

Chapter 5. Exercises for the Learning Analyst.

Suggested Exercises to Expand your Understanding of the Arpeggio System

1. **Sample Files.** For learning how to use Arpeggio, we have provided all the necessary files set up to perform an Arpeggio analysis, along with a set of instructions to walk you through several sample practice Arpeggio runs. The sample files include all necessary input files: *arpeggio.in* with all run parameters already set up; the Q matrix file *qmatrix.in*; and the scored student response data in file *score.in*. It also includes all the output files derived from performing the Arpeggio analysis. The Arpeggio novice can perform the run that is already set up and confirm that the output files are the same as those included. In addition, a number of straightforward modifications are suggested for the analyst to practice running Arpeggio.
2. **Make a “Check” folder..** Before beginning, the analyst should prepare two file folders, one called *LASampleRun*, and a second called *LASampleCheck*. The two folders should be identical, and should include all files that have been provided with this Arpeggio package. The analyst will perform the analysis in the first folder, and will use the second folder to confirm that the new results match the output files originally provided.
3. **Purpose of Arpeggio Analysis.** The purpose of this sample analysis is to calibrate the Fusion Model using the designated Q matrix given in *qmatrix.in* and the scored student response data given in the *score.in*. As part of the Arpeggio software, the model calibration step also produces posterior probabilities of mastery for each skill.
4. **Resources.** Be sure you refer to the other chapters and Appendices of the Manual as you go through these exercises: *arpeggio.in*; Arpeggio Run Modes, and Appendices with file formats.
5. Please check that you have the following files in the sample run folder:

	Name	Type	Modified
Input files:			
Arpeggio Control File	arpeggio.in	IN File	3/10/2005 1:46 PM
Data File	score.in	IN File	3/14/2005 5:19 PM
Q matrix input file	qmatrix.in	IN File	9/30/2004 3:31 PM
Executable Files:			
	Arpeggio3_1.exe	Application	2/3/2005 11:43 AM
	simarpeggulator3_1b.exe	Application	3/15/2005 2:57 PM
	fastclass3_1b.exe	Application	3/15/2005 2:58 PM
	tabulator3_1b.exe	Application	3/14/2005 1:26 PM
Output Files from Arpeggio:			
	log.txt	Text Doc	3/14/2005 7:55 PM
	Qmatrix.csv	MOffice	3/14/2005 5:21 PM
	ArpReport.csv	MOffice	3/14/2005 7:55 PM
	ExamReports.csv	MOffice	3/14/2005 7:55 PM
	itemparms.csv	MOffice	3/14/2005 7:55 PM
	Fitreports.csv	MOffice	3/14/2005 7:55 PM
Output Files from other executable programs:			
	JDEExamReports.csv	MOffice...	3/18/2005 9:44 AM
	EAPexamest.out	OUT File	3/18/2005 9:44 AM
	MAPexamest.out	OUT File	3/18/2005 9:44 AM
	16 file(s)		

6. One of the first steps in doing an Arpeggio analysis is to construct the data file. In this case, the data file is already set up for you, so we will inspect it to see the format you will be using in other analyses. Open *score.in* with an ASCII text editor:
- *The first line or row in score.in is a header row that provides meta-information about the data in the file, and the remaining rows consist of a rectangular array of scored item responses, with each line after line 1 corresponding to an examinee and each single-character column corresponding to item response scores.*
 - *Row 1 specifies the max score for each item. Since the sample data here consists of dichotomous items, all entries in row 1 are 1. If we had a case in which, say, item 1 is partial credit scored 0, 1, 2, 3 and item two is scored 0,1, 2, and item three is scored 0, 1, 2, 3, 4, 5, and all other items are dichotomous, then row 1 would be: 32511111...1.*
 - *Row 2 represents the scored item responses for student 1 (notice the offset (row 2) ←→(student 1)). Consequently if your dataset has 1000 examinees, then the file score.in will contain 1001 rows, the extra row is the row 1 header row that gives max score for each item.*
 - *Since all items for this sample data are dichotomous, all data entries from row 2 on are 0 (incorrect) or 1 (correct). If instead we had the case described above, with max item scores 325111...1, then we might have some examinees with scored response vector 21400110110011.... Notice in this polytomous case, that a data row 444101100101100... etc would be “illegal.” The reasons are that: item 1 score 4 is greater than max score 3 (as given in row 1). Similarly, item 2 score 4 is greater than max score 2 as given in row 1. The rest of the scores are within range. For example, item 3 score of 4 is a legal score since max score for item 3 is 5 according to row 1. **The Arpeggio software will catch illegal data.** It will show an error message on the screen and halt.*
 - *Notice that this version of Arpeggio is programmed to recognize the data from position 1 through I, where I is the total number of items. In this case there are 34 items, so the data for each examinee is in columns 1 through 34.*
 - *Note there are no student IDs in this file, so each row is exactly 34 characters long, and each character is either the max item score (row 1) or an examinee’s item score (rows 2 and below). An ID number CAN be stored for each record, but it must be stored to the RIGHT of the data. You would skip several columns and store the ID number from that point on. For example, in this dataset, since there are 34 items, you could start in column 40 and store a 10 digit ID for each student in columns 40-49. That was not done in this sample dataset.*
 - *One last caveat is to note that in this version of the software, each item must have max score 9 or less. There is no provision for individual item scores to require more than 1 character position.*
 - *For extensive operational or research work it is highly recommended that you program a statistical utility program that extracts your response data from your source data files and constructs the file score.in in the format that Arpeggio requires. That is easier and more reliable than extracting the data by hand each time you run Arpeggio.*
7. The next step is to construct the Q matrix. The Q matrix specifies for each item in the test, which skills are required for that item. The required format for the Q matrix is as shown in the sample file *qmatrix.in*.

