flexMIRT®: Flexible Multilevel Multidimensional Item Analysis and Test Scoring

User's Manual
Version 3.7

Authored by:
Carrie R. Houts, PhD
Li Cai, PhD

This manual accompanies a Release Candidate version of the flexMIRT^{TM} software. Please refer to the officially released product information (when available) for citation information.

Contents

List of Tables

List of Syntax Examples

List of Output

CHAPTER 1

Overview

Vector Psychometric Group, LLC, a North Carolina-based psychometric software and consulting company, is pleased to announce the immediate availability of flexMIRT$^{\text{TM}}$ Version 3.7, an update to our multilevel and multiple-group item response theory (IRT) software package for item analysis and test scoring. flexMIRT$^{\text{TM}}$ fits a variety of unidimensional and multidimensional IRT models to single-level and multilevel data using maximum marginal likelihood or modal Bayes via Bock-Aitkin EM (with generalized dimension reduction) or MH-RM estimation algorithms. The generalized dimension reduction EM algorithm, coupled with arbitrary user-defined parameter constraints, makes flexMIRT$^{\text{TM}}$ one of the most flexible IRT software programs on the market today and the MH-RM algorithm allows users to efficiently estimate high-dimensional models. Since Version 3.6, flexMIRT$^{\text{TM}}$ also implements fully Bayesian Markov chain Monte Carlo estimation methods to produce samples from the posterior distributions of item and group parameters. flexMIRT$^{\text{TM}}$ produces IRT scale scores using maximum likelihood (ML), maximum a posteriori (MAP), and expected a posteriori (EAP) estimation. Multiple imputation scoring is now available when using the MH-RM algorithm. flexMIRT$^{\text{TM}}$ (optionally) produces summed-score to IRT scale score (EAP) conversion tables for unidimensional and multidimensional IRT models. As for the item types, flexMIRT$^{\text{TM}}$ can estimate any arbitrary mixtures of 3-parameter logistic (3PL) model, logistic graded response model (which includes 2PL and 1PL as special cases), and the nominal categories model (including any of its restricted sub-models such as the generalized partial credit model, partial credit model, and rating scale model) for both single-level and multilevel data, in any number of groups.

flexMIRT$^{\text{TM}}$ also has some of the richest psychometric and statistical features. Specialized commands are available to make the fitting of Diagnostic Classification Models (DCMs) intuitive and relatively straight-forward.

flexMIRT$^{\text{TM}}$ includes capabilities for conducting exploratory factor analysis with analytic or target rotations for orthogonal or oblique factors, as well as the ability to include latent regression covariates. In addition to fully implementing many recently developed multilevel and multidimensional IRT models, flexMIRT$^{\text{TM}}$ supports numerous methods for estimating item parameter standard errors: Supplemented EM, empirical cross-product approximation, Fisher (expected) information matrix, Richardson extrapolation, forward difference, and sandwich covariance matrix. A multitude of model fit statistics for dimensionality analysis, item-fit testing, and latent variable normality diagnosis are included in flexMIRT$^{\copyright}$. Its multiple-group estimation features easily facilitate studies involving differential item function (DIF) and test linking (including vertical scaling).

A key innovation in flexMIRT$^{\text{TM}}$ is its ability to relax the ubiquitous multivariate normality assumption made in virtually all multidimensional IRT models. With an extended dimension reduction algorithm, it supports the non-parametric estimation of latent density shapes using empirical histograms and Gaussian kernel-based smoothing for both unidimensional and hierarchical (i.e., bifactor and testlet response theory) item factor models in any number of groups, with support for constraints on group means and variances. Finally, it has a full-featured built-in Monte Carlo simulation module that can generate data from all models implemented in flexMIRT$^{\text{TM}}$.

flexMIRT$^{\text{TM}}$ is easy to use. It has an intuitive syntax. It can import data natively in space-, comma-, or tab- delimited formats. Windows-based flexMIRT$^{\text{TM}}$, with a friendly graphical user interface (GUI), is available in both 32-bit and 64-bit flavors. flexMIRT$^{\text{TM}}$ was designed with cross-platform compatibility and large-scale production use from the beginning. The computational engine of flexMIRT$^{\text{TM}}$ is written using standard C++, which ensures that it can run on any platform where a C++ compiler exists. A modern memory allocation scheme helps flexMIRT$^{\text{TM}}$ efficiently handle thousands of items and millions of respondents, with no imposed upper limit on the size of the problem.

flexMIRT$^{\text{TM}}$ is fast. For multidimensional and multilevel models that permit dimension reduction, flexMIRT$^{\text{TM}}$ automatically reduces the amount of quadrature computation to achieve a dramatic reduction in computational time. As modern CPU architecture trends toward multi-core design, flexMIRT$^{\text{TM}}$ implements two ways to use multi-threading to further speed up computations by spreading the load automatically to the multiple available cores or processing units.

The system requirements for flexMIRT$^{\text{TM}}$ are minimal. Currently, flexMIRT$^{\text{TM}}$ is only available for machines running Windows 7 or above (natively or via an emulator/virtual machine). Administrative rights to the machine are required for installation and .NET 4.5 (which should automatically update with Windows) is also required. The free space needed for installation is negligible and there is no minimum requirement regarding the amount of available RAM. An active internet connection is required for license validation - if an internet connection is not available for flexMIRT$^{\text{TM}}$ please contact us for assistance. Despite the minimal requirements for flexMIRT$^{\text{TM}}$, the estimation of complex models may need significant CPU resources, RAM, and processing time.

This user's manual is intended both as a guide to those trying to navigate the complex features implemented in flexMIRT$^{\text{TM}}$ for the first time, and as a reference source. It is assumed that the user is familiar with the theory and applications of IRT. Please visit the website `http://www.flexMIRT.com` for product updates, technical support, and feedback.

CHAPTER 2

flexMIRT$^{\text{TM}}$ Syntax and Datafile Structure

We begin by discussing expectations for the structure of data files being submitted to flexMIRT$^{\text{TM}}$ as well as the basic structure of flexMIRT$^{\text{TM}}$ syntax. This chapter is intended to familiarize new users with how data files should be arranged (allowed delimiters, missing data placeholder values, etc.) as well as provide a brief introduction to the general structure of flexMIRT$^{\text{TM}}$ syntax files.

2.1. Datafile Structure

Data should be arranged so columns represent items and each observation (or response pattern) has its own row in the dataset. Data files must use space, tab, or comma delimiters between variables; flexMIRT$^{\text{TM}}$ is not able to interpret Fortran-type statements for data processing, so individual values *must* be separated by one of the previously noted delimiters. Data files submittted to flexMIRT$^{\text{TM}}$ may have a header row (AKA column labels/names in the first row). If the data file does have a header row then users should indicate this when providing the datafile name and location by also including the command `Header = Yes;` - by default flexMIRT$^{\text{TM}}$ assumes there is no header row and failing to include this command when a header row is present will likely result in data read-in errors. Additionally, when `Header = Yes;` is used, flexMIRT$^{\text{TM}}$ will pull the variable names from the header row and it is not necessary that users provide a separate `Varnames` statement that names the variables in the body of the flexMIRT$^{\text{TM}}$ syntax.

Missing data are indicated by -9 by default, although the missing data code may be modified to any numeric value between -127 and 127. flexMIRT$^{\text{TM}}$ will not intepret missing indicators commonly used by other statistical programs, such as a blank or ".,' correctly; such values should be recoded in the datafile prior to attempting to submit the file to the program.

flexMIRT$^{\text{TM}}$ has the capability to analyze/score only a subset of the available variables in a dataset. The optional `Select` and `Exclude` statements may be used when defining the analysis and allow for a subset of the variables initially read from the data file to be utilized. The default is for all variables in the datafile to be used for any scoring/analyses. `Select` allows users to specify a subset of the available variables to be analyzed/scored while `Exclude` allows users to tell which variables to exclude from analysis/scoring. For instance, if we had variables ID and V1-V12 in our data set, we could use either `Select = V1-V12;` or `Exclude= ID;` so that flexMIRT$^{\text{TM}}$ will not attempt to include the ID variable in the IRT analyses. Note that only one of these statements should be used in reference to a given data set; that is, `Select` and `Exclude` statements should not both be used in reference to the same datafile.

Although flexMIRT$^{\text{TM}}$ has some data manipulation capabilities (recoding, rekeying response using answer keys, etc.), it is expected that the user has completed necessary data management checks prior to submitting data to the program. An important check is to ensure that all response options have observations.

2.2. Syntax Overview

Syntax files for flexMIRT$^{\text{TM}}$ may be created in the GUI by selecting "New" or created using a flat text program (e.g., Notepad). If "New" is selected within the flexMIRT$^{\text{TM}}$ GUI, a command file containing the skeleton code for a basic analysis opens. Once you have modified this command file, it must be saved prior to being able to submit the analysis to flexMIRT$^{\text{TM}}$; the "Run" button is disabled until the file has been saved to prevent the skeleton code from being over-written. The file name extension for flexMIRT$^{\text{TM}}$ syntax files is *.flexmirt but the program is able to run syntax files with other extensions (e.g., *.txt).

Generally speaking, flexMIRT$^{\text{TM}}$ syntax files are divided into four required sections, each of which provides instructions for distinct parts of the analysis. The four sections, which will be discussed in more detail, are

1. `<Project>` – where you provide a title and a general description of the analysis for record-keeping.
2. `<Options>` – where the type of analysis and options of the analysis are specified.
3. `<Groups>` – where the input data and the IRT models are specified.
4. `<Constraints>` – where parameter constraints (such as equality or fixed parameter values) or univariate priors may be specified.

All four section headers must be present in every command file (e.g., even if no constraints are applied to the parameters, the `<Constraints>` section header must still be declared in the syntax). The other requirement is that all statements in the flexMIRT^{TM} syntax file must end with a semi-colon. Finally, the commands in the syntax are not case-sensitive (but file names on certain operating systems may be).

Within the `<Project>` section, there are two required commands, `Title = " ";` and `Description = " ";`. The name of the title and the contents of the description must be enclosed in double quotation marks.

The `<Options>` section is where the more technical aspects of the analysis are specified. It is in this section that analysis details such as convergence criteria, scoring methods, standard error estimation methods, or requests for various fit indices may be specified. The only required command is `Mode = ;`, where the user is asked to specify what type of analysis will be conducted. The four options available are `Classical`, `Calibration`, `Scoring`, and `Simulation`. Depending on the `Mode` of the analysis, there may be additional required commands, such as the type of IRT scoring to be conducted in a `Scoring` analysis. These details will be covered as they emerge in the examples.

As noted previously, `<Groups>` is the section where information regarding the groups of item response data is provided. This includes the name(s) of the file(s) containing the data set(s), the names of items, the number of groups and participants, and the number of response options for each item, among other things. Following the `<Groups>` section header, a group name must be provided for the first group (even if there is only one group). The group name is enclosed in percent signs (%), must start with a letter, and should be something easy to remember (e.g., `%Group1%` or `%Grade4%` or `%Female%` or `%US%`). The group names become important in specifying constraints in multiple-group analyses. The name is arbitrary, but spaces are not allowed when specifying group names. After a group name has been given, the data file name where the item responses are located is specified, as well as the number of participants in the group, variable names, the number of possible response categories for each item, and the IRT model to be estimated (e.g., `ThreePL`, `Graded(5)`, etc).

The final section of flexMIRT^{TM} syntax, `<Constraints>`, is reserved for specifying constraints and prior distributions on the item parameters and latent variable means and (co)variances. Although the section header is required, there are no mandatory commands. Constraints and prior distributions may be placed on almost any aspect of the model, including means, covariances, and the various item parameters. Details on how to specify constraints will be discussed in examples that utilize constraints.

CHAPTER 3

Basic Calibration and Scoring

In this chapter, we will familiarize users with flexMIRT^{TM} and its syntax by providing examples that demonstrate some standard classical test theory (CTT) and IRT analyses, such as calibration and scoring. All syntax and data files used in this manual are available in the Support section on the flexMIRT^{TM} website (`https://vpgcentral.com/software/flexmirt/support-v3-72/`).

3.1. Dichotomous Classical Test Theory Analysis

flexMIRT^{TM} now offers the ability to run CTT analyses, allowing users to obtain estimates of coefficient alpha, item-total correlations, and alpha-if-deleted, as well as printing observed frequencies and weighted summed score statistics for each item. The first several examples will use data from two test forms that resemble those used in North Carolina end-of-grade testing. The item data are responses to 12 multiple choice items collected from 2844 test-takers, with responses in the data file coded as correct (1) or incorrect (0).

Example 3-1: Classical test theory syntax

```
<Project>
Title = "CTT example";
Description = "12 dichotmous items, N = 2844 ";

<Options>
Mode = Classical;

<Groups>
%Group1%
```

```
File = "g341-19.dat";
Varnames = v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12;
N = 2844;
Ncats (v1-v12) = 2;
Model (_ALL_) = Graded(2);
ItemWeights (v2) = (0,0.5);
<Constraints>
```

Following the previous discussion, section headers (denoted by their enclosure within `<` and `>`) precede the commands. The `<Project>` header is followed by a title and description, both of which will be printed in the output. In the `<Options>` section, the `Mode` command declares this to be a classical test theory run via `Mode = Classical;`. After the `<Groups>` section header, a name for the first (and only) group is provided, which is enclosed within `%`s. Then the name of the data file (`File = " ";`), variable names (`Varnames = ;`), number of examinees (`N = ;`), the number of observed categories for each item (`Ncats( ) = ;`), and the IRT model ultimately intended to be fit to each item (`Model( ) = ;`) are provided. In this example, only the file name was provided in the `File` statement. When the command file (*.flexmirt) and the data file are in the same folder, the name of the data file is sufficient for flexMIRT^{TM} to locate it. If, however, the command file and the data are in different directories, it is necessary to specify the full path in the `File` statement. For example, an alternate statement might be `File = "C:\Users\VPG\Documents\flexMIRT\example 2-1\g341-19.dat";`.

Note that for the `Ncats` statement, a variable naming shortcut was used, which is only available when variable names end with consecutive integers. As seen in the syntax, `v1, v2, ..., v12` can be shortened to `v1-v12`. This shorthand may be employed in any statement where variable names are provided. In the `Model` statement, a different variable name shorthand is employed. Here, rather than listing out the variables to be analyzed we use a macro-type variable _ALL_ , which tells flexMIRT^{TM} to apply the `Model` statement to all items. This shorthand may be useful when variable names are not sequential (i.e., when something like V1-V12 cannot be used) but is somewhat limited in that all variables must have the same properties (e.g., all items must have the name number of response categories, be fit with the same item model). Note, also, that the_ALL_ macro variable is only available for use in the `<Groups>` section of flexMIRT^{TM} syntax; attempts to use it in the `<Constraints>` section will be ignored and will likely result in errors or unintended models.

In most cases, users will likely want to retain the default weighting of responses (e.g., a 0-response is given a weight of 0; a 1-response is given a weight of 1) but the `ItemWeights(vars) = ;` command in the `<Groups>` section of the syntax can be used to assign alternate weights (for instance, a 0-response is given a weight of 0; a 1-reponse is given a weight of 0.5 would result from the statement, `ItemWeights(v2) = (0, 0.5);`. Response category values as well as default and user-specified item weights will be reflected in the individual item reporting.

Output 3.1: CTT Output - Item Frequencies and Weighted Summed Score Statistics

```
Output Files
Text results and control parameters: 2PLM_example_CTT-ctt.txt

Item and (Weighted) Summed-Score Statistics for Group 1: Group1

Item 1:               v1
        Category/Weight:  0/ 0.0  1/ 1.0       .
                  Freq.:    1251    1593       0
Listwise-Complete Freq.:    1251    1593
    Average (wtd) Score:    6.72    9.12
  Std. Dev. (wtd) Score:    2.00    1.83

Item 2:               v2
        Category/Weight:  0/ 0.0  1/ 0.5       .
                  Freq.:     251    2593       0
Listwise-Complete Freq.:     251    2593
    Average (wtd) Score:    5.86    8.28
  Std. Dev. (wtd) Score:    2.35    2.12

Item 3:               v3
        Category/Weight:  0/ 0.0  1/ 1.0       .
                  Freq.:    1456    1388       0
Listwise-Complete Freq.:    1456    1388
    Average (wtd) Score:    7.00    9.18
  Std. Dev. (wtd) Score:    2.04    1.89

...
```

The first section of the CTT output, opened by the Output viewer and saved to a *-ctt.txt output file, provides observed item frequencies, both simple observed and list-wise complete (meaning for only those observations with non-missing responses to all items), as well as descriptive statisitics (mean and standard deviation) of weighted total summed scores associated with each item response. As noted previously, in most cases, users will likely want to retain the default weighting of responses (e.g., a 0-response is given a weight of 0; a 1-reponse is given a weight of 1) but if item weights were used, the user-supplied weights are reflected in the individial item reporting (such as shown for Item 2 in the “Category/Weight” line), otherwise, default weights are reported.

Following the summary information for the individual items, coefficient alpha and related values are reported for the scale as a whole. Additionally, the average response and standard deviations for each item are reported - in the case of 0/1 items this average would be P^{+}. Additionally, in this same table, item-total correlations as well as the expected alpha if the item were to be removed from the scale are also reported.

Output 3.2: CTT Output - Coefficient Alpha and Related Values

```
Classical item statistics are computed only for the listwise-complete data (N = 2844.00):

Coefficient alpha:  0.6515

          Response        Item-Total       Alpha
Item   Average  Std.Dev. Correlation  If Deleted
   1     0.560     0.496      0.3408      0.6224
   2     0.456     0.142      0.2457      0.6457
   3     0.488     0.500      0.2888      0.6336
   4     0.683     0.465      0.3331      0.6238
   5     0.570     0.495      0.2913      0.6329
   6     0.831     0.375      0.3344      0.6250
   7     0.951     0.216      0.2661      0.6400
   8     0.782     0.413      0.2848      0.6328
   9     0.904     0.295      0.3307      0.6290
  10     0.866     0.341      0.2974      0.6316
  11     0.444     0.497      0.2903      0.6332
  12     0.531     0.499      0.3287      0.6250
```

3.2. Single-Group 2PL Model Calibration

Within the IRT framework, the dichotomous responses of the previously described data set may be fitted with either the 2PL model, if guessing is assumed to be negligible, or with the 3PL model, which explicitly models guessing with an additional parameter. Example syntax for fitting the 2PL model is presented first.

Example 3-2: 2PL Calibration Syntax

```
<Project>
Title = "2PL example";
Description = "12 items, 1 Factor, 1 Group 2PL Calibration";

<Options>
Mode = Calibration;

<Groups>
%Group1%
File = "g341-19.dat";
Varnames = v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12;
N = 2844;
Ncats(v1-v12) = 2;
Model(v1-v12) = Graded(2);

<Constraints>
```

As with the previous example, section headers (denoted by their enclosure within < and >) precede the commands. Building to the previous syntax example, for the `Model` statement , the Graded Response Model, with two categories, is specified as `Graded(2)`. This model is formally equivalent to the 2PL model and, as such, flexMIRT^{TM} does not implement the 2PL model separately. Although not used in this example, it is useful to be aware that multiple `Ncat` and `Model` statements can be specified within one group, which is needed for an analysis of mixed item types. For example, if items 1 through 6 in the current data were dichotomous and items 7 through 12 were five-category polytomous items, two `Ncat` statements (i.e., `Ncats(v1-v6) = 2; Ncats(v7-v12)= 5;`) and two `Model` statements (i.e., `Model(v1-v6) = Graded(2); Model(v7-v12) = Graded(5);`) would be provided to fit appropriate models to the items.

There are no constraints placed on this model, so after the `<Constraints>` section header is declared, the command file syntax ends. At the completion of estimation, flexMIRT^{TM} automatically opens the results in the output viewer. The output is also saved into a text file named with the command file name and "*-irt.txt" as the extension. The command file used in the first example is "2PLM_example.flexmirt" and the output is saved to "2PLM_example-irt.txt". The output is presented in a straight-forward fashion, so those familiar with IRT and its parameters and fit values should find it easily understandable. For completeness, however, we will review the output and highlight relevant sections. Due to the length of the output file, it will be separated into parts which are discussed in turn.

Output 3.3: 2PL Calibration Output - Summary and Controls

```
flexMIRT(R) Engine Version 3.72 (64-bit)
Flexible Multilevel Multidimensional Item Response Modeling and Test Scoring
(C) 2013-2024 Vector Psychometric Group, LLC., Chapel Hill, NC, USA

2PLM example
12 items 1 Factor, 1 Group 2PLM Calibration

Summary of the Data and Dimensions
   Missing data code        -9
     Number of Items        12
     Number of Cases      2844
 # Latent Dimensions         1

Item  Categories       Model
   1           2      Graded
   2           2      Graded
   3           2      Graded
   4           2      Graded
   5           2      Graded
   6           2      Graded
   7           2      Graded
   8           2      Graded
   9           2      Graded
  10           2      Graded
  11           2      Graded
  12           2      Graded

Bock-Aitkin EM Algorithm Control Values
Maximum number of cycles:   500
Convergence criterion:    1.00e-04
Maximum number of M-step iterations:   100
Convergence criterion for iterative M-steps:    1.00e-07
Number of rectangular quadrature points:    49
Minimum, Maximum quadrature points:   -6.00,    6.00
Standard error computation algorithm: Cross-product of gradients

CPU processing times in seconds (may not equal clock time)
E-step computations:      0.13
M-step computations:      0.01
Standard error computations:      0.17
Goodness-of-fit statistics:      0.00
Total:       0.31

Output Files
Text results and control parameters: 2PLM_example-irt.txt
Technical information in a file: 2PLM_example-dbg.txt

Convergence and Numerical Stability
flexMIRT(R) engine status: Normal termination
Number of cycles completed:    27
Maximum parameter change (P#):   0.00008371 (   13)
First-order test: Convergence criteria satisfied
Condition number of information matrix: 46.0221
```

```
Log determinant of information matrix: 128.2635
Second-order test: Solution is a possible local maximum
```

In the first line of every output file, flexMIRT$^{\text{TM}}$ prints the version number and bit-version (64 bit or 32 bit) of the engine used to complete the analysis. If comparing results across different systems, users should ensure that the same engine version of flexMIRT$^{\text{TM}}$ is being used and also be aware that the 32-bit and 64-bit C++ compilers generate different object code so the order of the floating point operations may be different, which can result in small differences among reported values across the bit-versions of the same flexMIRT$^{\text{TM}}$ engine version. It is recommended that users occasionally log into their flexMIRT$^{\text{TM}}$ account and check if their version is the most current version of the program available and update (for free, provided a valid license) when necessary.

Following the flexMIRT$^{\text{TM}}$ version and copyright information, the output begins by printing the title and description provided in the `<Project>` section of the syntax file. A broad summary of the data and model is provided on the next 4 lines, listing the missing value code, number of items, sample size, and total number of dimensions. In the next section, the number of categories and the IRT models are listed for each item. The various control values are listed (e.g., convergence criteria, the maximum number of iterations, number of quadrature points used, number of free parameters in the model) – if any control values were changed from defaults, the information printed here serves as verification.

Next, the program processing time is listed, both in total and broken down into various stages. Any output files that were generated by the analysis are named in the next section. The final part of this subsection of the output reports if flexMIRT$^{\text{TM}}$ terminated normally and if the specified convergence criteria were met and a stable solution has been obtained. Specifically, the reported first-order test examines if the gradient has vanished sufficiently for the solution to be a stationary point. The second-order test tests if the information matrix is positive definite, a prerequisite for the solution to be a possible maximum. For the second-order test, reporting that the solution is a possible local maximum simply means that the program reached a statistically desirable solution. The other possible message that may be printed for the outcome of the second-order test is "Solution is not a maximum; caution is advised." **If a warning message is received for either the first- or second-order test, all parameter estimates should be taken as provi-**

sional and should not be used as final estimates, for future scoring, etc. but, rather, should be used to diagnose possible issues with the model/items.

The next section of the output provides the estimated item and group parameters.

Output 3.4: 2PL Calibration Output - Item and Group Parameters

```
*** Random effects calibration in Group 1: Group1

2PLM example
12 items 1 Factor, 1 Group 2PLM Calibration

2PL Items for Group 1: Group1
Item   Label   P#      a    s.e.   P#      c    s.e.      b    s.e.
  1      v1    2    1.05    0.07    1    0.30    0.05   -0.29    0.05
  2      v2    4    1.23    0.11    3    2.88    0.11   -2.35    0.15
  3      v3    6    0.84    0.06    5   -0.05    0.04    0.06    0.05
  4      v4    8    1.04    0.07    7    0.94    0.05   -0.90    0.06
  5      v5   10    0.85    0.06    9    0.33    0.04   -0.39    0.06
  6      v6   12    1.34    0.10   11    2.09    0.09   -1.56    0.09
  7      v7   14    1.90    0.17   13    4.29    0.21   -2.26    0.12
  8      v8   16    0.97    0.08   15    1.51    0.06   -1.56    0.10
  9      v9   18    1.89    0.15   17    3.35    0.16   -1.78    0.08
 10     v10   20    1.32    0.10   19    2.40    0.09   -1.82    0.10
 11     v11   22    0.87    0.06   21   -0.26    0.04    0.30    0.05
 12     v12   24    1.01    0.07   23    0.15    0.05   -0.15    0.05

2PLM example
12 items 1 Factor, 1 Group 2PLM Calibration

Group Parameter Estimates:
  Group   Label   P#     mu    s.e.   P#   s2    s.e.     sd    s.e.
     1     Group1        0.00    ----      1.00   ----   1.00    ----
```

Following a reprinting of the title and description, each item is listed in its own row, with the provided item label (“Label”), flexMIRT^{TM} assigned parameter number (“P#”), estimated item parameters, and standard error values going across. The IRT model fit in this example was the 2PLM, so the reported item parameters are the slope (labeled `a`) and intercept (labeled `c`). In addition, for unidimensional models the threshold/difficulty parameter (labeled `b`) is also reported; this value may be found from the other two parameters as $b = c/-a$. Below the item parameter table, the group parameters are printed. In this example, we had only one group, so the mean and variance were fixed at 0 and 1, respectively, for model identification. That these parameters were not estimated is indicated by both the lack of an assigned parameter number and the dashes in the standard error columns.

Output 3.5: 2PL Calibration Output - Information Functions

```
2PLM example
12 items, 1 Factor, 1 Group 2PL Calibration

Item Information Function Values at 15 Values of theta from -2.80 to 2.80 for Group 1: Group1
           Theta:
  Item  Label  -2.8  -2.4  -2.0  -1.6  -1.2  -0.8  -0.4  -0.0   0.4   0.8   1.2   1.6   2.0   2.4   2.8
     1       v1  0.07  0.10  0.13  0.18  0.22  0.26  0.28  0.27  0.24  0.20  0.16  0.12  0.08  0.06  0.04
     2       v2  0.35  0.38  0.36  0.31  0.24  0.17  0.12  0.08  0.05  0.03  0.02  0.01  0.01  0.00  0.00
     3       v3  0.05  0.07  0.09  0.11  0.13  0.15  0.17  0.18  0.17  0.16  0.14  0.12  0.10  0.08  0.06
     4       v4  0.12  0.16  0.20  0.24  0.26  0.27  0.25  0.22  0.18  0.13  0.10  0.07  0.05  0.03  0.02
     5       v5  0.07  0.09  0.12  0.14  0.16  0.17  0.18  0.17  0.16  0.14  0.12  0.09  0.07  0.06  0.04
     6       v6  0.24  0.33  0.41  0.45  0.42  0.35  0.26  0.18  0.11  0.07  0.04  0.03  0.02  0.01  0.01
     7       v7  0.70  0.89  0.85  0.62  0.38  0.20  0.10  0.05  0.02  0.01  0.01  0.00  0.00  0.00  0.00
     8       v8  0.17  0.20  0.22  0.24  0.23  0.21  0.17  0.14  0.11  0.08  0.06  0.04  0.03  0.02  0.01
     9       v9  0.39  0.64  0.85  0.87  0.67  0.42  0.23  0.12  0.06  0.03  0.01  0.01  0.00  0.00  0.00
    10      v10  0.29  0.38  0.43  0.43  0.37  0.29  0.20  0.13  0.08  0.05  0.03  0.02  0.01  0.01  0.00
    11      v11  0.04  0.06  0.08  0.10  0.13  0.15  0.17  0.19  0.19  0.18  0.16  0.14  0.11  0.09  0.07
    12      v12  0.06  0.09  0.12  0.16  0.20  0.23  0.25  0.25  0.24  0.20  0.17  0.13  0.09  0.07  0.05

Test Information:  3.56  4.38  4.86  4.83  4.41  3.87  3.38  2.97  2.61  2.29  2.01  1.77  1.58  1.42  1.30
   Expected s.e.:  0.53  0.48  0.45  0.45  0.48  0.51  0.54  0.58  0.62  0.66  0.71  0.75  0.80  0.84  0.88
Marginal reliability for response pattern scores: 0.64
```

The item and test information functions, values for which are presented in Output 2.3, are the focus of the next section of the output file. The title and description are printed once again, followed by the label for the information table. As may be seen, the values of theta used in the table range from -2.8 to 2.8, increasing in steps of 0.4. The information function values for each item are presented on the appropriate row and the Test Information Function (TIF) values, which sum the item information values at each theta value (plus the contribution from the prior), are printed towards the bottom of the table. Additionally, the expected standard error associated with each theta value is printed. Finally, the marginal reliability, which is defined as the prior variance minus the averaged error variance divided by the prior variance, is reported underneath the table (see, for example, Thissen & Orlando, 2001 for additional details). After the information and reliability section, the overall goodness of fit (GOF) indices are presented, which appear in the next output box.

Output 3.6: 2PL Calibration Output - Goodness of Fit Indices

```
Statistics based on the loglikelihood of the fitted model:
                        -2loglikelihood:    33408.05
      Akaike Information Criterion (AIC):    33456.05
    Bayesian Information Criterion (BIC):    33598.92

Full-information fit statistics of the fitted model:
                 Degrees
          G2  of freedom Probability       F0hat       RMSEA
     2062.50         696      0.0001      0.7252        0.03
                 Degrees
          X2  of freedom Probability       F0hat       RMSEA
     7710.31        4071      0.0001      2.7111        0.02
```

When no option is specified, the reporting of GOF values will provide only basic fit indices. These include the values for the $-2\times$Log Likelihood, the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and when appropriate, the Pearson X^2 statistic (labeled `X2`), the likelihood ratio statistic (labeled `G2`), their associated degrees of freedom, the estimated population discrepancy function value (labeled `F0hat`), and the RMSEA (Root Mean Square Error of Approximation; e.g., Browne & Cudeck, 1993).

3.3. Single-Group 3PL Model Calibration

The data in the previous section may also be suitable for the 3PL model, which explicitly models examinees' guessing.

Example 3-3: Single Group 3PL Calibration Syntax

```
<Project>
Title = "3PL example";
Description = "12 items, 1 Factor, 1 Group Calibration,
saving parameter estimates";

<Options>
Mode = Calibration;
GOF = Extended;
SavePRM = Yes;

<Groups>
%Group1%
File = "g341-19.dat";
Varnames = v1-v12;
N = 2844;
Ncats(v1-v12) = 2;
Model(v1-v12) = ThreePL;

<Constraints>
Prior Group1, (v1-v12), Guessing : Beta(1.0,4.0);
```

The syntax for this run differs from the previous one in several ways. In the `<Options>` section, the parameter estimates are to be saved to a separate output file (`savePRM = Yes;`) and additional GOF statistics were requested by specifying `GOF = Extended;` as an option. In the `<Groups>` section the fitted model is now the 3PL model. (`Model` keyword is `ThreePL`). As noted previously, our current data is scored (0/1) data, such as from multiple choice items - if raw multiple choice data is to be supplied, users will find the `Key` command (see Details of the Syntax chapter) useful for providing flexMIRT^{TM} with a scoring key.

In the `<Constraints>` section, a prior distribution is imposed on the `Guessing` parameters for all items. In this example the `Beta(1.0,4.0)` distribution was used, which corresponds to a prior sample size of 5, with $1.0/(1.0 + 4.0) = 0.2$ as its mode. This prior indicates that the prior guessing probability is equal to 0.2, as these multiple-choice items have 5 options. The `Normal` and `LogNormal` distributions are also available when specifying priors. In relation to the lower asymptote, the `Normal` prior applies a normal distribution prior on the *logit* of the lower asymptote parameter. A prior of `Normal(-1.39, 0.5)` is a reasonable prior for the lower asymptote, with a mode around 0.20 in the typical g metric. Other common normal priors for the logit-g parameter would be N(-1.09, 0.5) for items with 4 possible response options, N(-0.69, 0.5) for items with 3 response options, and N(0.0, 0.5) for True/False type items. The standard deviation of 0.5 is standard across all noted priors and is based primarily on experience, in that it provides a prior that is diffuse enough to allow estimated guessing parameters to move away from the mean value if necessary but not so diffuse as to be uninformative.

The output for this analysis is quite similar to that of the previous run, so only differences will be highlighted. As noted in the syntax discussion, flexMIRT^{TM} has been instructed to save the item and group parameter values into a separate file. This results in an additional file name, with the "*-prm.txt" extension, being listed in the output files. The layout of the item parameter output table (presented in Output 2.5) is essentially the same as before, with the slopes, thresholds, and difficulty values and their standard errors reported first. In the last four columns of the table, the estimates for the lower asymptote (both in its logit form and in terms of the pseudo-guessing probability) are reported, as well as standard errors.

Output 3.7: Single Group 3PL Calibration Output- Item Parameters

```
...
Output Files
Text results and control parameters: 3PLM_example-irt.txt
Text parameter estimate file: 3PLM_example-prm.txt
...
3PL example
12 items, 1 Factor, 1 Group Calibration, saving parameter estimates

3PL Items for Group 1: Group1
Item  Label  P#      a   s.e.  P#      c   s.e.      b   s.e.  P# logit-g   s.e.      g   s.e.
   1     v1   3   1.84   0.30   2  -0.56   0.26   0.31   0.10   1   -1.09   0.22   0.25   0.04
   2     v2   6   1.27   0.12   5   2.77   0.20  -2.17   0.26   4   -2.00   1.33   0.12   0.14
   3     v3   9   1.50   0.26   8  -0.98   0.29   0.65   0.11   7   -1.19   0.23   0.23   0.04
   4     v4  12   1.31   0.19  11   0.46   0.22  -0.35   0.21  10   -1.19   0.45   0.23   0.08
   5     v5  15   1.35   0.22  14  -0.42   0.25   0.31   0.15  13   -1.08   0.28   0.25   0.05
   6     v6  18   1.39   0.15  17   1.91   0.18  -1.37   0.23  16   -1.93   1.09   0.13   0.12
   7     v7  21   1.96   0.18  20   4.27   0.25  -2.18   0.17  19   -2.22   1.46   0.10   0.13
   8     v8  24   1.05   0.13  23   1.24   0.23  -1.19   0.34  22   -1.56   0.91   0.17   0.13
   9     v9  27   1.94   0.18  26   3.31   0.19  -1.71   0.15  25   -2.59   1.59   0.07   0.10
  10    v10  30   1.35   0.13  29   2.28   0.17  -1.69   0.22  28   -2.23   1.36   0.10   0.12
  11    v11  33   1.77   0.32  32  -1.40   0.34   0.79   0.08  31   -1.25   0.19   0.22   0.03
  12    v12  36   1.55   0.24  35  -0.52   0.23   0.33   0.11  34   -1.37   0.28   0.20   0.05
```

In addition to the added parameters and output files, the `GOF` statement was set to `Extended`, so additional fit statistics, namely marginal X^2 values and the local dependence (LD) statistics of Chen and Thissen (1997) are included in the output. A table containing these values is printed between the group parameter values and the Information Function values table. A large section of this table is presented below. The values in the second column of this table are the item-specific marginal X^2, which may be tested against a chi-square variable with degrees of freedom equal to the number of categories for that item minus 1. In this case, all items fit well at the level of univariate margins.

Output 3.8: Single Group 3PL Calibration Output- Extended GOF Table (Excerpt)

```
Marginal fit (Chi-square) and Standardized LD X2 Statistics for Group 1: Group1
     Marginal
Item  Chi2      1       2       3       4       5       6       7       8       9      10
   1   0.0
   2   0.0   -0.7n
   3   0.0   -0.7p    2.0n
   4   0.0    0.9p   -0.6n    0.2p
   5   0.0   -0.6n   -0.0n   -0.1n    5.3p
   6   0.0   -0.5p   -0.7n   -0.7n   -0.5p   -0.3n
   7   0.0    0.8n   -0.4p   -0.4n   -0.6n    0.3n   -0.7p
   8   0.0   -0.5p   -0.6p   -0.7p    0.6n   -0.4p    0.7p    0.0p
   9   0.0   -0.6p    0.7p   -0.4n   -0.4n   -0.5n   -0.7p    4.4p    0.2n
  10   0.0    2.5n    5.0p   -0.7n    0.1n   -0.5n   -0.0n   -0.0p   -0.7n    1.4p
  11   0.0   -0.6p    3.5n   -0.1n   -0.3p   -0.4p    0.8p    0.0n   -0.7p   -0.7n   -0.3p
  12   0.0   -0.7p   -0.6n    3.7p    0.3n   -0.0n   -0.3p   -0.7p   -0.6p    0.4n   -0.1p
```

The remaining entries of the table provide pairwise diagnostic information for possible local dependence. The values reported are *standardized* LD X^2 values, one for each item pair, followed by either the letter "p" or the letter "n". Phi correlations are computed for both the model-implied and observed bivariate tables - if the model implied correlation is lower than the observed correlation for a given item pair, a "p" is printed after the calculated X^2, meaning "positive" LD. If the model implied correlation is higher, an "n" is printed, indicating "negative" LD. Either a "p" or an "n" is printed after every value in the table. As the reported values are standardized X^2s, item pairs with values larger than 3.0 may need to be examined further for possible local dependence. It is important to recognize that the values reported are indices and are not intended to be used literally as X^2 values. Even with the provided guideline, users should also be aware that an examination of the overall pattern of findings, in addition to individual values, is important when interpreting results.

3.4. Single-Group 3PL EAP Scoring

There are situations when existing item parameters will be used and only scoring for a new set of data is desired. This next example will introduce a command file that does just that, using the saved parameter estimates from a prior calibration run.

Example 3-4: 3PL EAP Scoring Example

```
<Project>
Title = "3PL EAP scoring example";
Description = "12 items, 1 Factor, 1 Group 3PL Scoring";

<Options>
Mode = Scoring;
ReadPRMFile=  "3PLM_example-prm.txt";
Score = EAP;
saveSCO = Yes;

<Groups>
%Group1%
File = "g341-19.dat";
Varnames =v1-v12;
N = 2844;

<Constraints>
```

For scoring runs the `<Project>` section stays the same. The first change encountered is in the `<Options>` section, in that `Mode` is now set to `Scoring`. Using the command `ReadPRMFile`, the name of the file that contains the item parameter estimates is provided. In this case, the file was created by the previous calibration run. For the interested reader, details regarding the formatting and structure of the -prm file are found in the Simulation chapter of the manual.

The type of IRT scale score estimation is selected using the `Score = ;` command. For this example, Expected *A Posteriori* (EAP) scores will be saved. We may also specify `ML` for maximum likelihood scoring, `MAP` for Maximum *A Posteriori* scoring, `MI` for multiple imputations (only available when the MH-RM algorithim is used), or `SSC` for summed score to EAP conversion tables. We have also specified that the scores should be saved into a separate file, which will be labeled with the extension "*-sco.txt". The command requesting flexMIRT$^{\text{TM}}$ to output the estimated scores (`SaveSCO = Yes;`) is redundant for most scoring methods, but it is needed if individual scores are desired for the `SSC` scoring method. Without the `SaveSCO = Yes;` command, `SSC` will produce only the summed score to scale score conversion table. Notice that this may be a desirable feature in some situations, when only the conversion table is needed with actual scoring occurring at a much later stage.

The output of scoring runs differs from calibration runs so we have reproduced the entire scoring control output file, which has the extension"*-ssc.txt", in Output 2.7 below.

Output 3.9: 3PL EAP Scoring Output

```
3PL EAP scoring
12 items, 1 Factor, 1 Group Scoring

Summary of the Data and Dimensions
   Missing data code        -9
     Number of Items        12
     Number of Cases      2844
 # Latent Dimensions         1

Item  Categories       Model
   1           2         3PL
   2           2         3PL
   3           2         3PL
   4           2         3PL
   5           2         3PL
   6           2         3PL
   7           2         3PL
   8           2         3PL
   9           2         3PL
```

```
  10           2         3PL
  11           2         3PL
  12           2         3PL

Scoring Control Values
Response pattern EAPs are computed

Miscellaneous Control Values

Output Files
Text results and control parameters: 3PLM_score_example-ssc.txt
Text scale score file: 3PLM_score_example-sco.txt

3PL EAP scoring
12 items, 1 Factor, 1 Group Scoring

3PL Items for Group 1: Group1
    Item       a       c       b logit-g       g
       1    1.84   -0.56    0.31   -1.09    0.25
       2    1.27    2.77   -2.17   -2.00    0.12
       3    1.50   -0.98    0.65   -1.19    0.23
       4    1.31    0.46   -0.35   -1.19    0.23
       5    1.35   -0.42    0.31   -1.08    0.25
       6    1.39    1.91   -1.37   -1.93    0.13
       7    1.96    4.27   -2.18   -2.22    0.10
       8    1.05    1.24   -1.19   -1.56    0.17
       9    1.94    3.31   -1.71   -2.59    0.07
      10    1.35    2.28   -1.69   -2.23    0.10
      11    1.77   -1.40    0.79   -1.25    0.22
      12    1.55   -0.52    0.33   -1.37    0.20

3PL EAP scoring
12 items, 1 Factor, 1 Group Scoring

Group Parameter Estimates:
   Group               Label      mu      s2      sd
       1              Group1    0.00    1.00    1.00
```

As with the calibration output, the file starts with a general data summary and echoing of values that were provided to flexMIRT$^{\text{TM}}$. In the scoring mode, this section of the output is much shorter due to the fact that fewer control parameters are available for modification. Of note in this section is the line labeled "Scoring Control Values," under which the type of IRT scoring method used is printed. Below that, the output files generated are named and tables containing the item and group parameters, as read in from the parameter file, are printed to allow the user to verify the parameter values were read in as expected.

The individual scores were saved into a "*-sco.txt" file. The score file (3PLM_score_example-sco.txt) columns are arranged as follows: the group number, the observation number, the scale score, and the standard error associated with the scale score. Optionally, following the group number and the observation number, flexMIRT^{TM} will print the value of a user-supplied ID variable if the `caseID` command is utilized in the `<Groups>` section of the syntax.

3.5. Single-Group 3PL ML Scoring

As noted, maximum likelihood scores are also available. One issue that may occur when using ML scores is that extreme response patterns on either end of the continuum (e.g., all correct or all incorrect) are associated with undefined or unreasonable theta estimates. When `Score = ML;` the user is required to specify the desired maximum and minimum scores via the `MaxMLscore` and `MinMLscore` keywords, to obtain practical theta estimates for such cases.

Example 3-5: 3PL ML Scoring example(excerpt)

```
<Options>
Mode = Scoring;
ReadPRMFile=  "3PLM_example-prm.txt";
Score = ML;
SaveSCO = Yes;
 MaxMLscore= 3.5;
 MinMLscore= -5.5;

 <Groups>
 %Group1%
 File = "g341-19.dat";
 Varnames =v1-v12;
 N = 2844;
```

```

<Constraints>
```

This syntax is substantially the same as the previous example so only an excerpt is presented. The notable differences are the change from `Score = EAP;` to `Score = ML;` and the additional commands `MaxMLscore` and `MinMLscore` used to “rein in” the extreme ML estimates. Below, we present the first 10 cases of the associated -sco.txt file, where the scores were saved.

Output 3.10: 3PL ML Scoring Output

```
 1       1     2    -0.198234     0.573388
 1       2     3     1.427265     0.784667
 1       3     2     0.113716     0.680658
 1       4     3     1.427265     0.784667
 1       5     3    -0.675141     0.625639
 1       6     3     0.637464     0.606990
 1       7     3     1.382694     0.792826
 1       8     3    -1.205723     0.628273
 1       9     2     3.500000    99.990000
 1      10     3     0.859248     0.658306
```

Like the previous -sco file, the first column indicates group membership and the second column provides the flexMIRT^{TM} assigned observation number. The third column reports the number of iterations to reach the reported score. The fourth and fifth columns are the ML theta estimate and the associated SE, respectively. As can be seen for observation 9, our specified maximum of 3.5 was applied. Because the reported value was assigned as a result of our `MaxMLscore` statement, the reported SE is 99.99 to indicate that the score was assigned, rather than estimated.

3.6. Polytomous Classical Test Theory Analysis

The newly introduced `Mode = Classical;` is also available for use with polytomous item data, allowing users to obtain estimates of coefficient alpha, item-total correlations, and alpha-if-deleted, as well as printing observed frequencies and weighted summed score statistics for each item. All reported values are estimated using Pearson correlations. This example employs data from 3000 simulees, using parameter values reported in Table 1 of Edwards' (2009) paper as generating values. These data were constructed to mimic responses collected on the Need for Cognition Scale (Cacioppo, Petty, & Kao, 1984), which is an 18 item scale that elicits responses on a 5-point Likert-type scale, asking participants to indicate the extent to which the content of the item prompt describes them. We have introduced missing values and a response category that does not have any observed responses into this simulated data.

Example 3-6: Classical syntax - polytomous data

```
<Project>
Title = "CTT example";
Description = "CTT: 18 5-cat items, N = 3000";

<Options>
Mode = Classical;

<Groups>
%Group1%
File ="NCSsim_MISS.dat";
Varnames =v1-v18;
N = 3000;
Ncats(v1-v18) = 5;
Code(v1-v18) = (1,2,3,4,5), (0,1,2,3,4);
Model(v1-v18) = Graded(5);

<Constraints>
```

As with the previous CTT analysis syntax, we have set `Mode = Classical;`, specified the data file, variable names, sample size, the number of categories per item, and the expected model to be used when the items are calibrated in an IRT model. The only new command encountered in this example is the `Code = ;` statement in the `<Groups>` section. The internal representation of item response data in flexMIRT^{TM} is zero-based (i.e., response options must start at zero and go up sequentially). The raw item response data file for this example (NCSsim_MISS.dat) labels the categories as 1 through 5, so recoding is necessary. The `Code` statement does this for variables `v1-v18` by specifying the original values in the first pair of parentheses and, following a comma, the desired recoded values in a second set of parentheses. In the syntax, we have simply reduced each raw response value by one, making the reponse codes zero-based. An additional possible use of the `Code` command is something like `Code(v1-v5)=(1,2,3,4,5),(4,3,2,1,0);` which would both reverse code items (if needed), as well as make the recoded item response data zero-based.

Output 3.11: Polytomous Classical Analysis - Example Item Summaries

```
Item and (Weighted) Summed-Score Statistics for Group 1: Group1

Item 1:               v1
        Category/Weight:  0/ 0.0  1/ 1.0  2/ 2.0  3/ 3.0  4/ 4.0       .
                  Freq.:     254     772     688    1047     239       0
Listwise-Complete Freq.:     201     587     532     875     218
    Average (wtd) Score:   24.87   32.99   39.66   47.35   54.48
  Std. Dev. (wtd) Score:    9.96    9.87    9.06    8.46    7.87

Item 2:               v2
        Category/Weight:  0/ 0.0  1/ 1.0  2/ 2.0  3/ 3.0  4/ 4.0       .
                  Freq.:     137     575     561    1349     378       0
Listwise-Complete Freq.:     110     426     442    1089     346
    Average (wtd) Score:   21.25   29.40   36.98   44.92   53.90
  Std. Dev. (wtd) Score:    9.31    9.41    8.53    8.66    7.56

Item 3:               v3
        Category/Weight:  0/ 0.0  1/ 1.0  2/ 2.0  3/ 3.0  4/ 4.0       .
                  Freq.:     166     609       0    1135     503     587
Listwise-Complete Freq.:     166     609       0    1135     503
    Average (wtd) Score:   23.44   30.73     ---   43.99   52.15
  Std. Dev. (wtd) Score:    9.42    9.05     ---    8.17    8.22

...
```

As with the dichotomous classical example, individial item information is reported first. Due to the missing data in the NCSsim_MISS data file, the reported Frequency values and Listwise-Complete Frequency values now differ for all items. The missing responses are confined to item v3, which we can see by the 587 cases listed as "." in the v3 item summary. Additionally, in item v3, we find that recoded response category of 2 (originally 3 in the raw data file) has no observations; that is, no respondents used this middle category as a response option - this information will become important when we attempt to calibrate these items using this data.

Output 3.12: Polytomous Classical Analysis - Coefficient Alpha and Related Values

```
Classical item statistics are computed only for the listwise-complete data (N = 2413.00):

Coefficient alpha:  0.8975

          Response         Item-Total       Alpha
Item    Average  Std.Dev. Correlation  If Deleted
   1      2.133     1.133      0.6210      0.8895
   2      2.470     1.078      0.6593      0.8885
   3      2.497     1.259      0.6644      0.8878
   4      2.446     1.129      0.6471      0.8887
   5      2.530     1.057      0.5808      0.8909
   6      1.835     1.192      0.5304      0.8925
   7      2.230     1.165      0.5491      0.8918
   8      1.771     1.127      0.3527      0.8979
   9      1.767     1.156      0.4868      0.8938
  10      2.661     0.994      0.5584      0.8917
  11      2.542     1.057      0.6272      0.8895
  12      2.623     1.074      0.5871      0.8907
  13      1.821     1.104      0.5013      0.8933
  14      2.545     1.122      0.5682      0.8912
  15      2.453     1.093      0.6101      0.8900
  16      1.920     1.207      0.3851      0.8973
  17      2.317     1.198      0.5355      0.8923
  18      2.370     1.119      0.3081      0.8992
```

The coefficent alpha and related values follow the item summaries in the in -ctt.txt output file. As can be seen by the output label, flexMIRT^{TM} reports such values using only the listwise-complete data - as we found in the item summary section, item v3 had 587 missing values. The reported classical analyses reflect this (3000- 587 = 2413) appropriately.

3.7. Graded Model Calibration and Scoring

Example syntax for a combined calibration and scoring run is now presented. flexMIRT^{TM} code for the graded response model, which may be used with two or more ordered response options (e.g., Samejima, 1969), is also introduced.

Example 3-7: Graded Model Combined Calibration and Scoring Syntax with Recoding

```
<Project>
Title = "NCS example";
Description = "18 items, 1 Factor, 1 Group Calibration
```

```
 and Scoring";
<Options>
Mode = Calibration;
saveSCO = Yes;
Score = MAP;
GOF = Extended;

<Groups>
%Group1%
File ="NCSsim_MISS.dat";
Varnames =v1-v18;
N = 3000;
Ncats(v1,v2,v4-v18) = 5;
Ncats(v3) = 4;
Code(v1,v2,v4-v18) = (1,2,3,4,5), (0,1,2,3,4);
Code(v3) = (1,2,4,5), (0,1,2,3);
Model(v1,v2,v4-v18) = Graded(5);
Model(v3) = Graded(4);

<Constraints>
```

The standard syntax necessary for a calibration is supplied. Of note in this command file is that even though this is a calibration run, we have also requested `MAP` scores be saved. This combination of commands is a short-cut to a combined calibration and scoring run, in which the calibration is explicitly requested and the `SaveSCO=Yes;` command implicitly requests a scoring run, using the estimated parameter values from the calibration.

From examining the data, we know that the response categories in the data file start at 1 and go up and based on the previous classical output, we know that item v3 does not have any observed responses of 3. If submitted to flexMIRT with non-zero based and non-sequential categories (e.g., 1,2,4,5) an error will be returned. We have included two `Code` statements in our syntax, one for the 17 items needing only to be recoded to zero-based values and a separate recoding statement for v3, to address the missing response category (as well as obtain zero-based values).

flexMIRT$^{\text{TM}}$ makes no assumptions about the regularity of item intercepts, so it is not possible to estimate, say the boundary between response category 2 and category 3, when no observations used the available “3” response. Because of this, the fact that no raw 3/zero-based coding 2s were seen in item v3 needs to be addressed before submitting the calibration analysis. The most typical technique for managing such response anomolies is to collapse/recode the data- we have done so in our second `Code` statement, `Code(v3) = (1,2,4,5), (0,1,2,3);`. We find that the original response value of 3 has been omitted from our `Code` statement, because there are no observed 3s in the data set. Based on the values in the second half of the `Code` statement, we ensure that flexMIRT^{TM} will be provided with sequential, observed response categories; specifically, all 1s will be recoded to 0s, all 2s will be recoded to 1s, all 4s will be recoded to 2s and all 5s will be recoded to 3s. Because of this missing response category, item v3 also needs a separate `Model` statement from the other items as v3 only has 4 response categories, not the 5 response categories with observed responses seen in all other items.

With respect to the output, the primary changes from previous examples are the additional parameter values in the item parameter table. As may be seen from the two item parameter estimate output tables below there are multiple intercepts (or equivalently, thresholds) for each item. For convenience, both of these parameterizations (intercepts and thresholds) are presented in separate tables. However, only the intercept values will be saved to the -prm file, if it is requested. This behavior can be modified by adding the statement `SlopeThreshold= Yes;` to the `<Options>` section. When the `SlopeThreshold` keyword is invoked, threshold values, rather than intercept values, will be saved to the -prm file.

Output 3.13: Graded Model Calibration - Slopes and Intercepts

```
Graded Items for Group 1: Group1
Item Label P#    a    s.e.   P# c 1   s.e.  P#   c 2  s.e.  P#   c 3  s.e.  P#   c 4  s.e.
  1  v1    5   1.58  0.06   1  3.25  0.09   2  0.94  0.06   3 -0.41  0.05   4 -3.34  0.09
  2  v2   10   1.85  0.06   6  4.36  0.12   7  1.80  0.07   8  0.48  0.06   9 -2.92  0.09
  3  v3   14   1.88  0.07  11  3.89  0.12  12  1.07  0.07  13 -2.27  0.08
  4  v4   19   1.72  0.06  15  4.10  0.12  16  1.64  0.06  17  0.35  0.06  18 -2.61  0.08
  5  v5   24   1.45  0.05  20  4.35  0.12  21  1.72  0.06  22  0.47  0.05  23 -2.35  0.07
 ...
 16 v16   79   0.79  0.04  75  2.13  0.06  76  0.20  0.04  77 -0.59  0.04  78 -2.69  0.07
 17 v17   84   1.17  0.05  80  3.28  0.09  81  0.95  0.05  82  0.00  0.05  83 -2.07  0.06
 18 v18   89   0.61  0.04  85  2.83  0.08  86  1.16  0.05  87  0.21  0.04  88 -2.13  0.06
```

Output 3.14: Graded Model Calibration - Slopes and Thresholds

```
Graded Items for Group 1: Group1
Item Label P#  a    s.e.   b 1   s.e.   b 2   s.e.  b 3  s.e.  b 4   s.e.
 1    v1    5  1.58  0.06  -2.05  0.07 -0.60  0.04  0.26  0.03  2.11  0.07
 2    v2   10  1.85  0.06  -2.35  0.08 -0.97  0.04 -0.26  0.03  1.57  0.05
 3    v3   14  1.88  0.07  -2.06  0.07 -0.57  0.04  1.21  0.05
 4    v4   19  1.72  0.06  -2.38  0.08 -0.95  0.04 -0.21  0.03  1.51  0.05
 5    v5   24  1.45  0.05  -3.01  0.12 -1.19  0.05 -0.33  0.04  1.63  0.06
 ...
16   v16   79  0.79  0.04  -2.68  0.14 -0.25  0.05  0.75  0.06  3.39  0.18
17   v17   84  1.17  0.05  -2.80  0.11 -0.81  0.05 -0.00  0.04  1.77  0.07
18   v18   89  0.61  0.04  -4.62  0.31 -1.88  0.13 -0.35  0.07  3.47  0.23
```

3.8. 1PL Model Fitted to Response Pattern Data

Up to this point, analyses have been conducted on raw (individual) data, in which each line in the data set represents responses from an individual. flexMIRT^{TM} is also able to handle data that are grouped by response patterns. This example uses the well-known LSAT6 data set, in which the individuals have been grouped by response patterns. There are 1000 examinees and 5 dichotomous items. The first use of constraints in the `<Constraints>` section is also presented, demonstrating how to fit a 1PL model in flexMIRT^{TM}.

Example 3-8: 1PL Calibration and Scoring with Response Pattern Data

```
<Project>
Title = "LSAT 6";
Description= "5 Items 1PL N=1000 Grouped Data";

<Options>
Mode = Calibration;
GOF = Complete;
JSI = Yes;
HabermanResTbl = Yes;
SX2Tbl  = Yes;
FactorLoadings  = Yes;
SaveSCO = No;
Score =SSC;

<Groups>
```

```
16
17   %Group1%
18   File ="lsat6grp.dat";
19   Varnames = v1-v5, w;
20   Select= v1-v5; // only the first 5 variables
21   CaseWeight= w;
22   N=30;
23   /*  <- Starts a Block Comment
24   For grouped data, N = is rows in data set, not total N
25   Ends a Block Comment -> */
26   Ncats(v1-v5) = 2;
27   Model(v1-v5) = Graded(2);
28
29   <Constraints>
30   Equal(v1-v5), Slope;
```

An examination of the data file (lsat6grp.dat) shows that the data for these five items have been grouped by response patterns with the last variable in the file providing the number of observations per pattern. After naming the group and file name in the `<Groups>` section as usual, we use the `Select` statement to select only the first 5 variables that will be subjected to IRT analysis (i.e., we exclude the weighting variable, which was named "w" in the `Varnames` statement). Note that this is also the first time that we have encountered comments in the syntax file. A comment begins with 2 slashes, following the standard C/C++ convention. They may appear anywhere in the syntax, but not within a command statement. In addition, one may also use a pair of `/* */` to create entire blocks of comments that span more than one line, as shown around Line 23 syntax file.

Returning to the example syntax, we use the `CaseWeight` command to identify the variable that contains response pattern frequencies, (i.e., the number of cases per response pattern). The rest of the `<Groups>` statements, those defining number of categories, etc., are as before.

