IRT Command Language

Version 0.020301
March 30, 2002

Bradley A. Hanson

Table of Contents

1 Using ICL

IRT Command Language (ICL) is a computer program that can estimate parameters for the 1–, 2–, and 3–parameter logistic item response models for dichotomous items (Lord, 1980), and the partial credit and generalized partial credit models for polytomous items (Muraki, 1992). The 3–parameter logistic model (3PL) gives the probability of a correct response to item j as a function of the latent variable measured by the test the item is contained in (θ) as

$$\Pr(X_j = 1 \mid \theta) = c_j + \frac{1 - c_j}{1 + e^{-Da_j(\theta - b_j)}} ,$$

where X_j is a random variable representing a response to item j (where 1 indicates a correct response), a_j, b_j, and c_j are item parameters, and D is a scaling constant. In the two–parameter logistic model (2PL) the c–parameter is fixed at a specific value rather than estimated. In the one–parameter logistic model (1PL) the a– and c–parameters are fixed at specific values rather than estimated. For an item j with m_j possible responses $(0, 1, \ldots, m_j - 1)$ the generalized partial credit model (GPCM) gives the probability of response r as a function of latent variable θ as

$$\Pr(X_j = r \mid \theta) = \frac{e^{z_{jr}}}{1 + \sum_{k=1}^{m_j - 1} e^{z_{jk}}} ,$$

where

$$z_{ji} = \sum_{k=1}^{i} a_j(\theta - b_{jk}) ,$$

X_j is a random variable representing a response to item j, and a_j and $b_{jk}, k = 1, 2, \ldots, m_j - 1$, are item parameters. In the partial credit model (PCM) the a–parameter is fixed at a specific value rather than estimated. ICL can compute maximum likelihood or Bayes modal estimates of item parameters by finding the maximum of the marginal likelihood or posterior distribution, where the marginalization is over a discrete distribution of latent examinee proficiency in the population of examinees from which the data were sampled. The EM algorithm (Dempster, Laird, and Rubin, 1977; McLachlan and Krishnan, 1997) is used to compute the maximum likelihood or Bayes modal estimates. ICL handles single and multiple group estimation. In single group estimation it is assumed that all examinees are sampled from the same population, although different examinees may take different subsets of items. In multiple group estimation groups of examinees taking different, overlapping, subsets of items may be sampled from different populations. A description of the general algorithm used by the program in the single group case is presented in Woodruff and Hanson (1997) and Hanson (1998).

The next section discusses how to install and run ICL on three different operating systems: Windows, Macintosh, and Linux. Section 1.2 describes some basic ICL commands. These sections present basic information that is needed by all ICL users. The last two sections discuss ICL commands that allow more control over the program. These sections only need to be read if features are needed beyond those offered by the commands discussed in Section 1.2 (Basic ICL Commands).

The next chapter contains several examples. The first two examples use only the basic ICL commands discussed in Section 1.2. The remaining examples illustrate more advanced features of ICL.

Information about ICL, the latest version of ICL, this manual, and the ICL source code can be obtained from the ICL home page. ICL is part of the Social Science Measurement project hosted on the SourceForge web site. The development of ICL is managed through the SourceForge web site, which allows anyone interested to participate in the development of ICL.

ICL supercedes the Estimation Program for Dichotomous Item Response Models (EPDIRM) by adding the capability of estimating item parameters for polytomous IRT models. Most EPDIRM control files will require some minor changes to be used with ICL. Most importantly, the names of the three commonly used EPDIRM commands `epdirm_start`, `epdirm_end`, and `starting_values` have been changed to `allocate_items_dist` `release_items_dist`, and `starting_values_dichotomous`, respectively, in ICL. More details about EPDIRM commands that have changed in ICL can be found on this web page: http://www.b-a-h.com/software/irt/icl/epdirm_differences.html.

1.1 Installing and Running ICL

There are pre–compiled versions of ICL available for Windows, Macintosh, and Linux. The Linux source code can be used to compile ICL for other UNIX operating systems. In each case ICL is packaged as a single executable file. Installing ICL just involves copying this executable file to an appropriate location on the hard disk.

ICL runs by executing a sequence of commands. The commands are executed one by one in the order given. ICL can be run either interactively or in batch mode. When run interactively individual commands are typed for ICL to execute. In batch mode ICL processes a file containing a sequence of commands. The commands in the file are executed in the order in which they appear in the file. The following sections describe how to run the Windows, Linux, and Macintosh versions of ICL.

1.1.1 Running the Windows and Linux Versions of ICL

The Windows version of ICL requires a 32–bit version of Windows (95, 98, NT, 2000, XP). Even though the Windows version of ICL is a command–line program it will not run under DOS. There may be problems running ICL on very early versions of Windows 95. ICL requires the DLL file '`MSVCRT.DLL`' which was not included with the original Windows 95 release. If you have an early version of Windows 95 without '`MSVCRT.DLL`' you will need to upgrade your version of Windows or obtain a copy of the required DLL.

The Linux version of ICL should run under any Linux distribution. It is statically linked and does not depend on any Linux shared libraries.

ICL is command line program that runs in a command console. Under Windows it runs in a command prompt window, and under Linux it runs in a console window. ICL can be started from a command console in interactive mode by typing the name of the executable file and hitting the enter or return key. An interactive session is started in which ICL commands can be entered. This assumes the ICL executable file has been put in a directory that is in the list of directories searched for executable applications (i.e., the directory is contained in the list of directories given by the PATH environment variable). ICL can also be started in interactive mode by double clicking on the ICL icon. This will open a console window in which ICL commands can be entered. To stop an interactive ICL session use the

`exit` command, or close the console window in which ICL is running. The `source` command can be used to execute a sequence of commands from a file when ICL is run interactively. To use the `source` command type '`source`' followed by the name of the file containing the ICL commands to execute.

To run ICL in batch mode type the name of the ICL executable file followed by the name of a file containing ICL commands in a console window. The commands in the file will be executed and ICL will then quit. For example, the following command can be used to run the example program described in Section 2.1 (the name of the executable file is `icl`):

```
icl mondaty.tcl
```

1.1.2 Running the Macintosh Version of ICL

The Macintosh version of ICL should work on version 8.0 or higher of the classic Macintosh operating system (Mac OS). ICL will also probably work with Mac OS version 7, but has only been tested on Mac OS 8 and Mac OS 9. The Macintosh version of ICL will also run in the classic environment of Mac OS X.

ICL requires a Macintosh with a PowerPC processor (including the G3 or G4 processor). All recent Macintosh models have a PowerPC processor. ICL will not work on older Macintoshes which use the 68000 family of processors. The default memory allocation for ICL is 6 megabytes. For large problems the memory allocation will need to be increased.

ICL is a command line program that is run in a command console. Unlike Windows and Linux, the Mac OS does not have any built–in command console. The Macintosh version of ICL currently only supports interactive mode in a command window created by the program.

To start the Macintosh version of ICL double click on the ICL program icon. When the program begins a command window is displayed. ICL commands can be typed into this window and executed. The `source` command can be used to execute ICL commands contained in a file. To use the `source` command type '`source`' followed by the name of the file containing the ICL commands to execute.

File names used with commands such as `source` are relative to the current folder. When the program begins the current folder is the folder containing the ICL application. The full path of the current folder can be printed using the `pwd` command. A list of the files and folders contained in the current folder can be obtained using the `ls` command. The '`ls -l`' command lists additional information about the files and folders in the current folder.

The `cd` command can be used to change the current folder. It may be easier to change the current folder if files need to be accessed in another folder which would require the use of a full path name. On a Macintosh colons separate elements of a path name. For example, the command '`cd {Macintosh HD:work}`' will change the current folder to a folder named '`work`' on a disk named '`Macintosh HD`'. The brackets are needed around the path to the folder because there is a space in the path.

A strategy for using the Macintosh version of ICL that minimizes having to type long path names is to install the files to be used with ICL within folders that are contained in the folder containing the ICL application. For instance, if the '`examples`' folder that comes with the ICL distribution was placed in the same folder containing the ICL application then the following two commands could be used to run the given in Section 2.1:

```
cd :examples
```

```
source mondaty.tcl
```

The `cd` command in the above example illustrates that relative paths on the Macintosh begin with a colon. If it is necessary to deal with full path names a utility like CopyPaths is recommended. CopyPaths allows full paths to be copied from the Finder and pasted into an ICL command.

Text in the console window can be copied and saved using commands in the File and Edit menus. To quit the program selected Quit from the File menu, or use the `exit` command.

1.2 Basic ICL Commands

When ICL executes a sequence of commands in the order given. Commands can be entered interactively one by one, or read from a file. This section discusses command syntax and describes several basic ICL commands.

An ICL command has the following syntax

command arg1 arg2 arg3 ...

where *command* is the command name which is followed by the arguments to the command. The command name and the command arguments are separated from one another by white space (spaces or tabs).

Each ICL command appears on a separate line. A command can appear on two or more lines if each line before the last one containing the command ends with a backslash. Any line in which the first non–blank character is a pound sign (#) is treated as a comment. An example ICL command file is given below:

```
# Estimate item parameters using data set mondatx
# from Kolen and Brennan (1995)

# Write output to log file mondatx.log
output -log_file mondatx.log

# 36 dichotomous items to be modeled
allocate_items_dist 36

# Read examinee item responses from file mondatx.
# Each record contains the responses to
# 36 items for an examinee in columns 1-36.
read_examinees mondatx 36i1

# Compute starting values for item parameter estimates
starting_values_dichotomous

# Perform EM iterations for computing item parameter estimates.
# Maximum of 50 EM iterations.
EM_steps -max_iter 50

# Print item parameter estimates and discrete latent
# variable distribution.
print -item_param -latent_dist
```

```
# Release memory allocated to hold items and the
# latent variable distribution
release_items_dist
```

There are 7 ICL commands in this file. All lines beginning with a pound sign are comments. Blank lines are ignored. Detailed descriptions of the commands used in this command file are given below. This example is discussed in more detail in Section 2.1.

This section presents detailed descriptions of eight basic ICL commands — `allocate_items_dist`, `release_items_dist`, `read_examinees`, `starting_values_dichotomous`, `EM_steps`, `print`, `options`, and `output`. These eight commands are all that are needed to compute item parameter estimates in typical cases. Recall that ICL commands are executed in the order given. A restriction on the order in which the commands can be given is that the `read_examinees`, `starting_values_dichotomous`, `EM_steps`, and `print` commands can only occur after an `allocate_items_dist` command. In addition, the `output` and `options` commands will typically occur as the first commands to be executed, although they are not restricted to appearing before other commands.

Some arguments to ICL commands begin with a hyphen. These arguments are optional and can appear in any order. Arguments that do not begin with a hyphen must appear in a specific position within the command string (positional arguments). In some cases an option will be composed of two command arguments — one argument beginning with a hyphen and a second argument following the first which specifies a variable associated with that option. For example, the following command contains one positional argument and two command options.

```
# 60 dichotomous items will be modeled.
# 50 points in discrete latent variable distribution.
# Examinees are sampled from one population.
allocate_items_dist 60 -num_latent_dist_points 50 -num_groups 1
```

The first argument of the `allocate_items_dist` command is the number of items to be modeled. This argument is required and must be the first argument of the command. The second and third arguments (`-num_latent_dist_points` and `50`) are associated with one command option. These two arguments together specify that the number of points used for the discrete latent variable distribution is 50. The fourth and fifth arguments (`-num_groups` and `1`) are also associated with one command option. These two argument specify that the examinees were sampled from one group (single group estimation is used). The effect of the command would have been the same if the last two arguments were left off because the default number of groups is one if the '`-num_groups`' option is not specified.

The notation used in the command documentation in this section and following sections is as follows. The commands are presented in **bold** type, a command argument that is literal text (the text is exactly the same in every instance of the command) is presented in `fixed` type, and a command argument that is variable text (this text can vary from one instance of the command to another) is presented in *italic* type. Optional arguments are enclosed within a pair of question marks. In cases in which two arguments compose a command option and must go together both arguments are enclosed in a single set of question marks. The '`Tcl`' to the right of each command is an indication of the language the command was written in and can be ignored — it has no significance regarding how the command is used. The meaning of '`Tcl`' is explained in the next section which discusses advanced

ICL commands. The remainder of this section presents descriptions of the eight basic ICL commands.

allocate_items_dist *nitems ?*`-models` *list? ?*`-models_str` *string?* Tcl
?`-num_latent_dist_points` *npoints? ?*`-num_groups` *ngroups?*
?`-latent_dist_range` *list? ?*`-unique_points`*?*

The `allocate_items_dist` command is used to specify the number of items to be modeled, whether a dichotomous or polytomous model is to be used for each item, the number of examinee groups, the number of points in the discrete latent variable distribution, and the minimum and maximum points of the discrete latent variable distribution. The `allocate_items_dist` command allocates space to hold information for each item and for the latent variable distribution. An `allocate_items_dist` command must be given before any of the commands except `options` and `output` can be used. The value of the first argument is the total number of items to be modeled. In subsequent commands items are identified by item number. If the number of items specified is n then item numbers 1 through n are used to refer to these items in subsequent commands.

The `-models` argument is associated with a list containing an integer for each item. The integer 1 corresponding to an item means the item is modeled using a dichotomous model. If the integer corresponding to an item is greater than 1 that means the item is a polytomous item with that number of response categories modeled using a polytomous model. By default the three–parameter logistic model is used for dichotomous items and the generalized partial credit model is used for polytomous items. The default models to use for dichotomous and polytomous items can be set with the `--default_model_dichotomous` and the `-default_model_polytomous` arguments of the `options` command. If the `-models` option is not present the models for all items are assumed to be dichotomous. The `-models_str` argument identifies whether to use a dichotomous or polytomous model for each item in the same way as the `-models` option, except that the integers that indicate the model to use for each item are contained in a string. Each character in the string gives a model to use for one item. This means the `-model_str` argument can only be used when the number of response categories for all polytomous items is less than 10. If the number of response categories is greater than 9 for any item the `-models` argument must be used, rather than the `-models_str` argument. If the `-models` and a `-models_str` arguments are both present the `-models_str` argument is used to determine the model for each item. The following example illustrates the use of the `-models_str` argument.

```
# Ten total items are modeled. The
# first six items are modeled using the three-parameter
# logistic model, and the remaining four items are
# modeled using the generalized partial credit model.
# The seventh item has two response categories, the
# eighth and ninth items have 3 response categories,
# and the tenth item has four response categories.
allocate_items_dist 10 -models_str 1111112334
```

Examples of using the `-models` argument are given in Sections 2.7 and 2.8.

The `-num_latent_dist_points` option gives the number of categories used for the discrete latent variable distribution. The default value if the `-num_latent_dist_`

`points` option is not present is 40. The `-num_groups` option gives the number of groups of examinees in the data that are sampled from different populations. The default value if the `-num_groups` option is not present is 1. If the number of groups is greater than 1 multiple group estimation used, and a group identifier must be indicated for each examinee. The list given for the `-latent_dist_range` option should contain two numbers giving the minimum and maximum points of the discrete latent variable distribution, respectively. For example, `-latent_dist_range {-5 5}` indicates the minimum and maximum points of the latent distribution are -5 and 5. The default minimum and maximum points of the latent distribution if the `-latent_dist_range` option is not used are -4 and 4, respectively.

If the `-unique_points` option is present then unique points will be used for each latent variable distribution in each examinee group. If the `-unique_points` option is not present one set of points are used for the latent variable distributions in all examinee groups, although different latent distribution probabilities are used for each group. The `-unique_points` option is only relevant if more than one examinee group is specified with the `-num_groups` option. This option must be used with the `allocate_items_dist` command if the `-estim_dist_mean_sd` option or `-estim_dist_mean` option is used with the `EM_steps` command. Section 2.5 gives an example in which the `-unique_points` option is used.

The `allocate_items_dist` command assigns initial values to the item parameters for all items. The initial parameter estimates assigned to all a–, b–, and c–parameters are 1.0, 0.0, and 0.2, respectively. The same values are used for the a– and b–parameters of dichotomous and polytomous models. The values of the item parameters for individual items can be changed using the `item_set_param` and `item_set_params` commands. The `starting_values_dichotomous` command can be used to compute parameter starting values for dichotomous models.

The `allocate_items_dist` command initializes the prior distributions used for all item parameters to default values. The prior distributions used as defaults can be set using the `options` command. If the `options` command is not given the default item parameter prior distributions for the a–, b– and c–parameters are four-parameter beta with parameters (1.75, 3.0, 0.0, 3.0), (1.01, 1.01, -6.0, 6.0) and (3.5, 4.0, 0.0, 0.5), respectively, where the four numbers in the parameter list are the two shape parameters followed by the lower and upper limits of the distribution. The same default priors are used for the a–parameters and b–parameters of both dichotomous and polytomous models. The priors used for item parameters for individual items can be changed with the `item_set_prior` and `items_set_prior` commands.

The values of the points and weights (probabilities) used for the discrete latent variable distribution are also initialized by the `allocate_items_dist` command. There is one set of points (unless the `-unique_points` option is used) and a set of weights for each examinee group. The default points are equally spaced between minimum and maximum points indicated by the `-latent_dist_range` option, inclusive. The default weights are chosen so the resulting discrete distribution approximates a standard normal distribution. When there is more than one examinee group the initial weights are the same across groups. The points can be changed with the `dist_set_point` and `dist_set_points` commands. The weights can be changed by the `dist_set_prob` and `dist_set_probs` commands.

The `allocate_items_dist` command writes information to the log file specified using the `output` command including the version number of the ICL program, the number of items, the number of categories in the discrete latent variable distribution, the number of groups, and the default prior distributions used for item parameters. The command file is also listed, if a command file is being used. This information is not written to the log file if the '`output -no_print`' command is executed before the `allocate_items_dist` command.

EM_steps *?*`-max_iter` *iter? ?*`-crit` *d? ?*`-estim_dist`*? ?*`-scale_points`*? ?*`-no_print_iter`*? ?*`-no_mstep_iter_error`*? ?*`-estim_dist_mean_sd`*? ?*`-estim_dist_mean`*?* Tcl

The `EM_steps` command performs EM iterations to estimate item parameters and, optionally, the marginal latent variable distributions. An `allocate_items_dist` command must be given before this command can be used. The `-max_iter` options specifies the maximum number of EM iterations. If the maximum number of iterations is not specified a default value of 100 is used. The convergence criterion used to determine when to stop EM iterations is the maximum relative change in an item parameter estimate from the previous to the current iteration over all parameters across all items. The option `-crit` specifies the value this convergence criterion must be less than to stop. The default criterion if the `-crit` option is not specified is 0.001. If the convergence criterion is not met after the specified number of iterations a warning message is printed in the log file.

If the `-estim_dist` option is given an M–step to estimate the probabilities of the marginal latent variable distribution for the base examinee group is performed in each EM iteration, otherwise this distribution is fixed across EM iterations. The base examinee group is group 1 unless otherwise specified using the `-base_group` option on the `options` command. If the number of examinee groups is greater than one the latent variable distributions for groups other than group 1 are estimated regardless of whether the `-estim_dist` option is used.

If the `-scale_points` option is used the points of latent variable distribution are linearly transformed after every M–step so that the mean and s.d. in base examinee group are 0 and 1. This scale transformation is also applied to the parameter estimates for all items to put them on the same scale. When this argument is not present the points of the latent variable distribution are not changed, even when the latent variable probabilities for all groups are estimated. This argument only has an effect when the latent variable distribution of the base group is being estimated as requested with the `-estim_dist` option.

If the `-no_mstep_iter_error` option is used then the maximum number of iterations being exceeded in the M-step optimization is not considered an error that stops the calculation. Instead, a warning message is printed in the log file and the calculation continues. If this option is not present then if the maximum number of M–step iterations is exceeded for an item an error message is printed and the calculation is terminated. The maximum number of M–step iterations can be set for all items with the `-max_iter_optimize` option of the `options` command, or can be set for individual items with the `mstep_max_iter` command.

If the `-estim_dist_mean_sd` option is used the mean and standard deviation, rather than the probabilities, of the latent variable distributions are estimated. If this option is used the points of the latent variable distribution for all examinee groups except the base group are modified to be consistent with an estimated mean and standard deviation (the mean and standard deviation are fixed for the base group). The `-estim_dist_mean` option is similar to the `-estim_dist_mean_sd` option except that only the means of the latent variable distributions are estimated. These options require that different latent distribution points be used for different examinee groups as specified using the `-unique_points` option of the `allocate_items_dist` command. Section 2.5 gives an example in which the `-estim_dist_mean_sd` option is used.

After each iteration several numbers are written to the log file unless output has been suppressed with the `output -no_print` command: 1) the iteration number, 2) the maximum difference between a parameter estimate from the last and second to last iteration, 3) the maximum difference between a probability of the latent variable distribution from the last and second to last iteration if the latent variable distribution is being estimated (the number of examinee groups is greater than 1 or the `-estim_dist` option is used, and the `-estim_dist_mean_sd` or `-estim_dist_mean` options are not used), 4) the maximum difference between the latent variable distribution mean from the last and second to last iteration if the `-estim_dist_mean_sd` or `-estim_dist_mean` option is used, 5) the maximum differences in the s.d. between the last and second to last iteration if the `-estim_dist_mean_sd` option is used, and 6) the value of the marginal posterior at the value of the parameter estimates (this is the quantity being maximized by the EM algorithm). This information is also written to the screen unless the `-no_print_iter` option is used. A list containing these values at the last iteration is returned by the `EM_steps` command.

options *?*`-D` *d? ?*`-missing_resp` *resp? ?*`-base_group` *group?* *?*`-default_model_dichotomous` *model? ?*`-default_model_polytomous` *model? ?*`-max_iter_optimize` *max? ?*`-default_prior_a` *prior? ?*`-default_prior_b` *prior? ?*`-default_prior_c` *prior? ?*`-default_dist_range` *list?* Tcl

The `options` command sets global program options. The options set with this command influence the behavior of other commands such as the `allocate_items_dist` command. Consequently, the `options` command is typically one of the first commands executed.

