TIMMMIG: A PROGRAM FOR EXTRACTING MIGRATION TIME SERIES TABLES

Oliver Duke-Williams and Philip Rees

WORKING PAPER 93/13

SCHOOL OF GEOGRAPHY • UNIVERSITY OF LEEDS

TIMMMIG: A PROGRAM FOR EXTRACTING MIGRATION TIME SERIES TABLES

Oliver Duke-Williams
and Philip Rees

WORKING PAPER 93/13

School of Geography
University of Leeds
Leeds LS2 9JT

May 1993

CONTENTS

Abstract

This paper dsecribes the first of three stages involved in building a continuous time series of migration data for the United Kingdom and linking that database to user friendly software for extracting statistics. Stage (1) arranges the databases as three aggregated arrays for the period from mid-1975 to the most recent quarter. Stage (2) will create a parallel population database and stage (3) an individual migration database from mid-1983 onwards.

Sections 1 and 2 introduce and detail respectively the NHSCR data source. Section 3 describes the functionality of the TIMMIG software for accessing the data. Section 4 provides example applications of the software. Section 5 provides instructions for accessing and operating the software, using Leeds University's General Purpose Server (GPS) computer system. Section 6 provides a conclusion and describes plans for future developments of the software.

Appendices contain a list of the area identification codes that are necessary to use the software, together with listings of the various programs and subroutines which form the TIMMIG package.

Acknowledgements

The work is funded by a grant (A507 26 5019) under the ESCR / JISC 1991 Census of Population Programme for the Analysis of Migration flows. The migration data accessed by the software described here are produced by the Office of Population Censuses and Surveys from the National Health Service Central Register. Users of the data should provide the folowing acknowledgement: "Source: Crown Copyright (OPCS). Supplied through the ESRC / JISC Census Programme."

List of tables

List of figures

1. INTRODUCTION

1.1 Aim

The aim of this paper is to describe how to access an important national database on migration, supplied by the Office of Population Censuses and Surveys (OPCS). The data are provided on magnetic tape as the migrations made by individuals between sets of defined areas. These individual records need to be turned into meaningful cross tabulations of the migrations occurring between the areas or aggregations of them, classified by age, sex and time interval. The TIMMIG (TIMe series of MIGration data) program has been constructed to provide researchers (including students) with the means of generating such cross tabulations easily via a campus computing network. We envisage the fully developed version of the database and program being made available nationally, to the academic community. The immediate use to which TIMMIG will be put is to provide arrays of migration data for the period one year prior to the 1991 Census for comparison with the Census migration statistics.

The rest of section 1 provides the background to the migration data used and to the conceptual frameworks used. Section 2 of the paper provides a detailed record of the data files employed. Section 3 outlines the functionality of TIMMIG, which are illustrated in section 4. Finally, section 5 describes how the program can be used.

1.2 Introduction to the NHSCR migration data

The Census Offices of the UK (OPCS, General Registrars Office (Scotland), (GRO(S)) and Census Office of Northern Ireland CONI) liase closely with the National Health Service (NHS) in making statistical use of the patient registration records of the NHS. These serve, for many purposes, as a surrogate for a full population register (discontinued in 1948 after wartime use). One of the uses to which they are put is in the compilation of statistics on internal migration. A migration question has been asked in the dicennial Census since 1961, but it has only covered the period of the year prior to the date of the Census in the last two Censuses.

In theory, it should be possible to record in a continuously maintained register such as the NHS Central Register (NHSCR) all changes of address by the persons registering. However, there is no obligation on persons registering with the NHS to report a change of address until some form of service is sought. As a result no register of changes of address exists, although it may be possible in future, with the computerisation of the patient registration records in Family Health Service Authorities (FHSA) from the December quarter of 1990, to generate matrices of inter-area migration at any scale.

In practice, the Census Offices have confined their attentions to those changes of address that can easily be recorded in the NHS registration system. These are the changes of address that involve the transfer of the patient record from one FHSA to another, the introduction of a new patient record

from abroad or from the Armed Forces Medical Service or the deletion of an old patient record as a result of emigration or entry to the Armed Forces. Rosenbaum and Bailey (1991 p.24) summarise the process thus:

> " After someone moves and registers with a new doctor, the doctor notifies the relevant FHSA. If the move involves a change of FHSA, the new authority informs NHSCR by forwarding the name, address, date and place of birth, NHS number, and the FHSA or previous registration of patient. NHSCR then records the move and instructs the previous FHSA to send the patients documents to the new FHSA."

The date of birth and sex of the patient are recorded at the time of re-registration, making possible age and gender disaggregation of the derived datset.

These migration data are provided in a variety of forms including aggregated arrays and individual records. The data volumes involved are large: 35.9 million migration records from mid-1983 onwards. To handle these data volumes a step-by-step strategy was designed.
The steps of the strategy are as follows:

(1) Create and make accessible an *aggregate migration database* in the form of three aggregated arrays for the period mid-1975 to the most recent quarter (currently September 1992);

(2) Create a parallel *population database* to provide the population denominators in computations of occurrence-exposure migration rates;

(3) Create an *individual migration database* in compressed form of the individual migrations for the period mid-1983 onwards.

Step (1) builds on the arrays available from mid-1975 onwards in aggregated form but the age aggregations are fixed at 5 year age groups to 85+. Step (3) will make it possible to produce single year of age tables or to carry out aggregations not fixed to five year age intervals. This step will also introduce the full spatial disaggregation. The three arrays that constitute step (1) are as follows:

(a) *The origin-destination-time (ODT) array*

The origin-destination-time array for FHSAs (called Family Practitioner Committees prior to September 1990) has 97 origins and 97 destinations (94 FHSAs in England and Wales plus Northern Ireland, Scotland and the Isle of Man): migrations are aggregated over age and gender. The array is available, currently, for 17 single year time periods from mid-1975 / mid-1976 to mid-1991 / mid-1992.

(b) *The origin-age-sex-time (OAST) array*

The array contains out-migration totals from the 97 FHSAs, disaggregated by five year age groups from 0-4 to 85+ and by gender for 17 single year time periods from mid-75 / mid-76 to mid-91 / mid-92.

(c) *The destination-age-sex (DAST) array*

The array contains in-migration totals from the 97 FHSAs, disggregated by five year age groups from 0-4 to 85+ and by gender for 17 single year time periods from mid-75 / mid-76 to mid-91 / mid-92.

The step (2) database uses local-authority districts (in Great Britain) or OPCS's building blocks (prior to 1983) as the spatial unit. These are the units for which mid-year population estimates are prepared by OPCS and GRO(S), and which can be aggregated to form the populations of the larger units (FPCs / FHSAs in England and Wales; AHBs in Scotland) for which migration data are reported.

The step (3) database uses the expanded set of areas available from mid-year 1983 - the additional AHBs in Scotland - and from mid-1986 - the additional 5 FPCs / FHSAs that constitute 'Middlesex'.

1.3 Software and paper versions

This paper represents a first edition of a description and guide to the TIMMIG software; new edition(s) will be produced in line with developments in the software. The software at the time of writing is at version 2.0.

2. THE NHSCR DATA SOURCE

The data used to construct this migration database are the National Health Service Central Register (NHSCR) records referring to patient re-registrations. This section concentrates on a technical description on the source, structure and formats of the data, and of the primary methods of processing the data.

2.1 General description of the dataset

2.1.1 Source of data

The data are received quarterly on magnetic tapes from the Office of Population Censuses and Surveys (OPCS). These tapes are copied by staff at the University Computing Services (Leeds) onto blank magnetic tapes provided by the School of Geography, and the original tapes are then returned to OPCS.

2.1.2 Description of dataset

The dataset comprises a series of records giving details of the origin and destination of each individual who is being re-registered, together with some details about the individual (age, sex, year of birth). The records do not contain information that specifically identifies the person, such as names or NHS identification numbers. The structure of the record has changed over the time period that the database refers to. Table 1 illustrates the major changes to the record structure. Most of the changes concern the way in which the origin and destination are identified, with, over time, a greater number of different geographical areas being recognised; some of the changes however concern the actual format of the record.

2.1.3 Time lags in the data

The time-lag (which has been reduced since 1990) must be taken into account. Prior to December 1990, there was a time lag between a person migrating and this move being recorded on the Register. This time lag was due to a 'real' delay between a person moving house and registering with a new doctor, and in administrative delays in sending the forms to the NHS Central Register (at Southport, Edinburgh or Belfast) after this. The total time lag was estimated to be (on average) three months. As a result, magnetic tapes of registry data for a given quarter were assumed to refer to migration events which actually occurred in the *preceding* quarter (for example a tape containing registry data labelled as 'March quarter' (meaning, the three months to the end of March) would actually contain data about transfers taking place during the December quarter).

Since December 1990 computerisation of the Register has reduced administrative delays, and the average time-lag is now estimated to be one month. Consequently, the data on a given tape now

contains data about moves occurring during the quarter that the tape is labelled as referring to. That is, the tape for a given quarter contains re-registration data for the last two months of that quarter and the first month of the following quarter. For example, a tape labelled as 'March quarter' will contain records collated during February, March and April; these records are assumed to contain a one month time-lag, and thus refer to moves taking place in January, February and March.

2.2 NHSCR Tapes collection

Table 2 contains a list of the migration data tapes which the School of Geography owns at the time of writing. These are added to at quarterly intervals as new tapes are received from OPCS. The table is subdivided by quarters. For all tapes (that is, both pre- and post- computerisation of the Register), these subdivisions represent the quarters during which the migrations are estimated to have occurred. The 'tape number' column gives the index number given to the tape by the University Computing Service, Leeds.

The shading bands in table 2 are used to distinguish the specific tapes on which the data are held.

Table 1: Changes in the NHSCR record structure

Period	Form of Record	Number of Zones	Description
1975 quarter 3 to 1983 quarter 2	Table cell	97	Data taken from a series of output table produced by OPCS. Great Britain sub-divided into 97 zones comprised mostly of Family Practitioner Committee areas, but with 'Scotland' as a single zone.
1983 quarter 3 to 1986 quarter 2	Individual record	111	Primary Unit Data (PUD) made available (i.e. a series of individual records) as a 100% sample. The Scottish data are subdivided into 15 Scottish Area Health Boards (AHBs).
1986 quarter 3 to 1990 quarter 3	Individual record	115	'Middlesex', previously one category is subdivided into five separate areas.
1990 quarter 4 onwards.	Individual record	115	The Register data are computerised. This reduces the average time-lag between the patient moving and being re-registered from three months to one month. At the same time the record length is increased, to include categories such as date of birth and date of migration.

Notes:

1. A list of the geographical areas is included in the appendix A.
2. In column 1, all references are to the times during which migrations occurred. Prior to quarter 4 of 1990, the tapes were originally labelled as referring to the period when the data was compiled. (See paragraph 2.1.3).

Table 2: The NHSCR tapes collection

Quarter ending [1]	Tape number	Number of records [2]	Additional notes
Sep. 1983	F36903	1,137,600	
Dec. 1983	F36904	1,175,880	
Mar. 1984		977,192	
Jun. 1984	F36803	913,880	Tape contains both quarters
Sep. 1984		1,202,660	
Dec. 1984	F36802	1,091,010	Tape contains both quarters
Mar. 1985		902,886	
Jun. 1985		1,046,700	Tape contains both quarters
Sep. 1985		1,059,658	
Dec. 1985	F36834	1,176,168	Tape contains both quarters
Mar. 1986		1,174,390	
Jun. 1986		1,100,754	
Sep. 1986	F38374	1,267,886	Marked Dec '86 A
	F38375		Marked Dec '86 B
Dec. 1986	F38376	1,403,934	Marked Mar '87 A
	F38377		Marked Mar '87 B
Mar. 1987	F38371	1,021,022	Tape contains both quarters
Jun. 1987		1,089,702	
Sep. 1987	F38405	1,302,610	
Dec. 1987	F38406	1,411,830	
Mar. 1988	F38407	1,195,670	Tape contains both quarters
Jun. 1988		1,161,460	
Sep. 1988	F38372	1,212,408	
Dec. 1988	F38442	1,415,438	
Mar. 1989	F38443	1,122,974	Tape contains both quarters
Jun. 1989		1,191,496	
Sep. 1989	F38799	1,191,496	
Dec. 1989	F38801		
Mar. 1990	F38802		
Jun. 1990	F39084		
Sep. 1990			See Note 3
Dec. 1990		1,252,012	See Note 4
Mar. 1991	F39082	863,026	
Jun. 1991	F39083	963,416	
Sep. 1991	F39084	1,381,502	
Dec. 1991	F39085	1,191,372	
Mar. 1992	F39086	1,027,648	
Jun. 1992	F39087		
Sep. 1992		1,440,872	
		37,066,552	

Notes to Table 2:

1. This is the period the tapes were originally labelled as; not necessarily the period that the migration occurred during (see section 2.1.3).
2. This (unless noted otherwise) is the number of records quoted in the original documentation received from OPCS.
3. This is the last tape in the 'pre-computerisation' series. It refers to moves taking place in the third quarter of 1990.
4. This is the first tape in the 'post-computerisation' series, referring to moves during the fourth quarter of 1990.

2.3 Reading the magnetic tapes

2.3.1 General notes on MOVES5

The magnetic tapes are presently read using the University of Leeds' Amdahl computer; however this facility will shortly become unavailable and it will be necessary for some other system to be used. Primary processing of the tapes is done using a FORTRAN program called MOVES5; this is based on and adapted from a previous program, MOVES4 (Boden, 1989). The original program has been expanded to take into account changes in the structure of records on the magnetic tapes (as reflected in Table 1) over time, and to allow alternative processing methods to be used. A set of control variables within the body of the program (imode, imttype and iver) are used to select alternative configurations, depending on both the tape being read and the type of output required. Three alternative settings are used to read the magnetic tapes. These are required due to changes in the numbers of records in each block, and structure of the data within each record. Table 3 summarises the differences in the settings of these variables.

2.3.2 Discussion of IMTTYPE setting.

The settings of IMTTYPE are most important for the successful running of the program; they have been defined through trial and error attempts to read the tapes. As one block of data contains 16384 characters, there is a logical limit in the number of records which may be contained in one block. The settings for record length and records per block reflect this. However there still appears to be a minor mismatch in the number of records read by the software and the number quoted in the documentation (the number of records read is typically a few thousand (out of 1,000,000 plus) less then the number quoted by OPCS).

In the case of the tape containing data on moves during the third quarter of 1990, a special group of settings have been included. This is because the tape could not be satisfactorily read using either of the other two methods. This tape (which represents the last of the 'pre-computerisation' series) was produced at the same time as the 1990 quarter 4 tape (the first 'post-computerisation' tape), in order to avoid the break in continuity implied by the removal of the three-month time-lag (see section 2.1.3). It appears that significantly less than the logical number of records are included in each block on the tape, however the software has been set to read all logical records in order to preclude the possibility of some data being missed. The result of this is that the count of logical records read reported by MOVES5 is much higher than the number of records quoted by OPCS; however the number of records processed (having discarded blank records) is roughly the same as the number of records quoted.

Table 3: Settings of MOVES5 environment variables

Variable, setting	Effect
IMODE	
1	Read in arrays specified in the MOVES5 EXEC program (If processing first quarter of a given period, read in blank arrays, otherwise read in outputs of previous run). These arrays are incremented during the running of the program, and form the output.
2	Don't bother reading in or incrementing the arrays; output individual records in a (semi)compressed format.
IVER	
1	If imode=1, then the output matrices will have the same shape and order as those produced by MOVES4 (i.e. 97 spatial areas plus totals)
2	If imode=1, output matrices will use the expanded zonal ordering system (i.e. 117 spatial areas plus 8 spatially non-specific categories).
IMTTYPE	
1	Used for pre-computerisation tapes: Assumes that a block contains 584 records which are 28 characters long, and that the required data runs from char. 9 to char. 36 of a record.
2	Used for post-computerisation tapes: Assumes that a block contains 201 records which are 61 characters long, and that the required data runs from char. 13 to char. 36 of a record.
3	Used for hybrid tape: Assumes that a block contains 511 records which are 61 characters long, and that the required data runs from char. 13 to char. 36 of a record.

2.3.3 Maintenance of MOVES5 software.

MOVES5 uses 3 definitional files whilst running which may require adaptation or replacement in the future in the event of further changes in the NHSCR data etc. These files are: *fhsacode 28193* (a numerically ordered list of FHSA etc. codes and labels, correct as of 28/1/93); *iprog lut* (a look-up-table containing indexing data for fhsacode 28193. These are the values that are used in conjunction with the variable IVER to govern the ordering (and size) of the output matrices); *moves agecodes* (used to form single year data into (5 year) age groups).

The software is currently installed on the University's Amdahl mainframe computer. This is due to be taken off-line in the summer of 1993, and thus some notes are now given concerning the transferral of the software to a different (probably UNIX based) platform. The program is written entirely in standard FORTRAN 77 code, and thus should transfer directly to a new system that has a suitable FORTRAN 77 compiler. However the following should be noted:

- CMS filenames are composed of two parts; using a space in UNIX filenames, however will lead to difficulties in working with these files. Standard practice is to use a dot to separate the stem and the suffix of a filename.
- Under CMS, a script file (MOVES5 EXEC) was required to set-up FORTRAN and run the program; this will not be needed, as most (if not all) UNIX FORTRAN compilers produce executable output files. However, the EXEC file also contains important file definition information. These should be replaced with OPEN commands in the program code; e.g., the EXEC line ' FI 2 DISK IPROG LUT ' should be replaced with the FORTRAN line (in MOVES5) ' OPEN (2,FILE='IPROG.LUT') '. This, and other OPEN statements should be placed after the initial parameterisation statements (e.g. CHARACTER..., INTEGER... etc. and before the first use of the external file).

- The most significant area of adaptation is likely to be in tape handling. It is probable that the MOVES5 line that reads a block of data from the magnetic tape will have to be replaced with one that is more suited to tape handling under the UNIX operating system. However, assuming that tapes can still be read in a broadly similar fashion, the rest of the program should work without alteration.

- Individual records are written out in a semi-compressed form (i.e. the relevant codes are written in ASCII form) because of inflexibility in the use of direct access files under CMS. It should be possible under UNIX to perform both parts of the code compression within MOVES5 are write out the fully compressed record directly (see section 2.4 for a more general discussion of the data compression technique employed).

2.4 Compression of individual records.

One of the 'new' features of MOVES5 is that it can produce output files that consist of compressed individual records. These files will be used in a future implementation of the TIMMIG program allowing far greater flexibility in tabulation. As mentioned above, there is an inflexibility with the use of direct access files under CMS. For this reason the CMS implementation of MOVES5 is designed to output semi-compressed individual records with the following characteristics:

i. Only those records which have fields which have been fully interpreted are used.

ii. Only those records that are created at the point of arrival are used. (In most cases two records are produced - one at departure from the old FHSA and one at arrival at the new FHSA).

iii The compressed records contain four numbers. In the semi-compressed format these take the form of (ASCII encoded) integers from 0-255; in the fully compressed format these are further compacted to four single ASCII values (thus taking four bytes in total).

The structure of the record is:

First byte / number: Values from 0-127 indicate destination; high-bit used to flag sex=female
Second byte / number: Values from 0-127 indicate origin; high-bit used to flag sex=unstated
Third byte / number: Age in years
Fourth byte / number: Year of birth

The record contains both age and year of birth in order to permit cohort-based analysis. Clearly both age and year of birth can be contained in values of not more than 127; this means that the high bit (value=128) is spare in both these two bytes and may be additionally used to store further information (direct access records under both CMS and UNIX must have a length specified in an integer number of bytes, and therefore all bits should ideally be exploited).

3. THE FUNCTIONALITY OF TIMMIG

3.1 Introduction.

3.1.1 General

TIMMIG is designed to allow the user to extract tables of migration data for between chosen areas over a chosen time span. The program is operated by means of a command file, a short plain text document which contains a series of instructions for the program to follow. Under the present version of the software, the command file must by written by the user; it is hoped however that future versions will include an interactive program to allow the construction of command files using a menu based system.

3.1.2 Query type

Two different levels of query will be possible in the full version of TIMMIG. The first type will allow the user to aggregate tables which have previously been constructed (the step (1) database), the second type will allow the user to construct unique tables based on individual records (the step (3) database). Clearly both types have inherent advantages and disadvantages. When the program is fully implemented, both types will also access the step (2) population database, although type 1 queries will clearly be limited to the level of age disaggregation present in the step (1) database.

Type 1 queries can be performed quickly, but are obviously constrained by the level of detail present in the tables they use as their source data. These tables contain interaction flows for FHSAs in England and Wales, and for Northern Ireland and the Isle of Man, and a bulked set of flows for Scotland. All flows are totals for 12 month mid-year to mid-year periods, from 1975 onwards. An additional table contains a breakdown of the total flow to and from each area by gender and age (aggregated into five-year age groups, with the top group being 75+). Type 2 queries take much longer to carry out, but offer a greater level of detail. As well as making data for the Scottish Area Health Boards available, a number of additional categories (such as to and from the Armed Forces, HM Prisons and abroad) can be tabulated. Furthermore, age is recorded by single year.

The two types are best seen as being complementary: initial extraction of data can be performed using a fast type 1 query, and then the output from this can be used to determine which parts of the database are to be examined using a more detailed type 2 query (when this query type can be made).

3.2 Table layout.

3.2.1 Terminology

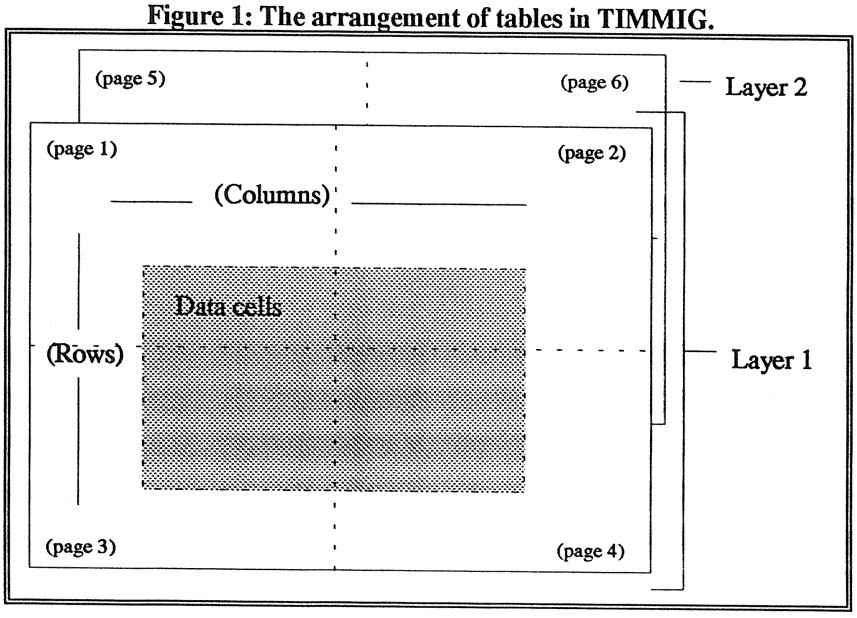
It should be noted that the discussion of the arrangement of TIMMIG tables uses the following definitions. A successful run of the TIMMIG program will result in a *table* being written to an output file. A table consists of *layers*, and each layer contains a number of *cells*, which are arranged into *rows* and *columns*, depending on the choices made by the user in the command file. The rows, columns and layers are termed the tables *dimensions*, and may be thought of as analogous to the x, y and z vertices of a cuboid.

Assuming that the table is to be printed out, it will be divided up into *pages*, which may contain all or part of a layer (but never parts of different layers). The portion that is printed on each page will have relevant row and column labels and dividing lines. This portion is referred to as a *sub-table*; the term should not be confused with the table as a conceptual whole.

3.2.2 Standard table layout

TIMMIG permits users to arrange tables in a flexible manner. Instructions must be given in the command file describing what variable(s) should be tabulated on each table dimension (i.e. row, column, layer). Figure 1 shows the conceptual relationship of the table dimensions by representing a typical table. In this example, the table has a sufficiently large number of rows and columns that a layer will not fit on one printed page. The page numbers shown in parentheses in the diagram illustrate the manner in which a large table will be distributed over several printed pages. In practice, row and column headings relevant to the portion of the layer will be included on all pages, not just the pages which represent the top and left hand side of the layer as is implied in the graphic.