Note that the graded model with 2 categories is specified in the `Model` statement, which is equivalent to a 2PL model. To obtain a 1PL model, the slope/discrimination parameters must be constrained equal across items. To accomplish this, an equality constraint is specified in the `<Constraints>` section using the `Equal` statement on the `Slope` parameters of items `v1-v5`. The keywords for other item parameters are `Intercept` for the location parameter in all models, `Guessing` for the guessing parameter/lower asymptote in the 3PL model, and `ScoringFn` for the scoring function parameters in the Nominal Categories model. See the Constraints section of the Details of the Syntax chapter for additional information on flexMIRT^{TM} constraints.

There are several new commands in the `<Options>` section. For the goodness-of-fit indices, the full gamut of available `GOF` indices (detailed in Chapter 9 of the manual) are requested by the keyword `Complete`. The `JSI` statement requests the Jackknife Slope Index (JSI), an LD detection tool, for all pairs of items. The `HabermanResTbl` command requests a table that lists the observed and expected frequencies and standardized residuals for each response pattern, as well as the associated SEs, EAPs and SDs. When `SX2Tbl` is requested, detailed item fit tables will be printed, in addition to the Orlando and Thissen (2000) $S - X^2$ item fit chi-square statistics. The command `FactorLoadings = Yes;` requests the printing of the factor loading values for each item, which may be converted from the respective slope values (e.g., Takane & de Leeuw, 1987; Wirth & Edwards, 2007). Even though the scores are not saved (`SaveSCO = No;`), the scoring method of `SSC` is specified. As noted previously, the `SSC` method will print the requested summed score to EAP scale score conversion table in the output, although no individual scores are calculated/saved. The commands introduced in this example add several new parts to the output file, discussed in the next pages.

The item parameters are printed in their customary order, but note that the 1PL equal-slope constraint specified in the syntax has been interpreted correctly, as evidenced by all values in the *a* column being equal. The factor loadings are presented below the typical item parameter table, with loadings appearing in the `lambda1` column.

Output 3.15: 1PL Grouped Data Output - Item Parameters and Factor Loadings

```
2PL Items for Group 1: Group1
 Item Label   P#      a     s.e.   P#      c   s.e.       b     s.e.
    1    v1    6   0.76    0.07    1   2.73   0.13   -3.61    0.32
    2    v2    6   0.76    0.07    2   1.00   0.08   -1.32    0.14
    3    v3    6   0.76    0.07    3   0.24   0.07   -0.32    0.10
    4    v4    6   0.76    0.07    4   1.31   0.08   -1.73    0.17
    5    v5    6   0.76    0.07    5   2.10   0.11   -2.78    0.25
```

```
LSAT 6
5 Items 1PL N=1000 Grouped Data

Factor Loadings for Group 1: Group1
 Item      Label lambda 1  s.e.
    1         v1     0.41  0.05
    2         v2     0.41  0.05
    3         v3     0.41  0.05
    4         v4     0.41  0.05
    5         v5     0.41  0.05
```

The next new portion of the output are the values and tables associated with the summed-score based item fit diagnostic (e.g., Orlando & Thissen, 2000, Cai, 2015), which are printed as a result of the `GOF = Complete` and `SX2Tbl = Yes;` commands, respectively. The $S - X^2$ value and table for the first item is presented. When the p-value associated with the $S - X^2$ value is less than the α-level (typically 0.05), the item should be examined for possible misfit.

Output 3.16: 1PL Grouped Data Output - Item Fit Diagnostics

```
Orlando-Thissen-Bjorner Summed-Score Based Item Diagnostic Tables and X2s
 Group   1: Group1
  Item 1 S-X2(3) = 0.2, p = 0.9809
             Category  0           Category  1
 Score     Observed    Expected    Observed    Expected
     0            3         2.4          10        10.6
     1           10         9.9          62        62.1
     2           23        23.3         212       211.7
     3           25        25.5         342       341.5
     4           15        14.9         298       298.1
```

By specifying `Score = SSC;` in the command file, we requested a summed score to EAP scale score conversion table be printed, which is shown next. With a total of 5 items, there are 6 possible summed scores which are listed in the first column. Going across a row, after the sum score, the EAP score and the posterior standard deviation associated with that particular summed score value are presented. The table also includes model-implied and observed probabilities for the summed scores, appearing in columns labeled `P` and `O`, respectively. Immediately after the table, the marginal reliability of the scale scores is printed, as well as the value of a chi-square statistic that provides a test of the underlying normality of the latent variable distribution.

Output 3.17: 1PL Grouped Data Output - Summed Score to Scale Score Conversion

```
Summed Score to Scale Score Conversion Table:
  Summed
   Score     EAP      SD         P         O
    0.00  -1.910   0.797 0.0023654 0.0030000
    1.00  -1.429   0.800 0.0208898 0.0200000
    2.00  -0.941   0.809 0.0885974 0.0850000
    3.00  -0.439   0.823 0.2274460 0.2370000
    4.00   0.084   0.841 0.3648997 0.3570000
    5.00   0.632   0.864 0.2958018 0.2980000
Summed score based latent distribution fit S-D2 = 0.9, p = 0.8166
Marginal reliability of the scaled scores for summed scores = 0.29427
```

Scrolling further down in the full output file, we encounter sections related to LD detection. We have previously discussed the Marginal fit and Standardized LD X^2 Statistics table, but the matrix immediately below that in the output is new. The `JSI = Yes;` statement in the command file produces a matrix (labeled Edwards-Houts-Cai Jackknife Slope Diagnostic Index) which provides values for a newly introduced LD detection method (Edwards & Cai, 2011), the calculation of which is described in detail later in the manual.

Output 3.18: 1PL Grouped Data Output - JSI matrix

```
Edwards-Houts-Cai Jackknife Slope Diagnostic Index for Group 1: Group1
  Item      1       2       3       4       5
     1    ---   -0.08    0.37   -0.17   -0.15
     2   0.07     ---    0.37   -0.17   -0.15
     3   0.07   -0.08     ---   -0.17   -0.15
     4   0.07   -0.08    0.37     ---   -0.15
     5   0.07   -0.08    0.37   -0.17     ---
```

For now, it is sufficient to know that a value noticeably larger than others in the matrix indicates an item pair that should be examined for possible LD.

The last new table is the result of the statement `HabermanResTbl=Yes;`. A portion of this table is reproduced below. In this table, observed response patterns are listed by row. For a given response pattern, the observed and model-implied frequencies, the standardized residual, EAP score, and standard deviation associated with the EAP score are printed.

Output 3.19: 1PL Grouped Data Output - Haberman Residual Table

```
Response Pattern Observed and Expected Frequencies, Standardized
Residuals, EAPs and SDs

Group 1: Group1
Item:              Frequencies    Standard
  1  2  3  4  5 Observed Expected Residual EAP[q |u]  SD[q |u]
  0  0  0  0  0     3.00     2.37     0.41     -1.91      0.80
  0  0  0  0  1     6.00     5.47     0.23     -1.43      0.80
  0  0  0  1  0     2.00     2.47    -0.30     -1.43      0.80
  0  0  0  1  1    11.00     8.25     0.96     -0.94      0.81
  0  0  1  0  0     1.00     0.85     0.16     -1.43      0.80
  0  0  1  0  1     1.00     2.84    -1.09     -0.94      0.81
  0  0  1  1  0     3.00     1.28     1.51     -0.94      0.81
  0  0  1  1  1     4.00     6.22    -0.89     -0.44      0.82
  0  1  0  0  0     1.00     1.82    -0.61     -1.43      0.80
  0  1  0  0  1     8.00     6.06     0.79     -0.94      0.81
  0  1  0  1  1    16.00    13.29     0.75     -0.44      0.82
...
  1  0  1  0  0     3.00     5.33    -1.01     -0.94      0.81
  1  0  1  0  1    28.00    25.83     0.43     -0.44      0.82
  1  0  1  1  0    15.00    11.69     0.97     -0.44      0.82
  1  0  1  1  1    80.00    83.30    -0.38      0.08      0.84
  1  1  0  0  0    16.00    11.39     1.37     -0.94      0.81
  1  1  0  0  1    56.00    55.17     0.12     -0.44      0.82
  1  1  0  1  0    21.00    24.96    -0.80     -0.44      0.82
  1  1  0  1  1   173.00   177.91    -0.41      0.08      0.84
  1  1  1  0  0    11.00     8.59     0.83     -0.44      0.82
  1  1  1  0  1    61.00    61.23    -0.03      0.08      0.84
  1  1  1  1  0    28.00    27.71     0.06      0.08      0.84
  1  1  1  1  1   298.00   295.80     0.15      0.63      0.86
```

3.9. The Nominal Categories Model

Our last set of basic examples will give demonstrations of using flexMIRT$^{\text{TM}}$ to fit the reparameterized Nominal Categories model (Thissen, Cai, & Bock, 2010, Thissen & Cai, 2016). As discussed in both Thissen and Steinberg (1986) and Thissen et al. (2010), models equivalent to the Partial Credit Model (Masters, 1982), and Muraki's (1992) Generalized Partial Credit Model, among others, may be obtained as restricted versions of the full-rank nominal model by the choice of contrast matrices and constraints.

3.9.1 Nominal Model

The first data set contains responses to items in a quality of life scale. First reported in Lehman (1988), the data set consists of responses from 586 patients to 35 items from the classic Quality of Life (QOL) Scale for the Chronically Mentally Ill. All items have 7 response categories: terrible; unhappy; mostly dissatisfied; mixed, about equally satisfied and dissatisfied; mostly satisfied; pleased; and delighted.

Example 3-9: Nominal Model Calibration

```
<Project>
Title = "QOL Data";
Description = "35 Items - Nominal";

<Options>
Mode = Calibration;
Etol = 1e-4;
Mtol = 1e-5;
Processors = 2;

<Groups>
%Group1%
File ="QOL.DAT";
Varnames = v1-v35;
N = 586;
Ncats(v1-v35) = 7;
Model(v1-v35) = Nominal(7);

<Constraints>
Fix(v1-v35), ScoringFn(1);
```

In the `<Groups>` section, we have identified the data file, named the variables, specified the number of response options, and chosen to fit the `Nominal` model with 7 categories to the data. In the `<Options>` section, the convergence criteria for both the E step and M iterations of the EM algorithm have been adjusted from the default values, by the `Etol` and `Mtol` commands, respectively, so that a tightly converged solution can be obtained.

Because the Nominal Categories model estimates parameters that differ from the models previously discussed, excerpts from the item parameter tables will be presented from the output and briefly discussed.

Output 3.20: Nominal Model Output - Item Parameters

```
Nominal Model Slopes and Scoring Function Contrasts for Items for Group 1: Group1
    Item             Label  P#       a    s.e. Contrasts  P# alpha 1    s.e.  P# alpha 2    s.e.  P# alpha 3    s.e.  P# alpha 4    s.e.  P# alpha 5    s.e.  P# alpha 6    s.e.
       1                v1   6    0.83    0.17     Trend        1.00    ----   1   -1.21    0.89   2    0.14    0.48   3    0.29    0.37   4    0.10    0.39   5   -0.19    0.37
       2                v2  18    0.44    0.17     Trend        1.00    ----  13   -1.01    1.71  14    0.06    0.95  15   -0.11    0.70  16    0.37    0.74  17   -0.12    0.61
...
      34               v34 402    0.49    0.13     Trend        1.00    ---- 397   -0.98    1.40 398    0.44    0.79 399    0.07    0.82 400    0.24    0.75 401   -0.13    0.56
      35               v35 414    0.34    0.14     Trend        1.00    ---- 409   -0.97    2.17 410    0.22    1.17 411    0.14    1.11 412    0.20    0.93 413   -0.20    0.82

Nominal Model Scoring Function Values Group 1: Group1
    Item             Label     s 1     s 2     s 3     s 4     s 5     s 6     s 7
       1                v1    0.00    0.79    1.15    1.31    3.09    4.38    6.00
...
      34               v34    0.00    1.10    1.44    1.82    3.10    3.92    6.00
      35               v35    0.00    0.92    1.35    1.69    3.32    4.18    6.00

Nominal Model Intercept Contrasts for Items for Group 1: Group1
    Item             Label Contrasts  P# gamma 1    s.e.  P# gamma 2    s.e.  P# gamma 3    s.e.  P# gamma 4    s.e.  P# gamma 5    s.e.  P# gamma 6    s.e.
       1                v1     Trend   7    0.15    0.15   8    1.64    0.50   9   -0.54    0.28  10    0.27    0.21  11    0.14    0.20  12    0.09    0.17
       2                v2     Trend  19    0.34    0.14  20    0.64    0.52  21   -0.42    0.30  22    0.05    0.22  23    0.23    0.22  24    0.26    0.20
...
      34               v34     Trend 403    0.26    0.12 404    1.53    0.46 405   -0.70    0.30 406    0.16    0.29 407   -0.02    0.25 408    0.04    0.19
      35               v35     Trend 415    0.43    0.15 416    1.12    0.59 417   -0.77    0.37 418    0.15    0.27 419    0.15    0.29 420    0.11    0.27

Original (Bock, 1972) Parameters, Nominal Items for Group 1: Group1
    Item             Label   Category:       1       2       3       4       5       6       7
       1                v1           a    0.00    0.66    0.95    1.09    2.55    3.62    4.96
                                     c    0.00    0.94    1.06    1.92    2.53    2.24    0.91
       2                v2           a    0.00    0.31    0.43    0.87    1.54    1.74    2.65
                                     c    0.00    0.67    0.44    1.86    2.24    2.36    2.02
...
      34               v34           a    0.00    0.54    0.70    0.89    1.52    1.92    2.93
                                     c    0.00    0.59    1.22    2.19    2.92    2.87    1.57
      35               v35           a    0.00    0.31    0.46    0.57    1.13    1.42    2.04
                                     c    0.00    0.66    0.93    2.37    3.39    3.45    2.58
```

The first 3 tables in Output 2.16 provide parameter estimates using the reparameterized version of the model (see Thissen et al., 2010). In the first table, the overall slope value, labeled `a`, the type of $\boldsymbol{\alpha}$ contrast matrix (`Ta`) used, and the estimated contrast values, labeled `alpha 1` – `alpha 3`, are reported. There are 7 categories so there are 6 scoring function contrasts. The second table, labeled `Nominal Model Scoring Function Values` reports the scoring function values for the categories, which are calculated as $\boldsymbol{T_a \alpha}$, where $\boldsymbol{\alpha}$ are the estimated scoring contrasts. The third table of item parameters reports the type of contrast matrix used for the intercepts (`Tc`) and the estimated $\boldsymbol{\gamma}$ parameters. For convenience, the final item parameter table of the output presented also reports the estimates in Bock's original parameterization.

The next example employs a set of data regarding numerical knowledge from pre-school aged children, most notably analyzed in Thissen and Steinberg (1988) to illustrate the concept of testlets. As described by Thissen and Steinberg (1988), the data were collected from 4 tasks that involved asking 592 young children to identify the numerals representing the quantities of 3 and 4 and, separately, match the numeral for each number to a correct quantity of blocks. From these items, called Identify3, Identify4, Match3, and Match4, pseudo-items were created that summarized a child's performance on both numbers. For the new pseudo-items, Identify and Match, the response options are 0 (child could do the requested action for neither number), 3 (child could identify/match 3 only), 4 (child could identify/match 4 only), and 34 (child could identify/match both 3 and 4 successfully). We will again fit the Nominal Model, but with some additional constraints imposed.

Example 3-10: Nominal Model Calibration 2

```
<Project>
Title =  "Pre-School Numerical Knowledge";
Description= "2 Redefined Items Nominal";

<Options>
Mode = Calibration;
SE = SEM;
Etol  = 5e-5;
Mtol  = 1e-9;
GOF = Extended;
M2  =  Full;
HabermanResTbl = Yes;
```

```
SX2Tbl  = Yes;
FitNullModel  = Yes;
SavePRM = Yes;

<Groups>
%Group1%
File ="PreschoolNum.dat";
Varnames =  Identify,Match,Identify3,Identify4,Match3,Match4;
Select= Identify, Match;
N = 592;
Ncats(Identify,Match) = 4;
Code1(Identify,Match) = (0,3,4,34),(0,1,2,3);
Model(Identify,Match) =  Nominal(4);
Ta(Identify,Match) =  Identity;
Tc(Identify,Match) =  Identity;

<Constraints>
Fix(Identify), ScoringFn(2);
Equal Group1, (Match), ScoringFn(2):  Group1, (Match), ScoringFn(3);
Equal Group1, (Match), Intercept(1): Group1, (Match), Intercept(2);

```

Within the `<Options>` section, there are several new commands. The method of standard error calculation has been changed from the default of empirical cross-product approximation (`Xpd`) to Supplemented EM algorithm (keyword is `SEM`; see Cai, 2008). We have also requested the full version of the limited information GOF M_2 statistic (Cai & Hansen, 2013; Maydeu-Olivares & Joe, 2005) and, additionally, specified that the zero-factor Null Model should be estimated and the associated incremental GOF indices be printed.

In the `<Groups>` section, we have identified the data file, named the variables, selected only the pseudo-items to retain for analysis, specified the number of response options, recoded the response options into zero-based sequential (but nevertheless nominal) codes, and chosen to fit the `Nominal` model with 4 categories to the data. The statements `Ta` and `Tc` are specific to the nominal model and refer to the contrast matrices for the scoring function and intercept parameters, respectively. The options for the scoring function and intercept contrasts include a Trend matrix (which uses a Fourier basis, augmented with a linear column; see Thissen et al., 2010), an Identity matrix, or a user-supplied matrix. In this example, both of the contrast matrices have been set to type `Identity` to facilitate equality constraints.

Within the `<Constraints>` section, we encounter a more complicated set of constraints than in the previous example. Substantively, these constraints are applied to the scoring functions and intercepts so that the solution matches with Thissen and Steinberg's (1988) preferred testlet solution. In terms of flexMIRT^{TM} coding, we do encounter something new in the `Equal` statements.

In both `Equal` statements, rather than setting a group of similar parameters across items (such as the slopes across items in the previous 1PL example) equal to another, we are selecting specific parameters to impose the equality constraint upon. Because of this, the form of the `Equal` statement is slightly more complex; because we are specifying different parameters we will need to use a multi-part statement where each part will exactly specify parameters. First, the type of constraint to be applied (`Equal` in this case) is listed and then syntax to identify the first parameter(s) to which the constraint will be applied is provided. We do this by listing the group name, followed by a comma, and then the item(s) which are to be affected by the constraint are then specified in parentheses, followed by another comma, and then the keyword for the specific parameter that is affected (slope, intercept, etc.), followed by a numeric indicator in paranetheses for the exact parameter. In the current example, the first `Equal` statement is specifying the second scoring function of item "Match" in "Group1". The colon indicates the end of the first part of the constraint and we will then specifiy the second parameter to be included in our equality (in our example, the third scoring function of item "Match" in "Group1"), using similar statements and ordering, after the colon. This formatting of constraints may have as many parts as is necessary (that is, it is not limited to only two parameters that can be set equal to one another), and can also be used to impose cross-group constraints (as will be demonstrated in later multiple-group examples.

Estimated item parameters from the current model are presented in Output 3.21.

Output 3.21: Nominal Model 2 Output - Item Parameters

```
Nominal Model Slopes and Scoring Function Contrasts for Items for Group 1: Group1
   Item     Label  P#       a    s.e. Contrasts  P# alpha 1    s.e.  P# alpha 2   s.e. P#  alpha 3   s.e.
      1  Identify   2    0.80    0.61  Identity        1.00    ----       0.00    ---- 1    1.21    0.47
      2     Match   6    0.82    0.62  Identity        1.00    ---- 8   0.46    0.31 8    0.46    0.31

Nominal Model Scoring Function Values Group 1: Group1
   Item     Label     s 1     s 2     s 3     s 4
      1  Identify    0.00    0.00    1.21    3.00
      2     Match    0.00    0.46    0.46    3.00

Nominal Model Intercept Contrasts for Items for Group 1: Group1
   Item     Label Contrasts  P# gamma 1    s.e.  P# gamma 2    s.e.  P# gamma 3    s.e.
      1  Identify  Identity   3   -0.86    0.14   4   -0.80    0.43  5   0.65    0.19
      2     Match  Identity   9   -0.42    0.28   9   -0.42    0.28  7   0.52    0.20

Original (Bock, 1972) Parameters, Nominal Items for Group 1: Group1
   Item     Label   Category:       1       2       3       4
      1  Identify           a    0.00    0.00    0.97    2.41
                            c    0.00   -0.86   -0.80    0.65
      2     Match           a    0.00    0.38    0.38    2.45
                            c    0.00   -0.42   -0.42    0.52
```

Although several new command statements were introduced in this example, only two result in additions to the output file. Both the `M2` and `FitNullModel` statements provide additional GOF information that is printed near the bottom of the output file, with the other overall GOF index values. The fitted model's $-2\times$log-likelihood, AIC, and BIC are printed as before, but additional fit statistics for the zero-factor null model are also included under the corresponding fitted model section. The estimation of the independence model permits the calculation of the non-normed fit index (NNFI; aka Tucker-Lewis Index). The usual full-information fit statistics are printed, but limited-information statistics, based on M_2, for the fitted and zero-factor model are also given. Taken together, the fit index values indicate that the nominal model provides excellent fit to this data.

Output 3.22: Nominal Model 2 Output - GOF values

```
Statistics based on the loglikelihood of the fitted model
                       -2loglikelihood:     2766.69
    Akaike Information Criterion (AIC):     2784.69
  Bayesian Information Criterion (BIC):     2824.14

Statistics based on the loglikelihood of the zero-factor null model
                       -2loglikelihood:     2885.67
    Akaike Information Criterion (AIC):     2895.67
  Bayesian Information Criterion (BIC):     2917.58

Full-information fit statistics of the fitted model
               Degrees
        G2  of freedom Probability       F0hat       RMSEA
      8.45           6      0.2064      0.0143        0.03
               Degrees
        X2  of freedom Probability       F0hat       RMSEA
      8.99           6      0.1739      0.0152        0.03

Full-information fit statistics of the zero-factor null model
               Degrees
        G2  of freedom Probability       F0hat       RMSEA
    127.43          10      0.0001      0.2153        0.14
               Degrees
        X2  of freedom Probability       F0hat       RMSEA
    124.58          10      0.0001      0.2104        0.14
  With the zero-factor null model, Tucker-Lewis (non-normed) fit index
  based on G2 is     0.97

  With the zero-factor null model, Tucker-Lewis (non-normed) fit index
  based on X2 is     0.96

Limited-information fit statistics of the fitted model:
                 Degrees
       M2  of freedom Probability       F0hat       RMSEA
     8.24           6      0.2207      0.0139        0.03
Note: M2 is based on full marginal tables.
Note: Model-based weight matrix is used.

Limited-information fit statistics of the zero-factor null model:
                 Degrees
       M2  of freedom Probability       F0hat       RMSEA
   124.58          10      0.0001      0.2104        0.14
Note: M2 is based on full marginal tables.
Note: Model-based weight matrix is used.

  With the zero-factor null model, Tucker-Lewis (non-normed) fit index
  based on M2 is     0.97
```

3.9.2 Generalized Partial Credit Model

We return to the QOL data again to illustrate Muraki's (1992) Generalized Partial Credit (GPC) model. The GPC model is a constrained special case of the nominal model. To set up a GPC model using the reparameterized nominal model, one fixes all scoring function contrasts. This results in the first scoring function contrast being fixed to 1.0 and the other contrasts fixed to 0.0. With such constraints in place, the category scoring function values are equal to $0, 1, \ldots, k$, which is the necessary structure for fitting the GPC. Because the GPC is such a popular special case of the nominal model we have added `GPC()` as an option to the `Model()= ;` statement. This option will automatically fix the scoring functions as needed, freeing the user from making such technical specifications. The following syntax demonstrates the use of this model keyword. As with the `Graded` model specification, when one uses the `GPC()` keyword in the `Model` statement, the model name is immediately followed by the number of categories in parentheses.

Example 3-11: Generalized Partial Credit Model

```
<Project>
Title = "QOL Data";
Description= "35 Items - GPC";

<Options>
Mode = Calibration;
Etol = 1e-4;
Mtol = 1e-5;
Processors = 2;

<Groups>
%Group1%
File = "QOL.DAT";
Varnames = v1-v35;
N=586;
Ncats(v1-v35) = 7;
Model(v1-v35) = GPC(7);

<Constraints>
```

3.9.3 Rating Scale Model

Andrich's (1978) Rating Scale (RS) model may also be obtained as a special case of the reparameterized nominal model. To obtain the RS model in flexMIRT^{TM} it is necessary to fix the scoring functions for all items. Additionally, we need to constrain the slopes for all items to be equal, as well as set each intercept to be equal across the items. This is all accomplished via the constraints imposed in Example 3-12.

Example 3-12: Rating Scale Model with Trend Contrasts

```
<Project>
Title = "QOL Data";
Description= "35 Items RS Using Default Trend Contrasts";

```

```
<Options>
Mode = Calibration;
SE = SEM;

<Groups>
%Group1%
File = "QOL.DAT";
Varnames = v1-v35;
N = 586;
Ncats(v1-v35) = 7;
Model(v1-v35) = Nominal(7);

<Constraints>
Fix(v1-v35), ScoringFn;
Equal(v1-v35),Slope;
Equal(v1-v35),Intercept(2);
Equal(v1-v35),Intercept(3);
Equal(v1-v35),Intercept(4);
Equal(v1-v35),Intercept(5);
Equal(v1-v35),Intercept(6);
```

The just-presented RS fitting used the flexMIRT^{TM} default trend contrasts. To obtain results for the RS model similar to those that would be produced by Multilog, the user may provide flexMIRT^{TM} with the triangle contrasts utilized by Multilog. This is done through the `Tc` command and, rather than using one of the available keywords, supplying the values that compose the desired contrast. The triangle contrast is shown in the `Tc` statement in the following example syntax.

Example 3-13: Rating Scale Model with Triangle Contrasts

```
<Project>
Title = "QOL Data";
Description= "35 Items RS Multi-log style Triangle Contrasts";

<Options>
Mode = Calibration;
```

```
SE = SEM;

<Groups>
%Group1%
File = "QOL.DAT";
Varnames = v1-v35;
N = 586;
Ncats(v1-v35) = 7;
Model(v1-v35) = Nominal(7);
Tc(v1-v35) = (
      0  0  0  0  0  0,
     -1 -1  0  0  0  0,
     -2 -1 -1  0  0  0,
     -3 -1 -1 -1  0  0,
     -4 -1 -1 -1 -1  0,
     -5 -1 -1 -1 -1 -1,
     -6  0  0  0  0  0 );

<Constraints>
Fix(v1-v35),ScoringFn;
Equal(v1-v35),Slope;
Equal(v1-v35),Intercept(2);
Equal(v1-v35),Intercept(3);
Equal(v1-v35),Intercept(4);
Equal(v1-v35),Intercept(5);
Equal(v1-v35),Intercept(6);
```

Thus far, we have covered how to use flexMIRT^{TM} to calibrate several of the most commonly used single-group unidimensional IRT models, and to obtain IRT scale scores. While these types of models are useful, the greatest strength of flexMIRT^{TM} is its ability to readily handle complex models (e.g., multidimensional, bifactor, testlet, etc.) and more complex data structures, such as multilevel data. In the next chapter, we will provide instruction and examples, building intuitively from the basic syntax already presented, to demonstrate how flexMIRT^{TM} syntax files are constructed to handle more complex models and data structures.

CHAPTER 4

Advanced Modeling

The previous chapter focused on basic unidimensional, single group models to introduce the flexMIRT$^{\text{TM}}$ syntax, output, and available IRT models. While these examples are useful in their own right, it is expected that a user working with real-world data will need more advanced modeling options. In this chapter, we will provide examples for using flexMIRT$^{\text{TM}}$ when multiple groups, non-normal theta distributions, or hierarchical data structures need to be modeled.

4.1. Multiple Group Calibration

The data set used to illustrate 2PL and 3PL examples in the previous chapter is actually made up of responses from two groups of examinees, 3rd and 4th graders. Using this already familiar data set, we will demonstrate a multiple group analysis.

Example 4-1: 2-Group 3PL Using Normal Group Parameters and Guessing Prior with EAP Scoring

```
<Project>
Title =  "G341-19";
Description= "12 Items 3PL 2 Groups
Calibration with Normal Prior, Saving Parameter Estimates,
Followed by Combined Scoring Run for EAP Scores";

<Options>
Mode = Calibration;
Progress = Yes;
SaveSCO = Yes;
Score = EAP;
```

```
SavePRM = Yes;
SaveINF = Yes;
FisherInf = 81,4.0;
SE = Xpd;
GOF = Extended;
M2 = Full;
FitNullModel = Yes;

<Groups>
%Grade4%
File ="g341-19.grp1.dat";
Varnames = v1-v12;
N=1314;
Ncats(v1-v12) = 2;
Model(v1-v12) = ThreePL;
BetaPriors(v1-v12) = 1.5;

%Grade3%
File ="g341-19.grp2.dat";
Varnames = v1-v12;
N=1530;
Ncats(v1-v12) = 2;
Model(v1-v12) = ThreePL;
BetaPriors(v1-v12) = 1.5;

<Constraints>
Free Grade3, Mean(1);
Free Grade3, Cov(1,1);
Equal Grade3, (v1-v12), Guessing:
      Grade4, (v1-v12), Guessing;
Equal Grade3, (v1-v12), Intercept:
      Grade4, (v1-v12), Intercept;
Equal Grade3, (v1-v12), Slope:
      Grade4, (v1-v12), Slope;
Prior Grade3, (v1-v12), Guessing : Beta(1.0,4.0);
```

Two groups (`%Grade4%` and `%Grade3%`) are specified in the `<Groups>` section. An important point to highlight is that the data for each group should be stored as individual files. To that end, "g341-19.dat" is split into 2 new files "g341-19.grp1.dat" and "g341-19.grp2.dat". As may be seen in the `N =` statements for each group, the original sample size of 2844 is now divided between the two groups. Within each group the basic specification statements (`VarNames`, `Ncats`, etc.) remain the same. There are simply now two sets of them, one per group. A new option not specific to multiple group analysis, is `BetaPriors`. This statement imposes a prior on the item uniquenesses in the form of a Beta(α,1.0) distribution with a user-specified α parameter, in this case $\alpha = 1.5$ (see Bock, Gibbons, & Muraki, 1988).

Within the `<Constraints>` section, we are allowing the mean and variance of the second group (Grade3) to be estimated by freeing those parameters via the first two constraint statements. Additionally, we have specified a model that assumes measurement invariance by constraining the item parameters to be equal across the two groups. Finally, we supplied a prior for the guessing parameters in the form of the beta distribution.

There are new commands encountered in the `<Options>` section as well. It should be noted that these newly encountered commands are not specific in any way to the multi-group format and are available for use with single group analyses as well. For instance, the `Progress` statement (via `Yes/No` keywords) determines whether flexMIRT$^{\text{TM}}$ will print progress updates in the console window. Using `SaveINF = Yes;`, we requested that the Fisher information function for the items and the total scale be saved to an external file and specified, via `FisherInf = 81,4.0;` command, that we would like information values calculated at 81 points, equally spaced from -4.0 to 4.0. The method of standard error calculation has been specified as empirical cross-products approximation, using `SE = Xpd` (all possible SE calculation methods and corresponding keywords are covered in the Details of the Syntax chapter).

4.2. Characterizing Non-Normal Population Distributions

This example is substantially the same as the analysis just discussed, with an additional option included in the `<Groups>` section. As the command file does not change drastically, only the modified sections are presented here. The full syntax file (g341-19.calib.EH.flexmirt) is available in the examples provided on the flexMIRT^{TM} website.

Example 4-2: 2-Group 3PL with Empirical Histogram (Excerpts)

```
<Options>
...
Etol   = 5e-4;

<Groups>
%Grade4%
File ="g341-19.grp1.dat";
Varnames = v1-v12;
N= 1314;
Ncats(v1-v12) = 2;
Model(v1-v12) = ThreePL;
BetaPriors(v1-v12) = 1.5;
EmpHist = Yes;
%Grade3%
File ="g341-19.grp2.dat";
Varnames = v1-v12;
N= 1530;
Ncats(v1-v12) = 2;
Model(v1-v12) = ThreePL;
BetaPriors(v1-v12) = 1.5;
EmpHist = Yes;
...
```

The primary point of interest here is the `EmpHist = Yes;` statement found as the last command provided for each group. With this option invoked, flexMIRT^{TM} will estimate the item parameters and the shape of the population (theta) distribution simultaneously. The latent variable distribution is estimated as an empirical histogram (e.g., Mislevy, 1984; Woods, 2007). By "empirical histogram" we mean the normalized accumulated posterior densities for all response patterns at each quadrature node - that is, an empirical characterization of the shape of the examinee ability distribution in the population. The use of an empirical histogram is in contrast to a typical calibration run in which the latent trait is assumed to be normally distributed. This estimation method is currently only available for single-level unidimensional and single-level bifactor (or testlet response models) with one general dimension. In the `<Options>` section, the convergence criterion for the E step of the EM algorithm has been increased from the default of 1e-4 to 5e-4 to accommodate the added uncertainty due to the estimation of the empirical histogram prior.

4.3. Multiple Group Scoring Only

Scoring runs with multiple groups proceed in much the same way as the single group scoring analyses. The current scoring example makes uses of the item parameters obtained from the first multiple group example (Example 3.1 which employed normal priors). The syntax is presented below and, as with the single group scoring syntax, in the `<Options>` section `Mode` is set to `Scoring`, the type of scores requested is specified (MAPs in this case) and the file containing the relevant item parameters (generated by our previous calibration run) is given using the `ReadPRMFile` statement. Because of the distinct groups, two sets of data descriptor statements (e.g., syntax for declaring group label, data file name, variable names, and sample size) are given. This example is very similar to the single group scoring runs, no further discussion is provided. For explications of any of the included syntax statements, the user should refer to a previous single-group example (e.g., Example 2.3).

Example 4-3: 2-Group 3PL MAP Scoring from Parameter Estimates

```
<Project>
Title =  "G341-19";
Description= "12 Items 3PL 2 Groups MAP
from Parameter Estimates";

<Options>
Mode = Scoring;
ReadPRMFile= "g341-19.calib.normal-prm.txt";
Score =MAP;
```

```

<Groups>
%Grade4%
File ="g341-19.grp1.dat";
Varnames = v1-v12;
N = 1314;
%Grade3%
File ="g341-19.grp2.dat";
Varnames = v1-v12;
N = 1530;
<Constraints>
```

In some situations, it may be useful to obtain the sum score conversion table, without actually having to score data. For example, when only the SSC table is wanted and the sample size is large enough that calculating scores for every individual would take a non-trivial amount of computing time, flexMIRT^{TM} is able to by-pass individual scoring and produce only the necessary table. The syntax to do so is straight-forward and presented below.

Example 4-4: 2-Group 3PL Summed Score to EAP Conversion Table

```
<Project>
Title =  "G341-19";
Description= "12 Items 3PL 2 Groups Summed Score to EAP from
Parameter Estimates Using Estimated Empirical Histogram Prior
Make Table Only";

<Options>
Mode = Scoring;
ReadPRMFile = "g341-19.calib.EH-prm.txt";
Score = SSC;
Quadrature  = (81,4.0);

<Groups>
%Grade4%
Varnames = v1-v12;
EmpHist = Yes;
```

```

%Grade3%
Varnames = v1-v12;
EmpHist = Yes;

<Constraints>
```

Just as in a typical scoring run, we have specified the parameter file (now using parameters obtained from the multiple-group calibration which estimated theta via an empirical histogram). Because we are using the empirical histogram parameter file, we must also supply a `Quadrature` command that matches the number and range of points used to originally estimate theta empirically. We have set the scoring method to produce the desired `SSC` (Summed Score to EAP Conversion) table. In the `<Groups>` section, we still provide labels and variable names for both groups as well as the `EmpHist = Yes;` statement to use the empirical histogram-estimated theta values in scoring, but the sample size and data file names are omitted because they are not needed for producing the conversion table without scoring any individuals.

4.4. DIF Analyses

Up to this point, the multiple group examples have assumed the invariance of item parameters across groups. However, when multiple groups are involved it is generally prudent to check that assumption by looking for differential item functioning (DIF) between groups. flexMIRT$^{\text{TM}}$ provides two general methods for DIF testing: 1) a DIF sweep for all items, and 2) a more focused examination, where candidate DIF items are tested with designated anchor items. Both methods utilize the improved Wald test (e.g., Langer, 2008). The DIF sweep procedure, which readers may be less familiar with, is detailed extensively in Woods, Cai, and Wang (2013). The DIF examples will again use the end-of-grade testing data. As the syntax is quite similar to what was presented earlier, only the changes will be noted in the excerpt. The full syntax file may be found in the corresponding folder on the flexMIRT$^{\text{TM}}$ support page.

Example 4-5: DIF: Test All Items (Excerpts)

```
...
<Options>
...
 DIF = TestAll;
 DIFcontrasts = (1.0 -1.0);
...
 <Constraints>
...
 Prior Grade3, (v1-v12), Guessing: Normal(-1.1,0.5);
 Prior Grade4, (v1-v12), Guessing: Normal(-1.1,0.5);
```

First, in conjunction with the use of the 3PL model, a logit-normal prior is applied in both groups to the guessing parameters, which was a previously unseen option. With respect to the DIF analysis, the relevant new statements are contained in the `<Options>` section. To enable the general DIF sweep, testing all items, `DIF = TestAll;` is specified. Also required when testing for DIF are values that supply flexMIRT^{TM} with the entries for a DIF contrast matrix, specifying how contrasts among groups should be constructed. In this example, we are conducting a straight-forward comparison of Grade3 against Grade4, so the statement appears as, `DIFcontrasts = (1.0 -1.0);` with spaces between the supplied values for each contrast. When k groups are present, $k-1$ contrasts must be specified, (e.g., for 3 groups, `DIFcontrasts = (2.0 -1.0 -1.0, 0.0 1.0 -1.0);`, with commas in between). Because the syntax file is free-form, the contrasts may appear on more than one line. For example,

```
DIFcontrasts = (
2.0 -1.0 -1.0,
0.0  1.0 -1.0
);
```

is entirely permissible. The above contrasts specify that, for the first comparison, group 1 is compared against the mean of groups 2 and 3 and, for the second contrast, that group 2 is compared against group 3. Users may find a review of contrasts within the context of post-hoc ANOVA analyses useful for conceptualizing the DIF contrasts supplied to flexMIRT^{TM}.

From the DIF analysis, an additional table is reported in the output, which we present on the next page. For each item that is present in both groups, a line in the table is added that reports the various Wald tests for that item. As we fit the 3PL to the items, we are provided with an overall test for DIF, a test of DIF with respect to the guessing parameter (labeled `X2g`), a test of the slope, assuming g is fixed across groups (labeled `X2a|g`), and a test of the intercept, assuming a and g are fixed across groups. Each of the reported Wald values also has the associated degrees of freedom and p-value printed.

Users are strongly encouraged to consult Woods et al. (2013) prior to using the `DIF = TestAll` command, as this paper provides a detailed description of the automated process being used. For example, the `TestAll` procedure expects that the model specified in the syntax has all item parameters fixed across groups and focal group parameters (mean(s) and variance(s)) freed; deviations from this may result in unexpected DIF results. Further, a review of the noted paper will reveal that the authors found the Wald-2 procedure, implemented via `DIF = TestAll;` in flexMIRT$^{\text{TM}}$, tends to result in inflated Type I error rates and suggested that this procedure may be best used for identifying anchors for a subsequent DIF analysis that examines only candidate items for DIF, rather than as a final DIF analysis from which conclusions regarding item functioning are drawn. Additionally, it is also recommended that when using the `TestAll` DIF sweep option that the standard error method be changed from the default. Conclusions reported in Woods et al. (2013) were based on the use of supplemented EM SEs (`SE = SEM;`). Users should also be aware that the automated process implemented via the `TestAll` command is only appropriate for use with unidimensional models; attempts to use this automated procedure with a MIRT model will result in an unidentified model and spurious results. However, the two-step process described in Woods et al. (2013) can be manually programmed by users wishing to conduct a DIF sweep in a MIRT model (in which the two-step procedure is completed via two separate syntax files).

Output 4.1: DIF Analyses Output Table

```
DIF Statistics for 3PL Items
Item numbers in:
 Grp1 Grp2 Total X2  d.f.        p      X2g  d.f.        p    X2a|g  d.f.        p  X2c|a,g  d.f.        p
    1    1       1.9     3  0.5876      0.1     1  0.8148      1.5     1  0.2275      0.4     1  0.5204
    2    2       7.7     3  0.0523      0.2     1  0.6773      4.5     1  0.0331      3.0     1  0.0832
    3    3       0.5     3  0.9228      0.0     1  0.9089      0.4     1  0.5484      0.1     1  0.7420
    4    4      17.0     3  0.0007      0.0     1  0.8686      0.3     1  0.5919     16.7     1  0.0001
    5    5       1.8     3  0.6122      0.1     1  0.7757      1.2     1  0.2758      0.5     1  0.4619
    6    6       0.7     3  0.8802      0.1     1  0.7920      0.5     1  0.4851      0.1     1  0.7374
    7    7       0.7     3  0.8758      0.0     1  0.8743      0.2     1  0.6353      0.4     1  0.5079
    8    8       8.9     3  0.0303      0.0     1  0.9190      7.2     1  0.0072      1.7     1  0.1935
    9    9       3.6     3  0.3084      0.2     1  0.6714      0.4     1  0.5474      3.1     1  0.0800
   10   10       1.5     3  0.6872      0.0     1  0.9073      0.7     1  0.4159      0.8     1  0.3706
   11   11       1.9     3  0.5967      1.0     1  0.3303      0.0     1  0.9027      0.9     1  0.3376
   12   12       0.7     3  0.8640      0.4     1  0.5050      0.0     1  0.9090      0.3     1  0.5962
```

flexMIRT^{TM} may also be used for more traditional DIF testing, estimating candidate DIF items relative to designated anchor items. Only relevant excerpts of the command file, “g341-19.difcand.flexmirt”, are presented. Again, the full file is available on the support page.

Example 4-6: DIF: Test Candidate Items (Excerpts)

```
<Options>
...
DIF = TestCandidates;
DIFcontrasts = (1.0 -1.0);

<Groups>
%Grade4%
File ="g341-19.grp1.dat";
Varnames = v1-v12;
N = 1314;
Ncats(v1-v12) = 2;
Model(v1-v12) = ThreePL;
DIFitems = v1,v2;
%Grade3%
File ="g341-19.grp2.dat";
Varnames = v1-v12;
N = 1530;
Ncats(v1-v12) = 2;
Model(v1-v12) = ThreePL;
DIFitems = v1, v2;

<Constraints>
Free Grade3, Mean(1);
Free Grade3, Cov(1,1);
Equal Grade3, (v3-v12), Guessing:
      Grade4, (v3-v12), Guessing;
Equal Grade3, (v3-v12), Intercept:
      Grade4, (v3-v12), Intercept;
Equal Grade3, (v3-v12), Slope:
      Grade4, (v3-v12), Slope;
     ...
```

The keyword used with the `DIF` statement is now `TestCandidates`, as opposed to `TestAll` in the previous example. The `DIFcontrasts` statement remains unchanged. To specify candidate DIF items, within the `<Groups>` section, we simply list the candidate item names using the `DIFitems` statement, found as the last entry within each group subsection of the excerpted code. Here, we are testing Items v1 and v2 for possible DIF. As can be seen in the `<Constraints>` section, the item parameters for the anchor items `(v3-v12)` are set equal across groups. The DIF table printed in the output is structured as in the previous example and will not be covered here.

4.5. Multiple Group Multilevel Model

Our next multiple group example will cover the flexMIRT^{TM} syntax options for handling nested data. Using the student cognitive outcomes data from the 2000 Program for International Student Assessment (see Adams & Wu, 2002), 31 mathematics items with mixed formats (multiple choice and constructed response) from students in both the United States and Ireland will be analyzed. The US data set is comprised of 2115 students, representing 152 different schools, and the Irish data set has 2125 student observations, coming from 139 different schools. From the general description of the data, it is clear that the complete set of data for this example has two groups (US and Irish students) and within each country, students (level-1 units) are nested within schools (level-2 units). The 31 items will be fit with a unidimensional model at each level (students and schools) within each group (country). Because the full command file (PISA00mathL2.flexmirt) is quite long, only commands relevant to multilevel aspect of the analysis will be highlighted.

Example 4-7: Multiple Group Multilevel Model (Excerpts)

```
<Groups>
%USA%
File ="PISA00Math.US.dat";
Missing = 9;
Varnames = SchID,StdID,
   ViewRoomQ1,  BricksQ1,
...
   CarpenterQ01,  PipelinesQ1;
Select = ViewRoomQ1, BricksQ1,
...
        CarpenterQ01,PipelinesQ1;
    CaseID = StdID;
    Cluster = SchID;
    Dimensions = 2;
    Between = 1;
    N = 2115;
 ...
  %Ireland%
    File ="PISA00Math.IE.dat";
 ...
    CaseID = StdID;
    Cluster = SchID;
    Dimensions = 2;
    Between = 1;
    N = 2125;
 ...
    <Constraints>
 ...
    Free USA, Cov(1,1);
    Free Ireland, Mean(1);
    Free Ireland, Cov(1,1);
```

The data files are specified as usual within each group. We have also set a missing data code using `Missing = 9;`, so all 9s in the data set will be interpreted as missing values, rather than the default missing value of -9. The variables are named in the typical fashion - the first two variables are of note because they indicate student ID numbers and school ID numbers, respectively, and will not be selected for IRT analysis. In general, a multilevel analysis has two kinds of latent dimensions, Between and Within. The Between dimensions, used for modeling level-2 (schools, in this case), will always precede the Within dimensions, used for modeling level-1 (students, in this example) when flexMIRT^{TM} is assigning dimensions. For this example, there are only two dimensions total so we specify the total number of dimensions to be used for the model (via `Dimensions = 2;`) and also denote that at level-2 there is one between-school dimension (`Between = 1;`). Considering the total number of dimensions is two, this implies that there is one within-school dimension at the student level. In the estimation and output, flexMIRT^{TM} will use the first dimension for between-school variations and the second dimension for within-school variations among students.

When a 2-level model has been requested, it is required that a variable defining the higher-level units be specified via the `Cluster` statement. In the syntax, we also provide a variable that identifies the first level data (individuals) with the `CaseID` statement. The latter can be convenient if the individual IDs are unique so that the scale scores computed by flexMIRT^{TM} can be merged with other data sets for further analysis.

The `<Constraints>` section contains extensive statements for setting up the necessary priors and equality constraints both within and across the two groups so that the desired hierarchical model is estimated. With the between-group equality constraints, the item parameters (for both level-1 and level-2 models) are set equal across the two countries, thereby creating anchoring so that the mean of Ireland schools can be freely estimated. The within-group equality constraints are specified so that the item slopes for the within dimension are equal to those for the between dimension. This implies that a random-intercept only model is fitted in each country, which permits the decomposition of between-school versus within-school variance.

Output 4.2: Two-Level Model - Means and Variances

```
Summary of the Data and Dimensions
               Group       USA   Ireland
   Missing data code         9         9
     Number of Items        31        31
 Number of L-2 units       152       139
 Number of L-1 units      2115      2125
 # Latent Dimensions         2         2
  Between Dimensions         1         1
   Within Dimensions         1         1
...
Group Latent Variable Means:
   Group              Label   P#   mu  1    s.e.   P#   mu  2    s.e.
       1                USA         0.00    ----         0.00    ----
       2            Ireland   79    0.36    0.04         0.00    ----

Latent Variable Variance-Covariance Matrix for Group  1: USA
   P# Theta  1    s.e.   P# Theta  2    s.e.
   78     0.52    0.07
          0.00    ----          1.00    ----

Latent Variable Variance-Covariance Matrix for Group  2: Ireland
   P# Theta  1    s.e.   P# Theta  2    s.e.
   80     0.11    0.02
          0.00    ----          1.00    ----
```

We can see from the provided output that the summary of data shows the multilevel syntax has been correctly interpreted. On average, the overall school-level achievement in Ireland is 0.36 standard deviations higher than that of the US. The more interesting result, however, is in the variance decompositions. In the US, with a fixed within-school variance of 1.0 (for identification), the between-school variance is estimated as 0.52. This implies an intra-class correlation of $0.52/(1 + 0.52) = 0.34$, showing a significant amount of school level variation among scores in the US. On the other hand, the intra-class correlation for Ireland is equal to $0.11/(1 + 0.11) = 0.10$, which implies that the school achievement-levels are much more homogeneous.

While the multilevel examples just discussed included two dimensions, technically making them multidimensional IRT (MIRT) models, they are multidimensional in a very specific way and actually undimensional at each of the two levels of analysis. In the next chapter, we cover MIRT models that exhibit more complex factor patterns, making the models multidimensional in a more dimensionally significant way.

CHAPTER 5

Multidimensional IRT

Although the previous chapter contained examples that were multidimensional, estimating two factors, they were also dealing with nested/hierarchical data. In the noted examples, the fitting of a multidimensional model was somewhat conflated with the multilevel modeling syntax so, in this chapter, we provide more straight-forward multidimensional examples.

5.1. Exploratory Factor Analysis with Analytic Rotation

The QOL dataset was introduced in Chapter 2, but a previously unmentioned aspect of the scale is it was designed to assess both the overall quality of life as well as the quality of life in 7 more specific subdomains (family, money, health, leisure, living, safety, and social), with each subdomain containing 4 to 6 items. Item 1 is a global quality of life item. We will be fitting a four factor EFA model to a subset of the items from the previously introduced QOL scale (items from the family, money, health, and leisure factors) and allowing the latent factors to correlate.

5.1.1 EFA with "Blind" Rotation

Example 5-1: EFA with Oblique CF-Quartimax Rotation

```
<Project>
Title = "QOL Data";
Description= "Items 2-21 - 4 Correlated Factors EFA";

<Options>
Mode = Calibration;
Quadrature = 21, 5.0;
Processors = 2;
```

```
Etol = 1e-3;
NewThreadModel = Yes;
FactorLoadings = Yes;

<Groups>
%Group1%
File ="QOL.DAT";
Varnames = v1-v35;
Select= v2-v21;
Ncats(v2-v21) = 7;
Model(v2-v21) = Graded(7);
BetaPriors(v2-v21) = 1.5;

Dimensions = 4;
//  Rotation options are: None/CFquartimax/CFvarimax/Target;
Rotation = CFQuartimax;
Oblique = Yes;

<Constraints>
```

In the `<Options>` section, the `Quadrature` command is encountered for the first time. This statement allows the user to specify the number of quadrature points to be used, 21 in this case, and the range over which those points should be spread (from -5.0 to 5.0 here). Reducing the number of quadrature points from the default value (49 points per dimension) may be necessary when fitting multidimensional models (exploratory and confirmatory) because the total number of points used is exponentially related to the number of dimensions. With higher dimensional models, the total number of quadrature points (and therefore processing time) may become unwieldy if the default value is maintained. **Users attempting to run these examples for themselves should be aware that even with reduced quadrature points and spread, the EFA examples still take a considerable amount of time to complete**. In extreme cases, flexMIRT^{TM} may use all available memory attempting to perform the quadrature calculations and be forced to abandon the analysis. If users encounter such a situation with a MIRT model, in which flexMIRT^{TM} reports a finishing time but no output pane is opened, reducing the number of quadrature points may resolve the issue. Another possible solution is to run the model using the Metropolis-Hastings Robbins-Monro

(MH-RM) algorithm (see Chapter 5 for MH-RM details).

Also in the `<Options>` section, we have allowed flexMIRT$^{\text{TM}}$ access to 2 processors, rather than the default of 1, to improve the speed of the analysis and have also specified `NewThreadModel= Yes;` which calls an updated implementation of multi-core processing that may be more efficient for high-dimensional models with a large number of quadrature points. Additionally, we have requested that flexMIRT$^{\text{TM}}$ provide factor loadings for each item, in addition to the standard IRT parameters.

In the `<Groups>`, the first several statements, specifying a datafile, selecting items, and specifying item models, etc., are similar to previous examples. We have included the `BetaPriors` command for all items in this example to help prevent Heywood cases in our EFA model. Specific to the multidimensional aspect of the current model, we find the `Dimensions` statement, which in most previous examples had been left at its default value of 1 and not explicitly stated. The selected items to be submitted for analysis are, substantively, related to the four factors of family, money, health, and leisure factors; thus, we have specified that our model should have 4 factors. Specific to the EFA that we want to conduct, the next command is related to the type of rotation we would like performed after the initial extraction. `Rotation` is the keyword that indicates to flexMIRT$^{\text{TM}}$ that an EFA is desired - to trigger an EFA run, a rotation other than `Rotation = None;` must be specified. All available rotations are covered in the Details of the Syntax chapter - for this example, we have selected the Crawford-Ferguson (CF) Quartimax rotation (e.g., Browne, 2001). Further, by setting `Oblique = Yes;` we have specified that the rotated factors should be allowed to correlate. If the `Oblique` statement were set to `No`, then the resulting rotated factors would be uncorrelated (orthogonal).

Output 5.1: EFA Output - Slope parameters for an unrotated solution

```
    Item              Label  P#     a 1    s.e.   P#     a 2    s.e.   P#     a 3    s.e.   P#     a 4    s.e.
       1                 v2   7    2.65    0.20          0.00    ----          0.00    ----          0.00    ----
       2                 v3  14    2.10    0.15   15     0.33    0.15          0.00    ----          0.00    ----
       3                 v4  22    4.07    0.33   23     0.51    0.21   24    -0.01    0.20          0.00    ----
       4                 v5  31    3.88    0.30   32     0.62    0.22   33     0.13    0.22   34    -0.03    0.19
       5                 v6  41    0.88    0.23   42     2.58    0.45   43     0.01    0.90   44     1.30    0.74
       6                 v7  51    0.55    0.15   52     1.46    0.27   53     0.14    0.51   54     0.74    0.42
       7                 v8  61    1.07    0.24   62     3.03    0.47   63     0.26    1.02   64     1.36    0.85
       8                 v9  71    0.94    0.22   72     2.71    0.42   73     0.28    0.93   74     1.28    0.79
       9                v10  81    0.72    0.13   82    -0.17    0.40   83     0.39    0.54   84     1.29    0.24
...
      16                v17 151    0.92    0.14  152     0.29    0.44  153     1.36    0.28  154     0.62    0.55
      17                v18 161    1.00    0.15  162     0.52    0.51  163     1.59    0.30  164     0.59    0.63
      18                v19 171    1.28    0.21  172     0.94    0.75  173     2.39    0.44  174     0.59    0.95
      19                v20 181    1.02    0.15  182     0.54    0.46  183     1.45    0.26  184     0.40    0.58
      20                v21 191    0.61    0.11  192     0.24    0.21  193     0.51    0.21  194     0.43    0.23
```

In the output produced by our EFA analysis, the typical IRT parameters (slopes and intercepts) are reported first. Of particular note in the slope parameter table (just presented) is the fact that not all items load on all slopes, as some users may have expected. Within the CFA context, EFA can be conceptualized as a minimally constrained confirmatory model, so while a large majority of the items load on all factors some constraints have been imposed to identify the model. Specifically, for an m-dimensional model, it is necessary to impose $(m(m-1))/2$ constraints for identification. flexMIRT$^{\text{TM}}$ meets this requirement by fixing the slope to zero on a given factor for any item where the item number is less than the factor number (e.g., item number two of the model, labeled v3, has fixed slopes on factors 3 and 4 as 2 [item number] is less than both 3 and 4 [factor numbers where slopes of v3 are fixed]).

After the item parameter tables, the factor loadings are printed, per our `FactorLoadings = Yes;` request. Below that we find the section labeled "...Rotated Loadings..." which are the estimates of primary interest.

Output 5.2: EFA Output - Rotated Factor Loadings and Factor Correlations

```
Oblique CF-Quartimax Rotated Loadings for Group 1: Group1
 Item                Label lambda 1 lambda 2 lambda 3 lambda 4
    1                   v2     0.85     0.06    -0.02     0.10
    2                   v3     0.77    -0.01    -0.00    -0.04
    3                   v4     0.92     0.00    -0.01    -0.02
    4                   v5     0.90    -0.03     0.04    -0.04
    5                   v6     0.00     0.04    -0.07    -0.88
    6                   v7     0.00     0.03     0.02    -0.69
    7                   v8     0.01    -0.02     0.03    -0.89
    8                   v9    -0.01    -0.00     0.04    -0.86
    9                  v10    -0.05     0.70     0.02     0.04
   10                  v11    -0.00     0.77    -0.04    -0.03
   11                  v12     0.01     0.67     0.06     0.01
   12                  v13     0.03     0.70    -0.02    -0.04
   13                  v14     0.03     0.75     0.01    -0.02
   14                  v15     0.09     0.49     0.19    -0.04
   15                  v16     0.03     0.06     0.66    -0.08
   16                  v17     0.01     0.14     0.66     0.07
   17                  v18    -0.00     0.06     0.74     0.01
   18                  v19    -0.02    -0.07     0.90    -0.03
   19                  v20     0.07    -0.02     0.72    -0.00
   20                  v21     0.08     0.16     0.28    -0.05

Oblique CF-Quartimax Rotated Factor Correlation Matrix for Group 1: Group1
          Theta  1 Theta  2 Theta  3 Theta  4
 Theta  1     1.00
 Theta  2     0.51     1.00
 Theta  3     0.51     0.61     1.00
 Theta  4    -0.38    -0.44    -0.52     1.00
```

As can be seen, following the specified oblique, CF-Quartimax rotation of the initial extraction values, all items are now loading on all factors. Typically, an EFA solution will be used to inform a subsequent confirmatory model. The rotated factor loading values may be examined to find groups of items that load saliently on each factor (such as items v2 - v5 on factor 1; v6 - v9 on factor 2). Below the rotated factor loadings, we find the non-zero correlations among the rotated factors, indicating that the `Oblique = Yes;` command was interpreted properly.

5.1.2 EFA with Target Rotation

The just-discussed example conducted analytic rotation with no information provided by the user regarding expectations. As noted in the introduction, however, theory and previous empirical work have provided us with information regarding how the items are expected to group together. flexMIRT^{TM} is able to accommodate such information through the use of rotation to a partially specified target (e.g., Browne, 2001).