- *The very first line of the Q matrix file (“H”) is a 6 character alphanumeric header*
 - *The second line gives the number of items in the test—34 in this example case.*
 - *The third line gives the number of skills—4 in this example.*
 - *Rows 4, 5, 6 and 7 are for documentation. You can use these as convenient for your analytic purposes..*
 - *This qmatrix.in file is a comma delimited file, consequently new fields begin or end where there are commas. The trailing comma at the end of each row is extremely required. If a trailing comma is missing your data will either be misread or the program will halt.*
 - *Row 8 is a header line that lists shortened versions of the field names in the form of column headings. In this example the field names are: ITEM,TOTAL,SKILL_1,SKILL_2,SKILL_3,SKILL_4, (notice the trailing comma). That means that in each row after row 8, the first piece of data is the item number or item ID, and the second piece of data (called TOTAL here) is the total number of skills required by the given item. The item number or item ID can be any number of alpha-numeric characters long, and goes from the beginning of the line to the first comma. The third and remaining fields are skill names. In this example they are generic SKILL_1, etc.*
 - *After the header row, starting in row 9 and continuing are the Q matrix specifications for each item. One row for each item. So if the number of items is I then the rows in the qmatrix.in file giving skills specifications for each item are row 9 through I + 8. In this example, check that qmatrix.in has skill information in row 9 through row 34 + 8 = 42.*
 - *Each item row starts with an item ID. This can be an alphanumeric string. The item IDs can be multiple number of characters long, since the file is comma delimited. In this example all the item ID’s are long and complicated—for example “accession_1” etc. You can have the item ID be whatever you want. You don’t need to use the term “accession” in each item ID.*
 - *For each item, the second entry in the row is the total number of skills required by that item. For example item 3 requires one skill, item 10 requires two skills.*
 - *In each item row, after the item ID and the total number of skills required for that item are given, the row specifies which skills are required for the item. In each item row, the item ID and total number of skills required for that item (each with its own comma) are followed by exactly K data locations ,each separated by commas, where K is the total number of skills for the test. For example the skill specifications for items 8-12 in your example are given in qmatrix.in as follows: is listed in the corresponding position, separated by commas. For example the one skill required by item 5 is skill 3, and the skill number 3 is entered in the third position after the total number of skills. The two skills required by item 10 are skills 3 and 4 and they are entered in positions 3 and 4. etc.*
- ```

accession_8,1,,2,,,
accession_9,1,,,3,,
accession_10,2,,,3,4,
accession_11,2,1,,,4,
accession_12,2,,,3,4,

```
- *These mean that item 8 requires only skill 2; item 9 requires only skill 3; item 10 requires skills 3 and 4; item 11 requires skills 1 and 4; and item 12 requires skills 3 and 4.*
  - *Notice the required trailing commas at the ends of each item row.*

- *The very last line in the file is a trailer code “T” it means the qmatrix file is ended.*
8. Next step is to construct the Arpeggio control file *arpeggio.in*. This file is complex. The reader should consult both the Arpeggio.in Chapter of this manual, and also the Arpeggio Run Modes Chapter.
- *It is best practice to always start with an existing arpeggio.in and change its entries as your new run requires. Since the programs in the Arpeggio Suite all expect arpeggio.in to have a certain format, with particular sections, in a given order, and with particular entries within each section, and since the programs are very sensitive to commas, spacing, etc, this is much easier and less error prone than entering all information from scratch.*
  - *The arpeggio.in control file has lots of features and options. In doing these exercises we will look at only a few of them. The Arpeggio.in Chapter reviews all control options.*
  - *The lines that begin with the symbol > are header lines, one for each section of arpeggio.in and specify what the immediately following lines represent. The first section is >Global, the next section is >Input, the next section is >Output, the next >MCMC, etc.*
  - *Within each section, the lines MUST be in the exact order given. For example, within >Global, the first line Global(1) must be # of examinees, the 2<sup>nd</sup> line Global(2) must provide an internal run code to select which portions of the program will run—1 = Arpeggio+Fitstats; 2 = Arpeggio+Gamestats; 3 = FitStats only. For the various analyses in suggested here we will run Arpeggio in with Global(2) = 1 = Arpeggio+FitStats.*
  - *Continuing the lines under >Global: line 3 Global(3) is a run ID that you can use for your own purposes—it is ignored by the software. Line Global(4) selects between two different models for handling partial credit item scoring. If all items are dichotomous, as they are in the sample folder, then the software ignores this line. It is necessary in any partial credit case.*
  - *>Global. In summary, we will list the lines within the >Global section. Please check how the Global section in your arpeggio.in file is set up.*
- Global(1) = # of examinees. If Global(1) is blank or is less than 200, the Arpeggio software uses ALL examinees in the *score.in* file. If it is a number N greater than 199 and strictly less than the total number of examinees in the *score.in* file the Arpeggio uses the first N records for its analysis. The allowed range for Global(1) is 0 to 20,000. A good way to see if your run parameters will work without wasting a lot of time is to do the very first run with Global(1) = 200. Arpeggio will take the first 200 records in your data file and do an Arpeggio run just on them. That analysis will be very fast.
- Global(2) = internal program mode as discussed above. For these exercises we always will run with Global(2) = 1
- Global(3) = an alphanumeric ID or phrase you can use in any way you want
- Global(4) = which partial credit model to use (See Templin, He, Roussos, Stout, 2003, and Bolt & Fu, 2004).
- *Next look at the >Input control section. These lines specify the names of all input files to be used in the next Arpeggio run. As above, we will use the notational convention of Input(m) to mean Line m under control section Input*