Data Management For Surveys & Trials

A practical Primer Using EpiData

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

Data Management for Surveys and Trials

A Practical Primer using EpiData

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

The EpiData Documentation Project

First Published in 2001 by *The EpiData Association*

This edition published 2001

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

Data management 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 in this course are:

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

The objective of this book is that you should understand and know how to carry out the *data management* aspects of a research study. This book uses the ***EpiData*** software package. ***EpiData*** 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 ***EpiData*** for this book for five reasons:

1. It has been specifically written for use in research studies with functions that are specifically designed to assist with each stage of 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.
4. It does not require a powerful computer to run it.
5. It can export data in formats that can be read by virtually every statistical, database, or spreadsheet package.

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

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 ***EpiData*** version 2.00 or later. This book makes extensive use of sample datasets which are supplied with this document.

We do not consider the details of statistical analysis in this book. Instead, we concentrate on the steps that must be taken before statistical analysis in order to produce clean (i.e. error free) data in a format amenable to statistical analysis.

Databases

Objectives of this section

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

- ❑ Understand the idea of a database, and the concepts of file, case and variable.
- ❑ Understand the types of variables used in ***EpiData*** and their attributes.
- ❑ Create a questionnaire and data entry forms using the editor functions of ***EpiData***.
- ❑ Create a database file using ***EpiData***.
- ❑ Enter data into a database file using ***EpiData***.

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 large 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.

Onchocerciasis 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 blackfly (*simulium*) that breed in fast-flowing tropical 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 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* (Merck). 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.

The questionnaire on page 32 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.

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.

Detailed information about the data can be found in Appendix 1.

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 verify data with the 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 are 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. Data may be exported for analysis by statistical, database, or spreadsheet package.

The sequence of events (1 - 10) will be considered in this book.

Overview of EpiData

EpiData was specially created to assist researchers to carry out epidemiological investigations. It has functions which carry the following tasks:

TEXT EDITOR functions that are used to create questionnaires, edit files that contain data-checking and data-coding rules, and write and edit text (such as the output from EpiData's data documentation functions).

DATA ENTRY functions that allow you to make data files from questionnaire files, enter, edit, and query data.

DATA CHECKING functions that allow you to add range checking, skip patterns, legal values, and complex coding rules to data as it is entered (*interactive* checking).

DATA CHECKING functions that allow you to apply range checking, skip patterns, legal values, and complex coding rules to data **after** it has been entered (*batch* checking).

DATA CHECKING functions that compare two files and report any differences between them (*validation*).

CODING and CALCULATION functions that allow you to create new variables or manipulate existing variables (e.g. calculation of latency periods, grouping ages etc.).

An **IMPORT** function that allows you to import data from text, dBase, and STATA files

An **EXPORT** function that allows you to export data to text, dBase, STATA, Excel, SAS, and SPSS format files for subsequent analysis.

DATA MANAGEMENT functions to append (i.e. join end-to-end), merge (i.e. join side-to-side), or restructure data files.

DOCUMENTATION functions that list data, describe file structures and data checking rules applied to files, produce frequency tables and descriptive statistics for variables in a file, and count records in one or more files using values entered into particular variables.

UTILITY functions that allow you to pack (i.e. remove deleted records) data files, compress data files, make backup copies of data files and associated questionnaires and data checking rules, copy file structures, create questionnaires from existing data files, and rename fields in a file.

EpiData does not contain any complex data analysis functions but data may be exported to a variety of common formats. ***EpiData*** use the same file format as EpiInfo version 6.xx (a popular MSDOS-based program widely used by field epidemiologists). Any program that can read EpiInfo version 6.xx data (.REC) files may also be used to analyse data entered using ***EpiData***. Available tools include the **`ANALYSIS`**, **`CSAMPLE`**, and **`EPINUT`** modules of EpiInfo version 6.xx, the **`ANALYSIS`** module of EpiInfo 2000, EpiMap version 2, and numerous add-in programs that perform logistic regression, conditional logistic regression, and survival analysis. Basic data analysis functions may be added to subsequent versions of ***EpiData***.

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 and functions. 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.

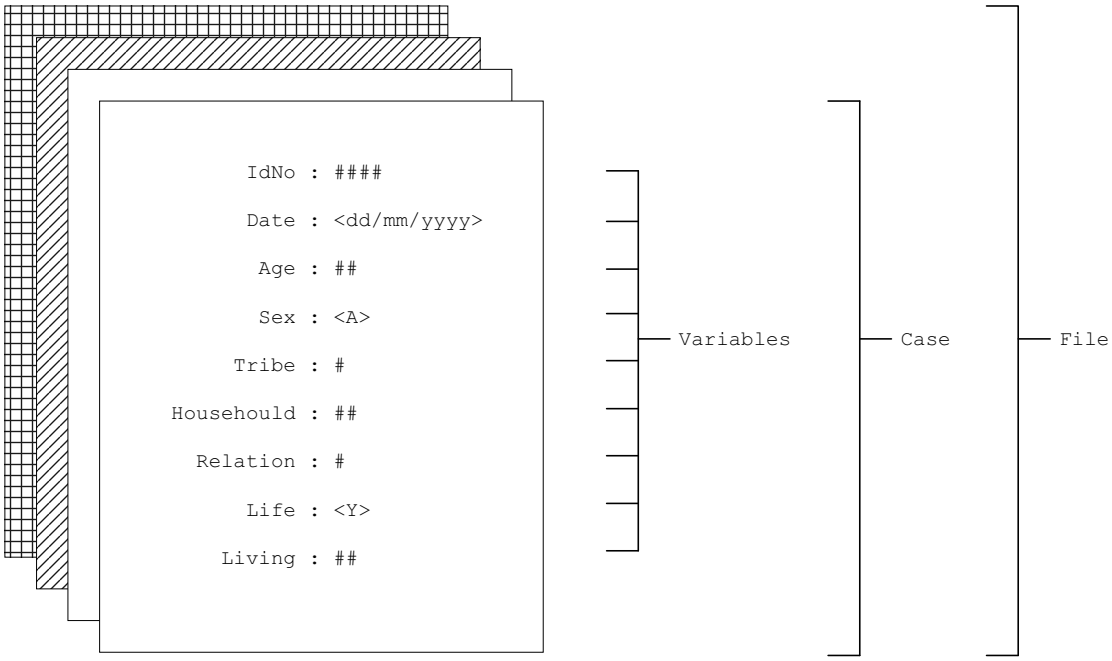
There are many different DBMS software packages available for use on personal computers, of which dBase (and derivatives such as FoxPro and Visual dBase), Paradox, Access, and Approach are probably the most widely used on personal computers. We show here how to set up a database using the ***EpiData***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

Once the layout of questionnaires or data collection forms have 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* or *key variable* common to each data file (e.g. individual identifying numbers) 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	Age	Sex	Tribe	Household Number	Relation To Head Of Household
1	10/11/1988	56	M	1	1	1
2	10/11/1988	49	F	1	1	2
3	10/11/1988	15	M	1	1	3
4	10/11/1988	28	M	2	1	6

A file with a structure such as this and 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 ***EpiData*** variable definitions is:

Name	Type	Length	*EpiData* 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 ***EpiData*** variable names:

- ❑ Must not exceed eight characters
- ❑ Must begin with a letter, not a number
- ❑ Can otherwise contain any sequence of letters and digits
- ❑ Must 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. **`Q17`**) as variable names.

Examples of **illegal** variable names are:

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

Write down three legal and three illegal variable names:

Legal Variable Names	Illegal Variable Names

EpiInfo version 6 (on which ***EpiData*** is based) allows you to create variable names that are ten characters long. ***EpiData*** can work with EpiInfo files that use ten character variable names but can only create files with eight character variable names. The eight character variable name limit was chosen to make it easy to export data to packages such as SPSS which also have an eight character variable name limit.

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. ***EpiData*** 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.
>
> ***EpiData*** 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 codes that use letters (e.g. ICD-10 which uses mixed letter and number codes). You can enter numbers into text and upper case text variables but you cannot easily 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).
>
> **BOOLEAN, 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' (which may also be entered as '1' and '0'). **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`**) and European (**`dd/mm/yyyy`**) 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 EpiData 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. ***EpiData*** also has a special type of **DATE** variable that is updated each time a record is changed.
>
> **AUTO ID NUMBER** variables that get incremented by one for every new record that is entered. **AUTO ID NUMBER** variables cannot be changed during date entry since their values are generated automatically.
>
> **SOUNDEX** variables are special text variables that apply SOUNDEX coding rules to text data as it is entered. SOUNDEX is a coding of words that can be used to anonymise (e.g.) the surnames of informants participating in a survey. This may be useful in (e.g.) surveillance systems for sexually transmitted disease. A SOUNDEX code is always in the format A-999, i.e. one upper-case letter, a hyphen, and three digits.

EpiData does not support the **PHONENUM** type variables that are available in EpiInfo version 6.xx but telephone numbers may be entered into ordinary text fields.

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 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.

Boolean, 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 statistical packages and with some statistical procedures (e.g. *marker variables* in regression analysis). You should check if your statistical software knows how to handle Boolean variable before using them in your data files. Boolean variables are automatically translated into numeric variables (coded 0 = No, 1 = Yes) when you export data from ***EpiData*** to STATA and Excel files.

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* within ***EpiData*** prior to the data being exported for analysis. 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 ***EpiData*** 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. When using dates in calculations and when comparing dates, ***EpiData*** works with dates as date-time serial numbers (i.e. the number of days since 31st December 1899) and can easily convert between dates and date-time serial numbers if required (see the 'Date and time functions' section of the help file for more details).

EpiData does not support time fields, but two functions (**`Time2Num`** and **`Num2Time`**) are provided that allow you to work with times in numeric variables (see the 'Date and time functions' section of the help file for more details). These functions permit the entry of times as numeric variables using the format *hh.mm* (0.00 - 23.59).

Defining variable type and length

EpiData 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 European format date. A variable defined as **<mm/dd/yyyy>** will hold an American format data. The characters used to define dates that are updated each time a record is changes are **<Today-dmy>** (European format) and **<Today-mdy>** (American format).
>
> **BOOLEAN**, **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**.
>
> **SOUNDEX** variables are defined in the same way as **UPPER CASE TEXT** variables except that the letter **S** is used to specify a **SOUNDEX** variable (e.g. **<S >**).
>
> **AUTO ID NUMBER** variables are defines in the same way as **UPPER CASE TEXT** variables except that the text **IDNUM** is used (e.g. **<IDNUM>**) to specify an **AUTO ID NUMBER** variable.

You should **not** use any of these special characters (i.e. **'_'**, **'<'**, **'>'**, and **'#'**) in ***EpiData*** questionnaire (.QES) files for any purpose other than defining variable type and length. ***EpiData*** also places a special meaning on the **'@'** character (it is used to align variables on data-entry forms) and this character should also be avoided.

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 easily perform mathematical operations with them. 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 ***EpiData*** 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 ***EpiData*** variable definition for the following variables:

Variable	Name	Type	Length	EpiData 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	EpiData definition
None	SYPHILIS (TONSILS)	Text	18	**<AAAAAAAAAAAAAAAAAA>**
KC60	A1	Text	2	**<AA>**
ICD-9	091.2	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. ICD-9 codes have three digits before the decimal point and one digit following the decimal point.

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 may 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*. Not-appropriate data is data that is missing because it is not appropriate to collect that data for a particular subject. For example, in a case-control study you will have subjects who are cases and subjects who are controls. Data such as onset times, onset dates, symptom histories, and the duration of symptoms would not be collected from controls. This data is missing not because it is unavailable but because it is inappropriate to collect it. 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.

