

Logistic Regression Differential Item Functioning (DIF) Analyses Using SAS

In this document I explain how to use SAS 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). **Proc logistic** in SAS 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.sas7bdat.” 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 SAS 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 **proc means.**) Here, I subset the original dataset (ch16.dif) into a new dataset LR.

```
data LR; set ch16.dif;  
total=total - 19.7843438;  
group=group - 0.383730;  
run;
```

To illustrate the **proc logistic** commands, I ran the 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). The item that will serve as the criterion variable (here, I1) is specified immediately after the **model** statement, and the variable(s) that will serve as the predictor(s) are specified after the “=” sign.

The specification (Event=1) indicates that the probability of the correct answer should be computed. The default in **proc logistic** is to model the probability of the lowest value of the criterion, which would be the probability of the incorrect answer for DIF analyses.

```
proc logistic data = LR ;  
model I1(Event='1')=total;  
proc logistic data = LR ;  
model I1(Event='1')=total group;  
proc logistic data = LR;* outest=I1;  
model I1(Event='1')=total|group;  
run;
```

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 Fit Statistics		
Criterion	Intercept Only	Intercept and Covariates
AIC	1237.880	1135.010
SC	1243.053	1145.354
-2 Log L	1235.880	1131.010

Model Fit Statistics		
Criterion	Intercept Only	Intercept and Covariates
AIC	1237.880	1133.032
SC	1243.053	1148.549
-2 Log L	1235.880	1127.032

Model Fit Statistics		
Criterion	Intercept Only	Intercept and Covariates
AIC	1237.880	1135.031
SC	1243.053	1155.721
-2 Log L	1235.880	1127.031

<i>Model</i>	<i>-2LL</i>	<i>LR difference</i>	<i>p-value</i>
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. I created the macro below to automate the process in SAS. In the subsequent text I explain how it works.

```
options nodsnferr mcompilenote=all mlogic mprint;
```

```
data LRresult; set _null_;
```

```
%macro log;
```

```
%do item=1 %to 31;
```

```
proc logistic data=LR outest=a&item noprint;
```

```
model i&item(Event='1')=total;
```

```
proc logistic data=LR outest=b&item noprint;
```

```
model i&item(Event='1')=total group;
```

```
proc logistic data=LR outest=c&item noprint;
```

```
model i&item(Event='1')=total|group;
```

```
run;
```

```
data temp(keep=item log1 log2 log3);
```

```
merge a&item(rename=(_LNLIKE_=log1)) b&item(rename=(_LNLIKE_=log2))
```

```
c&item(rename=(_LNLIKE_=log3));
```

```
item=&item;
```

```
run;
```

```
data LRresult; set LRresult temp;
```

```
run;
```

```
proc datasets; delete temp a&item b&item c&item;
```

```
run;
```

```
%end;
```

```
%mend log;
```

```
%log;
```

The **options** command specifies that:

- error messages should be ignored and processing should continue if errors are found (**nodsnferr**)
- notes for any errors and info on the macro's instructions should be displayed in the log (**mcompilenote=all**)
- execution of the macro commands should be displayed in the log (**mlogic**)
- the commands generated by the macro should be displayed in the log (**mprint**)

The command **%macro log** creates a macro named log that will carry out the sequence of commands following it. The macro is ended by the command **%mend log**.

The symbol “%” is used to identify macro commands. The sequence of commands begins with a do loop (**%do item= 1 %to 31**). The part of the command “item = 1 %to 31” specifies a looping variable (item) that begins at 1 and ends at 31. Thus, the commands within the loop will be run 31 times. Note that there must be a % before the word “to.”

The **%do** command is followed by the **proc freq** commands used previously to obtain MH statistics for item 1. Here, the name of the variable “item” is replaced by “**I&item**.” This indicates that the letter “I” should be concatenated with the current value of the looping variable “item.” Recall that the 31 items are named i1, i2, i3, and so on. The specification “**I&item**” will create these names sequentially as the loop iterates from 1 to 31.

The three sets of **proc logistic** commands come next. These are the same commands used previously to obtain the DIF statistics for Item 1, but there are a few additions.

The subcommand “**outest=a&item noprint;**” specifies that a SAS datafile should be created and named “a” plus the current value of the looping variable “item.” Thus, a datafile will be created for each of the 31 items as the macro loops through its 31 iterations. These datafiles will be named “a1,” “a2,” ... “a31.”

The **noprint** statement suppresses the printing of the full output from the logistic regression. The output is instead written to the 31 new datafiles.

The subcommand “**model i&item(Event='1')=total;**” specifies the name of the criterion variable as “I” plus the current value of the looping variable “item” in the same way as was done for the previous output datafiles. This will cause the macro to loop through the 31 items, using each one in turn as the criterion variable.

The next four lines repeat the logistic commands for the second and third logistic regressions, naming the output files “b1” up to “b31” and “c1” up to “c31.”

The next set of commands in the macro is:

```
data temp(keep=item log1 log2 log3);  
merge a&item(rename=(_LNLIKE_=log1)) b&item(rename=(_LNLIKE_= log2))  
c&item(rename=(_LNLIKE_=log3));  
item=&item;  
run;
```

These commands cause the macro to create a temporary file (“temp”) at each iteration of the loop into which selected output from the three datafiles containing the logistic regression output for each item are merged (“**merge a&item b&item, c&item**”).

The **rename** statements change the SAS output name of the -2 log likelihood values from “_LNLIKE_” to “log1,” “log2,” or “log3.”

The **keep** statement indicates that the item number and the three values of -2 log likelihood should be saved, and all other variables dropped.

Finally, commands:

```
data LRresult; set LRresult temp;  
run;  
proc datasets; delete temp a&item b&item c&item;  
run;
```

cause create a new datafile called "LRresult" into which each of the temp files are subsetting. The temp files and the “a,” “b,” and “c” files are then deleted as the information is now contained in the datafile “LRresults.”

The commands:

```
%end;  
%mend log;  
%log;
```

end the loop (**%end**), end the macro (**%mend log**), and call the macro (**%log**).

Calculating the LR DIF statistics

The commands below set the data “LRresult” in the file “result” and calculate the values of uniform (Uchi) and nonuniform (NUchi) DIF by subtracting the appropriate values of $-2 \times LL$ (see chapter 16 for more details). The p -values of Uchi and NUchi are calculated using the **probchi** function with one degree of freedom.

```
data result; set LRresult;  
Uchi=(log1-log2);  
NUchi=(log2-log3);  
prob1=1-probchi(Uchi,1);  
prob2=1-probchi(NUchi,1);  
run;
```

Saving values into an rtf file

Finally, the last set of commands creates an rtf file that contains a table with the item number, values of log1, log2, and log3, and values of Uchi, NUchi, and their associated p -values.

Values are sorted in descending order by Uchi and NUchi such that the items with the largest amounts of DIF are at the top.

Enclosing the relevant commands between the **ods rft** and **ods rft close** commands causes the rtf file to be created.

```
ods rft file="Your folder\LRoutput.rtf";  
proc sort data= result; by descending Uchi NUchi;  
proc print; format log1 log2 log3 Uchi NUchi prob1 prob2 F9.5;  
id item; var log1 log2 log3 Uchi NUchi prob1 prob2 ;  
run;  
ods rft close;
```