Mx Mx
Mx Mx

Statistical Modeling

Michael C. Neale

Mx: Statistical Modeling

by

Michael C. Neale
Department of Psychiatry
Medical College of Virginia
Virginia Commonwealth University
Richmond, Virginia, U.S.A.

Medical College of Virginia, Virginia Commonwealth University
Department of Psychiatry

First Published 1991
Second Edition 1994

Refer to Mx manual as:

Neale, M.C. (1994). *Mx: Statistical Modeling.* Box 710 MCV, Richmond, VA 23298: Department of Psychiatry. 2nd Edition.

Contents

List of Tables

List of Figures

Preface

The development of Mx owes much to LISREL and I acknowledge the pioneering effort put in by Karl Jöreskog and Dag Sörbom over the years. There are many who have supported and encouraged this effort in many different ways. I thank them all, and especially Lindon Eaves, Ken Kendler and John Hewitt since they also provided grant support[1] , and David Fulker for allowing modification of his notes on matrix algebra to be supplied as an appendix for this document. Jack McArdle and Steve Boker provided excellent path diagram drawing software, and Luther Atkinson suggested the binary file save option. Health Sciences Computing at the Medical College of Virginia, Virginia Commonwealth University provided excellent software, hardware and consultancy support. My family have had to put up with my working unreasonably long hours to put this package together, and I would like to thank them for their patience.

A number of users have been most helpful finding errors in the documentation or software or both, and for suggesting new features that would make Mx easier to use. I hope that all users will forward any comments, bug reports, or wish-lists to me. My current address is:

address	Department of Psychiatry
	Box 710 MCV
	Richmond VA 23298-0710, USA
phone	804 786 8590
fax	804 371 6577
email	neale@ruby.vcu.edu *(internet)*
	neale@vcuvax *(bitnet)*

and my order of preference for communication is email, fax, phone and snail mail. For problems encountered running Mx, email is especially useful when problem files are included. You can grab the code for a variety of operating systems via anonymous ftp to sapphire.gen.vcu.edu. Please have the courtesy (and self-interest)

[1]The author was supported by ADAMHA grants MH-40828, MH-45268, AG-04954 and MH-45127

to email me to let me know what you grab so that I can keep you informed of updates, bug reports etc.

The bad news is that this manual is quite long; the good news is that you don't need to read it all. There is a very brief background to structural equation modelling in Chapter 1. The "how to" part of the manual starts in Chapter 2.2, in which general syntax conventions and job structure are laid out, followed by description of the commands necessary to read data. Chapter 3 deals with the heart of Mx: how to define matrices and matrix algebra formulae for model-fitting, and ways of estimating and constraining parameters. Methods of changing the default fit-function, of decreasing and increasing the quantity (and quality) of the output, and for fitting sub-models efficiently, are described in Chapter 4.1. The last chapter supplies and briefly describes a number of example scripts. The Appendices describe the use of Mx under different operating systems, error codes, and some introductory matrix algebra.

Chapter 1

Introduction

1.1 Background to the Software

For many years, the analysis of human twin and family data has been hampered by the lack of good, general software that can be taught and used with ease. Mx has been written to fill some of this gap. In principle, any analysis can be done if the user is prepared to learn a programming language such as FORTRAN or C, and write programs as required to fit models, compute χ^2 and so on. From years of experience I know that this is not very efficient; programming is subject to subtle errors and is time-consuming. The philosophy of Mx is to protect the user from these errors, while at the same time providing the greatest ease and flexibility in model specification. The Mx command language has been written to maintain some consistency with LISREL (Jöreskog & Sörbom 1989) whose users should find Mx especially easy to learn.

The currently available Structural Equation Modelling (SEM) packages such as LISREL (Jöreskog & Sörbom, 1989), EQS (Bentler, 1989) and COSAN (Fraser, 1988) were written with one group in mind. Only after some time was the concept of multiple groups added to these programs, and they remain rudimentary in this respect. Multiple groups are the "stuff of life" of the analysis of twin and family data, so Mx was explicitly designed to make working with many groups as easy as possible. Other aims were to generalize the specification of multivariate models for data from relatives, and to allow boundary, linear and non-linear equality and inequality constraints.

Mx is not simply another SEM package. Of course it can fit any structural equation model by the currently popular methods, but it can do a number of other things too. It is possible to 'play' with matrix algebra, without fitting models, for the purposes of education or research. Models for genetic linkage can be fitted. Raw data with missing values can be analyzed by maximum likelihood. No doubt many other data analysis tasks can be performed; Mx is *flexible.*

MX obtains its flexibility by allowing the user to specify matrix expressions. Effectively, a matrix algebra interpreter has been tied to state-of-the-art software for numerical optimization (NAG, 1990). The interpreter has been designed so that matrix formulae are entered much the same way as they would be written. This brings conceptual model development and data analysis much closer together.

1.2 Translating Path Diagrams into Matrices

Any model for covariance structure could in principle be translated into an infinite number of diagrams, or have an infinite number of matrix representations. For example, Figure 1.1 shows four possible ways in which the variance of a single variable could be modelled. The first estimates the variance directly, and the second

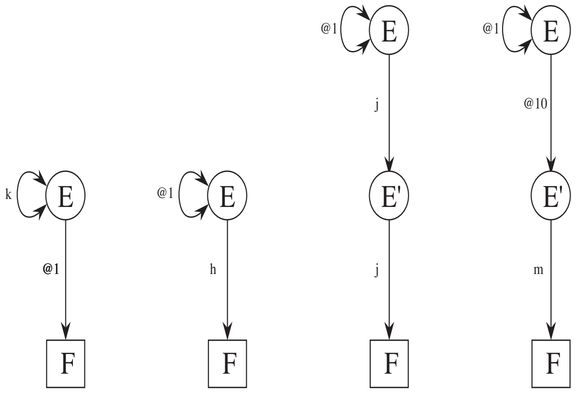


Figure 1.1: Four of the infinite variety of models for the variance of a single variable

shows a model for the square root of the variance (the standard deviation). The third and fourth estimate the fourth root of the variance and one-hundredth of the variance respectively. Using the latter two is definitely questionable, since much complexity has been added to the model with no increase in concepts. Choosing between the first two is more difficult. We note first that the second model implicitly constrains the variance to be positive. This constraint is advantageous for a model of a variance since an observed variance can never be negative. If we fit a model involving several different sources of variance, it can be important to keep them from becoming negative, even if when summed they result in a positive variance. Thus for most models, we select the second approach; in the analysis of genetically informative data it yields an intuitive model which generalizes easily to multivariate data analysis. The disadvantage of this approach is that the standard error of the path coefficient is less valid as a measure of significance (Neale *et al*, 1989).

For many people, path diagrams provide an intuitive summary of the model that they wish to test. Others prefer structural equations explicitly stating which variables are linear functions of which other variables. Still others prefer to use matrix algebra. All methods, if used properly, will yield precise mathematical expressions of a model. To make optimum use of MX we must translate the models into matrix algebra in an accurate (and hopefully elegant) fashion. The step from matrix equation to a functional model fitting program is relatively small, and much insight can be gained from the matrix formulation.

To translate a model from a path diagram we must know a number of things about the variables. Path analysis has been justly criticized for being used insufficiently explicitly (Karlin et al, 1983). Many users implicitly assume that all variables, latent as well as observed, are standardized to unit variance. Others do not, but fail to define the parameters that give the latent variables their variance. Here we follow the method of McArdle (1980, 1984) and McArdle and Boker (1990). First we discriminate between latent and observed variables by drawing boxes around observed variables and circles around latent variables. This is not a crucial distinction, the same covariance structure may be derived with or without the circles and boxes, but it adds considerably to the ease with which a model can be 'seen' by the reader. Second, we separate variables into two sets: those that are caused by other variables, and those that are not. If a variable does not have a unidirectional arrow pointing at it, we call it an *external* variable (many other terms are common for this class of variable, such as independent, exogenous, and source, but we will use external). On the other hand, if a variable is caused by another variable we call it an *internal variable* (other terms include dependent, endogenous, and target). These two sets are mutually exclusive; a variable cannot be both external and internal. External variables may correlate (via a two-headed arrow) with other external variables. They may not correlate in this fashion with internal variables. The only relationship permitted between an external variable and an internal variable is that the external variable causes the internal one, shown in a diagram as a single headed arrow going from the external to the internal variable.

Assuming that we are not drawing a path diagram of every event in the universe since the big bang, we must allow for variables that are not shown in the diagram, but which contribute variance to one or more variables in the diagram. We have already dealt with the possibility that external variables covary due to these non-depicted variables, by allowing for double-headed arrows between them. It remains to specify the *variance* of external variables. One of the most concise methods of doing this is to draw a double-headed arrow from a variable to itself. The value of such a path is the residual variance of that variable. Both external and internal variables may have these residual paths; if they are absent, they are assumed to be zero. This formulation agrees with the McArdle and Boker (1990) RAMPATH path diagram software conventions. Output from MX can be used with RAMPATH to visually display models, trace paths leading to variance and covariance of latent and observed variables, and to print publication quality path diagrams.

Path diagrams in this document follow two conventions:

1. External variables without explicit residuals have a variance of unity
2. Internal variables do not have any residual variance unless explicitly shown.

In certain circumstances, it is useful to partition external variables by placing them in different matrices. Although this can always be avoided, it can lead to increased conceptual clarity and numerical efficiency. Two external variables (either latent or observed) may be placed in different matrices if:

1. They do not correlate with each other
2. They do not both correlate with any third external variable.

A good example of this practice is the multivariate ACE model for twin data described in Section 5.2.4, in which genetic, shared environment and random environment components are estimated in separate matrices. Distinction between variables in this fashion may simplify requests for modification indices, or the fixing and freeing of parameters.

If there are no direct causal relationships between internal variables, the structural model may be specified with an $\mathbf{ABA}'$ formulation. The $\mathbf{B}$ matrix will contain the covariances among the external variables and the $\mathbf{A}$ matrix will have elements A_{ij} that represent paths from external variable j to internal variable j. If there are other variables 'lower' in the diagram, i.e. caused by the internal variables, they may be represented in another matrix and the $\mathbf{ABA}'$ formulation may be expanded to $\mathbf{CABA}'\mathbf{C}'$. This is only the case if a) the external variables are not causes of the second level of internal variables, and b) the second level of internal variables does not cause any variables at the second or first level. This approach may be continued through any number of levels of variables.

However, if an internal variable causes another internal variable at the same level or higher, then a different formulation is required. Both of these variables will need to be specified in the same matrix, and an $(\mathbf{I}-\mathbf{G})^{-1}$ formulation is needed to specify these effects. The rationale behind this approach is quite simple and general.

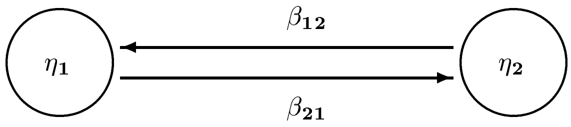
Consider the Path Diagram in Figure 1.2. This shows a feedback loop between two internal variables, which we call η variables. The total variance of η_1 and η_2 is the sum of the infinite geometric series:

$$\beta_{21}\beta_{12} + \beta_{21}^2\beta_{12}^2 + \beta_{21}^3\beta_{12}^3 \ldots$$

It is simple to show that if $|\beta_{21}\beta_{12}| < 1$ then the series converges. Let the sum of the series of n terms be called S_n, and let $\beta_{12}\beta_{21} = r$. Then

$$\begin{aligned} S_n &= r + r^2 + r^3 + \ldots r^n \\ rS_n &= \phantom{r +{}} r^2 + r^3 + \ldots r^n + r^{n+1} \end{aligned}$$

Figure 1.2: Feedback loop between two variables, η_1 and η_2.



Thus the difference between these two equations is:

$$(1-r)S_n = r - r^{n+1}$$

and so

$$S_n = \frac{r - r^{n+1}}{1-r}$$

which as n gets large tends to

$$\begin{aligned} S_n &= \frac{r}{1-r} \\ &= \frac{1}{1-r} - \frac{1-r}{1-r} \\ &= \frac{1}{1-r} - 1 \\ &= \frac{1}{1-\beta_{12}\beta_{21}} - 1 \end{aligned}$$

This formula generalizes to the case of a matrix of such effects between more than a pair of variables. In general (see LISREL 7 manual, Jöreskog & Sörbom, 1989, p. 34)

$$\mathbf{B} + \mathbf{B}^2 + \mathbf{B}^3 + \ldots = (\mathbf{I} - \mathbf{B})^{-1} - \mathbf{I}.$$

Another way to see this formulation is directly from the structural equations. Figure 1.3 shows a multivariate path diagram of the LISREL structural equation model, where η variables are caused by two sets of independent variables, ξ and ζ.

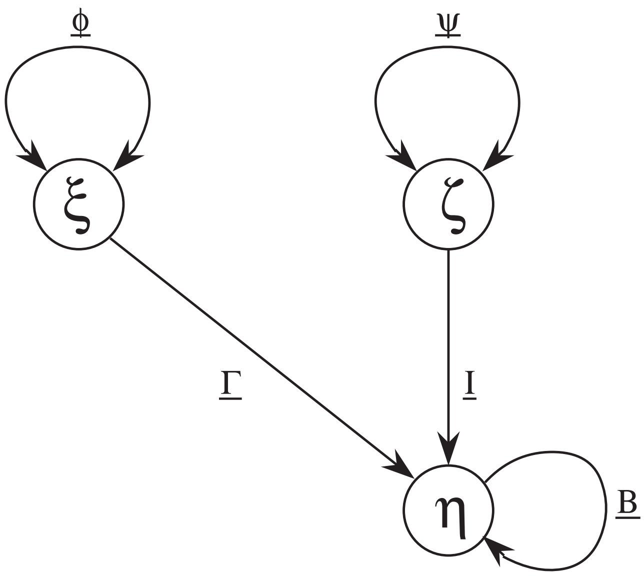


Figure 1.3: LISREL structural equation model for η variables

In addition, the η variables may cause each other, hence the unidirectional arrow from η to itself. Algebraically, the model for the η variables is:

$$\eta = \mathbf{B}\eta + \mathbf{\Gamma}\xi + \zeta$$

Here η appears on both sides of the equation, and we want it solely in terms of the other variables in the model. Hence

$$\begin{aligned}
\eta - \mathbf{B}\eta &= \mathbf{\Gamma}\xi + \zeta \\
(\mathbf{I} - \mathbf{B})\eta &= \mathbf{\Gamma}\xi + \zeta \\
(\mathbf{I} - \mathbf{B})^{-\mathbf{1}}(\mathbf{I} - \mathbf{B})\eta &= (\mathbf{I} - \mathbf{B})^{-\mathbf{1}}\mathbf{\Gamma}\xi + (\mathbf{I} - \mathbf{B})^{-\mathbf{1}}\zeta \\
\eta &= (\mathbf{I} - \mathbf{B})^{-\mathbf{1}}(\mathbf{\Gamma}\xi + \zeta)
\end{aligned}$$

This is the so-called Structural Equation Model in the LISREL manual (p.4) and together with the measurement models for $\mathbf{x}$ and $\mathbf{y}$, defines the entire LISREL model:

$$\begin{aligned}
\eta &= \mathbf{B}\eta + \mathbf{\Gamma}\xi + \zeta \\
\mathbf{y} &= \mathbf{\Lambda_y}\eta + \epsilon \\
\mathbf{x} &= \mathbf{\Lambda_x}\xi + \delta
\end{aligned}$$

In general, this model could be specified in Mx but to do so would reduce the efficiency and elegance with which models may be defined using general matrix formulae.

Chapter 2

Outline of Mx scripts and Data Input

2.1 Preparing Input Scripts

2.1.1 Commands, Numeric Input and Comments

Input files should be prepared with the text editor of your choice. If you use a wordprocessor (such as Wordperfect or Wordstar) the input file should be saved in DOS text (ASCII) format.

You may put comments anywhere in your input file using the character '!'. The Mx command processor ignores:

- All characters following `!` on any line
- Blank lines
- Anything after column 120

The processor is also entirely insensitive to case, except for filenames under UNIX. Basically, Mx reads two things: *keywords* and *numbers*. Unless explicitly stated otherwise, the first two letters of a keyword are sufficient to identify it. Keywords are separated by one or more blank spaces. Once the program has identified a keyword you can extend it to anything you like *as long as it doesn't have a blank character in it*, so `Data` and `Data_silly_words_` have the same effect.

Quite often, a keyword has the format `KEY=123` where `123` is a numeric value to be input. This is called a *parameter*. Mx ignores all (*including blanks*) non-numeric characters found between recognition of a parameter and reading a number, so that `NI=100` and `NInput_vars a lot of words 100` have the same effect.

2.1.2 Syntax Conventions

The syntax described for commands follows these conventions:

- alternatives are represented by `/`
- optional parameters or keywords are enclosed by { and }
- items to be substituted according to the specific application are enclosed by `<` and `>`

2.1.3 Job Structure

Mx has been written for multiple groups, since genetically informative data generally comprise information on different types of relatives which form distinct groups. In the first group of an Mx input file, you have to say how many groups there are (this is so that several Mx programs can be run in one file). A group begins with a title line that contains from 1 to 120 characters for reference. The second line is the Data line, and the group ends with an Output line. What happens in between varies according to what *type* of group it is. Currently there are 3 types:

- DATA - containing data to be analysed (default).
- CALCULATION - allowing matrix operations for output or to simplify structure.
- CONSTRAINT - expressing linear and non-linear constraints between the parameters.

Any number of each type of group can be specified, in any order. Unless one of the keywords `Const` or `Calc` appears on the data line, Mx expects to read a `Data` group. Effectively, there are 3 things to do:

- Supply the data
- Describe the model
- Request options

To do this, the input script will have the following structure:

1. User's TITLE
2. DATA: indicate group type: data/calculation/constraint
3. Read, select any observed data, labels
4. MATRICES: define at least one matrix
5. MODEL: define matrix formula: covariance/means/threshold/compute

6. Specify numbers and parameters, starting values, equality constraints
7. Request fit functions, statistical output and optimization options, multiple fit mode, save matrices and job specification
8. END command

Steps 1-3 supply data and are described in Section 2.2, 4-6 define the model (Section 3), and 7-8 requests output (Section 4.1). Constraint and calculation groups do not read any data, so they omit step 3.

2.1.4 Single Group Example

For example, an input file may look like this:

```
Simple MX example file
Data NGroups=1 NObservations=150 NInput_variables=2
CMatrix
1.2
.8 1.3
Matrices
A Full 2 1
D Diag 2 2
Covariance_model A*A' + D /
Specification A
1 2
Specification D
0 3
End
```

This would fit, by maximum likelihood (the default) a factor model to a covariance matrix calculated from 150 observations of two variables. The model is shown as a path diagram in Figure 2.1. Details of this example will be found in the following sections.

2.2 Data Groups

Every group has to begin with a Title line and a Data line. Optionally, these may be followed by reading of data. These commands are described in this section.

2.2.1 Title Line

The title line is purely for the user's reference, it is printed when Mx prints the parameter specifications and the parameter estimates for a group. It is most useful when there are multiple groups. The title line is recognized by its location rather than a keyword at the start of a line.

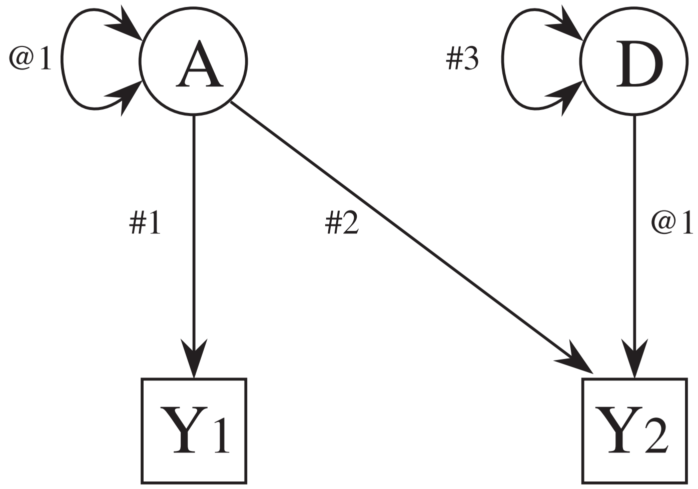


Figure 2.1: Free parameters are indicated by # and fixed parameters are indicated by @. Causal paths are shown as single headed arrows and correlational paths are shown as double-headed arrows.

2.2.2 Data Line

Syntax:

```
Data {Calc/Const} {NGroups=n NInput_vars=n NObservations=n}
```

where `Calc` *defines a calculation group and* `Const` *a constraint group, the default being a* `Data` *group*

Every group must have a data line. It has a number of parameters to indicate

i. if it is the first group, `NGroups`, the number of groups

ii. what kind of group is being input, and

iii. various characteristics (the number of input variables `NInput_vars` and the number of observations `NObservations`) of the data to be analyzed, if any .

The parameters may be specified in any order, and are summarized in Table 2.1. Note that `Data` groups must have `NInput_vars` and `NObservations` keywords. `Constraint` groups only require `NInput_vars`, and `Calculation` groups need no parameters except `NGroups` if it is the first group.

Table 2.1: Parameters of the Data line in Mx input files.

Parameter	Function	Required for group(s)
`Calc`	Specifies a calculation group	Calculation
`Const`	Specifies a constraint group	Constraint
`NGroups=` n	Number of groups	First group
`NInput_vars=` n	Number of input variables	Data, constraint
`NObservations=` n	Number of observations	Data

2.3 Commands for Reading Data

2.3.1 Covariance and Correlation Matrices

Syntax:

`CMatrix/KMatrix/PMatrix {Full} {File=filename}`

In a data group, a covariance matrix may be read using the keyword `CMatrix`. By default, `CMatrix` expects to read *the lower triangle* of an `NInput_vars` × `NInput_vars` matrix, from the input file. If the keyword `Full` appears, then a full matrix will be read. The matrix is read in free format, that is, the numbers are expected to be separated by one or more blank spaces or carriage returns. If the keyword `File` appears, then Mx will read the data from a file. This latter method is generally to be preferred, since it keeps the data in one place. If the data are changed, it is not necessary to change every file that uses these data. A FORTRAN format [in parentheses, e.g., `(6F10.5)`] for reading data can be read as the first line of a data file.

Correlation matrices (`KMatrix`) and matrices of polychoric and polyserial correlations (`PMatrix`) are read in the same way as covariance matrices (`CMatrix`). Although the diagonal elements of these matrices are all 1.0, and could in principle be omitted, Mx *insists on reading them.*

2.3.2 Asymptotic Variances and Covariances

Syntax:

`ACov/AVar/AInv {File=filename}`

In order to use asymptotic weighted least squares (AWLS see Section 4.2.1) or diagonally weighted least squares (DWLS, see Section 4.2.1) it is necessary to read a weight matrix. For compatability with LISREL and PRELIS (Jöreskog & Sörbom, 1986; 1993), Mx expects to receive a weight matrix multiplied by the number of observations. If the File= option is used, a PRELIS output file (created with the SA=filename or the SV=filename PRELIS commands) may be read. By default, Mx expects to receive an asymptotic weight matrix (ACov) whose size depends on (i) NInput_vars and (ii) whether a correlation matrix or covariance matrix has been input. If NInput_vars=k, then if CMatrix has been input, the number of rows in ACov is

$$p = k(k + 1)/2$$

or if PMatrix or KMatrix have been input, the number of rows in ACov is

$$q = k(k - 1)/2.$$

If you use PRELIS, please be sure to use PRELIS 2 instead of PRELIS 1, as the off-diagonal elements of the asymptotic weight matrices produced by PRELIS 1 appear quite inaccurate at the time of writing (PRELIS 1.20 and earlier). The weight matrices can thus be very large,

$$p(p + 1)/2$$

or

$$q(q + 1)/2.$$

An ACov line makes AWLS the default method of estimation for that group. If AWLS is requested on the Options line in a group without an ACov, and error will result. Similarly, DWLS is default if AVar is read.

Note that inverting the asymptotic covariance matrix can take an appreciable amount of time for large problems. Two facilities are available to combat this problem. First, the inverse of the matrix can be read instead. A simple Mx job could be used to invert and save the inverse, for example:

```
Commands to invert a 325x325 asymptotic weight matrix
Data Calculate NGroups=1
Matrices
P Symm 325 325
Compute  P~ /
Matrix P File=weight.asy
Output MX%E=weight.inv
```

The inverse of the asymptotic matrix (AInv), saved in the file weight.inv could be used in place of the matrix itself, with a command of the form:

```
AInv Full File=weight.inv
```

The Full keyword is essential here because Mx is agnostic about the symmetry of square matrices created in calculation groups. It is safer to assume that it is not symmetric to maintain consistency across applications. Alternatively, the inverse would implicitly be saved if the binary save feature described in Section 4.4.3 were used, which saves the whole job specifications.

2.3.3 Raw Data and Variable Length Files

Syntax:

```
Rawdata/VLength {File=filename}
```

Mx will read two types of raw data. `Rawdata` reads *rectangular* data, i.e. where every observation has the same number of input variables (`NInput_vars` on the `Data` line). This is appropriate if there are no missing data, or if missing data have been imputed. If the `Fix` keyword is used, the data are read from a file which should have a FORTRAN format as the first line. If the first line just has `*` or `(*)` on it, the data are read in *free format,* i.e. numbers are separated by one or more spaces or new line characters.

`VLength` is a variable length record reader, which allows reading of raw data where there may be missing values. The default (and mandatory) format for these data is *free.* A line with comments or `*` can be placed at the start of the file, but it will be ignored by Mx except for printing a warning and the line itself in the output file. The structure of a `VLength` file is as follows:

- number of input variables (k)
- identification codes for the k variables
- observed data for the k variables.

For every case, the number of input variables has to be on a line by itself. The identification codes must be integers that correspond to codes read by the `ICodes` command (see Section 3.5.2). For example, a file might contain the following:

```
3
1 2 3 .33 .62 .95
2
2 3 1.4 -2.2
1
2 .37
```

This example reads 3 variables for the first observation, with identification codes 1 2 3, and data values .33 .62 and .95. The second observation has no data for variable 1, but supplies data for 2 and 3, while the third supplies data for variable 2 alone. By default, data of this type are fitted using the raw maximum likelihood fit function (see Section 4.2.2).

It is quite simple to prepare `VLength` files with SAS or SPSS. However, caution should be exercised with SAS which uses a `.` for a missing value. Depending on the operating system under which you are running Mx, this dot may produce a file read error or be read as a zero. Here are a few lines of SAS code to output a `VLength` file from an array of two variables `V{2}`, either or both of which may be missing. The third and fourth lines need to be modified to declare the length of the array and to copy the required variables to the array into it. Certain applications may also need to change the format of the PUT statement that writes the data values.

```
DATA ONE; SET ZERO;
COUNT=0;
NVAR=2;                /*  Number of variables in total !!Change!! */
ARRAY V{2} AT1 AT2;    /*  Set up array for variables   !!Change!! */
DO I=1 TO NVAR;        /*  Count the non-missing observations */
 IF V{I} NE . THEN DO;
  COUNT+1;
 END;
END;
FILE MXVLFILE;          /* Filename for future Mx input !!Change!! */
IF COUNT NE 0 THEN DO; /* Write observations if there are any */
 PUT COUNT;
 DO I=1 TO NVAR;
  IF V{I} NE . THEN PUT I @@;             /* Write the identifiers */
 END;
 PUT;
 DO I=1 TO NVAR;
  IF V{I} NE . THEN PUT V{I} 13.6 +1 @@; /* Write the data values */
 END;
 PUT;
END;
```

2.3.4 Contingency Tables

Syntax:

`CTable <r> <c> {File=filename}`

Mx will read contingency tables of order `r` by `c`. `NInput_vars` must be 2 for a group reading a contingency table. Both `r` and `c` must be greater than 1 but they do not have to be equal. A contingency table contains frequency data (or counts) such that each cell C_{ij} indicates the number of observations falling in row category i and column category j. The frequencies supplied should be greater than or equal to zero. Mx will automatically handle incomplete ascertainment which the user can flag by supplying a negative number for cells that have not been ascertained. See Section 4.2.3 for an example that cross-classifies schizophrenic proband disease status with disease status in their relative. See Section 3.2.4 for details on specifying a matrix to contain estimates of thresholds.

The ordering of the categories should follow the natural numbering of the rows and columns, so that a table with a strong positive correlation between the variables would have large frequencies on the leading diagonal. Supplying a `CTable` changes the default fit function to the likelihood of observing the frequencies assuming a bivariate normal distribution of liability underlies the observed presence in a cell. See Section 4.2.3 for details on fitting structural models to contingency table data.