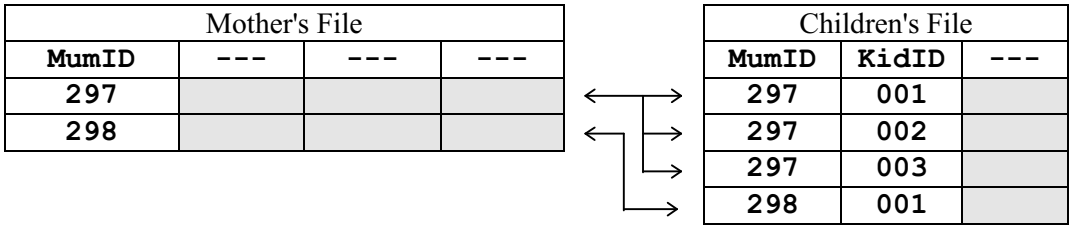
EpiData handles missing data automatically. Any field that is left empty at data-entry or uses missing data in calculations receives a special *missing data* code (. = missing).

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.

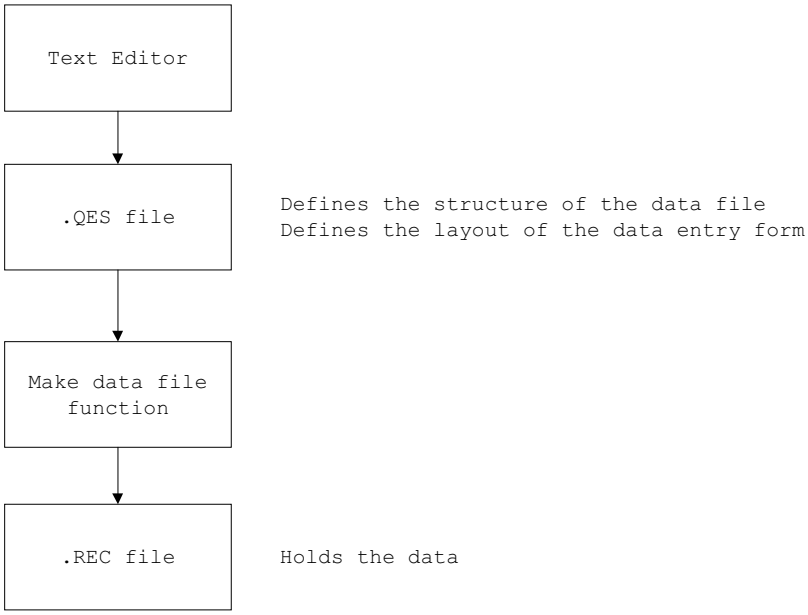
A database that consists of more than one linked data file is called a *relational* database. A database that consists of one data file (or many files of identical structure) is called a *flat-file* database.

Creating a data file

Creating a database file in ***EpiData*** is a two stage process:

1. First you use the text editor to create a screen questionnaire, or data entry form. This must include the variable names, types, and lengths. A questionnaire file **must** have the extension .QES. ***EpiData*** allows you to preview the data-entry form before saving the questionnaire (.QES) file and making a data (.REC) file.

2. The structure of this .QES file in turn defines the structure of the data file. In ***EpiData*** 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 form. Data are entered and stored in the record (.REC) file.

The **`Make data file`** function is used to create a record (.REC) file from a questionnaire (.QES) file.

The **`Enter data`** function is used to enter data into an existing record (.REC) file.

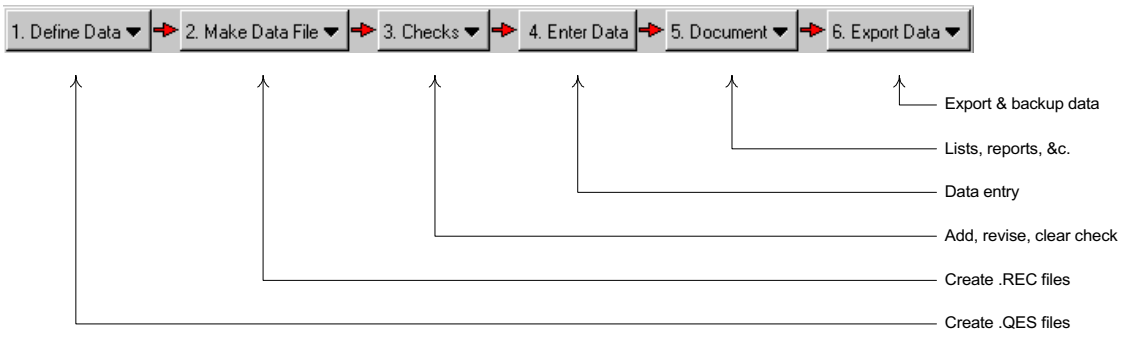
Starting *EpiData*

On different computer systems there will be different ways of starting the ***EpiData*** package. On most, however, it will be sufficient to select the ***EpiData*** item from the Windows Start Menu.

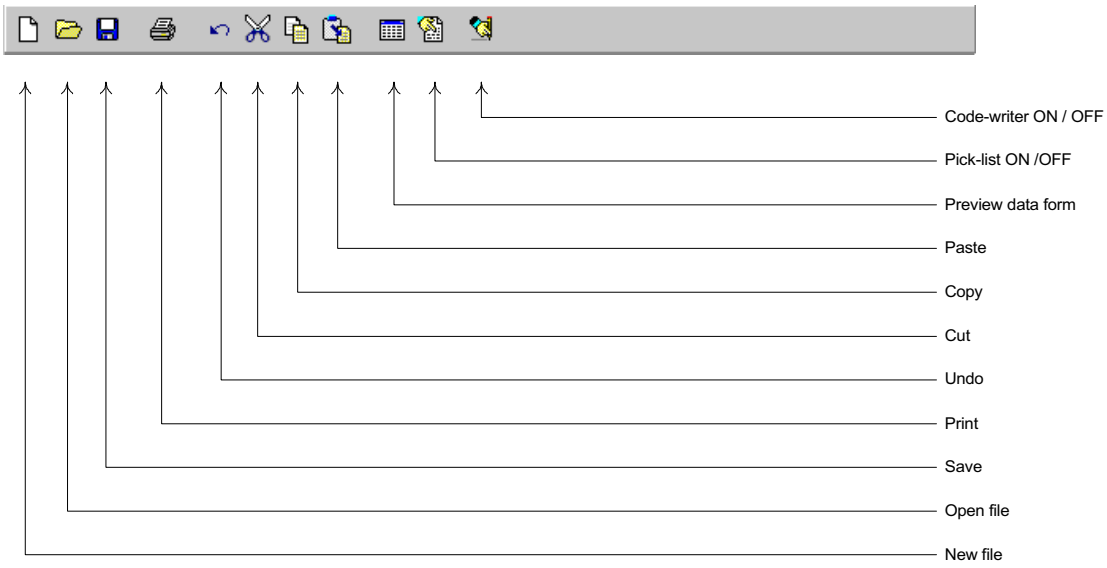
EpiData runs in a single window. Functions are selected from menus or from toolbars as in any other Windows application.

The two toolbars are called the 'work process toolbar' and the 'editor toolbar':

Work process toolbar



Editor toolbar



The toolbars may be switched on and off using the **`Windows -> Toolbars`** menu option. This book assumes ***EpiData*** toolbars are configured with the **`Work Process Toolbar`**, **`Editor Toolbar`**, and **`Hide Toolbars During Data Entry`** options set. These are the default options.

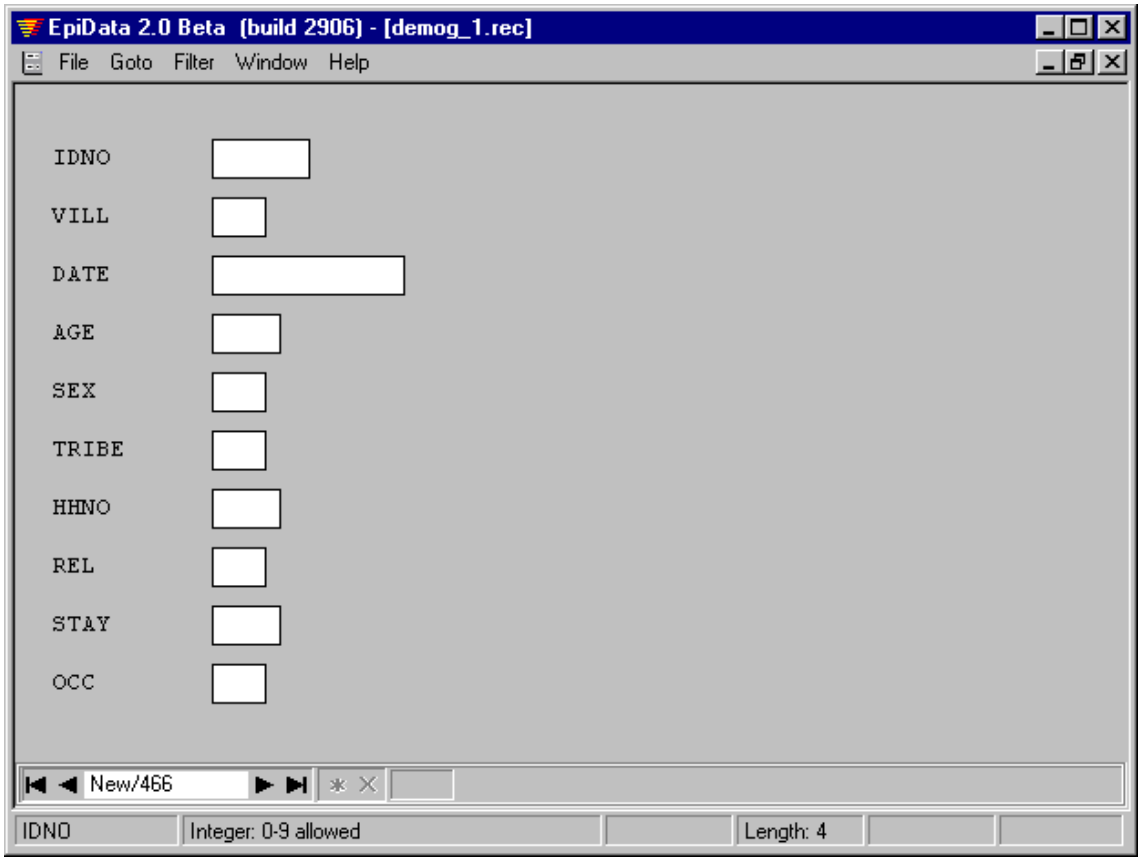
All of the functions available from the toolbars are also available as menu options. The toolbars provide shortcuts to the most common ***EpiData*** functions. Help is available from the **`Help`** menu. A brief guided tour of ***EpiData*** is provided and is available from the **`Help -> Tour of EpiData`** menu option.

Using *EpiData* to examine a dataset

In this exercise you will use ***EpiData*** to examine a *dataset*. ***EpiData*** provides four separate functions to examine a data file. You can browse data, list data, produce summary statistics, and display the structure of a data file. We will use each of these functions in turn.

Click the **`Enter data`** button on the work process toolbar. Select the file **`demog_1.rec`** and click the **`Open`** button.

EpiData will open the file and display a blank record ready to receive new data:



Do **not** enter any data now.

Using *EpiData* to examine a dataset

Click on the data navigation controls displayed in the bottom left corner of the ***EpiData*** window:

Control	Function
⏮	Display **first** record in the data file
◀	Display **previous** record in the data file
▶	Display **next** record in the data file
⏭	Display **last** record in the data file
✱	Display a **new** blank record ready to receive data
✕	**Mark** the currently displayed record **for deletion**

and browse through the records in the **`demog_1.rec`** data file.

Note that the number of the current record and the number of records in the data file are also displayed in the bottom left corner of the ***EpiData*** window.

