Guide to IRT Invariance Using the MIRT Package in R

Adam W. Meade

North Carolina State University

Current as of October 5, 2016

Contents

This document assumes the use of the current version of both R and R Studio. See https://www.rstudio.com/.

Getting Started

To get started, there are several packages required to use this guide. If these packages are not already installed, first run the following code:

```
#first load some needed libraries
install.packages("psych")
install.packages("lessR")
install.packages("mirt")


library("psych")
library("lessR")
library("mirt")
```

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. Let's go ahead and load it for later:

```
#########################################################################
#                                 Functions
#########################################################################
get.dif.items <- function(out.list,p.val,parms){
  dif.items <- NULL
  non.dif.items <- NULL
  for(i in 1:length(out.list)){  # loop list of items
    chi.sq <- out.list[[i]][2,6]  #2 groups, so second row, and 6th column
    df <- out.list[[i]][2,7]
    p <-  out.list[[i]][2,8]
    i.name <- names(out.list[i])
    d <- c(i.name,chi.sq,df,p,parms[i,])

    if(p < p.val){
      dif.items <- rbind(dif.items,d)
    }else{
      non.dif.items <- rbind(non.dif.items,d)
    }
  }
  if (!is.null(dif.items)) {
    dif.items <- data.frame(dif.items, row.names = NULL)
    colnames(dif.items)[1] <- "item"
    colnames(dif.items)[2] <- "chi.sq"
    colnames(dif.items)[3] <- "df"
    colnames(dif.items)[4] <- "p"

  }
  if (!is.null(non.dif.items)) {
    non.dif.items <- data.frame(non.dif.items, row.names = NULL)
    colnames(non.dif.items)[1] <- "item"
    colnames(non.dif.items)[2] <- "chi.sq"
    colnames(non.dif.items)[3] <- "df"
    colnames(non.dif.items)[4] <- "p"
  }
  r.list <- list(dif_items = dif.items, no_dif = non.dif.items)
  return(r.list)
}
#########################  END FUNCTIONS ########################
```

Next we will make some data for illustration purposes using a function.

```
########## Make the data with a lack of invariance ######
make.data <- function(N){
  set.seed(1234)
  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,]
```

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)
```

Looking at the output, it's clearly unidimensional.

Model Fit

Next, compute stat and examine plots.

```
########## check model fit ##############
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)
foc.fit
ref.fit <- itemfit(ref.model)
ref.fit

### 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
```

All looks good.

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.

Create a Constrained 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.

```
################## 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]]
constrained.parameters
```

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 #######
mirtCluster(4)  # speed up processing
dif.drop <- DIF(model.constrained, c('a1','d1','d2','d3'), scheme = 'drop',
seq_stat = .05)
dif.drop
## use the optional function to table the output
dif.drop.out <-
get.dif.items(out.list=dif.drop,p.val=.05,parms=constrained.parameters)
dif.drop.out
```

Examine the output:

```
$dif_items
     item chi.sq df      p    a1     d1     d2     d3
1  Item_1 17.706  4 0.0014 0.688 -0.133 -1.227 -2.189
2  Item_3 11.086  4 0.0256   1.2 -0.742 -1.711 -2.531
3  Item_5 11.569  4 0.0209 1.086  1.496  0.533 -0.504
4  Item_6  17.88  4 0.0013  1.18 -0.113 -1.095 -2.106
5  Item_7 10.227  4 0.0368 0.836 -0.443  -1.49 -2.451
6 Item_11  9.673  4 0.0463 0.739 -1.155 -2.071 -3.185
7 Item_13 84.557  4      0 0.857 -0.268 -1.275  -2.36
8 Item_14 76.865  4      0 1.045    0.3 -0.624 -1.583
9 Item_15 56.403  4      0 1.216 -0.396 -1.334 -2.218

$no_dif
     item chi.sq df      p    a1     d1     d2     d3
1  Item_2  5.308  4 0.2571 1.094 -0.452 -1.527 -2.548
2  Item_4  1.156  4 0.8853 0.231 -0.579 -1.598 -2.739
3  Item_8  5.054  4 0.2818 0.904 -0.471  -1.56 -2.577
4  Item_9  6.721  4 0.1514 0.754  0.269 -0.782 -1.768
5 Item_10  7.881  4  0.096 0.652 -0.563 -1.502 -2.445
6 Item_12  7.985  4 0.0921  0.68  0.333 -0.657 -1.745
```

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.

```
##### Run an anchor-item model #####
itemnames <- colnames(dat)
anc.items.names <- itemnames[c(2,8,9,10,12)]
test.items <- c(1,3:7,11,13:15)
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]]
anchor.parms
```

The above runs the model and also requests the parameters.

