# Guide to IRT Invariance Tests in R

Dr. Adam W. Meade, NC State University

2020-09-09

## Getting Started

To get started, there are several packages required to use this guide. If these packages are not already installed, or if you haven't updated in a while, first run the following code:

```
install.packages("psych")
install.packages("lessR")
install.packages("mirt")
```

If these packages are already installed and updated, skip the above and just load these libraries.

```
rm(list=ls())   # clear out old junk
library("psych")
library("lessR")
library('mirt')
```

## Optional Function

This step is completely optional, but I have written a function to help make it easier to identify differentially functioning (DF) and non-DF items. Note, older versions of MIRT used List format to provide results of DIF tests whereas newer version use a dataframe. Let's go ahead and load it for later:

```
get.dif.items <- function(f.data,p.val=.05,parms){
  r.warnings = ""
  keep.vars <- c("X2", "df", "p")  # just keep these variables
  f.data <- f.data[keep.vars]
  f.data$p = round(f.data$p,3)
  if(missing(f.data)) return('Missing model output out.list')
  f.data$sig <- ifelse(f.data$p < p.val,'dif','no_dif')
  if(!missing(parms)){
    if(nrow(f.data) == nrow(parms)){
      f.data <- cbind(f.data,parms)
    }else{
      r.warnings = "There number of item parameters doesn't match the number of items "
      r.warnings = paste(r.warnings,"given to get.dif.items. Item parameters omitted.")
    }
  }
  dif.items <- subset(f.data, sig == 'dif')
  no.dif.items <- subset(f.data, sig == 'no_dif')
  if(!missing(parms) && nrow(f.data) == nrow(parms)){
    if(nrow(no.dif.items)>1){
```

```
      no.dif.items <- no.dif.items[order(-no.dif.items$a1),]
    }
  }

  r.list <- list(dif_items = dif.items, no_dif = no.dif.items, warnings = r.warnings)
  return(r.list)
}
```

## Create data for illustration

Next we will create some data just to illustrate the process. When analyzing your own data, you can skip this step.

```
make.data <- function(N){
  set.seed(12345)
  a <- matrix(abs(rnorm(15,1,.3)), ncol=1)
  d <- matrix(rnorm(15,0,.7),ncol=1)
  d1 <- d2 <- cbind(d, d-1, d-2)  # b parameters for both groups
  d2[13:15, ] <- d1[13:15, ] + 1  # here is the DIF
  itemtype <- rep('graded', nrow(a))
  dataset1 <- simdata(a, d1, N, itemtype)
  dataset2 <- simdata(a, d2, N, itemtype)
  dat <- rbind(dataset1, dataset2)
  return(dat)
}
N <- 1000
dat <- make.data(N)
group <- c(rep('Ref', N), rep('Foc', N))
focal.data <- dat[1:1000,]
ref.data <- dat[1001:2000,]
rm(N)
```

Here we have specified our sample size per group and given the groups labels. In general, you want your reference group to be the first group.

# Step 1: Testing Assumptions

## Parallel Analysis

Start by examining dimensionality. I'm using parallel analysis from the Psych package. I like to do this for each group separately.

```
########### check dimensionality ########
fa.parallel(focal.data)
fa.parallel(ref.data)
```

Im hiding the output here, but it's clearly unidimensional.

## Model Fit

Next, compute stats and examine plots. I'm surpressing output here again, but you should certainly inspect your own data.

```
########## check model fit ##############
mirtCluster(4)  # speed up processing
foc.model <- mirt(focal.data, model = 1, itemtype = "graded", SE=TRUE)
M2(foc.model)
ref.model <- mirt(ref.data, model = 1, itemtype = "graded", SE=TRUE)
M2(ref.model)
(foc.fit <- itemfit(foc.model))
(ref.fit <- itemfit(ref.model))

### optional plotting
plots.foc <- list()
plots.ref <- list()
for(i in 1:ncol(dat)){
  plots.foc[[i]]<-itemfit(foc.model,empirical.plot = i)
  plots.ref[[i]]<-itemfit(ref.model,empirical.plot = i)
}
plots.foc
plots.ref
```

