum\_scores == 10], freq = F)



# Appendix 2: Scale Score Estimation Using Full-Pattern Scoring

# Use parameter estimates to simulate answers for 100 students  
# (Import student data here if using real answers)  
set.seed(88)  
sim\_thetas <- rnorm(100)  
sim\_responses <- sim(my\_ip, sim\_thetas)  
  
# Put the parameter estimates and standard errors into the list structure that irtoys functions expect  
# Note: ability estimation function does not need standard errors to run  
parameter\_list <- list(est = my\_ip, se = my\_se, vcm = NA)  
  
# Estimate student thetas, based on full-pattern scoring, using MLE (several other methods exist in this "ability" function)  
mod\_MLE<- ability(resp = sim\_responses, ip = parameter\_list, method = "MLE")  
  
# Look at the first five rows  
mod\_MLE[1:5, ]

## est sem n  
## [1,] 2.6572126 1.824988 15  
## [2,] 0.2634567 1.327549 15  
## [3,] 3.9999341 2.289135 15  
## [4,] -0.6151277 1.276231 15  
## [5,] -0.5962468 1.276476 15

# Convert these estimated thetas to scale scores and achievement levels  
ability\_df <- as.data.frame(mod\_MLE)  
colnames(ability\_df) <- c("Theta", "se(Theta)", "n")  
ability\_df$ScaleScore <- predict(theta\_to\_scale\_score, newdata = ability\_df)  
  
# Add achievement levels, based on known scale score cutoffs  
ability\_df$AchievementLevel <- 1  
ability\_df$AchievementLevel[ability\_df$ScaleScore > 470] <- 2  
ability\_df$AchievementLevel[ability\_df$ScaleScore > 500] <- 3  
ability\_df$AchievementLevel[ability\_df$ScaleScore > 530] <- 4  
  