Figure 1: The arrangement of tables in TIMMIG.



A further possibility of the arrangement of data on a page which can be noted occurs if a table has many columns but only a few rows. If this is the case then the program will print two or more sub-tables on a page, each one having its own row and column headings. For example, suppose a table was defined which had ten rows and twenty columns per layer and was printed on an A4 width printer (which will print six columns on a page). The first page in each layer would contain two sub-tables containing columns 1-6 and 7-12 of the layer, and the second page would contain two sub-tables, showing columns 13-18 and 19-20.

3.2.3 Layout of tables in export format

An important qualifier to the above description of table layout occurs if the user chooses the table to be printed in export format. This is a method of layout designed to facilitate the export of tables from TIMMIG to other packages. The format has been specifically designed with Microsoft Excel 4.0 in mind, but should be suitable for many other packages. The actual process of transferring data to Excel is discussed in Section 5.7. If the standard tabular format was read into a spreadsheet, complications would be introduced because of the vertical stacking of sub-tables.

In export format, no restriction is made on the width of output, so that when the file is read into a spreadsheet, it will be in a readily usable form. In effect, the program assumes an infinite page size (permitting any number of rows and columns per page). Most modern spreadsheet software can handle a large number of rows and columns. If problems are encountered then it is probably best to run the program several times, with appropriately altered settings, in order to produce a series of moderately sized tables rather than one very large one.

3.3 Arrangement of variables on table dimensions.

3.3.1 General

There are five variables which can be put onto any of the three table dimensions. All three dimensions must have a variable. The variables are: origin, destination, age, gender and time. These can be freely mixed (depending on the query type - see paragraph 3.3.4), but obviously can only be included once per table.

If a variable is not included on any dimension, then the tabulated cell will be totalled for that variable. For example, if the variables origin, gender and time were specified for row, column and layer respectively, then the data in each cell would refer to the flow from a given origin (depending on the row) of persons of a given gender (depending on the column) during a given time period (depending on the layer), to all destinations and for all age groups (because neither of these variables formed a dimension).

3.3.2 Collapsing two variables into a dimension

If the only arrangement of tables was as defined above, then clearly only three out of the five variables could ever be seen in at a table at one time, a situation which is not ideal. However it is possible to display two variables on one dimension. This process is called collapsing a dimension, because it allows the user to tabulate data for more categories than would otherwise be possible - a fourth dimension has effectively been 'collapsed' into the table. In the command file, either one or two variables can be chosen for each table dimension. In the output table, data is shown for all conditions of the second variable for each case of the first variable. For example, suppose a table was defined with 'origin' as the first row variable, and 'gender' as the second row variable. The resulting table would have a series of rows labelled:

$origin_1$: Male	[col_1]	...	[col_n]
$origin_1$: Female	[col_1]	...	[col_n]
...	...	...	...	...
$origin_n$: Male	[col_1]	...	[col_n]
$origin_n$: Female	[col_1]	...	[col_n]

As the program permits two variables to be assigned to any dimension, it is obviously possible to display all five variables in one table.

3.3.3 Totalling a variable

It is possible to include the total of a chosen variable as a dimension. This may be required in order to avoid large outputs. This is done by preceding the variable name in the command file by an ampersand character (e.g. row1=&origin). Example 2 in section 4 illustrates this.

3.3.4 Restrictions on the choice of variables

As explained above, it is possible to display all possible views of the database (i.e. all the categories can be displayed, in a user-defined arrangement). However, using type 1 queries, some combinations of variables are deemed to be invalid. There are essentially two types of table which TIMMIG produces under a type 1 query: interaction matrices of flows between different areas, and tables breaking down the flows from given areas by age and sex. The program will not permit tables of the form origin by destination by age / sex to be specified. This is because type 1 queries extract data from existing tables, and the age and sex data in these tables are in the form of national (UK) totals for each FHSA only; data is not contained regarding flows between specific FHSAs. If the user wishes to analyse the demographic structure of a flow between two specific areas, then a suitable table must be extracted using a type 2 query.

3.3.5 The use of variables as filters

As well as describing how a table will be disaggregated, the definition of variables also serves to create data filters; for example, if the origin is defined as Leeds and the destination as Nottingham, then the only data that will be tabulated refers to these regions. All variables will theoretically function as filters as well as table dimensions - for example if gender is restricted to males only, then only data about male migration will be tabulated, regardless of the table dimensions. However, this functionality is compromised in the initial implementation of TIMMIG. This version draws on previously aggregated tables (the step (1) database) which are limited in detail. It is important to remember the relationship between variables, and whether they serve as dimensions or filters in this version of TIMMIG.

There are essentially two types of tables which can be constructed - ones with the form [origin by destination] on the one hand, and ones with the form [{origin *or* destination} by {age and / or gender}] on the other. In the former type of table, specific origin and destination zones may be tabulated, but the age and gender settings do not apply. Whatever the definitions of age and gender for this table type, all persons are counted. In the latter type of table however, age and gender may be disaggregated, (and thus used as filter and/or dimension), but there is limited spatial disaggregation. One (only) of the spatial variables must be chosen as a table dimension. There is no difference between origin and destination areas in this case; the category is merely used as a list of zones that data is to be tabulated for. The output will show the interactions between the specific zones (as defined by the list which is chosen as a table dimension) and the whole of the rest of the country. The direction of flow of persons for which data will be extracted in this type of table is determined not by the choice of either 'origin' or 'destination' as a table dimension, but by the choice of 'net', 'in', or 'out migration' as the table statistic.

3.4 How to choose the parameters for analysis.

3.4.1 General

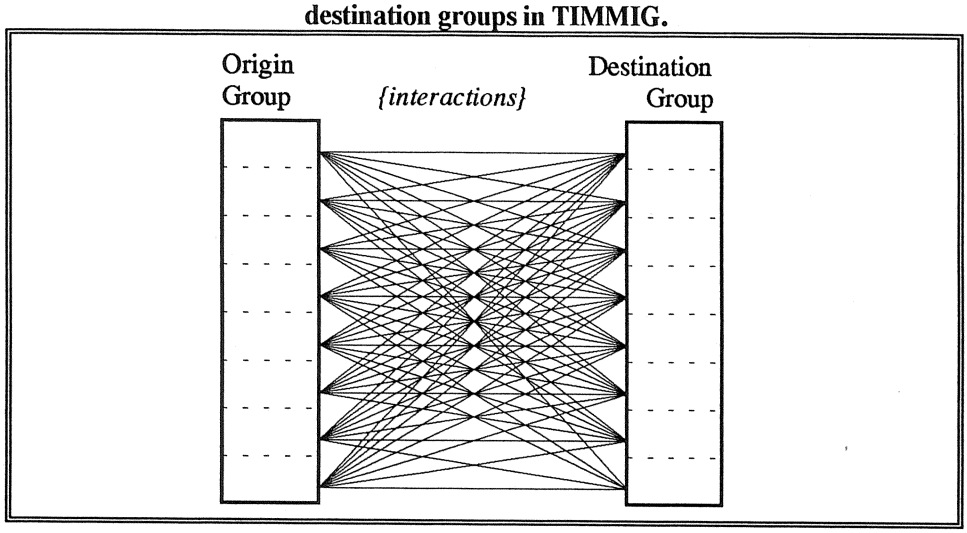
Because there are a wide range of options provided by TIMMIG it is important to decide before writing a command file what data you wish to extract. The most important factors to consider are the composition of the origin and destination groups of areas to be chosen, and the structure of the table that will be created.

3.4.2 Choosing the origin and destination areas

It is important to consider the level of detail that you require in the output. If you ask for too much detail in the output then it will be difficult to find the data that you are actually interested in, or to identify trends that may be of interest; on the other hand if you do not include sufficient detail you will have to run the program again to get a more appropriate set of results. The diagram below illustrates

the nature of the relationship between the origin and destination groups. Clearly, the more zones that are included in either group, the larger the output tables will be.

Figure 2 A diagrammatic representation of the relationship between the origin and the destination groups in TIMMIG.

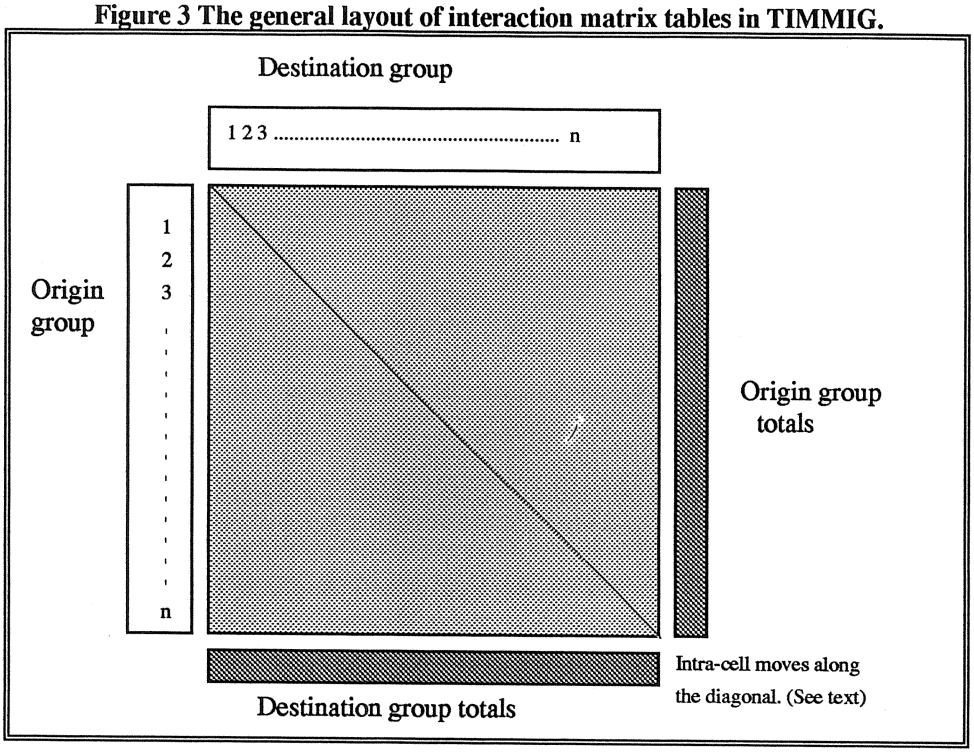


As figure 2 implies, interactions between areas within either of the groups are not specifically tabulated. If the user wishes to discover such interactions, then it would be necessary to define both the origin and the destination groups as including the areas of interest. For example, if the districts of West Yorkshire were defined as the origin group, and the districts of Greater Manchester as the destination zone, then the resulting tables (if printed) would contain data regarding the flows between districts in either group, but would not contain data about flows between districts within a group. If this information was required as well, it would be necessary to define both groups so as to include the districts of West Yorkshire and Greater Manchester. If both groups are defined as including exactly the same areas, then it is worth noting that the output tables may include a mirror-image of themselves. Figure 3 illustrates the general layout of such tables in TIMMIG output, by showing how a table which includes both origin and destination as table dimensions will contain a line of symmetry. It should be remembered, however, that as created tables are multidimensional, the line of symmetry will not always be so clear.

In order to optimise the amount of data that is extracted, analysis should ideally be performed at the coarsest geographical scale that includes the required data; i.e., using the examples mentioned above, if it was decided that the only data that was actually required was that concerning movements between West Yorkshire as a whole and Greater Manchester as a whole, then the data extraction should be made at county level, in order to avoid tabulating all the districts of the two areas. The same relationships between the two groups will exist at this (and all) scales: defining the county of West Yorkshire as one group and Greater Manchester as the other will produce output showing the total

moves between the two counties. If, however the two counties are included in the definition of both groups then the output will also show total moves between the districts within the county.

Figure 3 The general layout of interaction matrix tables in TIMMIG.



It should be remembered that all figures are summed from FHSA interactions, and that intra-FHSA moves are not recorded. Hence if an output table contains the same FHSAs in both groups then the figure for the moves within an FHSA (e.g. Leeds to Leeds) will be shown as zero, despite the fact that there are obviously many such moves (in fact, this should be the largest single flow from Leeds). A consequence of this is that all moves within a larger area (e.g. West Yorkshire to West Yorkshire) will be significantly under-represented. In the case of West Yorkshire, the internal flow would be the sums of flows between Leeds and all other districts, Bradford and all other districts etc.). It may be the case that the user decides that these flows are not wanted in the output, because they will adversely affect totals. For this reason an option is included in the program to ignore these 'intra-cell' moves.

3.4.3 Structuring a table to output

When the user has chosen which areas are to be included in his / her analysis, it is then necessary to consider how to structure the table - that is, which variables to assign to each table dimension. There are two important considerations in this: firstly, how easily tables can be read by the user, and secondly, how much space they will take up (in terms of both computer disk space and paper on which the table will be printed). Ideally, the user should plan tables that will contain no more detail than is required, and will maximise the efficiency of how it fits on to a sheet of paper.

4. EXAMPLE APPLICATIONS

4.1 General

This section contains samples of TIMMIG runs. A description of the program operation parameters is included for each example. The example command files can be copied using an option on the initial TIMMIG menu.

4.2 Discussion of examples

Example 1 illustrates a simple aggregation of out migration data by origin age and time; this example also shows the output generated on screen while TIMMIG runs. Example 2 shows data for out migrations from the districts of West Yorkshire and South Yorkshire. Note that both examples 1 and 2 shows migrations from specific zones to the rest of the country; but that this is achieved in slightly different ways, despite the fact that both command files contain the line 'destination=all;'. In example 1, a table of origins by age is generated. This information is always at the national level (that is, the data is for the interactions of given areas with the whole of the rest of the country). Age and gender disaggregated data are not available in the step (1) database for flows between specific areas. Thus, in example in, the line 'destination=all;' is effectively ignored (it would have made no difference if it had been defined as a set of given areas). In example 2, however, the table generated is of the form origin by destination. This data is available for specific areas,, and therefore, in order to gain the national interaction, it is necessary to define destination as 'all'. Example 2 also shows the use of the ampersand prefix to produce the total of a given variable, in the line 'layer1=&destination'. Had the ampersand not been included on this line, then the output would have contained 98 layers - one for each destination, and a layer of totals.

Examples 1 and 2 further illustrate a difference between the two table types (origin or destination by age and/or gender in the first case, and origin by destination in the second). The difference is due to the construction of the step (1) tables that type 1 queries access. If the 'Totals' line at the bottom of the table shown in example 1(c) is compared with the line for Leeds in the table that forms example 2(b), it can be seen that the two sets of values, which ostensibly show the same data have some small differences, namely that for each year, the values in example 1(c) are slightly lower than the values in example 2(b). This is because example 1(c) is drawn from tables that specifically include only 'male' and 'female' categories, whereas the tables from which 2(b) is compiled include additional data where the gender variable was not stated. It should be noted that, for type 1 queries, all tables that show data disggregated by age or gender will not include these additional data.

Example 3 illustrates the process of collapsing two variables into one table dimension, using a similar set of data to the first two examples. This example additionally features a totalled variable, (&age). The output of example 3 shows the way in which two variables share the same dimension - all

cases of variable2 are tabulated for each case of variable1. The fourth example, shows how data can be written out in an export format that is suitable for use with Microsoft Excel. A bar chart of the data is also illustrated. This example also shows the use of the special census year data set. This set uses a April 1 to March 31 concatentation of quarterly data, in order to mimic the year prior to the Census (in April 1991) as closely as possible.

The final example, number 5, shows the use of two further features. The simpler of these is the command 'intracell=off'. This stops the program from counting moves between FHSAs which are in the same zone. Thus, in this example moves between zones within the 'Rest of England' and 'Wales' groups will not be counted. The second feature illustrated by example 5 is the use of a user defined spatial aggregation. This is indicated by the command '*scale=*userdef1.agg*'. The file *userdef1.agg* is an alternative aggregation of FHSAs. The asterisk in the scale command is used to decalare a user defined file. The file must be in the same directory as the TIMMIG script. The required file will be copied at the same time as the command file *example5.cmd*, if this is copied using the option on the initial menu.

Example 1

1(a) Command file

```
start

author=A Geographer
title=An example command file for TIMMIG No.1
printer=narrow
filename=example1.out
output=table

# Temporal parameters...
tstart=1985
quarters=4
tperiods=5

# Spatial parameters...
scale=fhsa
origin=20;
destination=all;

# Arrangement of table...
row1=age
column1=time
layer1=origin
statistic=out migration

# Aggregation of age and sex...
age=fiveyear
gender=mf

end
```

1(b) Sample run-time messages

```
Found: example1.cmd
START FOUND
END FOUND
Making validity checks...
All categories processed OK.
Copying chosen files...
Determining array limits...
Preparing unique version of tabulation program...
Running program...
Counting...
Processing year:   1985
Processing year:   1986
Processing year:   1987
Processing year:   1988
Processing year:   1989
Exiting...
--TIMMIG2: Run log----------------------
Started at:
Tue May 18 09:41:03 BST 1993

Finished at:
Tue May 18 09:41:38 BST 1993
---------------------------------------
```

1(c) Output file

```
*******************************************************************************

 #####  ###  ##    ##  ##    ##  ###   ##
   #      #   # # # #  # # # #   #    #
   #      #   #  #  #  #  #  #   #    #
   #      #   #     #  #     #   #    # ##
   #    ###  #     #  #     #  ###    ##


*******************************************************************************

 Layout of pages in layer 1:
 [Output page no.]

       Columns ->
 Rows
  V    [  1 ]

-------------------------------------------------------------------------------

 Table shows:
 Age by Time by Origins
 Statistic:
 Out migration

-------------------------------------------------------------------------------

===============================================================================
Out migrations from Leeds mid 85 - 90
Oliver Duke-Williams
User:geo6odw       Date:Tue May 18 09:41:03 BST 1993         Page:  1
===============================================================================

===============================================================================
Origins                    | Time
Leeds                      |-----------------------------------------------------
___________________________|85q3-86q:86q3-87q:87q3-88q:88q3-89q:89q3-90q:Totals  :
Age                        |        :        :        :        :        :        :
-------------------------------------------------------------------------------
  0 -  4                   :     1361     1283     1553     1446     1354     6997
  5 -  9                   :     1143     1188     1260     1194     1076     5861
 10 - 14                   :      898      929      901      914      758     4400
 15 - 19                   :     1827     1877     1863     1799     1713     9079
 20 - 24                   :     5240     5445     5852     5855     5631    28023
 25 - 29                   :     3481     3626     3974     4451     4101    19633
 30 - 34                   :     1972     2118     2355     2429     2100    10974
 35 - 39                   :     1497     1476     1531     1574     1359     7437
 40 - 44                   :      901     1113     1115     1192     1035     5356
 45 - 49                   :      638      708      737      755      668     3506
 50 - 54                   :      537      592      631      647      566     2973
 55 - 59                   :      491      539      551      521      381     2483
 60 - 64                   :      456      509      547      501      343     2356
 65 - 69                   :      403      446      469      457      337     2112
 70 - 74                   :      271      256      273      249      246     1295
 75+                       :      687      787      891      891      749     4005
-------------------------------------------------------------------------------
Totals                     :    21803    22892    24503    24875    22417   116490
===============================================================================
```

Example 2

2(a) Command file

```
start

author=A Geographer
title=An example command file for TIMMIG No.2
printer=narrow
filename=example2.out
output=table

# Temporal parameters...
tstart=1985
quarters=4
tperiods=5

# Spatial parameters...
scale=fhsa
origin=13-21;
destination=all;

# Arrangement of table...
row1=origin
column1=time
layer1=&destination
statistic=out migration

# Aggregation of age and sex...
age=fiveyear
gender=mf

end
```

Notes:
i) The dimension layer1 is '&destination'. This causes only one layer to be calculated, which shows the totals (by origin and time, the other dimensions) for all destinations.
ii) Age and gender are not included as dimensions. By default therefore, table cells will be totalled for age and gender. Under a type 2 query, age and gender would be used as filters in such a case.

2(b) Output file

```
*******************************************************************************

 #####  ###  ##    ##  ##    ##  ###   ##
   #     #   # # # #  # # # #   #    #
   #     #   #  #  #  #  #  #   #    #
   #     #   #     #  #     #   #    # ##
   #    ###  #     #  #     #  ###    ##

*******************************************************************************

 Layout of pages in layer 1:
 [Output page no.]

       Columns ->
 Rows
  V    [  1 ]

-------------------------------------------------------------------------------

 Table shows:
 Origins by Time by Tot. Destinations
 Statistic:
 Out migration


-------------------------------------------------------------------------------

===============================================================================
Out migrations from Leeds mid 85 - 90
Oliver Duke-Williams
User:geo6odw       Date:Mon May 17 14:40:10 BST 1993         Page:  1
===============================================================================

===============================================================================
Tot. Destinations        | Time
Totals                   |-----------------------------------------------------
_________________________|85q3-86q:86q3-87q:87q3-88q:88q3-89q:89q3-90q:Totals  :
Origins                  |        :        :        :        :        :        :
-------------------------------------------------------------------------------
Barnsley                 :     4896     4988     5282     5165     4871    25202
Doncaster                :     7514     7672     8241     8137     7285    38849
Rotherham                :     6499     6487     6612     6707     6499    32804
Sheffield                :    16295    16457    17575    17929    15992    84248
Bradford                 :    13967    14318    14657    14989    14202    72133
Calderdale               :     5313     5315     5986     6164     5597    28375
Kirklees                 :    11098    11224    11924    11522    10787    56555
Leeds                    :    22035    23104    24579    24968    22442   117128
Wakefield                :     8630     8285     8778     8707     7688    42088
-------------------------------------------------------------------------------
Totals                   :    96247    97850   103634   104288    95363   497382
===============================================================================
```

Example 3

3(a) Command file

```
start

author=A Geographer
title=An example command file for TIMMIG No.3
printer=narrow
filename=example3.out
output=table

# Temporal parameters...
tstart=1988
quarters=4
tperiods=4

# Spatial parameters...
scale=fhsa
origin=17-21;
destination=all;

# Arrangement of table...
row1=time
row2=gender
column1=origin
layer1=&age
statistic=out migration

# Aggregation of age and sex...
age=fiveyear
gender=mf

end
```

Notes:
i) This example illustrates the process of collapsing two variables into a dimension. The row dimension will include both time and gender.
ii) The layer variable contains age as a totalled variable. This is because all the primary dimensions must be defined. As gender and origin have been included, and as this is a type 1 query, age is the only remaining variable. It is totalled, by using the ampersand prefix, in order to minimise the output.