2.3.5 Means

Syntax:

```
Means {File=filename}
```

A vector of means, length `NInput_vars` may be read. When fitting models by maximum likelihood, a model for the mean structure may be provided. The joint likelihood of the means and the covariances is maximized, enabling tests of hypotheses about equality of means across variables or across groups. Similar tests are possible when models are fitted by maximum likelihood to raw data. However, for balanced data (in which all variables have been obtained from all subjects) considerable computer time may be saved by fitting to the means and covariances.

2.3.6 Higher Moment Matrices

Syntax:

```
Skewness/Kurtosis {File=filename}
```

Matrices of skewness and kurtosis may be read with these commands. These are provided for future developments in Mx that will allow model fitting to these types of data in addition to means and covariances. Currently there is no facility to use matrices read in this way. However, model fitting with higher moments could be done with user-specified functions (see Section 4.2.4).

2.4 Label and Select Variables

2.4.1 Labelling Input Variables

Syntax:

```
Labels <list of labels> /
```

Labels may be given for the observed data by issuing a `Label` command, before the `Matrices` command. These labels may be used to select variables, for example:

```
Data NGroups=1 NInput_vars=3 NObservations=171
CMatrix File=Cov.mat
Labels ALC1 ALC2 AGE
Select ALC2 ALC1 /
```

would read the lower triangle of a 3×3 covariance matrix from the file Cov.mat, and label the variables ALC1 ALC2 and AGE. The variables ALC2 and ALC1 are then selected for analysis, changing their original order. Note that a space is required before the '/' character when using labels. See also Section 3.5.1 for details on labelling specified matrices.

2.4.2 Select Variables

Syntax:

```
Select <numlist or varlist> /
```

Variables may be selected for analysis using the `Select` command. It may be used to reorder data or to pick a selected number of variables for analysis. In either case, *a / must end the command.* `Select` accepts *integers* which correspond to the order of the input variable. More conveniently, `Select` will operate on variable labels (see Section 2.4.1). The command will work with raw data as supplied by the `Rawdata` or `VLength` commands (see Sections 4.2.2 and 2.3.3).

Select with Variable Length Data

In combination with the `VLength` command, it will change the identification codes to consecutive integers starting at 1. For example, if the following `Select` line was read:

```
Select 3 4 2 /
```

a `VLength` record of the form:

```
4
4 3 2 1  .1 .2 .3 .4
```

would effectively be changed to:

```
3
1 2 3 .2 .1 .3
```

thus the observation originally numbered 3 has become observation 1, observation numbered 4 has become observation 2, and observation numbered 2 has become observation 3. `Select` will automatically reduce the number of data vectors if there are no matches for a particular data vector and the codes in the `Select` line. The final number of vectors and observations used in the analysis is given in the output file.

`Select` cannot contain more numbers than the `NInput_vars` specified on the `Data` line. To do so would necessarily result in a singular correlation or covariance matrix. Likewise, the same variable cannot be selected twice.

2.5 Calculation and Constraint Groups

The use of calculation and constraint groups is very similar the use of groups that read data. The three types of group are fully command compatible with the exception of commands for reading data, which can be used by data groups alone.

2.5.1 Calculation Groups

The keyword `Calc` on the `Data` line indicates that the group is used for calculation. The calculated matrix formula from such a group is printed if the `RSiduals` command appears on the `Options` line. There are no restrictions on the type and dimensions of a matrix than can be produced with this command (other than memory limits). The result of the calculation may be used in later groups by using the `=%En` syntax when specifying a matrix, where `n` is the number of the calculation group. Note that there is a strict ordering within the input file; results cannot be taken from a calculation that has not yet occurred.

The `Calc` group provides a facility for printing out results of matrix operations. Any calculation group that is not followed by a constraint or data group is not calculated until the end of optimization, thus avoiding unnecessary waste of computer time.

2.5.2 Constraint Groups

Constraint groups, indicated by `Const` on the `Data` line, set up a matrix of nonlinear constraints. The matrix is symmetric, and is of order k, where k is the number of input variables (`NInput_vars`) specified on the input line. Thus the number of constraints in each such group is $k(k+1)/2$. Any nonlinear equality constraint can be expressed in the form

$$C\{\Theta\} = 0$$

where $C\{\Theta\}$ is a function of the free parameters. For example, if `NInput_vars=3`, six nonlinear constraints would be specified. The expected covariance matrix for the group would be structured:

	1	2	3
1	C_1		
2	C_2	C_3	
3	C_4	C_5	C_6

The optimizer would find values of the parameters that keep the C_i very close to zero. One way to think of these constraints is to imagine that the group has a very large sample size and has an observed covariance matrix full of zeroes.

Mx increases the number of degrees of freedom by the number of nonlinear constraints, which assumes that each constraint identifies a parameter. This is obviously not always the case, and the `DF` parameter on the `Options` line may be used to correct for failures of this assumption.

The optimization routine treats constraints in an intelligent fashion; if it finds the derivatives of the constraint functions with respect to certain parameters to be zero, it does not calculate them during optimization. This means that if some of the specified constraint functions are always zero, little additional computational cost is incurred. If there is no feasible point for the constraints, for example one of them always takes the value .5, an IFAIL=3 error message is returned.

Chapter 3

Building Models with Matrices

All groups, be they constraint, calculation, or data, require at least one matrix in order to operate. The next few sections describe the types of matrix that may be used, the operators that act on and between them, and ways of putting parameters and numbers into them.

3.1 Matrices Command

Syntax:

```
Matrices
<matrix name> <type> <rows> <columns> {= <name> <group> / Free}
<matrix name> <type> <rows> <columns> {= <name> <group> / Free}
.
.

<matrix name> <type> <rows> <columns> {= <name> <group> / Free}
```

A group must have the *3-letter* `MAT` command, followed by at least one matrix definition. The end of `Matrices` reading is signalled by an `Covariance`, `Means`, or `Threshold` command. Matrix names are restricted to one letter, from A to Z. The type of a matrix may be one of the 11 forms described in Table 3.1, and its row and column dimensions are specified with integers. Once the type and size of a matrix has been defined, it cannot be changed. Note that matrix definitions are *group specific;* for example, matrix A in group 1 does not have to be the same type or size as matrix A in group 2.

Optionally, a matrix may be *constrained* to equal a matrix previously specified. For example, we could use the command

Table 3.1: Types of matrices that may be specified in Mx. # free elements indicates the number of elements that can be altered by the user, where r is the number of rows and c the number of columns of the matrix.

Type	Structure	Shape	# of free elements
`Zero`	Every element is zero (null matrix)	Any	0
`Unit`	Every element is one (unit matrix)	Any	0
`Iden`	Identity matrix	Square	0
`IZero`	Identity | Zero partitioned matrix	Any	0
`ZIden`	Zero | Identity partitioned matrix	Any	0
`Diag`	Diagonal matrix	Square	r
`Sdiag`	Subdiagonal (zeros on & above diagonal)	Square	$r(r-1)/2$
`Stand`	Standardized (symmetric, ones on diagonal)	Square	$r(r-1)/2$
`Symm`	Symmetric	Square	$r(r+1)/2$
`Lower`	Lower triangular	Square	$r(r+1)/2$
`Full`	Full	Any	$r \times c$

```
A Symm 3 3 = Y2
```

to equate matrix A in this group to matrix Y in group 2. Note that the current group must be number three or greater. Several additional options allow constraints to other quantities found in previous groups, such as the observed or expected covariance matrix. The special codes for constraining a matrix to equal those defined or computed in previous groups are shown in Table 3.2. These add to the flexibility of Mx.

It is especially important to note that *none of the* `%E, %O, %M, %F` *and* `%P` *equalities may refer to groups that appear after the current group.* When matrices are constrained to be equal in this fashion, the type and row $\times$ column dimensions of the earlier matrix are retained. If the two specifications do not agree, a warning is printed. Both the number of rows and the number of columns must be supplied for square matrices, but only the first is used to define the size of the matrix.

All changeable elements of matrices are initialized at zero and are fixed parameters, unless the `Free` keyword is used, in which case each changeable element is specified as a different free parameter. Examples of the results of using the keyword `Free` are shown in Table 3.3.

More detail on specifying parameters in matrices is given in Sections 3.3 to 3.4.4.

Table 3.2: Special codes for constraining a matrix to equal one computed or defined in group n. Such equalities may be set only in groups $n+i$ where $i = 0, 1 \ldots Ng$ where Ng is the number of groups in total. Note: NI_n is the number of input variables in group n following any selection; NR and NC are respectively the number of rows and columns in a contingency table, and may be requested only if group n has such a table.

Symbol	Matrix quantity	Dimensions
`%O`n	Observed covariance (data) matrix	$\mathrm{NI}_n \times \mathrm{NI}_n$
`%E`n	Expected covariance matrix	$\mathrm{NI}_n \times \mathrm{NI}_n$
`%M`n	Expected mean vector	$1 \times \mathrm{NI}_n$
`%P`n	Expected proportions under bivariate normal	$\mathrm{NR}_n \times \mathrm{NC}_n$
`%F`n	Function value	1×1

3.2 Covariances, Model or M2 Command

Syntax:

```
Covariances/M2/Model A {operator B {operator C ...}} /
```

The covariance command uses the matrices specified following the `Matrices` line and special symbols to perform operations on or between them. A covariances statement may contain a single matrix and no operations, or it could be very complex. The command may extend over several lines and *must end in a /*

3.2.1 Matrix Operations

In ordinary algebra, operators such as $+$ $-$ $\times$ and $\div$ have an order of evaluation established by convention. Multiply and divide are done before addition and subtraction. Multiply and divide are done in left-to-right order if they appear consecutively, as are addition and subtraction. We could say then, that $\times$ and $\div$ have priority 1, and $+$ and $-$ have priority 2. These default priorities can be changed with the use of brackets () which specify that operations inside the brackets are done first. For example,

$$a + b \times c = a + bc$$

whereas

$$(a + b) \times c = ac + bc$$

A similar hierarchy has been established for the matrix operators in Mx, and again it may be revised by the use of brackets. Table 3.4 shows the matrix operators and their (default) order of evaluation.

Table 3.3: Examples of use of the matrices command to specify the dimensions of different matrix types. The keyword `Free` following each command makes each modifiable element in the matrix a separate free parameter, numbered in order as shown in the second column. In the third column, values of elements are shown, with ? representing a free parameter.

Example command	Parameter Matrix	Values
`A Zero 2 3 Free`	0 0 0 0 0 0	0 0 0 0 0 0
`B Unit 2 3 Free`	0 0 0 0 0 0	1 1 1 1 1 1
`C Iden 3 3 Free`	0 0 0 0 0 0 0 0 0	1 0 0 0 1 0 0 0 1
`D IZero 2 5 Free`	0 0 0 0 0 0 0 0 0 0	1 0 0 0 0 0 1 0 0 0
`E ZIden 2 5 Free`	0 0 0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 0 0 1
`F Diag 3 3 Free`	1 0 0 0 2 0 0 0 3	? 0 0 0 ? 0 0 0 ?
`G SDiag 3 3 Free`	0 0 0 1 0 0 2 3 0	0 0 0 ? 0 0 ? ? 0
`H Stand 3 3 Free`	0 1 2 1 0 3 2 3 0	1 ? ? ? 1 ? ? ? 1
`I Symm 3 3 Free`	1 2 4 2 3 5 4 5 6	? ? ? ? ? ? ? ? ?
`J Lower 3 3 Free`	1 0 0 2 3 0 4 5 6	? 0 0 ? ? 0 ? ? ?
`K Full 2 4 Free`	1 2 3 4 5 6 7 8	? ? ? ? ? ? ? ?

Table 3.4: Matrix operators available in Mx, together with their priority for evaluation. See also Table 3.5 for matrix functions.

Symbol	Name	Function	Example	Priority
~	Inverse	Inversion	`A~`	1
'	Transpose	Transposition	`A'`	1
^	Power	Element powering	`A ^ B`	2
*	Star	Multiplication	`A * B`	3
.	Dot	Dot product	`A . B`	3
@	Kron	Kronecker product	`A @ B`	3
%	Eldiv	Element division	`A % B`	3
+	Plus	Addition	`A + B`	4
-	Minus	Subtraction	`A - B`	4
|	Bar	Horizontal adhesion	`A | B`	5
_	Under	Vertical adhesion	`A _ B`	6

A line has been drawn between the first two operators (Inverse & Transpose) and the rest because inverse and transpose are *unary* operators, that is, they operate on *one* matrix. The rest form a single new matrix from two matrices, and are thus *binary* operators. These operators are now described in detail.

Readers unfamiliar with matrix algebra may benefit from reading Appendix C, where examples and exercises are given. Readers familiar with matrix algebra may wish to examine Tables 3.4 and 3.5 for the variety of available operators and functions, and use Sections 3.2.1 and 3.2.2 for reference.

Inverse ~

Only square matrices may be inverted, but they may be either symmetric or non-symmetric. The inverse of matrix $\mathbf{A}$ is usually written $\mathbf{A}^{-1}$ and implies that

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix. To request an inverse with Mx, we use the symbol `~`. If the inverse does not exist (possibly due to rounding errors), Mx will terminate with an error message. Some precautions can be taken to avoid this, such as supplying starting values that allow inversion, or putting boundary constraints on parameters to prevent their taking values that would lead to a singular matrix.

Transpose ′

Any matrix may be transposed. The transpose of $\mathbf{A}$ is written $\mathbf{A}'$. The order of the matrix changes from $r \times c$ to $c \times r$, as the rows become the columns and vice-versa.

Power ^

All the elements of a matrix may be raised to a power using the ^ symbol. The matrix to be powered may be of any dimension, but the power to which each element must be raised has to be specified in a 1×1 matrix. It is possible to use negative powers and non-integer exponents to indicate reciprocal functions and roots of elements, but note that it is not possible to raise a negative number to a non-integer power. For example, the cube of every element of a matrix would be obtained by `A^B` if `B` was a 1×1 matrix with 3 as its only element.

Star $*$

Star is the ordinary form of matrix multiplication. The elements of $\mathbf{A}$ $(m \times n)$ and $\mathbf{B}$ $(n \times p)$ are combined to form the elements of matrix $\mathbf{C}$ $(m \times p)$ using the formula

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \times B_{kj}$$

Matrices multiplied in this way have to be *conformable for multiplication.* This means that *the number of columns in the first matrix must equal the number of rows in the second matrix.*

For example, the matrix product $\mathbf{A} * \mathbf{B}$

$$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} * \begin{pmatrix} g & h \\ i & j \end{pmatrix} = \begin{pmatrix} a \times g + b \times i & a \times h + b \times j \\ c \times g + d \times i & c \times h + d \times j \\ e \times g + f \times i & e \times h + f \times j \end{pmatrix}$$

$$= \begin{pmatrix} ag + bi & ah + bj \\ cg + di & ch + dj \\ ge + fi & eh + fj \end{pmatrix}$$

Dot .

Dot is another type of matrix multiplication, which is done *element by element.* For two matrices to be multiplied in this way, they have to have the same dimensions, i.e. the number of rows in $\mathbf{A}$ has to be the same as the number of rows in $\mathbf{D}$, and the number of columns in $\mathbf{A}$ has to be the same as the number of columns in $\mathbf{D}$. Elements of the dot product are described by the formula

$$C_{ij} = A_{ij} \times D_{ij}$$

For example, the dot product $\mathbf{A}.\mathbf{D}$ is

$$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} . \begin{pmatrix} g & h \\ i & j \\ k & l \end{pmatrix} = \begin{pmatrix} a \times g & b \times h \\ c \times i & d \times j \\ e \times k & f \times l \end{pmatrix}$$

Kron $\otimes$

The *right* Kronecker product of two matrices $\mathbf{A} \otimes \mathbf{B}$ is formed by multiplying each element of $\mathbf{A}$ by the matrix $\mathbf{B}$. If $\mathbf{A}$ is of order $(m \times n)$ and $\mathbf{B}$ is of order $(p \times q)$, then the result will be of order $mp \times nq$. There are no conformability criteria for this type of product. The symbol $\otimes$ is imitated in Mx input files with the symbol `@`.

For example, the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is

$$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} \otimes \begin{pmatrix} g & h \\ i & j \end{pmatrix} = \begin{pmatrix} a \times g & a \times h & b \times g & b \times h \\ a \times i & a \times j & b \times i & b \times j \\ c \times g & c \times h & d \times g & d \times h \\ c \times i & c \times j & d \times i & d \times j \\ e \times g & e \times h & f \times g & f \times h \\ e \times i & e \times j & f \times i & f \times j \end{pmatrix}$$

Eldiv %

Eldiv does *element by element* division. For two matrices to be divided in this way, they have to have the same dimensions, i.e. the number of rows in $\mathbf{A}$ has to be the same as the number of rows in $\mathbf{D}$, and the number of columns in $\mathbf{A}$ has to be the same as the number of columns in $\mathbf{D}$. Elements of the result are described by the formula

$$C_{ij} = A_{ij} \div D_{ij}$$

If any element of $\mathbf{D}$ is zero, the corresponding cell in the result matrix is set to 10^{35}.

For example, the eldiv $\mathbf{A\%D}$ is

$$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} \begin{pmatrix} g & h \\ i & j \\ k & l \end{pmatrix} = \begin{pmatrix} a \div g & b \div h \\ c \div i & d \div j \\ e \div k & f \div l \end{pmatrix}$$

Plus +

Addition of matrices is performed *element by element.* For two matrices to be added, they have to have the same dimensions, i.e. the number of rows in $\mathbf{A}$ has to be the same as the number of rows in $\mathbf{D}$, and the number of columns in $\mathbf{A}$ has to be the same as the number of columns in $\mathbf{D}$. Elements of the sum, $\mathbf{C}$ are described by the formula

$$C_{ij} = A_{ij} + D_{ij}$$

For example, the sum $\mathbf{A} + \mathbf{D}$ is

$$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} + \begin{pmatrix} g & h \\ i & j \\ k & l \end{pmatrix} = \begin{pmatrix} a+g & b+h \\ c+i & d+j \\ e+k & f+l \end{pmatrix}$$

Minus $-$

Subtraction of matrices is performed *element by element.* For one matrix to be subtracted from another, they have to have the same dimensions, i.e. the number of rows in **A** has to be the same as the number of rows in **D**, and the number of columns in **A** has to be the same as the number of columns in **D**. Elements of the difference, **C** are described by the formula

$$C_{ij} = A_{ij} - D_{ij}$$

For example, the difference $\mathbf{A} - \mathbf{D}$ is

$$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} - \begin{pmatrix} g & h \\ i & j \\ k & l \end{pmatrix} = \begin{pmatrix} a-g & b-h \\ c-i & d-j \\ e-k & f-l \end{pmatrix}$$

Note that in Mx there is also a *unary* minus operator, so that an expression such as $-\mathbf{A}$ is legal. This operation changes the sign of each element of **A**

Bar |

Bar allows partitioning of matrices. Its operation is called horizontal adhesion because $\mathbf{A}|\mathbf{D}$ is formed by sticking **D** onto the right hand side of **A**. For two matrices to be adhered in this way, they have to have the same number of rows. If **A** $(m \times n)$ and **D** $(m \times p)$ are adhered, the result **C** is of order $(m \times (n + p))$.

For example, the operation $\mathbf{A}|\mathbf{D}$ is

$$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} | \begin{pmatrix} g & h \\ i & j \\ k & l \end{pmatrix} = \begin{pmatrix} a & b & g & h \\ c & d & i & j \\ e & f & k & l \end{pmatrix}$$

Under _

Under allows partitioning of matrices. Its operation is called vertical adhesion because **A_D** is formed by sticking **D** underneath **A**. For two matrices to be adhered in this way, they have to have the same number of columns. If **A** $(m \times n)$ and **D** $(p \times n)$ are adhered, the result **C** is of order $((m + p) \times n)$.

For example, if

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}$$

and

$$\mathbf{D} = \begin{pmatrix} g & h \\ i & j \\ k & l \end{pmatrix}$$

then

$$\mathbf{A\_D} = \begin{pmatrix} a & b \\ c & d \\ e & f \\ g & h \\ i & j \\ k & l \end{pmatrix}$$

3.2.2 Matrix Functions

A number of matrix functions, shown in Table 3.5, have been provided in Mx, which are especially useful for in specialized applications involving user-defined fitting-functions (`US`; see Section 4.2.4, page 54). Functions differ from operators because they take an *argument* enclosed by brackets (). This argument may be a single matrix name, or a complex matrix formula. The argument is evaluated *before* the function is applied, consistent with the rules for using brackets. Functions form a second set of unary operators (see Section 3.2). Descriptions of these functions follow.

Trace \tr()

The trace of a matrix is the sum of the elements on the leading diagonal, i.e.

$$\sum_{i=1}^{n} A_{ii}$$

It is only allowed for *square* matrices.

Determinant \det()

Properties of determinants, and ways of calculating them are discussed in Appendix C. `\det()` is only calculated for *square* matrices.

Sum \sum()

The sum of a matrix is the sum of all its elements, i.e.,

$$\sum_{i=1}^{r}\sum_{j=1}^{c} A_{ij}$$

Product \prod()

The product function of a matrix yields the product of all its elements, i.e.,

$$\prod_{i=1}^{r}\prod_{j=1}^{c} A_{ij}$$

Table 3.5: Matrix functions available in Mx. Restrictions are on rows (r) and columns (c) of input argument.

Keyword	Function	Restrictions	Result Dimensions
`\tr()`	Trace	$r = c$	(1×1)
`\det()`	Determinant	$r = c$	(1×1)
`\sum()`	Sum	$None$	(1×1)
`\prod()`	Product	$None$	(1×1)
`\max()`	Maximum	$None$	(1×1)
`\min()`	Minimum	$None$	(1×1)
`\abs()`	Absolute value	$None$	$(r \times c)$
`\cos()`	Cosine	$None$	$(r \times c)$
`\cosh()`	Hyperbolic Cosine	$None$	$(r \times c)$
`\sin()`	Sin	$None$	$(r \times c)$
`\sinh()`	Hyperbolic Sin	$None$	$(r \times c)$
`\tan()`	Tan	$None$	$(r \times c)$
`\tanh()`	Hyperbolic Tan	$None$	$(r \times c)$
`\exp()`	Exponent ($e^{\mathbf{A}}$)	None	$(r \times c)$
`\ln()`	Natural logarithm	None	$(r \times c)$
`\sqrt()`	Square root	None	$(r \times c)$
`\d2v()`	Diagonal to Vector	None	$(\min(r, c) \times 1)$
`\v2d()`	Vector to Diagonal	$r = 1$ or $c = 1$	$(\max(r, c) \times \max(r, c))$
`\m2v()`	Matrix to Vector	None	$(rc \times 1)$
`\vec()`	Matrix to Vector†	None	$(rc \times 1)$
`\vech()`	Lower triangle to Vector	None	$(rc \times 1)$
`\stnd()`	Standardize matrix	$r = c$	$(r \times c)$
`\eval()`	Real eigenvalues	$r = c$	$r \times 1$
`\evec()`	Real eigenvectors	$r = c$	$r \times r$
`\ival()`	Imaginary eigenvalues	$r = c$	$r \times 1$
`\ivec()`	Imaginary eigenvectors	$r = c$	$r \times r$

†`vec` vectorizes by columns, in contrast to `m2v`, which vectorizes by rows.

Maximum \max()

The maximum function of a matrix yields a 1×1 matrix containing the maximum of all its elements.

Minimum \min()

The minimum function of a matrix yields a 1×1 matrix containing the minimum of all its elements.

Absolute value \abs()

The abs function replaces all matrix elements with their absolute value.

Trigonometric functions \cos(), \sin() etc.

These functions replace all matrix elements with their appropriate trigonometric transformation, in radians.

Exponent \exp()

Any matrix is a legal argument for this function which replaces each element A_{ij} by $e^{A_{ij}}$.

Natural Logarithm \ln()

Any matrix is a legal argument for this function which replaces each element A_{ij} by $\ln A_{ij}$. If an element is less than 1×10^{-30} then the result is $\ln(1 \times 10^{-31})$. Although error messages would be more normal in such a situation, this behavior can be helpful in optimization.

Square Root \sqrt()

Any matrix is a legal argument for this function which replaces each element A_{ij} by $\sqrt{A}_{ij}$. If an element is less than zero, a fatal error occurs.

Diagonal to Vector \d2v()

The leading diagonal of any matrix is placed into a column vector with $\min(r, c)$ rows, i.e. r or c, whichever is less. e.g. if

$$\mathbf{A} = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \end{pmatrix}$$

then

$$\backslash\text{d2v}(\mathbf{A}) = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Vector to Diagonal Matrix \v2d()

A row or column vector is placed in the leading diagonal of a square matrix. e.g. if

$$\mathbf{E} = \begin{pmatrix} a & b & c & d \end{pmatrix}$$

then

$$\backslash\text{v2d}(\mathbf{E}) = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{pmatrix}$$

Matrix to Vector \m2v

A matrix is placed in a column vector, by rows. Thus if

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

then

$$\backslash\text{m2v}(\mathbf{A}) = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

This is similar to the function `\vec`; which places the matrix into a vector by columns, instead of rows.

Matrix to Vector \vec

A matrix is placed in a column vector, by columns. Thus if

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

then

$$\backslash\text{vec}(\mathbf{A}) = \begin{pmatrix} a \\ c \\ b \\ d \end{pmatrix}$$

Note that it is more efficient to use `\m2v(A)` than `\vec(A')` and more efficient to use `\vec(A)` than `\m2v(A')`. Both functions work for matrices of any shape.

Matrix to Vector \vech

All the elements on the diagonal and below are placed into a vector, by columns. Thus if

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

then

$$\backslash\text{vech}(\mathbf{A}) = \begin{pmatrix} a \\ c \\ d \end{pmatrix}$$

Like its counterparts `vec` and `m2v`, this function will operate on matrices of any shape, terminating at the last row or column, whichever is the lesser. Thus if

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}$$

then

$$\backslash\text{vech}(\mathbf{A}) = \begin{pmatrix} a \\ c \\ e \\ d \\ f \end{pmatrix}.$$

Standardize \stnd

This operation will convert a covariance matrix into a correlation matrix. Replacement of elements is made according to the formula:

$$\mathbf{A}^*_{\mathbf{ij}} = \frac{\mathbf{A}_{\mathbf{ij}}}{\sqrt{\mathbf{A}_{\mathbf{ii}}\mathbf{A}_{\mathbf{jj}}}}$$

The diagonal elements of $\mathbf{A}$ have to be greater than zero, and $\mathbf{A}$ has to be square.

Real Eigenvalues \eval

The real parts of the eigenvalues of a square matrix are placed in a column vector, in ascending order of size, smallest first.

Real Eigenvectors \evec

The real parts of the eigenvectors of a square matrix are placed in a square matrix, where column j contains the eigenvector corresponding to eigenvalue j, with eigenvalues sorted in ascending order of size, smallest first ($j = 1$).

Imaginary Eigenvalues \ival

The imaginary parts of the eigenvalues of a square matrix are placed in a column vector, in ascending order of size, smallest first.

Imaginary Eigenvectors \ivec

The imaginary parts of the eigenvectors of a square matrix are placed in a square matrix, where column j contains the eigenvector corresponding to eigenvalue j, with eigenvalues sorted in ascending order of size, smallest first ($j = 1$).

3.2.3 Means or M1 Command

Syntax:

```
Means/M1 A {operator B {operator C ...}} /
```

The `Means` command operates in the same way as the `Const` or `M2` command. It exists to facilitate the modelling of means. All the matrix operators and functions (Sections 3.2.1 and 3.2.2) may be used just as when specifying a model for covariances. A `/` must end the command. Currently, Mx will do nothing with models for means when applying the functions `LS, GLS, AWLS, DWLS`. Only the `ML`, `US` or `RM` fit functions make use of models for means.

3.2.4 Threshold Command

Syntax:

```
Threshold A {operator B {operator C ...}} /
```

The `Threshold` command operates in the same way as the `Means` to specify means. It exists to facilitate the modelling of thresholds when fitting to contingency table data. All the matrix operators and functions (Sections 3.4 and 3.5) may be used just as when specifying a model for covariances. A `/` must end the command. `Threshold` cannot be used with any fit function other than `CTable` (see Section 4.2.3).

Special restrictions apply to the dimensions of the matrix calculated in the `Threshold` command. The result must have 2 rows and must have at least d columns where $d = \max((r-1),(c-1))$, in other words at least one less than the number of rows or the number of columns in the contingency table, whichever is the greater. The first $(r-1)$ elements of the first row of the matrix will contain the thresholds that separate the rows. The first $(c-1)$ elements of the second row of the matrix will contain the thresholds that separate the columns. In fact, these are unstandardized row and threshold estimates, which may be standardized by dividing by the square root of the product of the two diagonal elements of the expected covariance matrix calculated by the `Covariance` or `Constraint` statement. This allows the testing of models that predict differences in variance between groups, but have equal thresholds.

