Data Management for Surveys and Trials

A Practical Primer using EpiInfo

Steve Bennett, Mark Myatt, Damien Jolley, and Andrzej Radalowicz

Brixton Books

First Published in 1996 by Brixton Books, Station Building, Llanidloes, Powys SY18 6EB

This edition published 1996

A CIP catalogue record for this book is available from the British Library

ISBN 1-873937-41-5

Coding and data entry are the Cinderellas of survey method, attracting little academic interest or concern compared with sampling, interviewing and tests of significance. Yet a survey, like the proverbial chain, is probably as good as its weakest link. And if enough care, thought and time are not devoted to these aspects of the study the validity and usefulness of the whole operation are jeopardised. We have no magical alternatives to the painstaking and methodical attention to detail which are needed for this part of the study. To do it well you need to be obsessional.

Anne Cartwright & Clive Seale, The Natural History of a Survey, 1990

The authors would like to thank Neal Alexander, Linda Williams, and the late Nicola Dollimore for their contribution to the courses on which parts of this book are based, Maria Quigley, Judith Glynn, and other teaching colleagues at the London School of Hygiene and Tropical Medicine for their comments on earlier versions, and Jimmy Whitworth for allowing us to use the onchocerciasis data.

Steve Bennet and Andrzej Radalowicz are supported by the Medical Research Council.

This book is dedicated to the memory of Nicola Dollimore.

Contents

Overview of the course

The objective of this book is that you should understand and know how to carry out the *data processing* aspects of a research study. This book uses the EpiInfo software package. EpiInfo was developed for epidemiological research (the study of illness in populations rather than in individuals), and the case study we use throughout this book is a medical one, but the principles of data management, and the details of data management activities are the same whatever the field of study.

We have chosen EpiInfo for this book for three reasons:

1. It has been specifically written for use in research studies with modules that are designed to assist with each stage the data management process.

2. It is easy to use. Although its features may be fewer than more sophisticated packages, the simplicity is a benefit in many situations, particularly for beginners.

3. It is distributed free of charge.

We emphasise that the objective is that you should learn the principles of data processing, so that even if you go on to work in another software package, the concepts and techniques that you have learned here using EpiInfo will remain valid.

Data processing takes place at **all** stages of a study, and should be planned at the very start. The objective is to produce data of the highest possible quality in a form suitable for statistical analysis. The stages of data processing that we shall consider are:

- ❑ Planning the data needs of a study
- ❑ Data collection
- ❑ Data entry
- ❑ Data validation and checking
- ❑ Data management and manipulation
- ❑ Data summary and presentation

The material is designed so that students work through the book at their own pace.

All of the exercises in this book require you to have access to EpiInfo version 6. This book makes extensive use of sample datasets which are supplied on the accompanying disk. You should copy **all** the files on this disk to a working directory on your hard disk before continuing.

We do not consider the details of statistical analysis in this book.

Databases

Objectives of this section

By the end of this section you should be able to:

- ❑ Appreciate the steps involved in data processing for a research study.
- ❑ Understand the idea of a database, and the concepts of file, case and variable.
- ❑ Understand the types of variables used in EpiInfo and their attributes.
- ❑ Create a questionnaire and data entry screen using the EPED module of EpiInfo.
- ❑ Create a database file using the ENTER module of EpiInfo.
- ❑ Enter data into a file using the ENTER module of EpiInfo.

The advantages of using a computer

Most epidemiological studies involve the collection of information (*data*), either by asking questions, or from hospital records, or from laboratory results. The use of a computer allows:

- ❑ Storage of enormous quantities of data.
- ❑ Ease of checking and correcting (editing) data.
- ❑ Ease of tabulation and presentation of results.
- ❑ Powerful and quick statistical analyses.

The computer can also be used to:

- ❑ Generate lists of study subjects who are to be seen again.
- ❑ Produce updated reports on the progress of the survey.

Case study: The River Blindness (Onchocerciasis) Study

Any investigation is likely to involve collecting data from several different sources using more than one type of questionnaire. At this stage, however, we shall imagine that our study uses only one type of questionnaire and see what happens to the data collected with it.

The study that we shall work with throughout this book is a trial of an intervention against river blindness (*onchocerciasis*) in Sierra Leone in West Africa. River blindness is a disease spread by the blackfly (*simulium*) that breed in fast-flowing rivers. The fly bites and injects the parasite larvae (*microfilariae*) under the skin. These mature and produce further larvae that may migrate to the eye where they may cause lesions leading to blindness.

The study is a randomised trial of the drug *Ivermectin*(Merck) against a placebo. An initial census was followed by five survey and treatment rounds at six monthly intervals.

The questionnaire on page 35 is similar to that used to collect baseline data for the study. It contains questions on background demographic and socio-economic factors, and on subjects' previous experience of onchocerciasis.

Reference

Whitworth JAG, Morgan D, Maude GH, Downhan MD, and Taylor DW (1991), *A community trial of ivermectin for onchocerciasis in Sierra Leone: clinical and parasitological responses to the initial dose,* Transactions of the Royal Society of Tropical Medicine and Hygiene, **85**, 92-6.

The progress of a questionnaire

The usual progress of the questionnaire and its data would be:

1. Interviewer collects data and completes questionnaire.

2. Interviewer checks questionnaire, and corrects any errors, returning to respondent if necessary.

3. Supervisor checks questionnaires, re-interviewing a sample of respondents.

4. A data entry clerk enters the data into the computer.

5. A different data entry clerk enters the data into the computer a second time.

6. The two data *files* are compared to find any typing errors, and errors corrected.

7. Either at the time of data entry, or afterwards, data are checked. The checks ensure that data are within allowable *ranges* (e.g. sex must be either male or female). Checks also ensure that data are *consistent* from one question to another (e.g. if respondent is pregnant then sex must be female!). Any errors found are corrected.

8. When the data are *clean*, there will usually be a need to create new variables or manipulate existing ones (e.g. calculation of latency periods, grouping age in five year bands etc.).

9. Data will need to be *linked* (or *related*) to data from other forms and questionnaires (e.g. linking interview data with laboratory data).

10. Tables and graphs will be produced.

11. Statistical analyses will be performed.

12. Data may need to be transferred to a different software package for advanced statistical analysis or graphics.

This is the sequence of events that we shall be considering in this book.

Overview of EpiInfo

EpiInfo version 6 is a computer package specially created to enable researchers to carry out epidemiological investigations. It consists of several modules, which carry out all the tasks described above, and more:

Module	Functions
EPED	Create questionnaires or write and edit text (such as output from the ANALYSIS module).
ENTER	Make data files from questionnaire files, enter, edit, and query data. EpiInfo v6.03 has an extended version of ENTER called ENTERx. This program works in exactly the same way as ENTER but uses 'high memory' allowing it to cope with larger and more complex files.
ANALYSIS	Batch checking of data, creation and manipulation of variables, production of tables and graphs, and statistical analysis.
CHECK	Add range checking, skip patterns, legal values, and complex coding rules to the data entry process.
IMPORT	Import data files from other programs into EpiInfo.
EXPORT	Export EpiInfo files to other file formats.
MERGE	Append, combine, update, or restructure data files.
STATCALC	Calculator for 2 x 2 tables, sample size calculations, and the chi-square test for trend.
CSAMPLE	Analyse data from complex sample (e.g. cluster sampled) surveys.
EPITABLE	Statistical and epidemiological calculator
EPINUT	Nutritional anthropometry
VALIDATE	Comparison of two copies of a data file
EPIGLUE	Create customised menu and help systems

The shaded modules are **not** covered in this book. Besides these modules there are two optional modules:

Module	Functions
EPIMAP	Production of maps from EpiInfo data files
LOGISTIC	Logistic and conditional logistic regression

EpiInfo and EpiMap were developed at CDC/EPO (Centres for Disease Control Epidemiological Program Office) in Atlanta and WHO/GPA (World Health Organisation Global Program on AIDS) in Geneva. Logistic was developed by Gerard Dallal and is part of his excellent StatTools™ suite of low-cost statistical and epidemiological software. The programs, manuals, and related products are distributed by Brixton Books.

Database definition

The core feature of a *database file* is that individual *records* are stored in a *file* that contains *data* (numbers and characters) and a *structure* that defines what the numbers and characters refer to.

A database file will be part of a database management system (DBMS). Such systems allow the user to perform a wide range of operations using a set of simple instructions. These include creating new files, opening existing files, entering new data, and sorting, searching and editing records. To use a DBMS for a specific project it is necessary to adapt the general system for the specific set of data in hand.

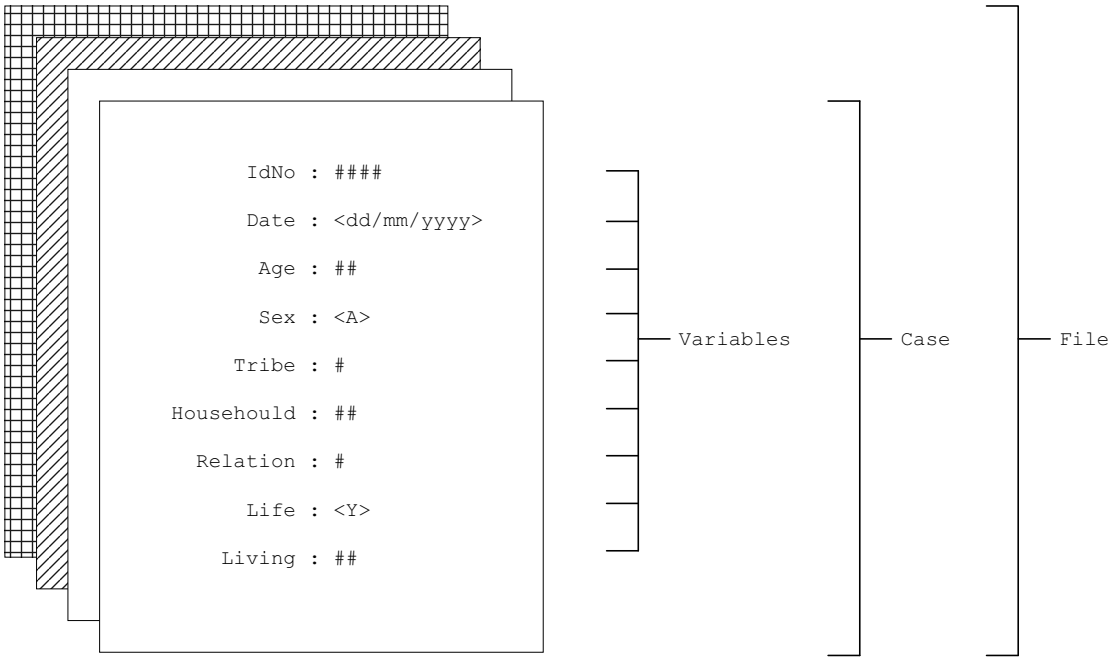
There are many different DBMS software packages available for use on personal computers, of which dBase (and derivatives such as FoxPro and Visual dBase), Paradox, Access, and Approach are probably the most widely used. We show here how to set up a database using the EpiInfo package.

Once the layout of a questionnaire or data collection form has been decided, the database structure can be defined. This will correspond directly to the data that is to be collected (although there may be questions on the form that lead to data that is not necessary or suitable for entering or further analysis).

Simple studies may have just a single data collection form. Complex studies may generate several forms. Separate data files will be created to store data from the separate forms. By ensuring that there is an *identification variable* common to each data file (e.g. individual IDNO) it is a simple job to link data from the different data files when required.

Datasets, cases, and files

A set of data are *stored* in the computer as a *file*. A file consists of a collection of *cases* (or *records*). Each case contains data in a series of *variables* (or *fields*). In our example the cases are individuals interviewed for the onchocerciasis baseline survey and the variables are the answers to the questions asked:



Once the layout of the questionnaire or data collection form has been decided the *structure* of the data file can be defined. This is a straightforward process since the structure will correspond to the data that is to be collected.

Data are usually represented in a table in which each *row* represents an individual case (record), and each *column* represents a variable (field). Only the first few questions are shown here for the first four respondents:

Survey Number	Date	Sex	Age	Tribe	Household Number	Relation To Head Of Household
1	10/11/1988	M	56	1	1	1
2	10/11/1988	F	49	1	1	2
3	10/11/1988	M	15	1	1	3
4	10/11/1988	M	28	2	1	6

A file with a structure such as this, with information about the variables such as their names and allowable ranges, is known as a *database file*.

Defining database file structure

To define the structure of a database file we need to specify for each variable a *name*, a *type*, and a *length*. The variable type chosen will depend on the type of data that the variable is to contain.

An example of the structure corresponding to eight variables in the onchocerciasis baseline questionnaire with the EpiInfo variable definitions is:

Name	Type	Length	EpiInfo definition
IDNO	numeric	4	####
DATE	date	8	<mm/dd/yyyy>
AGE	numeric	2	##
SEX	text	1	<A>
TRIBE	numeric	1	#
HHNO	numeric	2	##
REL	numeric	1	#
LVILL	logical (yes/no)	1	<Y>

Each variable has a *name*. Names allows us to refer to variables for data checking and analysis.

Each variable is of a certain *type*. The type you choose to assign to a variable depends upon the type of data it will contain. The most commonly used data types are *text*, *numeric*, *logical*, and *date*.

The *length* of a variable defines how much data it can hold. A text variable with length ten will be able to hold up to ten letters or numbers. A numeric variable with length three will be able to hold numbers between -99 and 999. The length of a variable must correspond to the maximum anticipated number of letters and / or numbers.

Variable names

In EpiInfo variable names:

- Must not exceed 10 characters
- Must begin with a letter not a number
- Can otherwise contain any sequence of letters and digits
- Should not contain any spaces or punctuation marks

Names can describe the variable they refer to (e.g. OCCUP is probably more informative than VAR17) but with large questionnaires it may be easier to use question numbers (e.g. Q017) as variable names.

Examples of **illegal** variable names are:

1DATE	(begins with a number)
LAST NAME	(contains a space)
COUNTRYOFORIGIN	(longer then ten characters)

Write down three legal and three illegal variable names:

Legal Variable Names	Illegal Variable Names

EpiInfo allows you to create variable names that are ten characters long. Other packages (such as SPSS/PC+) limit the length of variable names to eight characters. If you think that you might need to use another package to analyse your data then you should make sure that you limit the length of variable names to eight characters.

Variable types

Each variable must be of a certain type. The type you choose to assign a variable will depend on the type of data you wish it to contain. The EpiInfo package provides many different variable types:

> **TEXT variables** are used for holding information consisting of text and / or numbers. Text variables are useful for holding information such as names and addresses.
>
> EpiInfo has a special type of text variable called **UPPER CASE TEXT**. This is similar to the text type but can only hold upper case (i.e. capital) letters as well as numbers. If you enter lower case text into an upper case text variable it will automatically be converted into upper case text. This is useful because it avoids potential problems caused by inconsistent use of capital letters for data items such as postcodes. You can enter numbers into text and upper case text variables but you cannot perform mathematical operations with them.
>
> **NUMERIC variables** are used for holding numerical information. They can be used for holding *categorical* or *continuous* data. Numeric variables can be defined to hold either *integers* (whole numbers) or *real numbers* (numbers with a fractional part).
>
> **LOGICAL or YES/NO variables** are used for holding data that can have two possible states such as whether a respondent has been ill or not. Logical variables can hold either the character 'Y' or the character 'N'. Logical variables are sometimes called *binary categorical variables*.
>
> **DATE variables** are used to hold dates. Date variables can be used to hold data in the American (**`mm/dd/yyyy`** or **`mm/dd`**) and European (**`dd/mm/yyyy`** or **`dd/mm`**) formats. You can perform simple arithmetic such as addition and subtraction with date type variables. The advantage of using date type variables is that the ENTER module of EpiInfo will only allow you to enter valid dates. Date type variables also simplify any calculations as factors such as variable month length and leap years are accounted for.

EpiInfo also provides PHONENUM, LAST-UPDATE , and AUTO-INCREMENT variable types. These are of limited use in surveys and trials but are of more use in routine data collection activities such as communicable disease surveillance.

Data and variable types

When designing your own questionnaires and data files think carefully about the sort of data you want each variable to hold.

If you wish to perform mathematical operations with variables they **must** be of the numeric type. You should always use numeric variables for continuous data such as *unmodified measures* of height and weight. Statistical procedures such as *ANOVA*, *correlation*, and *regression* will only work with data stored in numeric variables.

You can use text or numeric variables to hold *categorical* data. Some investigators prefer to use numeric variables to hold categorical data. Categories are given numeric *codes*. Data coded in this way may be easier to use with other statistical packages and with some statistical procedures (e.g. *marker variables* in regression analysis). With categorical data it is a good idea to keep codes consistent across variables. This will help reduce errors at all stages of a survey.

If you use text variables to hold categorical data make sure that you use the upper case text type. This will avoid potential problems caused by inconsistent capitalisation of data items.

Logical or Yes/No variables hold a special type of categorical data. Some investigators prefer to use numeric variables to hold categorical data. Categories are given numeric codes. Data coded in this way may be easier to use with other statistical packages and with some statistical procedures (e.g. marker variables in regression analysis).

