Logistic Regression Differential Item Functioning (DIF) Analyses Using SPSS Syntax

In this document I explain how to use SPSS syntax to examine differential item functioning (DIF) using the logistic regression (LR) methods discussed in Chapter 16. Logistic regression procedures assess item DIF by regressing the dichotomous item response on the total score, grouping variable, and the interaction between the total score and the grouping variable in three separate analyses (see Chapter 16 for more explanation on how this works). The **LOGISTIC REGRESSION** command in SPSS can run these analyses and produce the necessary log likelihood statistics.

As explained in Chapter 16, DIF can be uniform or nonuniform. In uniform DIF, only the difficulty parameter (intercept or threshold) of the item differs across groups.

In non-uniform DIF, the discrimination (or slope) of the item differs across groups. Of course, items may display both uniform and nonuniform DIF (differences across groups in both difficulty and discrimination).

LR procedures can be used to test for both uniform and nonuniform DIF by comparing the log likelihood values obtained at different steps. The difference between the log likelihood values obtained from step 1 (total score only) and step 2 (total score and grouping variable) provide a test of uniform DIF. The difference between the log likelihood values obtained from steps 2 and 3 (total score, grouping variable, and their interaction) provide a test of nonuniform DIF (see Chapter 16 for more explanation).

The data for this example are from a 31-item multiple choice test for which gender DIF was assessed. Data were available from 500 male and 803 female students. The data are available in the file “DIF data.sav.” There are no missing data. For these data, males are coded as 0 and females are coded as 1.

Basic DIF testing requires that each item be tested separately. With a large number of items, this can be tedious. Because of this, I created a mini-program called a “macro” that uses SPSS syntax to automate the process of running the analysis sequentially for all items and collating the results of these analyses. In the rest of this document I first discuss the LR analysis for a single item. I then present and discuss the macro for the LR procedures. Finally, I present and discuss a program to combine and format the results of the macro-based analyses.

Logistic Regression Procedures for a Single Item

Before running the logistic regression analyses, both the grouping variable and the total score should be centered at their means to avoid possible confounding of these variables with their interaction. The centered versions of the grouping variable and total score should be used to create the interaction as well as in the logistic regressions for each item. The syntax below will accomplish this (note that I previously obtained the mean of each variable using the **DESCRIPTIVES** command):

```
COMPUTE c_total = total -19.784344.
EXECUTE.
COMPUTE c_group = group - .616270.
EXECUTE.
COMPUTE interact = c_total * c_group.
EXECUTE.
```

To illustrate **LOGISTIC REGRESSION** commands, I ran the three analyses for item 1 using the syntax below. Note that the logistic regression must be run three times, adding in an additional variable at each step (see Chapter 16 for more explanation).

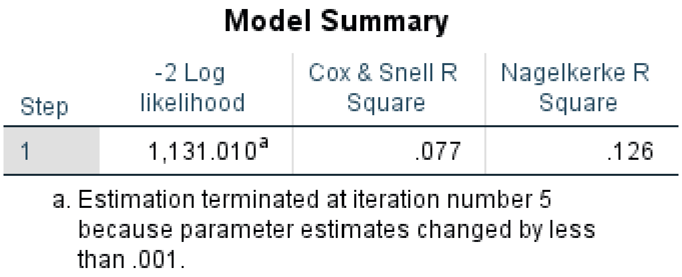
```
 LOGISTIC REGRESSION variables =I1
          /METHOD=ENTER c_total
          /CRITERIA=PIN(.05) POUT(.10) ITERATE(20).

LOGISTIC REGRESSION variables =I1
          /METHOD=ENTER c_total c_group
          /CRITERIA=PIN(.05) POUT(.10) ITERATE(20).

LOGISTIC REGRESSION variables =I1
          /METHOD=ENTER c_total c_group interact
          /CRITERIA=PIN(.05) POUT(.10) ITERATE(20).
```

The log likelihood values from the three analyses are shown below. I also created a table similar to Table 16.4 in the book to illustrate how the results could be presented.

Note that the LL values have been multiplied by -2, as is common (see Chapter 16, page 491). These values can be used to calculate the likelihood ratio tests for uniform and nonuniform DIF. Uniform DIF can be tested by comparing the -2LL values from the first and second analyses, and nonuniform DIF can be tested by comparing the -2LL values from the second and third steps.

For item 1 there appears to be uniform DIF, as indicated by the LR difference between steps 1 and 2 of 3.98, which is significant at the .05 level. The amount of DIF might be considered small, however, given the relatively low LR difference value.