The user should take care to supply starting values for thresholds that increase from left to right in both rows of the matrix calculated by the `Threshold` command. Ideal starting values are those that, when standardized, mark the $z-$scores on the normal distribution corresponding to the cumulative frequencies of the normal distribution of the row totals (first row of the calculated matrix) or the column totals (second row of the calculated matrix). For example, if the following contingency table was supplied as data:

```
CTable 3 2
20 180
40 360
20 180
```

then appropriate starting values for 2 row thresholds would be -.67 and +.67 (z-scores corresponding to the lower 25% and 75% of the normal distribution), and -1.28 would be appropriate for the starting value of the column threshold (z-score corresponding to the lower 10% of the normal distribution).

3.3 Putting Numbers in Matrices

This section describes three methods of entering numbers into matrices (see Section 3.4 for how to specify elements of matrices to be free, fixed or constrained parameters). In Section 3.1, we saw how matrices could be defined as one of 9 types, such as identity, symmetric, diagonal or full (see Table 3.1), and how their dimensions (rows, r and columns, c) were specified. On inspection of the table, we see immediately that types `ZE ID IZ` and `ZI` have no free elements at all. For example, there is nothing more to know about an `IZ` matrix which has 2 rows and 4 columns. It looks like this:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

and it *cannot be changed at all.* If it was altered, then it would no longer be an `IZ` matrix.

All 5 remaining matrix types have *modifiable* elements which may be altered with the commands `Matrix Start` and `Value`. The number of modifiable elements varies according to:

- The number of rows and columns in the matrix
- The *type* of the matrix

All modifiable elements of a matrix are initialized at zero. The order of elements in a matrix is left to right, by rows. For example, a symmetric (3×3) matrix would be read as:

```
1
2 3
4 5 6
```

See Table 3.3 for more examples on the patterning of matrices.

3.3.1 Matrix Command

Syntax:

```
Matrix <matrix name> {File=filename}
<numlist>
```

where `<numlist>` *is a free format list of numbers. Note that different syntax is required in multiple fit mode:*

Syntax:

```
Matrix <group number> <matrix name> {File=filename}
<numlist>
```

The list length required varies according to matrix type, and size as described at the start of this Section, on page 35. For example, suppose we specify a diagonal matrix **A** with 3 rows and 3 columns. The fourth column in Table 3.1 shows that the number of free elements is equal to r for diagonal elements, so we supply r elements. The command lines

```
Matrix A
.3  5  9
```

or, equivalently

```
Matrix-I-would-like-to-change-is A
0.3D+00
5  9.00000000
```

would result in matrix **A** as:

$$\begin{pmatrix} .3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}$$

The `Matrix` will operate on elements regardless of whether or not they have been specified as fixed or free parameters.

`Matrix` will read its elements from a file with a FORTRAN format on the first line. Such files may have been produced by an earlier run of MX, or by another program. LISREL matrix output files (produced by commands such as gamma=filename on the LISREL OU line) are fully compatible. The files must contain at least as many numbers as required to fill the changeable elements of the matrix specified (see page 35).

3.3.2 Start and Value Commands

Syntax:

```
Start/Value <value> <element list>/ All
```

where `<element list>` *consists of matrix elements (e.g. A 1 2 3) and may include the* `TO` *keyword*

Often, it is not convenient to provide a value for all the elements of a matrix, if only a small number need to be modified. Under these circumstances, it is easier to explicitly change elements by *name.* Elements of matrices may be referred to by up to three subscripts, according to the syntax

```
A {<group>} <row> <col>
```

If the matrix you wish to refer to is in the current group, the group number may be omitted. The numbers `<group> <row> <col>` may be separated by any number of non-numeric or blank characters, so that, for example, to put .5 in row 2 column 3 of group 1's A matrix, you could enter:

```
Value.5 A 1 2 3
```

will work the same as

```
Value.5 A(1,2,3)
```

or even

```
Value.5 A group1 row2 column3
```

but *please note* the *foolish command*

```
Value.5 A column3 group1 row2
```

would do something *entirely different,* placing .5 in the 2nd column of row 1 of matrix A in group 3. *N.B. It is only possible to modify matrices in the current or previous groups.*

The difference between `Start` and `Value` lies in their treatment of elements when the keywords `ALL` or `TO` are used (`-` is a synonym for `TO`). With the keyword `ALL`, `Start` assigns a starting value to every free parameter specified at that point in the input file. `Value` does the opposite – it assigns its value to every *fixed* matrix element specified up to that point. Although `Start` does the same thing if the `TO` keyword is specified, i.e. only apply its value to free parameters, `Value` behaves differently. It will assign a value to all elements in the same specified range, free parameter or fixed.

The `TO` keyword should be used only to specify a range of matrix elements *within the same matrix.*

3.4 Putting Parameters in Matrices

Parallel to the placement of numbers in matrices described in Section 3.3, there are facilities for putting parameters in matrices. Four of these commands are equivalent to the LISREL commands of the same name, and there is a new command (`Specification`) that greatly simplifies specification for models with lots of equality constraints. Note also that all modifiable elements of a matrix can be specified as different free parameters using the keyword `Free` after the matrix is specified (see Section 3.1).

3.4.1 Pattern Command

Syntax:

```
Pattern <matrix name> {File=filename}
<numlist>
```

where `<numlist>` *is a list of* `1`*'s and* `0`*'s. Note that different syntax is required in multiple fit mode:*

Syntax:

```
Pattern <group number> <matrix name> {File=filename}
<numlist>
```

The `Pattern` command is a simple method that has the same syntax as the LISREL command on which it was based. Following the `Pattern <matrix>` command, the user must provide the correct number (see `Matrix` command page 36) of `1`'s and `0`'s for that matrix. A `1` (or any non-zero value) indicates that the element is a free parameter (which may be constrained to equal another free parameter - see the `Equate` command on page 39), and a `0` indicates that the element is to be fixed.

3.4.2 Fix and Free Commands

Syntax:

```
Fix/Free <value> <element list>
```

where `<element list>` *is a list of matrix elements*

The `Fix` and `Free` commands operate directly on specific matrix elements or sets of matrix elements. `Fix` makes a parameter fixed (if it was `Free` before) and `Free` makes an element a free parameter to be estimated. Matrix elements are referred to by group, row and column as described in Section 3.3, page 37. The keywords `TO` and `ALL` (see page 37) may be used to specify ranges of matrix elements to be fixed or freed.

For example, suppose in group 1, matrix A was defined as symmetric, with 4 rows and columns. Initially it would be patterned with zeroes throughout. The command

```
Free a 1 2 1 - a 1 4 3
```

would give the following pattern:

$$\begin{pmatrix} 0 & 1 & 3 & 6 \\ 1 & 2 & 4 & 7 \\ 3 & 4 & 5 & 8 \\ 6 & 7 & 8 & 0 \end{pmatrix}$$

if this command was followed by

```
Fix A 1 4 2
```

the parameter specification would become:

$$\begin{pmatrix} 0 & 1 & 3 & 6 \\ 1 & 2 & 4 & 0 \\ 3 & 4 & 5 & 8 \\ 6 & 0 & 8 & 0 \end{pmatrix}$$

In symmetric matrices, references to the upper triangle are legal also; anything done to an element one side of the diagonal (A_{ij}) is done to the corresponding element on the other side (A_{ji}).

3.4.3 Equate Command

Syntax:

```
Equate <matrix> {<gp>} <r> <c> <matrix> {<gp>} <r> <c> ...} }
```

In order to constrain matrix elements to equal one another, the `Equate` command may be used. Its primary purpose is to specify equality constraints among parameters, but it can be used to copy a numeric value from one matrix element to another. There is a big conceptual difference between the *first* element specified in a list and the others. The fixed or free status of the first element is given to the remaining elements in the list, be they fixed or free. If the first element is a free parameter, the same parameter is copied to the other elements. If the first element is fixed, then a warning message is printed, to the effect that all other elements will be fixed. The value in the first element is then passed to the other elements in the list. The `Equate` command may be used within matrices, or across matrices in the same group, or across matrices in different groups. Note that it is *not* possible to use `Equate` to make an immovable element (such as an element of a matrix specified as type `ID`, or an off-diagonal element of a diagonal matrix) into a free parameter.

For example, given matrices specified in group 1 as follows:

```
Matrices
A Symm 3 3  FR
B Full 2 4
I Identity 6 6
```

the following `Equate` statements are legal:

```
Equate A 1 1 B(2,2) B 1 4
Equate B 2 1 A 1 2 3
Equate BANANA 2 2 APPLE 1 1 1
```

However, the following are *illegal:*

```
Equate A 1 1 I 2 2
Equate I 5 5 B 1 1
Equate A 1 2 C 1 1
Equate A 4 4 B 1 1
Equate A 2 2 G 4 1 1
```

They fail because: I is an identity matrix (1 & 2), `C` has not been specified (3), `A` does not have 4 rows and columns (4), and it is not possible to refer to an element of a matrix in a later group (5).

For large models with many constraints it is often more efficient to use the `Specification` command, or to seek repetitive structures in the model matrices and use partitioned matrices (see Section 5.2.4), or both. The kronecker product can be particularly useful when specifying repetitive partitioned matrix structures.

3.4.4 Specification Command

Syntax:

```
Specification <matrix name>
a { b {c}}
```

where `a, b` *and* `c` *are not necessarily distinct integers. Note that different syntax is required in multiple fit mode:*

Syntax:

```
Specification <group number> <matrix name>
a { b {c}}
```

Following the `Specification` command, the user supplies a list of numbers that varies in length according to the dimensions and type of the matrix (see Section 3.3). If a zero is supplied, it indicates that the element is to be fixed. Non-zero elements refer to free parameters, and the same number refers to the same parameter. For example, the command

```
Specification A
1 2 3 0 0 0
0 0 0 3 2 1
```

would be equivalent to the statements

```
Pattern A
1 1 1 0 0 0
0 0 0 1 1 1
Equate A 1 1 A 2 6
Equate A 1 2 A 2 5
Equate A 1 3 A 2 4
```

The second method becomes tedious and error-prone in large models. Users familiar with LISREL will recognize specification matrices in LISREL output, but not as input.

Note that the `Specification` and `Pattern` commands cannot be mixed in the same Mx job. This is for safety, because the opportunities for user error are too large.

3.4.5 Boundary Command

Syntax:

```
Bound low high <parlist> / ALL
```

where parlist is a list of matrix elements or a list of parameter `Specification` *numbers.*

Boundary Constraints

By default, parameters to be estimated are constrained to lie between -10000 and +10000. These limits can be increased or decreased with the `Bound` command. Boundaries may be supplied more than once for any parameter, but only the last `Bound` statement referring to a particular element is used. For example the statements

```
Boundary -1 1 all
Boundary .3 5 a 1 4 6 a 1 5 6
```

would change the limits for all parameters to -1 and +1, except those (if any) in elements `A 1 4 6` and `A 1 5 6`. The `TO` syntax may be used to specify ranges within matrices, so that

```
Boundary 0 1 X 1 2 to X 1 6
```

would make parameters in all elements between `X 1 2` and `X 1 6` lie take values between zero and one.

If the `Specification` command has been used to specify parameters in matrices, then it may be easier to refer to parameters with these numbers in a `Bound` command. Thus

```
Specification A
0 2 4 6
2 0 6 7
Boundary 0 10  2 4
```

would be permitted as a method of bounding parameters 2 and 4.

Inequality Constraints

The boundary command may be used to specify linear and non-linear inequality constraints[1]. Constraints specified in a constraint group are numbered consecutively by rows in the lower triangle of the `NInput_vars` × `NInput_vars` matrix. For example, if `NInput_vars=3` was specified in the first constraint group, it would have constraints numbered as follows:

1		
2	3	
4	5	6

Any following constraint group would be numbered in a similar fashion, but starting at 7. Normally, the upper and lower boundary for a constraint are both set at zero. These boundaries may be reset *using negative parameter numbers to refer to the constraint we wish to bound.* For example, to make the boundaries on constraint 2 zero and ten thousand, we would use the command:

```
Bound 0 10000 -2
```

While it is a little clumsy, this notation does permit say constraining parameter 2 to be greater than parameter 3. This could be achieved with the following constraint group:

```
Title constrain parameter 2 > parameter 3
Data Constraint NInput_vars=1
Matrices
a full 1 1
b full 1 1
Constraint a - b /
Specify a
2
```

[1]In order to understand this Section, familiarity with the Section 2.5.1 below is at least helpful.

```
Specify b
3
Boundary 0 10000 -1
End
```

3.5 Label Matrices and Select Variables

3.5.1 Labelling Matrices

Syntax:

```
Labels Row/Column <matrixname> <label-list>
```

Labels may be given for the row or column (or both) of any matrix that has free elements. Matrices without free elements (`ZE, ID, IZ,` and `ZI`) are never displayed so labels provided for these matrices will not appear on the output. The label-list contains labels separated by blanks or carriage returns. Labels *must not begin with a number* and be *up to 8 characters long.* More characters can be read, but Mx only regards the first eight as significant, and will only print the first eight on the output.

Labels may be given for the observed data by issuing a `Label` command before the matrices command, as described in Section 2.4.1.

3.5.2 Identification Codes

Syntax:

```
ICodes <numlist>
```

where `<numlist>` *is a number list of length* `NInput_vars`.

The `ICodes` command may be used in conjunction with the `RM` fit function (Section 4.2.2) to specify a non-standard structure of the expected covariance matrix. By default, the identification codes for the covariance matrix are `1 2 3 4...` For example, if `NInput_vars=3` then by default the expected covariance matrix has a structure like this:

	1	2	3
1	V_1		
2	C_{12}	V_2	
3	C_{13}	C_{23}	V_3

From which structure it would be possible to read data in a `VLength` file that had forms:

```
1
1 .1
1
2 .2
1
3 .3
2
1 2 .1 .2
2
1 3 .1 .3
2
2 3 .2 .3
3
1 2 3 .1 .2 .3
```

and any of these could be reordered. For example, if the following `VLenghth` data were input:

```
2
3 1 .9 .4
```

Mx would generate a covariance matrix of the form

	3	1
3	V_3	
1	C_{13}	V_1

if means are being estimated, they will also be selected appropriately, in this case selecting μ_3, μ_1 from an initial vector $(\mu_1 \mu_2 \mu_3)$

The `ICodes` command allows the default order `1 2 3`... to change, making an infinite variety of input data vectors readable. The repetition of a number is most useful for pedigrees of variable structure, for example, if the model generates the covariance between two parents and two children, data that come from families with more than two children may be handled. In this case, the `ICodes` command would be:

```
ICodes 1 2 3 3
```

And thus the covariance matrix looks like this:

	F	M	C_1	C_2
F	V_F			
M	C_{FM}	V_M		
C_1	C_{FC}	C_{MC}	V_C	
C_2	C_{FC}	C_{MC}	C_{CC}	V_C

If the following `VLength` data were read:

```
3
3 3 3 .2 .4 .6
```

then Mx would create the following covariance matrix for this data structure:

	C_1	C_2	C_3
C_1	V_C		
C_2	C_{CC}	V_C	
C_3	C_{CC}	C_{CC}	V_C

The fact that two 3's have been given allows the generation of the expected covariance matrix between any number of individuals with code 3.

Chapter 4

Setting Options for Fit Functions and Output

4.1 Options and End Commands

Syntax:

```
Options <multiple> <fitfunc> <statout> <optimpar> <write>
End
```

where `<multiple>` *starts multiple fit mode,* `<fitfunc>` *specifies the fit function,* `<statout>` *requests statistical output,* `<optimpar>` *requests optimization parameters, and* `<write>` *specifies the filenames to write matrices to files*

The `Options` and `Output` lines of Mx allow the specification of a wide variety of keywords and parameters to control the type of fit function used, the amount of output requested, file names for result matrices, and many others. The `Options` command does not signify the end of a group, so several `Options` lines may be given within any group. `Options` commands should follow `Model` or `Covariance` statements, and should not be followed by `Bounds` commands. The `Output` line may be extended beyond a single line. If the last two characters of the line are ' `C`' (with no blanks after the `C`) then parameters on the next line will be accepted. This facility is not implemented for any other command line. `Output` signifies the end of a group. To end a group without specifying any options, the `End` command may be used. So, for example,

```
Options RSiduals NAG=30
End
```

would be equivalent to

```
Output RSiduals NAG=30
```

Table 4.1: Default fit functions according to the type of data that have been read. `CMatrix` - covariance matrix; `KMatrix/PMatrix` - correlation matrix; `ACov` - asymptotic covariance matrix; `AVar` - asymptotic variance matrix; `Rawdata` - Raw data; `VLength` - variable length data; `CTable` - contingency table; ML - maximum likelihood; AWLS - asymptotic weighted least squares; DWLS - diagonal weighted least squares; RM raw maximum likelihood; ML_N - maximum likelihood assuming bivariate normal liability.

Input data	Default fit function
`CMatrix` `KMatrix` or `PMatrix`	ML
`CMatrix` `KMatrix` or `PMatrix` with `ACov`	AWLS
`CMatrix` `KMatrix` or `PMatrix` with `AVar`	DWLS
`Rawdata` `VLength`	RM
`CTable`	ML_N

4.2 Fit functions: Defaults and Alternatives

The default fit function for a group is set according to the type of data that are read. For example, if covariance matrices alone are read, the default is maximum likelihood. Table 4.1 shows the default fit functions selected by Mx for a given data input. Note that the method may change between groups. If a function that does not asymptote to χ^2 (e.g. RU or RM) is used in any group, then no χ^2 probability is given at the end of optimization. In general, the default fit function is appropriate for the data supplied. Mx does not provide for altering the input data from one type to another (e.g. converting a covariance matrix to a correlation matrix). However, it is a simple task to write a separate Mx input file to make this conversion.

4.2.1 Standard Fit Functions

There are several good introductions to the properties of different fit functions (e.g Jöreskog & Sörbom, 1989; Bentler, 1989). Controversy exists about the relative merits of the different methods in the face of assumption violations (see Kaplan, 1990), and it seems wise for the user to treat this information in the same way as a white wine from the Loire (drink youngest available). Currently, maximum likelihood is showing robustness in the face of violations of the assumptions of multivariate normality. Asymptotic weighted least squares generally performs better in the presence of kurtosis, but can be at least as badly affected by skewness as ML. See however, simulation work by Rigdon and Ferguson (1991) for problems with tetrachoric correlations.

In the following sections, the calculation of the fit functions is described, where

$\mathbf{S}$ is the observed covariance matrix, $\mathbf{\Sigma}$ is the expected covariance matrix, $\mathrm{tr}(\mathbf{A})$ indicates the trace of and $|\mathbf{A}|$ indicates the determinant of matrix $\mathbf{A}$. $\mathbf{S}$ and $\mathbf{\Sigma}$ are of order p and df is one less than the sample size used to calculate $\mathbf{S}$.

LS

The unweighted least squares fit function is calculated by the formula:

$$\mathrm{LS} = \mathrm{df}\left(\frac{\mathrm{tr}(\mathbf{S}-\mathbf{\Sigma})^2}{2}\right)$$

ML

When model fitting to covariance matrices, the maximum likelihood fit function is

$$\mathrm{ML} = \mathrm{df}\left\{\ln|\mathbf{\Sigma}| - \ln|\mathbf{S}| + \mathrm{tr}\mathbf{S}\mathbf{\Sigma}^{-1}) - p\right\}$$

and this is modified when both a mean vector x and a model for the means μ (see Section 3.2.3) are supplied, this function is augmented to become

$$\mathrm{ML}_{\mathrm{C+M}} = \mathrm{df}\left\{\ln|\mathbf{\Sigma}| - \ln|\mathbf{S}| + \mathrm{tr}\mathbf{S}\mathbf{\Sigma}^{-1}) - p + (x-\mu)'\Sigma^{-1}(x-\mu) + 1\right\}$$

In order for the `ML` fit function to be calculated, $\mathbf{\Sigma}$ must be positive definite. If, during optimization, the determinant of $\mathbf{S}$ is less than 10^{-30} then Mx uses a penalty function to try to steer optimization back towards a positive definite solution. The penalty function is

$$100\left\{p^2 + 10^{33}\left(\frac{10^{-30}-|\mathbf{S}|}{|\mathbf{\Sigma}|}\right)\right\}$$

If starting values supplied begin optimization in this region, it is difficult for the optimizer to escape this high plateau, so optimization may fail. To avoid this, starting values may be revised or the LSML fit function may be used to obtain sensible starting values for ML estimation.

GLS

Generalized least squares is based on the principles of Aitken (1934-35); see also Browne (1974)

$$\mathrm{GLS} = \frac{\mathrm{tr}\left(\mathbf{I}-\mathbf{S}^{-1}\mathbf{\Sigma}\right)^2}{2}$$

AWLS

Asymptotic weighted least squares follows from work by Browne (1982, 1984) and others. Effectively, the variance covariance matrix of the observed summary statistics is used as a weight matrix $\mathbf{W}$. Formally the fit function is

$$\mathrm{AWLS} = \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{n}\sum_{l=1}^{k}(S_{ij}-\Sigma_{ij})\,\mathbf{W}_{ij,kl}^{-1}\,(S_{kl}-\Sigma_{kl})$$

In LISREL, and by default in Mx, if a correlation matrix (`KMatrix` or `PMatrix`) is supplied, the above formula is modified to

$$\mathrm{AWLS}_{\mathrm{KM}} = \sum_{i=2}^{n}\sum_{j=1}^{i-1}\sum_{k=2}^{n}\sum_{l=1}^{k-1}(S_{ij}-\Sigma_{ij})\,\mathbf{W}_{ij,kl}^{-1}\,(S_{kl}-\Sigma_{kl})+\frac{df}{2}\,(1-\Sigma_{ii})^2$$

The term $\frac{df}{2}\,(1-\Sigma_{ii})^2$ exists to constrain the elements of the diagonal to equal one. If the model fails to meet this constraint, the fit function is inflated. Mx prints the amount of the fit function that is due to this component of model misspecification. The second term is not calculated (and does not contribute to the fit function) if the keyword `DI` appears on the `Options` line.

An alternative approach to maintaining a diagonal of ones would be to standardize the expected covariance matrix before calculating the AWLS fit function. Mx does this if the keyword `Start` appears on the `Options` line. Some care with starting values is needed here, since the solution for a model standardized in this way is necessarily *not* unique. A third approach would be to use a nonlinear constraint group (see Section 2.5.2) that specifies that the diagonal elements should equal unity.

DWLS

Diagonally weighted least squares is a simplified form of AWLS for use with large problems where the AWLS matrix becomes unmanageably large. It is a compromise with less statistical validity than AWLS and should be used with caution. `Select` does not work with DWLS to discourage its use... The fit function for DWLS is simply:

$$\mathrm{DWLS} = \sum_{i=1}^{n}\sum_{j=1}^{i}(S_{ij}-\Sigma_{ij})\,\mathbf{W}_{ij,ij}^{-1}\,(S_{ij}-\Sigma_{ij})$$

where $\mathbf{W}$ is a diagonal matrix of variances of the observed covariances. $\sum_{i=1}^{n}\sum_{j=1}^{i}$ is replaced by $\sum_{i=2}^{n}\sum_{j=1}^{i-1}$ if a correlation matrix is supplied as data.

LSML

This fit function starts with unweighted least squares, and takes parameter estimates from the solution as starting values for maximum likelihood estimation. This method is useful to avoid having to specify starting values that generate a positive definite covariance matrix. Its disadvantage is that it consumes more computer time than would supplying appropriate start values and using ML alone.

4.2.2 Analyzing Raw Data

When we have a sample of complete multinormal data, the summary statistics of means and covariance matrices are sufficient statistics to obtain maximum likelihood estimates of parameters (see the keyword `ML` above). It is common practise to remove from analysis any subject that has missing data. However, there are occasions when missing data result in the omission of a significant amount of data from analysis. If the number of types of missing data is small, for example, if there are really two sub-populations, one that has data on 6 tests, and one that lacks data on the fourth test, then covariance matrices could be computed separately for the two populations and models fitted separately to the different groups. Mx is flexible, allowing different groups to have different numbers of input variables.

Unfortunately, this multi-group approach breaks down if the number of sub-populations is large and the sample size for each group is too small to estimate a positive definite observed covariance matrix (this happens if the number of variables exceeds the number of subjects). For this reason, a number of methods of handling incomplete data are provided in Mx. `RM` (raw maximum likelihood) is really an extension of the multigroup method described above, but calculates twice the negative log-likelihood of the data for each observation. The procedure follows the theory described by Lange *et al.,* (1976). If there are k observed variables in the set, the normal probability density function of an observed vector x_i is

$$|2\pi\mathbf{\Sigma}|^{-n/2}\exp\left\{-\frac{1}{2}(\mathbf{x}_i-\mu_i)'\mathbf{\Sigma}^{-1}(\mathbf{x}_i-\mu_i)\right\}$$

where $\mathbf{\Sigma}$ is the population covariance matrix and μ_i is the (column) vector of population means of the variables, and $|\mathbf{\Sigma}|$ and $\mathbf{\Sigma}^{-1}$ denote the determinant and inverse of the matrix $\mathbf{\Sigma}$, respectively. The fit function is thus:

$$\mathrm{RM} = -\left\{k\log(2\pi)+\log|\mathbf{\Sigma}|+(\mathbf{x}_i-\mu_i)'\mathbf{\Sigma}^{-1}(\mathbf{x}_i-\mu_i)\right\},$$

If there are incomplete data, a separate group could be constructed for each different type of data vector. This could be rather tedious for anything beyond a very few types of vector, so Mx provides a second, more general approach. The approach is to create a variable length record file (Section 2.3.3). This allows the use of the above fit function, but with a variable length observed vector x. The appropriate mean vector μ and covariance matrix $\mathbf{\Sigma}$ is automatically created by

Mx for each observation. To save on computer time, the creation of μ and $\mathbf{\Sigma}$ (and importantly $\mathbf{\Sigma^{-1}}$) is done only if a vector is different in structure from the previous vector. Therefore, considerable CPU-time saving can be obtained if sorted data are supplied to Mx.

4.2.3 Contingency Table Analysis

Mx will allow only 2-way contingency tables to be analyzed, so the covariance structure is necessarily limited to two variables. Figure 4.1 shows a contour plot of the frequency distribution of two variables, X and Y. For an r by c contingency

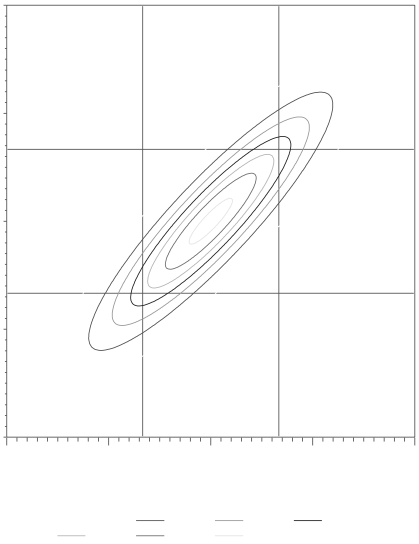


Figure 4.1: Contour plot showing a bivariate normal distribution with correlation r=.9 and two thresholds in the X and Y dimensions.

table, there are assumed to be $r-1$ row thresholds and $c-1$ column thresholds that separate the observed categories of individuals.

Twice the log-likelihood of the observed frequency data is calculated as:

$$\ln \mathrm{L_{CT}} = \sum_{i=1}^{r}\sum_{j=1}^{c} 2n_{ij} \ln \left\{ \frac{n_{ij}}{n_{..}p_{ij}} \right\}$$

where n_{ij} is the observed frequency in cell ij, p_{ij} is the expected proportion in cell ij, and $n_{..}$ is the total number of observations in the contingency table. The expected proportion in each cell is calculated by numerical integration of the bivariate normal distribution, performed by subroutine BIVNOR (Schervish, 1984). For example, the expected proportion with individual 1 lying in the category between threshold a and threshold b and with individual 2 lying in the category between threshold c and threshold d would be given by:

$$L_{1,2} = \int_a^b \int_c^d \phi(v_1, v_2)\, dv_2 dv_1 \tag{4.1}$$

where ϕ denotes the multinormal probability density function, and v_i is the liability of individual i.

Since n_{ij} is not estimated, the number of degrees of freedom associated with an $r \times c$ table is $rc - 1$. If z cells have not been ascertained, the number of degrees of freedom is reduced by z. In order to compute an approximate χ^2 statistic, twice the likelihood of the data under the model is subtracted from the likelihood of the observed data themselves, calculated as

$$\ln \mathrm{L}_{\mathrm{CTO}} = \sum_{i=1}^{r} \sum_{j=1}^{c} 2n_{ij} \ln \left\{ \frac{n_{ij}}{n_{..}} \right\}$$

See Section 3.2.4 for details on specifying thresholds for models fitted to contingency table data.

