Documentation for Beta-Binomial Program

I. General

This program allows one to fit a latent trait model with a distribution of the latent variable that is two-parameter beta to dichotomous data. Both constrained and unconstrained models are allowed. Data with separate groups may also be analyzed. Maximum likelihood estimates are produced through two different algorithms. The program is interactive, and results are written to a file 'BETA Date Time.LOG' in the current directory. (For "Date Time," the current date and time are written, so you won't accidentally overwrite old log files.) In addition to this documentation, a flowchart is provided that will aid in navigation through the program.

REQUIREMENTS: This version is a recent recompilation of a program originally designed to run in an extended DOS environment. The change to Windows NT/98 has not affected the numerical results, but may have adversely affected some presentation. Please tell us if anything does not seem as it should be. When you start up BETA, it may begin in full screen mode. You should probably shift to a windowed format (Alt-Enter) and then page so that the bottom of the window isn't cut off by some scroll bar; otherwise the page may blink when you type. You may not see a cursor—sorry about that—and so if you ever think the program has hung, it may be that BETA is waiting for input from you. (For some types of input, such as a number, if you press <CR> without specifying a number, BETA will keep on waiting for that number.)

DATA FILE: BETA reads in ASCII text file data in a number of forms. Data files in a number of different ASCII forms can be read--it is not necessary that the user know the format, though formatted input is still possible. The 'standard' type is considered to be sequential format, where first the pattern vector is stored in 1's and 0's, and then the frequency follows. It is only important that there be spaces between each value, including the 1's and 0's in the pattern representation. (A program to facilitate the data handling, CONCON, is included with BETA, and it stores data in such fashion). Data that are produced by the SPSS CROSSTABS /WRITE=ALL command are directly readable in the form in which SPSS makes them. This is a convenient way of producing contingency table data from individual data.

Thus to create the data for BETA from SPSS for 4 variables, A, B, C and D, originally scored (1=yes) (2=no), the syntax would be

```
RECODE A B C D (1=1) (2=0) (ELSE=SYSMIS).
FILE HANDLE OUT /NAME='C:\BETA\MYDATA.DAT'
PROCEDURE OUTPUT  OUTFILE=OUT.
CROSSTABS VARIABLES=A B C D (1,2).
  /TABLES=A by B by C by D
  /WRITE=ALL.
```

See an SPSS manual for details.

You may use this form to enter grouped data. In this case, the group must be the LAST variable used—it will be furthest right and changing least frequently. Because SPSS will write out a cell for each combination, even if it is empty, BETA just takes for granted that the first 2**N lines are for group 1, where N is the number of items, the second 2**N for group 2, etc. So it doesn't care what the group numbers are in this format, it just assigns them consecutively as would follow.

Also note that in this form, BETA expects the variables to be scored 1,2 and not 0,1, so don't recode; just tell BETA to read this in as SPSS crosstabs output.

Alternatively, you can convert it to standard form using CONCON. Standard form looks something like this:

```
0 0 0 0 34
0 0 0 1 23
0 0 1 0 45
0 0 1 1 88
... etc.
```

NOTE THAT THE VARIABLES USED TO INDICATE THE PATTERN MUST BE INTEGERS; if they have places to the right of the decimal in your SPSS system file, they will retain them in the CROSSTABS produced file. BETA can read in these non-integer values, but be sure to specify this file format (option 6 in the menu). In the standard form, the group data should be the last variable; group numbers must be consecutive integers—if they don't start at 1, BETA will recode them so that they do.

IMPORTANT NOTE: The data must be coded such that 1 is consistently either a positive or a negative response; the covariation between all items must be positive. If you are using data from multiple groups, it may not be possible to code the data such that this is true for all groups. In such cases, the model cannot be fit; you must collapse over the offending item(s).