The `-missing_resp` option is used to specify the character that indicates an examinee has not responded to an item. Any character except '`0`' and '`1`', which indicate an incorrect and correct response, can be used. The default character indicating a missing response is a period.

The `-D` option specifies the value of a scaling constant (D) in the logistic function for the 3PL, 2PL, and 1PL models. The value 1.7 makes the logistic ogive close to a normal ogive. If the `option -D` command is not given the value of 1.7 is used as the default. This option does not have an effect on the GPCM and PCM models.

The `-default_model_dichotomous` option specifies the default model used for an item modeled using a dichotomous model. The value of this option can be one of three values '`3PL`', '`2PL`', or '`1PL`' corresponding to the three-parameter, two-parameter, and

one-parameter logistic models. If the '`option -default_model_dichotomous`' command is not given the three-parameter logistic model is used as the default. The `-default_model_polytomous` option specifies the default model used for an item modeled using a polytomous model. The value of this option can be either '`GPCM`' or '`PCM`' corresponding to the generalized partial credit model and the partial credit model. If the '`option -default_model_polytomous`' command is not given the generalized partial credit model is used as the default. The model for individual items can be set with the `item_set_model` command.

The `-base_group` option specifies the examinee group that will be used as the base group. The latent variable distribution for the base group is fixed rather than estimated unless the `-estim_group` option is used with the `EM_steps` command. If the `-estim_group` option is used with the `EM_steps` command the mean and standard deviation of the base group latent variable distribution are fixed at zero and one.

The `-max_iter_optimize` option sets the maximum number of iterations used for each item in the optimization procedure used for the M–step and starting value calculations. The default value used if this option is not present is 150. To set the maximum number of iterations used in the optimization procedure for each item individually use the `mstep_max_iter` command.

The `-default_dist_range` option sets the minimum and maximum points to be used for the discrete latent variable distribution. If this option is not present the default values used for the minimum and maximum points of the discrete latent variable distribution are -4 and 4, respectively. The minimum and maximum points are specified as two numbers surrounded by brackets (a list of two numbers). For example, the following command will specify the minimum and maximum points at -5 and 5

```
options -default_dist_range {-5 5}
```

The third argument of this command shows that to specify a list of strings as a single command argument the elements of the list should be surrounded by brackets.

The `-default_prior_a`, `-default_prior_b`, and `-default_prior_c` options set the default prior distributions used for the a–, b–, and c–parameters for all items. The same default priors for a– and b–parameters are used for dichotomous and polytomous models. For each of these options the prior is specified by a list, where the first element of the list is the type of prior distribution and the remaining elements of the list are the parameters of the prior distribution. There are three distribution types possible: '`normal`', '`lognormal`' '`beta`', and '`none`', corresponding to the normal distribution, the lognormal distribution, the four-parameter beta distribution, and no prior distribution, respectively. For the normal and lognormal distributions two parameters need to be specified: a mean and standard deviation, in that order. For the four-parameter beta distribution the parameters that need to be specified are the two shape parameters, the lower limit, and the upper limit, in that order. For example,

```
# Specify prior distributions used by default in BILOG

# Lognormal prior for a-parameters with mean 0 and standard
# deviation 0.5 in the underlying normal distribution
options -default_prior_a {lognormal 0.0 0.5}
```

```
# No prior for b-parameters
options -default_prior_b none

# Two-parameter beta prior is used by BILOG for the
# c-parameters when the number of response options is 4.
options -default_prior_c {beta 6.0 16.0 0.0 1.0}
```

As the above example indicates, more than one `options` command can be used. The following example shows the default prior distributions that would be used if no `options` statements were used. These priors are also presented in the discussion of the `allocate_items_dist` command.

```
# Default prior for a used if option command is not used.
# The 10th, 25th, 50th, 75th, and 90th percentiles
# of this distribution are .34, .62, 1.05, 1.53, and 1.96,
# respectively. The mean is 1.11 and the mode is .82.
options -default_prior_a {beta 1.75 3.0 0.0 3.0}

# Default prior for b used if option command is not used
# This is an almost uniform prior on the interval -6
# to 6. It is better to not use an exact uniform
# distribution so that the slopes at
# the distribution limits are not infinite
options -default_prior_b {beta 1.01 1.01 -6.0 6.0}

# Default prior for c used if option command is not used.
# The 10th, 25th, 50th, 75th, and 90th percentiles
# of this distribution are .12, .17, .23, .30, and .35,
# respectively. The mean is .23 and the mode is .23.
options -default_prior_c {beta 3.5 4.0 0.0 0.5}
```

Separate prior distributions for parameters of individual items can be set with the `item_set_prior` and `items_set_prior` commands.

output *?*`-log_file` *file_name? ?*`-no_print`*?* Tcl

The output command is used to specify options for printed output. The `-log_file` argument is used to specify the name of a log file to which the printed output of commands is written. If the `-no_print` option is specified then written output of any command is suppressed.

print *?*`-item_param`*? ?*`-latent_dist`*? ?*`-latent_dist_moments`*?* Tcl
?`-no_heading`*? ?*`-items` *itemno? ?*`-format` *string? ?*`-item_model`*?*

The `print` command prints item parameter estimates, latent variable distributions, and moments to the log file. If the `-item_param` argument is present then the current item parameter estimates are printed to the log file. If the `-latent_dist` argument is present then the current discrete latent distributions for all examinee groups are printed to the log file. If the `-latent_dist_moments` argument is present then the mean and standard deviation of the current discrete latent distributions for all examinee groups are printed to the log file. If more than one of the `-item_param`,

`-latent_dist`, or `-latent_dist_moments` options is given the order in which the output is printed is: 1) the item parameters, 2) latent variable distributions, and 3) moments of the latent variable distributions. To print the options in a different order multiple `print` statements could be used:

```
# print latent variable distributions followed by item parameter
# estimates
print -latent_dist
print -item_param
```

If the `-no_heading` option is used no descriptive heading is printed before the item parameters and latent variable distribution. Item parameter estimates will only be printed for items corresponding to item numbers in the list following `-items`. If the `-items` option is not used item parameter estimates are printed for all items. If the `-item_param` option is not used then the `-items` option has no effect.

The `-format` option is used to specify the output format used for the item parameter estimates, weights of the latent variable distribution, and moments. The format is used for all item parameter estimates, for the weights of the latent variable distribution (fixed point format with six decimal places is used for the points of the latent variable distribution), and the mean and standard deviation. The format specified is used for all elements to be printed. To use separate formats for different elements to be printed use separate `print` commands as in the example below. The string giving the format is a C sprintf–like format — `%[width][.precision]char`, where `char` = '`f`' (fixed point), '`e`' (scientific notation), or '`g`' (fixed point or scientific notation, which ever takes less space). A more detailed description of the format specification is given in Appendix A. If a format is not specified the default format used for printing the item parameter estimates and distribution moments is `%.6f` (fixed point with six places after the decimal point), and the default format used for the distribution weights is `%.6e` (scientific notation with six places after the decimal point). If the `-model_item` option is present the model associated with each item (either 3PL, 2PL, 1PL, GPCM, or PCM) is printed between the item number and first parameter when the `-item_param` option is specified. An example of the `print` command is given below.

```
# Print the item parameter estimates and latent variable
# distribution using 8 digits after the decimal
# point rather than the default 6.
print -item_param -format %.8f
print -latent_dist -format %.8e
```

read_examinees *file resp_format ?group_format?* Tcl

The `read_examinees` command reads examinee responses to the items, and optionally an examinee group, from a file. The first argument is the name of the file to read examinee responses from. It is assumed that each line of the file contains responses to all items for one examinee. The second argument consists of a format list which indicates which columns in each record contain item responses. The syntax of a format list is explained below. The optional third argument consists of a format list which indicates which columns in each record contain the group number the examinee belongs to. The third argument is only needed if the number of groups indicated in the `allocate_items_dist` command is greater than 1. The `read_examinees` command returns the number of examinees for whom item responses were read.

The number of item responses read for an examinee must be equal to the number of items given on the `allocate_items_dist` command (the `read_examinees_missing` command can be used when examinee records do not contain the responses to all items). A '1' or '0' indicates a correct or incorrect response to the item, respectively. If the examinee did not respond to the item, for example if they did not receive the item, a '.' (period) should be given as the item response. The character that indicates an examinee did not respond to the item can be changed using the `options` command. The first item read for each examinee corresponds to item 1, the second item read corresponds to item 2, etc. The group numbers must be 1, 2, . . . , up to the number of groups.

The format list used for the second and third arguments contains strings in one of two forms: 1) `@`#, where # is an integer, which means move to column # of the input record, 2) ri#, where r and # are integers, which means read r item responses from r times # columns at the current location in the input record, where each response contains # characters. The r is optional, and if not given is assumed to be 1. If there is more than one string in the format list the list must be delimited by brackets ('`{`' and '`}`'). If there is only one string in the format the brackets are not necessary. Examples of the `read_examinee` command illustrating the use of format lists in the second and third argument are:

```
# Read item responses for 15 items from columns 3-7
# and columns 20-29 of each input record in file test.in
read_examinees test.in {@3 5i1 @20 10i1}

# Read item responses for 100 items from columns 2-101.
# Read group number from column 1
read_examinees test.in {@2 100i1} i1
```

Other examples illustrating the use of the `read_examinees` command are given in Chapter 2.

release_items_dist Tcl

The `release_items_dist` command indicates the completion of sequence of commands begun by an `allocate_items_dist` command. Memory and resources allocated by the `allocate_items_dist` command and subsequent commands are released, and the log file is closed. An `allocate_items_dist` command should be paired with a corresponding `release_items_dist` command.

starting_values_dichotomous *?*`-items` *list? ?*`-use_all`*?* *?*`-ignore_error`*?* Tcl

The `starting_values_dichotomous` computes starting values for the item parameter estimates of items model using a dichotomous model. An `allocate_items_dist` command must be given before this command can be used. The `-items` argument is followed by a list of item numbers for which starting values are to be computed. All these items must be modeled using a dichotomous model. If the `-items` option is not present starting values are computed for all items modeled using a dichotomous model.

If the option `-use_all` is present then all examinees are included in the computation of initial proficiencies used for computing starting values, even examinees who get all

items correct or all items incorrect. If the `-use_all` argument is not present examinees who get all items correct or incorrect are not included in computing the initial proficiencies used in computing the starting values. If the argument `-ignore_error` is present then the program will continue even if there was an error in computing the starting values, although a warning message will be printed. If the argument `-ignore_error` is not present the program will terminate if an error occurs in computing the starting values.

More information about the starting values is contained in the discussion of the `item_3PL_starting_values` command in the next section. The `starting_values_dichotomous` command calls the `item_3PL_starting_values` command.

Currently there is no command to produce starting values for items modeled using a polytomous model. Using the default values assigned to the a–parameters (1) and b–parameters (0) as starting values for polytomously modeled items appears to result in good parameter estimates, at the possible expense of more EM iterations. Starting values for any item can be manually assigned using the `item_set_param` and `item_set_params` commands.

1.3 ICL Commands

The commands presented in the previous section allow basic control of the program for estimating 3PL model item parameters, and latent variable distributions, including multiple group estimation. For many users the commands presented in the previous section are all that would be needed to accomplish what they want to do with ICL. This section discusses additional features of the ICL command language that allow more control over the operation of the program.

To process commands ICL uses an embedded Tool Command Language (Tcl) interpreter. Tcl is pronounced "tickle". Tcl is a scripting language designed to be embedded within applications in order to act as the command language for the application. There are a basic set of ICL commands written in C++ that are added to the basic Tcl interpreter. In addition, all commands that are part of the Tcl interpreter are available for use as ICL commands. Many of the ICL commands, including all the commands described in the previous section, are actually written in the Tcl language using the more basic commands written in C++. The Tcl source code for the ICL commands written in Tcl, including those discussed in the previous section, is contained in the file '`icl.tcl`' provided with the ICL distribution.

There are a few restrictions on the order in which the commands can be given. Most commands require the `new_items_dist` command be executed before they can be used (the `allocate_items_dist` command discussed in the previous section calls the `new_items_dist` command). Exceptions are commands that set default values used by other commands such as `set_default_D`, `set_default_model_dichotomous`, `set_default_model_polytomous`, `set_default_prior`, and `set_missing_resp`. Examinee item response data must be assigned using the `add_examinee` command before commands which compute estimates can be used (the `allocate_items_dist` or `new_items_dist` command must be executed before an `add_examinee` command can be executed).

The format of the command descriptions is the same as that described in the previous section. On the first line of the command description in the right margin is either '`Tcl`' or '`C++`', which indicates whether the command was written in Tcl or C++. A source

code command written in Tcl can be found in the file '`icl.tcl`' that comes with the ICL distribution. The source code for the commands written in C++ can be found in the source file '`swig_etirm.cpp`' which comes with the ETIRM distribution and the '`swig_icl.cpp`' that come with the ICL source code distribution.

add_examinee *responses ?group? ?count?* C++

The `add_examinee` command adds information for an examinee that is used for parameter estimation. The first argument is a list of integers giving the item responses for the examinee. The number of integers in this list must be the same as the total number of items as defined in a `allocate_items_dist` command. A zero represents an incorrect response and a one represents a correct response. A negative number indicates the examinee did not respond to the item. The second argument gives the group number to which the examinee belongs. The default value of the second argument if it is not present is 1. The third argument gives a count which indicates the number of times this response pattern should be counted in performing computations. The count can be non–integer. For example, if there were two examinees with the same response pattern in the same group they could be added with two `add_examinee` commands, or one `add_examinee` command with a count of 2. The default value of the third argument if it is not present is 1. The `add_examinee` returns the examinee number associated with the examinee added. The examinee number for the examinee added with the first `add_examinee` command is 1, etc. This number can be used in other commands which take an examinee number as an argument.

allocate_items_dist *nitems ?*`-models` *list? ?*`-models_str` *string?* Tcl
?`-num_latent_dist_points` *npoints? ?*`-num_groups` *ngroups?*
?`-latent_dist_range` *list? ?*`-unique_points`*?*

The `allocate_items_dist` command is documented in the previous section.

bootstrap_seed *seed* C++

Assign a seed for the random number generator used in the bootstrap_sample command. If this command is not used then the random number generator is initialized with an arbitrary seed.

bootstrap_sample C++

Generate a bootstrap sample of examinees. This command will modify the count associated with each examinee to be the number of times the examinee is included in the bootstrap sample. The `examinee_get_count` command can be used to return the count assigned to each examinee by the `bootstrap_sample` command. The counts associated with examinees prior to using the `bootstrap_sample` command are replaced.

delete_items_dist C++

The `delete_items_dist` command indicates the completion of a sequence of commands begun by a `new_items_dist` command. Memory and resources allocated by the `new_items_dist` command and subsequent commands are released. A `new_items_dist` command should be paired with a corresponding `delete_items_dist` command.

delete_estep *obj* C++

The `delete_estep` command disposes of an E–step object created with the `new_estep` command. The argument is a variable containing an E–step object. An example of using the `delete_estep` command is given in the description of the `estep_compute` command.

dist_get_point *cat ?group?* C++

Returns the point value for one category of the discrete latent variable distribution. The first argument is the number of the category for which the point value is returned: 1 for the first (lowest) category, 2 for the second category, etc. The second argument is the number of the examinee group (1, 2, . . .) for which the point value is returned. The default group used if the second argument is not given is 1. The third argument is only used if different latent distribution points are used for different examinee groups as specified in the `new_items_dist` or `allocate_items_dist` command.

dist_get_points *?group?* C++

Returns a list containing the point values for all categories of the discrete latent variable distribution. The argument gives the examinee group (1, 2, . . .) for which the point values should be returned. The default group used when the argument is not given is 1. The third argument is only used if different latent distribution points are used for different examinee groups as specified in the `new_items_dist` or `allocate_items_dist` command.

dist_get_prob *cat ?group?* C++

Returns the discrete probability for one category of the latent variable distribution for one examinee group. The first argument is the number of the latent variable category for which the probability is returned (1, 2, . . .). The second argument is the examinee group (1, 2, . . .) for which the probability should be returned. If the second argument is not present a default value of 1 is used.

dist_get_probs *?group?* C++

Returns a list of the discrete probabilities for all categories of the latent variable distribution for one examinee group. The argument is the examinee group (1, 2, . . .) for which the probabilities should be returned. If the argument is not present a default value of 1 is used.

dist_mean_sd *?group?* C++

Returns a list containing the mean and standard deviation of the discrete latent variable distribution in the examinee group indicated by the argument (the number of the examinee group— 1, 2, . . .). If the first argument is not present a default value of 1 is used. For example:

```
# find mean and standard deviation of latent variable
# distribution in groups 1 and 2
set moments1 [dist_mean_sd 1]
set moments2 [dist_mean_sd 2]
```

```
# print means for groups 1 and 2 to log file
puts_log "Means: [lindex $moments1 0], [lindex $moments2 0]"

# print standard deviations for groups 1 and 2 to log file
puts_log "s.d.'s: [lindex $moments1 1], [lindex $moments2 1]"
```

dist_scale *mean sd ?group?* C++

The `dist_scale` scales points of the discrete latent variable distribution so the mean and standard deviation of the distribution for the examinee group given by the last argument (the number of the examinee group — 1, 2, . . .) are equal to the values specified by the first and second arguments. If the last argument is not present the default value of 1 is used. If different latent distribution points are used for different groups the points for all groups are transformed to the new scale. A list is returned containing the slope and intercept of the scale transformation.

dist_set_point *cat point ?group?* C++

Sets the value of the latent variable for one category of the discrete latent variable distribution in one examinee group. The first argument is the number of the category to set: 1 for the first (lowest) category, 2 for the second category, etc. The second argument is the value that the point for that category should be set equal to. The third argument is a group number of the group for which the point is to be set. The default value of the third argument if it is not present is 1. The third argument is only used if different latent distribution points are used for different examinee groups as specified in the `new_items_dist` or `allocate_items_dist` command.

dist_set_points *list ?group?* C++

Sets the value of the latent variable for all categories of the discrete latent variable distribution in one examinee group. The first argument is a list giving values that the points of each category should be set equal to. There should be as many elements in the list as there are latent variable categories. The number of latent variable categories is set with the `allocate_items_dist` command or the `new_items_dist` command. The third argument is a group number of the group for which the points are to be set. The default value of the third argument if it is not present is 1. The third argument is only used if different latent distribution points are used for different examinee groups as specified in the `new_items_dist` or `allocate_items_dist` command.

dist_set_prob *cat prob ?group?* C++

Sets the discrete probability for one category of the latent variable distribution in one examinee group. The first argument is the number of the latent variable category for which the probability is set (1, 2, . . .). The second argument is the value the probability should be set to. The third argument is the examinee group (the number of the examinee group— 1, 2, . . .) for which the probability should be set. If the third argument is not present a default value of 1 is used. When the probability for a single category is set the remaining probabilities are not standardized so all probabilities sum to one. It is the responsibility of the user to make sure the sum is one after all probabilities have been assigned.

dist_set_probs *list ?group?* C++

Sets the discrete probabilities for all categories of the latent variable distribution for one examinee group. The first argument is a list of probabilities which will be assigned to the distribution. The number of elements in the list must be equal to the number of categories in the discrete latent variable distribution as set with the `allocate_items_dist` or `new_items_dist` command. The values in the list are standardized to sum to 1. The second argument is the examinee group (1, 2, . . .) for which the probabilities should be set. If the second argument is not present a default value of 1 is used.

dist_transform *slope intercept* C++

The `dist_transform` transforms the points of the discrete latent variable distribution to a new scale using a linear transformation given by the the arguments. If different latent distribution points are used for different groups the points for all groups are transformed to the new scale.

dist_unique_points C++

The `dist_unique_points` command returns a one if different latent distribution points are used for different examinee groups, as requested in the `new_items_dist` or `allocate_items_dist` command, otherwise the command returns zero.

