r2Winsteps: An R package for interfacing between R and the Rasch Modeling Software Winsteps

Daniel Anderson

2016-04-01

Contents

Contents

1 Introduction

The *r2Winsteps* package was developed to provide a convenient interface between *R* (R Core Team 2015) and the Rasch modeling software *Winsteps* (Linacre 2016). The package is not intended to encompass the full capabilities of *Winsteps*, but rather to provide a simple framework for estimating many commonly applied models. The primary features of the package include:

- Write control and data files for Winsteps with the `r2Winsteps()` function, which includes automatic detection of polytomous item types. Either the rating scale (default; Andrich 1978) or partial credit models (Masters 1982) can be estimated.
- Run Winsteps directly from R with the `runWinsteps()` function, which writes and executes a `.bat` file to call *Winsteps*. Item, person, and structure (if a polytmous model is estimated) parameters are returned in a list. Intermediary files (control, data, output, etc.) can be stored or discarded (default).
- Batch run a set of models with the `batchRunWinsteps()` files. Essentially calls `runWinsteps()`, but takes as its argument a list of data frames, with a different model fit and parameter estimates returned for each data frame in the list.

There were three primary motivation for developing *r2Winsteps*. First, *Winsteps* is a powerful software that can fit a range of Rasch models with impressive speed. By linking the software to *R*, a unified framework is provided where users can prep, explore, and analyze their data within the same environment. This is important not only for convenience and efficiency purposes, but also to facilitate transparency. When multiple software packages are used, and multiple script files stored, it can be easy for an outside researcher (or indeed, the same researcher at a later time) to get lost in the process used. This is also an important principle of reproducible research (Stodden, Leisch, and Peng 2014), and can facilitate dynamic document generation (Xie 2015) with Winsteps, where all tables and figures are generated in a report are generated dynamically. That is, as the model or data are updated, so too are the features within the document that depend on them. This document is an example of using *Winsteps* within *R* to generate a reproducible and dynamic document, via the *r2Winsteps* package (although in practice, most documents would include the code being hidden).

The second primary motivation for developing *r2Winsteps* was the powerful plotting capabilities in *R*, coupled with tremendous flexibility. *Winsteps* provides many excellent plots, available through the graphs menu, but they are limited in that the user has little control over their appearance (e.g., line types, colors, etc.). By contrast, *R* provides almost limitless flexibility, and multiple packages are available for facilitating plot creation (e.g., ggplot2 (Wickham 2009), lattice (Sarkar 2008)). The built-in plotting function in the *r2Winsteps* package are still under active development. These were all created with the base graphics, and should be fully customizable.

The third and final primary motivation for developing *r2Winsteps* was to develop an efficient method for batch processing analyses with *Winsteps*. While a batch processing option is available through *Winsteps*, one still must write each control and data file for each analysis. The *r2Winsteps* package allows for an efficient means of batch writing control and data files for analysis, which can then be estimated through *Winsteps* batch mode.

Note that the package is still in active development, and this vignette will be updated over time accordingly. The purpose of this vignette is to provide a few illustrated examples of using the package in its current stage. If you use the package, please reference it using the code below. If you find any bugs, please email them to daniela@uoregon.edu, or log them at https://github.com/DJAnderson07/r2Winsteps/issues.

```
citation(package = "r2Winsteps")
```

```
##
## To cite package 'r2Winsteps' in publications use:
##
##   Daniel Anderson (2015). r2Winsteps: A package for interfacing
##   between R and the Rasch modeling software Winsteps. R package
##   version 0.0.0.9000. https://github.com/DJAnderson07/r2Winsteps
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {r2Winsteps: A package for interfacing between R and the Rasch modeling software Winsteps
##     author = {Daniel Anderson},
##     year = {2015},
##     note = {R package version 0.0.0.9000},
##     url = {https://github.com/DJAnderson07/r2Winsteps},
##   }
```

1.1 Installation

For the time being, the *r2Winsteps* package is housed exclusively on *github*. Installation is straightforward via the *devtools* package. If you don't have *devtools* installed, you will need to first run the following:

```
install.packages("devtools")
```

Then, you just need to load the *devtools* package and install *r2Winsteps* directly from github.

```
library(devtools)
install_github("DJAnderson07/r2Winsteps")
```

The package also (obviously) requires a working installation of *Winsteps*, which is designed for Windows. If you are on a Windows machine, you should be good to go from here. However, if you'd like to use a Mac, you can do so through *Wine* (see https://www.winehq.org), a free, open- source "compatibility layer" for running Windows programs on OS X. For help getting Winsteps installed with *Wine*, see the documentation on *Using Wine with Winsteps and the r2Winsteps Package on Macs*.

One last Note: Older versions of Winsteps included different (less) output from the item and person files. The *r2Winsteps* package was built with *Winsteps* Version 3.90. If you have an earlier version of Winsteps, there are workarounds you can use in the options of the `read.pfile()` and `read.ifile()` functions, and you may want to consider modifying the source code for your particular installation.

2 Using the package: Example 1, LSAT data

Now that everything is installed, let's start with a simple example. We'll begin by fitting a Rasch model with dichotomous data, using the supplied LSAT data.

```
library(r2Winsteps)
data(LSAT)
head(LSAT)
```

```
##     ID    Sex Ethnicity Item 1 Item 2 Item 3 Item 4 Item 5
## 1 1086   Male     Black      0      0      0      0      0
```

```
## 2  978   Male     White      0      0      0      0      0
## 3  958   Male    Latino      0      0      0      0      0
## 4  987 Female     White      0      0      0      0      1
## 5 1123 Female     White      0      0      0      0      1
## 6 1004   Male     White      0      0      0      0      1
```

It's generally a good idea to inspect some preliminary data, so we can get an idea of what to expect from the model, and whether the assumptions of the model appear tenable. First, we'll estimate the proportion of examinees responding correctly to the items. Because the items are dichotomous, this is just the mean.

```
apply(LSAT[ ,4:8], 2, mean)
```

```
## Item 1 Item 2 Item 3 Item 4 Item 5
##  0.924  0.709  0.553  0.763  0.870
```

All items appear somewhat easy, but Item 1 is clearly the easiest while Item 3 is clearly the most difficult.

Next, we can compute the point-biserial correlation, by correlating the response vector for each item with a vector of raw scores. We'll compute the raw scores, and then compute the correlations.

```
raw <- rowSums(LSAT[ ,4:8])
round( sapply(LSAT[ ,4:8], function(i) cor(i, raw)), 2)
```

```
## Item 1 Item 2 Item 3 Item 4 Item 5
##   0.36   0.57   0.62   0.53   0.44
```

These are classical test theory indicators of *item discrimination*. The Rasch model assumes essentially equivalent item discrimination (technically 1.0), and so we're looking to see if any items appear wildly different from the others. It's also worth noting that item-fit statistics reported by *Winsteps*, such as the mean square outfit, are produced by evaluating the differences from (essentially) the *average* biserial correlation (see Wu and Adams 2012). These all appear reasonable, so let's go ahead and fit the model.

2.1 Fitting the model

In this case, because the model and data are straightforward, we simply need to call the `runWinsteps()` function, which requires the data be split into a data frame of item responses and a data frame of person demographics.

```
# Split data
itemsLSAT <- LSAT[ ,4:8]
demosLSAT <- LSAT[ ,1:3]

# Run model
parsLSAT <- runWinsteps(itemsLSAT, demosLSAT)
str(parsLSAT)
```

```
## List of 2
##  $ ItemParameters  :'data.frame':    5 obs. of  22 variables:
##   ..$ Entry               : int [1:5] 1 2 3 4 5
##   ..$ Difficulty          : num [1:5] -1.55 0.56 1.63 0.16 -0.81
##   ..$ Status              : int [1:5] 1 1 1 1 1
```

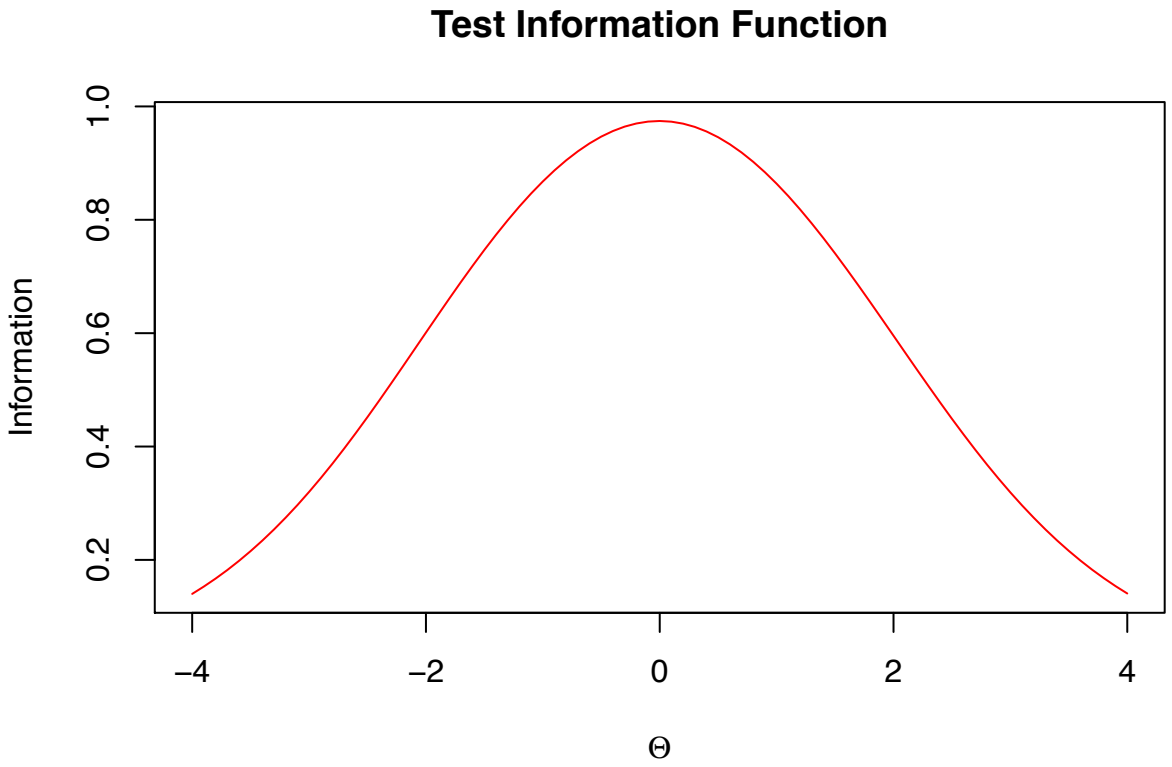
```
##   ..$ Count               : num [1:5] 1000 1000 1000 1000 1000
##   ..$ RawScore            : num [1:5] 924 709 553 763 870
##   ..$ SE                  : num [1:5] 0.13 0.08 0.08 0.09 0.11
##   ..$ Infit               : num [1:5] 1.01 0.98 1.01 0.99 1.01
##   ..$ Infit_Z             : num [1:5] 0.16 -0.5 0.27 -0.28 0.16
##   ..$ Outfit              : num [1:5] 1.05 0.97 1.01 0.98 1.02
##   ..$ Outfit_Z            : num [1:5] 0.41 -0.62 0.15 -0.29 0.27
##   ..$ Displacement        : num [1:5] 0 0 0 0 0
##   ..$ PointMeasureCorr    : num [1:5] 0.35 0.56 0.63 0.53 0.42
##   ..$ Weight              : num [1:5] 1 1 1 1 1
##   ..$ ObservMatch         : num [1:5] 89.6 69.4 65.1 73 83
##   ..$ ExpectMatch         : num [1:5] 89.8 67.9 65.9 72.6 83
##   ..$ PointMeasureExpected: num [1:5] 0.36 0.56 0.64 0.52 0.43
##   ..$ RMSR                : num [1:5] 0.29 0.45 0.45 0.43 0.36
##   ..$ WMLE                : num [1:5] -1.54 0.56 1.63 0.17 -0.8
##   ..$ Group               : int [1:5] 1 1 1 1 1
##   ..$ Model               : Factor w/ 1 level " R": 1 1 1 1 1
##   ..$ Recoding            : Factor w/ 1 level " .": 1 1 1 1 1
##   ..$ ItemID              : Factor w/ 5 levels " Item 1"," Item 2",..: 1 2 3 4 5
##  $ PersonParameters:'data.frame':    1000 obs. of  21 variables:
##   ..$ Entry               : int [1:1000] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Theta               : num [1:1000] -3.23 -3.23 -3.23 -1.72 -1.72 -1.72 -1.72 -1.72 -1.72 -1.72
##   ..$ Status              : int [1:1000] -1 -1 -1 1 1 1 1 1 1 1 ...
##   ..$ Count               : num [1:1000] 5 5 5 5 5 5 5 5 5 5 ...
##   ..$ RawScore            : num [1:1000] 0 0 0 1 1 1 1 1 1 1 ...
##   ..$ SE                  : num [1:1000] 1.93 1.93 1.93 1.21 1.21 1.21 1.21 1.21 1.21 1.21 ...
##   ..$ Infit               : num [1:1000] 1 1 1 1.09 1.09 1.09 1.09 1.09 1.09 1.54 ...
##   ..$ Infit_Z             : num [1:1000] 0 0 0 0.35 0.35 0.35 0.35 0.35 0.35 0.95 ...
##   ..$ Outfit              : num [1:1000] 1 1 1 0.73 0.73 0.73 0.73 0.73 0.73 1.59 ...
##   ..$ Outfit_Z            : num [1:1000] 0 0 0 0.18 0.18 0.18 0.18 0.18 0.18 0.82 ...
##   ..$ Displacement        : num [1:1000] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ PointMeasureCorr    : num [1:1000] 0 0 0 0.37 0.37 0.37 0.37 0.37 0.37 -0.08 ...
##   ..$ Weight              : num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ObservMatch         : num [1:1000] 100 100 100 80 80 80 80 80 80 80 ...
##   ..$ ExpectMatch         : num [1:1000] 100 100 100 80 80 80 80 80 80 80 ...
##   ..$ PointMeasureExpected: num [1:1000] 0 0 0 0.37 0.37 0.37 0.37 0.37 0.37 0.37 ...
##   ..$ RMSR                : num [1:1000] 0 0 0 0.39 0.39 0.39 0.39 0.39 0.39 0.46 ...
##   ..$ WMLE                : num [1:1000] -3.23 -3.23 -3.23 -1.41 -1.41 -1.41 -1.41 -1.41 -1.41 -1.41
##   ..$ ID                  : num [1:1000] 1086 978 958 987 1123 ...
##   ..$ Sex                 : Factor w/ 2 levels " Female "," Male   ": 2 2 2 1 1 2 2 2 1 1 ...
##   ..$ Ethnicity           : Factor w/ 5 levels " Asian"," Black",..: 2 5 3 5 5 5 4 5 2 5 ...
##  - attr(*, "class")= chr "r2Winsteps"
```