Many statistical packages do **not** support date type variables. If you use date type variables then you may need to convert data to a different format (e.g. calendar-month-in-century or date-time index number) before it can be used with other packages. Variables can be changed or *recoded* with the ANALYSIS module of EpiInfo. Sometimes you will be interested in a single data item (e.g. month or year) or you may need to use dates that do not conform to a Western calendar. In this case you may need to use a combination of numeric and text type variables to hold your data. The advantage of using date type variables is that the ENTER module of EpiInfo will only allow you to enter valid dates. Date type variables also simplify any calculations as factors such as variable month length and leap years are accounted for.

Defining variable type and length

EpiInfo variables are defined using special characters in questionnaire (.QES) files:

> **NUMERIC** variables are defined using the **'#'** character. A variable defined as **###** can hold three digits. If a decimal point is given then the variable will be in *fixed decimal* format. A variable defined as **#####.##** can hold numbers between -9999.99 and 99999.99.
>
> **TEXT** variables are defined using the underline **'_'** character. The length of the variable will be the number of underline characters used.
>
> **UPPER CASE TEXT** variables are defined by enclosing an upper case **'A'** within angle brackets. The number of characters between the **'<'** and **'>'** characters defines the length of the variable. A variable defined as **<AAA>** will be able to hold up to three upper case letters and numbers.
>
> **DATE** variables are defined by enclosing the required date format between angle brackets. A variable defined as **<dd/mm/yyyy>** will be able to hold a Long European date. The other date formats provided by EpiInfo are **<dd/mm>** (Short European), **<mm/dd/yyyy>** (Long American), and **<mm/dd>** (Short American).
>
> **LOGICAL** or **YES/NO** are defined by enclosing an upper case **'Y'** between angle brackets (i.e. **<Y>**). They are used for holding information that can have two possible states such as whether a respondent has been ill or not. Logical variables can hold either the character **'Y'** or the character **'N'**.

You should **not** use any of these special characters (i.e. **'_'**, **'<'**, **'>'**, and **'#'**) in EpiInfo questionnaire (.QES) files for any purpose other than defining variable type and length.

When designing your own data files think carefully about the sort of data you want each variable to hold. If you want to perform mathematical operations with variables then they should be of the numeric type. Some statistical procedures will only work if data are stored in numeric variables. It may also be useful to use numeric variables to hold categorical data for use with some statistical procedures.

You can enter numbers into text and upper case text variables but you cannot perform mathematical operations. If you wish to perform mathematical operations with variables (e.g. calculate means) they should be of the numeric type.

An advantage of using date variables is that EpiInfo will only allow you to enter valid dates. You can perform addition and subtraction with date variables. These calculations account for variable month length and leap years, and give an answer in days.

Database structure

Give a suitable structure (i.e. variable name, type, length) including the EpiInfo variable definition for the following variables:

Variable	Name	Type	Length	EpiInfo definition
Family Name	FAMNAME	Text	15	<AAAAAAAAAAAAAAA>
Date of Birth				
Province				
Temperature				
Hospital Number				
Age in years				
Ill today?				
Diagnosis				

When designing your own questionnaires and database files think carefully about the sort of data you want each variable to hold. The same data may be coded and stored in different ways. For example, a diagnosis of *primary syphilis of the tonsils* could be coded and stored in any of the following ways:

Coding	Coded	Type	Length	EpiInfo definition
None	SYPHILIS (TONSILS)	Text	18	<AAAAAAAAAAAAAAAAAA>
KC60	A1	Text	2	<AA>
ICD-9	91.2	Numeric	4	##.#

KC60 is a coding scheme used by the UK Department of Health for the surveillance of sexually transmitted infections and to measure GUM/STD clinic workload. ICD-9 is an international code used to record morbidity and mortality in a standardised form.

Null values

Sometimes data items will not be available or are not appropriate to collect for some respondents (e.g. age at menarche for male respondents). It is important that you take this into account when designing questionnaires, coding schemes, and data files. Data that is missing or not appropriate is called *null* data. There are two types of null data:

> When data are not available it is defined as *missing data*. It is bad practice to leave data entry spaces on the questionnaire or data entry screen empty because it can lead to confusion at data entry time. Always consider the codes to use when a value is missing. It is common practice to use 9, 99, 999 etc. to denote missing data. Coding missing data in this way will require data to be coded back to missing or null prior to analysis.
>
> When data are not available because it is not appropriate to collect it is defined as *not-appropriate*. Always consider the codes to use when data are not-appropriate. It is common practice to use 8, 88, 888 etc. to denote not-appropriate data. Coding not-appropriate data in this way may require data to be coded back to missing or null prior to analysis.

The coding scheme you decide to use for missing and non-appropriate data should be defined in advance and be consistent across variables.

EpiInfo handles missing data automatically. Any field that is left empty at data-entry or uses missing data in calculations receives a special *missing data* code (. = missing).

Identifying (ID) numbers

When designing a questionnaire or a database file it is important to include a variable that holds a unique value for each case. This makes finding both paper forms and individual cases in a database file easier should you need to query or edit a data item. This variable is called the *key* or *identifier* variable.

A survey will usually include several different forms related to each other. The key variable allows cases in one file to be linked to cases in another file.

In this example it is possible, using a key variable (MUMID), to link data from mothers and children:

Mother's File			
MumID	---	---	---
297			
298			

Children's File		
MumID	KidID	---
297	001	
297	002	
297	003	
298	001	

The use of a key variable ensures that data for each mother can be linked with data for that mother's children. Note that in this example the key variable is unique in the mother's file but not unique in the children's file. The combination of MUMID and KIDID uniquely identifies each child.

Another example of linking data between files is when a field questionnaire is linked to a laboratory report form for the same person.

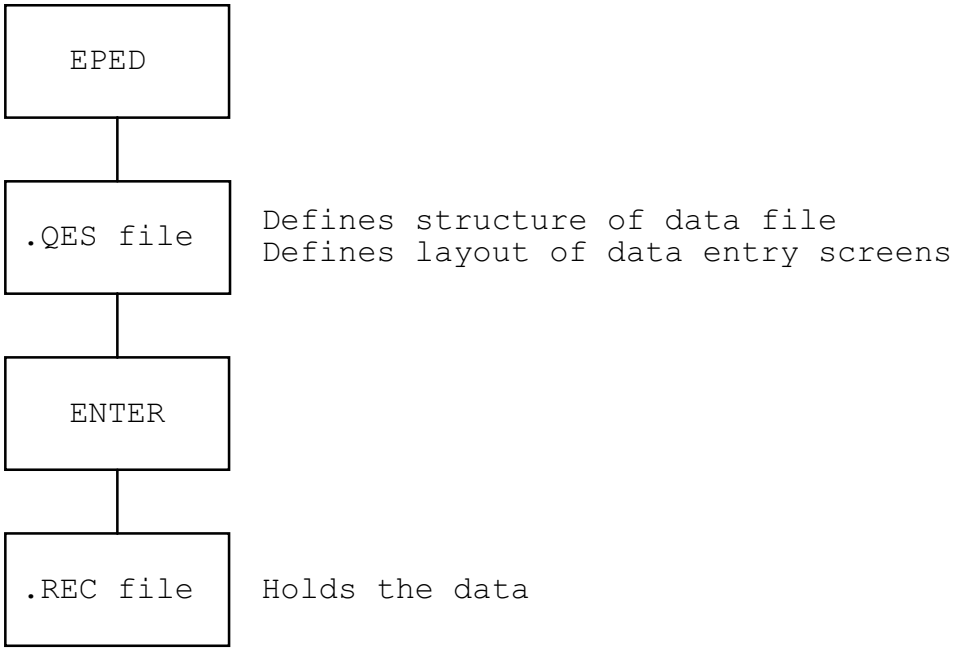
A database that consists of more than one linked data file is called a *relational* database. A database that consists of one data file (or many files of identical structure) is called a *flat-file* database.

Creating a data file

Creating a database file in EpiInfo is a two stage process:

1. First you use EPED to create a screen questionnaire, or data entry screen. This must include the variable names, types, and lengths. A questionnaire file **must** have the extension .QES.

2. The structure of this .QES file in turn defines the structure of the data file. You create this using the ENTER module. In EpiInfo a data file is called a *record file* and has the extension .REC.



The questionnaire (.QES) file defines the structure of the record (.REC) file and the layout of the data-entry screens. Data are entered and stored in the record (.REC) file.

The ENTER module is also used to enter data into an existing record (.REC) file.

Starting EpiInfo

On different computer systems there will be different ways of starting the EpiInfo package. On most, however, it will be sufficient to type:

```
epi6
```

at the DOS prompt and press the ENTER key.

Once EpiInfo has started you will see the *top-level menu*. Select **Programs** from the top-level menu. To select an item from the top-level menu you must first move the highlighted bar using the ← and → keys. When the highlighted bar is over the menu item you want to select (in this case **Programs**) press the ENTER key to select it. A *drop-down menu* will then appear.

Use the ↑ and ↓ keys to move the cursor bar. As you move the cursor bar up and down the **Programs** drop-down menu you will see the descriptions of the different modules on the bottom line of the screen. Use the ↑ and ↓ keys to move the cursor bar up and down and read the descriptions of the different modules.

You may get a more complete description of each module if you press the F1 key while the cursor bar is over the name of the module that you are interested in. Place the cursor bar over the **STATCALC calculator** menu item and press the F1 key. EpiInfo will display a window containing a description of the STATCALC module. Once you have read the description press ESC to return to the menu system.

Using ANALYSIS to browse through a dataset

ANALYSIS is the data analysis module of EpiInfo. Using ANALYSIS you can use simple commands to produce lists, frequencies, statistics, and graphs. In this exercise you will use ANALYSIS to examine a *dataset*.

Position the cursor bar over the menu option **`ANALYSIS of data`** on the **`Programs`** menu. Press [ENTER] to select ANALYSIS.

Once ANALYSIS starts the screen is divided into several distinct sections. The upper section is headed **`Output`** and the lower section is headed **`Commands`**. At the top of the screen are two lines giving the status information. For example:

```
Dataset: <None>                                Free memory 353K
Use READ to choose a dataset
```

The status information shows the name of the current data file and the amount of free memory in the system (the exact figure will depend on how your system has been configured).

The bottom line of the screen shows the *function keys* available from within ANALYSIS:

Key	Function
[F1]	Help
[F2]	Display menu of commands
[F3]	Display menu of variables in the current dataset
[F4]	Browse
[F5]	Toggle printer on and off
[F9]	DOS shell
[F10]	Quit ANALYSIS

We need to tell ANALYSIS the name of the dataset we want to look at. Type:

```
read demog_1.rec
```

And press [ENTER] . The name of the file (DEMOG_1.REC) and the number of records (466) will appear at the top of the screen showing that the file has been read.

Press the [F4] key to browse through the data. The first twenty-one records of DEMOG_1.REC appear on the screen, one record per line, with variables running across the line. Move around the file using the [↑] , [→] , [↓] , [←] , [PG UP] , [PG DN] , [END] , and [HOME] keys.

Press [F4] again to select full screen mode. You now have a single record displayed on the screen. The record number is displayed to the right of the menu bar at the bottom of the screen. Look at the menu bar and work out how to move to the next record and back to the previous one. Try this a few times. Press [F4] again to return to browse mode. When you have finished examining the data press [F10] to return to the main screen.

Press [F10] to return to the top-level menu

Making a screen questionnaire

Since our data comes from an onchocerciasis study we might call the questionnaire (.QES) file ONCHO.QES. We can use EPED module to create questionnaire (.QES) files.

The format of the questionnaire (.QES) file can be very simple. For example:

```
 IDNO:  ####
 Date:  <dd/mm/yyyy>
  Age:  ##
  Sex:  <A>
Tribe:  #
 HHNO:  ##
  Rel:  #
LVill:  <Y>
```

The questionnaire (.QES) file will usually be more elaborate and look more like the field questionnaire than this.

Whatever appears in the questionnaire (.QES) file will appear on the screen during data entry. It is worth spending a little time getting it to look good by lining up columns, giving adequate spaces to make it clearer to read, and putting groups of similar variables together. It is a matter of choice whether there is more than one variable on a line - the computer will read across the lines. For large data entry projects it is best to lay out the data entry screens with one variable per line with the variables lined-up one below the other. This can make data entry less error prone.

EPED

Select **EPED word processor** from the **Programs** menu. Once EPED starts the screen will be blank except the top and bottom lines of the screen.

The top line is called the *menu bar*. All the commands available within EPED are accessed from the menu bar. To select an item from the menu bar press the associated *function key*. For example to access the commands that deal with files (e.g. loading and saving files) you would need to press the F2 key. All commands are selected from menus of the *highlighted bar* type.

On the screen you will see a small flashing line or block. This is called the cursor. Whatever you type at the keyboard will be placed on the screen at the cursor position.

The bottom line of the screen is called the *status line* and conveys information about the document being edited. A typical status line might look like this:

```
1  UNTITLED       EPED      405944 P1   L1   C1  WW Ins
```

WordWrap (WW) means that text will automatically be *wrapped* onto the next line when it reaches the edge of the screen. This means that you do not have to press ENTER at the end of each line as you would with a typewriter. This is useful for writing documents and letters, but not for questionnaire files as it prevents you from using the full width of the screen.

In *insert* (Ins) mode, text is inserted before the character at the cursor position. What you type will not type over what is already on the screen but will be inserted into the text. The alternative is *overtype*, in which new text overwrites the previous text. It is best to keep to insert mode as this makes it less likely that you will lose text by typing over it. You can move between insert and overtype modes by using the **Set** menu or by pressing the INS key.

EPED has all the usual facilities of a simple word-processor (blocks, find and replace, etc.). These facilities are not covered in this book. If you wish to use these facilities you can explore the function keys and help system. Pressing F3 EPIAID and selecting **Word processing tutorial** will give you a tutorial on using EPED.

Setting up EPED

The EPED module performs three separate functions. These are:

- Word-processing and report writing
- Creating questionnaire (.QES) files for use with ENTER
- Editing command files (programs) for use with ANALYSIS and CHECK

Each function requires different editing features. This means that before you start entering text you have to tell EPED how to behave:

Function	WW/TXT/QES mode
Wordprocessing and report writing	**WW**
Creating .QES files for use with ENTER	**QES**
Editing command files	**TXT**

Select the **Set** menu by pressing F6 .

Select the option **WW/TXT/QES mode** from the **Set** menu by pointing to it with ↑ and ↓ keys. Press the SPACE bar until the mode reads **QES**. Press the ESC key to close the menu and return to the editing screen.

When EPED is in **QES** mode it places black blocks at the left and right sides of the screen at the top of each data entry screen.

Get help on **WW/TXT/QES mode** by pointing to it on the **Set** F6 menu and pressing F1 . EPED will display a window describing the command. When you have read the help text, press ESC . Press ESC again to close the menu. Get help for a few more commands. If there is more than one screen of help available **PgDn** will be displayed in the bottom right corner of the help window. Pressing PG DN will display the next screen of help information.

Creating a .QES file using EPED

We shall create a questionnaire (.QES) file using EPED. Variable names are typed onto the screen in the position and order you would like to use when entering data. Next to the variable names are typed the special characters that define the length and type of each variable. On the blank screen copy the questionnaire:

```
Onchocerciasis Baseline Survey - Personal Data

ID Number                              {IDNO}   ####
Date of Interview                      {DATE}   <dd/mm/yyyy>

Age in years                            {AGE}   ##
Sex (M or F)                            {SEX}   <A>
Tribe  (Mende 1, Other 2)              {TRIBE}  #
Household number                        {HHNO}  ##
Relation to head of household?          {REL}   #
Lived in village all your life?        {LVILL}  <Y>
Years living in this village?           {STAY}  ##
What is your main occupation?           {OCC}   #
```

EPED has a feature that makes it easy to define variable types and lengths. Hold down the CTRL key and press the Q key twice. You will be presented with a menu of variable types. Select the type you require. If you choose a type that allows for variables of different lengths (e.g. text, upper case text, or numeric) you will be prompted for the appropriate information. The variable definition will be inserted at the current cursor position. You can obtain the same result by selecting QUESTIONS from the **Txt** F4 menu.

EpiInfo will automatically take the first ten non-blank characters of text prior to a variable definition and make this the variable name. Some words such as ‘what’, ‘are’, and ‘of’ are discarded automatically (e.g. ‘Date of onset’ becomes DATEONSET). If you precede the text with a number, EpiInfo will add the prefix ‘N’ (e.g. the text ‘2. Age’ becomes N2AGE). This can lead to unsuitable variable names (e.g. YEARSLIVIN rather than STAY). The best way of overcoming this is to put { } around the characters that you want to be the variable name. Always create variables with unique, short, easily remembered, but meaningful names. For long questionnaires it may be easier to use question numbers (e.g. Q01) as variable names.

Complex coding information, while needed on paper collection forms would tend to clutter up the data entry screen and should not be included.

Creating a .QES file

Check your work carefully against the example questionnaire (.QES) file. When you are satisfied select **Save** by pressing F9 . When prompted give the filename ONCHO.QES and press the ENTER key. The questionnaire (.QES) file is saved to disk as ONCHO.QES.

Press F10 to return to the top-level menu.