EM_steps *?*`-max_iter` *iter? ?*`-crit` *d? ?*`-estim_dist`*? ?*`-scale_points`*? ?*`-no_print_iter`*? ?*`-no_mstep_iter_error`*? ?*`-estim_dist_mean_sd`*? ?*`-estim_dist_mean`*?* Tcl

The `EM_steps` command is documented in the previous section.

estep_compute *obj ?compute_post? ?store_post? ?items?* C++

The `estep_compute` command performs E–step computations using an E–step object created with the `new_estep` command. The value returned is the logarithm of the marginal posterior density computed using the current item parameter estimates. This is the value the EM algorithm is maximizing. The arguments are positional, so they must appear in the indicated order. The first argument is an E–step object created with the `new_estep` command. The items used to compute the posterior distribution for each examinee are determined when the E–step object is created. The current item parameter estimates are used for the E–step calculation. If the second argument is non–zero the posterior distributions are computed for each examinee, otherwise posterior distributions previously computed in an `estep_compute` command are used. If the second argument is not present the examinee posterior distributions are computed (the same as the second argument being non–zero). If the third argument is non-zero the posterior distributions for examinees are stored. The stored posterior distribution for an examinee can be obtained with the `examinee_get_posterior` command. If the third argument is not present the examinee posterior distributions are not stored (the same as the third argument being zero). The fourth argument is a list of item numbers indicating the items for which the n's and r's used in the next M–step are updated (see Woodruff and Hanson, 1997; Hanson, 1998). If the fourth argument is not present then the n's and r's are updated for all items

used to compute the examinee posterior distributions as indicated in the `new_estep` command. If an empty list is passed as the fourth argument the n's and r's are not updated for any items. The following is an example of how to use the `estep_compute` command. The example in Section 2.4 provides an illustration of the usefulness of the `estep_compute` command.

```
# Make a new E-step object and assign it to the variable e.
# Items 1, 3, 5-10, 15, and 18-22 will be used to compute
# the posterior latent variable distribution for each
# examinee in the E-step calculation.
set e [new_estep [concat 1 3 [seq 5 10] 15 [seq 18 22]]]

# compute E-step storing posterior distributions
# for all examinees
estep_compute $e 1 1

# print posterior mean (EAP estimate) for
# examinee 1 to log file
puts_log [examinee_posterior_mean 1]

# delete E-step object
delete_estep $e
```

Note that the value returned by the `estep_compute` command is the logarithm of the marginal posterior density at the values of the item parameter estimates, not the loglikelihood. The logarithm of the marginal posterior density is the loglikelihood summed with the logarithm of the prior densities of all item parameters. The loglikelihood for the current item parameter estimates can be computed using the following commands:

```
# Create new E-step object
set e [new_estep]

# Compute the loglikelihood. The empty list as last
# argument to estep_compute indicates that n's and
# r's will not be updated for any items. The log of
# the prior density is only added to the
# loglikelihood for items for which n's and
# r's are updated. Thus, the value returned
# is the loglikelihood.
set loglikelihood [estep_compute $e 1 0 {}]

# Print loglikelihood to log file
puts_log "Loglikelihood: $loglikelihood\n"

# Delete E-step object
delete_estep $e
```

examinee_get_count *examineeno* C++

Returns the count for the examinee corresponding to the examinee number given by the first argument. For an explanation of the examinee count see the description of the `add_examinee` command.

examinee_get_posterior *examinee* C++

The `examinee_get_posterior` command returns a list containing the posterior latent variable distribution for the examinee corresponding to the examinee number given by the argument. This distribution must have been previously stored for the examinee, with a command such as '`estep_compute $e 1 1`'.

examinee_get_group *examinee* C++

The `examinee_get_group` command returns an integer giving the group of the examinee corresponding to the examinee number given by the argument. The first group is indicated by a 1, the second group by a 2, etc.

examinee_set_group *examinee group* C++

The `examinee_set_group` command sets the group of the examinee corresponding to the examinee number given by the argument to the group number given by the second argument. The first group is indicated by a 1, the second group by a 2, etc.

examinee_posterior_mean *examinee* C++

The `examinee_posterior_mean` command returns the mean of the posterior latent variable distribution for an examinee — the expected a posterior (EAP) estimate of examinee proficiency. This distribution must have been previously stored for the examinee, with a command such as '`estep_compute $e 1 1`'. The argument is an examinee number for the examinee for which the posterior mean is returned. An example of using the `examinee_posterior_mean` command is given in the description of the `estep_compute` command and in the example in Section 2.3.

examinee_set_count *examineeno count* C++

Sets the count for the examinee corresponding to the examinee number given by the first argument to the value given by the second argument. For an explanation of the examinee count see the description of the `add_examinee` command.

examinee_set_posterior *examinee list* C++

The `examinee_set_posterior` command sets the probabilities of the posterior latent variable distribution for the examinee corresponding to the examinee number given by the first argument to the list of probabilities given by the second argument. The number of probabilities in the list given by the second argument must be equal to the number of categories in the discrete latent variable distribution as specified in the `new_items_dist` or `allocate_items_dist` command. The probabilities are standardized so they sum to one.

examinee_theta_MLE *examinee min_theta max_theta prec itemno* C++

The `examinee_theta_MLE` command returns a maximum likelihood estimate (MLE) of the latent variable for an examinee based on the examinee item responses and the

current item parameter estimates. The first argument is an examinee item number identifying the examinee for which the MLE is returned. The second and third arguments are the minimum and maximum values which the estimate can assume. The fourth argument is a precision representing the length of interval in which MLE is determined to lie. If the fourth argument is not present a value of 0.001 is used. The fifth argument is a list of item numbers identifying the items used to compute the MLE. If the fifth argument is not present the MLE will be computed using all items.

examinee_responses *examinee* C++

Returns a list of integers representing an examinee's responses to all items. The argument is an examinee number of the examinee for which the item responses are returned. The list returned contains integers where zero represents a response in the first response category and a response in the highest response category is indicated by an integer equal to the number of response categories minus one. A negative number for an item response indicates the examinee did not respond to the item.

examinee_response_str *examinee* C++

Returns a string representing an examinee's responses to all items. The argument is an examinee number of the examinee for which the item responses are returned. The string returned contains characters where zero represents a response in the first response category, a two represents a response in the second response category, etc. The character that represents a missing response is the character assigned using the `set_missing_resp` command (a period by default). This command is only useful when the number of response categories for all items is less than 10.

examinees_count *?group?* C++

Returns the sum of examinee counts in an examinee group, or across all examinee groups. The count for each examinee is the value assigned when the examinee is added with the `add_examinee` command. The count for an examinee can also be assigned by the `set_examinee_count` command or the `bootstrap_sample` command. The argument is an integer giving the examinee group for which the sum of the examinee counts is to be returned. If the argument is not present, or is equal to zero, the sum of examinee counts across all groups is returned. If the count for each examinee is 1 then the `examinees_count` command with no argument returns the same number as the `num_examinees` command.

get_base_group C++

Returns the number of the group used as the base group when standardizing the latent variable distribution (for instance, when the `-scale_points` option is specified in the `EM_steps` command). The base group can be set with the `set_base_group` command.

get_default_prior_param *param* C++

Returns a vector of parameters for the default prior distribution corresponding to the item parameter given by the argument ('`a`', '`b`', or '`c`'). The default prior can be set with the `set_default_prior` command.

get_default_prior_type *param* C++

Returns the name of the default prior distribution corresponding to the item parameter given by the argument ('a', 'b', or 'c'). The string returned is either '`normal`' (normal distribution), '`lognormal`' (lognormal distribution), '`beta`' (four–parameter beta distribution), or '`none`' (no prior distribution). The default prior can be set with the `set_default_prior` command.

get_responses *line offsets lengths* C++

The `get_responses` command reads item responses from a string given by the first argument and returns a list of integer item responses (0 corresponding to the first response category, 1 corresponding to the second response category, etc., and -1 indicating no response). The second argument is a list giving the zero-based offset of each item response in the string given by the first argument (the first character in the string is at offset zero). The third argument is a list giving the number of consecutive characters in the string that are to be read to obtain the item response for each item. The number of elements in the lists given by the second and third arguments do not need to be equal to the total number of items as specified in the `allocate_items_dist` or `new_items_dist` command. For example:

```
# Response record containing responses to five items beginning in
# column 3. The response to fourth item is missing.
set line {12010.1}

# Variable resp will contain the list
# {0 1 0 -1 1} of item responses
set resp [get_responses $line {2 3 4 5 6} {1 1 1 1 1}]
```

get_responses_missing *line offsets lengths items* C++

The `get_responses_missing` command reads item responses for a subset of items from a string given by the first argument and returns a list of integer item responses to all items (0 corresponding to the first response category, 1 corresponding to the second response category, etc., and -1 indicating no response). The second and third arguments are the same as the second and third arguments of the `get_responses` command. The fourth argument is a list of item numbers for which responses are to be read. The responses to the items indicated by the fourth argument are read from the string given by the first argument. Responses to any items not read from the string are assigned -1 (indicating a missing response).

item_cat_counts *itemno ?group?* C++

Returns a list giving the sum of examinee counts in each response category for the item corresponding to the item number given by the first argument. If the second argument is used the counts are return just for examinees in the examinee group indicated. If the second argument is not present, or is equal to zero, the counts are returned for examinees in all examinee groups. The count for each examinee is the value assigned when the examinee is added with the `add_examinee` command, or the value assigned to an examinee with the `set_examinee_count` command. If the count for each examinee is 1 the number of examinees who responded in each response category of the item is returned.

item_get_all_params *itemno* C++

Returns a list of values of all item parameters for the item corresponding to item number given by the argument. Both fixed and estimated parameters are returned. For the 3PL, 2PL, and 1PL models the list returned contains three elements corresponding to the values of the a–, b–, and c– parameters, respectively. For the 2PL and 1PL models the c–parameter is fixed, and for the 1PL model the a–parameter is fixed. For the GPCM and PCM the list returned contains a a–parameter followed by b–parameters in order of increasing response category. For the PCM the a–parameter is fixed.

item_get_model *itemno* C++

Returns a string describing model used for the item corresponding to the item number given by the argument, where '`3PL`' = three–parameter logistic, '`2PL`' = two–parameter logistic, '`1PL`' = one–parameter logistic, '`GPCM`' = generalized partial credit model, '`PCM`' = partial credit model.

item_get_name *itemno* C++

Returns a string containing the name of the item corresponding to the item number given by the argument. Item names are set with the `item_set_name` or `items_set_names` command. If the item was not assigned a name the item number (the command argument) is returned.

item_get_param *param itemno* C++

Returns the value of the item parameter for the parameter given by the first argument and the item corresponding to item number given by the second argument. The first argument is an integer giving the index of the parameter to return in the list of parameters for the item. For a three–parameter logistic model the indices for the a–, b–, and c– parameters are 1, 2, and 3. For a two–parameter logistic model the indices for the a– and b– parameters are 1, 2. for a one–parameter logistic model the index for the b–parameter is 1. For a generalized partial credit model the index for the a–parameter is 1, and the indices of the b–parameters are 2, 3, etc. For the partial credit model the indices for the b–parameters are 1, 2, etc.

item_get_params *itemno* C++

Returns a list of values of all estimated item parameters for the item corresponding to item number given by the argument. For the three–parameter logistic model the list contains the a–, b–, and c– parameters, in that order. For the two–parameter logistic model the list contains the list contains the a–parameter followed by the b–parameter. For the one–parameter logistic model the list contains the b–parameter. For the generalized partial credit model the list contains the a–parameter, followed by the b–parameters in order of increasing response category. For the partial credit model the list contains the b–parameters in order of increasing response category.

item_get_prior_type *param itemno* C++

Returns the type of prior distribution ('`normal`', '`lognormal`', '`beta`', or '`none`') used for the parameter given by the first argument for the item corresponding to the item

number given by the second argument. The first argument is an integer index giving the position of the parameter in the list of parameters for the item. The index corresponding to each parameter is given in the description of the `item_get_param` command.

item_get_prior_param *param itemno* C++

Returns a list containing the parameters of the prior distribution used for the parameter given by the first argument for the item corresponding to item number given by the second argument. The first argument is an integer index giving the position of the parameter in the list of parameters for the item. The index corresponding to each parameter is given in the description of the `item_get_param` command. The number of elements in the returned list depends on the prior distribution used for the item: 2 (mean and standard deviation) for the normal and lognormal distributions, 4 (two shape parameters, the lower limit and the upper limit) for the four–parameter beta distribution. If no prior distribution is used for the item an empty list is returned.

item_num_params *itemno* C++

Returns the number of estimated parameters for the item corresponding to item number given by the argument.

item_num_resp_cat *itemno* C++

Returns the number of response categories (number of possible item responses) for the item corresponding to item number given by the argument.

item_prob_resp *itemno response theta* C++

The `item_prob_resp` returns the probability that an examinee with a latent variable value given by the third argument would give the response given by the second argument to the item corresponding to the item number given by the first argument. The second argument should be an integer corresponding to an item response, where a response in the first response category is 0, a response in the second response category is 1, etc.

item_resp_count *itemno ?group?* C++

Returns the sum of the examinee counts for examinees responding to the item corresponding to the item number given by the first argument. If the second argument is used the count is return just for examinees in the examinee group indicated. If the second argument is not present, or is equal to zero, the count is returned for examinees in all examinee groups. The count for each examinee is the value assigned when the examinee is added with the `add_examinee` command, or the value assigned to an examinee with the `set_examinee_count` or `bootstrap_sample` command. If the count for each examinee is 1 the number of examinees who responded to the item is returned.

item_scale_params *itemno slope intercept ?ignore_priors?* C++

Transform the scale of the parameters for the item associated with the item number given by the first argument using the latent variable transformation with slope and

intercept given by the second and third arguments, respectively. The `item_scale_params` command returns zero if the parameters were successfully transformed, or one if the scaling would result in one or more parameters having values with zero prior density. If the fourth argument is zero then the command returns a 1 if any transformed parameter for the item has zero prior density, and the item parameters remain unchanged. If the fourth argument is non–zero then the parameters are transformed even if one of the transformed parameters has zero prior density, and the command always returns 0. The default value of the fourth argument if it is not present is zero.

item_set_all_params *itemno list* C++

Set the values of all item parameters for the item corresponding to the item number given by the first argument to the values given in the second argument. Values of both fixed and estimated item parameters are set. The order of the parameters in the list given by the second argument is given in the description of the `item_get_all_params` command. For example:

```
# Set the values of the fixed a and c parameters, and
# the estimated b parameter for item 5
# (modeled using the 1PL model) to 1.0, 0.0, and 0.2, respectively.
item_set_all_params 5 {1.0 0.0 0.2}
```

item_set_model *itemno model* C++

Sets the model used for the item corresponding to the item number given by the first argument to the model corresponding to the string given by the second argument. The string identifying the model should be equal to either '`3PL`' (three–parameter logistic), '`2PL`' (two–parameter logistic), '`1PL`' (one–parameter logistic), '`GPCM`' (generalized partial credit model), or '`PCM`' (partial credit model). The model for an item with more than 2 response categories cannot be changed to the 3PL, 2PL, or 1PL model.

item_set_name *itemno name* Tcl

The `item_set_name` command associates a name with an item. The first argument is an item number corresponding to the item for which the name is to be assigned. The second argument is a string giving the name for that item. Item names are only used to label items in output, consequently it is possible to assign names to some items and not others. For items without item names, the item number is used to label the item in output. Item numbers, not item names, are used as arguments in ICL commands to refer to items.

items_set_names *names ?itemnos?* Tcl

The `item_set_names` command associates names with items. The first argument is a list of item names. The second argument is a list giving item numbers corresponding to the list of item names given by the first argument. If the second argument is not present default values of 1, 2, . . . , up to the number of elements in the name list are used. The lists given by the first and second arguments must have the same number of elements. Item names are only used to label items in output, consequently it is possible to assign names to some items and not others. For items without item

names, the item number is used to label the item in output. Item numbers, not item names, are used as arguments in ICL commands to refer to items. An example of the `items_set_names` command is

```
# Assign item names to the first five items
items_set_names {item20 item21 item22 item23 item24}
```

item_set_param *param itemno value* C++

Set the value of the item parameter given by the first argument for the item corresponding to item number given by the second argument to the value given by the third argument. The first argument is an integer index giving the position of the parameter in the list of parameters for the item. The index corresponding to each parameter is given in the description of the `item_get_param` command.

item_set_params *itemno list* C++

Set the values of all item parameters for the item corresponding to the item number given by the first argument to the values given in the second argument. The order of the parameters in the list given by the second argument is given in the description of the `item_get_params` command. For example:

```
# Set the values of the a, b, and c parameters for item 5
# (modeled using the 3PL model) to 1.0, 0.0, and 0.2, respectively.
item_set_params 5 {1.0 0.0 0.2}
```

item_set_prior *param itemno prior ?list?* C++

Sets the prior distribution for the parameter given by the first argument of the item corresponding to item number given by the second argument. The first argument is an integer index giving the position of the parameter in the list of parameters for the item. The index corresponding to each parameter is given in the description of the `item_get_param` command. The type of prior distribution is given by the third argument. It must be one of '`normal`' (normal distribution), '`lognormal`' (lognormal distribution), '`beta`' (four–parameter beta distribution), or '`none`' (no prior distribution). The parameters of the prior distribution are given in a list as the last argument. The number of prior parameters depends on the prior distribution: two (mean and standard deviation) for '`normal`' and '`lognormal`', four (two shape parameters, lower limit, and upper limit) for '`beta`'. If '`none`' is specified as the prior distribution the last parameter is not needed. The default prior distributions for the item parameters that are used if an `item_set_prior` command is not present can be set by the '`-default_prior_a`', '`-default_prior_b`', and '`-default_prior_c`' options of the `options` command, or the `set_default_prior` command. The default prior distributions if the `item_set_prior` command is not used are presented in the description of the `options` command. An example of using the `item_set_prior` command is:

```
# set the prior distribution for the c parameter of item 10
# (modeled using the three-parameter logistic model) to
# four-parameter beta with shape parameters 1.5 and 1.5, lower
# limit 0.0 and upper limit 0.5. This is a symmetric unimodal
# distribution in the interval 0.0 to 0.5 with mode at 0.25.
item_set_prior 3 10 beta {1.5 1.5 0.0 0.5}
```

item_3PL_starting_values *?use_all? ?item_all? ?itemno?* C++

The `item_3PL_starting_values` computes starting values for the item parameter estimates for items modeled using the three–parameter logistic (3PL), two–parameter logistic (2PL), and one–parameter logistic (1PL) models. If the first argument is non–zero then all examinees are used to compute initial proficiencies used in computing the starting values, even examinees who get all items correct or all items incorrect, and all items are used to compute initial item difficulties used in computing the starting values, even items answered correctly or incorrectly by all examinees. The default value of the first argument if it is not present is zero. If the second argument is non–zero all items are used to compute initial values of examinee proficiency used in computing the starting values, even those for which starting values are not being computed. If the second argument is zero then only items for which starting values are being computed are used to compute initial examinee proficiencies. The default value of the second argument if it is not present is zero. The third argument is a list of item numbers for which starting values are computed. This list must only contain item numbers for items modeled using the 3PL, 2PL, or 1PL models. If the third argument is not present starting values are computed for all items modeled by the 3PL, 2PL, or 1PL models. The value returned is an integer giving the number of items for which minimization procedure used to compute starting values failed — zero indicates starting values were successfully computed for all items.

Starting values are computed by producing a rough estimate of latent proficiency for each examinee based on their item responses using the PROX procedure (Linacre, 1994). Nonlinear regressions with these proficiencies as independent variables and item responses as dependent variable are used to get starting values for the item parameters. For more details on how the starting values are calculated see the C++ source file '`Start3PL.h`' in the ETIRM distribution.

items_set_prior *item_param prior_dist ?prior_params? ?itemno?* Tcl

The `items_set_prior` command sets the prior distribution for one item parameter for a set of items. The first argument is an integer index giving the position of the parameter in the list of parameters for the item. The index corresponding to each parameter is given in the description of the `item_get_param` command. The second argument is the type of prior distribution – either '`normal`', '`lognormal`', '`beta`', or '`none`' corresponding to a normal distribution, a lognormal distribution, a four–parameter beta distribution, or no prior, respectively. The third argument is a list containing the parameters of the prior distribution. For the normal and lognormal distribution the parameters are the mean and variance. For the four-parameter beta distribution the parameters are the two shape parameters, the lower limit, and the upper limit. This argument is optional because it is not needed if '`none`' is specified as the type of prior. The last argument is a list of item numbers identifying the items for which the prior distribution of the indicated parameter should be set. If the last argument is not present then the prior is set for all items. It is assumed that the parameter index given by the first argument identifies the same parameter for all items.

mstep_dist *estep group* C++

Performs the M–step calculation for a discrete latent variable distribution for one examinee group. The first argument is an E-step object created using the `new_estep` command. The second argument gives the number of the group (1, 2, . . .) for which the M–step calculation giving estimates of the probabilities in the latent variable distribution is performed.

mstep_item_param *?*`-items` *itemno? ?*`-no_max_iter_error`*?* Tcl

The `mstep_item_param` command performs the M–step calculation for item parameter estimates. The maximum relative difference in parameters from previous iteration to current iteration is returned. The `-items` option specifies a list of item numbers for which the M–step calculation is performed. If this option is not present the M–step calculation is performed for all items. If the `-no_max_iter_error` option is present the maximum number of iterations being exceeded in the M–step optimization for an item is not considered an error, otherwise if the maximum number of M–step iterations for an item is exceeded the calculation stops at the point where the error occurred.

mstep_items *?ignore? ?itemno?* C++

The `mstep_items` command performs the M–step calculation for item parameter estimates. The first argument is an integer flag. If the first argument is non–zero then the maximum number of iterations in the M-step being exceeded is not treated as an error, otherwise if the maximum number of M-step iterations is exceeded processing stops at the point where the error occurred. The second argument is a list of item numbers for which the M–step calculation is performed. If the second argument is not present the M–step calculation is performed for all items. Zero is returned if the calculation is successful. If the first argument is non-zero, and the only error that occurs is that the maximum number of M-step iterations is exceeded, then the negative of number of items for which maximum number iterations was exceeded is returned. If the first argument is zero and an error occurs, or an error other than exceeding the maximum number of M-step iterations occurs, then the item number for which error occurred is returned and processing is stopped at the point of the error.

mstep_max_diff C++

Returns the maximum relative difference between parameters in the current and previous EM iteration computed the last time the `mstep_items` was executed. The maximum is over all parameters for all items.

mstep_max_iter *itemno max* C++

Sets the maximum number of iterations used in the M–step optimization procedure for the item corresponding to the item number given in the first argument to the value given by the second argument. The default maximum number of M–step iterations used for each item when this command is not given is 150. The maximum number of iterations set with this command also applies to the optimization procedure used to compute the starting values in `item_3PL_starting_values` command.

mstep_message *itemno* C++

Returns the message generated in the last call to `mstep_items` from the optimization procedure used for an item. The argument is the item number for which the message is returned. The possible integers returned and their associated messages are:

0 — Optimal solution found, terminated with gradient small.