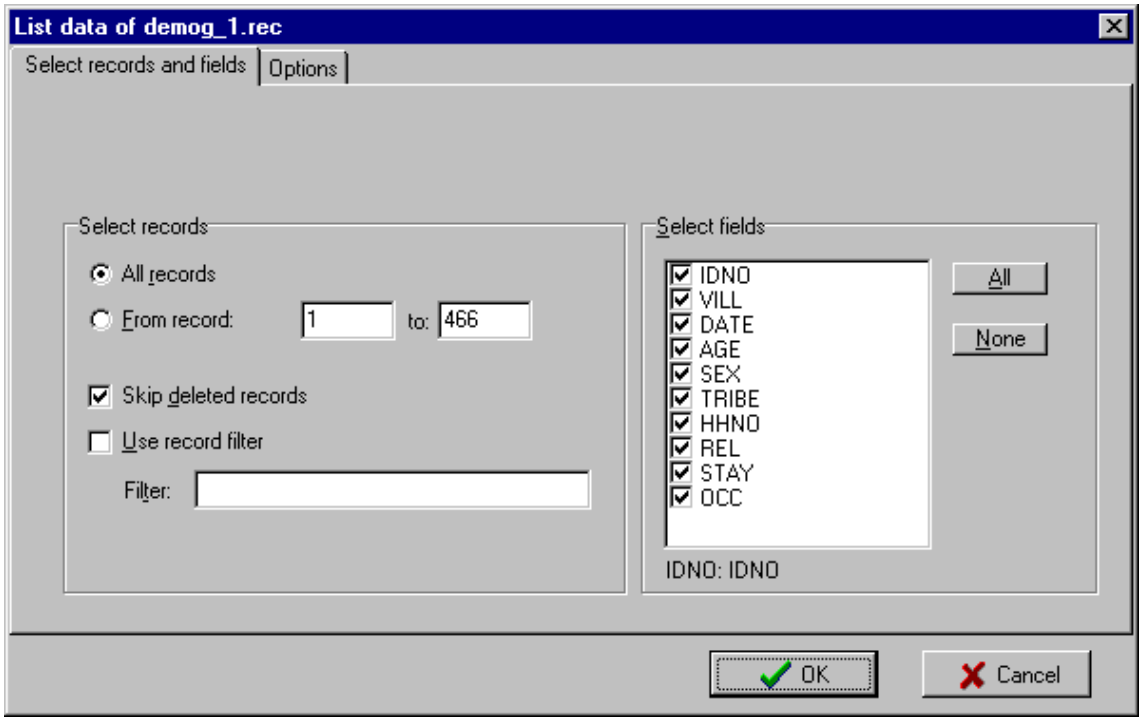
When you have finished examining the data click the close document window control (or select the **`File -> Close form`** menu option, or press CTRL + F4) to close the data form.

Note that records are not deleted immediately but are *marked for deletion*. Records that are marked for deletion can be included or excluded from some data checking, documentation, and export functions. Marking records for deletion instead of deleting records immediately is also a safety measure. Records that have been marked for deletion can be 'undeleted' using the delete data control or the **`Goto -> Undelete`** record menu option. Records that are marked for deletion can be permanently deleted using the **`Tools -> Pack data file`** menu option.

Using *EpiData* to examine a dataset

EpiData can also produce lists of data. Select the **`Document -> List Data`** menu option. Select the file **`demog_1.rec`** and click the **`Open`** button.

This will display the **`List Data`** dialog box:



Note that you can specify the records to list by specifying a range of record numbers, whether records that are marked for deletion are to be included in the list, a *filter* that specified which records are to be listed, the fields to be listed. Using the **`Options`** tab you may specify the width of the list, the number of columns in the list, a sort order based on an indexed variable, and whether value labels rather than the data values themselves should be listed.

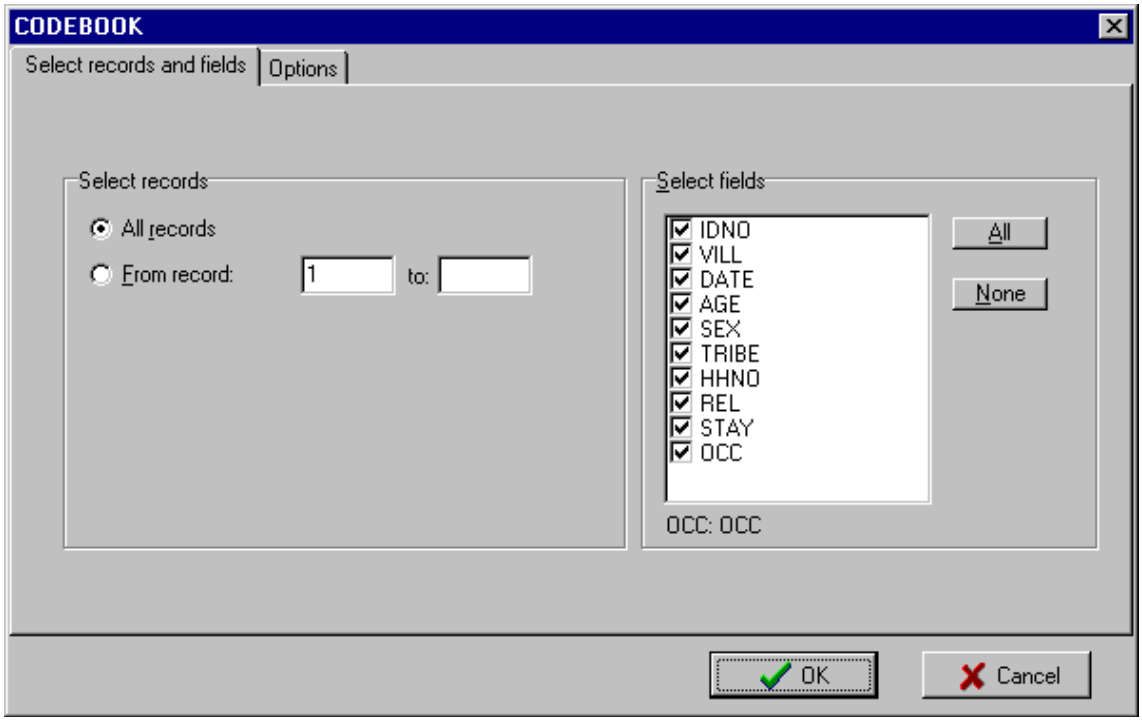
Accept the default options by clicking the **`OK`** button. This displays a list of all fields in all records.

Scroll through the list and examine the data listing. When you have finished examining the data click the close document window control (or select the **`File -> Close`** menu option, or press CTRL + F4) to close the data form. You do not need to save the data listing.

Using *EpiData* to examine a dataset

EpiData can also produce summary statistics from your data.

Select the **`Document -> Codebook`** menu option. Select the file **`demog_1.rec`** and click the **`Open`** button. This will display the **`Codebook`** dialog box:



Note that you can specify the records to include, the fields to be summarised, and, using the **`Options`** tab, the level of detail that will be used when displaying data checking rules (if any have been specified).

Accept the default options by clicking the **`OK`** button. This displays summary statistics for the file and for each variable.

Scroll through the *codebook* and examine the output. When you have finished examining the *codebook* click the close document window control (or select the **`File -> Close`** menu option, or press CTRL + F4) to close the data form. You do not need to save the *codebook*.

EpiData can also produce a report summarising the structure of a data file. Select the **`Document -> File Structure`** menu option. Select the file **`demog_1.rec`** and click the **`Open`** button. This will display a report summarising the structure of the selected data file.

Scroll through the report and examine the output. When you have finished examining the report click the close document window control (or select the **`File -> Close`** menu option, or press CTRL + F4) to close the data form. You do not need to save the report.

Making a data entry form

Since our data comes from an onchocerciasis study we might call the questionnaire (.QES) file **`oncho.qes`**. We can use the editor 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 spacing 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 and with the variables lined-up one below the other. This can make data entry less error prone.

