```
This file contains item information for data with stem bdicmf
   The following information can be modified:
     (Note the column numbers)
 The item label                 (Cols  1 -  8)
 The item type (M or S)         (Col   1     )
 An option label                (Cols 28 - 35)
 An option weight (use decimal) (Cols 40 - 45)
Caution!  Do not disturb the locations of other information.
**********************  Item    1  **************************
Sad                                     Item Label
Scale                                   Item Type
None                                    Correct Option
 4                                      Number of Options

Option Number  Data Char.    Label     Weight   Frequency
          1         0          0           .0      117
          2         1          1          1.0      296
          3         2          2          2.0      145
          4         3          3          3.0       90
Item omit frequency                                   0
**********************  Item    2  **************************
Future                                 Item Label
Scale                                   Item Type
None                                    Correct Option
 4                                      Number of Options

Option Number  Data Char.    Label     Weight   Frequency
          1         0          0           .0       92
          2         1          1          1.0      343
          3         2          2          2.0      143
          4         3          3          3.0       70
Item omit frequency                                   0
```

Figure 17: Setup of the TestGraf item information file `bdi.itm` for some Beck Depression Inventory data.

You will not need to modify the default choices offered to you, and can therefore progress through the analysis by responding to each choice by simply typing `Enter`. When TestSave is finished, you might want to inspect the `.par` and `.itm` files that it produces.

You can also try your hand at scale data by setting up the Beck Depression Inventory data. For this type

```
ts bdi
```

The only non-default choice required for these data is to specify the default item type as scale. But you could try your hand at specifying item and even option labels by modifying the `.itm` file afterwards.

5.2 Running TestGraf

Once the setup program TestSave has been run, one is ready to run the analysis program TestGraf. We have briefly described above the files which TestGraf requires for processing. Ordinarily, all that is now required is to type

```
tg filename
```

Here `filename` refers to the set of up to 8 characters that were used as the argument in the invocation of TestSave, and which forms the first part or stem of the names of the files set up by TestSave. For example, typing `tg psych101` invokes TestGraf and at the same time identifies the files

```
psych101.par
psych101.exo
psych101.ito
psych101.itm
psych101.lab
psych101.abi
```

that will be used by TestGraf.

The program will also set up two additional files:

`filename.out`: contains numerical information output by TestGraf, `filename.prb`: contains values of the option characteristic curves, which can then be used on subsequent runs of TestGraf to avoid the time taken to recompute curve values,

These files are described in more detail below.

If TestGraf is invoked without file stem, it will prompt you for this stem.

As we indicated above, the file with the extension `.par` contains the various parameters which control how TestGraf does its computation. Figure 5.1.7 shows how this file appears for the introductory psychology data. On the first run, you will probably want to leave this file as is.

The `.par` file contains various types of information:

- title lines, which can be any string of up to 40 characters
- numerical values, which can be either integer or decimal numbers. The number of examinees is an example.
- option lines, which can assume only a small number of values. Options are spelled out in the `.par` file produced by TestSave, but are actually identified by the first character only, and this can be in either upper or lower case. For the example, the display variable options are identified by N, E, F, or O, standing for Normal, Expected correct, Formula score, and Observed score, respectively. The technique for estimating rank order has as options N and F, standing for Number correct and Formula score, respectively.
- logical values, which can be either true/false, or off/on. The true or on state is indicated by an asterisk "*" in column 1, and the false or off state by a blank.

TestGraf first displays the contents of the `.par` file that you are permitted to change, and then asks if you wish to change any of the parameters. To change any value, move the highlighting to the item to be changed by using the up or down arrow keys, or clicking on the desired item with the mouse

Once the appropriate item is highlighted, type `Enter`. If the item is a numerical or a title line, a one-line window will appear. Just type in the new value followed by `Enter`, and the changed value will appear. If the appropriate item is logical, TestGraf will just switch the value and return the highlighting to the `Continue` line.

To proceed with the computation, type `Enter` when the `Continue` line is highlighted. To abort the analysis at this point, highlight the `Quit` line and press `Enter`.

Once all desired changes have been made, TestGraf proceeds with the computation.

TestGraf dynamically allocates memory at the time of execution. For most problems the memory required will be well within the limitations imposed by the 640K memory available on most personal computers using the MS/DOS operating system. TestGraf shouldn't have any trouble, for example, with exams with 100 or so items, each with 5 or fewer options, given to 20,000 or more examinees.

If, however, more memory is required than is available, an error message will result and the program will terminate. The error message that results is somewhat unpredictable, and commonly occurring messages are:

```
run-time error F6700: ALLOCATE
- heap space limit exceeded

Cannot execute TESTGRAF.EXE

Insufficient RAM memory available.
```

Should any of these messages appear, one solution is to remove any memory-resident programs that may be in your machine, such as notepad programs like Sidekick©, the MS/DOS application GRAPHICS, or communication software. These programs or applications, called TSR's, can take up a substantial amount of memory. After removing them by modifying the `autoexec.bat` file to disable these programs, rebooting and terminating the installation of these programs, TestGraf may then run without any problems.

Alternatively, the distribution diskette contains a reduced version of TestGraf called TestAnal that only computes option characteristic curve values and sets up the `.prb` file. Since a good deal of the code in TestGraf is devoted to generating displays, TestAnal requires much less memory. Once the TestAnal run is finished, TestGraf can then be run with the computation of curves option toggled off.

Future versions of TestGraf will be able to use extended memory, but as of this writing this is not possible.

If these measures fail, one may have to resort to cutting down the size of the problem by, for example, only looking at subsets of the items, or only using subsets of examinees.

5.2.1 The Parameters Controlling TestGraf and the .par File

We turn now to a detailed description of the `.par` file, and example of which is shown in Figure 5.2.1. Note that the line numbers are not part of the file, and that the actual line begins in with the character in the fourth column in this figure.

Each line following the two title lines contains a single parameter value. These parameters are of the following types:

- integer parameters
- option lines
- floating point or real parameter
- logical items

Each value is followed by a few words of description of its role in the program.

Note that the logical parameters must take one of two values in the first column: * or a blank. Character * means that an option is toggled on or set to "On", and a blank that it is disabled or set to "Off".

The following description gives further details about each parameter, including its default value if any. Capital letters are used for clarity only and are not required.

1. The initial line containing `Run` is used to signal to TestGraf that the data are to be analyzed.
2. The first line of title (only 40 characters used).
3. The second line of title (only 40 characters used).
4. The number of examinees (no default value, cannot be changed).
5. The number of items (no default value, cannot be changed).

```
1  Run
2  Beck Depression Inventory              |First  line of title
3  McGill University Student Data         |Second line of title
4  242                  Number of examinees or respondents
5  21                   Number of test or scale items
6  51                   Number of plotting points for displays
7  2                    Number of principal components displayed
8  Expected Score       Display variable option (N, E, F, or O)
9  Number Correct       Type of scoring (MC items only) (N or F)
10 0.36                 Bandwidth or smoothing parameter value
11                      Compute curve values
12                      Input ability values
13                      Only output .pwo plotting file
14                      Control plotting limits for abscissa
15 bdi     .exo                        File for data in examinee order
16 bdi     .ito                        File for data in item order
17 bdi     .out                        File for output of results
18 bdi     .prb                        File for curve values
19 bdi     .itm                        File for item info
20 bdi     .lab                        File for examinee labels
21 bdi     .abi                        File for input abilities
22                     Omit character
```

Figure 18: The `.par` file for the Beck Depression Inventory data. The line numbers are for reference further description, and do not appear in the file.

6. The number of points at which each curve is evaluated. (max 101, default 51)

7. The number of principal components of curve variation to be displayed. Two is usually adequate. (max 5, default 2).

8. The type of display variable to be used for the independent variable, proficiency, in the plots. The options allowed are:

   N or n ... use standard normal quantiles

   E or e ... use expected total score

   F or f ... use formula score (multiple choice exams only)

   O or o ... use actual total score, or the proficiency values were input

   (default E).

9. The type of score computed from the test or scale for ranking examinees. The options allowed are:

   N or n ... number of correctly answer items (multiple choice exams) or scale score (scales)

   F or f ... formula score (multiple choice exams only)

10. The smoothing parameter for kernel smoothing with a Gaussian kernel. By default this is set to $1.1N^{-1/5}$. The higher the smoothing parameter, the smoother the curve will be. TestGraf uses a fairly conservative (smallish) value, and a slightly larger value will give smoother curves.

11. When toggled on, the item characteristic curve values are computed. Otherwise the curve values are assumed to be in the `.prb` file produced by a previous run of TestGraf. Therefore, this parameter must be toggled on for the first analysis. (default on)

12. When toggled on, the proficiency values used for ranking the examinees for the smoothing step are input from the `.abi` file. Otherwise the values used for ranking examinees are the numbers of correctly answered items. (default off)

13. When toggled on TestGraf goes into noninteractive mode, and instead of producing displays on the video screen, generates plotting commands in the file `testgraf.pwo`. (default off)

14. When toggled on TestGraf permits the user to control the range of the independent or display variable. The program prompts the user to input the initial and final values to be plotted, and the number of tik or hash marks. (default off)

Also part of the `.par` file in lines 15 to 22 is a list of file names, and the omit character. TestGraf does not display these for modification, but they can always be modified by an editor prior to running TestGraf.

The files used by TestGraf are described as follows, again identifying the files with the corresponding variable names in the program. Most of these are set up by the TestSave run, but a couple, and most notably the `.abi` file, will need to be set up independently using an editor, and in a format which is described below.

`EXOIFILE` Up to 30 characters designating the name of the file containing the test to be analyzed organized with examinees as rows, or items nested within examinees. This file contains the response patterns for each examinee in unformatted or binary form, and is the first file output by the program TestSave. It must have the extension `.exo`.

`ITOIFILE` Up to 30 characters designating the name of the file containing the test to be analyzed organized with items as rows, or examinees nested within items. This file contains the response patterns for each item in unformatted or binary form, and is the first file output by program TestSave. It must have the extension `.ito`.

`OUTFILE` Up to 30 characters designating the file to contain the results or output from the analysis by TestGraf. It can be set to whatever you wish as long as it conforms to MS/DOS file naming conventions. The default extension for this file is `.out`.

`PRBFILE` Up to 30 characters designating the file to contain values of the option characteristic curves. These can be read in on subsequent runs of the program to avoid recalculating them. The default extension for this file is `.prb`.

`ITMFILE` Up to 30 characters designating the file to contain information for each item.

`LABFILE` Up to 30 characters designating the file containing examinee labels.

`ABIFILE` Up to 30 characters designating the file containing examinee proficiency values which are input for the smoothing phase. The first line of the file must contain a FORTRAN format specification for this information. This file is not set up by program TestSave, and must be provided by the user if the `ABIWRD` has the value `.TRUE.` It must have the extension `.abi`.

If maximum likelihood estimates of proficiency or latent trait are requested (option 7), TestGraf automatically sets up this file with these proficiency estimates, both as standard normal quantile values, and as expected number correct values. In this case the file is ready to use without modification for a subsequent analysis. Caution: The contents of a previously created `.abi` file will be lost in this case.

Also in the `.par` file on line 22 is a character used to indicate that an item was not presented to an examinee. By default TestSave sets this to the blank character.

Finally, we remind the user again that any of these parameter values, the title lines, or the correct answer key can either be modified by any editing program prior to running TestGraf, or can be modified interactively within a run of TestGraf itself.

5.2.2 Output from TestGraf in the `.out` File

TestGraf sets up a file with the extension `.out` and with the first part given by the argument string provided at invocation of the program. This file contains various types of numerical output from the program.

Test Characteristic Curve Values and Derivatives: The values of the test characteristic curve, $\eta(\theta_q), q = 1, \ldots, Q$, as well as the slopes of the curves at these argument values. Also output are the estimated standard errors of the curve values.

Expected Formula Score Function: This appears only for multiple choice exams.

Logistic Model Parameter Estimates: Estimates of the parameters of the three-parameter logistic model (3).

Test Information Function: If this TestGraf analysis option has been selected, the numerical values are tabulated.

Eigenvalues for the Principal Components Analysis: If the principal components option is chosen for TestGraf, the eigenvalues of the covariance matrix generated from the correct-answer curve values are output. These permit the user to decide how many principal components to display (the default is 2).

Each of these types of information can be accessed by an editing program, which can extract a particular type for further display or processing by another program.

5.2.3 Output from TestGraf in the `.prb` File

This file contains information that allows TestGraf to redisplay the results on subsequent runs without having to recalculate the curve values. These results can also be used in other software to produce customized plots.

The contents of this file are:

1. The title.
2. The number of display variable values used for plotting.
3. The values of the standard normal display variable. These are equally spaced and range from -2.5 to 2.5.
4. The values of the expected number correct or scale score display variable. These are given as integers after being multiplied by 1000.
5. For multiple choice tests only, the values of the expected formula score display variable, multiplied by 1000 and given as integers.
6. The values of the observed score display variable.

7. The values of the five standard normal quantiles used in the displays.

8. The values of the five expected total score quantiles used in the displays.

9. For multiple choice tests only, the values of the five expected formula score quantiles used in the displays.

10. The values of the five observed score quantiles used in the displays.

11. Characteristic curve values $P_{im}(\theta_q)$. The values of the curves at the argument values $\theta_1, \ldots, \theta_Q$ are saved in this file. Although these values are probabilities, and hence lie in the unit interval [0,1], TestGraf outputs them as integers between 0 and 1000 since this economizes on both the storage that they require within the program and on the size of the output file. These values can then be used in another plotting program if desired, but they must be divided by 1000.0 before hand.

    The main reason for outputting the option characteristic curve values is that TestGraf itself can read them in on subsequent invocations of the program, thereby saving considerable computation time.

12. Standard errors for the correct answer curve (multiple choice exams) or expected item score (scales). It is these values which define the vertical cross-hatching for this curve in the item displays. As with the curve values themselves, these standard errors are output by first multiplying them by 1000 and outputting them as integers. These are also in the `.prb` extension file.

5.2.4 Output from TestGraf in the `.abi` File

If the option is chosen of computing maximum likelihood proficiency estimates for each examinee, these are output to this file. These estimates are expressed both in terms of standard normal deviates, and in terms of expected number correct. Note that these values are output in such a way that they can be used as input on a subsequent TestGraf analysis, thereby producing an iterated set of estimates. See Section 9 for further details.

6 A Technical Description of TestGraf

A technical account of much of TestGraf can be found in the paper "A Kernel Smoothing Approach to the Estimation of Item Characteristic Curves" in Ramsay (1991). In this section a less technical discussion of how TestGraf works is offered, with a brief introduction to the idea of kernel smoothing. Some knowledge of modern test theory is presumed, such as can be found in Allen and Yen (1979), Hambleton, Swaminathan, and Rogers (1991), and Lord (1981).

6.1 Kernel Smoothing and Nonparametric Regression

One of the more dramatic developments in contemporary statistics has been the evolution of techniques for the direct estimation of functions. This contrasts with the older approach of specifying a family of functions which were defined by a certain number of parameters, and then using the data to estimate the parameters. The principal problem with the parametric approach was that no matter how many parameters were used, some application always managed to present itself that needs more flexibility than the parametric family could provide. And, moreover, if a generous number of parameters were used to give sufficient flexibility for almost all problems, then many problems calling for simpler functions were over-fitted. Overfitting data leads to many problems, among which are needlessly expensive computation, poor estimation properties for the parameters actually needed, and poor power for hypothesis tests. These issues are well known in the context of the linear model or multiple regression, where great stress is placed on using the minimal number of independent variables, and hence parameters required to fit the data.

The parametric function family used most widely to model the characteristic curve for the correct option in multiple choice exams is the three-parameter logistic or 3PL model

$$P(\theta) = c + (1-c)\frac{\exp[1.7a(\theta - b)]}{1+\exp[1.7a(\theta - b)]} .$$

Although only three parameters are involved, this apparently simple model has turned out to well illustrate the problems of parametric function estimation. For example, when the item is easy, there are typically virtually no data available for estimating the guessing parameter c, and then large changes in c can be compensated for by corresponding changes in the discrimination parameter a. The result is that a, which can play an important role in describing the item, is very poorly estimated. Thus, it is wise to set $c = 0$, called the 2PL model, when the item has very low difficulty. Even when the item is of moderate difficulty, the covariance between parameter estimates is high, and very large amounts of data are required to estimate them well, even though the function itself may be well defined. Thissen and Wainer (1982) discuss these and other problems with the 3PL model. Computer programs for fitting the 3PL model are comparatively slow, complex, and loaded with heuristic devices for preventing failure.