1 — Terminated with gradient small, solution is probably optimal.

2 — Terminated with step size small, solution is probably optimal.

3 — Lower point cannot be found, solution is probably optimal.

4 — Iteration limit exceeded.

5 — Too many large steps, function may be unbounded.

-1 — Analytic gradient check requested, but no analytic gradient supplied.

-2 — Analytic hessian check requested, but no analytic hessian supplied.

-3 — Illegal dimension.

-4 — Illegal tolerance.

-5 — Illegal iteration limit.

-6 — Minimization function has no good digits.

-7 — Iteration limit exceeded in line search.

-20 — Function not defined at starting value.

-21 — Check of analytic gradient failed.

-22 — Check of analytic hessian failed.

mstep_latent_dist *estep_obj ?*`-estim_base_group`*? ?*`-scale_points`*?* Tcl

The `mstep_latent_dist` command performs the M-step calculation for the probabilities of the discrete latent variable distributions. The maximum relative difference in distribution probabilities across points from the previous iteration to current iteration is returned. The first argument is an E–step object created with the `new_estep` command. If the optional `-estim_base_group` argument is present the probabilities of the latent variable distribution are estimated for the base group. If the `-estim_base_group` argument is not present only probabilities of the latent variable distribution for examinee groups other than the base group are estimated. If the optional `-scale_points` argument is used the points of latent variable distribution are linearly transformed so that the mean and s.d. in base examinee group are 0 and 1. This scale transformation is also applied to the parameter estimates for all items to put them on the same scale.

mstep_latent_dist_moments *estep_obj ?*`-mean_only`*?* Tcl

?`-estim_base_group`*?*

The `mstep_latent_dist_moments` command performs the M-step calculation to estimate the mean and standard deviation of the discrete latent variable distributions for all examinee groups except the base group for which the mean and standard deviation are fixed (unless the `-estim_base_group` option is used). This command modifies the points of the latent variable distribution for all groups except the base group to be consistent with an estimated mean and standard deviation in that group. The

probabilities of the latent variable distributions are not changed. The maximum relative difference in distribution means from the previous iteration to current iteration is returned. The first argument is an E–step object created with the `new_estep` command. If the optional `-mean_only` argument is present only the mean is estimated, not the standard deviation. If the optional `-estim_base_group` argument is present the mean and standard deviation of the base group are estimated. This command requires that different latent distribution points be used for different examinee groups as specified in the `new_items_dist` or `allocate_items_dist` command.

new_items_dist *nitems ?npoints? ?ngroups? ?models? ?range_list? ?unique_points?* C++

The `new_items_dist` command is used to specify the number of items to be modeled, and optionally, the number of points in the discrete latent variable distribution, the number of examinee groups, whether a dichotomous or polytomous model is to be used for each item, the minimum and maximum points of the discrete latent variable distribution, and whether unique points are used for the latent distributions in each group. Memory is allocated by the `new_items_dist` for the number of items and number of discrete latent variable points specified. The number of items and number of points in the discrete latent variable distribution can only be changed by the `new_items_dist` command. A `new_items_dist` command must be given before most other commands can be used.

The first argument gives the total number of items to be modeled. In subsequent commands items are identified by item number. If the number of items specified is n then item numbers 1 through n are used to refer to these items in subsequent commands. The second argument is the number of categories used for the discrete latent variable distribution. The default value if the second argument is not present is 40. The third argument is the number of groups of examinees in the data that are sampled from different populations. The default value if the third argument is not present is 1. Multiple group estimation is used if the number of groups is greater than 1, and in this case a group identifier must be indicated for each examinee.

The fourth argument is a list containing an integer for each item. The integer 1 corresponding to an item means the item is modeled using a dichotomous model. If the integer corresponding to an item is greater than 1 that means the item is a polytomous item with that number of response categories modeled using a polytomous model. If the fourth argument is not present the models for all items are assumed to be dichotomous.

The fifth argument is a list giving the minimum and maximum points of the discrete latent variable distribution. The first element of the list is the minimum point and the second element of the list is the maximum point of the latent variable distribution. The default minimum and maximum points of the latent variable distribution are -4 and 4 if the fifth argument is not present.

The sixth argument is an integer which if nonzero indicates different latent distribution points are used for different examinee groups. If the sixth argument is zero the same set of latent variable points is used for all examinee groups. The default value of the sixth argument if it is not present is zero.

The `new_items_dist` command initializes the IRT model used for each item and the prior distributions of all item parameters to default values. The model and prior distributions used as defaults can be set using the `set_default_model_dichotomous`, `set_default_model_polytomous`, and `set_default_prior` commands. If the `set_default_model_dichotomous` command is not used the default model for dichotomous items is the three-parameter logistic model. If the `set_default_model_polytomous` command is not used the default model for polytomous items is the generalized partial credit model. The default prior distributions for the item parameters are described in the documentation of the `allocate_items_dist` command.

The values of the discrete latent variable points and probabilities are set to initial values. There is a set of probabilities for each examinee group. There is one set of points for all examinee groups unless the fifth argument is nonzero, in which case there is a different set of points for each examinee group. The points are equally spaced in the range given by the fourth argument. The weights are chosen so the resulting discrete distribution approximates a standard normal distribution. The initial probabilities and points are the same across groups. The points can be changed with the `dist_set_point` and `dist_set_points` commands. The probabilities can be changed by the `dist_set_prob` and `dist_set_probs` commands.

new_estep *?items?* C++

Creates and returns a new E-step object to use for E-step computations. The optional argument is a list of item numbers indicating the items to be used in the E-step calculation to compute examinee posterior latent variable distributions. The `estep_compute` command is used to perform an E-step computation using an E-step object created with the `new_estep` command. An E-step object created with the `new_estep` command should be disposed of with the `delete_estep` command when it is no longer needed. Examples of using the `new_estep` command is given in the description of the `estep_compute` command and the example given in Section 2.4.

normal_dist_points *npoints min max ?mean? ?sd?* C++

Returns a list of points of a discrete distribution with the number of points given by first argument. The second and third arguments are minimum and maximum points for a standard normal distribution. The list of points returned are transformed equally spaced points between the second and third argument, inclusive. The transformation is computed so that the mean and standard deviation computing using these points with the probabilities returned by `normal_dist_prob` (using the same second and third arguments) equals the mean and standard deviation given by the fourth and fifth arguments. Thus, the minimum and maximum points returned will differ from the second and third argument if the fourth and fifth arguments differ from zero and one, respectively. If the fourth argument is not present the default value of zero is used, and if the fifth argument is not present the default value of one is used.

normal_dist_prob *npoints min max* C++

Returns a list of probabilities of a discrete distribution that approximates a standard normal distribution with the number of discrete points given by the first argument,

the minimum point given by the second argument, and the maximum point given by the third argument.

num_examinees C++
The `num_examinees` command returns the number of examinees for which data have been assigned using the `add_examinee` command.

num_groups C++
The `num_groups` command returns the number of examinee groups as set with the `allocate_items_dist` or `new_items_dist` command.

num_items C++
The `num_items` command returns the number of items as set with the `allocate_items_dist` or `new_items_dist` command.

num_latent_dist_points C++
The `num_latent_dist_points` command returns the number of points in the discrete latent variable distribution as set with the `allocate_items_dist` or `new_items_dist` command.

options *?*`-D` *d? ?*`-missing_resp` *resp? ?*`-base_group` *group? ?*`-default_model_dichotomous` *model? ?*`-default_model_polytomous` *model? ?*`-max_iter_optimize` *max? ?*`-default_prior_a` *prior? ?*`-default_prior_b` *prior? ?*`-default_prior_c` *prior? ?*`-default_dist_range` *list?* Tcl
The `options` command is documented in the previous section.

output *?*`-log_file` *file_name? ?*`-no_print`*?* Tcl
The `output` command is documented in the previous section.

print *?*`-item_param`*? ?*`-latent_dist`*? ?*`-latent_dist_moments`*? ?*`-no_heading`*? ?*`-items` *itemno? ?*`-format` *string? ?*`-item_model`*?* Tcl
The `print` command is documented in the previous section.

puts_log *?*`-nonewline`*? string* Tcl
The `puts_log` command writes a string to the log file. If the argument `-nonewline` is present a newline character is not written after the end of the string. The `-nonewline` argument must be the first argument if it is present.

read_examinees *file resp_format ?group_format?* Tcl
Documentation for the `read_examinees` command is given in the previous section.

read_examinees_channel *fileID resp_format ?group_format?* Tcl
The `read_examinees_channel` command reads examinee responses to the items, and optionally an examinee group, from a an open I/O channel (e.g., file or process pipeline). The first argument is an identifier for an I/O channel returned by the

Tcl `open` command. The remaining arguments are the same as those for the `read_examinees` command. This command functions the same as the `read_examinees` command except that it takes an open I/O channel rather than a file name as the first argument.

read_examinees_missing *file form_format itemNos resp_format ?group_format? ?group_conv?* Tcl

The `read_examinees_missing` command reads examinee responses to the items, and optionally an examinee group, from a file. While the `read_examinees` command requires responses to all items be read for every examinee, this command allows reading of examinee records containing responses to only some of the items. The responses to the items not read for an examinee are assumed missing. The first argument is the name of the file to read examinee responses from.

A set of items for which a group of examinees have responses is called a form. The second argument consists of a format list which indicates which columns in each record contain a form identifier for each examinee. The syntax of the format list is explained in the description of the `read_examinees` command. The form identifier can be any string, it does not need to be an integer. If the second argument is an integer then that integer is taken as the form identifier for all records in the file. This can be useful when responses to the different forms are contained in separate files. In this case each `read_examinees_missing` command is reading a file containing only one form so a form identifier for each examinee is not needed.

The third and fourth argument are names of Tcl arrays, where the indices of the array are the form identifiers (Tcl arrays can be indexed by general strings in addition to integers). Note the names of the arrays are used as arguments. The array name should not be preceded by a dollar sign when used as the third or fourth argument. The third argument is the name of an array containing lists of item numbers for which responses are to be read for examinees taking each form. The fourth argument is the name of an array containing format lists used to read examinee responses for each form. The number of elements in the arrays whose names are passed as the third and fourth arguments should be equal to the number of forms.

A '1' or '0' indicates a correct or incorrect response to the item, respectively. If the examinee did not respond to the item, for example if they did not receive the item, a '.' (period) should be given as the item response (the character which indicates a missing response can be changed with the `options` command).

The optional fifth argument consists of a format list which indicates which columns in each record contain a group identifier for the examinee. The fifth argument is only needed if the number of groups indicated in the `allocate_items_dist` command is greater than 1. If the fifth argument is an integer then that integer is taken as the group for all records in the file. This can be useful when responses for different groups are kept in separate files. The optional sixth argument is the name of an array that gives the group number (1 through the number of groups) corresponding to the group identifiers read for each record. The `read_examinees_missing` command returns the number of examinees for whom item responses were read.

The following example indicates how to use the `read_examinees_missing` command. An example using the `read_examinees_missing` command is presented in Section 2.5.

```
# Read data from a common item nonequivalent groups
# design. Two forms of a test (A1 and B2) are taken
# by different nonequivalent groups of examinees.
# Forms A1 and B2 both have 60 items, with 20 items
# in common between the two forms.
# Items 1-40 are the items unique to form A1.
# Items 41-60 are the items common to forms A1 and B2.
# Items 61-100 are the items unique to form B2.
# The input record contains form in columns 1-2 and
# responses to the 60 items taken in columns 3-62.

# For examinees taking form A1 responses to items
# 1-60 are read
set items(A1) [seq 1 60]

# For examinees taking form B2 responses to items
# 41-100 are read
set items(B2) [seq 41 100]

# Responses are read from columns 3-62 for both forms
set respFmt(A1) {@3 60i1}
set respFmt(B2) {@3 60i1}

# Group 1 are examinees taking form A1 and group
# 2 are examinees taking form B2
set groups(A1) 1
set groups(B2) 2

# The form identifier is read from the first two columns
# of the examinee record. For each examinee the
# identifier should be either A1 or B2.
set formFmt a2

# Read examinee responses from file test.dat
# In this case form identifier and group identifier
# are the same for all examinees.
read_examinees_missing test.dat $formFmt items respFmt \
    $formFmt groups
```

read_examinees_missing_channel *fileID form_format itemNos resp_format ?group_format? ?group_conv?* Tcl

The `read_examinees_missing_channel` allows reading of examinee records containing responses to only some of the items from an open I/O channel (e.g., file or process pipeline). The first argument is an identifier for an I/O channel returned by the Tcl `open` command. The remaining arguments are the same as those for the `read_examinees_missing` command. This command functions the same as the `read_examinees_missing` command except that it takes an open I/O channel rather than a file name as the first argument.

read_item_param *file ?*`-item_model`*? ?*`-no_item_numbers`*?* Tcl

The `read_item_param` command reads item parameters from a file. The first argument is the name of the file to read the item parameters from. The optional arguments are the same as for the `read_item_param_channel` command. The item parameters are read in the same way as described for the `read_item_param_channel` command.

read_item_param_channel *fileID ?*`-item_model`*? ?*`-no_item_numbers`*?* Tcl

The `read_item_param_channel` command reads item parameters from an open I/O channel (e.g., file or process pipeline). Each line read should contain an item number, followed by the parameters for that item. Optionally, the model used for the item may be present between the item number and first parameter. Lines are read until the end of the file, or until a blank line is read. The first argument is an identifier for an I/O channel returned by the Tcl `open` command. If the `-item_model` option is present the model used for each item (3PL, 2PL, 1PL, GPCM, or PCM) is read between the item number and first parameter. This model must match the model specified for the item with the `allocate_items_dist` or `new_items_dist` command. If the `-no_item_numbers` option is present it is assumed there are no item numbers present before the parameters on each line. In that case, the number for an item is taken to be the same as the line number. Both estimated and fixed parameters for each item are read. The order in which the parameters for an item should appear on a line is the same as that for the `item_get_all_params` command. The elements on each line need to separated by white space (spaces or tabs). It is not necessary that parameters for all items be read. The `read_item_param_channel` command will only assign parameters to items corresponding to an item number in the input file.

read_latent_dist *file ?*`-group` *groupno?* Tcl

The `read_latent_dist` command reads points and weights of the discrete latent variable distributions from a file. The first argument is the name of a file the latent distributions will be read from. The optional argument is the same as for the `read_latent_dist_channel` command. The latent distributions are read in the same way as described for the `read_latent_dist_channel` command.

read_latent_dist_channel *fileID ?*`-group` *groupno?* Tcl

The `read_latent_dist_channel` command reads points and weights of the discrete latent variable distributions from an open I/O channel (e.g., file or process pipeline). The argument is an identifier for an I/O channel returned by the Tcl `open` command from which to read the distributions. Each line read should contain the value for a discrete latent variable point, followed by the weights (probabilities) associated with that point for examinee groups 1, 2, up to the number of groups. The elements on each line need to separated by white space. The weights for each group should sum to one. The number of lines read should be equal the number of latent variable categories as indicated in the `allocate_items_dist` or `new_items_dist` command. In the case in which there are different latent variable points used for different examinee groups the single set of points read is assigned to all groups. The `-group` options specifies a single group for which the distribution should be read. When this option is used the first number on each line is read as the a point, and the second number on each line

is read as a weight for that group. If the `-group` option is used points and weights are only read for one group.

release_items_dist Tcl

The `release_items_dist` command is documented in the previous section.

rep *value number* Tcl

The `rep` command returns a list containing the value given by the first argument repeated the number of times given by the second argument. An example is:

```
# Assign x to be a list containing 60 1's.
set x [rep 1 60]
```

seq *min max ?inc?* Tcl

The `seq` command returns a list containing a sequence of integers where the first integer in the sequence is equal to the first argument, the last integer in the sequence is equal (or less than) the second argument, and the distance between consecutive integers in the sequence is equal to the third argument. The third argument is optional. If it is not present the default value of 1 is used. An example is:

```
# set the variable 'itemno' equal to the list of
# integers 1, 2, 3, ..., 60.
set itemno [seq 1 60]
```

set_base_group *group* C++

Set the base examinee group to the group corresponding to the group number given by the argument. The base group is used when standardizing the latent variable distribution (for instance, when the `-scale_points` option is specified in the `EM_steps` command).

set_default_D *D* C++

Set default value of logistic scaling constant (D) to the value of the command argument. A value of 1.7 makes the logistic ogive curve close to a normal ogive curve. If this command is not given 1.7 is used as the logistic scaling constant.

set_default_model_dichotomous *model* C++

Set the default model used for dichotomous items. The argument is a string equal to '`3PL`', '`2PL`', or '`1PL`' corresponding to the 3–, 2–, and 1–parameter logistic models, respectively. If this command is not given the default model used is the 3–parameter logistic. The model used for individual items can be set with the `item_set_model` command.

set_default_model_polytomous *model* C++

Set the default model used for polytomous items. The argument is a string equal to '`GPCM`' or '`PCM`' corresponding to the generalized partial credit model and the partial credit model, respectively. If this command is not given the default model used is the generalized partial credit model. The model used for individual items can be set with the `item_set_model` command.

set_default_prior *param priortype priorparam* C++

Set the default prior distribution used for the item parameter given by the first argument (a string equal to either '`a`', '`b`', or '`c`') to the prior distribution identified by the second and third arguments. The second argument specifies the type of prior distribution: '`normal`', '`lognormal`' '`beta`', and '`none`', for the normal distribution, the lognormal distribution, the four-parameter beta distribution, and no prior distribution, respectively. The third argument is a list containing the parameters of the prior distribution. For the normal and lognormal distributions two parameters need to be specified: a mean and standard deviation, in that order. For the four-parameter beta distribution the parameters that need to be specified are the two shape parameters, the lower limit, and the upper limit, in that order. For example,

```
# Specify lognormal with mean 0 and s.d. 1 as
# default prior for a-parameter
set_default_prior a lognormal {0.0 1.0}

# Specify a symmetric beta distribution with
# a mean of 0.2 as the default prior for the
# c-parameter
set_default_prior c beta {2.0 2.0 0.0 0.4}
```

The default prior distributions if the `set_default_prior` command is not used are given in the description of the `options` command.

set_missing_resp *char* C++

The `set_missing_resp` option is used to specify the character that indicates an examinee has not responded to an item. The command argument is the character that is used to represent a missing examinee response. Any character except '`0`' and '`1`', which indicate an incorrect and correct response, can be used. The default character indicating a missing response if the `set_missing_resp` command is not given is a period.

simulate_seed *seed* C++

Assign the random number generator seed used in the `simulate_responses` command. If this command is not used then the random number generator used for the `simulate_responses` command is initialized with an arbitrary seed.

simulate_responses *theta ?itemno?* C++

This command returns a list of simulated item responses. The first argument is the value of the latent variable for which item responses are to be simulated. The second argument is a list of item numbers for which responses will be generated. If the second argument is not present responses will be generated for all items. The list returned contains integers where a response in the first response category is represented by a 0, a response in the second response category is represented by a 1, and a response in the highest response category is indicated by an integer equal to the number of response categories for the item minus 1.

standardize_scale *mean sd ?group? ?ignore_prior?* Tcl

Transforms the points of the latent variable distribution so the mean and standard deviation of the distribution are equal to the first and second arguments in the group given by the third argument. The item parameters and latent distribution points in other groups are correspondingly transformed to be on the new scale. If the third argument is not present the base group is used. The base group can be set with the `set_base_group` command. If the fourth argument is zero then an error is generated if the transformation results in an item parameter with a zero prior density, otherwise no error is generated when a transformed item parameter has zero prior density. The default value of the fourth argument if it is not present is zero. A list containing the slope and intercept of the scale transformation is returned.

starting_values_dichotomous *?*`-items` *list? ?*`-use_all`*?* Tcl
?`-ignore_error`*?*

The `starting_values_dichotomous` command is documented in the previous section.

test_characteristic_curve *thetas ?itemno?* C++

The `test_characteristic_curve` command returns a list giving values of the test characteristic curve corresponding to specified values of the latent variable. The first argument is a list giving the values of the latent variable at which the test characteristic curve is to be computed. The second argument is a list of item numbers which define the test for which the test characteristic curve is computed. If the second argument is not present the test characteristic curve is computed using all items. The number of elements in the list returned is equal to the number of elements in the list given by the first argument.

transform_scale *slope intercept ?ignore_prior?* Tcl

Transform the item parameters and latent distribution points in all groups to a new scale using the linear transformation given by the arguments. If the third argument is zero then an error is generated if the transformation results in an item parameter with a zero prior density, otherwise no error is generated when a transformed item parameter has zero prior density. The default value of the third argument if it is not present is zero.

write_item_param_channel *fileID ?*`-format` *cformat? ?*`-item_model`*?* Tcl
?`-no_item_numbers`*? ?*`-items` *itemnos?*