The questionnaire (.QES) file has been saved to disk. When naming files always choose a sensible and easily recognisable name. The conventions for naming files are the same as for DOS (eight characters for the name and three characters for the extension separated by a full stop). You must use the extension .QES for questionnaire files.

Filenames **must** be unique. If you specify the name of a file that already exists than EPED will ask if you want to overwrite it. If you are not sure that you want to overwrite the file then specify a different filename.

When working on a long file it is best to save your work frequently - especially if where you work is prone to fluctuations or cuts in the electricity supply. You can save a document at any time by pressing F9 .

Make sure that the questionnaire (.QES) file you use to create the record file is the final version. If you need to make some changes to a record file (e.g. add a variable) after it has been created then you will need to use ENTER to restructure the data file from the revised .QES file.

The symbols `'<'`, `'>'`, `'_'`, and `'#'` are used by EpiInfo to define variables. Avoid using them anywhere else in the questionnaire (.QES) file.

Editing the questionnaire

Start EPED and press F2 **File** and select **Open file this window**. When prompted for a filename, type ONCHO.QES and press ENTER to recall the file that you just created.

Which variables are defined as numeric?

Name	EpiInfo definition

Which variable is defined as upper case text?

Name	EpiInfo definition

Which variable is defined as DATE?

Name	EpiInfo definition

Which variable is defined as YES/NO?

Name	EpiInfo definition

Editing the questionnaire

What are the names given to the first four variables in the database?

Name	EpiInfo definition

By default, EpiInfo will automatically pick up the first ten characters of text (excluding blanks) prior to a variable specification and make this the variable name. The best way of overcoming this is to put { } round the characters that you want to be the variable name. If the { } were removed from {LVILL} what name would EpiInfo give to that variable?

After the last variable (OCC) add the variables for treatment, height and weight from the field version of the questionnaire shown on the next page. Make sure the variable names and types are appropriate and that the screen is easy to read. You will need to replace the boxes on the field questionnaire with appropriate EpiInfo variable definitions. Experiment with the CTRL Q Q method of inserting the variable definitions. Coding information needed on the field questionnaires will tend to clutter up the data entry screens.

Type F2 and select **`Save file to ..`** and type the new file name and press ENTER . **Do not** use the F9 key as this will overwrite the original questionnaire (ONCHO.QES) which we will use later.

Leave EPED by pressing F10 .

Press F10 again to leave EpiInfo.

Field questionnaire

Onchocerciasis Baseline Survey - Personal Data

Name ..... IDNO |__|__|__|__|

Village .....

Date of interview DATE |__|__|__|__|__|__|__|__|

Age in years AGE |__|__|

Sex (M or F) SEX |__|

Tribe (Mende 1 Other 2) TRIBE |__|

Household number HHNO |__|__|

Relationship to head of household? REL |__|

1. Self	*5. Other Blood Relative*
2. Parent	*6. Spouse*
3. child	*7. Other Non-Blood Relative*
4. sibling	*8. Friend*

How long have you lived in this village? STAY |__|__| (years)

What is your main occupation? OCC |__|

0. None / Missing	*5. Office Work*
1. At Home	*6. Trading*
2. At School	*7. Housework*
3. Farming	*8. Mining*
4. Fishing	

What treatment have you had for onchocerciasis?

ONCHTRT |__|

0. None	*4. 1 & 2*
1. Nodulectomy	*5. 1 & 3*
2. Banocide	*6. 2 & 3*
3. Native Medicine	*7. 1 & 2 & 3*

Weight (kg) WEIGHT |__|__|

Height (cm) HEIGHT |__|__|__|

Creating a database file

Start EpiInfo and select **ENTER data** from the **Programs** menu.

In response to the request for a data file type ONCHO (the .REC extension is appended automatically) and press ENTER . As the file ONCHO.REC does not exist you should specify the option:

2. Create new data file from .QES file

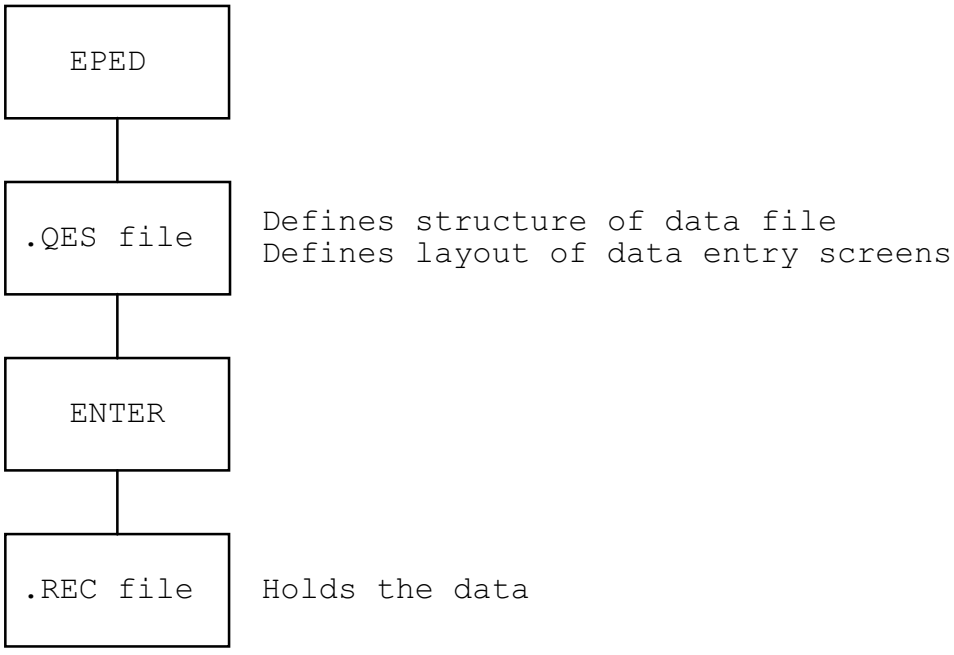
and press ENTER .

You will then be prompted for the name of the questionnaire (.QES) file. In response to the request for a questionnaire file type ONCHO (the .QES extension is appended automatically) and press ENTER .

Press ENTER again to confirm that you wish to proceed.

The ENTER module will use the questionnaire file (ONCHO.QES) to create a data file (ONCHO.REC) and then display the questionnaire on the screen, ready to receive data.

You have completed the process:



and may now move on to entering data that will be stored in the data (.REC) file.

Entering data

Fill in the blanks on the data-entry screen using the data for the following sample cases:

IDNO	DATE	AGE	SEX	TRIBE	HHNO	REL	LVILL	STAY	OCC
1001	01/05/1991	30	M	2	29	1	N	5	1
1002	02/05/1991	26	F	2	29	2	N	3	1
1003	02/05/1991	34	F	2	23	1	Y		7
1004	02/05/1991	23	M	2	12	1	N	2	4
1005	03/05/1991	19	F	1	47	3	Y		5
1006	03/05/1991	22	F	1	47	3	Y		6
1007	04/05/1991	39	M	2	13	3	Y		2
1008	04/05/1991	27	M	1	13	1	N	3	3
1009	04/05/1991	36	F	2	34	2	Y		2
1010	05/05/1991	32	F	1	37	1	Y		2

You can only enter the type of data shown on the prompt line at the bottom left of the screen. This may be different for each variable and depends on the variable types you specified in the questionnaire (.QES) file. If you enter data of the wrong type into a variable (or an invalid date into a date type variable) then ENTER will beep. Re-enter the correct data.

Pressing ENTER on its own (i.e. entering no data) will set a variable to 'missing'. This is a special value that is recognised as *missing* by the ANALYSIS module and excluded from any analysis. Statistical and database packages vary in the way they treat missing data so it may be best to use explicit codes for missing and not-available data which will have to be *recoded* before they can be used in ANALYSIS.

You may need to press ENTER to move onto the next variable.

The ↑ and ↓ keys allow you to move between variables. The HOME key will move the cursor to the first variable. The END key will move the cursor to the last variable.

After you have entered all the data for one case the message:

`Write data to disk (Y/N/<Esc>)?`

will be displayed at the bottom of the screen. Answer Y to this prompt. A new (blank) questionnaire will be displayed on the screen. Note that **`Rec = 2`** is displayed in the lower right corner of the screen. Enter data for all of the sample cases writing each to disk. Press F10 to return to the top-level menu.

The END key may also cause the **`Write data to disk (Y/N/<Esc>)?`** message to be displayed. If you answer N or ESC to this prompt ENTER will not save the case to the disk but present you with the same case which can then be edited.

Editing data

Data can also be edited after it has been saved to disk.

Select **ENTER data** from the **Programs** menu.

In response to the request for a data file type ONCHO (the .REC extension is appended automatically) and press ENTER three times. This file already exists and ENTER will not ask you for a questionnaire (.QES) file but display the questionnaire on the screen ready to receive data. The current record number is displayed in the lower right hand corner of the screen.

The F7 and F8 keys provide a means of moving from case to case within a data file. The F7 key moves one case back and the F8 key moves one case forward. The displayed case can be edited and then saved to disk by pressing the END key and answering Y to the prompt:

```
Write data to disk (Y/N/<Esc>)?
```

To continue entering new cases press CTRL N (i.e. hold down CTRL key and press N).

Use the F7 and F8 keys to move through the cases and review your work.

Edit a case by pointing to a variable with the cursor and changing its contents.

Save the edited case by pressing the END key and replying Y to the prompt:

```
Write data to disk (Y/N/<Esc>)?
```

Return to data entry mode by pressing CTRL N .

Searching for data

When you have many cases in a data file it can be time consuming to find a single case, or a particular set of cases. You can find a particular case (or set of cases) using CTRL F to place the ENTER module in *find mode*. When you press CTRL F the ENTER module displays a blank questionnaire. You can enter data into this questionnaire that *matches* the data held in the case(s) you wish to find (this example data is called the *search criteria* or the *search keys*). In this exercise we will search for members of household number 47.

Press CTRL F to place the ENTER module in find mode.

Move the cursor to the HHNO field.

Enter **`47`** into the HHNO field.

Press F3 to find the first matching case (i.e. the first case with HHNO = 47). ENTER finds all matching cases and displays them in a list. You can examine a case by moving the highlighted bar to it and pressing the ENTER key.

The following keys work in find mode:

Key	Function
F2	Find case by record number
F3	Find first matching case
F4	Find next matching case
SHIFT F4	Find previous matching case
F5	Print case
F6	Delete case
F7	Move back one case
F8	Move forward one case

You can edit cases in find mode. Press CTRL N to return to data entry mode and enter new cases. Press CTRL F to enter find mode and specify a new set of *search criteria*. Find mode is useful in finding individual records by their key or unique identifier variable.

Delete F6 does **not** actually delete data but marks it to be ignored by the ANALYSIS module of EpiInfo. F6 is a *toggle*. The first time you press F6 it will delete the record. If you press F6 on a record that has already been deleted it will be *undeleted*. Deleted records are marked with an asterisk `'*'` next to the record number in the bottom right corner of the data-entry screen.

Return to the top-level menu by pressing F10 .

Data Quality and Data Checking

Objectives of this section

By the end of this section you should be able to:

- ❑ List the types of error that can occur in data, how they arise, and how to avoid them.
- ❑ Understand the principles of data checking.
- ❑ Use the CHECK module to set up interactive checks.
- ❑ Double enter and compare data using the VALIDATE module.
- ❑ Double enter and compare data using the punch and verify method.
- ❑ Use the ANALYSIS module to implement batch checking of data.
- ❑ Use ANALYSIS and CHECK to implement consistency checks.

GIGO

GIGO is the most important acronym in computer science. It stands for *Garbage In Garbage Out*. Put simply, this means if your data are of poor quality then the results of any analysis will be unreliable. For this reason great emphasis is placed on checking data and minimising error.

Although the acronym GIGO originated in computer science it is important to realise that it applies to all data handling procedures at all stages of a survey.

The major roles of data management is to minimise error at all stages of a survey and not just at the computing stage. To minimise errors and check data effectively you should be aware of the types of error that can find their way into data.

Types of error

There are six main types of error that can find their way into your data:

Transposition e.g. 39 becomes 93. Transposition errors are usually typing or keyboard errors. These are common in the early stages of data entry and will tend to reduce dramatically as the data entry operators become more experienced with the data collection forms and data entry system. Transposition errors should be detected by *validation* after double entry, particularly if the data are entered by two different people.

Copying Errors e.g. 1 as 7, O as 0 etc. Copying Errors are another type of keyboard error and will also tend to reduce as the survey progresses. Another cause of copying errors is poorly filled in questionnaires. Copying errors should be detected by validation after double entry, particularly if the data are entered by two different people. Some copying errors will be eliminated by close supervision of data collection and ensuring that data are carefully and clearly recorded on the data collection forms.

Coding errors. Sometimes data are coded after collection. This involves adding a coding stage to the survey which can introduce error. Questionnaires should be piloted so that groups, treatments etc. can be coded directly onto the data collection form at the interview. Errors can also be minimised by using a consistent coding scheme.

Routing errors. The interviewer asks the wrong questions or asks questions in the wrong order. This is usually caused by a poorly designed questionnaire or badly trained data collection staff.

Consistency errors. Two or more responses are contradictory. These are usually caused by badly designed or worded questionnaires or badly trained data collection staff.

Range errors. Answers lie outside of probable, or possible, values. For most variables common sense or previous experience will decide the range (e.g. for haemoglobin measured in g/dl the lower limit would be 6 g/dl and the upper limit would be 18 g/dl).

Preventing errors

Preventing errors relies on an understanding of the type of errors that can occur. Whilst there are many specific types of error they can be broadly categorised as *response* error and *processing* error. Response error is the difference between the 'true' answer and what appears on the questionnaire. Processing error is the error that creeps in during the coding, data entry, and data analysis stages of a survey.

Response errors can arise because of questionnaire faults, errors made by the interviewer, or errors made by the subject. The best way of minimising response errors is a well designed and piloted questionnaire administered by well trained and motivated staff.

Processing errors are caused by the way data are handled after it has been collected. The principal way of preventing processing errors is extensive checking of data on the paper forms and on the computer. A well designed data entry system and well trained and motivated data entry staff are essential. Processing errors also occur at the data analysis stage so it is important to check the results of any recodes, transformations, and calculations. It is a good idea to perform procedures on a small sample of data first, so that the results can be analysed in detail for any mistakes.

Detecting errors in data

There are four methods of checking data. More than one method may be applied to the same data as each method has different strengths and weaknesses.

Manual Checking. Apply this test to a few completed questionnaires. It provides an initial assessment of data quality and highlights problems with the questionnaire and data collection process. Manual checking is particularly useful in detecting routing errors. Interviewers should check through each questionnaire immediately on completion. This is an important part of data checking as the error can be corrected in the field without the need for re-interviewing of the respondent. Data collection staff should be carefully trained in checking their work. Data supervisors should check through all questionnaires immediately after collection from interviewers. Checks should be made for completeness, correct routing, and clear handwriting. The researcher should check all questionnaires at the start of a survey. The researchers should continue to check samples of the questionnaires throughout the survey. Manual checking is very important because it takes place very early in the data *collection - entry - analysis* process. Problems can be detected and corrected in the field without the need for re-interview.

Checking during data entry (Interactive Checking). The CHECK module in EpiInfo allows for immediate detection and correction of problems with data as it is entered. Interactive checking is useful in picking up range, copying, consistency, and routing errors.

Checking after data entry (Batch Checking). It is possible to use the ANALYSIS module to check data after it has been entered. Producing frequency tables and scatter plots can highlight problems with data. It is also possible to perform range and consistency checks.

Validation (or Verification). This involves the data being entered twice into different files by different operators. The resulting files are then compared to each other to see if they are the same. Validation is useful in picking up transposition and copying errors. The VALIDATE program in EpiInfo provides validation. The ENTER program in EpiInfo provides *punch and verify* style validation. Data is entered twice into the same file. The second time it is entered each entry is compared with the data already entered. A message appears when entries do not match and data can be re-entered.

The type of checking you do will depend on the way your study is organised. If you are using experienced data entry staff who are not familiar with your data then interactive checking may slow the data entry process as each problem is referred to the data supervisor. A compromise approach is sometimes used. Data entry staff are warned of an error and asked to confirm that the data entered is the same as that on the data collection form. The operator can then correct or leave the error (so that it corresponds to what is on the data collection form). Errors are then detected with batch checking. This approach minimises the call on the supervisors time.

Correcting errors

Once errors have been detected the problem is what to do with them. If the error is detected in the field or soon after collection then it should be possible to re-interview and collect the correct data. Often this will involve asking only a few questions of the respondent.

If the error is detected later it may not be possible to re-interview. There are three options available for dealing with cases that have some errors in them:

> **Ignore the errors and use all the data**. This approach is fine **only** if you are absolutely certain that the errors are not systematic. Even if errors are not systematic they will add *noise* to the data and may create a *bias towards the null* in subsequent statistical analysis. This approach is very rarely used.
>
> **Mark erroneous data as missing and use the rest of the data**. This approach ensures that no obviously erroneous data are included in subsequent analysis. There is a risk that the data that is not obviously erroneous may be of poor quality. This is the most commonly used approach.
>
> **Throw out all erroneous cases**. This is the strong position and ensures that no erroneous data are included in subsequent analysis. Potentially erroneous data are filtered out on a 'guilt by association' basis. This approach may bias results depending who gets to decide on which records are to be excluded from subsequent analysis and is rarely used in practice.