Nonparametric regression is the term used for a wide range of techniques for direct estimation of a functional relationship between an independent variable x and a dependent variable y. These techniques vary in sophistication, computational convenience, and other important aspects. References on kernel smoothing are Green and Silverman (1994) and Eubank (1988), and an especially readable introduction can be found in Altman (1992).

The problem at hand is the estimation of an option characteristic curve, the function relating the probability of choosing an option to the value of the latent proficiency or trait variable. TestGraf uses what is perhaps the simplest and computationally most convenient procedure, *kernel smoothing.* Although there are techniques that are superior in various ways, the need to both process large amounts of data rapidly and to keep the program itself reasonably compact argue in favor of this simple but remarkably effective technique. Kernel smoothing is described as follows.

Suppose we have a set of independent variable values $x_i, i = 1, \dots, n$; and a corresponding set of dependent variable values y_i. This situation is plotted in Figure 19. Our objective is to estimate a smooth curve defined by function $\hat{f}$ with values $\hat{f}(x)$. For example, we might want to compute the value $\hat{f}(x_q)$ at an independent variable value x_q which may or may not coincide with any of the data values. We can refer to x_q as an *evaluation point.*

The principle exploited by kernel smoothing is that of *local averaging.* This means that we in effect compute an average of those values y_i which correspond to values x_i which are close to the target value x_q. For example, we could construct rules such as:

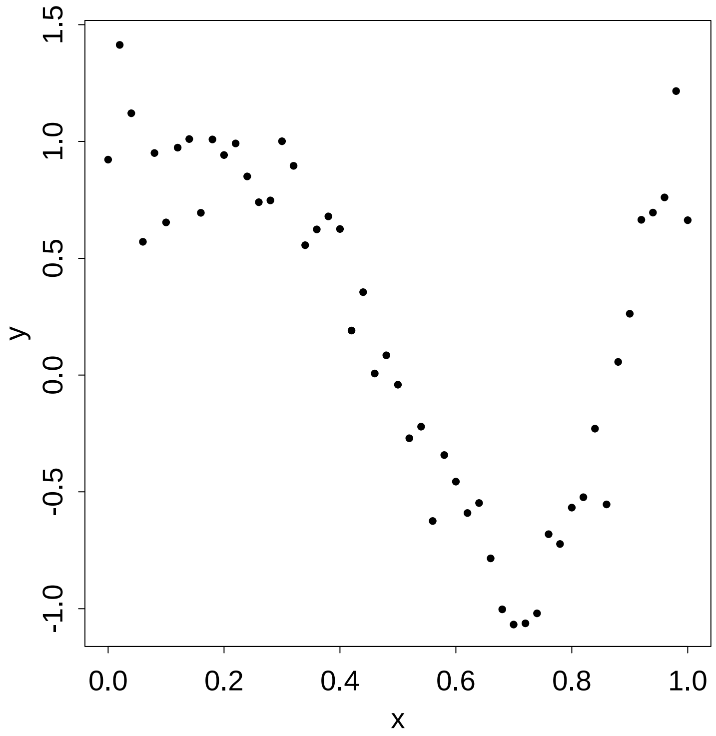

Figure 19: Some simulated data for kernel kmoothing generated by adding normally distributed error to values of the function $\cos(2\pi x^2)$.

nearest neighbor averaging: let $\hat{f}(x_q)$ be the arithmetic mean of the y_i's corresponding to the k x_i's closest to x_q,

uniform local averaging: let $\hat{f}(x_q)$ be the arithmetic mean of the y_i's corresponding to the x_i's which are no more than h units from x_q.

Each of these simple rules can be extended in various ways, and each is not unreasonable as it stands. Both exploit the notion that when estimating function values, we can borrow information from the values of the function at near points provided that we can assume that the function is relatively smooth. Indeed, you can think of the regular arithmetic mean involving the whole sample as an attempt to estimate a function which has a constant value.

The second rule, uniform local averaging, can be extended by broadening our concept of an average. Let the *weighted average* Ave_w be defined by

$$\text{Ave}_w = \sum_{i=1}^{n} w_i y_i \quad \text{where} \quad \sum_{i=1}^{n} w_i = 1 \ . \tag{9}$$

Note that the usual arithmetic mean makes use of weights $w_i = 1/n$, and since these weights do not vary over i, it can be called an *unweighted average.* Many estimates used in statistical work can be thought of as weighted averages. A particularly important example is the median, which, for odd-sized samples, is a result of using $w_i = 0$ for all values except for the central value, for which $w_i = 1$.

A *local average* can now be defined as one for which only the weights w_i corresponding to values of x_i close to a target value x_q are substantially larger than zero. Both of the schemes above are local averages for which the weights are equal for close values of x and zero otherwise, and they differ only in terms of how they define *close.*

A very general technique for defining local averages uses a *smoothing kernel function* K. This is a function with the following properties:

1. $K(u)$ is zero or positive for all values of argument u,
2. $K(0)$ is the maximum value taken by K,
3. $K(u)$ goes to zero as u deviates more and more in either direction from 0.

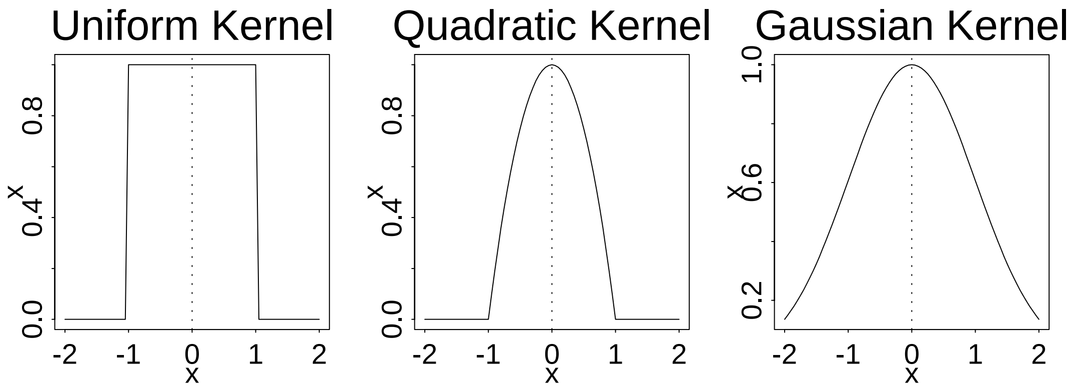

Figure 20: Three commonly used kernel functions $K(u)$ plotted as a function of u.

Here are three commonly used kernel functions:

uniform: $K(u) = 1$ if $|u| \leq 1$ and 0 otherwise,

quadratic or Epanechnikov: $K(u) = 1 - u^2$ if $|u| \leq 1$ and 0 otherwise,

Gaussian: $K(u) = \exp(-u^2/2)$.

Each of these is plotted in Figure 20.

We can now define a local average by defining the weights w_{iq} associated with the evaluation point x_q as follows:

$$w_{iq} = \frac{K[(x_i - x_q)/h]}{\sum_{j=1}^{n} K[(x_j - x_q)/h]} \ . \tag{10}$$

The denominator in this expression merely assures that the weights add to 1, so let us focus on the numerator. We see that argument u in the kernel function is the ratio $u_i = (x_i - x_q)/h$. This ratio defines a measure of the displacement between any observed value x_i and the evaluation point x_q. The parameter h, called the *smoothing parameter* or sometimes the *bandwidth parameter*, controls the size of this displacement measure: when h is relatively large, the ratio and hence the argument u_i will only be large for relatively large differences $x_i - x_q$; and when h is relatively small, the ratio will become large for comparatively small differences. Thus, h controls the amount of the data which is substantially weighted, and therefore that plays a role in defining the weighted average Ave_w.

We are now in a position to offer a complete definition of our kernel smoothing estimate $\hat{f}(x_q)$:

$$\hat{f}(x_q) = \sum_{i=1}^{n} w_{iq} y_i = \frac{\sum_{i=1}^{n} K[(x_i - x_q)/h] y_i}{\sum_{j=1}^{n} K[(x_j - x_q)/h]} . \tag{11}$$

Figure 21 displays the kernel smoothing functions defined by each of the kernel functions defined above. The value of the smoothing parameter h used for the uniform and quadratic kernels was 0.15, and $h = 0.1$ for the Gaussian kernel. What differences between the functions $\hat{f}$ do you notice?

We now consider more carefully the role that the smoothing parameter h plays in determining $\hat{f}$. Have a look at Figure 22. This shows the same data that appeared in Figures 19 and 21, using the Gaussian smoothing kernel, but now the dashed line indicates the function

$$f(x) = \cos(2\pi x^2)$$

used to generate the data. At each value x_i an error value e_i was added to $f(x_i)$ sampled randomly and independently from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.2$. We can now think in terms of how well the smoothing function $\hat{f}$ estimates the true function f.

The smoothing parameter h controls two mutually opposed factors:

Bias: The bias at evaluation point x_q is the difference $\text{E}[\hat{f}(x_q) - f(x_q)]$. The smaller h, the smaller this bias will tend to be, because the principal source of bias is the use of data values y_i associated with argument values x_i for which the true function values $f(x_i)$ are substantially different from $f(x_q)$. The bigger the displacement, $x_i - x_q$, the more of a problem this is going to be, especially in regions where f has high

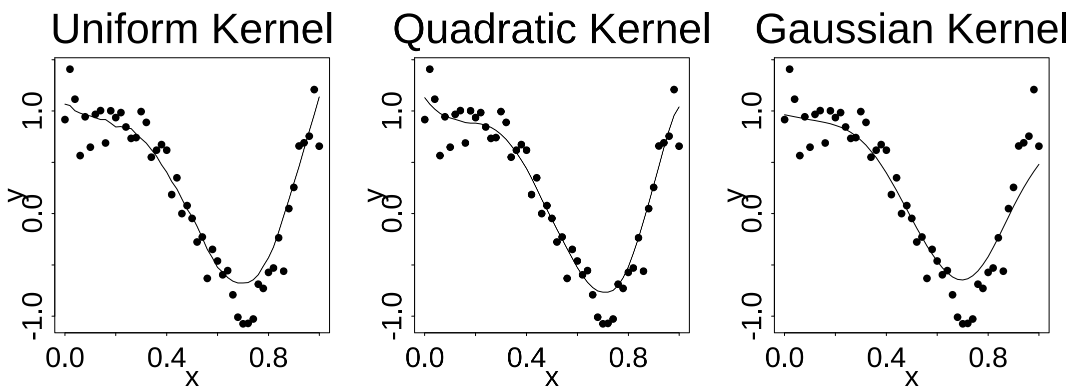

Figure 21: Kernel smoothing estimates $\hat{f}$ for three kernel functions.

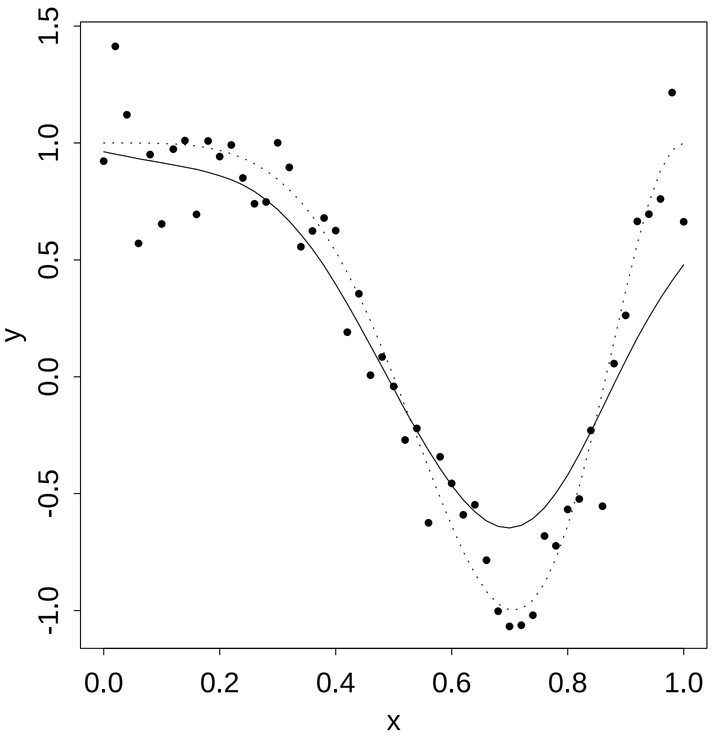

Figure 22: The function $f(x) = \cos(2\pi x^2)$ used to generate the data is indicated by a dotted line, while the Gaussian kernel smoothing estimate $\hat{f}$ is indicated by the solid line.

curvature such as on the right of Figure 22. Going for low bias therefore argues for small values of h.

Sampling Variance: We know that the sampling variance of an arithmetic mean of n observations drawn from a population with variance σ^2 is σ^2/n. That is, the sample variance goes down as the number of observations increases. The same holds for weighted averages, except that the sample variance depends on the number of observations receiving weight substantially greater than zero. Now, since $\hat{f}(x_q)$ is a weighted average, the larger the number of neighboring values over which the average is effectively taken, the less sensitive this value is going to be to errors in the values. This argues for larger values of h.

It is really mean squared error

$$\mathrm{MSE}[\hat{f}(x_q)] = \mathrm{E}[\{\hat{f}(x_q) - f(x_q)\}^2]$$

that we want to keep small. MSE, bias, and sampling variance are simply related by the equation

$$\mathrm{MSE} = \mathrm{Bias}^2 + \text{Sampling Variance}.$$

It turns out that MSE is minimized in a wide range of situations by letting h be proportional to $n^{-1/5}$. This implies that h does not decrease very rapidly as n increases:

$$\begin{aligned} 10^{-1/5} &= 0.63 \\ 100^{-1/5} &= 0.40 \\ 1000^{-1/5} &= 0.25 \\ 10000^{-1/5} &= 0.16\ . \end{aligned}$$

This does not help much for specific situations, but often a sensible value for h can be found by trying out various values and observing the amount of smoothing that they imply. Figure 23 shows how the smoothing function varies with h in this example using a Gaussian kernel. There are also data-driven procedures for determining the best smoothing parameter, but this is beyond the scope of this discussion.

In summary, kernel smoothing offers an easily understood and computationally efficient technique for estimating the values of a function directly, without using specific models defined in terms of parameters. Function estimation by smoothing is especially valuable where there are no strong reasons

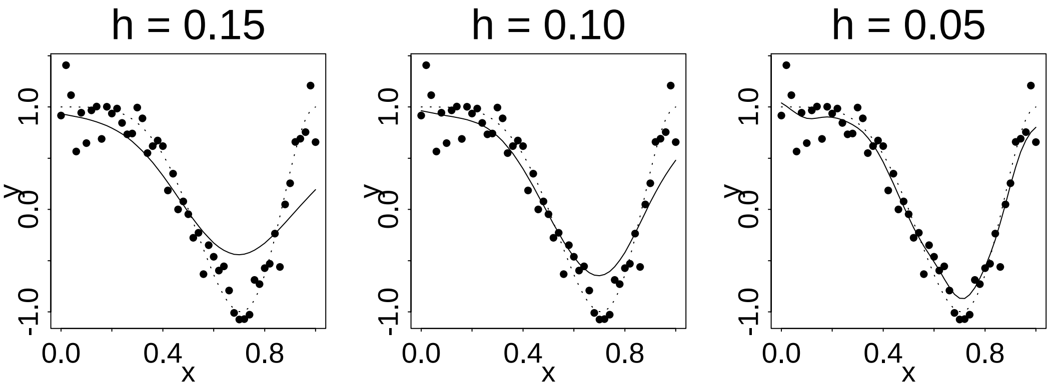

Figure 23: Smoothing with a Gaussian Kernel for Three Values of h The dotted curve is the true function f and the solid line is the kernel smoothing estimate $\hat{f}$.

for supposing that the function will have specific characterstics that can be modeled by a known parametric family of curves. But even where this may be the case, kernel smoothing can often be an attractive estimation technique because of its simplicity and computational speed.

6.2 Applying Kernel Smoothing to Estimating P_{im}

In our problem, the independent variable is proficience or latent trait value θ, and the dependent variable is the probability of choosing option m for item i. What the data offer are the actual choices. These can be summarized numerically by defining a new variable, called an *indicator variable*, as

$$y_{ima} = \begin{cases} 1 \text{ if examinee } a \text{ chooses this option,} \\ 0 \text{ if examinee } a \text{ does not choose this option .} \end{cases} \tag{12}$$

If we can tentatively associate an proficiency value θ_a with each examinee, then smoothing the relationship between these binary 0-1 values and examinee abilities will tend to estimate the probability function $P_{im}(\theta)$.