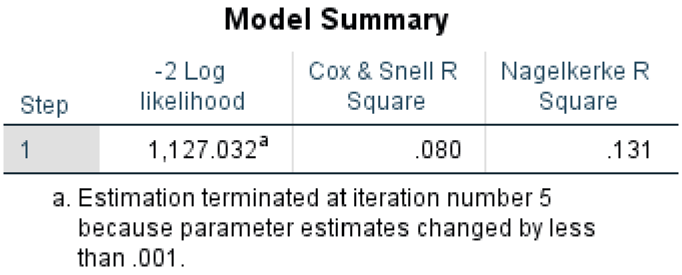
There does not appear to be any nonuniform DIF for this item, given that the -2LL values change only in the third decimal place between steps 2 and 3.



Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	1,131.010[a]	.077	.126

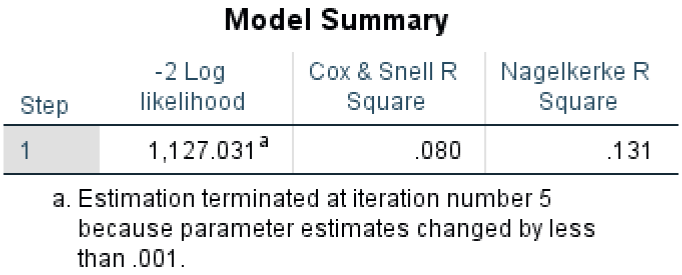
a. Estimation terminated at iteration number 5 because parameter estimates changed by less than .001.



Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	1,127.032[a]	.080	.131

a. Estimation terminated at iteration number 5 because parameter estimates changed by less than .001.



Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	1,127.031[a]	.080	.131

a. Estimation terminated at iteration number 5 because parameter estimates changed by less than .001.

Model	***-2LL***	***LR difference***	***p-value***
Total score	1131.010		
Total score and group	1127.032	3.978	.05
Total score, group, and interaction	1127.031	0.001	.98

Automating the Process

As noted previously, DIF must be tested for each item, which can become tedious when there are many items on a scale. To facilitate this process, I created an SPSS macro that automates this process. In the remainder of this document, I present the macro and discuss how it works. To run the three logistic regression steps, I created three separate macros. A separate program combines the results from the three analyses and calculates the LR tests and their significance values. I begin by presenting the macro syntax for the first logistic regression.

OMS Commands: Saving Selected Output into New Files

The **OMS** (Output Management System) procedure in SPSS can be used to select output from analyses and direct this to external files. In the set of OMS commands above, I select the tables and text from the output. Here, I specify the filename “LR1_coefficients.sav” to indicate that the file contains output from the first of the three logistic regression analyses.

The **IF COMMANDS** subcommand specifies that output under the heading “Logistic Regression” should be identified. The **SUBTYPES =** subcommand specifies that only output from tables with the heading “Model Summary” should be saved (note that this is heading for the tables containing the -2LL values shown previously).

The subcommand **DESTINATION FORMAT = SAV** specifies that the output should be saved into an SPSS system file (SAV file). If the output is based on analyses from multiple variables, as in this example where there are analyses from each of 31 items, the subcommand **NUMBERED = TableNumber** is quite useful. This subcommand creates a new variable that identifies which of the 31 (for this example) variables the output corresponds to.

Finally, the **OUTFILE = command** specifies the file (and drive) to which the output should be saved.

```
OMS
/SELECT TABLES TEXTS
/IF COMMANDS = [‘Logistic Regression’]
    SUBTYPES = [‘Model Summary’]
/DESTINATION FORMAT = SAV NUMBERED=TableNumber_
OUTFILE = ‘’your folder\LR1_coefficients.sav’.
```

Defining the Macros

The next set of commands, beginning with **DEFINE LRmac1 (arg1 = !TOKENS(1)**, contains the macro setup. The commands **DEFINE** and **!ENDDEFINE** indicate the beginning and end of the macro commands. The macro commands can include any SPSS command, including **COMPUTE** statements or SPSS procedures such as logistic regression. The use of the **“!”** symbol indicates to SPSS that the command is a macro command. **DEFINE** LRmac1 indicates that a macro named “LRmac1” is being created (names of macros are arbitrary but must follow the naming conventions for SPSS variables).

The “**!**” symbol can also serve as a wildcard placeholder that can take on different values that are specified in the body of the macro. This is illustrated by the subcommands (**arg1=!TOKENS(1)** and **/arg(2)=!TOKENS(2)**. These two subcommands define two “tokens” that are associated with the starting and ending values for the subsequent **!DO** loop.