Quality and validity

Most of the checks you can perform simply by examining questionnaires or data records relate largely to the validity of the data but not necessarily to the quality of the data. It is possible to check the quality of the data. Some examples that might be possible:

Question Redundancy: Ask important questions in different ways at different points in the interview. Responses can be compared. This is useful at the pilot stage and can help you decide which question elicits the most reliable response.

Confirmatory Checks: Incorporate checks within the questionnaire (e.g. ask about vaccination history, check against health card, ask to see vaccination scar).

External Records: Check a sample of questionnaires against external records (e.g. if asking about clinic visits then check against clinic records).

Expected distributions: It is likely that you will already know something (e.g. from the Census or previous studies) about the population under study. You should check that the distributions (e.g. age distribution, sex ratios, etc.) in your data are similar to those you expected.

Compare results from different interviewers: Use a variable on the data collection form that identifies the interviewer.

CHECK

The ENTER module provides a limited form of data checking. It will only allow you to enter numbers into numeric variables, valid dates into date variables, and Y or N responses into yes/no variables. Although you may use the ENTER module on its own it is better to use it with the CHECK module. The CHECK module contains some features that allow for interactive error checking, automatically coding of variables, and control over questionnaire routing. These functions are listed below:

Must-enter variables. CHECK allows you to specify that certain variables must be filled with a value other than missing.

Legal values. The input must match one of a specified list of values. The variable can be left blank unless it is designated as a *must-enter* variable.

Range Checks. The contents of a variable must lie between two bounding values. Text and upper case text variables may use range checks based on alphabetical order. CHECK allows you to mix range checks and legal values (e.g. for missing values).

Repeat variables. The variable will automatically hold the value for the previous case. This is useful for data that seldom changes (e.g. location codes).

Automatic coding. The input is compared against a pre-entered table of codes and 'phrases'. If a match is found another variable can be set to a code value.

Conditional jumps. CHECK allows you to check a variable for values that if found will cause the cursor to jump to a specified variable. If the tests fail the cursor moves to the next variable. Conditional jumps are used to implement questionnaire routing during data entry.

Programmed checks. CHECK also provides a block-structured programming language that allows you to program more complex checking procedures (e.g. consistency checks).

The CHECK module provides one type of data checking called *interactive checking* (i.e. checking the data as it is entered). The VALIDATE program allows for *batch* (i.e. all cases at once) checking of data that has been *double-entered*. Batch checks for range and consistency can be performed with the ANALYSIS module as can analytical checking for *outlier* (unusual) values.

Setting up checks

Select **`CHECK customize entry`** from the **`Programs`** menu.

Type the filename ONCHO (the .REC extension will be appended automatically) and press the ENTER key.

The ONCHO data entry screen appears (as with the ENTER program) but with a different set of function key prompts at the bottom of the screen. The most important function keys that work in CHECK are:

Key	Function
F1	Set **lower limit** for range
F2	Set **upper limit** for range
F3	**Repeat** value from previous case
F4	Data **must be entered** into this variable
F5	**Link** variables for codes and look-up values (used only with F8)
F6	Set a **legal** value
SHIFT F6	Display legal values for variable
CTRL F6	Delete the value in variable from the legal values list
F7	Set up **jumps** based on value in variable
SHIFT F7	Display jumps for variable
CTRL F7	Delete the jump for the value in variable
F8	Add **code** and **look-up value** for linked (F5) variables
SHIFT F8	Display codes and look-up values for linked (F5) variables
CTRL F8	Delete the code and look-up value in linked (F5) variables
F9	**Edit CHECK commands** associated with variable

Type **`1001`** into the IDNO variable. Press F1 to make this the *minimum* value allowed in this variable. Type **`3999`** into the IDNO variable. Press F2 to make this the *maximum* value allowed in this variable. With the cursor still in the IDNO variable press F4 to make this a *must-enter* variable. Note that a line at the bottom of the data entry screen changes to read:

```
IDNO: Valid values 1001 to 3999 You must enter data
```

to identify the checks you have added.

Move the cursor to the DATE variable and type **`01/05/91`** into the DATE variable. Press F1 to make this the minimum value allowed in this variable. Type **`01/06/91`** into the DATE variable. Press F2 to make this the maximum value allowed in this variable.

Move the cursor to the AGE variable and type **`12`** into the AGE variable. Press F1 to make this the minimum value allowed in this variable. Type **`70`** into the AGE variable. Press F2 to make this the maximum value allowed in this variable. Type **`99`** into the AGE variable. Press F6 to make this a *legal value* (**`99`** is used for missing data).

Setting up checks

Move the cursor to the SEX variable and type **M** into the SEX variable. Press F6 to make this a legal value. Type **F** into the SEX variable. Press F6 to make this a legal value.

Move the cursor to the TRIBE variable and type **1** into the TRIBE variable. Press F6 to make this a legal value. Type **2** into the TRIBE variable. Press F6 to make this a legal value. With the cursor still in the TRIBE variable press F3 to make this a *repeat* variable.

Move the cursor to the HHNO variable and type **01** into the HHNO variable. Press F1 to make this the minimum value allowed in this variable. Type **50** into the HHNO variable. Press F2 to make this the maximum value allowed in this variable. Type **99** into the HHNO variable. Press F6 to make this a legal value (**99** is used for missing data).

Move the cursor to the REL variable and type **1** into the REL variable. Press F1 to make this the minimum value allowed in this variable. Type **8** into the REL variable. Press F2 to make this the maximum value allowed in this variable.

Move the cursor to the LVILL variable and type **Y** into the LVILL variable and press F7 . Move the cursor to the OCC variable and press F7 again. The cursor will now jump automatically to the OCC variable when **Y** is entered into the LVILL variable during data-entry.

Move the cursor to the STAY variable and type **1** into the STAY variable and press F1 to make this the minimum value allowed in this variable. Type **70** into the STAY variable. Press F2 to make this the maximum value allowed in this variable. Type **99** into the STAY variable. Press F6 to make this a legal value (**99** is used for missing data).

Move the cursor to the OCC variable and type **0** into the OCC variable. Press F1 to make this the minimum value allowed in this variable. Type **8** into the OCC variable. Press F2 to make this the maximum value allowed in this variable.

Press F10 and confirm that you wish to save the checks.

Checking your checks

Use the ENTER module to enter some more data (make some up) into ONCHO.REC. Verify that the range, legal value, and conditional jumps work correctly.

Return to the top-level menu by pressing F10 .

Programming data entry

The CHECK module provides many more commands than are available using the function keys. The CHECK module creates a text file with the same name as the record (.REC) file but with the extension .CHK. The check (.CHK) file contains commands that are executed by the ENTER module at data entry time. The check (.CHK) file is a plain text file and can be edited using EPED or the simple editor built into the CHECK module.

Start CHECK. Type the filename ONCHO (the .REC extension will be appended automatically) and press the ENTER key.

Move the cursor to the IDNO variable and press the F9 **Edit Field** key. The screen changes to show the CHECK commands associated with the IDNO variable:

```
IDNO
  MUSTENTER
  RANGE 1001 3999
END
```

Press ESC to return to the main CHECK screen.

Point to each variable in turn and press F9 **Edit Field** to examine the CHECK commands associated with each variable. **Do not** alter any of the commands. Press ESC to return to the main CHECK screen.

By pressing F9 **Edit Field** within CHECK you can see that the CHECK commands are stored in blocks associated with each variable. The block has the general form:

```
VNAME
  INSTRUCTIONS
  INSTRUCTIONS
  INSTRUCTIONS
        .
        .
        .
  INSTRUCTIONS
END
```

Details of the commands available within CHECK can be found in the EpiInfo manual.

Specifying a KEY variable

CHECK provides a way of ensuring that a variable is unique (i.e. that no other case has the same value in a particular variable). This is useful for *key variables* and speeds up searches for data. It also ensures a measure of *data integrity* as it makes it difficult for the same person's data to be entered twice and thus counted twice in analysis and reports.

Move the cursor to the IDNO variable and press F9 **Edit Field**. CHECK displays the CHECK commands associated with this variable:

```
IDNO
  MUSTENTER
  RANGE 1001 3999
END
```

EDIT the CHECK commands associated with this variable to read:

```
IDNO
  MUSTENTER
  KEY UNIQUE
  RANGE 1001 3999
END
```

Press ESC to return to the main CHECK screen. Press F10 and confirm that you wish to save the checks. Now use the ENTER module to enter two cases (make up some data) with the same value in the IDNO variable into ONCHO.REC. ENTER will not accept the second case but pop up a window in the centre of the screen warning:

```
IDNO must be unique Press ESC
```

Press ESC and change the value in the IDNO variable and press END to save the changes. Answer Y to the prompt:

```
Write data to disk (Y/N/<Esc>)?
```

and press F10 to return to the top-level menu.

In versions of EpiInfo prior to 6.03 the ENTER module does not warn you of a duplicate KEY variable until you attempt to write the data to disk. It is possible to force ENTER to make a search of the data file for a duplicate IDNO immediately using the AUTOSEARCH command:

```
IDNO
  MUSTENTER
  KEY UNIQUE
  RANGE 1001 3999
  AUTOSEARCH
END
```

The CHECK command KEY used on its own (i.e. without UNIQUE) can be used to speed up searches without forcing the variable to hold unique values. When you use the CHECK commands KEY and KEY UNIQUE two new files with the same name as the record (.REC) file but with the extensions .IX and .DAT will be created. These files are called *index files* and speed up the way EpiInfo searches for data.

Data entry with legal value menus

The ENTER module allows you to enter data by picking options from menus. These menus are accessed by pressing F9 **Choices** in ENTER. This option only works with variables that have had legal values assigned to them with the CHECK module.

Start ENTER with the file ONCHO.REC. Enter some data (make it up). When you get to the TRIBE variable press F9 **Choices**. A menu will pop up. Select the data you want to enter by moving the highlighted bar using the ↑ and ↓ keys and pressing ENTER . Press F10 DONE and answer N to the prompt:

```
Write data to disk (Y/N/<Esc>)?
```

Using menus to enter categorical data that is not coded (e.g. full name of tribal group instead or 1 or 2) can speed up data entry and relieve data entry staff of unnecessary typing. When you use the F9 **Choices** option the pop-up menu will display a list of legal values with no clue as to what each code refers to. Luckily there is a CHECK command, COMMENT LEGAL, that allows us to use meaningful names when entering coded categorical data. Select CHECK from the **Programs** menu. Type the filename ONCHO (the .REC extension will be appended automatically) and press the ENTER key. Move the cursor to the TRIBE variable and press F9 **Edit Field**. CHECK displays the commands associated with this variable:

```
TRIBE
  REPEAT
  LEGAL
    1
    2
  END
END
```

EDIT the CHECK commands associated with this variable to read:

```
TRIBE
  REPEAT
  COMMENT LEGAL
    1 "MENDE"
    2 "OTHER"
  END
END
```

Press ESC to return to the main CHECK screen. Press F10 and confirm that you wish to save the checks. Start ENTER with the file ONCHO.REC. Enter some data (make it up). When you get to the TRIBE variable press F9 **Choices**. A menu will pop up. This time the menu has labels next to the codes. Select the data you want to enter by moving the highlighted bar using the ↑ and ↓ keys and pressing ENTER .

Press F10 DONE and answer N to the prompt:

```
Write data to disk (Y/N/<Esc>)?
```

You might like to add data-entry menus to the REL and OCC variables. The codes used for these variables are shown on the questionnaire on page 29 and in Appendix 1.

Limitations of CHECK

The CHECK module provides a very powerful tool for interactive data checking. Interactive checking is good at dealing with many sorts of error but cannot provide a complete shield against typing errors and digit transposition (e.g. 3 and 4 may both be valid occupation codes, 23 and 32 may both be valid ages). Also, data may be invalid on the data collection form and ENTER and CHECK will not allow the data to be entered. This can slow the data entry process as each offending case may need to be checked by the data supervisor as it is entered. This can be advantageous if data are entered soon after collection as unresolved queries can be referred back to the field. Interactive checking can be useful in bringing to light problems with data collection.

Another problem with interactive checking is that it is *prescriptive.* It does not allow any data that violate the data checking rules to be entered. This means that interactive checking is best used when there can be no controversial-but-correct (real-but-unlikely) values.

EpiInfo provides another method of data checking called *validation* (also known as *verification*). Validation involves data being entered twice into two separate files by two different data entry operators. The two files are then compared and any discrepancies can be checked against data collection forms.

With validation it is important that the data are entered by two different operators. This is to overcome errors caused by digit preferences (the same operator will tend to make the same mistakes in the same situations).

Validation works best when each case has a key or unique identifier variable. If a key variable is not used then data must be entered into the two files in the same order. This is difficult to guarantee so **always** use a key or unique identifier variable.

Double entry and validation

Use ENTER to create a data file called ONCHO1A.REC using ONCHO.QES. Do not enter any data but leave EpiInfo and return to DOS by pressing F10 twice. At the DOS prompt enter the command:

```
copy oncho.chk oncho1a.chk
```

to create a .CHK file with a filename appropriate to ONCHO1A.REC. We can copy the .CHK file as the files ONCHO.REC and ONCHO1A.REC have the same structure. The .CHK file contains the CHECK commands for a particular data file. The EpiInfo file types used so far are:

.QES	Defines the structure of the data file and the layout of the data-entry screens
.REC	Contains the data
.IX	Index file for KEY and KEY UNIQUE fields (optional)
.DAT	Index file for KEY and KEY UNIQUE fields (optional)
.CHK	Contains data CHECKing commands (optional)

Start EpiInfo and then start ENTER with the file ONCHO1A.REC. Fill in the blanks on the questionnaire using the data from the sample cases:

IDNO	DATE	AGE	SEX	TRIBE	HHNO	REL	LVILL	STAY	OCC
1001	01/05/1991	30	M	2	29	1	N	5	1
1002	02/05/1991	26	F	2	29	2	N	3	1
1003	02/05/1991	34	F	2	23	1	Y		7
1004	02/05/1991	23	M	2	12	1	N	2	4
1005	03/05/1991	19	F	1	47	3	Y		5

Enter data for the sample cases, writing each to disk. Press F10 to return to the top-level menu. Use ENTER to create a data file called ONCHO2A.REC using ONCHO.QES. Leave EpiInfo and return to DOS. Use the COPY command to create ONCHO2A.CHK. Start EpiInfo and then start ENTER with the file ONCHO2A.REC. Fill in the blanks on the questionnaire using the data from the same sample cases. Make two or three deliberate mistakes when entering data into this second file. Press F10 to return to the top-level menu.

Double entry and validation

Select **VALIDATE duplicate entry** from the **Programs** menu. In response to the prompt:

File 1

type ONCHO1A (the .REC extension is appended automatically) and press ENTER . In response to the prompt:

File 2

type ONCHO2A and press ENTER . Press ENTER again to confirm that you would like to send the output to a window on the screen. As the output is to be sent to a window on the screen the:

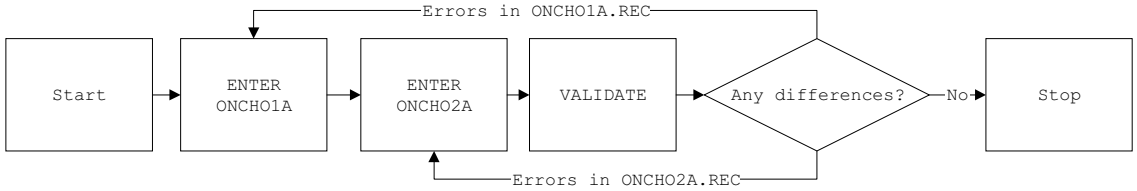
Output file name

prompt should be left blank. Press ENTER again to exclude any deleted records from the comparisons. With the cursor in the box next to

Use unique IDs

press the SPACE bar. A check-mark **[X]** should appear in the **Use unique IDs** box. Press ENTER twice to start the file comparison. Because you asked VALIDATE to **Use unique IDs** a list of fields in the data files will pop-up. There should be a diamond shaped check-mark **(♦)** next to the IDNO field. Press ENTER to confirm that this is the correct unique ID or *match field*. Press ENTER twice to start the file comparison. VALIDATE will report any discrepancies (by record number) showing the different values in each file.

Make a note of the problems. Press F10 to quit VALIDATE. Use ENTER to edit data in the two files and then use VALIDATE to compare the two files again. You may need to repeat this process until you have eliminated all of the discrepancies between the two files:



VALIDATE checks whether data entered into two separate files is the same. It does not check the validity of the data. CHECK allows you to check the validity of the data as it is entered. To ensure good quality data you should use both the CHECK and VALIDATE modules.

Double entry may seem an unnecessary step that adds time and expense to a study but it is an **essential** step in ensuring data quality and should always be performed. Some researchers hold the view that errors in data are random events and will not lead to bias in large studies. This is a mistaken view as many typing mistakes are caused by a systematic *digit preference* by the data entry operator and can lead to a *systematic bias* in the data.