We proceed as follows:

Rank: Estimate the rank r_a of the a^{th} examinee by ranking the values X_a of some statistic X, which will usually be the total score on the exam

Enumerate: Replace the ranks r_a by the quantiles of the standard normal distribution. These quantiles are the values dividing the area under the standard normal density function into $N+1$ equal areas of size $1/(N+1)$. We use these values as the trait values $\theta_a, a = 1, \ldots, N$.

Sort: Sort examinee response patterns $(X_{a1}, \ldots, X_{an})$ by the estimated proficiency rankings. That is, the a^{th} response pattern in the sorted test, $(X_{(a)1}, \ldots, X_{(a)n})$, is that of the a^{th} examinee by the size of θ_a.

Smooth: For the m^{th} option of the i^{th} item, estimate P_{im} by smoothing the relationship between the binary item-option indicator vector y_{ima} of length N and the proficiency vector $\theta_1, \ldots, \theta_N$. That is, we use the estimate

$$\hat{P}_{im}(\theta_q) = \sum_{a=1}^{N} w_{aq} y_{ima} = \frac{\sum_{a=1}^{N} K[(\theta_a - \theta_q)/h] y_{ima}}{\sum_{a=1}^{N} K[(\theta_a - \theta_q)/h]} \ . \tag{13}$$

> In this smoothing phase, it turns out that the smoothing parameter h can usually be set to something fairly close to $h = N^{-1/5}$. TestGraf uses by default $h = 1.1N^{-1/5}$.

However, TestGraf will often be applied to data for which the number of examinees is tens or hundreds of thousands, so that even this simple linear operation can be time-consuming, especially since it must be repeated for each option within each item. The following grouping or binning technique sacrifices little in accuracy and greatly speeds up the computation.

Assume that one wants an estimate of the option characterstic curve at display value θ_q. Instead of using the indicator values, TestGraf computes the average of the indicator values for values of θ_a falling within the limits $(\theta_{q-1} + \theta_q)/2$ and $(\theta_q + \theta_{q+1})/2$; that is, between the centers of the adjacent intervals, $[\theta_{q-1}, \theta_q]$ and $[\theta_q, \theta_{q+1}]$. For the smallest display value, the average is taken of values falling below the center of the first interval, and for the largest display value, the average is of values falling above the center of last interval. Indicate these Q averages by $\hat{p}_{imq}$. At the same time, the area under the standard normal curve between these interval centers is also computed; denote these areas by ϕ_q. The average binned values are then smoothed by the equation

$$\hat{P}_{im}(\theta_q) = \frac{\sum_{r=1}^{Q} \phi_r K[(\theta_q - \theta_r)/h]\hat{p}_{imr}}{\sum_{r=1}^{Q} \phi_r K[(\theta_q - \theta_r)/h]} \ . \tag{14}$$

Note that the summation is now over the much smaller set of display values (by default 51) rather than over the potentially enormous number of examinee indices. The binning or grouping process is, of course, very fast relative to the smoothing process. Note also, that the weights

$$w_{qr} = \frac{\phi_r K[(\theta_q - \theta_r)/h]}{\sum_{r=1}^{Q} \phi_r K[(\theta_q - \theta_r)/h]} \tag{15}$$

do not depend on the item or option, so that a matrix of order Q containing their values can be computed initially and then used for all curves. The use of binning prior to smoothing is discussed in detail in Härtle (1990), and he refers to the process as *warping*.

This smoothing approach to estimating P_{im} has a number of important advantages over parametric or semiparametric approaches:

1. It is very fast. Since the smoothing process involves simply taking a linear combination of observed values, which from a computational perspective is a very simple operation, computation tends to go from 500 to 1000 times faster than is typical for standard programs like LOGIST or BILOG.

2. Most previous approaches have concentrated on just estimating the characteristic curve for the correct option, thus forcing only right-wrong scoring. There is considerable information in choices among wrong answers for examinees, as we shall see later, and thus this approach is able to extract more information from the test about examinee proficiency, as well as provide useful information about how choices among wrong answers vary with proficiency.

3. It is a simple matter to estimate confidence limits for the value of a curve at any proficiency value θ. An estimate of the standard error of the estimated curve value $\hat{P}_{im}(\theta_q)$ is given by

$$\hat{s}(\theta_q) = \sum_{a=1}^{N} w_{ima}^2 \hat{P}_{im}(\theta_a)[1 - \hat{P}_{im}(\theta_a)] \ . \tag{16}$$

4. One is not constrained to have characteristic curves which can only exhibit features permitted by parametric models such as the 3PL model. That is, for example, characteristic curves can be nonmonotonic, meaning that they can both decrease and increase.

5. Simulation studies have shown that even when the data are sampled from a population where the curves belong to the 3PL family, the estimated characteristic curves have MSE's as good as those obtained by maximum likelihood estimation using the appropriate parametric models. That is, one appears to lose nothing in terms of efficiency of estimation.

A weakness of the approach is its use of total score to rank examinees. As we already know, total number correct is both a biased and inefficient estimate of examinee proficiency. However, since what is being estimated in this method is not proficiency but merely proficiency rank order, and since the smoothing process tends to average out modest errors in ranking, this limitation appears to not seriously affect the estimates of the characteristic curves provided the test has at least 15 or so items. On the other hand, when the items are of the scale type, excellent estimates of option characteristic curves have been obtained with as few as three items.

Once the characteristic curves are estimated for each item and option, one then proceeds to estimate the proficiency of the examinees based on these curves. These estimates can then be fed back into the process as a basis for ranking, and the curves re-estimated. This loop or iteration can be continued until the estimated curves and abilities do not change appreciably. Experience shows that this happens after only two or three iterations. Further details of this iterative refinement process are given in Section 9.

6.3 Other Details

Some additional comments are as follows:

1. The appropriate size of the bandwidth parameter h depends on the number of examinees or respondents, N. This version uses by default the value
$$h = 1.1N^{-0.2}$$

2. Confidence limit estimates are necessarily only approximations, since they are based on asymptotic behavior of the curves, and do not take into account bias in the estimate of the curve itself. The same smoothing process used to compute curve values can be used to compute these estimates, but is itself only an approximation to a more complete estimation procedure.

3. The estimation of the test information function presents many serious statistical problems, and the best approach to its estimation is still being researched by the author. The current procedure uses a parametric model for the option characteristic curves. The parameters of a logistic-quadratic model are estimated from the smoothing probability function estimates by a crude but fast regression procedure, and these are then used to compute the test information function. This is the best we have to offer at the moment, but be aware that any estimates will have a substantial sampling variation, and these estimates can be substantially biased. They are intended to give only a general idea of

the information content of the test, and then only over ranges of θ for which there are substantial numbers of observations.

4. The FORTRAN source code for TestGraf is distributed so that anyone curious about the inner workings of the program can see what is going on, and perhaps even experiment with modifications. Inquiries are welcomed, and the author can be reached at the electronic mail address

   `ramsay@psych.mcgill.ca`

   None of the source code is viewed by the author or anyone else as proprietary, and it can be used as desired. It would be unethical, of course, to redistribute any part of the code commercially for profit.

5. Although the current version of TestGraf uses kernel smoothing to estimate option characteristic curves, other smoothing procedures can also be used, and will probably play some role in future versions. Of these alternative smoothing techniques, the two most important appear to be

   - local polynomial smoothing. Hastie and Loader (1993) offer a readable and persuasive account of the advantages of this approach over kernel smoothing.
   - spline smoothing. Although rather more technical, spline smoothing promises to permit useful estimation of the derivatives of option characterstic curves as well as the curves themselves, which is essential to nonparametric estimation of the information function.

7 How to Produce PostScript Laser Printer Plots

It is possible to produce a high resolution plotted output of the displays in TestGraf using a laser printer that is able to process the PostScript plotting language.

The production of a file with PostScript commands appropriate to be sent to a printer is a two stage process. In the first stage, a set of high-level graphics commands are stored in a file with the fixed name `testgraf.pwo`. The extension `.pwo` indicates that this or any other file contains these commands. In the second stage these commands are processed by the program TestLasr to produce a file with the fixed name `testgraf.ps`. It is this file that is directed to the printer.

There are two procedures for producing the high-level graphics command language file `testgraf.pwo`. The first is to select the menu item `PostS` in the display control box while running TestGraf. The next and subsequent displays will also result in the appropriate graphics commands being generated by TestGraf. This action is toggled off by selecting this item again, and so on. This technique is very useful if only a few displays are to be recorded.

The second procedure involves generating all of the displays produced by TestGraf (except the dynamic displays) in one easy step. This involves invoking TestGraf and selecting the `Only output .pwo plotting file` logical option.

File `testgraf.pwo` will then contain a long series of plotting commands and values to plot.

TestGraf does have one option when running in this mode: to save paper for long tests or scales, the program defaults to producing six plots of the option characteristic curves per page for tests, and three for scales. But you can, if desired, have the program take up an entire page for each plot.

The plotting commands output by TestGraf are not themselves in the PostScript language. Instead, they are in a language developed at McGill University specifically for the types of plots produced by TestGraf. These commands must now be processed by an interpreter program to convert them to PostScript. The conversion of these commands into PostScript is done by the program TestLasr. Again, this program is invoked by simply typing `tl`. The result is a rather long file with the name `testgraf.ps` containing

PostScript commands.

This file `testgraf.ps` must then be directed to a PostScript laser printer by whatever means is appropriate on your system. For example, if the PostScript printer is connected to the parallel communications port `lpt1`, then one would type

```
print testgraf.ps
```

to produce the plots.

The file with the `testgraf.pwo` extension contains plotting commands in an intermediate plotting language that is fairly easy to interpret. This file is in standard ASCII format, and can be processed using any editor.

The commands fall into two categories:

- page setup or layout commands
- graphing commands

The purpose of the page layout commands is to prepare for plotting by specifying the characteristics of the page on which the plot is to be placed. Consequently, these commands use *page* or *physical coordinates*, and the unit is inches.

The graph setup commands produce the actual plot. But the coordinate system is that of the figure itself, which may be in entirely different units. For example, the program producing the commands may be setup to plot a figure in the unit rectangle with lower left position (0,0) and upper right position (1,1), even though the final plot is supposed to occupy a square 6 inches on a side on the paper. Consequently the coordinate system for the graphing commands is called the *graph coordinate system.* One of the purposes of the layout or setup commands is to specify the relationship between the two coordinate systems.

The graphing commands are divided into those that set up the graph, those that produce the graph itself, and those that control style.

Each page may contain multiple graphs, and the graphing commands for each graph must therefore be inside a pair of BeginPage and EndPage commands, and following the page setup commands. The general structure of a plotting job for a single page is as follows:

```
BeginPage
     .
BeginGraph
     .
EndGraph
     .
BeginGraph
     .
EndGraph
EndPage
```

This structure is repeated for each page.

The commands must use upper and lower case characters exactly as specified below. All commands must start at the beginning of the line. There may be blank lines. Comment lines begin with the percent sign, (`%`).

Most but not all commands require arguments. Numerical arguments must be separated by one or more spaces from each other and from the command. They may contain decimal points if necessary, but need not. As many lines as desired may be used.

Text or string arguments must be surrounded by quotes (`"`) and anything within the pair of quotes is considered part of the string, even if blank. If nothing is to be plotted, then there must be a pair of adjacent quotes (as in `""`). Multiple strings must be separated from each other and from the command by one or more spaces.

Some commands have a fixed number of arguments, and some have a variable number. Where there are a variable number, these are terminated by a semicolon.

Numerical arguments often specify coordinates. In this case, there must be two, and the x-coordinate or width is first.

7.1 Setup or Layout Commands

7.1.1 Page Setup Commands

These commands set up the plotting region on the page. They must occur before the BeginGraph command that sets up a graph. The coordinates used in several of these commands are those of the page, and are in inches.

BeginPage Specify the beginning of a new page. This must be the first command in a file, but the file may contain other BeginPage commands as well.

Arguments: None

Example: `BeginPage`

EndPage Specify the end of a page. This must be the last command in a file, but the file may contain other EndPage commands as well. In effect, this command is an instruction to the printer to actually plot the page.

Arguments: None

Example: `EndPage`

PageOrientation Specify whether the long side of the page is considered vertical (Portrait) or horizontal (Landscape).

Arguments: Portrait or Landscape

Example: `PageOrientation  Portrait`

PageSize Specify the physical size of the region to contain the plot on the paper. This cannot exceed the size of the paper used by the printer, which is normally 8.5 in. by 11 in. If the PageOrientation is Landscape, then the upper limits of the region would be 11 and 8.5; if Portrait, then 8.5 and 11. But normally the plotting region would be smaller than the physical size of the page itself.

Arguments: The width and height of the region in inches.

Example: `PageSize  6.5 9`

LeftMargin Specify the left margin between the left physical edge of the paper and the beginning of the plotting region. Note that the sum of the left margin and the width of the plotting region should not exceed the physical width of the paper, where width depends on whether PageOrientation is Landscape or Portrait.

Arguments: The left margin in inches.

Example: `LeftMargin 1`

BottomMargin Specify the bottom margin between the bottom physical edge of the paper and the bottom of the plotting region. Note that the sum of the bottom margin and the height of the plotting region should not exceed the physical height of the paper, where height depends on whether PageOrientation is Landscape or Portrait.

Arguments: The Bottom margin in inches.

Example: `BottomMargin 1`

7.1.2 Graph Setup Commands

These commands setup the graph within the plotting region. Remember that there may be multiple plots per plotting region or page, so these commands locate each graph within the region. These commands begin with the BeginGraph command and end with the EndGraph command, and must be contained within a pair of BeginPage and EndPage commands.

BeginGraph Specify the beginning of a new graph. There may be multiple graphs per page, so in effect this command is an instruction to set up the characteristics of a new graph, including possibly a new relationship between the page and graph coordinates.

Arguments: None

Example: `BeginGraph`

EndGraph Specify the end of a graph. This command will be the second last command in a file, but there may be multiple EndGraph commands within a page.

Arguments: None

Example: `EndGraph`

LowerLeftCorner Specify the origin of the graph within the plotting region. This is the point specified in inches within the plotting region from which the graph extends to the right and up.

Arguments: Horizontal and vertical position in inches.

Example: `LowerLeftCorner   4.5 4.0`

GraphSize Specify the size of the graph within the plotting region. It is this command combined with the GraphCoordinates command that specifies the relationship between the physical coordinates and the graph or plotting coordinates.

Arguments: Width and height of graph in inches.

Example: `GraphSize   4.5 4.0`

GraphCoordinates Specify the range of the x-coordinate or abscissa and the range of the y-coordinate or ordinate. These ranges are in the units of the graph rather than in inches. Each range is mapped on the corresponding physical range specified by the GraphSize command.

Arguments: Two x-values, the second larger than the first, and two y-values also in increasing order.

Example: `GraphCoordinates   1.5  8.5    1.65  7.65`

7.1.3 Graph Style Commands

These commands determine the font and line characteristics. They may occur anywhere after a BeginPage command, and there may be many of these, depending on how often fonts and line styles need to changed. The fonts and line characteristics set by these commands remain in effect until the next command that changes them.

SetFont Specify the font to be used in plotting characters and text. The font has a size specification and a style specification. The style is any style recognized by the PostScript printer, and is in the PostScript language. A PostScript manual should be consulted for the possible styles. Some common font styles are: Helvetica, Courier, Times, and Times-Roman. Bold or Narrow following a hyphen determines the weight of the lines in the characters.

Arguments: A numerical argument specifying the size, and a PostScript style specification determining the style.

Example: The following produces largish bold-face Swiss style text.

```
SetFont 20 Helvetica-Bold
```

The following produces small fine Times style text.

```
SetFont 12 Times-Narrow
```

LineWidth Specify the width of plotted lines in numbers of the printers unit, the em. A size of 0.5 gives a fairly fine line, and a size of 1.0 gives a line of medium boldness.

Arguments: A single numerical argument specifying the width.