The do loop to run the analyses sequentially for the 31 items is specified by the subcommand:

```
!DO !i = !arg1 !TO !arg2.
```

Note that both the **!DO** command and all its arguments (the looping index “i,” the two token names, and the specification **!TO**) must be preceded by the symbol “!”.

```
DEFINE LRmac1 (arg1 = !TOKENS(1)
            /arg2 = !TOKENS(1))
!DO !i = !arg1 !TO !arg2.
LOGISTIC REGRESSION variables = !CONCAT(I,!i)
/METHOD=ENTER c_total
 /CRITERIA=PIN(.05) POUT(.10) ITERATE(20).
  !DOEND
!ENDDEFINE.
Lrmac1 arg1 = 1 arg2 = 31.
OMSEND.
```

Values of the two tokens (“arg1” and “arg2”) are supplied in the later subcommand **LRmac1 arg1 = 1 arg2 = 31.** This subcommand specifies that the two tokens for the macro LRmac1 should have values of 1 and 31, respectively.

Values of the two tokens are employed in the subsequent **!DO** loop (**!DO !i = !arg1 !TO !arg2**) which specifies that the logistic regression analyses specified by the commands following the do loop statement should be carried out 31 times (from 1 to 31).

The logistic regression statement is also modified by the macro. The command:

```
LOGISTIC REGRESSION variables = !CONCAT(I,!i)
```

specifies that the variable names should be obtained by concatenating (**!CONCAT**) the letter I and the current value of the looping variable “i.” For example, in the first pass through the do loop, the variable name would be “i1,” and would change to “i2,” “i3,” and so on in subsequent loop iterations.

The **!DOEND** command specifies the end of the loop.

The **!ENDDEFINE** command specifies the end of the macro LRmac1.

Finally, the command **OMSEND** indicates the end of the output that should be sent to an external file.

Macros for the Remaining Logistic Regressions

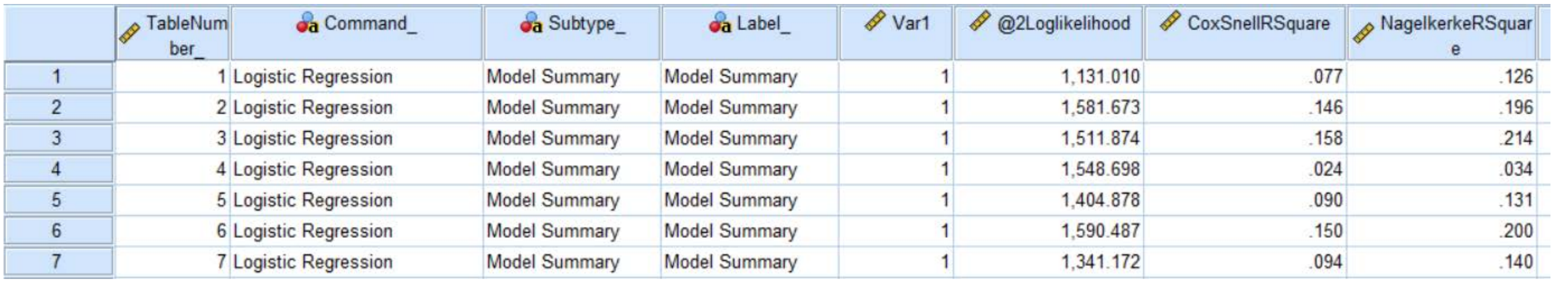
To obtain the analyses for the second and third logistic regression steps, I modify the code presented above by 1) specifying a different name for the OMS output file, 2) providing a different name for the macro, and 3) including the additional variable names on the **METHOD = ENTER** subcommand.

Below I present the code for these two regressions with the changes in italics.

```
OMS
/SELECT TABLES TEXTS
/IF COMMANDS = ['Logistic Regression']
    SUBTYPES = ['Model Summary']
/DESTINATION FORMAT = SAV NUMBERED=TableNumber_
OUTFILE = ‘your folder\LR2_coefficients.sav'.

DEFINE LRmac2 (arg1 = !TOKENS(1)
           /arg2 = !TOKENS(1))
!DO !i = !arg1 !TO !arg2.
LOGISTIC REGRESSION variables = !CONCAT(I,!i)
/METHOD=ENTER c_total c_group
 /CRITERIA=PIN(.05) POUT(.10) ITERATE(20) CUT(.5).
  !DOEND
!ENDDEFINE.
LRmac2 arg1 = 1 arg2 = 31.
OMSEND.

OMS
/SELECT TABLES TEXTS
/IF COMMANDS = ['Logistic Regression']
    SUBTYPES = ['Model Summary']
/DESTINATION FORMAT = SAV NUMBERED=TableNumber_
OUTFILE = ‘your folder\LR3_coefficients.sav'.

DEFINE LRmac3 (arg1 = !TOKENS(1)
           /arg2 = !TOKENS(1))
!DO !i = !arg1 !TO !arg2.
LOGISTIC REGRESSION variables = !CONCAT(I,!i)
/METHOD=ENTER c_total c_group interact
 /CRITERIA=PIN(.05) POUT(.10) ITERATE(20) CUT(.5).
  !DOEND
!ENDDEFINE.
LRmac3 arg1 = 1 arg2 = 31.
OMSEND.
```