Punch and verify

Punch and verify combines the convenience of interactive checking with the rigour of double entry and validation. With punch and verify, data are entered into the same file twice and discrepancies are detected as they occur. Enter the following sample cases into ONCHO1A.REC:

IDNO	DATE	AGE	SEX	TRIBE	HHNO	REL	LVILL	STAY	OCC
1006	03/05/1991	22	F	1	47	3	Y		6
1007	04/05/1991	39	M	2	13	3	Y		2

Press F10 to return to the top-level menu. Start ENTER and specify ONCHO1A.REC (again) as the data file and press ENTER . Specify option:

```
4. Reenter and verify records in existing data file
```

and press ENTER . ENTER now asks:

```
Names of Identifying Fields
```

Type IDNO and press ENTER twice to start entering data. Enter the data for the first sample case (IDNO = 1006). Make the deliberate mistake of entering AGE as **21**. ENTER recognises a discrepancy and displays a warning message:

```
Field Mismatch : AGE
You have entered a value which is different from
the value originally entered. Press <Esc> to try
again or <F9> to compare the old and new values.

Press <Esc> or <F9>
```

If you press ESC now you will be returned to the data entry screen. Do not press ESC . Instead press F9 . ENTER now displays:

```
Field Value Mismatch : AGE
Original:
  <F1>22
New:
  <F2>21
Press <F1> to keep Original value or <F2> to
accept New value.

Press <F1> or <F2>
```

You can now choose which is the correct value for AGE. In this case the correct value is **22** so press F1 . F1 sets the value to the first value entered and F2 sets the value to the second value entered.

Enter the rest of the data for both sample cases. Make some deliberate mistakes so that you can practice working with ENTER's punch and verify facility.

Data consistency

So far we have used ENTER, CHECK and VALIDATE to ensure the validity of individual data items. We have not checked whether data are *internally consistent*. *Consistency checks* are used to identify cases where two or more responses are contradictory. You can perform consistency checks with the CHECK module (for on-line or interactive checking) and ANALYSIS (for batch checking after data has been entered). You **cannot** perform consistency checks with the VALIDATE module.

It is up to you whether to use interactive consistency checking as this may slow the data entry process. Interactive checking is most useful if data are entered by field staff who collected it (familiar with the data but slow typists). If data are entered by clerks (fast typists but relying on the supervisor for all problems with the data) it can slow data entry. You should always perform batch consistency checks after the data has been entered. Typically batch checking will also apply the standard checks (range, legal values, etc.) as well as consistency checks.

The onchocerciasis data requires little consistency checking. One check we could apply is to check that no data (other then the 'missing' value) is entered into the STAY variable if **Y** is entered into the LVILL variable. Instead we will create a new data file that requires more consistency checks. Start EPED and set **WW/TXT/QES mode** to **QES**. On the blank screen type the sample questionnaire:

```
Identification Number                  {IDNO} ###

Age of Mother                          {AgeM} ##
Age at First Parity                    {AgeP} ##
Number of Live Births                {Births} ##
```

Check your work carefully. When you are satisfied select **Save** by pressing F9 . When prompted give the filename MUMS.QES and press ENTER . The questionnaire file (MUMS.QES) will be saved to disk. Press F10 to return to the top-level menu.

With this small number of variables there are two consistency checks that need to be made. The age of the mother (AGEM) should be greater than or equal to the age at first parity (AGEP). A check also needs to be made on the number of live BIRTHS against the number of 'fertile years' (AGEM - AGEP). It may seem sensible to query a *birth interval* of less than two years: The number of *fertile years* divided by BIRTHS should not be less than two:

```
(AGEM - AGEP) / BIRTHS >= 2
```

To correct for the first birth (where AGEM - AGEP = 0) we need to add two to AGEM to perform consistency checks in this way:

```
(AGEM + 2 - AGEP) / BIRTHS >= 2
```

Consistency checks

Use ENTER to create MUMS.REC (using MUMS.QES). Enter the following sample cases:

IDNO	AGEM	AGEP	BIRTHS
101	30	21	2
102	22	25	1
103	25	18	2
104	31	20	6
105	19	18	1
106	34	19	3
107	35	18	4
108	24	24	8
109	34	16	4
110	24	19	7

You may notice that the data contains some obvious errors. **Do not** attempt to correct these errors. Press F10 to return to the top-level menu.

Now that we have entered the data we will use the ANALYSIS module to perform batch checking.

ANALYSIS

ANALYSIS is the data analysis module of EpiInfo. Using ANALYSIS you can use simple commands to produce lists, frequencies, statistics, and graphs. In this exercise you will use analysis to perform batch-checking of data.

Position the cursor bar over the menu option **`ANALYSIS of data`** on the **`Programs`** menu. Press ENTER to select ANALYSIS.

Once ANALYSIS starts the screen is divided into several distinct sections. The upper section is headed **`Output`** and the lower section is headed **`Commands`**. At the top of the screen are two lines giving the status information:

```
Dataset: <None>                                Free memory 353K
Use READ to choose a dataset
```

The status information shows the name of the current data file and the amount of free memory in the system (the exact figure will depend on how your system has been configured).

The bottom line of the screen shows the *function keys* available from within ANALYSIS:

Key	Function
F1	Help
F2	Display menu of commands
F3	Display menu of variables in the current dataset
F4	Browse
F5	Toggle printer on and off
F9	DOS shell
F10	Quit ANALYSIS

The F1 key provides help. Press F1 at any time for help on what you are doing. The help provided is *context sensitive* so if you type a command and then press F1 ANALYSIS will respond with help for that particular command.

A menu of commands is available by pressing the F2 key.

A menu of variables in the data file is available by pressing the F3 key.

Entering commands in ANALYSIS

There are two methods of entering commands in ANALYSIS. You may either type the command directly at the keyboard or choose commands and variables from menus.

Pressing the F2 key brings up a menu of available commands. To select a command from the menu you must first move the highlighted bar using the arrow keys. When the highlighted bar is over the command you wish to use press ENTER to select it. To execute the command press ENTER again. With most commands you will need to specify at least one variable.

If you select the wrong command by mistake press ESC to clear the command line.

Pressing the F3 key brings up a menu of variables in the current dataset. Variables can be selected from the menu. To select a variable point to it using the arrow keys and press ENTER . To select a group of variables point to each in turn and press the + key. When you have selected all the variables you need press ENTER . If you select a variable by mistake you can deselect it using the − key.

Getting Help on Commands

There are two ways of getting help on commands in ANALYSIS. If you press F1 before you have typed a command then ANALYSIS will present a menu with options for help on general topics and specific commands. Options are selected from the menu in the usual way. If you want help on a specific command then type the command (e.g. FREQ) and press F1 . ANALYSIS will respond with help on the command you typed.

If there is more than one screen of help on a particular topic or command then **PgDn** will be displayed in the bottom right hand corner of the help window. Pressing PG DN will display the next screen of information.

Press ESC to return to ANALYSIS.

Choosing a data file

Before performing any checks or analysis you must first tell ANALYSIS which data file you want to work with (in this case MUMS.REC). The ANALYSIS command to retrieve a data file is READ followed by the name of the file. Once you have retrieved a data file the status lines on the screen will change to show the name of the dataset and the number of cases in the dataset.

Enter the command:

```
read mums.rec
```

and press ENTER .

For data files created using the ENTER module of EpiInfo you do not need to specify the .REC extension.

ANALYSIS can also read files created with dBase (or any application that can write a dBase format file). To read a dBase file you need to specify the .DBF extension.

If you cannot remember the name of the file you want to work with then issue the READ command on its own (i.e. without a file name). ANALYSIS will respond with a menu listing all the .REC files in the current directory.

Truth Tables

Before issuing commands to check data, it is good practice to list the types of errors you are looking for. To do this we use a *truth table* or *error table* to list the potential errors that can occur in the data. Because a consistency error can be caused by an error in a single data item checks should also be performed on single data items:

Condition	Type	Code
`AGEM < 14 or AGEM > 40`	Range	`AGEM`
`AGEP < 14 or AGEP > 40`	Range	`AGEP`
`AGEP > AGEM`	Consistency	`AGEP > AGEM`
`(AGEM + 2 - AGEP) / BIRTHS < 2`	Consistency	`BIRTH INT`
`BIRTHS < 1 or BIRTHS > 10`	Range	`BIRTHS`

Each possible error has a code assigned to it. We will use this code to assign error values to individual cases based on the types of error present in each case. The codes chosen simply reflect the nature of the error.

Error values

In the MUMS.REC data file there is no variable to hold the error codes. We need to create a new variable that will contain the error code for each case. The DEFINE command is used to create a new variable. The DEFINE command when used on its own creates a new variable of the numeric type. To create other variable types use the same special characters that define variable type and length in questionnaire (.QES) files. The command DEFINE ERR <AA> creates an upper case text variable of length two. Once we have defined a new variable (ERR) we need to give it an initial value (" " = no errors) and instruct ANALYSIS to assign error values for each case. The ERR variable should be long enough to hold all possible combinations of the error codes (40 characters in this example). You may need to create more than one variable or use shorter codes to check a file with lots of variables.

Enter the following commands one at a time:

```
define err <AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>
err = " "
if agem < 14 or agem > 40 then err = err + "AGEM, "
if agep < 14 or agep > 40 then err = err + "AGEP, "
if agep > agem then err = err + "AGEP > AGEM, "
if (agem + 2 - agep) / births < 2 then err = err + "BIRTH INT, "
if births < 1 or births > 10 then err = err + "BIRTHS, "
```

The IF ... THEN command sequence allows us to check for specific conditions and perform an action based on the outcome of that check.

All ANALYSIS commands work with all (selected) cases in the data file. This set-at-a-time processing is common to all statistical packages.

As the commands to perform batch checking will be used frequently it may be best to include them in a program file for periodic use on batches of cases. A program file is simply a list of instructions stored in a text file.

ANALYSIS provides an easy way of developing programs. First enter the commands interactively (as you have done already). After you have entered the commands use the SAVE command to save them in a program (.PGM) file. Program files are ordinary text files and can be edited using EPED or any other text editor. ANALYSIS program (.PGM) files should have the extension .PGM. The SAVE command usually saves the last 20 commands entered. If you want to save more than 20 commands then you should enter the command SET CMDSTACK = *nn* where *nn* is the number of commands you wish to save (up to a maximum of 99). You should use SET CMDSTACK **before** entering any of the commands you wish to save. The following ANALYSIS command will save previously entered commands to a file called BATCH.PGM:

```
save batch.pgm
```

You can *run* these commands again with the ANALYSIS command:

```
run batch.pgm
```

Listing error values

Enter the command:

```
list idno err
```

to list the error values for each case. Enter the command:

```
list
```

and compare the error values against the other variables in each case.

Press F10 to return to the top-level menu.

Once you have a list of errors for each case it is a relatively simple matter to check data against data collection forms, return to the field (if required) and correct the data.

You can use the ROUTE command to send a copy of the output to a file on disk (ROUTE *filename*) or to the printer (ROUTE PRINTER). Make sure that the printer is connected and 'on-line' before issuing the ROUTE PRINTER command. To stop sending output to a file or to the printer issue the command ROUTE SCREEN. The ROUTE command will only redirect the output of subsequent commands. If you want a record of an ANALYSIS session then the ROUTE command should be the first command you issue. It is best to ROUTE to a filename and then edit the output (using EPED) before printing.

Another way of viewing the error values is to use the BROWSE command (F4) or the UPDATE (F6) command. The UPDATE (F6) command allows you to edit the offending data values directly and see the effect of any edits on the value of the ERR variable.

You can scroll through the output generated by previous commands using the PG UP and PG DN key. CTRL PG UP and CTRL PG DN scrolls through the output a line at a time.

You can scroll through previously entered commands using the ↑ and ↓ keys.

Every EpiInfo data (.REC) file has a set of *system variables* which you do not see during data-entry but are visible to ANALYSIS. These variables are:

Variable	Contents	Definition
RECNUMBER	Record Number	##########
RECDELETED	Is this record deleted?	<Y>
RECVERIFIED	Has this record been verified?	<Y>
SYSTEMDATE	Current Date	<MM/DD/YYYY>
SYSTEMTIME	Current Time	<AAAAAAAAAAA>

You can use the RECVERIFIED variable to produce a list of records that have **not** been double-entered and verified (with either VALIDATE or ENTER's punch-and-verify facility) using the following ANALYSIS commands:

```
select recverified = "N"
list recnumber
```

Interactive consistency checks

It is also possible to use the CHECK program to perform consistency checks as data are entered. Whether you choose to do this depends on the way your study is organised. Interactive checking can slow the data entry process as each offending case may need to be checked by the data supervisor as it is entered. This can be advantageous if data are entered soon after collection. Unresolved queries can be referred back to the field. Interactive checking can be useful in bringing to light problems with data collection. This may be used to advantage in the early stages of a study.

Start CHECK and specify the file MUMS (the .REC extension is appended automatically) and press ENTER . Use the function keys to set up the following range checks:

Variable	Lower	Upper
IDNO	101	499
AGEM	14	40
AGEP	14	40
BIRTHS	1	10

Move the cursor to the variable AGEP and press F9 **Edit Field** and change the CHECK commands associated with this variable to read:

```
AGEP
  RANGE 14 40
  IF AGEP > AGEM
  THEN
    HELP " QUERY:  AgeP > AgeM "
    GOTO AGEP
  ENDIF
END
```

Press ESC to return to the CHECK main screen.

The IF ... THEN ... ENDIF command sequence allows us to test a case for a specific set of conditions and perform an action based on the outcome of the test.

The HELP command causes ENTER to suspend data entry and pop up a window on the screen containing the text enclosed in the quote (") characters. To resume data entry the data entry operator must press the ESC key.

The GOTO command causes ENTER to move the cursor to a specified variable. The GOTO command is used here to ensure that only valid data are entered by going back to the variable where the error was detected (this is not necessarily the offending variable).

Interactive consistency checks

Move the cursor to the BIRTHS variable and press F9 EDIT and change the CHECK commands associated with this variable to read:

```
BIRTHS
  RANGE 1 10
  IF (AGEM + 2 - AGEP) / BIRTHS < 2
  THEN
    HELP " QUERY: Birth interval < 2 "
    GOTO BIRTHS
  ENDIF
END
```

Press ESC to return to the main CHECK screen. Press F10 to save the checks and return to the top-level menu. Start ENTER and specify the MUMS file. Enter the sample data and verify that the consistency checks work:

IDNO	AGEM	AGEP	BIRTHS
111	30	21	2
112	22	25	1
113	25	18	2
114	31	20	6
115	19	18	1
116	34	19	3
117	35	18	4
118	24	24	8
119	34	16	4
120	24	19	7

Press F10 to return to the top-level menu.

Note that interactive consistency checks apply only to new data as it is entered and **not** to data already in the file. You should use ANALYSIS to check data that is already in the file.

The IF ... THEN ... ENDIF command sequence tests for a specific set of conditions and perform an action based on the outcome of the test.

The HELP command causes ENTER to suspend data entry and pop up a window on the screen containing the text enclosed in the quote (") characters. To resume data entry the data entry operator must press the ESC key.

The GOTO command causes ENTER to move the cursor to a specified variable and is used here to ensure that only valid data are entered by going back to the variable where the error was detected. This example highlights a problem with using interactive data-checking. A birth interval of less than two years is a *possible* correct value for BIRTHS (e.g. in the case of twins) but our checks will not allow for this. Interactive checks are *prescriptive* and are best used when there can be no controversial-but-correct values. If you get stuck using interactive checking you can usually force ENTER to accept 'faulty' data by pressing the END key.

Other forms of data checking

The ANALYSIS module can be used to provide another form of data checking using summary statistics. The following types of data checks are possible:

> **Duplicate** and **missing cases**. If you use the CHECK command KEY UNIQUE then it is unlikely that any duplicate cases were entered. The VALIDATE module will also point out duplicate cases. As a final check for duplicate cases produce a frequency table (using the FREQ command in ANALYSIS) of the unique identifier variable. This should produce a very long table with **no** counts greater than one. Check the table for consecutive values of the unique identifier variable. If values are not consecutive then a case may not have been entered.
>
> **Expected distributions**. It is likely that you will already know something (e.g. from the Census or previous studies) about the population under study such as age distribution, sex ratio etc. You should check that the distributions in your data are similar to those you expected using the FREQ and MEANS commands in ANALYSIS.

Representing data in graphical form often makes it easy to identify *outliers* (extreme values). The ANALYSIS module provides the PIE, BAR, HISTOGRAM, and SCATTER commands for this purpose. The HISTOGRAM command is useful in gauging the shape of the distribution of your data (e.g. *normal*, *uniform*, etc.).

Data Management

Introduction and objectives

In this section we will discuss some aspects of data management that can be performed with EpiInfo. Data management can be split into two broad categories and is performed by two modules of EpiInfo.

> The first category creates and manipulates individual variables in a dataset. Examples are classifying people whose ages are known into discrete age classes, and transforming malaria parasitaemia from parasites per 200 leucocytes to parasites per µl. These functions are performed by the ANALYSIS module.
>
> The second category deals with the use of several data sets simultaneously. Examples are appending a file that contains this year's data to a file containing last year's and earlier data, and linking a file containing laboratory results to a file containing field results. These functions are performed by the MERGE module (some of these functions are also available within the ANALYSIS module).