The function writes control and data files necessary to run a basic Rasch model for the supplied data. Additionally, a *.bat* file is written to the same directory, which is then executed to call *Winsteps* and run the analysis. By default, the control file is written to have *Winsteps* write out the person and item parameter estimates. These are then read back into R through other functions in the package. Note that one of the things that's nice about this approach, however, is that the person files are read back in with each demographic variable separated into its own column, whereas the Winsteps files contain all person identifying information in a single column.

2.2 Plotting the fitted model

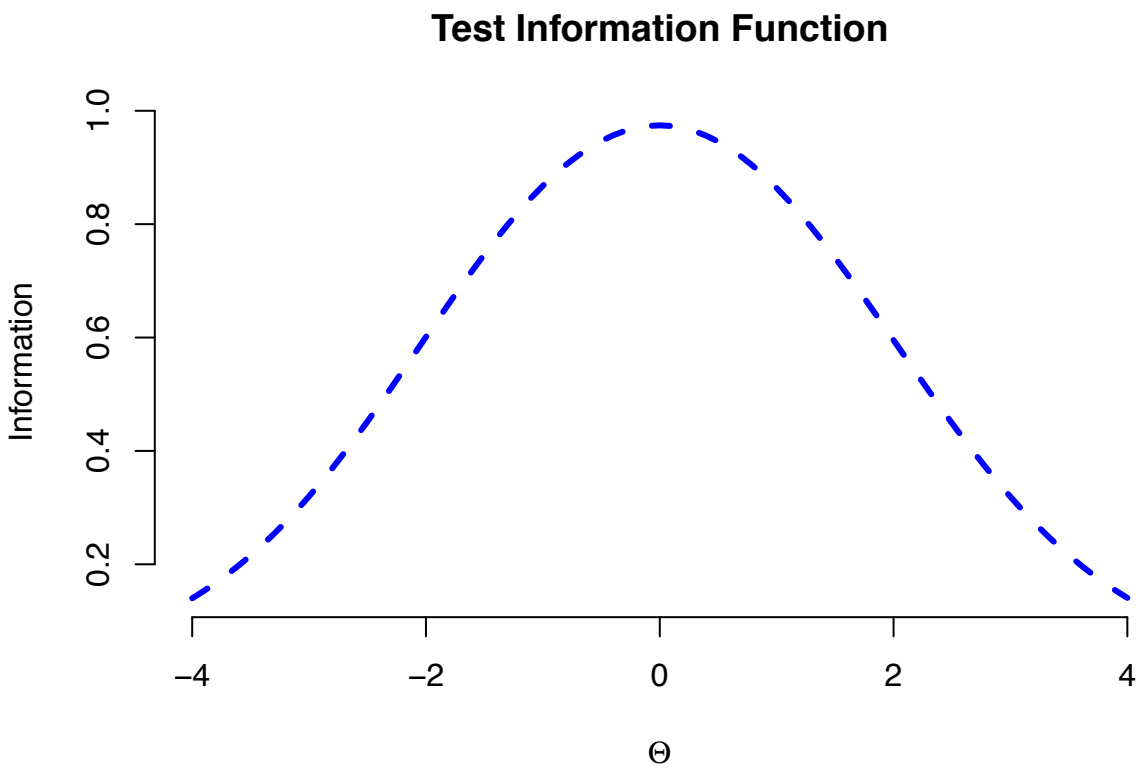
Generally, I like to explore plots of the fitted model before digging in too deep with the parameter estimates. Below, we'll go through some of the built-in plotting features. For example, we can view the test information function through

```
plot(parsLSAT)
```

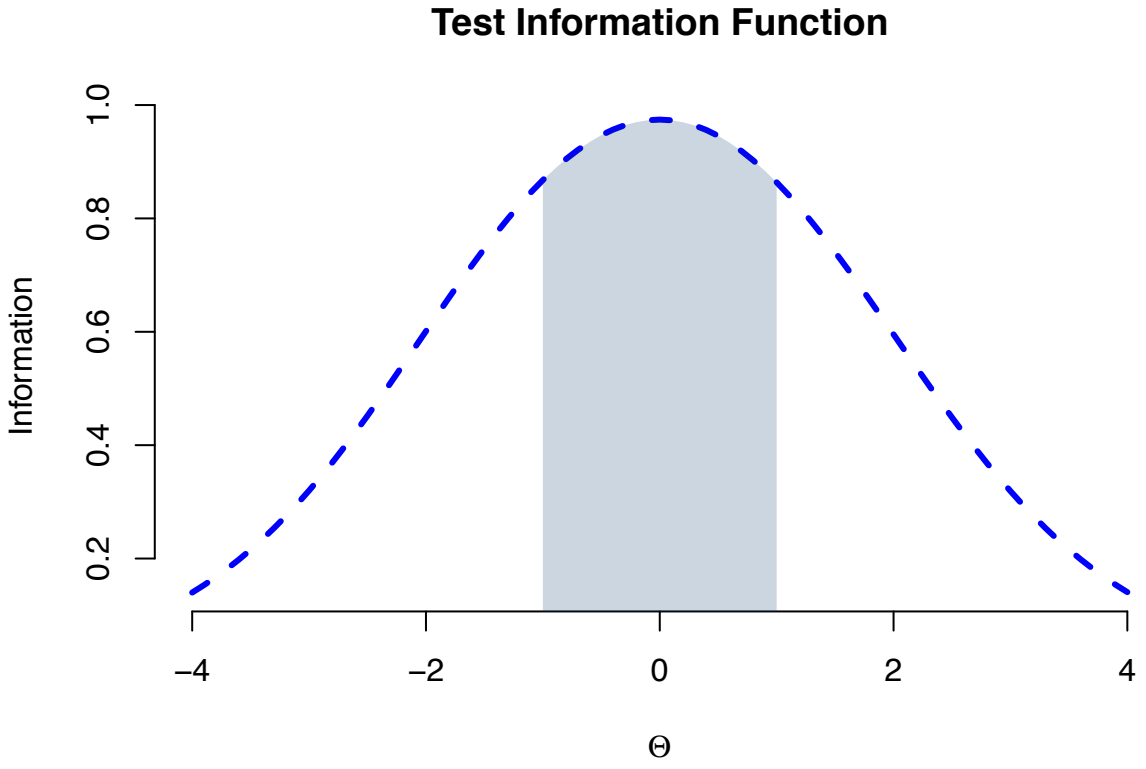


As mentioned in the introduction, this plot is easily modifiable. For example, we can remove the border and change the line color and type with the following code

```
tif <- plot(parsLSAT, store = TRUE, bty = "n", col = "blue", lty = 2, lwd = 3)
```

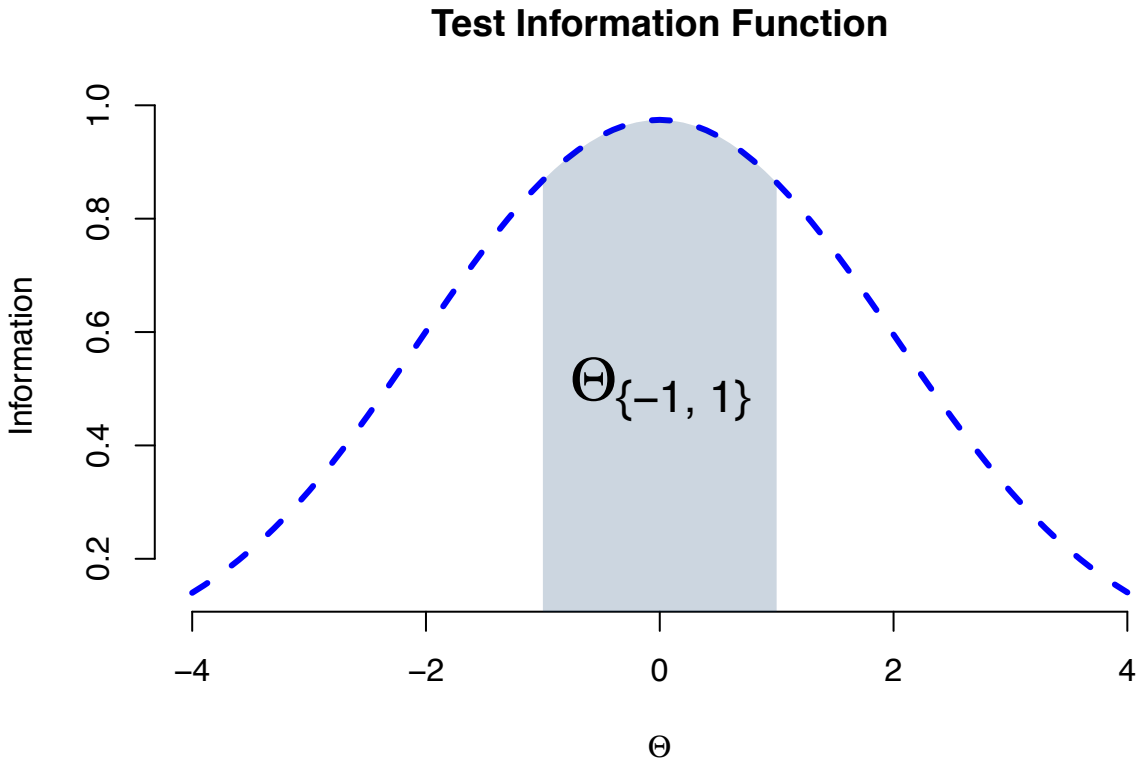


Note that this time I have stored the plot in a new object, `tif`, and included the optional argument `store = TRUE`, which will return the test information values under the range of theta values requested. We can use these for a multitude of purposes, including further plotting. For example, if we wanted to shade the area under the curve corresponding to items from −1, to 1, we could do so as follows



```
theta <- seq(-4, 4, 0.1) # default x-axis

x1 <- theta[which(theta == -1)]
x2 <- theta[which(theta == 1)]

y <- tif[which(theta == -1):which(theta == 1)]

polygon(x = c(x1, seq(x1, x2, length.out = length(y)), x2),
        y = c(0, y, 0),
        col = rgb(0, 0.2, 0.4, 0.2),
        border = FALSE)
```