Non-random Ascertainment

Mx will automatically calculate an ascertainment correction while calculating the likelihood of the incompletely ascertained data. For example, if we ascertain a sample of 60 probands from hospital records and examine their spouses, of whom 10 are observed to have the same disorder, then a 2×2 contingency table would be supplied as follows:

```
CTable 2 2
-1 -1
50 10
```

The `-1` in the cells of the first row indicate that subjects were not ascertained in these areas. The likelihood of the observed data must be corrected for the incomplete ascertainment of subjects for study. Effectively, as we omit certain classes of person from observation, so the likelihood of observing the remaining individuals increases. Mathematically this is expressed by dividing the likelihood by the proportion of the population remaining after ascertainment. We obtain this by subtracting the proportions in all omitted classes from the total population proportion (i.e. 1.0). In our example, assuming that individual 1 has to be above threshold, the proportion omitted is

$$\tilde{A} = \int_{-\infty}^{t} \int_{-\infty}^{t} \phi(v_1, v_2)\, dv_2 dv_1 + \int_{-\infty}^{t} \int_{t}^{\infty} \phi(v_1, v_2)\, dv_2 dv_1 \tag{4.2}$$

where t is the ascertainment threshold, v_1 and v_2 are the liability values of individuals 1 and 2, and ϕ is the multinormal probability density function. The likelihood corrected for ascertainment would simply be the likelihood as obtained before, but divided by $1 - \tilde{A}$

4.2.4 User-defined Fit Functions

If the `US` keyword appears on the `Options` line, the fit function for the group is to be user specified. In order for this to be the case, the matrix expression given as the model (`Constraint` or `Covariance` command) *must evaluate to a scalar*. There are no other rules. Any of the automatically defined fit functions `LS, ML, AWLS` etc. could be specified as user-defined functions, but it is generally less efficient to do so. User-defined functions are recommended only when the provided functions are not suitable. An example is given in Section 5.5

4.3 Statistical Output and Optimization Options

In this Section we discuss some of the statistics that Mx will compute automatically. While the range of these statistics is limited at present, the user should note that it is quite straightforward to compute his or her own functions of the parameters or goodness of fit statistics using calculation groups (see Section 2.5.1 for syntax and Section 5.2.4 for an example script that computes standardized estimates).

4.3.1 Appearance

Syntax:

```
Option NDecimals=n or Width=m
```

where `n` *is the number of decimal places, and* `m` *is the number of columns*

By default, Mx will print most numbers with three decimal places, or use exponential format if there are very small or very large numbers in a matrix. You may override this default with the `NDecimals` keyword, where `NDecimals=n` will print `n` decimal places of precision.

Mx prints up to 80-columns of output, which is suitable for viewing on an 80-column display or legal/letter/A4 paper (in portrait orientation) with a 10cpi font. This default may be changed with the option `Width=m` where `m` is the number of columns desired. At the present time, the `NDecimals` and `Width` parameters cannot be used together (sorry).

4.3.2 Suppressing Output

Syntax:

`Option NO_output`

Before describing ways in which Mx output can be increased, we note the valuable keyword `NO_output` which prevents printing of all output for a group. This option should be used by the ecologically-minded as often as possible. Even the environmentally unconscious may find it useful to reduce their disk-space usage, but some caution should be taken not to use it too frequently since valuable information that could reveal misspecification of the model might be missed.

4.3.3 Residuals

Syntax:

`Option RSiduals`

The `RSiduals` keyword requests that the observed matrix, the expected matrix, and the residuals (observed - expected) be printed. In calculation and constraint groups, only the expected matrix is printed, since neither has any data. Note that `RSiduals` is the only way to have the observed matrix printed, and may be useful if the `Select` command has been used. When means have been supplied, the observed and expected mean vectors will be printed. Expected means also appear when using maximum likelihood with raw data. With contingency tables, Mx prints the observed and expected frequencies and their difference.

Under asymptotic weighted least squares Mx prints two types of residual matrix. First, it prints the unweighted difference between the observed and expected correlations or covariances. Second, it prints a *weighted residual matrix,* which is calculated from the formula:

$$\mathrm{WRes}_{ij} = \sum_{k=2}^{n} \sum_{l=1}^{k-1} (S_{ij} - \Sigma_{ij})\, \mathbf{W}^{-1}_{ij,kl}\, (S_{kl} - \Sigma_{kl}) + \frac{df}{2}\,(1 - \Sigma_{ii})^2$$

(see Section 4.2.1 on page 4.2.1). The sum of these elements gives the fit function for the group (neglecting any penalty function for the diagonal of a correlation matrix). Note that not all elements will be positive, but that their sum is necessarily non-negative. Quite often, inspection of the weighted residuals will give a clearer idea of the cause of model failure than consideration of the unweighted residuals alone.

4.3.4 Adjusting Degrees of Freedom

Syntax:

`Option DFreedom=n`

where `n` *is the adjusted number of degrees of freedom*

If a correlation matrix is read instead of a covariance matrix, the number of statistics provided is usually less than when variances are also given. The amount of the reduction in information depends on the structure of the data. For example, if MZ and DZ twins have been measured on one variable, there are four statistics that are necessarily equal (the variances of twin 1 and twin 2 in the MZ and DZ groups). Only one of these statistics confers any information (it scales the size of the MZ and DZ covariances), so three degrees of freedom are lost, and `DF=-3` should be placed on the `Options` line. For multivariate twin data, `DF=-3k` should be used, where 3k is three times the number of variables on which each twin is measured. We can extend this idea to m groups of pedigrees of size n, each measured on k variables, in which case `DF=` $-k(mn-1)$ should be used.

4.3.5 Power Calculations

Syntax:

```
Option Power=alpha,df
```

where `alpha` *is the probability level of the test, and* `df` *are the degrees of freedom associated with it*

Power calculations are useful in a wide variety of contexts, especially experimental design and getting grants. For theoretical work, once one has established that it is possible in principle to detect an effect, a natural question is 'what are the chances of finding it with a sample of x many subjects?' See Neale & Cardon (1992) for a blow-by-blow account of how power calculations are performed using LISREL. They are much easier to perform using Mx; however, we would urge that users do not use the methods blindly — a little knowledge is a dangerous thing — but instead buy a copy of Neale & Cardon! If the function values computed by Mx may be considered as a χ^2, then the `Power` command will compute the power of the study to reject the hypothesis at significance level α (`alpha`) for the given number of degrees of freedom (`df`). In addition, the program very kindly works out the total sample size that would be required, given the current proportion of subjects in each group, to reject the hypothesis at various power levels from .25 to .99. See Section 5.2.2 for an example application of this method in the context of the classical twin study.

4.3.6 Confidence Intervals

Syntax:

```
Option CInterval=n
```

where `n` *is the desired percentage of the confidence interval; e.g.* `CI=90` *will give 90% confidence intervals*

In order to have some idea of magnitude of the difference in fit between two competing models, it is useful to have confidence limits on their χ^2 statistics. These are computed using the inverse non-central χ^2 distribution, in the following way. Mx searches for the χ^2 (with degrees of freedom equal to those of the model) whose non-central χ^2 equals the observed χ^2 goodness of fit of the model. This procedure is done twice, to find the lower and upper limits. An example will make this clearer. If we requested the 90% confidence limits of a χ^2 of 10 with 5df, then the lower limit would be calculated as the χ^2 whose 95% limit was 10; the upper limit would be computed as the χ^2 whose 5% limit was 10. Note that these two statistics do *not* always bound the χ^2 statistic itself, i.e. an observed χ^2 may lie outside its own confidence intervals. See (Browne and Cudeck, 1990).

4.3.7 Standard Errors

Syntax:

`Option SErrors`

In order to retain flexibility in model specification, there are certain costs to be incurred. One of these costs is that explicit differentiation of the fit function with respect to the parameters is not feasible. In Mx standard errors are computed using numerical methods (NAG routine E04XAF). Normally this routine fails to get central difference and forward difference approximations to the Hessian to agree, because the gradient is very close to zero. The technical output of routine E04XAF is automatically printed in the file NAGDUMP.OUT if NAG=n where n is greater than zero, is used on the `Options` line. Mx warns the user that the errors are approximate. In the output file they are printed after each matrix that contains free parameters. Definitely $\beta-$test at present. Note that it is possible to obtain more strictly correct standard errors when the second derivatives are known. For example, using variants of the LISREL model, the second derivatives of component matrices are known (see e.g., Jöreskog, 1973) and could be calculated after optimization by adding a few calculation groups at the end.

4.3.8 Randomizing Starting Values

Syntax:

`Option THard=n`

where `n` *is a positive or a negative integer*

If the parameter `THard` is set using `TH=n` where `n` is a positive integer, Mx will generate random starting values for all parameters and attempt to fit the model again. This attempt is likely to fail if no bounds are specified, because the default boundaries are -10000 and +10000, and the random values will be random from within these bounds. Most optimizers fail if starting values are too far away from the final solution; Mx has shown greater tolerance than LISREL in this respect.

`THard` can be very useful when exploring the identification of structural equation models. If data are generated with particular fixed values for the variable parameters (Θ_1), then optimization from a different set of starting values (Θ_2) should give a solution of the original values (Θ_1). This can be tested a number of times using `THard`. If sensible bounds are not given for the parameters, this test will likely fail because Θ_1 will not be recovered. The key to underidentification is finding a solution that fits perfectly, but with a parameter vector other than Θ_1. If this is the case, the hypothesis that the model is identified has been *falsified*. Finding a number of solutions at Θ_1 does not prove identification of the model, it merely increases the support for the hypothesis that it is identified. Of course, this method does not show that the model has been specified correctly; a completely misspecified model may be identified.

The `THard` parameter may be used with a negative integer to request repeat fits from the solution, resetting the covariance matrix to an identity matrix (see Section 4.3.10).

4.3.9 Jiggling Parameter Starting Values

Syntax:

```
Option Jiggle
```

Similar to the randomizing option described above, parameter start values can be jiggled. Jiggling replaces each parameter start value x_i with $x_i + .1(x_i + .5)$. This option can be useful to nudge Mx away from a saddle point which can be troublesome when using numerically estimated derivatives (Mx does this by default). An example of a common saddlepoint is when parameters are started at or very near to zero, and the estimates x and $-x$ have the same effect on the function value. Such situations are very common in structural equation models which feature quadratic forms in their expectations; the ACE genetic model is one such example.

When used in conjunction with a negative `THard` parameter, jiggling will occur each time the refit is attempted. This may cause estimates to drift from their initial values, especially if the parameter concerned has no effect on the fit function.

4.3.10 Changing Optimization Parameters

Mx uses NAG's routine E04UCF to perform numerical optimization in the presence of general linear, non-linear and boundary constraints. Certain versions of the code use the essentially equivalent routine NPSOL, obtained from Walter Murray and Philip Gill at Stanford University in place of E04UCF. Mx does its best to choose suitable values for the parameters that control optimization, taking into account the number of parameters to be estimated, the numerical precision of the function value, and so on. However, the enormous variety of types of optimization tasks that can in principle be requested with Mx means that the automatic selection of these parameters is not always ideal. In addition, difficulties in optimization may require examination of the optional output that E04UCF can generate. Mx allows the user to print these data and to alter the parameters as needed. There are also facilities for automatically performing some of the solutions of optimization difficulties suggested in the NAG manual.

In general, parameters have been set for NAG on the cautious side, so that many of the warning messages about NAG's IFAIL parameter being non-zero are spurious. This seems better than being misled by the program ending without a noise, but giving the wrong answer. Of least concern are IFAIL's of 1 (code GREEN). Next easiest to deal with are IFAIL=4 (code BLUE - it ran out of breath), for which the `IT=n` may be used. IFAIL=3 (code RED - bad news) occurs in connection with constraints, normally they have been misspecified. Printing the residuals in the constraint groups often helps. IFAIL=6 (also code RED - take note) is the most tricky. Sometimes it occurs because of ill-conditioning of the Hessian, which can be verified by examination of the NAG output file (see Section 4.3.10). A solution here may be to use the `TH=-n` which requests optimization from the current 'solution' with the Hessian reset to the identity matrix, `n` times. On other occasions, it may appear because of insufficient numerical precision, yet still be at the right place. If a parameter in your model is not identified, IFAIL=6 is quite likely.

Appropriate choice of starting values is always recommended. Many users start parameters at zero because this is the default value of free matrix elements. In practice, Mx attempts to avoid so doing in two ways. First any parameter in the range -.01 to .01 is started at .01, and second, the JI option may be used to further nudge the parameter value away from a possible saddle point at zero (see Section 4.3.9).

Setting NAG Parameters

The defaults given in this section are the defaults as set by Mx, not the defaults suggested by NAG.

NAG=n

If this statement appears on the `Options` line, the technical output from NAG is printed in a file called `NAGDUMP.OUT`. The value `n` controls the Major Print Level, the higher the number the more verbose the output file. Minimum output is written with `NAG=1` and maximum is written with `NAG=30`. The only printing that Mx does to this file is an initial value of the function. Mx rescales all functions to 1.0 to assist general optimization, so the function value printed by NAG is a proportion of this initial value.

DE

Sets all optimization parameters at the default values given in the NAG manual.

FEAS=n

Will control the Nonlinear Feasibility Tolerance, i.e. `FEAS=r` defines "the maximum acceptable absolute violations in linear and nonlinear constraints at a 'feasible' point; i.e., a linear constraint is considered satisfied if its violation does not exceed `r`" (NAG, 1990). It has no effect if there are no nonlinear constraints. Default = .00001

IT=n

Controls the number of major iterations. Should be increased if IFAIL=4 error messages occur. Default = 100.

LI=r

Linesearch tolerance. Default = .9 if no non-linear constraints are present, otherwise, Default = .3

OP=r

Sets the optimality tolerance, a parameter controlling the accuracy of the solution. Default = $1^{-(n+6)}$ where n is approximately $\log(F_1)$ where F_1 is the function value at the starting values.

STE=r

Infinite step size. Default = 10000.

FU=r

Specifies function precision. In general this should be set at a lower value than the required precision of the solution. Default = $1^{-(n+8)}$ where n is approximately $\log(F_1)$ where F_1 is the function value at the starting values.

Obtaining Extensive Debug Output: DB=1

If the parameter `DB=1` is set on the output line, together with `NAG=n` where `n` is greater than zero, additional information will be written to the NAGDUMP.OUT file. For each evaluation of the function to obtain the gradient of the parameter vector, the fit function value for each group, the total fit function, and the values of all the parameters are printed. On each evaluation of the function to obtain the constraint functions, the values of all the parameters and the constraint functions are printed. Note that the order of the parameters in the vector corresponds to the order used by NAG during optimization and not the order of parameter specifications given by the user, or printed by Mx. Note also that using this option for problems with more than a few parameters can result in enormous NAGDUMP.OUT files.

4.4 Fitting Submodels: Saving Matrices and Files

One of the powerful features of Mx is its ability to start again where it left off. An example of this has already been described in Sections 4.3.8 and 4.3.10 above, where repeated attempts to optimize are made either from the current solution or from randomized starting values. Here we describe how repeated fits may be made from the solution, allowing for changes in the model. This can be done manually, writing out matrices to files, or automatically within the same run, using the `Multiple` command, or from a saved binary file. For large problems, use of binary files can save a lot of time.

4.4.1 Fitting Submodels Using Multiple Fit Option

Syntax:

```
Option Multiple
```

If the keyword `Multiple` is included on the `Options` line, the next Mx input file is assumed to have a special form. It will consist of a single group, ending with an `Options` line. The only commands that may be used under the `Multiple` option are `Specification`, `Pattern`, `Matrix`, `Value`, `Start`, `Equate`, `Fix`, `Free` and `Options`. Data may not be changed, matrices may not be added, and the matrix formulae specified in the `Covariance` or `Means` statement cannot be altered. At some time in the future, these restrictions may be lifted.

There is no group structure to the input stream following an `Multiple` command; all modifications to the model must refer to matrices with a number identifying the particular group in which the matrix is to be found. This changes the usual form of the `Specification, Matrix,` and `Pattern` commands to include a group specification, which must be placed directly after the keyword, *before* the letter indicating the matrix. Thus

```
Specification A
```

becomes

```
Specification 2 A
```

if `A` was specified in group 2. As an example of the use of this command, consider the simple factor model presented in Section 2.1.4. We could test the significance of the covariance of the two variables by fixing parameter #2 to zero. Obviously this could be achieved by modifying the entire input file and running it, but the following will fit both models, and fits the second starting at the solution provided by the first.

```
Simple MX example file
Data NGroups=1 NObservations=150 NInput_vars=2
CMatrix
1.2
.8 1.3
Matrices
A Full 2 1
D Diag 2 2
Model A*A' + D /
Specification A
1 0
Specification D
0 3
Options Multiple RSiduals
End
Specification 1 A
1 2
End
```

Considerable computer time can be saved using `Multiple`, since the solution of a model often has parameter estimates which are close to those estimated under nested models of the same type. In general, we recommend fitting models starting with the simplest, and working up to more complex models. When working from complex to simple, the `Drop` command (see next section) can be useful. If you have more than a single set of nested models to test, saving the general model in an Mx binary save file (see Section 4.4.3) can save considerable time and effort.

4.4.2 Dropping Parameters from Model

Syntax:

```
Drop {@value} <parnumlist>
```

where parnumlist is a list of parameter numbers as used in the `Specification` *command, and @value is an optional value to fix at*

Quite often, equality constraints between parameters lead to model specifications with the same parameter in many different matrices or several groups or both. Fixing all occurrences of the parameter with the `Fix` parameter can be time-consuming and error prone, so the `Drop` command may be used instead. By default, all parameters whose specification number matches a number in the list following the `Drop` command will be fixed to zero. For example:

```
Drop 5 8 7
Drop 11 to 20
```

would fix to zero all occurences of parameters 5, 8, 7 and 11 to 20. Note that matrix addresses cannot be used in this command. It is possible to supply an optional value to the drop command, so that for example

```
Drop @.5 2 3
```

would fix all occurences of parameters 2 and 3 to 0.5

4.4.3 Reading and Writing Binary Files

Syntax:

```
Save <filename>
Get <filename>
```

where filename is the name of the file to be saved or retrieved - the extension .mxs is recommended

When the multiple fit option is implemented (see Section 4.4.1) the entire input job specifications, data, and estimates may be stored in binary format for rapid retrieval and estimation in subsequent fitting of submodels. Note that `Save` *must* follow the entire job specification. Thus for example the following would save the results of fitting the model, together with the complete model specification, in the file `acesave.mxs`:

```
.
.    ! Mx commands for first job
.
Options Multiple
End
Save acesave.mxs ! First command in multiple fit number 1
Fix all
End
```

The `Fix all` command simply stops the optimizer from trying to improve on the current solution by fixing all the parameters. To use this binary save file in another command file, we could use the following:

```
Title - using a binary file
Get acesave.mxs
Fix A 1 1 1
End
```

By retrieving a binary file, MX is automatically in the Multiple fit mode, so modifications can be made to the model and a further series of hypotheses tested. If `Get` is used in a separate job, a title line is required before this statement. Normally, parameter estimates after model fitting are stored, but if it is desired to save the user's starting values, it is possible to set the number of iterations to zero, use `Multiple`, and `Save` the starting values. When the binary file has been read with the `Get` command, it is necessary to remember to reset the `IT` parameter on the `Options` line.

4.4.4 Writing Matrices to Files

Syntax:

```
MXa= <filename>
```

where `a` *is the single-letter name of the matrix to be written, or one of %E %M %P %V*

MX will write matrices to files with this command. The first line has header information, including the Group number, the matrix name, type and dimensions. The matrix elements are then written in FORTRAN format (6D13.6). This file format is fully compatible with LISREL, so matrices output by MX can be read in as starting values for LISREL and *vice versa* (see Section 3.3.1). If the matrix name is `%E`, the expected covariance matrix will be written to the file. If the matrix name is `%M`, the expected mean matrix will be written to a file. For groups with raw maximum likelihood fit functions, %P will write a series of columns of information about the likelihood of individual pedigrees, as follows:

1. the likelihood function for that vector of observations
2. the square root of the Mahalanobis distance, $(x-\mu)'\Sigma^{-1}(x-\mu)$
3. the number of the observation in the active (i.e. post selection) dataset. Note that with selection this may not correspond to the position of the vector in the data file
4. the length of the vector (i.e. the family size if it is a pedigree with one variable per family member).

Finally, it is also possible to save a VL file,with the `%V` keyword. While this is not normally useful (since it would recreate the original data file), following the select command it can be advantageous. Subsequent reading of the selected data can be faster than reading the whole dataset and performing selection again. A second advantage is where there are problems with the data so that observations are seen to be particularly unlikely. MX will usually print out the observation, and the expected mean and standardized difference (see Section 5.4.2)

4.4.5 Creating RAMpath Graphics Files

Syntax:

```
Draw= <filename>
```

Structural equation models may be specified very simply in terms of three matrices. The first matrix, **S**, is symmetric and specifies all the two-headed arrows between all the variables (both latent and observed) in the diagram. The second matrix, **A**, is asymmetric and specifies all the single-headed arrows between all the variables in the model. Causal paths from variable i to variable j are specified in element $\mathbf{A}_{ji}$. For example, a path from variable 1 to variable 4 would be represented in element (4,1) of this matrix. The third matrix, **F**, is used to filter the observed variables from the latent variables for the purpose of model fitting. The development and application of this approach is succinctly described in the RAMpath manual (McArdle & Boker, 1990).

Iff[1] you specify a model with these three matrices, F, A and S, then RAMpath graphics files may be written and saved to a file with the `Draw` command. This file, largely consisting of a RAMpath input_model command, may be used as input for RAMpath to draw a diagram of your model to view or to produce publication-quality output on a postscript printer. An example of this approach is given in Section 5.2.3.

[1]Iff with two f's means 'if and only if'

Chapter 5

Example script files

There are very many different ways of setting up any particular model in Mx. As with programming, there is a compromise between compactness and comprehensibility that is set by the individual user. The most compact files are often the least comprehensible; the longest ones are the most prone to typographical errors, and can be very boring. Judicious use of comments can make for a short but comprehensible input file.

5.1 Using Calculation Groups

The examples in this section do not fit models; matrix formulae are simply evaluated and the results are printed.

5.1.1 General Matrix Algebra

Suppose we wish to find the inverse of the symmetric matrix:

$$\begin{array}{ccc} 1. & & \\ .23 & 2. & \\ .34 & .45 & 3. \end{array}$$

The following input file could be created:

```
Inverting Symmetric 3 x 3 example file
Data Calc NGroups=1
Matrices
A Symm 3 3
Compute A~ /
Matrix A
1.
```

```
.23 2
.34 .45 3.
Options RSiduals
End
```

The `RSiduals` command would print the expected matrix for this group, i.e. the result of evaluating the formula in the `Compute` command, i.e. the inverse of `A`. Note that the expected matrix of a calculation group is always considered to be *full* even if it is always symmetric for a particular example. Thus adding `Mx%e=inverted.mat` to the `Options` line above would write nine elements of the inverse of `A` matrix to the file `inverted.mat`

5.1.2 Assortative Mating 'D' Matrix

Multivariate phenotypic assortative mating (Van Eerdewegh, 1982; Vogler, 1985; Carey, 1986 Phillips et al, 1988; Fulker, 1988) leads to a predicted covariance matrix between husbands' and wives' phenotypes of the form:

$$\mathbf{\Sigma} = \left[\begin{array}{r|r} \mathbf{R_H} & \mathbf{R_H D' R_W} \\ \hline \mathbf{R_W D R_H} & \mathbf{R_W} \end{array}\right]$$

Thus the matrix $\mathbf{D}$ can be obtained from $\mathbf{R_W}^{-1}\mathbf{M}\mathbf{R_H}^{-1}$, where $\mathbf{M} = \mathbf{R_W D R_H}$, the off-diagonal block of correlations between phenotypes of husbands and phenotypes of wives. The following Mx input file will calculate matrix $\mathbf{D}$.

```
Calculation of full D matrix, 3 phenotypes husband & wife
Data Calculation NGroups=1
Matrices
A Symm 3 3
B Symm 3 3
C Full 3 3
Compute A~*C*B~ /
! Covariance of wives' variables
Matrix A
1
.4 .9
.3 .5 1.1
! Covariance of husbands' variables
Matrix B
1.2
.42 1.
.3 .47 .9
!Covariance between husband and wife variables.
Matrix C
.4 .1 .2
```

```
.05 .3 .12
.22 .11 .5
Options RSiduals
End
```

The relevant part of the output file looks like this:

```
  GROUP NUMBER:           1

 CALCULATION OF FULL D MATRIX, 3 PHENOTYPES HUSBAND & WIFE

  MATRIX A
 This is a SYMMETRIC matrix of order    3 by    3
    1.0000
    0.4000    0.9000
    0.3000    0.5000    1.1000

  MATRIX B
 This is a SYMMETRIC matrix of order    3 by    3
    1.2000
    0.4200    1.0000
    0.3000    0.4700    0.9000

  MATRIX C
 This is a FULL matrix of order    3 by    3
    0.4000    0.1000    0.2000
    0.0500    0.3000    0.1200
    0.2200    0.1100    0.5000

  EXPECTED MATRIX of this CALCULATION group
    0.4302   -0.2854    0.1497
   -0.3471    0.7770   -0.5237
    0.1361   -0.4908    0.7829
```

5.1.3 Pearson-Aitken Selection Formula

This example is a little more complex. In 1934, Aitken generalized Pearson's formulae for predicting the new mean and variance of variable when selection is performed on a correlated variable. Aitken's delightful paper shows how selection on a vector of variables, X_S, leads to changes in the covariance of correlated variables X_N that are not themselves selected. If the original (pre-selection) covariance matrix of X_S is $\mathbf{A}$, and the original covariance matrix of X_N is $\mathbf{C}$, and the covariance between

X_N and X_S is $\mathbf{B}$, then the original matrix may be written

$$\left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{B}' & \mathbf{C} \end{array}\right)$$

if selection transforms $\mathbf{A}$ to $\mathbf{D}$, the whole new matrix is given by:

$$\left(\begin{array}{r|r} \mathbf{D} & \mathbf{D}\mathbf{A}^{-1}\mathbf{B} \\ \hline \mathbf{B}'\mathbf{A}^{-1}\mathbf{D} & \mathbf{C}-\mathbf{B}'(\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{D}\mathbf{A}^{-1})\mathbf{B} \end{array}\right)$$

Likewise, if the original means are $(\mathbf{x_s} : \mathbf{x_n})'$ and selection modifies $\mathbf{x_s}$ to $\tilde{\mathbf{x_s}}$, the vector of means after selection is given by:

$$\mathbf{x_n} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{x_s} - \tilde{\mathbf{x_s}})$$

```
Pearson Aitken Selection formulae
! Idea is to give original means and covariances,
! and get new ones
!
! We have original matrix  A  B
!                          B’ C
!
! Which we following transformation of A becomes
!                          D | DA~B
!                       B’A~D| C-B’(A~-A~DA~)B
!
Data Calc NGroups=2
Matrices
A Symm 2 2
B Full 2 2
C Symm 2 2
D Symm 2 2
Compute
     D | D*A~*B_
B’*A~*D| C-B’*(A~-A~*D*A~)*B /
!
! Note that the underscore above is the UNDER operator,
!  not A with a horizontal bar over it!
!
Matrix A
1.
.7 1.
Matrix B
.6 .42
.42 .6
```

```
Matrix C
1.
.352 1.
Matrix D
1.
.4 1.
Options RSiduals
End
!
! The means are done in group 2.  We have original means of
! the non-selected variables, n, which are transformed to
! n + A~Bs where s is the deviation of the means of the
! selected variables from their original values.
!
Second group to calculate new means
Data Calc
Matrices
A Symm 2 2 =A1
B Symm 2 2 =B1
S Full 2 1
N Full 2 1
Compute N + A~*B*S /
Matrix N
3
4
Matrix S
2 1
Options RSiduals
End
The new covariance matrix and mean vector are printed as the
expected matrices of the calculation groups.
```

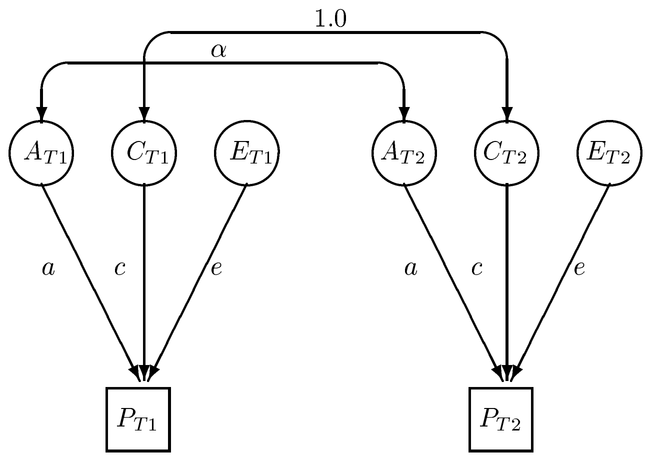
5.2 Model Fitting with Genetically Informative Data

The examples in this section demonstrate elementary use of Mx to fit models to data in the form of variance-covariance matrices.