Reformatting and Merging the New Datafiles

Running the entire set of commands creates three SPSS data files with the format below:



	TableNumber_	Command_	Subtype_	Label_	Var1	@2Loglikelihood	CoxSnellRSquare	NagelkerkeRSquare
1	1	Logistic Regression	Model Summary	Model Summary	1	1,131.010	.077	.126
2	2	Logistic Regression	Model Summary	Model Summary	1	1,581.673	.146	.196
3	3	Logistic Regression	Model Summary	Model Summary	1	1,511.874	.158	.214
4	4	Logistic Regression	Model Summary	Model Summary	1	1,548.698	.024	.034
5	5	Logistic Regression	Model Summary	Model Summary	1	1,404.878	.090	.131
6	6	Logistic Regression	Model Summary	Model Summary	1	1,590.487	.150	.200
7	7	Logistic Regression	Model Summary	Model Summary	1	1,341.172	.094	.140

Below I provide syntax that will strip the LR values from each of the three files, merge these into a single datafile, and compute the LR difference tests and their associated *p*-values.

I begin with the syntax to strip the LR values from the three datafiles and save these into new files.

Syntax for the first file (LR1_coefficients.sav) is shown first, followed by syntax for the other two files. Because the syntax for all three files is the same, I explain the syntax for the first and for the second and third files I simply indicate the changes in italics.

GET FILE=’*your folder*\LR1_coefficients.sav'.
DELETE VARS Command_ Subtype_ Label_ Var1 CoxSnellRSquare NagelkerkeRSquare.
RENAME VARIABLES (TableNumber_ = item) (@2Loglikelihood = LR1).
EXECUTE.
SAVE OUTFILE = ‘*your folder*\LR1.sav'.

After opening the file (**GET FILE =**), the **DELETE VARS** subcommand deletes all the variables except the TableNumber (which indicates the item number) and the LR value ((@2Loglikelihood).

The **RENAME VARIABLES** subcommand renames the item number and LR values and the new file is saved.

The file is then saved under a new name (**SAVE OUTFILE =**).

Syntax for the second file (LR2_coefficients.sav) is shown next:

GET FILE=’*your folder\LR2_coefficients.sav'*.
DELETE VARS Command_ Subtype_ Label_ Var1 CoxSnellRSquare NagelkerkeRSquare.
RENAME VARIABLES (**TableNumber_ =** item) **(@2Loglikelihood =** LR2).

```
EXECUTE.
SAVE OUTFILE = ‘your folder\LR2.sav'.
```

Lastly, syntax for the third file (LR3_coefficients.sav) is shown.

```
GET FILE=’your folder\LR3_coefficients.sav'.
DELETE VARS Command_ Subtype_ Label_ Var1 CoxSnellRSquare
NagelkerkeRSquare.
RENAME VARIABLES (TableNumber_ = item) (@2Loglikelihood = LR3).
EXECUTE.
SAVE OUTFILE = ‘your folder\LR3.sav'.
```

Merging the Three Files and Computing the LR DIF Statistics

The next set of commands merges the three files into a single file using the **MATCH FILES** command. The three files are read in using **/FILE =** subcommands.

The three files are matched by the item number (**BY item**) so that the three LR values for each item will be in the same row.