The `write_item_param_channel` writes item parameters for a set of items to an open I/O channel (e.g., file or process pipeline). Each line contains an item number followed by the all item parameter estimates for the item. Optionally, the model associated with the item is printed between the item number and first parameter. Both estimated and fixed parameters for each item are written. The order in which the parameters for an item appear on the line is the same as that described for the `item_get_all_params` command. If an item name has been assigned to an item using the `item_set_name` or `items_set_names` commands then the item name is used in place of the item number. The elements on each line are separated by tab

characters. The first argument is an I/O channel returned by the Tcl `open` command. The `-format` option specifies a C sprintf–like format — `%[width][.precision]char`, where `char` = '`f`' (fixed point), '`e`' (scientific notation), or '`g`' (fixed point or scientific notation, which ever takes less space). The default format if the `-format` option is not present is `%.6f` (fixed point decimal with six places after the decimal point). A more detailed description of the format specification is given in Appendix A. If the `-model_item` option is present the model associated with each item (either 3PL, 2PL, 1PL, GPCM, or PCM) is printed between the item number and first parameter. If the `-no_item_numbers` option is present item numbers are not written before the parameters on each line. The `-items` option specifies a list of item numbers identifying the items for which the item parameters should be printed. If the `-items` option is not present the item parameters are printed for all items. An example of using the `write_item_param_channel` command is

```
# open file param.out for writing
set fileID [open param.out w]

# write current item parameter estimates in scientific notation
# with five digits after the decimal point
write_item_param_channel  $fileID -format %.5e

# close file
close $fileID
```

write_item_param *file ?*`-format` *cformat? ?*`-item_model`*? ?*`-items` *itemnos?* Tcl

The `write_item_param` command writes item parameters for a set of items to a file. The first argument is the name of the file the parameters will be written to. If the file does not exist it will be created, and if it does exist the contents will be overwritten. The optional arguments are the same as for the `write_item_param_channel` command. The output is the same as that described for the `write_item_param_channel` command.

write_latent_dist_channel *fileID ?*`-point_format` *format? ?*`-weight_format` *wformat? ?*`-groups` *groups?* Tcl

The `write_latent_dist_channel` command writes the points and weights of the discrete latent variable distributions to an open I/O channel (e.g., file or process pipeline). Each line contains a point followed by the weights for examinee groups 1, 2, etc, if the same points are used for all group. The elements on each line are separated by a tab character. The first argument is an identifier for an I/O channel returned by the Tcl `open` command. The `-point_format` option The `-point_format` and `-weight_format` options specify C sprintf–like formats — `%[width][.precision]char`, where `char` = '`f`' (fixed point), '`e`' (scientific notation), or '`g`' (fixed point or scientific notation, which ever takes less space). A more detailed description of the format specification is given in Appendix A. The `-point_format` option specifies the format to use for the points, and the `-weight_format` option specified the format to use for the weights (the same format is used or all groups). If the `-point_format` option is not present the format `%.6f`

is used, and if the `-weight_format` option is not present the format `%.6e` is used. The `-groups` option specifies a list of group numbers of groups for which weights are to be written. If the `-groups` option is not present weights are written for all groups. If different latent variable points are used for different examinee groups, as specified in the `allocate_items_dist` or `new_items_dist` command, then the `-groups` option is required and can contain only a single group number. In this case the `write_latent_dist_channel` command must be called multiple times to write the distributions for multiple groups. `write_latent_dist_channel` command is

```
# open file test.out for writing
set fileID [open test.out w]

# Write current latent variable distribution points and weights
# using scientific notation with 5 digits after the decimal
# point for the weights, and 8 digits after the decimal point
# for the weights.
write_latent_dist_channel  $fileID -point_format %.5e \
     -weight_format %.8e
```

write_latent_dist *file ?*`-point_format` *format? ?*`-weight_format` *wformat? ?*`-groups` *groups?* Tcl

The `write_latent_dist` command writes the points and weights of the discrete latent variable distributions to a file. The first argument is the name of the file to which the latent distribution points and weights are written. This file is created if it does not exist, and is overwritten if it does exist. The optional arguments are the same as those for the `write_latent_dist_channel` command. The output is the same as that described for the `write_latent_dist_channel` command.

1.4 Using Tcl

Command processing in ICL is handled by an embedded Tool Command Language (Tcl) interpreter. Tcl (pronounced "tickle") is a scripting language that is used as the command language for the ICL application. The Tcl interpreter used in ICL has been extended to include the commands described in the previous two sections.

In addition to the commands documented in this manual, all commands that are part of the basic Tcl interpreter are available to use as ICL commands. Basic use of ICL does not require knowing any Tcl commands, just the commands specific to ICL. This manual does not provide a discussion of how to use the basic Tcl commands, only a description of the commands specific to ICL. Several sources are available that describe the built–in Tcl commands. A good book for learning how to use Tcl is Welch (1999). A few chapters of Welch (1999) are freely available online including Chapter 1, which provides a nice introduction to Tcl. Nelson (2000) is a reference book containing descriptions of all Tcl commands. Information about getting started with Tcl, including documentation, and an introduction to Tcl syntax, is available from the Tcl Developer Xchange.

Many of the ICL commands described in the previous sections are written in Tcl. They are contained in the file '`icl.tcl`' included with the ICL distribution. Any of these commands can be overridden by redefining the command. The Tcl `rename` command can be used to

give a command a new name so it can be redefined yet still be assessable under a different name.

2 Examples

This chapter presents some examples of using ICL. The command, input, and output files for each of these examples are included with the ICL distribution in the '`examples`' directory.

The first two examples use only the basic ICL commands presented in Section 1.2. These two examples cover the basic cases of single group and multiple group estimation. These examples illustrate all the commands needed to perform straightforward single group or multiple group estimation using ICL.

The remaining examples illustrate the use of ICL in some more complex cases using commands presented in Section 1.3 and some other Tcl commands. To fully understand the examples other than the first two some basic knowledge of Tcl beyond what is presented in this manual is needed. Section 1.4 provides some references for learning more about Tcl.

2.1 Single Group Estimation

This example shows how to use ICL for single group estimation. The data set used for this example is introduced in Chapter 4 of Kolen and Brennan (1995). These data consists of responses of two groups of examinees sampled from different populations to two 36 item test forms (Form X and Form Y). In this example only the Form Y data are used. There were 1638 examinees who took Form Y. In this example only item parameters are estimated. The discrete latent variable distribution is not estimated. For each EM iteration the discrete latent variable distribution is fixed at the initial value of an approximate standard normal distribution.

The command file for this example is named '`mondaty.tcl`'. Each command is preceded by a comment that provides information about the command. Running ICL using this command file produces a log file named '`mondaty.log`', which contains a listing of the command file as part of the output. The log file '`mondaty.log`' is given below. The top of the log file contains the version number of the program and the date the program was run followed by a listing of the command file. Each command in the command file is preceded by a comment that provides information about that command.

```
IRT Command Language (ICL)
Version 0.020301

Feb 28, 2002 06:29

Command file mondaty.tcl
------------------------------------------------------------------------
#
# mondaty.tcl
#
# Estimate item parameters for Form Y items
# using data for examinees who took Form Y.
# Example data from Chapters 4 and 6 Kolen and
# Brennan (1995)

# Write output to log file mondaty.out
```

```
output -log_file mondaty.log

# 36 items to be modeled
allocate_items_dist 36

# Read examinee item responses from file mondaty.dat.
# Each record contains the responses to
# 36 items for an examinee in columns 1-36.
read_examinees mondaty.dat 36i1

# Compute starting values for item parameter estimates
starting_values_dichotomous

# Perform EM iterations for computing item parameter estimates.
# Maximum of 50 EM iterations.
EM_steps -max_iter 50

# Print item parameter estimates and discrete latent
# variable distribution.
print -item_param -latent_dist

# end of run
release_items_dist
-------------------------------------------------------------------------

Number of items: 36
Number of latent variable points: 40
Number of examinee groups: 1

Default prior for a-parameters:
 beta  a: 1.750   b: 3.000   lower limit: 0.000    upper limit: 3.000
Default prior for b-parameters:
 beta  a: 1.010   b: 1.010   lower limit: -6.000   upper limit: 6.000
Default prior for c-parameters:
 beta  a: 3.500   b: 4.000   lower limit: 0.000    upper limit: 0.500

Read 1638 examinee records from file mondaty.dat

EM iterations
(iteration: parameter criterion, marginal posterior mode)
    1: 0.323389  -33397.5098
    2: 0.110038  -33389.3586
    3: 0.062705  -33387.3062
    4: 0.035834  -33386.6304
    5: 0.020535  -33386.3208
    6: 0.011804  -33386.1332
    7: 0.006828  -33386.0002
    8: 0.003986  -33385.8991
    9: 0.003053  -33385.8198
```

```
   10: 0.002604  -33385.7570
   11: 0.002277  -33385.7068
   12: 0.002034  -33385.6665
   13: 0.001827  -33385.6342
   14: 0.001643  -33385.6082
   15: 0.001479  -33385.5873
   16: 0.001331  -33385.5704
   17: 0.001198  -33385.5568
   18: 0.001077  -33385.5458
   19: 0.000969  -33385.5370

Item Parameter Estimates
(a, b, c for 3PL, 2PL, 1PL; a, b1, b2, ... for GPCM, PCM)
1   0.906423    -1.347660   0.208407
2   0.473684    -0.371264   0.121268
3   0.454707    -1.179623   0.205979
4   0.572846    -0.785926   0.176588
5   0.677359    -1.215177   0.307152
6   0.616771    -1.070789   0.278926
7   1.345043    0.120214    0.317249
8   0.492794    0.422721    0.321266
9   0.625573    -0.638649   0.141979
10  0.908488    -0.353344   0.173849
11  1.071849    -0.778552   0.107534
12  0.626706    -0.335110   0.087110
13  0.897964    0.039772    0.155317
14  0.764979    -0.277579   0.099294
15  1.149896    0.562483    0.324354
16  0.882140    0.570445    0.241832
17  0.656918    0.934675    0.253957
18  0.848905    -0.105520   0.068215
19  1.059801    -0.127841   0.192292
20  0.878602    0.397564    0.157621
21  0.361482    2.504522    0.215255
22  0.823831    -0.090087   0.142616
23  1.384274    0.519738    0.257000
24  1.516551    0.579964    0.235205
25  1.310816    0.620187    0.268282
26  1.088591    0.393237    0.193605
27  1.196990    0.910755    0.177657
28  1.343262    1.084245    0.237970
29  1.076227    0.682977    0.117473
30  0.671011    1.803170    0.080542
31  1.190248    1.323631    0.190957
32  1.096862    0.886164    0.107298
33  1.224951    1.029468    0.065284
34  1.344530    1.724109    0.102385
35  1.284450    1.908184    0.082827
36  1.043982    1.987125    0.129066
```

```
Discrete Latent Variable Distributions
(point, probability for group 1, 2, etc)
-4.000000   2.745344e-05
-3.794872   6.106663e-05
-3.589744   1.302378e-04
-3.384615   2.663153e-04
-3.179487   5.221329e-04
-2.974359   9.815038e-04
-2.769231   1.769004e-03
-2.564103   3.056973e-03
-2.358974   5.065011e-03
-2.153846   8.046278e-03
-1.948718   1.225563e-02
-1.743590   1.789790e-02
-1.538462   2.506079e-02
-1.333333   3.364442e-02
-1.128205   4.330694e-02
-0.923077   5.344755e-02
-0.717949   6.324468e-02
-0.512821   7.175402e-02
-0.307692   7.805385e-02
-0.102564   8.140824e-02
0.102564    8.140824e-02
0.307692    7.805385e-02
0.512821    7.175402e-02
0.717949    6.324468e-02
0.923077    5.344755e-02
1.128205    4.330694e-02
1.333333    3.364442e-02
1.538462    2.506079e-02
1.743590    1.789790e-02
1.948718    1.225563e-02
2.153846    8.046278e-03
2.358974    5.065011e-03
2.564103    3.056973e-03
2.769231    1.769004e-03
2.974359    9.815038e-04
3.179487    5.221329e-04
3.384615    2.663153e-04
3.589744    1.302378e-04
3.794872    6.106663e-05
4.000000    2.745344e-05
```

Following the listing of the command file the number of items, groups and discrete latent variable points that were specified are printed along with the default prior distributions that will used for each item parameter. All output up to this point is the result of the `allocate_items_dist` command. The `read_examinees` command results in the number of examinee records read from the input file being printed. The `EM_steps` results in information from each EM iteration being printed. For each iteration the iteration number is printed followed

by the criterion used to determine convergence. This is the largest relative difference of a parameter estimate from the previous and current iteration relative to the parameter estimate for the current iteration across all parameter estimates for all items. This value generally decreases at each iteration, although it is possible it could increase. The default convergence criterion of 0.001 is reached at iteration 19. Also reported for each iteration is the value of the marginal loglikelihood of the data over the latent variable distribution, or in the case of Bayes modal estimation the value of the marginal log posterior distribution at the mode. This is the value being maximized by the EM algorithm. It will increase at each iteration.

The item parameter estimates are then printed. This output is the result of the `-item_param` option of the `print` command. The item parameter estimates on each line are separated by a tab character. These values correspond to those in Table 6.5 of Kolen and Brennan (1995) which were computed using the same data.

At the end of the log file are the points and probabilities of the discrete latent variable distribution. This output is the result of the `-latent_dist` option of the `print` command. The points and probabilities are separated by a tab character. The distribution was not estimated in this example, so the points and probabilities printed represent the initial values which approximate a standard normal distribution on equally space points from -4 to 4.

2.2 Multiple Group Estimation

This example shows how to use ICL for multiple group estimation using data from a common-item nonequivalent groups equating design. The data set introduced in Chapter 4 of Kolen and Brennan is used for this example. This data set consists of two groups of examinees sampled from different populations taking two forms of a 36 item test (Form X and Form Y). There are 12 items in common to Form X and Form Y. Consequently, this is an example of a common–item nonequivalent groups design that is often used in equating studies. The Form Y data were used in the previous example. This example uses both the Form Y and Form X data.

Two approaches to obtaining item parameter estimates for data from a common–item nonequivalent groups design are separate and concurrent estimation. In separate estimate the item parameters for the two forms are separately estimated, and the two sets of item parameter estimates for the common items are used to calculate a scale transformation to put the Form X parameters on the same scale as the Form Y parameters. This is the approached used in Chapter 6 of Kolen and Brennan (1995) to obtain item parameter estimates for the two Forms. An alternative is concurrent estimation in which all data are used to estimate item parameters for all items simultaneously. In order to appropriately perform concurrent estimation separate latent variable distributions must be assumed for the groups that took the two forms, and multiple group estimation must be used (Bock & Zimowski, 1996). The multiple group estimation procedure involves estimating not only the item parameters but the latent variable distributions for the two groups.

The log file below (‘`mondat2.log`’) shows the results of running ICL on these data with default options using a command file ‘`mondat2.tcl`’. The discrete latent variable distribution of the group that took Form Y (group 1) is fixed at a discrete approximation to a standard normal distribution. The probabilities of the latent distribution for the group that took Form X (group 2) are estimated along with the item parameters. The command

file used for this run ('mondat2.tcl') is listed at the top of the log file. There is a comment before each command providing information about the command.

```
IRT Command Language (ICL)
Version 0.020301

Feb 28, 2002 06:30

Command file mondat2.tcl
------------------------------------------------------------------------
#
# mondat2.tcl
#
# Estimate item parameters for Form Y and Form X
# items fixing the latent variable distribution of
# the group that took Form X at a discrete approximation
# to a standard normal distribution and estimating
# the latent variable distribution of the group
# that took Form Y.
# Example data from Chapters 4 and 6 Kolen and
# Brennan (1995)

# Write output to log file mondat2.log
output -log_file mondat2.log

# 24 unique items on each of two forms and
# 12 common items for a total of 60
# items. Two groups specified
# for multiple group estimation
allocate_items_dist 60 -num_groups 2

# Read examinee item responses from file mondat.dat.
# Each record contains the responses to
# 60 items for an examinee in columns 2-61.
# The first 24 items are the unique items on
# Form Y, the second 12 items are common items,
# and the last 24 items are unique items on
# Form X. An integer in column 1 gives
# the examinee group: 1 for examinees
# who took Form Y, and 2 for examinees
# who took Form X
read_examinees mondat.dat {@2 60i1} {i1}

# Compute starting values for item parameter estimates
starting_values_dichotomous

# Perform EM iterations for computing item parameter estimates
# and probabilities of latent variable distribution for
# group 2.
EM_steps
```

```
# Print item parameter estimates, discrete latent
# variable distributions, and mean and s.d. of
# latent variable distributions.
print -item_param -latent_dist -latent_dist_moments

# end of run
release_items_dist
-------------------------------------------------------------------------

Number of items: 60
Number of latent variable points: 40
Number of examinee groups: 2

Default prior for a-parameters:
 beta  a: 1.750   b: 3.000   lower limit: 0.000    upper limit: 3.000
Default prior for b-parameters:
 beta  a: 1.010   b: 1.010   lower limit: -6.000   upper limit: 6.000
Default prior for c-parameters:
 beta  a: 3.500   b: 4.000   lower limit: 0.000    upper limit: 0.500

Read 3293 examinee records from file mondat.dat

EM iterations
(iteration: parameter criterion, dist criterion,
 marginal posterior mode)
    1: 0.325684  0.792375  -67324.1530
    2: 0.118463  0.032091  -67309.4932
    3: 0.067596  0.043489  -67304.2629
    4: 0.038600  0.040133  -67301.4630
    5: 0.022204  0.036295  -67299.6131
```

To save space results for iterations 6–47 have not been included

```
   48: 0.001029  0.004610  -67289.5678
   49: 0.001006  0.004627  -67289.5441
   50: 0.000983  0.004640  -67289.5211

Item Parameter Estimates
(a, b, c for 3PL, 2PL, 1PL; a, b1, b2, ... for GPCM, PCM)
1   0.902049    -1.346818   0.208466
2   0.471417    -0.366077   0.121348
3   0.572317    -0.777315   0.177926
4   0.673001    -1.209600   0.309646
5   1.338834    0.128340    0.317040
6   0.500505    0.456035    0.327627
7   0.907474    -0.344096   0.175273
8   1.069827    -0.778211   0.104336
9   0.897436    0.050535    0.156708
```

```
10  0.761472    -0.270436   0.099868
11  0.883390    0.581825    0.242873
12  0.654371    0.943493    0.253757
13  1.063666    -0.113581   0.195530
14  0.879993    0.408817    0.158874
15  0.823225    -0.077643   0.144995
16  1.394771    0.533100    0.258673
17  1.317835    0.633579    0.269809
18  1.093679    0.405745    0.195304
19  1.350875    1.093911    0.238635
20  1.084287    0.693878    0.118892
21  1.206940    1.330781    0.192186
22  1.110434    0.898678    0.109714
23  1.379741    1.718422    0.103069
24  1.295562    1.906052    0.082852
25  0.451529    -1.017349   0.264057
26  0.607908    -1.009752   0.316995
27  0.651341    -0.581375   0.128070
28  0.611004    -0.380763   0.102269
29  1.158889    0.568537    0.318312
30  0.848076    -0.205679   0.040316
31  0.271444    2.201852    0.130057
32  1.587493    0.606562    0.247714
33  1.568531    0.962962    0.197606
34  0.635111    1.844574    0.073585
35  1.279586    1.072499    0.068289
36  1.106592    1.882636    0.114267
37  0.471825    -2.429900   0.228530
38  0.750045    -0.934507   0.133897
39  1.491997    0.073402    0.288756
40  1.046136    -0.458091   0.324941
41  0.946046    0.117222    0.353344
42  1.244699    -0.439677   0.279729
43  0.962483    0.618718    0.374043
44  0.995185    0.325047    0.251834
45  1.294408    0.026028    0.268403
46  1.091252    0.393986    0.169489
47  0.976533    0.239023    0.273558
48  0.933876    0.146591    0.250714
49  0.637557    -0.012976   0.133525
50  1.138165    0.551257    0.217272
51  0.909770    0.668293    0.249267
52  1.122261    0.165681    0.068003
53  0.486087    1.018214    0.146719
54  0.917912    0.672956    0.092374
55  1.460418    1.080933    0.161300
56  0.988716    1.121005    0.145468
57  1.215465    1.488096    0.244659
58  0.867742    1.291361    0.087738
```

```
59  0.391847    3.706447    0.118318
60  0.828203    2.887799    0.107679

Discrete Latent Variable Distributions
(point, probability for group 1, 2, etc)
-4.000000   2.745344e-05    7.530285e-04
-3.794872   6.106663e-05    1.486021e-03
-3.589744   1.302378e-04    2.716769e-03
-3.384615   2.663153e-04    4.572037e-03
-3.179487   5.221329e-04    7.043583e-03
-2.974359   9.815038e-04    9.904101e-03
-2.769231   1.769004e-03    1.273877e-02
-2.564103   3.056973e-03    1.515239e-02
-2.358974   5.065011e-03    1.706718e-02
-2.153846   8.046278e-03    1.892539e-02
-1.948718   1.225563e-02    2.172727e-02
-1.743590   1.789790e-02    2.702136e-02
-1.538462   2.506079e-02    3.665557e-02
-1.333333   3.364442e-02    5.068457e-02
-1.128205   4.330694e-02    6.235741e-02
-0.923077   5.344755e-02    6.191401e-02
-0.717949   6.324468e-02    5.583147e-02
-0.512821   7.175402e-02    6.326347e-02
-0.307692   7.805385e-02    9.362840e-02
-0.102564   8.140824e-02    1.008225e-01
0.102564    8.140824e-02    6.235792e-02
0.307692    7.805385e-02    4.776633e-02
0.512821    7.175402e-02    5.470069e-02
0.717949    6.324468e-02    4.940951e-02
0.923077    5.344755e-02    3.896836e-02
1.128205    4.330694e-02    3.162236e-02
1.333333    3.364442e-02    1.748825e-02
1.538462    2.506079e-02    7.328682e-03
1.743590    1.789790e-02    4.497094e-03
1.948718    1.225563e-02    5.394483e-03
2.153846    8.046278e-03    7.521996e-03
2.358974    5.065011e-03    5.789346e-03
2.564103    3.056973e-03    1.955403e-03
2.769231    1.769004e-03    4.090775e-04
2.974359    9.815038e-04    9.151466e-05
3.179487    5.221329e-04    3.335142e-05
3.384615    2.663153e-04    2.377408e-05
3.589744    1.302378e-04    3.272659e-05
3.794872    6.106663e-05    7.717242e-05
4.000000    2.745344e-05    2.666780e-04

Moments of Latent Variable Distributions (group 1, 2, etc)
Mean:   0.000000    -0.446339
s.d.:   0.999646    1.131774
```

The output at the top of the log file that is produced by the `allocate_items_dist` and `read_examinees` commands is the same as that described in the first example. The information for each EM iteration produced by the `EM_steps` command includes the largest relative item parameter difference and the marginal posterior mode that were described in the first example as the first and third number after the iteration number. The second number after the iteration number was not present in the one group example. This number is the maximum relative difference in the estimated probabilities of discrete latent variable distribution for group 2 from the previous to current iteration (group 1 probabilities are not included because they were not estimated). This is analogous to the maximum relative difference in the estimated item parameters (the first number after the iteration number). Only the maximum relative difference in the item parameter estimates is used to determine whether the convergence criterion is met. The maximum relative difference in the latent distribution probabilities is presented, but is not used in determining whether the convergence criterion is met. This could be changed by modifying the source code of the `EM_steps` function in the file '`icl.tcl`'. The convergence criterion is met after 50 iterations.