Run the Final Invariance Tests

```
#################### Final round of DIF analyses #################
dif.anchor <- DIF(model_anchor, c('a1','d1','d2','d3'), items2test =
test.items, plotdif = TRUE)
dif.anchor
## use the optional function to table the output
dif.anchor.out <-
get.dif.items(out.list=dif.anchor,p.val=.05,parms=anchor.parms)
dif.anchor.out
```

Check out the output. Looks like the correct items were detected.

```
> dif.anchor.out
$dif_items
     item chi.sq df p    a1     d1     d2     d3
1 Item_13 94.622  4 0 0.753  0.342  -0.71 -1.697
2 Item_14 84.691  4 0 0.656 -0.502 -1.442 -2.386
3 Item_15 64.678  4 0 0.719 -1.018 -1.939 -2.953

$no_dif
     item chi.sq df      p    a1     d1     d2     d3
1  Item_1  7.514  4 0.1111 0.737  -0.01 -1.045 -1.976
2  Item_3  2.072  4 0.7225 1.288 -0.605 -1.655 -2.488
3  Item_4  1.114  4  0.892 0.274 -0.562 -1.617  -2.75
4  Item_5  3.916  4 0.4175 1.204  1.645  0.702 -0.426
5  Item_6  2.661  4 0.6161 1.287  0.067 -0.968 -2.036
6  Item_7  2.484  4 0.6475 0.832 -0.325 -1.366 -2.262
7 Item_11  3.511  4 0.4762  0.91 -0.386 -1.477 -2.496
```

Compute Effect Sizes

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

```
################## Compute the effect sizes ####################
empirical_ES(model_anchor, DIF=FALSE)  # test level stats
empirical_ES(model_anchor)             # item level stats
empirical_ES(model_anchor, DIF=FALSE, plot=TRUE)  # expected test score plots
empirical_ES(model_anchor, plot=TRUE)             # expected item score plots
itemplot(model_anchor, 13)             # further investigate item with DF
```

And the final output (test level):

```
> empirical_ES(model_anchor, DIF=FALSE)  # test level stats
          Effect Size Value
1                STDS 1.114
2                UTDS 1.787
3              UETSDS 1.114
4               ETSSD 0.189
5         Starks.DTFR 1.096
6               UDTFR 1.775
7              UETSDN 1.096
8 theta.of.max.test.D 0.513
9           Test.Dmax 1.241
```

Item level output:

```
> empirical_ES(model_anchor)             # item level stats
         SIDS  UIDS   SIDN  UIDN   ESSD theta.of.max.D  max.D mean.ES.foc mean.ES.ref
item.1  -0.112 0.112 -0.113 0.113 -0.333          2.538 -0.226       0.839       0.952
item.2   0.000 0.000  0.000 0.000  0.000         -0.636  0.000       0.816       0.816
item.3  -0.011 0.030 -0.013 0.032 -0.021          2.265 -0.112       0.750       0.760
item.4   0.013 0.028  0.012 0.030  0.137          2.932 -0.111       0.619       0.606
item.5  -0.014 0.059 -0.013 0.062 -0.027         -2.094  0.138       1.890       1.904
item.6  -0.052 0.059 -0.052 0.061 -0.092          1.790 -0.147       1.000       1.051
item.7  -0.051 0.051 -0.051 0.051 -0.138          0.398 -0.057       0.755       0.806
item.8   0.000 0.000  0.000 0.000  0.000         -0.636  0.000       0.766       0.766
item.9   0.000 0.000  0.000 0.000  0.000         -0.636  0.000       1.135       1.135
item.10  0.000 0.000  0.000 0.000  0.000         -0.636  0.000       0.717       0.717
item.11 -0.053 0.053 -0.051 0.051 -0.211          0.201 -0.060       0.456       0.508
item.12  0.000 0.000  0.000 0.000  0.000         -0.636  0.000       1.167       1.167
item.13  0.466 0.466  0.464 0.464  1.288          2.111  0.698       1.097       0.631
item.14  0.508 0.508  0.498 0.498  1.015          0.391  0.574       1.502       0.994
item.15  0.420 0.420  0.414 0.414  0.808          1.209  0.596       1.127       0.707
>
```

And the ETS plot. Item plots are also output but not presented here.