# Round the scale scores to 0 dp before reporting  
ability\_df$ScaleScore <- round(ability\_df$ScaleScore, 0)  
  
# View table:  
ability\_df

## Theta se(Theta) n ScaleScore AchievementLevel  
## 1 2.65721260 1.824988 15 550 4  
## 2 0.26345673 1.327549 15 504 3  
## 3 3.99993412 2.289135 15 575 4  
## 4 -0.61512770 1.276231 15 488 2  
## 5 -0.59624676 1.276476 15 488 2  
## 6 0.95215476 1.421211 15 517 3  
## 7 3.99992119 2.289130 15 575 4  
## 8 -3.70899024 1.792110 15 429 1  
## 9 -3.21898609 1.634530 15 439 1  
## 10 1.55941518 1.538792 15 529 3  
## 11 -0.49457846 1.278462 15 490 2  
## 12 0.38624123 1.340966 15 507 3  
## 13 1.66408195 1.562134 15 531 4  
## 14 -1.07122217 1.282270 15 479 2  
## 15 -0.31404380 1.284726 15 494 2  
## 16 -0.56574035 1.276954 15 489 2  
## 17 0.65761636 1.375746 15 512 3  
## 18 -1.43635137 1.304012 15 472 2  
## 19 -0.92013513 1.277702 15 482 2  
## 20 0.93581570 1.418484 15 517 3  
## 21 1.66109354 1.561456 15 531 4  
## 22 0.39932547 1.342482 15 507 3  
## 23 0.48304145 1.352571 15 509 3  
## 24 -0.87542548 1.276841 15 483 2  
## 25 -0.45728098 1.279470 15 491 2  
## 26 -1.37486449 1.299281 15 474 2  
## 27 -0.08949490 1.297319 15 498 2  
## 28 0.03120847 1.306243 15 500 3  
## 29 -0.31643314 1.284621 15 493 2  
## 30 -1.25857872 1.291522 15 476 2  
## 31 -1.32300168 1.295629 15 474 2  
## 32 0.71459938 1.383926 15 513 3  
## 33 -1.35073740 1.297544 15 474 2  
## 34 0.24136217 1.325292 15 504 3  
## 35 2.52413728 1.785608 15 547 4  
## 36 -1.40164053 1.301288 15 473 2  
## 37 -0.91132082 1.277515 15 482 2  
## 38 0.05269105 1.307987 15 500 3  
## 39 -1.13946382 1.285178 15 478 2  
## 40 3.99993575 2.289135 15 575 4  
## 41 1.71609660 1.574058 15 532 4  
## 42 3.99992284 2.289130 15 575 4  
## 43 -3.00341548 1.574423 15 443 1  
## 44 1.87619204 1.612082 15 535 4  
## 45 0.36528842 1.338573 15 506 3  
## 46 -1.05923252 1.281814 15 479 2  
## 47 0.03942169 1.306904 15 500 3  
## 48 1.06962053 1.441504 15 520 3  
## 49 0.55540620 1.361830 15 510 3  
## 50 0.44767795 1.348228 15 508 3  
## 51 1.24274512 1.473593 15 523 3  
## 52 0.65761636 1.375746 15 512 3  
## 53 -3.23641042 1.639634 15 438 1  
## 54 -0.78243793 1.275764 15 485 2  
## 55 -1.28352601 1.293056 15 475 2  
## 56 -1.46720224 1.306550 15 472 2  
## 57 -0.74229351 1.275595 15 485 2  
## 58 2.19796519 1.694406 15 541 4  
## 59 0.90201903 1.412919 15 516 3  
## 60 -1.93294199 1.358315 15 463 1  
## 61 0.88759272 1.410575 15 516 3  
## 62 -2.66866586 1.492162 15 449 1  
## 63 0.46216294 1.349992 15 508 3  
## 64 0.19095951 1.320321 15 503 3  
## 65 -0.28777685 1.285927 15 494 2  
## 66 -0.39298241 1.281559 15 492 2  
## 67 -0.78789773 1.275801 15 485 2  
## 68 1.96181185 1.633227 15 536 4  
## 69 -0.11392053 1.295695 15 497 2  
## 70 -1.34123126 1.296877 15 474 2  
## 71 0.58184205 1.365336 15 510 3  
## 72 0.86735780 1.407318 15 516 3  
## 73 -0.06423333 1.299063 15 498 2  
## 74 2.31705622 1.726818 15 543 4  
## 75 2.06921912 1.660537 15 538 4  
## 76 -0.04139416 1.300697 15 499 2  
## 77 0.64019726 1.373306 15 511 3  
## 78 0.92484530 1.416667 15 517 3  
## 79 -2.07408398 1.379022 15 460 1  
## 80 -0.15393361 1.293168 15 497 2  
## 81 -0.51347505 1.278008 15 490 2  
## 82 0.10146083 1.312118 15 501 3  
## 83 1.10624243 1.448077 15 520 3  
## 84 0.31819208 1.333349 15 505 3  
## 85 2.64398094 1.821017 15 549 4  
## 86 -0.70227504 1.275604 15 486 2  
## 87 -0.95351782 1.278491 15 481 2  
## 88 1.81067078 1.596280 15 534 4  
## 89 -3.58104780 1.748144 15 432 1  
## 90 -0.70650159 1.275595 15 486 2  
## 91 0.73915479 1.387543 15 513 3  
## 92 -2.08655558 1.380964 15 460 1  
## 93 -2.63144761 1.483845 15 450 1  
## 94 -1.55975682 1.314825 15 470 2  
## 95 3.89815069 2.249798 15 573 4  
## 96 -0.93174264 1.277962 15 482 2  
## 97 -0.22896747 1.288877 15 495 2  
## 98 -0.60837369 1.276314 15 488 2  
## 99 2.40235610 1.750664 15 545 4  
## 100 -1.21695254 1.289121 15 476 2

# Export the data   
write.csv(ability\_df, file = "Estimated\_Scores\_From\_Full\_Pattern\_Scores.csv", row.names = F)  
# The above file will appear in the file window on the bottom-right hand side of the screen  
# To download to local computer, select this file using the checkbox, then use More...Export...