Example: `LineWidth   1`

SetDash Specify the type of alternation between white space and line in a dashed line. This is essentialy the "setdash" command of the PostScript language, and a manual should be consulted for full details. The number of arguments is variable, but must be at least one. The second and subsequent arguments specify a cycle pattern between "on" line drawing and "off" line drawing, and thus can be used to set up complex patterns of white space and lines of varying lengths. The first argument is the number of units into the line taken before the cycle begins. The line drawing begins at that point as if the cycle had been repeated from the beginning. If this sounds complex, most dashed lines will require only two arguments: the first is 0 and the second indicates the length of the line segment and the intervening white space that is desired. Arguments 0 and 3, for example, produce a medium dashed line. If there is only one argument, the dashing is turned off and the line is plotted as solid. Don't forget the semicolon!

Arguments: A variable number of numerical values ended by a semi-colon.

Examples: The following sets up a solid line:

```
SetDash   0;
```

The following sets up a dashed line with medium length dashes:

```
SetDash   0 3;
```

The following sets up a dashed line with longer dashes:

```
SetDash   0 6;
```

The following sets up a dashed line alternating between medium and long length dashes:

```
SetDash   0  6 3 3 3;
```

7.1.4 Graph Production Commands

These commands actually do the work. They must follow a BeginGraph command and precede a EndGraph command, and what they do is determined by the graph setup commands and the graph style commands.

DrawAxes Draw a set of axes either as a box around the plot (argument Box) or without axes but with two arrows indicating the directions of increase of the two coordinates (Crossing).

Arguments: One argument that is either Box or Crossing.

Examples: `DrawAxes Box`

```
DrawAxes Crossing
```

CoordinatesX Plot tick marks and values at specified x-values on the abscissa.

Arguments: A variable number of pairs, each containing an x-value followed by a character string enclosed in quotes. The sequence of pairs is terminated by a semicolon.

Example: `CoordinatesX`

```
0.0 "0.0"
0.4 "0.4"
0.8 "0.8"
1.2 "1.2";
```

CoordinatesY Plot tick marks and values at specified y-values on the ordinate.

Arguments: A variable number of pairs, each containing a y-value followed by a character string enclosed in quotes. The sequence of pairs is terminated by a semicolon.

Example: `CoordinatesY`

```
0.0  "None"
0.5  "Half"
1.0  "All";
```

LabelX Plot a label for the abscissa. The label consists of three strings: the first plotted at the extreme left, the second centered, and the third at the extreme right.

Arguments: Three strings enclosed in quotes.

Example: This command plots a centered label.

```
LabelX "" "Time" ""
```

This command plots a label on the extreme left.

```
LabelX "" "" "Time"
```

LabelY Plot a label for the ordinate. The label consists of three strings: the first plotted at the bottom, the second centered, and the third at the top. The labels are plotted parallel to the axis.

Arguments: Three strings enclosed in quotes.

Example: This command plots a centered label.

```
LabelY "" "Acceleration" ""
```

Line Plot a line segment from an initial to a final point.

Arguments: Four numerical arguments: the x- and y-coordinates of the initial point and the x- and y-coordinates of the final point.

Example: `Line    0.2 0.3  0.4 0.8`

PolyLine Plot a line through a series of points specified by a sequence of (x,y) coordinate pairs.

Arguments: A variable length set of pairs of x- and y-coordinates terminated by a semicolon.

Example: `PolyLine`

```
0.2 0.3
0.4 0.8
0.6 1.2
;
```

XYPolyLine Plot a line through a series of points specified by a sequence of x-coordinates followed by a sequence of y-coordinates

Arguments: A variable length set of x-coordinates followed by a sequence of the same length of y-coordinates. The two sequences are terminated by a semicolon.

Example: The data in the PolyLine example above would be give to XYPolyLine as follows:

```
XYPolyLine  0.2 0.4 0.6   0.3 0.8 1.2 ;
```

LabelAtPoint Plot a string of characters at a point.

Arguments: A character string enclosed in quotes followed by the x- and y-coordinate at which the plotting is to begin. The characters are plotted horizontally.

Example: `LabelAtPoint   "TestGraf"  0.5 0.5`

VerticalReferenceLines Plot a series of dashed vertical lines at a set of x-axis positions.

Arguments: A variable number of numerical values terminated by a semicolon.

Example: `VerticalReferenceLines  0.2  0.5  0.8;`

HorizontalReferenceLines Plot a series of dashed horizontal lines at a set of y-axis positions.

Arguments: A variable number of numerical values terminated by a semicolon.

Example: `HorizontalReferenceLines 0.2 0.5 0.8;`

CircularArc Draw a circular arc with specified center, radius, and range of angles.

Arguments: Five numerical arguments: the x- and y-coordinates of the center of the circle, the radius of the circle, the angle in degrees measured counterclockwise from the horizontal at which the arc begins, and the angle at which it ends.

Example: This command draws an arc with center at (5,5), radius 2, through the from 0 to 90 degrees.

```
CircularArc  5 5  2  0 90
```

This command draws a circle with center at (1,1) and radius 1.

```
CircularArc  1 1  1  0 360
```

EllipticalArc Draw an elliptical arc with specified center, a radius for the x-direction, a radius for the y-direction, an angle through which the ellipse is to be rotated, and range of angles.

Arguments: Seven numerical arguments: the x- and y-coordinates of the center of the circle, the two radii for the x- and y-directions before rotation, an angle through which the ellipse is to be rotated counterclockwise, the angle in degrees measured counterclockwise from the horizontal at which the arc begins, and the angle at which it ends.

Example: This command draws an ellipse with center at (5,5), radius 2 in the x-direction before rotation, radius 1 in the y-direction, then rotated through 45 degrees.

```
EllipticalArc   5 5  1 2   45   0 360
```

Draw a circle with center at (1,1) and radius 1.

```
CircularArc  1 1  1  0 360
```

7.2 The Translation Program

The program TestLasr that translates these commands into PostScript commands is written in portable C code, and is available on request from the author.

The translation program requires two files of PostScript code called `header.ps` and `pwplot.ps` to be in the directory where the program is executed. These files provide the initial header lines of the PostScript file and some further PostScript code required by the output file.

If printed versions of multiple sets of data are to be produced, and one wishes to save the files, the `testgraf.pwo` file will have to be renamed prior to the next invocation of TestGraf.

8 How to Compare Results from Several Groups

8.1 Differential Item Functioning or DIF

One often wishes to explore ways in which two or more groups differ with respect to their responses to a specific test or scale. Broadly speaking, there are roughly two ways in which groups can differ:

1. in terms of the distribution of proficiency. That is, for example, a specific group may have both observed and latent scores whose mean and other aspects of the distribution differ from what other groups produce. This can happen for exams, for example, when a particular group comes from a disadvantaged educational background.

2. in terms of probabilities of choosing among options even when observed and latent scores are made equal. This is often called *differential item functioning*, or *DIF* in the psychometric literature. This can happen, for example, when one group tends to read or interpret the options in a different way than another, or is disadvantaged with respect to the specific *item*, but not necessarily with respect to the test as a whole.

It is the second type of variation, differential item functioning or DIF, that is the focus of this section, and of the program TestComp. The first type of variation is not unimportant, of course, but can be readily investigated using standard statistical tools. A recent collection of papers on DIF is Holland and Wainer (1993).

Differential item functioning, then, means that for two or more individuals, each with latent proficiency or trait level θ, the probability of choosing option m on item i, $P_{im}(\theta)$ is not constant, but varies from individual to individual or group to group. This concept can extend also to the omit response.

It is of fundamental importance to separate this type of intergroup response variation, DIF, from the more obvious intergroup variation of type 1. DIF can properly be called *item bias*, and is usually regarded as undesirable, whereas large group differences in score can be understandable, and not necessarily reflect badly on the nature of the test.

As an example, we would expect big differences in how grade five and grade eight students were able to handle a test of proficiency to manipulate fractions, and we wouldn't think badly of the test when these differences showed up. However, if we compare the performance of a very bright fifth grader with a somewhat mediocre proficiency eighth grader, and we know from other evidence that their fraction-handling skills are reasonably identical, then we would not be pleased to see a particular test item show real differences in performance. That is, performance on the item should depend only on how much a student knows about fractions, and not directly on the student's grade or other characteristics. DIF might come about in this case, for example, if the question used sophisticated sentence structure in posing the problem, and thus placed the fifth grade student at a disadvantage unrelated to math competency because of less developed language skills.

8.2 An Overview of How to Compare Subgroups

Before TestComp can be used to compare groups in order to detect DIF, the data for each subgroup must be separately analyzed by TestGraf. This, in turn, means that we have to do something to select only the data belonging to a particular subgroup. To be sure, this can always be done using editing software, or by writing a little program. However, it can also achieved by running TestSave separately for each subgroup in order to set up the corresponding TestGraf analysis. This approach keeps the original raw data file intact, and will usually be less work.

Thus, the first step will usually be to use TestSave to identify and select the responses for a specific subgroup from the master raw data file. The next subsection describes how to do this. Then TestGraf is run for each subgroup, and finally TestComp is run for the group as a whole. An example of this process is given in Section 10, as well as being illustrated in this Section.

On the other hand, if it is more convenient to set up separate subgroup files using an editor or some other type of program, then TestSave and TestGraf can be simply run on each subroup's data in the standard way.

8.3 Running TestSave for Each Subgroup

8.3.1 Subgroup Selection Items

As indicated in Sections 2.5 and 5, a new feature of TestSave is the possibility of selecting a specific subgroup, and producing files containing information for only this subgroup. This is done by designating one or more items as subgroup specification items. These items are not a part of the files produced by TestSave but are only used to select the subgroup.

In order to illustrate subgroup selection and comparison, we shall use data from an administration of the General Management Aptitude Test to 2735 prospective applicants to graduate business schools. The GMAT is developed by Educational Testing Service, and the data considered here are for 25 multiple choice items in a quantitative subscale within the test. For subgroup selection purposes, data are available on whether the examinees were male or female. The responses actually coded were not the original choices, but

- 0: the question was incorrectly answered
- 1: the question was correctly answered
- 2: the question was omitted, but subsequent questions were answered
- 3: the question and all subsequent questions were omitted

The first question posed by TestSave is whether these subgroup selection items are a part of the total number specified on the first line of the raw data file with the `.dat` extension. Suppose, for example, that we want to study the performance of the females only for the GMAT Quantitative subscale. The subscale has 25 items, but a complete record for an individual is as follows:

```
1110011111110111102033333 F
```

Note that the final character is an "F". This character indicates that the examinee associated with this record is a female. To set up the TestSave analysis, we need to indicate that 26 items are to be processed, and in this case that there are no characters of label. Thus the first line of the data file would be

```
26      0
```

The key line for selecting females would be

```
1111111111111111111111111 F
```

TestSave begins by asking if one or more items are to be used to select subgroups. We would answer "yes" to this question. The key line specifies which subgroup is to be used.

TestSave then asks how many subgroup selection items there are. In this case there is only one subgroup selection item. There can be up to twenty, and the selected subgroup must match the key line for all specified items. TestSave then asks for the indices of the subgroup selection items, which in this example would be 26 since the gender identification item is the last.

TestSave then sets up files for a TestGraf analysis for only those items not used to select subroups. In this example, the test actually has 25 rather than 26 items. TestSave also can assign the stems of the file names so that they are specific to this subgroup. This is done by supplying a second argument on the command line invoking TestSave.

This procedure must be repeated for as many subgroups as there are, each time changing the subgroup identification item in the key line. In the example above, a second run of TestSave would be required after changing "F" to "M".

8.3.2 Producing a Common `.itm` File

At this point a particular problem arises: in order for the results for the subgroups to be comparable, it is essential that the options for each item be the same for all groups. This will not automatically be the case, since the file with the extension `.itm` produced by TestSave lists only the options actually occurring in the raw data file that it analyzes. But there may be options that are not chosen by anybody in a particular subgroup.

This problem can be dealt with relatively simply, in two ways:

1. Prior to any TestGraf analysis, designate one of the subgroups as the standard, and simply copy that subgroup's `.itm file` to each of the other subgroups, so that the `.itm` files for the other groups are replaced by the master subgroup's `.itm` file. The most convenient procedure is to designate the largest subgroup as the master subgroup.

Be sure, however, that this master group has all the options listed for each item that are identified for any other subgroup; if not, simply add missing options to the appropriate line of the `.itm file` for the master subgroup before copying.

2. Edit the `.itm` file for each subgroup and add options that are present in some groups but not in that one. There is no harm in having options that are chosen by nobody; TestGraf simply estimates zero probability for choosing them. Be sure to also change the number of options line when an option is added.

8.3.3 Summary of the Procedure for the GMAT Data

To summarize, for the GMAT data the complete procedure could be as follows:

1. set up the first line of the raw data file `gmatq.dat` to be

   ```
   26    0
   ```

2. set up the second key line to be

   ```
   1111111111111111111111111 F
   ```

3. invoke TestSave with the command

   ```
   ts gmatq gmatqf
   ```

   and respond in the appropriate way to the questions asked by TestSave: "yes" to whether there are subgroup items, "1" to the question of how many there are, and "26" to the question of what the index is.

4. Now change the second key line to be

   ```
   1111111111111111111111111 M
   ```

5. Invoke TestSave again, but with the line

```
ts gmatq gmatqm
```

6. (optional) use an editor or simply display the two `.itm` files to ensure that the number of options listed for each item is the same. The options listed will not be in the same order (except for the first), but this doesn't matter. We shall use the male subgroup as the master subgroup. Having assured that each group has all the items listed that appear for the other subgroup, or, if necessary, adding missing options, we now copy the file `gmatqf.itm` to the file `gmatqm.itm` by the command

   ```
   copy gmatqf.itm gmatqm.itm
   ```

   Note, for example, that the `.itm` file for the females for item 9 indicates 4 options, with the characters 0, 1, 2, and 3, while the file for the males for this item indicates only three options with the characters 0, 1, and 2. This is because character 3 indicates "not reached", and no male failed to reach item 9.

7. Finally, two TestGraf analyses are then performed, one with the command line

   ```
   tg gmatqm
   ```

   and the other with the line

   ```
   tg gmatqf
   ```

We recall again that it is by no means essential to use TestSave to define the subgroups. One can simply extract the records by use of an editor or program corresponding to a subgroup, and put these in a separate raw data file. Once this is done for each subgroup, so that one has a raw data file for each subgroup, one then can run TestSave followed by TestGraf for each subgroup in turn. It will be essential between the TestSave and TestGraf analysis to ensure that each item has the same number of options for all subgroups, and that these are specified in the `.itm` file in the same order. This can be done by direct manipulation of one subgroup's `.itm` file, and then copying that file to the corresponding version for each of the other subgroups.

8.4 Running TestComp

TestComp produces five types of displays:

Option Characteristic Curve Plots: For each option within each item, the option characteristic curves are plotted. For purposes of displaying DIF, there is no point to using the normal quantiles as the display variable, but the other display variables available in TestGraf are also available here. Lack of DIF is manifested by option curves being nearly coincident. Of course, what can be viewed as coincident or identical depends on the sample sizes involved. When the number of examinees or respondents is modest, two curves can be expected to differ somewhat due simply to sampling variation in the estimates.

You can either choose the item to be displayed, or step through the items sequentially. Once an item is selected, all the options associated with the item are displayed sequentially. In the case of scale data, the final display for an item is the expected scale score.

Item Characteristic Curve Plots: This is an abbreviated version of the first option. For many purposes it suffices to look at DIF in terms of either the correct answer for multiple choice items, or the expected scale score for scale items. This reduces the number of displays per item to 1.

Pairwise Expected Score Plots: For any pair of subgroups, the expected number correct or total score values associated with the standard normal quantiles are plotted against each other. This plot summarizes differences in performance between the two groups rather than DIF. If, for a particular pair of subgroups, the performance is about the same, the relationship will appear as a nearly diagonal line (a truly diagonal line is plotted as a reference). On the other hand, if the subgroup associated with the vertical axis has an overall inferior level of performance, the relationship will appear below the diagonal reference line.

Dynamic Tetrahedron Plots: The four most active options are used to define tracks or curves in probability space, one for each group. Again, lack of DIF implies that the curves will lie along one another, although not necessarily beginning and ending at the same points. Again, some amount of discrepancy due to sampling variation must be allowed for, especially for extreme proficiency or trait values where there are few cases.

Total Score Distribution Plots: Like the expected score plots, this display is designed not to show DIF but rather differences in performance on the total test. The probability density functions for the groups are shown on the same plot.