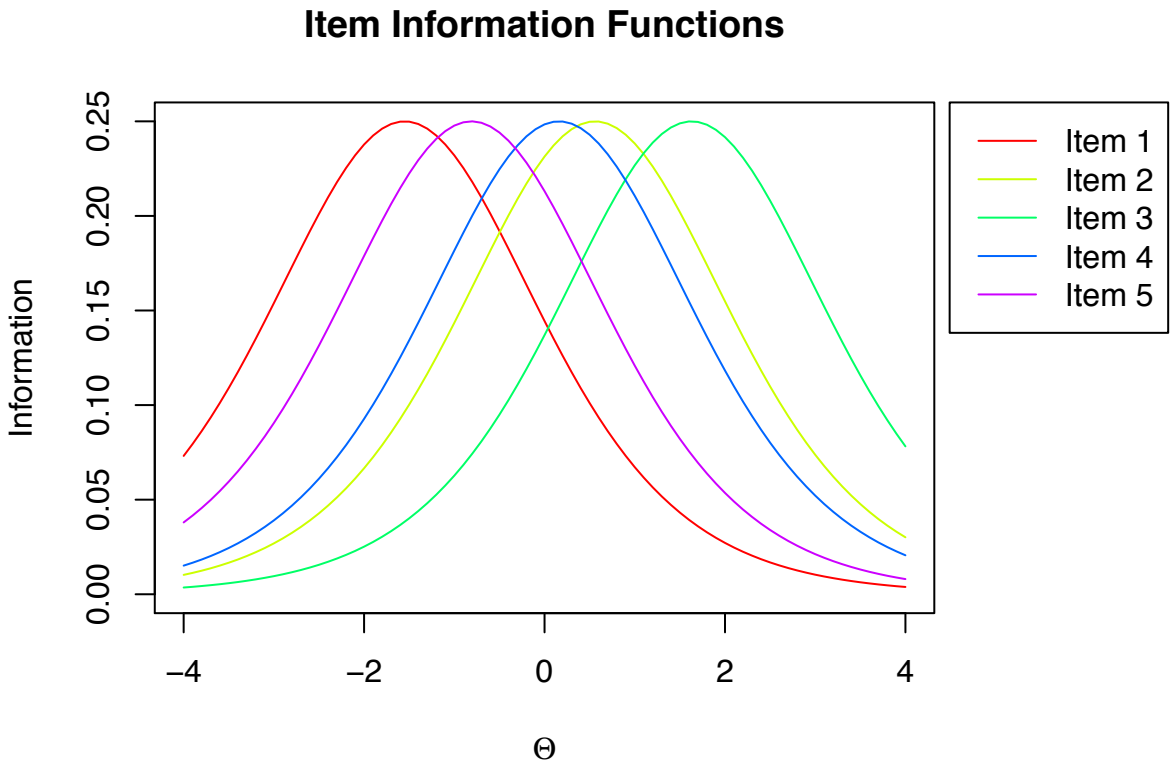
A more realistic situation may be shading the range in which the estimated reliability is above some threshold (e.g., 0.8). However, in this case, with only five items included in the analysis, the estimated reliability is low across the full ability range (with information peaking at a value < 1.0). We could take the plot one step further and annotate it to describe the shaded region.




```
text(0, 0.5, expression(Theta["{-1, 1}"]), cex = 2)
```

This briefly illustrates some of the flexibility and power of plotting in *R*. We can also explore some of the other default plots by including the additional `type` argument to `plot`. For example, if we wanted to view each of the item information functions, we could do so as follows

```
plot(parsLSAT, type = "IIFs")
```



We could alternatively only view a few (or one) item information function by including the additional `itemSelect` argument. For example, to see the curves for only items 1, 3, and 5, we could do so by passing the item locations or names to `itemSelect`

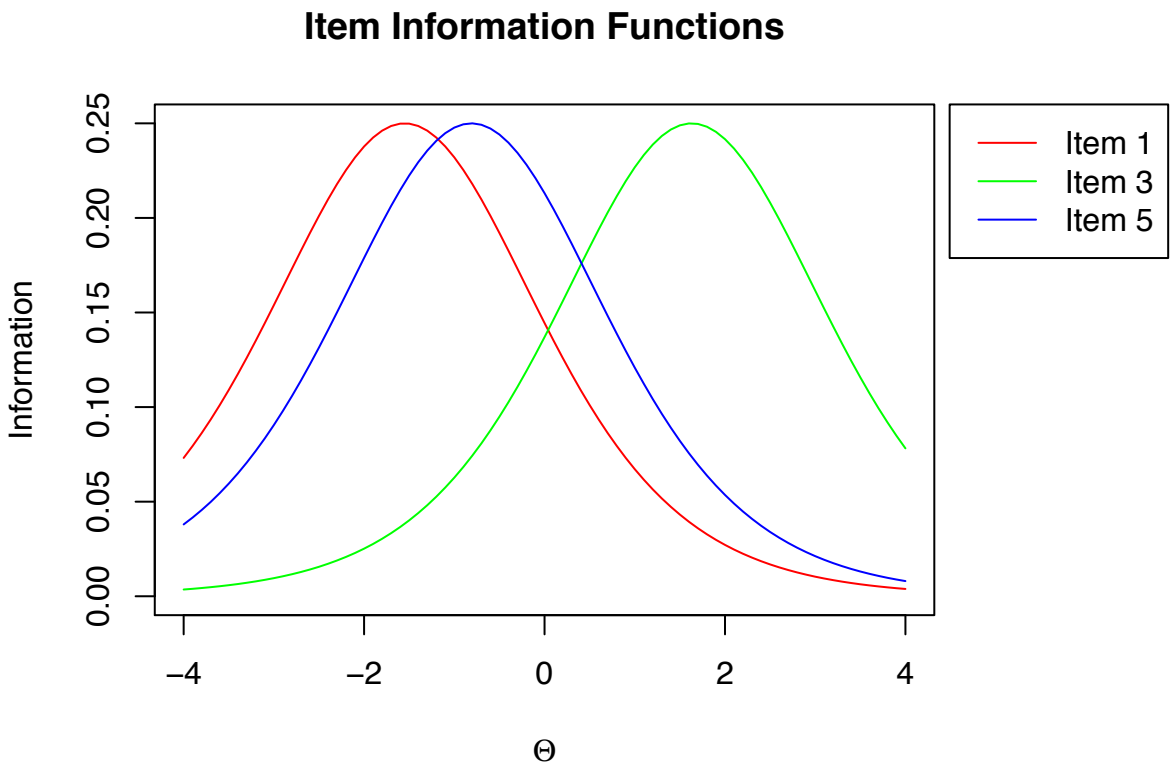
```
plot(parsLSAT, type = "IIFs", itemSelect = c(1, 3, 5))
```

or equivalently

```
plot(parsLSAT, type = "IIFs", itemSelect = c("Item 1", "Item 3", "Item 5"))
```

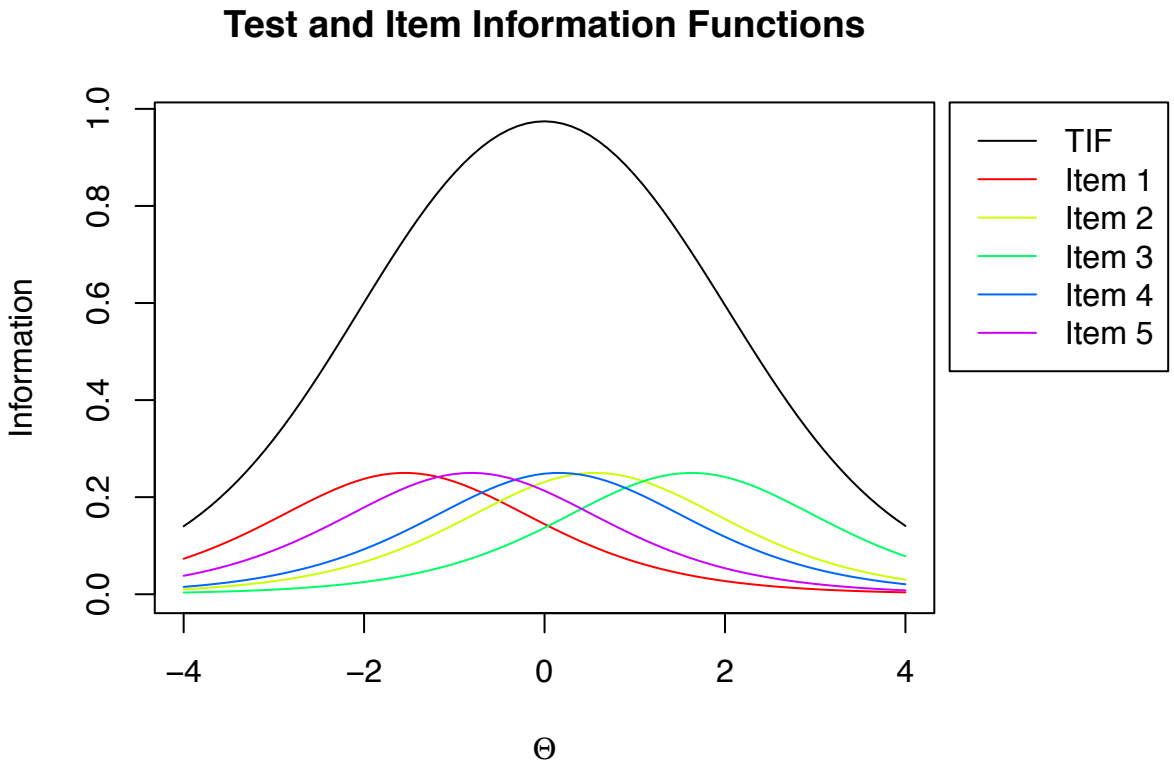
We could also pass the location of items to remove by including the `-` operator. The following is equivalent to the previous two code chunks producing the IIF plot for items 1, 3, and 5.

```
plot(parsLSAT, type = "IIFs", itemSelect = -c(2, 4))
```



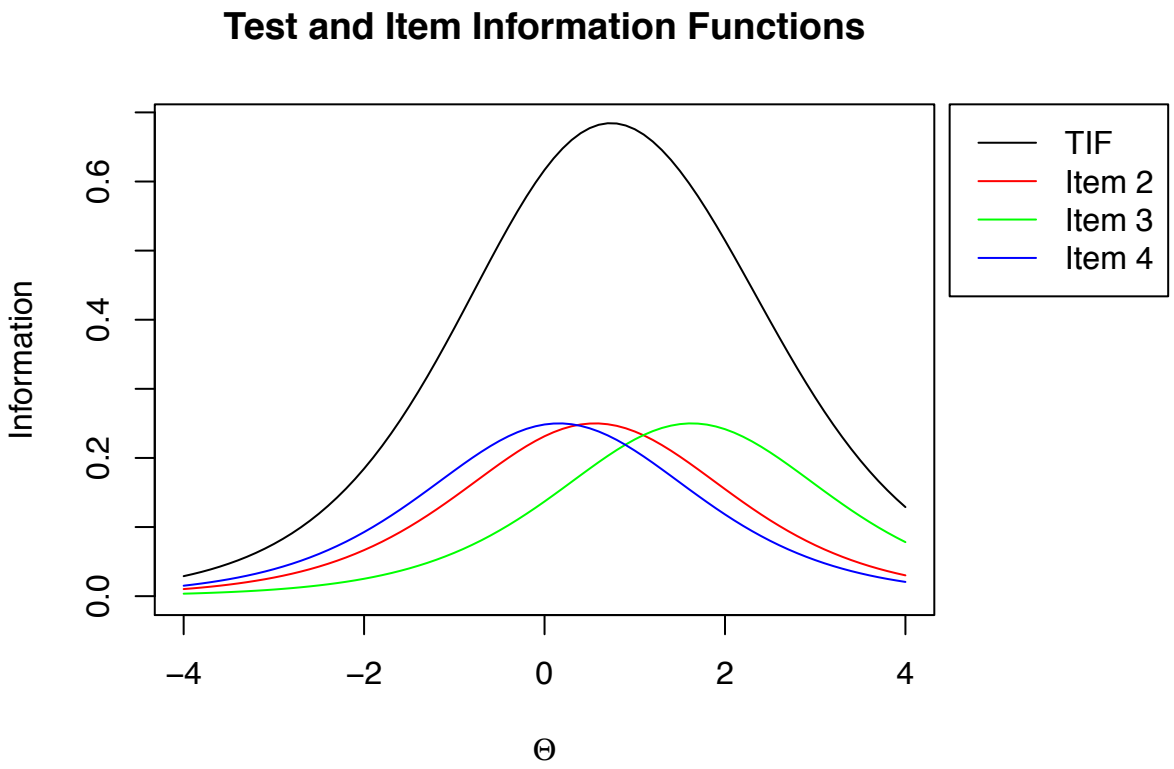
In some contexts, it may be helpful to view the test information function with the item information functions together, particularly if one wants to evaluate how the test information function may change with the exclusion of one or more items. This is possible through `type = "TIF/IIF"`.

```
plot(parsLSAT, type = "TIF/IIF")
```



If we drop the two easiest items, 1 and 5, we can evaluate how the test information function changes.