## Check for Sufficient Number of Category Responses

With Likert-type scales, often there will be very few respondents endorsing some of the extreme response options. If you haven't already done so, consider collapsing response categories where there are fewer than 20 respondents and/or where the SE associated with the item b parameter is greater than around .4. Different items can have different numbers of response options, but the number of response options must be the same across groups. Note - these are unpublished rules of thumb, so use at your own risk.

```
apply(ref.data, 2, table)
```

```
##   Item_1 Item_2 Item_3 Item_4 Item_5 Item_6 Item_7 Item_8 Item_9 Item_10
## 0    398    611    573    315    481    367    320    593    694     752
## 1    189    170    174    218    199    238    183    207    148     123
## 2    178    108    124    207    146    194    175    105     81      69
## 3    235    111    129    260    174    201    322     95     77      56
##   Item_11 Item_12 Item_13 Item_14 Item_15
## 0     272     545     236     241     313
## 1     188     178     186     190     215
## 2     207     117     191     201     200
## 3     333     160     387     368     272
```

```
apply(focal.data, 2, table)
```

```
##   Item_1 Item_2 Item_3 Item_4 Item_5 Item_6 Item_7 Item_8 Item_9 Item_10
## 0    395    632    539    327    474    346    317    569    713     726
## 1    186    146    209    206    170    242    186    208    144     154
## 2    170    121    136    207    152    204    184    117     90      71
## 3    249    101    116    260    204    208    313    106     53      49
##   Item_11 Item_12 Item_13 Item_14 Item_15
## 0     261     544     416     408     513
## 1     174     173     209     176     223
## 2     199     115     157     192     145
## 3     366     168     218     224     119
```

# Step 2: Get Item Parameters (optional)

## Freely Estimated Model

This step is also optional, but usually you'll want to look at your item parameters and put them in your manuscript somewhere. Once you start doing invariance tests, you will be constraining parameters to be the same for some items. This is a good time to go ahead and get unconstrained item parameters for your manuscript.

```
model.free <- multipleGroup(dat, 1, group)
```

```
coef(model.free, simplify = TRUE)    # for the manuscript
```

```
## $Foc
## $items
##            a1     d1     d2     d3
## Item_1  1.237  0.533 -0.454 -1.505
## Item_2  1.246 -0.585 -1.624 -2.600
## Item_3  1.114 -0.370 -1.346 -2.322
## Item_4  0.897  0.902 -0.158 -1.214
## Item_5  1.120  0.098 -0.939 -1.914
## Item_6  0.378  0.565 -0.440 -1.422
## Item_7  1.193  0.966 -0.019 -0.960
## Item_8  0.958 -0.458 -1.645 -2.606
## Item_9  0.932 -0.967 -1.943 -2.828
## Item_10 0.665 -1.212 -2.100 -3.010
## Item_11 1.102  1.214  0.184 -0.879
## Item_12 1.342 -0.240 -1.271 -2.163
## Item_13 1.350  1.570  0.437 -0.616
## Item_14 1.116  1.408  0.329 -0.686
## Item_15 0.683  0.876 -0.113 -1.078
##
## $means
## F1
##  0
##
## $cov
##    F1
## F1  1
##
##
## $Ref
## $items
##            a1     d1     d2     d3
## Item_1  1.076  0.523 -0.401 -1.343
## Item_2  1.168 -0.690 -1.581 -2.679
## Item_3  0.936 -0.192 -1.280 -2.336
## Item_4  0.865  0.826 -0.164 -1.208
## Item_5  1.175  0.139 -0.751 -1.707
## Item_6  0.540  0.681 -0.374 -1.412
## Item_7  1.289  1.006 -0.025 -1.040
## Item_8  0.870 -0.324 -1.428 -2.396
## Item_9  0.873 -1.042 -2.028 -3.204
## Item_10 0.766 -1.087 -2.199 -3.228
```

```
## Item_11 0.871  1.198  0.304 -0.635
## Item_12 1.529 -0.261 -1.319 -2.217
## Item_13 1.257  0.438 -0.682 -1.661
## Item_14 1.066  0.469 -0.400 -1.496
## Item_15 0.837 -0.060 -1.169 -2.247
##
## $means
## F1
##  0
##
## $cov
##    F1
## F1  1
```

