Data collection and management guidelines for Foothills Model Forest research projects

1. Introduction

Research projects often involve the collection of a large volume of data. The data then have to be processed and analysed, with results and summaries being prepared for publication in some form. For this sequence to proceed smoothly, the project requires a well-defined system of data management.

Data management is or provides:

- serious and cost effective;
- guided by a preconceived plan;
- basic organization and maintenance at the record and file levels;
- a guarantee of the integrity and security of data;
- reliable, understandable data;
- good database design that promotes exploration and utilization of data.

Data management is *not*:

- simply entering data into a computer or a file;
- simply storing data in a notebook, computer, or file cabinet;
- working with or analyzing data with computers and software;
- using data without regard to their source, quality, and original purpose;
- keeping data to oneself or in a form that no one else can understand.

Perils of Data Mismanagement

Data can be lost through accident or disaster, corrupted through mishandling or neglect, rendered legally indefensible because of inadequate documentation and quality assurance, or found to be useless beyond a narrow purpose because of poor database design. Data can also be incorrect. Remember Murphy's Law (of Data): *without attention, whatever can go wrong with the data will go wrong,* and the problems will not be discovered until a large group of people depend on the data for quick decisions. The ultimate cost of poorly managed data can be astronomical, but most major problems can be avoided with good data management practices, procedures, and policies.

The main stages of the data management process in a research project are as follows:

1. determine study objectives;
2. design study and data to be collected;
3. data entry;

4. data verification;
5. data validation;
6. data archiving.

Data entry - is the initial set of operations where data from paper field forms or field notebooks are transcribed or typed into a computerized form (i.e., a database or spreadsheet). When data are gathered or stored digitally in the field (e.g., on a datalogger), data entry is the transfer of data (downloaded) to a file in an office computer where they can be further manipulated. Specific procedures for electronic data transfer are not discussed here, but the general procedures apply for those data, too.

Getting data from field projects into the computer seems a fairly simple process. However, without proper preparation and some simple guidelines, the quality and integrity of the data will be debatable. Three steps are needed in the data entry process to ensure that the resulting database is certifiably accurate.

Data Verification - Data verification immediately follows data entry and involves checking the accuracy of the computerized records against the original source, usually hard copy field records. Although the goal of data entry is 100% correct entries, it is rarely accomplished. The *verification* phase is the verification of the accuracy of all entries by comparison with the original source to identify and correct errors. When the computerized data are verified as accurately reflecting the original field data, the hard copies can be archived and most activities with the data can be done with the computer.

Data Validation - Although data may be correctly transcribed from the original field forms (data entry and verification), they may not be accurate or logical. For example, entries of stream pH of 25.0 or a temperature of 95°C in data files raise doubt about their accuracy, and such entries almost certainly are incorrect--whether or not they were properly transcribed from field forms. This process of reviewing computerized data for range and logic errors is *validation*. It can be done during data verification *only* if the operator is comprehensively knowledgeable about the data. More often, validation is a separate operation carried out by a project specialist *after* verification to identify generic and specific errors in particular data types. Corrections or deletions of logical or range errors in a data set also require notations in the original paper field records about how and why the data were changed. Modifications of the field data should be clear and concise but preserve the original data entries or notes (i.e., no erasing!).

2. Software for handling data

Database (DBMS) packages (e.g. Access) should be used for the entry and manipulation of data. From this package, data can easily be exported using an ODBC (open database connection) to most statistical programs for analysis purposes.

Database programs are very good at manipulating (e.g. sorting, selecting, counting) many records or rows. They are also able to handle hierarchical data structures. A database program should be used for both the entry and management of data.

3. Data entry

In designing a suitable system for data entry, consideration must be given to several aspects of the data. These are discussed in turn.

Understand the structure of the data

Few projects generate simple data; most have a complex structure with more than one flat file which must be linked in a clearly defined way. It is essential that both the flat file components and the links are fully specified, to ensure that the information meets the database requirements of completeness, integrity and minimum redundancy (or duplication) of information. Modern, relational database software makes this task fairly easy. Spreadsheet software does not - in fact it can make the task more difficult.