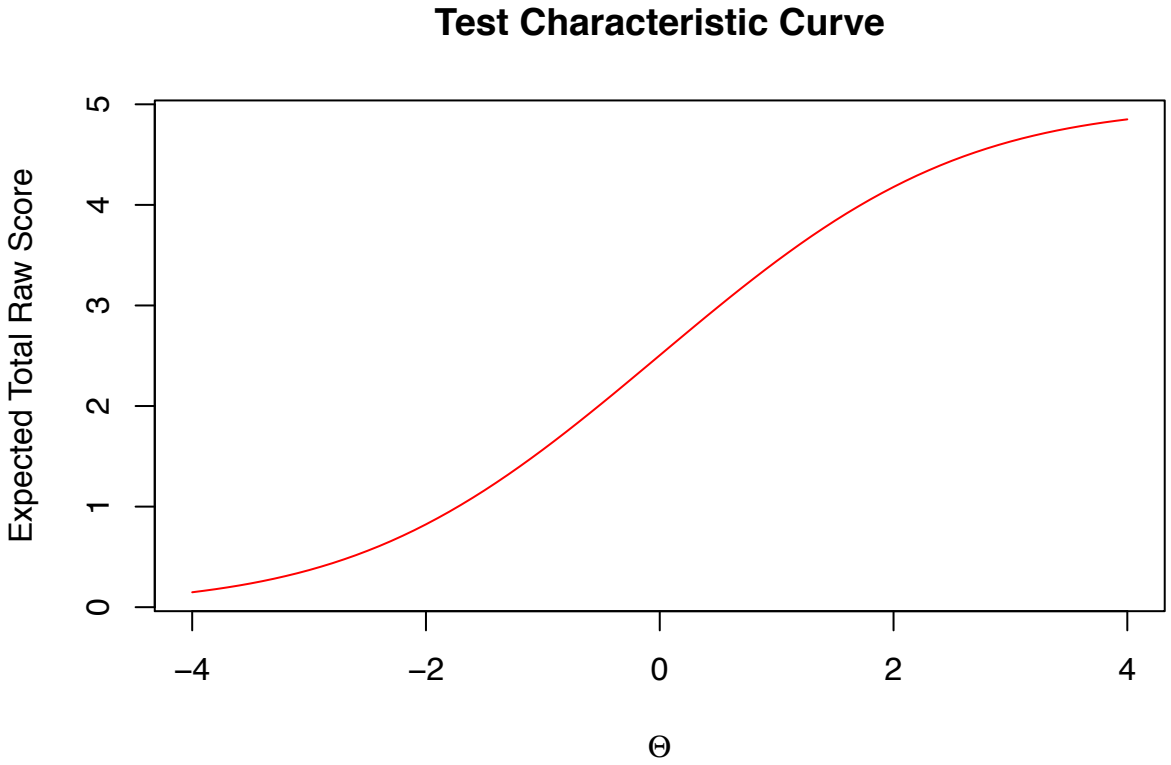
```
plot(parsLSAT, type = "TIF/IIF", itemSelect = -c(1, 5))
```



Predictably, the distribution shifts toward the more difficult end of the scale.

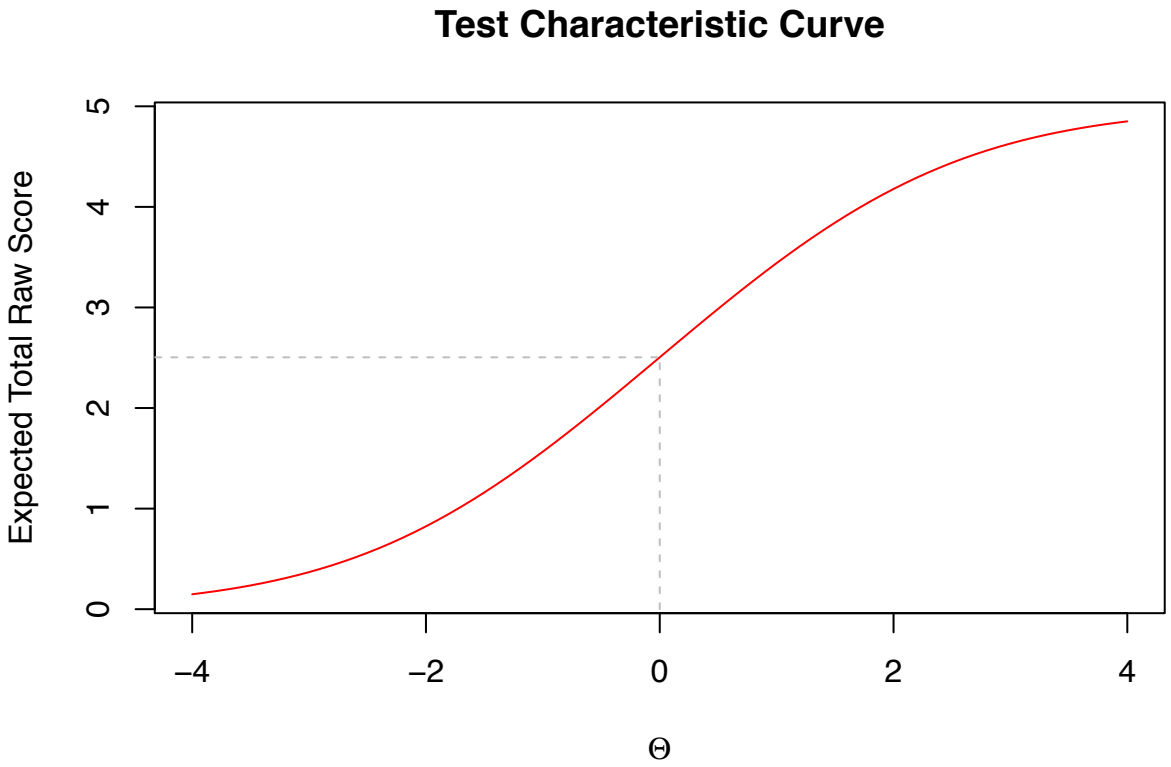
Finally, item and test characteristic curves can also be helpful when evaluating tests, as they provide the expected raw score, given theta. The test characteristic is computed as

```
plot(parsLSAT, type = "TCC")
```



It may be helpful to add some reference lines here. For example, what raw score corresponds with a theta estimate of 0 (or average)? To add this reference line, we may want to rerun the TCC with the optional `store = TRUE` to retain the expected raw scores.

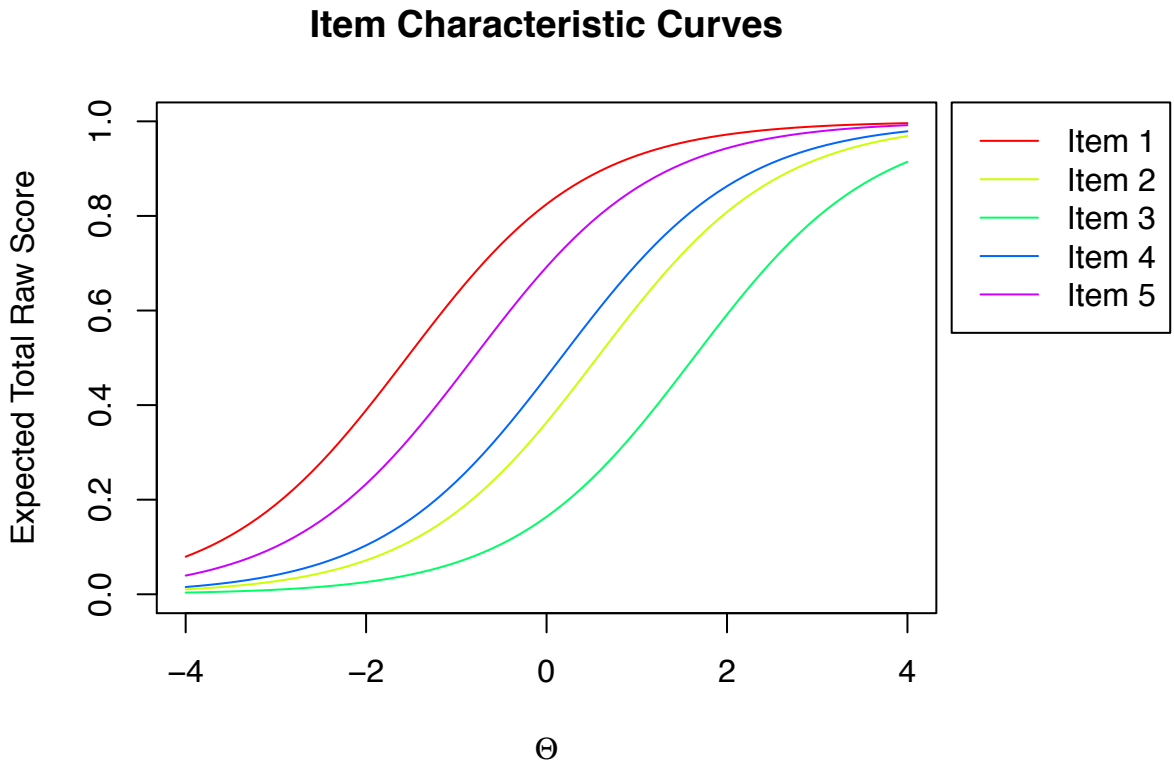
```
tcc <- plot(parsLSAT, type = "TCC", store = TRUE)

segments(x0 = 0, x1 = 0,
            y0 = 0, y1 = tcc[which(theta == 0)],
        col = "gray", lty = 2)

segments(x0 = -5, x1 = 0,
            y0 = tcc[which(theta == 0)], y1 = tcc[which(theta == 0)],
        col = "gray", lty = 2)
```



In other words, examinees of average ability would be expected to score, on average, approximately 2.5 raw score points on the sample of 4 items on the LSAT.

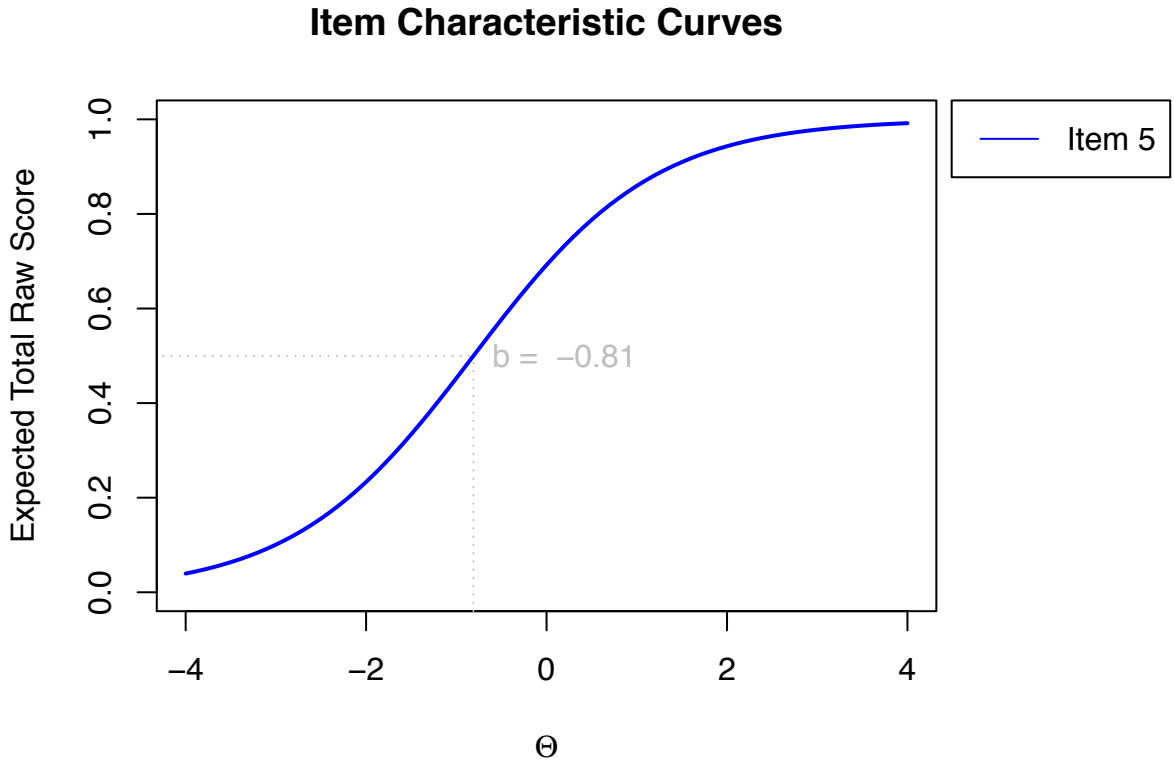
The final built-in plot for dichotomous data that is currently available are the item-characteristic curves. Again, we can plot all, or just some of the items.

```
plot(parsLSAT, type = "ICCs")
```



The inflection point, or the point at which respondents have a 50% probability of correctly responding to the item, corresponds to the item difficulty. An example is shown for Item 5, below.