Creating a .QES file using the *EpiData* editor

We shall create a questionnaire (.QES) file using the ***EpiData*** editor. Variable names are typed into the editor window 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. Click the **`Define Data`** button on the work process toolbar and the **`New .QES file`** option. This will open a new editor window. In the editor window 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}  #
```

The ***EpiData*** editor has two features that makes it easy to define variable types and lengths:

> The **field pick list** displays a dialog box that allows you to specify variable definitions and insert them into the questionnaire. The field pick list may started (and stopped) from the editor toolbar, from the **`Edit`** menu, or by pressing CTRL + Q .
>
> The **code writer** watches what you type. If you type some field definition characters (e.g. **`#`**, **`_`**, **`<A`**, **`<d`**, **`<m`**, **`<Y`**, **`<I`**, **`<S`**) ***EpiData*** will prompt you for the information required (e.g. length, number of decimal places) to complete the variable definition. The code writer may be started (and stopped) from the editor toolbar, from the **`Edit`** menu, or by pressing CTRL + W .

You can type variable definitions directly into the editor window without using either the field pick list or the code writer. You need only use these functions if you find them useful.

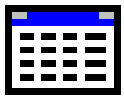
EpiData will automatically take the first eight non-blank characters of text prior to a variable definition and make this the variable name (this default behaviour can be changed, see Appendix 2 for details of how to change ***EpiData*** default behaviours.). Some words such as 'what', 'are', and 'of' are discarded automatically (e.g. **`Date of onset`** becomes **`DATEONSE`**). If you precede the text with a number, ***EpiData*** will add the prefix **`N`** (e.g. the text **`'2.Age'`** becomes **`N2AGE`**). This can lead to unsuitable variable names (e.g. **`YEARSLIV`** 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 questionnaire (.QES) file

Check your work carefully against the example questionnaire.

You can see what the data entry form will look like by clicking the preview data form button:



on the editor toolbar, selecting the **`Data File -> Preview Data Form`** menu option, or by pressing CTRL + T . You can switch between the editor and preview data form window using the Windows menu, clicking on the relevant tabs at the bottom of the ***EpiData*** window, or by using CTRL + ↹ .

The preview data form window is not updated automatically when you make changes to the questionnaire. You must request a new preview (i.e. by clicking the *preview data form* button on the editor toolbar, selecting the **`Data file -> Preview data form`** menu option, or by pressing CTRL + T) in order for the preview data form window to be updated to reflect any changes you may have made to the questionnaire.

When you are satisfied save the questionnaire file by clicking the *save* icon on the editor toolbar or by selecting the **`File -> Save`** menu option. When prompted give the filename **`oncho.qes`** and click **`Save`**. The questionnaire (.QES) file is saved to disk as **`oncho.qes`**. Close the editor window.

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 any Windows program. 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 ***EpiData*** 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 CTRL + S .

The characters **<**, **>**, **_**, and **#** are used by EpiInfo to define variables. Avoid using them anywhere else in the questionnaire (.QES) file. You should also avoid using the **`@`** character.

Editing the questionnaire (.QES) file

Click the *open file* icon on the editor toolbar or select **`File -> Open`**. Select the file **`oncho.qes`** and click the **`Open`** button to recall the file that you just created.

Which variables are defined as **NUMERIC**?

Name	*EpiData* definition

Which variable is defined as **UPPER CASE TEXT**?

Name	*EpiData* definition

Which variable is defined as **DATE**?

Name	*EpiData* definition

Which variable is defined as **YES/NO**?

Name	*EpiData* Definition

Editing the questionnaire (.QES) file

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

Name	*EpiData* definition

By default, ***EpiData*** will automatically use the first eight characters of text (excluding blanks) prior to a variable specification and make this the variable name. This may not give you the variable names that you want. The best way of overcoming this is to put **{ }** round the characters that you want to be the variable name. If the **{ }** characters were removed from **{LVILL}** what name would ***EpiData*** 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 ***EpiData*** variable definitions. Experiment with the field pick list and code writer methods of inserting the variable definitions. Coding information is needed on the field questionnaires but will tend to clutter up the data entry form.

Save the file using a new file name of your own choosing. **Do not** overwrite the original questionnaire (**oncho.qes**) which we will use later.

Close the editor window. Close ***EpiData***.

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 data (.REC) file

Start ***EpiData*** and click the **`Make Data File`** button on the work process toolbar.

In response to the request for a .QES file select **`oncho.qes`**.

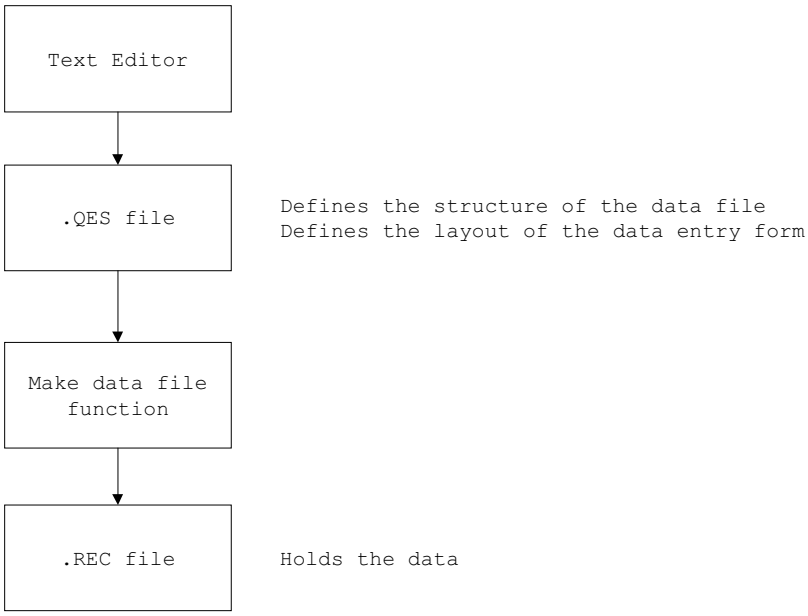
In response to the request for the name of the data file specify **`oncho.rec`**.

Click the **`OK`** button.

In response to the request for a data file label type **`Onchocerciasis Baseline Survey`**.

EpiData will create a data (.REC) file **`oncho.rec`** using the information contained in the .QES file (**`oncho.qes`**) and report that the file has been created. Click the **`OK`** button.

You have completed the process:



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

Entering data

Click the **Enter Data** button on the work process toolbar. Specify the file **oncho.rec** and click the **Open** button. ***EpiData*** will display the data entry form for **oncho.rec** ready to receive data.

Fill in the blanks on the data entry form 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 status bar at the bottom of the ***EpiData*** window. 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 ***EpiData*** will warn you. 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 ***EpiData***. 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 data can be analysed.

You may need to press ENTER to move onto the next variable if the data you enter does not completely fill a particular variable.

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

After you have entered all the data for one case (record) the message **Save record to disk?** will be displayed. Click the **Yes** button or press ENTER . A new (blank) data form will be displayed. Note that the text **New/1** is displayed in the lower right corner of the ***EpiData*** window. Enter data for all of the sample cases, writing each to disk. When you have finished, close the data entry form.

If you click the **No** button when asked to **Save record to disk?** You will be able to edit the data you have just entered. To save this move to the end of the record and press ENTER , or select the **Goto -> New Record** menu option, or type CTRL + N , or click the new record (*****) data control and then click the **Yes** button on the **Save record to disk?** message box.

Editing data

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

Click the **`Enter Data`** button on the work process toolbar. Specify the file **`oncho.rec`** and click the **`Open`** button. ***EpiData*** will display the data entry form for **`oncho.rec`** ready to receive data.

The data controls:

Control	Function
	Display **first** record in the data file
	Display **previous** record in the data file
	Display **next** record in the data file
	Display **last** record in the data file
	Display a **new** blank record ready to receive data
	**Mark** the currently displayed record **for deletion**

provide a means of moving from case to case within a data file. These functions are also available from the **`Goto`** menu and also have keyboard shortcuts (these are shown on the **`Goto`** menu).

Move through the cases that you just entered and review your work.

Edit a case by clicking on a variable and changing its contents. Save the edited case by pressing CTRL + END followed by ENTER and clicking the **`Yes`** button when asked to **`Save record to disk?`** You will also see this prompt if you change any data and move to a different record or a new blank record before saving the changed data.

Display a new blank record by pressing CTRL + N .

You may have noticed that ***EpiData*** provides several different ways of accessing the same function. For example, the function to display a new blank record can be accessed using the menus (**`Goto -> New Record`**), using a keyboard shortcut (CTRL + N), or by clicking the relevant data control. Use whichever method you feel most comfortable with.

Searching for data

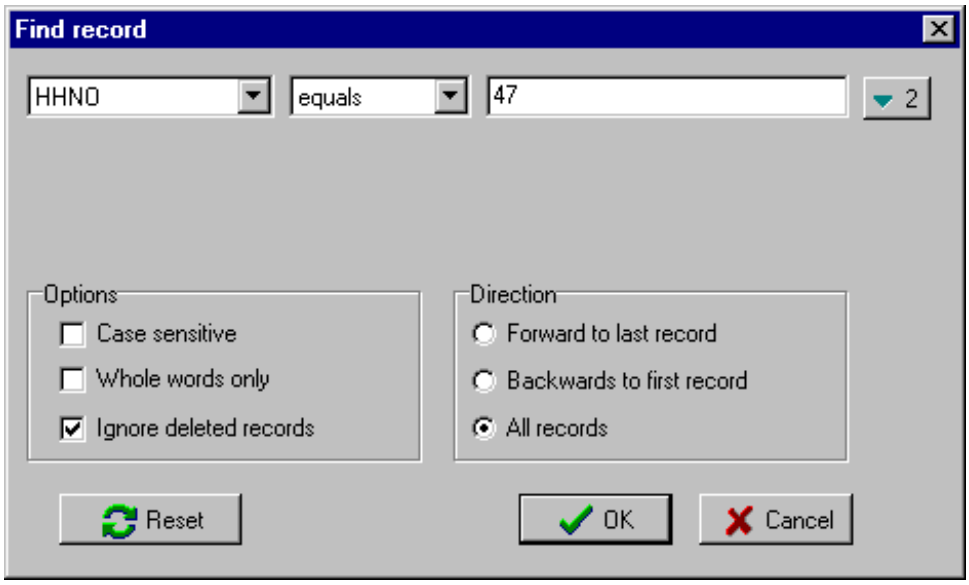
When you have many cases in a data file it can be time consuming to find a single case, or a particular set of cases. You can find a particular case (or set of cases) using **the Goto -> Find Record** menu option or the keyboard shortcut CTRL + F . This will display a dialog box where you can enter the data that *matches* the data held in the case(s) you wish to find. In this exercise we will search for members of household number 47.

Press CTRL + F to display the find record dialog box.

In the first drop-down list select the variable name **HHNO**.

Make sure that the second drop-down list specifies the **equals** option.

Enter **47** into the remaining text box:



Click the **OK** button.

EpiData displays the first matching case (i.e. the first case with **HHNO = 47**). To display the next matching case select the **Goto -> Find Again** menu option or press the F3 key.

You can edit any displayed record. Press CTRL + N to enter a new case. Press CTRL + F to specify a new set of *search criteria*.

Searches can be performed on up to three different fields by searching on the full contents of a field (**equals**), the start of a field (**begins with**), or part of a field (**contains**). If you search for data in more than one field then ***EpiData*** returns records which meet **all** of the search criteria. Experiment with the record finding functions.

Try searching for data in more than one field at the same time. Start by clicking on the 

button.

Close the **oncho.rec** data entry form.

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 ***EpiData*** to set up interactive checks.
- ❑ Double enter and compare data using ***EpiData***.
- ❑ Use ***EpiData*** to implement batch checking of data.
- ❑ Use ***EpiData*** 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)**. ***EpiData*** provides functions that allow 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)**. ***EpiData*** can also check data after it has been entered.
>
> **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. ***EpiData*** provides functions for double-entry and validation of data.

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.

Data checking functions in *EpiData*

Even without specifying data checks, ***EpiData*** 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`**, **`N`**, **`1`**, or **`0`** into **YES/NO** variables. It is much better, however, to specify more specific checks and ***EpiData*** provides functions that allow you to do this. These functions include:

Must-enter variables. You can 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. You may mix range checks and legal values (e.g. for missing value codes).

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

Conditional jumps. You can 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. ***EpiData*** also provides an easy-to-use block-structured programming language that allows you to program more complex checking procedures (e.g. consistency checks).

The **`Checks`** function provides one type of data checking called *interactive checking* (i.e. the data is checked as it is entered). The **`Validate Files`** function allows for *batch* (i.e. all cases at once) checking of data that has been *double-entered*.

Batch checks for range and consistency may also be performed using the CHECK programming language which is built into ***EpiData***.

Setting up checks

Start ***EpiData*** if it is not already started. Click the **Checks** button on the work process toolbar and select the **Add / Revise** option from the drop-down menu. Select the file **oncho.rec** and click the **Open** button. The data entry form appears again but is accompanied by a dialog box that allows you to specify data checking rules:

Make sure that **IDNO** is displayed in the drop-down list at the top of this dialog box and type **1001-3999** in the **Range/Legal** box and press ENTER to confirm this.

Click the  button next the **Must-enter** drop-down list and select **Yes** to make this a must-enter field:

Click the **Save** button to save the checks.

Setting up checks

Select the **AGE** variable (you can select the variable using the drop down box or by clicking on the variable's data entry box on the data entry form) and type **12-70,99** into the **Range/Legal** box to specify an allowable range of values of 12 to 70 and a legal value of 99 (for missing data). Remember to press ENTER to confirm the check.

Select the **SEX** variable. Click the button marked **+** next to the **Value label** box add codes and labels for *male*, *female*, and *unknown* sex in the editor window:

```
LABEL Label_SEX
  M Male
  F Female
  U Unknown
END
```

This sets up a value and label set that specifies the values you may enter into the **SEX** field as well as the meaning of each value (the *value label*). Click the **Accept and Close** menu option. Note that the **Value label** box now contains the name of the value and label set **label_sex**.

Select the **TRIBE** variable. As **TRIBE** is likely not to vary much between batches of questionnaires you should set **Repeat** to **Yes** for this variable. Click the button marked **+** next to the **Value label** box and type the following data into the editor window:

```
LABEL Label_TRIBE
  1 Mende
  2 Other
END
```

Click the **Accept and Close** menu option. Note that the **Value label** box now contains the name of the value and label set **label_tribe**. Select the **HHNO** variable and specify a range of 1 to 50 and allow the entry of 99 for missing data. Select the **REL** field and specify the following value and label set:

```
LABEL Label_REL
  1 Self
  2 Parent
  3 Child
  4 Sibling
  5 "Other blood"
  6 Spouse
  7 Other
  8 Friend
END
```

When a value label consists of more than one word, the value label must be enclosed in double quotes (") as is done with **"Other blood"** above.

Setting up and checking your checks

Select the **LVILL** variable and type **Y>OCC** in the **Jumps** box and press ENTER to specify that a value of **Y** entered into the **LVILL** variable will cause a jump to the **OCC** variable.

Select the **OCC** variable and specify a value and label set for the occupations (coded 0, 1, ..., 8) of none, home, school, farming, fishing, office, trading, housework, and mining.

When you have specified all of these checks click the **Save** button to save the checks and then click the **Close** button.

Enter some more data into **oncho.rec** (make some up) and verify that the must-enter, repeat, range, legal value, and jumps checks that you specified work correctly.

Note that you can use the F9 or + key to display value and label sets as a pick-list.

If your checks do not work as expected then close the data entry form and use **Checks -> Add / Revise** to edit your checks. Enter some more data into **oncho.rec** (make some up) and verify that the checks work correctly.

Close the data entry form when you are satisfied that your checks are working as expected.

Programming data entry

When you specify checks, ***EpiData*** 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 at data entry time. The check (.CHK) file is a plain text file and can be edited using the ***EpiData*** editor.

Click the **Checks** button on the work process toolbar and select the **Add / Revise** option from the drop-down menu. Select the file **oncho.rec** and click the **Open** button.

Move the cursor to the **IDNO** variable and click the **Edit** button. A small editor window shows the CHECK commands associated with the **IDNO** variable:

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

Click the **Cancel** menu option or press ESC to close the editor window.

Point to each variable in turn and click the **Edit** button to examine the CHECK commands associated with each variable. **Do not** alter any of the commands.

EpiData CHECK language commands are stored in blocks associated with each variable. The block has the general form:

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

Details of the ***EpiData*** CHECK language can be found in the ***EpiData*** help file.

Specifying a KEY variable

EpiData 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 analyses and reports.

Move the cursor to the **IDNO** variable and click the **Edit** button. ***EpiData*** displays the CHECK commands associated with this variable:

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

Edit the CHECK commands associated with this variable to read:

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

Click the **Accept and Close** menu option. Click the **Save** button to save the checks and then click the **Close** button.

Enter two cases (make up some data) with the same value in the **IDNO** variable into **oncho.rec**. ***EpiData*** displays a warning message and will not accept the duplicate value for the second case.

Click the **OK** button and change the value in the **IDNO** variable and save the new record. Close the data entry form when you are finished.

The CHECK language 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 commands **KEY** and **KEY UNIQUE** a new file with the same name as the record (.REC) file but with the extension .EIX is created. This *index file* speeds up the way ***EpiData*** searches for data.

Using value and label sets during data entry

When you specify a value and label set you are specifying both the legal values for a field as well as the meaning of each of the legal values. ***EpiData*** uses this information to create pick lists during data entry (i.e. using F9 or +) as well as to limit the entries allowed in a field.

Value and label sets are also used when data are exported to packages, such as SAS, SPSS and STATA, that can use value labels.

We can also use the value and label sets to display information on the data entry form during data entry. To do this we need to use another CHECK language command.

Click the **Checks** button on the work process toolbar and select the **Add / Revise** option from the drop-down menu. Select the file **oncho.rec** and click the **Open** button.

Select the **SEX** variable and click the **Edit** button. Edit the CHECK commands associated with this variable to read:

```
SEX
  TYPE COMMENT
  COMMENT LEGAL USE label_sex
END
```

Click the **Accept and Close** menu option.

The **TYPE COMMENT** command causes ***EpiData*** to display the label associated with an entered value during data entry.

The **COMMENT LEGAL USE label_sex** command was inserted automatically when you specified the **label_sex** value and label set earlier:

```
LABEL Label_SEX
  M Male
  F Female
  U Unknown
END
```

And causes ***EpiData*** to use this value and label set to specify the values you may enter into the **SEX** field as well as the meaning of each value (the *value label*). The **COMMENT LEGAL USE** command allows you to use one value and label set for more than one variable if required. For example, you could use the command **COMMENT LEGAL USE label_sex** in the CHECK command block of any variable that needed to use the same value and label set.

Edit the CHECK commands for the **TRIBE**, **REL**, and **OCC** variables to include a **TYPE COMMENT** command.

Click the **Save** button to save the checks and then click the **Close** button.

Enter another case (make up some data) into **oncho.rec**. Note that ***EpiData*** now displays the appropriate value label next to the variables that have the **TYPE COMMENT** command in their CHECK language block. Close the data entry form.

Limitations of interactive data checking

EpiData provides very powerful functions 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 ***EpiData*** may 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 (i.e. real-but-unlikely) values.

EpiData 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 (specified using **KEY UNIQUE** in the check (.CHK) file). 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

We will first create two new data files and set up data checking rules. ***EpiData*** makes this easy by providing functions specifically designed to help you do this.

Close any open editor windows and data entry forms and select the **`Tools -> Copy Structure`** menu option. Select the file **`oncho.rec`** and click the **`Open`** button. In the text box labelled **`File name`** type **`oncho1a.rec`** and check the **`Copy check file`** option. Click the **`OK`** button. This will create an empty data file called **`oncho1a.rec`** with exactly the same structure as **`oncho.rec`** and also create a file called **`oncho1a.chk`** which contains the data checks that you specified for **`oncho.rec`**. ***Epidata*** will inform you that the files have been copied. Click the **`OK`** button.

Enter the following data into the **`oncho1a.rec`** data file:

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

Close the data entry form when you have entered all five cases.

Use **`Tools -> Copy Structure`** to make a new data file called **`oncho1b.rec`** based on **`oncho.rec`** and enter the same sample cases into this file. Make two or three deliberate mistakes when entering data into this second file.

Close the data entry form when you have entered all five cases.

Double entry and validation

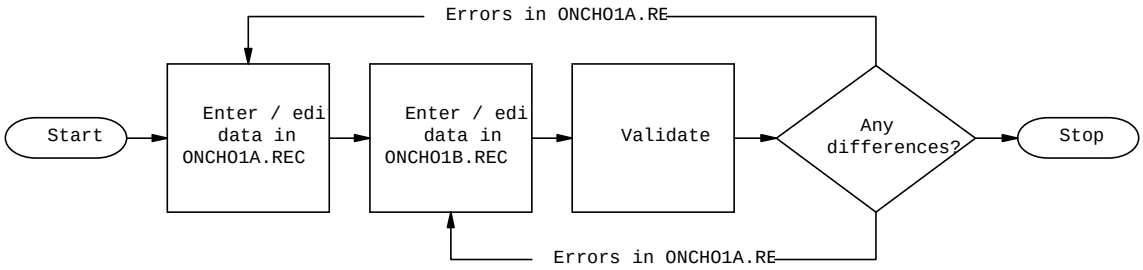
Select **`Document -> Validate Files`** menu item. In response to the prompts for the names of the two files specify (you can type the filenames or select from a file browser by clicking the folder icon next to the entry boxes) the files **`oncho1a.rec`** and **`oncho1b.rec`** and click the **`OK`** button.

Specify **`IDNO`** as the *key* field (i.e. the field that will be used to identify the records in each file that 'belong' to each other) and click the **`OK`** button.

EpiData responds with a list of discrepancies (by **`IDNO`**) between the two files displayed in an editor window. The list is a text file which you can edit, save, or print.

Print the file or make a note of the reported discrepancies. Close the editor window. Edit the data in the two files so that they are the same and use **`Document -> Validate Files`** 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:



Validation checks whether data entered into two separate files is the same. It does not check the validity of the data. Interactive checking allows you to check the validity of the data as it is entered. To ensure good quality data you should use both interactive checking and validation.

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.

Data consistency

So far we have used ***EpiData*** 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 CHECK language commands for both interactive and batch data checking.

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 the 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.

Use the ***EpiData*** editor to create a new questionnaire file:

```
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 save the questionnaire file as **mums.qes**. Use this file to create a new data file called **mums.rec**.

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**):

```
AGEM >= 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 (i.e. where **AGEM - AGEP = 0**) we need to add **2** to **AGEM** to perform consistency checks in this way:

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

Consistency checks

Enter the following data into the **`mums.rec`** data file:

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.

Close the data entry form when you have entered all ten cases.

Now that we have entered the data we will use the ***EpiData*** editor to create a file containing the consistency checking rules for this data.

Consistency checks

EpiData provides three CHECK language commands (**CONSISTENCYBLOCK**, **REPORT**, and **CHECK**) for specifying batch consistency checks:

The **CHECK** command provides an example of a *good* item of data. For example:

```
CHECK "Ages not consistent" AGEM >= AGEP
```

Specifies that a *good* record is one in which the age of the mother (**AGEM**) is greater than or equal to (>=) the age at first parity (**AGEP**). The text between the double quotes (**"**) describes the purpose of the consistency check.

The **REPORT** command specifies which variable is used to identify records that fail consistency checks. The use of the **REPORT** command is optional. If you do not use the **REPORT** command ***EpiData*** will display record numbers.

The **CONSISTENCYBLOCK** command acts as a 'wrapper' around the **CHECK** and **REPORT** commands. **CONSISTENCYBLOCK** is always paired with an **END** command. The **CONSISTENCYBLOCK** command allows you to include consistency checks in the normal interactive check file.

Open a new editor window and type the following CHECK language commands:

```
CONSISTENCYBLOCK
  REPORT IDNO
  CHECK "Ages not consistent" AGEM >= AGEP
  CHECK "Birth interval" (AGEM + 2 - AGEP) / BIRTHS >= 2
END
```

Save these commands in a file called **mums.chk** and close the editor window.

Select the **Document -> Consistency Checks** menu option and specify the files **mums.rec** and **mums.chk**. Click the **OK** button. ***EpiData*** will respond with a report listing the value of the **IDNO** variable for records that do not pass the consistency checks. Examine the report and compare the results against the data (shown on the previous page). Close the editor window when you are finished.

In the above example we specified consistency checks. If you add a **CONSISTENCYBLOCK** to an existing check (.CHK) file you can use special forms of the **CHECK** command to apply basic interactive checks as batch checks:

```
CHECK "Legal values" CHECKLEGAL
CHECK "Ranges" CHECKRANGE
CHECK "MustEnter" CHECKMUSTENTER
```

Interactive consistency checks

It is also possible 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.

Specify the following range checks for the **mums.rec** data:

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

Move the cursor to the **AGEP** variable click the **Edit** button. 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
```

Click the **Accept and Close** menu option.

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 displays a message.

The **GOTO** command causes the cursor to move 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

Edit the commands associated with the **BIRTHS** variable to read:

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

Save these changes and 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

Close the data entry form when you are finished.

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 **Document -> Consistency Checks** to check data that is already in the file.

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.