# Appendix 3: Diagnostics

## Diagnostics and other test plots:

# Simulate data  
set.seed(88)  
sim\_thetas <- rnorm(100)  
sim\_responses <- sim(my\_ip, sim\_thetas)  
  
# Classical Test Theory EDA metrics:  
# Note: Because the simulated data is pre-graded, we're saying the "answer key" is 1, 1, 1, 1, 1....  
ctt <- tia(sim\_responses, key = rep(1, 15))  
  
# Show Cronbach's alpha for this "test":  
ctt$testlevel$alpha

## [1] 0.2134204

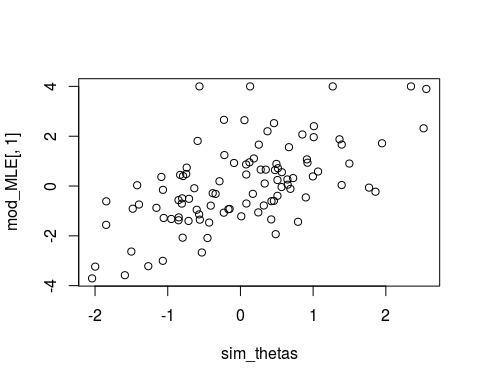
# Show CTT item-level statistics for first five items  
ctt$itemlevel[1:5, ]

## Prop. correct Item-sum cor. Alpha without  
## Item1 0.64 0.3553741 0.04551212  
## Item2 0.51 0.1998617 0.13001584  
## Item3 0.56 0.3043689 0.07040776  
## Item4 0.67 0.3264939 0.06688087  
## Item5 0.52 0.3511876 0.03941584

# How does the MLE ability estimation perform compared to the "true thetas" from our simulation?  
cor(mod\_MLE[,1], sim\_thetas)

## [1] 0.6044257

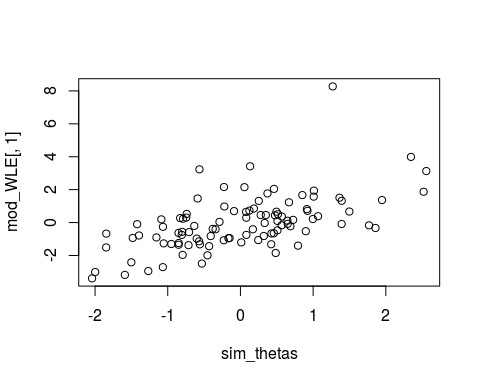
plot(sim\_thetas, mod\_MLE[,1])



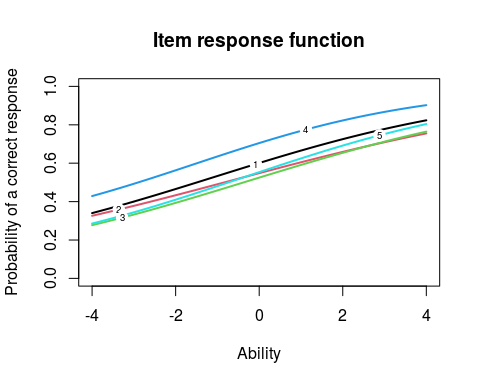
# How do other ability estimation methods perform?  
mod\_WLE <- ability(resp = sim\_responses, ip = parameter\_list, method = "WLE")  
cor(mod\_WLE[,1], sim\_thetas)

## [1] 0.577338

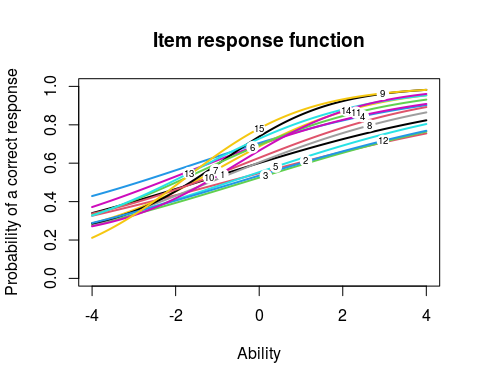
plot(sim\_thetas, mod\_WLE[,1])