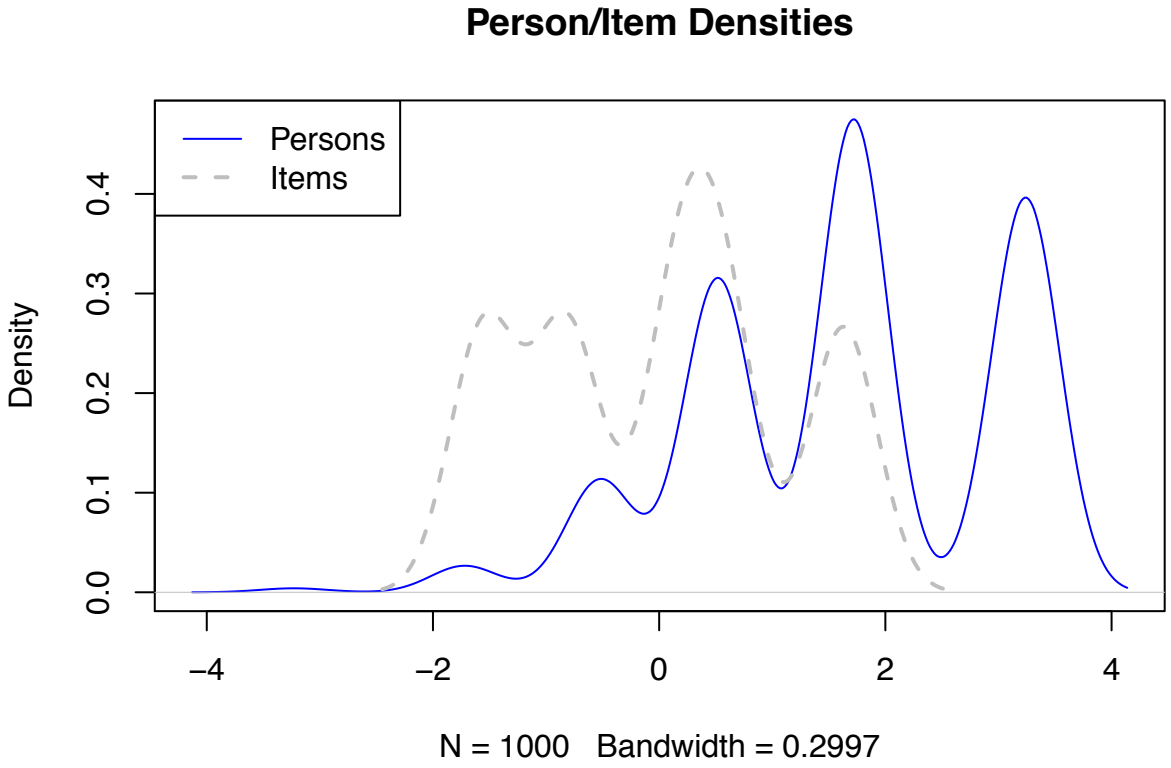
```
itm5 <- plot(parsLSAT, type = "ICCs", itemSelect = c("Item 5"), store = TRUE,
    col = "blue", lwd = 2)

i5b <- parsLSAT$ItemParameters$Difficulty[5]

segments(x0 = -4.25, x1 = i5b,
            y0 = 0.5, y1 = 0.5, col = "gray", lty = 3)
segments(x0 = i5b, x1 = i5b,
            y0 = -0.04, y1 = 0.5, col = "gray", lty = 3)

text(x = 0.2, y = 0.5, paste("b = ", round(i5b, 3)), col = "gray")
```



There are a couple of things to note here. First is that I extracted the item difficulty, `i5b`, prior to producing the line segments (we'll talk more about working with the actual parameter estimates momentarily). Second is that I had to work with a little bit of trial and error to get the lines to extend directly to the margins (e.g., `y0 = -0.04`). I then annotated the plot with the `text` function to print the estimated item difficulty.

It's important to note that the above only illustrates the built-in plotting features (which are still in development and expanding). However, the flexibility of *R* allows for essentially any plot can be produced. For example, it may be helpful to view the person and item distributions together, through overlayed density plots (which may be incorporated as a default plot at a later date). Below is one method for producing such a plot

```
personD <- density(parsLSAT$PersonParameters$Theta)
itemD <- density(parsLSAT$ItemParameters$Difficulty, bw = personD$bw)

plot(personD, col = "blue",
    main = "Person/Item Densities")
lines(itemD, col = "gray", lty = 2, lwd = 2)
legend("topleft",
    c("Persons", "Items"),
    col = c("blue", "gray"),
    lty = c(1, 2),
    lwd = c(1, 2))
```



Note that in this case the densities are rather "jagged". This is, again, because only 5 items were used in the model. However, we do see that the person distribution is shifted slightly above the item distribution.

2.3 Parameter Estimates

Parameter estimates from a fitted model are returned in the form of a named *list*. When a model with dichotomous data is estimated, the list will contain two data frames: the item and person parameters. However, if a polytomous model is estimated (next example), the list will also contain the *structure* file, which are the Rasch-Andrich thresholds. In this case, only the item and person parameters are reported.

```
length(parsLSAT)
```

```
## [1] 2
```

```
names(parsLSAT)
```

```
## [1] "ItemParameters"   "PersonParameters"
```

We can now inspect the results by subsetting the list. For example, we can look at items by

```
itmsLSAT <- parsLSAT$ItemParameters
itmsLSAT
```

```
##   Entry Difficulty Status Count RawScore   SE Infit Infit_Z Outfit
## 1     1      -1.55      1  1000      924 0.13  1.01    0.16   1.05
## 2     2       0.56      1  1000      709 0.08  0.98   -0.50   0.97
```

```
## 3     3       1.63      1  1000      553 0.08  1.01    0.27   1.01
## 4     4       0.16      1  1000      763 0.09  0.99   -0.28   0.98
## 5     5      -0.81      1  1000      870 0.11  1.01    0.16   1.02
##   Outfit_Z Displacement PointMeasureCorr Weight ObservMatch ExpectMatch
## 1     0.41            0             0.35      1        89.6        89.8
## 2    -0.62            0             0.56      1        69.4        67.9
## 3     0.15            0             0.63      1        65.1        65.9
## 4    -0.29            0             0.53      1        73.0        72.6
## 5     0.27            0             0.42      1        83.0        83.0
##   PointMeasureExpected RMSR  WMLE Group Model Recoding  ItemID
## 1                 0.36 0.29 -1.54     1     R        .  Item 1
## 2                 0.56 0.45  0.56     1     R        .  Item 2
## 3                 0.64 0.45  1.63     1     R        .  Item 3
## 4                 0.52 0.43  0.17     1     R        .  Item 4
## 5                 0.43 0.36 -0.80     1     R        .  Item 5
```

In this case, the test was really small (only five items) and so we can pretty much view everything we need here. In other cases, the output may be too large, and so further subsetting can be helpful. Let's look at the person parameters

```
persLSAT <- parsLSAT$PersonParameters
head(persLSAT)
```

```
##   Entry Theta Status Count RawScore   SE Infit Infit_Z Outfit Outfit_Z
## 1     1 -3.23     -1     5        0 1.93  1.00    0.00   1.00     0.00
## 2     2 -3.23     -1     5        0 1.93  1.00    0.00   1.00     0.00
## 3     3 -3.23     -1     5        0 1.93  1.00    0.00   1.00     0.00
## 4     4 -1.72      1     5        1 1.21  1.09    0.35   0.73     0.18
## 5     5 -1.72      1     5        1 1.21  1.09    0.35   0.73     0.18
## 6     6 -1.72      1     5        1 1.21  1.09    0.35   0.73     0.18
##   Displacement PointMeasureCorr Weight ObservMatch ExpectMatch
## 1            0             0.00      1         100         100
## 2            0             0.00      1         100         100
## 3            0             0.00      1         100         100
## 4            0             0.37      1          80          80
## 5            0             0.37      1          80          80
## 6            0             0.37      1          80          80
##   PointMeasureExpected RMSR  WMLE   ID      Sex Ethnicity
## 1                 0.00 0.00 -3.23 1086  Male        Black
## 2                 0.00 0.00 -3.23  978  Male        White
## 3                 0.00 0.00 -3.23  958  Male       Latino
## 4                 0.37 0.39 -1.41  987  Female      White
## 5                 0.37 0.39 -1.41 1123  Female      White
## 6                 0.37 0.39 -1.41 1004  Male        White
```

The first six rows are shown above out of the 1,000 respondents. The mean square outfit is often a useful indicator not only of item functioning, but also of person fit to the model expectations (with high values indicating unexpected responses). Let's calculate the percentage of persons with a mean square outfit between 0.7 and 1.3 (these are fairly arbitrary cutoffs).

```
nrow( subset(persLSAT, Outfit >= 0.7 & Outfit <= 1.3) ) / nrow(persLSAT)
```

```
## [1] 0.538
```

The above indicates that aprroximately 54% of respondents did not have unexpected responses (given the range we specified as acceptable).

We may also want to compile some of this information into a table. We can use the *knitr* package (Xie 2015) to produce tables quickly (other packages, like *xtable* are more flexible, but require more investment (Dahl 2014)). For example, we can first create a data frame of item parameter estimates, containing only the estimates we'd like to report.

```
itmTbl <- itmsLSAT[c("ItemID", "Difficulty", "SE", "Infit", "Outfit",
            "PointMeasureCorr")]
```

We can then call the `kable` function from `knitr` to produce a nice looking table.

```
library(knitr)
kable(itmTbl,
        row.names = FALSE,
        align = c("l", rep("c", ncol(itmTbl) -1)),
        caption = "Summary of Item Parameter Estimates")
```

Table 1: Summary of Item Parameter Estimates

ItemID	Difficulty	SE	Infit	Outfit	PointMeasureCorr
Item 1	-1.55	0.13	1.01	1.05	0.35
Item 2	0.56	0.08	0.98	0.97	0.56
Item 3	1.63	0.08	1.01	1.01	0.63
Item 4	0.16	0.09	0.99	0.98	0.53
Item 5	-0.81	0.11	1.01	1.02	0.42

Similarly, we may want to produce a summary table of the person parameters. We'll produce a summary table that reports the estimates for each unique theta estimate (corresponding to each possible raw score).

```
persSummary <- persLSAT[!duplicated(persLSAT$Theta), ]
persSummary <- persSummary[c("RawScore", "Theta", "SE")]
```

This time, however, we'll use *xtable* to make the table a little fancier.

```
library(xtable)

names(persSummary) <- c("Raw Score", "$\\theta$", "$SE(\\theta)$")

persTab <- xtable(persSummary,
    caption = "Raw Score to Theta Mapping",
    align = c("l", "c", "c", "c"))

options(xtable.comment = FALSE)
print(persTab, caption.placement = "top",
               include.rownames = FALSE,
               sanitize.text.function = function(x) x)
```

These are both relatively simple examples, but demonstrate what was discussed in the introduction about producing dynamic documents. If we changed the data or model, these tables would be updated automatically, which helps efficiency and can reduce errors.

Table 2: Raw Score to Theta Mapping

Raw Score	θ	$SE(\theta)$
0.00	-3.23	1.93
1.00	-1.72	1.21
2.00	-0.52	1.03
3.00	0.52	1.03
4.00	1.72	1.21
5.00	3.24	1.94

2.4 Diving Deeper

In some ways, *r2Winsteps* is limited relative to interacting directly with *Winsteps*, and it will likely never have the full features of *Winsteps* built-in (again, the purpose of the package is to provide an interface for commonly applied models). However, *r2Winsteps* may still be useful by providing an automated "starting point" for control and data files, particularly if you are already working within *R*. The wrapper functions `batch.pfile` and `batch.ifile` (and `batch.sfile` if a polytomous model is fit) can also be used to read the parameter estimates back into R, where the results can be explored further. In what follows, each of the steps taken by the `runWinsteps` function is shown so that users can (a) understand the function better and, more importantly, (b) modify specific aspects of the control file to suit their needs.

The `runWinsteps` function begins by writing control and data files for the given data supplied, via the `r2Winsteps` function.

```
r2Winsteps(itemsLSAT, demosLSAT)
```

This writes control and data files to the working directory, such as those shown below.