Example 5-2: EFA with Target Rotation

```
<Project>
Title = "QOL Data";
Description= "Items 2-9 - 2D EFA with target rotation";

<Options>
Mode = Calibration;
Quadrature = 21, 5.0;
Processors = 2;
Etol = 1e-3;
NewThreadModel = Yes;
FactorLoadings = Yes;
SavePCC = Yes;

<Groups>
%Group1%
File ="QOL.DAT";
Varnames = v1-v35;
Select= v2-v9;
Ncats(v2-v9) = 7;
Model(v2-v9) = Graded(7);
BetaPriors(v2-v9) = 1.5;

Dimensions= 2;
Rotation = Target;
Oblique = Yes;

```

```
UnspecifiedTargetElement = 9;
Target = (
    9 0,
    9 0,
    9 0,
    9 0,
    0 9,
    0 9,
    0 9,
    0 9);

<Constraints>
```

In the example syntax for target rotation, the commands are largely unchanged from the previous example, although we have reduced the number of items/dimensions used in the interest of analysis run time and specified `Rotation = Target;` to indicate that target rotation is desired. Also notice that in the `<Options>` section, we have included the command `SavePCC= Yes;` which creates an additional *-pcc.txt output file containing the eigenvalues of the polychoric correlation matrix, the unique elements of the polychoric correlation matrix of the analyzed items, as well as item thresholds.

When target rotation is used, it is necessary to provide flexMIRT^{TM} a matrix which will serve as the target for the rotation. By this, we mean flexMIRT^{TM} will perform the rotation so all loadings specified as 0 in the target matrix will be as close to zero as possible while the unspecified elements of the matrix (the 9's in the example target matrix) have no expectations/desired values. As can be seen, the theoretical item groupings noted in the introduction (e.g., items v2 - v5 comprise a "family QOL" factor) are set up here by leaving each subset of items as unspecified elements on one of the factors.

Output 5.3: EFA Output - Factor Loadings Rotated to a Target

```
Oblique Target Rotated Loadings for Group 1: Group1
 Item              Label Target lambda 1 Target lambda 2
    1                 v2      ?    -0.87   0.00     0.08
    2                 v3      ?    -0.75   0.00    -0.06
    3                 v4      ?    -0.93   0.00     0.01
    4                 v5      ?    -0.90   0.00    -0.04
    5                 v6   0.00     0.01      ?    -0.87
    6                 v7   0.00    -0.01      ?    -0.71
    7                 v8   0.00    -0.01      ?    -0.89
    8                 v9   0.00     0.00      ?    -0.88
```

The output from a target rotation is similar to other analyses up to the rotated factor loadings table. In the reported "...Rotated Loadings..." section shown above flexMIRT^{TM} reports estimated, rotated factor loading values. Additionally, it also reprints the target matrix, column by column, it attempted to rotate the solution to, with unspecified elements represented by ?s.

5.2. Confirmatory Multidimensional Model - 4 Correlated Factors

The typical progression of modeling new item sets generally goes from EFA to CFA, using EFA results and theory to inform the factor pattern of the confirmatory model. The first EFA example with non-target rotation provided good evidence that a 4-factor solution is supported by the data but also contained many items that loaded on factors they are not theoretically related to. To assess the suitability of a more parsimonious model, we will fit a confirmatory four factor model to the subset of the items (v2-v21) we previously submitted to EFA, allowing items to load on/have slope parameters for only those factors to which they are expected to be related.

Example 5-3: Confirmatory MIRT Model with Correlated Factors - Graded

```
<Project>
Title = "QOL Data";
Description= "Items 2-21,  4 Correlated Factors CFA"
 "NOTE: THIS EXAMPLE TAKES ~1hr TO COMPLETE;

<Options>
Mode = Calibration;
Quadrature  = 21, 5.0;
Processors  = 4;
SavePRM = Yes;
FactorLoadings = Yes;

<Groups>
%Group1%
File ="QOL.DAT";
Varnames = v1-v35;
Select= v2-v21;
Dimensions = 4;
Ncats(v2-v21) = 7;
Model(v2-v21) = Graded(7);

<Constraints>
Fix(v2-v21),Slope;

Free(v2-v5),Slope(1);
Free(v6-v9),Slope(2);
Free(v10-v15),Slope(3);
Free(v16-v21),Slope(4);

Free Cov(2,1);
Free Cov(3,1);
Free Cov(3,2);
Free Cov(4,1);
Free Cov(4,2);
Free Cov(4,3);
```

Points of note in the syntax, relevant to the fitted multidimensional model, are the `Dimensions` statement in the `<Groups>` section and the various constraints encountered in the `<Constraints>` section. As described earlier, we are fitting a model with four factors, hence the `Dimensions = 4;` statement. Within the `<Constraints>` section, we are assigning the items to the appropriate factors.

The first step in assigning items to factors is to fix all slope values to 0, via the `Fix (v2-v21),Slope;` statement. This allows us, in the next step, to free the individual items to load on only the appropriate factors. As can be seen in the four `Free` statements, we are specifying that Items 2 through 5 load on the first factor, Items 6 through 9 load on the second factor, Items 10 through 15 load are assigned to the third factor, and the remaining items are assigned to factor 4. This corresponds to the description provided earlier, in which Items 2 to 5 belong to the "Family" factor only and Items 6 to 9 belong to only the "Money" factor and so on. The final constraints, such as `Free Cov(2,1);`, are used to tell flexMIRT$^{\mathrm{TM}}$ to freely estimate the covariance/correlation between the factors. Using the lower triangle of the variance/covariance matrix of the latent factors to determine positions, the variance of the first factor is found at Cov(1,1), the variance of the second factor at Cov(2,2) and the covariance of those two factors is located in Row 2, Column 1 (i.e., Cov(2,1)) of the matrix.

Additionally, in the `<Options>` section, the `Quadrature` command is again encountered. As with the previous EFA examples, reducing the number of quadrature points from the default value (49 points per dimension) may be necessary when fitting MIRT models because the total number of points used is exponentially related to the number of dimensions. With higher dimensional models, the total number of quadrature points (and therefore processing time) may become unwieldy if the default value is maintained.

Within the output, the various sections are arranged as with the other examples. In the output file, the factor loadings are printed below the estimated item parameters, as requested by the `FactorLoadings = Yes;` statement. As seen before, the group means of the latent variables are reported followed by the estimated latent variable variance/covariance matrix. This section of the output is presented below.

Output 5.4: Correlated Four Factor Confirmatory Model Output - Loadings, Means and Variances

```
Factor Loadings for Group 1: Group1
 Item              Label lambda 1  s.e. lambda 2  s.e. lambda 3  s.e. lambda 4  s.e.
    1                 v2     0.83  0.03     0.00  ----     0.00  ----     0.00  ----
    2                 v3     0.78  0.03     0.00  ----     0.00  ----     0.00  ----
    3                 v4     0.93  0.02     0.00  ----     0.00  ----     0.00  ----
    4                 v5     0.92  0.02     0.00  ----     0.00  ----     0.00  ----
    5                 v6     0.00  ----     0.87  0.03     0.00  ----     0.00  ----
    6                 v7     0.00  ----     0.72  0.05     0.00  ----     0.00  ----
    7                 v8     0.00  ----     0.90  0.02     0.00  ----     0.00  ----
    8                 v9     0.00  ----     0.88  0.03     0.00  ----     0.00  ----
    9                v10     0.00  ----     0.00  ----     0.66  0.06     0.00  ----
   10                v11     0.00  ----     0.00  ----     0.74  0.05     0.00  ----
   11                v12     0.00  ----     0.00  ----     0.71  0.05     0.00  ----
   12                v13     0.00  ----     0.00  ----     0.72  0.05     0.00  ----
   13                v14     0.00  ----     0.00  ----     0.78  0.04     0.00  ----
   14                v15     0.00  ----     0.00  ----     0.72  0.05     0.00  ----
   15                v16     0.00  ----     0.00  ----     0.00  ----     0.77  0.04
   16                v17     0.00  ----     0.00  ----     0.00  ----     0.73  0.04
   17                v18     0.00  ----     0.00  ----     0.00  ----     0.76  0.04
   18                v19     0.00  ----     0.00  ----     0.00  ----     0.84  0.03
   19                v20     0.00  ----     0.00  ----     0.00  ----     0.75  0.04
   20                v21     0.00  ----     0.00  ----     0.00  ----     0.49  0.07

QOL Data
Items 2-21, 4 Correlated Factors.

NOTE: THIS EXAMPLE TAKES ~1hr TO COMPLETE

Group Latent Variable Means:
   Group              Label   P#   mu  1    s.e.   P#   mu  2    s.e.   P#   mu  3    s.e.   P#   mu  4    s.e.
       1             Group1         0.00    ----         0.00    ----         0.00    ----         0.00    ----

Latent Variable Variance-Covariance Matrix for Group  1: Group1
   P# Theta  1    s.e.   P# Theta  2    s.e.   P# Theta  3    s.e.   P# Theta  4    s.e.
         1.00    ----
  141    0.41    0.04         1.00    ----
  142    0.55    0.03  143    0.50    0.04         1.00    ----
  144    0.55    0.04  145    0.56    0.03  146    0.69    0.03         1.00    ----
```

The pattern of the estimated loadings indicates that the items were assigned to the factors as expected, with certain loadings restricted to 0.00 and having no associated SE. Because there is only a single group, the latent variable means were also restricted to zero and have no associated SE, indicating they were not estimated. The final section in the output excerpt reports the correlation matrix of the latent variable - for example, the correlation between the first two standardized factors is estimated to be 0.41, with a SE of 0.04.

5.3. The Bifactor Model

The item bifactor model is a special multidimensional IRT model, in which each item is restricted to load on, at most, two factors. The typical set up for a bifactor model is one factor on which all the items load, termed the general factor, and several additional factors, termed group-specific factors, onto which only subsets of the items load. A common substantive example of this would be a reading test made up of several short reading passages that each have several items probing the students comprehension for a particular passage. The general factor in this case would be “Reading” and each passage would have its own specific factor, onto which only the items related to that passage load.

Chief among the benefits of the item bifactor model is computational efficiency in which full-information maximum marginal likelihood estimation of item parameters can proceed with 2-dimensional integration regardless of how many group-specific factors are in the model, due to a result first discovered by Gibbons and Hedeker (1992). The bifactor model in flexMIRT$^{\text{TM}}$ is in fact obtained as a two-tier model (Cai, 2010a). flexMIRT$^{\text{TM}}$ is fully capable of multiple-group bifactor or two-tier modeling as detailed in Cai, Yang, and Hansen (2011). Also note that Bradlow, Wainer, and Wang’s (1999) testlet response theory model can be understood as a further restricted bifactor model with within-item proportionality constraints on the general and group-specific slopes (see also, Glas, Wainer, & Bradlow, 2000). With its generalized capability to impose constraints, flexMIRT$^{\text{TM}}$ is able to estimate testlet models too.

5.3.1 Bifactor Model Using the Reparameterized Nominal Categories Model

Here, we use the full 35 item Quality of Life data to illustrate a bifactor analysis, specifying a general QOL factor and 7 specific factors to capture the subdomains mentioned earlier. Gibbons et al. (2007) reported a graded bifactor analysis of this data set using the original 7 categories and we will replicate this structure but employ a full-rank nominal model.

Example 5-4: Bifactor Structure - Nominal Model

```
<Project>
Title = "QOL Data";
Description = "35 Items Bifactor Nominal Model";

<Options>
Mode = Calibration;
Quadrature = 21, 5.0;
Etol = 1e-4;
Mtol = 1e-5;
```

```

<Groups>
%Group1%
File ="QOL.DAT";
Varnames = v1-v35;
N = 586;
Ncats(v1-v35) = 7;
Model(v1-v35) = Nominal(7);
Dimensions = 8;
Primary= 1;

<Constraints>
Fix(v1-v35, ScoringFn(1);

Fix(v1-v35), Slope;

Free(v1-v35), Slope(1);
Free(v2-v5), Slope(2);
Free(v6-v9), Slope(3);
Free(v10-v15), Slope(4);
Free(v16-v21), Slope(5);
Free(v22-v26), Slope(6);
Free(v27-v31), Slope(7);
Free(v32-v35), Slope(8);

```

With regard to the bifactor structure, in the `<Groups>` section, a multidimensional model with 8 dimensions is first specified. This is the total number of domains/factors, both general and specific. The `Primary` command specifies, out of the total number of dimensions, how many are primary/general dimensions. As noted in the data description, the QOL scale is made of 1 general domain and 7 subdomains.

Once the dimensionality of the model is set up in the `<Groups>` section, the details of which items load onto which specific factors are given in the `<Constraints>` section. The first constraint serves to fix all item slopes to zero, facilitating the subsequent freeing of parameters. The next 8 lines set up the factor structure by 1) freeing all the item slopes on the first dimension, which will be the general factor, and 2) assigning the items to the appropriate subdomains, based on the item content. *Note that it is absolutely necessary* *to ensure that the primary dimension(s) precede the group-specific dimensions when freeing items to load on the factors.* flexMIRT$^{\text{TM}}$ automatically reduces the dimensionality of integration to 2 for this model and, consequently, the analysis runs extremely fast.

The `Model(v1-v35) = Nominal(7);` statement in the `<Groups>` section indicates that we wish to fit the Nominal Model (for 7 response categories) to the items. In the current code, we have opted not to modify either the scoring functions contrasts (which can be accomplished via the `Ta(var) = ;` statement) or the intercept contrast matrix (which can be accomplished via the `Tc(var) = ;` statement). Without changes, the default option of Trend matrices will be used for both sets of parameters. In the `<Constraints>` section, the first constraint imposed fixes the first scoring function contrast for all 35 items (the default value is 1.0), as required for identification (see Thissen et al., 2010). With the default contrast matrices, the fixed first scoring function contrast of 1.0 ensures that the first and last scoring function values are fixed to 0 and 6, respectively. The remaining 5 scoring function values are reparameterized by the contrast matrices and become estimable parameters (as shown below).

In the `<Options>` section, the number of quadrature points is reduced for time-efficiency, as was done with the other MIRT models, using the `Quadrature` statement and the tolerance values are adjusted down to obtain tighter solutions.

Output 5.5: Nominal Model Scoring Function Estimates - Excerpt

```
Nominal Model Scoring Function Values (s) under Dimension 1 for Group 1: Group1
    Item            Label    s  1     s  2     s  3     s  4     s  5     s  6     s  7
       1               v1    0.00     0.76     1.18     1.46     3.29     4.61     6.00
       2               v2    0.00     0.79     1.53     2.58     4.18     4.77     6.00
...
      15              v15    0.00     1.32     1.09     2.25     3.22     4.16     6.00
      16              v16    0.00     1.52     1.71     2.65     3.96     4.99     6.00
      17              v17    0.00     1.81     2.47     2.78     3.91     4.69     6.00
...
      34              v34    0.00     1.48     1.40     2.32     3.53     4.45     6.00
      35              v35    0.00     1.39     1.10     1.65     3.32     4.19     6.00
```

An examination of the output excerpt presented shows that a number of general dimension scoring function values exhibit reversal of categories from the presumed order (such as in Items 13, 15, 34, and 35) or category boundaries that are very close together (such as categories 2 and 3 in Items 16). As

Preston, Reise, Cai, and Hays (2011) explained, when the estimated scoring function values deviate from the presumed order of $0, 1, 2, \ldots, 6$, the nominal model provides additional category-specific information that other ordinal polytomous IRT models (e.g., partial credit or graded) do not provide. As noted earlier, the item responses were collected on a 7-point categorical response scale indicating the extent to which the respondent is satisfied with the content probed in the item, where: 0=terrible; 1=unhappy; 2=mostly dissatisfied; 3=mixed, about equally satisfied and dissatisfied; 4=mostly satisfied; 5=pleased; and 6=delighted. There may be some uncertainty as to how people interpret the order of the middle categories.

5.3.2 Bifactor Model Using the 2PL

Due to the undesirable properties when analyzing the QOL data with all 7 categories, we will again demonstrate how flexMIRT$^{\text{TM}}$ may be used to recode data, this time combining some response categories, rather than the simple "reduce each value by one" shown in Example 2-5 to get zero-based item responses for analysis. We will fit the same bifactor factor pattern to the QOL data, but rather than utilizing all 7 categories, we will collapse the data into dichotomous responses and fit the 2PLM.

Example 5-5: Bifactor Structure - 2PL

```
<Project>
Title = "QOL Data";
Description = "35 Items Bifactor Showing Respond Collapsing";

<Options>
Mode = Calibration;
Quadrature = 21, 5.0;
SavePRM = Yes;
Processors = 4;

<Groups>
%Group1%
File ="QOL.DAT";
Varnames = v1-v35;
N = 586;
Ncats(v1-v35) = 7;
```

```
Code(v1-v35) = (0,1,2,3,4,5,6),(0,0,0,1,1,1,1);
Model(v1-v35) = Graded(2);
Dimensions = 8;
Primary = 1;

<Constraints>
Fix(v1-v35),Slope;

Free(v1-v35),Slope(1);
Free(v2-v5),Slope(2);
Free(v6-v9),Slope(3);
Free(v10-v15),Slope(4);
Free(v16-v21),Slope(5);
Free(v22-v26),Slope(6);
Free(v27-v31),Slope(7);
Free(v32-v35),Slope(8);
```

The `Code` command is used to collapse the 7 original categories into 2. A point of note here is that the `Ncats` value provided refers to the number of categories in the raw data file, not the number of categories that will be used for modeling, which is specified with the `Model` statement. An examination of the `Code` statement shows that original responses of 0,1, and 2 will be recoded into 0s and raw responses of 3-6 will become 1s after the recoding by the program. The bifactor structure remains unchanged from the previous example, as may be seen in the `<Constraints>` section.

5.3.3 Testlet Response Model Using the 2PL

As mentioned when introducing the bifactor model, the testlet model can be obtained as a special case of the bifactor model. Due to interest in such models, we present an example of the testlet response model, still using the QOL dataset from the previous examples.

Example 5-6: Testlet Model - 2PL - Excerpt

```
 <Project>
 Title="QOL Data";
 Description="35 Items Graded Testlet Response Model";
...
   <Constraints>
   Fix(v1-v35),Slope;
   Free(v1-v35),Slope(1);
   Free(v2-v5),Slope(2);
   Free(v6-v9),Slope(3);
   Free(v10-v15),Slope(4);
   Free(v16-v21),Slope(5);
   Free(v22-v26),Slope(6);
   Free(v27-v31),Slope(7);
   Free(v32-v35),Slope(8);

   Equal G,(v2-v5),Slope(1) : G,(v2-v5),Slope(2);
   Equal G,(v6-v9),Slope(1) : G,(v6-v9),Slope(3);
   Equal G,(v10-v15),Slope(1) : G,(v10-v15),Slope(4);
   Equal G,(v16-v21),Slope(1) : G,(v16-v21),Slope(5);
   Equal G,(v22-v26),Slope(1) : G,(v22-v26),Slope(6);
   Equal G,(v27-v31),Slope(1) : G,(v27-v31),Slope(7);
   Equal G,(v32-v35),Slope(1) : G,(v32-v35),Slope(8);

   Free Cov(2,2);
   Free Cov(3,3);
   Free Cov(4,4);
   Free Cov(5,5);
   Free Cov(6,6);
   Free Cov(7,7);
   Free Cov(8,8);
```

To save space, we focus on the <Constraints> section for this example - the `<Options>` and `<Groups>` are similar to previous examples. The full code for this example is available on the flexMIRT$^{\text{TM}}$ support page.

Using the first group of statements in the `<Constraints>` section, we set up the bifactor structure, exactly as was done in the previous two examples, by fixing all slopes with the `Fix` statement and then free slopes onto factors as needed to achieve the appropriate structure, including non-overlapping testlets.

The second set of statements, comprised of `Equal` statements, is used to set slopes across factors equal within items. As briefly noted earlier, the testlet response model is a constrained form of the bifactor model imposing within-item proportionality constraints; equality is the form of proportionality employed here.

Finally, with the third group of constraints, we free the variances of all the specific factors. The rationale for including the testlet parameters in this manner (via equality constraints and freed specific variances), as well as the interpretation of the model parameters is provided in Glas et al. (2000), specifically in what they label Alternative 1.

As noted in various places throughout the MIRT chapter, high-dimensional models estimated with the default Bock-Aitkin EM algorithm can take an extremely long time to complete or may not complete at all due to computational limitations of the method. In the next chapter, we cover an alternate estimation method, based on Markov chain Monte Carlo (MCMC) principals, that is able to estimate such high-dimensional models easily.

CHAPTER 6

Alternative Estimation Methods for High-Dimensional and Complex Models

Although Bock and Aitkin's (1981) EM algorithm (BAEM) made IRT parameter estimation practical, the method does have shortcomings with the most noticeable one being its limited ability to generalize to truly high-dimensional models. This is due primarily to the need to evaluate high-dimensional integrals in the likelihood function for the item parameters. As the number of dimensions of a model increases linearly, the number of quadrature points increases exponentially, making BAEM estimation unwieldy and computationally expensive for models with more than three or four latent factors.

Two alternative estimation methods which avoid the so-called "curse of dimensionality" associated with BAEM estimation have been implemented in flexMIRT©: a Metropolis-Hastings Robbins-Monro (MH-RM) algorithm (Cai, 2010b, 2010c) and Markov chain Monte Carlo (MCMC)-based estimation (e.g., Edwards, 2010). This chapter serves as a detailed examination of the particulars of MH-RM and MCMC estimation and flexMIRT™ settings and syntax commands used when estimating models via these alternate methods. We will briefly discuss these two methods to acquaint those users less familiar with these modern estimation methods, cover the available flexMIRT™ options and syntax commands that are specific to these estimation routines, and provide working examples with discussion.

6.1. Overview of MH-RM Estimation

In brief, the MH-RM algorithm is a data augmented Robbins-Monro type (RM; Robbins & Monro, 1951) stochastic approximation (SA) algorithm driven by the random imputations produced by a Metropolis-Hastings sampler (MH; Hastings, 1970; Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953), which is a sampling method-based on the principles of MCMC. The MH-RM algorithm is motivated by Tittering's (1984) recursive algorithm for incomplete data estimation, and is a close relative of Gu and Kong's (1998) SA algorithm. It can also be conceived of as an extension of the Stochastic Approximation EM algorithm (SAEM; Celeux & Diebolt, 1991; Celeux, Chauveau, & Diebolt, 1995; Delyon, Lavielle, & Moulines, 1999) when linear sufficient statistics do not naturally exist in the complete data model.

Within the flexMIRT$^{\text{TM}}$ implementation, the MH-RM iterations improve on initial start values (Stage I) and further improves upon these values during a supplemented EM-like stage (Stage II) - both Stage I and Stage II use a fixed number of cycles. Stage III is home to the MH-RM algorithm and where the primary estimation of parameters occurs. Cycles in Stage III terminate when the estimates stabilize.

Cycle $j+1$ of the MH-RM algorithm for multidimensional IRT consists of three steps:

- *Imputation.* In the first step, conditional on provisional item and latent density parameter estimates $\boldsymbol{\beta}^{(j)}$ from the previous cycle, random samples of the individual latent traits $\boldsymbol{\theta}^{(j+1)}$ are imputed using the MH sampler from a Markov chain having the posterior of the individual latent traits $\pi(\boldsymbol{\theta}|\boldsymbol{Y},\boldsymbol{\beta}^{(j)})$ as the unique invariant distribution.

- *Approximation.* In the second step, based on the imputed data, the complete data log-likelihood and its derivatives are evaluated so that the ascent directions for the item and latent density parameters can be determined later.

- *Robbins-Monro Update.* In the third step, RM stochastic approximation filters are applied when updating the estimates of item and latent density parameters. First, the RM filter will be applied to obtain a recursive stochastic approximation of the conditional expectation of the complete data information matrix. Next, we use the RM filter again when updating the new parameter estimates. Cai (2010c) showed that the sequence of parameters converges with probability 1 to a local maximum of the likelihood of the parameter estimates, given the observed data.

For more detailed descriptions of the estimation method, see Cai (2010b, 2010c).

6.2. Overview of MCMC Estimation

Markov chain Monte Carlo methods evolved from work done on the Manhattan project by Nicholas Metropolis and colleagues (see Metropolis & Ulam, 1949; Metropolis et al., 1953). A generalization produced by Hastings (1970) led to the common form of the algorithm we see now. The resulting algorithm (the Metropolis-Hastings algorithm) is a very general MCMC method. The powerful aspect of the contribution, as explained by originators of the method, "...is that we avoid dealing with multiple integrations or multiplications of the probability matrices, but instead sample single chains of events." (Metropolis & Ulam, 1949, p.339)

MCMC is perhaps most commonly associated with Bayesian inference and statistics. As noted by several authors though, MCMC is actually not inherently Bayesian (see Geyer, 1996 for a more detailed discussion). Despite that, we suspect the vast majority of situations where users will see MCMC will be in the Bayesian context. While our goal is not to explain Bayesian statistics, we must define a few terms before proceeding. Critical to our discussion is the idea of a posterior distribution, which also necessitate the explanation of a prior distribution. The posterior is the product of the likelihood and the prior. The likelihood is a function we should be familiar with from maximum likelihood (ML). In ML, we choose parameters such that the likelihood of the data given those parameters is maximized. This is the same likelihood we find in Bayesian statistics. The difference is that we also find a prior distribution, which, depending on which Bayesian you ask, is either a summary of existing evidence about a parameter or a statement of the analyst's subjective belief about what the parameters is. The focus of inference is then the posterior distributions of our parameters.

MCMC estimation methods can be thought of as Monte Carlo integration using Markov chains (Gilks, Richardson, & Spiegelhalter, 1996a). Monte Carlo integration works by drawing samples from $\pi(.)$, some target distribution of interest, and then computing averages to approximate expectations. The difficulty is making independent draws from $\pi(.)$ By creating a Markov chain with $\pi(.)$ as its target distribution, one is able to make dependent draws from the distribution of interest. A Markov chain is a sequence of random events such that what happens at time t+1 depends on the state of the chain at time t and not any prior state. Once the Markov chain has converged (i.e., is stationary), samples from that chain will approximate samples from the distribution of interest. In models like IRT, the distributions of interest are posterior distributions for item and person parameters. Point estimates can be taken as means or modes of these distributions and we can learn about the variability in our estimates by calculating the standard deviation of the posterior (i.e., a kind of standard error).

Applications of MCMC in the IRT world are now fairly commonplace. Significant early efforts by Albert (1992), Patz and Junker (1999a, 1999b), and Fox and Glas (2001) paved the way. More specific information about MCMC as it relates to IRT can be found in Edwards (2010) and Wirth and Edwards (2007). There are hundreds of papers and dozens of books on the topic, but we have found Gilks, Richardson, and Spiegelhalter (1996b) and chapters 10 and 12 in Gill (1996) to be especially useful.

6.3. Alternative Estimation Method-Specific Syntax Options

Below provides all MH-RM and MCMC-specific commands collected together. The only required statements for MH-RM or MCMC to run is the algorithm call, which is bolded in the table. All other commands are optional, but **users should review the descriptions provided in the following pages as there are syntax statements whose defaults should to adjusted on a model-by-model basis to obtain converged, stable, and trustworthy solutions.**

Syntax Display 6.1: Engine Mode Settings to Call MH-RM or MCMC Estimation and MH-RM Specific Commands

```
<Options>
Algorithm = MHRM/MCMC;
   //Shared MH-RM and MCMC commands
RndSeed = 1842;
Imputations = 1;
Score = MI;
  // Note: other scoring methods can be used with MH-RM and MCMC,
 but Score = MI; is only available when non-BAEM est. is used

SaveMCO = Yes/No;
ProposalStd = 1.0;
ProposalStd2 = 1.0;

 //MH-RM-specific commands
Burnin = 10;
Thinning = 10;
Stage1 = 200;
Stage2 = 100;
Stage3 = 2000;
Stage4 = 0;
InitGain = 0.10;
Alpha = 1.0;
Epsilon = 1.0;
WindowSize = 3;
MCsize = 2500;
```

```
 //MCMC-specific commands or defaults
Burnin = 500;
Thinning = 25;
ItemProposalStd = 0.1;
MaxCycle = 500;

<Groups>
Covariates = vars/0;
L2covariates = 0;
CovariateCorr =  0;
```

The alternate estimation algorithm are available for calibration runs, scoring-only runs (particularly for when plausible values/multiple imputation theta estimates are desired), or combined calibration and scoring runs. To call the one of the alternate estimation algorithms, only `Algorithm = MHRM;` or `Algorithm = MCMC;`is required; all other listed commands are optional, with the default values shown above.**However, users should review the descriptions provided in the following pages as there are syntax statements whose defaults should to adjusted on a model-by-model basis to obtain converged, stable, and trustworthy solutions.**

`Imputations` controls the number of imputations from the MH step per RM cycle. A larger value will lead to smoother behavior from the algorithm. In most cases, a value of 1 should be sufficient for obtaining accurate point estimations, however this number may be increased for better standard errors. Additionally, when `Score = MI;`, which is only available with non-BAEM estimation, the value of `Imputations` will also control the number plausible values drawn from individual theta posteriors and saved to the -sco file.

The `Burnin` statement controls the number of draws that are discarded from the start of the MH sampler. These draws are discarded to avoid using values that were sampled before the chain had fully converged to the target distribution. `Burnin = 10;` tells flexMIRT^{TM} to discard the first 10 values obtained by the MH sampler.

Thinning refers to the sampling-method based estimation practice of retaining only every kth draw from a chain. Thinning is used, in part, to reduce the possible autocorrelation that may exist between adjacent draws. The `Thinning` statement sets the interval for the MH sampler - meaning if `Thinning = 10;`, every 10th draw by the sampler will be retained.

`SaveMCO = Yes;` is used to save out, when `Algorithm = MHRM;`, the MH-RM Stage 1 iteration history or, when `Algorithm = MCMC;`, the drawn parameter values into a separate -mco output file. **When using MCMC estimation, the values in the –mco file should be plotted to examine MCMC chain convergence.** In the -mco file when using MCMC estimation, the columns are the individual estimated parameters, with parameters reported in parameter number order (which is printed in the -irt file). The first row of the -mco file are the starting values for the parameters, the next rows will be the draws discarded as the burn-in (e.g., if `Burnin=250;`, the next 250 rows will be draws from those 250 cycles). Following the burn-in rows, the thinned main cycle rows draws are reported. If `Thinning = 10;` and `MaxCycle = 1000`, that means 1000 main cycle draws will be in the -mco file, but 1000*10 draws were completed to get those values. Users are directed to Edwards (2010) for an oveview of examining MCMC chain convergence and relevant references.

`ProposalStd` and `ProposalStd2` control the dispersion of the Metropolis proposal densities for the first and second level of the specified model, respectively. If a single level model is specified, only the `ProposalStd` command will be used. **Although default values of 1.0 have been set for both of these commands, these values need to be adjusted on a case-by-case basis**. The values used will depend on the complexity of the model, the number of items, the type of items (e.g., dichotmous, polytomous), and the model fit to the items (e.g., Graded, 3PL, Nominal). The user should choose `ProposalStd` and `ProposalStd2` values so that the long-term average of the acceptance rates (which is printed for each iteration in the progress window) for level 1 and level 2 (if applicable) are around 0.5 for lower dimensional models ($<=$ 3 factors) and in the 0.2 - 0.3 range for higher dimensional/more complex models. Generally speaking, increasing the `ProposalStd` value will result in lowered acceptance rates while decreasing the value will result in higher acceptance rates. Users are directed to Roberts and Rosenthal (2001) for optimal scaling, choice of dispersion constants, and long-term acceptance rates of Metropolis samplers.

`Stage1` determines the number of Stage I (constant gain) cycles. The Stage I iterations are used to improve default or user-supplied starting values for the estimation that occurs in Stage II. `Stage2` specifies the number of Stage II (Stochastic EM, constant gain) cycles. The Stage II iterations are used to further improve starting values for the MH-RM estimation that occurs in Stage III. `Stage3` sets the maximum number of allowed MH-RM cycles to be performed.

`Stage4` determines the method by which SEs are found. If `Stage4 = 0;` then SEs will be approximated recursively (see Cai, 2010b). If a non-zero value is given then the Louis formula (Louis, 1982) is used directly. If the Louis formula is to be used, the supplied value determines the number of iterations of the SE estimation routine; this number will typically need to be large (i.e., 1000 or more).

`InitGain` is the gain constant for Stage I and Stage II. If the algorithm is initially taking steps that are too large this value may be reduced from the default to force smaller steps.

`Alpha` and `Epsilon` are both Stage III decay speed tuning constants. `Alpha` is the first tuning constant and is analogous to the `InitGain` used in Stages I and II. `Epsilon` controls the rate at which the sequence of gain constants converge to zero. Specified `Epsilon` values must be in the range (0.5, 1], with a value closer to 1 indicating a faster rate of convergence to zero.

`WindowSize` allows the user to set the convergence monitor window size. Convergence of the MH-RM algorithm is monitored by calculating a window of successive differences in parameter estimates, with the iterations terminating only when all differences in the window are less than 0.0001. The default value of the window size is set at 3 to prevent premature stoppage due to random variation.

`MCSize` is the Monte Carlo size for final log-likelihood, AIC, and BIC approximations.

When using MCMC, the item parameter draws are divided up into independent segments in the flexMIRT$^{\text{TM}}$ implemented Metropolis-Hastings within Gibbs set up. `ItemProposalStd` sets the proposal dispersion for item parameter segments.

`MaxCycle` controls the total number of MCMC draws to be accepted following the specified number of burn in cycles. For instance, if `MaxCycle = 1000` that means that 1000 draws will be printed to the -mco file but the total number of completed draws will be MaxCycle number of draws multipled by the specified thinning value.

Within the `<Groups>` section there are handful of commands, regarding covariates, that are only available when one of the alternate estimation methods is used. During calibration, `Covariates` is used to supply flexMIRT$^{\text{TM}}$ with a list of variables that will serve as predictors of the latent variable estimates. It is expected that continuous variables will be supplied as covariates or, for categorical covariates, that appropriate codings (effect coding, dummy coding, etc.) into multiple variables have been constructed prior to being supplied to flexMIRT$^{\text{TM}}$. When used in simulation, `Covariates` is used to specify the number of covariates that should be generated as part of the model.

The `L2covariates` statement is used to tell flexMIRT$^{\text{TM}}$ how many of the covariates supplied in the `Covariates` should be used as predictors of the level-2 (Between) latent variables. For example, during calibration when `L2covariates` is set to a non-zero value, the first x variables listed in the `Covariates` statement will be implemented as predictors of the higher-level latent variable(s). When used in with a simulation, `L2covariates` indicates that the first x simulated covariates will apply to level-2 factors only.

`CovariateCorr` is used, during simulation, to set the generating correlation value among the simulated covariates. Even when more than 2 covariates are present, `CovariateCorr` should still be a single value - for simplicity, flexMIRT$^{\text{TM}}$ is will induce only an equicorrelation matrix among covariates.

Although not listed in the previous syntax summary, also specific to non-BAEM estimation is the `Beta(?,?)` matrix used in the `<Constraints>` section. This matrix, to be used in conjunction with Free, Fix, Equal, and Value statements, allows users to have detailed control of which covariates are applied to a given latent variable and impose constraints upon the estimated regression coefficients.

6.4. Examples Using MH-RM Estimation

All of the calibration examples presented previously in the manual are able to be estimated using MH-RM estimation. MH-RM estimation allows for `GOF = Extended;` and `GOF = Complete;` to be called, making some GOF indices available. When `GOF = Complete;` is used in conjunction with MH-RM estimation, the stochastic theta variant of the Yen-Bock item diagnostic X^2 values (e.g., Yen, 1981; Bock, 1960) is reported. However, requests for certain additional GOF output (the Haberman residuals table, JSI, M_2 statistic, etc.) will be ignored by flexMIRT$^{\text{TM}}$ because their behavior, in conjunction with the MH-RM algorithm, is still under research.

6.4.1 Confirmatory MIRT Model with Correlated Factors

For a first example, we will refit the confirmatory 4 correlated factors model applied to the 35-item QOL dataset, originally presented in Example 4-3.

Example 6-1: MH-RM: Confirmatory MIRT Model

```
<Project>
Title = "QOL Data";
Description= "Items 2-21,  4 Correlated Factors CFA: MH-RM";

<Options>
Mode = Calibration;
Algorithm=MHRM;
ProposalStd= 0.45;
Processors  = 4;
FactorLoadings  = Yes;

<Groups>
%Group1%
File ="QOL.DAT";
Varnames = v1-v35;
Select= v2-v21;
Dimensions= 4;
Ncats(v2-v21) = 7;
Model(v2-v21) = Graded(7);

<Constraints>
Fix(v2-v21),Slope;
Free(v2-v5),Slope(1);
Free(v6-v9),Slope(2);
Free(v10-v15),Slope(3);
Free(v16-v21),Slope(4);
Free Cov(2,1);
Free Cov(3,1);
Free Cov(3,2);
Free Cov(4,1);
Free Cov(4,2);
Free Cov(4,3);
```

In the `<Options>` section, we have specified that flexMIRT^{TM} should use the MH-RM algorithm via `Algorithm = MHRM;` and are opting to use the default random seed value to start the draws. In addition, we have set `ProposalStd = 0.45;`. This value was adjusted from the default of 1.0 because a first run of the algorithm was observed to have an acceptance rate lower than is desirable. As noted earlier, lowering the `ProposalStd` value will generally lead to higher acceptance rates. Outside of the `<Options>` section, no syntax was changed from the example presented in Chapter 4; the same model was fit by both the MH-RM and BAEM estimation routines.

Output 6.1: Correlated Four Factor Model Output - Processing Pane Output

```
MH-RM: Stage I
#    1; AR: 0.44,0.00; Fn:   -25561.90
#    2; AR: 0.43,0.00; Fn:   -25150.88
#    3; AR: 0.44,0.00; Fn:   -24761.94
#    4; AR: 0.41,0.00; Fn:   -24337.34
#    5; AR: 0.42,0.00; Fn:   -23975.69
#    6; AR: 0.41,0.00; Fn:   -23666.61
#    7; AR: 0.41,0.00; Fn:   -23338.13
#    8; AR: 0.41,0.00; Fn:   -23060.52
#    9; AR: 0.40,0.00; Fn:   -22792.52
#   10; AR: 0.41,0.00; Fn:   -22552.57
 ...
MH-RM: Stage II
#    1; AR: 0.20,0.00; Fn:   -19978.51
#    2; AR: 0.19,0.00; Fn:   -19932.53
#    3; AR: 0.20,0.00; Fn:   -19946.00
#    4; AR: 0.20,0.00; Fn:   -20066.33
#    5; AR: 0.21,0.00; Fn:   -20002.37
...
MH-RM: Stage III
#    1; AR: 0.20,0.00; Fn:   -19997.94; Gam: 0.50; W: 0; P#:  12; Chg:   0.1687
#    2; AR: 0.20,0.00; Fn:   -20017.97; Gam: 0.33; W: 0; P#: 127; Chg:   0.0631
#    3; AR: 0.21,0.00; Fn:   -20011.88; Gam: 0.31; W: 0; P#:  60; Chg:   0.0752
#    4; AR: 0.20,0.00; Fn:   -19975.84; Gam: 0.26; W: 0; P#:  41; Chg:   0.0583
#    5; AR: 0.19,0.00; Fn:   -20046.35; Gam: 0.24; W: 0; P#:  41; Chg:   0.0413
```

As noted previously, acceptance rates are printed in the processing pane for each iteration at all three stages of the MH-RM estimation. Given the complex structure of the model and the IRT model applied to the items, the desired goal for our fitted model is to have the reported acceptance rate have an average value between 0.2 and 0.3. Reading across the output, flexMIRT^{TM} is reporting the iteration number, the letters "AR" for acceptance rate, followed by the level-1 acceptance and then the level-2 acceptance rate. The current model had only 1 level, so all level-2 values will be 0.00. The value labeled Fn is an indicator that the program is or is not working towards a converged solution. While not the log likelihood, it may be interpreted in a similar fashion, meaning that less negative values indicate a better fit. Examining the iteration history of the analysis, the acceptance rate starts somewhat high (0.48) but drops down to the near desired level by the end of Stage I; during this time, the `Fn` value also rapidly drops initially and then stabilizes. For all of Stages II and III, both the acceptance rate and the function value fluctuate, with the acceptance rate staying within the desired value of 0.2-0.3. While the output should be closely examined for issues, this pattern of values in the processing pane is an early indicator that the MH-RM algorithm is working towards a good solution.

For bookkeeping purposes, the long-term acceptance rate will be reported in the -irt file with the other previously noted assessments of stability and convergence.

Output 6.2: Correlated Four Factor Model Output - Loadings, Means and Variances

```
Convergence and Numerical Stability
flexMIRT(R) engine status: Normal termination
Number of cycles completed:  1308
Maximum parameter change (P#):   0.00005380 (  138)
MH-RM latent trait sampler acceptance rate (L1,L2): (0.198,0.000)
First-order test: Convergence criteria satisfied
Condition number of information matrix: 867.5188
Second-order test: Solution is a possible local maximum
```

Below, we present the same section of the output was that given for Example 4-3.

Output 6.3: Correlated Four Factor Model Output - Loadings, Means and Variances

```
Factor Loadings for Group 1: Group1
 Item              Label lambda 1  s.e. lambda 2  s.e. lambda 3  s.e. lambda 4  s.e.
    1                 v2     0.83  0.03     0.00  ----     0.00  ----     0.00  ----
    2                 v3     0.78  0.03     0.00  ----     0.00  ----     0.00  ----
    3                 v4     0.92  0.01     0.00  ----     0.00  ----     0.00  ----
    4                 v5     0.92  0.01     0.00  ----     0.00  ----     0.00  ----
    5                 v6     0.00  ----     0.87  0.02     0.00  ----     0.00  ----
    6                 v7     0.00  ----     0.72  0.04     0.00  ----     0.00  ----
    7                 v8     0.00  ----     0.90  0.02     0.00  ----     0.00  ----
    8                 v9     0.00  ----     0.88  0.02     0.00  ----     0.00  ----
    9                v10     0.00  ----     0.00  ----     0.66  0.05     0.00  ----
```

```
   10              v11    0.00  ----     0.00  ----     0.74  0.04     0.00  ----
   11              v12    0.00  ----     0.00  ----     0.71  0.04     0.00  ----
   12              v13    0.00  ----     0.00  ----     0.71  0.04     0.00  ----
   13              v14    0.00  ----     0.00  ----     0.77  0.04     0.00  ----
   14              v15    0.00  ----     0.00  ----     0.72  0.04     0.00  ----
   15              v16    0.00  ----     0.00  ----     0.00  ----     0.77  0.04
   16              v17    0.00  ----     0.00  ----     0.00  ----     0.73  0.04
   17              v18    0.00  ----     0.00  ----     0.00  ----     0.76  0.04
   18              v19    0.00  ----     0.00  ----     0.00  ----     0.84  0.03
   19              v20    0.00  ----     0.00  ----     0.00  ----     0.74  0.04
   20              v21    0.00  ----     0.00  ----     0.00  ----     0.48  0.06

QOL Data
Items 2-21 Correlated Factors

Group Latent Variable Means:
   Group              Label   P#   mu  1    s.e.   P#   mu  2    s.e.   P#   mu  3    s.e.   P#   mu  4    s.e.
      1               Group1        0.00    ----         0.00    ----         0.00    ----         0.00    ----

Latent Variable Variance-Covariance Matrix for Group  1: Group1
   P# Theta  1    s.e.   P# Theta  2    s.e.   P# Theta  3    s.e.   P# Theta  4    s.e.
          1.00    ----
  141     0.39    0.03          1.00    ----
  142     0.53    0.03  143     0.48    0.03          1.00    ----
  144     0.53    0.03  145     0.54    0.03  146     0.67    0.02          1.00    ----
```

From the excerpt of the output provided, it may again be observed that the output from the MH-RM is formatted identically to output from BAEM estimations. Comparing across the two estimation routine results, the parameter estimates are nearly identical, indicating that the BAEM and MH-RM estimation routines are reaching a similar converged solution. In favor of the MH-RM algorithm, however, is the fact that the total time needed to estimate the model and generate output was around 4 minutes, compared to the 54 minutes used by the default BAEM.

6.4.2 Exploratory MIRT Model

EFAs are also able to be estimated via MH-RM estimation, making the high-dimensional nature of such models less of a hindrance, especially with respect to analysis completion time. Here we re-estimate the target rotation example from the previous chapter using MH-RM.

Example 6-2: MH-RM: EFA

```
<Project>
Title = "QOL Data";
Description= "Items 2-9, 2D EFA: MH-RM";

<Options>
Mode = Calibration;
Algorithm=MHRM;
```

```
RndSeed= 10;
ProposalStd= 0.35;
Processors  = 2;
NewThreadModel  = Yes;
FactorLoadings  = Yes;
 SavePCC  = Yes;

<Groups>
%Group1%
File ="QOL.DAT";
Varnames = v1-v35;
Select= v2-v9;
Ncats(v2-v9) = 7;
Model(v2-v9) = Graded(7);
BetaPriors(v2-v9) = 1.5;

Dimensions= 2;
Rotation  = Target;
Oblique =Yes;

UnspecifiedTargetElement =9;
Target = (
    9 0,
    9 0,
    9 0,
    9 0,
    0 9,
    0 9,
    0 9,
    0 9);

<Constraints>
```

As can be seen in the example MH-RM syntax, all commands for the EFA are identical to the previously presented example using BAEM estimation, with the exception of the MH-RM specific commands in the `<Options>` section. As in the previous example, we have set `Algorithm` to `MHRM` and adjusted the `ProposalStd` value to obtain a desirable acceptance rate. We have also chosen to specify a different random seed value, primarily to demonstrate the option.

Output 6.4: Correlated 2D EFA Model Output - Target-rotated Factor Loadings

```
Oblique Target Rotated Loadings for Group 1: Group1
 Item              Label Target lambda 1 Target lambda 2 Target lambda 3 Target lambda 4
    1                 v2      ?     0.85   0.00    -0.03   0.00    -0.04   0.00     0.04
    2                 v3      ?     0.77   0.00    -0.05   0.00    -0.06   0.00     0.01
    3                 v4      ?     0.94   0.00     0.03   0.00     0.05   0.00    -0.00
    4                 v5      ?     0.90   0.00     0.04   0.00     0.04   0.00    -0.05
    5                 v6   0.00    -0.00      ?     0.90   0.00    -0.05   0.00     0.08
    6                 v7   0.00    -0.00      ?     0.70   0.00    -0.03   0.00    -0.01
    7                 v8   0.00     0.01      ?     0.90   0.00     0.04   0.00    -0.03
    8                 v9   0.00    -0.01      ?     0.87   0.00     0.01   0.00    -0.04
    9                v10   0.00    -0.07   0.00    -0.05      ?    -0.81   0.00     0.05
   10                v11   0.00     0.01   0.00     0.04      ?    -0.72   0.00     0.02
   11                v12   0.00     0.01   0.00    -0.00      ?    -0.61   0.00    -0.09
   12                v13   0.00     0.04   0.00     0.04      ?    -0.64   0.00    -0.01
   13                v14   0.00     0.01   0.00     0.01      ?    -0.85   0.00     0.06
   14                v15   0.00     0.08   0.00     0.04      ?    -0.52   0.00    -0.17
   15                v16   0.00     0.01   0.00     0.07   0.00    -0.05      ?    -0.69
   16                v17   0.00    -0.01   0.00    -0.08   0.00    -0.12      ?    -0.69
   17                v18   0.00    -0.02   0.00    -0.01   0.00    -0.04      ?    -0.76
   18                v19   0.00    -0.04   0.00     0.02   0.00     0.10      ?    -0.95
   19                v20   0.00     0.06   0.00    -0.00   0.00     0.06      ?    -0.77
   20                v21   0.00     0.08   0.00     0.05   0.00    -0.13      ?    -0.31
```

A comparison of the output for the EFAs using target rotation finds, again, that the MH-RM solution is highly similar to the rotated solution obtained from BAEM estimation. However, it is not expected that the unrotated loadings will be exactly the same across the BAEM and MH-RM estimation and minor perturbations of the unrotated estimates can results in somewhat different rotated solutions. A detailed comparison of the factor patterns and factor correlations across the two solutions finds that the factors labeled “lambda 1” and “lambda 2”, across the different estimation methods, are actually reflections of each other. This can be seen in the similar magnitude of the factor loadings and factor correlations but reversed signs across the outputs (e.g., item v1 in the BAEM output has a lambda 1 rotated loading of -0.86, while in the MH-RM solution it has a rotated loading of 0.85). Regardless of which output is used though, the decisions regarding which items should compose the individual factors in a confirmatory model would be exactly the same. Even as the rotated solutions are comparable across the two estimation methods, note that the MH-RM estimation is much more efficient. A similar analysis using the four factors included in the other EFA and CFA examples with this data took a total of 90 seconds in MH-RM, compared to well over an hour needed for the BAEM estimation to complete.

6.4.3 Multilevel Model

With both confirmatory and exploratory factor analysis examples, we have demonstrated the time benefit that can be obtained through the use of the flexMIRT^{TM} MH-RM algorithm when estimating high dimensional models. The MH-RM as implemented in flexMIRT^{TM} is also able to fit multilevel models, a feature that, to our knowledge, is not available in any other commercially available software. We will provide syntax for and output from fitting such a model. The data used in this example was simulated to have six dichotomous items with responses from 2000 individuals- these 2000 individuals are nested within 100 higher-order units, 20 per unit. This type of structure could be seen if 2000 students from 100 different schools took a brief assessment in which items were graded as right/wrong.

Example 6-3: MH-RM: Multilevel MIRT Model

```
<Project>
Title = "Fit Two-level MIRT Model to Simulated Data";
Description=  "6 Items, 100 L2 units, 20 respondents within each";

<Options>
Mode = Calibration;
Algorithm=MHRM;
ProposalStd= 1.0;
ProposalStd2= 1.0;
Processors  = 2;
MCsize = 10000;

<Groups>
%Gr%
File =File = "simL2.dat";
Varnames = v1-v6,l2id;
Select=  v1-v6;
Cluster = l2id;
Dimensions= 4;
Between= 2;
N = 2000;
Ncats(v1-v6) = 2;
Model(v1-v6) = Nominal(2);

<Constraints>
Fix(v1-v6),Slope; // fix all slopes to begin with
Free(v1-v3),Slope(1); // level-2 factor 1
Free(v4-v6),Slope(2); // level-2 factor 2
Free(v1-v3),Slope(3); // level-1 factor 1
Free(v4-v6),Slope(4); // level-1 factor 2
```

```

Equal Gr,(v1-v3),Slope(1):Gr,(v1-v3),Slope(3);// cross-level equality
Equal Gr,(v4-v6),Slope(2):Gr,(v4-v6),Slope(4);

Free Cov(1,1);
Free Cov(2,2);
Free Cov(2,1);
Free Cov(4,3);

```

As before, we have told flexMIRT^{TM} to use the MH-RM algorithm and provided a random number seed. In this example, we are using both the `ProposalStd` and `ProposalStd2` keywords because we now have two levels to our model. The adjustments to the `ProposalStd` will affect the level-1 acceptance rate and adjustments to the `ProposalStd2` value will affect the level-2 acceptance rates. In the `<Groups>` section we inform flexMIRT^{TM} we will be fitting a model with 4 latent dimensions (`Dimensions = 4;`), two of which will be used as higher-level "between" factors (`Between = 2;`), and we then indicate the variable that will supply the level-2 group membership (`Cluster = l2id;`). Within the `<Constraints>` section, we create a structure that has 2 "between" and 2 "within" factors (via the first group of `Fix` and `Free` statements) and assign items to factors as desired. We also constrain the slopes across the levels to equality using `Equal` statements and then allow the variances of the level-2 factors and the covariance of the within factors and the covariance of the between factors to be freely estimated. It is worth noting here that there is nothing outside of the `<Options>` section that is uniquely related to the requested MH-RM estimation - this specific model can be estimated using the default BAEM, albeit with noticeable time loss. Larger models with more factors would most likely require MH-RM to complete successfully.

We will cover the processing pane output prior to addressing the results of the estimation. In particular, we will highlight the differences between the previous example and the processing pane output for a model with a second level.

Output 6.5: Two-Level MIRT Model - Processing Pane Output

```
MH-RM: Stage I
#    1; AR: 0.46,0.66; Fn:   -12385.84
#    2; AR: 0.47,0.65; Fn:   -12167.32
#    3; AR: 0.47,0.67; Fn:   -12072.87
#    4; AR: 0.47,0.61; Fn:   -12053.52
#    5; AR: 0.47,0.64; Fn:   -11981.70
#    6; AR: 0.47,0.63; Fn:   -11900.24
#    7; AR: 0.47,0.66; Fn:   -11823.63
#    8; AR: 0.47,0.61; Fn:   -11758.62
#    9; AR: 0.47,0.61; Fn:   -11684.70
#   10; AR: 0.47,0.65; Fn:   -11661.23
#   11; AR: 0.47,0.61; Fn:   -11630.73
...
MH-RM: Stage II
#    1; AR: 0.41,0.56; Fn:   -10856.40
#    2; AR: 0.41,0.54; Fn:   -10979.40
#    3; AR: 0.41,0.57; Fn:   -10942.47
#    4; AR: 0.41,0.56; Fn:   -10906.11
#    5; AR: 0.41,0.57; Fn:   -10924.03
...
MH-RM: Stage III
#    1; AR: 0.41,0.58; Fn:   -10949.66; Gam: 0.50; W: 0; P#:  12; Chg:   0.0109
#    2; AR: 0.41,0.53; Fn:   -10909.03; Gam: 0.33; W: 0; P#:  12; Chg:   0.0124
#    3; AR: 0.41,0.55; Fn:   -10931.30; Gam: 0.31; W: 0; P#:   7; Chg:   0.0156
#    4; AR: 0.41,0.59; Fn:   -10941.05; Gam: 0.25; W: 0; P#:  13; Chg:   0.0120
```

While the general structure of the reported values is the same as the previous processing pane, rather than "0.00" being consistently repeated as the second value after the `AR:` as in the first example, there is now a value that changes with each iteration - this is the acceptance rate for the second level. To be concrete, in Iteration 1 of Stage I, the reported level-1 acceptance rate is 0.47 and the level-2 acceptance rate is 0.63. As noted before, the desired range of acceptance rates is between 0.20 and 0.30 for large or complex models, with values around 0.50 being acceptable for smaller, less dimensionally complex models. Output 5.5 presents the parameters for the estimated model. It may again be observed that the item and group parameter output from the MH-RM is formatted identically to output from BAEM estimation. Additionally, it appears that the estimation completed successfully, resulting in reasonable item and group parameter values and SEs.

Output 6.6: Multilevel MIRT Model - Item and Group Parameters

```
Fit Two-level MIRT Model to Simulated Data
6 Items, 100 L2 units, 20 respondents within each

2PL Items for Group 1: Gr
    Item               Label   P#    a  1    s.e.   P#    a  2    s.e.   P#    a  3    s.e.   P#    a  4    s.e.   P#       c    s.e.
       1                  v1    7    1.60    0.10         0.00    ----    7    1.60    0.10         0.00    ----    1   -0.96   0.08
       2                  v2    8    1.86    0.11         0.00    ----    8    1.86    0.11         0.00    ----    2   -0.45   0.07
       3                  v3    9    0.98    0.06         0.00    ----    9    0.98    0.06         0.00    ----    3    0.72   0.06
       4                  v4         0.00    ----   10    1.78    0.13         0.00    ----   10    1.78    0.13    4    1.74   0.11
       5                  v5         0.00    ----   11    1.62    0.12         0.00    ----   11    1.62    0.12    5   -1.72   0.10
       6                  v6         0.00    ----   12    0.88    0.06         0.00    ----   12    0.88    0.06    6   -0.30   0.05

Fit Two-level MIRT Model to Simulated Data
6 Items, 100 L2 units, 20 respondents within each

Group Latent Variable Means:
   Group               Label   P#   mu  1    s.e.   P#   mu  2    s.e.   P#   mu  3    s.e.   P#   mu  4    s.e.
       1                  Gr         0.00    ----         0.00    ----         0.00    ----         0.00    ----

Latent Variable Variance-Covariance Matrix for Group  1: Gr
   P# Theta  1    s.e.   P# Theta  2    s.e.   P# Theta  3    s.e.   P# Theta  4    s.e.
   13     0.72    0.12
   15     0.46    0.10   14     0.87    0.14
          0.00    ----          0.00    ----          1.00    ----
          0.00    ----          0.00    ----   16     0.49    0.02          1.00    ----
```

6.4.4 Multilevel Bifactor Model

The multilevel capabilities of flexMIRT^{TM} may be applied to models with factor structures beyond the “Between” and “Within” factors used in the previous example. In the following example, we demonstrate MH-RM estimation of a multilevel IRT model with a bifactor structure applied to the Level-1/“Within” portion of the model. The data used in this example was simulated to have six dichotomous items with responses from 2000 individuals- these 2000 individuals are nested within 100 higher-order units, 20 per unit. The structure could be seen if 2000 students from 100 different schools took a brief assessment in which items were graded as right/wrong and the content of the assessment was intended to measure general math ability as well specific math subtopics, such as multiplication and fractions.

Example 6-4: MH-RM: Multilevel Bifactor

```
<Project>
Title =  "Two-level Bifactor";
Description= "6 Items, 100 L2 units, 20 subjects within each";

<Options>
Mode = Calibration;
Algorithm=MHRM;
ProposalStd= 1.0;
ProposalStd2= 2.0;
Processors  = 4;
MCsize = 500;
Rndseed = 874;
SavePRM  = Yes;
SaveMCO = Yes;

<Groups>
%Group1%
File = "L2bifacsim.dat";
Varnames = v1-v6, l2id;
Select = v1-v6;
Cluster = l2id;
Dimensions= 4;
Between = 1;
```

```
N  = 2000;
Ncats(v1-v6) = 2;
Model(v1-v6) = Graded(2);

<Constraints>
Fix(v1-v6),Slope;  // fix all slopes to begin with
Free(v1-v6),Slope(1);  // between factor
Free(v1-v6),Slope(2);  // within-general
Free(v1-v3),Slope(3);  // within-specific 1
Free(v4-v6),Slope(4);  // within-specific 2
```

The syntax for this multilevel bifactor model is constructed using “building blocks” that have been presented in previous examples. To account for the multilevel aspect of the data, we have specified `Cluster = l2id;` to provide flexMIRT^{TM} the variable that contains Level-2 group information as well as setting `Between = 1;`, telling flexMIRT^{TM} a multilevel model is intended and to initiate the multilevel estimation routine. For the bifactor portion of the model, as with previous bifactor examples, we have set `Dimensions` to the appropriate number (4 total = 1 Between factor, 3 Within factors - of which 1 is a general factor and 2 are specific) and then assigned items to factors, based on the Between and Within portions of the model and the content-related bifactor structure, in the `<Constraints>` section.