## Step 3: Baseline Model

We will use likelihood ratio tests (LRTs). To start, we need to create a baseline model in which all items have parameters constrained across items. We also will take a look at the parameters in the constrained model. I'll surpress the output here, but we are also getting item parameters constrained to be equal across groups.

```
################## Baseline Model ############################
model.constrained <- multipleGroup(dat, 1, group,
        invariance = c(colnames(dat), 'free_means', 'free_var'))
(constrained.parameters <- coef(model.constrained,simplify = TRUE)[[1]][[1]])
```

## Step 4: First Round of LRTs

First, test each of the items by freeing the parameters of each item, one at a time. The other items serve as the anchor items (i.e., the all-others-as-anchors model).

```
#################### First round of DIF analysesb - All Others As Anchors #################
(dif.drop <- DIF(model.constrained, c('a1','d1','d2','d3'), scheme = 'drop', seq_stat = .05))
```

```
##              AIC     AICc   SABIC      HQ     BIC      X2 df     p
## Item_1    -4.618   -4.075   5.078   3.609  17.786  12.618  4 0.013
## Item_2     3.201    3.743  12.896  11.427  25.604   4.799  4 0.309
## Item_3    -7.819   -7.277   1.876   0.407  14.585  15.819  4 0.003
## Item_4     4.543    5.085  14.238  12.769  26.947   3.457  4 0.484
## Item_5   -10.774  -10.232  -1.079  -2.548  11.629  18.774  4 0.001
## Item_6     1.191    1.734  10.887   9.417  23.595   6.809  4 0.146
## Item_7     0.436    0.978  10.131   8.662  22.839   7.564  4 0.109
## Item_8    -4.639   -4.097   5.056   3.587  17.764  12.639  4 0.013
## Item_9     3.398    3.941  13.094  11.624  25.802   4.602  4 0.331
## Item_10   -1.962   -1.420   7.733   6.264  20.441   9.962  4 0.041
## Item_11  -14.171  -13.629  -4.476  -5.945   8.233  22.171  4 0.000
## Item_12   -3.361   -2.819   6.334   4.865  19.042  11.361  4 0.023
## Item_13 -102.548 -102.006 -92.852 -94.322 -80.144 110.548  4 0.000
## Item_14  -57.339  -56.797 -47.644 -49.113 -34.935  65.339  4 0.000
## Item_15 -104.997 -104.455 -95.302 -96.771 -82.593 112.997  4 0.000
```

```
## use the optional function to table the output
get.dif.items(f.data=dif.drop,p.val=.05,parms=constrained.parameters)
```

```
## $dif_items
##                 X2 df     p sig        a1         d1           d2         d3
## Item_1   12.617505  4 0.013 dif 1.1232715  0.6298066 -0.320057304 -1.3095206
## Item_3   15.818965  4 0.003 dif 0.9922609 -0.1809792 -1.209024123 -2.2204892
## Item_5   18.774232  4 0.001 dif 1.1031775  0.2197826 -0.734651243 -1.6890229
## Item_8   12.639474  4 0.013 dif 0.8762771 -0.3046022 -1.440640155 -2.4009223
## Item_10   9.962481  4 0.041 dif 0.6934429 -1.0798484 -2.075703890 -3.0409324
## Item_11  22.170960  4 0.000 dif 0.9444870  1.2822498  0.332114956 -0.6575168
## Item_12  11.361130  4 0.023 dif 1.3980824 -0.1167109 -1.155452094 -2.0455828
## Item_13 110.547872  4 0.000 dif 1.2719896  1.0762762 -0.005587984 -0.9825108
## Item_14  65.338954  4 0.000 dif 1.0961640  1.0153050  0.066249884 -0.9696149
## Item_15 112.997023  4 0.000 dif 0.7576384  0.4724196 -0.532885307 -1.5044003
##
## $no_dif
##              X2 df     p    sig        a1         d1          d2         d3
```

```
## Item_7 7.564119  4 0.109 no_dif 1.2134407  1.0972210  0.09394862 -0.8810099
## Item_2 4.799253  4 0.309 no_dif 1.1903195 -0.5260877 -1.49005357 -2.5256397
## Item_9 4.601646  4 0.331 no_dif 0.8922566 -0.9205046 -1.90027658 -2.9154160
## Item_4 3.456962  4 0.484 no_dif 0.8671231  0.9450889 -0.07923670 -1.1286627
## Item_6 6.808646  4 0.146 no_dif 0.4449100  0.6626897 -0.36459751 -1.3718975
##
## $warnings
## [1] ""
```