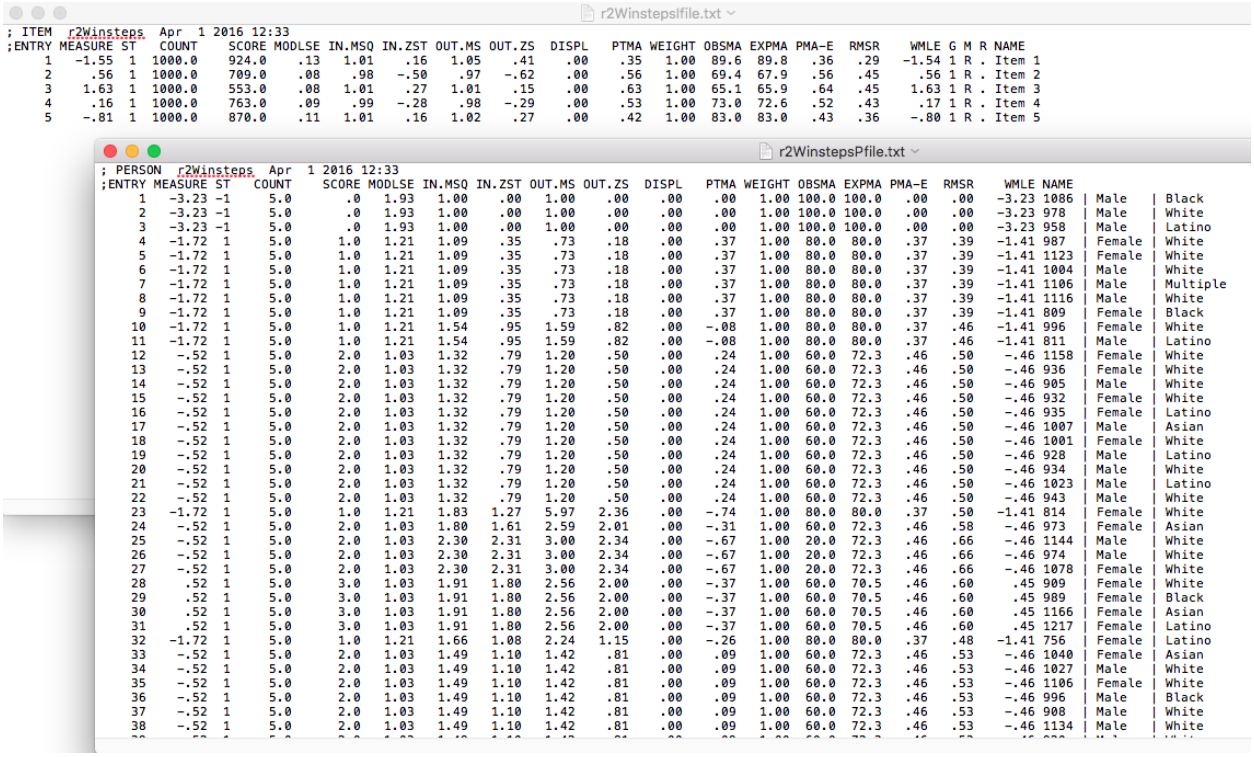
r2WinstepsCntrl.txt

```
&INST
TITLE = r2Winsteps
DATA =  r2WinstepsDta.txt
ITEM1 =  26
NI =  5
NAME1 = 1
NAMLEN =  24
XWIDE = 1
CODES = 0 1
TOTALSCORE = YES
UDECIMALS = 2
IFILE =  r2WinstepsIfile.txt
PFILE =  r2WinstepsPfile.txt
;
;
;
;
@ID = 1E5
@Sex = 6E11
@Ethnicity = 12E19
&End
Item 1
Item 2
Item 3
Item 4
Item 5
END NAMES
```

r2WinstepsDta.txt

```
1086 | Male   | Black    00000
978  | Male   | White    00000
958  | Male   | Latino   00000
987  | Female | White    00001
1123 | Female | White    00001
1004 | Male   | White    00001
1106 | Male   | Multiple 00001
1116 | Male   | White    00001
809  | Female | Black    00001
996  | Female | White    00010
811  | Male   | Latino   00010
1158 | Female | White    00011
936  | Female | White    00011
905  | Male   | White    00011
932  | Female | White    00011
935  | Female | Latino   00011
1007 | Male   | Asian    00011
1001 | Female | White    00011
928  | Male   | Latino   00011
934  | Male   | White    00011
1023 | Male   | Latino   00011
943  | Male   | White    00011
814  | Female | White    00100
973  | Female | Asian    00101
1144 | Male   | White    00110
974  | Male   | White    00110
1078 | Female | White    00110
909  | Female | White    00111
```

The control file could then be modified to accommodate any options or models that can be fit by Winsteps. After running the model, item and person files should be written to the working directory, as shown below.



r2WinstepsIfile.txt

```
; ITEM  r2Winsteps  Apr  1 2016 12:33
;ENTRY MEASURE ST   COUNT    SCORE MODLSE IN.MSQ IN.ZST OUT.MS OUT.ZS  DISPL   PTMA WEIGHT OBSMA EXPMA PMA-E  RMSR    WMLE G M R NAME
    1   -1.55  1  1000.0    924.0   .13   1.01    .16   1.05    .41   .00   .35   1.00  89.6  89.8   .36   .29   -1.54 1 R . Item 1
    2    .56  1  1000.0    709.0   .08    .98   -.50    .97   -.62   .00   .56   1.00  69.4  67.9   .56   .45    .56 1 R . Item 2
    3   1.63  1  1000.0    553.0   .08   1.01    .27   1.01    .15   .00   .63   1.00  65.1  65.9   .64   .45   1.63 1 R . Item 3
    4    .16  1  1000.0    763.0   .09    .99   -.28    .98   -.29   .00   .53   1.00  73.0  72.6   .52   .43    .17 1 R . Item 4
    5   -.81  1  1000.0    870.0   .11   1.01    .16   1.02    .27   .00   .42   1.00  83.0  83.0   .43   .36   -.80 1 R . Item 5
```

r2WinstepsPfile.txt

```
; PERSON  r2Winsteps  Apr  1 2016 12:33
;ENTRY MEASURE ST   COUNT    SCORE MODLSE IN.MSQ IN.ZST OUT.MS OUT.ZS  DISPL   PTMA WEIGHT OBSMA EXPMA PMA-E  RMSR    WMLE NAME
    1   -3.23 -1    5.0       .0   1.93   1.00    .00   1.00    .00   .00   .00   1.00 100.0 100.0   .00   .00   -3.23 1086 | Male   | Black
    2   -3.23 -1    5.0       .0   1.93   1.00    .00   1.00    .00   .00   .00   1.00 100.0 100.0   .00   .00   -3.23 978  | Male   | White
    3   -3.23 -1    5.0       .0   1.93   1.00    .00   1.00    .00   .00   .00   1.00 100.0 100.0   .00   .00   -3.23 958  | Male   | Latino
    4   -1.72  1    5.0      1.0   1.21   1.09    .35    .73    .18   .00   .37   1.00  80.0  80.0   .37   .39   -1.41 987  | Female | White
    5   -1.72  1    5.0      1.0   1.21   1.09    .35    .73    .18   .00   .37   1.00  80.0  80.0   .37   .39   -1.41 1123 | Female | White
    6   -1.72  1    5.0      1.0   1.21   1.09    .35    .73    .18   .00   .37   1.00  80.0  80.0   .37   .39   -1.41 1004 | Male   | White
    7   -1.72  1    5.0      1.0   1.21   1.09    .35    .73    .18   .00   .37   1.00  80.0  80.0   .37   .39   -1.41 1106 | Male   | Multiple
    8   -1.72  1    5.0      1.0   1.21   1.09    .35    .73    .18   .00   .37   1.00  80.0  80.0   .37   .39   -1.41 1116 | Male   | White
    9   -1.72  1    5.0      1.0   1.21   1.09    .35    .73    .18   .00   .37   1.00  80.0  80.0   .37   .39   -1.41 809  | Female | Black
   10   -1.72  1    5.0      1.0   1.21   1.54    .95   1.59    .82   .00  -.08   1.00  80.0  80.0   .37   .46   -1.41 996  | Female | White
   11   -1.72  1    5.0      1.0   1.21   1.54    .95   1.59    .82   .00  -.08   1.00  80.0  80.0   .37   .46   -1.41 811  | Male   | Latino
   12    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 1158 | Female | White
   13    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 936  | Female | White
   14    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 905  | Male   | White
   15    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 932  | Female | White
   16    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 935  | Female | Latino
   17    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 1007 | Male   | Asian
   18    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 1001 | Female | White
   19    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 928  | Male   | Latino
   20    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 934  | Male   | White
   21    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 1023 | Male   | Latino
   22    -.52  1    5.0      2.0   1.03   1.32    .79   1.20    .50   .00   .24   1.00  60.0  72.3   .46   .50    -.46 943  | Male   | White
   23   -1.72  1    5.0      1.0   1.21   1.83   1.27   5.97   2.36   .00  -.74   1.00  80.0  80.0   .37   .50   -1.41 814  | Female | White
   24    -.52  1    5.0      2.0   1.03   1.80   1.61   2.59   2.01   .00  -.31   1.00  60.0  72.3   .46   .58    -.46 973  | Female | Asian
   25    -.52  1    5.0      2.0   1.03   2.30   2.31   3.00   2.34   .00  -.67   1.00  20.0  72.3   .46   .66    -.46 1144 | Male   | White
   26    -.52  1    5.0      2.0   1.03   2.30   2.31   3.00   2.34   .00  -.67   1.00  20.0  72.3   .46   .66    -.46 974  | Male   | White
   27    -.52  1    5.0      2.0   1.03   2.30   2.31   3.00   2.34   .00  -.67   1.00  20.0  72.3   .46   .66    -.46 1078 | Female | White
   28     .52  1    5.0      3.0   1.03   1.91   1.80   2.56   2.00   .00  -.37   1.00  60.0  70.5   .46   .60     .45 909  | Female | White
   29     .52  1    5.0      3.0   1.03   1.91   1.80   2.56   2.00   .00  -.37   1.00  60.0  70.5   .46   .60     .45 989  | Female | Black
   30     .52  1    5.0      3.0   1.03   1.91   1.80   2.56   2.00   .00  -.37   1.00  60.0  70.5   .46   .60     .45 1166 | Female | Asian
   31     .52  1    5.0      3.0   1.03   1.91   1.80   2.56   2.00   .00  -.37   1.00  60.0  70.5   .46   .60     .45 1217 | Female | Latino
   32   -1.72  1    5.0      1.0   1.21   1.66   1.08   2.24   1.15   .00  -.26   1.00  80.0  80.0   .37   .48   -1.41 756  | Female | Latino
   33    -.52  1    5.0      2.0   1.03   1.49   1.10   1.42    .81   .00   .09   1.00  60.0  72.3   .46   .53    -.46 1040 | Female | Asian
   34    -.52  1    5.0      2.0   1.03   1.49   1.10   1.42    .81   .00   .09   1.00  60.0  72.3   .46   .53    -.46 1027 | Male   | White
   35    -.52  1    5.0      2.0   1.03   1.49   1.10   1.42    .81   .00   .09   1.00  60.0  72.3   .46   .53    -.46 1106 | Female | White
   36    -.52  1    5.0      2.0   1.03   1.49   1.10   1.42    .81   .00   .09   1.00  60.0  72.3   .46   .53    -.46 996  | Male   | Black
   37    -.52  1    5.0      2.0   1.03   1.49   1.10   1.42    .81   .00   .09   1.00  60.0  72.3   .46   .53    -.46 908  | Male   | White
   38    -.52  1    5.0      2.0   1.03   1.49   1.10   1.42    .81   .00   .09   1.00  60.0  72.3   .46   .53    -.46 1134 | Male   | White
```

We can read the person file back into *R* as follows

```
pers <- batch.pfile(dir = "./assets/data/")
head(pers)
```

```
##   Entry Theta Status Count RawScore   SE Infit Infit_Z Outfit Outfit_Z
## 1     1 -3.23     -1     5        0 1.93  1.00    0.00   1.00      0.00
## 2     2 -3.23     -1     5        0 1.93  1.00    0.00   1.00      0.00
## 3     3 -3.23     -1     5        0 1.93  1.00    0.00   1.00      0.00
## 4     4 -1.72      1     5        1 1.21  1.09    0.35   0.73      0.18
## 5     5 -1.72      1     5        1 1.21  1.09    0.35   0.73      0.18
## 6     6 -1.72      1     5        1 1.21  1.09    0.35   0.73      0.18
##   Displacement PointMeasureCorr Weight ObservMatch ExpectMatch
## 1            0             0.00      1         100         100
## 2            0             0.00      1         100         100
## 3            0             0.00      1         100         100
## 4            0             0.37      1          80          80
## 5            0             0.37      1          80          80
## 6            0             0.37      1          80          80
##   PointMeasureExpected RMSR  WMLE  v20      v21     v22
## 1                0.00 0.00 -3.23 1086  Male      Black
## 2                0.00 0.00 -3.23  978  Male      White
## 3                0.00 0.00 -3.23  958  Male     Latino
## 4                0.37 0.39 -1.41  987  Female    White
## 5                0.37 0.39 -1.41 1123  Female    White
## 6                0.37 0.39 -1.41 1004  Male      White
```

There's a few things to note here. First, the function name itself may not be very intuitive. Why not `read.pfile`? The answer is that the function was designed to be flexible enough to read 1...n person files into R. If more than one person file is read in, then the function returns a list of the data frames. The second is that we seem to have lost the names of our demographic variables. This is because the function does not "know", by itself, what those variables represent (although `runWinsteps` does, because it was supplied the original data). We can provide the function with a vector of names, and they will be input.

```
pers <- batch.pfile(demNameL = list(c("ID", "Sex", "Ethnicity")),
            dir = "./assets/data/")
head(pers)
```

```
##   Entry Theta Status Count RawScore   SE Infit Infit_Z Outfit Outfit_Z
## 1     1 -3.23     -1     5        0 1.93  1.00    0.00   1.00     0.00
## 2     2 -3.23     -1     5        0 1.93  1.00    0.00   1.00     0.00
## 3     3 -3.23     -1     5        0 1.93  1.00    0.00   1.00     0.00
## 4     4 -1.72      1     5        1 1.21  1.09    0.35   0.73     0.18
## 5     5 -1.72      1     5        1 1.21  1.09    0.35   0.73     0.18
## 6     6 -1.72      1     5        1 1.21  1.09    0.35   0.73     0.18
##   Displacement PointMeasureCorr Weight ObservMatch ExpectMatch
## 1            0             0.00      1         100         100
## 2            0             0.00      1         100         100
## 3            0             0.00      1         100         100
## 4            0             0.37      1          80          80
## 5            0             0.37      1          80          80
## 6            0             0.37      1          80          80
##   PointMeasureExpected RMSR  WMLE   ID      Sex Ethnicity
## 1                 0.00 0.00 -3.23 1086  Male        Black
## 2                 0.00 0.00 -3.23  978  Male        White
## 3                 0.00 0.00 -3.23  958  Male       Latino
## 4                 0.37 0.39 -1.41  987  Female      White
## 5                 0.37 0.39 -1.41 1123  Female      White
## 6                 0.37 0.39 -1.41 1004  Male        White
```

Importantly, the vector of demographic names must be provided in a list. This is so that multiple vectors of names can be provided to the function. By default, the `batch.pfile` function will try to read in all the files with the pattern "Pfile" in the directory provided (or the current working directory if none is provided). Different patterns can be provided, via the `pat` argument. Alternatively, a vector of file names can be provided, and only those files will be read in.