TestComp is invoked by typing `tc` or `testcomp` followed by a set of stems, one for each group. If TestComp is invoked without this list, it will prompt you for the stems. It computes the number of groups by counting the number of stems supplied. For example, in the case of the GMAT data, we would type

```
tc gmatqm gmatqf
```

The curves associated with various groups can be given different colors. The colors available depend on the display adaptor used by your particular computer, of course. Indices of colors are stored in a file with the name `testcomp.chr`. For example, for the GMAT analysis invoked as above, the indices might be 12 and 9. These can be stored one per line, or on the same line separated by one or more blanks. On a VGA display adapter, index 12 plots in a pinkish color, and 9 in a blue color.

In any case, the curves are also identified with numbers, corresponding to the order of the groups as identified in the command line. In the above example, curve 1 would be females and 2 would be males.

8.5 Summary Measures of DIF

It is common practice to attempt to summarize the amount of DIF for a specific option characteristic curve by a single number, denoted here by β. TestComp extends this practice somewhat by displaying two types of DIF summary statistics at the right of the option characteristic plots. It is usual in practice to refer to one of the groups as the *reference group*, and to the remainder as *focal groups*. The reference group is usually some standard baseline group with respect to which DIF is to be assessed for usually smaller specific groups. For example, a group designated as "whites" is often used when looking for DIF for blacks. TestComp assumes that the first file named in the command line is the reference group.

A DIF summary index for a comparison between a specific focal group and the reference group is computed in TestComp as follows. Let the proportion of the reference group having display variable value θ_q be indicated by p_{Rq}. And let $P_{im}^{(R)}(\theta)$ and $P_{im}^{(F)}(\theta)$ stand for option characteristic curve values for the reference and focal groups, respectively. Then

$$\beta_F = \sum_{q=1}^{Q} p_{Fq}[P_{im}^{(F)}(\theta) - P_{im}^{(R)}(\theta)] \tag{17}$$

is the index. It measures the expected or average discrepancy between the focal group curve and the reference group curve. If the focal group is disadvantaged on average, this index will be negative. The index β_F is calculated for each focal group and displayed opposite the file stem name for that group in the plot.

When there are multiple focal groups, it is also desirable to have a composite index summarizing the total DIF in all groups. This composite index is

$$\beta_R = \sqrt{(N_G - 1)^{-1} \sum_{q=1}^{Q} p_{Rq}[P_{im}^{(F)}(\theta) - P_{im}^{(R)}(\theta)]^2} \tag{18}$$

where N_G is the number of groups, including the reference group. This index is the a root-mean-square measure of DIF across groups, and will always be positive. It is displayed opposite the reference group file stem name.

When the analysis is complete, the DIF indices along with standard error estimates are put in file `testcomp.out`.

8.6 Comparing Males and Females on the GMAT

We can now see what happens when we compare the performance of two genders on the GMAT Quantitative subscale.

Figures 24 and 25 show the option characteristic curves for the correct answer for items 21 and 22, respectively, on this test. The curve identified with the "1" is for the males, and with "2" for females. There does not appear to be any serious DIF for item 21, but there is for item 22. This latter is a difficult item, of course, but the substantially lower curve for

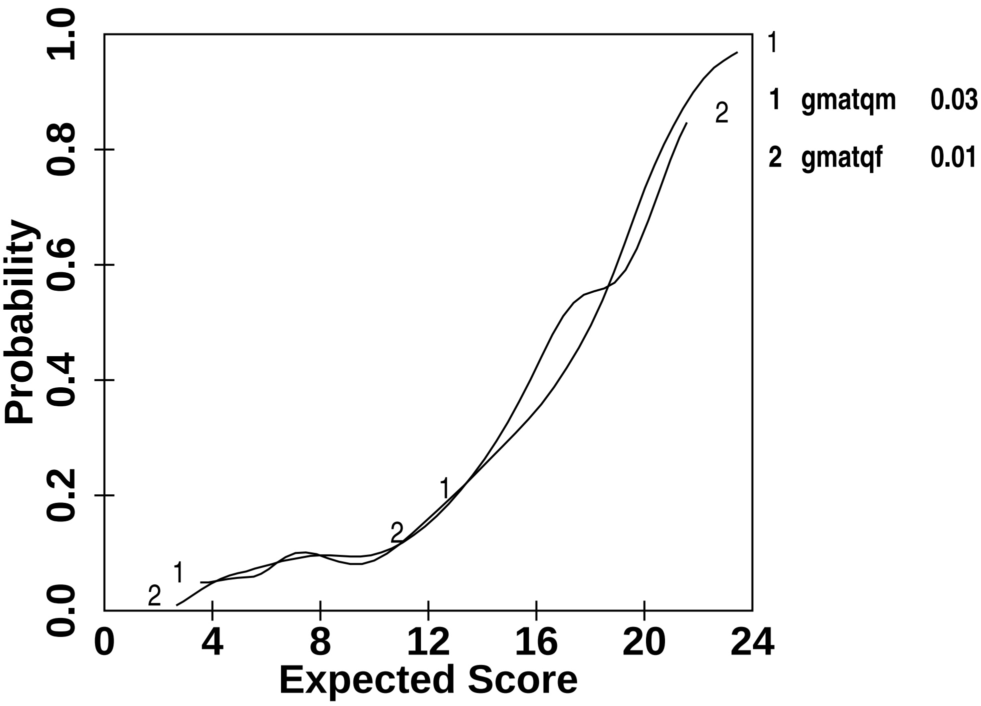

Figure 24: The correct answer characteristic curve for item 21 for the GMAT quantitative subscale. The curve labelled 1 is for males, and the curve labelled 2 is for females. This item exhibits very little DIF, or item bias.

females indicates that the item is disproportionately difficult for females for some reason. This DIF means that, for two students of equal proficiency but different gender, the female is less likely to get the item right, especially if her level of proficiency corresponds to an expected number correct in the 15 range. The difference in probability at that level is about 0.15.

The overall summary DIF statistic β_1 is -0.05. This value is lower because the difference between the curves for any specific expected score is weighted in equation (17) by the probability of getting that expected score for the focal group. We see in Figure 28 that the modal expected score for females is about 10, and at that value and below the curves in Figure 25 are less far apart than they are at 15, a value reached by comparatively few females.

Since the test is fairly short, the presence of DIF in any one item means that the total score on the test cannot be considered a DIF-free or unbiassed basis for assessing proficiency, since the score is contaminated by the DIF in

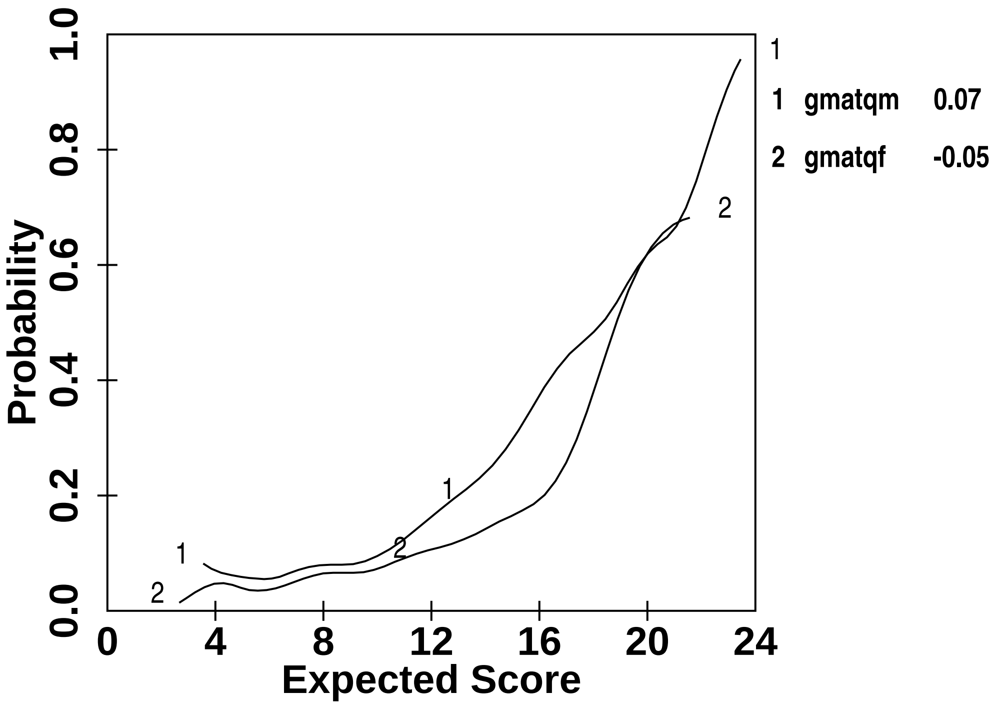

Figure 25: The correct answer characteristic curve for item 22 for the GMAT quantitative subscale. The upper curve labelled 1 is for males, and the lower curve labelled 2 is for females. This item exhibits a substantial amount of DIF, or item bias.

item 22. It would be prudent to re-assess DIF with this item dropped from the test. We do this by taking following the following steps:

1. Run TestSave for both groups but with entry in the key line for item 22 removed and with only 24 items indicated (or 25 if TestSave is used to select the subgroups). One might use new file stems such as `gmatqm24` and `gmatqf24` for these analyses.

2. Run TestGraf for each subgroup for the 24-item test, and ask for computation of maximum likelihood estimates of proficiency by selecting TestGraf analysis option 7. This will produce two new files with names such as `gmatqm24.abi` and `gmatqf24.abi`.

3. Now run TestGraf again for two groups, but using all 25 items, and using the two 24-item `.abi` files to input proficiency values, rather than letting TestGraf use total score to rank examinees. One can input these abilities by either copying the two 24-item `.abi` files to the corresponding files with appropriate stems for the 25-item analyses, or by using an editor to modify the `.par` files to access the 24-item files. In either case, the logical item indicating input abilities must now be toggled on.

4. The TestGraf results for the two groups are now not affected by DIF in item 22, even though the option characteristic curves are computed and displayed for this item. TestComp may now be run to display the DIF for item 22.

Figure 26 shows that the estimated DIF remains roughly the same as in Figure 25 for most proficiency values, but does differ somewhat for very high proficiencies. In fact, we see the DIF in favor of males continuing over the entire upper range of proficiencies, and this seems rather more reasonable.

We can also have a look at how the two groups perform on the GMAT Quantitative scale. Figure 27 shows that if we plot expected number correct for the two groups corresponding to the various standard normal quantile values against each other, the result is a relationship falling well above the diagonal dashed line. Group 1 is the males, and, for example, we can read off the plot that the males getting about 13 correct correspond in quantile position to females getting only about 11.5 correct. That is, the median female is about 1.5 items behind the median male, and, moreover, this deficit

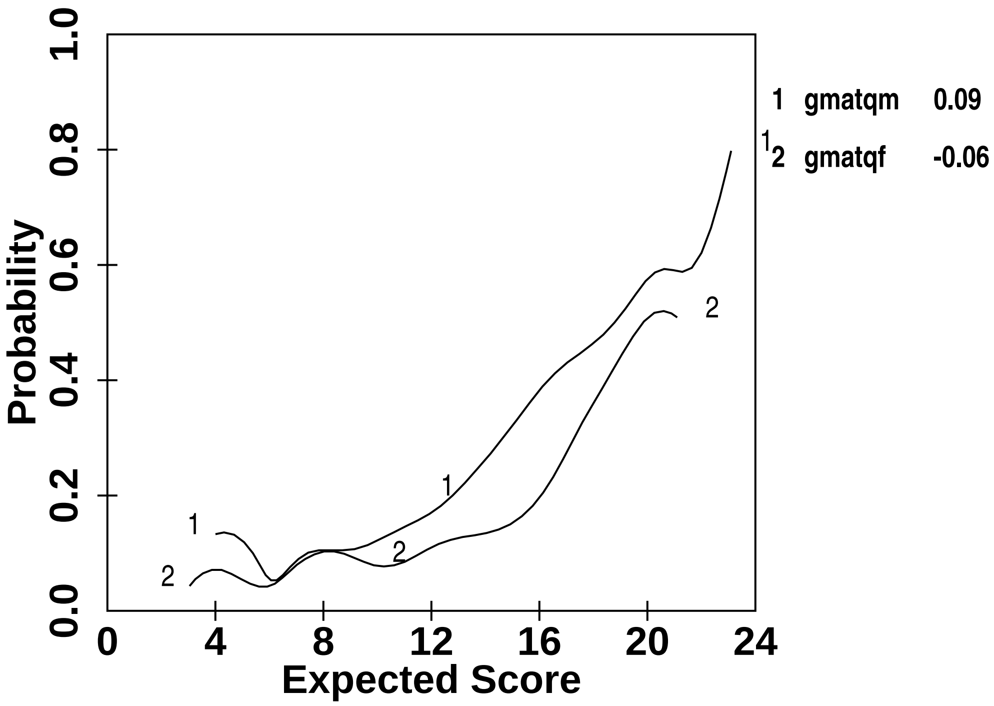

Figure 26: The correct answer characteristic curve for item 22 for the GMAT quantitative subscale. The upper curve labelled 1 is for males, and the lower curve labelled 2 is for females. The influence of item 22 on the estimation of proficiency has been eliminated.

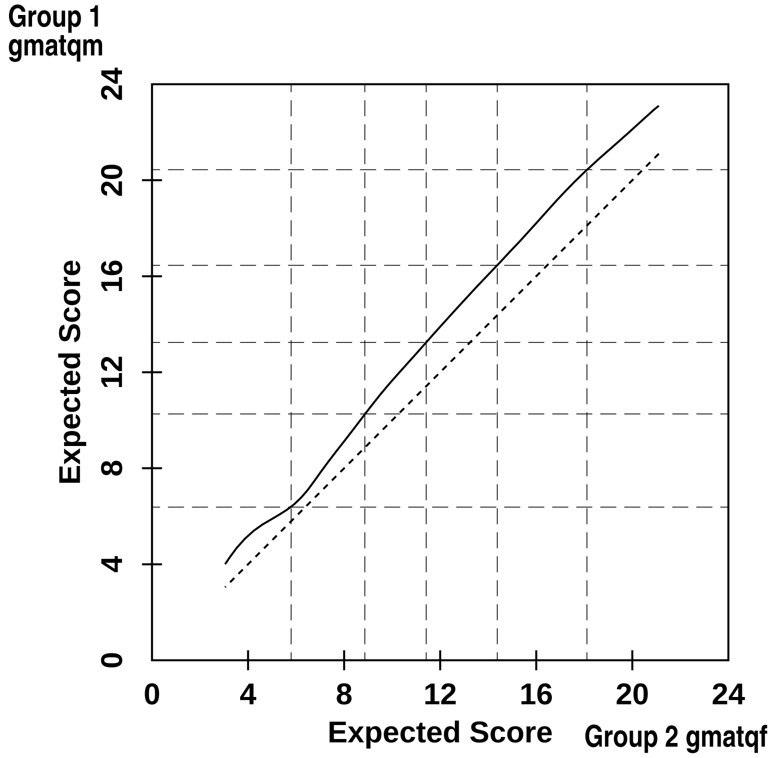

Figure 27: The solid curve shows the relationship between expected number correct for males (Group 1) and females (Group 2) for The GMAT Quantitative Subscale. The dashed diagonal line indicates the relationship were there no difference in performance for the two groups. The horizontal and vertical dashed lines indicate the 5%, 25%, 50%, 75%, and 100% quantiles for the two subgroups, males and females, respectively.

in performance is roughly maintained for most of the quantile levels, although the discrepancy is somewhat less for the lowest ranking examinees.

Another indication of the difference in performance on the test is given in Figure 28, where the distribution of numbers correct is shown for the two groups. There is a difference of about 2 in the modal values for the two groups, and while there are a fair number of males expected to score higher than 20, there are very few females in this high range.