6.4.5 Fixed Effects Calibration

flexMIRT^{TM} has the ability to perform fixed effects (aka fixed theta) calibration, in which the user supplies individual theta values, which are fixed, to be used in calibrating items. This mode of calibration is potentially useful in field-testing experimental items. Fixed effects calibration is only available in conjunction with MH-RM estimation. The keyword to induce flexMIRT^{TM} to perform fixed effects calibration is `FixedTheta`. `FixedTheta` is a group-specific command that is used to set the variable(s) that contain(s) the known, individual theta values for that group in the calibration. By implementing fixed effects calibration in flexMIRT^{TM} as group-specific, it is possible to construct a calibration in which one group is based on fixed theta values and the other group is based on random, estimated thetas.

Example 6-5: MH-RM: Fixed Effects Calibration

```
<Project>
Title =  "LSAT 6 -  Fixed Effects Calibration";
Description = "5 Items 1PL N=1000";

<Options>
Mode = Calibration;
Algorithm= MHRM;
GOF= Extended;
Stage1 = 0;
Stage2 = 0;

<Groups>
%Group1%
File ="lsat6.dat";
Varnames = v1-v5, id, theta;
Select= v1-v5;
N = 1000;
Ncats(v1-v5) = 2;
Model(v1-v5) = Graded(2);
FixedTheta = theta;

<Constraints>
Equal(v1-v5), Slope;
```

For any group that will be calibrated with fixed ability values, theta is read-in as part of the datafile for that group. In this example, the theta values for `%Group1%` are in the variable *theta*, the last variable listed in the `Varnames` statement. As in earlier examples, because we have provided variables that are not to be calibrated, we then use a `Select` statement to specify only those items which will be subjected to IRT analysis. To provide flexMIRT^{TM} with the theta values, we use the statement `FixedTheta = theta;` to tell the program the variable that holds the fixed theta values. Although not seen in this example, `FixedTheta` works under multidimensionality - currently, the number of variables listed in `FixedTheta =` *`vars`* `;` must match the number of factors specified via `Dimensions` for that group.

Of final note in the fixed effects calibration syntax, the `Stage1` and `Stage2` keywords in the `<Options>` section, which determine the number of constant gain and stochastic EM, constant gain cycles of the MH-RM estimation method, are both set to 0. This is recommended for a majority of fixed effects calibrations in flexMIRT^{TM} because these cycles do not contribute much in the fixed effects case and eliminating them improves the time-efficiency of the estimation. However, if flexMIRT^{TM} is unable to reached a converged and stable solution, `Stage1` and `Stage2` may be set to non-zero values to help improve the starting values at Stage 3. Additionally, users may find it necessary to adjust the MH-RM step values from their defaults when fitting the 3PL item model with fixed effects calibration - the guessing parameter of this model generally requires a different step length (adjusted with the `Alpha` and `InitGain` statements) than other parameters and maintaining the default values will often result in "Feasible Set Violation" warnings in the flexMIRT^{TM} processing pane and an unconverged or unstable solution.

When `FixedTheta` is specified and the program can successfully run , flexMIRT^{TM} will print in the main output a notification that fixed effects calibration was performed, prior to reporting the item parameter estimates for the group. Such a notification is presented in the example program output. Also with regard to the output, please note that for a single-group fixed effects analysis or a fixed effects analysis that applies to all the groups in a multiple-group run, the SE of the log-likelihood approximation will be reported as 0.

Output 6.7: MH-RM: Fixed Effects Calibration Output - Item Parameters

```
*** Fixed effects calibration in Group 1: Group1

LSAT 6 -  Fixed Effects Calibration
5 Items 1PL N=1000

2PL Items for Group 1: Group1
    Item             Label   P#       a    s.e.   P#       c    s.e.       b    s.e.
       1                v1    6    2.96    0.08    1    3.59    0.14   -1.21    0.06
       2                v2    6    2.96    0.08    2    1.35    0.08   -0.45    0.03
       3                v3    6    2.96    0.08    3    0.33    0.08   -0.11    0.03
       4                v4    6    2.96    0.08    4    1.76    0.09   -0.59    0.03
       5                v5    6    2.96    0.08    5    2.79    0.11   -0.94    0.05
```

6.4.6 Crossed Random Effects Models

When using non-BAEM estimation methods, flexMIRT^{TM} also has the ability to fit crossed random effects models (e.g., de Boeck, 2008; Van den Noortgate, De Boeck, & Meulders, 2003). These models are useful in several different settings, including when users wish to estimate random, rather than fixed, item effects, in addition to the typical random effect that is used for observations/abilities.

Example 6-6: MH-RM: Crossed Random Effects Calibration

```
<Project>
Title =  "Two-level CFA";
Description = "Crossed Random Effect";
<Options>
Mode = Calibration;
Algorithm=MHRM;
RndSeed = 10;
Processors = 2;
ProposalStd= 2.0;
ProposalStd2 = 2.0;
Stage1 = 2000;
Stage2 = 100;
MCSize = 1;
SaveMCO = Yes;
SaveSCO = Yes;
Score = EAP;
SavePRM = Yes;

<Groups>
%Gr%
File ="simL2crossed.dat";
Varnames = v1, indiv_ID, item_fam, obs_cnt, theta, delta;
Select= v1;
Cluster = indiv_ID;
Block = item_fam;
Crossed = Yes;
Ncats(v1) = 2;
Model(v1) = Graded(2);
Dimensions = 2;
Between= 1;

<Constraints>
Value(v1), Intercept, 0.5;
```

An examination of the datafile used for this example would find that it is formatted such that each individual has multiple rows of data (i.e., the data is in “long” format). To understand the data and syntax, it may also help users to consider the alternate “wide” format, in which rows are individuals and each column represents an item - in reformatting the data to the long format, we could have named the variable “indiv_ID” as *row* and the “item_fam” variable could have been named *column*, so that the variable names and the values reflect the data position in the wide format (e.g., row = 4 and column = 2 would be individual 4’s response to the second item). The `<Options>` section has been optimized for this particular analysis (`ProposalStd` and `ProposalStd2` adjusted to obtain desirable acceptance rates, etc.) but the particulars specific to the crossed random effects model are found in the `<Groups>` section.

To ensure the data are interpreted correctly, we include the `Cluster` statement to let flexMIRT^{TM} know these are not independent observations and that observations with the same indiv_ID/row variable value belong to the same individual. The `Crossed = Yes;` statement is used to tell flexMIRT^{TM} that a crossed random effect model is desired and the `Block` statement supplies flexMIRT^{TM} with the variable that that is crossed with individuals, rather than nested within individuals. In this case, `Block = item_fam;` is used to tell flexMIRT^{TM} that a random effect should be used for the item_fam/column variable. Because a crossed random effects model is, by definition, a multilevel model, we have specified two total dimensions and stated that one of the two dimensions is to be treated as a level-2 dimension (`Between = 1;`), which, as discussed in previous multilevel examples, means that the first factor will be assigned as the “between” factor by the program. Output for crossed random effects model analyses is in the typical -irt format, discussed in previous examples.

6.5. Examples Using MCMC Estimation

With the exception of the previous fixed effect calibration run, all of the calibration examples presented previously in the manual, going back even to the first chapter, are able to be estimated via MCMC estimation. Users should be aware when selecting an estimation method that all requests for GOF output will be ignored by flexMIRT^{TM} when using `Algorithm= MCMC`.

For the first MCMC example, we will use the multilevel, bifactor model estimated with MH-RM earlier in the chapter. The majority of the syntax (all of the <Groups> and <Constraints> sections) are the same as in the previous syntax, so we will focus on only the MCMC-specific syntax.

6.5.1 Multilevel Bifactor Model

Example 6-7: MCMC: Multilevel Bifactor (excerpt)

```
<Project>
Title =  "Two-level Bifactor";
Description= "6 Items, 100 L2 units, 20 subjects within each";

<Options>
Mode = Calibration;
Algorithm=MCMC;
BurnIn = 500;
Thinning = 10;
MaxCycle = 1000;

ProposalStd= 1.0;
ProposalStd2= 2.0;
ItemProposalStd= 0.05;

Processors = 4;
Rndseed = 874;
SavePRM  = Yes;
SaveMCO = Yes;
```

The first point of note is that we have specified `Algorithm = MCMC;` in the second line of the `<Options>`. Given the complexity of the model, we have increased the length of the burn-in from the default value, decreased the thinning interval to 10, and have set the maximum number of MCMC cycles to 1000 (greater than the default of 500). Futher down, we find that the `ProposalStd` and `ProposalStd2` values, optimized in the previous MH-RM run of this analysis, have the same values as were used during the MH-RM estimation. We also find the MCMC-specific command of `ItemProposalStd` has been changed from the default value of 0.1 to 0.05.

6.5.2 Models with Covariates

Our final example for the Alternate Estimation Methods chapter presents an analysis with covariates. **The inclusion of covariates is only available when calibrating models with non-BAEM algorithms**. Note that while the following example is multilevel, there is no restriction on the models that may be used with covariates - any item model, unidimensional, multiple group, and multidimensional models, etc. are all acceptable for use when covariates are to be included. In the full syntax (available on the flexMIRT^{TM} support page) it can be seen that MCMC estimation was used. Covariates may also be used in conjunction with MH-RM estimation. **User's should be aware that flexMIRT^{TM} expects that covariates will not have missing values - users should address missingness in all covariate variables (e.g., multiple imputation of missing values) during data processing in their preferred general statistical software. If not resolved prior to flexMIRT^{TM} analysis, missing values in a covariate variable (e.g., -9) will be treated as usable/correct covariate values and will negatively affect the accuracy of estimates**.

Example 6-8: MCMC: Multilevel Model with Covariates (excerpt)

```
<Groups>
%G%
File ="simL2.g1.dat";
Varnames =  v1-v12, l2id, l1id, theta1-theta6, l2cov, l1cov;
CaseID = l1id;
Cluster =  l2id;

N v1-v12;
Ncats(v1-v5) = 2;
Model(v1-v5) = Graded(2);

Dimensions = 2;
Between = 1;

Covariates = l2cov, l1cov;
L2covariates= 1;

<Constraints>
```

```

Equal G, (v1-v12), Slope(1): G, (v1-v12),  Slope(2);
Free G,  Cov(1,1);

Free Beta(1,1);
Free Beta(2,2);
```

The inclusion of covariates takes place in the `<Groups>` and `<Constraints>` sections of the syntax. When covariates are specified, they are group-specific; our current example has only one group, but multiple-group runs may include some covariates in one group and more or fewer or no covariates in another. Further, it is not necessary that the same covariates be used across all groups.

Within the `<Groups>` we can see that the data file and variable names have been supplied as usual. We have provided the program with an individual ID variable via the `CaseID` statement and a level-2 unit ID with `Cluster` as in the previous multilevel example. We select variables and specify item models in the typical fashion and set up a unidimensional-at-each-level multilevel model with the `Dimensions` and `Between` statements. Finally, we find the `Covariates` statement which is used to select variables that will be used as predictors of the latent traits. All desired covariates are listed in the `Covariates` statement separated by commas. To specify that we have a covariate which will be predicting the Between (rather than the Within factor) of our multilevel model we set `L2covariates` to a non-zero value. flexMIRT^{TM} will scan through the list of variables in the `Covariates` and select out the first x (the value set in `L2covariates`) variables and treat them as covariates of the higher-level factor(s).

In the `<Constraints>` section, we set which covariates are predicting which factors by freeing individual elements of the Beta matrix, which collects all the regression coefficients of the latent variables on covariates. The Beta matrix will have the dimensions of (number of latent variables X number of covariates), with the columns representing the covariates in the order in which they were supplied in the `Covariates` statement. For our current example, we want the variable "l2cov" to predict our Between (level-2) factor and the variable "l1cov" to predict the Within (level-1) factor of our model. The Beta matrix, with freely estimated elements represented by 1s, would be:
In our syntax, we have freed the element `Beta(1,1)` to regress the Between latent variable on the first covariate from the `Covariate` statement (which we

Table 6.1: Beta matrix for group G

Latent Variable	Covariate1 (l2cov)	Covariate 2 (l1cov)
theta1 (Between)	1	0
theta2 (Within)	0	1

defined as a level-2 covariate in the `<Groups>` section) and we have freed the element `Beta(2,2)`, which allows the second covariate listed in the `Covariate` statement to predict the Within latent variable.

The output for an analysis including covariates is very similar to the output from previous examples. As seen in the provided excerpt, flexMIRT^{TM} has included additional information regarding the covariates in the summary printed at the beginning of the *-irt.txt output file. In addition to the typical item and group parameters, we now have a section of output labeled "Latent Regression Coefficients" that reports the estimated regression coefficients of the latent variables predicted by the covariates.

```
Summary of the Data and Dimensions
   Missing data code        -9
     Number of Items        12
 Number of L-2 units       100
 Number of L-1 units      1000
 # Latent Dimensions         2
  Between Dimensions         1
   Within Dimensions         1
Number of Covariates         2
  Level-2 Covariates         1
...

Latent Regression Coefficients:

   Group  1: G
                   Covariate  1          Covariate  2
             P#    Beta    s.e.     P#    Beta    s.e.
Theta  1     26    1.40    0.11           0.00    ----
Theta  2           0.00    ----     27    0.44    0.04
```

CHAPTER 7

Simulation

Simulation studies are common in the IRT literature. Compared to other commercially available IRT software programs (e.g., Bilog, IRTPro, Multilog, and Parscale), flexMIRT^{TM} is unique in that it facilitates simulations by offering the user the capability of generating data from all implemented IRT models, including multidimensional, bifactor, and multilevel models. The command syntax for generating data has the same basic structure as the model-fitting syntax. In general, it will typically involve a relatively small amount of code in the syntax file, leaving the majority of the model settings and generating parameter values in the parameter file. Alternatively, one can fully specify the models and generating parameter values directly in the syntax, without using the parameter file, or using it only as a supplement. Note that a parameter file saved from a calibration run is compatible for use in simulations. This provides a convenient mechanism so that the user may conduct simulations with generating parameters found from empirical data analysis.

7.1. A Syntax-Only Example Using Value Statement

In this example, we will use the syntax file to specify simulation of a data set consisting of 1000 simulees' responses to 4 graded items, each scored with 3 categories. The generating parameter values are specified in the `<Constraints>` section. In simulation mode, the `Value` statement can be used to specify generating parameters, as is done in the following example.

Example 7-1: Simulation - Graded Model Syntax Only

```
<Project>
Title = "Simulate Data";
Description= "4 Items Graded Syntax Only";

<Options>
Mode = Simulation;
Rndseed = 7474;

<Groups>
%G%
File ="sim1b.dat";
Varnames = v1-v4;
N = 1000;
Model(v1-v42) = Graded(3);

<Constraints>
Value(v1),Slope,0.7;
Value(v2),Slope,1.0;
Value(v3),Slope,1.2;
Value(v4),Slope,1.5;
Value(v1,v2),Intercept(1),2.0;
Value(v1,v2),Intercept(2),1.0;
Value(v3,v4),Intercept(1),0.0;
Value(v3,v4),Intercept(2),-1.0;
```

In the `<Options>` section, flexMIRT^{TM} is set to run in Simulation mode. When this is case, a random number seed must be supplied via the `Rndseed` statement. The group specification remains the same as in calibration runs. In this case, the `File` statement instructs flexMIRT^{TM} to save the simulated item responses to “sim1a.dat”. The variables are named and the total number of simulees is set to 1000. The `Model` statement sets the generating IRT model to the graded response model with 3 categories. Next, in the `<Constraints>` section, the generating parameter values are set. For instance, the first constraint says that the value of the slope parameter for item v1 should be set to 0.7. The 5th constraint specifies that the first intercept term for item v1 and v2 should be equal to 2.0. In a single group, the group name can be omitted, so

`Value (v1), Slope,0.7;` and `Value G,(v1), Slope,0.7;` would have the same effect. For multiple group simulations, the group name must be specified in each `Value` statement.

Output 7.1: Simulation Control Output - Graded Model Syntax Only

```
...
Summary of the Data and Dimensions
   Missing data code        -9
     Number of Items         4
     Number of Cases      1000
 # Latent Dimensions         1

Item  Categories       Model
   1           3      Graded
   2           3      Graded
   3           3      Graded
   4           3      Graded

Miscellaneous Control Values
Random number seed:        7474
...
Graded Items for Group 1: G
    Item       a     c 1     c 2
       1    0.70    2.00    1.00
       2    1.00    2.00    1.00
       3    1.20    0.00   -1.00
       4    1.50    0.00   -1.00

Graded Items for Group 1: G
    Item       a     b 1     b 2
       1    0.70   -2.86   -1.43
       2    1.00   -2.00   -1.00
       3    1.20   -0.00    0.83
       4    1.50   -0.00    0.67
...
Group Parameter Estimates:
   Group              Label      mu      s2      sd
       1                  G    0.00    1.00    1.00
```

The first part of the simulation control output echoes the user-supplied specifications, (e.g. number of items, number of simulees, the number of categories, IRT model types, and the random number seed). The generating item parameters are then printed. One can see that the syntax statements are correctly interpreted. When everything is set up correctly, a simulated data set is generated and saved. The first and last three rows of this data set are shown below. The first 4 columns are the item responses (in 3 categories coded as 0, 1, and 2). Column 5 is a case ID column. Column 6 contains the generating theta value for that simulee.

Output 7.2: Simulated Data Set - Graded Model Syntax Only

```
2 2 0 2 0 1.69411
0 0 0 0 1 -1.33926
2 2 0 0 2 -0.0399422
...
2 0 0 0 997 0.0584387
2 2 0 1 998 -0.672594
2 2 0 2 999 -0.203111
```

As noted previously, simulations may either be conducted by using `Value` statements to provide generating parameter values or through the use of a parameter file. To demonstrate this point, we will now generate data with the parameters values just used, but supply them to flexMIRT^{TM} via a parameter file.

7.2. The First Example Redone Using Parameter File

Example 7-2: Simulation - Graded Model Using PRM File

```
<Project>
Title = "Simulate Data";
Description= "4 Items Graded Use PRM File";

<Options>
Mode = Simulation;
Rndseed = 7474;
ReadPRMFile = "sim1b-prm.txt";

<Groups>
%G%
File ="sim1b.dat";
```

```
Varnames =  v1-v4;
N = 1000;

<Constraints>
```

The ReadPrmFile statement is invoked to read the parameter file. The group specification only contains the file name, the variable names, and the number of simulees. Note that the <Constraints> section is empty because the values of the generating parameters are set in the parameter file. The contents of parameter file are shown in Table 6.1.

Table 7.1: Simulation - Graded Model PRM File Content

1	v2	1	1	2	3	2	1	1.0
1	v3	1	1	2	3	0	-1	1.2
1	v1	1	1	2	3	2	1	0.7
1	v4	1	1	2	3	0	-1	1.5

The parameter file has 4 rows, corresponding to the 4 items. Take row 1 as an example. The entry in the first column of 1 specifies that this is a row containing item parameters. This is a predefined option. The other option is 0, which corresponds to group specifications. The second column specifies the variable name. Note that the items are not ordered as v1 to v4, but there is no problem since the items are checked against the list of variable names in the syntax file and matched automatically. Column 3 indicates that this item belongs to group 1. Column 4 indicates the number of factors - in this case, the model is unidimensional. The next two columns, taken together, set the IRT model to graded (type 2 in column 5) with 3 categories (column 6). Then the item intercepts (from the intercept corresponding to the lowest category to the highest category) and the item slope are specified for each item. The resulting simulated data set is exactly the same as the previous example.

Output 7.3: Simulated Data Set - Graded Model Using PRM File

```
2 2 0 2 0 1.69411
0 0 0 0 1 -1.33926
2 2 0 0 2 -0.0399422
...
2 0 0 0 997 0.0584387
2 2 0 1 998 -0.672594
2 2 0 2 999 -0.203111
```

7.3. The Parameter File Layout

Although the progression of columns of the parameter file must proceed in a prescribed fashion, the general format of the file is plain ASCII, in which delimiters may be spaces or tabs. Column headers are not permitted, which is why the order of values in the columns must follow a predetermined progression. For arranging the values in the parameter file, we have found it easiest to use a spreadsheet program (e.g., Excel) to initially set up the entries and, when complete, export/paste the values into the flat ASCII text editor. As noted earlier, there are set options from which the user must select for certain columns. Table 7.2 provides a summary of these options.

Table 7.2: Key Codes for Columns With Set Options

Column 1 (Entry Type)		Column 5 - Items (IRT Model)		Column 5 - Groups (Prior Type)		Select Columns (Nominal Contrast Type)	
Value	Meaning	Value	Meaning	Value	Meaning	Value	Meaning
0	Group	1	3PL	0	Normal Prior	0	Trend
1	Item	2	Graded	1	EH/KDE Prior	1	Identity
		3	Nominal	2	DCM probs	2	User-specified

As indicated in Table 7.2, the first column of the parameter file indicates whether the information to be supplied across the row refers to an item or a group. The information that goes into the various subsequent columns differs somewhat, depending on whether item or group is indicated in column 1, so the arrangement for these two parts of the parameter file will be discussed separately.

7.3.1 Column Progression for Groups

With a `0` in the first column of the parameter file, flexMIRT$^{\text{TM}}$ will interpret all subsequent information on the line as applicable to a group. The 2nd column is used to supply a group label, which should correspond to the label provided in the syntax file. The third column indicates the group number for the given group, (i.e., 1 if it is the first group listed in the syntax file, 2 if it belongs to the second group, etc.). Column 4 requires the user to specify the number of latent variables in the generating model, specific to the group just specified. The 5th column is reserved for indicating the type of population distribution (prior) to be applied to the latent trait. The acceptable values are 0 for normal and 1 for empirical histogram (see Table 7.2). If a normal prior is chosen, the subsequent columns will supply the latent factor mean(s) and then (co)variance(s).

The latent factor mean values are entered in order, with the mean of factor 1 listed first, followed by mean 2, mean 3 and so on. For specifying (co)variance values, the unique elements of the latent factor variance/covariance matrix are entered in row-major order. To demonstrate this point, suppose the following 4×4 factor covariance matrix is to be submitted to flexMIRT$^{\text{TM}}$:

	f1	f2	f3	f4
f1	1	2	4	7
f2	2	3	5	8
f3	4	5	6	9
f4	7	8	9	10

The progression of variance/covariance values entered in the parameter file should be

```
1 2 3 4 5 6 7 8 9 10
```

If an empirical histogram prior is chosen, the user must supply additional information. Column 6 specifies the total number of quadrature points by which the histogram is defined and Column 7 requires a maximum value for the quadrature points, which determines the range over which the points are to be distributed. Similar to specifying `Quadrature` or `FisherInf` in the `<Options>` section, the quadrature points are spread symmetrically around zero, with the minimum and maximum value determined by the entry in Column 7. After the EH specification, the mean(s) and unique element(s) in the covariance matrix are listed in the subsequent columns, following the same progression of entries just covered. The histogram itself is supplied next. For instance, if the entry in Column 6 is `101` and Column 7 is `5`, then 101 points will be evenly spaced over the range of -5 to 5. The user must then supply, as the last 101 entries of this particular row, the 101 ordinates of the empirical histogram (i.e., the heights of the histogram at the 101 quadrature points). An example of simulating data with EH priors will be presented later.

The ordering of means and then (co)variance values is slightly changed when covariates are to be simulated as well. Following the mean values for all latent factors, the Beta matrix values (that is, the generating regression coefficients for the covariate(s)) are supplied in row-major order prior to the (co)variance value entries.

7.3.2 Column Progression for Items

The first four columns of a row supplying item level information have the same generic content as groups. First, a value indicating item information will follow (i.e., `1`) is given in Column 1, then a variable name is assigned in Column 2. This name must correspond to a variable in the `Varnames` statement in the command file. Column 3 assigns the item to a group number, and Column 4 provides the number of latent factors for that group. At this point (Column 5), the meaning of column entries differs from what was covered in the groups subsection. The generating IRT model for the item is set in Column 5, with models coded as indicated in Table 7.2. For instance, an item to be generated using graded model parameters would have a 2 in the 5th column. Column 6 supplies the number of possible response categories for the item.

Once an IRT model is chosen, the item parameters must be entered, one value per column. The progression of expected parameters varies slightly depending on the which IRT model was selected in Column 5. If the 3PL model was selected, the guessing parameter (in logit metric) is given in Column 7, followed by the intercept parameter, and then all discrimination/slope parameter values (if there is more than one factor). If the Graded model is selected in Column 5, the expected order for parameter values is the ordered intercept parameters (starting at the lowest category's in Column 7 and going up) followed by the discrimination parameter(s).

With respect to the Nominal model, the parameters are entered in the reparameterized form detailed in Thissen et al. (2010). The type of scoring contrast is given in Column 7, using one of the code values given in Table 7.2. If Trend (`0`) or Identity (`1`) contrasts are specified in Column 7, the scoring function contrast values (of which there are the number of categories minus one) must be supplied starting from Column 8. After those $m - 1$ values are given, the slope parameter(s) are entered. Immediately after the slope parameter, the next column (the column number of which will vary depending on the number of categories and slopes used) is used to indicate the contrast type for the intercept parameters. Again, if Trend or Identity contrasts are selected, the columns immediately following the contrast selection contain the appropriate number of intercept parameter values (again always equal to the number of categories minus one).

In the rare occasion that user-supplied contrasts are selected for either the scoring functions or the intercept parameters, it is necessary to supply the contrast coefficient values as well as the contrasts. In other words, the built-in contrasts are automatically computed, but any user-defined contrasts need to be entered into the program. The contrast coefficients are given after the contrast selection column (e.g., Column 7) but before the actual contrast values. The contrast coefficients of a user-supplied contrast matrix are expected to be in row major order. That is, for a contrast matrix applied to nominal items with k categories, the coefficients from that $k \times (k-1)$ matrix would be entered in the following order: $T(1,1), T(1,2), T(1,3), \ldots, T(1,k-1), T(2,1), \ldots, T(2,k-1), \ldots, T(k,k-1)$.

7.4. A 3PL Model Simulation Example

In this example, we will simulate item responses to 10 3PL items for a single group of 500 simulees. The latent trait distribution is standard normal.

Example 7-3: Simulation - 3PL Model Normal Population

```
<Project>
Title =  "Fit Model";
Description=  "10 Items 3PL N=500 Normal Priors";

<Options>
Mode = Simulation;
Rndseed = 1;
ReadPRMFile = "genparams.txt";
<Groups>
%Group1%
File = "sim2.dat";
Header = Yes;
Varnames = v1-v10;
N = 500;

<Constraints>
```

As always, the command file begins with the `<Project>` section where a title and general description are provided. In the `<Options>` section we have set the engine mode to `Simulation` and, as required when `Mode = Simulation;` provided a seed value for the random number generator which will be used to create the data. Because we are not using `Value` statements, we have also provided a file name where item and group parameters may be found.

In the `<Groups>` section, we label our group and use the `File` statement to provide a name for file where the data will be stored once generated. We want to simulate data for 10 variables so we provide labels for 10 variable names with the `Varnames` statement. We have also included the optional statement `Header = Yes;` which will write column labels/a header row into the datafile that is being created. Finally, using `N`, we specify the number of cases we want generated. The variable and group labels provided here must match those provided in the second column of the parameter file. As may be seen here, and as mentioned previously, the syntax command file is only used to set up the very basic structure of the analysis and the specifics are left to be input through the parameter file, to which we will now turn our attention.

As covered in Section 6.1, the actual parameter file to be submitted to flexMIRT^{TM} (Ex1genparms.txt, in this case) may not contain a header row/ column names but they are included in Table 7.3 for illustrative purposes only. In this parameter file, we have arranged all the item specifications first and the group parameters are in the bottom row. This is completely arbitrary and the rows of the file may be arranged in any way the user wishes. However, the columns must follow the arrangement previously detailed. Reading across for the first item, the `1` in the first column denotes that the values to follow refer to an item. We then provide a variable name, `v1`, and assign this item to the first group (which is the only group in this example). The next column indicates that there is 1 latent factor in the model and the subsequent column (Column 5) says that we will be providing parameters for the 3PL model (indicated by the `1`). The 6th column value tells flexMIRT^{TM} that the item will have two response options. Based on the number of factors, the IRT model, and the number of response categories, we now supply the appropriate number of item parameters, in this case 3 parameters (1 for the lower asymptote, 1 for the intercept, and 1 for the slope). The order of parameters values must be guessing (in the logit metric), intercept (in the $c = -a * b$ metric, where b is the difficulty/threshold), and then the slope parameter (a).

Table 7.3: Simulation - 3PL Parameter File with Labeled Columns

Type	Label	Group #	# Factors	Model	# Cat	logit g	c	a
1	v1	1	1	1	2	-1.34	-0.69	1.30
1	v2	1	1	1	2	-0.97	-1.30	1.41
1	v3	1	1	1	2	-1.64	0.42	1.05
1	v4	1	1	1	2	-2.10	0.40	1.26
1	v5	1	1	1	2	-1.59	-1.35	2.43
1	v6	1	1	1	2	-1.72	-0.56	1.67
1	v7	1	1	1	2	-1.13	-0.26	3.12
1	v8	1	1	1	2	-1.71	-1.56	2.09
1	v9	1	1	1	2	-2.79	0.73	1.17
1	v10	1	1	1	2	-1.66	-0.14	1.10
Type	Label	Group #	# Factors	Prior Type	Mean	Var		
0	Group1	1	1	0	0.00	1.00		

The final row of the file is used to describe the group. The first `0` indicates that the row is describing a group. The second column provides the group a label, `Group1`, which should correspond to the label provided in the command file. We assign the `Group1` label to the first group by the `1` in the 3rd column, and specify that a unidimensional model will be generated, by indicating 1 factor in the 4th column. As noted in Table 7.2, the fifth column of a Group line indicates the type of prior distribution to be applied (in this case a normal prior, signified by the `0`). After a normal prior is indicated, a group mean of `0.00` and a variance of `1.00` are supplied.

7.5. Simulating Data with Non-Normal Theta Distributions

In this example, we will cover data simulated from models with non-normal population distribution. This is accomplished using the empirical histograms method (previously used in Calibration Example 3-2). The empirical histogram prior may only be used for unidimensional or bifactor models (testlet response model included), as previously noted in the discussion of Example 3-2.

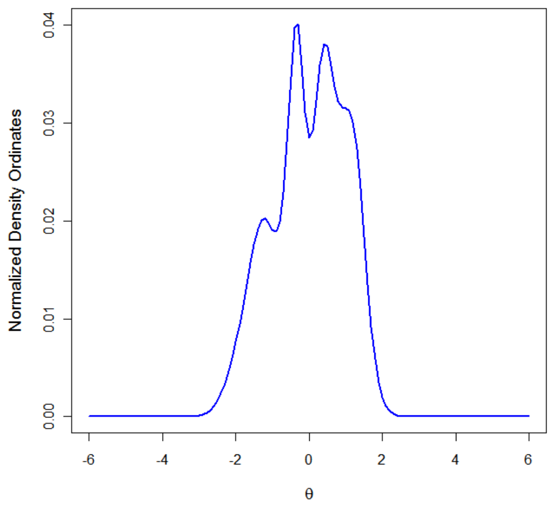
Example 7-4: Simulation - Empirical Histogram Prior

```
<Project>
Title = "Simulate Data";
Description= "10 Items 3PL N=500 Non-Normal Priors";

<Options>
Mode = Simulation;
Rndseed = 1;
ReadPRMFile = "EHgenparams.txt";
Quadrature = 121, 6.0;

<Groups>
%Group1%
File ="sim2.dat";
Varnames = v1-v10;
N = 500;
EmpHist = Yes;

<Constraints>
```

The command file for this example is substantively the same as the previous, although we have changed the parameter file name. The added feature here is that a non-normal distribution will be specified in the generating parameter file (EHgenparams.txt). This change only substantially effects that values entered into the Group row of the parameter file. However, because we are using the empirical histogram parameter file, we must also supply a `Quadrature` command in our syntax file that matches the number and range of points used to construct the empirical histogram.

An examination of the generating parameter file will find that the 5th column of the Group row been changed from the value of `0`, seen in Table 7.3, to a value of `1`. As noted in Table 7.2, this indicates that an empirical histogram prior will be used. The 6th column then provides the number of points to be used in constructing the histogram (`121`) and the next value indicates the range over which the points are spread (from -6 to 6 here, as indicated by the value of `6` in the 7th column of the Group row). Following the specification of the empirical histogram parameters, the latent variable mean(s) and covariances are listed. After all means and covariances are listed, the heights of the empirical histogram are provided. For our current example, we need to provide 121 values, per the entry in Column 6. Figure 7.1 provides a graphical display of the empirical histogram provided in the parameter file, to demonstrate the extent of non-normality.

Figure 7.1: Empirical Histogram From EHgenparams.txt



7.6. Simulating Bifactor Model Data

We will be using item parameters obtained from our bifactor analysis in Example 4-5 to simulate items with a bifactor structure from the 2PL model. As noted previously, when not employing `Value` statements, only the very basic structure of the analysis is supplied in the simulation syntax file and the details of the model are left to the parameter file. Table 7.4 presents a labeled version of the parameter file (QOLbifac-prm.txt) for illustrative purposes.

Example 7-5: Simulation - Bifactor Structure From Existing Parameter File

```
<Project>
Title  = "QOL Example";
Description=  "Simulation From PRM File 35 2PL Items Bifactor";

<Options>
Mode = Simulation;
ReadPRMfile=  "QOLbifac-prm.txt";
Rndseed = 7474;

<Groups>
%Group1%
File = "QOLbifacsim.DAT";
Varnames = v1- v35;
Dimensions = 8;
Primary = 1;
N = 1000;

<Constraints>
```

Per the previous discussion of the parameter file arrangement, the first column is for entering whether the values to follow on the row refer to an item or a group, with `1` denoting items and `0` denoting groups. We will now follow the first line across, discussing the columns in turn. The second supplies a variable name for the line and the third column specifies to which group the item belongs. In this case, there is only one group, so the entire column consists of `1`s. In Column 4, we specify that there are 8 factors in the model

Table 7.4: Labeled Parameter File for Bifactor Model Simulation

Type	Label	Grp#	#Fac	Model	#Cat	c	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
1	v1	1	8	2	2	2.13	2.26	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	v2	1	8	2	2	3.78	2.31	2.51	0.00	0.00	0.00	0.00	0.00	0.00
1	v3	1	8	2	2	1.69	1.62	1.42	0.00	0.00	0.00	0.00	0.00	0.00
1	v4	1	8	2	2	3.76	3.86	4.89	0.00	0.00	0.00	0.00	0.00	0.00
1	v5	1	8	2	2	2.87	2.77	3.08	0.00	0.00	0.00	0.00	0.00	0.00
1	v6	1	8	2	2	0.86	2.77	0.00	3.20	0.00	0.00	0.00	0.00	0.00
1	v7	1	8	2	2	1.38	1.38	0.00	1.50	0.00	0.00	0.00	0.00	0.00
1	v8	1	8	2	2	0.49	2.60	0.00	2.86	0.00	0.00	0.00	0.00	0.00
1	v9	1	8	2	2	0.30	1.78	0.00	1.97	0.00	0.00	0.00	0.00	0.00
1	v10	1	8	2	2	3.09	1.91	0.00	0.00	1.74	0.00	0.00	0.00	0.00
1	v11	1	8	2	2	2.57	1.42	0.00	0.00	0.71	0.00	0.00	0.00	0.00
1	v12	1	8	2	2	1.88	1.40	0.00	0.00	0.72	0.00	0.00	0.00	0.00
1	v13	1	8	2	2	2.30	1.57	0.00	0.00	0.13	0.00	0.00	0.00	0.00
1	v14	1	8	2	2	5.16	4.02	0.00	0.00	3.80	0.00	0.00	0.00	0.00
1	v15	1	8	2	2	2.09	2.49	0.00	0.00	0.66	0.00	0.00	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	v34	1	8	2	2	2.95	1.67	0.00	0.00	0.00	0.00	0.00	0.00	1.30
1	v35	1	8	2	2	3.22	1.40	0.00	0.00	0.00	0.00	0.00	0.00	0.44
Type	Label	Grp#	#Fac	Prior	μ_1	$\cdots$	μ_8	σ_{11}	σ_{21}	σ_{22}	σ_{31}	σ_{32}	σ_{33}	$\cdots$
0	Group1	1	8	0	0.00	$\cdots$	0.00	1.00	0.00	1.00	0.00	0.00	1.00	$\cdots$

and Columns 5 and 6, respectively, tell flexMIRT$^{\text{TM}}$ that the IRT model is the graded model (type 2) with two possible response categories. The combination of the these three values informs flexMIRT$^{\text{TM}}$ to expect 9 item parameters, one intercept and 8 slopes. Scanning down the columns of slope parameters, it becomes clear that the a_1 column is for the general factor, with all item loading onto it, and the next 7 columns represent the group-specific factors, onto which only a subset of items load.

Turning now to the group specification in the bottom row of the table, the first column signifies that this line is supplying group information, due the value of `0`. Labeling and assigning the group number and the number of factors in the model, all similar to what was done in the item rows, comes next. In Column 5, we enter information regarding the type of population distribution prior. We have chosen normal priors (via the option `0`). Because we have chosen normal priors, all that is left to specify are the latent variable means for each of the eight factors, followed by the unique elements of the covariance matrix.

7.7. Simulating Group Differences

As seen in the multiple group calibration examples, each group has its own self-contained subsection within the `<Groups>` section. While this can lead to some redundancy in the naming of variables, etc., it does all for a great deal of flexibility across groups. For example, flexMIRT^{TM} does not require that the number of items, the number of categories, the kind of IRT models employed, or the number of latent dimensions remains the same across groups. One may specify one group to have a normal prior distribution and use empirical histograms in the other group. In particular, one may even set up a multilevel model for one group and a single level model for the other.

The current example simulates group differences in both population means and variances as well as item parameters. Each group has 10 items. Group 1 has mean 0, variance 1. Group 2 has the same variance but a mean at 0.2. Group 3 has a mean of -0.2 and variance of 1.5. The items are for the most part 3PL items, with the exception of Item 1 in Group 3. The lower asymptote parameter is set to 0 for that item, making it a 2PL item. Also, the intercept parameter for Item 9 in Group 2 is chosen such that its threshold value is 0.5 standard deviation higher than the corresponding items in Groups 1 and 3.

Example 7-6: Simulation - DIF and Group Mean Differences

```
<Project>
Title =  "Simulate Data";
Description= "3 Groups 10 Items N=1000 (mostly) 3PLs
Group 1 N( 0.0,1.0), Group 2 N( 0.2,1.0), Group 3 N(-0.2,1.5)
Item 9 in Group 2 has 0.5 higher threshold than the
corresponding item in Groups 1 and 3.
Item 1 in Group 3 is a 2PL item.";

<Options>
Mode = Simulation;
Rndseed = 7474;
ReadPRMFile =  "genparams.txt";

<Groups>
%Group1%
File ="group1.dat";
Varnames = v1-v10;
N = 1000;
%Group2%
File ="group2.dat";
```

```
Varnames = v1-v10;
N = 1000;
%Group3%
File ="group3.dat";
Varnames = v1-v10;
N = 1000;

<Constraints>
```

The `<Options>` section remains substantially unchanged from previous examples. Within the `<Groups>` section, we now have labels and specifications for the three distinct groups for which we wish to generate data. Following the other examples, the details of the simulation are set up in the parameter file (genparams.txt). The contents of the parameter file (presented, in part, in Table 7.7) are now discussed.

Table 7.5: Simulating DIF and Group Mean Differences - Parameter Values

1	v1	1	1	1	2	-1.33972	-0.68782	1.301362
1	v2	1	1	1	2	-0.96814	-1.29641	1.409536
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	v9	1	1	1	2	-2.79306	0.730972	1.172468
1	v10	1	1	1	2	-1.65629	-0.14263	1.102673
1	v1	2	1	1	2	-1.33972	-0.68782	1.301362
1	v2	2	1	1	2	-0.96814	-1.29641	1.409536
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	v9	2	1	1	2	-2.79306	0.144738	1.172468
1	v10	2	1	1	2	-1.65629	-0.14263	1.102673
1	v1	3	1	2	2	-0.68782	1.301362	
1	v2	3	1	1	2	-0.96814	-1.29641	1.409536
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	v9	3	1	1	2	-2.79306	0.730972	1.172468
1	v10	3	1	1	2	-1.65629	-0.14263	1.102673
0	Group1	1	1	0	0	1		
0	Group2	2	1	0	0.2	1		
0	Group3	3	1	0	-0.2	1.5		

As before, the first column still indicates whether the row refers to an item or a group, and the second column provides a label. The first indication that this is a multiple group problem appears in Column 3, which is the column designated for assigning the row (item or group) to a specific group defined in the syntax. In previous examples, with only one group, all rows displayed a `1` in this column, where now the first 10 items (4 of which are displayed in the table) have `1`s, the next 10 rows have `2`s and so on.

With respect to the differences across the groups, as noted earlier, Item v1 in group 3 is a 2PL item, so it is specified as a graded model with two response categories, while item v1 in both Groups 1 and 2 are 3PL items. DIF also occurs for Item v9. The same generating IRT model is used and the guessing and discrimination parameter values are identical across groups, but in Group 2, the difficulty parameter value is 0.5 higher (in the *b* metric) than the value in Groups 1 and 3. Mean differences among the groups are also generated. The last 3 rows refer to Groups 1 through 3, respectively. After labeling, specifying group membership, and declaring the number of factors, normal population distributions are chosen for all groups (via the `0`s in the 5 column). The sixth column provides the population means and the seventh column provides population variances.

7.8. Simulating a Multilevel Bifactor Model with Covariates

Our final example will illustrate how one may simulate item responses according to a multilevel bifactor model in two groups with covariates at either both or only one level of analysis. We will generate data for 12 dichotomously scored graded items in two separate groups.

Example 7-7: Simulation - Multilevel, Multiple Group Model with Bifactor Structure and Covariates

```
<Project>
Title =  "Two-level Bifactor Model (Two Groups)";
Description= "12 Items: 1 L2 factor,
5 L1 factors with 1 General, 4 Specific,
100 L2 units, 10 respondents within each";

<Options>
Mode = Simulation;
```

```
Rndseed = 7471;
ReadPRMFile =  "MGsimL2 covariates-prm.txt";

<Groups>
%G1%
File =  "simL2.g1.dat";
Varnames = v1-v12;
Dimensions = 6;
Primary = 2;
Between = 1;
N = 1000;
Nlevel2 = 100;
Covariates = 2;
L2covariates = 1;
CovariateCorr = 0.4;

%G2%
File= "simL2.g2.dat";
Varnames = v1-v12;
Dimensions = 6;
Primary = 2;
Between = 1;
N = 1000;
Nlevel2 = 100;
Covariates = 1;

<Constraints>
```

Skipping down to the `<Groups>` section and focusing on the first group's subsection, we have provided the data file name, variable labels, and the total number of dimensions of the generating model with the `Dimensions` statement. The command `Between = 1;` indicates that the model has one level-2 (between) latent variable and 5 level-1 (within) latent variables. The `Nlevel2` command is required when simulating multilevel data and determines the number of level-2 units (e.g., schools in which students are nested). In this case, the number of level-2 units is equal to 100. In conjunction with the total number of level-1 cases specified in the `N = 1000;` line, one can deduce that the number of level-1 units within each level-2 unit is equal to 10. flexMIRT^{TM} will distribute the total sample size as evenly as possible across the level-2 units.

If the sample size value is not a multiple of the number of level-2 units, the "remainder" observations will be added to the final level-2 unit. Note that while the specifications for the second group are the same as those in the first group in this example, it is by no means a requirement. As mentioned earlier, the number of items, type of IRT model, and the latent dimensionality can be different across groups.

In addition to the multiple levels, we have also requested that the model be generated with covariates. In group `G1`, we have specified that two covariates should be generated (`Covariates = 2;` and one of the covariates should predict the level-2 (Between) factor (`L2covariates`). We have also set the correlation among the 2 covariates to use a generating values of 0.4. In the second group, we have specified a simpler covariate pattern, in which there is just one level-1 covariate to be generated. When generating data with covariates, the ordering of values in the parameter file for group rows are slightly changed from the typical order discussed previously. We present the first several columns of the parameter file to be used in the above simulation.

Table 7.6: Parameter file excerpt: Simulating multiple groups with multilevel covariates

0	G1	1	6	0	0	0	0	0	0	0	1.3	0	0	0.5	0	0	0	0	0	0	0	0	0.5	0
0	G2	2	6	0	0.2	0	0	0	0	0	0	0.4	0	0	0	0	0.25	0	1	0	0	1	0	0

We previously detailed the ordering of the first several columns of lines related to groups and the meaning of these columns has not changed. We still see a 0 in the first column, indicating the line refers to a group, the group label, the group number, the number of factors, and a 0 indicator telling flexMIRT$^{\mathrm{TM}}$ to use a normal prior for the latent variables. Again, as with previous parameter files, the means of all the factors (in this example 6) are then presented. In group `G1` the means of all factors are set at zero; in the second group, the first factor (which will be the Between (level-2) factor is set at 0.2 and the following 5 values set the means of all the level-1 factors in the group to 0.

In a simulation without covariates, the next set of entries would be the variance-covariance matrix values. However, for this example we are using covariates in both groups, so we must first supply the entries of the Beta matrix, which defines the regression coefficients of each latent variable on the covariates. For group 1, the Beta matrix of 6 latent variables by 2 covariates we want to simulate from is detailed in the following table.

Table 7.7: Beta matrix with generating regression coefficient values for group G1

Latent Variable	Covariate1	Covariate 2
theta1	1.3	0
theta2	0	0.5
theta3	0	0
theta4	0	0
theta5	0	0
theta6	0	0

Entries in the table are regression coefficients of the latent variables on the covariates. Referring to the syntax, we see that in group `G1` we specified one level-2 covariate, which we have set at a regression coefficient of 1.3 in the table, which leaves one level-1 covariate. This level-1 covariate is set as a predictor of general factor of the level-1 model (theta 2) with a regression coefficient of 0.5. Comparing the table to the entries of the parameter file, we can see that the rows of the Beta matrix have been "unstacked" and the 12 regression coefficients from the table (including all 0 entries) are simply listed by row. After the last beta value has been specified, we then see the start of the variance-covariance entries, with the variance of the first factor in group `G1` set to 0.5 and the covariance of factor 2 and factor 1 set at 0.

For the second group, in the syntax we stated that there is only 1 covariate for this group and, as `L2covariates` has not been changed from its default of 0, the covariate will apply to the level-1 latent variables. The group `G2` Beta matrix, which will be 6 latent variables by 1 covariate, is specified with the covariate predicting only the general factor of the level-1 model, with a regression coefficient of 0.4. Again, the rows of the beta matrix have been "unstacked" and the values are listed across in the parameter file. After the 6 beta matrix entries, the next column starts the variance-covariance matrix for group `G2` with the variance of the Between factor set at 0.25.

Output 7.4: Simulation Control Output for Multilevel Bifactor Model

```
Two-level Bifactor Model (Two Groups)
12 Items: 1 L2 factor, 5 L1 factors with 1 General, 4 Specific,
100 L2 units, 10 respondents within each

Summary of the Data and Dimensions
               Group        G1        G2
   Missing data code        -9        -9
     Number of Items        12        12
 Number of L-2 Units       100       100
 Number of L-1 Units      1000      1000
 # Latent Dimensions         6         6
  Between Dimensions         1         1
   Within Dimensions         5         5
 Dimension Reduction  Yes[  2]  Yes[  2]
Number of Covariates         2         1
  Level-2 Covariates         1         0

...
Miscellaneous Control Values
Random number seed:        7471
...
2PL Items for Group 1: G1
    Item    a  1    a  2    a  3    a  4    a  5    a  6       c
       1    2.00    2.00    1.50    0.00    0.00    0.00   -1.50
       2    1.50    1.50    1.50    0.00    0.00    0.00   -0.50
       3    1.00    1.00    1.20    0.00    0.00    0.00    0.50
       4    2.00    2.00    0.00    1.20    0.00    0.00    1.50
       5    1.50    1.50    0.00    1.50    0.00    0.00   -1.50
       6    1.00    1.00    0.00    1.80    0.00    0.00   -0.50
       7    1.50    1.50    0.00    0.00    1.20    0.00    0.50
       8    1.50    1.50    0.00    0.00    1.50    0.00    1.50
       9    1.00    1.00    0.00    0.00    1.50    0.00   -1.50
      10    2.00    2.00    0.00    0.00    0.00    1.80   -0.50
      11    2.00    2.00    0.00    0.00    0.00    2.00    0.50
      12    1.00    1.00    0.00    0.00    0.00    1.50    1.50
...
Group Latent Variable Means:
   Group              Label   mu  1   mu  2   mu  3   mu  4   mu  5   mu  6
       1                 G1    0.00    0.00    0.00    0.00    0.00    0.00
       2                 G2    0.20    0.00    0.00    0.00    0.00    0.00
Latent Variable Variance-Covariance Matrix for Group  1: G1
 Theta  1 Theta  2 Theta  3 Theta  4 Theta  5 Theta  6
     0.50
     0.00     1.00
     0.00     0.00     1.00
     0.00     0.00     0.00     1.00
     0.00     0.00     0.00     0.00     1.00
     0.00     0.00     0.00     0.00     0.00     1.00

Latent Variable Variance-Covariance Matrix for Group  2: G2
 Theta  1 Theta  2 Theta  3 Theta  4 Theta  5 Theta  6
     0.25
     0.00     1.00
     0.00     0.00     1.00
     0.00     0.00     0.00     1.00
     0.00     0.00     0.00     0.00     1.00
     0.00     0.00     0.00     0.00     0.00     1.00

Latent Regression Coefficients:

   Group  1: G1
          Covariate  1  Covariate  2
Theta  1          1.30          0.00
Theta  2          0.00          0.50
Theta  3          0.00          0.00
Theta  4          0.00          0.00
Theta  5          0.00          0.00
Theta  6          0.00          0.00

   Group  2: G2
          Covariate  1
Theta  1          0.00
Theta  2          0.40
Theta  3          0.00
Theta  4          0.00
Theta  5          0.00
Theta  6          0.00
```

The simulation control output shows that the input syntax and the parameter values are specified correctly. The item parameters are invariant across groups so only parameters from Group 1 are shown here. The within-item equality of the slopes implies that the 5 level-1 dimensions (`a2 - a6`) follow a testlet response theory model (restricted bifactor) with 4 testlets of 3 items each. Furthermore, the equality of slopes on the Within general dimension (`a2`) and the Between general dimension (`a1`) implies that a random-intercept model is set up that provides a decomposition of the Between and Within variances for the general trait. The mean of group 1 is 0.20 lower than group 2, but the group 1 Between variance is 0.50 while the group 2 Between variance is 0.25. Additionally, the changing number of covariates across groups and their specified beta regression coefficient values can be see in the "Latent Regression Coefficients" tables reported for each group. Note that when flexMIRT^{TM} simulates covariates, it will only simulate normally distributed continuous covariates - that is, dichotomous or categorical covariates are currently unable to be simulated by the program.

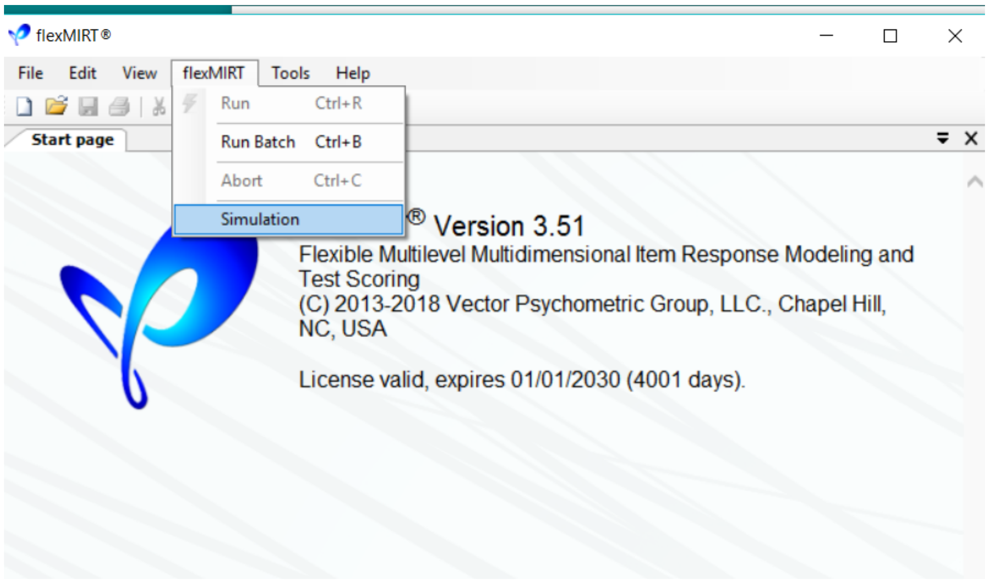
7.9. Internal Simulation Function

Here we provide an overview for using the internal simulation function (ISF), which allows users to automate numerous replications of a given simulation and subsequent calibration. This function uses existing simulation and calibration syntax files and merely provides a convenient way of obtaining replications; it is not an alternative method for constructing simulation syntax files. Additionally, the ISF is intended primarily for use in item and group parameter recovery simulation studies; even if scoring is requested in the supplied calibration syntax files, scores (typically saved in a -sco file) will not be produced/saved. If the recovery of person parameters is to be assessed, the ISF will not be useful and an alternate method of conducting calibration/scoring on replications should be utilized.

7.9.1 Accessing the Internal Simulation Function

The ISF is available through the flexMIRT^{TM} GUI only; specifically, the ISF is not available if using the program through the command prompt or when it is being called by another program, such as R. The ISF is found under the "flexMIRT" menu bar option of the flexMIRT^{TM} GUI.

Figure 7.2: Accessing the ISF through the GUI



® Version 3.51
Flexible Multilevel Multidimensional Item Response Modeling and Test Scoring
(C) 2013-2018 Vector Psychometric Group, LLC., Chapel Hill, NC, USA

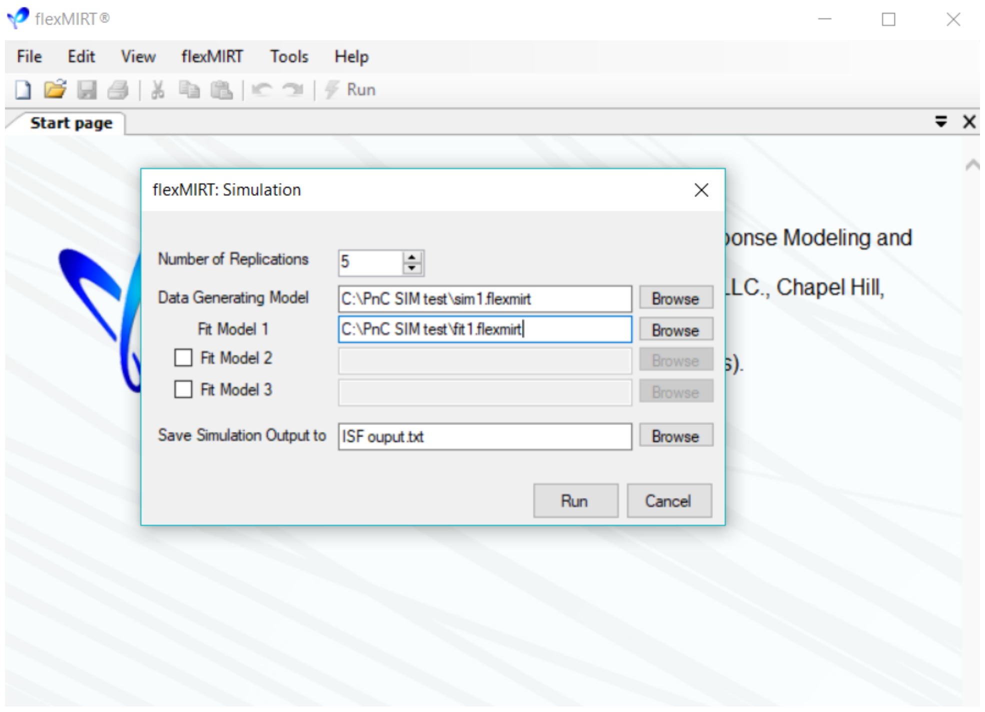
License valid, expires 01/01/2030 (4001 days).

Once "Simulation" is selected, a pop-up will appear where the user enters the information for the simulation study to be conducted. As can be seen in the next image, the user specifies the desired number of replications, the name and location of the existing simulation syntax file which will be responsible for the data creation, and the location(s) and name(s) of the calibration syntax for the model(s) that the users wishes to fit to the simulated data.

flexMIRT^{TM} will output the simulated data sets to external files by default. The simulated datasets will be output to the file name specified in the simulation syntax file but will have the replication number added to the output file name. For instance, if we are outputting the data to the file “SIM.dat" via the `File = "SIM.dat";` command in the simulation syntax, the individual replications will be saved to files “SIM-0.dat,” “SIM-1.dat,” etc. However, in the calibration syntax, to direct flexMIRT^{TM} to the datafile, we will still use the statement `File = "SIM.dat";` - without any mention of the replication number.

When all the syntax files have been identified and a name for the output file is specified in the “Save Simulation Output to:” blank, the “Run” button will be enabled.