Identify the types of information being collected

Try to foresee the full range of different types of data that will be collected, e.g. plot data may consist of crop yield from all plants in the plot, number of plants with pods for harvest, total pod weight and number of dead plants. Build facilities in the data collection sheet for recording all such information. Often data will be collected from the same plot on a number of sampling occasions. Dates of such records must be kept, with space available on the recording sheet for notes about the plot or farm at that specific time. Such secondary information will be valuable at the data analysis stage to explain any curious behaviour of the data.

Specify the measurement units and precision

Ensure that the units of measurement used for all quantitative variables are documented in the field descriptions within the data tables. Changes in measuring instruments, or in field and research staff, or in methods of data collection, may bring about changes in measurement units. Consideration must be given at an early stage of the database design to allow for such changes to be incorporated into the data recording system.

Specify clearly the precision (number of decimal places) to which all measurements are to be recorded. The number of significant digits should match the real precision of the measuring instruments or recording devices.

3.1 Data entry and checking during collection and entry

We consider primarily the data that are collected in field books or survey sheets. First, we discuss the overall strategies that can be adopted for data keying and for checking, and then give separate guidelines on the two aspects.

3.2 Strategy for data entry and checking

When planning a strategy for data entry, clearly distinguish between the data entry / data checking / data management activities and that of data analysis. The ultimate aim should be a fully-documented archive of checked, correct, reliable data that can be subjected to scientific scrutiny without raising any doubts in the minds of subsequent researchers. Unfortunately, many worthwhile research projects do not achieve this.

The process of data entry will normally involve a skilled person who designs the system, while more junior staff (e.g. trained data entry operators or field staff) carry out the actual keying. Checking is done both at the time of keying and afterwards.

When planning the system, aim to make the data entry stage as simple as possible. For example, in a replicated experiment it should never be necessary to type variety names or long treatment codes for each plot. A single letter or number is usually sufficient. Then, either the data entry system can insert the full code, or the full names may be available in a separate, "look-up" file. Simplifying the keying process will speed the task, make it less tedious and hence also less error-prone.

The logical checking phase should be done by trained staff who understand the nature of the data. Usually this phase involves preliminary analyses and plotting of data.

In practice, the data entry and checking steps are usually designed at the same time. The way the data checking is undertaken will, however, depend on who is entering the data. Non-skilled staff should be expected to key exactly what they see on the data sheets or field books, and the logical checks (e.g. checks to rule out pregnant males, or minimum greater than maximum temperature) should be done by scientifically-trained staff after the (double) entry is complete. In that way, reasoned decisions can be made about what to do. If scientists are keying the data themselves, then the entry and full data checking can proceed together.

3.3 Guidelines for data entry

These guidelines may be summarised as "Do the data entry promptly, simply and completely."

- The data should be entered in their "raw" form - i.e. directly from the original recording sheets or fieldbooks - whenever possible. They are therefore entered in the same order that they are collected.

- All the data should be entered. Entering "just the important variables, so they can be analysed quickly," limits the possibilities for checking, which can make use of relationships between variables. Often when short-cuts are attempted, the full data entry has to re-start from the beginning, or (more usually) the remaining variables are never entered.

- No hand calculations should be done prior to data entry. Software can be used to transform data into the appropriate units for checking and analysis, e.g. grams per plot to kilograms per hectare, or to take averages of replicated readings, etc.

- One of the variables entered should give a unique record number (which can be used as a primary key). In field experiments this can be the plot or sub-plot number or a number automatically created by the database program (i.e. autonumber). One way to do this is to create a unique ID field for each table using the name of the table in the field name, so that the field type and purpose is easily identified. For example, in a table named T_Tree, we create a field name T_TreeID, which is a autonumber field that provides a unique identifying number for each record in the table.

- In field experiments, the position of each plot should be entered. This enables data (and residuals during analysis) to be tabulated, or plotted in their field positions - very useful for checking purposes. Where plots are regularly spaced, with no gaps, the position can be derived from the plot number. Otherwise, two extra columns are keyed giving the UTM co-ordinates.

- **The data should be entered promptly - i.e. as soon as possible after data collection.** For example, where measurements are made through the season, they should normally be entered as they are made. This speeds the whole process, because the data entry task at the end of the trial or survey is then not so large and daunting. It also helps the checking, because some checks can indicate unusually large changes from the previous value, and odd values can then be verified immediately. Feedback of any problems that are noticed to field data collectors can help maintain the data quality.