Thus, our conclusion is that, although females do not get scores as high as their male counterparts on the GMAT Quantitative subscale, there is little item bias or discrimination against them within the test, except for item 22. It should be pointed out, however, that the DIF in item 22 is nowhere

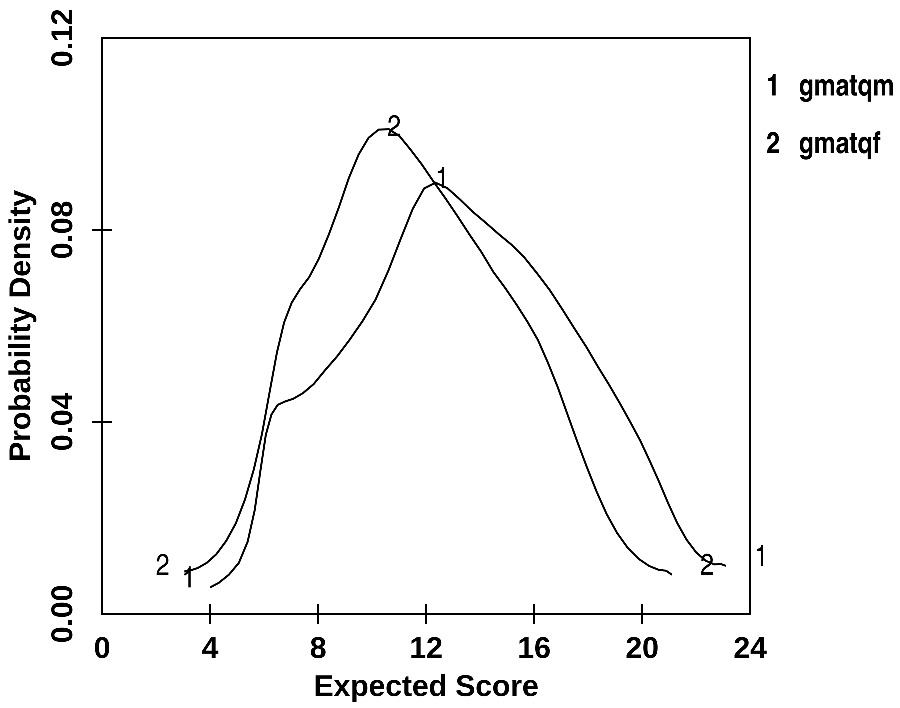

Figure 28: The distribution of expected number correct for males (Group 1) and females (Group 2) for the GMAT Quantitative Subscale.

nearly large enough to account for the poorer female performance, since it would only predict a difference of about 0.15 for examinees who are scoring at the median or above. Also, there is no substantial DIF for very low or very high proficiency levels, while the deficit for the test as a whole remains high. Consequently, we must look elsewhere to explain the discrepancy in overall performance.

8.7 DIF in the Beck Depression Inventory

In this example we look at data from the Beck Depression Inventory with a focus on seeing whether performance on specific items differs for males and females.

The data in this case were kindly provided by Aaron T. Beck, and are described in Beck, Rush, Shaw and Emery (1979). They are the responses of 282 males and 366 females with levels of depression ranging from minimal to severe, we shall refer to these data as the clinical sample for the BDI. An extended TestGraf analysis of the properties of this scale can be found in Santor, Ramsay, and Zuroff (1994).

In this example we focus on item 14, which is

0 I don't feel that I look any worse than I used to.

1 I am worried that I am looking old or unattractive.

2 I feel that there are permanent changes in my appearance that make me look unattractive.

3 I believe that I look ugly.

The result of comparing the two groups for this item for option 0 is in Figure 29, and for the expected score on this item is in Figure 30. We see that there is an obvious difference between males and females for all but the most extreme levels of depression. In general males appear to be rather less concerned with their appearance, and their expected score is lower by about 0.3 over most of the score range. This is primarily due to the fact that males at low to moderate levels of depression are much more likely to choose option 0 than females. A male appears less likely to relate his appearance to his sense of well-being.

There were also differences for Item 10, which was

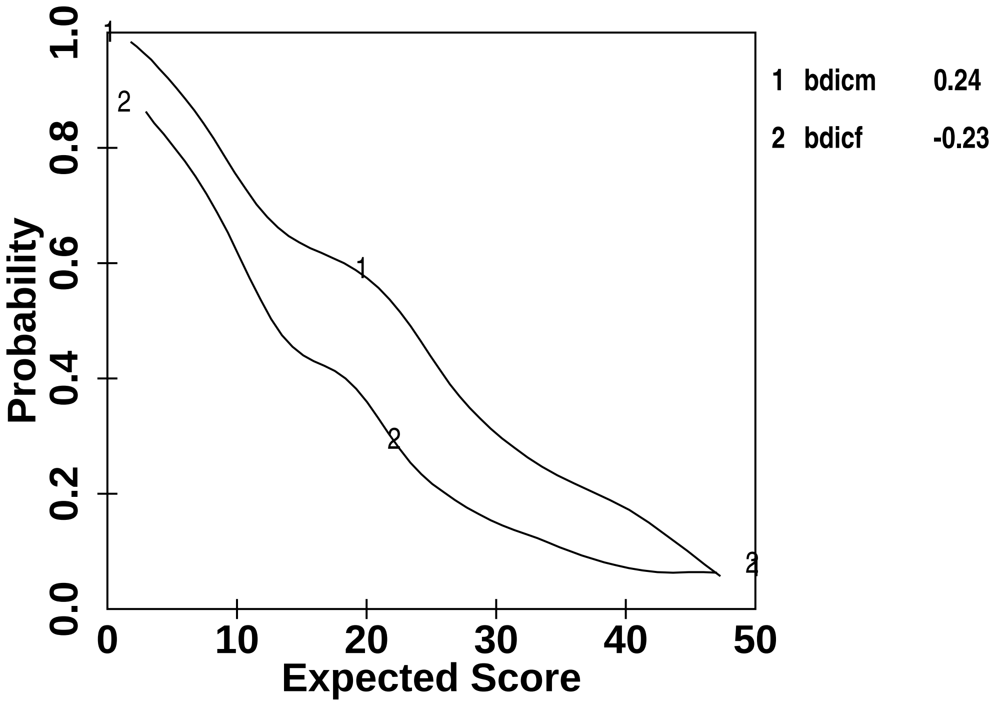

Figure 29: The option characteristic curve for item 14 and option 0 from the Beck Depression Inventory. Option 0 reads, "I don't feel that I look any worse than I used to." The line labelled 1 is for males and the line labelled 2 is for females.

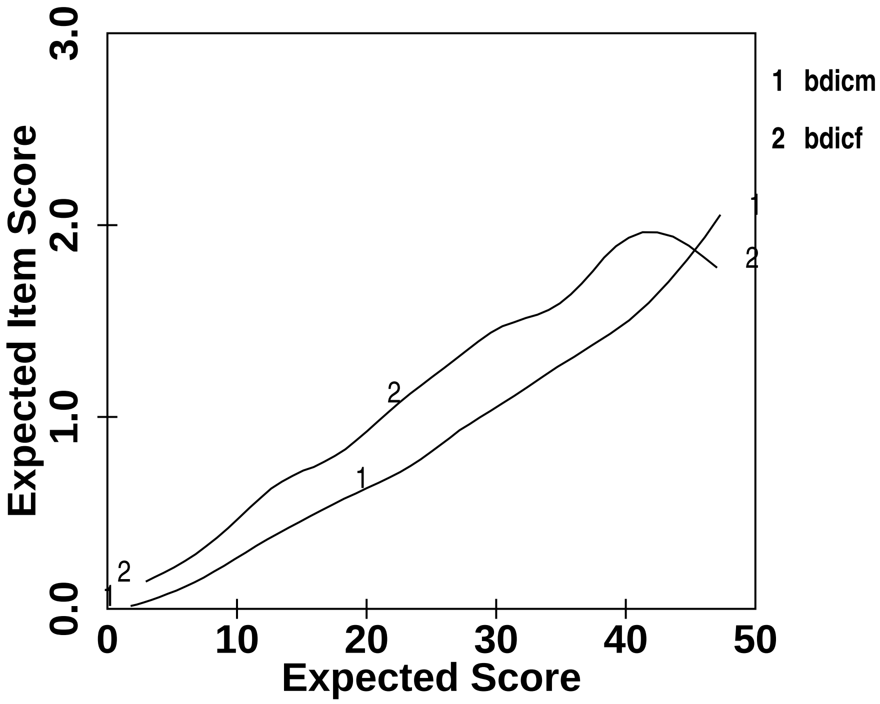

Figure 30: The expected scale score for item 14 from the Beck Depression Inventory.

0 I don't cry any more than usual.

1 I cry more now than I used to.

2 I cry all the time now.

3 I used to be able to cry, but now I can't cry even though I want to.

However, the differences for the remaining 19 items were minimal.

Figure 31 indicates that there is also a tendency for females to have higher BDI scores by about 2 to 3 over males at the same quantile position for levels of depression that are from low to medium. Given the much higher referral rate among females, and the historical tendency to view depression as more of a female problem, this difference is surprisingly small.

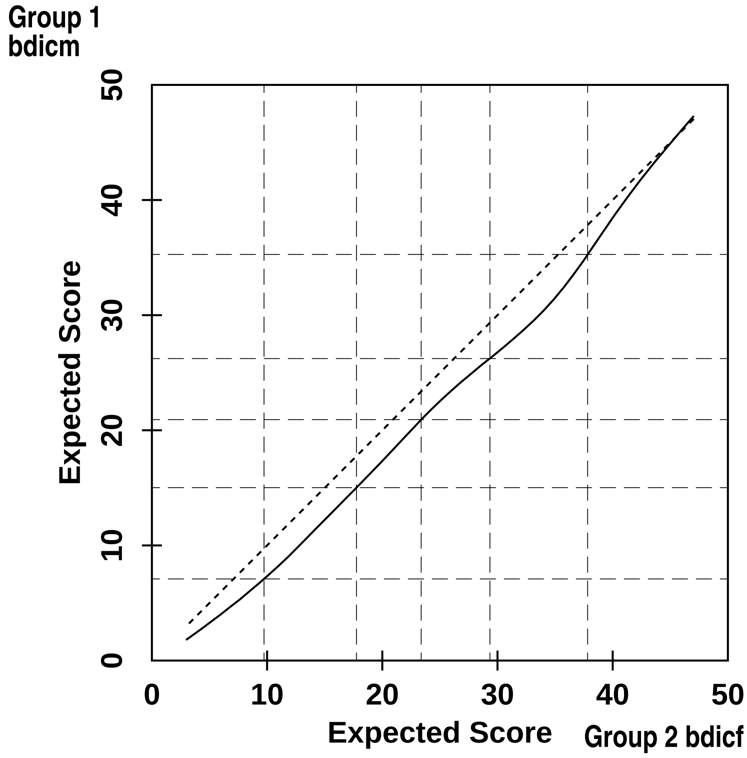

Figure 31: Expected scale scores for the entire Beck Depression Inventory for males and females. Any point on the solid line corresponds to respondents occupying equivalent quantile positions in their samples. The diagonal dashed line indicates identical performance. Vertical and horizontal dashed lines represent the 5%, 25%, 50%, 75%, and 100% quantiles for the two subgroups.

9 Refining Scores by Iterating TestGraf

By default TestGraf orders examinees by using a number correct or scale score computed from each examinee's responses. This score is determined by weights for options provided by the user of TestGraf. In the case of a multiple choice examination, the weights are either 1 for the correct answer or zero for any option which is incorrect. In the case of a psychological scale, the weights are input in the file with the extension `.itm` and are fixed by the designer of the scale.

Once TestGraf has estimated the option characteristic curves, $P_{im}(\theta)$, however, one can consider discarding these weights in favor of estimates of proficiency or trait value θ computed by maximum likelihood estimation (TestGraf analysis option 7). This technique, abbreviated MLE, uses the information in the responses more efficiently than any a priori weighting scheme is likely to do.

One might think of this estimation MLE process as one of optimal rather than fixed weighting of the options. MLE weights option choices by taking into account the following factors:

- the overall quality of the item in terms of the information it provides about θ. That is, some items will yield a lot of information about proficiency or trait level, and others will have only a modest relationship to whatever the scale measures. MLE weights the former items more heavily than the latter.

- how informative each option is. Within items, not all options are going to be equally informative. For example, in multiple choice examinations not all wrong answers are going to be equally indicative of a low value of proficiency; some are more dumb than others. In the case of psychological scales, the scale designer usually uses equally spaced weights for the ordered options (0, 1, 2, and 3 in the Beck Depression Inventory, for example), but varying the spacing may be more appropriate for some items. MLE weights informative options more heavily than noninformative ones, and in effect assigns numerical scores to options that are optimal.

- just how heavily an option is weighted also depends on θ itself. Some options and items are highly informative for certain ranges of proficiency or trait, but not so for others. Difficult exam questions are informative for high performing students, and give little information about the bottom of the class. MLE in effect assigns the weight to an option that is appropriate to the value of θ that is being estimated.

All this means that one may choose to use TestGraf analysis option 7 to provide what are almost certainly better estimates of proficiency or trait level than were provided by the original score. Note that TestGraf can only estimate values of θ at the values being plotted, so if this option is being used, it may be desirable to set the number of plotted values to its maximum value 101.

Which raises another issue. If the estimates of θ are so much better than the original scores, why not use these as a basis for a new TestGraf analysis of the same data? Good idea! Just follow these steps after the initial run of TestGraf in which maximum likelihood estimates have been computed:

1. set the logical parameter in the TestGraf menu indicating that proficiency values are to be input rather than computed to `*`, and

2. rerun TestGraf using analysis option 7 again.

This process can clearly be repeated any number of times, with the hope that each analysis refines or improves the estimates of θ.

Actually, however, one usually sees these estimates stabilize after only a very few iterations, and in many cases only one iteration will suffice.

As an example, here is what happens with the Beck Depression Inventory for the clinical sample. In the upper part of Figure 32 we see the maximum likelihood estimates at each iteration plotted against those for the previous iteration for three iterations. On the first iteration the previous scores are the raw scale scores. In the lower figures are plotted the changes in score values. We see that on the first iteration a fairly large number of cases change their score values by as much as four up or down. By the second iteration only a few show such changes. On the third iteration the score changes are less than three for all but two cases. For each of these cases the relative credibility curves are bimodal, so that the maximum likelihood estimate is oscillating between two maxima from one iteration to the next. There is little one can do about this, and for all practical purposes we can regard the estimates as sufficiently stabilized by the third iteration.

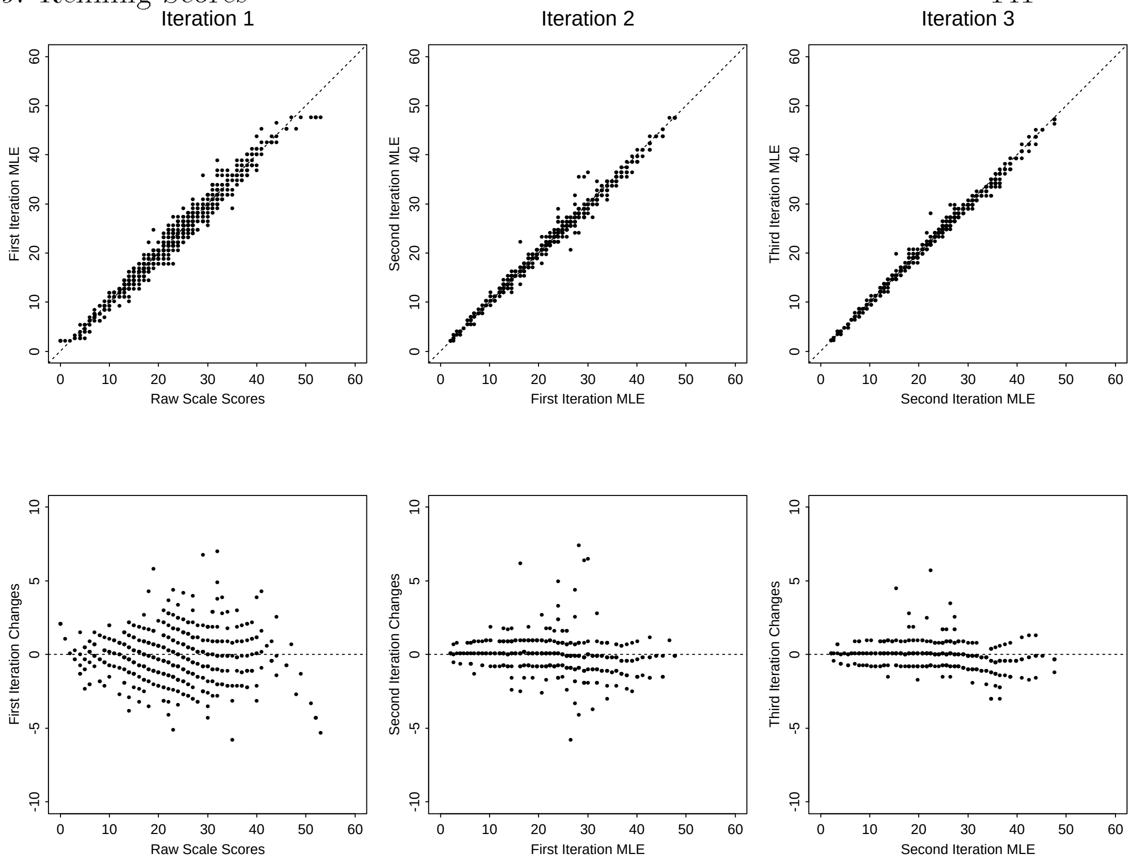

Figure 32: The relationship between successive maximum likelihood score estimates of depression level for the Beck Depression Inventory. In the upper plots the maximum likelihood score estimates of depression level for the Beck Depression Inventory at each iteration are plotted against the estimates on the previous iteration. For the first iteration the previous estimates are the raw scale scores. In the lower plots the changes in scores are plotted against the scores on the previous iteration.