Figure 7.3: Completed ISF pop-up window



Number of Replications: 5
Data Generating Model: C:\PnC SIM test\sim1.flexmirt
Fit Model 1: C:\PnC SIM test\fit1.flexmirt
Fit Model 2
Fit Model 3
Save Simulation Output to: ISF ouput.txt

Once the simulation has been submitted, flexMIRT^{TM} will begin the process of simulating data and fitting the specified model(s) to the data. The progress of each replication is printed in the engine viewing pane, as with any one-off simulation or calibration. When the specified number of replications have completed, a single output file is created. The format of the ISF output file is optimized to be read into some other software program (e.g., SAS, R) and as such, appears quite different from the normal flexMIRT^{TM} output. The output file has no labeled sections or column headers, etc.. Due to this, the layout of the ISF output file will be described in detail.

7.9.2 The Simulation Output File

The first 6 columns in the output for every model summary will contain the same information. Column 1 has the simulation replication number. Column 2 prints what is called the return code and is an indicator of successful replications. If the return code is 0, the replication completed successfully; if it takes any value other than 0, a problem was encountered. Column 3 reports the log likelihood, Column 4 gives the number of estimation cycles that were completed to fit the model, the 5th column reports the time the calibration run took, and the 6th column gives the inverse of the condition number. The inverse condition number is equal to the ratio of the smallest to largest eigenvalue of the asymptotic covariance matrix of the item parameters; if it is very small, it shows instability of the obtained solution.

Output 7.5: Beginning Columns of Output from 5 Replications

```
1 0 -4.42522e+003 46 1.41000e-001 8.61952e-003 2.87409e+000 1.99261e+000 7.62795e-001
2 0 -4.44206e+003 40 1.25000e-001 1.52923e-002 3.00412e+000 1.88355e+000 7.38170e-001
3 0 -4.28512e+003 41 1.25000e-001 8.18986e-003 2.78522e+000 1.87415e+000 7.44810e-001
4 0 -4.33888e+003 31 1.09000e-001 1.63897e-002 2.74887e+000 1.86261e+000 6.77367e-001
5 0 -4.37110e+003 29 1.09000e-001 1.82420e-002 2.58602e+000 1.84299e+000 8.12819e-001
```

At the seventh column, reporting of the item and latent variables parameters begins. The parameters are reported in the same order as they are numbered in a corresponding one-off calibration run output. It is strongly recommended that a single run of the calibration file(s) be conducted prior to employing the ISF, to both ensure the syntax works properly and to obtain the ordering of parameters. For example, below is the item parameter portion of the flexMIRT^{TM} output for a single run fitting the model found in the file

“Fit1.flexmirt”.

Output 7.6: Item Parameters with Parameter Number Labels

```
Graded Items for Group 1: G1
    Item              Label  P#       a     s.e.  P#     c 1     s.e.  P#     c 2     s.e.
       1                 v1   3    0.81     0.13   1    2.59     0.14   2    1.84     0.11
       2                 v2   6    1.14     0.15   4    2.69     0.16   5    2.01     0.13
       3                 v3   9    1.41     0.15   7    2.92     0.18   8    0.20     0.09
       4                 v4  12    1.25     0.13  10    2.88     0.16  11    0.13     0.08
       5                 v5  15    2.13     0.28  13    2.76     0.24  14    1.70     0.18
       6                 v6  18    0.54     0.09  16    2.92     0.14  17    0.20     0.07
```

The order in which the parameters will be printed in the ISF output corresponds to the values list in the columns labeled “P#” in the output. From this output excerpt, we can see that in the ISF output file, the two intercept parameters and then the slope parameter for Item v1 will be printed first, as indicated by the 1, 2, and 3, respectively, in the “’P#” columns. These values are followed by the intercept parameters and slope for Item v2, and so on. Once the item parameters have been reported, the latent variable mean(s) and covariance(s) are printed. After all point estimates are given, the respective standard error for each parameter is printed, in the same order as the point estimates.

Following all parameter point estimates and SEs, the next six columns of the ISF output are fixed. flexMIRT^{TM} will report, in order, the G^2, X^2, and M_2 overall fit statistics. It should be noted here that the M_2 column will always print, regardless of whether the statistic was requested or not. If the statistic was not requested, via the `GOF` and `M2` statements in the `<Options>` section, placeholder values of 0.0000 will be printed and are not indicative of the actual value of the M_2 statistic, had it been requested.

The overall fit statistics are followed by three members of the power-divergence family of fit statistics (e.g., Cressie & Read, 1984) for assessing latent distribution fit, specifically the summed score likelihood based G^2, D^2, and X^2. These indices have been found to accurately detect departures from normality within the latent distribution while appropriately “ignoring” other forms of model misspecification such as multidimensionality (Li & Cai, 2012).

Output 7.7: ISF output - Switch from Model 1 to Model 2 Values

```
... 1.29309e-314 1.47651e-317 | 1 0 -4.52136E+03 31 ...
... 9.70097e-315 1.47651e-317 | 2 0 -4.40664E+03 24 ...
... 2.32371e-315 1.47651e-317 | 3 0 -4.44256E+03 24 ...
... 1.04469e-314 1.47651e-317 | 4 0 -4.45391E+03 21 ...
... 1.02812e-314 1.47651e-317 | 5 0 -4.58027E+03 22 ...
```

If the ISF was instructed to fit multiple models to the data, then the values for next model will follow, on the same line, after the latent distribution fit statistics of the current model. The new model reporting begins with the replication number and follows the same general order of reporting detailed previously. If we fit two models to a given set of simulated data, the output columns at the point where the reporting for the first model ends and the second model begins will look something like the values presented in Output 6.7 (although we have added the break line for emphasis). Note, however, that the numbering of parameters may change from model to model. As stated before, a single run of any model being fit is strongly recommended before using the ISF.

CHAPTER 8

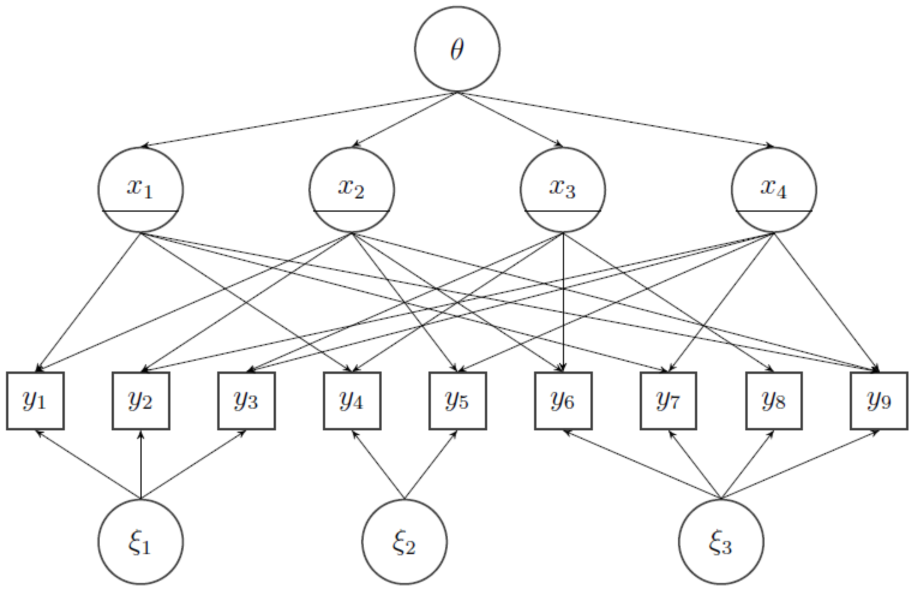
Diagnostic Classification Models

Over the last several decades, a growing field of interest and research has been the development and estimation of latent variable models that locate respondents in multidimensional space for the purpose of diagnosis; in this context, "diagnosis" includes the usual meaning, such as diagnosing a psychological disorder, but also encompasses activities such as identifying a student's mastery of content areas in an educational setting. Such models have been variously termed diagnostic classification models (DCMs), cognitive diagnosis models, cognitive psychometric models, restricted latent class models, and structured IRT models, among others. The goal of this chapter is not to provide a didactic on DCMs (interested readers are directed to Rupp, Templin, and Henson (2010) for a thorough discussion of DCMs and related models) but to demonstrate, for those already familiar with the models, how flexMIRT$^{\mathrm{TM}}$ may be used to fit them.

8.1. General DCM Modeling Framework

Complete coverage of the models flexMIRT$^{\text{TM}}$ fits to DCMs is provided in the Models chapter, but we will briefly introduce the models here. Figure 8.1 provides a schematic for a generic flexMIRT$^{\text{TM}}$ DCM with a single higher-order factor to facilitate the discussion of the mathematical equations.

Figure 8.1: Generic Higher-order DCM



The extended DCMs used by flexMIRT$^{\text{TM}}$ combine a linear predictor with a link function. The linear predictor may be represented as:

$$\begin{aligned} \eta = & \alpha + \sum_{k_1=1}^{K} \beta_{k_1} x_{k_1} + \sum_{k_1=1}^{K} \sum_{k_2=1}^{(k_1-1)} \beta_{k_1,k_2} x_{k_1} x_{k_2} \\ & + \text{higher-order interaction terms} + \sum_{k=1}^{p} \lambda_k \xi_k \end{aligned} \tag{8.1}$$

where α, β, and λ are item parameters, xs are 0/1 attributes, and ξs are continuous latent variables. The kernel (linear predictor) η can be turned into item response probabilities through the use of appropriate link functions, such as the logit presented below.

$$P(y_i = 1|\mathbf{x}, \xi) = \pi_i(1|\mathbf{x}, \xi) = \frac{1}{1 + \exp[-(\eta)]}$$

For the higher-order portion of the DCM, flexMIRT$^{\text{TM}}$ is modeling the x probabilities, (i.e., $P(x_1, ..., x_K|\theta_1, \ldots, \theta_q) = \prod_{k=1}^{K} P(x_k|\theta_1, \ldots, \theta_q)$) with a set of (higher-order) latent variables (θ), so that each of the attributes, xs, is treated as if it is an observed item.

8.2. DCM Specific Syntax Options

Syntax Display 8.1: `<Options>`, `<Groups>`, and `<Constraints>`- DCM Specific Options

```
<Options>
DMtable  = Yes/No;

<Groups>
Attributes  = ?;
InteractionEffects  = (?,?,...);
Generate  = (?,?),(?,?),...;

DM = group;

<Constraints>
Coeff group, (vars), parameter, ?;
Fix group,(vars), MainEffect;
Free group,(vars), Interaction(?,?,..);
```

In the `<Options>` section, the user may control whether or not the (sometimes large) diagnostic classification probability table is printed in the output via the `DMtable` command.

`Attributes` is used to set the number of main effects present in the skill space that is being assessed. This is the keyword that triggers flexMIRT$^{\text{TM}}$ to model discrete latent variables, rather than the default continuous latent variables.

`InteractionEffects` instructs flexMIRT$^{\text{TM}}$ to automatically generate all possible interaction effects. For instance, with 4 attributes one may specify

`InteractionEffects = (2,3,4);` and, in addition to the 4 main effects, the program with also generate the 2nd, 3rd, and 4th order interaction terms of the attributes as dimensions of the model. If only the second order interaction effects are desired, those would be created by specifying `InteractionEffects = (2);`.

Similar to the `InteractionEffects` keyword, `Generate` sets up higher order interactions effects. However, rather than creating all possible interaction effects `Generate` creates only those effects specified in the statement. For example, `Generate = (3,6,7),(4,7);` is used to generate the interaction effect of attributes 3, 6, and 7 and, separately, the interaction effect of attributes 4 and 7. There is no limit to the number of interaction effects that may be specified in a `Generate` statement and the order in which interaction effects are specified does not matter.

The `DM` command stands for "diagnostic model" and is used to set the group containing observed data - this is the group to which the higher-order DM will be applied. The `DM` command is optional; one could, in principle, fit a model without the DM (higher-order) latent variables.

In the `<Constraints>` section, the `Coeff` command allows users to impose proportionality restrictions on the item parameters (i.e., `MainEffects` and `Interactions`). For instance, a DINO model may be parameterized as a logistic model with interactions but the interaction coefficients are -1.0 times the main effect slopes. With the `Coeff` keyword, we can change the sign of those specific interaction terms, effectively enabling flexMIRT^{TM} to fit more DCMs than it would be capable of fitting if restricted to only the `MainEffect` and `Interaction` keywords.

The `Fix` and `Free` commands noted in the `<Constraints>` section aren't limited to those particular commands or to DCMs, but are provided to show that rather than using the typical IRT parameter keywords `Slope`, etc.), users may employ the parameter keywords `MainEffect` and `Interaction` to allow syntax specification that is in keeping with the DCM conceptualization/terminology for the models.

8.3. DCM Modeling Examples

In this section, we will provide several examples of DCMs fit in flexMIRT^{TM} - all examples will use default BAEM estimation as DCMs have not been implemented for either MH-RM or MCMC estimation. We will discuss basic examples that demonstrate how to fit several of the more well-known DCMs that appear in the literature and then, using datasets that have been fit elsewhere, demonstrate the comparability of flexMIRT^{TM} estimates to those from other programs and estimation methods.

8.3.1 Introductory DCM Examples

Here we provide straight-forward demonstrations of several widely discussed models. We hope that in addition to providing simple, concrete examples for the chosen models, from these examples users will begin to see the flexibility flexMIRT^{TM} has in fitting DCMs, gain insight into the details of our syntax for DCMs, and be able to extrapolate this to fit the wide variety of alternative DCMs that are not specifically covered but may be fit using flexMIRT^{TM}. For the examples in this section, we will use simulated data for 15 dichotomous items that were generated to measure four underlying attributes. The Q-matrix, which assigns items to attributes, is presented in Table 8.1. As can be seen, the majority of items use only 1 attribute, but the final 3 items are assigned to both attributes 3 and 4. These items could be fit with a variety of models and we will present examples for three different DCMs.

Table 8.1: Q-matrix for Basic DCM Demonstrations

Item	Attribute 1	Attribute 2	Attribute 3	Attribute 4
Item 1	1	0	0	0
Item 2	1	0	0	0
Item 3	1	0	0	0
Item 4	0	1	0	0
Item 5	0	1	0	0
Item 6	0	1	0	0
Item 7	0	0	1	0
Item 8	0	0	1	0
Item 9	0	0	1	0
Item 10	0	0	0	1
Item 11	0	0	0	1
Item 12	0	0	0	1
Item 13	0	0	1	1
Item 14	0	0	1	1
Item 15	0	0	1	1

Basic C-RUM Fit to Simulated Data

For the first example, the majority of items will only depend on the main effect of a single attribute and we will fit the compensatory reparameterized unified model (C-RUM) to the final three items. The C-RUM model (e.g., Hartz, 2002; Choi, Rupp, & Pan, 2013) is a DCM that, when items are informed by multiple attributes, allows for mastery of one attribute to compensate for non-mastery on other attributes. The C-RUM estimates for each item an intercept term and as many slope terms for an item as there are "1" entries in the Q-matrix. As detailed in Choi et al. (2013), C-RUM is a special case of the log-linear diagnostic classification model (LDCM) in which all interaction terms are set to 0, meaning only main effects are estimated.

Example 8-1: C-RUM Model Fit To Simulated Data

```
<Project>
Title = "Calibrate and Score Simultaed DM Data";
Description= "C-RUM Model";

<Options>
Mode = Calibration;
MaxE   = 20000;
MaxM   = 5;
Mtol   = 0;
Etol   = 1e-5;
SE = REM;
SaveCOV = Yes;
SavePRM = Yes;
SaveSCO = Yes;
Score = EAP;

<Groups>
%G%
File ="DMsim.dat";
Varnames = v1-v15
N= 3000;
Ncats(v1-v15) = 2;
Model(v1-v15) = Graded(2);

```

```
Attributes = 4;
Dimensions = 4;

%D%
Varnames = a1-a4;
DM = G;

<Constraints>
Fix G, (v1-v15),MainEffect;
Free G, (v1-v3),MainEffect(1);
Free G, (v4-v6),MainEffect(2);
Free G, (v7-v9,v13-v15),MainEffect(3);
Free G, (v10-v15),MainEffect(4);
```

The first points of note in the syntax occur in the `<Options>` section. The maximum number of E-steps has been increased from the default, the E tolerance and M tolerance values have been decreased, and the SE calculation method has been set to REM (Richardson extrapolation method), as experience has found the REM SEs to be preferable to estimates from other methods. These changes in the `<Options>` section are made to improve the resulting SE estimates and are recommended for any DCMs fit using flexMIRT©.

The specifications for the DCM begin in the `<Groups>` section. Although the naming of the groups is arbitrary, all examples will use the custom of defining group `G` as the group containing observed data and group `D` as the group containing the higher-order latent dimensions/attributes. As with a typical IRT model, in group `G` we read in our data from a named file, provide flexMIRT™ with the variable names, and specify the sample size. The items were simulated as correct/incorrect items, so `Ncats` has been set to 2 and we will fit the 2PLM to all items. For the first statement specific to the DCM, we tell flexMIRT™ that 4 latent attributes will be used. Because the C-RUM model uses only main effect terms, the total number of dimensions is equal to the number of attributes.

In the D group, we inform flexMIRT™ that the higher-order portion of the model we are setting up will be applied to the group with observed data, group `G`, using the `DM = G;` statement. We then name the four attributes via `Varnames = a1- a4;`. If no other information is given regarding categories or a model to be fit, flexMIRT™ by default will assume there are two categories for each of the attributes and will fit the attributes with the 2PLM. In the

`<Constraints>` section, we assign items onto attributes according to the Q-matrix. First, we fix all items to 0 and then free items onto the appropriate main effect and interaction terms. Items 1 - 12 only have main effects for a single attribute specified in the `<Constraints>` section. From the Q-matrix, we know that items 13 - 15 will load on both attributes 3 and 4. These specifications are incorporated in the statements `Free G,(v7-v9, v13-v15), MainEffect(3);` and `Free G,(v10-v15), MainEffect(4);`, in which items 13 - 15 appear in the list of items freed onto both attribute 3 and attribute 4.

The first section of the output excerpt presents the item parameter estimates, with freely-estimated versus fixed-at-zero parameters mirroring the Q-matrix specifications. The factors labeled a1-a4 represent main effects/slope parameters for attributes 1 - 4, respectively, and c is the intercept parameter.

The next section of interest is labeled `Diagnostic IRT Attributes and Cross-classification Probabilities for Group 1:  G`, which reports the estimated proportion of the population in each attribute profile. For instance, it is expected that about 15% of the population have mastered none of the measured attributes (attribute profile [0 0 0 0]), around 4 % have mastered attributes 2 and 4 (profile [0 1 0 1]), and 18.4% of the population have mastered all 4 attributes (profile [1 1 1 1]).

Output 8.1: C-RUM Output - Item Parameters and Attribute Profile Posterior Probabilities

```
    Item          Label   P#    a  1    s.e.   P#    a  2    s.e.   P#    a  3    s.e.   P#    a  4    s.e.   P#       c    s.e.
       1             v1   24    1.89    0.14         0.00    ----         0.00    ----         0.00    ----    1   -1.68    0.07
       2             v2   25    1.91    0.15         0.00    ----         0.00    ----         0.00    ----    2   -1.14    0.07
       3             v3   26    1.91    0.17         0.00    ----         0.00    ----         0.00    ----    3   -0.63    0.06
       4             v4         0.00    ----   27    2.43    0.25         0.00    ----         0.00    ----    4   -0.34    0.07
       5             v5         0.00    ----   28    2.23    0.17         0.00    ----         0.00    ----    5   -1.16    0.08
       6             v6         0.00    ----   29    1.88    0.33         0.00    ----         0.00    ----    6    1.54    0.07
       7             v7         0.00    ----         0.00    ----   30    1.93    0.14         0.00    ----    7   -1.29    0.10
       8             v8         0.00    ----         0.00    ----   31    2.16    0.15         0.00    ----    8   -0.85    0.09
       9             v9         0.00    ----         0.00    ----   32    1.69    0.13         0.00    ----    9   -0.37    0.08
      10            v10         0.00    ----         0.00    ----         0.00    ----   36    2.47    0.18   10   -0.05    0.11
      11            v11         0.00    ----         0.00    ----         0.00    ----   37    1.68    0.17   11    0.77    0.11
      12            v12         0.00    ----         0.00    ----         0.00    ----   38    1.92    0.20   12    1.14    0.12
      13            v13         0.00    ----         0.00    ----   33    0.10    0.41   39    0.96    0.42   13    2.77    0.22
      14            v14         0.00    ----         0.00    ----   34    0.70    0.28   40    1.03    0.26   14    1.60    0.14
      15            v15         0.00    ----         0.00    ----   35    0.55    0.15   41    1.85    0.20   15   -1.38    0.13

Diagnostic IRT Attributes and Cross-classification Probabilities for Group   1: G

 Pattern         Prob
 0 0 0 0   0.15002832
 0 0 0 1   0.16623281
 0 0 1 0   0.03889021
 0 0 1 1   0.19192117
 0 1 0 0   0.00750342
 0 1 0 1   0.03906408
 0 1 1 0   0.00531547
...
 1 0 1 0   0.00165291
 1 0 1 1   0.07127746
 1 1 0 0   0.00033627
 1 1 0 1   0.01527851
 1 1 1 0   0.00093495
 1 1 1 1   0.18411789
```

We will next cover the structure of the individual scoring file (-sco) that was requested in the `<Options>` section. In the -sco output excerpt presented, we have given only the first 10 cases and have included line breaks (//) and a separator (|) not in the output file for ease of presentation.

The first two columns of the -sco file give the group and observation number, similar to a standard flexMIRT^{TM} IRT score output file. Recall that for this example, there are 4 attributes and 16 possible attribute profiles. The estimated probability of the respondent having mastered each of the attributes individually is given, starting after the observation number. For example, for observation 1, the probability they have mastery of attribute 1 is 0.054, for attribute 2 it is 0.128 and for attribute 3 it is 0.165 and for attribute 4 it is 0.375. The following four columns, after the separator, are the estimated standard errors for those probabilities, respectively.

The entries that have been moved to the second line of each observation contain the estimated attribute posterior variance/covariance matrix. For observation 1, those values would enter the matrix as:

	a1	a2	a3	a4
a1	0.050755			
a2	0.023774	0.109961		
a3	0.024396	0.033344	0.137477	
a4	0.029047	0.049426	0.051856	0.234379

In what we have made the third line for each respondent are the posterior probabilities for each of the possible attribute profiles. As noted earlier with 4 attributes, there are 16 possible profiles and, therefore, 16 values on the third line for this example. For observation 1, the posterior probability of being in attribute profile 1 is 0.548, the probability of possessing attribute profile 2 is 0.204 and so on. The order in which the attribute profile probabilities are given matches the order of the probability profiles used in the output table "Diagnostic IRT Attributes and Cross-classification Probabilities." Looking back at the output presented for this example, we find that attribute profile 1 corresponds to [0 0 0 0] (meaning no attributes are mastered) and attribute profile 2 corresponds to [0 0 0 1] (meaning only attribute 4 has been mastered), and so on.

Output 8.2: C-RUM Output –SCO Output

```
  1     1     0.053631    0.125782    0.164556    0.375016  |   0.225289    0.331604    0.370779    0.484127   //
0.050755    0.023774    0.109961    0.024396    0.033344    0.137477    0.029047    0.049426    0.051856    0.234379 //
0.547785    0.203755    0.044491    0.055075    0.023108    0.040386    0.005129    0.026639    0.002596    0.009569    0.000926    0.010020    0.000507    0.007738    0.000442    0.021832  //
1   -0.863639    0.559221    0.312728
  1     2     0.094169    0.197099    0.208524    0.753622  |  0.292064    0.397808    0.406253    0.430901    //
0.085301    0.036159    0.158251    0.035482    0.043046    0.165042    0.021494    0.037324    0.035284    0.185676   //
0.219748    0.436824    0.014035    0.092845    0.009270    0.086583    0.001618    0.044908    0.001060    0.020890    0.000298    0.017201    0.000207    0.016893    0.000142    0.037478  //
2   -0.458755    0.526603    0.277310
  1     3     0.028209    0.072130    0.100814    0.199482  |  0.165569    0.258703    0.301083    0.399611   //
0.027413    0.011188    0.066927    0.011742    0.016077    0.090651    0.017207    0.027676    0.029037    0.159689  //
0.720026    0.121128    0.045985    0.025745    0.024803    0.019606    0.004329    0.010169    0.003468    0.005783    0.000973    0.004762    0.000554    0.003819    0.000379    0.008472  //
1   -1.088950    0.574875    0.330481
  1     4     0.009269    0.137086    0.038002    0.705494  |  0.095827    0.343938    0.191201    0.455820  //
0.009183    0.003258    0.118294    0.001570    0.007045    0.036558    0.002476    0.028187    0.007495    0.207772  //
0.278809    0.554228    0.003301    0.021836    0.011761    0.109853    0.000381    0.010562    0.000199    0.003930    0.000010    0.000600    0.000039    0.003178    0.000005    0.001307  //
2   -0.674111    0.528978    0.279818
  1     5     0.002525    0.007245    0.018326    0.233881  |  0.050183    0.084809    0.134128    0.423297  //
 0.002518    0.000148    0.007193    0.000266    0.000372    0.017990    0.001356    0.002651    0.004997    0.179181  //
 0.753729    0.219111    0.008923    0.008633    0.002798    0.003822    0000091    0.000367    0.000539    0.001554    0.000028    0.000237    0.000009    0.000111    0.000001    0.000045  //
1   -1.155748    0.576725    0.332611
  1     6     0.002825    0.007694    0.376743    0.057087  |  0.053073    0.087379    0.484570    0.232008  //
0.002817    0.000224    0.007635    0.001179    0.002216    0.234808    0.001130    0.001577    0.016181    0.053828  //
0.601189    0.018925    0.334564    0.035049    0.002232    0.000330    0.003394    0.001492    0.000430    0.000134    0.001052    0.000963    0.000007    0.000010    0.000044    0.000185  //
  1     7     0.004550    0.083366    0.023717    0.309510  |  0.067300    0.276435    0.152167    0.462292  //
0.004529    0.001691    0.076416    0.000765    0.004134    0.023155    0.002547    0.028993    0.007709    0.213714  //
0.653688    0.243148    0.007739    0.009580    0.027575    0.048194    0.000892    0.004634    0.000468    0.001724    0.000024    0.000263    0.000091    0.001394    0.000012    0.000573  //
1   -1.047601    0.567131    0.321638
  1     8     0.020510    0.009435    0.131043    0.184563  |  0.141736    0.096676    0.337448    0.387943  //
0.020089    0.001817    0.009346    0.008108    0.002867    0.113871    0.011562    0.004088    0.029010    0.150500  //
0.731348    0.123032    0.075446    0.042239    0.002715    0.002146    0.000765    0.001798    0.003466    0.005778    0.001571    0.007685    0.000060    0.000411    0.000066    0.001474  //
1   -1.123851    0.575411    0.331098
  1     9     0.126701    0.227366    0.460304    0.624168  |  0.332637    0.419131    0.498422    0.484337  //
0.110648    0.052320    0.175671    0.049910    0.064527    0.248424    0.043652    0.064029    0.085670    0.234582  //
0.275306    0.195548    0.076165    0.180042    0.011614    0.038759    0.008781    0.087085    0.001326    0.009335    0.001612    0.033299    0.000259    0.007549    0.000769    0.072550  //
1   -0.403618    0.536756    0.288107
  1    10     0.020842    0.009465    0.131177    0.184754  |  0.142855    0.096829    0.337594    0.388098  //
0.020408    0.001846    0.009376    0.008236    0.002886    0.113970    0.011746    0.004110    0.029093    0.150620  //
0.731100    0.122991    0.075420    0.042224    0.002714    0.002145    0.000765    0.001797    0.003522    0.005872    0.001596    0.007809    0.000061    0.000418    0.000067    0.001497  //
1   -1.123361    0.575390    0.331074
```

Finally, in what has been moved to the fourth line for each respondent the most likely attribute profile is listed, followed by the individual's estimated theta score for the higher-order latent variable, as well as the the SD and variance estimates for the higher-order theta estimate. For respondent 1, the most likely attribute profile is profile 1, corresponding to attribute profile [0 0 0 0] (mastery of no attributes); this attribute can be found to have the highest posterior probability listed on the line above.

Basic DINA Fit to Simulated Data

For the second example, we will fit the deterministic input, noisy-and-gate (DINA) model to the final three items. Unlike the C-RUM, this model is non-compensatory in nature, not allowing mastery on one attribute to make up for lack of mastery on other attributes. This is accomplished in the LDCM framework by constraining main effects to zero and allowing only the most complex attribute interaction term for an item to take on a non-zero value.

Example 8-2: DINA Fit to Simulated Data

```
<Project>
Title = "Calibrate and Score Simulated DM Data";
Description = "DINA Model";

<Options>
Mode = Calibration;
MaxE  = 20000;
MaxM  = 5;
Etol  = 1e-5;
SE = REM;
SaveCOV = Yes;
SavePRM = Yes;
SaveSCO = Yes;
Score = SSC;

<Groups>
%G%
File ="DMsim.dat";
Varnames = v1-v15
N = 3000;
Ncats(v1-v15) = 2;
Model(v1-v15) = Graded(2);

```

```
Attributes = 4;
InteractionEffects = (2);
Dimensions = 10;  //4 ME and 6 2nd order ints.

%D%
Varnames = a1-a4;
DM = G;

<Constraints>
Fix G, (v1-v15),MainEffect;
Free G, (v1-v3),MainEffect(1);
Free G, (v4-v6),MainEffect(2);
Free G, (v7-v9),MainEffect(3);
Free G, (v10-v12),MainEffect(4);

// item 13-15 fit with DINA model depends on int of attribs 3,4;
Free G, (v13-v15),Interaction(3,4);
```

All syntax in the `<Options>` section remains unchanged from the previous example, with the exception of requesting sum score to EAP conversion tables (via `Score = SSC;`). In the `G` group, we have declared the data file, variable names, sample size, number of categories, and IRT model to be fit to each item, as in the previous example. We again specify that there are 4 attributes with main effects, but because we are using the DINA model we will also need to construct dimensions that will represent the interaction effects. This is accomplished via `InteractionEffects = (2);` which is requesting that all second-order interaction terms (that is, attribute 1 with attribute 2, attribute 1 with attribute 3, etc.) be created for the model. Because we have requested all second-order interactions (of which there are 6 possible with the 4 attributes), the total number of dimensions for the final model is 10 (4 main effect terms and 6 second-order interactions). Syntax in the `D` group remains unchanged from the previous example.

In the `<Constraints>` section, as with the previous example, we initially fix all items to zero and then free items onto the appropriate attributes. Unlike the previous example, which used only main effects, we are also using the parameter keyword `Interaction` to free items 13-15 onto the interaction of attributes 3 and 4. As noted when introducing the model, for items that depend on more than one attribute, only the highest order interaction terms is allowed to be non-zero, so we do not free items 13-15 onto the main effects of attributes 3 or 4.

Because the majority of the output has the same structure as the previous example's, regardless of specific DCM fit to items, we will only briefly discuss outputted estimates. For this example, we highlight the output that relates to the item parameters and the summed score to EAP conversion table in the -ssc file that is printed due to `Score = SSC`;.

The output excerpt presents the estimated parameters for the items, with estimated versus fixed-at-zero loadings mirroring the Q-matrix specifications. The factors labeled `a1 - a4` represent main effects for attributes 1 - 4 and `a5 - a10` represent the requested 2nd order interactions. Items 1 - 12 have estimates for main effects on only the attributes specified in the Q-matrix and item 13-15 have estimates on only the interaction of attributes 3 and 4, which is labeled as `a10` in the parameter table. One may notice that dimensions `a5 - a9` that have no items assigned to them - these dimensions correspond to the unused second order interaction effects that were created via the `InteractionEffects` keyword. The estimated item values reported are in the LDCM parameterization, but the classic DINA parameters of guessing and slippage may be obtained by conversion. Guessing is found from the reported flexMIRT^{TM} parameters as $exp(c)/(1 + exp(c))$ and one minus the slippage term $(1 - s)$ is found as $exp(c + a)/(1 + exp(c + a))$.

Output 8.3: DINA output - Item Parameter Estimates

```
2PL Items for Group 1: G
Item           Label  P#    a  1    s.e.   P#    a  2    s.e.   P#    a  3    s.e.   P#    a  4    s.e.   P#    a  5    s.e.   P#    a  6    s.e.
   1              v1  24    1.87    0.14         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----
   2              v2  25    1.93    0.15         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----
   3              v3  26    1.90    0.17         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----
   4              v4        0.00    ----   27    2.38    0.25         0.00    ----         0.00    ----         0.00    ----         0.00    ----
   5              v5        0.00    ----   28    2.23    0.18         0.00    ----         0.00    ----         0.00    ----         0.00    ----
   6              v6        0.00    ----   29    1.87    0.32         0.00    ----         0.00    ----         0.00    ----         0.00    ----
   7              v7        0.00    ----         0.00    ----   30    2.01    0.15         0.00    ----         0.00    ----         0.00    ----
   8              v8        0.00    ----         0.00    ----   31    2.16    0.15         0.00    ----         0.00    ----         0.00    ----
   9              v9        0.00    ----         0.00    ----   32    1.63    0.13         0.00    ----         0.00    ----         0.00    ----
  10             v10        0.00    ----         0.00    ----         0.00    ----   33    2.53    0.17         0.00    ----         0.00    ----
  11             v11        0.00    ----         0.00    ----         0.00    ----   34    1.70    0.18         0.00    ----         0.00    ----
  12             v12        0.00    ----         0.00    ----         0.00    ----   35    1.90    0.21         0.00    ----         0.00    ----
  13             v13        0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----
  14             v14        0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----
  15             v15        0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----

Item           Label  P#    a  7    s.e.   P#    a  8    s.e.   P#    a  9    s.e.   P#    a 10    s.e.   P#       c    s.e.
  1               v1        0.00    ----         0.00    ----         0.00    ----         0.00    ----    1   -1.67    0.07
  2               v2        0.00    ----         0.00    ----         0.00    ----         0.00    ----    2   -1.14    0.07
  3               v3        0.00    ----         0.00    ----         0.00    ----         0.00    ----    3   -0.63    0.06
  4               v4        0.00    ----         0.00    ----         0.00    ----         0.00    ----    4   -0.34    0.07
  5               v5        0.00    ----         0.00    ----         0.00    ----         0.00    ----    5   -1.17    0.08
  6               v6        0.00    ----         0.00    ----         0.00    ----         0.00    ----    6    1.54    0.07
  7               v7        0.00    ----         0.00    ----         0.00    ----         0.00    ----    7   -1.55    0.13
  8               v8        0.00    ----         0.00    ----         0.00    ----         0.00    ----    8   -1.09    0.11
  9               v9        0.00    ----         0.00    ----         0.00    ----         0.00    ----    9   -0.52    0.09
  10             v10        0.00    ----         0.00    ----         0.00    ----         0.00    ----   10    0.04    0.09
  11             v11        0.00    ----         0.00    ----         0.00    ----         0.00    ----   11    0.83    0.11
  12             v12        0.00    ----         0.00    ----         0.00    ----         0.00    ----   12    1.24    0.11
  13             v13        0.00    ----         0.00    ----         0.00    ----   36    0.74    0.30   13    3.11    0.17
  14             v14        0.00    ----         0.00    ----         0.00    ----   37    1.33    0.22   14    2.01    0.11
  15             v15        0.00    ----         0.00    ----         0.00    ----   38    1.58    0.12   15   -0.52    0.08
```

In the -ssc file that was generated by the scoring request, we find tables that provide conversions from sum scores to the probability of mastery for each attribute. We present an excerpt of these table below.

Output 8.4: DINA Summed Score Conversion Tables from -ssc File

```
Summed Score to Attribute Profile Probability Conversion Table:
  Summed
   Score  EAP  1  EAP  2  EAP  3  EAP  4   SD  1   SD  2   SD  3   SD  4         P  Error Covariance Matrix
    0.00   0.001   0.001   0.026   0.005   0.030   0.026   0.158   0.070 0.0000002     0.000876
                                                                                       0.000003     0.000700
                                                                                       0.000115     0.000045     0.024902
                                                                                       0.000017     0.000007    -0.000092     0.004948

    1.00   0.001   0.001   0.029   0.010   0.032   0.035   0.168   0.097 0.0000093     0.001025
                                                                                       0.000006     0.001256
                                                                                       0.000147     0.000085     0.028217
                                                                                       0.000035     0.000017    -0.000168     0.009475

    2.00   0.001   0.002   0.036   0.021   0.036   0.049   0.186   0.143 0.0001508     0.001332
                                                                                       0.000013     0.002420
                                                                                       0.000214     0.000173     0.034470
                                                                                       0.000084     0.000041    -0.000363     0.020508
...


   13.00   0.760   0.795   0.990   0.983   0.427   0.404   0.097   0.129 0.0805989     0.182430
                                                                                       0.027666     0.163028
                                                                                       0.002884     0.002254     0.009474
                                                                                       0.005347     0.003948     0.001326     0.016527

   14.00   0.898   0.889   0.997   0.992   0.303   0.315   0.053   0.087 0.0356330     0.091595
                                                                                       0.013076     0.099016
                                                                                       0.000808     0.000623     0.002789
                                                                                       0.001972     0.001464     0.000291     0.007550

   15.00   0.966   0.947   0.999   0.996   0.180   0.225   0.027   0.060 0.0078999     0.032539
                                                                                       0.004564     0.050540
                                                                                       0.000175     0.000147     0.000739
                                                                                       0.000572     0.000493     0.000057     0.003578
```

For this example, each summed score is listed with four EAP values, corresponding to the 4 attributes specified in the model, 4 SD values, the expected proportion of respondents who will obtain the listed summed score (listed in the column labeled `p`), and the error variance/covariance matrix associated with that summed score. For example, given a summed score of 0, the probability that the individual has mastered attribute 1 is 0.001, the probability that they have mastered attribute 3 is 0.026 and so on. For individuals with a summed score of 15, those probabilities increase to 0.966 and 0.999, respectively. Rather than simply providing the most likely attribute profile associated with a summed score, flexMIRT$^{\text{TM}}$ prints the individual attribute probabilities so researchers have sufficient information to apply mastery/indifference/non-mastery cut-off values appropriate for the specific assessment at hand.

Basic DINO Fit to Simulated Data

The final basic example using this simulated dataset will fit the disjunctive, deterministic noisy-or-gate (DINO) model. Unlike the previous models, the DINO allows for both non-zero main effect and interaction terms. However, and again using the LDCM framework, constraints are placed such that the main effect slopes are set equal and the interaction term is -1 times the main effect constrained-to-equality slope. This is done because an item may be endorsed/answered correctly if any one of the attributes is mastered and there is assumed to be no added benefit to having mastery of several/all attributes over mastering only one or a subset of the attributes; the -1 applied to the interaction term serves to remove such a "benefit" from the model.

Example 8-3: DINO Fit to Simulated Data

```
<Project>
Title = "Calibrate and Score Simulated DM Data";
Description= "DINO Model";

<Options>
Mode = Calibration;
MaxE  = 20000;
MaxM  = 5;
Etol  = 1e-5;
SE = REM;
SaveCOV = Yes;
SavePRM = Yes;
SaveSCO = Yes;
Score = EAP;


<Groups>
%G%
File ="DMsim.dat";
Varnames = v1-v15
N=3000;
Ncats(v1-v15) = 2;
Model(v1-v15) = Graded(2);

Attributes = 4;
Generate  = (3,4);
Dimensions = 5;  //4 ME and 1 2nd order int.
```

```

%D%
Varnames  a1-a4;
DM = G;

<Constraints>
Fix G, (v1-v15),MainEffect;

Free G, (v1-v3),MainEffect(1);
Free G, (v4-v6),MainEffect(2);
Free G, (v7-v9),MainEffect(3);
Free G, (v10-v12),MainEffect(4);

// item 13-15 fit with DINO model that depends on int of attribs 3,4;
Free G, (v13-v15),MainEffect(3);
Free G, (v13-v15),MainEffect(4);
Free G, (v13-v15),Interaction(3,4);

Equal G, (v13), MainEffect(3):
      G, (v13), MainEffect(4):
      G, (v13), Interaction(3,4);

Coeff G, (v13), Interaction(3,4), -1;

Equal G, (v14), MainEffect(3):
      G, (v14), MainEffect(4):
      G, (v14), Interaction(3,4);

Coeff G, (v14), Interaction(3,4), -1;

Equal G, (v15), MainEffect(3):
      G, (v15), MainEffect(4):
      G, (v15), Interaction(3,4);

Coeff G, (v15), Interaction(3,4), -1;
```

Again, the maximum number of E-step and M-steps have been increased from the default, the E tolerance and M tolerance values have been decreased, and the SE calculation method has been set to REM in the `<Options>` section. As noted in the output from the previous example, there were several latent dimensions that had no items loading onto them. Here, rather than using the general `InteractionEffects` keyword, we have used the more targeted `Generate` to construct only those interaction terms which will be used in the model. As seen in the Q-matrix and discussed earlier, only the interaction of attributes 3 and 4 is needed and is specified by the `Generate = (3,4);` statement. If additional interactions were needed, they may be added by placing a comma after the last interaction term and listing the other desired interactions. For example, if the 3-way interaction of attributes 2, 3 and 4 was desired in addition to the 3-4 interaction, we could specify that using a single generate statement of `Generate = (3,4), (2,3,4);`.

As before, we free the first 12 items onto the appropriate single attributes. Following the comment in the syntax, we free items 13 -15 on both main effect terms for attributes 3 and 4, as well as the interaction of the two attributes. To induce the equality specified in the model, we set the main effects and interaction terms equal to each of the items via a statement such as `Equal G, (v13), MainEffect(3):G, (v13), MainEffect(4):G, (v13), Interaction(3,4);` and then apply the -1 to the interaction with a coefficient constraint of the form `Coeff G, (v13), Interaction(3,4), -1;`.

Output 8.5: DINO Output - Item Parameters

```
Item Label  P#   a  1    s.e.  P#    a  2    s.e.   P#    a  3    s.e.   P#    a  4    s.e.   P#    a  5    s.e.   P#       c    s.e.
  1     v1  24   1.89    0.14        0.00    ----         0.00    ----         0.00    ----         0.00    ----    1   -1.68   0.08
  2     v2  25   1.90    0.15        0.00    ----         0.00    ----         0.00    ----         0.00    ----    2   -1.14   0.07
  3     v3  26   1.90    0.17        0.00    ----         0.00    ----         0.00    ----         0.00    ----    3   -0.63   0.06
  4     v4       0.00    ----  27    2.47    0.26         0.00    ----         0.00    ----         0.00    ----    4   -0.33   0.07
  5     v5       0.00    ----  28    2.23    0.17         0.00    ----         0.00    ----         0.00    ----    5   -1.15   0.08
  6     v6       0.00    ----  29    1.91    0.34         0.00    ----         0.00    ----         0.00    ----    6    1.55   0.07
  7     v7       0.00    ----        0.00    ----   30    1.84    0.13         0.00    ----         0.00    ----    7   -1.18   0.09
  8     v8       0.00    ----        0.00    ----   31    2.17    0.14         0.00    ----         0.00    ----    8   -0.79   0.09
  9     v9       0.00    ----        0.00    ----   32    1.68    0.13         0.00    ----         0.00    ----    9   -0.32   0.08
 10    v10       0.00    ----        0.00    ----         0.00    ----   33    2.42    0.17         0.00    ----   10    0.12   0.10
 11    v11       0.00    ----        0.00    ----         0.00    ----   34    1.63    0.17         0.00    ----   11    0.87   0.10
 12    v12       0.00    ----        0.00    ----         0.00    ----   35    1.91    0.20         0.00    ----   12    1.25   0.11
 13    v13       0.00    ----        0.00    ----   36    1.06    0.29   36    1.06    0.29   36   -1.06    0.29   13    2.74   0.21
 14    v14       0.00    ----        0.00    ----   37    1.51    0.20   37    1.51    0.20   37   -1.51    0.20   14    1.60   0.14
 15    v15       0.00    ----        0.00    ----   38    2.53    0.14   38    2.53    0.14   38   -2.53    0.14   15   -1.66   0.13
```

As with the DINA model, the estimated item values reported are in the LDCM parameterization but DINO parameters of guessing and slippage may be obtained by conversion. Again, guessing is found from the reported parameters as $exp(c)/(1 + exp(c))$ and one minus the slippage term $(1 - s)$ is found as $exp(c + a)/(1 + exp(c + a))$. The a term used in the conversion should be the value reported in one of the main effect columns, not the negative value associated with the interaction term.

One aspect of the output that we haven't discussed as of yet is the model fit. We present the -2 log likelihood, AIC, and BIC values for each of the three models fit to our simulated data in Table 8.2. As with the output for more standard IRT models, these values are reported at the bottom of the -irt flexMIRT^{TM} output file. As can be seen in Table 8.2, the fit of the models indicates that using DINO for the final three items was the preferred model based on all three reported fit indices. It is informative to know that the same data set was used in all three examples and this data was generated using the DINO model for the last three items. The advantage for the DINO is sometimes small, but that is most likely a result of the majority of items being fit by single main effects, the specification of which did not vary regardless of the model that was used for the items with more complex attribute relationships.

Table 8.2: Model Fit for Basic DCMs Fit to Simulated Data

	CRUM	DINA	DINO
-2loglikelihood:	45799.53	45873.59	45798.31
Akaike Information Criterion (AIC):	45881.53	45949.59	45874.31
Bayesian Information Criterion (BIC):	46156.21	46204.17	46128.89

Basic DINA with Testlet Structure

For the final example in this section, we employ a new simulated dataset that contains responses to 24 dichotomous items that are to be modeled with 4 attributes. In addition to being generated from a DCM, this data was simulated to incorporate a testlet structure, such as when multiple reading items use the same passage as a prompt. A graphical depiction of such a model was given in Figure 8.1, where ys are the observed items, xs are the discrete attributes, and ξs are the testlet effects.

For the 24 item dataset we will be modeling, items 1 - 6 are members of testlet 1, items 7 - 12 belong to testlet 2, items 13-18 to testlet 3, and items

19-24 are members of the final testlet. The Q-matrix for mapping items onto attributes supplied in Table 8.3.

Table 8.3: Q-matrix for Testlet-DINA Example

Item	Attribute 1	Attribute 2	Attribute 3	Attribute 4
Item 1	1	1	0	0
Item 2	0	0	1	1
Item 3	1	0	1	0
Item 4	0	1	0	1
Item 5	1	0	0	1
Item 6	0	1	1	0
Item 7	1	1	0	0
Item 8	0	0	1	1
Item 9	1	0	1	0
Item 10	0	1	0	1
Item 11	1	0	0	1
Item 12	0	1	1	0
Item 13	1	1	0	0
Item 14	0	0	1	1
Item 15	1	0	1	0
Item 16	0	1	0	1
Item 17	1	0	0	1
Item 18	0	1	1	0
Item 19	1	1	0	0
Item 20	0	0	1	1
Item 21	1	0	1	0
Item 22	0	1	0	1
Item 23	1	0	0	1
Item 24	0	1	1	0

As seen in syntax, the `<Options>` section is largely unchanged from the previous examples, although we have reduced the number quadrature points from the default, due to the high-dimensional nature of this problem. In the first group, `G`, the data file, variable names, etc. have been given. As with previous DCM examples, we specify the number of attributes (`Attributes = 4;`) and also request that flexMIRT^{TM} create dimensions for the interaction effects. From the Q-matrix, every possible second order interaction is needed, so we use `InteractionEffects = (2);` to generate those dimensions automatically. To incorporate the testlet structure, we will fit a bifactor-type model, in which the DCM main effect and interactions are the primary/general dimensions and the testlet effects are considered the specific dimensions. To indicate this to flexMIRT^{TM} we set `Primary = 10;` which is the total number of DCM dimensions (4 main effects + 6 interactions) and set the total number of dimensions for the model equal to 14 (10 DCM dimensions + 4 testlet effect dimensions).

In the `<Constraints>` section, the first group of statements is used to assign items to the appropriate interaction terms, based on the Q-matrix, for the DINA model. The second group of constraints, beginning with `Free G,(v1-v6),Slope(11);` is used to assign items to their testlet dimension. As noted earlier, the first 10 latent dimensions are taken up by the DCM, so the first available dimension for a testlet effect is dimension 11, hence `Slope(11)` in the noted `Free` statement. The specific dimensions are specified as a restricted bifactor model, where the testlet effect is presumed to be a testlet-specific random intercept term. This requires the equality of the slopes within factors and is accomplished with the final group of constraints. With the final `Equal` constraint, we are fitting a Rasch-type model to the higher-order dimensions, constraining the slopes of attribute 1 through attribute 4 in group `D` to equality.

Example 8-4: Testlet Structure with DINA model

```
<Project>
Title =  "Fit HO-DINA Model with Testlets";
Description=  "HO-DINA Model - 4 attributes";
<Options>
Mode = Calibration;
MaxE   = 10000;
Etol   = 1e-4;
Mtol   = 1e-7;
Quadrature   = 25,6.0;
Processors   = 2;
GOF = Extended;
SE = REM;
SaveDBG = Yes;
SaveSCO = Yes;
```

```
Score = EAP;

<Groups>
%G%
File ="fitDINAt.dat";
Varnames = v1-v24
Ncats(v1-v24) = 2;
Model(v1-v24) = Graded(2);

Attributes = 4;
InteractionEffects = (2);  //6 possible 2nd order ints.
Primary = 10;    //4+6 = number of dimensions to DCM
Dimensions = 14;  //total dims, including testlet effects

%D%
Varnames  a1-a4;
DM = G;

<Constraints>
Fix G, (v1-v24),MainEffect;

Free G,(v1,v7,v13,v19),Interaction(1,2);
Free G,(v3,v9,v15,v21),Interaction(1,3);
Free G,(v5,v11,v17,v23),Interaction(1,4);
Free G,(v6,v12,v18,v24),Interaction(2,3);
Free G,(v4,v10,v16,v22),Interaction(2,4);
Free G,(v2,v8,v14,v20),Interaction(3,4);

Free G,(v1-v6),Slope(11);
Free G,(v7-v12),Slope(12);
Free G,(v13-v18),Slope(13);
Free G,(v19-v24),Slope(14);

Equal G,(v1-v6),Slope(11);
Equal G,(v7-v12),Slope(12);
Equal G,(v13-v18),Slope(13);
Equal G,(v19-v24),Slope(14);

Equal D,(a1-a4),Slope(1)
```

8.3.2 Replicating Published Examples

Because users are most likely acquainted with fitting DCMs in other programs, we also provide syntax and output for well-known examples to demonstrate that comparable parameter estimates may be obtained from flexMIRT™ and benefits of either computation-time savings or simplicity of syntax are obtained.

The first example is a replication of Example 9.2 presented in *Diagnostic Measurement: Theory, Methods and Applications* by Rupp et al. (2010). The simulated data presented in this example uses 7 dichotomous items that measure 3 attributes. The Q-matrix given by the authors is duplicated here. With

Table 8.4: Q-matrix for Rupp, Templin, & Henson's Example 9.2

Item	Attribute 1	Attribute 2	Attribute 3
Item 1	1	0	0
Item 2	0	1	0
Item 3	0	0	1
Item 4	1	1	0
Item 5	1	0	1
Item 6	0	1	1
Item 7	1	1	1

the Q-matrix, we are now able to generate our flexMIRT™ DCM syntax.

Example 8-5: Rupp, Templin, & Henson - Example 9.2

```
<Project>
Title = "Rupp Templin Henson Example 9.2";
Description= "Saturated LDCM model - 7 items, 3 attributes";

<Options>
Mode = Calibration;
MaxE   = 20000;
Etol   = 1e-6;
MaxM   = 50;
Mtol   = 1e-9;
GOF = Extended;
```

```
SE = REM;
SaveCOV = Yes;
SavePRM = Yes;
SaveSCO = Yes;
Score = EAP;

<Groups>
%G%
File ="ch9data.dat";
N = 10000;
Varnames = v1-v7,truec;
Select = v1-v7;
Ncats(v1-v7) = 2;
Model(v1-v7) = Graded(2);
Attributes = 3;
InteractionEffects = (2,3); // generate 2nd- and 3rd-order ints.
Dimensions = 7; // 3 main + 3 2nd-order + 1 3rd-order

%D%
DM = G;
Varnames = a;
Ncats(a) = 8;
Model(a) = Nominal(8);
Tc(a) =Identity;

<Constraints>
Fix G,(v1-v7),MainEffect;
Free G,(v1),MainEffect(1);      // 1
Free G,(v2),MainEffect(2);      // 2
Free G,(v3),MainEffect(3);      // 3
Free G,(v4),MainEffect(1);      // 1
Free G,(v4),MainEffect(2);      // 2
Free G,(v4),Interaction(1,2);   // 1x2
Free G,(v5),MainEffect(1);      // 1
Free G,(v5),MainEffect(3);      // 3
Free G,(v5),Interaction(1,3);   // 1x3
Free G,(v6),MainEffect(2);      // 2
Free G,(v6),MainEffect(3);      // 3
Free G,(v6),Interaction(2,3);   // 2x3
Free G,(v7),MainEffect(1);     // 1
Free G,(v7),MainEffect(2);     // 2
Free G,(v7),MainEffect(3);       // 3
Free G,(v7),Interaction(1,2);    // 1x2
Free G,(v7),Interaction(1,3);    // 1x3
Free G,(v7),Interaction(2,3);    // 2x3
Free G,(v7),Interaction(1,2,3);  // 1x2x3

Fix D,(a),Slope;
Fix D,(a),ScoringFn;
Value D,(a),ScoringFn(1),0.0;
```

The items were simulated as correct/incorrect items, so `Ncat=` has been set to 2 and we will fit the 2PLM to all items. As noted in the introduction, there are 3 attributes measured by the items, so `Attributes = 3;`. From the Q-matrix, we can see that Items 4-6 all measure two of the attributes and Item 7 measures all 3; to accommodate this aspect of the item assignment, we ask flexMIRT^{TM} to generate all second and third order effects for us via the `InteractionEffects` command. Finally, we set `Dimensions = 7`, as the total number of dimensions of the model is 3 main effects + 3 second order interactions + 1 third order interaction.

In the `D` group, we inform flexMIRT^{TM} that the diagnostic model we are setting up will be applied to group `G`, using the `DM = G;` statement. Within the `D` group, we specify that the attribute variable will be called `a` and that it has 8 categories, reflecting the 8 possible attribute profiles. Additionally, we tell flexMIRT^{TM} that the attribute variable will be fit with the nominal model and we use an identity matrix for the nominal model intercepts.

In the `<Constraints>` section, we assign items onto attributes according to the Q-matrix. First, we fix all items to 0 and then free items onto the appropriate main effect and interaction terms. For example, from the Q-matrix we see that Item 4 is informed by attributes 1 and 2 and the interaction of those attributes. In the `<Constraints>` section we free Item 4 with the statements `Free G,(v4), MainEffect(1);`, `Free G,(v4), MainEffect(2);`, and `Free G,(v4),Interaction(1,2);`. The final three constraints are applied to the higher-level group, group `D`. The first two statements fix the slope and scoring function values and the final `Value` statement is used to fix the first scoring function value to 0 for model identification purposes.

With the syntax created, we can now run the DCM in flexMIRT$^{\text{TM}}$ and turn to the output.

Output 8.6: Rupp, Templin, Henson Output - Item Parameters

```
Rupp Templin Henson Example 9.2
LDCM model - 7 items, 3 attributes

2PL Items for Group 1: G
    Item              Label  P#    a  1    s.e.  P#    a  2    s.e.  P#    a  3    s.e.  P#    a  4    s.e.  P#    a  5    s.e.  P#    a  6    s.e.  P#    a  7    s.e.  P#       c    s.e.
       1                 v1  15    1.87    0.06        0.00    ----        0.00    ----        0.00    ----        0.00    ----        0.00    ----        0.00    ----   1   -0.91    0.05
       2                 v2        0.00    ----  16    2.04    0.21        0.00    ----        0.00    ----        0.00    ----        0.00    ----        0.00    ----   2   -1.01    0.15
       3                 v3        0.00    ----        0.00    ----  17    2.01    0.14        0.00    ----        0.00    ----        0.00    ----        0.00    ----   3   -0.96    0.10
       4                 v4  18    2.05    0.17  19    1.86    0.20        0.00    ----  20    1.03    0.34        0.00    ----        0.00    ----        0.00    ----   4   -2.44    0.11
       5                 v5  21    1.79    0.33        0.00    ----  22    1.72    0.36        0.00    ----  23    1.38    0.58        0.00    ----        0.00    ----   5   -2.24    0.24
       6                 v6        0.00    ----  24    2.15    0.12  25    2.10    0.11        0.00    ----        0.00    ----  26    1.11    0.28        0.00    ----   6   -2.54    0.05
       7                 v7  27    2.47    0.22  28    2.13    0.26  29    2.06    0.28  30    0.81    0.42  31    0.75    0.44  32    0.98    0.44  33   -1.30    0.84   7   -3.63    0.10

Diagnostic IRT Attributes and Cross-classification Probabilities for Group   1: G

 Pattern         Prob
   0 0 0    0.24810019
   0 0 1    0.07355267
   0 1 0    0.09927395
   0 1 1    0.09119598
   1 0 0    0.08634180
   1 0 1    0.09134911
   1 1 0    0.09358603
   1 1 1    0.21660027

Rupp Templin Henson Example 9.2
LDCM model - 7 items, 3 attributes

GPC Items for Group 2: D
    Item              Label  P#       a    s.e.       b    s.e.     d 1     d 2    s.e.     d 3    s.e.     d 4    s.e.     d 5    s.e.     d 6    s.e.     d 7    s.e.     d 8    s.e.
       1                  a        0.00            ----    ----       0    ----    ----    ----    ----    ----    ----    ----    ----    ----    ----    ----    ----    ----    ----

Rupp Templin Henson Example 9.2
LDCM model - 7 items, 3 attributes

Nominal Model Slopes and Scoring Function Contrasts for Items for Group 2: D
    Item              Label  P#       a    s.e. Contrasts  P# alpha 1    s.e.  P# alpha 2    s.e.  P# alpha 3    s.e.  P# alpha 4    s.e.  P# alpha 5    s.e.  P# alpha 6    s.e.  P# alpha 7    s.e.
       1                  a        0.00    ----     Trend        1.00    ----        0.00    ----        0.00    ----        0.00    ----        0.00    ----        0.00    ----        0.00    ----

Nominal Model Scoring Function Values Group 2: D
    Item              Label     s 1     s 2     s 3     s 4     s 5     s 6     s 7     s 8
       1                  a    0.00    1.00    2.00    3.00    4.00    5.00    6.00    7.00

Nominal Model Intercept Contrasts for Items for Group 2: D
    Item              Label Contrasts  P# gamma 1    s.e.  P# gamma 2    s.e.  P# gamma 3    s.e.  P# gamma 4    s.e.  P# gamma 5    s.e.  P# gamma 6    s.e.  P# gamma 7    s.e.
       1                  a  Identity   8   -1.22    0.52   9   -0.92    0.43  10   -1.00    0.13  11   -1.06    0.31  12   -1.00    0.37  13   -0.97    0.24  14   -0.14    0.05

Original (Bock, 1972) Parameters, Nominal Items for Group 2: D
    Item              Label   Category:       1       2       3       4       5       6       7       8
       1                  a           a    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
                                      c   -0.00   -1.22   -0.92   -1.00   -1.06   -1.00   -0.97   -0.14
```

The output section labeled `2PL Items for Group 1:  G` presents the item parameters for the 7 items, with estimated versus fixed at zero loadings mirroring the Q-matrix specifications, with the factors labeled a1-a3 representing main effects for attributes 1 - 3 and a4 - a7 representing the 2nd and 3rd order interactions. For those wishing to compare flexMIRT^{TM} to MPLUS (the program used to estimate the book example), this item parameter section corresponds with the "New/Additional Parameters" section of MPLUS output and the point estimates agree with those given in Figure 9.15 (pg. 218) of Rupp et al. (2010).