Relational data entry

In the onchocerciasis example, we have data from several sources:

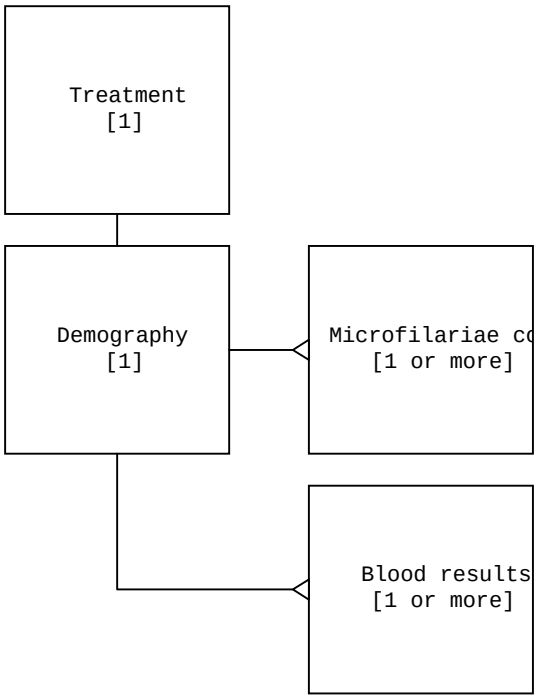
Each person has **one** demographic record

Each person has **one or more** microfilariae count records

Each person has **one or more** blood results records

Each person has **one** treatment code record

This can be expressed better in a diagram:

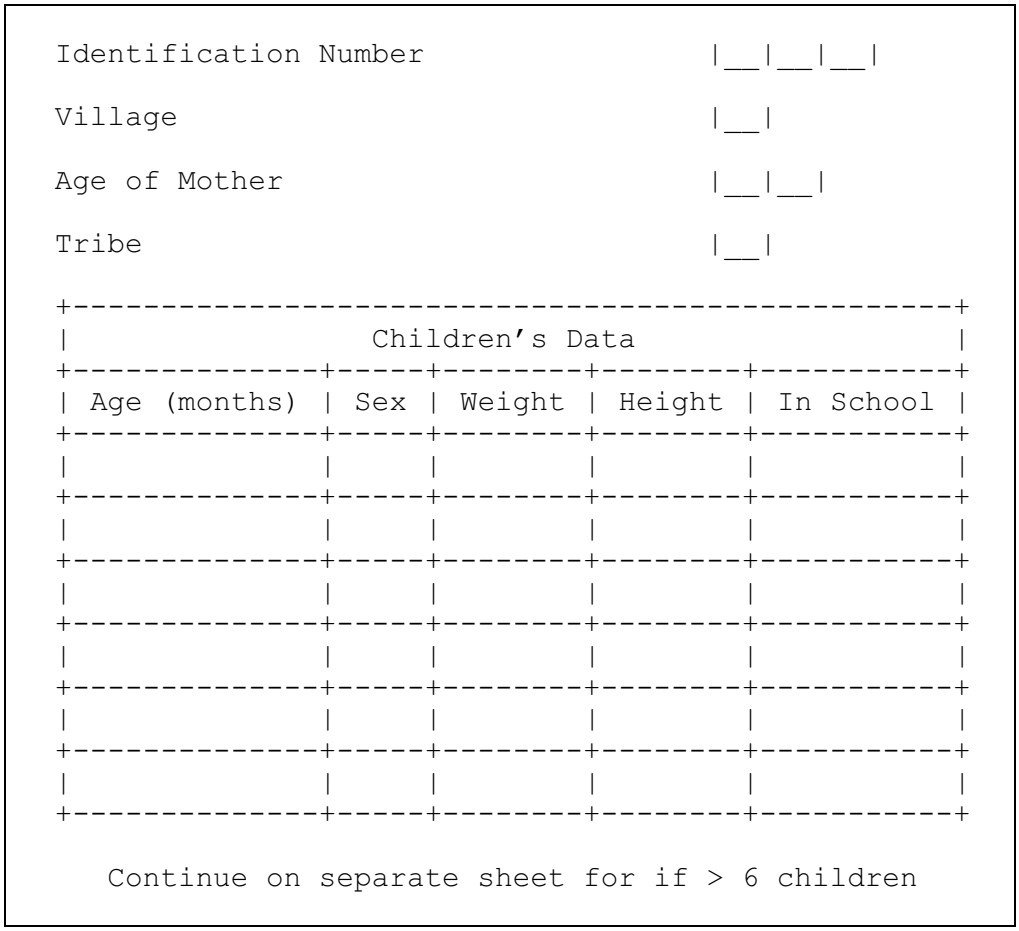


With the onchocerciasis example, the data come from separate sources (questionnaires, laboratory records for microfilariae counts, laboratory records for blood results, and the randomisation schedule) and it is a relatively simple matter to enter data from each source into a separate data file and combine them as required (this is covered in the next section of this manual).

Some data, however, will arrive from a single source but will contain data about two (or more) different type of subject. As an example, imagine a survey that collects data on mothers and their children. In this example, each data collection form will contain data about **one** mother and **one or more** children. We could create one file with enough space for up to ten children but this would result in a very wide file with many empty variables. Such a file would also be difficult to analyse if, for example, you wanted to produce a summary measure of the weight of the children surveyed. In this case we want a way of entering the data into two separate files - one for mothers and another for children. ***EpiData*** provides functions that allow you to do just that.

Relational data entry

The form below is a shortened and simplified version of a data collection form that might be used to collect data on mothers and their children:



To enter this data we need to create two files - one for mothers and another for children.

Relational data entry

Create a new questionnaire (.QES) file called **`mothers.qes`** which looks like this:

```
Identification Number                  {IDNO} ###

Village                                {Vill} #
Age of Mother                          {AgeM} ##
Tribe                                 {Tribe} #
```

Use this file to create a new data (.REC) file called **`mothers.rec`**. This file will hold the mothers' data.

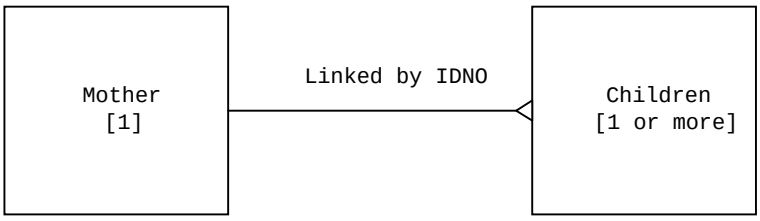
Create a new questionnaire (.QES) file called **`children.qes`** which looks like this:

```
Identification Number                  {IDNO} ###

Age (months)                            {Age} ##
Sex                                     {Sex} #
Weight (kg)                              {Wt} ##.#
Height (cm)                              {Ht} ###.#
In School?                            {School} <Y>
```

Use this file to create a new data (.REC) file called **`children.rec`**. This file will hold the children's data.

Note that both files contain a variable called **`IDNO`**. This variable has the **same** name (**`IDNO`**), type (**NUMERIC**), and length (three digits) in both files. This variable is used to link or *relate* the two files together and specifies which children belong to a particular mother (and vice-versa):



It is important that the linking variable is identical (i.e. in name, type, and length) in both files.

Relational data entry

EpiData provided CHECK language commands to manage relational data entry:

> **KEY** is used with the linking variable (in this case **IDNO**) of the *child* file (in this case **children.rec**). This command ensures that searches / filters and other functions and commands that use the linking variable will work as fast as possible.
>
> **KEY UNIQUE** is used with linking variable (in this case **IDNO**) in the *parent* file (in this case **mothers.rec**). This command ensures that the linking variable is unique in the parent file (i.e. that children may only have one mother) and that searches / filters and other functions and commands that use the linking variable will work as fast as possible.
>
> **RELATE** is used in the variable where the link between the two files will be activated. In this example it makes sense to place the **RELATE** command in the **TRIBE** variable block.

Use the data checking functions of ***EpiData*** to add the following checks:

Data (.REC) File	Variable	CHECK command block
children.rec	IDNO	IDNO KEY END
mothers.rec	IDNO	IDNO KEY UNIQUE END
	TRIBE	TRIBE RELATE IDNO CHILDREN.REC END

Close all windows and open the **mother.rec** file for data entry. Note that both the **mothers.rec** and the **children.rec** files are opened and that each is displayed on a separate (tabbed) data entry form.

Enter some data into **mothers.rec** (make some up). When you have filled the **TRIBE** variable the focus should shifts to **children.rec**. Note that the **IDNO** variable in **children.rec** is filled automatically with the value that you entered into **mothers.rec**. You can enter more than one record into **children.rec**. When you have completed entering all the children for one mother press F10 to return to **mothers.rec** where you may now enter data for another mother (and her children)

Note that you can click on the tabs at the bottom of the data entry forms to move between the files but you cannot enter data this way. You can only examine data. Relational data entry remains under the control of the **RELATE** command.

Data Management

Introduction and objectives

In this section we will discuss some aspects of data management that can be performed with ***EpiData***. Data management can be split into two broad categories:

> 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.
>
> 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.

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.

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 ***EpiData*** data (.REC) files. Interactive checks were specified for legal values and ranges. Batch consistency checks were applied. Data were entered twice and validated.

Five 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
blood.rec	Blood results
micro.rec	Microfilariae counts
tmtcodes.rec	Randomisation codes

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

Open the file **demog_1.rec** in ***EpiData*** using **Data -> Enter** and browse through the file. Take note of the variable names and values.

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 **blood.rec**, **micro.rec** and **tmtcodes.rec**.

Another way of looking at the structure of a data file is to use the **Document -> File Structure** function. This function displays a list of variables in the current data file with information about their type and length and details of checks, if any have been specified.

You can also examine the data using the **Document -> Codebook** function (summary statistics) and the **Document -> List Data** function (list of data).

If you use **the Document -> List Data** function to examine data you might find it quicker and easier to examine just the first few records (e.g. the first fifty records) of each data file.

Concatenation

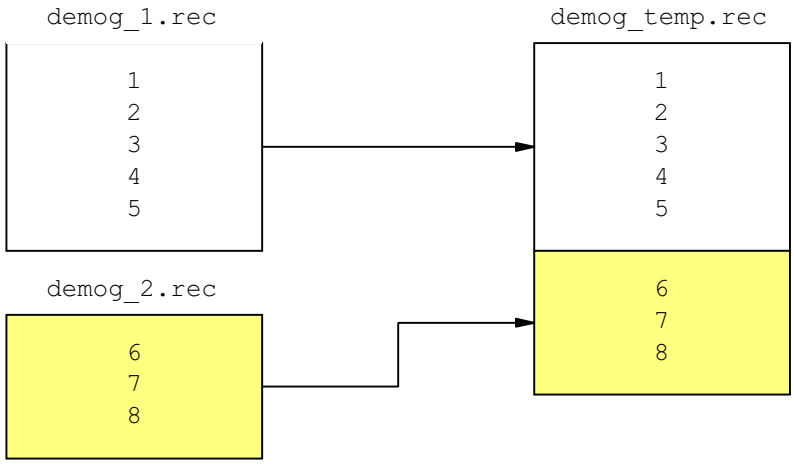
Demographic questionnaires were filed by **IDNO** in the field office. The data entry task was simplified by entering date into three separate files for **IDNO**s 1-999, 1000-1999 and 2000+. The same questionnaire (.QES) file was used and the end result was three different files of exactly the same structure (**demog_1.rec**, **demog_2.rec**, and **demog_3.rec**). We want to join these three files end-to-end to create a single file for analysis.

The process of joining files end-to-end is called *concatenation*. The **Data -> Append / Merge** function in ***EpiData*** will allow us to concatenate (i.e. join together end-to-end) different data files. At each pass, **Append / Merge** can join two files to make a combined file.

Close all editor windows and data forms. Select the **Data -> Append / Merge** menu option. Select the files **demog_1.rec** and **demog_2.rec** and click the **OK** button.

In the box labelled **Resulting data file** type the file name **demog_temp.rec** and click the **Append** button.

The **Append** operation will create a new data file, called **demog_temp.rec**, that contains the records from both input files:



You can check this by selecting the function **Document -> Data Entry Notes** and specifying the file **demog_temp.rec**.

EpiData keeps track of operations performed on files in Data Entry Notes (.NOT) files.

Concatenation

Close all editor windows and data forms.

Repeat the **Append** operation but this time concatenate the **demog_temp.rec** file with **demog_3.rec**. Call your output file **demog.rec**.

Select the function **Document -> Data Entry Notes** and specifying the file **demog.rec**. There should be 1625 records in the file **demog.rec**.

The file **demog_temp.rec** is no longer required and may safely be 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 sample dataset.

Close all windows and open the data file **blood.rec**.

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**

Close the data form.

Select the option **Document -> Codebook** and specify the file **blood.rec**. Choose to produce a code book for all records. Examine the output for the **EOSIN**, **PCV**, and **MPS** variables. You should be able to see that **9999**, **99**, and **9** are reported as the maximum values for each variable.

Close the code book window and select the option **Document -> Count Records**. Specify the file **blood.rec**. Select the **EOSIN** variable and click the **OK** button. The **Count Records** function produces a basic frequency table of the **EOSIN** variable. Scroll through the output and note that there are 207 records in which **EOSIN** is recorded as **9999** (missing).