4. Guidelines for data checking

The objective is that the data to be analysed should be of as high a quality as possible. **Therefore the process of data checking begins at the data collection stage and continues until, and during, the analysis.**

4.1 Checks when the data are collected

- **Data should be collected and recorded carefully**. Consider what checks can be incorporated into the data collection routine. For example, the smallest and largest measurements could have a one-line note to verify - and perhaps explain - their exceptional nature. This will confirm that they were not written in error. The most efficient means of accurate data collection is the use of a data logger, where validation rules can be written for each field (for example, for a field that records dbh values, the validation rule on the field can be set to > 0 and < 90; when a value outside the range is entered (128, for example), a window pops up, saying that the value is outside the defined range and does the user want to change the value (perhaps the dbh was 12.8 and a decimal point was missed during the entry).

4.2 Checks while the data are being entered

- Always use software for data keying that has some facilities for data checking. Many of these can be built into the tables used in Access .

- Recognise that ignoring the data entry guidelines given above may be counter-productive for data checking. For example, changing the order of the data, transforming yields to kg/ha or calculating and entering only the means from duplicate readings can all lead to errors in calculation. It also makes it more difficult to check the computerised records against the original records.

- **Do not trust reading or visually comparing the computerised data with the original records. Though often used, it is not a reliable method of finding key-entry errors. Print out the data to check it.**

- Consider using double entry, where the second keying is done by a different person. This does not take much longer than visual comparison and is a far better form of validation. Modern data-entry software has facilities for a system of double-entry with immediate or subsequent comparison of values.

- Build in further checks that your software allows. The simplest are range (dbh > 0.5 but < 70 cm) checks, but other logical checks can also be used.

5. Guidelines for data verification after entry

Verification is done to ensure that all the data were entered and accurately transcribed.

1. Two verifiers - Two people are best for the verification of the data. This process involves two sets of paper. It is faster than data entry and thus generates a greater opportunity for confusion or losing one's place when working alone. Verification is best accomplished with one person reading the original data sheets (the reader) and the second the same data on the printout (the checker). In the remaining discussion a pair of people working together is assumed.

2. Comparison of data and noting differences on the printout - The reader reads the original data (field forms) out loud so that the checker can compare the original data with the data entries on the printout. The three common types of error are duplicated records (entered twice), missing records (inadvertently skipped during entry), and misspellings (wrong number or code). The checker controls the speed of the reader and halts the reader when a discrepancy is found. **When an error in the printout (=computerized records) is found, the correction to be made is noted in red on the printout and not on the original data sheets.**

After verifying the data from each field sheet, the reader should date and initial the original field form at the top (or where indicated), stating that verification was done. The reading and checking is continued until all the data sheets in a data set are compared. Thereafter, an original set of data sheets with completion marks (both entry and verification) and a set of printouts with needed corrections marked in red are available.

3. Correction of identified errors in the computer files - The application for data entry is used to correct the errors as indicated on the printout. **Each correction is made separately (i.e., no search and replace that may have unexpected consequences).** As each correction is made, the red mark on the printout should be checked with green. When all identified errors are corrected in the computer file, the printout is inspected again for any corrections that were missed (red without green check). Finally, the printout is initialed and dated at the top to indicate that all errors were corrected. The printout is saved with the original field form because it serves as direct evidence of the completion of entry and verification.

4. Simple summary analyses - Simple summary statistics of the entered data can be done on the computer. This is important because even when care is taken up to this point, a duplicate or omitted entry may have been overlooked. For example, the number of known constant elements, such as the number of sampling sites, plots per site, or dates per sample can be viewed. The same question can be posed in different ways; differences in the answer provide clues to errors. **The more checks that can be devised to test the completeness of the data, the greater is the confidence that the data are completely verified.** A handy tool in Access is the “Find Unmatched Records” query, which can be used to find records in one table that don’t have a matching record in another table. This has been useful for the FMF NDP program to identify sites in data tables that are not present in the table that holds information on the site visit (the rule being that any record in the database must have a visit id that relates to information on when the site was sampled).