5.2.1 ACE Genetic Model for Twin Data

If data are collected from identical or monozygotic (MZ) twins and from fraternal or dizygotic (DZ) twins, it is possible to estimate parameters of a model that includes

Figure 5.1: The parameter α is fixed at 1 for MZ twins and at .5 for DZ twins. All latent variables have a variance of 1.0



the effects of additive genes (A), shared or family environment (C) and random or unique environment (E). This model is shown in Figure 5.1 as a path diagram. A LISREL input file for the model was provided by Heath *et al* (1989), which could be adapted very easily using `Equate` statements in Mx. However, the MX script illustrates use of the `Specification` command to modify it slightly. Heath *et al* allow for phenotypic interaction, but this is left for a later example (Section 5.2.5)

```
A C E model fitted to the Heath (1989) data
!     on alcohol consumption.
! Group 1.... Monozygotic twin pairs
    ! Anything after a '!' is a comment
Data NGroups=2 NInput_variables=2 NObservations=171
Labels Alc_t1 Alc_t2
CMatrix
1.28
0.766 1.194
Matrices
A Full 2 6
B Symmetric 6 6
```

```
Covariance_model A*B*A' /
Specification A
1 2 3 0 0 0
0 0 0 1 2 3
Labels Col A
At1 Ct1 Et1 At2 Ct2 Et2
Labels Row A
Pt1 Pt2
Matrix B
1
0 1
0 0 1
1 0 0 1
0 1 0 0 1
0 0 0 0 0 1
Labels Row B
At1 Ct1 Et1 At2 Ct2 Et2
Labels Col B
At1 Ct1 Et1 At2 Ct2 Et2
Options RSiduals
End

Group 2.... Dizygotic twin pairs
Data NInput_variables=2 NObservations=101
CMatrix
1.077
0.463 0.962
Labels Alc_t1 Alc_t2
Matrices
A Full 2 6 =A1
B Symm 6 6
Covariance_model A*B*A' /
Matrix B
1
0 1
0 0 1
.5 0 0 1
0 1 0 0 1
0 0 0 0 0 1
Labels Row B
At1 Ct1 Et1 At2 Ct2 Et2
Labels Col B
At1 Ct1 Et1 At2 Ct2 Et2
```

```
Start .6 All
Options RSiduals
End
```

5.2.2 Power Calculation for the Classical Twin Study

Our next example uses the same model as in the preceding Section, but in this case we calculate the power of the classical twin study to detect the effects of common environmental variation. The particular case we wish to examine is where the true population variance comprises 20% additive genetic, 30% shared environment, and 50% random environment variance, but we fit a model *without* shared environment variation to simulated MZ and DZ covariance matrices. This example is discussed in some detail in Neale & Cardon (1992); here we reproduce their results obtained with rather more effort using LISREL and some statistical tables.

There are two stages to the power calculation. First, fixed values of the parameters a, c and e are given, and a two-group Mx script simply calculates the expected covariances under the model, and saves them to two files, mzsim.cov and dzsim.cov. The next problem (preferably in the same input file, though this isn't essential) fits a model of additive genetic and random environmental variance only. The complete input file looks like this:

```
A C E model for power calculation   Part 1: simulate the data
Data Calc NGroups=2       !Note that it is just a calculation group
Matrices                  !Define matrices for the model
A Full 1 3
B Symmetric 6 6
I Identity 2 2
Compute (I@A)*B*(I@A)' /  !Formula for expected MZ cov
Labels Col A
At1 Ct1 Et1
Matrix A
.5477 .4472 .7071     ! To simulate:
!                       30% additive genetic   (.5477^2=.3)
!                       20% common environment (.4472^2=.2)
!                       50% random environment (.7071^2=.5)
!
!  Note that I@A will generate a matrix of the form
!  .6  0 .8  0  0  0
!   0  0  0 .6  0 .8
!  because I is an identity matrix and @ is the kronecker product.
!  as required for the model.  We reduce the chance of errors
!  when modifying the script by adopting this formulation here.
!
Matrix B
```

```
1
0 1
0 0 1
1 0 0 1
0 1 0 0 1
0 0 0 0 0 1
! Some labels to make the output a bit prettier
Labels Row B
At1 Ct1 Et1 At2 Ct2 Et2
Labels Col B
At1 Ct1 Et1 At2 Ct2 Et2
! This is the crucial bit where we save the file
Options mx%e=mzsim.cov
Options NO_output
End

Group 2.... Dizygotic twin pairs
Data Calc
Matrices
A Full 1 3 =A1
B Symm 6 6
I Iden 2 2
Compute (I@A)*B*(I@A)' /
Matrix B
1
0 1
0 0 1
.5 0 0 1
0 1 0 0 1
0 0 0 0 0 1
! And next we save the DZ file
Options mx%e=dzsim.cov
Options NO_output  ! No output for this group as nothing new, really
End

! The simulation is complete.  Now the next, stacked, problem
! will fit the wrong model to the simulated data

A C E model for power calculation
Data NGroups=2 NInput_variables=2 NObservations=1000
Labels Twin_1 Twin_2
CMatrix Full File=mzsim.cov
Matrices
```

```
A Full 1 3
B Symmetric 6 6
I Identity 2 2
Covariance_model (I@A)*B*(I@A)' /
Specification A
1 0 3            ! False model with no genetic effects
Labels Col A
At Ct Et
Labels Row A
Pt1 Pt2
Matrix B
1
0 1
0 0 1
1 0 0 1
0 1 0 0 1
0 0 0 0 0 1
Labels Row B
At1 Ct1 Et1 At2 Ct2 Et2
Labels Col B
At1 Ct1 Et1 At2 Ct2 Et2
Options RSiduals
End

Group 2.... Dizygotic twin pairs
Data NInput_variables=2 NObservations=1000
CMatrix Full File=dzsim.cov
Labels Twin_1 Twin_2
Matrices
A Full 1 3 =A1
B Symm 6 6
I Iden 2 2
Covariance_model (I@A)*B*(I@A)' /
Matrix B
1
0 1
0 0 1
.5 0 0 1
0 1 0 0 1
0 0 0 0 0 1
Labels row B
At1 Ct1 Et1 At2 Ct2 Et2
Labels col B
```

```
At1 Ct1 Et1 At2 Ct2 Et2
Start .7 ALL     ! Note different starting values from simulation
Options RSiduals Power= .05,1  ! For .05 level of sig. and 1 d.f.
End
```

The relevant part of the output is at the end, where we see that for the specified sample sizes of 1000 pairs each of MZ and DZ twins, the χ^2 goodness-of-fit is 11.35, as found by Neale & Cardon:

```
 Chi-squared fit of model =    11.349
  Degrees of freedom =           4
  Probability =      0.023
 Akaike's Information Criterion =     3.349

 Power of this test, at the 0.0500 significance level
 with   1. df is 0.920572
 Based on your combined observed sample size of        2000.
 The following sample sizes would be required to reject
 the hypothesis:

  Power         Total N
  .25             292.
  .50             677.
  .75            1223.
  .80            1383.
  .90            1852.
  .95            2290.
  .99            3238.
```

Because we requested that this statistic be treated as a χ^2 for 1 degree of freedom for the purposes of calculating power at the .05 level of significance (`POwer=.05, 1`), the output gives the power of the test, followed by the sample sizes that would be required to obtain certain commonly used levels of power.

5.2.3 RAM Specification of Model for Twin Data: Graphic Output

Here is a two group example, a phenotypic interaction PACE model (see Section 5.2.5 for more details), specified using the three matrix approach of McArdle & Boker (1990). Details of this method, and the syntax of the `Draw` command can be found in Section 4.4.5.

The title for the diagram is taken from the title of the group in the Mx input file, and the labels for the variable are taken from labels of the columns of the **S** matrix. The draw commands in this file produce two files, mx.ram and dz.ram. I don't like the way RAMpath draws interaction between the phenotypes, but there is a certain irrefutable logic in having causal arrows always going out the bottom of a variable and in the top. You can easily edit the RAMpath file to get rid of the @ signs if you like. Note that specifying models à la RAMpath is didactically very clear but computationally very inefficient, since the inverse of the maximally dimensioned **A** matrix is required. Using calc groups can reduce this burden if CPU time is at a premium but diagrams are desired.

```
Phenotypic interaction PACE model, Heath 1989 data
! Demonstration of RAM specification and output
Data NGroups=2 NInput_vars=2 NObservations=171
Cmatrix Symm
1.280
0.766 1.194
Matrices
s Symm 8 8
i Iden 8 8
a Full 8 8
f ZIden 2 8
Covariances F*(I-A)~*S*(I-A)~'*F'/
Matrix s
1
0 1
0 0 1
1 0 0 1
0 1 0 0 1
0 0 0 0 0 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Label row a
a1 c1 e1 a2 c2 e2 p1 p2
Label col a
a1 c1 e1 a2 c2 e2 p1 p2
Label row s
a1 c1 e1 a2 c2 e2 p1 p2
Label col s
a1 c1 e1 a2 c2 e2 p1 p2
Specify a
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

```
1 2 3 0 0 0 0 4  ! This is where the parameters are
0 0 0 1 2 3 4 0
Start .6 a 7 1 a 7 3
Options RSiduals Draw=mz.ram
End

Group 2.... DZ twins
Data NInput_vars=2 NObservations=101
Cmatrix Symm
1.077
0.463 0.962
Matrices
s SY 8 8
i Iden 8 8
a Full 8 8 = a1
f ZIden 2 8
Covariances F*(I-A)~*S*(I-A)~'*F'/
Matrix s
1
0 1
0 0 1
.5 0 0 1
0 1 0 0 1
0 0 0 0 0 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Label row s
a1 c1 e1 a2 c2 e2 p1 p2
Label col s
a1 c1 e1 a2 c2 e2 p1 p2
Bound -.99 .99 3
Bound 0 5 1 2
Options RSiduals Draw=dz.ram
End
```

5.2.4 Cholesky Decomposition for Multivariate Twin Data

Any positive definite matrix may be transformed to a product of a lower triangular matrix and its transpose, i.e.

$$\mathbf{\Sigma} = \mathbf{LL}'$$

This *Cholesky* decomposition is unique for a given positive definite matrix $\mathbf{\Sigma}$. It has a simple graphic representation as a path diagram (Figure 5.2) where the first latent factor loads on all variables, the second on all variables except the first, the

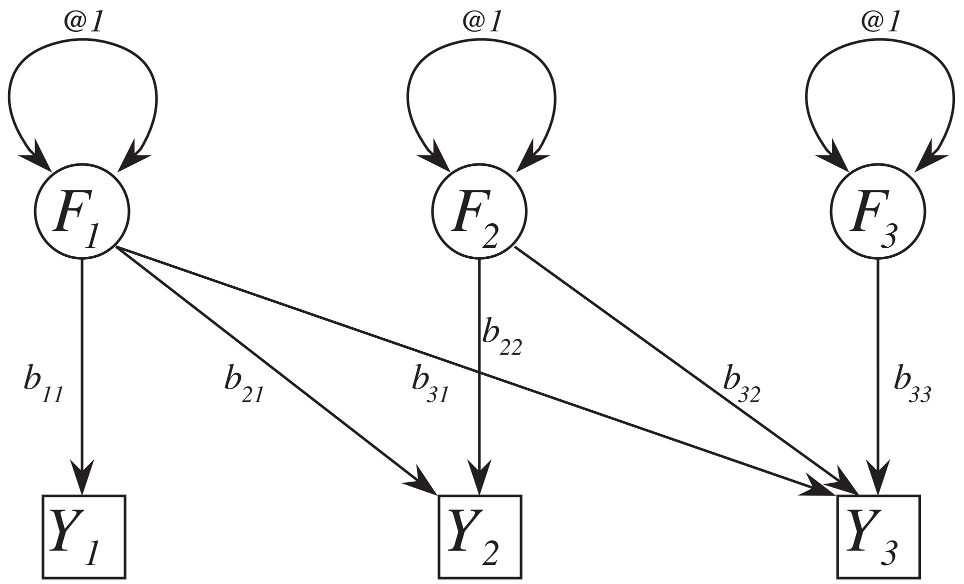


Figure 5.2: Cholesky or triangular factor model for three variables Y1 Y2 and Y3

third on all variables except the first two, and so on.

In multivariate genetic analysis, a Cholesky decomposition of separate genetic and environmental covariance matrices is possible. Thus the additive genetic, shared environment and random environment factors in the simple ACE model (Figure 5.1) has a multivariate counterpart where the phenotypes P_1 and P_2 are replaced by vectors of observed phenotypes, and the latent variables A C and E are replaced by matrices of factors. The path coefficients a c and e are replaced by triangular matrices of factor loadings according to the Cholesky decomposition. There are a number of ways of approaching such a model in Mx. The standard way, to set it up as one would a LISREL model, would prove very awkward to alter to allow for increased or decreased numbers of variables. Here we use an input file that separates the genetic, shared and random environmental factors into different calculation groups that generate genetic, shared and random environment covariance matrices. It is then a simple matter to form the expected covariance matrices for twin data as partitioned matrices containing linear combinations of these matrices. It is then simplicity itself to fix a parameter to zero, as only one character has to be changed from a 1 to a zero.

The example includes data from individuals without cotwins, as well as MZ and DZ twin pairs. Estimates from a model such as this depend on the size of the observed variances, and can be difficult to interpret. Estimates of the proportion of variance and covariance due to each source can be obtained using the eldiv operator

(%) (groups 7-9).

```
!
! Trivariate Cholesky 'Independent Pathways' model.
!
! Additive Genetic factors are group 1 matrix H
! Random Environ. factors are group 2 matrix E
! Common Environ. factors are group 3 matrix C
!
! Data are Extraversion, Neuroticism and CESD Depression
!
This group JUST calculates, no fitting is done.
! Genetic factors
Data Calculation NGroups=9
Matrices
H Lower 3 3 Free
Compute H*H'  /
End

Title 2nd Group also for calculation
! Specific Envt.
Data Calculation
Matrices
E Lower 3 3 Free
Compute E*E'  /
End

Title 3rd Group also for calculation
! Shared Envt.
Data Calculation
Matrices
C Lower 3 3 Free
Compute C*C'  /
End

!
! Now get to the actual data, and use the results of
!  groups 1, 2 and 3 (%E1 %E2 & %E3)
!
Unmatched twins
Data NInput_vars=3 NObservations=449
Cmatrix Symm
 .102216
-.021648 .0827805
```

```
-.005792 .0177622 .0138577
Matrices
G Symm 3 3 = %E1
E Symm 3 3 = %E2
C Symm 3 3 = %E3
Covariances (G + E + C) /
End

MZ twins with cotwins  Extraversion Neuroticism and Depression
Data NInput_vars=6 NObservations=456
Cmatrix
.0887349
-.018651  0.08036
-.001031 .0185154 .0139989
.0366848 -.008858 -.002873 0.097803
.0013777 .0291787 .0110424 -.014907 .0818947
-0.00122 .0103675 .0050655 -.005258 .0172896 0.013687
Matrices
G Symm 3 3 = %E1
E Symm 3 3 = %E2
C Symm 3 3 = %E3
Covariances
             (G + E + C| G + C _
                  G + C| G + E + C )
/
Options RSiduals
End

!
!
DZ twins with cotwins  Extraversion Neuroticism and Depression
Data NInput_vars=6 NObservations=357
CMatrix
0.100808
-0.02249 .0785764
-.003347 .0173292 .0150047
-0.00471 .0025537 -.002045 0.102181
.0032861 .0081708 .0078504 -.023836 .0808707
-.001514 .0053548 .0036505 -.005488 .0164667 .0143083
Matrices
G Symm 3 3 = %E1
E Symm 3 3 = %E2
C Symm 3 3 = %E3
```

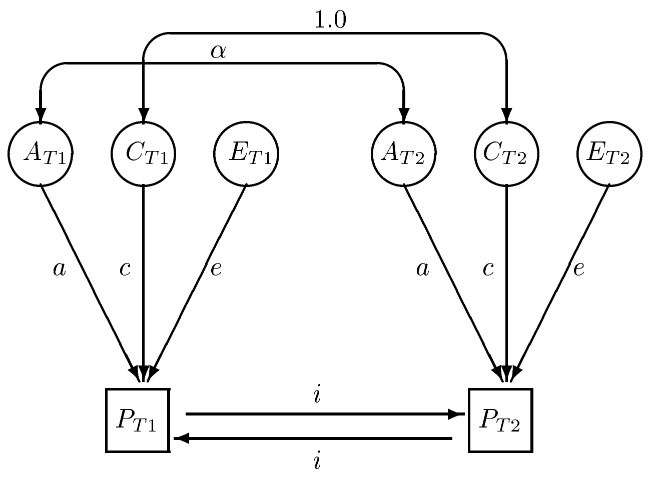
```
h Diag 1 1
Covariances
!
! Covariance matrix bit.  Note H as diag matrix of .5's
! that models DZ genet cov.  By using the Kron operator,
! every element of G is multiplied by .5, in accordance with
! genetic theory that DZ twins share only half their genes.
!
      (G + E + C| H @ G + C _
       H @ G + C| G + E + C )
/
Matrix Half
.5
Start .5 All
Boundary -1 1 All
Options RSiduals IT=200
End

Calculation of standardized solution - Additive Genetic
Data Calc
Matrices
A Symm 3 3 =%E1
P Symm 3 3 =%E4
Compute A%P /
Options RSiduals
End

Calculation of standardized solution - Shared Environment
Data Calc
Matrices
C Symm 3 3 =%E3
P Symm 3 3 =%E4
Compute C%P /
Options RSiduals
End

Calculation of standardized solution - Random Environment
Data Calc
Matrices
E Symm 3 3 =%E2
P Symm 3 3 =%E4
Compute E%P /
Options RSiduals
```

Figure 5.3: Path for additive genetic (A), shared environment (C) and specific environment (E) effects on phenotypes (P) of pairs of twins ($T1$ and $T2$). Path i models direct phenotypic effects of a twin on his or her cotwin. The parameter α is fixed at 1 for MZ twins and at .5 for DZ twins.




```
End
```

5.2.5 PACE Model: Reciprocal Interaction between Twins

Figure 5.2.5 shows a path diagram similar to the ACE model for twin data shown in Figure 5.1; There is now a path (i) from the phenotype of a twin to that of his or her cotwin. This is reciprocal interaction between dependent variables. It can easily be shown (see Jöreskog and Sörbom, 1989; also Section 1.2) that a matrix representation of this process is to use the formulation $(\mathbf{I}-\mathbf{B})^{-1}$, where $\mathbf{B}$ is a full matrix whose element b_{jk} represents the causal effects of variable k on variable j. In this case, the parameter i has been bounded to lie between -1 and 1, though this is not necessary.

```
Phenotypic interaction model, fit to Heath 1989 data.
! Female MZ Twins
Data NGroups=2 NInput_vars=2 NObservations=171
CMatrix
```

```
1.28
0.766 1.194
Labels Alc_t1 Alc_t2
Matrices
G Full 2 6
P Symm 6 6
B Full 2 2
I Identity 2 2
Covariance_model (I-B)~*(G*P*G')*(I-B)~'/
Labels row G
B4int_t1 B4int_t2
Labels col G
At1 Ct1 Et1 At2 Ct2 Et2
Specification G
1 2 3 0 0 0
0 0 0 1 2 3
Labels row P
At1 Ct1 Et1 At2 Ct2 Et2
Labels col P
At1 Ct1 Et1 At2 Ct2 Et2
Matrix P
1
0 1
0 0 1
1 0 0 1
0 1 0 0 1
0 0 0 0 0 1
Labels col B
B4int_t1 B4int_t2
Labels row B
Alc_t1 Alc_t2
Specification B
0 4
4 0
Options RSiduals
End

Phenotypic interaction model, Heath alcohol data
! Female DZ twins
Data NInput_vars=2 NObservations=101
Cmatrix Symm
1.077
0.463 0.962
```

```
Matrices
G Full 2 6 =G1
P Symm 6 6
B Full 1 2 =B1
I Identity 2 2
Covariance_model (I-B)~*(G*P*G')*(I-B)~'/
Labels row P
At1 Ct1 Et1 At2 Ct2 Et2
Labels col P
At1 Ct1 Et1 At2 Ct2 Et2
Matrix P
1
0 1
0 0 1
.5 0 0 1
0 1 0 0 1
0 0 0 0 0 1
Start .6 g 1 1 - g 2 6
Boundaries -.99 .99 4
Boundaries 0 5 1 2 3
Options RSiduals
End
```

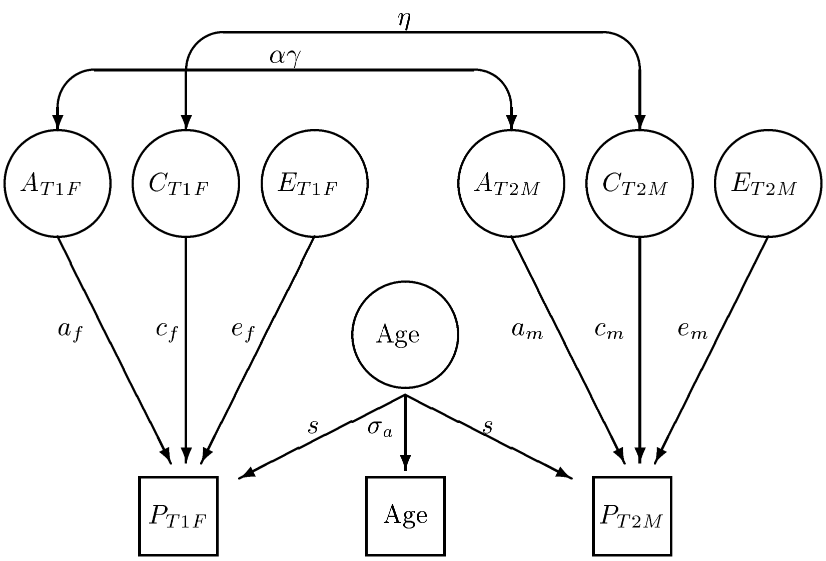
5.2.6 Scalar and Non-scalar Sex Limitation; Age Regression

The model and data in this section were taken from Neale & Martin (1989) who fitted a model of scalar sex-limitation to data from 5 groups of twins who reported subjective impressions of drunkenness following a challenge dose of alcohol. The model here involves additive genetic, shared and random environment components (A C and E; see example in Section 5.2, and Figure 5.1) but allows these to differ in their effects on the phenotypes of males and females. In addition, the regression of the phenotypes on Age is modelled, so that there are parameters for the variance of age (σ_a^2) and for the regressions (s). A path diagram of the model is shown in Figure 5.4.

In the fifth group, two commented out lines (beginning with !) indicate how the model could be modified to test the fit of a model that allowed for different genes in the two sexes. Two features are of particular note in this example. First is the use of the group-specific matrix constraint (e.g. `M Full3 7 =M2`) which makes for a brief input file. Second is the use of boundary constraints to prevent the estimation of parameters of opposite sign in the two sexes. Considerable effort is required to implement this simple constraint in LISREL (see Neale & Martin, 1989).

```
Age correction Sex limitation model MZ FEMALES
```

Figure 5.4: Path model for sex-limited additive genetic (A), shared environment (C) and specific environment (E) effects on phenotypes (P) of pairs of twins ($T1$ and $T2$) The parameter α is fixed at 1 for MZ twins and at .5 for DZ twins. Either γ or η may be estimated with data from twins, but not both.



```
Data NGroups=5 NInput_vars=7 NObservations=43
CMatrix File=DRUNK.COV
Matrices
F Full 3 7
P Symm 7 7
Covariances F*P*F' /
Specification F
1 2 3 0 0 0 4
0 0 0 3 2 1 4
0 0 0 0 0 0 9
Matrix P
1
0 1
0 0 1
0 0 0 1
0 1 0 0 1
1 0 0 0 0 1
0 0 0 0 0 0 1
Options RSiduals
End

Age correction Sex limitation model MZ MALES
Data NInput_vars=7 NObservations=42
CMatrix File=DRUNK.COV
Matrices
P Symm 7 7 =P1
M Full 3 7
Covariances M*P*M' /
Specification M
5 6 7 0 0 0 4
0 0 0 7 6 5 4
0 0 0 0 0 0 9
Options RSiduals
End

Age correction Sex limitation model DZ FEMALES
Data NInput_vars=7 NObservations=44
CMatrix File=DRUNK.COV
Matrices
F Full 3 7 =F1
P Symm 7 7
Covariances F*P*F' /
Matrix P
```

```
1
0 1
0 0 1
0 0 0 1
0 1 0 0 1
.5 0 0 0 0 1
0 0 0 0 0 0 1
Options RSiduals
End

Age correction Sex limitation model DZ MALES
Data NInput_vars=7 NObservations=38
CMatrix File=DRUNK.COV
Matrices
M Full 3 7 =M2
P Symm 7 7 =P3
Covariances M*P*M' /
Options RSiduals
End

Age correction Sex limitation model DZ OPP SEX
Data NInput_vars=7 NObservations=39
CMatrix File=DRUNK.COV
Matrices
P Symm 7 7
O Full 3 7
Covariances O*P*O' /
Specification O
1 2 3 0 0 0 4
0 0 0 7 6 5 4
0 0 0 0 0 0 9
Matrix P
1
0 1
0 0 1
0 0 0 1
0 1 0 0 1
.5 0 0 0 0 1
0 0 0 0 0 0 1
!Free P 6 1
!Boundary 0 0.5 P 6 1
Boundary 0 5 1 2 3 5 6 7
Start .6 All
```

```
Options RSiduals
End
```

5.2.7 Multivariate Assortative Mating: Modelling D

In this section we return to the transformation of the data described in Section 5.1.2. The question is now, how do we fit a model with parameters in the **D** matrix, so that we can explore the significance of marital assortment both within and between variables. For example, is there selection of wealthy men by attractive women, or is it just that attractiveness and wealth are correlated, and partners choose each other on the grounds of wealth alone? Neale & McArdle (1990) published a transformation of the matrix equation in Section 5.1.2 which allowed the fitting of LISREL and COSAN models to marital data. While LISREL is widely available, the implementation of the model is not straightforward, involving phantom latent variables (Rindskopf, 1984). The model is very easy to implement in Mx, as is shown below. We have already shown how estimates of parameters in the **D** matrix may be obtained directly (Section 5.1.2); here we show how to test specific hypotheses about direct and indirect homogamy. Phillips et al. (1988) reports data on general intelligence, educational level, extraversion, anxiety, tough-mindedness and independence in both husbands and wives. The first diagonal element of **D** therefore represents direct homogamy for intelligence; by fixing this parameter to zero we test the statistical significance of the process.

```
Assortative mating: Phillips data, test that d 1 1 is zero
Data NGroups=1 NInput_vars=12 NObservations=334
CMatrix File=asmat.cov
Matrices
h Symm 6 6 fr
w Symm 6 6 fr
d Full 6 6 fr
M2       h | h*d'*w_
     w*d*h | w       /
Start 1. h 1 1 h 2 2 h 3 3 h 4 4 h 5 5 h 6 6
Start 1. w 1 1 w 2 2 w 3 3 w 4 4 w 5 5 w 6 6
Fix  d 1 1
Options RSiduals
End
```

5.3 Fitting Models with Non-linear Constraints

5.3.1 Principal Components

Any symmetric matrix **C** may be decomposed to a product $\mathbf{ABA}'$ where **B** is a real diagonal matrix and **A** is a real orthogonal matrix, i.e. $\mathbf{AA}' = \mathbf{I}$. The elements of **B** are *eigenvalues* of **C** and the columns of **A** are the *eigenvectors* of **C**. In general, if we fitted a model $\mathbf{ABA}'$ where **A** was full and **B** was diagonal, it would be underidentified, since there would be more parameters in the model than in the observed covariance matrix **C**. However, we can supply identifying constraints that **A** is *orthogonal*. In the Mx input file, these constraints are imposed in group 2, by setting $\mathbf{AA}' - \mathbf{I} = \mathbf{0}$. This is not an efficient method of obtaining eigenvalues and eigenvectors of a matrix, but it does highlight nonlinearly constrained optimization. For eigenvalue and eigenvector functions, see Section 3.2.2 and Table 3.5.