3(b) Output file

```
*******************************************************************************

 #####  ###  ##    ##  ##    ##  ###    ##
   #      #   # #  # #  # #  # #   #    #
   #      #   #  #  #  #  #  #   #    #
   #      #   #     #  #     #   #    # ##
   #     ###  #     #  #     #  ###    ##


*******************************************************************************

 Layout of pages in layer 1:
 [Output page no.]

       Columns ->
 Rows
  V    [  1 ]

-------------------------------------------------------------------------------

 Table shows:
 Time        by Gender by Origins by Tot. Age
 Statistic:
 Out migration                                                                 j


-------------------------------------------------------------------------------

===============================================================================
Out migrations from Leeds mid 88 - 92
Oliver Duke-Williams
User:geo6odw        Date:Mon May 17 15:25:58 BST 1993          Page:  1
===============================================================================

===============================================================================
Tot. Age                  | Origins
Totals                    |-----------------------------------------------------
                          |Bradford:Clderdal:Kirklees:Leeds    :Wakefiel:Totals  :
Time        by Gender     |        :        :        :         :        :        :
-------------------------------------------------------------------------------
88q3-89q2   :Male         :      7221     3059     5651     11877     4102    31910
88q3-89q2   :Female       :      7627     3092     5839     12998     4575    34131
89q3-90q2   :Male         :      6831     2710     5098     10531     3621    28791
89q3-90q2   :Female       :      7286     2878     5668     11886     4058    31776
90q3-91q2   :Male         :      5906     2518     4531     10041     3417    26413
90q3-91q2   :Female       :      6181     2631     4928     11248     3796    28784
91q3-92q2   :Male         :      6710     2649     5272     11296     3813    29740
91q3-92q2   :Female       :      7017     2867     5682     12390     4294    32250
-------------------------------------------------------------------------------
Totals                    :     54779    22404    42669     92267    31676   243795
===============================================================================
```

Example 4

4(a) Command file

```
start

author=A Geographer
title=An example command file for TIMMIG No.4
printer=narrow
filename=example4.out
output=excel

# Temporal parameters...
tstart=census
quarters=4
tperiods=5

# Spatial parameters...
scale=fhsa
origin=20;
destination=all;

# Arrangement of table...
row1=age
column1=origin
column2=gender
layer1=time
statistic=net migration

# Aggregation of age and sex...
age=fiveyear
gender=mf

end
```

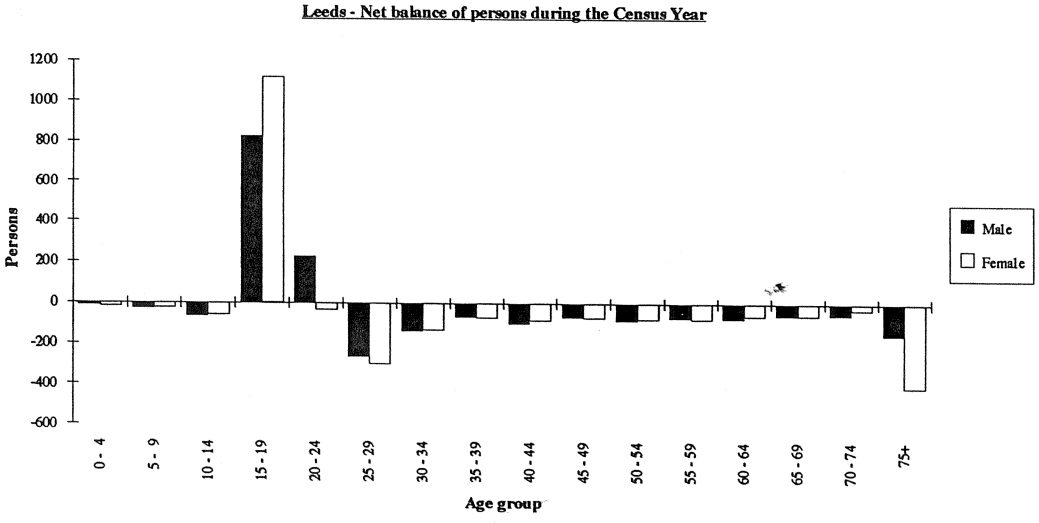
Notes:
i) This file has been designed to illustrate the Excel / export format. One effect of this is that the author and title text is not used.
ii) The file specifies Census as the time period. The setting for tperiods will be ignored.
iii) Note that there are two column variables.

4(b) Output file

```
Time;Census;

Age;Leeds:Males  ; Leeds:Females  ;Leeds:Totals  ;

  0 -  4;  -14;  -18;  -32;
  5 -  9;  -29;  -27;  -56;
 10 - 14;  -67;  -60;  -127;
 15 - 19;  825;  1120;  1945;
 20 - 24;  225;  -34;  191;
 25 - 29;  -267;  -302;  -569;
 30 - 34;  -138;  -134;  -272;
 35 - 39;  -69;  -72;  -141;
 40 - 44;  -103;  -87;  -190;
 45 - 49;  -71;  -73;  -144;
 50 - 54;  -86;  -79;  -165;
 55 - 59;  -72;  -80;  -152;
 60 - 64;  -77;  -65;  -142;
 65 - 69;  -61;  -62;  -123;
 70 - 74;  -58;  -33;  -91;
 75+;  -157;  -416;  -573;
Totals;  -219;  -422;  -641;
```

Notes:
This output is formatted for import to MS-Excel, not for analysis as text.

4(c) Typical graphical output from Excel



Example 5

5(a) Command file

```
start

author=A Geographer
title=An example command file for TIMMIG No.5
printer=narrow
filename=example5.out
output=table

# Temporal parameters...
tstart=1980
quarters=4
tperiods=10

# Spatial parameters...
scale=*userdef1.agg
origin=all;
destination=all;

# Arrangement of table...
row1=origin
column1=time
layer1=&destination
statistic=net migration
intracell=off

# Aggregation of age and sex...
age=fiveyear
gender=mf

end
```

Notes:
This example introduces two new features; the use of a user defined spatial aggregation, and the 'intracell=off' command. Note that the file 'userdef1.agg' is copied at the same time as the example command files if they are copied from the TIMMIG starting menu.

5(b) Output file

```
*******************************************************************************

 #####  ###  ##    ##  ##    ##  ###   ##
   #      #   # # # #  # # # #   #    #
   #      #   #  #  #  #  #  #   #    #
   #      #   #     #  #     #   #    # ##
   #     ###  #     #  #     #  ###    ##

*******************************************************************************
 Layout of pages in layer 1:
 [Output page no.]
       Columns ->
 Rows
  V    [  1 ]
-------------------------------------------------------------------------------
 Table shows:
 Origins by Time by Tot. Destinations
 Statistic:
 Net migration

===============================================================================
An example command file for TIMMIG
A Geographer
User:geo6odw        Date:Tue May 25 14:39:39 BST 1993          Page:  1
===============================================================================
===============================================================================
Tot. Destinations        | Time
Totals                   |-----------------------------------------------------
_________________________|80q3-81q:81q3-82q:82q3-83q:83q3-84q:84q3-85q:85q3-86q:
Origins                  |       :       :       :       :       :       :
-------------------------------------------------------------------------------
Barnsley                 :     -450     -660    -1010    -1044    -1068    -1437
Doncaster                :       90    -1320     -520    -1776     -986     -825
Rotherham                :      490     -580     -650    -1282     -828    -1057
Sheffield                :     -710     -280    -1090    -2163    -2109    -3058
Bradford                 :    -2210     -950    -1650    -1501    -2881    -2154
Calderdale               :     -110     -340      -40     -482      336      369
Kirklees                 :     -930     -730    -1100     -594    -1479     -869
Leeds                    :    -2520    -2300     -970    -2623    -1903    -2475
Wakefield                :       30     -810    -1200    -1319     -960    -1912
Humberside               :    -1740     -780     -710    -1329    -1184     -586
North Yorkshire          :     2570     3550     3640     5957     5294     4129
Rest of England          :     9210     4540    13000    24430    16705    23847
Scotland                 :    -2810     -310    -6870   -14184   -11354   -15685
Wales                    :     1540     3450     1930     1180     4877     4969
Northern Ireland         :    -3010    -2520    -2590    -3530    -2644    -3677
Isle of Man              :      560       40     -170      260      184      421
-------------------------------------------------------------------------------
Totals                   :        0        0        0        0        0        0
===============================================================================
===============================================================================
Tot. Destinations        | Time
Totals                   |-----------------------------------------------------
_________________________|86q3-87q:87q3-88q:88q3-89q:89q3-90q:Totals  :
Origins                  |       :       :       :       :       :
-------------------------------------------------------------------------------
Barnsley                 :    -1043     -688      780      -13    -6633
Doncaster                :    -1376      135      371      226    -5981
Rotherham                :    -1108     -580     1014       96    -4485
Sheffield                :    -2075    -3785    -1900     -819   -17989
Bradford                 :    -3030     -562      304    -1363   -15997
Calderdale               :      154       -8      521      113      513
Kirklees                 :    -2072     -905       22     -819    -9476
Leeds                    :    -4041    -1152    -2044    -2137   -22165
Wakefield                :     -119       38     1108      360    -4784
Humberside               :     -812     3068     3667     1854     1448
North Yorkshire          :     7388     6593     6588     3458    49167
Rest of England          :    20618     5034    -9588   -11273    96523
Scotland                 :   -16092   -19548   -10854     7757   -89950
Wales                    :     8993    17528    15991     4858    65316
Northern Ireland         :    -6438    -6672    -7113    -3712   -41906
Isle of Man              :     1053     1504     1133     1414     6399
-------------------------------------------------------------------------------
Totals                   :        0        0        0        0        0
===============================================================================
```

Note that one layer is fitted onto one page in this example.

5. ACCESSING AND OPERATING TIMMIG

5.1. Access to TIMMIG

5.1.1 General

The program, database and other files used in TIMMIG are held on a common access library facility on the GPS computer in Leeds. A user name on the GPS is therefore required to be able to use the program. The program is run by means of a UNIX script file - a program containing a series of consecutive UNIX commands - which extracts the program and data into a directory and then runs the program. When the program has finished, all temporary files will be removed. During the running of the program, a number of files are created in the users' normal file space, which are automatically deleted afterwards. These are used to pass data between different sub-programs. The names of these temporary files are given below. In order to avoid naming conflicts user should make sure that a) they do not have existing files with these names in the directory from which TIMMIG will be run and b) that output files do not take any of these names. The simplest way to ensure that condition a) is met is of course to create a new directory that will only be used for running TIMMIG jobs. It can be seen from Table 4 that several of the files are preceded by dots; this has the effect of making them invisible if a normal list of the directory is made (i.e. by using the UNIX command ls). In order to see such files, the option -a has to be appended to the ls command, forcing it to list all files.

Table 4: Prohibited filenames in the TIMMIG directory.

.controls	.dat	.makegen	.orderbook	.params	.pid
.tmgver	.who	.workfiles	glabels.o	gcountv1.o	gcountv2.o
gentab.o	goutput.o	gread.o	gtab		

5.1.2 Setting up a directory and copying the script file

The users first task should be to set up a directory (or move to an existing one) from which they wish to work. Output files will be written to this directory.

The command **cd** <*directory name*> is used to change directories.
The command **mkdir** <*directory name*> will create a new directory.

i. Use these commands to create and enter a suitable directory. N.B: The command **cd** (on its own) will return you to your home directory (the directory that you are in when you log in), and the command **pwd** (present working directory) will remind you of the directory that you are currently in.

ii. Enter the command **cp ~geo6lib/oliverdw/Timmig2 Timmig2**

This will copy the UNIX script file *Timmig2* into your current directory. Note that the copy command, cp, uses the syntax **cp *filename1 filename2*** in this instance. In the above example, Timmig2 has been used as filename2 (the name that the copy of the file will be given) - however any name could be given.

5.1.3 Running the program

Once you have copied the script file, the program can be run be entering the command **Timmig2** - or the name that you have given the program. As with all UNIX commands, the command is case-significant.

5.1.4 Running TIMMIG locally

TIMMIG has to extract a number of data files whilst it works. Under normal conditions, it does this by copying them to the /tmp directory. From time to time this may result in an error, because there is not enough space on the /tmp disk. This will cause the program to crash. The program can thus be run with an option instructing the program to copy files to the local directory. This is done by including the option -l on the command line i.e. by entering the command **Timmig2 -l**. The program will delete the copied files after it has finished running; however it will take up approximately 1 megabyte of disk space whilst it runs. The amount of available space the user has can be checked by issuing the command **qu -v**. This is measured in kilobytes - thus there must be at least 1000k free space as declared by qu -v in order to be able to run the program locally.

5.2: Running the program

5.2.1 General

The program is currently operated by using a command file - a text file which contains a series of instructions. An interactive interface which allows users to build a command file through a series of menus should be available shortly. A help file is kept on the geo6lib disk, which can be copied and printed out. The file contains a list of ID codes for different spatial scales, together with a copy of the command definitions on the following pages. There are two versions of this file - one is in Windows WRITE format, the other is in Postscript format. The file(s) can be copied using the command(s):

cp ~geo6lib/oliverdw/general/miniman.wri miniman.wri [For the WRITE version]
cp ~geo6lib/oliverdw/general/miniman.ps miniman.ps [For the Postscript version]

This file can then be printed out to use as a set of reference sheets. The Postscript version can be printed directly using a Postscript compatible printer. The WRITE version should be transferred to

a PC which is capable of running MS-Windows. The file can then be printed using the standard Windows printing procedures. See section 5.6 for help with transferring a file to a PC. **N.B.** The Write copy of the mini manual is in a binary form. It is necessary to transfer it as such. This can be done either by typing **binary** at the ftp prompt before retrieving the file, or by using the command **bget miniman.wri**. [bget=binary_get].

5.2.2 Running the program

When the program is started, the user will be prompted with a menu as shown in Figure 4. Currently this supports only 1 major option - running a prepared command file - put should be expanded to include more options in the future.

Figure 4: The initial menu in TIMMIG

```
+================================================+
|                                                |
|   T I M M I G - Version 2.0(Rev B) - May 1993  |
|------------------------------------------------|
|                                                |
| NOTES:                                         |
|   This version enabled for type 1 queries only!|
|                                                |
|__MAIN MENU_____________________________________|
| 1 - Run a prepared command file                |
| :                                              |
| 8 - List current directory                     |
| 9 - Quit                                       |
|                                                |
|__Copy documents to current directory___________|
| 10 - PostScript version of reference pages     |
| 11 - MSWindows Write version of reference pages|
| 12 - Example command files                     |
| 13 - User definable template files             |
|                                                |
+================================================+

Please enter your choice:
```

Users should enter the number of the required option. On selecting option one, the program will prompt the user to enter the name of the command file to be run. If the filename supplied can not be found, then an error message will be returned, and the program will return to the main menu. The 'quit' option can be used at this stage to leave the program. Once a valid command file has been found, the program runs automatically. It can be terminated at any stage by pressing CTRL-C on the keyboard. The program intercepts this stop signal, and attempts to 'clean up' its run, by erasing temporary files. However, it is possible that either some files will be missed by this process, or that some error messages might be produced. If CTRL-C is used to stop the program running, than the user is advised to use the command '**ls -a**' to list the files in the current directory and then use the **rm** command to delete any files which have been missed (see Table 4).

5.3 Construction of command files

5.3.1 Naming command files

The command file can be given any name that is valid in UNIX, but must be less then twenty letters long.

5.3.2 Structure of command files

Command files can contain a mixture of instructions, blank lines and comment lines. The file should be created as a blank ASCII text file. This can be done using simple text editors such as **emacs** or **vi** on a UNIX system, or the MS-DOS **edit** or MS-Windows **notepad** programs on a PC. It should be noted that the first thing the program does is to convert the command file to lower case characters, thus any upper case characters used will not appear as upper case in the output file. Blank lines and comment lines may be placed anywhere in the command file. Comment lines start with the character '#', the user can place any required test after this character, in order to comment on what the command file is going to do. Instructions take the form '*command=instruction*', for example 'origin=all'. The first instruction is a command file must be the command 'start', and the last instruction must be 'end'. All other instruction lines must be placed between these two commands.

5.3.3. Command file instructions

The following pages list the range of command that are presently available in TIMMIG, and describe the range and effect of different settings.

5. ACCESSING AND OPERATING TIMMIG

5.1. Access to TIMMIG

5.1.1 General

The program, database and other files used in TIMMIG are held on a common access library facility on the GPS computer in Leeds. A user name on the GPS is therefore required to be able to use the program. The program is run by means of a UNIX script file - a program containing a series of consecutive UNIX commands - which extracts the program and data into a directory and then runs the program. When the program has finished, all temporary files will be removed. During the running of the program, a number of files are created in the users' normal file space, which are automatically deleted afterwards. These are used to pass data between different sub-programs. The names of these temporary files are given below. In order to avoid naming conflicts user should make sure that a) they do not have existing files with these names in the directory from which TIMMIG will be run and b) that output files do not take any of these names. The simplest way to ensure that condition a) is met is of course to create a new directory that will only be used for running TIMMIG jobs. It can be seen from Table 4 that several of the files are preceded by dots; this has the effect of making them invisible if a normal list of the directory is made (i.e. by using the UNIX command **ls**). In order to see such files, the option **-a** has to be appended to the ls command, forcing it to list all files.

Table 4: Prohibited filenames in the TIMMIG directory.

.controls	.dat	.makegen	.orderbook	.params	.pid
.tmgver	.who	.workfiles	glabels.o	gcountv1.o	gcountv2.o
gentab.o	goutput.o	gread.o	gtab		

5.1.2 Setting up a directory and copying the script file

The users first task should be to set up a directory (or move to an existing one) from which they wish to work. Output files will be written to this directory.

The command **cd** *<directory name>* is used to change directories.
The command **mkdir** *<directory name>* will create a new directory.

i. Use these commands to create and enter a suitable directory. N.B: The command **cd** (on its own) will return you to your home directory (the directory that you are in when you log in), and the command **pwd** (present working directory) will remind you of the directory that you are currently in.

ii. Enter the command **cp ~geo6lib/oliverdw/Timmig timmig**

This will copy the UNIX script file *Timmig* into your current directory. Note that the copy command, cp, uses the syntax **cp *filename1 filename2*** in this instance. In the above example, timmig has been used as filename2 (the name that the copy of the file will be given) - however any name could be given.

5.1.3 Running the program

Once you have copied the script file, the program can be run be entering the command **timmig** - or the name that you have given the program. As with all UNIX commands, the command is case-significant.

5.1.4 Running TIMMIG locally

TIMMIG has to extract a number of data files whilst it works. Under normal conditions, it does this by copying them to the /tmp directory. From time to time this may result in an error, because there is not enough space on the /tmp disk. This will cause the program to crash. The program can thus be run with an option instructing the program to copy files to the local directory. This is done by including the option -l on the command line i.e. by entering the command **timmig -l**. The program will delete the copied files after it has finished running; however it will take up approximately 1 megabyte of disk space whilst it runs. The amount of available space the user has can be checked by issuing the command **qu -v**. This is measured in kilobytes - thus there must be at least 1000k free space as declared by qu -v in order to be able to run the program locally.

5.2: Running the program

5.2.1 General

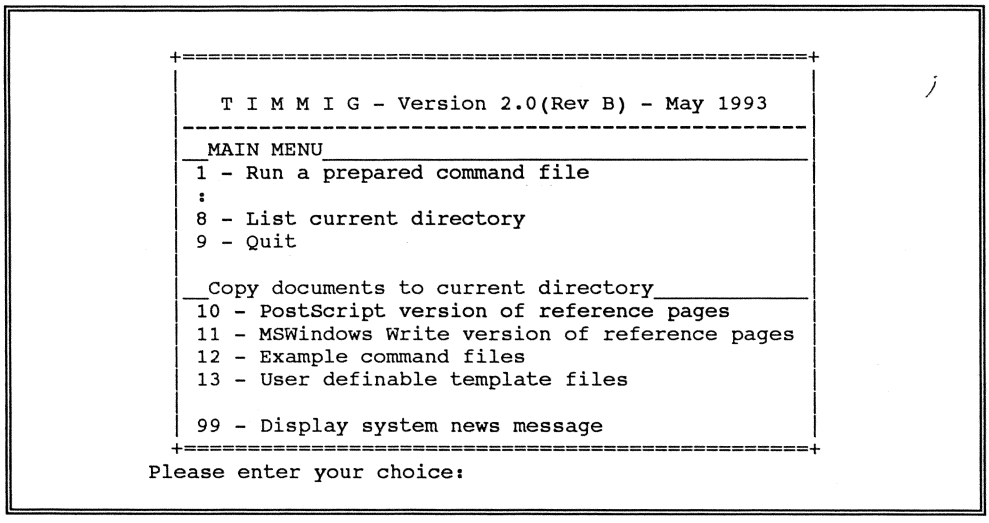
The program is currently operated by using a command file - a text file which contains a series of instructions. An interactive interface which allows users to build a command file through a series of menus should be available shortly. A help file is kept on the geo6lib disk, which can be copied and printed out. The file contains a list of ID codes for different spatial scales, together with a copy of the command definitions on the following pages. There are two versions of this file - one is in Windows WRITE format, the other is in Postscript format. The file(s) can be copied using options from the intial menu and then be printed out to use as a set of reference sheets. The Postscript version can be printed directly using a Postscript compatible printer. The WRITE version should be transferred to a PC which is capable of running MS-Windows. The file can then be printed using the standard Windows printing procedures. See section 5.6 for help with transferring a file to a PC. **N.B.** The Write copy of the mini manual is in a binary form. It is necessary to transfer it as such. This can be

done either by typing **binary** at the ftp prompt before retrieving the file, or by using the command **bget miniman.wri**. [bget=binary_get].

5.2.2 Running the program

When the program is started, the user will be prompted with a menu as shown in Figure 4. Currently this supports only 1 major option - running a prepared command file - put should be expanded to include more options in the future.

Figure 4: The initial menu in TIMMIG



Users should enter the number of the required option. On selecting option one, the program will prompt the user to enter the name of the command file to be run. If the filename supplied can not be found, then an error message will be returned, and the program will return to the main menu. The 'quit' option can be used at this stage to leave the program. Once a valid command file has been found, the program runs automatically. It can be terminated at any stage by pressing CTRL-C on the keyboard. The program intercepts this stop signal, and attempts to 'clean up' its run, by erasing temporary files. However, it is possible that either some files will be missed by this process, or that some error messages might be produced. If CTRL-C is used to stop the program running, than the user is advised to use the command '**ls -a**' to list the files in the current directory and then use the **rm** command to delete any files which have been missed (see Table 4).

5.3 Construction of command files

5.3.1 Naming command files

The command file can be given any name that is valid in UNIX, but must be less then twenty letters long.

5.3.2 Structure of command files

Command files can contain a mixture of instructions, blank lines and comment lines. The file should be created as a blank ASCII text file. This can be done using simple text editors such as **emacs** or **vi** on a UNIX system, or the MS-DOS **edit** or MS-Windows **notepad** programs on a PC. It should be noted that the first thing the program does is to convert the command file to lower case characters, thus any upper case characters used will not appear as upper case in the output file. Blank lines and comment lines may be placed anywhere in the command file. Comment lines start with the character '#', the user can place any required test after this character, in order to comment on what the command file is going to do. Instructions take the form '*command=instruction*', for example 'origin=all'. The first instruction is a command file must be the command 'start', and the last instruction must be 'end'. All other instruction lines must be placed between these two commands.

5.3.3. Command file instructions

The following pages list the range of command that are presently available in TIMMIG, and describe the range and effect of different settings.

COMMAND FILE TERMS: General parameters

start - to start a command sequence

Notes:

i) This command must be given - the program will terminate if the command cannot be found.

end - to end a command sequence

Notes:

i) This command must be given - the program will terminate if the command cannot be found.

author - An optional line of blank text, which can be used to add the name of the user to output.

Notes:

i) Only one line should be used - any text on subsequent lines will be ignored.

ii) All text will be converted into lower case lettering

iii) If 'table' has been selected as the output style, then the authors user name will automatically be printed on the page headers as well as the 'author' text.

title - An optional line of text, which can be used to describe the data in the output table.

Notes:

i) Only one line should be used - any text on subsequent lines will be ignored.

ii) All text will be converted into lower case lettering

iii) If 'table' has been selected as the output style, then the program will generate a description of the table, and print this on the burst page.

output - the output mode. Options are:

table -a formatted table with all labels, dividers and additional text. A cover sheet is included as well, showing the layout of pages, and some diagnostic information.

naked - only numerical information is included, no dividers. The data remains in a fixed format layout.

excel - row and column labels are included, values are separated by semicolons. This is a format designed to be imported into MS-Excel 4.0. The process of doing this is described in section 5.7. Some additional text is retained, the cover sheet that is produced in 'table' mode is not included.

printer - allows user to select a printer width. Options are:

narrow - page width is set to 80 columns. This mode should be used if the user wants to study the output on screen.

wide - page width is set to 132 columns, the width of a standard line printer.

filename - the name of the output file.

This should obey standard UNIX conventions. A directory can be included as part of the filename. e.g. If the directory that timmig is run form includes a sub directory called 'results', the filename 'results/out.001' could be given. The name should not be more than 36 characters in total.

COMMAND FILE TERMS: Time parameters

tstart - the start of the time-series. Options are:

year(quarter) - choose a year, and starting quarter if necessary

census - when using a type 1 query, the sets the program up to use a dataset most closely corresponding to the twelve month period prior to the 1991 Census

Notes:

i) For type 1 queries only a year need be specified, e.g. tstart=1985

ii) For type 2 queries. a starting quarter is required as well. e.g. tstart=1985q2

iii) The year should not be specified for the census option. Use tstart=census

quarters - the (integer) number of quarters in a time period.