This section is designed to demonstrate some common tasks involved in the management of real datasets. The material covers:

- ❑ Recoding missing value codes to system missing values.
- ❑ Defining and creating new variables.
- ❑ Merging datasets.
- ❑ Follow-up and recall.

Background

Onchocerciasis, or more familiarly river blindness, is a common and debilitating disease of the tropics. It is mainly a chronic disease, affecting the skin and eyes of afflicted persons. Its pathology is thought to be due to the cumulative effects of inflammatory responses to immobile and dead microfilariae in the skin and eyes. Microfilariae are tiny worm-like parasites, deposited in the skin by flies that infest some tropical rivers. The worms are detectable by microscopic examination of skin samples, usually snipped from around the hips; severity of infection is measured by counting the average number of worms per microgram of skin examined.

A double-blind-placebo-controlled trial was designed to study the clinical and parasitological effects of repeated treatment with a newly developed drug called Ivermectin. Subjects were enrolled from six villages in Sierra Leone (West Africa), and initial demographic and parasitological surveys were conducted between June and October 1987. Subjects were randomly allocated to either the Ivermectin treatment group or the placebo control group. Randomisation was done in London. Neither the clinical survey teams nor the study population knew the meaning of the codes used to label the two treatments.

Follow-up parasitology and repeated treatment was performed for five further surveys at six monthly intervals. The principal outcome of interest was the comparison between microfilarial counts both before and after treatment, and between the two treatment groups.

The data

As a demonstration, we will use the onchocerciasis dataset, introduced earlier in this book. Detailed information about the data can be found in Appendix 1.

Orientation

Data from this trial were collected and entered into an EpiInfo data (.REC) file using the ENTER module. The CHECK module was used for validation of legal codes at data entry. Data were entered twice and validated using the VALIDATE module.

Several files were used to store the data:

Filename	Contents
DEMOG_1.REC	Baseline demography file
DEMOG_2.REC	Baseline demography file
DEMOG_3.REC	Baseline demography file
MICRO.REC	Microfilariae counts
BLOOD.REC	Results from blood samples
TMTCODES.REC	Randomisation codes

Full details of each file's structure and coding scheme are shown in the Appendix 1.

Start ANALYSIS and issue the command:

```
read demog_1.rec
```

to retrieve the first baseline demography file. Press F4 to BROWSE through the file. Take note of the variable names and values. Use the arrow keys to move through the file. Press the ESC key to return to the ANALYSIS main screen.

Examine the contents of all these demography files (DEMOG_1.REC, DEMOG_2.REC, and DEMOG_3.REC). Check that these files have the same structure. Examine the contents of the other data files (MICRO.REC, BLOOD.REC, and TMTCODES.REC).

Another way of looking at the structure of a data file is to use the VARIABLES command. The VARIABLES command displays a list of variables in the current data file with information about their type and length. The F3 key also displays a list of variables but without information about their type and length.

Concatenation

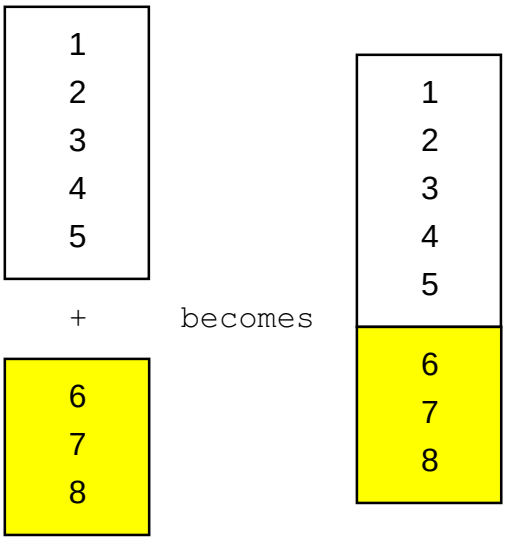
Demographic questionnaires were filed by IDNO in the field office. The data entry task was simplified by entering date into three separate files for IDNOs 1-999, 1000-1999 and 2000+. The same .QES file was used, but the end result was three different files. We want to join the three files end-to-end to create a single file for analysis.

The process of joining files end-to-end is called *concatenation*. The MERGE module in EpiInfo will allow us to concatenate (i.e. join together end-to-end) different EpiInfo files. At each pass, the MERGE module will concatenate two files to make a combined file.

Start the MERGE module. Fill in the prompts as follows:

File 1:	**DEMOG_1**
File 2:	**DEMOG_2**
Output file:	**TEMP**
Merge options:	**Concatenate**

The MERGE module will create a new EpiInfo data file, called TEMP, that contains the records from both input files:



Press ENTER to clear the MERGE information box.

Concatenation

Repeat the MERGE but this time concatenate TEMP.REC file with DEMOG_3.REC. Call your output file DEMOG.REC. Press ENTER to clear the MERGE information box. Press F10 to exit MERGE and return to the top-level menu.

Start the ANALYSIS module. Type the command:

```
read demog.rec
```

And press the ENTER key. EpiInfo will display the total number of records in the complete file. There should be 1625 records.

You may get a message **RUNTIME ERROR 101** when creating new files. This means that your disk (or network allowance) is full. Return to DOS and delete some *unwanted* files.

It is also possible to concatenate files with ANALYSIS. To do this you should list the names of the files to be concatenated in the READ command:

```
read demog_1.rec demog_2.rec demog_3.rec
```

The files **must** have **identical** structures. The files are concatenated temporarily and the end-to-end join will be broken when you quit ANALYSIS. You can make the join permanent by creating a new file using the ROUTE *filename* and WRITE RECFILE commands:

```
read demog_1.rec demog_2.rec demog_3.rec
erase demog.rec
route demog.rec
write recfile
```

If you use this method of concatenating files make sure that the output file does not already exist (as is done with the ERASE command in the example above) before writing to it as WRITE RECFILE will add records to an already existing file and this may not be what you want to happen. If the output file already exists and has a different structure from the files you are trying to concatenate then the resulting file will be unusable.

Both MERGE and ANALYSIS require that the files to be concatenated have **identical** structures.

The file TEMP.REC is no longer required and can be safely deleted.

Missing values

Most datasets will contain missing values. Missing values are often recorded using impossible values for each field. In the BLOOD.REC data file, a PCV of 99 is not possible, so this value is used to *flag* a missing value. Missing values have to be treated carefully during analysis or they will 'pollute' the results. Appendix 1 shows the missing value codes for each variable in the dataset. Most statistical packages (including EpiInfo) have ways of removing missing values from analysis.

In the ANALYSIS module, enter the command:

```
read blood.rec
```

Press F4 to browse through the data. There are five variables:

Variable	Contents	Missing
IDNO	Respondent ID number	
SR	Survey round	
EOSIN	Eosinophil count	9999
PCV	Packed cell volume	99
MPS	Presence of malaria parasites	9

Press ESC to return to the ANALYSIS main screen. To produce a histogram of the PCV variable, enter the command:

```
histogram pcv
```

The missing values are to the right hand side of an otherwise symmetric bell-shaped distribution. If the missing values are included they will pollute the analysis. Enter the command:

```
means pcv /n
```

These commands will show the mean, standard deviation and quartiles of the distribution of PCV. The standard deviation is huge, and the quartiles are wrong. Use HELP F1 and select MEANS to find out what is the purpose of the **'/N'** at the end of the MEANS command. To remedy this, use the recode command:

```
recode pcv 99=.
```

The period (.) at the end of this command is important. It is EpiInfo's special code for a missing value. Repeat the HISTOGRAM and MEANS commands. The missing values are now excluded from the analysis. Recode the EOSIN and the MPS variables and save the cleaned data with the commands:

```
recode eosin 9999=.
recode mps 9=.
erase bloodcln.rec
route bloodcln.rec
write recfile
```

Creating and transforming variables

The principal outcome variables in this study are:

1. the presence or absence of microfilariae in skin samples

2. the average density of microfilariae per milligram of skin

In the data file MICRO.REC the variables MFRIC and MFLIC contain the numbers of microfilariae observed in each right and left skin sample. The variables SSRIC and SSLIC record the diameters of the right and left skin samples. It is common to convert the diameter of a skin snip to its mass using the formula:

```
(mass in mg) = 0.495 * (diameter in mm) - 0.076
```

In ANALYSIS issue the READ command to access the file MICRO.REC:

```
read micro.rec
```

Use BROWSE F4 to familiarise yourself with this file.

Recode missing values to the EpiInfo missing values using the commands:

```
recode mfric 999=.
recode mflic 999=.
recode ssric 0=.
recode sslic 0=.
```

We need to define a new variable to contain the total number of microfilariae in both samples. We should define this new variable before it is used:

```
define mftot ####
```

Note that the same special characters used in questionnaire (.QES) files are used with the DEFINE command to specify variable type and length. Issue the following command to find the sum of the left and right microfilariae counts and store the result in the newly defined variable MFTOT:

```
mftot = mflic + mfric
```

Note that MFTOT will be set to missing if **either** MFLIC or MFRIC contain missing values.

Next, define a binary variable to indicate the presence of any microfilariae:

```
define mfany #
recode mftot to mfany 0=0 1-hi=1
```

HI is a special keyword that stands for *all higher value*s. There is also a keyword LO that stands for *all lower values*.

Creating and transforming variables

To see the overall percentage positive for microfilariae across all samples issue the command:

```
freq mfany
```

To get similar percentages for each survey separately issue the commands:

```
set lines = on
set percents = on
tables sr mfany
```

It is wise to save your data (.REC) file regularly if you are doing many recodes, in case of error or system failure. Use a different filename (or the ERASE command) if you do not wish to append (add to the end of) an existing file. For example, at this stage:

```
erase mftemp.rec
route mftemp.rec
write recfile
```

Press F10 to quit ANALYSIS. Re-start ANALYSIS and issue the command:

```
read mftemp.rec
```

A better approach is to use a program (see page 97) to perform recodes and transformations.

Creating and transforming variables

Create a new variable called MFDENS to hold the density per mg of microfilariae. We want a numeric variable with one decimal place so enter the commands:

```
define mfdens ###.#
mfdens = mftot / (0.4955 * (sslic + ssric) - 0.076)
```

To see the mean density of microfilariae for all samples issue the command:

```
means mfdens /n
```

To get the mean density separately for each survey round issue the command:

```
means mfdens sr /n
```

Save the results of your recoding in a new file with the commands:

```
erase microcln.rec
route microcln.rec
write recfile
```

Press F10 to return to the top-level menu.

The comparison between surveys is **not** valid because the same people were used at both surveys. Later we will rearrange this data file so that comparisons can be made *within* subject.

File splitting

Start ANALYSIS and READ the recently created microfilariae file, MICROCLN.REC. Browse through the data to check that it contains what you expect. We want to select only those results for the baseline survey. Enter the commands:

```
select sr = 1
```

Note the selection criterion is shown in the status line at the top of the screen. Create a new file called MICRO1.REC to hold data for the first survey round. Enter the commands:

```
erase micro1.rec
route micro1.rec
write recfile idno sr mfany mfdens
```

As we only need the newly created variables MFANY and MFDENS in later analyses we can save disk space and processing time by writing to disk only the variables we want. Create a similar file for the data from the fifth survey. First clear the current SELECTion criteria by entering the command:

```
select
```

SELECT only the results for the fifth survey by entering the command:

```
select sr = 5
```

We must be careful to use different names for the variables we save, so that when the time comes to merge the data files, we can differentiate between the initial and the later results. Enter the commands:

```
define mfany5 #
define mfdens5 ###.#
mfany5 = mfany
mfdens5 = mfdens
erase micro5.rec
route micro5.rec
write recfile idno sr mfany5 mfdens5
```

We have split the file MICRO.REC into two files called MICRO1.REC and MICRO5.REC:

Use READ and F4 to browse through MICRO1.REC and then through MICRO5.REC. How many records are in each file? Why are there fewer records in MICRO5 than in MICRO1? Is this a problem for the analysis?

Press F10 to return to the top-level menu.

File merging

A proper analysis of the microfilariae results from this study requires a link to be made between the before and after results for each subject. In comparing the density of infection we should consider the *difference* in density after treatment compared with that before treatment. To do this we need to have the results linked by subject identifier. EpiInfo provides two ways to perform such a link. Using the MERGE module, we can create a permanent linked file, with the before and after results stored in the same file. Another, sometimes more efficient, way to match the before and after results is to use the RELATE command in the ANALYSIS module.

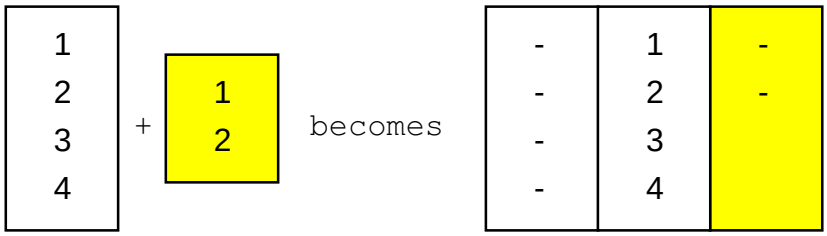
From the **`Programs`** menu, select the MERGE module. Fill in the prompts as follows:

File 1:	**MICRO1**
File 2:	**MICRO5**
Output file:	**MICRO15**
Merge options:	**Join**

Press the ENTER key twice. With the cursor in the check-box next to 'IDNO' press the SPACE bar to mark this field as the match field. Press ENTER . With the cursor in the check box next to:

`Include unmatched records`

press the SPACE bar. Press the ENTER key twice. MERGE will join the files MICRO1.REC and MICRO5.REC side-by-side matching each record using the IDNO variable and create a new file called MICRO15.REC that contains microfilarial counts for each subject from both surveys:



Press ENTER to clear the MERGE information box. Press F10 to return to the top-level menu. Use the ANALYSIS module to READ MICRO15. Browse F4 through the file and check that the file contains information from both the first and the fifth surveys.

Two files can be MERGEd side-by-side **only** if they share a common linking variable that has the same name, type, and length in both files. The MERGE operation can be *inclusive* or *exclusive* depending on whether you want all cases in the output file or just those with matching cases in both files. It is important to differentiate between files with *unique* matching identifiers (i.e. each file has only one case for each identifier) and files with multiple matches (e.g. mothers' and children's files). MERGE can deal with either situation but the file with unique identifiers must be named first on the MERGE input screen.

File merging

The MERGE module was used to link two files together side-by-side. The same operation can be performed using the RELATE command in the ANALYSIS module.

Use the ANALYSIS module to READ DEMOG.REC. Use BROWSE F4 to familiarise yourself with this file.

Before specifying the relation we will define a recoded age variable that will be useful in later work:

```
define ageg #
recode age to ageg 1-4=1 5-19=2 20-39=3 40-59=4 60-hi=5
```

Now attach the microfilariae results to the current dataset with the command:

```
relate idno micro15
```

Browse F4 through the file. Check that all the results are now available for every subject in the one record.

It is instructive to see how EpiInfo labels the different variables. Press the F3 **Variables** key and note that the variable names from the RELATEd file are preceded by the filename and a period. The variable name MICRO15.MFANY, for example, refers to the MFANY variable in the file MICRO15.REC. This is a common practice in relational database systems. Such a convention is particularly useful if variable names are shared between the related files.

The RELATE command does not link the two files permanently. The link is lost if you read another file or quit the ANALYSIS module. Enter the commands:

```
erase onchall2.rec
route onchall2.rec
write recfile
```

to save the data from the linked file as ONCHALL2.REC.

Multiple RELATEs

The RELATE command can be used to link several files together. Enter the commands:

```
read demog.rec
relate idno micro1
relate idno micro5
```

Browse F4 through this file and check that it contains the same information as ONCHALL.REC. Notice that the variable names are ambiguous. This is because MICRO1 and MICRO5 have variables with the same names. Press ESC to return to the main ANALYSIS screen. Examine the variable names by pressing F3 . ANALYSIS can differentiate between variables with similar names from different RELATEd files. The variables can be addressed by a double-barrelled name consisting of the filename and the variable name separated by a period or full-stop (.) character. Thus you can ask for:

```
tables micro1.mfany micro5.mfany5
```

The RELATE command does not link the two files permanently. The link is lost if you read another file or quit the ANALYSIS module. You may use the ROUTE *filename* and WRITE RECFILE commands to save linked files.

If you use WRITE RECFILE with RELATEd files that have the same variable names ANALYSIS will **not** retain the filename prefix and you'll end up with an ambiguously defined file structure.