The `-item_param` option of the `print` command results in the item parameter estimates being printed in the log file. The first 24 item parameter estimates correspond to unique items on Form Y, the next 12 item parameter estimates correspond to common item on Form X and Y, and the last 24 item parameter estimates correspond to unique items on Form X. These item parameter estimates can be compared to the item parameter estimates reported in Table 6.8 of Kolen and Brennan (1995). Kolen and Brennan (1995) estimated the item parameters using separate estimation. Thus, there are two sets of item parameter estimates for the common items in Table 6.8 of Kolen and Brennan (1995), and the item parameter estimates for the items on Form X have been transformed to be on the scale of the item parameter estimates for Form Y using the two sets of common item parameter estimates. In the concurrent run presented in this section parameter estimates for all items are obtained in one run, so there is only one set of item parameter estimates for the common items, and there is no scaling step needed to put the item parameter estimates for Form X and Y on the same scale.

The `-latent_dist` option of the `print` command results in the latent variable distributions for all groups being printed in the log file. The points, which are common to all groups, are printed first followed by the associated probabilities for groups 1 (Form Y) and 2 (Form X). The `-latent_dist_moments` option of the `print` command results in the mean and standard deviation of the latent variable distributions for both groups being printed. In this case the latent variable distribution for group 1 was fixed to be a discrete approximation to a standard normal distribution across all iterations. Thus, the mean and standard deviation of the latent variable distribution for group 1 are approximately 0 and 1. The latent variable distribution for group 2 was estimated and its mean is less than 0 and standard deviation is greater than 1. Thus, group 2, who took Form X is a lower performing group than group 1, who took Form Y.

It may be more reasonable to estimate the latent variable distribution for Group 1 rather than fixing it as a discrete approximation of a standard normal distribution. The following log file ('`mondat3.log`') shows output from running the command file '`mondat3.tcl`' in which the latent variable distributions for both groups are estimated. The only change from the command file '`mondat2.tcl`', besides the change of the log file name, is that the '`EM_steps`' command is changed to '`EM_steps -estim_dist -scale_points -max_iter`

200'. The `-estim_dist` option indicates the latent variable distribution will be estimated for Group 1. The `-scale_points` option results in the points of the discrete latent variable distribution being linearly transformed so the mean and standard deviation of the Form X distribution are 0 and 1 after each M–step. The item parameters are also transformed using the same scale transformation. In the '`mondat3.tcl`' run the points of the latent variable distribution were not changed during the EM iterations.

```
IRT Command Language (ICL)
Version 0.020301

Feb 28, 2002 06:30

Command file mondat3.tcl
------------------------------------------------------------------------
#
# mondat3.tcl
#
# Example data from Chapters 4 and 6 Kolen and
# Brennan (1995)
# Estimate item parameters for Form Y and Form X
# items while at the same time estimating
# the latent variable distribution of the groups
# that took Form X and Form Y. After each M-step
# the points of the discrete latent variable distribution
# are linearly transformed so the mean and standard
# deviation of the Form X distribution are 0 and 1.
# The item parameters are also tranformed using
# the same scale transformation.

# Write output to log file mondat3.log
output -log_file mondat3.log

# 24 unique items on each of two forms and
# 12 common items for a total of 60
# items. Two groups specified
# for multiple group estimation
allocate_items_dist 60 -num_groups 2

# Read examinee item responses from file mondat.dat.
# Each record contains the responses to
# 60 items for an examinee in columns 2-61.
# The first 24 items are the unique items on
# Form Y, the second 12 items are common items,
# and the last 24 items are unique items on
# Form X. An integer in column 1 gives
# the examinee group: 1 for examinees
# who took Form Y, and 2 for examinees
# who took Form X
read_examinees mondat.dat {@2 60i1} {i1}
```

```
# Compute starting values for item parameter estimates
starting_values_dichotomous

# Perform EM iterations for computing item parameter estimates
# and probabilities of latent variable distributions for
# groups 1 and 2. Scale points of latent variable distribution
# after each M-step so the mean and s.d. in group 1 are
# 0 and 1. Allow a maximum of 200 EM iterations.
EM_steps -estim_dist -scale_points -max_iter 200

# Print item parameter estimates, discrete latent
# variable distributions, and mean and s.d. of
# latent variable distributions.
print -item_param -latent_dist -latent_dist_moments

# end of run
release_items_dist
------------------------------------------------------------------------

Number of items: 60
Number of latent variable points: 40
Number of examinee groups: 2

Default prior for a-parameters:
 beta  a: 1.750   b: 3.000   lower limit: 0.000    upper limit: 3.000
Default prior for b-parameters:
 beta  a: 1.010   b: 1.010   lower limit: -6.000   upper limit: 6.000
Default prior for c-parameters:
 beta  a: 3.500   b: 4.000   lower limit: 0.000    upper limit: 0.500

Read 3293 examinee records from file mondat.dat

EM iterations
(iteration: parameter criterion, dist criterion,
 marginal posterior mode)
    1: 0.325684  0.792375  -67322.8364
    2: 0.106845  0.046843  -67309.3143
    3: 0.071285  0.044809  -67301.8025
    4: 0.053653  0.041071  -67296.5794
    5: 0.042593  0.037904  -67292.7518
```

To save space results for iterations 6–174 have not been included

```
  175: 0.001018  0.007281  -67274.2664
  176: 0.001012  0.007297  -67274.2602
  177: 0.001007  0.007313  -67274.2539
  178: 0.001002  0.007328  -67274.2477
  179: 0.000997  0.007342  -67274.2416
```

```
Item Parameter Estimates
(a, b, c for 3PL, 2PL, 1PL; a, b1, b2, ... for GPCM, PCM)
1   0.856376    -1.325967   0.246916
2   0.487127    -0.308145   0.132614
3   0.606201    -0.589769   0.234970
4   0.657235    -1.129413   0.346636
5   1.521015    0.212157    0.330804
6   0.613257    0.633843    0.374111
7   0.974694    -0.222569   0.208512
8   1.089917    -0.670979   0.148203
9   0.999468    0.148514    0.182302
10  0.816117    -0.154468   0.133906
11  1.047074    0.629296    0.262510
12  0.799468    0.956607    0.277609
13  1.203771    0.013975    0.229156
14  1.013782    0.475916    0.181176
15  0.942895    0.075590    0.192940
16  1.637067    0.570556    0.268125
17  1.526627    0.655456    0.277041
18  1.240873    0.458696    0.208509
19  1.612816    1.041685    0.241950
20  1.292494    0.712256    0.133058
21  1.525688    1.231749    0.198312
22  1.310676    0.880316    0.117730
23  1.827061    1.528146    0.106721
24  1.715056    1.664267    0.085240
25  0.473728    -0.874199   0.290485
26  0.635767    -0.858958   0.350641
27  0.692582    -0.455842   0.159864
28  0.646188    -0.287047   0.122906
29  1.323328    0.593159    0.325955
30  0.893672    -0.139091   0.050774
31  0.292861    2.142789    0.138894
32  1.787622    0.620910    0.250937
33  1.816469    0.929786    0.199521
34  0.749538    1.695184    0.081839
35  1.500056    1.025111    0.072655
36  1.379407    1.683228    0.117269
37  0.473628    -2.360666   0.238489
38  0.778453    -0.846108   0.146216
39  1.611046    0.119856    0.291455
40  1.137375    -0.356627   0.340054
41  1.047684    0.180112    0.363714
42  1.349114    -0.345247   0.293295
43  1.063927    0.628013    0.378435
44  1.083287    0.355466    0.256584
45  1.409989    0.080253    0.273988
46  1.193209    0.418889    0.174239
47  1.075626    0.281456    0.280996
```

```
48  1.033795    0.200959    0.260957
49  0.676102    0.043045    0.142039
50  1.259503    0.562194    0.221469
51  1.000726    0.668022    0.253145
52  1.205289    0.203870    0.071685
53  0.531999    1.009371    0.155838
54  1.012227    0.674949    0.097987
55  1.611799    1.039383    0.161956
56  1.089525    1.076782    0.147321
57  1.364772    1.398204    0.245398
58  0.969074    1.228746    0.091101
59  0.426559    3.453718    0.119237
60  0.948066    2.613977    0.108208

Discrete Latent Variable Distributions
(point, probability for group 1, 2, etc)
-3.364293   1.803924e-04    3.836721e-03
-3.187785   3.439385e-04    6.775798e-03
-3.011276   6.478347e-04    1.061181e-02
-2.834768   1.220672e-03    1.458895e-02
-2.658260   2.319404e-03    1.751716e-02
-2.481751   4.432547e-03    1.845248e-02
-2.305243   8.344814e-03    1.739466e-02
-2.128735   1.479886e-02    1.529827e-02
-1.952227   2.310089e-02    1.343777e-02
-1.775718   2.940365e-02    1.292471e-02
-1.599210   2.894974e-02    1.499117e-02
-1.422702   2.243898e-02    2.214244e-02
-1.246193   1.559435e-02    3.923349e-02
-1.069685   1.238178e-02    6.626064e-02
-0.893177   1.455648e-02    7.615204e-02
-0.716669   2.704838e-02    5.109651e-02
-0.540160   5.800437e-02    3.077938e-02
-0.363652   8.605510e-02    3.968360e-02
-0.187144   7.997061e-02    1.073430e-01
-0.010635   7.311614e-02    1.177310e-01
0.165873    6.984679e-02    3.405008e-02
0.342381    5.393648e-02    2.858519e-02
0.518889    6.512597e-02    7.194925e-02
0.695398    9.282164e-02    4.820522e-02
0.871906    6.094744e-02    2.622353e-02
1.048414    4.582051e-02    3.776838e-02
1.224923    3.137408e-02    2.441026e-02
1.401431    2.009088e-02    5.311097e-03
1.577939    3.411260e-02    2.108668e-03
1.754447    2.047829e-02    4.140958e-03
1.930956    4.673844e-04    1.208310e-02
2.107464    2.329692e-06    7.573391e-03
2.283972    5.642149e-08    6.068035e-04
```

```
2.460481    2.188147e-08    1.684229e-05
2.636989    7.299836e-08    6.483347e-07
2.813497    6.986436e-07    9.188993e-08
2.990006    8.306627e-06    7.331026e-08
3.166514    7.679156e-05    3.431198e-07
3.343022    4.471454e-04    7.985940e-06
3.519530    1.533559e-03    7.065380e-04

Moments of Latent Variable Distributions (group 1, 2, etc)
Mean:   0.000000    -0.414384
s.d.:   1.000000    1.111024
```

The format of the output is the same as for '`mondat2.log`'. In this case it took 179 iterations for the convergence criterion to be met. Note that the points of the latent variable distribution are different from those given in '`mondat2.log`'. The points have been adjusted at each M–step after new probabilities for Group 1 have been estimated so that the mean and standard deviation of the distribution for Group 1 are 0 and 1.

Lewis (1985) suggested that fixing the points of a discrete latent variable distribution fixes the scale of the latent variable and it is not necessary to adjust the points after each M–step so the mean and standard deviation for one of the groups was fixed at some value. The following log file '`mondat4.log`' shows the output from running the command file '`mondat4.tcl`' in which the `-scale_points` option is not used with the `EM_steps` command. When this option is not used the distributions for both groups are estimated, but the points are not adjusted after each M–step.

```
IRT Command Language (ICL)
Version 0.020301

Feb 28, 2002 06:31

Command file mondat4.tcl
------------------------------------------------------------------------
#
# mondat4.tcl
#
# Example data from Chapters 4 and 6 Kolen and
# Brennan (1995)
# Estimate item parameters for Form Y and Form X
# items while at the same time estimating
# the latent variable distribution of the groups
# that took Form X and Form Y.

# Write output to log file mondat4.log
output -log_file mondat4.log

# 24 unique items on each of two forms and
# 12 common items for a total of 60
# items. Two groups specified
# for multiple group estimation
allocate_items_dist 60 -num_groups 2
```

```
# Read examinee item responses from file mondat.dat.
# Each record contains the responses to
# 60 items for an examinee in columns 2-61.
# The first 24 items are the unique items on
# Form Y, the second 12 items are common items,
# and the last 24 items are unique items on
# Form X. An integer in column 1 gives
# the examinee group: 1 for examinees
# who took Form Y, and 2 for examinees
# who took Form X
read_examinees mondat.dat {@2 60i1} {i1}

# Compute starting values for item parameter estimates
starting_values_dichotomous

# Perform EM iterations for computing item parameter estimates
# and probabilities of latent variable distributions for
# groups 1 and 2. Points of the latent variable distribution
# will not be adjusted after each M-step so the mean and
# standard deviation of the distribution in Group 1 are
# zero and one. Allow a maximum of 200 EM iterations.
EM_steps -estim_dist -max_iter 200

# Print item parameter estimates and discrete latent
# variable distributions.
print -item_param -latent_dist -latent_dist_moments

# end of run
release_items_dist
-------------------------------------------------------------------------

Number of items: 60
Number of latent variable points: 40
Number of examinee groups: 2

Default prior for a-parameters:
 beta  a: 1.750   b: 3.000   lower limit: 0.000    upper limit: 3.000
Default prior for b-parameters:
 beta  a: 1.010   b: 1.010   lower limit: -6.000   upper limit: 6.000
Default prior for c-parameters:
 beta  a: 3.500   b: 4.000   lower limit: 0.000    upper limit: 0.500

Read 3293 examinee records from file mondat.dat

EM iterations
(iteration: parameter criterion, dist criterion,
 marginal posterior mode)
    1: 0.325684  0.792375  -67323.1840
```

```
   2: 0.105335  0.046843  -67309.8870
   3: 0.070159  0.044621  -67302.4966
   4: 0.052663  0.040925  -67297.3592
   5: 0.041683  0.037754  -67293.5908
```

To save space results for iterations 6–101 have not been included

```
 102: 0.001043  0.004270  -67272.7131
 103: 0.001032  0.004273  -67272.6868
 104: 0.001021  0.004276  -67272.6608
 105: 0.001010  0.004279  -67272.6351
 106: 0.001000  0.004282  -67272.6095

Item Parameter Estimates
(a, b, c for 3PL, 2PL, 1PL; a, b1, b2, ... for GPCM, PCM)
1   0.792187    -1.488947   0.250090
2   0.449416    -0.396068   0.131935
3   0.560405    -0.695451   0.235830
4   0.611584    -1.259582   0.352839
5   1.413598    0.172905    0.331471
6   0.570020    0.631022    0.375348
7   0.906584    -0.292693   0.211084
8   1.005828    -0.787173   0.147190
9   0.927100    0.104662    0.183414
10  0.754307    -0.226343   0.133831
11  0.973068    0.622823    0.263161
12  0.742809    0.976510    0.278349
13  1.115883    -0.042044   0.229698
14  0.941303    0.457787    0.182049
15  0.873259    0.024504    0.193532
16  1.524570    0.559444    0.268728
17  1.426198    0.651852    0.278065
18  1.153788    0.438952    0.209293
19  1.505390    1.066314    0.242397
20  1.200714    0.711681    0.133490
21  1.421214    1.270939    0.198612
22  1.220486    0.893008    0.118370
23  1.718087    1.587893    0.107191
24  1.604955    1.734869    0.085500
25  0.437941    -1.000497   0.292070
26  0.587873    -0.982818   0.352770
27  0.641365    -0.547027   0.162099
28  0.596239    -0.371596   0.122613
29  1.227959    0.582892    0.326236
30  0.826785    -0.208243   0.051555
31  0.269964    2.259852    0.138474
32  1.665245    0.613380    0.251479
33  1.690027    0.945642    0.199753
34  0.696628    1.771485    0.082339
```

```
35  1.393049    1.048473    0.072851
36  1.277264    1.759533    0.117280
37  0.437292    -2.617455   0.238206
38  0.718595    -0.974767   0.147245
39  1.491584    0.070535    0.291452
40  1.054100    -0.440028   0.342014
41  0.974644    0.138248    0.364723
42  1.257864    -0.424564   0.296475
43  0.989702    0.620969    0.379072
44  1.005329    0.326430    0.257210
45  1.310624    0.030317    0.275020
46  1.109688    0.395445    0.175172
47  0.996796    0.246629    0.281574
48  0.956088    0.158842    0.261263
49  0.627481    -0.006744   0.144300
50  1.171036    0.550009    0.222195
51  0.930581    0.665177    0.254119
52  1.116668    0.162424    0.072275
53  0.493380    1.032689    0.156504
54  0.939198    0.671059    0.098524
55  1.500060    1.063250    0.162197
56  1.008182    1.104636    0.147332
57  1.266904    1.451279    0.245540
58  0.899117    1.268914    0.091459
59  0.396571    3.664500    0.119563
60  0.879832    2.765618    0.108336

Discrete Latent Variable Distributions
(point, probability for group 1, 2, etc)
-4.000000   8.573090e-05    1.759295e-03
-3.794872   1.769717e-04    3.341082e-03
-3.589744   3.567996e-04    5.769281e-03
-3.384615   7.091941e-04    8.972447e-03
-3.179487   1.400644e-03    1.247364e-02
-2.974359   2.754413e-03    1.545960e-02
-2.769231   5.341754e-03    1.718867e-02
-2.564103   9.935387e-03    1.750107e-02
-2.358974   1.686753e-02    1.699215e-02
-2.153846   2.448576e-02    1.675436e-02
-1.948718   2.861434e-02    1.813852e-02
-1.743590   2.644297e-02    2.302729e-02
-1.538462   2.069402e-02    3.429648e-02
-1.333333   1.637503e-02    5.299516e-02
-1.128205   1.663285e-02    6.704495e-02
-0.923077   2.499488e-02    5.840790e-02
-0.717949   4.788184e-02    4.305661e-02
-0.512821   7.852908e-02    4.909310e-02
-0.307692   8.836300e-02    9.805626e-02
-0.102564   8.353821e-02    1.161418e-01
```

```
0.102564    7.381653e-02    5.094832e-02
0.307692    5.958968e-02    3.972880e-02
0.512821    7.315067e-02    6.565200e-02
0.717949    9.569996e-02    5.007554e-02
0.923077    6.267682e-02    3.443563e-02
1.128205    4.251842e-02    3.500934e-02
1.333333    3.074335e-02    1.713434e-02
1.538462    2.680885e-02    5.252996e-03
1.743590    3.137364e-02    3.602199e-03
1.948718    7.387146e-03    7.021542e-03
2.153846    1.680712e-04    1.042986e-02
2.358974    3.080710e-06    3.425411e-03
2.564103    3.670952e-07    2.742574e-04
2.769231    3.729715e-07    1.455538e-05
2.974359    1.430882e-06    1.393448e-06
3.179487    8.794689e-06    4.257355e-07
3.384615    5.057309e-05    4.874361e-07
3.589744    2.102855e-04    1.916602e-06
3.794872    5.768263e-04    2.104291e-05
4.000000    1.034723e-03    5.002338e-04

Moments of Latent Variable Distributions (group 1, 2, etc)
Mean:   -0.060974   -0.509101
s.d.:   1.087421    1.204460
```

In this case the EM iterations converged in less than 200 iterations. Note that the points of the latent distribution are the same as those in '`mondat2.log`' — they have not been changed during the EM iterations. Consequently, the mean and standard deviation for Group 1 differs from zero and one, as does the mean and standard deviation for Group 2.