The item files can be read in equivalently

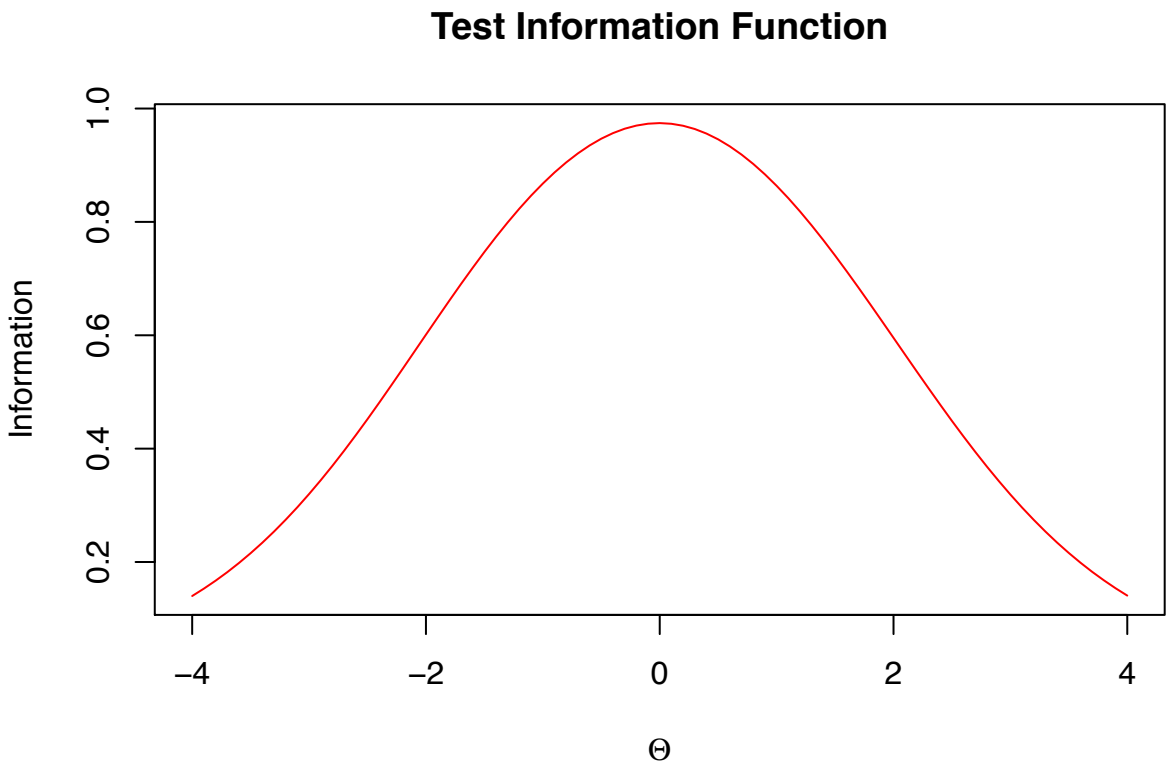
```
itms <- batch.ifile(dir = "./assets/data/")
head(itms)
```

```
##   Entry Difficulty Status Count RawScore   SE Infit Infit_Z Outfit
## 1     1      -1.55      1  1000      924 0.13  1.01    0.16   1.05
## 2     2       0.56      1  1000      709 0.08  0.98   -0.50   0.97
## 3     3       1.63      1  1000      553 0.08  1.01    0.27   1.01
## 4     4       0.16      1  1000      763 0.09  0.99   -0.28   0.98
## 5     5      -0.81      1  1000      870 0.11  1.01    0.16   1.02
##   Outfit_Z Displacement PointMeasureCorr Weight ObservMatch ExpectMatch
## 1     0.41            0             0.35      1        89.6        89.8
## 2    -0.62            0             0.56      1        69.4        67.9
```

```
## 3     0.15            0            0.63      1        65.1        65.9
## 4    -0.29            0            0.53      1        73.0        72.6
## 5     0.27            0            0.42      1        83.0        83.0
##   PointMeasureExpected RMSR  WMLE Group Model Recoding  ItemID
## 1                 0.36 0.29 -1.54     1     R        .  Item 1
## 2                 0.56 0.45  0.56     1     R        .  Item 2
## 3                 0.64 0.45  1.63     1     R        .  Item 3
## 4                 0.52 0.43  0.17     1     R        .  Item 4
## 5                 0.43 0.36 -0.80     1     R        .  Item 5
```

At this point, we pretty much are back to where we were after running `runWinsteps`. However, if we want to utilize the plotting functions, we need to combine the results into a list with the names `ItemParameters` and `PersonParameters`, and specify it as of class `r2Winsteps`. All the default plotting functions will then be available.

```
l <- list(ItemParameters = itms, PersonParameters = pers)
class(l) <- "r2Winsteps"
plot(l)
```



This facilitates increased flexibility, while maintaining a method for accessing the default plots. The `runWinsteps` function does all of the above automatically. Note that it's also possible to keep any of the intermediary files, such as the item and person parameters (in text files) through the optional `keep` argument.

3 Polytomous Models

The `runWinsteps` function will automatically detect the scoring of items. If there are only two categories, a standard Rasch model will be fit. However, if there are multiple categories, a polytomous model will be fit instead. Andrich's rating scale model (Andrich 1978) will be fit by default. However, Masters' partial credit

model (Masters 1982) can also be fit. At present, plotting methods are only available for the partial credit model, although extensions to the rating scale model are planned.

To illustrate fitting a polytomous model, we'll work with the example0dat file supplied by *Winsteps*. For convenience, this file has been included as part of the *r2Winsteps* installation, as the *science* dataset. According the Winsteps user manual, these data contain "the responses of 75 children to 25 rating-scale items. The responses are 0-dislike, 1-neutral, 2-like" (Linacre 2016, 74). These ratings correspond to the children's liking of science test items.

```
data(science)
head(science)
```

```
##   item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11
## 1     1     2     1     1     1     0     2     0     1      2      2
## 2     2     2     2     2     2     2     2     2     2      2      2
## 3     2     2     1     1     0     1     1     0     1      2      2
## 4     1     0     1     0     0     1     0     1     2      2      1
## 5     1     0     1     0     1     0     1     0     0      1      1
## 6     1     0     1     1     2     1     1     0     1      1      1
##   item12 item13 item14 item15 item16 item17 item18 item19 item20 item21
## 1      2      2      0      2      1      1      2      2      0      2
## 2      2      2      2      2      2      2      2      2      2      2
## 3      2      2      1      2      2      1      2      2      1      2
## 4      2      2      1      1      1      1      2      2      0      2
## 5      1      1      0      0      1      1      2      2      1      1
## 6      2      1      0      1      0      1      2      2      1      0
##   item22 item23 item24 item25 Sex LastName FirstName MiddleName
## 1      1      0      2      0   M  Rossner      Marc     Daniel
## 2      2      2      2      2   M  Rossner  Lawrence         F.
## 3      1      1      1      1   M  Rossner      Toby         G.
## 4      1      1      1      1   M  Rossner   Michael         T.
## 5      1      1      1      0   F  Rossner   Rebecca         A.
## 6      1      2      1      0   M  Rossner        Tr        Cat
```

The rating scale model could be fit to these data exactly as we fit the dichotomous model.

```
ratingScale <- runWinsteps(science[ ,1:25], science[ ,26:ncol(science)])
str(ratingScale)
```

```
## List of 3
##  $ ItemParameters  :'data.frame':    25 obs. of  22 variables:
##   ..$ Entry               : int [1:25] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Difficulty          : num [1:25] -0.4 -0.71 0.42 1.75 2.42 0.31 1.1 1.67 0.71 -1.49 ...
##   ..$ Status              : int [1:25] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ Count               : num [1:25] 75 75 75 75 75 75 75 75 75 75 ...
##   ..$ RawScore            : num [1:25] 109 116 88 52 37 91 69 54 80 130 ...
##   ..$ SE                  : num [1:25] 0.21 0.22 0.19 0.2 0.22 0.19 0.19 0.2 0.19 0.26 ...
##   ..$ Infit               : num [1:25] 0.55 0.93 0.57 0.89 2.3 0.81 0.97 1.1 1.18 0.78 ...
##   ..$ Infit_Z             : num [1:25] -3.48 -0.39 -3.54 -0.68 5.61 -1.37 -0.13 0.71 1.26 -1.06 ...
##   ..$ Outfit              : num [1:25] 0.49 0.72 0.54 0.91 3.62 0.76 1.01 1.21 1.17 0.57 ...
##   ..$ Outfit_Z            : num [1:25] -2.53 -1.02 -3.05 -0.44 7.27 -1.38 0.13 1.19 1.02 -1.14 ...
##   ..$ Displacement        : num [1:25] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ PointMeasureCorr    : num [1:25] 0.64 0.58 0.72 0.6 0.05 0.61 0.59 0.51 0.53 0.5 ...
```

```
##   ..$ Weight              : num [1:25] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ObservMatch         : num [1:25] 77 74.3 73 67.6 52.7 70.3 50 47.3 55.4 78.4 ...
##   ..$ ExpectMatch         : num [1:25] 61.7 64.4 57.7 60.1 68.1 58.3 54.7 59 56.6 77.1 ...
##   ..$ PointMeasureExpected: num [1:25] 0.49 0.46 0.55 0.61 0.61 0.54 0.59 0.61 0.57 0.38 ...
##   ..$ RMSR                : num [1:25] 0.42 0.52 0.46 0.55 0.79 0.55 0.6 0.61 0.67 0.4 ...
##   ..$ WMLE                : num [1:25] -0.39 -0.7 0.42 1.74 2.41 0.31 1.1 1.66 0.71 -1.47 ...
##   ..$ Group               : int [1:25] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ Model               : Factor w/ 1 level " R": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ Recoding            : Factor w/ 1 level " .": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ItemID              : Factor w/ 25 levels " item1"," item10",..: 1 12 19 20 21 22 23 24 25 2 .
##  $ PersonParameters:'data.frame':    75 obs. of  22 variables:
##   ..$ Entry               : int [1:75] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Theta               : num [1:75] 0.61 6.07 1.1 0.26 -0.67 -0.08 2.71 0.97 -0.08 0.38 ...
##   ..$ Status              : int [1:75] 1 0 1 1 1 1 1 1 1 1 ...
##   ..$ Count               : num [1:75] 25 25 25 25 25 25 25 25 25 25 ...
##   ..$ RawScore            : num [1:75] 30 50 34 27 19 24 44 33 24 28 ...
##   ..$ SE                  : num [1:75] 0.34 1.84 0.36 0.34 0.35 0.34 0.48 0.35 0.34 0.34 ...
##   ..$ Infit               : num [1:75] 0.95 1 0.44 0.72 0.88 1.62 1.84 1 1.41 0.71 ...
##   ..$ Infit_Z             : num [1:75] -0.1 0 -2.57 -1.14 -0.4 2.14 1.89 0.1 1.5 -1.17 ...
##   ..$ Outfit              : num [1:75] 0.83 1 0.39 0.7 1.33 2.39 1.1 0.82 1.83 0.66 ...
##   ..$ Outfit_Z            : num [1:75] -0.43 0 -1.92 -0.98 1.04 3.51 0.39 -0.4 2.35 -1.16 ...
##   ..$ Displacement        : num [1:75] 0 0.01 0 0 0 0 0 0 0 0 ...
##   ..$ PointMeasureCorr    : num [1:75] 0.7 0 0.79 0.7 0.48 0.2 0.43 0.76 0.37 0.75 ...
##   ..$ Weight              : num [1:75] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ObservMatch         : num [1:75] 68 100 88 72 60 56 76 64 52 64 ...
##   ..$ ExpectMatch         : num [1:75] 59.5 100 61.8 59.8 60.7 59.5 79.3 61.8 59.5 60.2 ...
##   ..$ PointMeasureExpected: num [1:75] 0.63 0 0.6 0.64 0.64 0.65 0.44 0.61 0.65 0.64 ...
##   ..$ RMSR                : num [1:75] 0.57 0 0.37 0.5 0.54 0.75 0.57 0.57 0.7 0.5 ...
##   ..$ WMLE                : num [1:75] 0.6 6.07 1.08 0.26 -0.66 -0.08 2.65 0.96 -0.08 0.37 ...
##   ..$ Sex                 : Factor w/ 2 levels " F "," M ": 2 2 2 2 1 2 2 2 2 2 ...
##   ..$ LastName            : Factor w/ 62 levels " Airehead   ",..: 48 48 48 48 48 48 62 31 53 27 ...
##   ..$ FirstName           : Factor w/ 66 levels " Alan       ",..: 36 33 61 39 49 63 7 53 38 17 ...
##   ..$ MiddleName          : Factor w/ 19 levels ""," A.",," Baby",..: 5 7 9 18 2 4 1 19 1 16 ...
##  $ StructureFiles  :'data.frame':    3 obs. of  2 variables:
##   ..$ Category: int [1:3] 0 1 2
##   ..$ delta   : num [1:3] 0 -0.86 0.86
##  - attr(*, "class")= chr "r2Winsteps"
```

Notice now that there is one additional returned element: the structure file. Because we fit the rating scale model, there is only one structure that defines the categories for all items

```
ratingScale$StructureFiles
```

```
##   Category delta
## 1        0  0.00
## 2        1 -0.86
## 3        2  0.86
```

We can fit the partial credit model to the same data by supplying the additional argument `partialCredit = TRUE`.