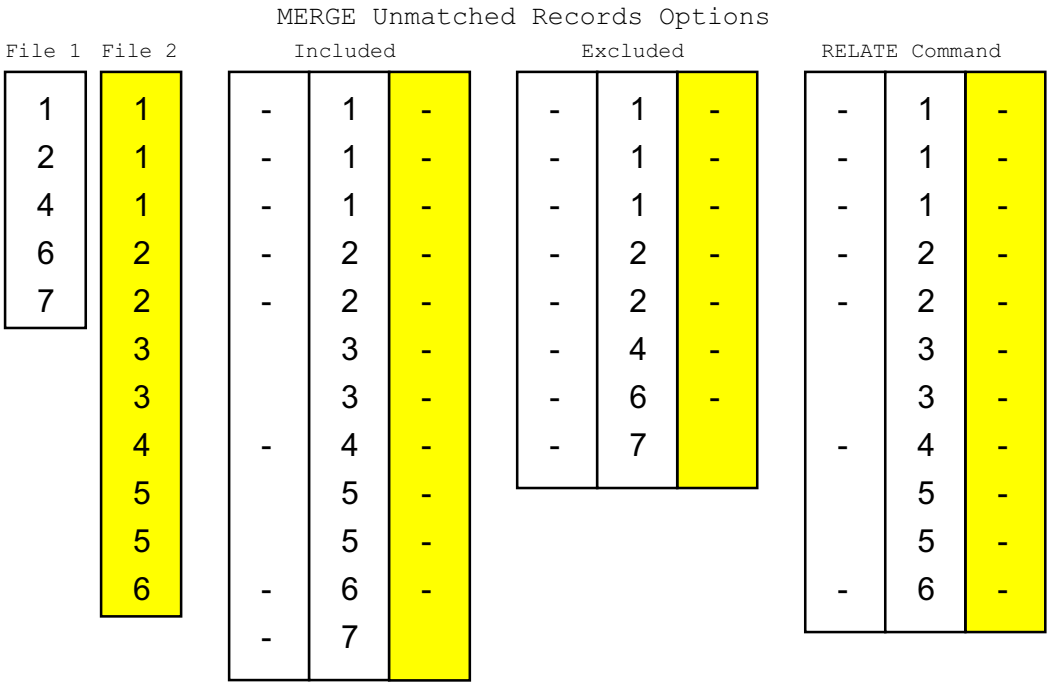
The base file accessed with the READ command defines the unit of analysis. Files accessed with RELATE can only add data to each existing case. The RELATEd file must have unique identifying variables. For example, if you wanted to link data about mothers to data about children then the children's file would be the READ file and the mother's file would be the RELATEd file.

It is safe to delete the temporary files (MFTEMP.REC, MICRO1.REC, MICRO5.REC). You cannot delete open files using the ERASE command in ANALYSIS or when using the DEL command in a DOS shell (F9). You can close all open files in ANALYSIS with the command:

```
close
```

MERGE and RELATE

The MERGE module and the RELATE command in ANALYSIS provide broadly similar functions. MERGE is more versatile as you can choose to include or exclude records that do not match the identifier variable in the first file:



The RELATE command in ANALYSIS excludes records that do not match with records in the first file. With MERGE you specify the 'parent' file as the first file and the 'children' file as the second file but with the RELATE command in ANALYSIS the 'children' file should be the first file (i.e. the file specified with the READ command) and the 'parent' file should be the file specified with the RELATE command. The ANALYSIS commands:

```
read parent.rec
relate idvariable children.rec
```

is equivalent to using MERGE with EXCLUDE. The ANALYSIS commands:

```
read children.rec
relate idvariable parent.rec
```

is equivalent to using MERGE with INCLUDE. A minor difference between the RELATE command in ANALYSIS and MERGE with INCLUDE is that RELATE will **not** include records for which a record does not exist in the first file. Relate only adds data to records **already** in the first file.

A major difference between MERGE and RELATE is that MERGE is permanent and RELATE is temporary. You need to use the ROUTE *filename* and WRITE RECFILE commands to make a permanently RELATEd file.

MERGE and RELATE

In some situations it will not be possible to work with READ and RELATE to retrieve multi-file datasets. In the following *clinical* dataset:

patients *may* have many visits and at each visit a patient *may* have many tests. In this case READ and RELATE commands in ANALYSIS **cannot** retrieve the complete dataset. The commands:

```
read tests.rec
relate id visits.rec
relate id patients.rec
```

result in the data for patients **without** recorded tests being **excluded**. Consider the result of the first RELATE command between TESTS.REC and VISITS.REC:

Tests			
ID	Date	Type	Result
1001	23/03/96	Routine	+
1001	23/03/96	TPHA	-
1002	24/03/96	Routine	-
1003	24/03/96	Routine	++

Visits		
ID	Visit	Visiting
1001	23/03/96	DTC
1002	24/03/96	AJR
1003	24/03/96	DTC
1004	24/03/96	DTC
1005	24/03/96	AJR
1006	24/02/96	DTC

This produces a dataset with data from both files:

From Tests				From Visits	
ID	Date	Type	Result	Visit	Visiting
1001	23/03/96	Routine	+	23/03/96	DTC
1001	23/03/96	TPHA	-	23/03/96	DTC
1002	24/03/96	Routine	-	24/03/96	AJR
1003	24/03/96	Routine	++	24/03/96	DTC

with the visit records for patients 1004, 1005, and 1006 excluded. RELATE will **not** add records for which a record does not already exist in the first file. RELATE will only **add data to records already in the first file** (TESTS.REC in this example). In these situations you **must** use MERGE with INCLUDE to retrieve the complete dataset.

It is very easy to accidentally include or exclude records with MERGE and RELATE. You should **always** check the results of any MERGE or RELATE operation before continuing with further data management or data analysis.

Another Use for MERGE

MERGE has another important function that is difficult to achieve using the RELATE command in ANALYSIS. This is the UPDATE function which allows you to change or add to data in one file based on the data in another file:

File 1

ID	V1	V2
11	1	1
12	1	1
13	1	
14	1	1

File 2

ID	V1	V2
11	1	2
12	2	2
13		2
15	2	2

Updated File

ID	V1	V2
11	1	2
12	2	2
13	1	2
14	1	1
15	2	2

The UPDATE function in MERGE is commonly used in surveillance systems in which it is important to be able to update records based on subsequent submissions of records with the same identifiers.

Follow-up and Recall

Many studies involve recalling participants for follow-up (e.g. for test of cure) or for recall (e.g. for a further round of treatment). You can use EpiInfo to help you manage follow-ups and recalls.

Use EPED to create the following questionnaire (.QES) file:

```
        {Id} Number : ####
        {Full Name} : <AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>
        {Address 1} : <AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>
        {Address 2} : <AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>
        {Address 3} : <AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>
        {Address 4} : <AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>
{Inter}vention Date : <dd/mm/yyyy>
   {Follow}-Up Date : <dd/mm/yyyy>
```

Save the file as FOLLOWUP.QES. Use ENTER to create a data (.REC) file called FOLLOWUP.REC. Enter some data into this file. Make sure you enter some INTERvention dates from **before** six months ago and that you leave some FOLLOW-up dates blank for these cases. Once you have entered some sample data you should quit ENTER and start ANALYSIS.

You can instruct ANALYSIS to list the cases with overdue follow-ups (here we assume a follow-up period of six months or 183 days) using the following commands:

```
read followup.rec
define due <A>
due = "-"
if (systemdate - inter > 183)  and follow = . then due = "+"
select due = "+"
list id fullname inter follow
```

Follow-up and Recall Letters

You can use ANALYSIS to automatically print recall letters for cases with overdue follow-ups. The following ANALYSIS commands (which are best stored as a program file - see page 97) will print the FULLNAME, address variables, and ID number above the text of a follow-up letter which is stored in a file called FULETTER.TXT (which you will need to create using EPED):

```
cmd fuletter
  type "@fullname"
  if address1 <> . then type "@address1"
  if address2 <> . then type "@address2"
  if address3 <> . then type "@address3"
  if address4 <> . then type "@address4"
  type " "
  type "Our Ref: @id"
  type " "
  type "Dear @fullname"
  type " "
  type fuletter.txt
end
read followup.rec
define due <A>
due = "-"
if (systemdate - inter > 183)  and follow = . then due = "+"
route printer
if due = "+" then fuletter
process
route screen
quit
```

The CMD command in ANALYSIS allows you to define a block of commands (typed between CMD and END) as a single command. In the above example the CMD command is used to define a new command called FULETTER which prints name, address, and identifier data followed by the text of the file FULETTER.TXT. The command:

```
if due = "+" then fuletter
```

Ensures that this occurs **only** for records with DUE = "+". The PROCESS command instructs ANALYSIS to work through the data file one record at a time and perform all calculations that have been specified in the following forms:

Form	Example
<expression>	due = "+"
if … then <expression>	if delay > 183 then due = "+"
if … then <command>	if due = "+" then fuletter

The '@' sign before the name, address, and identifier variables in the TYPE commands instructs ANALYSIS to use the contents of a variable in the record that is currently being PROCESSed rather than list the contents of that variable for every record.

Summary and Presentation

Introduction

Before conducting any formal statistical tests or data analysis process it is essential to prepare simple summaries and tabulations. These perform several functions:

- ❑ Screening for undetected data entry and coding errors.
- ❑ Describing the study population.
- ❑ Providing an overview of results.
- ❑ Helping you select the appropriate statistical analysis.

In this Section the onchocerciasis dataset is used to demonstrate methods of data summary using EpiInfo. These cover:

- ❑ Producing frequency tables and cross-tabulations.
- ❑ Producing graphs.
- ❑ Programming EpiInfo to produce repeat analyses.

Creating tables

EpiInfo can produce straightforward frequency tables and cross-tabulations of categorical variables. The basic commands to request these are FREQ and TABLES. For the onchocerciasis data it is important to describe the baseline demography and microfilarial loads. This will justify the choice of this population as a model for the use of Ivermectin. The usual measures of microfilariae load are:

- Prevalence of any microfilariae
- Geometric mean density of microfilariae per mg of skin across the population.

The geometric mean is quoted (rather than the arithmetic mean) because distributions are often highly skewed.

Start ANALYSIS and READ the file ONCHALL.REC. To obtain a breakdown of the age structure of the population issue the command:

```
freq ageg
```

Note that this includes relative frequencies and cumulative frequencies. FREQ also calculates summary statistics but these are **not** relevant here. One way to obtain a separate age distribution for males and females is to enter the commands:

```
select sex = 1
freq ageg
select
select sex = 2
freq ageg
select
```

A quicker and more effective way of obtaining an age-sex breakdown is with the command:

```
tables ageg sex
```

Note the arrangement of the variables: the first (AGEG) is vertical and the second (SEX) horizontal. Produce a table of the age-specific prevalence of microfilariae with percentages by entering the commands:

```
set lines = on
set percents = on
tables ageg mfany
```

Two percentages are produced for each cell of the table. The upper value is *row* percentage. The lower value is the *column* percentage. The command SET LINES = ON makes the table easier to read by printing separating lines between the cells of the table.

Summarising quantitative data

To obtain the mean and other descriptive statistics for the variable AGE issue the command:

```
means age /n
```

Use HELP F1 and select MEANS to find out what is the purpose of the **'/N'** at the end of the MEANS command.

If you were simply to enter the command:

```
means age
```

You would obtain a listing of all the data followed by the summary statistics. This table is identical to the one produced by the FREQ command.

We shall now produce the geometric mean densities of microfilariae. EpiInfo does not report the geometric mean automatically so we must calculate the arithmetic mean of the logarithm of density plus one (added to avoid attempting to take the logarithm of zero) and then report its antilogarithm plus one.

Create a new variable called LDENS to hold the log(density) per mg of microfilariae. We want a numeric variable with two decimal places, so enter the commands:

```
define ldens ##.##
ldens = ln(mfdens + 1)
```

One is added to the MFDENS to avoid taking the logarithm of zero. Use the MEANS command to find the overall means:

```
means ldens /n
```

Use a calculator or a set of log tables to work out the geometric mean density for the entire population. Use the MEANS command to find the median of the untransformed variable MFDENS:

```
means mfdens /n
```

And check that your calculated geometric mean is reasonably close to this.

To produce age-specific geometric mean densities enter the command:

```
means ldens ageg /n
```

Use a calculator or a set of log tables to work out the geometric mean density for each age group.

Saving output

Summary data can be saved to a file. This can later be printed out or incorporated into a report by importing it into a word-processor. Use the ROUTE command with a filename to send the output of ANALYSIS commands to a file:

```
erase agetab.out
route agetab.out
tables ageg mfany
route screen
```

The last ROUTE command closes the output file and instructs ANALYSIS to send output to the screen only.

The tabulated prevalence data have been written to a file called AGETAB.OUT on your disk. You can verify this by quickly 'shelling out' to DOS from ANALYSIS with F9 and issuing the DOS command:

```
type agetab.out | more
```

To see the output as a DOS text file. The output of the command TABLES AGEG MFANY has been saved in the file AGETAB.OUT. This is a plain text file and can be imported directly into any word-processor.

Issue the DOS command:

```
exit
```

To return to ANALYSIS. **Do not start EpiInfo again** or you will have two copies of EpiInfo running and neither will function correctly.

You can use the ROUTE command to send output to the printer with the command:

```
route printer
```

issued **before** the commands whose output you would like printed. The ROUTE command is always issued **before** the commands whose output you would like to save or print. The ROUTE command will **not** save or print the output of previously issued commands.

For a tidier output you may need to set the correct page length using the SET PAGE command which sets the number of lines per page and characters per line:

```
set page = 70,78
route printer
```

In this case we are setting the page size to 70 lines of 78 characters. These values are suitable for A4 paper on a dot matrix printer. When you have finished enter the command ROUTE SCREEN to turn off output to the printer.

Creating EpiInfo graphics

The quality and content of graphical displays of data are dependent on the intended audience. During data analysis, a quick and simple picture or display is very useful to help interpretation and to find outlying observations. Later, one would want to produce publication-quality graphics for inclusion in a final report or manuscript.

Here we shall be concerned only with the 'rough & ready' pictures that EpiInfo produces. There are five basic graphics commands in EpiInfo: BAR, HISTOGRAM, LINE, PIE, and SCATTER. We shall survey their actions briefly.

The BAR command or the PIE command can be used to display the frequency distribution of a categorical variable. Enter the command:

```
bar vill
```

Press any key to return to the ANALYSIS screen and enter the command:

```
pie vill
```

Titles can be added to a plot with the TITLE command:

```
title 1 onchocerciasis intervention study
```

The title can consist of up to three lines for plots (e.g. BAR, PIE, etc.) and up to five lines for commands that produce text output (e.g. FREQ, TABLES, MEANS, etc.). The number immediately after the word TITLE tells EpiInfo on which title line to place the rest of the command.

To see the effect of the TITLE command issue the command:

```
bar vill
```

Creating EpiInfo graphics

The HISTOGRAM command should be used to plot quantitative data. EpiInfo **will not** choose appropriate groups, as you can see with the command:

```
histogram age
```

You can choose your own cut-points, such as those we already have:

```
histogram ageg
```

But this ignores the differing widths of the age groups. Use the RECODE command to generate equal width intervals. Enter the commands:

```
define ageg2 <AAAAAAAAAAAAAA>
recode age to ageg2 by 20
histogram ageg2
```

To see how a BAR chart differs from a HISTOGRAM in EpiInfo issue the command:

```
bar ageg2
```

The distribution of a quantitative variable broken down by groups of another variable can be shown using the LINE command:

```
line ageg2 mfany
```

The SCATTER command can be used to examine the joint distribution of two quantitative variables. Enter the command:

```
scatter age mfdens
```

A regression line can be added by appending `'/R'` to the command:

```
scatter age mfdens /r
```

This makes little sense for a skewed variable such as MFDENS. With a log transformation, however, it is more appropriate:

```
scatter age ldens /r
```

The SCATTER command is also useful to check the consistency of numeric variables. Enter the command:

```
scatter age stay
```

Explain the very non-random pattern shown by the data points, and then comment on the outlying points that lie above the line AGE = STAY. Can you list the IDNOs of these points so that their data records can be checked?

Press F10 to quit ANALYSIS and return to the top-level menu.

Creating EpiInfo programs

ANALYSIS is a highly *interactive* program, in the sense that commands are executed as they are entered. This is very appealing but can lead to difficulties when you want to reproduce your work at a later date. Also, you might find yourself typing the same commands repeatedly, and it would be convenient to be able to automate the process. This is where program files can be used. A program file is simply a text file that contains ANALYSIS commands. The commands in a program file are the same as those used at the **EPI>** prompt in ANALYSIS.

Programs are particularly useful when you wish to carry out repeated analyses on routine data (e.g. surveillance data). Long series of tabulations or complex analysis can be produced by simply typing the name of the program file. Program files help avoid processing errors. As you enter the commands with a text editor it is possible to check the file for mistakes before executing the commands.

The most convenient way to create a program file is to use EPED. You should use the filename extension .PGM for your program files, so that you, EpiInfo and others will recognise them in the future.

Start EPED and set **Txt** mode. Type the sample program:

```
erase onchall.out
route onchall.out
read onchall.rec
set statistics = off
relate idno tmtcodes
tables drug mfany
tables drug mfany5
define ldens ##.#
ldens = ln(mfdens + 1)
means ldens drug /n
define ldens5 ##.#
ldens5 = ln(mfdens5 + 1)
means ldens5 drug /n
route screen
quit
```

These commands are **not** being executed immediately so you can use the ↑ , ↓ , → , ← and DEL keys as normal. Check your work carefully. Select **Save** by pressing F9 . When prompted give the filename ONCHALL.PGM and press the ENTER key. Press F10 to return to the top-level menu.

Start the ANALYSIS module and enter the command:

```
run onchall.pgm
```

Program (.PGM) files are ideal for storing code for batch checking of data and for routine reports such as recall lists in clinical or community trials.