The next section of interest is labeled `Diagnostic IRT Attributes and Cross-classification Probabilities for Group 1:  G`, which reports the estimated proportions of respondents in each attribute profile. This section in flexMIRT^{TM} corresponds to the MPLUS output section labeled `FINAL CLASS COUNTS AND PROPORTIONS FOR THE LATENT CLASSES...` and the reported values match those given in Figure 9.13 of Rupp et al. (2010).

There are some difference between flexMIRT^{TM} and MPLUS. For example, MPLUS fixes the latent variable mean of the last latent class (attribute profile 8 in this case) to 0, while in flexMIRT^{TM} we have constrained the first latent class to 0. This results in differences in the latent attribute parameter estimates, but as we show in the below table these differences are simply a matter of choice of identification constraints and do not affect the primary estimates of interest (i.e., the estimated profile proportions).

Table 8.5: Generating Latent Class Mean Values and flexMIRT^{TM} and MPLUS Estimates for Rupp, Templin, & Henson's Example 9.2

	c1	c2	c3	c4	c5	c6	c7	c8	sum
				generating					
μ	0	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	0	
$\exp(\mu)$	1	0.367879	0.367879	0.367879	0.367879	0.367879	0.367879	1	4.207277
υ	0.2481	0.0913	0.0913	0.0913	0.093	0.0913	0.0913	0.2481	
				flexMIRT					
$\hat{\mu}$	0	-1.21583	-0.91595	-1.00082	-1.05552	-0.99914	-0.97495	-0.13578	
$\exp(\hat{\mu})$	1	0.296464	0.400136	0.367578	0.348011	0.368196	0.377211	0.87303468	4.030631
$\hat{\upsilon}$	0.2481	0.0736	0.0993	0.0912	0.0863	0.0914	0.0936	0.2166	
				MPLUS					
$\hat{\mu}$	0.136	-1.08	-0.78	-0.865	-0.92	-0.863	-0.839	0	
$\exp(\hat{\mu})$	1.145682	0.339596	0.458406	0.421052	0.398519	0.421894	0.432142	1	4.617291
$\hat{\upsilon}$	0.2481	0.0736	0.0993	0.0912	0.0863	0.0914	0.0936	0.2166	

We will next cover the contents of the requested file that contains the individual scores.

Output 8.7: Rupp, Templin, Henson Output - -sco File

```
  1      1      0.162486    0.019853    0.598993  |  0.368896    0.139494    0.490102  |  0.136084   -0.002661    0.019459   -0.018766   -0.008690    0.240200 \\
0.300930    0.517296    0.016153    0.003135    0.083426    0.078495    0.000499    0.000066 2    0.000000    1.000000    1.000000
  1      2      0.041052    0.047476    0.036249  |  0.198411    0.212654    0.186910  |  0.039367   -0.001706    0.045222   -0.001260   -0.001504    0.034935 \\
0.875910    0.035804    0.047016    0.000217    0.040582    0.000228    0.000243    0.000000  1    0.000000    1.000000    1.000000
  1      3      0.198872    0.190374    0.160731  |  0.399151    0.392596    0.367282  |  0.159322   -0.019122    0.154132   -0.024583   -0.024751    0.134896 \\
0.481882    0.147611    0.165898    0.005738    0.172862    0.007272    0.018627    0.000110 1    0.000000    1.000000    1.000000
  1      4      0.041052    0.047476    0.036249  |  0.198411    0.212654    0.186910  |  0.039367   -0.001706    0.045222   -0.001260   -0.001504    0.034935 \\
0.875910    0.035804    0.047016    0.000217    0.040582    0.000228    0.000243    0.000000 1    0.000000    1.000000    1.000000
  1      5      0.475195    0.091060    0.900778  |  0.499384    0.287695    0.298960  |  0.249385   -0.031539    0.082768   -0.008865   -0.007338    0.089377 \\
0.029570    0.415907    0.013637    0.065691    0.053279    0.410184    0.002736    0.008996 2    0.000000    1.000000    1.000000
  1      6      0.880070    0.979322    0.977730  |  0.324880    0.142303    0.147561  |  0.105547   -0.001801    0.020250   -0.001495   -0.000300    0.021774 \\
0.000006    0.000673    0.001170    0.118081    0.000154    0.019844    0.020940    0.839131 8    0.000000    1.000000    1.000000
  1      7      0.981067    0.979636    0.846212  |  0.136289    0.141243    0.360745  |  0.018575   -0.000279    0.019950   -0.001603   -0.002010    0.130137 \\
0.000007    0.000100    0.001301    0.017525    0.001116    0.019142    0.151364    0.809445 8    0.000000    1.000000    1.000000
  1      8      0.150848    0.679496    0.012669  |  0.357900    0.466670    0.111843  |  0.128093   -0.034142    0.217781   -0.001397   -0.005747    0.012509 \\
0.228668    0.009347    0.608329    0.002808    0.082029    0.000460    0.068305    0.000054 3    0.000000    1.000000    1.000000
  1      9      0.751563    0.081414    0.638255  |  0.432107    0.273469    0.480505  |  0.186716   -0.029184    0.074785    0.013218   -0.028909    0.230886 \\
0.071059    0.127968    0.032030    0.017380    0.232326    0.487234    0.026330    0.005674 6    0.000000    1.000000    1.000000
  1     10     0.143402    0.935046    0.910959  |  0.350483    0.246444    0.284802  |  0.122838   -0.025556    0.060735   -0.031276    0.000838    0.081112 \\
0.001455    0.028628    0.043541    0.782973    0.005167    0.029704    0.038878    0.069654 4   -0.000000    1.000000    1.000000
```

As in the previous -sco output, we have given the first 10 cases and have added a line break and separators. The first two columns of the -sco file give the group and observation number, similar to a standard flexMIRT^{TM} IRT score output file. Recall that for this example, there are 3 attributes and 8 possible attribute profiles. The probability of the respondent having mastered each of the attributes individually is given, starting after the observation number. For example, for observation 1, the probability they have mastery of attribute 1 is 0.16, for attribute 2 it is 0.02 and for attribute 3 it is 0.60. The next three columns are the estimated SEs for those probabilities, respectively. The last six entries on the first line of each observation contain the unique elements of the estimated attribute variance/covariance matrix.

Finally, on what has been moved to the second line for each respondent (but is a continuation of the same in the actual -sco file), the posterior probabilities for each of the attribute profiles (8 possible for this example) are given. For observation 1, the posterior probability of being in attribute profile 1 is 0.30, the probability of being in attribute profile 2 is 0.52 and so on. The order in which the probabilities are given matches the order used in the output table "Diagnostic IRT Attributes and Cross-classification Probabilities." Referring to the output presented for this example, we find that attribute profile 1 corresponds to 0 0 0 (meaning no attributes are mastered) and attribute profile 2 corresponds to 0 0 1 (meaning only attribute 3 has been mastered), etc.. After the probabilities for each of the attribute profiles are presented, the most likely attribute profile for the respondent is given; for respondent 1 that is attribute profile 2, for respondents 6 and 7, that is attribute profile 8. The last three values given in the -sco file are the higher-order latent variable point estimate followed by the SD and variance for that estimate. In this case, all individual theta estimates for the higher-order variable are zero and all SD and variance estimates are 1.00 because the nominal model was fit, which has zero slope and is a reparameterization of the profile probabilities.

Unstructured Higher-Order Latent Space

It is also possible to the fit DCMs in flexMIRT^{TM} without using the higher-order latent variable at all. This is accomplished by removing all references to the D group in setting up the model. We present this syntax below. The only changes are the removal of the `%D%` group from the `<Groups>` section as well as omitting the parameter constraints that were applied to that group.

Example 8-6: Rupp, Templin, & Henson - Example 9.2 without D group

```
<Project>
Title = "Rupp Templin Henson Example 9.2";
Description= "Saturated LDCM model - 7 items, 3 attributes";

<Options>
Mode = Calibration;
MaxE  = 20000;
Etol  = 1e-6;
MaxM  = 50;
Mtol  = 1e-9;
GOF = Extended;
SE = REM;
SaveCOV = Yes;
SavePRM = Yes;
SaveSCO = Yes;
Score = EAP;

<Groups>
%G%
File ="ch9data.dat";
N = 10000;
Varnames = v1-v7,truec;
Select = v1-v7;
Ncats(v1-v7)  = 2;
Model(v1-v7) = Graded(2);
Attributes = 3;
InteractionEffects = (2,3); // generate 2nd- and 3rd-order ints.
Dimensions = 7;// 3 main + 3 2nd-order + 1 3rd-order

<Constraints>
Fix G,(v1-v7),MainEffect;
Free G,(v1),MainEffect(1);     // 1
Free G,(v2),MainEffect(2);     // 2
Free G,(v3),MainEffect(3);     // 3
Free G,(v4),MainEffect(1);     // 1
Free G,(v4),MainEffect(2);     // 2
Free G,(v4),Interaction(1,2);  // 1x2
Free G,(v5),MainEffect(1);      // 1
Free G,(v5),MainEffect(3);      // 3
Free G,(v5),Interaction(1,3);   // 1x3
Free G,(v6),MainEffect(2);      // 2
Free G,(v6),MainEffect(3);      // 3
Free G,(v6),Interaction(2,3);   // 2x3
Free G,(v7),MainEffect(1);      // 1
Free G,(v7),MainEffect(2);      // 2
Free G,(v7),MainEffect(3);      // 3
Free G,(v7),Interaction(1,2);   // 1x2
Free G,(v7),Interaction(1,3);   // 1x3
Free G,(v7),Interaction(2,3);   // 2x3
Free G,(v7),Interaction(1,2,3);// 1x2x3
```

With respect to the output, presented next, while there are some slight changes in the estimates, the reported point estimate values for the item parameters remain largely unchanged from the calibration that employed the higher-order latent variable. Differences in the estimated profile probabilities are not seen until the 4th or 5th decimal place. If one looked at the -sco file, there are also only slight changes to the reported values; the most noticeable difference between the -sco file from the model employing the higher-order variable and the current example is that final three values discussed in the previous -sco file (the higher-order latent variable point estimate, SD, and variance estimates) are no longer present.

Output 8.8: Rupp, Templin, Henson Output - Item Parameters without D group

```
2PL Items for Group 1: G
    Item              Label   P#    a  1    s.e.   P#    a  2    s.e.   P#    a  3    s.e.   P#    a  4    s.e.   P#    a  5    s.e.   P#    a  6    s.e.   P#    a  7    s.e.   P#       c    s.e.
       1                 v1    8    1.87    0.07         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----    1   -0.91    0.04
       2                 v2         0.00    ----    9    2.04    0.07         0.00    ----         0.00    ----         0.00    ----         0.00    ----         0.00    ----    2   -1.01    0.04
       3                 v3         0.00    ----         0.00    ----   10    2.01    0.07         0.00    ----         0.00    ----         0.00    ----         0.00    ----    3   -0.96    0.04
       4                 v4   11    2.05    0.18   12    1.86    0.18         0.00    ----   13    1.03    0.31         0.00    ----         0.00    ----         0.00    ----    4   -2.44    0.13
       5                 v5   14    1.79    0.17         0.00    ----   15    1.72    0.17         0.00    ----   16    1.38    0.32         0.00    ----         0.00    ----    5   -2.24    0.11
       6                 v6         0.00    ----   17    2.15    0.19   18    2.10    0.20         0.00    ----         0.00    ----   19    1.11    0.37         0.00    ----    6   -2.54    0.14
       7                 v7   20    2.47    0.57   21    2.13    0.58   22    2.06    0.64   23    0.81    0.80   24    0.75    0.85   25    0.98    0.85   26   -1.30    1.32    7   -3.63    0.43

Diagnostic IRT Attributes and Cross-classification Probabilities for Group   1: G

 Pattern          Prob
   0 0 0    0.24809775
   0 0 1    0.07355411
   0 1 0    0.09927471
   0 1 1    0.09119567
   1 0 0    0.08634185
   1 0 1    0.09134873
   1 1 0    0.09358578
   1 1 1    0.21660141
```

de la Torre Subtraction Example

In a demonstration of a higher order latent trait DCM to non-simulated data, de la Torre and Douglas (2004) fit a higher-order DINA model to a 20-item fraction subtraction test given to 2144 examinees. In the noted paper, an 8 attribute model was estimated using MCMC, with final parameter estimates based on averaging the estimates from 10 parallel chains, each with 20000 iterations of which the first 10000 were discarded as burn-in. We provide syntax and output of a flexMIRT^{TM} replication of this analysis, using a publicly available subset of the noted data (N = 536).

Example 8-7: de la Torre Subtraction Syntax

```
<Project>
Title = "Tatsuoka Subtraction Data";
Description= "Restricted Higher-order DINA Model";

<Options>
Mode = Calibration;
MaxE   = 20000;
MaxM   = 5;
Mtol   = 0;
Etol   = 1e-5;
GOF = Extended;
SE = REM;
SaveCOV = Yes;
SavePRM = Yes;
SaveSCO = Yes;
Score = EAP;
Processors = 4;
NewThreadModel = Yes;

<Groups>
%G%
File ="subtraction.csv";
Varnames = v1-v20;
Ncats(v1-v20) = 2;
Model(v1-v20) = Graded(2);

```

```
Attributes = 8;
Generate = (4,6,7),(4,7),(2,3,5,7),(2,4,7,8),(1,2,7),(2,5,7,8),
              (2,5,7),(7,8),(2,4,5,7), (2,7),(1,7),
              (2,5,6,7),(1,2,3,5,7),(2,3,5,7);
Dimensions = 22; // 8 MEs + 14 generated higher ord ints

%D%
Varnames  = a1- a8;
DM = G;

<Constraints>
FixG, (v1-v20),MainEffect;
Free G,(v1),Interaction(4,6,7); //3rd-order int of attr 4,6,7
Free G,(v2),Interaction(4,7); //2nd-order int of attr 4,7
Free G,(v3),Interaction(4,7); //2nd-order int of attr 4,7
Free G,(v4),Interaction(2,3,5,7);//4th-order int of 2,3,5,7
Free G,(v5),Interaction(2,4,7,8);//4th-order int of 2,4,7,8
Free G,(v6), MainEffect(7); //main effect of attr 7
Free G,(v7),Interaction(1,2,7); //3rd-order int of attr 1,2,7
Free G,(v8), MainEffect(7); // main effect of attr 7
Free G,(v9), MainEffect(2); // main effect of attr 2
Free G,(v10),Interaction(2,5,7,8); //4th-ord int of 2,5,7,8
Free G,(v11),Interaction(2,5,7); //3rd-order int of attr 2,5,7
Free G,(v12),Interaction(7,8); // 2nd-order int of attr 7,8
Free G,(v13),Interaction(2,4,5,7); //4th-ord int of 2,4,5,7
Free G,(v14),Interaction(2,7); //2nd-order int of attr 2,7
Free G,(v15),Interaction(1,7); //2nd-order int of attr 1,7
Free G,(v16),Interaction(2,7); //2nd-order int of attr 2,7
Free G,(v17),Interaction(2,5,7); //3rd-order int of attr 2,5,7
Free G,(v18),Interaction(2,5,6,7); //4th-ord int of 2,5,6,7
Free G,(v19),Interaction(1,2,3,5,7);//5th-ord int att 1,2,3,5,7
Free G,(v20),Interaction(2,3,5,7); //4th-order int 2,3,5,7

Equal D,(a1-a8),Slope; // "restricted" higher order model
```

As with the previous DCM example, the maximum number of E-steps has been increased from the default, the E tolerance and M tolerance values have been decreased, and the SE calculation method has been set to REM in the `<Options>` section. As noted earlier, these changes are made primarily to improve the resulting SE estimates. We have also requested the newly implemented multi-core processing model be used for this analysis via `NewThreadModel = Yes;`. This model is requested due to the high-dimensional nature of the current problem. Additional information regarding the new thread model is available in the Details of the Syntax chapter.

In the first group section, we have used the more targeted `Generate` to construct only those interaction terms which will be used in the model. The needed interaction effects are generated in the order they will be used for the items (e.g., item 1 requires the interaction of attributes 4, 6, and 7) but this is not necessary - as noted earlier, as many interactions as needed may be constructed from a single `Generate` statement and the order in which interactions are listed does not matter. Counting the interactions specified in the `Generate` statement, we find there are 14 interactions needed to correctly assign items to correspond to the Q-matrix. Adding the 14 interactions with the 8 main effects gives us a total of 22 latent dimensions in the model.

In the `<Constraints>` section, all main effect terms are initially fixed to 0 and then `Free` statements are added to replicate the Q-matrix for the problem (which may be found in Table 8 on pg. 347 of de la Torre & Douglas, 2004). Because a DINA is fit to all items, only the highest-order interaction of attributes for each item is specified and main effect terms are specified for only those items that depend on a single attribute (i.e., items 6, 8, and 9). The last statement in the `<Constraints>` section sets the latent attribute slope parameters to be equal across attributes, producing a restricted higher-order model.

As with all examples, the full output file is available on the support pae. The item parameter table for this example is extremely wide (22 latent dimension columns, an intercept column, and all associated SEs) and is not presented here due to size. However, we have summarized the flexMIRT$^{\text{TM}}$ parameter estimates in Table 8.6. Please note that the slopes are presented in a single column in this table for efficiency of space; the estimated slope values are not all on the same latent dimension, as can be seen in the full output file. In addition to the flexMIRT$^{\text{TM}}$ estimates, we also present the original slippage and guessing parameter values reported by de la Torre and Douglas (2004) in their Table 9 and the flexMIRT$^{\text{TM}}$ estimates converted to this parameterization. Comparing the point estimates across the two programs, one notices some differences but overall the values are quite similar. Regressing the original guessing parameter estimates onto the converted flexMIRT$^{\text{TM}}$ values gives an intercept value close to zero (0.004), a regression coefficient on 0.98, and an R^2 value of 0.99. A similar comparison of the slippage values also results in a low intercept (0.004), a regression coefficient near 1 (0.99), and a high R^2 value (0.99). Given the comparability of the parameter values, it is interesting to note that flexMIRT^{TM} completed its calibration in 20.5 seconds, highlighting the extreme efficiency that may be obtained when estimating higher-order DCMs via MML.

Table 8.6: Guessing and Slippage Parameters from flexMIRT and MCMC

		flexMIRT			de la Torre & Douglas	
Item	Intercept (c)	Slope (a)	g	1 - s	g	1 - s
v1	-3.27	5.42	0.04	0.90	0.04	0.90
v2	-3.46	6.72	0.03	0.96	0.03	0.96
v3	-5.60	7.56	0.00	0.88	0.00	0.88
v4	-1.25	3.31	0.22	0.89	0.22	0.89
v5	-0.83	2.36	0.30	0.82	0.30	0.82
v6	-4.68	7.77	0.01	0.96	0.03	0.96
v7	-3.56	4.97	0.03	0.80	0.03	0.81
v8	-0.22	1.67	0.45	0.81	0.44	0.81
v9	-1.51	2.63	0.18	0.75	0.18	0.75
v10	-3.50	4.80	0.03	0.79	0.03	0.79
v11	-2.67	5.18	0.07	0.93	0.07	0.93
v12	-1.93	5.00	0.13	0.96	0.13	0.96
v13	-4.14	4.84	0.02	0.67	0.02	0.67
v14	-3.04	5.71	0.05	0.94	0.05	0.93
v15	-3.41	5.58	0.03	0.90	0.04	0.90
v16	-2.19	4.21	0.10	0.88	0.10	0.88
v17	-3.08	4.92	0.04	0.86	0.04	0.86
v18	-1.95	3.70	0.12	0.85	0.13	0.85
v19	-3.76	4.89	0.02	0.76	0.02	0.76
v20	-4.31	5.95	0.01	0.84	0.01	0.84

Note: g is found from flexMIRT parameters as $exp(c)/(1+exp(c))$ and $1-s$ is found as $exp(c+a)/(1+exp(c+a))$.

CHAPTER 9

Details of the Syntax

This section provides detailed coverage of all the syntax statements and associated options that are available in flexMIRT^{TM}. Required statements are listed in bold, optional statements are in normal font. When keywords are required with statements, the possible options are listed after the statement and the default setting is underlined. If numeric values are required with a statement, such as for a convergence criterion, the default value is presented. If there is no default value, a question mark will denote where values need be entered.

9.1. The Project Section

Syntax Display 9.1: `<Project>` section commands

```
Title = "  ";
Description  = "  ";
```

Both of these statements must be present in the `<Project>` section, although the actual content within the quotes may be left blank.

9.2. The Options Section

The `<Options>` section is where the type of analysis to be conducted is specified and where technical details of the analysis may be modified, including convergence criteria, scoring methods, and the level of detail desired in the output. As indicated by the bold, the `Mode` statement is the only required command and determines the type of analysis that will be conducted. If `Mode = Scoring;` or `Mode = Simulation;` is selected, some additional technical commands, covered in the next group of statements become required. The setting of Progress determines if flexMIRT^{TM} will print detailed progress information in the console window (e.g., iteration number and log-likelihood value). TechOut determines if controls values such as tolerance values, processing times, names of outputted files, etc. will be printed in the preamble of the output. NumDec determines the number of decimal places reported for item parameters in the output file - the default is 2. The other commands allow for the optional saving of additional output into separate files. SavePRM, the item and group parameter estimates; SaveSCO, the individual IRT scale scores, SaveCOV refers to the covariance matrix of the parameter estimates; SaveINF the Fisher information function values of the items and the test; SaveICC, category response probabilites for items and the overall test **for unidimensional models**; SaveDBG, additional technical information; SavePCC, the eigenvalues of the polychoric correlations matrix, followed by the unique elements of the polychoric correlation matrix, followed by a table of item thresholds, and SaveSSP, the normalized summed score posteriors. Note that unless summed scores (`Score = SSC;`) or the full gamut of GOF indices (`GOF = Complete;`) are requested, the SSP output file will be blank. Each of the these additional output requests are saved to files with corresponding extensions; that is, the SaveINF command results in an output file with "*-inf.txt" appended to the command file name; SaveCOV appends "*-cov.txt", and so on. With the SaveEtbl command, users may request that the tables produced by the last E-step of the Bock-Aitkin expectation-maximization algorithm be saved to the -dbg output file; note that SaveDBG = Yes; must also be specified to obtain this additional reporting.

Syntax Display 9.2: `<Options>` - Engine Mode and Output Statements

```
<Options>
Mode= Calibration/Scoring/Classical/Simulation;
Progress  =  Yes/No;
TechOut  = Yes/No;
NumDec = 2;
SavePRM = Yes/No;
SaveSCO = Yes/No;
SaveCOV = Yes/No;
SaveINF = Yes/No;
SaveICC = Yes/No;
SaveDBG = Yes/No;
```

```
SaveEtbl = Yes/No;
SavePCC = Yes/No;
SaveSSP = Yes/No;
FactorLoadings  = Yes/No;
DMtable =  Yes/No;
NormalMetric3PL = Yes/No;
SlopeThreshold = Yes/No;
```

The `FactorLoadings` statement is invoked when, in addition to the IRT parameters, normal metric factor loadings should be printed as part of the general output file. The factor loading values are converted from the IRT parameters, as described in Wirth and Edwards (2007), among others.

Specific to DCMs, the command `DMtable` controls whether the diagnostic classification probability table is printed in the *-irt file. The default is to print the table, but this may be modified by changing the command to `DMtable= No;`. Modifying this setting may be helpful if the probability table will be large.

The final two statements in this section are provided primarily to facilitate the use of parameter estimates obtained from other programs, but users may find them convenient for other purposes. Both `NormalMetric3PL` and `SlopeThreshold` are used to print, save out, or read-in parameters that are in more traditional IRT metrics than the MIRT-compatible logistic metric flexMIRT$^{\text{TM}}$ uses; these keywords are only available for use in conjunction with unidimensional models.

When `NormalMetric3PL = Yes;`, the program will print normal-metric parameter estimates in the output and save those values into the -prm file, if requested. Specifically, flexMIRT$^{\text{TM}}$ will 1) print/save $(a/1.702)$ rather than the typical logistic metric slope values, 2) instead of the intercept $(c = b * -a)$ flexMIRT$^{\text{TM}}$ will print/output the difficulty (b) parameter value, and 3) rather than the logit-guessing parameter, the program will report/save the customary g value. Note that even if the normal metric keyword is used during calibration, any priors that may be specified are applied to the MIRT-compatible metrics (prior applied to intercept, not difficulty; normal guessing prior still applied to logit-g distribution, etc). If `NormalMetric3PL` is set to `Yes` for a scoring run, then flexMIRT$^{\text{TM}}$ will expect the all parameter values read-in from the -prm file for scoring to be in the normal metric.

By default, flexMIRT$^{\text{TM}}$ prints both the intercept and difficulty parameters in the output for unidimensional models, but saves only the intercept values into a requested -prm file. When `SlopeThreshold = Yes;`, rather than saving the intercept values, the program will write the difficulty (b) values to the -prm file. If `SlopeThreshold` is set to `Yes` in conjunction with a 3PL model, the psuedo-guessing parameters (gs) will be printed/saved, rather than the default logit-guessing parameter values. Users should be aware that, unlike the behavior when `NormalMetric3PL` is used, the `SlopeThreshold` keyword has no effect on the metric of the slope; a values printed to the -prm are in the logistic metric, regardless of the `SlopeThreshold` setting.

Syntax Display 9.3: `<Options>` - Technical Specifications

```
<Options>
 ...
Score =  EAP/MAP/ML/SSC/MI;
MaxMLscore = ?;
MinMLscoree = ?;
Mstarts =  Yes/No;
Rndseed =?;
ReadPRMFile = "*.txt";
SE = Xpd/Mstep/SEM/Sandwich/Fisher/FDM/REM;
SmartSEM = Yes/No;
Perturbation = 0.001;
FisherInf = 1, 0.0;
PriorInf = Yes/No;
logDetInf = Yes/No;
Quadrature = 49, 6.0;
MaxE = 500;
MaxM = 100;
Etol = 1e-4;
Mtol = 1e-9;
SEMtol = 1e-3;
SStol = 1e-4;
Processors = 1;
NewThreadModel = Yes/No;
SparseMatrix = Yes/No;
```

As stated previously, when `Mode` is set to something other than `Calibration`, additional statements are required. It was noted earlier that specifying `SaveSCO = Yes;` during a calibration run triggers a combined scoring run. If a scoring run is specified, either by explicitly setting `Mode` to `Scoring` or implicitly by requesting a combined scoring run, an IRT scoring method is required. As may be deduced from the lack of an underlined scoring option, there is no default. The options available for scoring are Expected *A Posteriori* (keyword is `EAP`), Maximum *A Posteriori* (keyword is `MAP`), Maximum Likelihood (keyword is `ML`), Multiple Imputation (keyword is `MI`) which is only available when `Algorithm = MHRM;` or `Algorithm = MCMC;`, and Summed Score Conversions to EAP scores (keyword is `SSC`), which in addition to individual scores provides a conversion table in the -sco output file that translates summed scores to EAP scale scores.

When using ML scoring, there are two additional commands that must be employed. Users are required to specify the desired maximum and minimum scores to be assigned by flexMIRT^{TM} using the `MaxMLscore` and `MinMLscore` keywords, respectively. There are no default values for the ML minimum and maximum scores. **Efforts have been made to make ML scoring as robust as possible, but it is not recommended for use with multidimensional models. Because ML scoring does not use information from the population distribution, essential statistical information about the population distribution (e.g., factor inter-correlations, means, and variances) is ignored when scoring individual dimensions of MIRT models. Additionally, ML scoring can lead to score information matrices that are not positive definite, making the SEs for some estimates undefined.**

flexMIRT^{TM} offers a multiple dispersed starting value option for use with ML and MAP scoring runs. For unidimensional models, the default theta starting value in MAP/ML scoring is 0. When `Mstarts = Yes;` flexMIRT^{TM} will do 12 additional restarts using a range of starting values from -2.75 to 2.75, in step sizes of 0.5. The best solution (as determined by the highest log-likelihood) will be picked from the 13 values. Note that using `Mstarts = Yes;` will increase the time needed to complete scoring, but provides protection against local modes in 3PL or nominal model scoring.

When `Score = MI;`, the value of `Imputations` controls the number plausible values drawn from individual theta posteriors and saved to the -sco file. Again, note that `Score = MI;` is only available when `Algorithm = MHRM` or `Algorithm = MCMC`.

When a `Simulation` run is specified by the `Mode` command, the random number generator must be "seeded". An integer value must be explicitly provided using the `Rndseed` command. There is no default value.

The `ReadPRMFile` command is used to read parameters values from a text file to flexMIRT$^{©}$, which may be used as starting values for a calibration run, as fixed item parameters for a scoring run, and as true generating parameter values for a simulation run. The proper layout for a parameter file is covered in detail in the previous Simulation chapter.

flexMIRT$^{\text{TM}}$ has several options available for computing the standard errors of estimated item/group parameters. The standard error method may be changed by the user through the `SE` command. The default method is empirical cross-product approximation (keyword is `Xpd`). Standard errors may also be calculated via the supplemented EM algorithm (keyword is `SEM`; Cai, 2008), from the Fisher (expected) information matrix (keyword is `Fisher`), via the sandwich covariance matrix (keyword is `Sandwich`), using the forward difference method (keyword is `FDM`; e.g., Jamshidian & Jennrich, 2000), or from the Richardson extrapolation method (keyword is `REM`; e.g., Jamshidian & Jennrich, 2000). `Mstep` values from the EM algorithm can be requested although, technically, the M-step standard errors are incorrect. They tend to be too small, but under some circumstances (e.g., huge calibration sample size and thousands of item parameters), it may be more efficient to bypass the more elaborate (and correct) standard error computations and focus only on item parameter point estimates.

As an additional note, three of the available standard error methods have additional options that may be adjusted by the user. By default the supplemented EM algorithm SEs come from an optimized window of iterations. If you find that the SEs are failing to converge in that optimized window, the optional command `SmartSEM` may be useful. By setting `SmartSEM = No;` it will allow flexMIRT$^{\text{TM}}$ to use the full EM iteration history, which may lead to converged standard errors. The `Perturbation` command controls a value used by both the forward difference and Richardson extrapolation methods of SE estimation. These methods require the selection of a perturbation constant that is used in the denominator of the finite difference approximation; the choice of this perturbation constant may exert substantial influence on the accuracy of the final item parameter SE estimates (Tian, Cai, Thissen, & Xin, 2013). As noted above, the flexMIRT$^{\text{TM}}$ default of this perturbation constant is set to 0.001, the value found to produce the consistently least poor estimates from the two methods when studied in both simulations and as applied to real data (Tian et al., 2013). While the forward difference and Richardson extrapolation methods are available to users, Tian et al. (2013) found them to be less accurate than the supplemented EM algorithm SEs when applied to standard IRT models, with no noticeable savings with respect to time when the smart window for the supplemented EM SEs was employed.

For undimensional models, the information function for the individual items and full scale is printed in the general output (i.e., the "*-irt.txt" file from calibration, the "*-ssc.txt" file from scoring) using theta values from -2.8 to 2.8, changing by steps of 0.4. The `SaveINF` option allows the user to output the information function values to a separate "*-inf.txt" file, which may be useful for additional computations external to flexMIRT^{TM} or when information functions are to be plotted. By default, the "*-inf.txt" file has only one theta value (at 0.0) saved. The `FisherInf` command allows the number of points and the symmetric minimum/maximum values of those points to be modified. For instance, if information values were desired for 81 points with theta values ranging from -4.0 to 4.0, the command to obtain this output, in conjunction with `SaveINF = Yes;`, would be `FisherInf = 81, 4.0;`, with the total number of points listed first and the maximum theta value to be used listed after comma. Using the `PriorInf` command, the user controls if flexMIRT^{TM} includes the contribution from the prior in the test information output - the default setting of `Yes` includes the prior contribution.

Note that for multidimensional models when `SaveINF = Yes;`, flexMIRT^{TM} will try to print, by default, the trace of the Fisher information matrix calculated at every grid point as defined by the direct product of the points specified in the `FisherInf` command. For high dimensional models, this is not advisable and may result in flexMIRT^{TM} abandoning the analysis due to exhausting all available memory resources when attempting to complete the calculations. If the `logDetInf = Yes;` command is used, the information will be taken as the log determinant of the Fisher information matrix, rather than the trace.

The `Quadrature` command, which can modify the number of quadrature points used in the E step of the EM algorithm, is specified in a similar fashion as the `FisherInf` command. The default quadrature rule has 49 rectangular points over -6 to +6.

The next several statements listed within the technical specifications deal with convergence criteria values and maximum numbers of iterations. `MaxE` determines the maximum allowed number of E-steps in the EM algorithm, `MaxM` the number of allowable iterations in each of the iterative M-steps. `Etol` and `Mtol` set the convergence criteria for the E- and M- steps, respectively. `SEMtol` sets the convergence criterion for the Supplemented EM algorithm used in calculating standard errors, and finally `SStol` determines the criterion for declaring two summed scores as equivalent (useful for scoring with weights).

There are several optional command statments that users may find valuable when conducting large analyses, such as batch runs or individual analyses with high-dimensional models, a large sample, and/or a large number of items to calibrate/score. The `Processors` statement lets the user determine how many processors (threads) flexMIRT^{TM} can utilize; the maximum number of possible threads is determined by the number of processing cores in your system. When `NewThreadModel = Yes;` a multi-core processing model is employed. The new multi-threading model is an updated implementation of multi-core processing that may be more efficient for very high-dimensional models with a large number of quadrature points to evaluate. The level of granularity is smaller than the legacy threading model implemented in flexMIRT$^{©}$. In general, the legacy threading model trades more memory for speed, and this new model does not do that nearly as much, so the parallel speed-up may not be as significant as the classical approach for low dimensional analyses with small to moderate N. When the `NewThreadModel` is active, `Fine (may be more optimal for high dimensionality)` will be printed in the -irt output file to describe the `Parallelization granularity`, while it will be reported as `Coarse` when the original threading model is used. Note that the NewThreadModel option is specific to `Algorithm = BAEM` and will have no effect on analyses using MCMC or MH-RM estimation. The command `SparseMatrix` is implemented primarily for use in large-scale testing situations - when `SparseMatrix = Yes;` is used in conjunction with the default cross-product approximation method for estimating SEs, flexMIRT^{TM} will assume a large and sparse information matrix, leading to time-savings for large calibration runs. When `SparseMatrix = Yes;` is used <u>in combination with the new multi-threading model</u>, parallel speed-up for very large N and many (read, thousands of) items may be much more significant than the legacy model, due to additional optimizations that improve memory utilization and work-sharing efficiency. The `SparseMatrix` option has no effect on models with Primary, Between, and/or Cluster keywords in the current implementation. The `SparseMatrix = Yes;` setting will be indicated in the -irt file on the line reporting the standard error computation algorithm as: `Cross-product (sparse matrix approximation)`. User are encouraged to experiment with the threading options.

Another subset of commands available in the `<Options>` section deals with the calculation and reporting of various GOF and local dependence (LD) indices.

Syntax Display 9.4: `<Options>` - GOF and LD Indices Commands

```
<Options>
...
GOF =  Basic/Extended/Complete;
CTX2Tb = Yes/No;
LDTblWidth = 10;
SX2Tbl = Yes/No;
MinExp = 1.0;
M2 = None/Full/Ordinal/OrdinalSW/Mixed;
HabermanResTbl = Yes/No;
JSI = Yes/No;
FitNullModel = Yes/No;
StartValFromNull = Yes/No;
...
```

The `GOF` statement controls the extent of GOF indices that are reported. `Basic`, `Extended`, and `Complete` are the three possible levels of GOF reporting. `Basic`, the default value, prints the $-2\times$ log likelihood, the AIC/BIC values, and when appropriate, the likelihood ratio (G^2) and Pearson X^2 full-information fit statistics. The `Extended` option results in all the indices included in the `Basic` reporting, as well as the marginal fit X^2 for each item and standardized LD X^2 for each item pair (e.g., Chen & Thissen, 1997). The last keyword, `Complete`, prints all indices included in the `Extended` reporting and also includes, per group, the item-fit index $S-X^2$ (Orlando & Thissen, 2000). Additionally, under `GOF = Complete;` mode, an additional statistic $S-D^2$ is printed below the SSC to EAP conversion table. This statistic is distributed approximately as a central X^2 variable, with degrees of freedom equal to [(number of summed scores - 1) - 2], and provides a test of the latent variable distribution normality assumption.

In both the `Extended` and `Complete` GOF modes, the standardized LD X^2 statistics for each item pair may be suppressed by setting the `CTX2Tbl` command to `No`. As noted in the discussion of the LD statistic values of Chen and Thissen (1997) in previous chapters, phi correlations are examined to determine if difference between the observed and model implied item-correlation is positive or negative. When polytomous data are involved, there may be times that the contingency table of two items needs to be collapsed from the edges to the middle so that the cells are not too sparse. Even if collapsing occurs for the Chen-Thissen LD calculations, the phi correlations (determining the "p" or "n" after each LD value), will be based on the original (non-collapsed) tables. The `LDTblWidth` statement is used to control how many variables wide the LD table will be, with the default value set to 10. If processing the Chen-Thissen LD matrix values for additional review, users may wish to set `LDTblWidth` equal to the total number of analyzed variables to obtain the full Chen-Thissen LD value matrix that is not broken across multiple sections in the -irt output file.

The `SX2Tbl` command is invoked when the user wishes to be able to examine the detailed item fit tables used in the calculation of the $S - X^2$ statistics. As described in Orlando and Thissen (2000), when expected frequencies for responses become small, the accuracy of the X^2 approximation deteriorates. flexMIRT^{TM} employs an algorithm to evaluate the expected probabilities prior to final calculations and collapse the table when needed. The `MinExp` command allows the user to set the value at which expected counts are considered too small, (the default value for `MinExp` is set at 1.0) and when expected counts lower than the value set by `MinExp` are encountered, the summed score groups are collapsed towards the center.

The `M2` statement requests that the limited-information fit statistics M_2, first introduced by Maydeu-Olivares and Joe (2005) (`Full`) and expanded in Maydeu-Olivares, Cai, and Hernandez (2011) (`OrdinalSW`), Cai and Hansen (2013) (`Ordinal`), and Monroe and Cai (2015) (`Mixed`) be printed. The `None` option suppresses M_2. It should be noted that, regardless of `M2` request, the M_2 values will not be printed unless `GOF` is set to something other than `Basic`. Computationally, users should also be aware that in one of the last steps of computing the M_2, a matrix quadratic form is computed. It is accumulated in flexMIRT^{TM} with multiple threading enabled (i.e., when `Processors` is set to a value greater than 1) to reduce the amount of time required. This means, however, that the numbers are combined in potentially indeterminate order if multiple threads are used. Typically this shouldn't be a problem, but sometimes the tiny numerical differences can accumulate and lead to lack of precision large enough (when the resulting matrices are further inverted) that differences in the final value are noticeable. User's encountering such a situation are encouraged to use `Processors = 1;` to obtain stable and precise

M_2 values.

The command `HabermanResTbl` is invoked when standardized residuals for each response pattern, sometimes referred to as Haberman Residuals (see Haberman, 1979), are desired. In addition to the standardized residuals, the observed and expected frequencies, as well as the associated EAPs and SD for each response pattern are also reported in the requested table.

There is an additional LD index that flexMIRT^{TM} can compute with the command `JSI`. The statement `JSI = Yes;` requests that the Jackknife Slope Index (JSI; Edwards & Cai, 2011), a newer LD detection technique, be calculated. Based on the observation that locally dependent items often exhibit inflated slopes, the JSI procedure finds, for each item pair, the change in the slope parameter of item j when item k is removed from the scale. Specifically, a single JSI values is obtained by

$$JSI_{j(k)} = \frac{a_j - a_{j(k)}}{se\big(a_{j(k)}\big)}, \tag{9.1}$$

where a is the IRT slope estimate, j indexes the item impacted and k indexes the removed item and $se\big(a_{j(k)}\big)$ is the SE of the item-removed slope parameter. The resulting n item by n item matrix, with empty diagonals, is inspected by the user and item pairs with JSI values substantially larger than the other values should be noted as possibly exhibiting LD.

The final two statements of this group are tied to the Null (independence) model, needed for incremental fit indices. The `FitNullModel` command, when invoked with the keyword `Yes`, tells flexMIRT^{TM} to fit the Null model and print the resulting basic GOF measures in the general output file. When the fitting of the Null model is requested, it becomes possible for the Tucker-Lewis Index to be calculated. Additionally, when the zero-factor Null model is requested, the resulting parameter values from fitting this model may be used as starting values for the IRT model of interest by supplying the command, `StartValFromNull = Yes;`. Obviously, it is necessary that the Null model is requested, via the `FitNullModel` command, for the `StartValFromNull` command to be functional.

Within the <Options> section, there are three commands related to the implementation of DIF analyses. The first command is a request for a DIF analysis, `DIF`. By default, no DIF analysis is conducted, but this may be modified by the user to request either a comprehensive DIF sweep (keyword is `TestAll`), testing all items, or a focused DIF analysis which looks for DIF only in the candidate items supplied by the user (keyword is `TestCandidates`).

Syntax Display 9.5: <Options> - DIF Analysis Commands

```
<Options>
...
DIF =  None/TestAll/TestCandidates;
DIFcontrasts = (...);
OrthDIFcontrasts = Yes/No;
```

If the `TestCandidates` option is selected, the `DIFitems` command's inclusion in the <Groups> section is required. When a DIF analysis is requested by invoking one of the `DIF` options, `DIFcontrasts` becomes a required command. This command is used to supply flexMIRT^{TM} with the DIF contrast matrix, which is used in constructing the contrasts among groups. The total number of contrasts is one fewer than the total number of groups being analyzed. For a 2 group DIF analyses, that means there is one allowable contrast, which will typically be of the form `DIFcontrasts = (1 -1);`. For analyses with more than 2 groups, any ANOVA contrasts can be specified. The Helmert contrast (which is orthogonal) is a reasonable choice, (e.g., for 3-groups:

```
DIFcontrasts = (
2.0 -1.0 -1.0,
0.0  1.0 -1.0);
```

When group size is highly unbalanced, `OrthDIFcontrasts` offers additional re-weighting of the contrast coefficients such that they become orthogonal with respect to the group sizes (Langer, 2008). This option is turned off by default.

Finally within the `<Options>` section, there are several commands that are specific to or used primarily by the non-BAEM estimation algorithms.

Syntax Display 9.6: `<Options>` -Alternate Estimation Method-Specific Options

```
<Options>
Algorithm = MHRM/MCMC;
   //Shared MH-RM and MCMC commands
RndSeed = 1842;
Imputations = 1;
Burnin = 10; MH-RM default / 500;  MCMC default
Thinning = 10;  MH-RM default / 25;  MCMC default
SaveMCO = Yes/No;  Note: the -mco file is always written with MCMC
ProposalStd = 1.0;
ProposalStd2 = 1.0;

Score = MI;  // Note: all scoring methods can be used,
but Score = MI; is only available when non-BAEM est. is used

//MH-RM-specific commands
Stage1 = 200;
Stage2 = 100;
Stage3 = 2000;
Stage4 = 0;
InitGain = 0.10;
Alpha = 1.0;
Epsilon = 1.0;
WindowSize = 3;
MCsize = 2500;

//MCMC-specific commands or defaults
ItemProposalStd = 0.1;
MaxCycle = 500;
```

The alternate estimation algorithm are available for calibration runs, scoring-only runs (particularly for when plausible values/multiple imputation theta estimates are desired), or combined calibration and scoring runs. To call the one of the alternate estimation algorithms, only `Algorithm = MHRM;` or

`Algorithm = MCMC;`is required; all other listed commands are optional, with the default values shown above. **However, users should review the descriptions provided in the following pages as there are syntax statements whose defaults should to adjusted on a model-by-model basis to obtain converged, stable, and trustworthy solutions.**

`Imputations` controls the number of imputations from the MH step per RM cycle. A larger value will lead to smoother behavior from the algorithm. In most cases, a value of 1 should be sufficient for obtaining accurate point estimations, however this number may be increased for better standard errors. Additionally, when `Score = MI;`, which is only available with non-BAEM estimation, the value of `Imputations` will also control the number plausible values drawn from individual theta posteriors and saved to the -sco file.

The `Burnin` statement controls the number of draws that are discarded from the start of the MH sampler (in either MH-RM or MCMC estimation). These draws are discarded to avoid using values that were sampled before the chain had fully converged to the target distribution. `Burnin = 10;` tells flexMIRT^{TM} to discard the first 10 values obtained by the MH sampler.

Thinning refers to the sampling-method based estimation practice of retaining only every kth draw from a chain. Thinning is used, in part, to reduce the possible autocorrelation that may exist between adjacent draws. The `Thinning` statement sets the interval for the MH sampler(in both MH-RM and MCMC estimation) - meaning if `Thinning = 10;`, every 10th draw by the sampler will be retained.

`SaveMCO = Yes;` is used to save out, when `Algorithm = MHRM;`, the MH-RM Stage 1 iteration history or, when `Algorithm = MCMC;`, the drawn parameter values into a separate -mco output file. While the -mco file can be suppressed with MH-RM estimation, it will always be produced when using MCMC estimation. **When using MCMC estimation, the values in the –mco file should be plotted to examine MCMC chain convergence.** In the -mco file when using MCMC estimation, the columns are the individual estimated parameters, with parameters reported in parameter number order (which is printed in the -irt file). The first row of the -mco file are the starting values for the parameters, the next rows will be the draws discarded as the burn-in (e.g., if `Burnin=250;`, the next 250 rows will be draws from those 250 cycles). Following the burn-in rows, the thinned main cycle rows draws are reported. If `Thinning = 10;` and `MaxCycle = 1000`, that means 1000 main cycle draws will be in the -mco file, but 1000*10 draws were completed to get those values. Users are directed to Edwards (2010) for an oveview of examining

MCMC chain convergence and relevant references.

`ProposalStd` and `ProposalStd2` control the dispersion of the Metropolis proposal densities for the first and second level of the specified model, respectively. If a single level model is specified, only the `ProposalStd` command will be used. **Although default values of 1.0 have been set for both of these commands, these values need to be adjusted on a case-by-case basis**. The values used will depend on the complexity of the model, the number of items, the type of items (e.g., dichotmous, polytomous), and the model fit to the items (e.g., Graded, 3PL, Nominal). The user should choose `ProposalStd` and `ProposalStd2` values so that the long-term average of the acceptance rates (which are printed for each iteration in the progress window) for level 1 and level 2 (if applicable) are around 0.5 for lower dimensional models (<= 3 factors) and in the 0.2 - 0.3 range for higher dimensional/more complex models. Generally speaking, increasing the `ProposalStd` value will result in lowered acceptance rates while decreasing the value will result in higher acceptance rates. Users are directed to Roberts and Rosenthal (2001) for optimal scaling, choice of dispersion constants, and long-term acceptance rates of Metropolis samplers.

`Stage1` determines the number of Stage I (constant gain) cycles. The Stage I iterations are used to improve default or user-supplied starting values for the estimation that occurs in Stage II. `Stage2` specifies the number of Stage II (Stochastic EM, constant gain) cycles. The Stage II iterations are used to further improve starting values for the MH-RM estimation that occurs in Stage III. `Stage3` sets the maximum number of allowed MH-RM cycles to be performed.

`Stage4` determines the method by which SEs are found. If `Stage4 = 0;` then SEs will be approximated recursively (see Cai, 2010b). If a non-zero value is given then the Louis formula (Louis, 1982) is used directly. If the Louis formula is to be used, the supplied value determines the number of iterations of the SE estimation routine; this number will typically need to be large (i.e., 1000 or more).

`InitGain` is the gain constant for Stage I and Stage II. If the algorithm is initially taking steps that are too large this value may be reduced from the default to force smaller steps.

`Alpha` and `Epsilon` are both Stage III decay speed tuning constants. `Alpha` is the first tuning constant and is analogous to the `InitGain` used in Stages I and II. `Epsilon` controls the rate at which the sequence of gain constants converge to zero. Specified `Epsilon` values must be in the range (0.5, 1], with a value closer to 1 indicating a faster rate of convergence to zero.

`WindowSize` allows the user to set the convergence monitor window size. Convergence of the MH-RM algorithm is monitored by calculating a window of successive differences in parameter estimates, with the iterations terminating only when all differences in the window are less than 0.0001. The default value of the window size is set at 3 to prevent premature stoppage due to random variation.

`MCSize` is the Monte Carlo size for final log-likelihood, AIC, and BIC approximations.

When using MCMC, the item parameter draws are divided up into independent segments in the flexMIRT^{TM} implemented Metropolis-Hastings within Gibbs set up. `ItemProposalStd` sets the proposal dispersion for item parameter segments.

`MaxCycle` controls the total number of MCMC draws to be accepted following the specified number of burn-in cycles. For instance, if `MaxCycle = 1000` that means that 1000 draws will be printed to the -mco file but the total number of completed draws will be MaxCycle number of draws multipled by the specified thinning value.

9.3. The Groups Section

The `<Groups>` section includes commands for specifying group names, number of items, models to calibrate, as well as more advanced features, such as commands for setting up the structure of hierarchical or multidimensional models, as well as empirical histogram priors.

Syntax Display 9.7: `<Groups>` - General Data/Model Descriptors 1

```
<Groups>
...
%GroupName%
File= "*.dat/*.txt/*.csv";
Header = Yes/No;
Varnames = vars;
Select = vars;
Exclude = vars;
Missing = -9;
N = ?;
Ncats(vars)= 2;
```

```
Model(vars) = Graded(2)/ThreePL/Nominal(?)/GPC(?);
Ta(vars) = Trend/Identity/(...);
Tc(vars) = Trend/Identity/(...);
```

Prior to any other group commands, a label must be assigned to a group, and it will become a token that all subsequent command statements may refer to. Even if only one group is present it must be given a label. The name provided to the group is arbitrary and at the discretion of the user; however, spaces should not be used when naming groups and the chosen name must be enclosed by percent signs.

The `File` statement is for specifying the path and file name in which item response data is held (or to be saved for `Simulation` runs). The number of cases in the data file is set using the `N` statement. The `File` and `N` commands are listed as required here, but there is one exception: When the engine is placed under `Scoring` mode, and scoring method `SSC` is selected, the input file is optional when only the score conversion table is made without scoring individuals.

The `Header` statement is used when the variables names/a header row is present in the data set that is being read in. Use of `Header = Yes;` means that flexMIRT^{TM} will pull the variable names from the first row of the provided data and will not need to be supplied manually by the user. Note that if `Header=Yes;` is present, then a `Varnames` statement should not be used.

`Varnames` assigns labels to variables by which they will be referred to later in commands and in the output. Variable names are separated by commas following the equals sign of the `Varnames` statement. When variable names end with consecutive integers (e.g., v1, v2, ..., v100) a short-cut is available that can be both time and space saving. Rather than listing all the variables, a dash between the name of the first variable and the last variable (i.e., v1-v100) will indicate to flexMIRT^{TM} that all variables between v1 and 100 are desired.

The optional `Select` and `Exclude` statements allows for a subset of the variables initially read from the data file to be submitted to IRT analysis. The default is for all variables in the datafile to be used for analyses. `Select` allows users to specify a subset of the available variables to be analyzed while `Exclude` allows users to tell which variables to exclude from analysis. For instance, if we had variables ID and V1-V12 in our dataset, we could use either `Select = V1-V12;` or `Exclude= ID;` so that flexMIRT^{TM} will not include the ID variable in the IRT analyses. Note that only one of these statements should be used in reference to a given dataset; that is, `Select` and `Exclude` statement should not both be used in reference to the same datafile.

Missing values may be represented by a specific number in data files. The default missing value is -9, but that may be modified with the `Missing` command. However, missing values may only be numeric values between -127 and 127. Common missing value symbols, such as a dot or a blank space will not be interpreted correctly by the program.

The number of valid response categories *observed in the data file* is specified with the `NCats` statements. Within the parentheses of the `NCats` statement, the names of the variables affected by the command are listed. For example, if Item 24, 28, and 32 - 40 all had four response options, this could be entered as `NCats(v24,v28,v32-v40) = 4;` in flexMIRT^{TM} syntax.

The `Model` command follows the same general format as `Ncats`, but instead of placing a number after the equals sign, one of the four keywords for the IRT model must be chosen. For example, to fit the graded model with four categories to the hypothetical items previously mentioned, the `Model` statement is `Model(v24,v28,v32-v40) = Graded(4);`. The question marks in parentheses following the `Graded`, `Nominal`, and `GPC` keywords in Example Box 8.7 are place-holders for the number of categories the model should accommodate. When items are of mixed formats (multiple choice, free response, Likert-type scale), multiple `NCats` and `Model` statements may be needed to correctly describe the data and the chosen IRT models.

The commands `Ta` and `Tc` are used in conjunction with the nominal categories model, supplying contrasts for the scoring function and intercepts, respectively. The default for both `Ta` and `Tc` is a trend contrast matrix (keyword is `Trend`), but an identity matrix (keyword is `Identity`) which allows for equality constraints, or a user-supplied matrix (denoted by (...) in the syntax box) may also be employed. A user-supplied intercept contrast matrix was demonstrated in Example 2.11; a scoring function matrix would be specified in a similar fashion. Multiple `Ta` and `Tc` statements may be supplied, allowing for a large degree of flexibility in the parameterization of the nominal model.

We have covered the required commands within the `<Groups>` section as well as several optional commands. There are additional `<Groups>` section commands, that although still general purpose, will see more limited use.

Syntax Display 9.8: `<Groups>` - General Data/Model Descriptors 2

```
<Groups>
...
Code(vars) = (?),(?);
Key(vars) = (?);
CaseID = var;
FixPrior = Yes/No;
LatentDistribution =Gaussian/EH/KDE;
Bandwidth = 1;
EmpHist = Yes/No;
PosteriorOut =  Yes/No;
ItemWeights(vars) = (?);
BetaPriors(vars) = ?;
CaseWeight = var;
```

The internal representation of item response data in flexMIRT™ is zero-based (i.e., response options must start at zero and go up). That being said, the program is capable of recoding the data, when old and new values are specified with the `Code` command. If the first 10 items are read from a data file that had four response options (1, 2, 3, 4), it would be necessary to recode these to zero-based, prior to fitting the IRT model. The recode statement could be `Code(v1-v10) = (1,2,3,4),(0,1,2,3);`. The original values are listed first and the values to which they should be recoded appear within the second set of parentheses. If it is necessary to also collapse the responses, so that the original codes 1 and 2 become a single response option 0 and the original 3 and 4 become a second option 1, that `Code` command could be `Code(v1-v10) = (1,2,3,4),(0,0,1,1);`. Note that recoding does not change the values of the number of response options in the data file, as defined by the `NCats` statements.

The `Key` statement also provides a specific form of recoding that may be useful when the observed responses from multiple choice items have been read in. Rather than writing numerous `Code` statements to convert the observed responses into correct/incorrect 0/1 data, the `Key` statement allows the user to provide a scoring key for the items and instruct flexMIRT^{TM} to score the items internally. For instance, if the correct responses to Items 1 through 4 were 2, 0, 1, and 3, respectively, the `Key` statement to score these items would be `Key(v1-v4) = (2,0,1,3);`. Multiple `Code` and `Key` statements are permitted, to easily accommodate situations in which items have variable numbers of response options or, when scoring a large number of items, multiple scoring keys would result in greater user-readability of the command file by limiting command line length. It should be noted that both the `Code` and the `Key` commands are completely internal to flexMIRT^{TM} and the original data file will not be modified. Although flexMIRT^{TM} can do some basic data management, it is not built to be a data management tool. It is generally advisable to perform all necessary data cleaning and data management in a general statistical software package (e.g., SAS, R, SPSS) before exporting a space-, comma-, or tab-delimited file for flexMIRT^{TM} analysis.

`CaseID` is used to specify the variable that provides unique identifiers for the individuals. If used during scoring, the ID values will be saved to the -sco file, along with the estimated scores.

The `FixPrior` statement is used to fix the latent distribution prior to density values supplied in a -prm.txt file, rather than assuming it is a standard normal prior. These supplied values will define the height of the rectangle at each quadrature node, such as the density values that may be found through the use of empirical histograms during calibration. It is expected that the number of supplied densities values will match the number of quadrature points specified in the `Quadrature` statement and will proceed from the density associated with the smallest quadrature node up to the density value of the highest node.