```
Principal components ABA' with constraints
!                     to keep A orthogonal
Data NGroups=2 NInput_vars=3 NObservations=100
Cmatrix Symm
   1.
  .6  .9
  .4  .2  .7
Matrices
A Full 3 3
B Diag 3 3
Covariances A*B*A'/
Specification A
1 2 3
4 5 6
7 8 9
Specification B
10 11 12
Start 1.0 A(1,1) A(2,2) A(3,3) B(1,1) to B(3,3)
Options LS
End

Here is the constraint A*A'=I
Data CON NInput_vars=3
Matrices
A Full 3 3 = A1
I Identity 3 3
Model A*A'-I/
Options LS
End
```

5.3.2 Fitting a PACE Model to Contingency Table Data on MZ and DZ Twins

In order to fit a model with additive genetic, common and specific environment components to categorical data collected from twins, we are faced with two possibilities. One, we could use PRELIS or similar software to estimate tetrachoric or polychoric correlation matrices and associate asymptotic weight matrices, or two, we could fit directly to the contingency tables. The advantage of the latter approach is when there is phenotypic interaction between the twins. Phenotypic interaction would lead to different variances between groups, or in the case of categorical data, to proportionate group differences in the thresholds. This example uses a simple PACE model (see the example shown in Section 5.2.5) fitted to 2×2 contingency tables obtained from MZ and DZ twin pairs. There is no information on the total variance in these data; but there is information on the relative magnitude of the variance in MZ and DZ groups. Therefore, it is necessary to constrain the total variance prior to interaction to unity. This is done in the third group.

```
Categorical data analysis.  PACE model
Data Ninput=2 NGroup=3
CTable 2 2
30  20
19  60
Matrices
A fu 2 6
B fu 2 2
I id 2 2
P Symm 6 6
T Full 2 1
Thresholds T /
Covariances (I-B)~* A*P*A' *(I-B)~ /
Specification A
1 2 3 0 0 0
0 0 0 1 2 3
Start .6 A 1 1 - A 2 6
Specification T
4 5
Specification B
0 6
6 0
Boundary -.99 .99 6
Matrix P
1
0 1
0 0 1
1 0 0 1
0 1 0 0 1
0 0 0 0 0 1
Options RSiduals
End
```

```
Test of categorical data analysis.  DZ twins in ACE model
Data Ninput=2
CTable 2 2
30  30
29  50
Matrices
A fu 2 6 =A1
B fu 2 2 =B1
I id 2 2
P Symm 6 6
T fu 2 1 =T1
Thresholds T /
Covariances (I-B)~* A*P*A' *(I-B)~ /
Matrix P
1
0 1
0 0 1
.5 0 0 1
0 1 0 0 1
0 0 0 0 0 1
Options RSiduals
End

Constraint group to ensure a*a + c*c + e*e = 1
Data Constraint Ninput=1
Matrices
S Full 1 3
I Identity 1 1
Constraint I - S*S'   /
Specification S
1 2 3
Options None Iterations=300
End
```

5.3.3 Twins and their Parents: Cultural and Genetic Transmission

The model described has been developed extensively in the univariate and multivariate case (Eaves et at, 1978; Fulker, 1982; Neale & Fulker, 1984; Vogler, 1985; Fulker, 1988; Phillips et al, 1988; Cardon, Fulker & Jöreskog, 1991). In order to preserve consistency with the ACE model presented for twin data alone, the same separation of environmental effects is made here, following the last of the referenced papers instead of the earlier treatments. A path diagram of the model is shown in Figure 5.5.

The development of a covariance structure model for this design is not simple. We describe two approaches: (i) direct use of RAM theory in which all variables in the model are represented in two matrices, and the Pearson-Aitken selection formula (see Section 5.1.3) is used to handle assortative mating, and (ii) an computationally efficient approach that progresses in a stepwise fashion from the top to the bottom of the diagram. Approach (ii) is recommended for general use except where fast hardware gives approach (i) comfortably quick turnaround.

RAM Theory Approach

As discussed in Section 4.4.5, structural equation models may be specified very simply in terms of three matrices. The first matrix, **S**, is symmetric and specifies all the two-headed arrows between all the variables (both latent and observed) in the diagram. The second matrix, **A**, is asymmetric and specifies all the single-headed arrows between all the variables in the model. Causal paths from variable i to variable j are specified in element $\mathbf{A}_{ji}$. The third matrix, **F**, is used to filter the observed variables from the latent variables for the purpose of model fitting. This example is a relatively inefficient approach to fitting this model, but it illustrates the flexibility of Mx to implement theory-driven models explicitly.

```
!
! Rose social fears data
!
!  Full 9-Phenotype model for all pedigree types
!
! P->C transmission & P--P assortment
!
G1 - covariance in the absence of assortative mating
Data Calculation NGroups=13
Matrices
A Full 19 19
I Iden 19 19
S Symm 19 19
Compute (I-A)~*S*(I-A)~' /

Labels Row A
!1  2    3    4    5  6  7    8  9 10   11 12 13
! 14 15   16   17  18  19
PH PW PMZ1 PMZ2 PDZ AH AH' CH EH AW AW' CW EW
AMZ CE EMZ1 EMZ2 ADZ EDZ
Labels Col A
PH PW PMZ1 PMZ2 PDZ AH AH' CH EH AW AW' CW EW
```

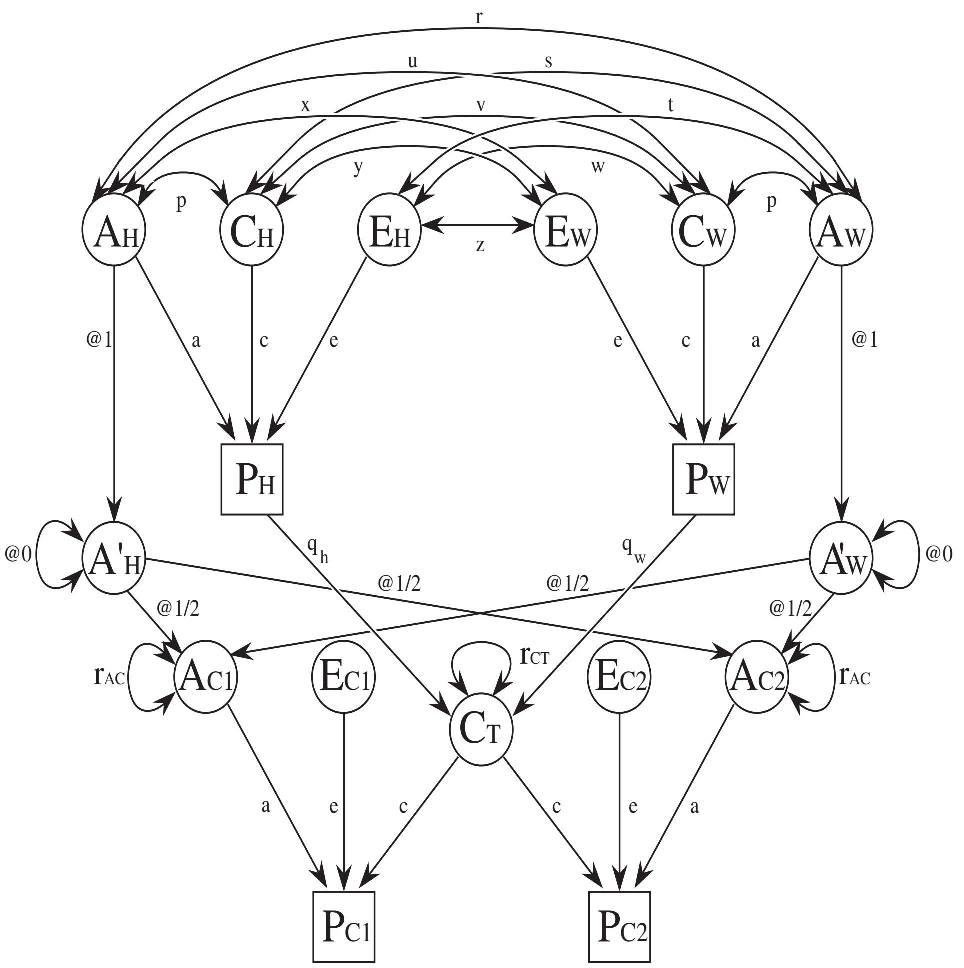


Figure 5.5: Model of mixed genetic and cultural transmission. Phenotypes of a husband and wife (P_H and P_W) directly affect the shared environment of their children (CT). Assortative mating based on phenotypes generates covariance between the latent variables of the parents (parameters r to z). The additive genetic and shared and specific environmental effects (parameters a, c and e) and the covariance of C and A (parameter p) are assumed to be equal across generations. Reduced genetic correlation between parents and offspring may be modelled if paths from A_H to A'_H and from A_W to A'_W are free parameters, with corresponding alteration of the residuals on A'_H and A'_W to maintain unit variance
.

```
AMZ CE EMZ1 EMZ2 ADZ EDZ

Specification A
0 0 0 0 0  1 0 2 3 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 1  0 2 3 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 1 2  3 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 1 2  0 3 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 2  0 0 1 3

0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0

0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
4 4 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0

0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0

Value 1.0  A 7 6  A 11 10
Value 0.5  A 14 7   A 14 11  A 18 7   A 18 11

Labels Row S
!1  2    3    4   5  6  7   8  9 10  11 12 13
! 14 15   16   17  18  19
PH PW PMZ1 PMZ2 PDZ AH AH' CH EH AW AW' CW EW
AMZ CE EMZ1 EMZ2 ADZ EDZ

Specify S
0
0 0
0 0 0
0 0 0 0
0 0 0 0 0

0 0 0 0 0  0
```

```
0 0 0 0 0  0 0
0 0 0 0 0  8 0 0
0 0 0 0 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0

0 0 0 0 0  0 0 0 0 0  0
0 0 0 0 0  0 0 0 0 8  0 0
0 0 0 0 0  0 0 0 0 0  0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 5
0 0 0 0 0  0 0 0 0 0  0 0 0 0 6

0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 5
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0

Labels Col S
!1  2    3    4   5  6  7   8  9 10  11 12 13
! 14 15   16   17  18  19
PH PW PMZ1 PMZ2 PDZ AH AH' CH EH AW AW' CW EW
AMZ CE EMZ1 EMZ2 ADZ EDZ
Start 1.0 S 6 6 S 8 8 S 9 9 S 10 10 S 12 12 S 13 13 S 16 16
Start 1.0 S 17 17 S 19 19
Options RSiduals
End

Group 2  Get covariance of parent phenotypes with other variables
Data Calculation
Matrices
X IZero  2 19
R Symme 19 19 = %E1
Y ZIden 19 17
Compute X*R*Y /
Options RSiduals
End

Group 3 - Get covar. of variables that are not parents' phenotypes
Data Calculation
Matrices
R Symm  19 19 = %E1
Z ZIden 17 19
Compute Z*R*Z' /
Options RSiduals
```

```
End

Group 4 - Obtaining Covariance Matrix of Parents
Data Calc
Matrices
J IZero  2 19
B Full  19 19 = %E1
Compute J*B*J' /
Options RSiduals
End

Group 5 - Handle the effects of assortative mating
Data Calculation
Matrices
A FU  2  2 =%E4
B FU  2 17 =%E2
C FU 17 17 =%E3
I ID  2  2
M SY  2  2
Compute       (A+M)|(A+M)*A~*B
_     ((A+M)*A~*B)'|C-B'*(A~*(I-(A+M)*A~))*B /
! Compute    (A+M)|(A+M)*B
! _     ((A+M)*B)'|C-B'*-M*B /     ! REDUCED FORM
Specify M
0
7 0
Start .1 M 2 1
Options RSiduals
End

Group 6 - Constraints on kids' G-E variance and covariance
Data Constraint NInput_vars=2
Matrices
E Full 19 19 =%E5
C Stan 2 2
F Full 2 19
Constraint F*E*F'-C /
Specify C
8
Matrix F
0 0 0 0 0  0 0 0 0 0  0 0 0 1 0  0 0 0 0
0 0 0 0 0  0 0 0 0 0  0 0 0 0 1  0 0 0 0
Labels Col F
```

```
Labels Row A
!1  2    3    4   5  6  7   8  9 10  11 12 13
! 14 15   16   17  18  19
PH PW PMZ1 PMZ2 PDZ AH AH' CH EH AW AW' CW EW
AMZ CE EMZ1 EMZ2 ADZ EDZ
Options RSiduals
End

Group 7 - VA phobia fear data: MZ twins & their parents
!Data NI=8 NO=340
!PMatrix File=mzphob.cor
!ACov File=mzphob.asy
!Labels
! T1SOC T1ACRO T2SOC T2ACRO
! P3SOC P3ACRO P4SOC P4ACRO
!Select P3SOC P4SOC T1SOC T2SOC /
Data NInput_vars=4 NObservations=144
CMatrix
.903
.113 1.1053
.2547 .0864 .9467
.1602 .2052 .4938 .8947
Matrices
C FU 19 19 = %E5
F IZ 4 19
Covariances F*C*F'/
Options RSiduals
End
! DZ data

Group 8 - Rose social criticism DZ  twins & their parents
!Data NI=8 NO=227
!PMatrix File=dzphob.cor
!ACov File=dzphob.asy
!Labels
! T1SOC T1ACRO T2SOC T2ACRO
! P3SOC P3ACRO P4SOC P4ACRO
!Select P3SOC P4SOC T1SOC T2SOC /
Data NInput_vars=4 NObservations=106
CMatrix
1.0811
.2111 .9063
.1443 .2398 1.2079
```

```
.1001 .1551 .3246 1.0318
Matrices
C FU 19 19 = %E5
F FU 4 19
Covariances F*C*F'/
Matrix F
1 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 1 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 1 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0
0 0 0 0 1  0 0 0 0 0  0 0 0 0 0  0 0 0 0
Options RSiduals IT=500
End

Group  9 - Standardization
Data Calc
Matrices
A Full 1 3
S Stand 3 3
Covariances A.A@(A*S*A')~ /
Specification A
1 2 3
Specification S
8
0 0
Options RSiduals
End

Group 10 Expected correlation matrix
Data Calc
Matrices
A Full 19 19 = %E5
!Compute \stnd(A) /
Compute A /
Options RSiduals NO_output
End

Group 11 Getting expected matrices for phenotypes
Data Calc
Matrices
J IZero 5 19
A Full 19 19 = %E10
Compute J*a*j' /
Options RSiduals
```

```
End

Group 12 Getting Covariances among Parents' latent variables
Data Calc
Matrices
F fu 6 19
A fu 19 19 = %E5
Compute F*A*F' /
Start 1 f 1 6  f 1 8  f 1 9  f 2 10 f 2 12 f 2 13
Options RSiduals
End

Group 13  - Parameter estimates
Data Calculation
Matrices
P Full 1 8
Compute P/
Labels COL P
A C E Q ResGs ResCt Mu  S
Specification P
1 2 3 4  5     6    7   8
Start .7 P 1 1 P 1 3
Start .5 P 1 5
Start 1 P 1 6
Start .1 P 1 4 P 1 8
Bound .001 1 P 1 1 P 1 2 P 1 3 P 1 5 P 1 6
Bound -1 1 P 1 4 P 1 7 P 1 8
Options TH=-1 Multiple
End

Save ramrose3.mxs
! No assortment
Drop 7
Options Multiple
End
! No genes
get ramrose3.mxs
Drop 1 8
Options Multiple
End
! No VCT
get ramrose3.mxs
Drop 4 8
```

```
Options Multiple
End
! No CE at all
Get ramrose3.mxs
Drop 2 4 8
Options Multiple
End
! No resemblance
Drop 1 7
End
```

Computationally Efficient Approach

It is possible to get much faster turnaround with computationally efficient approaches to structural modelling of twins and their parents. The constraints are calculated in group 1, to generate the (3×3) block of correlations between husband latent variables and wife latent variables.

$$\begin{array}{cccc} & AH & CH & EH \\ AW & r & s & t \\ CW & u & v & w \\ EW & x & y & z \end{array}$$

In the absence of sex differences in genetic or environmental effects, this matrix is symmetric.

The second group constructs the 6×6 (symmetric) correlation matrix between the parents' latent variables by concatenating the within-person parts (matrix `D` of group 1) with the off-diagonal across-parents part calculated in group 1 (matrix `%E1`).

$$\begin{array}{ccccccc} & AH & CH & EH & AW & CW & EW \\ AH & 1 & & & & & \\ CH & p & 1 & & & & \\ EH & 0 & 0 & 1 & & & \\ AW & r & s & t & 1 & & \\ CW & u & v & w & p & 1 & \\ EW & x & y & z & 0 & 0 & 1 \end{array}$$

This matrix is used in group 3 (matrix `P`) to calculate the covariance between the parents' phenotypes and the latent variables of the offspring. Group 4 introduces the cultural transmission parameters. Of note here is the matrix `S`, which contains residual parameters to equate the genetic (and environmental) variance in children to those of the parents. Also present is additional covariance between the environments of the children (parameter 9, element `S 5 5`), due to sources other than parental phenotypes. The fifth group specifies the nonlinear constraints, keeping the variances of AC1, AC2, EC1 and EC2 at unity, and equating the covariance of AC2 and EC2 to that in the parental generation. The sixth group keeps the variance of parents at unity. Standardization of the model in this way helps to keep optimization well conditioned. As shown, group 7 is useless and could be omitted. However, it is instructive to include it to show how the model could be altered to specify a model with reduced genetic correlation between parents and children. Variance parameters are supplied in the data groups (8 and 9) where the observed data are supplied and the fit function is calculated. Note the key difference between MZ and DZ twins is that in the former, genetic variance comes from the same genotype (elements `L 8 3 3` and `L 8 4 3`), whereas in the latter genetic variance is caused by two different (though correlated) genotypes via elements `L 9 3 3` and `L 9 4 4`). The tenth group simply prints out parameter estimates.

We recognize that this specification in matrices is not as simple as might be obtained if a specification in terms of expected correlations was implemented in a FORTRAN program. However, modifications to the Mx code are relatively simple to make. For example, to change the model of assortative mating the first group could be altered. In the example code the necessary components to allow for reduced parent-child genetic covariance are present but commented out. Additionally, facilities for dropping parameters to fit reduced models or for adding different data summaries make this example a convenient starting point for comprehensive analysis of data from all types of nuclear family and twin and parent data.

```
!
! Rose Fear data:  Social phobia on twins and parents
!
! Twins and parents: Genetic and cultural transmission model.
!                    P--P assortment
!                    Reduced P-C genetic correlation is
!                              commented out group 7
!
Group 1 - Assortative mating constraints
Data Calculation NGroups=10
Matrices
A Full 1 3
D Standardized 3 3
M Full 1 1
Compute (A*D)'*M*(A*D) /
Labels Column A
A C E
Labels Row A
P
Specification A
1 2 3
Matrix A
```

```
.7 .5 .5
Labels Row D
A C E
Labels Col D
A C E
Specification D
4
0 0
Specification M
5
End

Group 2 - construct the overall covariance matrix of parents'
!         latent variables
Data Calculate
Matrices
D Stan 3 3 = D1
E Full 3 3 = %E1
Compute
  (D|E
  _E|D) /
Labels row E
Ah Ch Eh
Labels col E
Aw Cw Ew
End

Group 3 - precalculate parents model
Data Calculate
Matrices
P Symmetric 6 6 = %E2
G Full 4 6
R Diagonal 4 4
Compute G*P*G' + R/
! For reduced genetic correlation between parent & child model
! parts commented out with ! are needed
Specification G
1 2 3 0 0 0
0 0 0 1 2 3
0 0 0 0 0 0 !99 0 0 0 0 0
0 0 0 0 0 0 !0 0 0 99 0 0
Labels row G
PH PW AH' AW'
```

```
Labels column G
AH CH EH AW CW EW
Value 1 G 3 1 G 4 4
! Specification r
! 0 0 100 100
End

Group 4 - precalculate middle of model
Data Calculate
Matrices
P Symmetric 4 4 = %E3
F Full 5 4
S Symmetric 5 5
Compute F*P*F'+S /
Labels row F
PH PW AC1 AC2 CT
Labels col F
PH PW AH' AW'
Specification F
 0 0 0 0
 0 0 0 0
 0 0 0 0
 0 0 0 0
 6 7 0 0
Value 1 F 1 1 F 2 2
Value .5 F 3 3  F 3 4  F 4 3  F 4 4
Labels row S
PH PW AC1 AC2 CT
Labels col S
PH PW AC1 AC2 CT
Specification S
0
0 0
0 0 8
0 0 0 8
0 0 0 0 9
Start .5 S 3 3   S 4 4
Start 1 s 5 5
End

G5 - Constraints on genetic environmental variance and
!    covariance in children
Data Constraint NInput_vars=2
```

```
Matrices
D Standardized 3 3 = D1
!Correlation matrix of AH CH EH
H Full 5 5 = %E4
!Correlation matrix of PH PW AC1 AC2 CE
Y IZero 2 3
!Filter matrix to get corr matrix AH CH
Z ZIden 2 5
!Filter matrix to get corr matrix AC2 CE
Constraint Y*D*Y' - Z*H*Z' /
Options NO_output
End

G6 - Fourth constraint for standardization
Data Constraint NInput_vars=1
Matrices
H Full 5 5 = %E4
Z IZ 1 5
I Identity 1 1
Constraint I - Z*H*Z' /
Options NO_output
End

Group 7 - new constraint for genetic transmission parameter
! The commented out parts of this group are needed if reduced
! genetic correlation between parent and child model is to be
! fitted, and the covariance statement should be commented out
Data Calculation ni=1
! Change the above to Data Constraint ni=1 for Rg model
Matrices
A Symm 1 1
B Symm 1 1
I Iden 1 1
Constraint a /
! Delete the above line for Rg model and uncomment
! the next five lines:
!constraint (a*a+b)-i /
!Specification a
!99
!Specification b
!100
Options None
End
```

```
Group 8 - Rose fear data: MZ twins & their parents
Data NInput_vars=4 NObservations=144
Labels
Husband Wife MZ_Twin1 MZ_Twin2
CMatrix
.903
.113 1.1053
.2547 .0864 .9467
.1602 .2052 .4938 .8947
Matrices
C Full 5 5 = %E4
L Full 4 5
V Diagonal 4 4  !Matrix of standard deviations
E Diagonal 4 4  !Matrix to add random environment for children
Covariances V*(L*C*L'+E*E)*V'/
Specification V
10 10 10 10
Matrix V
1 1 1 1
Labels row V
PH PW PT1 PT2
Labels column V
PH PW PT1 PT2
Labels row E
Dummy Dummy Et1 Et2
Labels column E
Dummy Dummy Et1 Et2
Specification E
0 0 3 3
Boundary 1 5 v 1 1 v 2 2 v 3 3 V 4 4
Labels row L
Ph Pw Pmzt1 Pmzt2
Labels column L
Ph Pw At1 At2 Ct
Specification L
0 0 0 0 0
0 0 0 0 0
0 0 1 0 2
0 0 1 0 2
Value 1 L 1 1 L 2 2
Options RSiduals
End
```

```
Group 9 - Rose social criticism DZ  twins & their parents
Data NInput_vars=4 NObservations=106
Labels
Husband Wife DZ_Twin1 DZ_Twin2
CMatrix
1.0811
.2111 .9063
.1443 .2398 1.2079
.1001 .1551 .3246 1.0318
Matrices
C Full 5 5 = %E4
L Full 4 5
V Diag 4 4 = V8
E Diag 4 4 = E8
Covariances V*(L*C*L'+E*E)*V'/
Specification L
0 0 0 0 0
0 0 0 0 0
0 0 1 0 2
0 0 0 1 2
Value 1 L 1 1 L 2 2
Labels row E
Dummy Dummy Et1 Et2
Labels column E
Dummy Dummy Et1 Et2
Labels row L
Ph Pw Pdzt1 Pdzt2
Labels column L
Ph Pw At1 At2 Ct
Start 1.5 v 1 1 v 2 2 v 3 3 V 4 4
Boundary .5 2 v 1 1 v 2 2 v 3 3 V 4 4
Options RSiduals IT=500
End

Group 10 - summary printout of parameter estimates
Data Calculation
Matrices
P Full 1 10
Compute P/
Specification P
1 2 3 4 5 6 7 8 9 10
Labels c p
```

```
a c e s mu zf zm res(a) res(c) vp
Boundary -.99 .99 4 5 6 7
Boundary 0 1 1 2 3 8 9
Options Multiple
End
!
! Re-fit the model with Father-child and Mother-child
!  cultural transmission set equal
!
Equate f 4 5 1 f 4 5 2
End
```

5.4 Fitting Models to Raw Data

When models are fitted to raw data, it is normal to provide a model for the means as well as for the covariances. Otherwise, there is little difference in the approaches. Mx computes minus twice the log-likelihood of the data, with an arbitrary constant that is a function of the data. Thus there is no overall measure of fit, but there are relative measures of fit, since differences in fit function between submodels are distributed as χ^2.

5.4.1 Estimating Means and Covariances

This section demonstrates maximum likelihood estimation using complete, balanced raw data. A cholesky decomposition (see Figure 5.2) is used for the covariance structure, and the means are estimated separately. Alternative models for covariances or means or both could be used if desired.

```
!
! ML fitting to raw data simulated
! with SAS, whose PROC COR COV gave:
!
! VARIABLE         N         MEAN        STANDARD
!                                        DEVIATION
! P1            1000     0.00182388     0.98499439
! P2            1000    -0.98608262     1.40083917
! P3            1000     2.05400385     1.79139557
!
! COVARIANCE MATRIX
!               P1       P2       P3
! P1       0.970214 0.506058 0.620529
! P2       0.506058  1.96235 0.807754
! P3       0.620529 0.807754   3.2091
```

```
!
! Cholesky for covariance structure
!
ML example, calculation of likelihood for each observation.
Data NInput_vars=3 NObservations=1000 NGroups=1
RA File=[NEALE.MX.EXAMPLES]MLPED.RAW
Matrices
M Full  1 3 Free
S Lower 3 3 Free
Means M /
Covariances S*S' /
Matrix_start_values S
1 0 0
.6 .8 0
.6 .0 .8
Options RM
End
```

The relevant part of the output from this job shows good agreement with the SAS results, calculating means and covariances in the usual fashion, as would be expected for a sample size of 1000. The estimates of the variances are slight underestimates since the ML estimate of a variance is biased, having denominator n instead of $n-1$. Multiplying the ML estimates by 1000/999 we recover the calculated covariances precisely.

```
  ML EXAMPLE, CALCULATION OF LIKELIHOOD...

  MATRIX M
 This is a FULL matrix of order    1 by    3
    0.0018   -0.9861    2.0536

  MATRIX S
 This is a constrained FULL matrix of order    3 by    3
    0.9692    0.5056    0.6199
    0.5056    1.9604    0.8069
    0.6199    0.8069    3.2059
```

5.4.2 Variable Pedigree Sizes

When there are many different possible configurations of data, it is most convenient to use a variable length data file (see Section 2.3.3). This information can be read by Mx and the likelihood of the data may be calculated for any structural model for the covariances and the means. In this example, a Cholesky decomposition (see Figure5.2) of the expected covariance matrix is specified in Group 1.

```
!
! ML fitting to raw variable length data.
!
! Cholesky decomposition for the expected covariance matrix
!
! Also matrix expression for means,
!  - in this case just a simple vector with free parameters.
!
Variable pedigree size ML example.
Data NInput_vars=3 NObservations=1000 NGroups=1
VL File=[neale.mx.examples]UNBALANCED.RAW
Matrices
M Full 1 3 Free
S Lower 3 3 Free
Means M /
Covariances S*S' /
Start .7 s 1 1 - s 3 3
Options RM
End
```

Typical lines of the data file 'unbalanced.raw' look like this:

```
2
1 2   0.5550   -1.1114
3
1 2 3   1.6442   -0.1319  3.609508
3
1 2 3  -0.2145   -1.2193  5.011667
3
1 2 3   2.2274   -1.9423  0.714351
```

There can be problems when beginning to fit models to VL data. One commonly encountered difficulty is that at the starting values, the likelihood is effectively zero for one or more pedigrees. Since Mx is going to try to take the logarithm of the likelihood, some corrective action is required. During iteration, a penalty function is used, but this rarely helps the case where there are poor starting values. To help guide the user to the problem with the starting values, Mx prints out 3 columns of information: the observed and expected means, and then the standardized difference between them. Now if these differences are large (say more than 3 for any variable), it would be a good idea to change starting values of either the means (to make the expected ones closer to the observed) or the variances (to make the standardized difference less). If the starting values of the means are very bad, then it would make sense to change them; however, if they are not, the error may occur with another vector in the dataset, in which case modifying the starting values to increase the expected variances should help. If not, examine the expected covariance matrix; sometimes large expected covariances can make a particular pairings of scores rather unlikely. It is usually better to supply starting values that specify a diagonal variance-covariance matrix, since the overall likelihood is simply the product of the likelihoods of the individual variables. If each of these likelihoods is reasonable, e.g. a standardized difference of less than two, then the overall likelihood will not produce problems unless the number of variables in the vector is very large.

5.5 User-defined Fit Functions

This is a simple example to illustrate the use of a user-defined fit function, in this case least-squares. The model statement evaluates to a scalar which is minimized by Mx. Note that this approach is generally less efficient than using built-in formulae available in Mx.