Repeat the **Count Records** function for the **PCV** and **MPS** variables.

There are a substantial number of missing values for each variable. These missing values must be set to missing prior to analysis. Most data analysis software will provide functions to identify or recode missing values but you can also do this in ***EpiData***.

Recoding missing values

EpiData provides CHECK language commands that allow you to recode missing values as well as create and transform variables. Recoding instructions are specified in a **RECODEBLOCK** block in a .CHK file.

Close all windows. Open an editor window and type the following CHECK language commands:

```
RECODEBLOCK
  IF EOSIN = 9999 THEN
    CLEAR EOSIN
  ENDIF
  IF PCV = 99 THEN
    CLEAR PCV
  ENDIF
  IF MPS = 9 THEN
    CLEAR MPS
  ENDIF
END
```

Save these commands in a file called **blood_missing.chk** and close the editor window.

Select the **Tools -> Recode Data** menu option and specify the files **blood.rec** and **blood_missing.chk**. Click the **OK** button. ***EpiData*** will respond with a message telling you how many records will be changed. Click the **OK** button to recode the data.

After performing the recode operation ***EpiData*** reports that it has made the changes and created a backup of the original data file as **blood.old.rec**. Click the **OK** button.

Examine the file **blood.rec** and check that the missing values codes have been recoded to missing (blank) values.

Check the data entry notes file for **blood.rec** to see a record of the **Recode Data** operation.

In this example, we recoded the missing values to blanks, using the CLEAR command. Blank fields are recognised by ***EpiData*** and EpiInfo as missing.

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 the right and left skin samples. The variables **SSRIC** and **SSLIC** record the diameters of the right and left skin samples.

Examine the data in the file **micro.rec**. and familiarise yourself with this file.

There are four things that we need to do with this file before it is ready for data analysis. These are:

1. Recode missing value codes to missing (blank) values.

2. Create a variable that contains the numbers of microfilariae observed in both skin snips.

3. Create a variable that indicates the presence of microfilariae in either skin snip.

4. Create a variable to hold the density per mg of microfilariae.

EpiData provides functions that allow us to perform each of these tasks.

Recoding missing values

Close all windows and open a new editor window. Type the following CHECK language commands:

```
RecodeBlock
  if MFRIC = 999 then
    clear MFRIC
  endif
  if MFLIC = 999 then
    clear MFLIC
  endif
  if SSRIC = 0 then
    clear SSRIC
  endif
  if SSLIC = 0 then
    clear SSLIC
  endif
End
```

Save these commands in a file called **micro_missing.chk** and close the editor window.

Note that ***EpiData*** is not sensitive to the case of CHECK language commands. **IF**, for example, may be written as **IF**, **If**, **iF**, or **if**.

Select the **Tools -> Recode Data** menu option and specify the files **micro.rec** and **micro_missing.chk**. Click the **OK** button. ***EpiData*** will respond with a message telling you how many records will be changed. Click to **OK** button to recode the data. ***EpiData*** reports that it has made the changes and created a backup of the original data file as **micro.old.rec**. Click the **OK** button.

Examine the file **micro.rec** and check that the missing values codes have been recoded to missing (blank) values. Check the data entry notes file for **micro.rec** to see a record of the recode operation.

Creating and transforming variables

To add variables to the file we will edit the questionnaire (.QES) file, add variable definitions for the new variables, and restructure the data file so that it includes these new variables.

In this example we do not have the original questionnaire (.QES) file that was used to create the **micro.rec** file so we need to recreate it. Close all windows and select the **Tools -> QES File from REC File** menu option. Specify the files **micro.rec** and **micro.qes** and click the **OK** button. ***EpiData*** will report that it has created a .QES file. Click the **OK** button

Open the **micro.qes** file in the editor and add the new variable definitions for **MFTOT**, **MFANY**, and **MFDENS**:

```
IDNO        ####
SR          #
MFRIC       ###
SSRIC       #.#
MFLIC       ###
SSLIC       #.#
MFTOT       ####
MFANY       #
MFDENS      ###.#
```

Save the file and close the editor window. Select the menu option **Tools -> Revise File** and specify the files **micro.qes** and **micro.rec** and click the **OK** button. ***EpiData*** will restructure the **micro.rec** file and create a backup file called **micro.old.rec** (note that a backup file is made when any process that changes the structure of a file or the data in the file is run). Click the **OK** button.

Examine the file **micro.rec** and verify that the new variables have been added.

Now that we have added the new variables we can specify the CHECK language instructions needed to ensure that they hold the required data.

Creating and transforming variables

Close all windows and open a new editor window. Type the following CHECK language commands:

```
RecodeBlock
  if (mflic = .) and (mfric <> .) then
    mftot = mfric
  endif
  if (mfric = .) and (mflic <> .) then
    mftot = mflic
  endif
  if (mflic <> .) and (mfric <> .) then
    mftot = mflic + mfric
  endif
  if mftot = 0 then
    mfany = 0
  endif
  if mftot > 0 then
   mfany = 1
  endif
  if (mftot <> .) and (sslic <> .) and (ssric <> .) then
    mfdens = mftot / (0.4955 * (sslic + ssric) - 0.076)
  endif
End
```

Save these commands in a file called **micro_recode.chk** and close the editor window.

Select the **Tools -> Recode Data** menu option and specify the files **micro.rec** and **micro_recode.chk**. Click the **OK** button. ***EpiData*** will respond with a message telling you how many records will be changed. Click to **OK** button to recode the data. ***EpiData*** reports that it has made the changes and created a backup of the original data file as **micro.old.rec**. Click the **OK** button.

Examine the file **micro.rec** and check the results of the recode operation. Check the data entry notes file for **micro.rec** to see a record of the recode operation.

The **RECODEBLOCK** is a little complicated. This is because we need to account for missing values in the data. If we did not check for missing values then ***EpiData*** would report a problem each time it could not calculate the result of a transformation due to missing values. In the **IF...THEN** statements, a single period (**.**) signifies *missing*.

The final calculation (i.e. for **MFDENS**) uses a standard formula to convert the diameter of a skin snip to its weight:

```
(weight in mg) = 0.4955 * (diameter in mm) - 0.076
```

File splitting

Close all windows. Select the **Document -> Count Records** menu option. Specify the file **micro.rec**. Specify the field **SR**. Click the **OK** button.

Examine the output. You should see that the file contains the data from the first and the final survey rounds. There are fewer records for **SR = 5** because of loss to follow up problems in the study. You can also use the **Count Records** function to examine the **IDNO** variable in **micro.rec**. If you do this you will see that most **IDNO**s appear twice (subjects with data at both survey rounds) and others appear only once (subjects with data at only one survey round).

Data analysis requires the data from each survey round to be stored on the same record so that (e.g.) a comparison of the density of microfilariae between the two survey rounds can be made. To do this, we must first split the **micro.rec** file into two files (one for each survey round) and then join the two file together side-by-side using the **IDNO** variable to identify which records belong with each other.

Close all open windows. Select the **Data -> Export -> EpiData** menu option. Specify the file **micro.rec** and click the **Open** button.

In the box marked **Export to:** type the filename **micro_sr1.rec**.

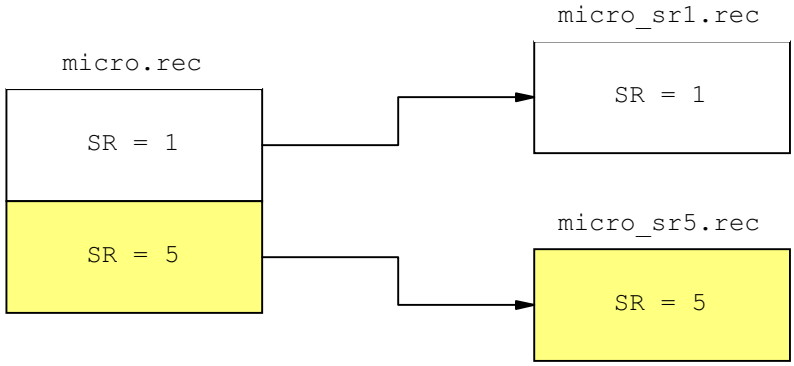
In the box marked **Filter:** type the expression **SR = 1**.

Make sure that the option **Use record filter** is checked.

Click the **OK** button. This will create a new file (**micro_sr1.rec**) that contains only the records for the first survey round. ***Epidata*** reports the results of the **Export** operation. Click the **OK** button.

Repeat this process to create a file called **micro_sr5.rec** that contains only the records for the final survey round (i.e. records were **SR = 5**).

We have split the file **micro.rec** into two separate files called **micro_sr1.rec** and **micro_sr5.rec**:



Examine both files and check that they contain the data you expect them to.

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 (**IDNO**).

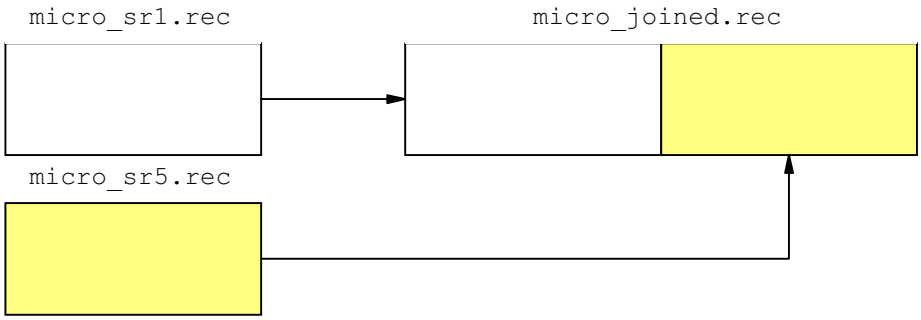
One problem that must be resolved before joining the two files side-by-side is that both files use the same variable names. We must first change the variable names in one of the files so we know which variable belongs to which survey round.

Close all open windows. Select the **Tools -> Rename Fields** menu option. Specify the file **micro_sr5.rec** and click the **Open** button. Type the new variable names next to the old variable names:

Field Name	Label	New field name
IDNO	IDNO	
SR	SR	
MFRIC	MFRIC	MFRIC5
SSRIC	SSRIC	SSRIC5
MFLIC	MFLIC	MFLIC5
SSLIC	SSLIC	SSLIC5
MFTOT	MFTOT	MFTOT5
MFANY	MFANY	MFANY5
MFDENS	MFDENS	MFDENS5

and click the **Save and close** button. ***EpiData*** applies the changes and makes a backup of the original file. Click the **OK** button.

Now we have renamed the variables in **micro_sr5.rec** we can join the two files together. Select the menu **Data -> Append / Merge** option and specify the files **micro_sr5.rec** and **micro_sr1.rec** in that order and click the **OK** button. In the box labelled **Resulting data file:** type the file name **micro_joined.rec**. Click on the **Merge** tab and specify the field **IDNO** as the key (i.e. linking) field. Click the **Merge** button. This creates a file called **micro_joined.rec** which contains the data from both files merged side-by-side based on the values of the **IDNO** variable:



Examine the **micro_joined.rec** file. Note that the variable names have changed but the labels in front of the variables have not been changed.