II. Program Structure

There are four major chunks of the program, accessible from a menu.

i) CHANGE CERTAIN PARAMETERS

One chunk is to change certain default values that affect program flow or the iterative procedures used to find MLEs. Changable parameters are the maximum number of iterations, the tolerance for the iterative procedure. One can also choose whether to proceed directly to a set of starting values (see below), or go systematically through the data, viewing different normalizations through 2 by 2 tables. One can also choose whether the iterative procedure writes additional information to the logfile, such as the values of different parameters at each stage in the iteration, various warning messages, what subroutine the program is in, etc. These options lead to a huge LOG file and should only be used when one suspects a problem with the iterative procedure. Another default is whether to try to have BETA handle overflows (i.e., parameters getting too large). If your model converges instantly with a bad fit and this default is ON, try turning it off.

ii) ANALYZE DATA FROM DISK--initial

The second chunk brings in a datafile and computes preliminary estimates from the 2 way tables, fixing one variable for the purpose of estimation. One can go through all the variables, fixing each in turn, or do just one. If the default is changed from the interactive mode to the direct mode (see previous paragraph), this stage is skipped, and the start values are loaded into memory. NOTE: the current default is to skip the interactive mode. Preliminary fits are obtained, but experience shows that poor preliminary fits may be deceiving. The initial parameter values for each preliminary fit are written to the log file; it is possible to produce complete information on

the preliminary fit (see below). The preliminary fits are relisted if all variables have been tried, to guide the choice of start values for iterative estimation.

NOTE: If you are analyzing data that are from separate groups, we recommend you use the default file format—the other file formats should work, but have not been tested. In the default format (or in the user specified format) the grouping variable must be in between the indicator variables and the frequency; it must go from 1 to G where G is the number of groups (i.e. there can be no skipped group numbers). If your data are not in this form, use CONCON and a text editor to get them in shape. If the proper number of groups are not detected, BETA will warn you. (Don't try loading grouped data in a different form—BETA may not notice and everything will go kerflooey.) If you use the crosstabs write output format, make sure the grouped data is last (see above). If you use a frequencies only format, the first 2**NI patterns will be assumed to be from the first group, the second 2**NI from the second, etc. BETA will number the groups automatically. You should have a return between groups.

iii) ANALYZE DATA FROM DISK--MLE

The third chunk allows one to compute maximum likelihood estimates iteratively. Given the peculiar case of the near-unidentifiability of the model, one has a choice whether to place a restriction on the model by fixing one parameter (an item parameter), or whether to fit an unrestricted model. An unrestricted model may take a long time to converge, and require certain choices of fitting procedure (see below). More important, the standard errors of the parameters will be far larger than the values of the parameters themselves (with the possible exception of the b distribution parameter). A restricted model, while having reasonable standard errors, is not generally the true Maximum Likelihood estimate for the full parameter space, and in some cases (depending on the fixing value chosen), this may lead to the error of a model being rejected that actually fits.

One way of handling this complication is to fit both restricted and unrestricted models. An unrestricted model should be fit using the simplex method (see below), with a very low tolerance (below 10E-15 for safety). This method is less susceptible to local minimums than the conjugate gradient method. Because the log-likelihood function has a very long, almost but not completely flat ridge along the "line" of relatively constant ratios of item parameters, the algorithm will creep up to the top and stop there. Because of the peculiarities of the difference between the power-of-a-beta-function and a proper beta function, this true maximum will probably have very low values of the parameters compared to the start values (perhaps two orders of magnitude less for the item parameters).

One can also fit restricted models, and compare the difference in fit as a one-degree-of-freedom test, and see if a significant loss occurs with the restricted case. Finally, one can first find the global maximum, then fit a restricted model where one of the parameters is fixed to the value found at this maximum. One can then derive standard errors for the reduced set of parameters and, if feeling guilty, recompute the probability of the chi-square as having one less degree of freedom than it seems.

GROUPED DATA

If your data are grouped, you can fit models that constrain some parameters to be the same for all groups, but others may differ (any constrained parameters must be equal for ALL groups, not just some). You will be presented

with a list of the parameters in the model, with the distribution parameters first. Put an 'x' under any parameter you wish to be constrained to be the same for all groups; leave all other spaces blank. Remember to take into account any previous constraints you may have placed on the parameters.

For examples of grouping hypotheses, and how they tranlate into these constraints, we consider a four variable, two group case.