The Group 1 latent variable distributions in '`mondat2.log`' and '`mondat4.log`' have the same points, and the latent variable distributions for Group 1 in '`mondat2.log`' and '`mondat3.log`' both have a mean and standard deviation of zero and one. Are the parameter estimates in '`mondat2.log`' and '`mondat4.log`' on the same scale (points the same), or are the parameter estimates for '`mondat2.log`' and '`mondat3.log`' on the same scale (mean and s.d. in Group 1 the same)? Plotting the parameter estimates shows that the parameter estimates from '`mondat2.log`' and '`mondat4.log`' appear to be on the same scale (the parameters are scattered about an identity line), whereas the parameter estimates from '`mondat2.log`' and '`mondat3.log`' do not appear to be on the same scale (the a–parameter and b–parameter estimates are scattered around a line that is not an identity line). A comparison of the parameter estimates from '`mondat3.log`' and '`mondat4.log`' shows that are very similar to one another but are on different scales (they fall very near to a straight line that is not the identity line). These results suggest that Lewis (1985) was correct and the scale is fixed by the fixing the points of the discrete latent variable distribution. This suggests a preference for not using the `-scale_points` option when the `-estim_dist` option is used with the `EM_steps` command.

2.3 EAP and MLE Theta Estimates for Examinees

In this example maximum likelihood (MLE) and Bayesian estimates of latent proficiency are computed for individual examinees. The Bayesian estimates are computed using the mean of the posterior latent variable distribution for each examinee. These Bayesian estimates are referred to as EAP (expected a posterior) estimates. The data used are the same as for the example in Section 2.1 — 1638 examinees taking Form Y of a 36 item test. This example reads in previously computed item parameter estimates, and item responses for all examinees. This information is used to compute MLE and EAP estimates of latent proficiency for all 1638 examinees. A number correct score is also computed and written for each examinee. The item parameter estimates needed in this example could be obtained by adding the following commands to the command file in Section 2.1 ('`mondaty.tcl`') before the `release_items_dist` command.

```
# Write parameter estimates with 8 digits after
# the decimal point
write_item_param mondaty.par -format %.8f
```

These commands write the item parameter estimates to a file named '`mondaty.par`'. The following command file (`mondaty_theta.tcl`) illustrates how to use the item parameter estimates in '`mondaty.par`' and item response data in '`mondaty.dat`' to compute posterior latent variable distributions for each examinee and write the mean of these posterior distributions, along with number correct score, to a file '`mondaty.theta`'.

```
#
# mondaty_theta.tcl
#
# Compute EAP and MLE latent variable estimates for
# each examinee who took Form Y reading using the
# previously computed item parameter estimates.
# Write EAP and MLE estimates and number correct score
# for each examinee to file 'mondaty.theta'.
# Example data from Chapters 4 and 6 Kolen and
# Brennan (1995)

# Supress written output from subsequent ICL commands
output -no_print

# 36 items to be modeled
allocate_items_dist 36

# Read examinee item responses from file mondaty.dat.
# Each record contains the responses to
# 36 items for an examinee in columns 1-36.
read_examinees mondaty.dat 36i1

# Read previously computed item parameter estimates
read_item_param mondaty.par

# Create E-step object needed to compute
# posterior latent variable distributions for
```

```
# examinees
set estep [new_estep]

# Use E-step object to compute posterior distribution
# for each examinee. The second argument being equal to 1
# indicates the posterior will be computed for each
# examinee. The third argument being equal to 1 indicates
# that the posterior for each examinee will be stored
# with the examinee to allow the examinee_posterior_mean
# command to be used for the examinee.
estep_compute $estep 1 1

# E-step object only needed for the estep_compute command,
# so can be deleted.
delete_estep $estep

# Open file to contain estimates
set eapfile [open mondaty.theta w]

# Write EAP and MLE estimates and number correct for each examinee on
# a separate line of the output file
for {set i 1} {$i <= [num_examinees]} {incr i} {

    # compute number correct
    set resp [examinee_responses $i]
    set numcorrect 0
    foreach r $resp {
        if {$r > 0} then {incr numcorrect}
    }

    # get examinee posterior mean (EAP estimate)
    set eap [examinee_posterior_mean $i]

    # get examinee MLE estimate
    set mle [examinee_theta_MLE $i -6.0 6.0]

    # Write EAP and MLE estimates and number correct. The first
    # argument to the format command indicates that the second and
    # third arguments to the format command will be written as
    # floating-point numbers with 6 digits after the decimal point and
    # that the fourth argument will be written as an integer, with
    # a tab character separating the numbers.
    puts $eapfile [format "%.6f\t%.6f\t%d" $eap $mle $numcorrect]
}

# close output file
close $eapfile

# end of run
```

```
release_items_dist
```

2.4 Pretest Item Calibration

This example illustrates estimation of pretest item statistics using simulated CAT data. The simulated data are taken from Ban, Hanson, Wang, Yi, and Harris (2001). In this simulated CAT there were 30 operational items administered to each examinee taken from an operational item pool of 520 items. Each examinee also received the same 10 pretest items. The goal is to use item parameter estimates for the operational items along with the item response data to estimate item parameters for the pretest items. This example shows how to use ICL to implement what Ban, et. al. (2001) refer to as the MMLE/Multiple–EM Cycle method of pretest item estimation.

Below the output produced by the command file '`pretest.tcl`' is given. This output includes a listing of the command file. This command file uses the command `ReadItemResp` which reformats data from the input file into a form that can be used by the ICL `add_examinee` command. This command is defined in the file '`pretest_dat.tcl`', which is presented in Section 2.9.

```
IRT Command Language (ICL)
Version 0.020301

Feb 28, 2002 06:43

Command file pretest.tcl
------------------------------------------------------------------------
#
# pretest.tcl
#
# Implement MMLE/Multiple-EM Cycle
# method of pretest item calibration and
# scaling described in:
#
# Ban, J., Hanson, B. A., Wang, T., Yi, Q., & Harris, D. J. (2001). A
# comparative study of on-line pretest item-calibration/scaling methods
# in computerized adaptive testing. Journal of Educational Measurement,
# 38(3), 191-212.

# number of operational items administered
# to each examinee
set adminOperItems 30

# total number of operational items used for CAT
set totOperItems 520

# number of pretest items administered to
# each examinee
set preItems 10

# write output to log file pretest.log
```

```
output -log_file pretest.log

# Total number of items is sum of number of operational
# and pretest items
allocate_items_dist [expr {$totOperItems+$preItems}]

#  Create list of item numbers for operational items
set operItemNo [seq 1 520]

# Set priors of all operational item parameters to none, since
# operational item parameters are not estimated. This allows
# any values of the operational item parameters to be read, even
# those for which the prior density using the default prior is zero
# (an error is reported if a parameter is read for which the
# prior density is zero).

# Set priors for a-parameters to none for operational items
items_set_prior 1 none {} $operItemNo

# Set priors for b-parameters to none for operational items
items_set_prior 2 none {} $operItemNo

# Set priors for c-parameters to none for operational items
items_set_prior 3 none {} $operItemNo

# Read examinee item responses using command ReadItemResp
# from file al40cf1.txt.
# The ReadItemResp command is defined
# in file pretest_dat.tcl.
source pretest_dat.tcl
ReadItemResp al40cf1.txt

# Read item parameters for operational items
# from file pool.par
read_item_param pool.par

# Create list of item numbers for pretest items
set preItemNo [seq 521 530]

# Compute starting values for pretest items.
# The first argument (1) indicates all
# items (including those answered correctly or
# incorrectly by all examinees) and all examinees
# (even those to get all items correct or all items
# incorrect) will be used to
# compute initial difficulties and proficiencies
# from which the starting values are computed.
# The second argument (0) indicates only
# the pretest items, rather than all items, are
```

```
# used to compute the initial examinee proficiencies.
item_3PL_starting_values 1 0 $preItemNo

# Create E-step object used to compute
# examinee posterior distributions based
# on operational items
set eoper [new_estep $operItemNo]

# Compute examinee posterior distributions based on
# only operation item responses, and store posteriors
# for each examinee. The second argument (1)
# indicates examinee posterior distributions
# are computed. The third argument (1)
# indicates that these posterior distributions
# are stored for each examinee. The fourth
# argument is an empty string which indicates
# that n's and r's are not updated for
# any items - the purpose of this command
# is to compute examinee posterior distributions,
# not n's and r's for any of the items.
estep_compute $eoper 1 1 {}

# E-step object no longer needed, so delete
delete_estep $eoper

# Create E-step object used in E-step
# for computing pretest item parameter
# estimates
set eall [new_estep]

# Compute E-step for pretest items using
# examinee posteriors computed with operational
# items. The second argument (0) indicates
# that examinee posterior distributions
# are not computed. Instead, posterior
# distributions previously computed and
# stored (in estep_compute command above)
# are used. The third argument (0) indicates
# that examinee posterior distributions are
# not stored for examinees (this is redundant,
# given the second argument is zero but must
# still be present).
estep_compute $eall 0 0 $preItemNo

# Loop over EM iterations.
# To implement the MMLE/One-EM Cycle method
# discussed in Ban, et. al. use just one
# iteration.
for {set iter 1} {$iter <= 100} {incr iter} {
```

```
   # M-step
   set maxreldiff [mstep_item_param -items $preItemNo]

   # E-step
   # Second argument to estep_compute (1) indicates
   # posterior distibutions will be computed
   # for examinees. These are used to update n's and
   # r's for items given by the item numbers
   # in the list that is the last argument.
   # The third argument to estep_compute (0) indicates
   # the examinee posterior distributions computed
   # will not be stored.
   set loglike [estep_compute $eall 1 0 $preItemNo]

   # Write iteration information to log file and to screen
   set iterinfo [format {%5d: %.6f %.4f} $iter $maxreldiff $loglike]
   puts_log $iterinfo
   puts $iterinfo

   # Quit EM iterations if convergence criterion is met
   if {$maxreldiff < 0.001} then break
}

# delete E-step object
delete_estep $eall

# Write parameter estimates for pretest items to
# log file. Uses global variable icl_logfileID
# defined in icl.tcl.
puts_log "\nItem parameter estimates for pretest items"
write_item_param_channel $icl_logfileID -format %.6f -items $preItemNo

# End of run
release_items_dist
------------------------------------------------------------------------

Number of items: 530
Number of latent variable points: 40
Number of examinee groups: 1

Default prior for a-parameters:
 beta  a: 1.750   b: 3.000   lower limit: 0.000    upper limit: 3.000
Default prior for b-parameters:
 beta  a: 1.010   b: 1.010   lower limit: -6.000   upper limit: 6.000
Default prior for c-parameters:
 beta  a: 3.500   b: 4.000   lower limit: 0.000    upper limit: 0.500

    1: 0.951852 -22816.2122
```

```
    2: 0.125647 -22807.7872
    3: 0.027547 -22807.5322
    4: 0.006083 -22807.5215
    5: 0.001334 -22807.5211
    6: 0.000290 -22807.5211

Item parameter estimates for pretest items
521 1.293347    0.843491    0.076837
522 1.097611    -0.909632   0.308161
523 0.825760    -0.259481   0.125992
524 1.360402    -0.123295   0.143632
525 1.336759    -0.572269   0.193322
526 1.318537    1.230668    0.170896
527 1.291393    0.598819    0.199266
528 1.223074    1.755223    0.074243
529 1.439228    -1.914188   0.218686
530 1.693259    0.267899    0.083983
```

Convergence was achieved in six EM iterations. Item parameter estimates for the 10 pretest items are printed at the end of the log file.

2.5 Multiple Group Estimation with Normal Distributions

This example is a continuation of the multiple group estimation example in Section 2.2. The same data used for the example in Section 2.2 are used in this example, although in this example the data are read in a different format. This example shows how to estimate just the mean and standard deviation of the latent variable distribution in Group 2 using fixed probabilities for the discrete latent variable distributions in Groups 1 and 2. This is accomplished by allowing separate points for the discrete latent variable distributions for Groups 1 and 2. For Group 1 the points are fixed throughout the EM iterations. For Group 2 the points are changed, although the weights are fixed, to allow the mean and standard deviation of the discrete latent variable distribution for Group 2 to be updated during the EM iterations.

This example uses the `read_examinees_missing` command to read the data from the original separate data sets for the two forms rather than the merged data set used in the example in Section 2.2. To allow different latent distribution points to be used for the two examinee groups the `-unique_points` option is used with the `allocate_items_dist` command.

The log file produced by the command file '`mondat5.tcl`' is given below. The initial discrete latent variable distribution is printed, as well as the discrete latent variable distribution after the EM iterations. At the end of the EM iterations the points of the latent variable distribution for Group 2 have changed, but the weights for Group 2 and the points and weights for Group 1 have not changed.

```
IRT Command Language (ICL)
Version 0.020301

Feb 28, 2002 06:32
```

```
Command file mondat5.tcl
-------------------------------------------------------------------------
#
# mondat5.tcl
#
# Example data from Chapters 4 and 6 Kolen and
# Brennan (1995)
# Estimate item parameters for Form Y and Form X
# items while at the same time estimating
# the mean and s.d. of the latent variable distribution
# of the group that took Form X (Group 2).

# Write output to log file mondat5.log
output -log_file mondat5.log

# 24 unique items on each of two forms and
# 12 common items for a total of 60
# items. Two groups specified
# for multiple group estimation.
# The -unique_points option allows different
# discrete latent distribution points to be
# used for the different group. This allows
# the mean and standard deviation of
# group 2 to be estimated.
allocate_items_dist 60 -num_groups 2 -unique_points

# Read examinee item responses for Form Y from
# file mondaty.dat and item responses for Form X
# from file mondatx.dat using read_examinees_missing
# command.
# Each record contains the responses to
# items in columns 1-36. The responses
# to the 12 common items on each form are in
# columns 3, 6, 9, ..., 36, and the responses
# to the 24 unique items on each form are in
# the other columns (1, 2, 4, 5, ..., 35).

# Item numbers are assigned such that the first 24
# items are the unique items on Form Y,
# the second 12 items are common items,
# and the last 24 items are unique items on
# Form X.

# Item numbers for Form Y in the order in which
# they are read from file mondaty.dat. Forms
# are not read from the input record since the examinees
# who take each form are read from separate files.
# In this case integers need to be used as indices for
# forms. The index of Form Y is 1.
```

```
set items(1) [list 1 2 25 3 4 26 5 6 27 7 8 28 \
    9 10 29 11 12 30 13 14 31 15 16 32 \
    17 18 33 19 20 34 21 22 35 23 24 36]

# Item numbers for Form X in the order in which
# they are read from file mondatx.dat. Forms
# are not read from the input record since the examinees
# who take each form are read from separate files.
# In this case integers need to be used as indices for
# forms. The index of Form X is 2.
set items(2) [list 37 38 25 39 40 26 41 42 27 43 44 28 \
    45 46 29 47 48 30 49 50 31 51 52 32 \
    53 54 33 55 56 34 57 58 35 59 60 36]

# Item responses are in columns 1-36 of input record
# for both forms.
set respFmt(1) 36i1
set respFmt(2) 36i1

# Read Form Y data (group 1)
# The second argument being 1 indicates all examinees
# read took the form associated with index 1 (Form Y).
# The fifth argument being 1 indicates all examinees
# are in group 1.
read_examinees_missing mondaty.dat 1 items respFmt 1

# Read Form X data (group 2)
# The second argument being 2 indicates all examinees
# read took the form associated with index 2 (Form X).
# The fifth argument being 2 indicates all examinees
# are in group 2.
read_examinees_missing mondatx.dat 2 items respFmt 2

# Compute starting values for item parameter estimates
starting_values_dichotomous

# Perform EM iterations for computing item parameter estimates
# and mean and s.d. of latent variable distribution for
# group 2.
EM_steps -estim_dist_mean_sd

# Print item parameter estimates and discrete latent
# variable distributions, and moments of
# latent variable distributions.
print -item_param -latent_dist -latent_dist_moments

# end of run
release_items_dist
-------------------------------------------------------------------------
```

```
Number of items: 60
Number of latent variable points: 40
Number of examinee groups: 2

Default prior for a-parameters:
 beta  a: 1.750   b: 3.000   lower limit: 0.000    upper limit: 3.000
Default prior for b-parameters:
 beta  a: 1.010   b: 1.010   lower limit: -6.000   upper limit: 6.000
Default prior for c-parameters:
 beta  a: 3.500   b: 4.000   lower limit: 0.000    upper limit: 0.500

Read 1638 examinee records from file mondaty.dat
Read 1655 examinee records from file mondatx.dat

EM iterations
(iteration: parameter criterion, mean criterion, sd criterion,
 marginal posterior mode)
    1: 0.325684  0.307720  0.085317  -67330.9426
    2: 0.121390  0.022953  0.013473  -67308.9803
    3: 0.069812  0.014435  0.015063  -67302.3718
    4: 0.041066  0.010214  0.012612  -67299.7890
    5: 0.023979  0.007613  0.010576  -67298.4860
    6: 0.013849  0.005964  0.009086  -67297.6683
    7: 0.009451  0.004884  0.007983  -67297.0721
    8: 0.008185  0.004146  0.007137  -67296.5982
    9: 0.007235  0.003621  0.006464  -67296.2041
   10: 0.006493  0.003229  0.005911  -67295.8689
   11: 0.005892  0.002923  0.005441  -67295.5807
   12: 0.005389  0.002674  0.005033  -67295.3316
   13: 0.004956  0.002465  0.004670  -67295.1158
   14: 0.004576  0.002283  0.004343  -67294.9284
   15: 0.004236  0.002122  0.004046  -67294.7658
   16: 0.003930  0.001976  0.003772  -67294.6245
   17: 0.003651  0.001843  0.003520  -67294.5018
   18: 0.003396  0.001721  0.003286  -67294.3952
   19: 0.003160  0.001607  0.003069  -67294.3025
   20: 0.002942  0.001502  0.002866  -67294.2220
   21: 0.002740  0.001404  0.002677  -67294.1521
   22: 0.002553  0.001312  0.002500  -67294.0913
   23: 0.002379  0.001226  0.002335  -67294.0385
   24: 0.002217  0.001146  0.002181  -67293.9926
   25: 0.002040  0.001071  0.002037  -67293.9527
   26: 0.001927  0.001001  0.001901  -67293.9180
   27: 0.001798  0.000935  0.001776  -67293.8878
   28: 0.001677  0.000874  0.001658  -67293.8616
   29: 0.001563  0.000817  0.001548  -67293.8388
   30: 0.001457  0.000764  0.001445  -67293.8190
   31: 0.001359  0.000714  0.001349  -67293.8018
```

```
   32: 0.001267  0.000667  0.001259  -67293.7868
   33: 0.001181  0.000623  0.001175  -67293.7738
   34: 0.001101  0.000582  0.001096  -67293.7625
   35: 0.001027  0.000544  0.001023  -67293.7526
   36: 0.000957  0.000508  0.000955  -67293.7440

Item Parameter Estimates
(a, b, c for 3PL, 2PL, 1PL; a, b1, b2, ... for GPCM, PCM)
1   0.904065    -1.342600   0.209782
2   0.472086    -0.365138   0.121636
3   0.573082    -0.776638   0.177935
4   0.674595    -1.204559   0.310948
5   1.341454    0.127767    0.317220
6   0.498942    0.447334    0.325796
7   0.908865    -0.344550   0.175079
8   1.072157    -0.776767   0.104713
9   0.897653    0.048058    0.155941
10  0.762528    -0.270404   0.099967
11  0.883926    0.580379    0.242801
12  0.654135    0.940711    0.253342
13  1.064732    -0.114612   0.195305
14  0.881212    0.408263    0.159102
15  0.823425    -0.079773   0.144293
16  1.395526    0.531817    0.258702
17  1.317010    0.631790    0.269624
18  1.095121    0.404573    0.195341
19  1.350874    1.092516    0.238634
20  1.084274    0.692115    0.118706
21  1.207214    1.329384    0.192204
22  1.109889    0.896887    0.109516
23  1.381480    1.716702    0.103132
24  1.296218    1.904260    0.082859
25  0.450195    -1.103938   0.233453
26  0.607664    -1.061523   0.293607
27  0.654993    -0.597720   0.118159
28  0.608414    -0.415396   0.087000
29  1.153304    0.567131    0.318498
30  0.846807    -0.214362   0.036832
31  0.272671    2.192607    0.129640
32  1.567267    0.604057    0.247199
33  1.547175    0.963907    0.197080
34  0.639390    1.838167    0.074147
35  1.269842    1.075187    0.068386
36  1.132177    1.867853    0.115002
37  0.509168    -2.300033   0.214556
38  0.771774    -0.933231   0.123109
39  1.462344    0.064269    0.288840
40  0.993275    -0.537367   0.297272
41  0.908665    0.079528    0.343232
```

```
42  1.168415    -0.514173   0.252219
43  0.911395    0.600598    0.367448
44  0.970805    0.311458    0.248061
45  1.243030    0.002410    0.262458
46  1.070866    0.383164    0.166861
47  0.926583    0.204012    0.262102
48  0.894838    0.111030    0.239211
49  0.638914    -0.034594   0.126210
50  1.106195    0.546097    0.215084
51  0.880720    0.661729    0.245532
52  1.109415    0.153550    0.065874
53  0.485861    0.989449    0.141474
54  0.902095    0.665986    0.089723
55  1.417103    1.092786    0.160834
56  0.972486    1.126737    0.144983
57  1.214783    1.492747    0.245017
58  0.888044    1.283014    0.089996
59  0.431652    3.532102    0.124595
60  0.874993    2.797930    0.108361

Discrete Latent Variable Distribution for Group 1
-4.000000   2.745344e-05
-3.794872   6.106663e-05
-3.589744   1.302378e-04
-3.384615   2.663153e-04
-3.179487   5.221329e-04
-2.974359   9.815038e-04
-2.769231   1.769004e-03
-2.564103   3.056973e-03
-2.358974   5.065011e-03
-2.153846   8.046278e-03
-1.948718   1.225563e-02
-1.743590   1.789790e-02
-1.538462   2.506079e-02
-1.333333   3.364442e-02
-1.128205   4.330694e-02
-0.923077   5.344755e-02
-0.717949   6.324468e-02
-0.512821   7.175402e-02
-0.307692   7.805385e-02
-0.102564   8.140824e-02
0.102564    8.140824e-02
0.307692    7.805385e-02
0.512821    7.175402e-02
0.717949    6.324468e-02
0.923077    5.344755e-02
1.128205    4.330694e-02
1.333333    3.364442e-02
1.538462    2.506079e-02
```

```
1.743590    1.789790e-02
1.948718    1.225563e-02
2.153846    8.046278e-03
2.358974    5.065011e-03
2.564103    3.056973e-03
2.769231    1.769004e-03
2.974359    9.815038e-04
3.179487    5.221329e-04
3.384615    2.663153e-04
3.589744    1.302378e-04
3.794872    6.106663e-05
4.000000    2.745344e-05

Discrete Latent Variable Distribution for Group 2
-4.699982   2.745344e-05
-4.480500   6.106663e-05
-4.261018   1.302378e-04
-4.041536   2.663153e-04
-3.822054   5.221329e-04
-3.602572   9.815038e-04
-3.383090   1.769004e-03
-3.163608   3.056973e-03
-2.944126   5.065011e-03
-2.724644   8.046278e-03
-2.505162   1.225563e-02
-2.285680   1.789790e-02
-2.066198   2.506079e-02
-1.846716   3.364442e-02
-1.627234   4.330694e-02
-1.407752   5.344755e-02
-1.188270   6.324468e-02
-0.968788   7.175402e-02
-0.749306   7.805385e-02
-0.529824   8.140824e-02
-0.310342   8.140824e-02
-0.090860   7.805385e-02
0.128622    7.175402e-02
0.348104    6.324468e-02
0.567586    5.344755e-02
0.787068    4.330694e-02
1.006550    3.364442e-02
1.226032    2.506079e-02
1.445513    1.789790e-02
1.664995    1.225563e-02
1.884477    8.046278e-03
2.103959    5.065011e-03
2.323441    3.056973e-03
2.542923    1.769004e-03
2.762405    9.815038e-04
```

```
2.981887    5.221329e-04
3.201369    2.663153e-04
3.420851    1.302378e-04
3.640333    6.106663e-05
3.859815    2.745344e-05

Moments of Latent Variable Distributions (group 1, 2, etc)
Mean:   0.000000    -0.420083
s.d.:   0.999646    1.069596
```