10 Analyses of the Advanced Placement Chemistry Exam

Each year Educational Testing Service, on behalf of the College Board, designs and administers a number of examinations for high school students wishing to receive credit at their colleges of choice for advanced level work in their schools. The complete examinations as well as other supporting materials are published annually after they are scored, and can be obtained from Advanced Placement Program, P. O. Box 6672, Princeton NJ 08541.

The test contains items called free-response in addition to the usual multiple choice items, and these are graded by trained and monitored markers. Test items in these exams are very carefully selected and reviewed before being used, and a typical Advanced Placement Exam is of about as high a quality as one could reasonably expect of a test designed to measure knowledge at a high school level.

In this section we shall use TestGraf to review the 1989 Advanced Placement Chemistry Exam, administered to 18,462 students. The large sample size and the fact that the structure of the exam is rather more complex than most classroom exams means that we can use a wide range of the capabilities of TestGraf.

10.1 An Overview of the Exam

The examination consists of two parts:

1. A 75 item multiple choice section. Each item has five options, and students were allowed one hour and thirty minutes to complete this section. This section was administered first.

2. A free response section, also allocated one hour and thirty minutes. In this section, students were required to:

    - do one required problem, with a score from 0 to 9,
    - do one of two other problems, each with a score from 0 to 9,
    - solve five out of eight possible chemical reactions, each scored from 0 to 3, for a total score of from 0 to 15, and

- complete three out of five possible essays, each scored from 0 to 8.

It is the second free response section that complicates the analysis of the exam because, first, students have a choice among problems, reactions, and essays, and second, scores on these types of items are what are called *graded response* in that they are ordered, and numerical weights were used to grade free-response items. They are, therefore, scale items. An additional factor to be taken into account is the great *speededness* of the exam, meaning that a significant proportion of the students were unable to complete the multiple choice items in the time allotted.

The data to be analyzed for the free response section consisted of a code for each possible problem and for each possible essay, but a single composite score for the chemical reactions. Thus, any student would necessarily have an "omit" response recorded for two of the three problems, and also for two of the five essays. The total number of items to be analyzed are therefore 84, but any one examinee will only have 81 recorded responses. The maximum grade achievable is 132.

Two additional items of data were also made made available:

1. the gender of the examinee
2. the racial or ethnic group of the examinee, with the following groups coded:
    - American Indian
    - Black
    - Mexican
    - Asian
    - Puerto Rican
    - Latin American or other Hispanic groups
    - Whites
    - Other

We are also interested in the exploration of possible sources of item bias using these groups.

The first decision taken with respect to these data was to regard the omitting of a multiple choice item as a specific response, coded in the file as a "0". This is because there are very large numbers of omits, and an omitted response is likely to convey useful information about examinee proficiency since it is reasonable to expect that weaker students will take longer to answer questions, and therefore be unable to complete more items.

In the free-response section, however, we can distinguish between an item omitted as a consequence of the student not having time to try or giving up, and items omitted because not all problems and essays had to be done. For these final 9 items, therefore, an omit because the item was not chosen by the student was coded as a blank.

10.2 Reviewing the Multiple Choice Section

As mentioned above, each of the 75 multiple choice items had five options, but since we are coding omit as an additional option, there are actually six options per item.

We begin with a straightforward TestGraf analysis of these multiple choice items, using all of the default options. The initial TestSave setup phase to construct the files used by TestGraf took about 3.5 minutes on a personal computer with a Intel 80486 50 megaherz processor. The initial TestGraf analysis in which curve values were computed took about 2.5 minutes to the point where the display menu appeared.

Figures 33, 34, and 35 show the option characteristic curves for items selected from the beginning, middle, and end of the exam. We note that the median expected number correct is about 34, and that 95% of the examinees score between about 17 and 57, making this a rather difficult exam by classroom standards. With this number of examinees, the curves are very tightly defined, and there is noticeable imprecision only at very extreme proficiencies. We see that item 1 is rather easy. The omit response 0 begins to show up for item 39, and by the time we get to item 75, it is the dominant "wrong" answer, and is even more popular than the right answer over all but very high proficiency values.

Figure 36 gives us an overview of the multiple choice part of the test by displaying the correct option curves in terms of the first two principal components of shape variation among them. As is usual in this display, easy items are on the right and hard ones on the left; highly discriminating items

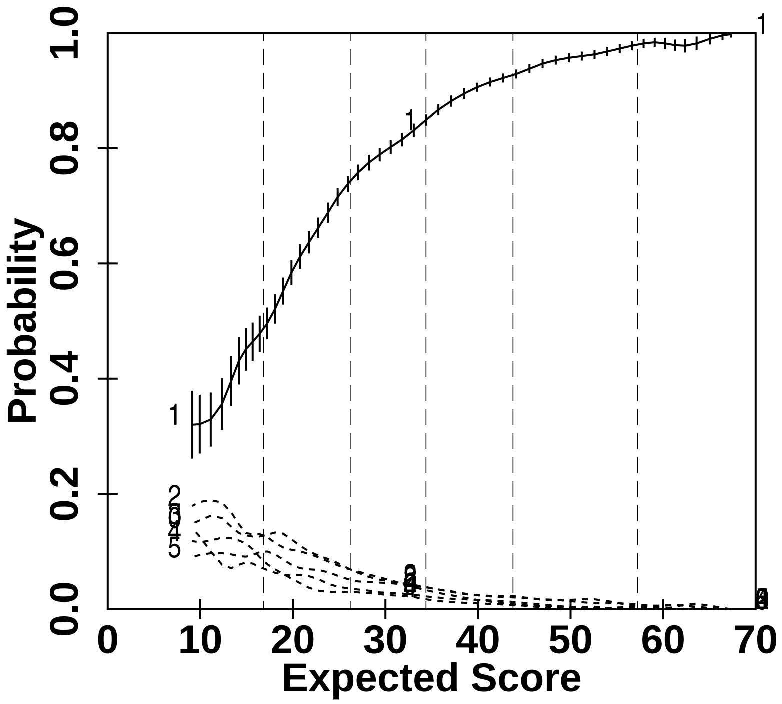

Figure 33: Option characteristic curves for item 1 for the multiple choice section of the Advanced Placement Chemistry Exam.

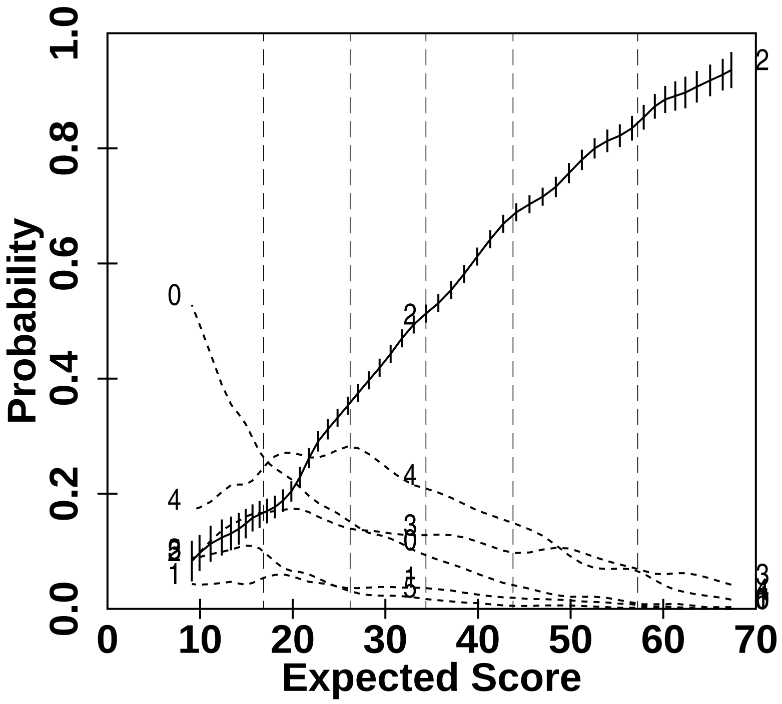

Figure 34: Option characteristic curves for item 39 for the multiple choice section of the Advanced Placement Chemistry Exam.

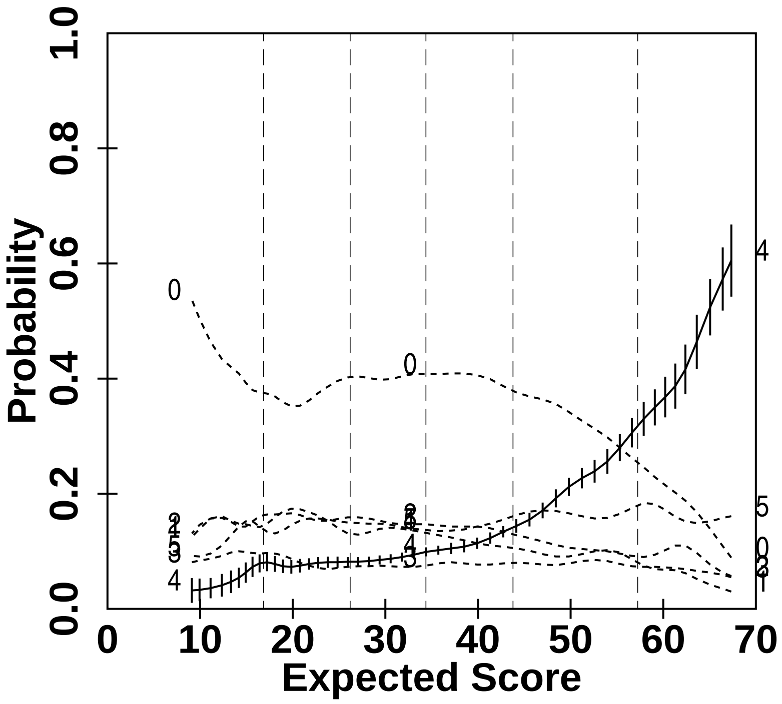

Figure 35: Option characteristic curves for item 75 for the multiple choice section of the Advanced Placement Chemistry Exam.

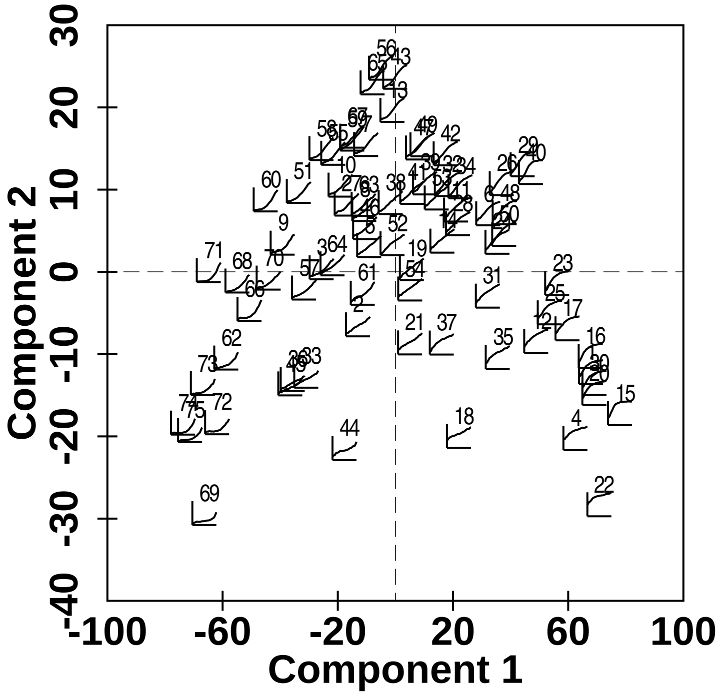

Figure 36: The correct option characteristic curves for the Advanced Placement Chemistry exam displayed as a function of their principal component scores in a principal components analysis of shape.

are on the top and less discriminating ones are on the bottom. We see that the test contains an abundance of highly discriminating items concentrated in the middle of the proficiency range, and hence the sharp peak in the information function. There are only a few (18, 22, 44 and 69) poorly discriminating items.

Figure 37 shows the test information curve as a function of standard normal quantiles. The test is obviously most powerful for examinees near median proficiency, and the fall-off in information for very high proficiency examinees is somewhat disappointing. Although one might suppose from items 75 that there is no lack of difficult items, in fact items such as this are difficult only because they are near the end of the test, where the main competing response is omit. On the other hand, this mid-range power is entirely appropriate to the purposes of the test, which was designed to assist colleges to decide when a student could be exempted from college-level introductory

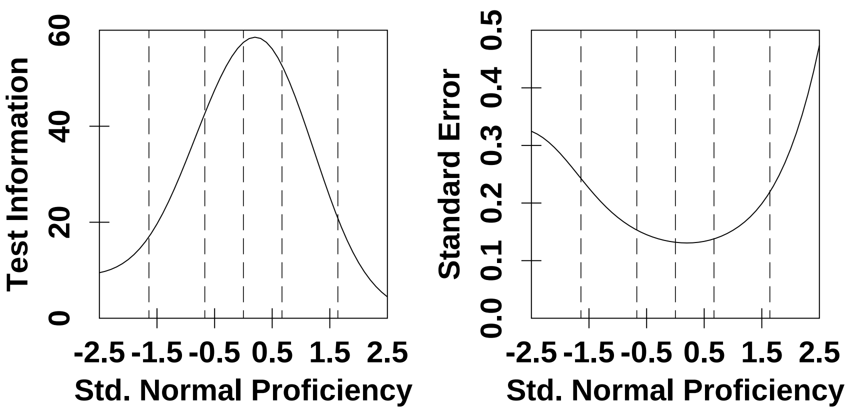

Figure 37: The test information function and the standard error function for the multiple choice section of the Advanced Placement Chemistry Exam.

chemistry.

Almost all of the difficult items occur near the end of the test, again illustrating the strong speededness of the test. We are left with the tantalizing and frustrating question of whether there were very proficient students in a more profound sense who score well below what they could if they had had as much time as they needed. Were students whose first language is not English unduly handicapped in this exam? One would not imagine, on the other hand, that proficiency and speed of response had nothing to do with one another; clearly weak students will tend to take longer answering questions and therefore finish fewer.

10.3 Reviewing the Free Response Section

Figure 38 gives the option characteristic curves for the first problem item, scored from 0 to 9. The first problem is required, and the students had

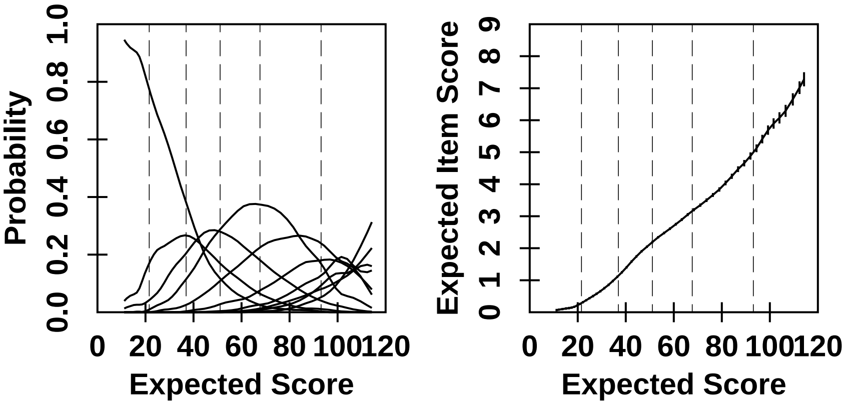

Figure 38: Option characteristic curves and expected item score for the first required problem for the complete Advanced Placement Chemistry Exam. The problem score ranges from 0 to 9.

to choose between the second two. The option characteristic curves are typical of graded response items in that the peaks for curves associated with higher levels tend to occur more to the right. The expected item score curve indicates that this is a highly discriminating item.