5. The making and storing of backup of the data - A copy of the verified data file(s) is made and stored where instructed. Filenames must have space to accommodate version numbers. A second copy of the file(s) with appropriate documentation is given to the project and data managers. A copy of the original field forms is made. The printout is attached to the original field form and stored in the specified area. The copy of the original form is stored in a second location (i.e. in another building). The project manager must identify the file cabinets for storage.

6. Data validation strategies

Step-by-step instructions are not possible for data validation because each data set has unique measurement ranges, sampling precision, and accuracy. Nonetheless, validation is a critically important step in the certification of the data. Invalid data commonly consist of slightly misspelled species names or site codes, the wrong date, or out-of-range errors in parameters with well defined limits (e.g., elevation). But more interesting and often puzzling errors are detected as unreasonable metrics (e.g., stream temperature of 70°C) or impossible associations (e.g., a tree 245 feet in diameter and only 3 feet high). These types of erroneous data are called *logic errors* because using them produces illogical (and incorrect) results. The discovery of logic errors has direct, positive consequences for data quality and provides important feedback to the methods and data forms used in the field. Validation, therefore, cannot be ignored.

When possible, the data entry software should be programmed to do the initial validation. The simplest validation during data entry is range checking, such as ensuring that a user attempting to enter a dbh of 300 gets a warning and the opportunity to enter a correct value between 0.5 and 70. Not all fields, however, have appropriate ranges that are known in advance, so knowledge of what are reasonable data and a separate, interactive validation stage are important. The data entry application should also use look up tables for standardized text items where spelling errors can occur. For example, rather than typing in a species name (where a misspelling can generate a new species in the database), the name should be selected from a list of valid species and picked for automatic entry into the species field. Again, lists are not appropriate for all written fields but should be used when appropriate.

One of the most important tasks of rigorous validation is the return to the original data media (*and* the printout, 2nd copy, etc.) to make corrections and notations about the errors that were found and fixed in the digital files. Without annotating the original field forms, the digital and paper records are out of synch. If this is discovered without adequate documentation, *all* data are rendered suspect. This task is so important that it must be repeated more strongly:

When validation errors are found in the original data, the computer files and the original field records must be corrected. Only when the original forms are annotated with the same corrections is the correspondence between computerized files and field forms kept exact. Failure to correct the original field data forms creates havoc and doubt about the integrity of the data if it is later discovered that the field data and the computerized data do not match. Changes on the field forms must be clearly marked, so that the original data (i.e., the mistake) and the correction are legible (i.e., **the original data are not erased; e.g., the incorrect data can be circled or crossed out with a single line***).* The same correction notations are made on the other copies of the field forms.

The following generic suggestions can help develop a validation strategy for most data sets, and examples of validation strategies (and strange errors) are also provided.

1. Cataloguing the error types found in each data set - **When particular validation errors are found, it is important to make a list of them for that data set**. Notes on the error(s) should include a description, how detected, and how corrected. Simple, generic errors and more esoteric and cryptic errors must be documented. This list of errors is a valuable reference for the next validation session and ultimately for building formal validation procedures into the data entry process and other automated, post-entry error-checking routines.

2. Exploratory data analysis to look for outliers - Database, graphic, and statistical tools can be used for ad-hoc queries and displays of the data. Histograms, line plots, and basic statistics reveal possible logic and range errors. Such exploratory techniques identify obvious outliers. Some of these may appear unusual but prove to be quite valid after confirmation. **Noting correct but unusual values in documentation of the data set saves other users from repeating the same confirmation.**

3. Modification of field data forms to avoid common mistakes - With a catalog of validation errors and exploratory data results in hand, the field data forms as the source of the logic errors can be reevaluated. Often minor changes, small annotations, or adding check boxes to a field form remove ambiguity about what to enter on the form. In fact, any time the same type of validation errors occur repeatedly in different data sets, the field form --not the field crew-- is usually at fault. Repeated validation errors can also mean that protocol(s) or field training is faulty, which must be recognized and corrected.