## Step 5: Specify a New Baseline Model using Anchor Items

We will use the A5 method from Meade and Wright (2012) in which we will choose five anchor items with the largest A parameters. Note that the get.dif.items function will sort non-dif items by the A parameter if supplied. I have manually specified the items that are anchors based on the output above. Specifically, I am specifying Items 2, 4, 6, 7, and 9 based on their A parameters and non-DIF initial results.

```
itemnames <- colnames(dat)
anc.items.names <- itemnames[c(2,4,6,7,9)]
test.items <- c(1,3,5,8,10:15)
model_anchor <- multipleGroup(dat, model = 1, group = group,
        invariance = c(anc.items.names, 'free_means', 'free_var'))
(anchor.parms <-coef(model_anchor,simplify = TRUE)[[1]][[1]])
```

The output here is not very interesting so we will proceed with the final round of invariance tests.

## Step 6: Run the Final Invariance Tests

```
(dif.anchor <- DIF(model_anchor, c('a1','d1','d2','d3'), items2test = test.items, plotdif = TRUE))
```

```
#################### Final round of DIF analyses #################
## use the optional function to table the output
get.dif.items(f.data=dif.anchor,p.val=.05)
```

```
## $dif_items
##              X2 df p sig
## Item_13 103.851  4 0 dif
## Item_14  71.323  4 0 dif
## Item_15 119.647  4 0 dif
##
## $no_dif
##            X2 df     p    sig
## Item_1  2.862  4 0.581 no_dif
## Item_3  6.604  4 0.158 no_dif
## Item_5  6.268  4 0.180 no_dif
## Item_8  4.042  4 0.400 no_dif
## Item_10 6.352  4 0.174 no_dif
## Item_11 8.568  4 0.073 no_dif
## Item_12 1.816  4 0.770 no_dif
##
## $warnings
## [1] ""
```

Using our custom function makes it easy to see that the correct items were identified as DIF items (Items 13-15)

## Step 7: Compute Effect Sizes

The last step is to compute effect size estimates, as described in Meade (2010).

### Test-Level Effect Sizes

```
empirical_ES(model_anchor, DIF=FALSE)  # test-level effect sizes
```

```
##           Effect Size      Value
## 1                STDS -1.1113795
## 2                UTDS  2.0559964
## 3              UETSDS  1.1113795
## 4               ETSSD -0.1630801
## 5         Starks.DTFR -1.0828614
## 6               UDTFR  2.0227532
## 7              UETSDN  1.0828614
## 8 theta.of.max.test.D  0.2529304
## 9           Test.Dmax -1.2979990
```