Notes:

i) This category is ignored for type 1 queries. These queries can only be run for 4 quarters periods starting in the third quarter of a year.

tperiods - the (integer) number of periods (of length *quarters*) that the time series is to run for.

Notes:

i) The program makes an attempt to intelligently compensate for unusable time period classifications. Firstly, the length of the total time run (i.e. *tperiods* * *quarters*) is considered. If it is calculated that the time run will end at a point for which data is not available (that is, at present, a quarter after the second quarter of 1992), the program will recursively reduce the value of *tperiods* by one, until a valid end value is found. The program will abort with an error message if it can not fit any time periods with the specified length (in quarters) into the available data.

e.g. To specify a run of 5 years from 1985-86, the following setting would be applied:

```
tstart=1985      [ The start of the time series ]
quarters=4       [ Four quarters in each time period ]
tperiods=5       [ The length of the time series ]
```

scale - the spatial scale at which zones will be identified. Numbering of zones within these scales is shown in appendix A. Current options are:

fhsa - FHSAs of England and Wales, Scottish AHBs, Northern Ireland and the Isle of Man

county - Counties of England and Wales, Scottish regions, Northern Ireland and the Isle of Man

nuts2 - NUTS2 level regions of the United Kingdom.

stregion - Standard regions of the United Kingdom. (Note that Greater London and 'Rest of the South East' are defined as separate regions).

country - England, Wales, Scotland, Northern Ireland and Isle of Man, all as single zones

**<*userdefined*>* - Users can specify a customised spatial scale.

Notes:

i) For a full description of how to create and use customised scales, see section 5.4

ii) Note that for type 1 queries, the program automatically selects a trimmed down version of the spatial scale, which does not include Scottish Area Health Board areas, or the additional categories such as 'Armed Forces'.

iii) The 'county' files are aggregation into the counties of England and Wales, and the Scottish regions. This arrangement also conforms to NUTS level 3 zoning.

iv) The 'stregion' file conforms to NUTS level 1 zoning for all areas apart from the South East of England. NUTS level 1 groups Greater London and other parts of the South East into one zone; TIMMIG however separates Greater London and the surrounding counties into two zones.

v) TIMMIG includes the Isle of Man in addition to the standard NUTS zones of the United Kingdom.

COMMAND FILE TERMS: Spatial parameters (ii) - origin and destination

origin - a list of origin zones.

destination - a list of destination zones.

origin and destination both have the same options, and must be specified using the same rules.

Options are:

all - use all zones at that scale.

number list - a list of zone ID numbers - each zone used in each scale has a specific number.

The syntax for a number list is as follows:

i) Numbers should be separated by either a space or a comma.

ii) The number list can take up any number of lines

iii) The number list must be terminated with a semicolon.

iv) The numbers can be in any order.

v) A continuous range of numbers can be specified using a dash e.g. '16-23'. Backwards ranges (eg.11-3) are not allowed. If such a range was required, the individual zone ID numbers would have to be specified, separated by commas or spaces.

e.g.. origin=1,2,12-23;
destination=all;

row1,row2 - Primary and secondary row variables.

column1, column2 - Primary and secondary column variables.

layer1, layer2 - Primary and secondary layer variables.

All table dimension parameters have the same syntax. They must be one of the following options:

origin - tabulate by origin zone(s)

destination - tabulate by destination zone(s)

age - tabulate by age groups

gender - tabulate by genders

time - tabulate by time periods

&<variable> - Use the total of *variable* as a dimension.

Notes:

i) The categories *row1*, *column1* and *layer1* must be defined.

ii) All secondary terms (*row2, column2* and *layer 2*) are optional (they are used to collapse a second dimension into one table vertex - see section 3.3.2).

iii) Each variable can only be included once.

iv) For type 1 queries, there is a prohibition on tables of the form origin by destination by age/gender, as this information can not be extracted from the source tables.

v) Variables selected for row1, column1 and layer1 may be tabulated in a totalled form by prefixing the variable name with the ampersand character (&). This has two important effects. Firstly, and secondary variable selected for the dimension (e.g. row2 ,if row1 has been prefixed by &) will be ignored, and totals will not be automatically calculated in that dimension.

statistic - what the cell numbers actually show. Current options are:

net migration - the net flow from origin zone(s) or to destination zone(s)

in migration - the flow from origin zone(s) to destination zone(s)

out migration - the flow from destination zone(s) to origin zone(s)

Notes:

i) In type 1 queries, the statistic also depends on the choice a table dimensions. If the table is of the form origin by destination, flow will be in terms of total persons between specific zones. If the table contains age or gender as a dimension however, flow will be in terms of persons (as disaggregated by the choice of age and gender parameters) between tabulated origins or destinations and the whole of the rest of the United Kingdom (irrespective of either spatial scale of definition of origin and destination groups). This is due to the structure of the source tables (the step (1) database) for type 1 queries.

intracell - This is an optional command which allows intra-cell migration to be ignored. Options are:

on - The default

off - Do not count intra-cell flows

Notes:

i) Intra-cell flows become important at coarser spatial scales. For example, at the country level, the largest tabulated flow will be 'England to England'. If intra-cell flows are turned off, this cell would remain at zero.

age - the pattern of age grouping. Present options are:

fiveyear - Use five year age groups (0-4,5-9,...,75+)

singleyr - Use single years of age: This option is only available for type 2 queries.

*****<*userdefined*> - Use a user defined age grouping

Notes:

i) At present, fiveyear is an enforced option for type 1 queries.

ii) For a full description of how to create and use a customised age grouping see section 5.4

gender - which sexes to include in the aggregation. Present options are:

m - to include males

f - to include females

u - to include the unstated category where applicable (when using a type 2 query)

One or more of the above letters should be included, e.g. gender=mf

Notes:

i) The processing of age and gender depend on both the query type and the structure of the table.

For type 2 queries (when these are made possible), age and gender will have two possible roles. Firstly, they act as a filter for records to be processed. Individual records will only be processed if they are within the bounds established by the settings for age and gender. Secondly, they can be chosen to be used as table dimensions, and the tabulated output will be disaggregated accordingly.

For type 1 queries, the gender and age categories are effectively bypassed for tables of the form origin by destination. Such tables are compiled from existing tables of data which include data for both genders and all ages, but are not disaggregated by either category. Thus age and gender can not be used as either filters or table dimensions. For tables of the form [origin or destination] by age and/or gender, age and gender will be treated in the same way as for type 2 queries, that is they can be used as both filters and dimensions. However the flexibility of this is currently limited.

5.4: Using user defined scales and ages

5.4.1 General

Both scale and age may be included as a user defined option. The process of creating and using a customised scale or pattern of age grouping is described in this section. Both age and scale can be customised by a similar process, and are dependent on the same condition. Template files should be copied from the geo6lib archive, and altered using a standard text editor. The altered file should be given a suitable name, and must be placed in the same directory as the TIMMIG script. In the current version of TIMMIG, customised scales and age groupings must be prepared by manually editing the template file. It is hoped that future versions will include an interactive utility to assist the construction of customised files. When this is available, it will be listed as an option on the initial menu that is displayed when the command 'timmig' is entered.

5.4.2 Copying the template files

Copies of the template files can be extracted using the relevant options on the initial TIMMIG menu. The files will be copied into the current directory.

5.4.3 The structure of the scale template file

The file template.agg is constructed as follows:

- The first four lines form a header, in which descriptive comments can be stored. This may be altered, but should always be four lines long.
- The fifth line contains a number, and a text label. The number is equal to the total number of different zones in the new spatial scale, and the text is a descriptive label for the scale. The number is formatted (in FORTRAN) as I3.
- The next 125 lines consist of a list of fundamental areas, that is, the areas that are defined at the most discrete level . This is the same definition as used in the file *fhsa_ful.agg* to define the full FHSA / AHB based spatial zoning system. The names of these zones are shown on the left hand side of the line, as a reference. On the right hand side of the line is a number. This is the number of the new zone of which the FHSA or AHB will become a part. This number should be an integer, and the new zones should be numbered sequentially from 1 to the number of zones (which should then be stores on line 5).
- Following the last fundamental zone (category 125 - Psychiatric or Prisoner) is another 4 line descriptive comment. Again, this comment can be altered, but should always be 4 lines long.
- After this second comment are a list of labels for the zones in the new scale, together with their corresponding numbers. These should be the same number of labels (with one label to a line) as there are new zones. The labels and numbers are formatted as (A24,I3) in FORTRAN. The zone number in this section is included for reference only - the labels should be sequentially ordered, the first label corresponding to the new zone 1.

5.4.4 The structure of the age template file

The file template.age is constructed as follows:

- The first four lines form a header, in which descriptive comments can be stored. This may be altered, but should always be four lines long.
- The fifth line contains a number. This is the total number of age groups in the new grouping.
- The remainder of the file consists of an array of numbers. There are 16 numbers per line, which contain the translation codes for eight single years. The numbers are paired: single year of age (as it will be read from individual records in type 2 queries), and then the number of the age group that this year falls into. The template file is a copy of the fiveyear.age file thus the first line starts ' 0 1 1 1 2 1 3 1 4 1 5 2 '...; this indicates that the first five single year ages (from under 1, which is coded as '0', to '4') will form the first age group, while the sixth single year age -'5' - will be the first year in the second age group.

5.5: Printing out your results

5.5.1 Direct printing

The output from TIMMIG can be printed out using the command **lpr**. This can be appended with various options. The most important is the -P option. This is used to name the printer to which output will be sent. A list of printers can be displayed by entering the command printers. The line printer in the Baines Wing is called fuji6, thus to print to this printer, you should use the command **lpr -Pfuji6** *<filename>*. Previous versions of the TIMMIG program required users to reformat the carriage control characters in output using the UNIX fpr utility; this is no longer necessary as the program changes the carriage control characters internally.

The UNIX command **man lpr** will describe fully the command lpr and its possible options.

5.5.2 Indirect printing

For various reasons, the above method of printing files may be either impossible or undesirable. If this is the case, then the user may want to print the file using a PC. The methods of transferring a file to a PC are discussed in the following section (5.6). Once transferred, the file can be printed as a text file to a suitable printer, or processed for a laser printer using MS-Windows or any other standard method.

5.6 File transfer

5.6.1 General

You may wish to transfer the output file to a PC in order to either edit it, to print it using a printer connected to a PC or to work on the file using another package. The methods of transferring a file from a UNIX host to a PC are numerous; this section summarises one that can be used on the University of Leeds campus network - ftp (File Transfer Protocol). This is the most common way of transferring files between computers. The following instructions describe how to use this facility on the University of Leeds campus network. There are two approaches to transferring files - either *sending* them to another computer (i.e. the transfer is initiated from the place where the original copy of the file is) or *getting* then (i.e. the transfer is initiated from the place where the new copy is to be made). The first part of this section describes some general commands that are used in both approaches.

5.6.2 General ftp principles

All ftp sessions have a broadly similar nature. Firstly you must establish a connection with another computer, and then you must (in most cases) login to the computer. Once this is done, files can be transferred. After files have been transferred, the connection between computers should be closed. This process automatically logs the user out of the remote system as well. The ftp session can then be ended. It is important to bear in mind the relationship between the remote and local systems in ftp. The local system is the one that the ftp session is started from, the remote one is the system that the user connects with. It is possible therefore that the 'local' system, in this respect, may be in a different building, or even in a different country, whilst it is quite likely that the 'remote' system is the computer that you are sitting at.

Some common commands that are used are listed in table5 below. Note that not all of the above commands will necessarily be supported by all ftp software. Typing 'help' should give a list of all the commands that are supported by the ftp software that you are using.

Table 5: Common FTP commands

ftp	to start an ftp session.
open	to make a connection with another computer.
ls	to list the files in the current directory on the remote system.
lls	to list the files in the current directory on the local system.
cd	to change directory on the remote system.
lcd	to change directory on the local system.
put	to copy files from the local to the remote systems.
mput	multiple put - uses standard wild card characters (e.g. mput * to put all files in a directory).
get	to copy files from the remote to the local systems.
mget	multiple get - uses standard wild card characters.
close	to break the connection with the remote system.
bye	to end the ftp session.

5.6.3 Using a Novell server as the local system

This method assumes that you have logged out of the GPS but are still logged on to a Novell server (such as West-01).

i) Enter **ftp**. The system should respond by giving the prompt 'ftp>'.

ii) Enter **open gps**. This establishes a connection with the GPS.

Note: Steps i) and ii) can be combined, by entering **ftp gps**.

iii) You will now be prompted to enter your user name and password for the GPS.

iv) When you have successfully logged in, use **cd** to change to the directory where the files that you want to copy are located. The command **ls** can be used to check the directory listing.

v) Enter **get** to get a copy of the files you want.

Note: The get command can be used in several ways:

get on its own will prompt you to supply the local and remote names of the file.

get <*filename1*> will copy the file called *filename1*. The computer will copy this name as closely as possible for the name of the copy.

get <*filename1*> <*filename2*> will copy *filename1*, and give the copy the name *filename2*.

vi) When you have copied all the required files, enter **close** to end the connection with the GPS.

vii) Enter **bye** or **quit** to end the ftp session.

5.6.4 Using the GPS as the local system

Note: This method assumes that you are still logged onto the GPS, and that you are accessing it using the Clarkson University telnet program CUTCP.

i) Remain logged onto the GPS

i) Make sure you are in the same directory as the file(s) that you wish to transfer.

iii) Type **ALT-T**.(i.e. press T whilst holding down the ALT key). This is a keyboard shortcut, which sends the command **ftp** *<IP number>* to the GPS. The IP number is a unique number which identifies the PC that you are using; the command thus instructs the GPS to open a connection with that PC (In this instance, the GPS is the local system, and the PC that you are using is the remote system). This connection could be achieved by entering **ftp** and **open** *<IP number>* as separate commands, but only if you know the IP number of the machine that you are using!

Note: If this does not work, it is possible that the computer you are working at is using a communications program other than Clarkson University telnet. The keyboard shortcut which generates 'ftp <IP number> is very common however. ALT-F is sometimes used to do this. Many telnet programs use the combination ALT-H to display a list of keyboard macros.

iv) If you have successfully made a connection with the PC that you are using, you will be prompted for login information. However most cluster based PCs do not require this information. You should press return when prompted for a user name, and **ALT-W** when prompted for a password. You should then get a message saying that you are logged in.

v) Enter **put** or **send**. These commands are used to transfer a file from a host (in this case the GPS) to a client (in this case the PC you are working at). The syntax of the command is the same as for 'get' (see instruction v) of section 5.6.3 above).

Note: Many of the cluster PCs prevent users writing to the C:\ drive. You should make sure that you send files to a drive that you do have write permission on. This includes the F:\ drive if you are also logged on to a Novell server. Alternatively, files can be sent to a formatted floppy disk in the A:\ drive (assuming that there will be enough space on the floppy disk). This can be achieved by including the drive letter in the name of the remote file, e.g. **put myfile a:\myfile**.

vi) When you have transferred all the files that are required, use **close** to end the connection.

vii) Type **bye** or **quit** to end the ftp session.

5.7 Import into Excel

The *excel* output format is designed to minimise difficulties in transferring data to this package. Once a copy of the TIMMIG output file has been transferred to a PC, it can be imported into Excel using the following instructions. N.B. This assumes the user has some familiarity with the usage of Windows based packages. The instructions use the symbol '/' to indicate nested menus - for example the instruction File/Open/*<filename>* would mean that the user should select File, and from the new menu select Open, and then enter *filename* in the dialogue box.

i) Start Excel.

ii) Go to the **File / Open** dialogue box.

iii) Select **List Files of Type / All types (*.*)** [In the lower left corner]

iv) Click the **Text** button [Right hand side of the dialogue box]

v) Select **Semicolon** as the delimiter in the pop-up box. (Note: Semicolons are used as delimiters rather than the more syandard comma, because of the use of commas in some area place names).

vi) Click the **OK** button of the pop-up box

vii) In the directory panel (in the middle of the dialogue box) find the directory that your TIMMIG file is in.

viii) Double-click the name of your file in the panel on the left of the dialogue box.

This should open up a new worksheet with the TIMMIG output properly laid out. (There may be a few errors in the layout for large files, which should be corrected using the standard Excel cut and paste facilities).

6. CONCLUSIONS

6.1 Conclusion

This paper has discussed the creation and operation of a flexible querying program to access a database on migration of people in the Unied Kingdom. The software provides a useful resource tool for the extraction of new datasets. The program structure is flexible and modular, allowing for significant expansion and development, and it is hoped to adapt the software for use by the wider academic community.

6.2 Future developments

There are a number of developments that are planned for future versions of the software. These can be separated into improvements in functionality, and improvements in user-friendliness. The programs involved have been subject to several revisions and waves of restructuring, and thus it is expected that continuing maintenance and editing of the code should optimise storage requirements and improve the speed of operation.

6.2.1 Improvements in functionality

These improvements mainly centre on the introduction of two additional datasets - population information, and the individual NHSCR records. Much of the code to handle these datasets has been written and prototype routines have been successfully tested. The main drawback at present involves the speed of aggregation of tables from individual records; alternative processing algorithms are being tested, although the main question at the moment would seem to be how to balance between minimising the volume of data and the level of indexing included in the data.

The introduction of the population data will necessitate changes to the exiting statistic command. It is envisaged that future versions of the software will have a greater range of options for the statistc command, and the command itself will be split into two parts, which will represent a numerator and an (optional) demnoinator. This will allow the tabulation of rates of migration. A new command will also be introduced to allow data (such as rates) to be indexed to a particular year (that is, the chosen year will have values set to 100, and all other years / time periods will display data as a number relative to 100).

6.2.2 Improvements in user-friendliness

The most pressing area of concern is the use of user defined spatial scales. These currently have to be prepared manually by altering a template file, and as with any file read by a FORTRAN program, this process is dependent on the file being correctly formatted. It is planned to introduce an interactive program which will prompt users to supply the necessary information. The program will write out correctly formatted data to a new file, which can then be used in a command file.

A further possible enhancement, but one with a lower priority, is the use of an interactive program to help users construct command files. Minor improvements which are planned include better file handling capabilities. These should include the ability of the user to specify directories in which to search for files, and the directory to which output file should be written.

REFERENCES

Boden, P. (1989), The analysis of internal migration in the United Kingdom using Census and National Health Service Central Register data, *Unpublished Ph.D. thesis*, School of Geography, University of Leeds, Leeds

Rosenbaum, M. and Bailey, J. (1991), Movement within England & Wales during the 1980s, as measured by the NHS Central Register, *Population Trends*, 65, 24-34

APPENDICES

Appendix A.1: Tables of regions by scale

Note that scales listed as {type 2} refer to the scale when applied to type 2 queries, whilst scales listed as {type 1} refer to the scale when applied to type 1 queries.

A1.1: Scale: fhsa {type 2}

1	Northern Ireland	43	Nottinghamshire	85	Dudley
2	Ayr and Arran	44	Cambridgeshire	86	Sandwell
3	Border	45	Norfolk	87	Solihull
4	Argyll and Clyde	46	Suffolk	88	Walsall
5	Fife	47	City,Hackney,Newham,Tr Hamlets	89	Wolverhampton
6	Greater Glasgow	48	Redbridge, Waltham Forest	90	Hereford & Worcester
7	Highland	49	Barking, Havering	91	Shropshire
8	Lanark	50	Camden, Islington	92	Staffordshire
9	Grampian	51	Kensington, Chelsea, Westminster	93	Warwickshire
10	Orkney	52	Richmond, Kingston	94	Bolton
11	Lothoian	53	Merton, Sutton, Wandsworth	95	Bury
12	Tayside	54	Croydon	96	Manchester
13	Forth Valley	55	Lambeth, Southwark, Lewisham	97	Oldham
14	Western Isles	56	Bromley	98	Rochdale
15	Dumfries and Galloway	57	Bexley, Greenwich	99	Salford
16	Shetland	58	Enfield and Haringey	100	Stockport
17	Scotland (general / n.f.s)	59	Barnet	101	Tameside
18	Isle of Man	60	Hillingdon	102	Trafford
19	Gateshead	61	Brent and Harrow	103	Wigan
20	Newcastle	62	Middlesex (general)	104	Liverpool
21	South Tyneside	63	Ealing, Hammersmith,Hounslow	105	St Helens & Knowsley
22	North Tyneside	64	Bedfordshire	106	Sefton
23	Sunderland	65	Buckinghamshire	107	Wirral
24	Cleveland	66	Essex	108	Cheshire
25	Cumbria	67	Hertfordshire	109	Lancashire
26	Durham	68	Berkshire	110	Clwyd
27	Northumberland	69	East Sussex	111	Dyfed
28	Barnsley	70	Hampshire	112	Gwent
29	Doncaster	71	Isle of Wight	113	Gwynedd
30	Rotherham	72	Kent	114	Mid Glamorgan
31	Sheffield	73	Oxfordshire	115	Powys
32	Bradford	74	Surrey	116	South Glamorgan
33	Calderdale	75	West Sussex	117	West Glamorgan
34	Kirklees	76	Avon	118	Not stated
35	Leeds	77	Cornwall	119	Abroad: Native land unstated
36	Wakefield	78	Devon	120	Abroad: Last residence Eire
37	Humberside	79	Dorset	121	Abroad: Last res. other than Eire
38	North Yorkshire	80	Gloucestershire	122	Return Migrant
39	Derbyshire	81	Somerset	123	Armed Forces
40	Leicestershire	82	Wiltshire	124	Armed Forces Dependent
41	Lincolnshire	83	Birmingham	125	Psychiatric / Prisoner
42	Northamptonshire	84	Coventry		

A1.2: Scale: fhsa {type 1}

1	Northern Ireland	34	Barking, Havering	67	Solihull
2	Scotland	35	Camden, Islington	68	Walsall
3	Isle of Man	36	Kensington, Chelsea, Westminster	69	Wolverhampton
4	Gateshead	37	Richmond, Kingston	70	Hereford & Worcester
5	Newcastle	38	Merton, Sutton, Wandsworth	71	Shropshire
6	South Tyneside	39	Croydon	72	Staffordshire
7	North Tyneside	40	Lambeth, Southwark, Lewisham	73	Warwickshire
8	Sunderland	41	Bromley	74	Bolton
9	Cleveland	42	Bexley, Greenwich	75	Bury
10	Cumbria	43	Middlesex	76	Manchester
11	Durham	44	Bedfordshire	77	Oldham
12	Northumberland	45	Buckinghamshire	78	Rochdale
13	Barnsley	46	Essex	79	Salford
14	Doncaster	47	Hertfordshire	80	Stockport
15	Rotherham	48	Berkshire	81	Tameside
16	Sheffield	49	East Sussex	82	Trafford
17	Bradford	50	Hampshire	83	Wigan
18	Calderdale	51	Isle of Wight	84	Liverpool
19	Kirklees	52	Kent	85	St Helens & Knowsley
20	Leeds	53	Oxfordshire	86	Sefton
21	Wakefield	54	Surrey	87	Wirral
22	Humberside	55	West Sussex	88	Cheshire
23	North Yorkshire	56	Avon	89	Lancashire
24	Derbyshire	57	Cornwall	90	Clwyd
25	Leicestershire	58	Devon	91	Dyfed
26	Lincolnshire	59	Dorset	92	Gwent
27	Northamptonshire	60	Gloucestershire	93	Gwynedd
28	Nottinghamshire	61	Somerset	94	Mid Glamorgan
29	Cambridgeshire	62	Wiltshire	95	Powys
30	Norfolk	63	Birmingham	96	South Glamorgan
31	Suffolk	64	Coventry	97	West Glamorgan
32	City,Hackney,Newham,Tr Hamlets	65	Dudley	*	
33	Redbridge, Waltham Forest	66	Sandwell		