6.1 Sample validation problems.

Below are four examples of logic errors discovered in data sets. The examples are informative. They demonstrate how errors can hide and give some generic and specific approaches to finding them. For data validation, the most useful adage is: Seek and ye' shall find. The most effective mechanism for avoiding tedious validation is to get the

right data into the computer in the first place, i.e., having a comprehensive set of standard operating procedures and data-collecting protocols for quality control, namely, clear field methodologies, a well trained and happy field staff, well organized field forms, and data entry applications with simple built-in validation. Last but not least, exploring the data looking for logic errors is also a good way to get to know the data intimately. Finding errors furthers understanding of what is represented.

1. Wrong date - A simple typo during data entry creates a logical set of data for a day, month, or year in which samples were never taken. This can become puzzling if the data are sorted by date--thus moving the entry away from its true neighbors. If sorting creates the appearance of missing data where a record should have been, the apparently appropriate correction may actually *create* duplicate records in the file rather than fix the errors--leaving the original problem unresolved. Even when left in the original order, however, date errors may go undetected because checkers can sometimes see what the readers say--especially when the month and day are the items of focus and an incorrect year digit is not examined. A summary analysis counting the total records of the data set will also be correct. A check of the number of dates or samples per year often reveals an erroneous year by revealing too many samples or a year that does not belong, whereas the rest of the data records reveal where the correction is needed. Identifying site code errors, etc. is a similar process for incorrect values not identified during verification.

2. Wild temperatures - Stream temperatures can show wild variations and yet be completely verifiable and valid. For example, some older data or the occasional spurious recent record may have been taken in Fahrenheit rather than in Celsius. The difference in the recorded number(s) is large. This is a protocol problem and not a data question, but if quality control during data collection was lax, these types of errors are often found only during data validation or (more annoyingly) data analysis. Routinely producing a box-plot or histogram of numerical data reveals drastic outliers, and when the original data forms are consulted, true outliers vs. errors in measurement scale or units and the correction of the files (i.e., convert the measurement to the appropriate units) become apparent.

3. Trees that shrink - Many vegetation monitoring programs include remeasuring trees in permanent plots every five years. In one survey, the project manager discovered that some of the remeasured trees were getting smaller--recent DBHs (diameters at breast height) were less than the original measurements 5 years earlier. Tree trunks of live trees do not get smaller. Some serious detective work revealed that the data were entered accurately (verifiable), but seemingly slight-to-moderate differences in the accuracy and exact methodology occurred between field crews. A search-and-compare program was written to parse the data and identify and scale the differences between trees, revealing the extent of the damaged data. Unfortunately, this problem could not be fixed by editing the data files. Rather, it revealed a previous protocol problem that resulted in data of poor quality and data that are useless for the original purpose.

7. Guidelines for data file editing

Data sets are rarely static; they often change from additions, corrections, and improvements from summary and analysis. These guidelines outline basic strategies for editing data files to update records, add new records, or change the record structure. The process requires (1) only changes that improve or update the data and maintain data integrity and (2) documentation of everything done to the data set.

All data must be validated as truthful and representative given the standard operating procedures of their collection. Proper preparation for and documentation of all changes during editing is important. **Practice careful version control during editing to ensure that changes are incremental and that roll-back to a previous editing session is possible until such time as the file being changed is certified as correct, up to date, and ready for archiving.**

7.1 Before editing

1. **Notes of edits. The editor should carefully note all changes to a data file** - These working notes may not become part of the permanent record of the data but are necessary for reconstructing the strategy of changing a file during editing. Whether these working notes are saved, a formal written summary and explanation should be created from each editing session, including a listing of all changes and when and by whom they were made. The editing report becomes part of the documentation permanently associated with the data file. A scratch copy of the *data edit report* in Figure 1 can ease note-taking. Also, detailed notes may later prove invaluable as a guide for specific editing in subsequent sessions.

2. **Work on only copies of files, one-at-a-time** - Work should never be performed on the only copy (i.e., the original) of a file. A working copy of the file is made and given a slightly different name. Choosing a shortened name for the working copy can facilitate loading, saving, and version control procedures. For example, when working with the file EA_Trees.DBF, the name during the editing session can be changed to EA_Trees*2*.DBF, where the '*2*' is the version number. If two or more files are edited simultaneously (i.e., if they are relational), use similarly coded version numbers to remind yourself that they are both at the same stage of editing.