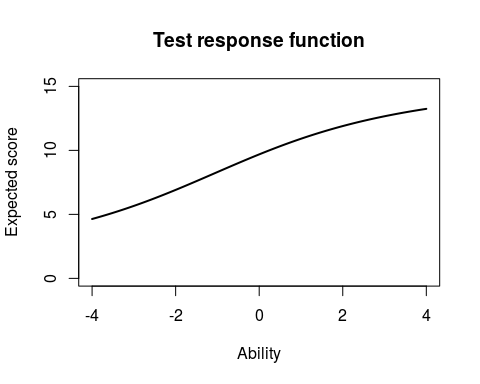
# Item Characteristic Curves for first five items (this package calls them "item response functions")  
plot(irf(my\_ip, items = 1:5), co = NA, label = T)



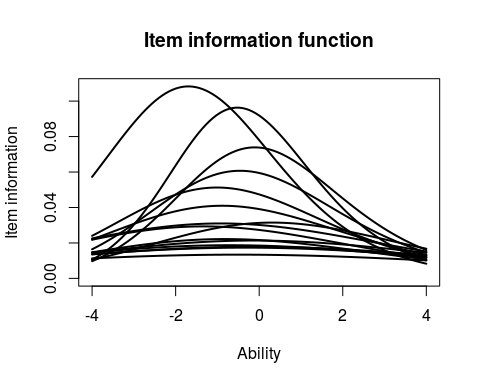
# ICCs for all items:  
plot(irf(my\_ip), co = NA, label = T)



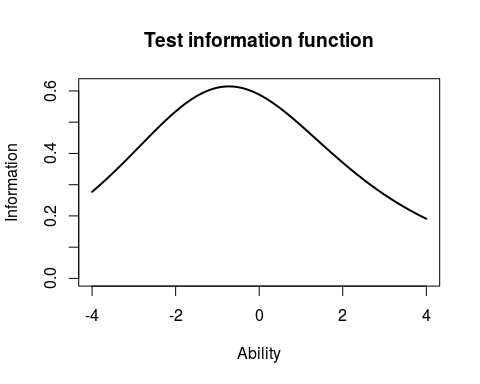
# Test Characteristic Curve (this package calls this a "test response function")  
plot(trf(my\_ip))



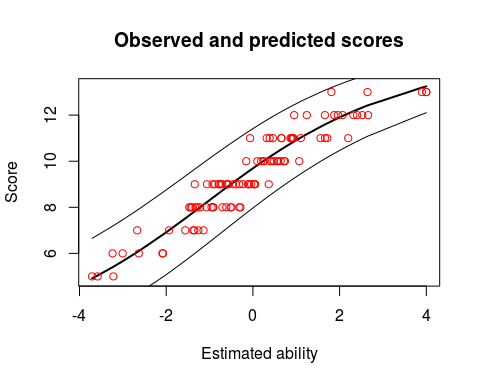
# Overlaid item information curves  
plot(iif(my\_ip))



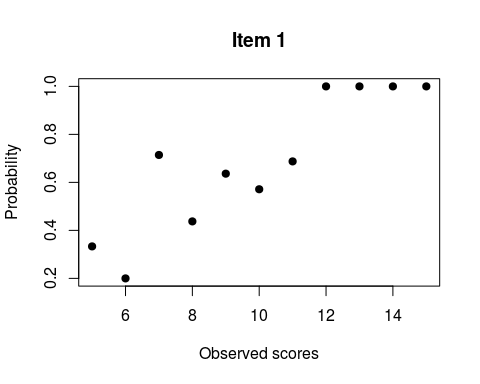
# Test information function  
plot(tif(my\_ip))



# Cool plot of observed sum scores and predicted sum scores against estimated ability, with +/- 1se bands  
scp(sim\_responses, my\_ip)



# "Empirical response function" for a selected item: observed sum scores vs. percent correct on this question  
erf(sim\_responses, 1)



## x y  
## 5 5 0.3333333  
## 6 6 0.2000000  
## 7 7 0.7142857  
## 8 8 0.4375000  
## 9 9 0.6363636  
## 10 10 0.5714286  
## 11 11 0.6875000  
## 12 12 1.0000000  
## 13 13 1.0000000  
## 14 14 1.0000000  
## 15 15 1.0000000