A1.3: Scale: county {type 2}

1	Northern Ireland	26	Northamptonshire	51	West Midlands
2	Strathclyde	27	Nottinghamshire	52	Hereford & Worcester
3	Borders	28	Cambridgeshire	53	Shropshire
4	Fife	29	Norfolk	54	Staffordshire
5	Highland	30	Suffolk	55	Warwickshire
6	Grampian	31	Greater London	56	Greater Manchester
7	Islands	32	Bedfordshire	57	Merseyside
8	Lothian	33	Buckinghamshire	58	Cheshire
9	Tayside	34	Essex	59	Lancashire
10	Central	35	Hertfordshire	60	Clwyd
11	Dumfries & Galloway	36	Berkshire	61	Dyfed
12	Scotland (general / n.f.s)	37	East Sussex	62	Gwent
13	Isle of Man	38	Hampshire	63	Gwynedd
14	Tyne & Wear	39	Isle of Wight	64	Mid Glamorgan
15	Cleveland	40	Kent	65	Powys
16	Cumbria	41	Oxfordshire	66	South Glamorgan
17	Durham	42	Surrey	67	West Glamorgan
18	Northumberland	43	West Sussex	68	Not stated
19	South Yorkshire	44	Avon	69	Abroad: Native land unstated
20	West Yorkshire	45	Cornwall	70	Abroad: Last residence Eire
21	Humberside	46	Devon	71	Abroad: Last res. other than Eire
22	North Yorkshire	47	Dorset	72	Return Migrant
23	Derbyshire	48	Gloucestershire	73	Armed Forces
24	Leicestershire	49	Somerset	74	Armed Forces Dependent
25	Lincolnshire	50	Wiltshire	75	Psychiatric / Prisoner

A1.4: Scale: county {type 1}

1	Northern Ireland	20	Suffolk	39	Somerset
2	Scotland	21	Greater London	40	Wiltshire
3	Isle of Man	22	Bedfordshire	41	West Midlands
4	Tyne & Wear	23	Buckinghamshire	42	Hereford & Worcester
5	Cleveland	24	Essex	43	Shropshire
6	Cumbria	25	Hertfordshire	44	Staffordshire
7	Durham	26	Berkshire	45	Warwickshire
8	Northumberland	27	East Sussex	46	Greater Manchester
9	South Yorkshire	28	Hampshire	47	Merseyside
10	West Yorkshire	29	Isle of Wight	48	Cheshire
11	Humberside	30	Kent	49	Lancashire
12	North Yorkshire	31	Oxfordshire	50	Clwyd
13	Derbyshire	32	Surrey	51	Dyfed
14	Leicestershire	33	West Sussex	52	Gwent
15	Lincolnshire	34	Avon	53	Gwynedd
16	Northamptonshire	35	Cornwall	54	Mid Glamorgan
17	Nottinghamshire	36	Devon	55	Powys
18	Cambridgeshire	37	Dorset	56	South Glamorgan
19	Norfolk	38	Gloucestershire	57	West Glamorgan

A1.5 Scale: nuts2 {type 2}

1	Northern Ireland	16	Leicestershire, Northants	31	Shropshire, Staffs
2	Brdrs,Cntral,Fife,Lthian,Tyside	17	Lincolnshire	32	Greater Manchester
3	Dumfries&Galloway, Strathclyde	18	East Anglia	33	Merseyside
4	Highland & Islands	19	Greater London	34	Cheshire
5	Grampian	20	Bedfordshire, Herts	35	Lancashire
6	Scotland (general / n.f.s)	21	Bucks, Berks, Oxfordshire	36	Clwyd,Dyfed,Gwynedd,Powys
7	Isle of Man	22	Essex	37	Gwent, Mid,S,W,Glamorgan
8	Northumbs,Tyne & Wear	23	Surrey, E&W Sussex	38	Not stated
9	Cleveland, Durham	24	Hampshire, Isle of Wight	39	Abroad: Native land unstated
10	Cumbria	25	Kent	40	Abroad: Last residence Eire
11	South Yorkshire	26	Avon, Gloucs, Wilts	41	Abroad: Last res. other than Eire
12	West Yorkshire	27	Cornwall, Devon	42	Return Migrant
13	Humberside	28	Dorset, Somerset	43	Armed Forces
14	North Yorkshire	29	West Midlands	44	Armed Forces Dependent
15	Derbyshire, Notts	30	Hereford & Worcs, Warks	45	Psychiatric / Prisoner

A1.6: Scale: nuts2 {type 1}

1	Northern Ireland	12	Leicestershire, Northants	23	Cornwall, Devon
2	Scotland	13	Lincolnshire	24	Dorset, Somerset
3	Isle of Man	14	East Anglia	25	West Midlands
4	Northumbs,Tyne & Wear	15	Greater London	26	Hereford & Worcs, Warks
5	Cleveland, Durham	16	Bedfordshire, Herts	27	Shropshire, Staffs
6	Cumbria	17	Bucks, Berks, Oxfordshire	28	Greater Manchester
7	South Yorkshire	18	Essex	29	Merseyside
8	West Yorkshire	19	Surrey, E&W Sussex	30	Cheshire
9	Humberside	20	Hampshire, Isle of Wight	31	Lancashire
10	North Yorkshire	21	Kent	32	Clwyd,Dyfed,Gwynedd,Powys
11	Derbyshire, Notts	22	Avon, Gloucs, Wilts	33	Gwent, Mid,S,W,Glamorgan

A1.7: Scale: stregion {type 2}

1	Northern Ireland	8	East Midlands	15	Abroad: Native land unstated
2	Scotland	9	West MIdlands	16	Abroad: Last residence Eire
3	Wales	10	East Anglia	17	Abroad: Last res. other than Eire
4	Isle of Man	11	South East (exc. London)	18	Return Migrant
5	North	12	Greater London	19	Armed Forces
6	North West	13	South West	20	Armed Forces Dependent
7	Yorkshire & Humberside	14	Not stated	21	Psychiatric / Prisoner

A1.8 Scale: stregion {type 1}

1	Northern Ireland	6	North West	11	South East (exc. London)
2	Scotland	7	Yorkshire & Humberside	12	Greater London
3	Wales	8	East Midlands	13	South West
4	Isle of Man	9	West MIdlands		
5	North	10	East Anglia		

A1.9: Scale: country {type 2]

1	Northern Ireland	6	Not Stated	11	Armed Forces
2	Scotland	7	Abroad: Native land unstated	12	Armed Forces Dependent
3	Wales	8	Abroad: Last residence Eire	13	Psychiatric/Prisoner
4	Isle Of Man	9	Abroad: Last res. other than Eire		
5	England	10	Return Migrant		

A1.10 Scale: country {type 1}

1	Northern Ireland	4	Wales
2	Scotland	5	England
3	Isle Of Man		

Appendix B1: Listing of MOVES5

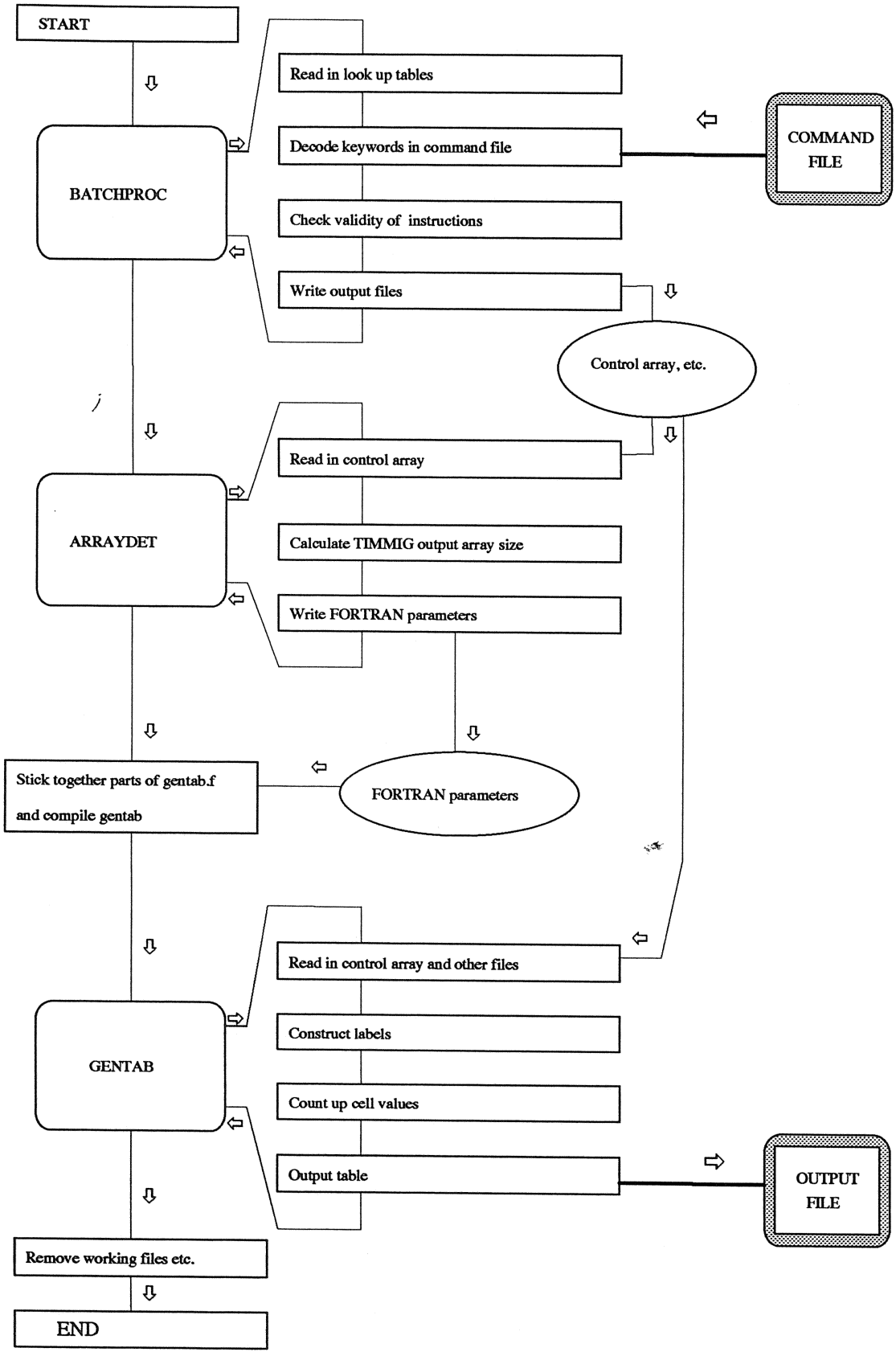
```
      PROGRAM MOVES5

      IMPLICIT REAL*8 (A-H,M,O-Z)
      CHARACTER*16384 REC
      CHARACTER*28 REC2(584)
      CHARACTER*2 FILLER
      CHARACTER*1 FILTER
      CHARACTER*4 OUTREC
      INTEGER ROSA(125,3,101),RDSA(125,3,101),ROD(125,125)
      INTEGER ICODE(0:99999),ITAPE(125),IAGCOD(101),IPROG(125,2)
C-----CONTROL CODES-------------------------------------------------------+
C IMODE REFERS TO TYPE OF OUTPUT (NORMAL / COMPRESSED )                   |
C   IMODE=1 - MAINLY OUTPUT                                               |
C   IMODE=2 - COMPRESSED OUTPUT                                           |
C IMTTYPE REFERS TO MAG TAPE STYLE                                        |
C   IMTTYPE=1 - PRECOMPUTERISED TAPES                                     |
C   IMTTYPE=2 - POST COMPUTERISED TAPES                                   |
C   IMTTYPE=3 - F39095 HYBRID                                             |
C IVER REFERS TO PROGRAM VERSION                                          |
C   IVER=1 - RETAINS MOVES4 ORDERING ETC                                  |
C   IVER=2 - FULL EXPANDED ORDERING                                       |
C-------------------------------------------------------------------------+
      OPEN (38,ACCESS='DIRECT',RECL=4)
      IMODE=2
      IMTTYPE=2
      IVER=1
C--FILL IN ICODE LOOK UP TABLE--------------------------------------------
      DO 10 N=1,4
 10      READ (2,'(A2)') FILLER
      DO 20 N=1,125
 20      READ (2,'(2I4)') (IPROG(N,I),I=1,2)
      DO 30 N=1,4
 30      READ (3,'(A2)') FILLER
      DO 40 N=1,125
         READ (3,'(I5)') ITAPE(N)
 40      ICODE(ITAPE(N))=IPROG(N,IVER)
      READ (4,'(7(4X,I3))') (IAGCOD(IA),IA=1,101)
      WRITE (6,*) 'AGES READ OK'
C--MAIN CONTROL SECTION---------------------------------------------------
      IF (IMTTYPE.LT.1.OR.IMTTYPE.GT.3) THEN
       WRITE (6,*) 'INCORRECT SETTING FOR IMTTYPE!'
       STOP
      ENDIF
      IF (IMODE.EQ.1) THEN
         DO 50 I=1,104
 50         READ (8,'(15F8.0)') (ROD(I,J),J=1,104)
         DO 60 I=1,104
           DO 60 IS=1,3
             READ (9,'(15F8.0)') (ROSA(I,IS,IA),IA=1,101)
 60          READ (10,'(15F8.0)') (RDSA(I,IS,IA),IA=1,101)
         WRITE (6,*) 'INARRAYS READ OK'
      ENDIF
      WRITE (6,*) 'PROCESS SECTION ENTERED'
C---INITIALIZE COUNTERS---------------------------------
      IR=1
      NRCT=0
      NRCP=0
      NRAB=0
      NROD=0
      NRD2=0
      NFLT=1
C---IMTTYPE DEPENDENCIES:
C  IST: START OF RECORD IN SUBSTRING
C  IND: END OF RECORD IN SUBSTRING
C  ILRL: LOGICAL LENGTH OF RECORD
C  INLR: LOGICAL NO OF RECORDS PER BLOCK
      IF (IMTTYPE.EQ.1) THEN
         WRITE (6,*) 'PROCESSING AS PRE-COMPUTERISED TAPE...'
         IST=9
         IND=36
         ILRL=28
         ILNR=584
      ENDIF
      IF (IMTTYPE.EQ.2) THEN
         WRITE (6,*) 'PROCESSING AS POST-COMPUTERISED TAPE...'
         IST=13
         IND=36
         ILRL=61
         ILNR=201
      ENDIF
      IF (IMTTYPE.EQ.3) THEN
         WRITE (6,*) 'PROCESSING AS DEC90/90Q3 TAPE...'
         IST=13
         IND=26
         ILRL=61
         ILNR=584
      ENDIF

C---MAIN LOOP-------------------------------------------------

 70   CONTINUE
C     IF (IR.GT.2) GOTO 200
      READ (1,'(A16384)',END=200) REC
      DO 100 I=1,ILNR
      REC2(I)=REC((IST+((I-1)*ILRL)):(IND+((I-1)+ILRL)))
      NRCT=NRCT+1
C---IMPOSE FILTER ON REC2 (IF NFLT=1)-------------------------------
      IF (NFLT.EQ.0) GOTO 80
      DO 80 IFLT=1,23
        FILTER=REC2(I)(IFLT:IFLT)
        IF (FILTER.EQ.';') GOTO 100
        IF (FILTER.LT.'0'.OR.FILTER.GT.'9') REC2(I)(IFLT:IFLT)='0'
 80   CONTINUE
C---NOW READ FROM REC2 FIELDS
      READ (UNIT=REC2(I),FMT=101,END=200) IDEST,IORIG,ISEX,IYEAR,
     * ITYMO,IAGE,IDAY,IDIR
101   FORMAT (2I5,I1,I3,I2,2I3,I1,5X)
C-------DUMP TEST VALUES?---------------------------------------------
      IF (IR.EQ.1) THEN
       WRITE (37,'(A24)') REC2(I)
      ENDIF
C-----PROCESS REC2 FIELDS-------------------------------------------
      IF (ISEX.EQ.0) GOTO 100
      IF (IDIR.EQ.2) THEN
       NRD2=NRD2+1
       GOTO 100
      ENDIF
      IF (ICODE(IORIG).EQ.ICODE(IDEST)) THEN
       NROD=NROD+1
       GOTO 100
      ENDIF
      IF (IAGE.EQ.999) IAGE=100
      IAGE=IAGE+1
      IF (ISEX.EQ.9) ISEX=3
      IF (IYEAR.EQ.999) IYEAR=255
      IOR=ICODE(IORIG)
      IDS=ICODE(IDEST)
C---INCREMENT ARRAYS OR WRITE ABBREVIATED RECORD---------------
      IF (IMODE.EQ.1) THEN
        ROD(IOR,IDS)=ROD(IOR,IDS)+1
        IF (IOR.GE.98.OR.IDS.GE.98) THEN
          NRAB=NRAB+1
          GOTO 100
        ENDIF
        ROSA(IOR,ISEX,IAGE)=ROSA(IOR,ISEX,IAGE)+1
        RDSA(IDS,ISEX,IAGE)=RDSA(IDS,ISEX,IAGE)+1
        NRCP=NRCP+1
      ENDIF
      IF (IMODE.EQ.2) THEN
       NRCP=NRCP+1
       IF (ISEX.EQ.2) IDS=IDS+128
       IF (ISEX.EQ.3) IOR=IOR+128
       IF (IYEAR.LT.0.OR.IYEAR.GT.255) WRITE (6,*) IYEAR
       WRITE(38,'(4I3)') IOR,IDS,IAGE,IYEAR
      ENDIF
C---END OF LOOP-----------------------------------------------------
100   CONTINUE
      RR=IR
      IF ((RR/100).EQ.INT(RR/100)) THEN
       WRITE (6,*) 'BLOCK ',IR,' HAS BEEN PROCESSED'
      ENDIF
      IF ((RR/1000).EQ.INT(RR/1000)) THEN
       WRITE (6,*) 'SUMMARY VALUES:',NRCT,NRCP
      ENDIF
      IR=IR+1
110   GOTO 70
C---END OF TAPE-----------------------------------------------------
200   WRITE (6,*) 'END OF TAPE'
      WRITE (6,*) 'LOGICAL RECORDS READ:',NRCT
      WRITE (6,*) 'NO OF RECORDS USED:',NRCP
      WRITE (6,*) 'NO RECS ABROAD :',NRAB
      WRITE (6,*) 'NO RECS IDIR=2:',NRD2
      WRITE (6,*) 'NO RECS OR=DES:',NROD
      NURT=NROD+NRD2+NRAB+NRCP
      WRITE (6,*) 'TOTAL PROCESSED RECS:',NURT
      IF (IMODE.EQ.1) THEN
        DO 210 I=1,104
 210      WRITE (11,'(15F8.0)') (ROD(I,J),J=1,104)
        WRITE (6,*) 'ROD WRITTEN OK'
        DO 220 I=1,104
          DO 220 IS=1,3
            WRITE (13,'(15F8.0)') (ROSA(I,IS,IA),IA=1,101)
 220        WRITE (14,'(15F8.0)') (RDSA(I,IS,IA),IA=1,101)
        WRITE (6,*) 'ROSA AND RDSA WRITTEN OK'
      ENDIF
      STOP
      END
```

Appendix B2: the TIMMIG command script

B2.1 Script listing

```
#!/bin/sh
trap "echo Stopped, attempting to clean up - some working files may not be erased.; rm .dat .who .pid .orderbook
.controls .params .tmgver .makegen g*.o .workfiles /tmp/*.$$*;exit" 2 3
dir="/tmp/"

if [ $# -gt 0 ]
then
while [ $# -gt 0 ]
do
case $1 in
      -l)
          dir="./"
          echo Running locally...
          ;;
      *)
          echo Unknown option $1
          ;;
 esac
 shift
done
fi

touch .runlog
echo --TIMMIG2: Run log----------------------- > .runlog
echo Started at: >> .runlog
date >> .runlog
echo >> .runlog
echo $dir > .path
echo $$ > .pid
whoami > .who
date > .dat
cp /home/gps/geo6lib/oliverdw/timl/batchproc ${dir}batchproc.$$
cp /home/gps/geo6lib/oliverdw/timl/arraydet ${dir}arraydet.$$
${dir}batchproc.$$
errstat=$?

if [ $errstat -eq 2 ]
then
rm .who .dat .pid
rm ${dir}*.$$
exit
fi

if [ $errstat -eq 3 ]
then
echo Exiting program because a fatal error was found in the command file...
rm .who .dat .pid
rm ${dir}*.$$
exit
fi

echo Copying chosen files...

for i in `cat .workfiles`
do
if [ -f $i ]
then
cp $i ${dir}$i.$$
else
cp /home/gps/geo6lib/oliverdw/tim2/$i ${dir}$i.$$
fi
done

echo Determining array limits...
${dir}arraydet.$$
echo Preparing unique version of tabulation program...
cat /home/gps/geo6lib/oliverdw/timl/gen01.f .params /home/gps/geo6lib/oliverdw/timl/gen02.f > gentab.f
cp ../oliverdw/timl/makegen .makegen
cp ../oliverdw/timl/goutput.o goutput.o
cp ../oliverdw/timl/gread.o gread.o
cp ../oliverdw/timl/glabels.o glabels.o
cp ../oliverdw/timl/gcountv1.o gcountv1.o
cp ../oliverdw/timl/gcountv2.o gcountv2.o
make -f .makegen -s 2>/dev/null
rm .makegen
rm goutput* gread* glabels* gcountv* gentab*
echo Running program...
cp gtab ${dir}gtab.$$
${dir}gtab.$$

echo Exiting...
rm .orderbook .params .pid .tmgver .who .dat .controls .workfiles
rm ${dir}*.$$* gtab
echo Finished at: >> .runlog
date >> .runlog
echo ---------------------------------------- >> .runlog
cat .runlog
rm .runlog
exit
```

B2.2 Diagramatic representation of TIMMIG structure



B2.3 A note about the commands used.

Almost all parts of TIMMIG are written in standard FORTRAN 77; there are however a few instances where SUN FORTRAN extension commands have been used. These may well be present in other modern FORTRAN compilers, but may limit portability to older compilers. N.B. Standard FORTRAN 77 uses upper case letters. The programs in TIMMIG wrere written in lower case lettering; this however can be simply changed to upper case using standard UNIX commands.

The extension commands which are used are:

lnblnk - This is an integer function, which returns the length of a string upto the last non-blank character in that string. e.g. *lnblnk ('ABC ')* would return a value of 3.

call exit (*n*) - This is an alternative to the stop command. It quits the program as stop does, but additionally set the UNIX environment variable $? to the value of *n*. This is useful for monitoring the error status of the program. The instances of call exit could be replaced by the standard command stop, but this would lead to messier results if the program was ended early (either by the quit command, or because of an unrecoverable error).

i=system (*string*) - This is an integer function which instructs the operating system to carry out the command that *string* is equivalent to; e.g. *i=system ('ls')* would print a listing of the current directory on the screen. The most significant system calls are the ones in *gcountv1.f* which repeatedly extract data files into a standard file, and then delete that file before extracting the next data file. This is done in order to conserve memory usage, and to allow the data to be permanently stored in a compressed form. These system call could therefore be avoided, but at the expense of memory and disk capacity.

open (.....,form='print') This is a special file handling option. The operating system strips off any FORTRAN carriage control characters contained in the output and replaces them with standard carriage control characters.