### Item-Level Effect Sizes

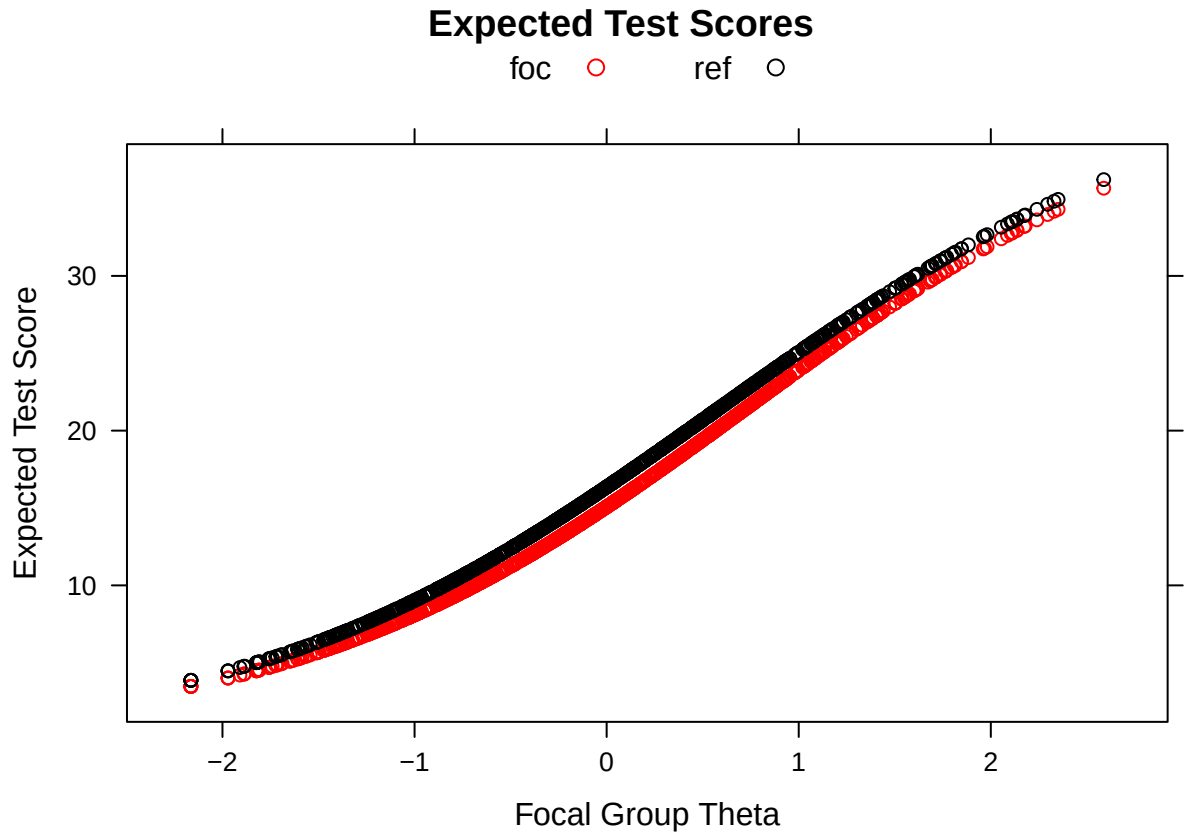
```
empirical_ES(model_anchor)  # item-level effect sizes
```

```
##           SIDS  UIDS   SIDN  UIDN   ESSD theta.of.max.D  max.D mean.ES.foc
## item.1   0.052 0.062  0.049 0.062  0.090         -1.031  0.103       1.266
## item.2   0.000 0.000  0.000 0.000  0.000          0.478  0.000       0.674
## item.3   0.046 0.070  0.041 0.071  0.105          2.588 -0.178       0.815
## item.4   0.000 0.000  0.000 0.000  0.000          0.478  0.000       1.389
## item.5   0.097 0.097  0.097 0.097  0.180          1.370  0.173       1.073
## item.6   0.000 0.000  0.000 0.000  0.000          0.478  0.000       1.246
## item.7   0.000 0.000  0.000 0.000  0.000          0.478  0.000       1.476
## item.8   0.084 0.084  0.082 0.082  0.227          0.127  0.096       0.750
## item.9   0.000 0.000  0.000 0.000  0.000          0.478  0.000       0.493
## item.10  0.022 0.027  0.023 0.029  0.100          2.588  0.155       0.434
## item.11  0.102 0.114  0.099 0.115  0.200         -1.644  0.232       1.672
## item.12  0.032 0.057  0.035 0.059  0.053          1.685  0.177       0.879
## item.13 -0.552 0.552 -0.535 0.535 -0.900          0.202 -0.660       1.158
## item.14 -0.436 0.436 -0.425 0.425 -0.798          0.122 -0.495       1.231
## item.15 -0.557 0.557 -0.548 0.548 -1.487          0.013 -0.597       0.861
##         mean.ES.ref
## item.1        1.214
## item.2        0.674
## item.3        0.769
## item.4        1.389
## item.5        0.976
## item.6        1.246
## item.7        1.476
## item.8        0.666
## item.9        0.493
## item.10       0.412
## item.11       1.570
```

```
## item.12       0.847
## item.13       1.710
## item.14       1.667
## item.15       1.418
```