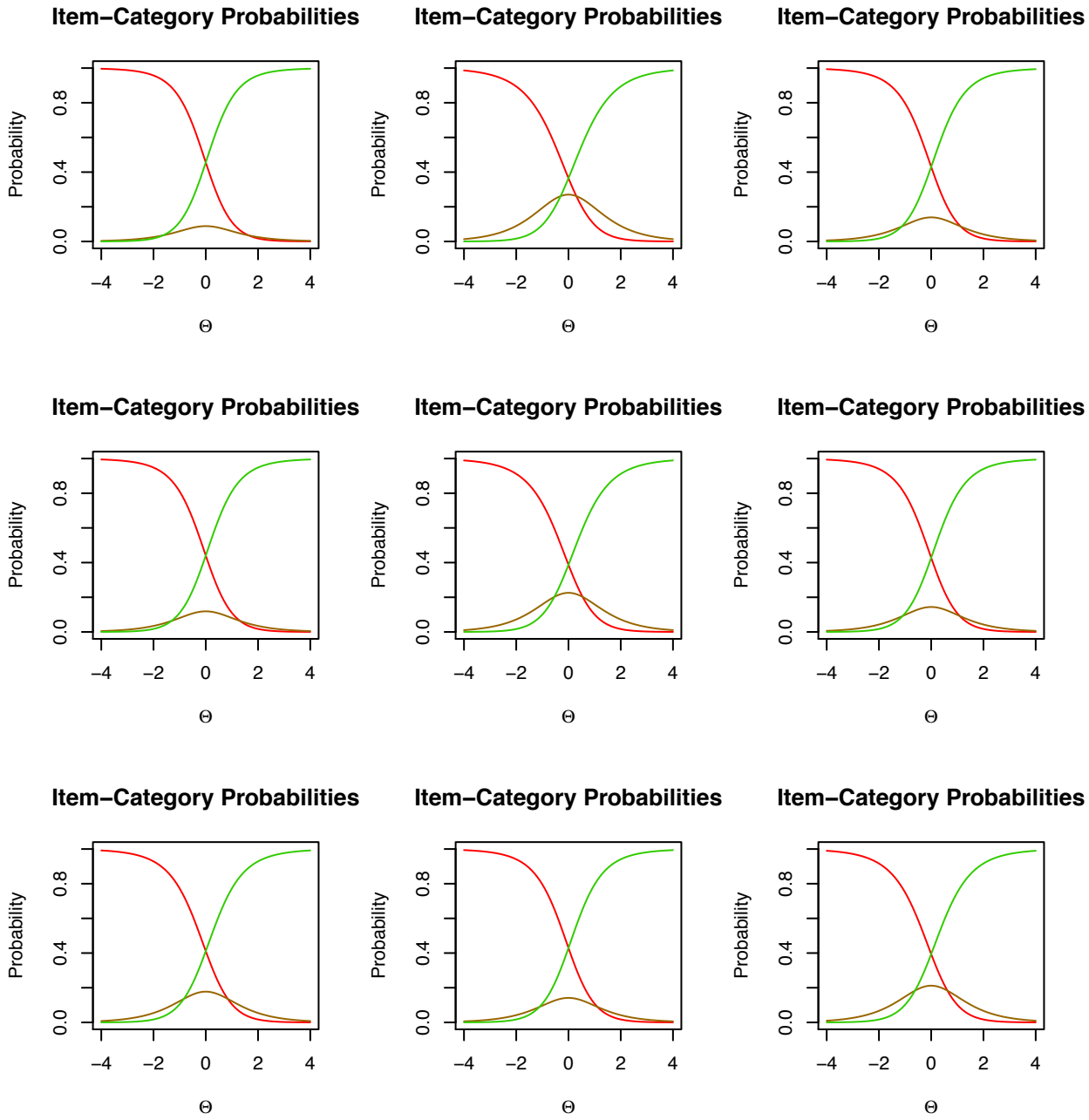
```
pc <- runWinsteps(science[ ,1:25], science[ ,26:ncol(science)],
       partialCredit = TRUE)
str(pc)
```

```
## List of 3
##  $ ItemParameters  :'data.frame':    25 obs. of  22 variables:
##   ..$ Entry               : int [1:25] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Difficulty          : num [1:25] -0.89 -0.61 0.28 1.8 2.19 0.16 0.99 1.64 0.6 -1.57 ...
##   ..$ Status              : int [1:25] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ Count               : num [1:25] 75 75 75 75 75 75 75 75 75 75 ...
##   ..$ RawScore            : num [1:25] 109 116 88 52 37 91 69 54 80 130 ...
##   ..$ SE                  : num [1:25] 0.23 0.2 0.2 0.21 0.22 0.2 0.19 0.21 0.18 0.26 ...
##   ..$ Infit               : num [1:25] 0.73 0.75 0.66 0.99 2.28 0.88 0.97 1.17 1.11 0.78 ...
##   ..$ Infit_Z             : num [1:25] -1.93 -1.49 -2.63 -0.03 5.28 -0.79 -0.17 1.14 0.83 -1.09 ...
##   ..$ Outfit              : num [1:25] 0.67 0.56 0.62 0.98 3.98 0.82 1 1.23 1.13 0.56 ...
##   ..$ Outfit_Z            : num [1:25] -1.83 -1.44 -2.51 -0.05 6.98 -1.01 0.08 1.41 0.76 -1.14 ...
##   ..$ Displacement        : num [1:25] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ PointMeasureCorr    : num [1:25] 0.64 0.58 0.72 0.6 0.05 0.61 0.59 0.51 0.53 0.5 ...
##   ..$ Weight              : num [1:25] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ObservMatch         : num [1:25] 74.3 73 74.3 71.6 52.7 68.9 52.7 45.9 56.8 78.4 ...
##   ..$ ExpectMatch         : num [1:25] 66.5 62.5 60.4 61.5 68.7 60.3 55.2 60.2 54.5 77 ...
##   ..$ PointMeasureExpected: num [1:25] 0.48 0.46 0.55 0.59 0.63 0.54 0.59 0.6 0.57 0.38 ...
##   ..$ RMSR                : num [1:25] 0.43 0.5 0.47 0.54 0.81 0.55 0.6 0.61 0.67 0.4 ...
##   ..$ WMLE                : num [1:25] -0.89 -0.6 0.28 1.8 2.18 0.17 0.99 1.64 0.6 -1.55 ...
##   ..$ Group               : int [1:25] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ Model               : Factor w/ 1 level " R": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ Recoding            : Factor w/ 1 level " .": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ItemID              : Factor w/ 25 levels " item1"," item10",..: 1 12 19 20 21 22 23 24 25 2 .
##  $ PersonParameters:'data.frame':    75 obs. of  22 variables:
##   ..$ Entry               : int [1:75] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Theta               : num [1:75] 0.49 5.98 0.99 0.14 -0.77 -0.2 2.63 0.86 -0.2 0.26 ...
##   ..$ Status              : int [1:75] 1 0 1 1 1 1 1 1 1 1 ...
##   ..$ Count               : num [1:75] 25 25 25 25 25 25 25 25 25 25 ...
##   ..$ RawScore            : num [1:75] 30 50 34 27 19 24 44 33 24 28 ...
##   ..$ SE                  : num [1:75] 0.35 1.84 0.36 0.34 0.34 0.34 0.48 0.36 0.34 0.34 ...
##   ..$ Infit               : num [1:75] 0.96 1 0.46 0.73 0.81 1.61 1.85 1.01 1.42 0.74 ...
##   ..$ Infit_Z             : num [1:75] -0.07 0 -2.29 -1.04 -0.71 2.08 1.93 0.13 1.54 -0.98 ...
##   ..$ Outfit              : num [1:75] 0.84 1 0.42 0.66 1.4 2.51 1.18 0.85 2.01 0.66 ...
##   ..$ Outfit_Z            : num [1:75] -0.31 0 -1.44 -1 1.2 3.44 0.5 -0.21 2.53 -0.95 ...
##   ..$ Displacement        : num [1:75] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ PointMeasureCorr    : num [1:75] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ Weight              : num [1:75] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ObservMatch         : num [1:75] 68 100 88 76 64 56 76 64 52 64 ...
##   ..$ ExpectMatch         : num [1:75] 61.7 100 63.4 59.9 60 59.1 79.3 63.9 59.1 61 ...
##   ..$ PointMeasureExpected: num [1:75] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ RMSR                : num [1:75] 0.56 0 0.38 0.5 0.53 0.75 0.57 0.56 0.71 0.5 ...
##   ..$ WMLE                : num [1:75] 0.48 5.98 0.98 0.14 -0.77 -0.2 2.57 0.85 -0.2 0.25 ...
##   ..$ Sex                 : Factor w/ 2 levels " F "," M ": 2 2 2 2 1 2 2 2 2 2 ...
##   ..$ LastName            : Factor w/ 62 levels " Airehead   ",..: 48 48 48 48 48 48 62 31 53 27 ...
##   ..$ FirstName           : Factor w/ 66 levels " Alan       ",..: 36 33 61 39 49 63 7 53 38 17 ...
##   ..$ MiddleName          : Factor w/ 19 levels ""," A.",," Baby",..: 5 7 9 18 2 4 1 19 1 16 ...
##  $ StructureFiles  :'data.frame':    74 obs. of  3 variables:
##   ..$ Item    : int [1:74] 1 1 1 2 2 2 3 3 3 4 ...
```

```
##   ..$ Category: int [1:74] 0 1 2 0 1 2 0 1 2 0 ...
##   ..$ delta   : num [1:74] 0 -1.64 1.64 0 -0.3 0.3 0 -1.13 1.13 0 ...
##  - attr(*, "class")= chr "r2Winsteps"
```

Now, the category structure file includes Rasch-Andrich thresholds for each item. We can view the item-category probabilities by specifying `type = ICP`. By default all item category probabilities will be produced, and it's often helpful to specify a range, or only one item. For example, we can view the category probabilities for the first nine items with

```
par(mfrow = c(3, 3))
plot(pc, type = "ICP", itemSelect = 1:9, legend = FALSE)
```



Notice the plot was specified with `legend = FALSE`, to provide more plotting room. However, it's clear which categories are represented. The red curve represents the bottom category, with a decreasing

probability as theta increases. The brown curve represents the middle category, increasing in probability to a point, but then decreasing in probability. Finally, the green curve represents the top category, monotonically increasing with theta.

4 Batch Processing

References

Andrich, David. 1978. "A Rating Formulation for Ordered Response Categories." *Psychometrika* 43 (4). Springer: 561–73.

Dahl, David B. 2014. *Xtable: Export Tables to LaTeX or HTML.* https://CRAN.R-project.org/package=xtable.

Linacre, John M. 2016. *A User's Guide to WINSTEPS MINISTEP Rasch-Model Computer Programs.* Chicago IL. http://www.winsteps.com/index.htm.

Masters, Geoff N. 1982. "A Rasch Model for Partial Credit Scoring." *Psychometrika* 47 (2). Springer: 149–74.

R Core Team. 2015. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Sarkar, Deepayan. 2008. *Lattice: Multivariate Data Visualization with R.* New York: Springer. http://lmdvr.r-forge.r-project.org.

Stodden, Victoria, Friedrich Leisch, and Roger D Peng. 2014. *Implementing Reproducible Research.* CRC Press.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. http://had.co.nz/ggplot2/book.

Wu, M, and RJ Adams. 2012. "Properties of Rasch Residual Fit Statistics." *Journal of Applied Measurement* 14 (4): 339–55.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr: 2nd Edition.* Chapman; Hall/CRC.