Hypothesis 1: items are invariant, but groups have different distributions. (a common null model for item-bias tests). Whether restricted or unrestricted in terms of setting the metric, you would enter

```
ab1234
  XXXX (your input)
```

Hypothesis 2: distribution of latent trait is same, but item parameters are different (as in an treatment-control experiment). Again, whether or not the metric was set by a restriction, you would enter

```
ab1234
XX
```

Hypothesis 3: complete independence--both groups are different in distribution and in item parameters. While this is the same as adding the likelihood ratio chi-squares for the tests independently, it may be a null model, and one may wish to compute it using the grouped data. If fitting unconstrained models (i.e. no item parameter is constrained to set the metric), you do not enter any constraints at all under "ab1234." But if you wish to test independently constrained models, they must have the same variable set to the same value. For example, if metrics for both groups were identified by setting K1 to 1, you would both give this restriction in response to "Fix one parameter?" AND enter

```
ab1234
  X
```

SCORES

Both the intitial and maximum likelihood estimates produce estimated frequencies, chi-squares, and residuals. Optionally, one can get the derivatives and standard errors of estimates of parameters, estimated average scores on the latent variable for each cell, and plot the distribution of the latent variable. The scores can be written to a current file (Name=<filename-base>.SCR) which is saved on exit or when a new datafile is loaded. These scores can then be read in by SPSS and regressions of one set of scores on another made. To read in the data under SPSS, choose FILE->READ ASCII DATA and FREEFIELD FORMAT. The data will be loaded with each normalization a case and each cell a variable. You can save templates for frequently used numbers of variables. Here is the SYNTAX for four variables--you can paste it from here into a SPSS syntax box:

```
SET
  UNDEFINED= WARN.
DATA LIST
  FILE='C:\BETA\filename.SCR' FREE /c1 * c2 * c3 * c4 * c5 * c6 * c7
 * c8 * c9 * c10 * c11 * c12 * c13 * c14 * c15 * c16 * .
EXECUTE.
```

Once the Data are read in, Transpose the matrix through DATA->TRANSPOSE.

This will put them in the form of each cell being a case, and having four variables, each variable a normalization. A new variable will be created that is the label for each cell. You can then SCATTERPLOT and label the points by the case label Case_lbl, or do regressions.

iv) GENERATE DATA

The fourth chunk generates data from a set of parameters. The program will remember the last parameters used, if you want to make variations. You can also add a disturbance term which will, proportional to the estimated frequency, perturb the final count, for various experiments. This is set under CHANGE CERTAIN PARAMETERS. If a datafile is in memory, you can compare the fit of these entered parameters to the data, or use them as start values for iterative fitting. This is therefore the way to start a routine at an arbitrary place. You can save the generated data as a TYPE 4 (default) data file, and re-read it in for analysis.

III. Routines Used

A. Iterative fitting

Two different procedures are currently implimented, both taken from Numerical Recipes in Fortran. The first is a conjugate gradient method, which uses the first derivatives to take successive turns maximizing along one dimension, and then along a dimension conjugate to that, etc. Currently, without being adapted to the peculiarities of this model and the bounds of the parameters, this method does not seem as efficient as it could be. It is okay for when one parameter is restricted. This method is the Polak-Ribiere adaptation of the Fletcher Reeves algorithm.

The second method--the Downhill simplex Method--involves the creation of a N+1 dimensional simplex of parameter points (where N is the number of parameters) and the worst point of the simplex is replaced, moving the simplex as a whole steadily (but slowly) to the maximum. Because the simplex does not contract until it has to, it retains a wider range of alternative points to the maximum that is being zeroed in on by the conjugate gradient method, and is less likely to fall into a local maximum. This method is from Nelder, J. A. and Mead, R. COMPUTER JOURNAL v. 7(1976):308-313.

When using this method, restarting the algorithm at the stopping point is helpful in avoiding premature maximums. The routine will select a new simplex by perturbing around the current maximum, and will begin searching anew. Such a restart is not helpful with the conjugate gradient technique. While the conjugate gradient technique will generally converge in a number of iterations on the order of 100, the conjugate gradient will be of an order higher, and therefore the number of maximum iterations should be high for this method. The iterations are quite fast. In both cases, in an un-restricted model, the tolerance for the algorithms should be very low, since small differences in likelihood are associated with big differences in parameter estimates.