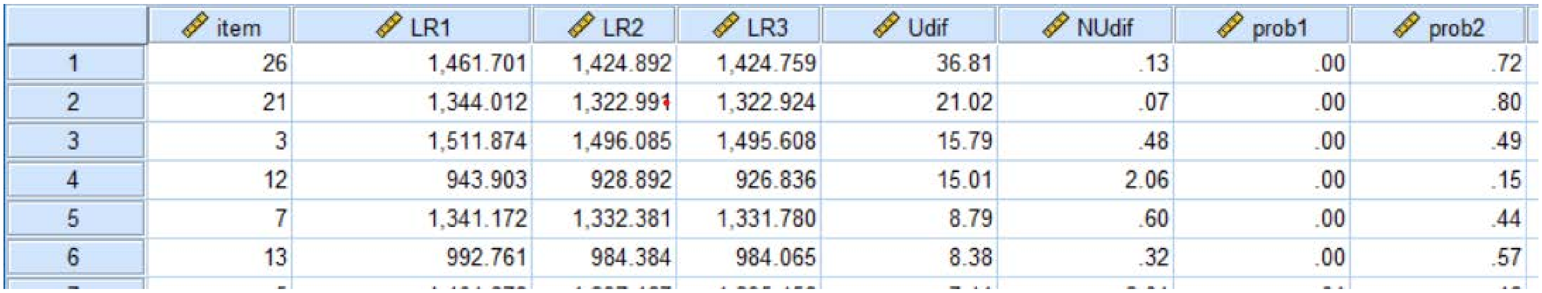
Finally, the merged file is saved using the **SAVE OUTFILE =** command. I use the **KEEP =** subcommand to keep only the item number and three LR values.

The four **COMPUTE** commands create new variables that are the tests of uniform DIF (Udif), nonuniform DIF (NUdif) and their associated *p*-values (prob1 and prob2).

The merged datafile is then saved (**SAVE OUTFILE =**).

```
MATCH FILES
  /FILE=’your folder\LR1.sav'
  /FILE=’your folder\LR2.sav'
  /FILE=’your folder\LR3.sav'
  /BY item.
EXECUTE.
SAVE OUTFILE = ‘your folder\LR.sav'
 /KEEP = item LR1 LR2 LR3.


COMPUTE Udif=LR1-LR2.
COMPUTE NUdif = LR2-LR3.
COMPUTE prob1 = 1- cdf.chisq(Udif, 1).
COMPUTE prob2 = 1- cdf.chisq(NUdif, 1).
EXECUTE.

SAVE OUTFILE = ‘your folder\LR.sav'.
```

The merged datafile will look like this:



	item	LR1	LR2	LR3	Udif	NUdif	prob1	prob2
1	26	1,461.701	1,424.892	1,424.759	36.81	.13	.00	.72
2	21	1,344.012	1,322.991	1,322.924	21.02	.07	.00	.80
3	3	1,511.874	1,496.085	1,495.608	15.79	.48	.00	.49
4	12	943.903	928.892	926.836	15.01	2.06	.00	.15
5	7	1,341.172	1,332.381	1,331.780	8.79	.60	.00	.44
6	13	992.761	984.384	984.065	8.38	.32	.00	.57

Saving LR Values as Output or Text Documents

Although not strictly necessary, researchers may wish to print the LR values and their associated *p*-values in their SPSS output and/or save these into an external text file.

The two sets of commands below will accomplish these.

The first set of commands calls up the merged dataset (**DATASET NAME**) and sorts the items from high to low by their levels of Udif and NUdif (**SORT CASES BY**; the (**D**) after each variable indicates sorting should be done in descending order). Of course, the items can be left in their canonical order, if desired.

The **PRINT** command causes specified values to be printed in the output. Here, I use FORTRAN format specifications to output the data in a readable manner.

The **WRITE OUTFILE =** command causes the specified values to be written to an external text file. I use FORTRAN specifications to format the data.

```
DATASET NAME LR.
SORT CASES BY Udif(D) NUdif(D).
PRINT/ item (F2.0, 2X) LR1 to LR3 (3(F9.4, 4X)) Udif to prob2 (4(F5.4, 4X)).

WRITE OUTFILE ='my folder\LRoutput.txt'/
   Item (F2.0)  LR1 to LR3  (3(F9.4, 4X))  Udif to prob2 (4(F5.4, 4X)).
EXECUTE.
```

The file created by the **WRITE OUTFILE** command looks like this (here I show values for just the first few items):

```
26  1461.7005    1424.8924    1424.7591    36.81    .1333    .0000    .7151
21  1344.0117    1322.9914    1322.9239    21.02    .0675    .0000    .7951
 3  1511.8737    1496.0854    1495.6084    15.79    .4769    .0001    .4898
12   943.9025     928.8918     926.8357    15.01    2.056    .0001    .1516
 7  1341.1723    1332.3814    1331.7805    8.791    .6009    .0030    .4382
13   992.7605     984.3841     984.0650    8.376    .3191    .0038    .5722
```