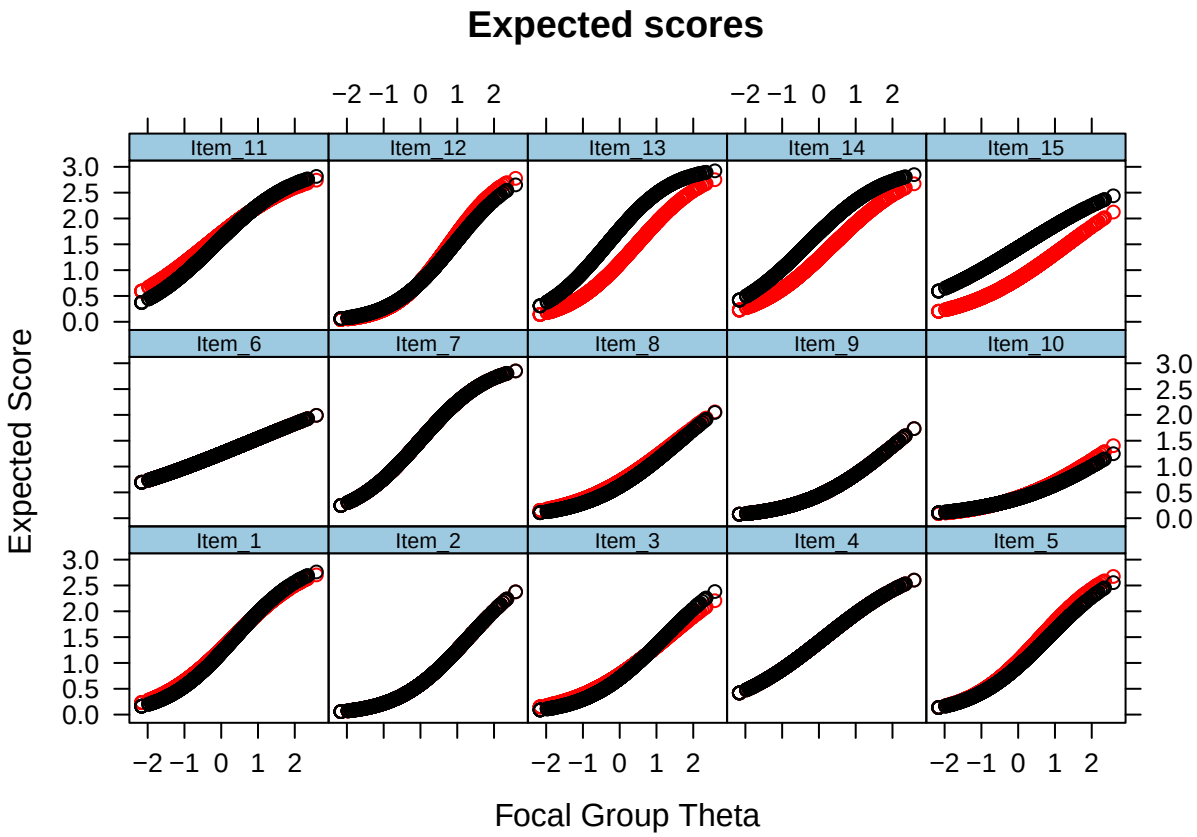
### Test-Level Plot

```
expected.test.plot <- empirical_ES(model_anchor, DIF=FALSE, plot=TRUE)
expected.test.plot
```