```
!
! User defined fit function - least squares
!
User defined function to fit to a correlation matrix
!                    by least squares
Data NInput_vars=3 NGroupies=1
CMatrix Symm
1
.2 1
.3 .4 1
Matrices
A Symm 3 3 = %O1
B Stan 3 3 Free
Covariances \tr((a-b)*(a-b))/
Options USer RSiduals
End
```

5.6 Correction for Ascertainment

In Section 4.2.3, we described how proband-ascertained ordinal data could be used to estimate population covariances or genetically informative parameters. This same truncate selection might be applied using a screening instrument, so that only individuals above a certain threshold value are sampled. If such an ascertainment scheme was applied in a pairwise fashion, such that only pairs in which at least one of the pair was above threshold, the likelihood of these observations would require correction for the necessarily missing pairs concordant for being below threshold.

Mathematically, this can be expressed as a double integral of the bivariate normal distribution:

$$L_{\tilde{A}} \quad = \quad \int_{-\infty}^{t}\int_{-\infty}^{t} \phi(v_1, v_2)\, dv_2 dv_1$$

where t is the ascertainment threshold, v_1 and v_2 are the liability values of individuals 1 and 2, and ϕ is the multinormal probability density function. The likelihood of a pair of observations x_1 and x_2 given the ascertainment scheme is therefore:

$$L(x_1, x_2|A) \quad = \quad \frac{\phi(x_1, x_2)}{1 - \int_{-\infty}^{t}\int_{-\infty}^{t} \phi(v_1, v_2)\, dv_2 dv_1}$$

If we use twice the negative log-likelihood as a function to minimize, then the ascertainment correction becomes more clearly distinct:

$$\begin{aligned} -2\ln L(x_1, x_2|A) &= -2\left\{\ln\left(\phi(x_1, x_2)\right) - \ln\left(1 - \int_{-\infty}^{t}\int_{-\infty}^{t} \phi(v_1, v_2)\, dv_2 dv_1\right)\right\} \\ &= -2\ln\left(\phi(x_1, x_2)\right) + 2\ln\left(1 - \int_{-\infty}^{t}\int_{-\infty}^{t} \phi(v_1, v_2)\, dv_2 dv_1\right) \end{aligned}$$

when we have m pairs, the likelihoods are summed over $j = 1 \ldots m$, giving

$$= \quad -2\ln\left(\phi(x_{1j}, x_{2j})\right) + 2m\ln\left(1 - \int_{-\infty}^{t}\int_{-\infty}^{t} \phi(v_1, v_2)\, dv_2 dv_1\right)$$

The first term is the function value calculated by Mx when fitting to raw data. The second term may be calculated by obtaining the value of the integral from a dummy categorical data group using zero observed frequencies in each cell. The threshold in this group should be fixed at the population value, and the correlation should be constrained to equal the correlation of these two variables as estimated from the ascertained data. The expected proportions under the bivariate normal distribution are passed as a matrix using the `%P` constraint as described in Section 3 on page 22. Note that this example does not follow the uppercase convention of uppercase for minimum input needed.

```
Simulated twin data.  Raw ML estimation
Data Ni=2 Ngroups=3 No=1000
Raw_data file=[neale.sas]mzasc.dat
Matrices
M Full 1 2
R Stan 2 2 free
Mean M /
Covariance R /
Matrix M
```

```
0 0
Bound -.99 .99 R 1 2
Options RSiduals
End

Dummy group to calculate expected cell proportions
Data Ni=2
CTable 2 2
0 0           ! It's full of zeros so it
0 0           ! contributes zero to the function
Matrices
T Full 2 1
R Stan 2 2 =R1
Thresholds T /
Covariance R /
Matrix T
1.282
1.282
Options RSiduals
End

Calculate ascertainment correction
Data Ni=0
Matrices
I iden 1 1
J izero 1 2
P full 2 2 = %P1
T full 1 1
Compute T*\ln(I-J*P*J') /
\index{matrix types!constrained!}
Matrix T
2000      ! twice the sample size of group 1
Options User-defined RSiduals Multiple
End
```

Appendix A

Using Mx under different operating systems

Currently Mx is available for VAX VMS, several unix systems (e.g. Vax BSD4.3, Sun-OS, IBM AIX, Dec Ultrix, HP), and MSDOS 3.0 (or higher, with or without Microsoft Windows) or OS2, for IBM compatible 386SX and above. Porting to other environments presents little difficulty, since the code is written in standard FORTRAN. Versions for most operating systems can be downloaded via binary ftp to sapphire.gen.vcu.edu

A.1 Using Mx under VAX VMS

Mx for VMS is distributed with command file (MX.COM) which deals with file extensions and checks to see whether the user has sufficient memory resources. Typically one user at a site, the administrator, will keep the command and executable files with read and execute permission set for all users. Then if users define the symbol mx with a command such as

```
$ MX :== "@DISK1:[BOSS.MX]MX.COM DUMMY"
```

where DISK1 is the name of the diskdrive, BOSS is the name of the administrator, and BOSS.MX is the name of the subdirectory in which the mx.com and mx.exe files are stored. The above symbol definition will permit users to run Mx either interactively or as a batch job. The only difference is that the former will produce a mesmerizing display of the Mx logo and the latter will free up your terminal to do something else while it runs. The output file may be read while the program is running, though emptying the print buffer has been reduced to improve performance.

We recommend that input files have the naming convention `CUTENAME.MX` where `CUTENAME` is a name of your choice. It allows you run Mx with the following syntax:

```
$ MX CUTENAME
```

or alternatively, you can run the job in batch with

```
$ BAT/Q=whateveryoulike MX CUTENAME
```

for short jobs, ignore the /Q= bit. With this syntax, the output will be in a file called `CUTENAME.MXO`; if you wish to use other extensions or names for input and output files, you could, for example, create a file called `BADNAME.ABC`, and use the syntax:

```
$ MX BADNAME.ABC AWFULNAME.XYZ
```

which would create an output file called `AWFULNAME.XYZ`. The command

```
$ MX BADNAME.ABC
```

would generate an output file called `BADNAME.MXO`

We can extend the idea of filename extensions a little further to include: covariance matrices, .COV; correlation matrices, .COR; contingency tables, .CTG; matrices, .MAT; variable length files, .VL; weight matrices, .ASY; inverted weight matrices, .ASI; vectors of means, .MEA; mx save files, .MXS. Of course, it doesn't make any difference to the program what you call the files, but some widely-used conventions such as these help you and other users to understand the content of directory when you (or your colleagues) look at it six months later.

A.2 Using Mx under UNIX

Mx may be used very simply in UNIX by typing

```
mx <inputfile >outputfile
```

and the parameter `&` may be added to run jobs in batch. For very cpu intensive applications the command

```
nice mx <inputfile >outputfile
```

may be used to run the job at lower priority to avoid overtaxing the system.

The IBM RISC 6000 series of computers has good facilities for editing files and executing programs. The INED editor, which is invoked with the command e, offers a user-defined menu system. Thus while editing an Mx input file, batch execution can be initiated simply by selecting an option on the menu. The menu file should be edited to have something like

```
|Batch run Mx on file being edited  |screen|
    (~neale/bin/mx <$FILE >"$FILE"o ; cat ~neale/bin/job.done) &|
```

where neale/bin would be changed to reflect the path where Mx is to be found, and job.done is an ascii file containing a suitable message such as"I have just finished another Mx job for you". This menu option should be configured to save files. It will then begin background Mx execution with a couple of keystrokes. Copies of the necessary profile file for the editor are supplied with the Mx program file; additional copies may be obtained from the author. The only gripe I have with the 6000 series INED editor is that somehow it will place an unrecognizable but invisible character at the start of the next line if the last character on the previous line is underscore (_). Thus matrix formulae spanning several lines should be written, e.g.,

```
 A*B*C' | C
_     C | D*E*D'
```

instead of

```
 A*B*C' | C_
      C | D*E*D'
```

which will not always work under the INED system. I don't have time to investigate this system-dependent problem, so it is likely to stay for a while — unless anyone has any suggestions?

A.3 Using Mx under MSDOS

MX is written in about 12,000 lines of FORTRAN, however it links to a further 20,000 lines of NPSOL code for optimization, so the resultant .exe file does not run within the 640K limit. Fortunately, the Lahey compiler allows binding of a loader to the code which permits Mx to use memory beyond the 640K limit in 386 and 486 machines. Naturally the presence of RAM disks etc. would decrease normally available machine memory, so a utility whatmem.exe is supplied with the program to allow you to check what is actually available. If the program runs out of memory, it will use virtual memory (disk space) instead, but obviously this can drastically decrease performance. The use of Stacker (file compression software) also seems to slow things down a lot, especially for the first run in a multiple fit file.

Note the difference between your computer running out of memory and Mx running out of workspace. Currently, the PC version is configured with 100,000 double precision words of workspace; larger workspace versions are available by special request.

MX now no longer requires that the output files be specified on the command line. This means that under windows and similar environments, .mx files can be associated with the mx.exe program. Double clicking on a .mx file will then run mx and create an output file of the same name but with .mxo as its extension.

A.4 Using Mx under VM

VM is a commonly encountered operating system on IBM mainframes such as the 3081 and 3090 series. It is not my favorite operating system. Porting Mx to this environment was more difficult than to any other, and the `Save` and `Get` functions for binary files are not yet operational. The saving grace of these machines is their speed, although they are now matched by cheaper and easier to use unix workstations.

Appendix B

Error Messages

B.1 General Input Reading

If, while looking for a number, a (non-comment) input line with only non-numeric characters on it (e.g. `MATRIX P`) is found, the program issues a warning:

```
**WARNING** Non-numeric characters found when trying to read a
number
```

This is to alert the user to what is probably an error of not reading enough numbers, either because:

- input lines are too long (120 column maximum)
- the matrix was given the wrong dimensions in the Matrices command
- too few numbers have been supplied

This is not a fatal error, but normally some other problem will arise. Similarly, if too many numbers have been provided for a matrix, (or a mistake was been made when defining the type or dimensions of the matrix) the program will usually try to read a number as a keyword, ask something like "Just what is this keyword supposed to mean?" and stop.

Sometimes an integer overflow will occur if too many digits have been read in free format for a number. Try to keep the number of digits to 9 or less. If you really need more precision, use the exponential format (D+00) or read data from a file.

B.2 Error Codes

This is a list of the error codes reported by Mx. The error messages are supposed to be self-explanatory, but they are usually quite brief.

1 *First input line after title must be DATA.* Will occur if the Title line has been forgotten.

2 *Data line must have NI and NO specified.* (Data groups only)

3 *End of file while trying to read title.*

4 *Must specify Matrices at this point...*

5 *Not a legal matrix name.* Use A-Z, one letter only

6 *Illegal matrix type...* Check for typos.

7 *Sorry, I seem to be at the end of your input file.* Check NG is correct.

8 *Number of selected variables must be less than or equal to NI ...* Otherwise you won't be analyzing something sensible and positive definite.

9 *Error - no variables selected for analysis.*

10 *Terribly sorry, I don't have enough workspace.*

11 *Please try not to refer to matrices that you haven't specified.*

12 *Please use only PATTERN or SPECIFICATION throughout.* Pattern and Specification cannot be used in the same input file.

13 *Error - no matrix specified...*

14 *More than 3 dimensions specified.* After a matrix name, there should be a maximum of 3 numbers to identify the element.

15 *Error - Matrix has not been specified.*

16 *Matrix has been specified not to have free elements.* See Table 3.1 to figure out which types of matrix can have free elements.

17 *You cannot alter off-diagonal elements of a diagonal matrix.*

18 *Incorrect element of a Subdiagonal matrix specified.* You can't alter elements on or above the diagonal of as Subdiagonal matrix

19 *Incorrect element of a Standardized matrix specified.* You can't alter elements on the diagonal of a Standardized matrix.

20 *Please use 2 or 3 dimensional format for elements.* Array references should be group,row,col, with group optional.

21 *Sorry, I can't find your matrix element.* I really should tidy up sometime

22 *I'm sorry, I don't understand this part of DATA line, so I'm ignoring it.* Note that Mx is not LISREL and won't fiddle about changing data structures using the MA= command syntax. You can always do it explicitly in a little Mx job which will be good for your immortal soul.

23 *Sorry, I couldn't invert your expected matrix.* If the method is ML, the expected matrix has to be invertible throughout optimization. Mx will try to avoid non-positive definite areas with a penalty function.

24 *Sorry, I couldn't invert your asymptotic matrix.* If the asymptotic matrix is not positive definite, it must be fixed. Check the Select command for repetitions of a number, if it is used.

25 *Strangely, you seem to have requested an unknown matrix operation.* IF you get this one, memory is screwed up somehow, and you should check your input file carefully before sending the problem to Mike Neale.

26 *Sorry, your model matrix has different dimensions from your data matrix.* A common silly mistake. Carefully figure out the dimension of your model, check NI parameter and Select command if used.

27 *Sorry, your expected matrix is not symmetric.* The matrix formula you provide should yield a matrix that is symmetric, if it is to be fitted to data.

28 ***Sorry, you must have symmetric or full model*** Same as previous message.

29 *Please try not to specify inverse for non-square matrices.* Generalized inverses are not available. If you are desperate, try transforming to a partitioned matrix that has a square matrix of full rank at one end.

30 *Sorry, had trouble transposing a matrix.* This is an unlikely error.

31 *Sorry, I couldn't find the determinant of a matrix.* I thought I put it on the shelf here somewhere... Probably zero.

32 *Uh-oh...* there's a problem with a binary operator. This can happen when evaluating an illegal matrix expression, but it is unlikely.

33 *Sorry, the matrix addition screwed up.* Very unlikely.

34 *The matrices you wish to add are not conformable for addition.* Quite likely. If at first you don't succeed, check check and check again

35 *The matrices you wish to multiply are not conformable for multiplication.* Be sure that you are using the right type of multiplication for your application, as well as checking the dimensions of the matrices you wish to multiply.

36 *You seem to have an unknown matrix type.* Very unlikely.

37 *I seem to be using an unknown fit function.* Very unlikely.

38 *You must specify 3 numbers for boundary constraints.* Lower limit, upper limit and parameter number - or matrix element.

39 *First character after BOUNDARY must be alphanumeric.* Alphanumeric means alphabetic abcdefghijklmnopqrstuv or 01234356789.

40 *Sorry, to request AWLS for this group, you should have input an asymptotic covariance matrix.*

41 *I don't know to what you want me to equate this matrix.* Some error in the =Mi command to equate matrices. Equatee must be on the same input line.

42 *Sorry, I can't make this matrix equal to a matrix that you haven't supplied yet.* Reorder your groups if it isn't a typo.

43 *Uh-oh... I got stuck inverting a matrix while calculating expected matrix...* This can be awkward. Sometimes starting values or changing the boundaries on parameters can help.

44 *Uh-oh... I'm having trouble reading a number in D or E format.* Probably a typo in the data.

45 *Sorry - could you put the =filename on the same line as the FI, please?*

46 *Awfully sorry, I couldn't open a file for you.* Probably a spelling mistake in the filename.

47 *I deeply regret that your equality constraint refers to a non-existent matrix.*

48 *OH NO! The IOP parameter calling MSOFAR is wrong.* I have no idea what this error means. The good thing about it is that you are not likely to get it.

49 *Please don't mix the numeric and array references on a Bound line!* Use only parameter numbers or only matrix elements.

50 *There I was, looking for a number and - blow me down...* Probably an error in specifying the dimensions of a matrix. The end of file was found before the number.

51 *It seems rather strange to me to have a + or - sign without a number after it.* Well, wouldn't it seem strange to you?

52 *You seemed to put a * in the middle of two numbers but WRONGLY.* Do not pass GO, do not collect $200

53 *Uh-oh. Now there are too many numbers in a list.*

54 *Nincompoop. Your data matrix has to be positive definite for GLS.* Probably a little harsh, this message.

55 *Ouch! do not try to change NG...* This used to cause big headaches.

56 *Awfully sorry, old chap, you're trying to | between matrices that...* Check conformability.

57 *Awfully sorry, old chap, you're trying to _ between matrices that...* Check conformability

58 *Uh-oh! Your formula has an illegal character.* Edit your input file and arrest this character immediately. On the IBM RISC 6000 it can occur spuriously and irrationally if you leave blank spaces at the start of a line following an underscore. Heaven knows why.

59 *Uh-oh! Your matrix expression has a mistake in it. Please fix it.* This could be unmatched parentheses, a missing operator or a missing matrix. Sorry that it isn't more specific... Also, the matrix formula is sensitive to untrapped memory problems. One known possibility is that you have tried to do something to a range of matrices from different groups, e.g. Start 1 A 1 1 1 - A 4 3 3

60 *There seems to a problem with your format in your data file.*

61 *Oh dear! The model you specified does not give the same number of rows...* Check the order of the model.

62 *Stack has overflowed - kick Mike Neale.* This is not necessary to fix the job. Your complicated expression ought to be simplified by using a CALC group to precalculate part of it.

63 *The matrices you wish to subtract are not conformable for subtraction.* If this message doesn't add up to you, go back to elementary school.

64 *I'm terribly sorry, stack 1 has overflowed.* Simplify your expression with CALC groups

65 *I'm terribly sorry, stack 2 has overflowed.* Simplify your expression with CALC groups.

66 *An undefined matrix has been encountered in the matrix formula.* Look for typos in the formula, and in the Matrices command. You're using a matrix that hasn't been specified for this group.

67 *You seem to have missed out an operator...* Check the matrix formula for matrix names that are more than one letter.

68 *At first you put % but then there was neither O nor E nor R after it.* So put O or E or R after it!

69 *You seem to be referring to parameter specifications for an Observed or Expected matrix...* There is no way that you are going to be allowed to do this.

70 *I want you to get this right.* If you equate to the Observed or... Get it right.

71 *The matrix you are trying to equate to the O matrix is bigger...* Make it smaller...

72 *The matrix you are trying to equate to the expected matrix is...* Make it the same size.

73 *You have to give NG after CA if the first group is a CALC group.* So do it.

74 *Tut-tut! you are trying to dot-product two matrices that have.* This is different from ordinary matrix multiplication after all.

75 *I tried to read another group and hit the end of file instead.* NG is probably wrong.

76 *Look buster, if you want to define your own function...* See Section 4.2.4

77 *At first I thought you were going to use a function, but then...* You must have boobed somewhere.

78 *Sorry, I only calculate the determinant of SQUARE matrices.*

79 *Sorry, I only calculate the trace of SQUARE matrices.*

80 *I say, if you use the \EXPonent operator, you must make sure that the...*You won't get this error message.

81 *I can't equate this matrix to the observed data of that group, because it...*

82 *Hmmm sorry I don't understand this keyword...*

83 *HEY! I thought you were going to say = but where's the =??*

84 *You can only have MA=CM,PM, or KM right now - sorry!*

85 *Just WHAT is this keyword supposed to mean?...* This is commonly encountered in the middle of a list of numbers if the list is too long. This may be because you have given too many numbers for the type and size of the matrix concerned, or alternatively you may have specified the dimensions of that matrix incorrectly. Note that you should only supply numbers for the modifiable elements of a matrix, which depends heavily on the type of matrix. See Section 3.3 for details on numbers of elements in different types of matrices.

86 *Imagine this: I'm reading a number and I see a character, so I think...* Do be less surprising in your input.

87 *I'm sorry, I can't write a matrix that doesn't exist to a file.* The matrix concerned has to be defined for that group.

88 *To output matrices to files, use* Mx *with 1 letter after it (exceptions %E %M %P %V).*

89 *AAAAGH! You can't assign parameters to the raw data vector.*

90 *If you want to fit to the raw data vectors, you must put them in a vector.*

91 *No, I won't let you do this.* It would overwrite the first raw data observation...

92 *Uh-oh, attempt to take log of zero or negative value imminent.*

93 *In order to use Maximum Likelihood to raw data, it is necessary to supply both...*

94 *Sorry, the expected matrix is singular just now.*

95 *Just WHAT is this keyword supposed to mean?.*

96 *You have tried to convert something that wasn't a vector into a matrix.*

97 *At your starting values, evaluation of the log-likelihood made me take...* Nothing for it... change the starting values. This could be a problem if there are some gruesome outliers, in which case you'd have to edit your data... See Section 5.4.2 for details of how to interpret and respond to the diagnostics printed along with this error message.

98 *Your boundary constraint refers to a parameter not yet specified.*

99 *You have referred to a non-existent row of a matrix.* Mx has a lousy imagination when it comes to that sort of thing.

100 *You have referred to a non-existent column of a matrix.* There are lots of non-existent columns in Athens and Rome.

101 *You can't use the keyword FU when reading diagonal weight matrices or means.*

102 *SORRY I don't select DWLS with correlation matrices yet.* Try again after midnight? Seriously, I'm sorry about this.

103 *Error - you seem to have a non-integer value for the type of person...* Make sure that you have got your VL file right. Be extra careful about SAS missing values.

104 _Please, you must tell me the covariance matrix structure in terms of id codes._

105 _There is someone in this pedigree whose id has not been given._ The mysterious stranger needs to be identified and deleted.

106 _Your model does not define the COVARIANCE between TWO people with this ID._ See Section 2.3.3

107 _An element of a matrix that you have tried to \sqrt()..._

108 _You tried to convert a vector of the wrong length to a Symmetric matrix._ Hah, didn't think I'd notice eh?

109 _I am only prepared to raise the elements of a matrix to the power of a scalar._ Do you have any better ideas? Kroneckerize it or something like that?

110 _You tried to convert a vector of the wrong length to a FULL square matrix._ Silly billy.

111 _Eigenvalues of square matrices only, please_

112 _Eigenvectors of square matrices only, please_

113 _I'm afraid that if you want to fit to raw data you MUST supply models for BOTH means and covariances. You have forgotten a model for the means_ Of course, you may have asked for the wrong kind of fit function on the OPTIONS line.

114 _I'm afraid that if you want to fit to raw data you MUST supply models for BOTH means and covariances. You have forgotten a model for the covariances_ Of course, you may have asked for the wrong kind of fit function on the OPTIONS line.

115 _If you are using Multiple fits, it is impossible to change the matrix formulae - and you can't change boundaries either_

116 _If you wish to Specify, Pattern, or Matrix a matrix, you have to specify the group number **on the same line** BEFORE the Specification, Pattern or Matrix statement_

117 _Please try *not* to refer to non-existent groups!_

118 _When using the Multiple option you *MUST* use 3 numbers to specify a matrix element: Group #, row # and column #_

119 _I'm *so* embarassed. I ran out of workspace._

120 _To read labels for a matrix, you must use the syntax: LABEL ¡R or C¿ Name+_

121 *Labels can't be given for a non-existent matrix* Actually they can, but then the program stops immediately.

122 *Labels may not begin with a number because it could confuse me later on.*

123 *Labels have not been provided for the data, but you seem to be using them to select variables. Use numbers or "Give me labels or give me ..."*

124 *It would seem that you are trying to select a variable that you never supplied. I expect that, being human, you made a mistake in the Labels or Select list*

125 *I can't select a variable that doesn't exist. Make sure that NI is correct or fix the Select list*

126 *Sorry, you can use covariance structures ONLY in constraint or calculation groups; means not allowed*

127 *Incorrect element of a lower triangular matrix specified*

128 *Hey! You must have NInputvars=2 to use contingency tables* A contingency table effectively cross-tabulates two variables, hence NInputvars must be two on the Data line.

129 *I'm afraid that if you want to fit to contingency tables you MUST supply models for BOTH thresholds and covariances. You have forgotten a model for the thresholds* Alternatively you may have requested the wrong fit function on the Options line.

130 *I'm afraid that if you want to fit to contingency tables you MUST supply models for BOTH thresholds and covariances. You have forgotten a model for the covariances* Perhaps you requested the wrong fit function on the Options line?

131 *You must supply a matrix expression for the thresholds that will evaluate to a matrix with 2 rows and with at least as many columns as one less than the number of row categories or the number of column categories, whichever is greater i.e. max(nrowcat-1,ncolcat-1)* See Section 3.2.4 for details about thresholds.

132 *Sorry, but during optimization I have been asked to calculate a bivariate integral with a correlation of 1 or more. This is very unreasonable of you Please fix your model so that this doesn't happen. Use boundaries or something.* Note that the *correlation* is critical here, not the covariance. The correlation is calculated from the expected covariance matrix (the result of the model or covariance statement) as $\mathrm{cov}_{ij}/\sqrt{\mathrm{var}_{ii}\mathrm{var}_{jj}}$

133 *Yes I will fit to n x n contingency tables BUT... n must be greater than or equal to 2* I know a $1 \times n$ is feasible but I haven't written the code for it yet, OK?

134 *If you want me to save or get an Mx binary file then you MUST supply a filename on the same line* It's just one of those things in life that you have to do.

135 *We must have misunderstood each other somehow. You can't get expected proportions %P+ from a group that isn't using contingency tables*

136 *I think memory is screwed up because Mike Neale has screwed up. Call him on 804 786 8590. Or better still, email him on* neale@ruby.vcu.edu *or* neale@vcuruby (bitnet). Has screwed up or is screwed up?

137 *Well well well. You would like graphics. If so why don't you put an = after the DR command???* Mx likes = signs before filenames. I don't know how it got into this habit.

138 *To fit to mean structures as well as covariances you should provide *both* observed means and a model for them (as well you know!) I'm going to use the covariances, but the mean - I'm just going to ignore it OK?* Watch out for this one if you don't seem to get any action with the means.

139 *Really I'm very very sorry about this. It is not under my control. Unless of course I was to use a better language than FORTRAN77 I suppose... However enough philosophising, the problem is that we have reached the end of file too soon. Since you are using * format, you should put one case per line. That is to say, there should be NObs lines each with NI variables in a RA data file* Sometimes error messages are not just explicit, they are introspective. How would you like to be a computer program? Could be our species' destiny.

140 *Really I'm very very sorry about this. It is not under my control. Unless of course I was to use a better language than FORTRAN77 I suppose... However enough philosophy, the problem is that we have reached the end of file too soon. Make sure that your format is OK' Also, remember there should be NObs * NI numbers in your RA data file* See previous error message for sci-fi remark.

141 *Very funny. Hahaha. You want me to standardize a non-square matrix? Just how am I supposed to do that?* If you get any ideas, let me know.

142 *It is with great sadness that I have to tell yo u that I couldn't standardize your matrix because the number I'm supposed to divide by is too small. Perhaps you could avoid this problem with boundary constraints.* Poor machine, it tried!

143 *You are in terrible danger. Don't say multiple until the last group. I know I could have just remembered for you, but I'm lazy too!* There are only so many hours in the day.

144 *Sorry, old chap. You can't specify boundary constraints after Options* I might remedy this problem one day, but for now just meekly go back and put boundary constraints before *the first* option line in this group.

145 *Cough, ahem... can you please give me a matrix that has 1 row and 2 column for the power transformation??* Currently it is only possible to apply a transformation to all variables within each vector. This should be upgraded if there is ever support for Mx; call your political representative now to safeguard its future...

146 *For heaven's sake! Can't you bound your constant so that it is *Greater* than minus the minimum observation??!* Really most nursery school children have a good idea why this should be done. Exponentiating (especially for non-integer exponents) numbers that are less than zero is mathematically awkward, requiring complex numbers and so on. To avoid complex arithmetic, whose implications are unclear to me in this context, Mx demands that if, say, your minimum observation is -3.1, the constant required would have to be greater than +3.1.

147 *YIKES! There's something funny about the power function you request. Remember that after PO you should supply 2 numbers. The significance level (alpha) (0.0 ¡ alpha ¡ 1), and the degrees of freedom of the test (df ¿ 0). Sometimes NAG chokes if alpha is close to 0 or 1* I can't think why this shouldn't be self-explanatory. The numbers should be on the same line as the PO keyword. Note that POwer in this context refers to calculating the statistical power of the study (see Section 5.2.2).

148 *I can calculate confidence intervals in the range 0 to 100. Please try to stay within these bounds'* Everyone has their limits, you know.

Some of these error messages are a little informal. I apologise. I'm terribly terribly sorry and I won't do it again.

What I am really sorry about is if Mx gives you the wrong error message. This is quite rare, but occasionally it is possible to screw up memory, and one of the sensitive areas is the matrix formula (since it contains junk instead of a reverse-polish version of your input formula). One untrapped source is if you say