The `LatentDistribution` statement is used to set how the latent variable prior/population distribution is handled. The default of `LatentDistribution = Gaussian;` will result in an assumed Normal distribution being used. If the latent distribution is thought to be non-normal, the options of `EH` (for empirical histograms [EHs]) or `KDE` (for kernel density estimation) may be employed by the user to empirically estimate the shape of the prior distribution. When `LatentDistribution = EH`, flexMIRT^{TM} will calibrate item parameters against the estimated EH prior, which is based on the normalized accumulated posterior densities for all the response patterns at each of the quadrature nodes, and flexMIRT^{TM} will save the EH prior in the -irt and -prm file. Additionally, for a concurrent scoring run or any future scoring runs that use the saved parameter file containing the EH values, the program will recognize that the group contains EH weights and will use them to compute EAPs or SSCs.

Similarly, when `LatentDistribution = KDE;` is used, the shape of the prior distribution will be characterized by Gaussian KDE. The shape of the distribution is characterized by the sum of the standard normal distribution kernels at each quadrature node; these summed values (collectively, the KDE prior) are reported in the -irt and -prm output files. The smoothness (or lack thereof) of the final KD estimated distribution's shape is controlled by the `Bandwidth` statement, which has a default value of 1.0. Generally speaking, smaller `Bandwidth` values will result in more jagged distributional shapes while larger `Bandwidth` values result in a more smooth shape. Users are encouraged to try several different bandwidth values to ensure that optimal smoothing is obtained. As with the use of EH, for a concurrent scoring run or any future scoring runs that use the saved parameter file containing the KDE values, the program will recognize that the group contains KDE weights and will use them to compute EAPs or SSCs. EH estimation may only be used with single-level unidimensional or bifactor/testlet models. When used with a bifactor-type model, only the general dimension will be characterized using the EHs; all other dimensions will be assumed Gaussian. KDE may be used with unidimensional models or applied to the general factor(s) of a two-tier model (e.g., Cai, 2010a).

The `EmpHist` statement is a legacy command, retained for backwards compatability, used to enable the EH characterization of the latent variable prior distribution, simultaneously estimated with the item parameters. `EmpHist = Yes;` and `LatentDistribution = EH;` result in the same behavior from the program.

The `PosteriorOut` statement is used to request that flexMIRT^{TM} print the density of the latent variable posterior distribution at each quadrature node. This statement will be ignored when MH-RM estimation is used and is not available for bifactor-type models (specifically when `Primary` is set to a value greater than zero) or for multilevel models (when `Between` is set to a value greater than zero). The requested density values of the posterior will be printed to the general *-irt.txt output file.

While category weights are typically taken to be the same that the category value (e.g., a response of 3 is assigned a weight of 3), this does not have to be the case. If alternate category weights for items are desired, the `ItemWeights` statement may be used to assign such alternative weights to items. For dichotomously scored items, a statement of `ItemWeights(v1-v3)`

`= (0.0,0.5);` could be used to assign, for variables v1 to v3, a weight of 0 to all 0 responses and a weight of 0.5 to all 1 responses. For polytomous items, the same format is holds. Suppose variables v4 to v8 had 5 possible responses categories, weights alternate to the default values of 0,1,2,3,4 could be assigned with the command, `ItemWeights(v4-v8) = (0.0,0.1,2.0,3.0,3.1);`. Due to this statement, scores of 0 are assigned a category weight of 0, 1s a weight of 0.1, 2s a weight of 2.0, all 3s given a weight of 3.0 and responses of 4 a weight of 3.1. Multiple `ItemWeights` statements are permitted, to handle situations when the number of responses options per item is variable.

`BetaPriors` is used to assign item-specific beta distribution priors to the uniquenesses; the rationale and use of applying priors to the item-uniquenesses as a method of preventing Heywood cases was discussed in Bock et al. (1988) on pp. 269-270 . The beta distribution applied in flexMIRT^{TM} has an α parameter set to the user-specified value and the β parameter fixed at 1. For example, the statement `BetaPriors(v1-v5) = 1.6;` would apply to the uniquenesses of Items 1 through 5 a prior distribution of the form Beta(1.6, 1). Multiple `BetaPriors` statements are permitted.

The `CaseWeight` command is needed when response pattern by frequency (aka grouped) data is entered. It provides the program with the variable name for the column that contains the number of observations (or weights) for each response pattern.

In addition to the general purpose syntax `<Groups>` section statements, there are also specialized commands available, such as for use with MIRT models including EFAs, diagnostic classification models, and multilevel models.

Syntax Display 9.9: `<Groups>` - Specialized Data/Model Commands

```
<Groups>
...
Dimensions = 1;
Primary = 0;

Between = 0;
Nlevel2 = ?;
Cluster = var;

Crossed = No/Yes;
Block = var;

Rotation  =  None/CFquartimax/CFvarimax/Target;
Oblique = Yes/No;
UnspecifiedTargetElement = 9;
Target = ( ...,  ...);

Attributes = ?;
InteractionEffects = (?,?,...);
Generate = (?,?), (?,?),...;
DM = group;

DIFitems = vars;
```

The `Dimensions` statement is used to specify the total number of dimensions in the model. The default is a single-level unidimensional model (i.e., `Dimensions = 1;`). Changing the number of dimensions is the first step in setting up multidimensional and multilevel models. The `Primary` statement is used to set the number of primary, or general, dimensions/factors that will be used in a two-tier multidimensional model. In a standard bifactor or testlet model, `Primary=1;` although, for complex models, more than one primary dimension is allowed.

Several of the syntax statements in the currently discussed syntax display are closely associated with multilevel models. To initiate the basic structure for a 2-level model, the `Between` command is used, which specifies that number of latent variables at level-2 (aka Between). When multilevel data are simulated, it is necessary to tell flexMIRT$^{\text{TM}}$ how many level-2 units should be generated. This is specified by the `Nlevel2` command. This command is only necessary when simulating multilevel data, as the program is able to determine the number of unique level-2 units from the data file, provided that the `Cluster` command is employed.

`Cluster` names the variable that contains unique identifying values for the level-2 units (e.g., schools in which students are nested).

The `Crossed` and `Block` statements are employed when a user wishes to fit a crossed random effect model (e.g., de Boeck, 2008, Van den Noortgate et al., 2003), which, by definition, are multilevel models. `Crossed=Yes;` is used to tell flexMIRT$^{\text{TM}}$ that a crossed random effects model is desired and the `Block` statement is used to denote the variable within the dataset that tracks the feature (often item family) that is "crossed" with persons and fitted as a random, rather than a fixed, effect. Crossed random effect models are only available when non-BAEM estimation is used.

Four commands are available that are specific to EFA. `Rotation` is the keyword that indicates to flexMIRT$^{\text{TM}}$ that an EFA is desired, with a default value of `Rotation = None;` indicating a CFA should be fit. The available rotations are Crawford-Ferguson (CF) family Quartimax, CF-Varimax, and target rotation (e.g., Browne, 2001). Each of these rotations may be orthogonal (uncorrelated factors) or oblique (correlated factors), as determined by the `Oblique` statement, which defaults to "Yes." Note that the initial extraction values for an EFA (that is, the unrotated solution) are always printed in the -irt output file. Therefore, if an unrotated EFA solution is desired, users must set `Rotation` to something other than "None" to trigger the EFA module, but may then simply ignore the rotated solution in the output.

When target rotation is specified, it become required that a target factor pattern matrix be supplied. This is done via the `Target` statement, which accepts the matrix submitted within parentheses with spaces separating elements and commas delineating row ends. The target matrix should be comprised of only 0s and the value set via the `UnspecifiedTargetElement` command, which has a default of 9. Note that `UnspecifiedTargetElement` may be changed to an alternate value, but it is restricted to numeric values between -127 and 127. An example of a target matrix may be found in both the MIRT and MH-RM chapters of this manual. The next four commands are used exclusively for the fitting of diagnostic classification models. `Attributes` is used to set the number of main effects present in the skill space that is being assessed. This is the keyword that triggers flexMIRT^{TM} to model discrete latent variables, rather than the default continuous latent variables. `InteractionEffects` instructs flexMIRT^{TM} to automatically generate all possible interaction effects. For instance, with 4 attributes one may specify `InteractionEffects = (2,3,4);` and, in addition to the 4 main effects, the program with also generate the 2nd, 3rd, and 4th order interaction terms of the attributes as dimensions of the model. If only the second order interaction effects are desired, those would be created by specifying `InteractionEffects = (2);`.

Similar to the `InteractionEffects` keyword, `Generate` sets up higher order interactions effects. However, rather than creating all possible interaction effects `Generate` creates only those effects specified in the statement. For example, `Generate = (3,6,7),(4,7);` is used to generate the interaction effect of attributes 3, 6, and 7 and, separately, the interaction effect of attributes 4 and 7. There is no limit to the number of interaction effects that may be specified in a `Generate` statement and the order in which interaction effects are specified does not matter.

The `DM` command stands for "diagnostic model" and is used to set the group containing observed data - this is the group to which the higher-order DM will be applied. The `DM` command is optional; one could, in principle, fit a model without the DM (higher-order) latent variables.

The final command `DIFitems` from the "Specialized Data/Model Commands" section provides flexMIRT^{TM} with the names of items to be tested for DIF during a targeted DIF analysis. This command is required when `DIF= TestCandidates;` has been invoked in the `<Options>` section. The names of candidate items are simply listed, separated by commas, (e.g., `DIFitems = v1-v4,v8;`)

Syntax Display 9.10: `<Groups>` - Commands Available only with non-BAEM estimation

```
<Groups>
...
Covariates = vars/0;
L2covariates = 0;
CovariateCorr =  0;
```

Within the `<Groups>` section there are handful of commands, regarding covariates, that are only available when one of the alternate estimation methods is used. During calibration, `Covariates` is used to supply flexMIRT^{TM} with a list of variables that will serve as predictors of the latent variable estimates. It is expected that continuous variables will be supplied as covariates or, for categorical covariates, that appropriate codings (effect coding, dummy coding, etc.) into multiple variables have been constructed prior to being supplied to flexMIRT^{TM}. When used in simulation, `Covariates` is used to specify the number of covariates that should be generated as part of the model.

The `L2covariates` statement is used to tell flexMIRT^{TM} how many of the covariates supplied in the `Covariates` should be used as predictors of the level-2 (Between) latent variables. For example, during calibration when `L2covariates` is set to a non-zero value, the first x variables listed in the `Covariates` statement will be implemented as predictors of the higher-level latent variable(s). When used in with a simulation, `L2covariates` indicates that the first x simulated covariates will apply to level-2 factors only.

`CovariateCorr` is used during a simulation run to set the generating correlation value among the simulated covariates. Even when more than 2 covariates are present, `CovariateCorr` should still be a single value - for simplicity, flexMIRT^{TM} is will induce only an equicorrelation matrix among covariates.

9.4. The Constraints Section

Several of the examples in the previous chapters have shown that there is no required statement in the `<Constraints>` section, and that it may be left blank after the (required) section header. However as models become more complex, especially if they include multiple groups and multiple dimensions, constraints will more than likely become necessary.

Listed in the left-most column of Table 9.1 are the types of constraints available. `Value` indicates the starting or fixed value of a parameter. `Free` indicates a parameter which is freely estimated, `Equal` indicates parameters which will be constrained equal, `Fix` fixes a parameter, `Coeff` applies a provided coefficient to a given parameter - enabling proportionality restrictions, `AddConst` applies a additive constant to specified parameters, and `Prior` provides a prior distribution for that parameter.

Table 9.1: Constraint Commands and Keywords

Constraint Type	Parameter	Prior Distribution
Free	Intercept(?)	Normal(μ, σ)
Equal	Slope(?)	logNormal(μ, σ)
Fix	Guessing	Beta($\alpha - 1, \beta - 1$)
Value	ScoringFn(?)	
Coeff	Mean(?)	
AddConst	Cov(?,?)	
Prior	Beta(?,?)	
	MainEffect(?)	
	Interaction(?,?,...)	

Presented in the second column of Table 9.1 are the keywords for parameters that may have constraints applied to them. `Intercept` refers to the location parameter. `Slope` refers to the slope/discrimination parameter. `Guessing` is the lower asymptote parameter in the 3PL model. `ScoringFn` is the scoring function contrast parameters of the nominal model. `Mean` and `Cov` refer to the mean and covariance matrix elements of the latent variable distribution. `Beta` refers to the matrix of covariates predicting the latent variables. `MainEffect` and `Interaction` are available so users may employ syntax specifications that are in keeping with the conceptualization/terminology used in diagnostic classification models.

The question marks in the parentheses following the parameter keywords permits the referencing of specific parameters. For example, when there are 4 latent variables in the model, `Slope(1)` refers to the slope on the first factor, `Mean(2)` refers to the mean of the second factor, and `Cov(3,2)` refers to the covariance of factors 3 and 2. When an item has 5 categories, `Intercept(2)` in a graded model or `ScoringFn(3)` in a nominal model each refers to specific item parameters.

For the `Free`, `Equal`, and `Fix` constraints, the general format for applying restrictions to the model is the same. First, the type of constraint to be applied and the group to which it is being applied are listed, followed by a comma. The items which are affected by the constraint are then specified in parentheses, followed by another comma and then the keyword for the parameter that is affected (e.g., `Slope`, `Guessing`, etc). For a unidimensional model in `Group1`, the slope parameters for v1 through v5 can be set equal using the following syntax:

```
Equal Group1, (v1-v5), Slope;
```

If there is only one group, the group name can be omitted.

```
Equal (v1-v5), Slope;
```

The following syntax fixes the second slope of (v1,v2) to 0 (the default value unless modified by `Value` statement).

```
Fix (v1,v2), Slope(2);
```

The following syntax fixes the second factor mean of `Group2` to 0 (the default value unless modified by `Value` statement).

```
Fix Group2, Mean(2);
```

The following syntax fixes the second factor variance of `Group2` to 1 (the default value unless modified by `Value` statement).

```
Fix Group2, Cov(2,2);
```

The following syntax fixes the covariance between factors 2 and 3 in `Group2` to 0 (the default value unless modified by `Value` statement).

```
Fix Group2, Cov(3,2);
```

The following syntax allows the first latent factor to be predicted by the second covariate in `Group2`.

```
Free Group2, Beta(1,2);
```

The following syntax fixes the first scoring function contrast for (v1) to 1 (the default value unless modified by `Value` statement).

```
Fix (v1),ScoringFn(1);
```

The following syntax fixes the second scoring function contrast for (v1) to 0 (the default value unless modified by `Value` statement).

```
Fix (v1),ScoringFn(2);
```

The following syntax frees the third slope for items v1-v5 in `Group2`.

```
Free Group2,(v1-v5),Slope(3);
```

Cross-group constraints are also allowed. The basic constraint format is still used, with the addition of the constraint in the second group following a colon. In one of our multiple group examples, it was desired that the item parameters for 12 items be the same across the 2 groups. To accomplish this, the `Equal` constraints in the following excerpted code were employed.

Syntax Display 9.11: Constraints Across Groups, Specifying a Prior Distribution, and Freely Estimating Theta

```
...
   <Constraints>
   Free Grade3, Mean(1);
   Free Grade3, Cov(1,1);
   Equal Grade3, (v1-v12), Guessing:
         Grade4, (v1-v12), Guessing;
   Equal Grade3, (v1-v12), Intercept:
         Grade4, (v1-v12), Intercept;
   Equal Grade3, (v1-v12), Slope:
         Grade4, (v1-v12), Slope;
   Prior Grade3, (v1-v12), Guessing : Beta(1.0,4.0);
```

As can be seen, for the three constraints that set the item parameters across groups, the general constraint format of *type group*, *(items)*, *parameter* is maintained for both groups, with a colon between the listing for the Grade 3 and Grade 4 groups. If more than two groups are to be constrained equal, additional colons, followed by the additional group information may be added. Also of note in this syntax excerpt, by default, the latent variable mean vector and covariance matrix are initially constrained to zero and identity, respectively. Using constraints, the first two lines in the example above specify that the mean and variance for the latent trait are to be freely estimated in the Grade 3 group, relative to the fixed values of the Grade 4 group which has a mean of 0 and variance of 1.

The format of the `Value`, `Coeff`, and `AddConst` constraints is similar to the previous types of constraints but with an additional piece of information. After specifying the constraint type, group, items, and parameter, a numeric value is specified as well which is separated from the parameter keyword by a comma. For example, a coefficient command could be something like, `Coeff G, (v1-v10), Slope, 1.702;` where we are applying a coefficient, in group G, for variables v1-v10, of 1.702 to the slopes, effectively incorporating the scaling constant D into the estimation.

The `Value` statement is a more general purpose constraint command. In calibration mode, it can be used to specify the values of fixed parameters or it can be used to supply starting values. In scoring mode, instead of using a parameter file (which is more convenient), one can in principle use the `Value` statement to fix item and group parameters to the calibrated estimates. In simulation mode, the `Value` statement can be used to specify generating parameters. For a `Value` constraint on one or more parameters, if the parameters are free, the numeric values supplied become starting values. If the parameters are fixed, `Value` can be used to change the default fixing values. For example, for a unidimensional model in a single group, the following syntax results in a set of constraints that fixes the item slopes to 1.0 and then frees the estimation of the variance of the latent variable, effectively creating a Rasch-type model.

```
Fix (v1-v5), Slope;
Value (v1-v5), Slope, 1.0;
Free Cov(1,1);
```

For applying prior distributions, the format is quite similar to the other constraints, but with the addition of a distribution type (and its associated pair of parameter values) after a colon. The syntax still follows the general constraint format of *type group*, *(items)*, *parameter*, with the addition of the requested prior distribution following a colon. In the across groups constraint example previously used, we specified a beta prior for the guessing parameters. The prior chosen was a beta distribution with $\alpha - 1 = 1.0$ and $\beta - 1 = 4.0$. As noted in Table 9.1, the `Normal` distribution and the `logNormal` distributions are also available when assigning priors. For the `Normal`, the prior mean and standard deviation are needed. For the `logNormal`, the prior mean and standard deviation on the logarithmic scale are needed. For the `Normal` prior, if it is used on the guessing parameter, it is in fact imposed as a logit-normal prior on the logit of the guessing probability.

CHAPTER 10

Models

In this chapter, some more technical aspects of the IRT model implemented in flexMIRT$^{\text{TM}}$ are discussed. We begin with a generalized formulation, focusing on the linear predictor portion of the model. More specific issues are discussed next.

10.1. A Generalized Formulation

Consider a linear predictor for a level-1 unit i (e.g., individual) in level-2 unit j (e.g., school) in a group g (e.g., country):

$$\boldsymbol{\eta}_{ijg} = \mathbf{A}_g^B \boldsymbol{\theta}_{jg} + \mathbf{A}_g^W \boldsymbol{\theta}_{ijg},$$

where $\boldsymbol{\eta}_{ijg}$ is an $n_g \times 1$ vector of linear predictors of the n_g items in this group, $\mathbf{A}_g^B$ is an $n_g \times p$ matrix of item slopes on the p level-2 (between) latent dimensions or latent attributes (and interaction terms), and $\mathbf{A}_g^W$ is an $n_g \times q$ matrix of item slopes on the q level-1 (within) latent dimensions or latent attributes (and interaction terms), with $\boldsymbol{\theta}_{jg}$ and $\boldsymbol{\theta}_{ijg}$ being the between and within latent dimensions or attributes (and attribute interactions). Note that both $\mathbf{A}_g^B$ and $\mathbf{A}_g^W$ are implicitly assumed to be functions of d unknown structural parameters in $\boldsymbol{\xi}$, subject to appropriate identification conditions or user specified constraints.

At the item level, the IRT model is a conditional density $f_{\boldsymbol{\xi}}(y_{ijkg}|\eta_{ijkg}) = f_{\boldsymbol{\xi}}(y_{ijkg}|\boldsymbol{\theta}_{jg}, \boldsymbol{\theta}_{ijg})$, where $k = 1, \ldots, n_g$ and y_{ijkg} is the item response to item k from level-1 unit i in level-2 unit j in group g. Conditionally on the latent variable levels the item responses are independent such that

$$f_{\boldsymbol{\xi}}(\mathbf{y}_{ijg}|\boldsymbol{\theta}_{jg}, \boldsymbol{\theta}_{ijg}) = \prod_{k=1}^{n_g} f_{\boldsymbol{\xi}}(y_{ijkg}|\boldsymbol{\theta}_{jg}, \boldsymbol{\theta}_{ijg}).$$

To introduce latent covariates, we consider the following regression model

$$\begin{pmatrix} \boldsymbol{\theta}_{ijg} \\ \boldsymbol{\theta}_{jg} \end{pmatrix} = \mathbf{B} \begin{pmatrix} \mathbf{x}'_{ijg} \\ \mathbf{x}'_{jg} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\epsilon}_{ijg} \\ \boldsymbol{\epsilon}_{jg} \end{pmatrix},$$

where $\mathbf{x}_{ijg}$ is the vector of level-1 fixed covariates for level-1 unit i in level-2 unit j and $\mathbf{x}_{jg}$ is the vector of level-2 covariates for level-1 unt j, and $\boldsymbol{\epsilon}$'s contain the disturbance terms for latent variables at each level. The parameters in $\mathbf{B}$ and error covariances of $\boldsymbol{\epsilon}$ are modeled as structural parameters as part of $\boldsymbol{\xi}$. In general, it would not make sense to regress a level-2 latent variable on level-1 covariates.

Let the distribution of $\boldsymbol{\theta}_{ijg}$ be $f_{\boldsymbol{\xi}}(\boldsymbol{\theta}_{ijg}|\mathbf{x}_{ijg}, \mathbf{x}_{jg})$. The level-1 latent variables can be integrated out as

$$f_{\boldsymbol{\xi}}(\mathbf{y}_{ijg}|\boldsymbol{\theta}_{jg}, \mathbf{x}_{ijg}, \mathbf{x}_{jg}) = \int f_{\boldsymbol{\xi}}(\mathbf{y}_{ijg}|\boldsymbol{\theta}_{jg}, \boldsymbol{\theta}_{ijg}) f_{\boldsymbol{\xi}}(\boldsymbol{\theta}_{ijg}|\mathbf{x}_{ijg}, \mathbf{x}_{jg}) d\boldsymbol{\theta}_{ijg}.$$

Assuming further conditional independence of level-1 units on $\boldsymbol{\theta}_{jg}$, the marginal distribution (given fixed covariates) of all item responses in a level-2 unit is

$$f_{\boldsymbol{\xi}}(\mathbf{Y}_{jg}|\mathbf{X}_{jg}) = \int \prod_{i=1}^{N_j} f_{\boldsymbol{\xi}}(\mathbf{y}_{ijg}|\boldsymbol{\theta}_{jg}, \mathbf{x}_{ijg}, \mathbf{x}_{jg}) f_{\boldsymbol{\xi}}(\boldsymbol{\theta}_{jg}|\mathbf{x}_{jg}) d\boldsymbol{\theta}_{jg},$$

where N_j is the number of level-1 units in level-2 unit j, $f_{\boldsymbol{\xi}}(\boldsymbol{\theta}_{jg})$ is the distribution of the level-2 latent variables, and $\mathbf{Y}_{jg} = \{\mathbf{y}_{1jg}, \ldots, \mathbf{y}_{N_j jg}\}$ is the collection of item response patterns for all n_g items from the N_j level-1 units in level-2 unit j of group g. Similarly, $\mathbf{X}_{jg}$ collects together all the level-1 and level-2 fixed covariates.

Once the item responses are treated as fixed upon observation, the marginal likelihood of $\mathbf{Y}_{jg}$ is defined as

$$L(\boldsymbol{\xi}|\mathbf{Y}_{jg}, \mathbf{X}_{jg}) = f_{\boldsymbol{\xi}}(\mathbf{Y}_{jg}|\mathbf{X}_{jg}).$$

Taking logs and summing over the (assumed) independent level-2 units, as well as groups, the marginal log-likelihood for the structural parameters is

$$\log L(\boldsymbol{\xi}|\mathbf{Y}_1, \ldots, \mathbf{Y}_g, \mathbf{X}_1, \ldots, \mathbf{X}_g) = \sum_{g=1}^{G} \sum_{j=1}^{J_g} L(\boldsymbol{\xi}|\mathbf{Y}_{jg}, \mathbf{X}_{jg}),$$

where G is the number of groups, J_g is the number of level-2 units in group g, and $\mathbf{Y}_g = \{\mathbf{Y}_{1g}, \ldots, \mathbf{Y}_{J_g g}\}$ is the collection of item response patterns for all level-1 and level-2 units in group g, and $\mathbf{X}_g = \{\mathbf{X}_{1g}, \ldots, \mathbf{X}_{J_g g}\}$ collects together the covariates.

10.2. Item Response Models

For the kth row in $\boldsymbol{\eta}_{ijg}$, the linear predictor is equal to $\eta_{ijkg} = \mathbf{a}_{kg}^{B}\boldsymbol{\theta}_{jg} + \mathbf{a}_{kg}^{W}\boldsymbol{\theta}_{ijg}$, where $\mathbf{a}_{kg}^{B}$ is the set of p slopes on the between latent variables and $\mathbf{a}_{kg}^{W}$ is the set of q slopes on the within latent variables.

10.2.1 Model with Pseudo-Guessing Parameter

This is the multilevel and multidimensional extension of the 3PL model. Let the correct/endorse (1) response probability be

$$P_{\boldsymbol{\xi}}(y_{ijkg} = 1|\eta_{ijkg}) = \text{guess}_{kg} + \frac{1 - \text{guess}_{kg}}{1 + \exp[-(c_{kg} + \eta_{ijkg})]},$$

where $\text{guess}_{kg} = \frac{1}{1+\exp[-\kappa_{kg}]}$ and in which guess_{kg} is the item-specific pseudo-guessing probability for item k in group g and c_{kg} is the group- and item-specific intercept. Consequently, $P_{\boldsymbol{\xi}}(y_{ijkg} = 0|\eta_{ijkg}) = 1.0 - P_{\boldsymbol{\xi}}(y_{ijkg} = 1|\eta_{ijkg})$.

10.2.2 Graded Response Model

Suppose item k has K graded categories. Let the cumulative response probabilities be

$$\begin{aligned}
P_{\boldsymbol{\xi}}(y_{ijkg} \geq 0|\eta_{ijkg}) &= 1.0, \\
P_{\boldsymbol{\xi}}(y_{ijkg} \geq 1|\eta_{ijkg}) &= \frac{1}{1 + \exp[-(c_{kg,1} + \eta_{ijkg})]}, \\
&\vdots \\
P_{\boldsymbol{\xi}}(y_{ijkg} \geq K-1|\eta_{ijkg}) &= \frac{1}{1 + \exp[-(c_{kg,K-1} + \eta_{ijkg})]}, \\
P_{\boldsymbol{\xi}}(y_{ijkg} \geq K|\eta_{ijkg}) &= 0.0,
\end{aligned}$$

where the c's are item intercepts and the boundary cases are defined for consistency. Then the category response probabilities are

$$P_{\boldsymbol{\xi}}(y_{ijkg} = l|\eta_{ijkg}) = P_{\boldsymbol{\xi}}(y_{ijkg} \geq l|\eta_{ijkg}) - P_{\boldsymbol{\xi}}(y_{ijkg} \geq l+1|\eta_{ijkg}).$$

10.2.3 Nominal Categories Model

The nominal model fit by flexMIRT$^{\text{TM}}$ is the multilevel extension of the reparameterized nominal model described by Thissen et al. (2010) and subsequently revised by Thissen and Cai (2016) to allow for dimension-specific scoring functions that may be useful for response style analysis (Falk & Cai, 2016). The dimension-indexing creates some deviations from the more general and standard notation used throughout so we will only discuss the less general case described by Thissen et al. (2010) with dimension-invariant scoring function values. Users interested in response style modeling are encouraged to consult Thissen and Cai (2016) for the full details. Suppose item k has K nominal categories. Category l's response probability is defined as

$$P_{\boldsymbol{\xi}}(y_{ijkg} = l | \eta_{ijkg}) = \frac{\exp(s_{kg,l}\eta_{ijkg} + c_{kg,l})}{\sum_{m=0}^{K-1} \exp(s_{kg,m}\eta_{ijkg} + c_{kg,m})},$$

where $s_{kg,l}$ is the scoring function value for category l and $c_{kg,l}$ is the intercept value for category l. Dropping the kg subscript for a minute, Thissen et al. (2010) pointed out that for identification, the following restrictions should be in place: $s_0 = 0$, $s_{K-1} = K - 1$, $c_0 = 0$. This can be accomplished by reparameterizaton. Let

$$\mathbf{s} = \begin{pmatrix} s_0 \\ \vdots \\ s_{K-1} \end{pmatrix} = \mathbf{T}_a \begin{pmatrix} 1 \\ \boldsymbol{\alpha} \end{pmatrix}, \text{ and } \mathbf{d} = \begin{pmatrix} c_0 \\ \vdots \\ c_{K-1} \end{pmatrix} = \mathbf{T}_c \boldsymbol{\gamma}.$$

The vector $\boldsymbol{\alpha}$ is a $(K-2) \times 1$ vector of scoring function contrasts that defines the ordering of categories, and $\boldsymbol{\gamma}$ is a $(K-1) \times 1$ vector of intercept contrasts. The matrices $\mathbf{T}_a$ and $\mathbf{T}_c$ are fixed $K \times (K-1)$ matrices of contrast coefficients. By an appropriate choice of $\mathbf{T}$, the identification restrictions will be automatically satisfied. Thissen et al. (2010) propose the use of the following linear-Fourier contrast matrix

$$\mathbf{T} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 1 & t_{2,2} & \cdots & t_{2,(K-1)} \\ 2 & t_{3,2} & \cdots & t_{3,(K-1)} \\ \vdots & \vdots & & \vdots \\ K-1 & 0 & \cdots & 0 \end{pmatrix},$$

where a typical element $t_{l,m}$ for $l = 1, 2, \ldots, K$ and $m = 1, 2, \ldots, K-1$ takes its value from a Fourier sine-series:

$$t_{l,m} = \sin\left\{\frac{\pi(l-1)(m-1)}{K-1}\right\}.$$

For instance, for $K = 4$, the contrast matrix is

$$\mathbf{F} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & .866 & .866 \\ 2 & .866 & -.866 \\ 3 & 0 & 0 \end{pmatrix},$$

and for $K = 5$, the contrast matrix is

$$\mathbf{F} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & .707 & 1 & .707 \\ 2 & 1 & 0 & -1 \\ 3 & .707 & -1 & .707 \\ 4 & 0 & 0 & 0 \end{pmatrix}$$

For the nominal model, one can verify that identification conditions are all satisfied. For the GPC model, all the $\boldsymbol{\alpha}$ contrasts are set to zero.

Appendices

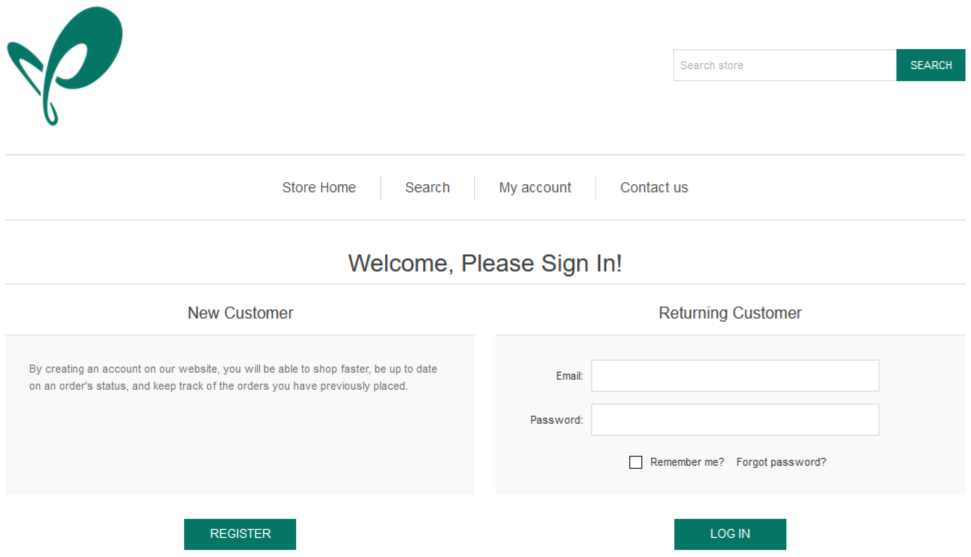
APPENDIX A

Quick-Start Installation and User Guide

This guide is designed to provide an overview for installing flexMIRT^{TM} from the webpage and initializing the program for use.

A.1. Installing flexMIRT

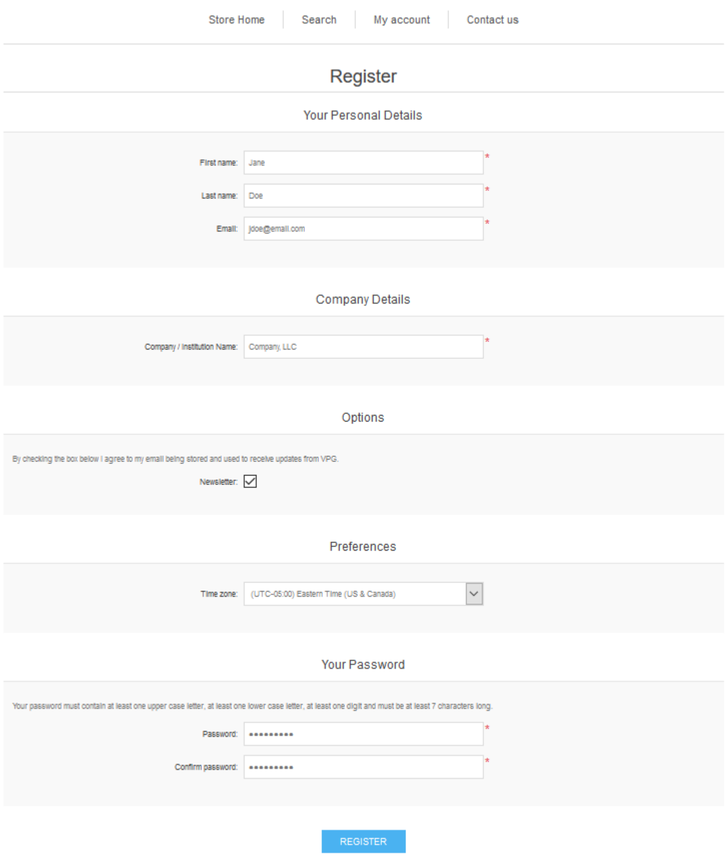
To obtain the flexMIRT^{TM} installer, you must first register for an account on the VPG Store. The registration page may be found by selecting the My Account link in the flexMIRT^{TM} website banner or visited directly at `https://store.vpgcentral.com/`. From your account page, you will be able to change your password, view your current flexMIRT^{TM} license and expiration date information, manage licenses on your various computers (with a purchased academic license users are allowed up to 3 separate installs of the program), as well as make payments for new or renewed licenses.



By creating an account on our website, you will be able to shop faster, be up to date on an order's status, and keep track of the orders you have previously placed.

After selecting Register, you will be taken to the registration page and asked to enter your name, email address, and other information, including your desired password.

Once you have submitted this information via the Register button, an email confirming your registration information will be sent to the email address you provided. The confirmation email will originate from websupport@vpgcentral.com - please add this email address to your safe list. You may now log in to your VPG Store account.



Register

Your Personal Details

First name: Jane *
Last name: Doe *
Email: jdoe@email.com *

Company Details

Company / Institution Name: Company, LLC *

Options

By checking the box below I agree to my email being stored and used to receive updates from VPG.

Newsletter: ✓

Preferences

Time zone: (UTC-05:00) Eastern Time (US & Canada)

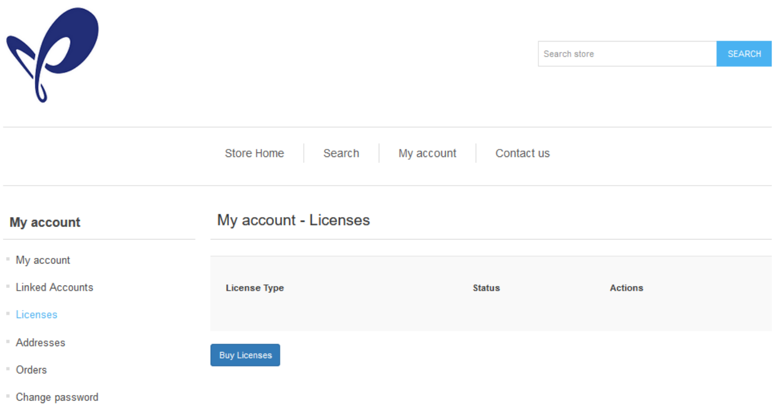
Your Password

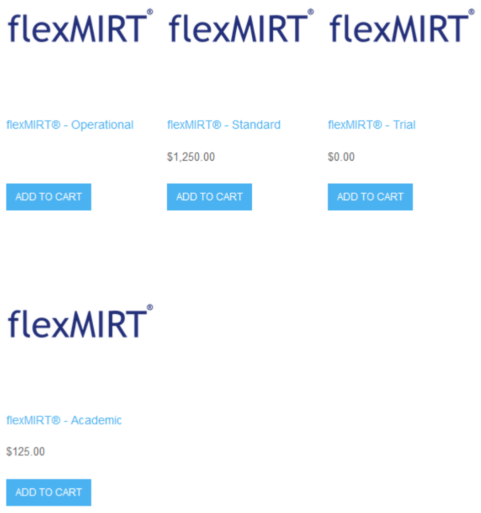
Your password must contain at least one upper case letter, at least one lower case letter, at least one digit and must be at least 7 characters long.

Password: ••••••••• *
Confirm password: ••••••••• *

To gain access to the free trial or any paid license you would like to purchase, go to the Licenses page within your account which can be found on the left hand side of your screen.

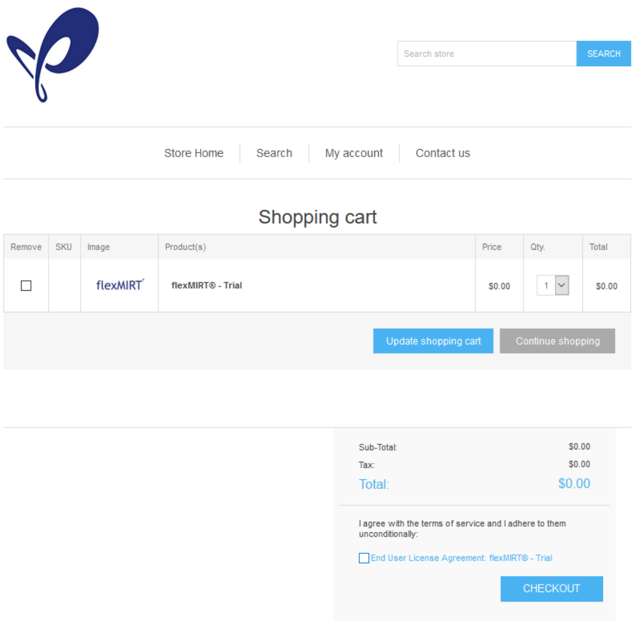
Once on the License page, click the “Buy Licenses” button - you may then use the “Add to Cart” buttons under the various flexMIRT^{TM} versions to purchase a full license (Academic or Standard) at this time or select the “flexMIRT- Trial” which has a cost of $0.00.





Once you have added you selected version of flexMIRT^{TM} to your cart, you will be able to begin the check out process. During this process, you will need to agree to the flexMIRT End User License Agreement (EULA), which can be reviewed by clicking on the blue link just above the “CHECKOUT” button. You will not be able to move forward with your purchase until the license agreement box has been checked.

Once the EULA has been accepted and the “CHECKOUT” button pushed, you will be taken to the check-out process where you will need to complete



SEARCH

Store Home | Search | My account | Contact us

Shopping cart

Remove	SKU	Image	Product(s)	Price	Qty.	Total
☐		flexMIRT®	flexMIRT® - Trial	$0.00	1	$0.00

Update shopping cart | Continue shopping

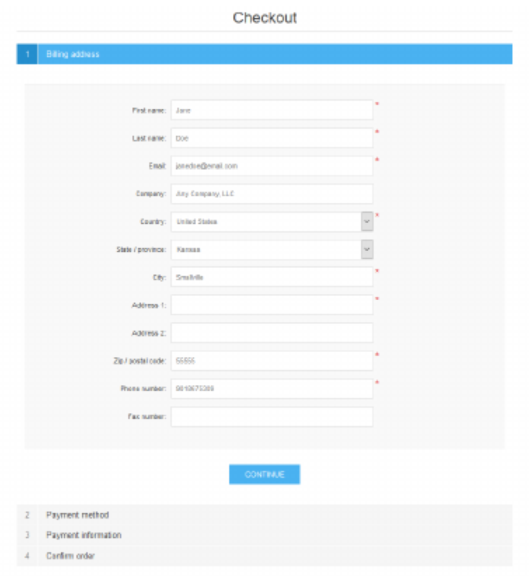
Sub-Total:	$0.00
Tax:	$0.00
Total:	$0.00

I agree with the terms of service and I adhere to them unconditionally:

☐ End User License Agreement: flexMIRT® - Trial

CHECKOUT

additional information.



Checkout

1 Billing address

First name:	Jane *
Last name:	Doe *
Email:	janedoe@email.com *
Company:	Any Company, LLC
Country:	United States *
State / province:	Kansas
City:	Smallville *
Address 1:	*
Address 2:	
Zip / postal code:	55555 *
Phone number:	9810675309 *
Fax number:	

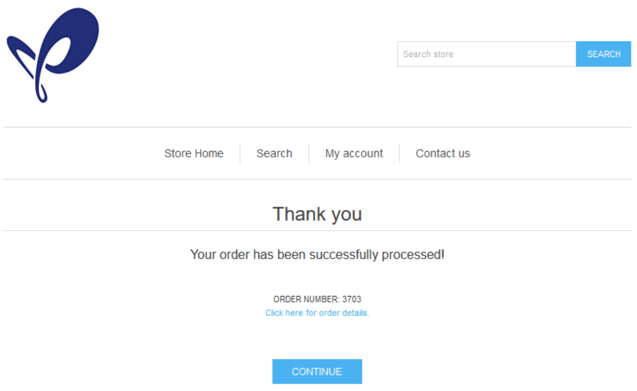
CONTINUE

2 Payment method

3 Payment information

4 Confirm order

Complete all information as necessary and use the "CONFIRM" button to complete your order. Once completed, you will be taken to an order summary page. You will also receive an email with a copy of your invoice.



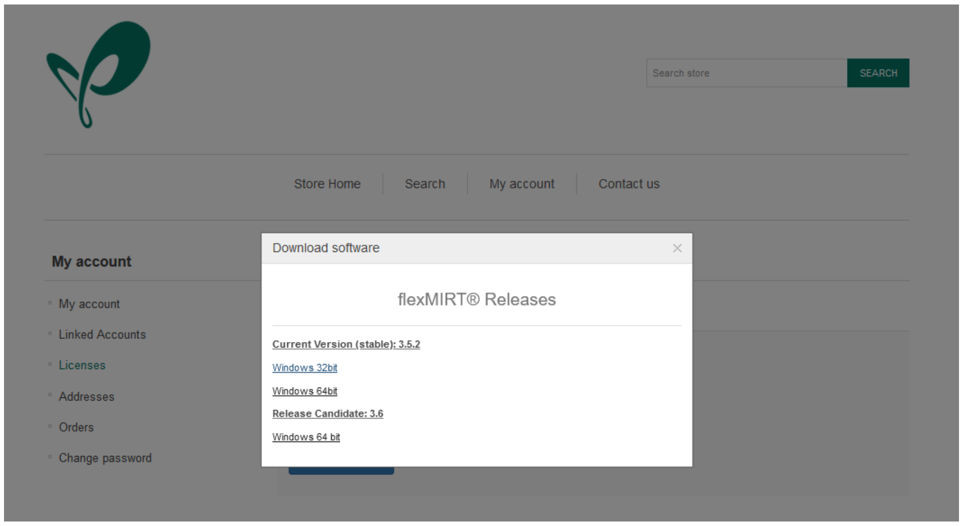
Thank you

Your order has been successfully processed!

ORDER NUMBER: 3703
Click here for order details.

Use the "My Account" link at the top of the page to return to your account home page.

Go into your Licenses page again and you should now see the license you just purchased available for use. To access the flexMIRT^{TM} installer, click the "Manage" button. After selecting Download Software, a pop-up will appear where you will select the version of flexMIRT^{TM} appropriate for your computer.



flexMIRT® Releases

Current Version (stable): 3.5.2

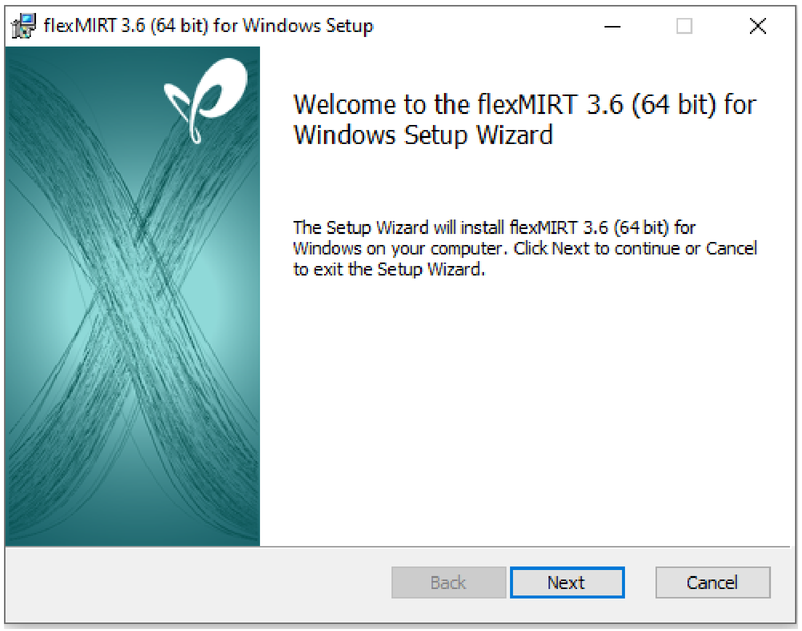
Windows 32bit

Windows 64bit

Release Candidate: 3.6

Windows 64 bit

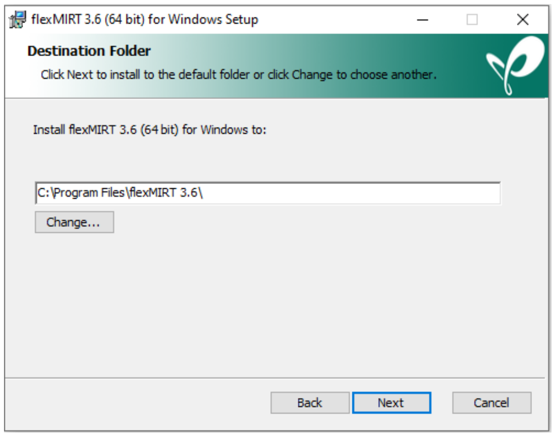
The flexMIRT^{TM} installer should be saved to a folder where you will be able to locate it later. Locate the downloaded installer (flexMIRTInstaller.msi) and double-click to activate it. Click "Next" to begin the flexMIRT^{TM} installation process.


flexMIRT 3.6 (64 bit) for Windows Setup

Welcome to the flexMIRT 3.6 (64 bit) for Windows Setup Wizard

The Setup Wizard will install flexMIRT 3.6 (64 bit) for Windows on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

After agreeing to license terms, the next step of the installer will ask which folder you wish to install the program into. If you would like flexMIRT^{TM} installed somewhere other than the default location, this can be accomplished by selecting the "Change" button and directing the installer to the desired folder location.


flexMIRT 3.6 (64 bit) for Windows Setup

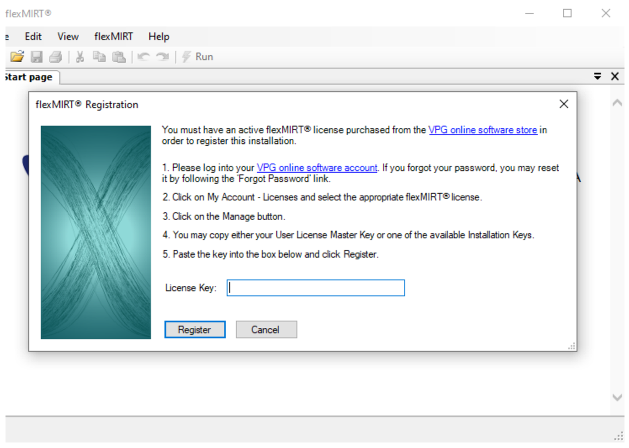
Destination Folder

Click Next to install to the default folder or click Change to choose another.

Install flexMIRT 3.6 (64 bit) for Windows to:

C:\Program Files\flexMIRT 3.6\

Follow the remaining steps, as directed by the installer, until the program indicates that it has finished installing. Once flexMIRT^{TM} have been installed, a shortcut will appear on the desktop. Double-click this to initiate the program. On the first start-up flexMIRT^{TM} will open a pane where you must register the software; you must supply an installation code associated withyou flexMIRT^{TM} license, which can be obtained from the VPG Store following the instructions on the pop-up.



flexMIRT® Registration

You must have an active flexMIRT® license purchased from the VPG online software store in order to register this installation.

1. Please log into your VPG online software account. If you forgot your password, you may reset it by following the 'Forgot Password' link.

2. Click on My Account - Licenses and select the appropriate flexMIRT® license.

3. Click on the Manage button.

4. You may copy either your User License Master Key or one of the available Installation Keys.

5. Paste the key into the box below and click Register.

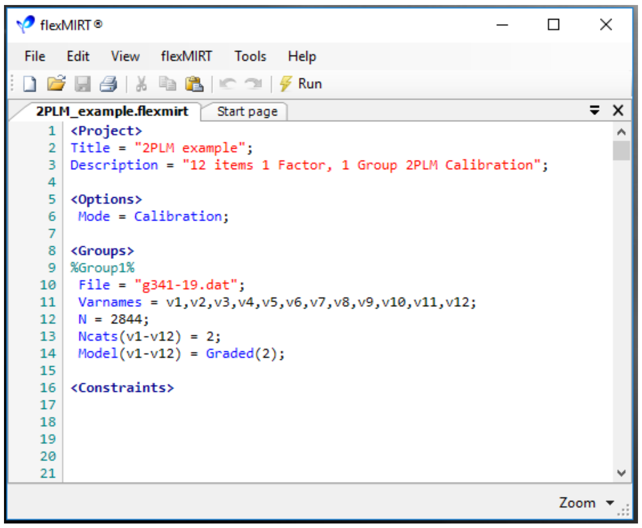
License Key:

A.2. Using flexMIRT

With the software registered, you are now able to conduct analyses. The flexMIRT^{TM} support page (`https://vpgcentral.com/software/flexmirt/support-v3-72/`) has the User's Manual and numerous example syntax command files with accompanying datasets, to help you become acquainted with the types of analyses that may be conducted and how command files should be structured. By selecting "New" under "File"' or using the "New" icon, a command file containing necessary statements for a basic analysis opens. Once you have modified this command file, it must be saved prior to being able to submit the analysis to flexMIRT$^{©}$; the "Run" button is disabled until the file has been saved to prevent the example code from being over-written.

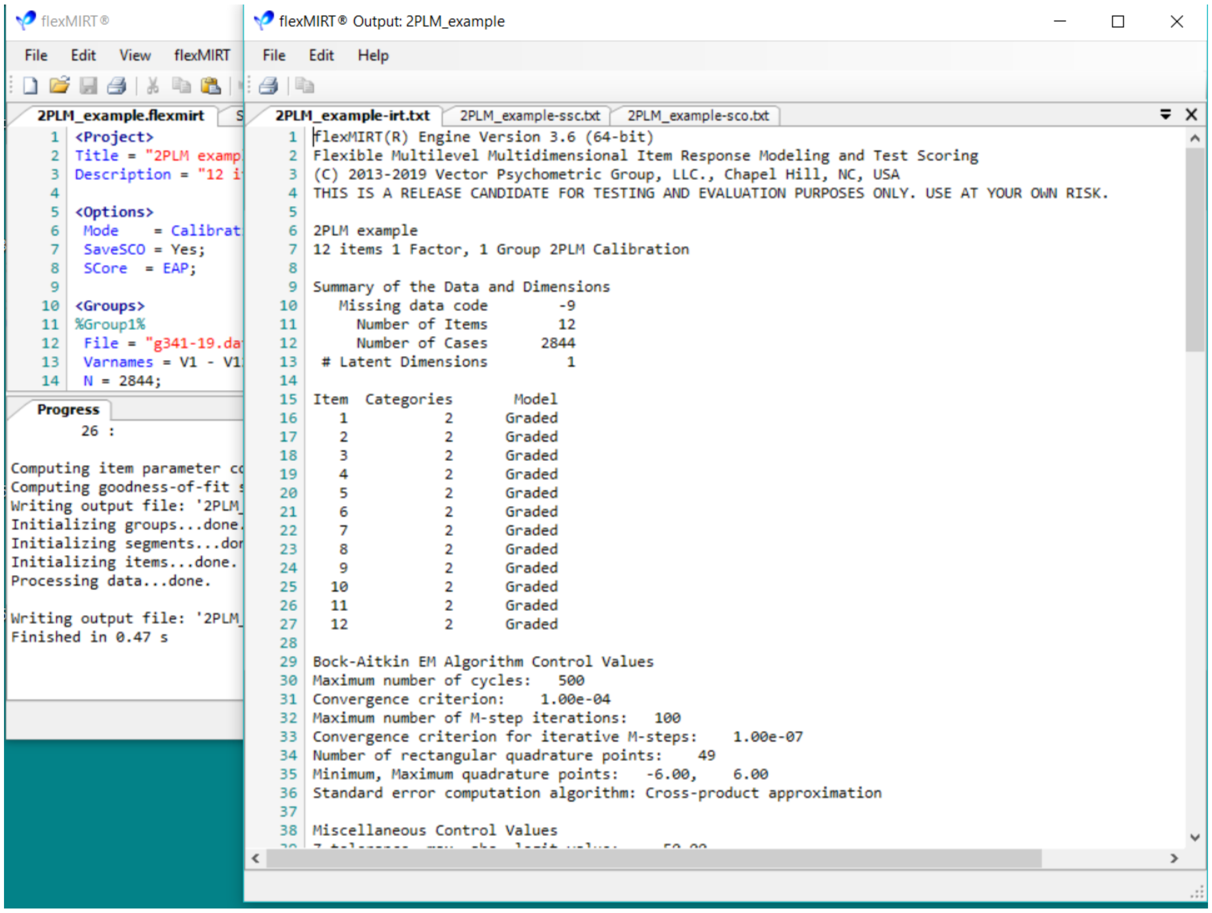
To provide a brief tutorial, we will focus on one of the first examples, found in the folder Example 3-2. Existing files are opened using the Open command under the File option of the processing pane, which is the pane opened when the program is first initiated. Once the desired flexMIRT^{TM} command file is located and selected, it will be opened in a separate pane, labeled Syntax Editor: with the file name listed after.

The command file is submitted for analyses by clicking the "Run" button on the far right of the icon toolbar. flexMIRT^{TM} will print its progress (e.g., EM iteration number, etc.) in the Progess pane that will appear beneath the



```
<Project>
Title = "2PLM example";
Description = "12 items 1 Factor, 1 Group 2PLM Calibration";

<Options>
 Mode = Calibration;

<Groups>
%Group1%
 File = "g341-19.dat";
 Varnames = v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12;
 N = 2844;
 Ncats(v1-v12) = 2;
 Model(v1-v12) = Graded(2);

<Constraints>
```

syntax and, when complete, will open the Output Viewer pane, which will contain all results. Additionally, the results are written to an external text file, which may be opened at a later time with flexMIRT$^{\text{TM}}$ or a text editor such as Notepad.



```
<Project>
Title = "2PLM examp
Description = "12 i

<Options>
 Mode     = Calibrat
 SaveSCO = Yes;
 SCore   = EAP;

<Groups>
%Group1%
 File = "g341-19.da
 Varnames = V1 - V1
 N = 2844;
```

```
      26 :

Computing item parameter co
Computing goodness-of-fit s
Writing output file: '2PLM_
Initializing groups...done.
Initializing segments...dor
Initializing items...done.
Processing data...done.

Writing output file: '2PLM_
Finished in 0.47 s
```

```
flexMIRT(R) Engine Version 3.6 (64-bit)
Flexible Multilevel Multidimensional Item Response Modeling and Test Scoring
(C) 2013-2019 Vector Psychometric Group, LLC., Chapel Hill, NC, USA
THIS IS A RELEASE CANDIDATE FOR TESTING AND EVALUATION PURPOSES ONLY. USE AT YOUR OWN RISK.

2PLM example
12 items 1 Factor, 1 Group 2PLM Calibration

Summary of the Data and Dimensions
   Missing data code         -9
     Number of Items         12
     Number of Cases       2844
 # Latent Dimensions          1

Item  Categories       Model
   1           2      Graded
   2           2      Graded
   3           2      Graded
   4           2      Graded
   5           2      Graded
   6           2      Graded
   7           2      Graded
   8           2      Graded
   9           2      Graded
  10           2      Graded
  11           2      Graded
  12           2      Graded

Bock-Aitkin EM Algorithm Control Values
Maximum number of cycles:    500
Convergence criterion:     1.00e-04
Maximum number of M-step iterations:    100
Convergence criterion for iterative M-steps:    1.00e-07
Number of rectangular quadrature points:    49
Minimum, Maximum quadrature points:   -6.00,     6.00
Standard error computation algorithm: Cross-product approximation

Miscellaneous Control Values
```

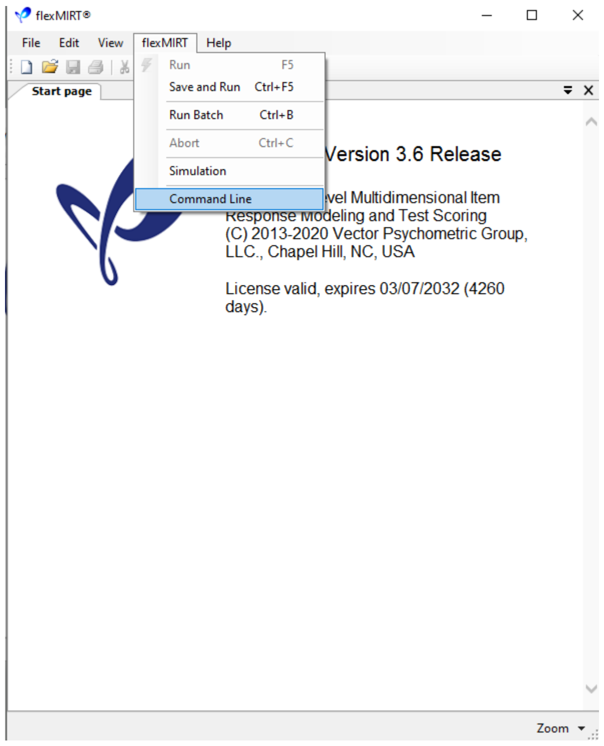
APPENDIX B

flexMIRT via the Command Line Interface

This appendix provides a brief overview of using flexMIRT^{TM} via the Command Line Interface (CLI)/Command Prompt, which may be useful to those wishing to batch submit a large number of files or to control flexMIRT^{TM} via another program (e.g., R).