File merging

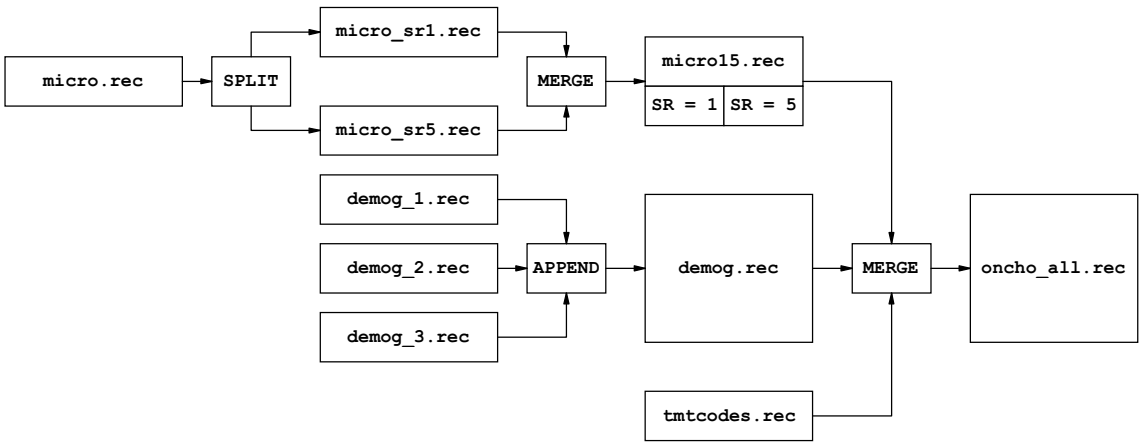
When you merge files together you may produce a file that contains much more data than is actually required. In this case we only need to keep the variables **IDNO**, **MFANY5**, **MFDENS5**, **MFANY**, **MFDENS**.

We can use the **Export** function to create files containing a subset of variables from the original file. Close all open windows. Select the **Data -> Export -> EpiData** menu option. Specify the file **micro_joined.rec** and click the **Open** button. In the box labelled **Export to:** type the file name **micro15.rec**. Make sure that only the variables **IDNO**, **MFANY5**, **MFDENS5**, **MFANY**, **MFDENS** are checked and click the **OK** button. Examine the file **micro15.rec** and check that it contains the required data.

We can now join this file (**micro15.rec**) to the demography file (**demog.rec**). Close all windows. Select the **Data -> Append / Merge** menu option and specify the files **micro15.rec** and **demog.rec** in that order and click the **OK** button. In the box labelled **Resulting data file:** type the file name **micro_demog.rec**. Click on the **Merge** tab and specify the field **IDNO** as the key (i.e. linking) field. Click the **Merge** button. This creates a file called **micro_demog.rec** which contains the data from both files merged side-by-side based on the values of the **IDNO** variable. Examine the **micro_demog.rec** file. Check that this file contains data from both files.

Repeat this process merging the files **micro_demog.rec** and **tmtcodes.rec** by **IDNO.** Call this file **oncho_all.rec**. Examine the **oncho_all.rec** file. Check that this file contains data from both files.

We have now completed a complex set of splits, appends (concatenations), and merges:



The data are almost ready for analysis.

If you are familiar with EpiInfo version 6, you will recognise the logic above, which could also have been accomplished using the **MERGE** module of EpiInfo version 6 or the **RELATE** command in the **ANALYSIS** module of EpiInfo version 6.

Tidying up

You may have noticed that the data entry form for the **oncho_all.rec** file is not particularly neat. This is because it contains data from several files and because we have excluded some variables. We can make the data entry form tidier by creating a new questionnaire (.QES) and revising the data file.

Close all open windows and select **the Tools -> QES File from REC File** menu option. Specify the files **oncho_all.rec** and **oncho_all.qes** and click the **OK** button. ***EpiData*** will report that it has created a .QES file.

Open the **oncho_all.qes** file in the editor and tidy the questionnaire (.QES) file:

```
{IDNO}        ####
{VILL}        #
{AGE}         ##
{SEX}         #
{TRIBE}       #
{HHNO}        ##
{REL}         #
{STAY}        ##
{OCC}         #
{DRUG}        ____
{MFANY}       #
{MFANY5}      #
{MFDENS}      ###.#
{MFDENS5}     ###.#
```

Note that we have dropped the **DATE** variable, re-ordered the variables, and specified variable names using curly brackets. It is **very important** that we do not change variable names. If we change variable names then data will be lost. Be very careful when editing questionnaire (.QES) files that will be used to restructure data files.

Save the file. Select the menu option **Tools -> Revise File** and specify the files **oncho_all.qes** and **oncho_all.rec** and click the **OK** button. ***EpiData*** will restructure the **oncho_all.rec** file. You will be warned of any loss of data. In this case we have dropped the **DATE** variable and we can choose the **Ignore** the warning. ***EpiData*** creates a backup file called **oncho_all.old.rec**. Click the **OK** button.

Examine the file **oncho_all.rec** and check that it holds the required data and the format is clean and tidy.

The **oncho_all.rec** file now contains the data required for a thorough analysis of the data from the onchocersiasis study. This file can be exported to you favourite analysis package or analysed directly using EpiInfo data analysis tools.

An exercise for you

You may have noticed that we have not merged the **`blood.rec`** file with the other example data files. This task is left as an exercise for you.

The file **`blood.rec`** is organised like the file **`micro.rec`** with data for subjects taken at survey rounds one and five in the same file. Use ***EpiData*** to:

1. Split **`blood.rec`** into two files (one for each survey round)

2. Rename the variables in one or both of the resulting files to indicate which survey round the data comes from.

3. Merge the two new data files so that the data for the two survey rounds are joined side-to-side on the same record.

4. Merge this combined file with **`oncho_all.rec`**.

5. Tidy up the final data file.

Remember to check your work at each stage.

Exporting data

Once you have entered, checked, validated, recoded, concatenated, and merged data you will probably need to perform some data analysis on the resulting dataset. This may require you to export your data into a format that can be read into your preferred data analysis package. ***EpiData*** use the same file format as EpiInfo version 6.xx (a popular MSDOS-based program widely used by field epidemiologists). Any program that can read EpiInfo version 6.xx data (.REC) files may also be used to analyse data entered using ***EpiData***. Available tools include the **`ANALYSIS`**, **`CSAMPLE`**, and **`EPINUT`** modules of EpiInfo version 6.xx, the **`ANALYSIS`** module of EpiInfo 2000, EpiMap version 2, and numerous add-in programs that perform logistic regression, conditional logistic regression, and survival analysis.

EpiData provides a set of functions that allow you to export data in a variety of common file formats (text, dBase III, Excel, Stata, SPSS, SAS, and ***EpiData***). All of these functions are available from the **`Data in/out -> Export`** menu.

Each export format has a set of options:

Range of records (by record number) to export

Skip deleted records

Export records that match a specified criteria

Export a subset of records

Export a subset of fields

As well as options specific to particular file formats:

Text

Add double quotes around the contents of text variables

Specify the character used to separate the contents

Stata

The format (i.e. Stata version) used for the exported file

The data file format for Stata includes information on data file label, variable labels and value labels. Export to SAS creates a SAS (.SAS) command file. Value labels are used in the SAS command file as value label definitions. Run (**`submit`**) the command file in SAS to load the data into SAS. Export to SPSS creates an SPSS syntax (.SPS) file and a text (.TXT) file containing the raw data. Value labels are used in the SPSS syntax file as value label definitions. Run the syntax file in SPSS to load the data into SPSS and use the SPSS **`save`** command to create an SPSS system file.

Exporting to ***EpiData*** is useful if you want to split files (as we did in the file splitting exercise) or if you want to create drop variables from a dataset (e.g. name and address data) prior to analysis.

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.

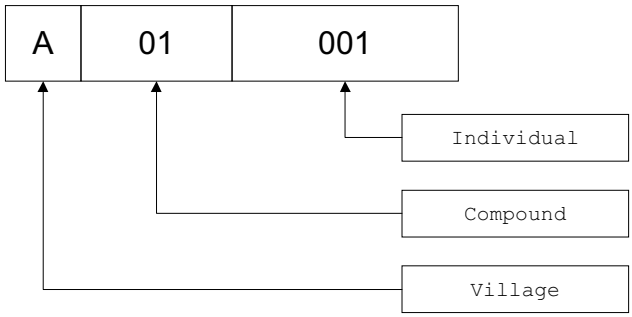
The number and nature of each of these tasks needs to be estimated beforehand, in order to quantify the resources needed to carry them out.

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), including paper forms.
- ❑ 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, ***EpiData*** can handle the data management requirements of large studies. The ANALYSIS module of EpiInfo v6.xx 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.

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 '98/ME. Other combinations of hardware (e.g. Apple Macintosh or PowerPC) or operating system (NT, MacOS, BeOS, Linux) 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. ***EpiData*** 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/yyyy	
	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 *microfilariae counts*	IDNO	Subject ID number		
	SR	Survey round	1 ... 5	
	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 ... 5	
	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		

Appendix 2
EpiData Options

EpiData options

EpiData allows you to change many of the ways in which it works by selecting the **`File -> Options`** menu option. Here is a list of the things that you can change:

Options Page	Item	Description	Recommended
Editor	Font	Font used in the text editor	A fixed pitch font such as `Courier New`
	Background	Background colour	White
	Tabs / indents	Number of spaces inserted by the ⇥ key	4
Show data form	Font	Font for forms	A fixed pitch font such as `Courier New`
	Background	Colour of forms	Grey
	Field colour	Background colour for data entry fields	White
	Highlight active field	Show the currently selected field in a different colour	On Bright yellow
	Field style	The way data-entry fields appear on the form	Flat with border
	Line height	Space between lines on the form	1½ lines
	Tabs / indents	Number of pixels inserted when using the `@` character in .QES files	40
Create data file	Field names	Use automatically generated field names based on the text before the field definition or use the first word on the line of the field definition	Automatic field names
	Letter case	Letter case for field names when exported	Upper-case
Documentation	Font	Font used for documentation	A fixed pitch font such as `Courier New`
	Background	Background colour	White
Advanced	First ID number	Starting value for **`<IDNUM>`** type fields	1
	Error messages	Show errors caused by CHECK commands	Set this on when developing CHECK code that includes a lot of calculations.
	Language	Language used for menus, messages, &c.	
File associations	File types	Specify which file types will open in ***EpiData*** when double-clicked in the Windows Explorer	All

Appendix 3
EpiData Field Types

Type	Example	Description
ID number	**<IDNUM>**	Automatic ID number. Incremented for each new record entered. Cannot be changed during data entry. The default first value of an ID number in a new data file is 1, but this can be changed using the **File -> Options -> Advanced** menu option.
Numeric	**###** **###.##**	Accepts entry of numbers.
Text	**___** **_____**	Accepts all characters. The number of underscore characters defines the length of the field.
Upper-case text	**<A>** **<A >**	Accepts all characters. Entries are converted to upper case. The length of the field is defined by the number of characters between the **<** and **>** symbols.
Boolean	**<Y>**	Accept only **Y**, **N**, **1**, **0** and space as legal entries. An entry of **1** is converted to **Y**. An entry of **0** is converted to **N**.
Date	**<dd/mm/yyyy>** **<mm/dd/yyyy>**	Accept valid dates in European (day/month/year) and American (month/day/year) formats.
Today's date	**<today-dmy>** **<today-mdy>**	Filled automatically with the current date. Cannot be edited. If a previously saved record containing a today's date field is modified and saved then the today's date field is updated to the current date.
Soundex	**<S>** **<S >**	Accept all characters. Only the last word entered is used to create the Soundex code. Soundex is a coding of words that can be used to anonymise (e.g.) the surnames of informants participating in a survey. Soundex coding is described in the ***EpiData*** help file.

Appendix 4
EpiData File Types

File extension	Type / function
.QES	**Q**u**ES**tionnaire - defines the variable names, types and lengths as well as the layout of the data entry form for a file.
.REC	**REC**ord - holds entered data.
.CHK	**Ch**ec**K** - holds the data checking and recoding rules for a file.
.NOT	**NOT**e - holds notes that you write during data entry using **File -> Data Entry Notes**.
.LOG	**LOG** files - files created by data documentation functions.
.EIX	***Epi****Data* **I**nde**X** Files - speed up searches for data.