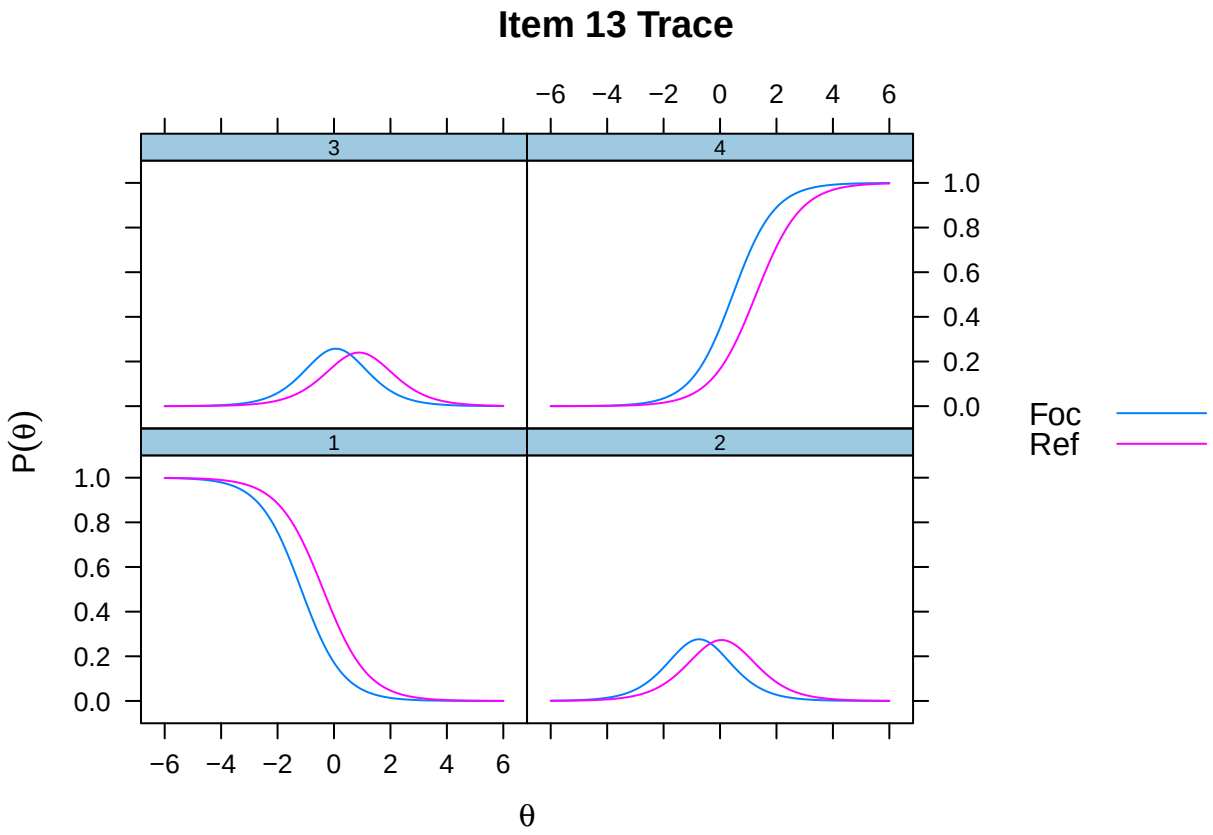
## Item-Level Plots You should also examine item-level DIF plots

```
expected.item.plots <- empirical_ES(model_anchor, plot=TRUE)
expected.item.plots
```



If you want to plot only a single item in more detail, you can do so like this

```
itemplot(model_anchor, 13) # further investigate item with DF
```



## Plots are lattice graphs which can be manipulated after the fact

```
expected.test.plot$main <- "ETS for Reference and Focal Groups"
expected.test.plot$legend$top$args$key$text[[1]] <- c('Focal', 'Reference')
expected.test.plot
```