```
start .5 A 1 2 3 to A 2 2 3
```

in which case everything between the memory addresses of A of group 1 and A of group 2 gets overwritten with .5 A little care can go a long way; so can a little more foolproof programing which I shall try to provide as soon as I can. Please let me know of any nonsensical error messages that you get; elucidation is prerequisite for elimination.

Appendix C

Introduction to Matrices

C.1 The Concept of a Matrix

A matrix is a table of numbers or symbols laid out in *rows* and *columns*.

$$\text{e.g.} \quad \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{pmatrix}$$

The table is enclosed in () or [] in most texts.

It is conventional to specify the configuration of the matrix in terms of Rows $\times$ Columns and these are its *dimensions*. Thus the first matrix above is of dimensions 3 by 2 and the second is a 3×3 *square* matrix.

The most common occurrence of matrices in behavioral sciences is the *data matrix* where the *rows are subjects* and the *columns are measures*.

$$\text{e.g.} \quad \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \end{array} \begin{array}{c} \begin{array}{cc} Wt. & Ht. \end{array} \\ \begin{bmatrix} 50 & 20 \\ 100 & 40 \\ 150 & 60 \\ 200 & 80 \end{bmatrix} \end{array}$$

It is convenient to let a single letter symbolize a matrix. This is written in UPPERCASE **boldface.**

Thus we might say that our data matrix is $\mathbf{A}$, which in handwriting we would underline with either a straight or a wavy line. Sometimes a matrix may be written ${}_4\mathbf{A}_2$ to specify its dimensions. When a matrix consists of a single number, it is called a *scalar*; when it consists of single column (row) of numbers it is called a column (row) *vector*. Vectors are normally represented as a **bold** lowercase. Thus the weights of our four subjects are

$$\begin{bmatrix} 50 \\ 100 \\ 150 \\ 200 \end{bmatrix} = \mathbf{a}$$

C.2 Matrix Algebra

Matrix algebra defines a set of operations that may be performed on matrices. These operations include addition, subtraction, multiplication, inversion (multiplication by the inverse is similar to division) and transposition.

C.2.1 Transposition

A matrix is transposed when the rows are written as columns and the columns are written as rows. This operation is denoted by writing $\mathbf{A}'$ or $\mathbf{A}^T$. In our example,

$$\mathbf{A}' = \begin{pmatrix} 50 & 100 & 150 & 200 \\ 20 & 40 & 60 & 80 \end{pmatrix}$$

a row vector is usually written

$$\mathbf{a}' = \begin{pmatrix} 50 & 100 & 150 & 200 \end{pmatrix}$$

Clearly, $(\mathbf{A}')' = \mathbf{A}$.

C.2.2 Matrix Addition and Subtraction

Matrices may be *added* and to do so they must be of the *same dimension.* They are then said to be *conformable for addition.* Each element in the first matrix is added to the corresponding element in the second matrix to form the same element in the solution.

$$\text{e.g.} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} + \begin{pmatrix} 8 & 11 \\ 9 & 12 \\ 10 & 13 \end{pmatrix} = \begin{pmatrix} 9 & 15 \\ 11 & 17 \\ 13 & 19 \end{pmatrix}$$

or symbolically,

$$\mathbf{A} + \mathbf{B} = \mathbf{C}.$$

You *cannot add*

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} + \begin{pmatrix} 8 & 10 \\ 9 & 11 \end{pmatrix}$$

Subtraction works in the same way as addition, e.g.

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} - \begin{pmatrix} 2 & 5 \\ 2 & 5 \\ 2 & 5 \end{pmatrix} = \begin{pmatrix} -1 & -1 \\ 0 & 0 \\ 1 & 1 \end{pmatrix}$$

which is written

$$\mathbf{A} - \mathbf{B} = \mathbf{C}.$$

C.2.3 Matrix Multiplication

Matrices *may be multiplied* and to do so they must be *conformable for multiplication.* This means that *adjacent columns and rows must be of the same order.* For example, the matrix product ${}_{\mathbf{3}}\mathbf{A}_{\mathbf{2}} \times {}_{\mathbf{2}}\mathbf{B}_{\mathbf{2}}$ may be calculated; the result is a 3×2 matrix. In general, if we multiply two matrices ${}_{\mathbf{i}}\mathbf{A}_{\mathbf{j}} \times {}_{\mathbf{j}}\mathbf{B}_{\mathbf{k}}$, the result will be of order $i \times k$.

Matrix multiplication involves calculating a *sum of cross products* among *rows of the first matrix* and *columns of the second matrix* in all possible combinations.

$$\text{e.g.} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 1 \times 1 + 4 \times 2 & 1 \times 3 + 4 \times 4 \\ 2 \times 1 + 5 \times 2 & 2 \times 3 + 5 \times 4 \\ 3 \times 1 + 6 \times 2 & 3 \times 3 + 6 \times 4 \end{pmatrix}$$

$$= \begin{pmatrix} 9 & 19 \\ 12 & 26 \\ 15 & 33 \end{pmatrix}$$

This is written

$$\mathbf{AB} = \mathbf{C}$$

The only exception to the above rule is multiplication by a *single number* called a scalar. Thus for example,

$$2 \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 2 & 8 \\ 4 & 10 \\ 6 & 12 \end{pmatrix}$$

by convention we write this

$$2\mathbf{A} = \mathbf{C}.$$

It is not possible to use this convention directly in Mx; however, it is possible to define a 1×1 matrix with the constant as the sole element, and use the kronecker product.

The simplest example of matrix multiplication is to multiply a vector by itself. If we premultiply a column vector by its transpose, the result is a scalar called the *inner product.* For example, if

$$\mathbf{a}' = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

then the inner product is

$$\mathbf{a}'\mathbf{a} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = 1^2 + 2^2 + 3^2 = 14$$

which is the sum of the squares of the elements of the vector $\mathbf{a}$. This has a simple graphical representation when $\mathbf{a}$ is of dimension 2×1 (see Figure 1).

C.2.4 Multiplication Exercises

Let

$$\mathbf{A} = \begin{pmatrix} 3 & 6 \\ 2 & 1 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 1 & 0 & 3 & 2 \\ 0 & -1 & -1 & 1 \end{pmatrix}$$

1. Form $\mathbf{AB}$.
2. Form $\mathbf{BA}$. (Careful, this might be a trick question!)

Let

$$\mathbf{C} = \begin{pmatrix} 3 & 6 \\ 2 & 1 \end{pmatrix}, \mathbf{D} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

1. Form $\mathbf{CD}$.
2. Form $\mathbf{DC}$.
3. In ordinary algebra, multiplication is *commutative*, i.e. $xy = yx$. In general, is matrix multiplication commutative?
4. Show for two (preferably non-trivial) matrices conformable for multiplication that $(\mathbf{AB})' = \mathbf{B}'\mathbf{A}'$

Let

$$\mathbf{E}' = \begin{pmatrix} 1 & 0 & 3 \\ 1 & 2 & 1 \end{pmatrix}$$

1. Form $\mathbf{E}(\mathbf{C} + \mathbf{D})$.
2. Form $\mathbf{EC} + \mathbf{ED}$.
3. In ordinary algebra, multiplication is *distributive*, i.e. $x(y + z) = xy + xz$. In general, is matrix multiplication distributive?

C.3 Equations in Matrix Algebra

Matrix algebra provides a very convenient short hand for writing sets of equations. For example, the pair of *simultaneous equations*

$$\begin{aligned} y_1 &= 2x_1 + 3x_2 \\ y_2 &= x_1 + x_2 \end{aligned}$$

may be written

$$\mathbf{y}' = \mathbf{A}\mathbf{x}$$

i.e.

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Also if we have the following pair of equations:

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

$$\mathbf{x} = \mathbf{B}\mathbf{z}$$

Then

$$\begin{aligned} \mathbf{y} &= \mathbf{A}(\mathbf{B}\mathbf{z}) \\ &= \mathbf{A}\mathbf{B}\mathbf{z} \\ &= \mathbf{C}\mathbf{z} \end{aligned}$$

where $\mathbf{C} = \mathbf{A}\mathbf{B}$. This is very convenient notation compared with direct substitution. The LISREL structural equations are written in this general form, i.e.

Real variables (y) = Matrix × hypothetical variables.

To show the simplicity of the matrix notation, consider the following equations:

$$\begin{aligned} y_1 &= 2x_1 + 3x_2 \\ y_2 &= x_1 + x_2 \\ x_1 &= z_1 + z_2 \\ x_2 &= z_1 - z_2 \end{aligned}$$

Then we have

$$\begin{aligned} y_1 &= 2(z_1 + z_2) + 3(z_1 - z_2) \\ &= 5z_1 - z_2 \\ y_2 &= (z_1 + z_2) + (z_1 - z_2) \\ &= 2z_1 + 0 \end{aligned}$$

From $\mathbf{y} = \mathbf{ABz}$, where

$$\mathbf{A} = \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and

$$\mathbf{AB} = \begin{pmatrix} 5 & -1 \\ 2 & 0 \end{pmatrix},$$

or

$$\begin{aligned} y_1 &= 5z_1 - z_2 \\ y_2 &= 2z_2 \end{aligned}$$

C.4 Calculation of Covariance Matrix from Data Matrix

Suppose we have a data matrix $\mathbf{A}$ with rows corresponding to subjects and columns corresponding to variables. We can calculate a mean for each variable and replace the data matrix with a matrix of *deviations from the mean.* That is, each element a_{ij} is replaced by $a_{ij} - \mu_j$ where μ_j is the mean of the j^{th} variable. Let us call the new matrix $\mathbf{X}$. The covariance matrix is then simply calculated as:

$$\frac{1}{N-1}\mathbf{X}'\mathbf{X}$$

where N is the number of subjects.

For example, suppose we have the following data:

X	Y	$X - \bar{X}$	$Y - \bar{Y}$
1	2	-2	-4
2	8	-1	2
3	6	0	0
4	4	1	-2
5	10	2	4

So the matrix of deviations from the mean is

$$\mathbf{X} = \begin{pmatrix} -2 & -4 \\ -1 & 2 \\ 0 & 0 \\ 1 & -2 \\ 2 & 4 \end{pmatrix}$$

and therefore the covariance matrix of the observations is

$$\begin{aligned} \frac{1}{N-1}\mathbf{X}'\mathbf{X} &= \frac{1}{4}\begin{pmatrix} -2 & -1 & 0 & 1 & 2 \\ -4 & 2 & 0 & -2 & 4 \end{pmatrix}\begin{pmatrix} -2 & -4 \\ -1 & 2 \\ 0 & 0 \\ 1 & -2 \\ 2 & 4 \end{pmatrix} \\ &= \frac{1}{4}\begin{pmatrix} 10 & 12 \\ 12 & 40 \end{pmatrix} \\ &= \begin{pmatrix} 2.5 & 3.0 \\ 3.0 & 10.0 \end{pmatrix} = \begin{pmatrix} S_x^2 & S_{xy} \\ S_{xy} & S_y^2 \end{pmatrix} \end{aligned}$$

The correlation is

$$\frac{S_{xy}}{\sqrt{S_x^2 S_y^2}} = \frac{S_{xy}}{S_x S_y}$$

In general, a correlation matrix may be calculated from a covariance matrix by pre- and post-multiplying the covariance matrix by a *diagonal matrix* $\mathbf{D}$ in which each diagonal element d_{ii} is $\frac{1}{S_i}$, i.e. the reciprocal of the standard deviation for that variable. Thus in our two variable example, we have:

$$\begin{pmatrix} \frac{1}{S_x} & 0 \\ 0 & \frac{1}{S_y} \end{pmatrix}\begin{pmatrix} S_x^2 & S_{xy} \\ S_{xy} & S_y^2 \end{pmatrix}\begin{pmatrix} \frac{1}{S_x} & 0 \\ 0 & \frac{1}{S_y} \end{pmatrix} = \begin{pmatrix} 1 & R_{xy} \\ R_{xy} & 1 \end{pmatrix}$$

Transformations of Data Matrices

Matrix algebra provides a natural notation for *transformations*. If we premultiply the matrix ${}_{\mathbf{i}}\mathbf{B}_{\mathbf{j}}$ by another, say ${}_{\mathbf{k}}\mathbf{T}_{\mathbf{i}}$, then the rows of $\mathbf{T}$ describe linear combinations of the rows of $\mathbf{B}$. The resulting matrix will therefore consist of k rows corresponding to the linear transformations of the rows of $\mathbf{B}$ described by the rows of $\mathbf{T}$. A very simple example of this is premultiplication by the *identity matrix* (written $\mathbf{I}$), which merely has 1's on the leading diagonal and zeroes everywhere else. Thus the transformation described by the first row may be written as 'multiply the first row by 1 and add zero times the other rows.' In the second row, we have 'multiply the second row by 1 and add zero times the other rows,' and so the identity matrix transforms the matrix **B** into the same matrix. For a less trivial example, let our data matrix be **X**, then

$$\mathbf{X}' = \begin{pmatrix} -2 & -1 & 0 & 1 & 2 \\ -4 & 2 & 0 & -2 & 4 \end{pmatrix}$$

and let

$$\mathbf{T} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

then

$$\begin{aligned} \mathbf{Y}' &= \mathbf{T}\mathbf{X}' \\ &= \begin{pmatrix} -6 & 1 & 0 & -1 & 6 \\ 2 & -3 & 0 & 3 & -2 \end{pmatrix}. \end{aligned}$$

In this case, the transformation matrix specifies two transformations of the data: the first row defines the sum of the two variates, and the second row defines the difference (row 1 - row 2). In the above, we have applied the transformation to the raw data, but for these linear transformations it is easy to apply the transformation to the covariance matrix. The covariance matrix of the transformed variates is

$$\begin{aligned} \frac{1}{N-1}\mathbf{Y}'\mathbf{Y} &= \frac{1}{N-1}(\mathbf{T}\mathbf{X}')(\mathbf{T}\mathbf{X}')' \\ &= \frac{1}{N-1}\mathbf{T}\mathbf{X}'\mathbf{X}\mathbf{T}' \\ &= \mathbf{T}\mathbf{V}_{\mathbf{x}}\mathbf{T}' \end{aligned}$$

which is a useful result, meaning that linear transformations may be applied directly to the covariance matrix, instead of going to the trouble of transforming all the raw data and recalculating the covariance matrix.

Determinants

For a square matrix **A** we may calculate a scalar called the *determinant* which we write as $|\mathbf{A}|$. In the case of a 2×2 matrix, this quantity is calculated as

$$|\mathbf{A}| = a_{11}a_{22} - a_{12}a_{21}$$

The determinant has an interesting geometric representation. For example, consider two standardized variables that correlate r. This situation may be represented graphically by drawing two vectors, each of length 1.0, having the same origin and an angle cosine r between them (see figure 2). It can be shown (i.e. this is a toughie that involves symmetric square root decomposition of matrices, eigenvalues etc. that I'm not going to do here) that the area of the triangle OV_1V_2 is $.5\sqrt{|\mathbf{A}|}$. Thus as the correlation r increases, the angle between the lines decreases, the area decreases and *the determinant decreases.* For two variables that correlate perfectly, the determinant of the correlation (or covariance) matrix is zero. For larger numbers of variables, the determinant is a simple function of the hypervolume in n-space; if any single pair of variables correlates perfectly then the determinant is zero. In addition, if one of the variables is a linear combination of the others, the determinant will be zero.

To calculate the determinant of larger matrices, we employ the concept of a *cofactor.* If we delete row i and column j from an $n \times n$ matrix, then the determinant of the remaining matrix is called the *minor* of element a_{ij}. The cofactor, written A_{ij} is simply:

$$A_{ij} = (-1)^{i+j} \text{minor } a_{ij}$$

The determinant of the matrix $\mathbf{A}$ may be calculated as

$$|\mathbf{A}| = \sum_{i=1}^{n} a_{ij} A_{ij}$$

where n is the order of $\mathbf{A}$.

The determinant of a matrix is related to the concept of *definiteness* of a matrix. In general, for a null column vector $\mathbf{x}$, the quadratic form $\mathbf{x}'\mathbf{A}\mathbf{x}$ is always zero. For some matrices, this quadratic is zero *only* if $\mathbf{x}$ is the null vector. If $\mathbf{x}'\mathbf{A}\mathbf{x} > 0$ for all non-null vectors $\mathbf{x}$ then we say that the matrix is *positive definite.* Conversely, if $\mathbf{x}'\mathbf{A}\mathbf{x} < 0$ for all non-null $\mathbf{x}$, we say that the matrix is *negative definite.* However, if we can find some non-null $\mathbf{x}$ such that $\mathbf{x}'\mathbf{A}\mathbf{x} = 0$ then the matrix is said to be *singular*, and its determinant is zero. As long as no two variables are perfectly correlated, and there are more subjects than measures, a covariance matrix calculated from data on random variables will be *positive definite.* LISREL will complain (and rightly so!) if it is given a covariance matrix that is not positive definite. The determinant of the covariance matrix can be helpful when there are problems with model-fitting that seem to originate with the data. However, it is possible to have a matrix with a positive determinant yet which is negative definite (consider $-\mathbf{I}$ with an even number of rows), so the determinant is not an adequate diagnostic. Instead we note that all the eigenvalues of a positive definite matrix are greater than zero. Eigenvalues and eigenvectors may be obtained from software packages and the numerical libraries listed above[1].

[1] Those readers wishing to know more about the uses of eigenvalues and eigenvectors may consult Searle (1982) or any general text on matrix algebra.

Inverse of a Matrix

Just as there are many uses for the operation of division in ordinary algebra, there are many valuable applications of the inverse of a matrix. We write the inverse of the matrix $\mathbf{A}$ as $\mathbf{A^{-1}}$, and one of the most important results is that

$$\mathbf{AA^{-1}} = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix. In this case, the multiplication operation is commutative, so it is also true that

$$\mathbf{A^{-1}A} = \mathbf{I}$$

There are many computer programs available for inverting matrices. Some routines are general, but there are often faster routines available if the program is given some information about the matrix, for example whether it is symmetric, positive definite, triangular or diagonal. Here we describe one general method that everyone should use at least once in their lives for at least a 3×3 matrix.

Procedure for Inversion of a General Matrix

In order to invert a matrix, the following four steps can be used:

1. Find the determinant
2. Set up the matrix of cofactors
3. Transpose
4. Divide by the determinant

For example, the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 1 & 5 \end{pmatrix}$$

1.

$$|\mathbf{A}| = (1 \times 5) - (2 \times 1) = 3$$

2.

$$\begin{aligned} A_{ij} &= \begin{bmatrix} (-1)^2 \times 5 & (-1)^3 \times 1 \\ (-1)^3 \times 2 & (-1)^4 \times 1 \end{bmatrix} \\ &= \begin{pmatrix} 5 & -1 \\ -2 & 1 \end{pmatrix} \end{aligned}$$

3.

$$A'_{ij} = \begin{pmatrix} 5 & -2 \\ -1 & 1 \end{pmatrix}$$

4.

$$\mathbf{A^{-1}} = \frac{1}{3}\begin{pmatrix} 5 & -2 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{5}{3} & \frac{-2}{3} \\ \frac{-1}{3} & \frac{1}{3} \end{pmatrix}$$

To verify this, we can multiply $\mathbf{AA^{-1}}$ to obtain the identity matrix:

$$\frac{1}{3}\begin{pmatrix} 5 & -2 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 1 & 5 \end{pmatrix} = \frac{1}{3}\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The result that $\mathbf{AA^{-1}} = \mathbf{I}$ may be used to solve the pair of simultaneous equations:

$$\begin{aligned} x_1 + 2x_2 &= 8 \\ x_1 + 5x_2 &= 17 \end{aligned}$$

which may be written

$$\begin{pmatrix} 1 & 2 \\ 1 & 5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 8 \\ 17 \end{pmatrix}$$

i.e.

$$\mathbf{Ax} = \mathbf{y}$$

premultiplying both sides by the inverse of $\mathbf{A}$, we have

$$\begin{aligned} \mathbf{A^{-1}Ax} &= \mathbf{A^{-1}y} \\ \mathbf{x} &= \mathbf{A^{-1}y} \\ &= \frac{1}{3}\begin{pmatrix} 5 & -2 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 8 \\ 17 \end{pmatrix} \\ &= \frac{1}{3}\begin{pmatrix} 6 \\ 9 \end{pmatrix} \\ &= \begin{pmatrix} 2 \\ 3 \end{pmatrix} \end{aligned}$$

which may be verified by substitution.

For a larger matrix it is more tedious to compute the inverse. Let us consider the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}$$

1. The determinant is

$$\begin{aligned} |\mathbf{A}| &= +1\begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix} - 1\begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} + 0\begin{vmatrix} 1 & 0 \\ 1 & -1 \end{vmatrix} \\ &= +1 + 1 + 0 \\ &= 2 \end{aligned}$$

2. The matrix of cofactors is:

$$\begin{aligned} A_{ij} &= \begin{bmatrix} +\begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix} & -\begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} & +\begin{vmatrix} 1 & 0 \\ 1 & -1 \end{vmatrix} \\ -\begin{vmatrix} 1 & 0 \\ -1 & 0 \end{vmatrix} & +\begin{vmatrix} 1 & 0 \\ 1 & 0 \end{vmatrix} & -\begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} \\ +\begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} & -\begin{vmatrix} 1 & 0 \\ 1 & 1 \end{vmatrix} & +\begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} \end{bmatrix} \\ &= \begin{pmatrix} 1 & 1 & -1 \\ 0 & 0 & 2 \\ 1 & -1 & -1 \end{pmatrix} \end{aligned}$$

3. The transpose is

$$A'_{ij} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & -1 \\ -1 & 2 & -1 \end{pmatrix}$$

4. Dividing by the determinant, we have

$$\mathbf{A^{-1}} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & -1 \\ -1 & 2 & -1 \end{pmatrix} = \begin{pmatrix} .5 & 0 & .5 \\ .5 & 0 & -.5 \\ -.5 & 1 & -.5 \end{pmatrix}$$

which may be verified by multiplication with $\mathbf{A}$ to obtain the identity matrix.

References

Aitken, A. C. (1934). Note on selection from a multivariate normal population. *Proceedings of the Edinburgh Mathematical Society B*, *4*, 106–110.

Aitken, A.C. (1934-35) On least squares and the linear combination of observations. *Proceedings of the Royal Society of Edinburgh,* **55:**42-48

Akaike, H. (1987) Factor analysis and AIC. *Psychometrika* **52,** 317-332.

Bentler, P.M. (1989) EQS Structural Equations Program Manual. Los Angeles: BMDP.

Browne, M.W. (1974). Generalized least squares estimators in the analysis of covariance structures. *South African Statistical Journal* **8:**1-24.

Browne, M.W. (1982). Covariance structures. In D.M. Hawkins (ed.): *Topics in applied multivariate analysis.* Cambridge: Cambridge University Press.

Browne, M.W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology.* **37:**1-21.

Browne, M. W. and Cudeck, R. (1990). Single sample cross-validation indices for covariance structures. *Multivariate Behavioral Research* **24:**445-455.

Cardon, L.R., Fulker, D.W. & Jöreskog, K. (1991) A LISREL model with constrained parameters for twin and adoptive families. *Behavior Genetics* **21:**327-350

Carey, G. (1986). A general multivariate approach to linear modeling in human genetics. *Am. J. Hum. Genet.* **39:**775-786

Eaves, L.J., Last, K., Young, P.A., Martin, N.G. (1978) Model fitting approaches to the analysis of human behaviour. *Heredity* **41:**249-320.

Fraser, C. (1988). *COSAN User's Guide* Unpublished documentation, Centre for Behavioural Studies in Education, University of New England, Armidale NSW Australia 2351.

Fulker, D.W. (1982). Extensions of the classical twin method. In *Human genetics, part A: The unfolding genome* (pp. 395–406). New York: Alan R. Liss.

Fulker, D.W. (1988) Genetic and cultural transmission in human behavior. In: B.S. Weir, E.J. Eisen, M. Goodman & G. Namkoong (eds.) *Proceedings of the Second International Conference on Quantitative Genetics* Masschusetts: Sinauer Associates Inc.

Heath, A.C., Neale, M.C., Hewitt, J.K., Eaves, L.J., & Fulker, D.W. (1989a) Testing structural equation models for twin data using LISREL. *Behavior Genetics*, **19:**9-36

Heath, A.C., Eaves, L.J., & Martin, N.G. (1989b). The genetic structure of personality III. Multivariate genetic item analysis of the EPQ scales. *Personality and Individual Differences* **10:**877-888.

Jöreskog, K.G. (1967). Some contributions to maximum likelihood factor analysis. *Psychometrika,* **32:**443-482

Jöreskog, K.G. & Goldberger, A.S. (1972). Factor analysis by weighted least squares. *Journal of the American Statistical Association* **10:**631-639.

Jöreskog, K.G. (1973). A general method for estimating a linear structural equation system. In: A.S. Goldberger & O.D. Duncan (Eds.) *Structural equation models in the social sciences.* New York: Seminar Press.

Jöreskog, K.G. & Sörbom, D. (1986). *PRELIS: A Preprocessor for LISREL.* Scientific Software: Mooresville, Indiana.

Jöreskog, K.G. & Sörbom, D. (1989). *LISREL 7: A Guide to the Program and Applications* 2nd Edition. Chicago, Illinois: SPSS Inc.

Jöreskog, K.G. & Sörbom, D. (1993). *New features in PRELIS 2* Chicago: Scientific Software International, Inc.

Kaplan, D. (1990) Evaluating and modifying covariance structure models: A review and recommendation. *Multivariate Behavioral Research* **25:**137-155.

Karlin, S., Cameron, E.C. & Chakraborty, R. (1983). Path analysis in genetic epidemiology: A critique. *American Journal of Human Genetics.* **35:**695-732.

Lange, K., Westlake, J., Spence, M. (1976) Extensions to pedigree analysis: III. Variance components by the scoring method. *Annals of human genetics* **39:**485-491.

Maxwell, A.E. (1977) *Multivariate Analysis in Behavioral Research* New York: John Wiley.

McArdle, J.J. (1980). Causal modelling applied to psychonomic systems simulation. *Behavior Research Methods and Instrumentation* **12:**193-209

McArdle, J.J. & Boker, S.M. (1990). RAMpath: Path diagram software. Denver:

Data Transforms Inc.

NAG (1990) The NAG Fortran Library Manual, Mark 14. Oxford: Numerical Algorithms Group.

Neale, M.C. & Cardon, L.R. (1992) *Methodology for genetic studies of twins and families* Dordrecht, NL: Kluwer Academic Publishers.

Neale, M.C. & Fulker, D.W. (1984). A bivariate path analysis of fear data on twins and their parents. *Acta Geneticae Medicae et Gemellologiae*, **33:**273–286.

Neale, M.C., Heath, A.C., Hewitt, J.K., Eaves, L.J., & Fulker, D.W. (1989). Fitting genetic models with LISREL: Hypothesis testing. *Behavior Genetics* **19:**37-50.

Neale, M.C. & Martin, N.G. (1989). The effects of age, sex and genotype on self-report drunkenness following a challenge dose of alcohol *Behavior Genetics* **19:**63-78.

Neale, M.C. & McArdle, J.J. (1990). The analysis of assortative mating: A LISREL model *Behavior Genetics* **20:**287-298

Phillips, K., Fulker, D.W., Carey, G. & Nagoshi, C.T. (1988). Direct marital assortment for cognitive and personality variables. *Behav. Genet.* **18:**347-356.

Rigdon, E. E. and Ferguson, C. E. (1991) The performance of the polychoric correlation coefficient and selected fitting functions in confirmatory factor analysis with ordinal data. *Journal of Marketing Research* **28:**491-497

Rindskopf, D.A. (1984) The use of phantom and imaginary latent variables to parameterize constraints in linear structural models. *Psychometrika* **49:**37-47.

Schervish, M.J. (1984) Multivariate normal probability with error bounded. *Appl. Stat.* **33:**81-94

Searle, S.R. (1982). *Matrix Algebra Useful for Statistics.* New York: John Wiley.

Van Eerdewegh, P. (1982). *Statistical Selection in Multivariate Systems with Applications in Quantitative Genetics.* Unpublished doctoral dissertation, Washington University, St. Louis, Mo.

Vogler, G.P. (1985). Multivariate path analysis of familial resemblance. *Genet. Epid.* **2:**35-54.