B.1. flexMIRT via the CLI - single file run

Begin by accessing the Command Prompt - a short cut can be found under the flexMIRT menu (see below).

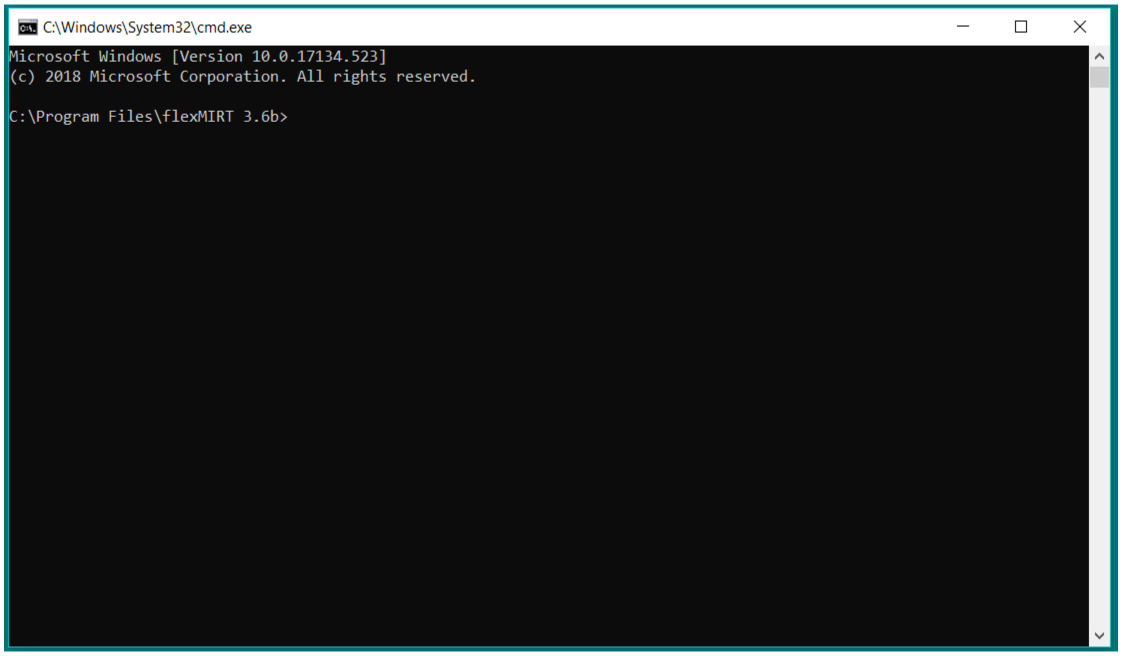


Version 3.6 Release

...vel Multidimensional Item Response Modeling and Test Scoring
(C) 2013-2020 Vector Psychometric Group, LLC., Chapel Hill, NC, USA

License valid, expires 03/07/2032 (4260 days).

The Command Prompt window will open and, for an individial run, all that remains is to supply the command file name with the appropriate WinFlexMIRT call.


C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Program Files\flexMIRT 3.6b>
```

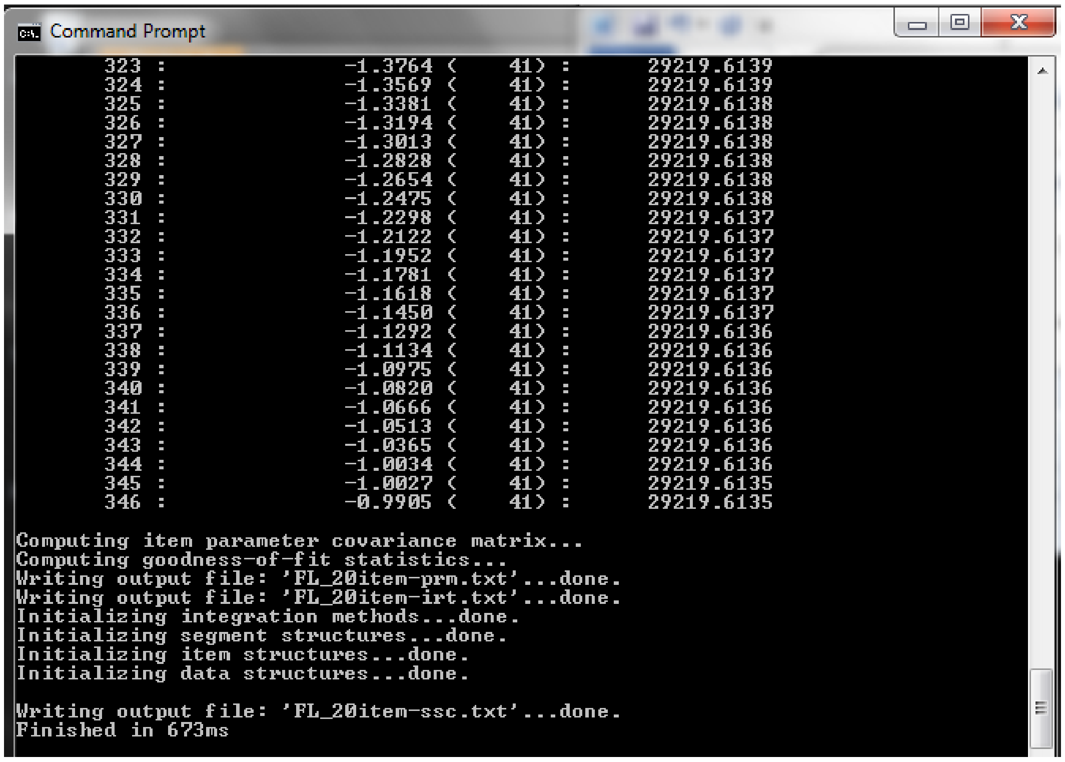
If the command file is found in the same folder as the executable, the call is of the form:

```
winflexmirt -r mysyntax.flexmirt
```

where the "`-r`" tells WinFlexMIRT (the name of the graphical user interface[GUI]) to run in the CLI and `mysyntax.flexmirt` is the command file you wish to run. If the command file is found in a different folder or has spaces in the file name then the full path to the command file must be given in quotation marks. For example, if we wished to submit the command file "FL_20item.flexmirt" for analysis and it is located in the folder "C:\Users\VPG\Documents\FL analyses\Final scale\", the WinFlexMIRT call would be:

```
winflexmirt -r "C:\Users\VPG\Documents\FL analyses\Final
scale\FL_20item.flexmirt"
```

Once a command file has been successfully submitted, the same progress information that is shown in the progress pane of the flexMIRT^{TM} GUI will be printed in the CLI.



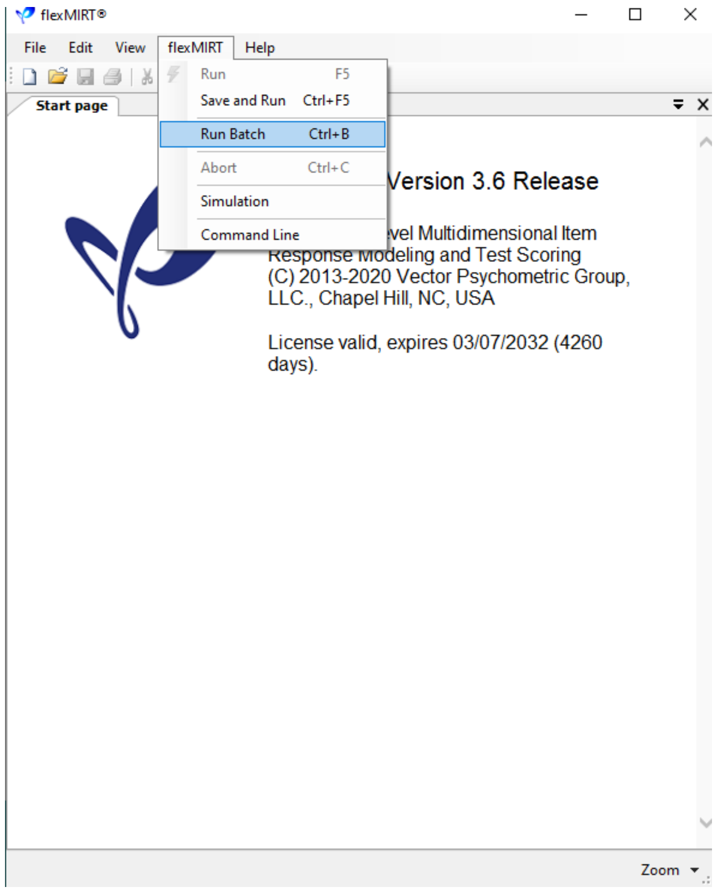
Command Prompt

```
        323 :                -1.3764 (     41) :       29219.6139
        324 :                -1.3569 (     41) :       29219.6139
        325 :                -1.3381 (     41) :       29219.6138
        326 :                -1.3194 (     41) :       29219.6138
        327 :                -1.3013 (     41) :       29219.6138
        328 :                -1.2828 (     41) :       29219.6138
        329 :                -1.2654 (     41) :       29219.6138
        330 :                -1.2475 (     41) :       29219.6138
        331 :                -1.2298 (     41) :       29219.6137
        332 :                -1.2122 (     41) :       29219.6137
        333 :                -1.1952 (     41) :       29219.6137
        334 :                -1.1781 (     41) :       29219.6137
        335 :                -1.1618 (     41) :       29219.6137
        336 :                -1.1450 (     41) :       29219.6137
        337 :                -1.1292 (     41) :       29219.6136
        338 :                -1.1134 (     41) :       29219.6136
        339 :                -1.0975 (     41) :       29219.6136
        340 :                -1.0820 (     41) :       29219.6136
        341 :                -1.0666 (     41) :       29219.6136
        342 :                -1.0513 (     41) :       29219.6136
        343 :                -1.0365 (     41) :       29219.6136
        344 :                -1.0034 (     41) :       29219.6136
        345 :                -1.0027 (     41) :       29219.6135
        346 :                -0.9905 (     41) :       29219.6135

Computing item parameter covariance matrix...
Computing goodness-of-fit statistics...
Writing output file: 'FL_20item-prm.txt'...done.
Writing output file: 'FL_20item-irt.txt'...done.
Initializing integration methods...done.
Initializing segment structures...done.
Initializing item structures...done.
Initializing data structures...done.

Writing output file: 'FL_20item-ssc.txt'...done.
Finished in 673ms
```

When the run has completed successfully, the output files may be found in the folder in which the syntax file is located.

B.2. flexMIRT batch mode

flexMIRT^{TM} also allows for users to submit multiple syntax files via a single batch file. Batch processing can be accessed via the GUI, under the flexMIRT menu.

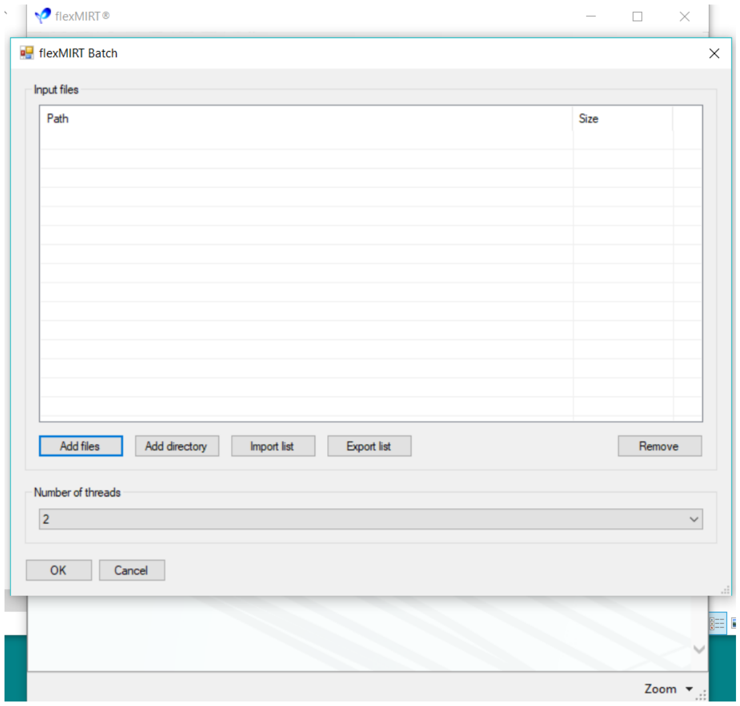


Version 3.6 Release

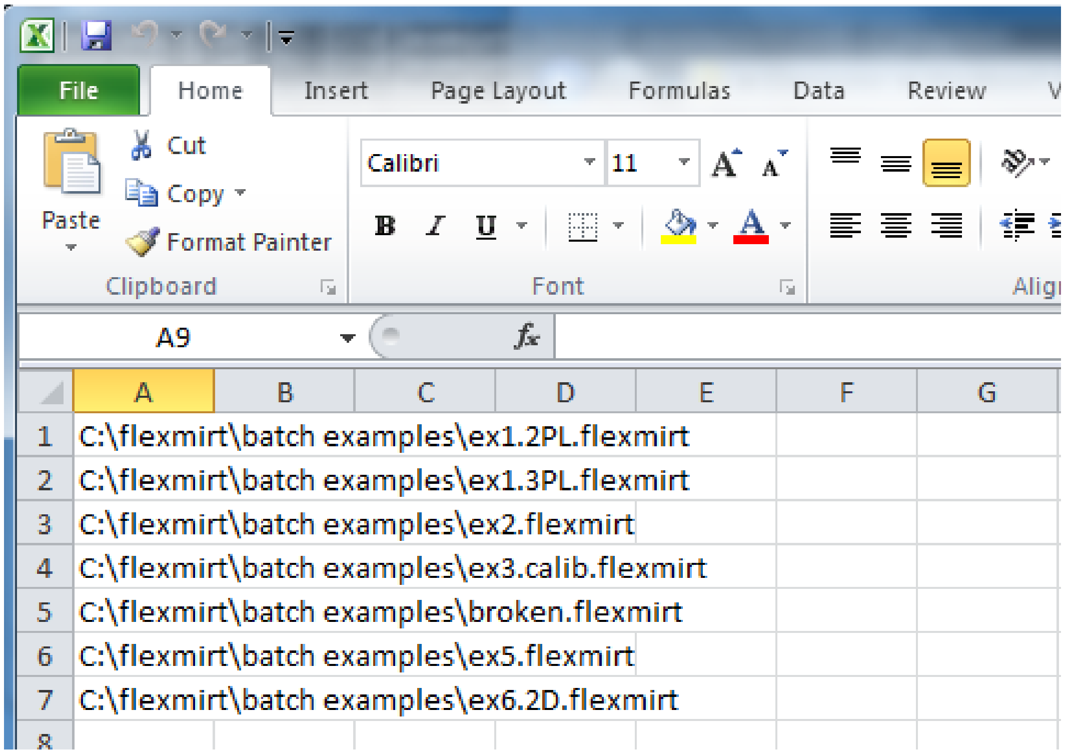
...vel Multidimensional Item Response Modeling and Test Scoring
(C) 2013-2020 Vector Psychometric Group, LLC., Chapel Hill, NC, USA

License valid, expires 03/07/2032 (4260 days).

which will open a new pop-up window.

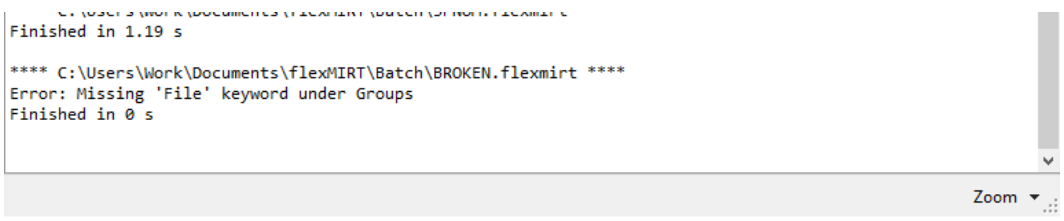


Users may constuct their batch file by selecting individual files (via the "Add files" button), selecting whole folders (via the "Add directory" button) which will add any *.flexmirt syntax files in the folder to the Input Files list, or by using the "Import List" button. When using the "Import List" button to import an existing batch file, the file should consist of a single column of syntax files, in which the user provides the full path and file names of the command files to be run. When importing an batch file, users should ensure that the flexMIRT^{TM} syntax files in the batch file ALWAYS be referenced with the full path. Below is an example batch file, saved as a CSV in Excel. Plain text files (*.txt) are also permissable and should be arranged such that each line contains one syntax file path and name.



	A
1	C:\flexmirt\batch examples\ex1.2PL.flexmirt
2	C:\flexmirt\batch examples\ex1.3PL.flexmirt
3	C:\flexmirt\batch examples\ex2.flexmirt
4	C:\flexmirt\batch examples\ex3.calib.flexmirt
5	C:\flexmirt\batch examples\broken.flexmirt
6	C:\flexmirt\batch examples\ex5.flexmirt
7	C:\flexmirt\batch examples\ex6.2D.flexmirt

Once the batch file has been submitted, progress will be printed in the Progress pane, as with single analysis runs. If a number greater then 1 is selected in the "Number of Threads" box, then multiple Progress panes will be opened. Once flexMIRT^{TM} has processed all of the syntax files named in the batch, it will report that it has finished the batch and print the total time used. Users should review all of the Progress pane printing after batch completion, paying attention to any files in which an issue was encountered, such as the error reported below for the BROKEN.flexmirt syntax file from a completed batch run.



```
Finished in 1.19 s

**** C:\Users\Work\Documents\flexMIRT\Batch\BROKEN.flexmirt ****
Error: Missing 'File' keyword under Groups
Finished in 0 s
```

The typical output files for the individual runs submitted in batch mode will be in the same folder as the corresponding *.flexmirt syntax file.

APPENDIX C

flexMIRT$^{\text{TM}}$ in an Operational Context

To assist users working with flexMIRT$^{\text{TM}}$ in an operational setting, where calibration and/or scoring runs may include hundreds of items and many thousands of observations, this appendix describes and demonstrates settings that can be optimized for large problems, as well discuss other features of flexMIRT$^{\text{TM}}$ that may be useful in an operational setting or that are commonly used in an large-scale testing situation.

C.1. Optimizing Estimation Speed

When working with very large item sets and/or sample sizes, it becomes important to optimatize estimation processes so that analyses can be completed in a timely fashion. Below we demostrate with three examples, based on the same simulated dataset of 400 items and 190,000 observations, the time savings that may be achieved, depending on the flexMIRT$^{\text{TM}}$ settings chosen and the model being used.

Example C-1: Speed Optimatization Examples - Base syntax

```
<Project>
Title =  "Fit a UIRT 2PL Model";
Description= "Using legacy BAEM multi-threading method";

<Options>
Mode = Calibration;
SE=Xpd;
Processors = 4;
NewThreadModel = No; //default value is Yes

```

```
<Groups>
%Grade2%
File ="Group.dat";
Varnames = v1-v400;
N=190000;
Ncats(v1-v400) = 2;
Model(v1-v400) = Graded;

<Constraints>
```

The `Processors` statement lets the user determine how many processors (threads) flexMIRT^{TM} can utilize; the maximum number of possible threads is determined by the number of processing cores in your system. When multiple cores are specified for flexMIRT^{TM} to use, the estimation process will be distributed across those cores/processing units, which can result in the increased estimation speed/reduced wait time before output files are available. However, the Bock-Aitkin EM algorithm isn't easy to parallelize because there is quite a bit of synchronization needed between E- and M-step; there is usually a barrier so that processors must wait for all members of the team to finish before proceeding to the next task. That being said, the originally implemented multiple-core processing model employed when `NewThreadModel = No;` may be optimal for low dimensional analyses with small to moderate N.

Example C-2: Speed Optimatization - Modification 1

```
<Project>
Title =  "Fit a UIRT 2PL Model";
Description= "Using updated BAEM multi-threading
method for large data sets";

<Options>
Mode = Calibration;
SE=Xpd;
Processors = 4;
NewThreadModel = Yes; // This is the default

<Groups>
%Grade2%
```

```
File ="Group.dat";
Varnames = v1-v400;
N=190000;
Ncats(v1-v400) = 2;
Model(v1-v400) = Graded;

<Constraints>
```

In the first modification of our speed optimatization example, we have updated only the `NewThreadModel` option from `No` to `Yes`. When `NewThreadModel = Yes;` an updated multi-core processing model is employed. The new multi-threading model is an updated implementation of multi-core processing that may be more efficient for very high-dimensional models with a large number of quadrature points to evaluate. The level of granularity is smaller than the legacy threading model implemented in flexMIRT^{TM}. In general, the legacy threading model trades more memory for speed, and this new model does not do that nearly as much, so the parallel speed-up may not be as significant as the classical approach for low dimensional analyses. When the `NewThreadModel` is active, `Fine (may be more optimal for high dimensionality)` will be printed in the -irt output file to describe the `Parallelization granularity`, while it will be reported as `Coarse` when the original threading model is used. Note that the NewThreadModel option is specific to `Algorithm = BAEM` and will have no effect on analyses using MCMC or MH-RM estimation.

Example C-3: Speed Optimatization - Modification 2

```
<Project>
Title =  "Fit a UIRT 2PL Model";
Description= "Using updated BAEM multi-threading
method and SparseMartix command";

<Options>
Mode = Calibration;
SE=Xpd;
Processors = 4;
NewThreadModel = Yes; // This is the default
SparseMatrix = Yes; // No is the default

```

```
<Groups>
%Grade2%
File ="Group.dat";
Varnames = v1-v400;
N=190000;
Ncats(v1-v400) = 2;
Model(v1-v400) = Graded;

<Constraints>
```

In this example, in addition to the new threading method being employed, the SparseMatrix command in the `<Options>` has also been used. The command `SparseMatrix` is implemented primarily for use in large-scale testing situations - when `SparseMatrix = Yes;` is used in conjunction with the default cross-product approximation method for estimating SEs, flexMIRT$^{\text{TM}}$ will assume a large and sparse information matrix, leading to time-savings for large calibration runs. When `SparseMatrix = Yes;` is used *in combination with the new multi-threading model*, parallel speed-up for very large N and many (read, thousands of) items may be much more significant than the legacy model, due to additional optimizations that improve memory utilization and work-sharing efficiency. The `SparseMatrix` option has no effect on models with Primary, Between, and/or Cluster keywords in the current implementation. The `SparseMatrix = Yes;` setting will be indicated in the -irt file on the line reporting the standard error computation algorithm as: `Cross-product (sparse matrix approximation)`. User are encouraged to experiment with the threading options. At present, the use of the `SparseMatrix` command will suppress GOF output to Basic reporting level to avoid costly computations necessary for the derivation of GOF index values.

Table C.1: Time Comparisons (in seconds) of Thread Models and Use of SparseMatrix command

Processing Step	Base (NewThreadModel = No;)	Modification 1 (NewThreadModel = Yes;)	Modification 2 (NewThreadModel = Yes and SparseMatrix = Yes;)
E-step computations	39.25	95.77	22.64
M-step computations	0.46	0.21	0.24
Standard error computations	102.67	105.66	14.09
Goodness-of-fit statistics	0	0	0
Total	142.38	201.64	36.98

As can be seen in the above table, especially for the large n, large K, low dimensional analyses like the one shown in the examples, there can be significant time savings in the SE computation step through the combined use of `SparseMatrix = Yes;` with `NewThreadModel = Yes;` in the `<Options>` of the syntax. It can also be seen that the `NewThreadModel` may not always the best option in terms of efficiency for BAEM estimation; this is why users are encouraged to experiment with settings to identify the most efficient settings for each analysis.

C.2. Score Conversion Table from Existing Parameters

A feature that may be useful in the operational context is the creation of a summed to EAP score conversion (SSC) table (e.g., Cai, 2015) that provides the most likely EAP score associated with each possible summed score that can be obtained from the set of items. This SSC table can be generated by flexMIRT^{TM} without an associated datafile. The syntax below demonstrate the syntax for such a run, which only requires that the existing item parameters are provided in the expected -prm file structure (see The Parameter File Layout section of the Simulation chapter and the Labeled Output Files Appendix for additional details on -prm file layout).

Example C-4: SSC Table from Existing Parameters

```
<Project>
Title = "SSC Table from item prms";
```

```
Description= "No data file";

<Options>
Mode = Scoring;
Score=SSC;

ReadPRMFile= "Existing_prms.txt";

<Groups>
%G1%
Varnames = v1-v400;

<Constraints>

```

Note that in the syntax, we do not provide a data file name in the `<Groups>` section, nor do we need to provide information related to the number of categories or item models for each item, as that information is provided to flexMIRT^{TM} through the parameter file, via the `ReadPRMFile` statement in the `<Options>` section of the syntax. However, users do need to ensure that the group and variable names provided in the -prm file are the same as those used in the syntax. Note that it is also possible to construct a SSC table from a subset of items in the parameter file; the desired items would be specified via a `Select` statement in the `<Group>` secton of the syntax. **Users are reminded that the SSC tables are only valid for deriving scores for observations where all items have non-missing responses.**

C.3. ML Scoring Considerations

In operational contexts, it is often mandated that a certain method for calculating person-estimates (i.e., scores) should be used and at some testing / certification agencies, the preferred method of scoring is the use of maximum likelihood estimates. ML scores can be requested from flexMIRT^{TM} by setting `Score = ML;` in the `<Options>` section of your syntax.

When using ML scoring, there are two additional commands that must be employed. Users are required to specify the desired maximum and minimum scores to be assigned by flexMIRT^{TM} using the `MaxMLscore` and `MinMLscore` keywords, respectively. There are no default values for the ML minimum and maximum scores and the program will produce an error if `Score = ML;` is used without also setting the minimum and maximum allowed values.

Efforts have been made to make ML scoring as robust as possible, but it is not recommended for use with multi-dimensional models. Because ML scoring does not use information from the population distribution, essential statistical information about the population distribution (e.g., factor inter-correlations, means, and variances) is ignored when scoring individual dimensions of MIRT models. Additionally, ML scoring can lead to score information matrices that are not positive definite, making the SEs for some estimates undefined.

APPENDIX D

Plotting

In unidimensional IRT, graphs (e.g., tracelines and information curves) form an integral component of item analysis in practice. flexMIRT^{TM} is built to handle multilevel and multidimensional IRT models. In this context, there currently exists no universally accepted consensus with regards to graphical procedures. We do realize, however, that there is a need for publication quality graphics and to meet that need flexMIRT^{TM} will now produce upon request, for unidimensional models only, a separate output file, the *-icc.txt, which contains the values necessary to construct tracelines in the user's preferred plotting program. **Please note that the -icc output file is only available for single factor models.**

D.1. Plotting ICCs and TCCs

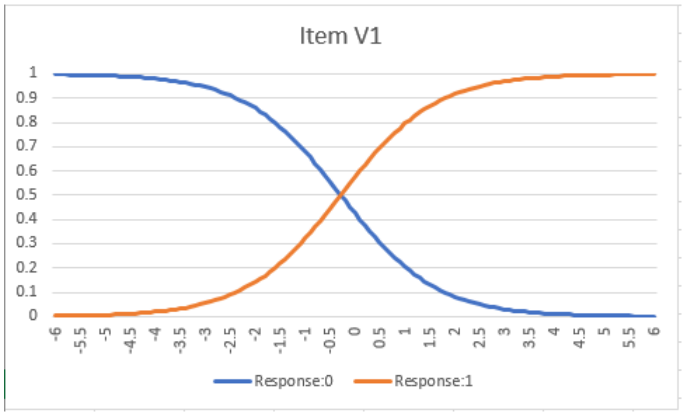
By including `SaveICC = Yes;` in the `<Options>` section of your syntax, users may request that the -icc.txt file be saved. The available -icc file contains values for each item and the overall test/scale which may be used to construct traceline plots. These values are calculated at 121 points over theta value ranging from -6 to 6 (and are evenly spaced 0.1 apart: $-6.0, -5.9, -5.8, ..., 6.0$); the range and number of points reported in the -icc output file is not able to modified by users.

We provide a labeled -icc file from the first example (2PLM_example.flexmirt) to demonstrate the meaning of each column.

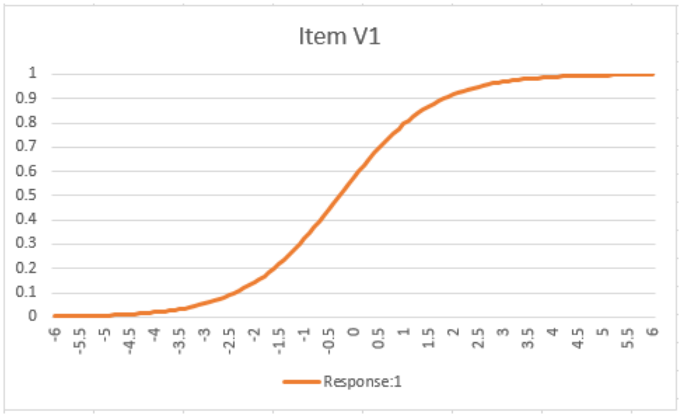
Table D.1: Labeled -icc file (excerpt)

Grp #	Item #	Label	Response Category	-6.0	-5.9	...
1	1	v1	0	0.9975	0.9973	
1	1	v1	1	0.0025	0.0027	
1	2	v2	0	0.9887	0.9873	
1	2	v2	1	0.0113	0.0127	
1	3	v3	0	0.9938	0.9933	
1	3	v3	1	0.0062	0.0067	
1	4	v4	0	0.9951	0.9946	
1	4	v4	1	0.0049	0.0054	
...						
1	10	v10	0	0.996	0.9954	
1	10	v10	1	0.004	0.0046	
1	11	v11	0	0.9958	0.9954	
1	11	v11	1	0.0042	0.0046	
1	12	v12	0	0.9973	0.997	
1	12	v12	1	0.0027	0.003	
1	TCC	Group1	0	0.0613	0.068	

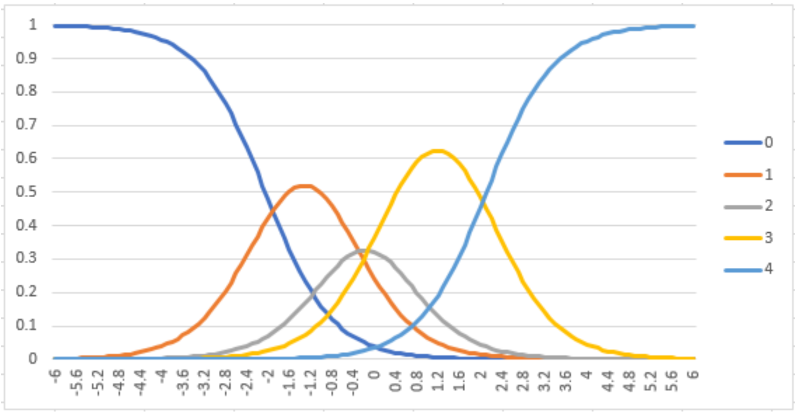
These values can be directly exported to the graphing program of your choice and item characteristic curves (ICCs)/traceline plots are easily constructed, with no additional calculations needed for standard plots. Using the full set of values provided for the dichotmous item V1, the below plot was made in Excel.



To obtain a more typical trace line plot for V1, we can exclude the 0-response category line such as below.



When requested in conjunction with polytomous models, the values in the -icc output file may also be used directly to create category response function plots, such as below (using item V1 of the previous NCS example that used the GRM).



Test characteristic curves are made by plotting the values in the line labeled "TCC" in a similar fashion.

D.2. Plotting IIFs and TIFs

Upon request, flexMIRT^{TM} will also output item and test information function values to a separate *-inf.txt output file. To request information be saved, users must include `SaveINF = Yes;` in the `<Options>` section of their syntax. The default number of points to be written to the -inf is a single point at theta =

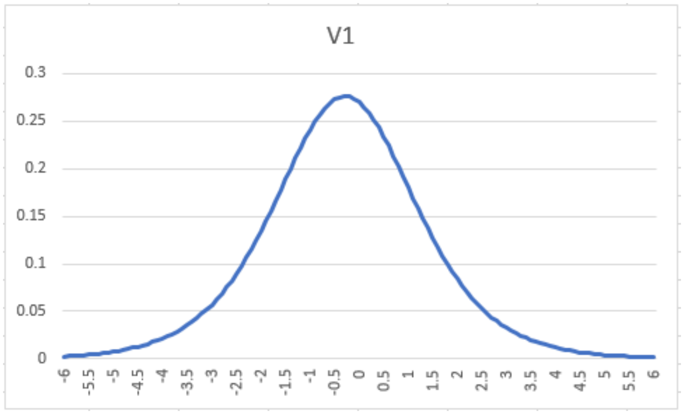
0; to construct a comprehensive information function plot, users will need to also include the `FisherInf = (?,?);` statement, which will tell flexMIRT at how many points and spread across what range of theta information should be calculated. For instance, to match the theta values used in the -icc file, the user would specify `FisherInf = (121,6);` Because flexMIRT^{TM} will not label the theta values in the -inf file, it is important for users to know how they are deteremined: Regardless of the number of points and the minimum/maximum value specified, flexMIRT^{TM} will always place the theta points evenly across the range - note that many choices can result in theta values that are spaced at non-round intervals. For instance, `FisherInf = (14,4);` would result in the 14 points being spaced from -4 to 4 at intervals of (4+4)/(14-1) = 0.61538.

We present a labeled version of the -inf from the first syntax example below. Note that "P" is the label used for the test information values.

Table D.2: Labeled -inf file (excerpt)

Grp #	Item #	theta1	theta2	...
1	1	0.0027	0.003	
1	2	0.0167	0.0188	
1	3	0.0043	0.0047	
1	4	0.0053	0.0059	
...				
1	9	0.0012	0.0015	
1	10	0.0069	0.0079	
1	11	0.0032	0.0034	
1	12	0.0028	0.0031	
1	P	1.0691	1.0773	

With the -inf obtained from flexMIRT^{TM}, the information function values transferred into the plotting program of choice, and the theta values determined, item information functions (IIFs) and the test information function (TIF) may be easily plotted. Porting the -inf values from the first example into Excel, we are able to produce the below IIF for item V1 without requiring any additional calculations.



The TIF plot may be created in a similar manner, using the values from the line labeled e “P” of the -inf output file.

APPENDIX E

Labeled Output Files

Due to the lack of labels in some of the output files that flexMIRT^{TM} creates, we provide this appendix to give labeled examples of several file examples, specifically -sco and -prm files. For each example, we will provide labels at the top of each column, which would not normally appear, and a small selection of lines from the output file.

Table E.1: Labeled -sco file - EAP scores, 1 Factor

Grp#	flexMIRT Obs #	$\hat{\theta}$	SE($\hat{\theta}$)
1	1	-0.179	0.501
1	2	0.957	0.609
1	3	0.033	0.576
1	4	0.957	0.609
1	5	-0.519	0.515

Table E.2: Labeled -sco file - MAP scores, 1 Factor

Grp#	flexMIRT Obs #	# iterations to MAP	$\hat{\theta}$	SE($\hat{\theta}$)
1	1	2	-0.150	0.494
1	2	3	0.939	0.564
1	3	2	0.077	0.567
1	4	3	0.939	0.564
1	5	2	-0.489	0.518

Note: Scores from maximum likelihood (`Score = ML;`) will have a similar "# of iterations to score" column.

Table E.3: Labeled -sco file - MI scores, 1 Factor

Imputation #	Grp#	flexMIRT Obs #	$\hat{\theta}$
1	1	1	-0.504647
1	1	2	1.601748
1	1	3	4.335706
1	1	4	1.212268
1	1	5	1.015626

Table E.4: Labeled -sco file - EAP scores, 2 Factors

Grp#	flexMIRT Obs #	$\hat{\theta}_1$	$\hat{\theta}_2$	SE($\hat{\theta}_1$)	SE($\hat{\theta}_2$)	$\hat{\sigma}_{11}$	$\hat{\sigma}_{21}$	$\hat{\sigma}_{22}$
1	1	-0.141	-0.466	0.258	0.355	0.066705	0.001763	0.126136
1	2	0.192	1.407	0.281	0.295	0.079119	0.001470	0.087174
1	3	2.023	1.684	0.527	0.392	0.277999	0.015970	0.153786
1	4	1.311	0.840	0.334	0.290	0.111558	0.003131	0.083973
1	5	0.667	1.182	0.262	0.284	0.068758	0.001159	0.080632

Table E.5: Labeled -sco file - EAP scores, 2 Factors with User-supplied ID Variable

Grp#	flexMIRT Obs #	User ID	$\hat{\theta}_1$	$\hat{\theta}_2$	SE($\hat{\theta}_1$)	SE($\hat{\theta}_2$)	$\hat{\sigma}_{11}$	$\hat{\sigma}_{21}$	$\hat{\sigma}_{22}$
1	1	sub1	-0.141	-0.466	0.258	0.355	0.066705	0.001763	0.126136
1	2	sub2	0.192	1.407	0.281	0.295	0.079119	0.001470	0.087174
1	3	sub3	2.023	1.684	0.527	0.392	0.277999	0.015970	0.153786
1	4	sub4	1.311	0.840	0.334	0.290	0.111558	0.003131	0.083973
1	5	sub5	0.667	1.182	0.262	0.284	0.068758	0.001159	0.080632

Table E.6: Labeled -sco file - MAP scores, 2 Factors

Grp#	flexMIRT Obs #	# of Iterations to MAPs	$\hat{\theta}_1$	$\hat{\theta}_2$	SE($\hat{\theta}_1$)	SE($\hat{\theta}_2$)	$\hat{\sigma}_{11}$	$\hat{\sigma}_{21}$	$\hat{\sigma}_{22}$
1	1	3	-0.146	-0.405	0.245	0.338	0.059838	0.001456	0.114251
1	2	4	0.200	1.409	0.263	0.287	0.068942	0.001183	0.082246
1	3	5	1.838	1.643	0.494	0.368	0.244128	0.011428	0.135356
1	4	4	1.261	0.831	0.296	0.275	0.087689	0.002205	0.075827
1	5	4	0.677	1.190	0.247	0.270	0.061086	0.000914	0.072911

Table E.7: Labeled -prm file - GRM with 1 Factor, 1 Group

Type	Label	Grp #	# of Factors	Model	# of Cats	c_1	c_2	c_3	c_4	a
1	v1	1	1	2	5	3.24905e+000	9.45119e-001	-4.10822e-001	-3.33909e+000	1.58287e+000
1	v2	1	1	2	5	4.35083e+000	1.79341e+000	4.76297e-001	-2.91482e+000	1.85045e+000
1	v3	1	1	2	5	4.04014e+000	1.60919e+000	2.87393e-001	-2.39810e+000	1.78740e+000
1	v4	1	1	2	5	4.11180e+000	1.64584e+000	3.54674e-001	-2.61476e+000	1.72958e+000
1	v5	1	1	2	5	4.34774e+000	1.71393e+000	4.73887e-001	-2.35381e+000	1.44323e+000
Type	Label	Grp#	# of Factors	Prior	μ_1	σ_{11}				
0	Group1	1	1	0	0.00	1.00				

Table E.8: Labeled -prm file - 2PL with 6 Factors, 1 Group

Type	Label	Grp #	# of Factors	Model	# of Cats	c	a_1	a_2	a_3	a_4	a_5	a_6
1	v1	1	6	2	2	2.12982e+000	2.26116e+000	0.00000e+000	0.00000e+000	0.00000e+000	0.00000e+000	0.00000e+000
1	v2	1	6	2	2	3.77704e+000	2.30913e+000	2.51024e+000	0.00000e+000	0.00000e+000	0.00000e+000	0.00000e+000
1	v3	1	6	2	2	1.69155e+000	1.61782e+000	1.41918e+000	0.00000e+000	0.00000e+000	0.00000e+000	0.00000e+000
1	v4	1	6	2	2	3.76464e+000	3.86443e+000	4.89228e+000	0.00000e+000	0.00000e+000	0.00000e+000	0.00000e+000
1	v5	1	6	2	2	2.86759e+000	2.76758e+000	3.08101e+000	0.00000e+000	0.00000e+000	0.00000e+000	0.00000e+000
Type	Label	Grp#	# of Factors	Prior	μ_1	$\cdots$	μ_5	σ_{11}	σ_{21}	σ_{22}	σ_{31}	$\cdots$
0	Group1	1	6	0	0.00	$\cdots$	0.00	1.00	0.00	1.00	0.00	$\cdots$

Table E.9: Labeled -prm File - 3PL with 1 Factor, 3 groups

Type	Label	Grp #	# of Factors	Model	# of Cats	logit g or c	c or a	a
1	v1	1	1	1	2	-1.33972	-0.68782	1.301362
1	v2	1	1	1	2	-0.96814	-1.29641	1.409536
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	v9	1	1	1	2	-2.79306	0.730972	1.172468
1	v10	1	1	1	2	-1.65629	-0.14263	1.102673
1	v1	2	1	1	2	-1.33972	-0.68782	1.301362
1	v2	2	1	1	2	-0.96814	-1.29641	1.409536
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	v9	2	1	1	2	-2.79306	0.144738	1.172468
1	v10	2	1	1	2	-1.65629	-0.14263	1.102673
1	v1	3	1	2	2	-0.68782	1.301362	
1	v2	3	1	1	2	-0.96814	-1.29641	1.409536
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	v9	3	1	1	2	-2.79306	0.730972	1.172468
1	v10	3	1	1	2	-1.65629	-0.14263	1.102673
Type	Label	Grp#	# of Factors	Prior	μ_1	σ_{11}		
0	Group1	1	1	0	0	1		
0	Group2	2	1	0	0.2	1		
0	Group3	3	1	0	-0.2	1.5		

Note that for item v1 in group 3, a different model (the 2PL, rather than the 3PL) is used. This is why the item parameter columns "logit g or c" and "c or a" are labeled as they are; the columns have different meanings, depending on the model associated with an item.

Table E.10: Labeled -cov file

P#	1	2	3	4	5	6	7	8	9	10
1	4.17345e-02	4.09706e-02	2.59980e-04	-2.95393e-03	4.04479e-04	-2.74287e-03	2.55595e-04	-2.28171e-03	5.17929e-06	-1.00769e-03
2	4.09706e-02	6.49320e-02	-7.64639e-04	-5.74476e-03	-3.38999e-04	-5.18577e-03	-6.35205e-04	-3.02963e-03	-5.71362e-04	4.69316e-04
3	2.59980e-04	-7.64639e-04	8.08072e-03	8.74185e-03	3.85735e-04	-3.05882e-03	2.02028e-04	-6.85395e-04	6.19991e-05	-8.16099e-04
4	-2.95393e-03	-5.74476e-03	8.74185e-03	3.48221e-02	-9.91978e-04	-1.12891e-02	-1.04386e-03	-2.86823e-04	-1.16237e-03	-3.72284e-03
5	4.04479e-04	-3.38999e-04	3.85735e-04	-9.91978e-04	5.84921e-03	3.83641e-03	4.37631e-04	-6.07546e-04	5.53338e-04	-3.24452e-04
6	-2.74287e-03	-5.18577e-03	-3.05882e-03	-1.12891e-02	3.83641e-03	5.41417e-02	-2.99643e-03	-1.14565e-02	-3.26232e-03	-4.33139e-03
7	2.55595e-04	-6.35205e-04	2.02028e-04	-1.04386e-03	4.37631e-04	-2.99643e-03	9.66083e-03	1.03843e-02	8.23114e-05	-7.89883e-04
8	-2.28171e-03	-3.02963e-03	-6.85395e-04	-2.86823e-04	-6.07546e-04	-1.14565e-02	1.03843e-02	3.42259e-02	-7.30998e-04	-3.87220e-03
9	5.17929e-06	-5.71362e-04	6.19991e-05	-1.16237e-03	5.53338e-04	-3.26232e-03	8.23114e-05	-7.30998e-04	1.73943e-02	1.75030e-02
10	-1.00769e-03	4.69316e-04	-8.16099e-04	-3.72284e-03	-3.24452e-04	-4.33139e-03	-7.89883e-04	-3.87220e-03	1.75030e-02	4.05049e-02

The error variance-covariance matrix associated with estimated parameters from a calibration can be requested from flexMIRT^{TM} by including the `SaveCOV = Yes;` statement in the `Options` of the syntax. The full error covariance matrix with comma-delimited values is provided and values are ordered based on the `P#` available in the -irt output file. That is, in the above table, the value in cell (1,1) is the error variance of the parameter in the -irt output file with a `P#` label of 1, in cell (2,1) and (1,2) the error covariance of parameter numbers 1 and 2 is reported, etc.

APPENDIX F

Licensing Information

Purchasing a license for flexMIRT^{TM} is easy. If you intend to use flexMIRT^{TM} for academic purposes, simply register at (`http://flexmirt.vpgcentral.com/Account/LogOn` and, once logged in, follow the instructions on-screen to download a trial version. At any point in your trial, you can log in to your account and purchase a flexMIRT^{TM} license. Your yearly subscription will begin the moment you purchase the program. Newer versions of flexMIRT^{TM} will be made available at no cost, provided a valid license. It is recommended that users periodically log into their flexMIRT^{TM} account and check if their version is the most current version of the program available and update if necessary.

Each single-user academic license is good for three installs of flexMIRT^{TM} so you can easily work with flexMIRT^{TM} on your academic, home, and laptop computers at no additional cost. If additional installations are required, pricing information may be obtained by contacting sales@VPGcentral.com.

Your flexMIRT^{TM} license does not automatically renew. You will be asked after your year with flexMIRT^{TM} if you would like to renew your subscription. If you desire a different subscription length, please contact sales@VPGcentral.com for pricing information.

F.1. Commercial use of flexMIRT

This version of flexMIRT^{TM} is intended for **non-commercial use only** (as defined in the flexMIRT^{TM} End User's Licensing Agreement). If you are interested in purchasing flexMIRT^{TM} for commercial use, please contact sales@VPGcentral.com.

F.2. Classroom Discounts for flexMIRT

For educators who are interested in using flexMIRT^{TM} in their classroom, bulk educational discounts are available. Please contact sales@VPGcentral.com for more information.

References

Adams, R., & Wu, M. (2002). *PISA 2000 technical report.* Paris: Organization for Economic Cooperation and Development.

Albert, J. H. (1992). Bayesian estimation of normal ogive item response curves using Gibbs sampling. *Journal of Educational Statistics*, *17*, 251-269.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, *43*, 561–573.

Bock, R. D. (1960). *Methods and applications of optimal scaling.* Chapel Hill, NC: L. L. Thurstone Psychometric Laboratory.

Bock, R. D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika*, *46*, 443–459.

Bock, R. D., Gibbons, R., & Muraki, E. (1988). Full-information item factor analysis. *Applied Psychological Measurement*, *12*, 261–280.

Bradlow, E. T., Wainer, H., & Wang, X. (1999). A Bayesian random effects model for testlets. *Psychometrika*, *64*, 153–168.

Browne, M. W. (2001). An overview of analytic rotation in exploratory factor analysis. *Multivariate Behavioral Research*, *36*, 111–150.

Browne, M. W., & Cudeck, R. (1993). Alternative ways of assessing model fit. In K. Bollen & J. Long (Eds.), *Testing structural equation models* (pp. 136–162). Newbury Park, CA: Sage.

Cacioppo, J. T., Petty, R. E., & Kao, C. F. (1984). The efficient assessment of need for cognition. *Journal of Personality Assessment*, *48*, 306–307.

Cai, L. (2008). SEM of another flavour: Two new applications of the supplemented EM algorithm. *British Journal of Mathematical and Statistical Psychology*, *61*, 309–329.

Cai, L. (2010a). A two-tier full-information item factor analysis model with applications. *Psychometrika*, *75*, 581–612.

Cai, L. (2010b). High-dimensional exploratory item factor analysis by a Metropolis-Hastings Robbins-Monro algorithm. *Psychometrika*, *75*, 33–57.

Cai, L. (2010c). Metropolis-Hastings Robbins-Monro algorithm for confirmatory item factor analysis. *Journal of Educational and Behavioral Statistics*, *35*, 307–335.

Cai, L. (2015). Lord-Wingersky algorithm version 2.0 for hierarchical item factor models with applications in test scoring, scale alignment, and model fit testing. *Psychometrika*, *80*, 535–559.

Cai, L., & Hansen, M. (2013). Limited-information goodness-of-fit testing of hierarchical item factor models. *British Journal of Mathematical and Statistical Psychology*, *66*, 245–276.

Cai, L., Yang, J. S., & Hansen, M. (2011). Generalized full-information item bifactor analysis. *Psychological Methods*, *16*, 221–248.

Celeux, G., Chauveau, D., & Diebolt, J. (1995). *On stochastic versions of the EM algorithm (Tech. Rep. No. 2514).* The French National Institute for Research in Computer Science and Control.

Celeux, G., & Diebolt, J. (1991). *A stochastic approximation type EM algorithm for the mixture problem (Tech. Rep. No. 1383).* The French National Institute for Research in Computer Science and Control.

Chen, W. H., & Thissen, D. (1997). Local dependence indices for item pairs using item response theory. *Journal of Educational and Behavioral Statistics*, *22*, 265–289.

Choi, H.-J., Rupp, A. A., & Pan, M. (2013). Standardized diagnostic assessment design and analysis: Key ideas from modern measurement theory. In M. M. C. Mok (Ed.), *Self-directed learning oriented assessments in the Asia-Pacific* (pp. 61–85). New York, NY: Springer.

Cressie, N., & Read, T. R. C. (1984). Multinomial goodness-of-fit tests. *Journal of the Royal Statistical Society: Series B*, *46*, 440–464.

de Boeck, P. (2008). Random item IRT models. *Psychometrika*, *73*, 533–559.

de la Torre, J., & Douglas, J. A. (2004). Higher-order latent trait models for cognitive diagnosis. *Psychometrika*, *69*, 333–353.

Delyon, B., Lavielle, M., & Moulines, E. (1999). Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, *27*, 94–128.

Edwards, M. C. (2009). An introduction to item response theory using the Need for Cognition scale. *Social and Personality Psychology Compass*, *3/4*, 507–529.

Edwards, M. C. (2010). A Markov chain Monte Carlo approach to confirmatory item factor analysis. *Psychometrika*, *75*, 474–497.

Edwards, M. C., & Cai, L. (2011, July). *A new procedure for detecting depar-*

tures from local independence in item response models. Paper presented at the annual meeting of American Psychological Association, Washington, D.C. Retrieved from `http://faculty.psy.ohio-state.edu/edwards/documents/APA8.2.11.pdf`

Falk, C. F., & Cai, L. (2016). A flexible full-information approach to the modeling of response styles. *Psychological Methods*, *21*, 328–347.

Fox, J.-P., & Glas, C. A. W. (2001). Bayesian estimation of a multilevel IRT model using Gibbs sampling. *Psychometrika*, *66*, 269-286.

Geyer, C. J. (1996). Estimation and optimization of functions. In W. R. Gilks, S. Richardson, & D. J. Spiegelhalter (Eds.), *Markov chain Monte Carlo in practice* (p. 241-258). New York, NY: Chapman and Hall.

Gibbons, R. D., Bock, R. D., Hedeker, D., Weiss, D. J., Segawa, E., Bhaumik, D. K., ... Grochocinski, V. J. (2007). Full-information item bifactor analysis of graded response data. *Applied Psychological Measurement*, *31*, 4–19.

Gibbons, R. D., & Hedeker, D. (1992). Full-information item bifactor analysis. *Psychometrika*, *57*, 423–436.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996a). Introducing Markov chain Monte Carlo. In W. R. Gilks, S. Richardson, & D. J. Spiegelhalter (Eds.), *Markov chain Monte Carlo in practice* (p. 1-19). New York, NY: Chapman and Hall.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (Eds.). (1996b). *Markov chain Monte Carlo in practice.* New York, NY: Chapman and Hall.

Gill, J. (Ed.). (1996). *Bayesian methods: A social and behavioral sciences approach.* New York, NY: Chapman and Hall.

Glas, C. A. W., Wainer, H., & Bradlow, E. T. (2000). Maximum marginal likelihood and expected a posteriori estimation in testlet-based adaptive testing. In W. J. van der Linden & C. A. W. Glas (Eds.), *Computerized adaptive testing: Theory and practice* (pp. 271–288). Boston, MA: Kluwer Academic Publishers.

Gu, M. G., & Kong, F. H. (1998). A stochastic approximation algorithm with Markov chain Monte-Carlo method for incomplete data estimation problems. *The Proceedings of the National Academy of Sciences*, *95*, 7270–7274.

Haberman, S. J. (1979). *The analysis of qualitative data.* New York: Academic Press.

Hartz, S. M. (2002). *A Bayesian framework for the unified model for assessing cognitive abilities: Blending theory with practicality.* (Unpublished doctoral dissertation). Department of Statistics, University of Illinois at Urbana-Champaign.

Hastings, W. K. (1970). Monte Carlo simulation methods using Markov chains and their applications. *Biometrika*, *57*, 97–109.

Jamshidian, M., & Jennrich, R. I. (2000). Standard errors for EM estimation. *Journal of the Royal Statistical Society: Series B*, *62*, 257–270.

Langer, M. M. (2008). *A reexamination of Lord's Wald test for differential item functioning using item response theory and modern error estimation* (Unpublished doctoral dissertation). Department of Psychology, University of North Carolina at Chapel Hill.

Lehman, A. F. (1988). A quality of life interview for the chronically mentally ill. *Evaluation and Program Planning*, *11*, 51–62.

Li, Z., & Cai, L. (2012, July). *Summed score likelihood based indices for testing latent variable distribution fit in item response theory.* Paper presented at the annual International Meeting of the Psychometric Society, Lincoln, NE. Retrieved from http://www.cse.ucla.edu/downloads/files/SD2-final-4.pdf

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society: Series B*, *44*, 226–233.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, *47*, 149–174.

Maydeu-Olivares, A., Cai, L., & Hernandez, A. (2011). Comparing the fit of IRT and factor analysis models. *Structural Equation Modeling*, *18*, 333–356.

Maydeu-Olivares, A., & Joe, H. (2005). Limited and full information estimation and testing in 2^n contingency tables: A unified framework. *Journal of the American Statistical Association*, *100*, 1009–1020.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of state space calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 1087–1092.

Metropolis, N., & Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, *44*, 335-341.

Mislevy, R. (1984). Estimating latent distributions. *Psychometrika*, *49*, 359–381.

Monroe, S., & Cai, L. (2015). Evaluating structural equation models for categorical outcomes: A new test statistic and a practical challenge of interpretation. *Multivariate Behavioral Research*, *50*, 569-583.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, *16*, 159–176.

Orlando, M., & Thissen, D. (2000). Likelihood-based item-fit indices for dichotomous item response theory models. *Applied Psychological Measurement*, *24*, 50–64.

Patz, R. J., & Junker, B. W. (1999a). A straightforward approach to Markov chain Monte Carlo methods for item response models. *Journal of Educational and Behavioral Statistics*, *24*, 146-178.

Preston, K., Reise, S., Cai, L., & Hays, R. (2011). Using the nominal response model to evaluate response category discrimination in the PROMIS emotional distress item pools. *Educational and Psychological Measurement*, *71*, 523–550.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407.

Roberts, G. O., & Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, *16*, 351–367.

Rupp, A. A., Templin, J., & Henson, R. A. (2010). *Diagnostic measurement: Theory, methods, and applications.* New York: The Guilford Press.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometric monograph No. 17.*

Takane, Y., & de Leeuw, J. (1987). On the relationship between item response theory and factor analysis of discretized variables. *Psychometrika*, *52*, 393–408.

Thissen, D., & Cai, L. (2016). Nominal categories models. In W. J. van der Linden (Ed.), *Handbook of item response theory: Vol. 1.* (pp. 51–74). Boca Raton, FL: Chapman & Hall/CRC.

Thissen, D., Cai, L., & Bock, R. D. (2010). The nominal categories item response model. In M. Nering & R. Ostini (Eds.), *Handbook of polytomous item response theory models: Developments and applications* (pp. 43–75). New York, NY: Taylor & Francis.

Thissen, D., & Orlando, M. (2001). Item response theory for items scored in two categories. In D. Thissen & H. Wainer (Eds.), *Test scoring* (pp. 73–140). New York, NY: Routledge.

Thissen, D., & Steinberg, L. (1986). A taxonomy of item response models. *Psychometrika*, *51*, 567–577.

Thissen, D., & Steinberg, L. (1988). Data analysis using item response theory. *Psychological Bulletin*, *104*, 385–395.

Tian, W., Cai, L., Thissen, D., & Xin, T. (2013). Numerical differentiation methods for computing error covariance matrices in item response theory modeling: An evaluation and a new proposal. *Educational and Psychological Measurement*, *73*, 412–439.

Tittering, D. M. (1984). Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society: Series B*, *46*, 257–267.

Van den Noortgate, W., De Boeck, P., & Meulders, M. (2003). Cross-classfication multilevel logistic models in psychometrics. *Journal of Educational and Behavioral Statistics*, *28*, 369–386.

Wirth, R. J., & Edwards, M. C. (2007). Item factor analysis: Current approaches and future directions. *Psychological Methods*, *12*, 58–79.

Woods, C. (2007). Empirical histograms in item response theory with ordinal data. *Educational and Psychological Measurement*, *67*, 73–87.

Woods, C., Cai, L., & Wang, M. (2013). The Langer-improved Wald test for DIF testing with multiple groups: Evaluation and comparison to two-group IRT. *Educational and Psychological Measurement*, *73*, 532–547.

Yen, W. M. (1981). Using simulation results to choose a latent trait model. *Applied Psychological Measurement*, *5*, 245–262.

Index