Figure 39 displays the information functions for both the multiple section and the entire test for the Advanced Placement Chemistry Test as well as for the Introductory Psychology Exam and GMAT Quantitative Test. For comparison purposes the information functions are divided by the number of items in the test to give a information per item measure. We see that the scale type problem items at the end of the chemistry test have the effect of spreading the information function out by raising the information levels for the examinees of extreme proficiency. This is consistent with the relatively flat information function typical of scale items, as seen, for example in Figure 11 for the Beck Depression Inventory. The improvement in the information

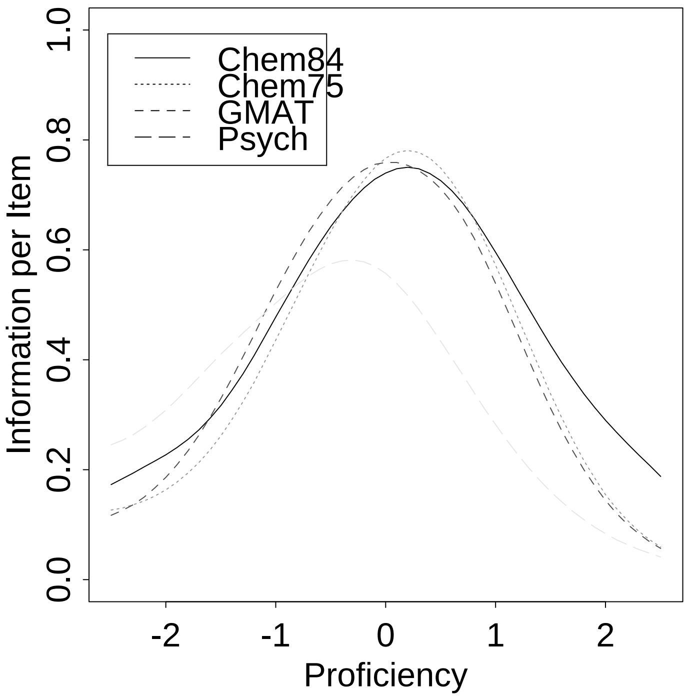

Figure 39: Information functions for four exams. For comparison purposes the information function has been divided by the number of items. The functions labelled Chem75 and Chem84 are for the multiple choices section and the complete test for the Advanced Placement Chemistry Exam, respectively.

for the strongest examinees is particularly pleasing. The GMAT exam looks very much like the multiple choice section of the chemistry exam. But the inferior quality of the Introductory Psychology Exam is obvious, as well as the shift in its power to the low end of examinee proficiency.

Our overall assessment of the test at this point is that almost all items, including the free response items, contribute very effectively to making this a powerful instrument for assessing chemistry mastery, especially at the mid-range of proficiency.

10.4 Gender DIF

Gender differences in overall performance in science courses is a very heavily researched topic, and experience indicates that we should expect some differences to be apparent in these data. However, item bias, or DIF, is something else, and is a problem of acute interest to professional testing institutions such as Educational Testing Service. They usually go to great lengths to spot potential sources of bias in items before using them.

Figure 40 results from a comparison of males and females using TestComp, and confirms that there is an important difference in performance in favor of males. The expected total test scores for females at the 25%, 50%, and 75% quantiles are 33, 45, and 60, respectively, while the corresponding scores for males are 40, 54, and 71. Thus, in the central half of the distribution, males in the corresponding central half score from 7 to 11 points higher on the average than the corresponding females.

Item 39 shows substantial DIF, as indicated in Figure 41, where we see that males at around median proficiency have a probability of getting the right answer that is about 0.1 higher than for females with the same expected total scale score. This figure also shows that most of the DIF for this item is due to the higher probability that females will omit this item.

TestComp also revealed an important amount of DIF in items 12, 16, 18, 24, 31, 36, 62, and 73. Of these, the bias was in favor of the females in items 12 and 36. None of the free-response items showed any DIF, however. While it is unfortunate that some item bias escaped the best efforts of the test designers, it should be emphasized that the amount of bias is much too small to explain the large differences in performance for the test as a whole.

10.5 DIF for Whites, Asians and Blacks

Among the 18,462 examinees, 12,287 identified themselves as white, 3,435 as Asian, and 590 as black. Again, there are well-publicized performance differences among these groups in tests of this nature, and Figures 42 and 43 show that this is indeed the case. There is a slight edge for Asians over whites, and a very large discrepancy between the performance of blacks relative to both other groups.

But what about item bias? Well, it turns out that there is very little evident bias among the 84 items. Figure 44 shows what is the worst case:

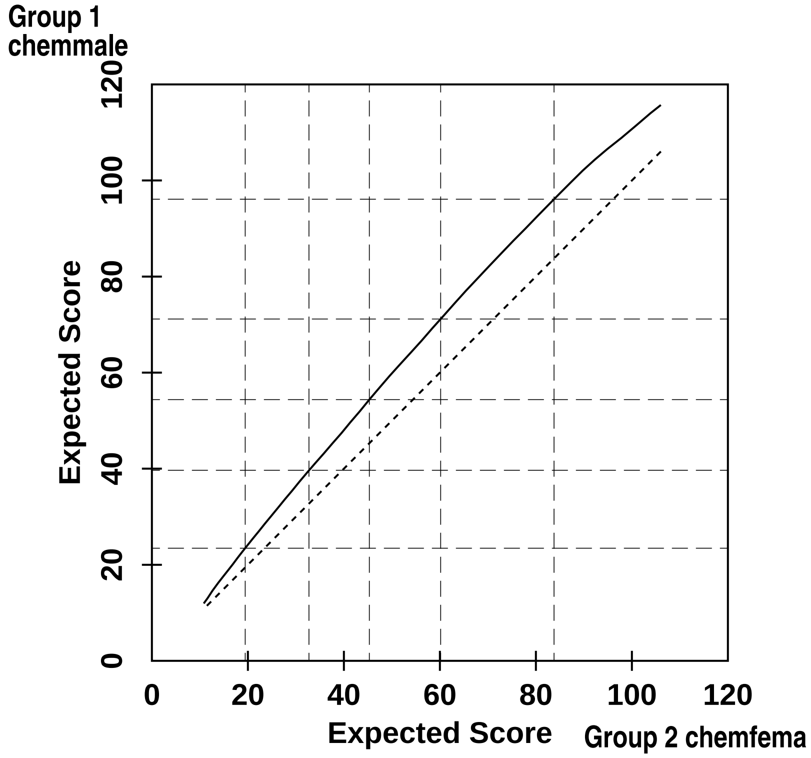

Figure 40: Expected total scores for males plotted against those for females at corresponding quantile levels for the Advanced Placement Chemistry Exam. Males outperform females at 5 points or more for all but the lowest proficiency ranges.

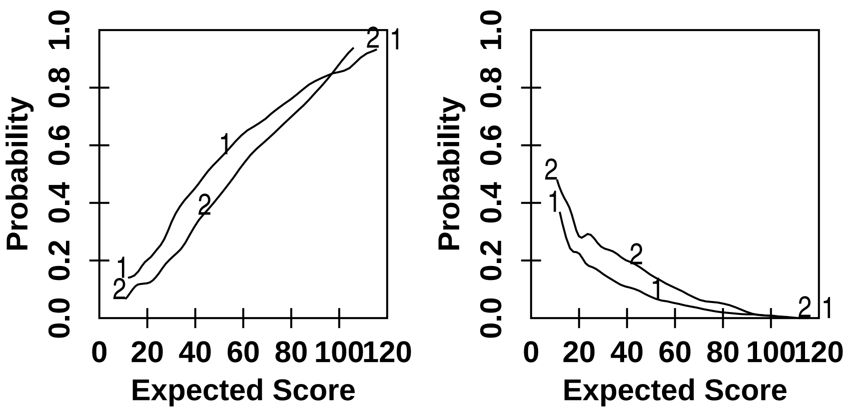

Figure 41: The left figure shows the correct answer characteristic curves for the males (1) and females (2), and the right figure shows the characteristic curves for omitting the item for item 39.

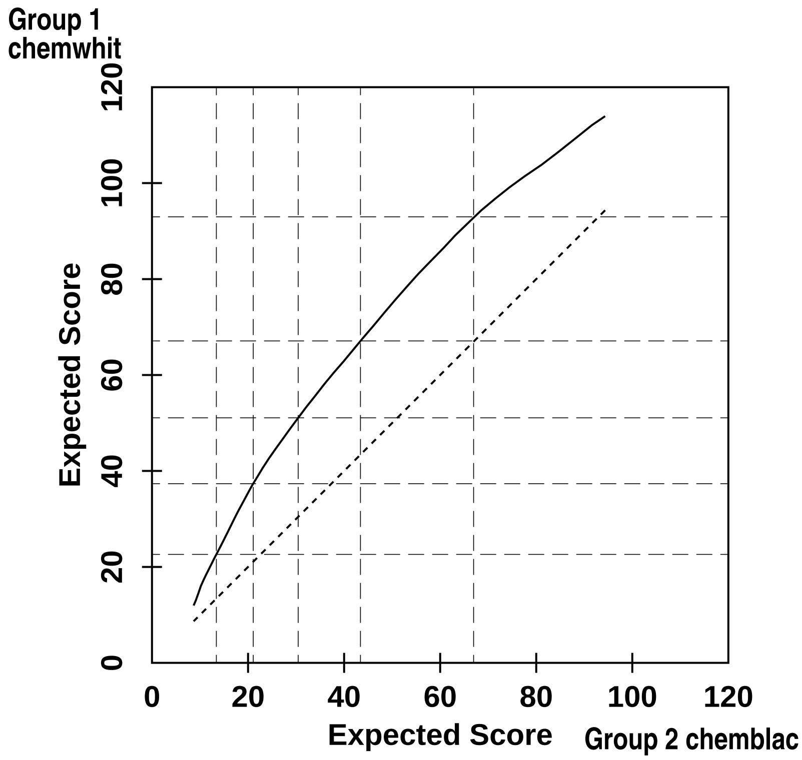

Figure 42: Expected total scores for pairs of the ethnic/racial group categories, White, and Black.

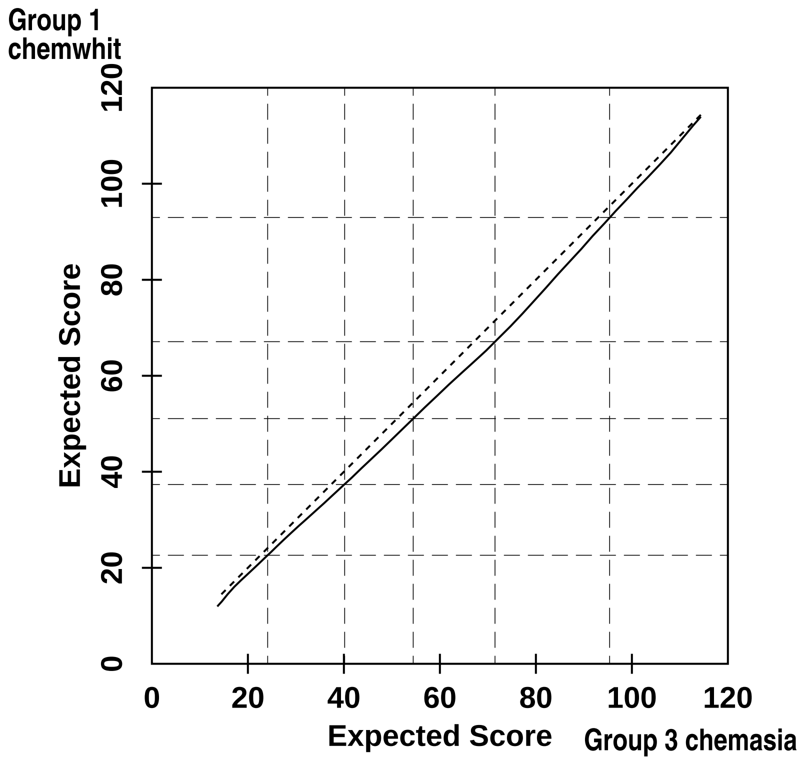

Figure 43: Expected total scores for pairs of the ethnic/racial group categories, White, and Asian.

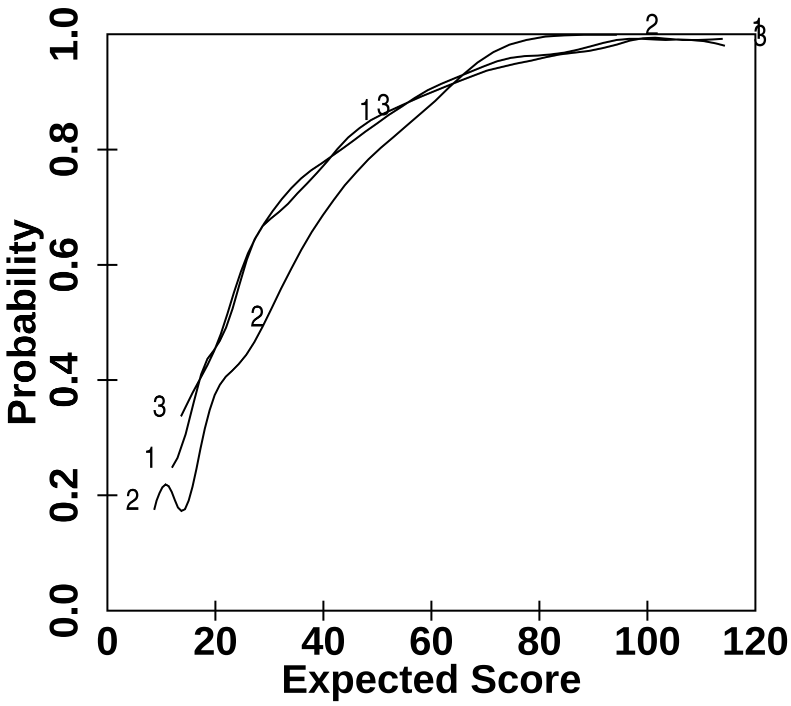

Figure 44: Expected item scores for the ethnic/racial group categories, white (1), Asian(3), and black (2) on item 16. This is the only noticeable case of DIF for these groups.

item 16. This is a rather easy item, but one which blacks find somewhat more difficult than the other two groups. Still, given the smaller sample size of this group, and the large number of items over which we searched for this type of DIF, this seems pretty tolerable.

11 References

Allen, M. J. & Yen, W. M. (1979) *Introduction to Measurement Theory.* Belmont, Cal.: Wadsworth.

Altman, N. (1992) An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46,* 175-185.

Beck, A. T., Rush, A., Shaw, B., & Emery, G. (1979) *Cognitive Therapy of Depression.* New York: Guilford Press.

Eubank, R. L. (1988). *Spline Smoothing and Nonparametric Regression.* New York: Marcel Dekker.

Green, P. J. and Silverman, B. W. (1994) *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach.* London: Chapman and Hall.

Härtle, W. (1990) *Applied Nonparametric Regression.* Cambridge: Cambridge University Press.

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991) *Fundamentals of Item Response Theory.* Newbury Park: Sage Publications.

Hastie, T. and Loader, C. (1993) Local regression: Automatic kernel carpentry. *Statistical Science, 8,* 120-143.

Holland, P. W. & Wainer, H. (Eds) (1993) *Differential Item Functioning.* Hillsdale, N.J.: Lawrence Erlbaum.

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

Lord, F. M. & Novick, M. R. (1968) *Statistical Theories of Mental Test Scores.* Reading, Mass.: Addison-Wesley.

Mislevy, R. J., & Bock. R. D. (1982) *BILOG: Item analysis and test scoring with binary logistic models* [Computer Program]. Mooresville, IN: Scientific Software.

Parker, D. B. (1991) *AcroSpin.* Salt Lake City: ACROBITS, P. O. Box 26871, Salt Lake City UT 84126-0871.

Ramsay, J. O. (1991) Kernel smoothing approaches to nonparametric item characteristic curve estimation. *Psychometrika, 56*, 611-630.

Santor, D. A., Ramsay, J. O., & Zuroff, D. C. (1994) Nonparametric item analyses of the Beck Depression Inventory: Evaluating gender item bias and response option weights. *Psychological Assessment, 6* 255-270.

Silverman, B. (1986) *Density estimation for statistics and data analysis.* London: Chapman and Hall.

Thissen, D. & Wainer, H. (1982) Some standard errors in item response theory. *Psychometrika, 47,* 397-412.

Wingersky, M. S., Patrick, R., & Lord, F. M. (1988) *LOGIST Users Guide.* Princeton, NJ: Educational Testing Service.