Appendix B3: Listing of batchproc

B3.1 Listing of batproc.f

```
C*********************************************************************

C**** Initial shell for Timmig2                                       *
C*********************************************************************

      program batproc

C**** Declare arrays at maximal size ****

      character*80 textout(4)
      character*60 filename
      character*24 stats(5)
      character*20 fname
      character*14 labels(20)
      character*12 vars(5),datafiles(80),keyfiles(2)
      character*8 aggfile(5),ages(2)
      character*6 pno
      character*4 years(20)
      character*2 ptst
      integer lablen(20),igr(125,2),control(30)
      integer logcat(20),dfstatus(80),system
      logical exists

      do 10 n=1,30
 10      control(n)=0

 20   call clearscreen

      write (6,*) '+================================================+'
      write (6,*) '|                                                |'
      write (6,*) '|     T I M M I G - Version 2.0 - May 1993       |'
      write (6,*) '|------------------------------------------------|'
      write (6,*) '|                                                |'
      write (6,*) '| NOTE: Only listed options are currently enabled |'
      write (6,*) '|                                                |'
      write (6,*) '|---MAIN MENU------------------------------------|'
      write (6,*) '|                                                |'
      write (6,*) '| 1 - Run a prepared command file                |'
c      write (6,*) '| 2 - Interactivly prepare a command file        |'
c      write (6,*) '| 3 - < To be added...>                          |'
      write (6,*) '| :                                              |'
      write (6,*) '| 8 - List current directory                     |'
      write (6,*) '| 9 - Quit                                       |'
      write (6,*) '|                                                |'
      write (6,*) '+================================================+'
      write (6,*)
 30   write (6,*) 'Please enter your choice:'
      read (5,*) il
      if (il.lt.1.or.il.gt.9) then
         write (6,*) 'Not a valid choice! Please try again.'
         goto 30
      endif
      if (il.ne.1.and.il.lt.8) then
         write (6,*) 'This option not ready yet! Please try again.'
         goto 30
      endif
      call clearscreen

      if (il.eq.8) then
      write (6,*) '==TIMMIG: Directory listing=====================''//
     *'=====================''
      i=system ('ls')
      write (6,*) '================================================''//
     *'=====================''
      write (6,*)
      write (6,*) '-Press return to continue-'
      read (5,*)
      goto 20
      endif

      if (il.eq.1) then
       write (6,*) 'Please enter name of command file:'
       read (5,*) fname
       inquire (file=fname,exist=exists)
       if (.not. exists) then
         call clearscreen
         write (6,*) '===ERROR!==========================================='
         write (6,*) 'File: ',fname
         write (6,*) '  This could not be found, please try again.'
         write (6,*) '============================================='
         write (6,*)
         write (6,*) '-Press return to continue-'
         read (5,*)
         goto 20
       endif
       call clearscreen
       write (6,*) 'Found: ',fname
c--currently enforces type 1 query
       open (2,file='.tmqver')
       write (2,'(a1)') '1'
       close (2)
      endif

      if (il.eq.9) call exit (2)

C**** Specify program constants ****

      filename(1:37)='/home/gps_06/geo6odw/progs/batchproc/'
      filename(38:)='control/batproc.con'
      open (1,file=filename)

      channel=1
      call fileheader (channel)

      read (1,'(6x,i3)') nage
      read (1,'(6x,i3)') nagg
      read (1,'(6x,i3)') ncat
      read (1,'(6x,i3)') nmax
      read (1,'(6x,i3)') ntim
      read (1,'(6x,i3)') nyrt

      close (1)

C**** Get process number and fix path flag (1=local, 0 otherwise)

      open(1,file='.pid')
      read (1,'(a6)') pno
      close (1)

      open (1,file='.path')
      read (1,'(a2)') ptst
      close (1)
      i=system ('/bin/rm .path')
      if (ptst.eq.'./') control(28)=1

C**** Main section of program ****
C** (Subroutine constraints always in alphabetical order) **

      call readtabs (ages,aggfile,datafiles,dfstatus,labels,lablen,
     * nage,nagg,ncat,nyrt,stats,vars,years)

      call interpbat (ages,aggfile,control,datafiles,dfstatus,
     * fname,igr,keyfiles,labels,logcat,nage,nagg,ncat,nmax,nyrt,
     * stats,textout,vars,years)

      call batoutput (control,datafiles,igr,keyfiles,nmax,pno,textout)

      stop
      end
```

Program batproc.f:

This sets up and runs the batch processor. It also contains the menu shell.

B3.2 Listing of readtabs.f

```
C*****************************************************************
C**** Subroutine to read in tables etc.                         *
C*****************************************************************

      subroutine readtabs (ages,aggfile,datafiles,dfstatus,
     * labels,lablen,nage,nagg,ncat,nyrt,stats,vars,years)

      character*60 filename
      character*37 stem
      character*24 stats(5)
      character*14 labels(ncat)
      character*12 vars(5),datafiles(80)
      character*8 aggfile(nagg),ages(nage)
      character*4 years(nyrt)
      integer lablen(ncat),dfstatus(80)

c--stem will change when progs are installed on geo6lib!
c--Read aggfiles etc anyway: Can safely ignore if running for timmig1

      channel=1
      stem='/home/gps_06/geo6odw/progs/batchproc/'
      filename(1:37)=stem

      filename(38:)='control/aggfiles.lut'
      open (1,file=filename)
      call fileheader (channel)
      do 10 n=1,nagg
 10      read (1,'(a8)') aggfile(n)
      close (1)

      filename(38:)='control/agefiles.lut'
      open (1,file=filename)
      call fileheader (channel)
      do 20 n=1,nage
 20      read (1,'(a8)') ages(n)
      close (1)


      filename(38:)='control/yearstem.lut'
      open (1,file=filename)
      call fileheader (channel)
      do 30 n=1,nyrt
 30      read (1,'(a4)') years(n)
      close (1)

      filename(38:)='control/batproc.cat'
      open (1,file=filename)
      call fileheader (channel)
      do 40 n=1,ncat
 40      read (1,'(a14,i2)') labels(n),lablen(n)
      close (1)

      filename(38:)='control/statsetc.lut'
      open (1,file=filename)
      call fileheader (channel)
      do 50 n=1,5
 50      read (1,'(a12,a24)') vars(n),stats(n)
      close (1)

      filename(38:)='control/datafile.lut'
      open (1,file=filename)
      call fileheader (channel)
      do 60 n=1,80
 60      read (1,'(8x,a12,1x,i1)') datafiles(n),dfstatus(n)
      close (1)

      return
      end
```

Subroutine readtabs.f:

This reads in various look up tables etc for use with the batch processor.

B3.3. Listing of interpbat.f

```
C*************************************************************
C**** A subroutine to encode the text strings etc. produced *
C**** by readbat subroutine                                 *
C*************************************************************

      subroutine interpbat (ages,aggfile,control,datafiles,dfstatus,
     * fname,igr,keyfiles,labels,logcat,nage,nagg,ncat,nmax,nyrt,
     * stats,textout,vars,years)

      character*80 textout(4),line
      character*60 filename
      character*36 temptext
      character*24 stats(5)
      character*20 fname
      character*14 labels(ncat)
      character*12 vars(5),keyfiles(2),datafiles(80)
      character*8 aggfile(nagg),ages(nage),userfile
      character*4 years(nyrt)
      integer igr(nmax,2),aggz,control(30)
      integer timetab(21,4),dfstatus(80)
      integer logcat(ncat),varflags(6)

C**** Initialize codes ****

      iversion=0
      icensus=0
      iqstart=0
      ntper=0
      iquarters=0
      iyear1=0
      iyear2=0
      irow1=0
      irow2=0
      icol1=0
      icol2=0
      ilay1=0
      ilay2=0
      istat=0
      iage=0
      iagegr=0
      ioutput=0
      iprinter=0
      igend=0
      allorig=0
      alldest=0
      norig=0
      ndest=0

      do 5 n=1,ncat
 5       logcat(n)=0
      do 6 n=1,6
 6       varflags(n)=0

c---indentify timmig version type

      open (1,file='.tmgver')
      read (1,'(i1)') iversion
      close (1)

C**** Store miscellaneous text
C**** 1: Read date & user

      open (1,file='.who')
      read (1,'(a36)') temptext
      close (1)
      textout(3)(1:5)='User:'
      textout(3)(6:)=temptext
      open(1,file='.dat')
      read (1,'(a36)') temptext
      close (1)
      textout(3)(20:24)='Date:'
      textout(3)(25:)=temptext(1:)

C**** Process file... ****
c---find start indicator-----------------------------------

      open (1,file=fname)
 20   read (1,'(a80)',end=5000) line
      if (line(1:5).ne.'start') goto 20
      write (6,*) 'START FOUND'

c---cycle until end indicator is reached-------------------

 30   read (1,'(a80)',end=5010) line
      if (line(1:1).eq.' '.or.line(1:1).eq.'#') goto 30
      if (line(1:3).eq.'end') then
         close(1)
         write (6,*) 'END FOUND'
         goto 500
      endif
c        write (6,*) line
c----------------------------------------------------------
      if (line(1:5).eq.'title') then
         if (logcat(1).ne.0) then
           call deferr (1,labels,ncat)
         endif
         textout(1)=line(7:)
         logcat(1)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:6).eq.'author') then
         if (logcat(2).ne.0) then
           call deferr (2,labels,ncat)
         endif
         textout(2)=line(8:)
         logcat(2)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:8).eq.'quarters') then
         if (logcat(3).ne.0) then
           call deferr (3,labels,ncat)
         endif
         read (line(10:12),'(i2)') iquarters
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:6).eq.'tstart') then
         if (logcat(4).ne.0) then
           call deferr (4,labels,ncat)
         endif
         if (iversion.eq.1) then
            ni=index(line,'census')
            if (ni.ne.0) then
               icensus=1
               goto 70
            endif
         endif
         do 60 n=1,nyrt
         ni=index(line,years(n))
         if (ni.ne.0) then
            iyear1=n
            if (iversion.eq.2) then
               read (line(ni+5:ni+5),'(i1)') iqstart
               if (iqstart.lt.1.or.iqstart.gt.4) iqstart=0
            endif
         endif
 60      continue
 70      continue
         logcat(4)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:8).eq.'tperiods') then
         if (logcat(5).ne.0) then
           call deferr (5,labels,ncat)
         endif
         read (line(10:12),'(i3)') ntper
         logcat(5)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:5).eq.'scale') then
         if (logcat(6).ne.0) then
           call deferr (6,labels,ncat)
         endif
         ni=index(line,'*')
         if (ni.ne.0) then
            read (line(ni+1:ni+9),'(a8)') userfile
            iscale=nagg+1
            keyfiles(1)(1:8)=userfile
            goto 80
         endif
         do 80 n=1,nagg
            ni=index(line,aggfile(n))
            if (ni.ne.0) then
               iscale=n
               keyfiles(1)(1:8)=aggfile(n)
            endif
 80      continue
         keyfiles(1)(9:12)='.agg'
         logcat(6)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:6).eq.'origin') then
         if (logcat(7).ne.0) then
            call deferr (7,labels,ncat)
         endif
         ni=index(line,'all')
         if (ni.ne.0) then
            allorig=1
            norig=nmax
            do 100 n=1,nmax
               igr(n,1)=n
 100           goto 110
         endif

         call numproc (control,igr,line,nmax,1)

         norig=control(14)

 110     logcat(7)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:11).eq.'destination') then
         if (logcat(8).ne.0) then
            call deferr (8,labels,ncat)
         endif
         ni=index(line,'all')
         if (ni.ne.0) then
            alldest=1
            ndest=nmax
            do 140 n=1,nmax
               igr(n,2)=n
 140           goto 150
         endif

         call numproc(control,igr,line,nmax,2)

         ndest=control(15)

 150     logcat(8)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:6).eq.'output') then
         if (logcat(9).ne.0) then
            call deferr (9,labels,ncat)
         endif
         ni=index(line,'excel')
         if (ni.ne.0) ioutput=3
         ni=index(line,'naked')
         if (ni.ne.0) ioutput=2
         ni=index(line,'table')
         if (ni.ne.0) ioutput=1
         logcat(9)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:8).eq.'filename') then
         if (logcat(10).ne.0) then
            call deferr (10,labels,ncat)
         endif
         textout(4)=line(10:)
         logcat(10)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:4).eq.'row1') then
         if (logcat(11).ne.0) then
            call deferr (11,labels,ncat)
         endif
         do 170 n=1,5
            ni=index(line,vars(n))
            if (ni.ne.0) then
               irow1=n
               if (index(line,'&').ne.0) varflags(1)=1
            endif
 170     continue
         logcat(11)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:4).eq.'row2') then
         if (logcat(12).ne.0) then
            call deferr (12,labels,ncat)
         endif
         do 180 n=1,5
            ni=index(line,vars(n))
            if (ni.ne.0) then
               irow2=n
            endif
 180        continue
         if (irow2.eq.0) goto 30
         logcat(12)=1
         goto 30
      endif
c----------------------------------------------------------
      if (line(1:7).eq.'column1') then
         if (logcat(13).ne.0) then
            call deferr (13,labels,ncat)
         endif
         do 190 n=1,5
            ni=index(line,vars(n))
            if (ni.ne.0) then
               icol1=n
               if (index(line,'&').ne.0) varflags(3)=1
```

```
          endif
 190      continue
          logcat(13)=1
          goto 30
        endif
c-------------------------------------------------------------------------------
        if (line(1:7).eq.'column2') then
          if (logcat(14).ne.0) then
             call deferr (14,labels,ncat)
          endif
          do 200 n=1,5
             ni=index(line,vars(n))
             if (ni.ne.0) then
                icol2=n
             endif
 200      continue
          if (icol2.eq.0) goto 30
          logcat(14)=1
          goto 30
        endif
c-------------------------------------------------------------------------------
        if (line(1:6).eq.'layer1') then
          if (logcat(15).ne.0) then
             call deferr (15,labels,ncat)
          endif
          do 210 n=1,5
             ni=index(line,vars(n))
             if (ni.ne.0) then
                ilay1=n
                if (index(line,'&').ne.0) varflags(5)=1
             endif
 210      continue
          logcat(15)=1
          goto 30
        endif
c-------------------------------------------------------------------------------
        if (line(1:6).eq.'layer2') then
          if (logcat(16).ne.0) then
             call deferr (16,labels,ncat)
          endif
          do 220 n=1,5
             ni=index(line,vars(n))
             if (ni.ne.0) then
                ilay2=n
             endif
 220         continue
             if (ilay2.eq.0) goto 30
          logcat(16)=1
          goto 30
        endif
c-------------------------------------------------------------------------------
        if (line(1:9).eq.'statistic') then
          if (logcat(17).ne.0) then
             call deferr (17,labels,ncat)
          endif
          do 230 n=1,3
             ni=index(line,stats(n))
             if (ni.ne.0) istat=n
 230      continue
          logcat(17)=1
          goto 30
        endif
c-------------------------------------------------------------------------------
        if (line(1:3).eq.'age') then
           if (logcat(18).ne.0) then
              call deferr (18,labels,ncat)
           endif

           ni=index(line,'*')
           if (ni.ne.0) then
              read (line(ni+1:ni+9),'(a8)') userfile
              keyfiles(2)(1:8)=userfile
              iage=nage+1
              goto 250
           endif

           do 240 n=1,nage
              ni=index(line,ages(n))
              if (ni.ne.0) then
                 keyfiles(2)(1:8)=ages(n)
                 iage=n
                 goto 250
              endif
 240       continue
 250       keyfiles(2)(9:12)='.age'
           logcat(18)=1
           goto 30
        endif
c-------------------------------------------------------------------------------
        if (line(1:6).eq.'gender') then
          if (logcat(19).ne.0) then
             call deferr (19,labels,ncat)
          endif
          ni=index(line,'m')
          if (ni.ne.0) then
             control(18)=1
             igend=1
          endif
          ni=index(line,'f')
          if (ni.ne.0) then
             control(19)=1
             igend=1
          endif
          ni=index(line,'u')
          if (ni.ne.0) then
             control(20)=1
             igend=1
          endif
          logcat(19)=1
          goto 30
        endif
c-------------------------------------------------------------------------------
        if (line(1:7).eq.'printer') then
          if (logcat(20).ne.0) then
             call deferr (20,labels,ncat)
          endif
          ni=index(line,'narrow')
          if (ni.ne.0) iprinter=1
          ni=index(line,'wide')
          if (ni.ne.0) iprinter=2
          logcat(20)=1
          goto 30
        endif

c-------------------------------------------------------------------------------
        goto 30

 500    continue
        write (6,*) 'Making validity checks...'

c---reset values for census yr
        if (icensus.eq.1) then
           iyear1=1
           ntper=1
        endif


c---basic checks first, version dependent checks second...

        if (ioutput.eq.0) then
           write (6,*) '*********************************************'
           write (6,*) 'Error!: Output missing or badly defined.'
           write (6,*) 'Action: Defaulting to tabular output.'
           write (6,*) '*********************************************'
           ioutput=1
        endif
        if (iprinter.eq.0) then
           write (6,*) '*********************************************'
           write (6,*) 'Error!: Printer missing or badly defined.'
           write (6,*) 'Action: Defaulting to narrow output.'
           write (6,*) '*********************************************'
           iprinter=1
        endif
        if (iage.eq.0) then
           write (6,*) '*********************************************'
           write (6,*) 'Error!: Age missing or badly defined.'
           write (6,*) 'Action: Defaulting to fiveyear.'
           write (6,*) '*********************************************'
           iage=2
           keyfiles(2)='fiveyear.age'
        endif
        if (istat.eq.0) then
           write (6,*) '*********************************************'
           write (6,*) 'Error!: Statistic missing or badly defined.'
           write (6,*) 'Action: Defaulting to net flow.'
           write (6,*) '*********************************************'
           istat=1
        endif
        if (igend.eq.0) then
           write (6,*) '*********************************************'
           write (6,*) 'Error!:Gender missing or badly defined.'
           write (6,*) 'Action:Defaulting to male, female & unstated.'
           write (6,*) '*********************************************'
           control(18)=1
           control(19)=1
           control(20)=1
        endif
        if (iscale.eq.0) goto 5060
        if (irow1.eq.0.or.icol1.eq.0.or.ilay1.eq.0) goto 5090
        if (norig.eq.0.and.ndest.eq.0) goto 5100

        if (iversion.eq.1) then

           if (iquarters.ne.4) then
              write (6,*) '*********************************************'
              write (6,*) 'Error!:Quarters not set to 4 in timmig1.'
              write (6,*) 'Action:Resetting to quarters=4.'
              write (6,*) '*********************************************'
              iquarters=4
           endif

           if (iscale.eq.1) then
              write (6,*) '*********************************************'
              write (6,*) 'Error!: Full FHSA/UK mode not currently'
              write (6,*) '        available with Timmig MKI'
              write (6,*) 'Action: Resetting to FHSA/EW mode.'
              write (6,*) '*********************************************'
              iscale=2
              keyfiles(1)='fhsa_ews.agg'
           endif

           if (iyear1.eq.0) goto 5030

           iqstart=3

        endif

        if (iversion.eq.2) then

           if (iquarters.lt.1.or.iquarters.gt.4) then
              write (6,*) '*********************************************'
              write (6,*) 'Error!: Quarters missing or badly defined.'
              write (6,*) 'Action: Defaulting to quarters=4.'
              write (6,*) '*********************************************'
              iquarters=4
           endif

           if (iyear1.eq.0.or.iqstart.eq.0) goto 5030

        endif


C**** Check that zones selected are within bounds for scale selected

        if (iscale.eq.nagg+1) then
          filename=keyfiles(1)
        else
          filename ='/home/qps_06/geo6odw/progs/batchproc/general/'
     *        //keyfiles(1)
        endif
        open(1,file=filename,status='old')
        channel=1
        call fileheader(channel)
        read (1,'(i3)') aggz
        close (1)

        if (allorig.eq.1) norig=aggz
        if (alldest.eq.1) ndest=aggz

        do 510 n=1,norig
 510       if (igr(n,1).gt.aggz) goto 5150

        do 520 n=1,ndest
 520       if (igr(n,2).gt.aggz) goto 5150


C**** Check for zone repetition ****

 530    continue
        do 560 group=1,2
         if (group.eq.1) imax=norig
         if (group.eq.2) imax=ndest
         do 560 n1=1,imax-1
          itmp1=igr(n1,group)
          do 560 n2=n1+1,imax
           itmp2=igr(n2,group)
           if (itmp1.eq.itmp2) then
              write (6,'(a5,i3,a10)') 'Zone ',itmp1,' repeated.'
              write (6,*) ' Removing subsequent occurrence(s)...'
              do 550 n3=n2+1,imax
 550           igr(n3-1,group)=igr(n3,group)
              imax=imax-1
              if (group.eq.1) norig=norig-1
              if (group.eq.2) ndest=ndest-1
           endif
 560    continue

C**** Check time periods ****
c--first, build a reference table...
        ntper2=ntper
        n1=1
        n2=3
        i1=1
 570    timetab(n1,n2)=i1
        i1=i1+1
        n2=n2+1
        if (n2.eq.5) then
          n2=1
          n1=n1+1
        endif
        if (n1.lt.22) goto 570

        if (iversion.eq.1) then
           iyrlchk=timetab(iyear1,3)
           if (iyear1+ntper.gt.nyrt) then
              ntper=nyrt-iyear1
           endif
        endif

        if (iversion.eq.2) then
```

```
      iyrlchk=timetab(iyear1,iqstart)
      if (iyrlchk.eq.0) goto 5020

      if (dfstatus(iyrlchk).ne.1) goto 5020

 580  iqtot=ntper*iquarters
      itest=iyrlchk+iqtot
      if (itest.gt.80) then
         if (ntper.eq.1) goto 5030
         ntper=ntper-1
         goto 580
      endif
      if (dfstatus(itest).eq.0) then
         if (ntper.eq.1) goto 5030
         ntper=ntper-1
         goto 580
      endif
   endif
   if (ntper.ne.ntper2) then
      write (6,*)
      write (6,*) '*****************************************'
      write (6,*) 'Error!: Too many time periods defined.'
      write (6,*) 'Action: No. of periods reduced to ',ntper
      write (6,*) '*****************************************'
      write (6,*)
   endif


C**** Check applicability of variables ****

C**** 1a: Check no vars are repeated (General filter)

   control(6)=irow1
   control(7)=irow2
   control(8)=icol1
   control(9)=icol2
   control(10)=ilay1
   control(11)=ilay2

   do 600 n=6,10
     do 600 n2=n+1,11
 600    if (control(n).eq.control(n2).and.control(n).ne.0) goto 5160

C**** 1b: Check table isn't invalid for .tml (Specific filter)

   if (iversion.eq.1) then
      flagag=0
      flaggn=0
      flagor=0
      flagds=0
      flagas=0
      do 610 n=6,11
         if (control(n).eq.1) flagor=1
         if (control(n).eq.2) flagds=1
         if (control(n).eq.4) flagag=1
         if (control(n).eq.5) flaggn=1
 610     continue
      if (flaggn.eq.1.or.flagag.eq.1) flagas=1
      if (flagor.eq.1.and.flagds.eq.1.and.flagas.eq.1) then
         goto 5050
      endif
      if (control(20).eq.1.and.flaggn.eq.1) then
          write (6,*) '*****************************************'
          write (6,*) 'Error!: Gender un-stated defined as a table'
          write (6,*) '        dimension in type 1 query.'
          write (6,*) 'Action: Ignoring.'
          write (6,*) '*****************************************'
          control(20)=0
       endif
   endif

C**** Add 10 to variable values for any (of r1,c1,l1) flagged with &
C--  and automatically ignore the secondary variable

   do 620 n=1,5,2
      if (varflags(n).eq.1) then
         control(n+5)=control(n+5)+10
         control(n+6)=0
      endif
 620  continue

C**** 2: Check age settings

   if (iage.eq.nage+1) then
      filename=keyfiles(2)
   else
      filename ='/home/gps_06/geo6odw/progs/batchproc/general/'
     *        //keyfiles(2)
   endif
   open(1,file=filename,status='old')
   channel=1
   call fileheader(channel)
   read (1,'(i3)') iagegr
C---- have to check age groups fully for version 1
   if (iversion.eq.1) iagegr=16
   close (1)

C**** Store control codes ****

   control(1)=iquarters
   control(2)=iyear1
   control(3)=iqstart
   control(4)=ioutput
   control(5)=istat
C   control(6)=irow1
C   control(7)=irow2
C   control(8)=icol1
C   control(9)=icol2
C   control(10)=ilay1
C   control(11)=ilay2
   control(12)=iagegr
   control(14)=norig
   control(15)=ndest
   control(16)=aggz
   control(17)=ntper
C   control(18) - male
C   control(19) - female
C   control(20) - not stated
   control(21)=iprinter
   control(22)=iyrlchk
   control(24)=icensus
C   control(28) - path flag (1=local)
   control(29)=nyrt
   control(30)=nmax

   write (6,*) 'All categories processed OK.'

   return

C**** Error messages etc. ****

 5000 write (6,*) 'Failed: Could not find start indicator.'
   call exit (3)

 5010 write (6,*) 'Failed: End indicator was not present.'
   call exit (3)

 5020 write (6,*) 'Failed: <look-up fail type 1 for iyear1,iqstart>'
   call exit (3)

 5030 write (6,*) 'Failed: tstart was missing or badly defined.'
   call exit (3)

 5040 write (6,*) 'Failed: Selected time parameters do not allow for'
   write (6,*) '        even one period within data available.'
   call exit (3)

 5050 write (6,*) 'Failed: A table displaying origins by destinations'
   write (6,*) '        by age and/or gender was chosen. This is'
   write (6,*) '        invalid under the .tml mode.'
   call exit (3)

 5060 write (6,*) 'Failed: Scale missing or badly defined.'
   call exit (3)

 5090 write (6,*) 'Failed: Key variable [row1, column1 or layer1]'
   write (6,*) '        missing or badly defined.'
   call exit (3)

 5100 write (6,*) 'Failed: Origins and/or destinations not defined.'
   call exit (3)

 5120 write (6,*) 'Failed: Statistic missing or badly defined.'
   call exit (3)

 5130 write (6,*) 'Failed: Age missing or badly defined.'
   call exit (3)

 5140 write (6,*) 'Failed: Gender missing or basly defined.'
   call exit (3)

 5150 write (6,*) 'Failed: Zone number is too high for chosen scale.'
   call exit (3)

 5160 write (6,*) 'Failed: Two or more variables have been defined'
   write (6,*) '        as the same.'
   call exit (3)

   end

C---------------------------------------------------------------------

   subroutine deferr (i,labels,ncat)

   character*14 labels(ncat)

   write (6,*) '===ERROR!=======================================/'
   write (6,*) 'Duplication of category: ',labels(i)
   write (6,*) '-------------------------------------------------'
   write (6,*) 'Prior case has been overridden.'
   write (6,*) 'Program continuing...'
   write (6,*) '=================================================='
   return
   end
```

```
      subroutine numproc (control,igr,line,nmax,igroup)

      character*80 line
      character*4 number
      character*1 test
      integer igr(nmax,2),control(30)
c      write (6,*) 'Processing numbers for group',igroup
      num1=0
      num2=0
      ngrp=0
      nf1=0
      nf2=1
      if (igroup.eq.1) then
         ntst=8
         ipoint=14
      endif
      if (igroup.eq.2) then
         ntst=13
         ipoint=15
      endif
 10   continue
      if (ntst.gt.80) then
         ntst=1
         read (1,'(a80)')line
      endif
      test=line(ntst:ntst)
      if (test.eq.' '.and.nf1.eq.0) then
         ntst=ntst+1
         goto 10
      endif

      if (test.ge.'0'.and.test.le.'9') then
         nf1=nf1+1
         number(nf1:nf1)=test
         ntst=ntst+1
         goto 10
      endif

      if (test.eq.'-') then
         nf2=nf2+1
         if (nf2.gt.2) goto 100
         read (number(1:nf1),*) num1
         nf1=0
         ntst=ntst+1
         goto 10
      endif

      if (test.eq.' '.or.test.eq.','.or.test.eq.';') then
         if (nf2.eq.1) then
            ngrp=ngrp+1
            read (number(1:nf1),*) num
            igr(ngrp,igroup)=num
            nf1=0
         endif
         if (nf2.eq.2) then
            read (number(1:nf1),*) num2
            if (num1.gt.num2) goto 100
            do 20 numstar=num1,num2
               ngrp=ngrp+1
20             igr(ngrp,igroup)=numstar
            nf1=0
            nf2=1
         endif
         if (test.eq.';') then
            goto 30
         else
            ntst=ntst+1
            goto 10
         endif
      endif

      goto 110

30    control(ipoint)=ngrp
      return

100   write (6,*) 'Failed: Zone-number processing error.'
      if (igroup.eq.1) write (6,*) 'Origin group:'
      if (igroup.eq.2) write (6,*) 'Destination group:'
      if (num1.gt.num2) then
         write (6,'(a10,i3,a4,i3,a12)') 'The range ',num1,'to',num2,
     *        ' is illegal.'
         call exit (3)
      endif
      if (nf2.gt.2) then
         write (6,*) 'Too many parameters defined for a range.'
         call exit (3)
      endif
      if (ngrp.gt.nmax) then
         write (6,*) 'Too many zones defined.'
         call exit (3)
      endif
      write (6,*) 'Unclassifiable syntax error.'
      call exit (3)

110   write (6,*) 'Failed: Illegal character in number list: ',test
      call exit (3)

      end
```