2.6 Bootstraping Item Parameter Estimates

This example shows how to use the `bootstrap_sample` command to bootstrap item parameter estimation. This example uses the same data used in the example of Section 2.1. Bootstrap samples are generated, and for each bootstrap sample item parameter estimates are computed. The item parameter estimates for each sample are saved. The bootstrap results can be used to compute standard errors or confidence intervals for item parameter estimates.

```
#
# mondaty_boot.tcl
#
# Bootstrap item parameter estimates using data mondaty.dat.
# For each bootstrap replication the parameter
# estimates for all items are written to
# one line of the output file.

# Number of bootstrap replications.
# Probably should be at least 100 to compute
# standard errors of item parameter estimates.
set nboot 10

# Do not print any output to log file
output -no_print

# 36 items
allocate_items_dist 36

# Read item responses from mondaty.dat
read_examinees mondaty.dat 36i1

# Open file to contain parameter estimates
# for each bootstrap sample.
set fileID [open mondaty_boot.out w]

# Set seed for random number generator used
# for bootstrap
bootstrap_seed 295736287

# loop over bootstrap replications
```

```
foreach b [seq 1 $nboot] {

    # generate bootstrap sample
    bootstrap_sample

    # calculate starting values for item parameters
    starting_values_dichotomous

    # calculate item parameter estimates
    set niter [EM_steps]

    # Print number of EM iterations used for this sample
    puts "Bootstrap sample $b: $niter"

    # write parameter estimates for all items
    # to one line of output file separated by tabs
    foreach i [seq 1 [num_items]] {
        puts -nonewline $fileID \
           [joinf [item_get_params $i] "\t" %.6f]
        if {$i < [num_items]} {
            puts -nonewline $fileID "\t"
        } else {
            puts -nonewline $fileID "\n"}
    }
}

# close output file
close $fileID

release_items_dist
```

2.7 Simulating Item Responses

This example demonstrates how to use ICL to simulate item responses to 36 dichotomous items modeled using the three–parameter logistic model and 4 polytomous items modeled using the generalized partial credit model. The item parameter estimates for the 36 dichotomous items are those produced in the example of Section 2.1. The item parameter estimates for the dichotomous items needed in this example could be obtained by adding the following commands to the command file in Section 2.1 (‘`mondaty.tcl`’) before the `release_items_dist` command.

```
# Write parameter estimates with 8 digits after
# the decimal point
write_item_param mondaty.par -format %.8f
```

Each line in the file written by the example contains simulated item responses for an examinee in the first 40 columns, followed by a space and the simulated examinee proficiency (theta).

```
#
# sim_resp.tcl
```

```
#
# Simulate responses to 36 dichotomous items
# using the three-parameter logistic (3PL) model,
# and four polytomous items using the
# generalized partial credit model (GPCM).

# Number of examinees to simulate
set num_examinees 2000

# Name of file to contain simulated responses
set sim_file sim_resp.dat

# Name of file containing item parameters for dichotomous items
set par_file mondaty.par

# Supress written output from subsequent ICL commands
output -no_print

# Set default item parameter priors to none since
# parameters are not being estimated, but only used
# to generate item responses.
options -default_prior_a none -default_prior_b none
options -default_prior_c none

# Create list giving model to use for each item.
# The first 36 items are dichotomous items modeled
# using the three-parameter logistic model, and the last four items
# are 4 category polytomous items modeled by
# the generalized partial credit model.
# The variable 'model' is a list containing
# 36 1's followed by four 4's.
set model [concat [rep 1 36] [rep 4 4]]

# 40 items to be modeled
allocate_items_dist 40 -models $model

# Read item parameters for the 36 dichotomous items.
# These are parameter estimates produced by
# mondaty.tcl.
read_item_param $par_file

# Assign parameters for the four polytomous items.
# The order of the parameters for each item is: a, b1, b2, b3.
item_set_params 37 [list 1.0 0.5 0.0 1.0]
item_set_params 38 [list 0.75 -2.0 0.0 2.0]
item_set_params 39 [list 0.5 0.0 -0.5 -1.0]
item_set_params 40 [list 1.5 -1.0 0.5 0.0]

# Set seed of random number generator used to
```

```
# simulate item responses.
simulate_seed 4967363

# Set seed of random number generator used to
# simulate examinee thetas.
normal_seed 5630837

# Open file to contain simulated item responses
if [catch {open $sim_file w} fileID] {
   error "Could not open $sim_file"
}

# Loop over simulated examinees
for {set i 0} {$i < $num_examinees} {incr i} {

   # Simulate value of latent variable for an examinee
   set theta [rand_normal]

   # Simulate item responses for an examinee
   set r [simulate_response_str $theta]

   # Write simulated item responses followed by
   # a space and the simulated theta formatted
   # to have 6 digits after the decimal point.
   puts $fileID "$r [format %.6f $theta]"
}

close $fileID

# end of run
release_items_dist
```

2.8 Estimation of Dichotomous and Polytomous Items

This example illustrates estimating item parameters for a mix of 36 dichotomous items modeled by the three–parameter logistic model, and 4 polytomous items modeled by the generalized partial credit model. Each examinee takes all 40 items. The data used are those simulated in the previous example. Below the output produced by the command file '`mondaty_poly.tcl`' is given below.

```
IRT Command Language (ICL)
Version 0.020301

Feb 28, 2002 06:23

Command file mondaty_poly.tcl
------------------------------------------------------------------------
#
# mondaty_poly.tcl
#
```

```
# Estimate parameters for 36 dichotomous items
# modeled using the three-parameter logistic (3PL) model,
# and four polytomous items modeled using the
# generalized partial credit model (GPCM) using
# data simulated by polysim.tcl.

# Name of file containing item responses
set resp_file sim_resp.dat

# Write output to mondaty_poly.log
output -log_file mondaty_poly.log

# Create list giving model to use for each item.
# The first 36 items are dichotomous items modeled
# using the three-parameter logistic model, and the last four items
# are 4 category polytomous items modeled by
# the generalized partial credit model.
# The variable 'model' is a list containing
# 36 1's followed by four 4's.
set model [concat [rep 1 36] [rep 4 4]]

# 40 items to be modeled
allocate_items_dist 40 -models $model

# Read examinee item responses.
# Responses to items are at the beginning
# of each record
read_examinees $resp_file 40i1

# Compute starting values for 3PL items
starting_values_dichotomous

# Perform EM iterations for computing item parameter estimates.
EM_steps

# Print item parameter estimates
print -item_param

# end of run
release_items_dist
-------------------------------------------------------------------------

Number of items: 40
Number of latent variable points: 40
Number of examinee groups: 1

Default prior for a-parameters:
 beta  a: 1.750   b: 3.000   lower limit: 0.000    upper limit: 3.000
Default prior for b-parameters:
```

```
 beta  a: 1.010   b: 1.010   lower limit: -6.000   upper limit: 6.000
Default prior for c-parameters:
 beta  a: 3.500   b: 4.000   lower limit: 0.000    upper limit: 0.500

Read 2000 examinee records from file sim_resp.dat

EM iterations
(iteration: parameter criterion, marginal posterior mode)
    1: 1.000000  -49839.8854
    2: 0.156021  -49813.0236
    3: 0.050711  -49806.9268
    4: 0.020425  -49804.6537
    5: 0.013235  -49803.4438
    6: 0.009502  -49802.6581
    7: 0.007414  -49802.0978
    8: 0.006466  -49801.6785
    9: 0.005663  -49801.3557
   10: 0.004977  -49801.1018
   11: 0.004392  -49800.8988
   12: 0.003877  -49800.7345
   13: 0.003436  -49800.6002
   14: 0.003056  -49800.4897
   15: 0.002729  -49800.3982
   16: 0.002457  -49800.3222
   17: 0.002216  -49800.2589
   18: 0.002002  -49800.2061
   19: 0.001813  -49800.1619
   20: 0.001644  -49800.1250
   21: 0.001493  -49800.0941
   22: 0.001360  -49800.0682
   23: 0.001241  -49800.0466
   24: 0.001133  -49800.0284
   25: 0.001035  -49800.0133
   26: 0.000945  -49800.0005

Item Parameter Estimates
(a, b, c for 3PL, 2PL, 1PL; a, b1, b2, ... for GPCM, PCM)
1   0.784632    -1.494295   0.216542
2   0.509761    -0.213177   0.174918
3   0.480782    -1.230036   0.171035
4   0.556064    -0.634404   0.230179
5   0.660246    -1.424265   0.235329
6   0.717690    -0.881038   0.313296
7   1.232697    0.008603    0.273957
8   0.406700    0.248871    0.275591
9   0.685756    -0.508994   0.191637
10  0.975991    -0.317898   0.202043
11  1.238413    -0.657687   0.183721
12  0.716615    -0.189311   0.144466
```

```
13  0.827056    0.107852    0.184588
14  0.708840    -0.244146   0.114232
15  1.148081    0.551510    0.320706
16  0.849328    0.653557    0.269533
17  0.552343    0.959053    0.214288
18  0.900992    -0.033334   0.101448
19  1.004164    -0.117059   0.207532
20  0.848080    0.364010    0.150887
21  0.336177    2.703949    0.226385
22  0.894207    0.021954    0.183050
23  1.215446    0.505058    0.270023
24  1.320996    0.530310    0.210530
25  1.394385    0.597443    0.247531
26  1.010578    0.361881    0.208454
27  1.342652    0.967083    0.207249
28  1.310263    1.108302    0.230058
29  1.163402    0.789757    0.153663
30  0.851033    1.626691    0.095071
31  1.229428    1.410455    0.205899
32  0.938290    0.960744    0.107012
33  1.200863    1.030150    0.057851
34  1.460793    1.711502    0.113126
35  1.294422    1.889917    0.094107
36  0.867795    2.252590    0.134263
37  0.918482    0.695774    -0.096936   1.064804
38  0.755907    -1.959096   -0.031002   2.069068
39  0.486778    0.116498    -0.528472   -0.930240
40  1.469481    -1.098740   0.584653    -0.002246
```

2.9 Data Processing Using Tcl

Data processing was required to produce the some of the input data sets used in the previous examples. The examples in this section show how to perform this data processing using the built–in Tcl commands in ICL. These examples illustration how the processing needed to put data into a form usable by ICL can be performed by ICL itself without needing to use a separate program.

The first example shows how separate data sets for Form X and Form Y were merged into the data set used as input for the example in Section 2.2 (multiple group estimation). This step can be avoided, as shown in Section 2.5, by using the `read_examinees_missing` command to read the data directly from the two individual files instead of using a merged data set.

The command file to merge these data sets (‘`mergedat.tcl`’) is given below. Comments in the command file explain the commands used.

```
#
# mergedat.tcl
#
# Combine separate data files mondatx.dat and mondaty.dat for
# Forms X and Y into a single data file (mondat.dat) where each line
```

```
# contains the responses of the unique items for Form Y,
# the common items, and the unique items for Form X.
# Responses to items an examinee did not take are indicated
# by a period.


# Convert item responses from file for one form
# to format used for combined file.
# The item responses to 36 items are in columns 1-36.
# The common items are items 3, 6, ..., 36,
# and the remaining items are unique to the form.
#
# Arguments
#   file    Name of file containing item responses for one form
#   outID   Open file channel of output file
#   group   Group number for this data (1 or 2)
proc WriteResponses {file outID group} {

    # try to open input file
    if {[catch {open $file} inId]} {
        error "Cannot open $file"
    }

    # set string of missing responses to 24 unique
    # items on the form not taken
    set missing [string repeat . 24]

    # read records in input file
    while {[gets $inId line] > 0} {

        # initialize strings containing common and
        # unique items
        set common {}
        set unique {}

        # put item responses in string allresp
        # item responses are in columns 1-36
        set allresp [string range $line 0 35]

        # loop over item responses separating
        # common and unique items
        set itemno 1
        foreach i [split $allresp {}] {
            if {$itemno % 3 == 0} then {
                append common $i
            } else {
                append unique $i
            }
            incr itemno
```

```
        }

        # Unique items for form taken by group 1
        # are written first, followed by common items,
        # then unique items for form taken by
        # group 2
        if {$group == 1} then {
            puts $outID "$unique$common$missing"
        } else {
            puts $outID "$missing$common$unique"
        }
    }

    close $inId
}

# open output file
if {[catch {open mondat.dat w} out]} {
    error "Could not open output file"
}

# write Form Y data
WriteResponses mondaty.dat $out 1

# write Form X data
WriteResponses mondatx.dat $out 2

# close output file
close $out
```

The second data processing example is the ReadItemResp procedure used in Section 2.4 (pretest calibration).

```
#
# pretest_dat.tcl
#
# Read simulated data for pretest example using
# simulated data from Ban, Hanson, Wang, Yi, & Harris (2000).
# Each simulated examinee received a CAT of 30 operational items
# and 10 pretest items. The 30 operational items
# were chosen from a pool of 520 items. The 10 pretest
# items were taken by all examinees.
# Each record in the input data set consists of
# the item numbers of the operational items
# taken by the examinee in columns 15-104 (3 digits per item)
# and the operational item responses in columns 105-134.
# The pretest item responses are in columns 145-154.
# This procedure reads a record for each examinee, creates
# an item response vector of 530 responses for the examinee
# and calls the add_examinee command using the item response
# vector.
```

```
# The argument is the file from which to read the data.
proc ReadItemResp {file} {

    # Number of items
    set adminOperItems 30
    set totOperItems 520
    set preItems 10

    # open input file
    if [catch {open $file} fileId] {
        error "Cannot open $file"
    }

    # loop over records
    # The gets command reads one record into variable 'line'
    while {[gets $fileId line] > 0} {

        # initialize list of all item responses
        set allResp [list]

        # Read operational item responses
        set posNum 14
        set posResp 104
        for {set i 0} {$i < $adminOperItems} {incr i 1} {
            # Read item number of operational item
            set operItemNo [string range $line $posNum \
                [expr {$posNum+2}]]

            # strip leading zeros from item number
            string trimleft $operItemNo 0

            # Assign item response for operational item
            # to array itemResp.
            # Change 1 to 0 and 2 to 1
            # (2 indicates a correct response and 1 an
            # incorrect response in the original file)
            set r [string index $line $posResp]
            set itemResp($operItemNo) [string map {1 0 2 1} $r]
            incr posNum 3
            incr posResp
        }

        # write operational item responses to list of all responses
        for {set i 1} {$i <= $totOperItems} {incr i} {
            if {[info exists itemResp($i)] == 1} then {
                # add item response to list
                lappend allResp $itemResp($i)
            } else {
                # add -1 indicating the examinee did not respond
```

```
                # to the item
                lappend allResp -1
            }
        }

        # Remove all elements from itemResp to initialize
        # for next examinee.
        unset itemResp

        # read pretest item responses and add them to
        # list of item responses
        set preResp [string map {1 0 2 1} [string range $line 144 153]]
        set allResp [concat $allResp [split $preResp {}]]

        # Add examinee to data used for estimation
        add_examinee $allResp

    }
    close $fileId
}
```

Appendix A Format Specifiers for C sprintf Function

This appendix describes the format specifiers used in the `print`, `write_item_param_channel`, `write_item_param`, `write_latent_dist_channel`, and `write_latent_dist` commands which follow the convention for the C sprintf function. This format specifier is also used for the Tcl `format` command. This description was created by Paul B. Patton.

All format specifiers are in the following form, where square brackets indicate optional parts:

```
%[flag][width][.prec]type
```

```
[flag]   What flag specifies
-------  ----------------------------------
 none    Right-justify; pad with 0 or blank
           on left
   -     Left-justify; pad spaces on right
   +     Always begin with + or -
 blank   Print sign for negative values only

   #     Use alternate form for these types
          c,s,d,i,u  (no effect)
          o        Prepend 0 to nonzero value
          x or X   Prepend 0x or 0X
          e, E, f  Always use decimal point
          g or G   Don't strip trailing zeros
```

```
[width]   Effect on Output
--------  ----------------------------------
  n       At least n characters, blank-padded
  0n      At least n characters, zero-padded
```

```
[.prec]   Effect on Output
--------  ----------------------------------
  none    Default precision
  .n      s         At most n characters
  .n      e,E,f,g,G Digits to right of point
  .0      e,E,f     No digits right of point
                       and no point unless #
```

```
type   Format of Output
-----  ----------------------------------
  d    signed decimal integer
  o    unsigned octal integer
  u    unsigned decimal integer
```

```
x    unsigned  hexadecimal integer (a-f)
X    unsigned  hexadecimal integer (A-F)

f    Floating point [-]dddd.ddd
e    Floating point [-]d.ddd e [+/-]ddd
E    Floating point [-]d.ddd E [+/-]ddd
g    e or f, whichever uses less space
G    E or f, whichever uses less space

c    Single character
s    Character string
%    The % character, literally
```

License

ICL is distributed under the following license. Additional licenses under which some source code used in ICL is copyrighted are provided with the ICL source code distribution.

References

Ban, J. Hanson, B. A., Wang, T., Yi, Q., & Harris, D. J. (2001). A comparative study of online pretest item–calibration/scaling methods in CAT. *Journal of Educational Measurement*, *38*, 191–212.

Bock, R. D., & Zimowski, M. F. (1996). Multiple group IRT. In W. J. van der Linden and R. K. Hambleton (Eds.), *Handbook of modern item response theory*. New York: Springer-Verlag.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, *39*, 1–38

Hanson, B. A. (1998). IRT Parameter Estimation using the EM Algorithm. [Available at http://www.b-a-h.com/papers/note9801.html]

Kolen, M. J., & Brennan, R. L. (1995). *Test equating methods and practices*. New York: Springer

Lewis, C. (1985). Discussion. In D. J. Weiss (Ed.), *Proceedings of the 1982 Item Response Theory and Computerized Adaptive Testing Conference* (pp. 203-209). Minneapolis: University of Minnesota, Department of Psychology, Computerized Adaptive Testing Laboratory.

Linacre, J. M. (1994). PROX with missing data. *Rasch Measurement Transactions*, *8(3)*, 378.

Lord, F. M. (1980). *Applications of item response theory to practical testing problems*. Hillsdale, N.J.: Lawrence Erlbaum Associates.

McLachlan, G. J., & Krishnan, T. (1997). *The EM algorithm and extensions*. New York: John Wiley & Sons.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, *16(2)*, 159–176.

Nelson, C. (2000). *Tcl/Tk Programmer's Reference*. Berkeley: Osborne/McGraw–Hill.

Welch, B. B. (1999). *Practical programming in Tcl and Tk* (3rd Edition). Upper Saddle River, NJ: Prentice Hall.

Woodruff, D. J., & Hanson, B. A. (1997). *Estimation of item response models using the EM algorithm for finite mixtures*. Paper presented at the Annual Meeting of the Psychometric Society (Gatlinburg, Tennessee, June). [Available at http://www.b-a-h.com/papers/paper9701.html]