Creating EpiInfo programs

Watch as ANALYSIS processes all your commands, and writes the output both to the screen and into the file ONCHALL.OUT. You will be returned to the top-level menu when the program finishes.

Start EPED. Select the **File** menu by pressing F2 and select the **Open file this window** option. Type the filename ONCHALL.OUT and press ENTER . The output from the program appears on the screen. Check the output of the program and return to the top-level menu by pressing F10 .

You might find some errors in your program while it is running or in the output file. Carefully examine the results from ANALYSIS to find the offending commands. Edit ONCHALL.PGM and run the file again.

You can also edit program files from within ANALYSIS using the EDIT command. The command:

```
edit onchall.pgm
```

issued from within ANALYSIS will allow you to edit the ONCHALL.PGM program file. The following keys work within the ANALYSIS program editor:

Key	Function
F2	Save file
F3	Open file
F7	Mark **start** of block
F8	Mark **end** of block
CTRL K V	Move block
CTRL K C	Copy block
CTRL K Y	Delete block
CTRL Y	Delete line
CTRL Q F	Find ...
CTRL Q A	Find & replace ...
F10	Quit

Exporting EpiInfo data

EpiInfo has at least two advantages over conventional database management systems for data management and analysis:

- ❑ Rapid and comprehensive data entry and validation procedures.
- ❑ Automatic calculation of useful epidemiological statistics, such as the odds ratio, risk ratio, Mantel-Haenzel estimates, etc.

Like every computer program EpiInfo has limitations. There are many statistical calculations which EpiInfo cannot do. The data analyst may find themselves wanting to move data out of EpiInfo and into another program. EpiInfo provides a module, called EXPORT, to perform these data exports.

Start the EXPORT module. The EXPORT menu shows the range of choices available to you for exporting your data:

LOTUS: Use this for spreadsheet packages (e.g. Lotus, Quattro, Excel). This is also a convenient way to get an entire .REC file into charting programs.

SYSTAT, Statpac, SAS, SPSS/PC+, MULTLR, EpiStat, EGRET: To move your data into another statistics program on the PC. SAS files can also be used with the mainframe version of SAS.

SPSS-X: Select this option if you want to move your data onto a mainframe to use the SPSS-X program. This is different from the SPSS/PC+ conversion.

dBase 2/3/4: These options will create a dBase file. Most statistical packages, databases, word-processors (e.g. for mail-merge of recall letters), and spreadsheets should be able to read a dBase format file.

BASIC, FIXED: Use these (plain text) formats when all else fails.

Fill in the prompts on the EXPORT screen as follows:

Input file name:	**ONCHALL**
Output format:	**11 (SPSS/PC+)**
Output file name:	**ONCHALL**

EXPORT supplies the filename extensions (in this case .SPS). Press the ENTER key to begin conversion. Progress is reported on the screen. Press ENTER to clear the information box. Press F10 to return to the top-level menu.

The SPSS/PC+ file created by EXPORT is a text file that needs to be processed as an *include* file by SPSS/PC+ before the data can be analysed. You may need to edit the .SPS file before using it with SPSS for Windows™ (or use the SPSS-X export option).

Cleaning up after yourself

By going through the exercises presented in this book you will have created many temporary and working files. You should delete these files (or copy them to another location) if you want to run through the worked examples again. To delete these files you should type:

```
cleanup
```

at the DOS prompt. This runs the DOS batch file CLEANUP.BAT which was provided with the example data files. To go through the exercises in this book again you will need the following files:

```
blood.rec
cleanup.bat
demog_1.rec
demog_2.rec
demog_3.rec
micro.rec
onchall.rec
tmtcodes.rec
```

These files are on the accompanying disk.

The Data Processing Needs of a Study

Planning the data processing needs of a study

The other sections of this book consider the progress of a single questionnaire through the stages of a survey, or the manipulation of data once it has been entered into the computer. In this section we shall consider the data processing needs of the study as a whole.

It is important that the data processing needs are carefully planned when the study is being designed, and that the costs for adequate hardware, software and personnel are included in the budget. Otherwise there is the real danger of a backlog of forms building up, delaying analysis of the data. In addition, in studies where repeat visits to the participants are planned, the computer may be used to prepare, for example, lists of persons to be visited. In this case, a hold-up in data processing could lead to delays in the fieldwork. As well as delays to the analysis or field work, inadequate resources for the data processing aspects of a study can lead to breakdowns in quality control at different points in the survey process resulting in poor quality data.

As an example, throughout this section, we shall take an overall view of the onchocerciasis study, taking into account its primary objective, as a trial of the effectiveness of the drug Ivermectin (Merck) in reducing the symptoms of the disease.

References

Sections 13.2 and 13.3 of *Methods for field trials of interventions against tropical diseases: a toolbox* by PG Smith and RH Morrow (Oxford University Press, 1991)

Choosing and using a microcomputer for tropical epidemiology. I. Preliminary considerations by P Byass (Journal of Tropical Medicine and Hygiene 1989, 92, 282-7).

Processing Data: The Survey Example by LB Bourque and VA Clark (Sage, 1992).

Data sources

In planning the data processing needs, the first stage is to summarise the data sources in the study. It is important to include all sources of data (including secondary data, interview schedules and questionnaires, laboratory forms, registers of events, etc.), whether computerised or not. This will involve an outline of the study plan.

The Ivermectin trial took place in six villages, and included everybody in those villages aged over one year of age. A census was carried out which showed the total population of the six villages to be about 1750 persons. This was followed by a baseline survey to record basic demographic data such as name, age, sex, occupation, residence.

Coincident with this survey, and then again every six months for a total of five surveys, a clinical survey was conducted to record symptoms of onchocerciasis such as skin lesions, nodules or itching, and an ophthalmic survey to record eye data and visual acuity. These surveys were conducted in one village every few weeks, thus covering the six villages in about five months. Samples of blood and skin snips taken during the clinical survey were analysed and the results stored in laboratory records.

At each survey round, information about deaths occurring since the previous survey was recorded. Baseline data was recorded about new entrants to the study (children reaching their first birthday or in-migrants).

Each individual was randomised to one of eighteen codes, nine of which were allocated to receive Ivermectin, and the other nine to receive placebo. Note that in the example exercise in this book only two codes were used for simplicity but more codes are desirable to protect blinding should the code need to be broken for a single individual (e.g. in case of serious illness or suspected severe allergic reaction to the drug). The allocation of codes was done in the UK, so that both the researchers and the villagers were blind to the treatment given. After each survey round, a treatment file was created with the weight of each subject and the appropriate dose (according to weight recorded at the clinical survey). This file was used to prepare treatment cards, to each of which was stapled a blister pack containing the appropriate number of tablets. The subjects were treated, and this fact recorded on the treatment file.

From the study outline, a list of the primary data sources can be drawn up, including the number, frequency and size of each:

Forms needed	Quantity	Frequency	Characters per record
census	1750	once	50
baseline survey	1750	once	54
death records	unknown	once	
new entrant records	unknown	once	
clinical survey	1750	five times (every six months)	45
ophthalmic survey	1750	five times (every six months)	31
laboratory records	1750	five times (every six months)	48
treatment file	1750	five times (every six months)	32

Tasks arising from the data

Further tasks will follow from this list of primary data sources, and will in turn generate secondary data:

- ❑ Combination of the census, the baseline survey data and the randomisation codes into a population register.
- ❑ Maintenance of this register and updating with details of new births, deaths and migrants.
- ❑ Using the register to generate lists of subjects for clinician and ophthalmologist to visit at each survey round.
- ❑ Production of sticky labels showing name, age, sex, address, ID no. of each subject to be affixed to blank clinical and ophthalmic questionnaires, and to laboratory forms.
- ❑ Using the register and most recent survey data to produce (using the treatment file) a list of subjects to be visited for treatment by each field worker. This list was printed double-spaced to allow the field worker to record treatment given and other information.
- ❑ Production of a sticky label with treatment code and number of pills for each subject to be treated.
- ❑ Production of lists of those subjects not seen at the clinical or ophthalmic survey or not treated at treatment round, for subsequent follow-up.

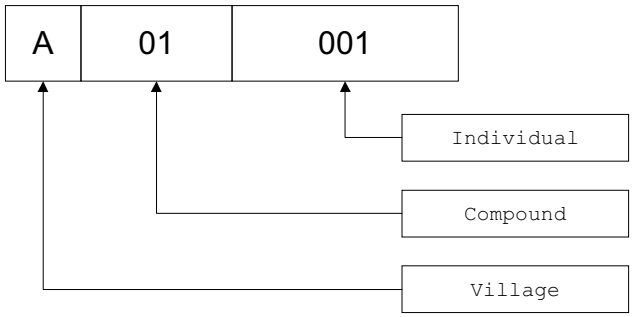
The number and nature of each of these tasks needs to be estimated beforehand, in order to quantify the resources needed to carry them out.

Linking of data files

Files need to be linked with a unique identifying number. In the Ivermectin trial a simple four digit number (IDNO) was appended initially to the baseline survey data. In practice it is often helpful to use an identifier which combines information about the person's residence. For example, in a study in The Gambia which covered about twenty villages, the identifying number was of the form:

A01001

The initial letter indicated the village, the first two digits indicated the compound within that village, and the final three digits indicated the individual within that compound:



This kind of hierarchical numbering scheme is particularly useful when information is recorded at different levels of the hierarchy. For example, prevalence of illness among individuals is related to (e.g.) sanitation within the compound.

Personnel requirements

When planning the number and level of staff required, the following general data processing tasks need to be considered in addition to those mentioned above:

- Distribution and tracking of forms to and from interviewers in the field.
- Raising queries with field staff about data, re-entering corrected or additional data.
- Setting up and maintaining systems for checking that each form goes through all the necessary stages of data entry. It is useful to mark the forms in some way so that it is obvious which ones have gone through which processes. It can be useful to have clearly marked trays for forms ready for (or having completed) a certain stage in the process.
- Checking data entry, either by double data entry or an alternative method. Often checking programs are written which need to be run periodically.
- Backing up data files. Need to decide who will do this, how often and onto what (one copy of all program and data files should be stored off-site in case of theft or fire).
- Virus checking. Who will keep virus checking programs up-to-date and how often will they be run to check computers?
- Protecting subject confidentiality.
- Documenting programs

Administrative duties, leave and sickness also need to be taken into account.

Data-entry clerks and coders

Unless trained data entry clerks and coders are already available, they will need to be hired and trained. The level of person hired depends on the extent to which they will be required to use their judgement. Byass (1989) found that secondary school leavers could be trained for straightforward data coding and entry. Bourque and Clark (1992) advise that, as the work of data processing personnel tends to be repetitive and tedious and demands attention to detail, it is important to explain during training the importance and relevance of their work to the whole research process. If open questions and qualitative data are to be coded then it is generally necessary to employ data processing personnel with a higher level of education than completion of secondary school.

The number of different tasks (such as those listed above) that a data entry clerk will perform varies from study to study. Obviously a main component of a data entry clerk's duties is to enter data, and in order to estimate the amount of time needed for this it is necessary to consider the amount of data. For the Ivermectin study the major load will be the processing over each six month period of the 1750 clinical, ophthalmic, laboratory and treatment forms. This will average out as 1750 x 4 / 26 = about 270 forms per week, or 540 since they will be double entered.

The requirement for data entry clerks can be estimated from this. If the average size of forms is around 40 characters this will entail entry of 21,600 characters per week. Byass (1989) found in The Gambia that secondary school leavers could be trained to type quite quickly and handle over 100 records of 100 characters per day (a total of 50,000 characters per week). This indicates that one clerk should be sufficient for this study (although it is usually preferable to have different people typing in the two entries of double entered data). In countries and urban areas where trained keyboard operators are available, far higher data entry speeds can be expected.

The data manager

For large studies it is advisable to appoint a data manager. This person will be responsible for

- ❑ Supervising the data entry clerks.
- ❑ Organising the ‘flow’ of the questionnaires.
- ❑ Ensuring the quality of the data.
- ❑ Ensuring the security of the data (backups, confidentiality).
- ❑ Carrying out the tasks arising from the data such as production of lists and labels.
- ❑ Installing and maintaining the hardware.
- ❑ Installing and maintaining software.
- ❑ Producing summaries of the data to enable the investigator to monitor progress.

Software requirements

As you have seen, EpiInfo can handle the requirements of large studies. The ANALYSIS module of EpiInfo can perform most of the statistics that will be required for analysing data from a survey or trial but it is likely that another statistics package will be required to perform multivariate or repeated measures analysis. The choice of statistical package is best left to the statistician who will be responsible for the data analysis. In the case of clinical trials the choice of statistics package may be dictated by the licensing authority.

For complex datasets you may opt for a specialised suite of programs to be written (for example in a database package such as Access) incorporating functions such as double data entry, range and consistency checks, and the linking and merging files. If you choose this route you should insist that everything is fully documented. The maintenance of undocumented programs can be very difficult and may require a complete rewrite of all programs if the original programmer is no longer available for advice. If a decision is made to use a package other than EpiInfo for the bulk of the work, then you would also need a word-processing package and a statistical package. The cost of these will need to be considered, and time and resources allocated for training.

Hardware requirements

Hardware specifications change from month to month making it difficult to give anything more than general guidelines:

- Always go for the 'industry standard' in both hardware and operating system. At the time of writing this meant an IBM compatible personal computer (PC) based upon the Intel Pentium processor running Microsoft Windows '95. Other combinations of hardware (e.g. Apple Macintosh or PowerPC) or operating system (OS/2) may be appealing but may prove difficult to support. Always buy from a reputable manufacturer. Do not be tempted by bargain prices.

- Buy the highest specified equipment you can afford. Processor speed is important. The amount of memory (RAM) can also greatly effect system performance. Make sure that the hard disk is large enough to hold at least twice the anticipated size of data and program files. A large monitor (17 or 21 inch) may prove useful for data analysis machines. Machines that are to be used for data-entry purposes only can be less highly specified. EpiInfo can run well on modestly specified machines.

- All machines should have some form of backup mechanism. For small surveys you can rely on floppy disks but for larger surveys you will need a tape backup device. You should also purchase a small fireproof box or safe for storage of backup disks or tapes.

You should have at least two computers for any survey. This will allow the survey to continue if one machine should fail but it also allows data to be entered on one machine whilst the other machine is used for administrative or data-management tasks. In remote locations it will help if the computers are identical so that a single set of spare parts will suffice for both machines and one machine can always be kept running by cannibalising the other. Portable computers are attractive but have limited keyboards and screens and are difficult to repair. They are more expensive than desktop machines and are more susceptible to theft.

The printer you use will depend on the size and complexity of the survey. For large surveys you may require a dedicated printer to produce lists, reports, and the output of analysis. Such a printer will be optimised for speed of output (not quality) and be able to handle wide continuous stationary (dot-matrix printer). If you will need to produce follow-up letters then a printer capable of handling standard letter paper and producing high quality text will be required (ink-jet or laser printer). Cheap colour ink-jet printers are available which will produce high quality text and graphics but tend to be expensive to run. In remote locations make sure that you buy a good supply of paper and other consumable items (such as ink, ribbons, or cartridges) at the start of the survey. Always buy from a reputable manufacturer. Do not be tempted by bargain prices.

In remote or developing country locations you may also need to purchase an uninterruptible power supply (UPS) as this will protect your computers against damaging 'spikes' in the mains supply, maintain the supply for a short period (enough time to save any open files) to protect against cut in mains supply (black-outs), and provide good quality power during periods of low voltage caused by excess demand (brown-outs). Air conditioning systems are not required as most computers are equipped with internal fans but dust and humidity can cause problems.

Appendix 1
Files and Variables

File	Variable	Contents	Values / Codes	Missing
DEMOG_x.REC *baseline demography*	IDNO	Subject ID number		
	VILL	Village number	1 ... 6	
	DATE	Date of interview	dd/mm/yy	
	AGE	Age in years	Positive integers	99
	SEX	Sex	1 = Male 2 = Female	9
	TRIBE	Tribe code	1 = Mende 2 = Other	
	HHNO	Household number	Positive integers	99
	REL	Relation in household	1 = Self 2 = Parent 3 = Child 4 = Sibling 5 = Other blood 6 = Spouse 7 = Other non-blood 8 = Friend 9 = Other	0
	STAY	Years in village	Positive integers 99 = Missing	99
	OCC	Occupation code	1 = At home 2 = At school 3 = Farming 4 = Fishing 5 = Office work 6 = Trading 7 = Housework 8 = Mining 9 = Other	0
MICRO.REC *mircofilariae counts*	IDNO	Subject ID number		
	SR	Survey round	1 ... 6	
	MFRIC	MF count (right)	Positive integers	999
	MFLIC	MF count (left)	Positive integers	999
	SSRIC	Skin diameter (right)	Positive real numbers	0
	SSLIC	Skin diameter (left)	Positive real numbers	0
BLOOD.REC *blood samples*	IDNO	Subject ID number		
	SR	Survey round	1 ... 6	
	EOSIN	Eosinophil count	Positive integers	9999
	PCV	Packed cell volume	Positive integers	99
	MPS	Malaria parasites	0 = Negative 1 = Positive	9
TMTCODES.REC *treatment codes*	DRUG	Drug batch	IVER = DRUG PLAC = PLACEBO	
	IDNO	Subject ID number		

Index

Index

Index