Subroutine interpbat.f

The main engine of batchproc. There are two main parts to the main routine; firstly, processing the command file to extract and decode the key terms, and secondly, validating the command file instructions. Note that there are two additional subroutines included in *interpbat.f*. A small routine *deferr*, is used to generate error warnings. The second subroutine, *numproc*, is used to decode the number lists extracted from the origin and destination instructions.

B5.4 Listing of batoutput.f

```
C*************************************************************
C**** A subroutine to display the processed / encoded       *
C**** batch file                                            *
C*************************************************************

      subroutine batoutput (control,datafiles,igr,keyfiles,
     * nmax,pno,textout)

      character*80 textout(4)
      character*36 filename(5)
      character*12 keyfiles(2),datafiles(80)
      character*6 pno
      character*5 path
      integer control(30),igr(nmax,2)

c---set path string

      if (control(28).eq.0) then
         path='/tmp/'
         ipl=5
      else
         path='./   '
         ipl=2
      endif

c---write data ordering file..

      open (2,file='.orderbook')
      istart=control(22)
      itotal=0
      do 20 n1=1,control(17)
         iend=istart+(control(1)-1)
         do 10 n2=istart,iend
            itotal=itotal+1
c            write (6,'(i3,a12,1x,i2)') n2,datafiles(n2),n1
 10         write (2,'(a12,1x,i2)') datafiles(n2),n1
 20         istart=istart+control(1)
      close(2)

      control(23)=itotal

C**** Write other files

      filename(1)='.workfiles'
      open (2,file=filename(1))
      write (2,'(a12)') (keyfiles(n),n=1,2)
      close (2)

      filename(2)=path(1:ipl)//'misctext.txt.'//pno
      open (2,file=filename(2))
      write (2,'(a80)') (textout(n),n=1,4)
      close (2)

      filename(3)='.controls'
      open (2,file=filename(3))
      write (2,'(10i5)') (control(n),n=1,30)
      close (2)

      filename(4)=path(1:ipl)//'origingr.arr.'//pno
      open (2,file=filename(4))
      write (2,'(16i4)') (igr(n,1),n=1,control(14))
      close (2)

      filename(5)=path(1:ipl)//'destingr.arr.'//pno
      open (2,file=filename(5))
      write (2,'(16i4)') (igr(n,2),n=1,control(15))
      close (2)

      return
      end
```

subroutine batoutput.f:

This produces outptu from the batch processor which is passed to subsequent programs. The key outputs are the control array, containing a variety of variables, and the two number arrays origingr and destingr.

Appendix B4: Listing of arrayprep.f

```
C**** program to determine size of array required by gentab
C**** and to prepare the necessary FORTRAN lines

      program arrayprep

      integer control(30)

C**** Read controls array

      open (1,file='.controls')
      read (1,'(10i5)'),(control(n),n=1,30)
      close(1)

      imr1=1
      imr2=1
      imc1=1
      imc2=1
      iml1=1
      iml2=1

      irow1=control(6)
      irow2=control(7)
      icol1=control(8)
      icol2=control(9)
      ilay1=control(10)
      ilay2=control(11)
      iage=control(12)
      ingr1=control(14)
      ingr2=control(15)
      iaggz=control(16)
      itime=control(17)
      isex=control(18)+control(19)+control(20)

      if (irow1.eq.1) imr1=ingr1
      if (irow2.eq.1) imr2=ingr1
      if (icol1.eq.1) imc1=ingr1
      if (icol2.eq.1) imc2=ingr1
      if (ilay1.eq.1) iml1=ingr1
      if (ilay2.eq.1) iml2=ingr1

      if (irow1.eq.2) imr1=ingr2
      if (irow2.eq.2) imr2=ingr2
      if (icol1.eq.2) imc1=ingr2
      if (icol2.eq.2) imc2=ingr2
      if (ilay1.eq.2) iml1=ingr2
      if (ilay2.eq.2) iml2=ingr2

      if (irow1.eq.3) imr1=itime
      if (irow2.eq.3) imr2=itime
      if (icol1.eq.3) imc1=itime
      if (icol2.eq.3) imc2=itime
      if (ilay1.eq.3) iml1=itime
      if (ilay2.eq.3) iml2=itime

      if (irow1.eq.4) imr1=iage
      if (irow2.eq.4) imr2=iage
      if (icol1.eq.4) imc1=iage
      if (icol2.eq.4) imc2=iage
      if (ilay1.eq.4) iml1=iage
      if (ilay2.eq.4) iml2=iage

      if (irow1.eq.5) imr1=isex
      if (irow2.eq.5) imr2=isex
      if (icol1.eq.5) imc1=isex
      if (icol2.eq.5) imc2=isex
      if (ilay1.eq.5) iml1=isex
      if (ilay2.eq.5) iml2=isex

c---allow 1 additional row, col and layer for totals

      irtot=(imr1*imr2)+1
      ictot=(imc1*imc2)+1
      iltot=(iml1*iml2)+1

c---adjust for any vars flagged with '&' -

      if (irow1.gt.5) irtot=1
      if (icol1.gt.5) ictot=1
      if (ilay1.gt.5) iltot=1

      open (2,file='.params')

      write (2,'(6x,a22)') 'character*24 rlab(125)'
      write (2,'(6x,a19,i3,a1)') 'character*24 grlab(',iaggz,')'
      write (2,'(6x,a19,i5,a8,i5,a1)') 'character*24 rlabs(',
     * irtot,'),llabs(',iltot,')'

      icno=1
      if (icol2.gt.0) icno=2
      write (2,'(6x,a19,i5,a3)') 'character*24 clong(',ictot,',2)'
      write (2,'(6x,a19,i5,a1,i1,a1)') 'character*10 clabs(',ictot,
     * ',',icno,')'
      write (2,'(6x,a21,i3,a1)') 'character*12 agelabs(',iage,')'
      write (2,'(6x,a17,i5,a1,i5,a1,i5,a1)') 'integer outarray(',
     * irtot,',',ictot,',',iltot,')'

      write (2,'(6x,a12,i3,a3)') 'integer igr(',iaggz,',2)'
      write (2,'(6x,a19)') 'integer control(30)'
      write (2,'(6x,a19)') 'integer agemap(101)'
      write (2,'(6x,a18)') 'integer znum (125)'
      write (2,'(6x,a18)') 'integer dflut (80)'
      write (2,'(6x,a5,i5)') 'nrow=',irtot
      write (2,'(6x,a5,i5)') 'ncol=',ictot
      write (2,'(6x,a5,i5)') 'nlay=',iltot
      write (2,'(6x,a7,i1)') 'ncolab=',icno
      write (2,'(6x,a5,i3)') 'nagz=',iaggz
      write (2,'(6x,a5,i2)') 'nyrt=',control(29)
      write (2,'(6x,a5,i3)') 'nmax=',control(30)

      close(2)

      stop
      end
```

Program arrayprep.f:

This reads in the contol array produced by *batoutput.f* (see B3.4), and uses it to calculate the size of the array required to print the table specified in the comand file. The program then writes an output file which comprises a series of FORTRAN statements, which are then compiled as part of the *gentab* control routine (see B5).

Appendix B5: Listing of gentab

B5.1 *gen01.f* and *gen02.f*

```
C*******************************
C**** Prog amalgam test
C*******************************

      program gentab
      character*80 headertxt(3)
      character*48 aggtitle
      character*24 vlabs(5)
      character*12 datafiles(80)
      character*6 pno
```

```
C** Read environment variables etc.

      open (1,file='.pid')
      read (1,'(a6)') pno
      close (1)

      open (1,file='.tmgver')
      read (1,'(i1)') iversion
      close (1)

      call gread (agemap,aggtitle,control,datafiles,dflut,
     *     grlab,headertxt,igr,nagz,nmax,pno,zlab,znum)


      call glabels (agelabs,agemap,clabs,clong,control,datafiles,
     *     dflut,grlab,igr,llabs,nagz,ncol,ncolab,nlay,nrow,
     *     rlabs,vlabs)


      if (iversion.eq.1) then
         call gcountv1 (agemap,control,datafiles,dflut,igr,nagz,ncol,
     *     nlay,nmax,nrow,nyrt,outarray,znum)
      else
         call gcountv2 (agemap,control,datafiles,dflut,igr,nagz,ncol,
     *     nlay,nmax,nrow,outarray,znum)
      endif

c--indexing etc. functions to go here if selected


      call goutput (clabs,control,headertxt,llabs,ncol,ncolab,
     *     nlay,nrow,outarray,rlabs,vlabs)

      stop
      end
```

Program gentab:

The tabulation part of TIMMIG. The two portions of code in this section, gen01.f (left) and gen02.f (right) are joined together, with the text file produced by arraydet sandwiched in between, in order to form the gentab module. This module must be complied at run-time.

B5.2 Listing of gread.f

```
C*************************************************************
C** A subroutine to read in variable data etc.              *
C*************************************************************

      subroutine gread (agemap,aggtitle,control,datafiles,dflut,
     * grlab,headertxt,igr,nagz,nmax,pno,zlab,znum)

      character*80 outfile,headertxt(3)
      character*48 aggtitle
      character*36 filename
      character*24 zlab(nmax),grlab(nagz)
      character*12 workfiles(2),datafiles(80)
      character*6 pno
      character*5 path

      integer agemap(101),control(30),igr(nagz,2)
      integer naggz,znum(nmax),dflut(80)
      integer convert(97)

      data convert /1,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,
     *     33,34,35,36,37,38,39,40,41,42,43,44,45,46,64,65,66,67,68,69,
     *     70,71,72,73,74,75,47,48,49,50,51,52,53,54,55,56,57,62,76,
     *     77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,
     *     97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,
     *     112,113,114,115,116,117/

c***fix path string

      open(1,file='.controls')
      read (1,'(1015)') (control(n),n=1,30)
      close (1)

      if (control(28).eq.0) then
         path='/tmp/'
         ipl=5
      else
         path='./   '
         ipl=2
      endif

C** Read workfiles file and then read variable files

      open (1,file='.workfiles')
      read (1,'(a12)') (workfiles(n),n=1,2)
      close (1)

C** Line 1 of workfiles will be .agg file

      channel=1
      filename=path(1:ipl)//workfiles(1)//'.'//pno
      open (1,file=filename)
      call fileheader (channel)

c** no. of regions; title; data
      read (1,'(i3,a48)') naggz,aggtitle
      do 20 n=1,nmax
 20      read (1,'(a24,i3)') zlab(n),znum(n)
      call fileheader (channel)
      do 30 n=1,nagz
 30      read (1,'(a24)') grlab(n)
      close (1)


C** Line 2 of workfiles will be .age file

      filename=path(1:ipl)//workfiles(2)//'.'//pno
      open (1,file=filename)
      call fileheader (channel)
      read (1,'(i3)') iagegr
      read (1,'(8(4x,i4))') (agemap(n),n=1,101)
      close (1)

C** Read standard files


      filename=path(1:ipl)//'origingr.arr.'//pno
      open (1,file=filename)
      read (1,'(16i4)')(igr(n,1),n=1,control(14))
      close(1)

      filename=path(1:ipl)//'destingr.arr.'//pno
      open(1,file=filename)
      read (1,'(16i4)')(igr(n,2),n=1,control(15))
      close (1)

      filename=path(1:ipl)//'misctext.txt.'//pno
      open(1,file=filename)
      read(1,'(a80)') (headertxt(n),n=1,3)
      read(1,'(a80)') outfile
      close(1)

c--outfile is the name of the output file, open it now
      open(12,file=outfile,form='print')

      open(1,file='.orderbook')
      do 40 n=1,control(23)
 40      read (1,'(a12,1x,i2)') datafiles(n),dflut(n)
      close(1)

c--the .orderbook still needs to be read for timmig1 mode in order
c--to use it for label building


      return

      end
```

Subroutine gread.f:

This is used to read in various look up tables. The tables to read are dictated by the file .workfiles, which is produced by batoutput.f and reflects the choices made in the caommand file. Files will be read from either the local directory or /tmp, depending on the path control variable, which is originally set up in the TIMMIG command script.

B5.3 Listing of glabels.f

```
C*************************************************************
C** A subroutine to build labels arrays for gentab          *
C*************************************************************

      subroutine glabels (agelabs,agemap,clabs,clong,control,datafiles,
     * dflut,grlab,igr,llabs,nagz,ncol,ncolab,nlay,nrow,rlabs,
     * vlabs)

      character*24 grlab (nagz),varlabsin(8),vlabs(5)
      character*24 rlabs(nrow),llabs(nlay)
      character*24 clong(ncol,2),testlab
      character*10 clabs(ncol,ncolab)
      character*12 yearlabs(84),sexlab(3),datafiles(80)
      character*12 agelabs(control(12))
      character*1 divider

      integer control(30),igr(nagz,2),isex(3),agemap(101),dflut(80)
      integer lnblnk

      common /testlabel/ testlab

      divider=':'
      if (control(4).eq.3) divider=';'
c     write (6,*) 'Creating table labels...'

C** Initialise time labels

      istart=1
      do 10 n=1,control(17)
         if (control(1).eq.1) then
          yearlabs(n)='19'//datafiles(n)(5:8)
         else
          iend=istart+(control(1)-1)
          yearlabs(n)=datafiles(istart)(5:8)//'-'//datafiles(iend)(5:8)
          istart=istart+control(1)
         endif
 10   continue
      if (control(24).eq.1) yearlabs(1)='Census  '

C** Initialise gender labels

      sexlab(1)='Male'
      sexlab(2)='Female'
      sexlab(3)='Not-stated'
      do 45 n=1,3
 45     isex(n)=0
      isx=0
      if (control(18).eq.1) then
         insex=insex+1
         isex(insex)=1
      endif
      if (control(19).eq.1) then
         insex=insex+1
         isex(insex)=2
      endif
      if (control(20).eq.1) then
         insex=insex+1
         isex(insex)=3
      endif

C** Initialise age-group labels

c--find start

      iage=control(12)
      do 50 n=1,101
         il=agemap(n)
 50      if (il.eq.1) goto 55
 55   continue

      iagegr=1
      istart=n
      itest=n
 60   if (iagegr.gt.iage) goto 70
      il=agemap(istart)
      if (iagegr.eq.iage) then
         write (agelabs(iage),'(i3,a1)') istart-1,'+'
         goto 70
      endif
      i2=agemap(itest+1)
      if (il.ne.i2) then
         if (istart.eq.itest) then
             write (agelabs(iagegr),'(i3)') istart-1
             iagegr=iagegr+1
            istart=istart+1
            itest=istart
            goto 60
         endif

         if (istart.ne.itest) then
            write (agelabs(iagegr),'(i3,a2,i3)') istart-1,' -',itest-1
            iagegr=iagegr+1
            istart=itest+1
            itest=istart
            goto 60
         endif
      endif
      itest=itest+1
      goto 60

 70   continue

C** Set up row, column and layer labels

      varlabsin(1)='Origins'
      varlabsin(2)='Destinations'
      varlabsin(3)='Time'
      varlabsin(4)='Age'
      varlabsin(5)='Gender'
      varlabsin(6)='Net migration'
      varlabsin(7)='Out migration'
      varlabsin(8)='Return migration'

      do 80 n=1,3
         itarg=(2*n)+4
         if (control(itarg).gt.5) then
           vlabs(n)='Tot. '//varlabsin(control(itarg)-10)(1:19)
         else
            vlabs(n)=varlabsin(control(itarg))
         endif
         if (control(itarg+1).ne.0) then
            vlabs(n)(11:14)=' by '
            vlabs(n)(15:24)=varlabsin(control(itarg+1))(1:10)
         endif
 80   continue
      vlabs(4)=varlabsin(control(5)+5)
c--row labels

      rlabs(nrow)='Totals'

      if (control(6).eq.1) ival1=control(14)
      if (control(6).eq.2) ival1=control(15)
      if (control(6).eq.3) ival1=control(17)
      if (control(6).eq.4) ival1=iage
      if (control(6).eq.5) ival1=insex
      if (control(7).eq.0) ival2=1
      if (control(7).eq.1) ival2=control(14)
      if (control(7).eq.2) ival2=control(15)
      if (control(7).eq.3) ival2=control(17)
      if (control(7).eq.4) ival2=iage
      if (control(7).eq.5) ival2=insex

      n=0
      do 150 n1=1,ival1
      do 150 n2=1,ival2
        n=n+1
         if (control(6).eq.1) then
           rlabs(n)=grlab(igr(n1,1))
         endif
         if (control(6).eq.2) then
           rlabs(n)=grlab(igr(n1,2))
         endif
         if (control(6).eq.3) then
          rlabs(n)=yearlabs(n1)
         endif
         if (control(6).eq.4) then
            rlabs(n)=agelabs(n1)
         endif
         if (control(6).eq.5) then
            rlabs(n)=sexlab(isex(n1))
         endif
150   continue

      if (control(7).ne.0) then
        n=0
        do 160 n1=1,ival1
        do 160 n2=1,ival2
        n=n+1
         if (control(7).eq.1) then
            rlabs(n)(12:12)=':'
            rlabs(n)(13:23)=grlab(igr(n2,1))(1:11)
         endif
         if (control(7).eq.2) then
            rlabs(n)(12:12)=':'
            rlabs(n)(13:23)=grlab(igr(n2,2))(1:11)
         endif
         if (control(7).eq.3) then
            rlabs(n)(12:12)=':'
            rlabs(n)(13:23)=yearlabs(n2)(1:11)
         endif
         if (control(7).eq.4) then
            rlabs(n)(12:12)=':'
            rlabs(n)(13:23)=agelabs(n2)(1:11)
         endif
         if (control(7).eq.5) then
            rlabs(n)(12:12)=':'
            rlabs(n)(13:23)=sexlab(isex(n2))(1:11)
         endif
160     continue
      endif

c--column labels

      clong(ncol,1)='Totals'

      if (control(8).eq.1) ival1=control(14)
      if (control(8).eq.2) ival1=control(15)
      if (control(8).eq.3) ival1=control(17)
      if (control(8).eq.4) ival1=iage
      if (control(8).eq.5) ival1=insex
      if (control(9).eq.0) ival2=1
      if (control(9).eq.1) ival2=control(14)
      if (control(9).eq.2) ival2=control(15)
      if (control(9).eq.3) ival2=control(17)
      if (control(9).eq.4) ival2=iage
      if (control(9).eq.5) ival2=insex

      n=0
      do 170 n1=1,ival1
      do 170 n2=1,ival2
        n=n+1
        if (control(8).eq.1) then
          clong(n,1)=grlab(igr(n1,1))
         endif
         if (control(8).eq.2) then
            clong(n,1)=grlab(igr(n1,2))
         endif
         if (control(8).eq.3) then
            clong(n,1)=yearlabs(n1)
         endif
         if (control(8).eq.4) then
            clong(n,1)=agelabs(n1)
         endif
         if (control(8).eq.5) then
            clong(n,1)=sexlab(isex(n1))
         endif
170   continue

      if (control(9).ne.0) then
        n=0
        do 180 n1=1,ival1
        do 180 n2=1,ival2
        n=n+1
         if (control(9).eq.1) then
            clong(n,2)=grlab(igr(n2,1))
         endif
         if (control(9).eq.2) then
            clong(n,2)=grlab(igr(n2,2))
         endif
         if (control(9).eq.3) then
            clong(n,2)=yearlabs(n2)
         endif
         if (control(9).eq.4) then
            clong(n,2)=agelabs(n2)
         endif
         if (control(9).eq.5) then
            clong(n,2)=sexlab(isex(n2))
         endif
180     continue
      endif

c--layer labels

      llabs(nlay)='Totals'

      if (control(10).eq.1) ival1=control(14)
      if (control(10).eq.2) ival1=control(15)
      if (control(10).eq.3) ival1=control(17)
      if (control(10).eq.4) ival1=iage
      if (control(10).eq.5) ival1=insex
      if (control(11).eq.0) ival2=1
      if (control(11).eq.1) ival2=control(14)
      if (control(11).eq.2) ival2=control(15)
      if (control(11).eq.3) ival2=control(17)
      if (control(11).eq.4) ival2=iage
      if (control(11).eq.5) ival2=insex

      n=0
      do 190 n1=1,ival1
      do 190 n2=1,ival2
        n=n+1
        if (control(10).eq.1) then
          llabs(n)=grlab(igr(n1,1))
         endif
```

```
         if (control(10).eq.2) then
            llabs(n)=grlab(igr(n1,2))
         endif
         if (control(10).eq.3) then
           llabs(n)=yearlabs(n1)
         endif
         if (control(10).eq.4) then
            llabs(n)=agelabs(n1)
         endif
         if (control(10).eq.5) then
            llabs(n)=sexlab(isex(n1))
         endif
 190   continue

       if (control(11).ne.0) then
         n=0
         do 200 n1=1,ival1
         do 200 n2=1,ival2
         n=n+1
          if (control(11).eq.1) then
             llabs(n)(12:12)=':'
             llabs(n)(13:24)=grlab(igr(n2,1))(1:10)
          endif
          if (control(11).eq.2) then
             llabs(n)(12:12)=':'
             llabs(n)(13:24)=grlab(igr(n2,2))(1:10)
          endif
          if (control(11).eq.3) then
             llabs(n)(12:12)=':'
             llabs(n)(13:24)=yearlabs(n2)(1:10)
          endif
          if (control(11).eq.4) then
             llabs(n)(12:12)=':'
             llabs(n)(13:24)=agelabs(n2)(1:10)
          endif
          if (control(11).eq.5) then
             llabs(n)(12:12)=':'
             llabs(n)(13:24)=sexlab(isex(n2))(1:10)
          endif
 200      continue
       endif

c--- abbreviation of (column) labels

       do 250 n1=1,ncol
          do 250 n2=1,ncolab
             testlab=clong(n1,n2)
             islen=lnblnk(testlab)
             if (islen.ge.10) call abbreviate(islen)

             islen=lnblnk(testlab)
 250         clabs(n1,n2)=testlab(1:8)//divider//' '
             return
             end

c-------------------------------------------------------------------

       subroutine abbreviate (isl)

       character*24 testlab
       character*1 tsl

       common /testlabel/ testlab

       do 20 n=1,isl-1
          if (testlab(n:n).eq.' ') then
             do 10 i=n,isl-1
 10             testlab(i:i)=testlab(i+1:i+1)
             testlab(isl:isl)=' '
             isl=isl-1
             itrim=itrim+1
             if (isl.lt.10) goto 50
          endif
 20    continue
       do 40 n=1,isl-1
          iflag1=0
          tsl=testlab(n:n)
          if (tsl.eq.'a'.or.tsl.eq.'e'.or.tsl.eq.'i') iflag1=1
          if (tsl.eq.'o'.or.tsl.eq.'u') iflag1=1
          if (iflag1.eq.1) then
             do 30 i=n,isl-1
 30             testlab(i:i)=testlab(i+1:i+1)
             testlab(isl:isl)=' '
             isl=isl-1
             if (isl.lt.10) goto 50
          endif
 40    continue
 50    continue


       return
       end
```

Subroutine glabels.f:

Constructs the labels to be used in output for each table dimension. Where necessary, concatenates two labels (for collapsed variables) or abbreviates labels.

B5.4 Listing of gcountv1.f

```
C*************************************************************
C**  A subroutine to count values in outarray
C*************************************************************
      subroutine gcountv1 (agemap,control,datafiles,dflut,
     *     igr,nagz,ncol,nlay,nmax,nrow,nyrt,outarray,znum)
      character*200 systring
      character*12 datafiles(80),filename
      character*8 namestr
      character*6 pidstr
      character*5 path
      character*4 name2
      integer agemap(101)
      integer igr(nagz,2),znum(nmax),dflut(80)
      integer outarray(nrow,ncol,nlay),control(30)
      integer zlut(125,2),isex(3)
      integer movlyr(125,125),mov2yr(125,16,4)
      integer ylut(18),system
      if (control(28).eq.0) then
         path='/tmp/'
         ipl=5
      else
         path='./   '
         ipl=2
      endif
c---blank out array (not necessary?)
      do 10 n1=1,nrow
         do 10 n2=1,ncol
            do 10 n3=1,nlay
 10            outarray(nrow,ncol,nlay)=0
      write (6,*) 'Counting...'

c---establish test contraints etc.
            do 20 i=1,nmax
         do 20 j=1,3
 20      zlut(i,j)=0
      do 30 i=1,control(14)
         i1=igr(i,1)
         do 30 i2=1,nmax
            if (znum(i2).eq.i1) zlut(i2,1)=i
 30   continue
      do 40 i=1,control(15)
         i1=igr(i,2)
         do 40 i2=1,nmax
            if (znum(i2).eq.i1) zlut(i2,2)=i
 40   continue
c       do 45 i1=1,nmax
c 45     write (6,*) i1,zlut(i1,1),zlut(i1,2)
c---pointers for gender variable
      iage=control(12)
      isex(1)=control(18)
      isex(2)=control(19)
      isex(3)=control(20)
      insex=control(18)+control(19)
c--establish pointer multipliers...
      if (control(7).eq.0) irm=1
      if (control(7).eq.1) irm=control(14)
      if (control(7).eq.2) irm=control(15)
      if (control(7).eq.3) irm=control(17)
      if (control(7).eq.4) irm=iage
      if (control(7).eq.5) irm=insex
      if (control(9).eq.0) icm=1
      if (control(9).eq.1) icm=control(14)
      if (control(9).eq.2) icm=control(15)
      if (control(9).eq.3) icm=control(17)
      if (control(9).eq.4) icm=iage
      if (control(9).eq.5) icm=insex
      if (control(11).eq.0) ilm=1
      if (control(11).eq.1) ilm=control(14)
      if (control(11).eq.2) ilm=control(15)
      if (control(11).eq.3) ilm=control(17)
      if (control(11).eq.4) ilm=iage
      if (control(11).eq.5) ilm=insex
c***Read in movl or mov23 files for chosen periods
c---read file will depend on table definition
c---if table is orig*dest use movl
c---if table is orig OR dest * age/sex use mov23
c--asflag determines tabletype
c--odflag determines orig or dest focussed type2
      asflag=0
      odflag=0
      do 50 n=6,11
         if (control(n).eq.4.or.control(n).eq.5) asflag=1
         if (control(n).eq.1) odflag=1
         if (control(n).eq.2) odflag=2
 50   continue
      if (asflag.eq.1) then
         itabtype=2
      else
         itabtype=1
      endif

c---first create a ref table for which years to use...
            do 60 n=1,nyrt
 60      ylut(n)=0
      iyr1=control(2)
      iyr2=control(2)+control(17)-1
      if (control(24).eq.1) then
         iyr1=1
         iyr2=1
         itabtype=itabtype+2
      endif
      do 70 n=iyr1,iyr2
 70      ylut(n)=1

c---read valid years and extract required values
      open (1,file='.pid')
      read (1,'(a6)') pidstr
      close (1)
      itim=0
      do 150 nyr=1,iyr2
       if (ylut(nyr).eq.0) goto 150
       if (control(24).eq.0) write (6,*) 'Processing year: ',1974+nyr
       if (control(24).eq.1) write (6,*) 'Processing Census year'
       nm1=1974+nyr
       nm2=nm1+1
       write (namestr,'(2i4)') nm1,nm2
       read (namestr,'(2x,a2,2x,a2)') name2(1:2),name2(3:4)
       if (itabtype.eq.1) filename='mig1//name2//'.tml'
       if (itabtype.eq.2) filename='mig2//name2//'.tml'
       if (itabtype.eq.3) filename='mig2cnss.tml'
       if (itabtype.eq.4) filename='mig2cnss.tml'
       itim=itim+1
       systring='zcat -geo61ib/oliverdw/archive3/'//filename//
     *      ' > '//path(1:ipl)//'movtml.'//pidstr
       i=system (systring)
       open (1,file=path(1:ipl)//'movtml.'//pidstr)
        if (itabtype.eq.1.or.itabtype.eq.3) then
          do 80 n1=1,125
             read (1,'(16i5)') (movlyr(n1,n2),n2=1,125)
 80        continue
        else
          do 90 n1=1,125
            do 90 ir=1,4
 90            read (1,'(16(i5))') (mov2yr(n1,ia,ir),ia=1,16)
        endif
       close (1)
       systring='/bin/rm  '//path(1:ipl)//'movtml.'//pidstr
       i=system (systring)

       if (itabtype.eq.1.or.itabtype.eq.3) then
         iag=1
         isx=1
         iptr=0
         iptl=0
         iptc=0
         do 100 ior=1,125
           do 100 ids=1,125
             iorig=zlut(ior,1)
             idest=zlut(ids,2)
             if (iorig.eq.0.or.idest.eq.0) goto 100
c             write (6,*) iorig,idest
             call pointers (control,iag,icm,idest,ilm,iorig,iptc,iptl,
     *            iptr,irm,isx,itim,nmax,znum)
             if (control(5).eq.1) ival=(movlyr(ids,ior)-movlyr(ior,ids))
             if (control(5).eq.2) ival=movlyr(ior,ids)
             if (control(5).eq.3) ival=movlyr(ids,ior)
              outarray(iptr,iptc,iptl)=outarray(iptr,iptc,iptl)+ival
c              if (itim.eq.1) then
c                 write (6,*) iorig,ior,idest,ids
c                 write (6,*) iptr,iptc,iptl,ival
c              endif
 100       continue
         else
          iptr=0
          iptl=0
          iptc=0
          do 130 ior=1,125
             izone=zlut(ior,odflag)
                        if (izone.eq.0) goto 130
             if (control(18).eq.1) then
                isx=1
                do 110 iag=1,16
                    call pointers (control,iag,icm,izone,ilm,izone,
     *                   iptc,iptl,iptr,irm,isx,itim,nmax,znum)
                    if (control(5).eq.1) then
                      ival=mov2yr(ior,iag,1)-mov2yr(ior,iag,3)
                    endif
                    if (control(5).eq.2) then
                       ival=mov2yr(ior,iag,3)
                    endif
                    if (control(5).eq.3) then
                       ival=mov2yr(ior,iag,1)
                    endif
 110              outarray(iptr,iptc,iptl)=outarray(iptr,iptc,iptl)+ival
              endif
              if (control(19).eq.1) then
                isx=1
                if (control(18).eq.1) isx=2
                do 120 iag=1,16
                    call pointers (control,iag,icm,izone,ilm,izone,
     *                   iptc,iptl,iptr,irm,isx,itim,nmax,znum)
                    if (control(5).eq.1) then
                       ival=mov2yr(ior,iag,2)-mov2yr(ior,iag,4)
                    endif
                    if (control(5).eq.2) then
                       ival=mov2yr(ior,iag,4)
                    endif
                    if (control(5).eq.3) then
                       ival=mov2yr(ior,iag,2)
                    endif
 120              outarray(iptr,iptc,iptl)=outarray(iptr,iptc,iptl)+ival
              endif
 130          continue

            endif
  150     continue

c-----calc totals
      nrlim=nrow-1
      nclim=ncol-1
      nllim=nlay-1
      if (control(6).gt.5) nrlim=nrow
      if (control(8).gt.5) nclim=ncol
      if (control(10).gt.5) nllim=nlay
      do 180 n1=1,nrlim
       do 170 n2=1,nclim
        do 160 n3=1,nllim
           nx1=1
           nx2=1
           nx3=1
           nx4=1
           nx5=1
           nx6=1
           nx7=1
           if (n1.eq.nrow) nx1=0
           if (n2.eq.ncol) nx2=0
           if (n3.eq.nlay) nx3=0
           if (nx1.eq.0.or.nx2.eq.0) nx4=0
           if (nx1.eq.0.or.nx3.eq.0) nx5=0
           if (nx2.eq.0.or.nx3.eq.0) nx6=0
           if (nx1.eq.0.or.nx2.eq.0.or.nx3.eq.0) nx7=0
           ival=outarray(n1,n2,n3)
           outarray(nrow,n2,n3)=outarray(nrow,n2,n3)+(ival*nx1)
           outarray(n1,ncol,n3)=outarray(n1,ncol,n3)+(ival*nx2)
           outarray(n1,n2,nlay)=outarray(n1,n2,nlay)+(ival*nx3)
           outarray(nrow,ncol,n3)=outarray(nrow,ncol,n3)+(ival*nx1*nx2)
           outarray(nrow,n2,nlay)=outarray(nrow,n2,nlay)+(ival*nx1*nx3)
           outarray(n1,ncol,nlay)=outarray(n1,ncol,nlay)+(ival*nx2*nx3)
           outarray(nrow,ncol,nlay)=outarray(nrow,ncol,nlay)+(ival*nx7)
 160        continue
 170     continue
 180  continue
      close(1)
      return
      end

c-------------------------------------------------------------------
      subroutine pointers (control,iag,icm,ids,ilm,ior,iptc,iptl,iptr,
     *     irm,isx,itim,nmax,znum)
      integer control(30),znum(nmax)
c      write (6,*) iag,icm,ids,ilm,ior,iptc,iptl,iptr,irm,isx,itim
      if (control(6).eq.1) ibasr=ior
      if (control(6).eq.2) ibasr=ids
      if (control(6).eq.3) ibasr=itim
      if (control(6).eq.4) ibasr=iag
      if (control(6).eq.5) ibasr=isx
           if (control(7).eq.0) iaddr=1
      if (control(7).eq.1) iaddr=ior
      if (control(7).eq.2) iaddr=ids
      if (control(7).eq.3) iaddr=itim
      if (control(7).eq.4) iaddr=iag
      if (control(7).eq.5) iaddr=isx
           if (control(8).eq.1) ibasc=ior
      if (control(8).eq.2) ibasc=ids
      if (control(8).eq.3) ibasc=itim
      if (control(8).eq.4) ibasc=iag
      if (control(8).eq.5) ibasc=isx
                if (control(9).eq.0) iaddc=1
      if (control(9).eq.1) iaddc=ior
      if (control(9).eq.2) iaddc=ids
      if (control(9).eq.3) iaddc=itim
      if (control(9).eq.4) iaddc=iag
      if (control(9).eq.5) iaddc=isx
           if (control(10).eq.1) ibasl=ior
      if (control(10).eq.2) ibasl=ids
      if (control(10).eq.3) ibasl=itim
      if (control(10).eq.4) ibasl=iag
      if (control(10).eq.5) ibasl=isx
           if (control(11).eq.0) iaddl=1
      if (control(11).eq.1) iaddl=ior
```

```
      if (control(11).eq.2) iaddl=ids
      if (control(11).eq.3) iaddl=itim
      if (control(11).eq.4) iaddl=iag
      if (control(11).eq.5) iaddl=isx
      iptr=((ibasr-1)*irm)+iaddr
      iptc=((ibasc-1)*icm)+iaddc
      iptl=((ibasl-1)*ilm)+iaddl
           if (control(6).gt.5) iptr=1
      if (control(8).gt.5) iptc=1
      if (control(10).gt.5) iptl=1

c     write (6,*) iptr,ibasr,irm,iaddr
c     write (6,*) iptc,ibasc,icm,iaddc
c     write (6,*) iptl,ibasl,ilm,iaddl
c     write (6,*) iag,icm,ids,ilm,ior,iptc,iptl,iptr,irm,isx,itim

      return
      end
```

Subroutine gcountv1.f

Tabulation routine for type all type 1 queries. [An alternative routine will be used for type 2 queries]. Uses the subroutine *pointers* to work out which cell in the out array to increment.

B5.5 Listing of goutput.f

```
C*****************************************************************
C** A generalised output subroutine                              *
C*****************************************************************

      subroutine goutput (clabs,control,headertxt,llabs,ncol,ncolab,
     * nlay,nrow,outarray,rlabs,vlabs)

      character*130 lines(5)
      character*80 headertxt(3)
      character*24 rlabs(nrow),llabs(nlay),vlabs(5)
      character*10 clabs(ncol,ncolab)
      character*6 burstxt
      integer outarray(nrow,ncol,nlay),control(30),lnblnk

c--- width (no. of cols per page) & output style etc...

      if (control(21).eq.2) iwidth=2
      if (control(21).eq.1) iwidth=1

      istyle=control(4)

      itab=1
      if (nrow.le.20) itab=2
      if (nrow.le.10) itab=3
      if (nrow.le.5) itab=4


      do 10 n=1,130
         lines(1)(n:n)='-'
         lines(2)(n:n)='='
         lines(3)(n:n)='_'
         lines(4)(n:n)=':'
         lines(5)(n:n)='*'
 10   continue

c--- burst page
c---- this will be by-passed if raw data is preferred

      if (istyle.eq.1) then
         write (12,'(a1)') '1'
         write (12,'(lx,a79)') lines(5)(1:79)
         write (12,*)
         write (12,*)' #####  ###  ##   ##  ##   ##  ###   ##'
         write (12,*)'   #     #   # # # #  # # #   #   #'
         write (12,*)'   #     #   #  #  #  #  #   #   #'
         write (12,*)'   #     #   #     #  #     #   # ##'
         write (12,*)'   #    ###  #     #  #     #  ###  ##'
         write (12,*)
         write (12,*)
         write (12,'(lx,a79)') lines(5)(1:79)
         write (12,*)
         write (12,*) ' Layout of pages in layer 1:'
         icolpage=((iwidth*6)*itab)
         rcol=ncol
         rcolpage=icolpage
         rpgw=rcol/rcolpage
         if (rpgw.ne.int(rpgw)) rpgw=int(rpgw+1)
         rrow=nrow
         rpgh=rrow/40
         if (rpgh.ne.int(rpgh)) rpgh=int(rpgh+1)
         itmp=1
         ipgw=int(rpgw)
         ipgh=int(rpgh)
         write (12,*)' [Output page no.]'
         write (12,*)
         if (ipgw.eq.1) ill=1
         if (ipgw.eq.2) ill=6
         if (ipgw.ge.3) ill=11
         write (12,*)'       Columns ',lines(1)(1:ill),'>'
         write (12,*)' Rows'

         if (ipgw.eq.1) then
            do 20 i=1,ipgh
               if (i.lt.ipgh) burstxt='   |  '
               if (i.eq.ipgh) burstxt='   V  '
               write (12,'(a6,lx,a1,i3,a2)') burstxt,'[',itmp,' ]'
 20            itmp=itmp+1
         endif

         if (ipgw.eq.2) then
            do 30 i=1,ipgh
               if (i.lt.ipgh) burstxt='   |  '
               if (i.eq.ipgh) burstxt='   V  '
               write (12,'(a6,lx,a1,i3,a7,i3,a2)') burstxt,
     *              '[',itmp,' ]    [',itmp+1,' ]'
 30            itmp=itmp+2
         endif

         if (ipgw.ge.3) then
            do 40 i=1,ipgh
               if (i.eq.1) burstxt=' Rows '
               if (i.gt.1.and.i.lt.ipgh) burstxt='   |  '
               if (i.gt.1.and.i.eq.ipgh) burstxt='   V  '
               write (12,'(a6,2x,a1,i3,a2,a7,a1,i3,a2)') burstxt,
     *              '[',itmp,' ]',' ... ','[',itmp+ipgw-1,' ]'
 40            itmp=itmp+ipgw
         endif

         write (12,*)
         write (12,'(lx,a79)') lines(1)(1:79)
         write (12,*)
         write (12,*) ' Table shows:'
         write (12,*) ' ',vlabs(1)(1:lnblnk(vlabs(1))),' by ',
     *       vlabs(2)(1:lnblnk(vlabs(2))),' by ',
     *       vlabs(3)(1:lnblnk(vlabs(3)))
         write (12,*) ' Statistic:'
         write (12,*) ' ',vlabs(4)
         write (12,*)
         write (12,*)
         write (12,'(lx,a79)') lines(1)(1:79)
      endif

c--- initial geometry

      ipage=0
      ic1=1
      ic2=iwidth*6
      ir1=1
      ir2=40
      ill=1

c--- come in here for new page

 50   continue
      if (istyle.ge.2) ic2=ncol
      if (istyle.ge.2) ir2=nrow
      ipage=ipage+1

c-- itab2 is set to permit n tables per page for low nrow values

      itabcount=0

      if (istyle.eq.1) then
         write (12,'(a1)') '1'
         call line (iwidth,2,lines)
         write (12,'(lx,a79)') (headertxt(n)(1:79),n=1,2)
         write (12,'(lx,a60,a6,i3)') headertxt(3)(1:60),' Page:',ipage
         call line (iwidth,2,lines)
      endif

c--- table header
c--- come in here for new table only

 60   continue

      if (ir2.gt.nrow) ir2=nrow
      if (ic2.gt.ncol) ic2=ncol

      itabcount=itabcount+1
      if (istyle.eq.1) then
       write (12,*)
       call line(iwidth,2,lines)
       write (12,'(lx,a24,a1,lx,a24)') vlabs(3),'|',vlabs(2)
       if (iwidth.eq.2) then
       write (12,'(lx,a24,a1,a103)') llabs(ill),'|',lines(1)(1:103)
       else
       write (12,'(lx,a24,a1,a53)') llabs(ill),'|',lines(1)(1:54)
       endif
       write (12,'(lx,a24,a1,12a9)') lines(3)(1:24),'|',
     *  (clabs(n,1),n=ic1,ic2)
       if (ncolab.eq.2) then
        write (12,'(lx,a24,a1,12a9)') vlabs(1),'|',
     *   (clabs(n,2),n=ic1,ic2)
       else
        write (12,'(lx,a24,a1,12(8x,a1))') vlabs(1),'|',
     * (lines(4)(icn:icn),icn=ic1,ic2)
       endif
       call line (iwidth,1,lines)
      endif

      if (istyle.eq.3) then
         write (12,*)
         write (12,*) vlabs(3)(1:lnblnk(vlabs(3))),':',
     *        llabs(ill)(1:lnblnk(llabs(ill))),':'
         write (12,*)
         if (ncolab.eq.1) then
            write (12,*) vlabs(1)(1:lnblnk(vlabs(1))),':',
     *           (clabs(n,1)(1:lnblnk(clabs(n,1))),n=ic1,ic2)
         else
            write (12,*) vlabs(1)(1:lnblnk(vlabs(1))),':',
     *     (clabs(n1,1)(1:8),':',clabs(n1,2),n=ic1,ic2)
         endif
         write (12,*)
      endif

c--- output section

      ilinecount=0

      if (istyle.eq.1) then
        do 70 n=ir1,ir2
         ilinecount=ilinecount+1
         if (n.eq.nrow) call line (iwidth,1,lines)
         write (12,'(lx,a23,a1,1219)') rlabs(n)(1:23),':',
     *    (outarray(n,icn,ill),icn=ic1,ic2)
         if (n.eq.ir2) call line (iwidth,2,lines)
 70     continue
      endif

      if (istyle.eq.2) then
        do 80 n=ir1,ir2
         ilinecount=ilinecount+1
         write (12,*) (outarray(n,icn,ill),icn=ic1,ic2)
 80     continue
        write (12,*)
      endif

      if (istyle.eq.3) then
        do 90 n=ir1,ir2
         ilinecount=ilinecount+1
         write (12,*) rlabs(n)(1:lnblnk(rlabs(n))),':',
     *    (outarray(n,icn,ill),':',icn=ic1,ic2)
 90     continue
        write (12,*)
      endif


c-- more columns?

      if (ncol.gt.ic2) then
         ic1=ic2+1
         ic2=ic2+(iwidth*6)
         if (itabcount.lt.itab) then
            goto 60
         else
            goto 50
         endif
      endif
```

```
c-- more rows?

      if (nrow.gt.ir2) then
         ic1=1
         ic2=iwidth*6
         ir1=ir2+1
         ir2=ir1+40
         goto 50
      endif

c-- more layers?

      if (nlay.gt.ill) then
         ic1=1
         ic2=iwidth*6
         ir1=1
         ir2=40
         ill=ill+1
         goto 50
      endif

      close(12)

      return
      end

c-----------------------------------------------------------------------

      subroutine line (iwd,itype,lines)

      character*130 lines(4)

      if (iwd.eq.2) then
        write (12,'(1x,a130)') lines(itype)(1:130)
      else
        write (12,'(1x,a79)') lines(itype)(1:79)
      endif

      return
      end
```

Subroutine goutput.f

Writes out the table / excel matrix, plus the burst sheet if required.

Produced By
School of Geography
University of Leeds
Leeds LS2 9JT
From Whom Copies May Be Ordered