3. Work on a subset of the data whenever possible - Corruption of data that does not need editing must be avoided. For example, if a field named TRANSECT must be adjusted for only 1 year in a multi-year file, the records of that one year are best isolated before editing. This can be done by splitting the original file into two parts, editing the one part, and then recombining the parts. (The entire file and the separate pieces are tracked in one's notes.) Another approach is the use of a query tool in a database program to allow access only to the desired records.

7.2 Editing strategy

1. Working file information - **The names of the file(s) to be worked on, the initial date(s), size(s) and the number(s) of records are recorded in the notes**. If a copy of the file to work on is renamed, the name of the file that includes a temporary version number (e.g., EA_TREES.DBF is the original EA_TREES2.DBF file.) is also recorded.

2. List and sequence of the changes - **Before beginning a data editing session, what is about to be accomplished is written down precisely in as much detail as practical. If several steps are needed to fix a file, they should be written down separately and examined carefully *before any editing begins* to evaluate whether any one change may adversely affect later steps. This examination may also reveal how to arrange a cascade of changes to be most efficient. If order of actions is important, an explanation is entered in the notes before beginning.**

A common example of poor planning is using global search and replace functions indiscriminately, such as wanting to increment a few numbers by one. One may start by changing all 1s to 2s and quickly discover one cannot distinguish the original 2s any more and all the 1s in compound numbers and other codes were also changed to 2s--not what was intended. For this example, one would record beforehand the proper strategy of replacing the highest number first and then work downward and restrict edit to the field(s) of interest.

3. Defining the tools for editing - The computer program(s) or file editor(s) for making changes to a file are listed in the notes. This needs to be stated only once for the editing session, but if some unusual feat of magic to accomplish a difficult task is performed or some advanced feature is tried for the first time, the steps should be recorded in detail.

7.3 During editing

1. Making notes - If you have properly planned and documented your strategy for the editing session, there is little left but to carry it out. Remember to record each step in the edit process, including names and versions of files**--especially noting any changes in the number of records as a result of an edit**. It is best to number the editing steps in your notes, and edit version tracking can be facilitated by using the step number as the temporary version number as each step is completed (e.g., S2.DBF is the *result* of step 2 in your notes of the editing process).

2. Frequent saving of work - Files must be saved frequently. If a small series of relatively simple updates are made, the editor may wish to complete all of them before saving because the repetition of the task would be easy. At that point, the editing may be completed and the file renamed as a new version. However, if a step in the editing process requires numerous changes to records, the file is best saved in 5 minute intervals or even after each record is completed. In the latter case, running notes are kept of the last edited record edited before each last save.

Microsoft Access automatically saves the record you are adding or editing as soon as you move the insertion point to a different record, or close the form or datasheet you are working on. So each change made to the database is automatically saved. To save the database as different versions that reflect the editting steps, go to Windows explorer and copy the file, renaming it to reflect the step that is going to be made to it (i.e. S3.mdb would be the datafile on which you would perform step 3). Then S3.mdb can be copied and renamed to S4.mdb and step 4 can then be performed on S4.mdb.

3. **Tracking of versions of the edited file - All intermediate, numbered versions are kept until the full edit is complete.**

7.4 After editing

1. Backup of the new file and the version series - Copies of the edited file and its intermediate versions are made immediately to diskette, even if only for temporary storage. This prevents a power outage or hard disk crash from nullifying editing efforts.

2. Review of pre-edit notes - The desired changes are checked against the notes made during editing to double-check that all changes were indeed made.

3. Formal documentation of the file edit. - A formal statement of what was done to the file is made and includes the operator's name, the date, the file(s) that were changed, a concise list of the changes and the reasons for the changes, and the version series used during the edit. Each record that was changed does not have to be listed if the change was more or less global. But records that were separately adjusted for a particular reason (i.e., to correct an error) should be identified individually. The documents that accompany the file--including edit summaries--should detail the entire history of the file, no matter how minor the change was (for example, changing a single date in a 20,000 record file still needs an explanation).

4. Archiving of the edited file and associated documentation - When the editing and documentation are complete, the guidelines for formal archiving of the file and its associated documentation are followed. Data set documentation and archiving are described below.

5. Filing the edit session report - A paper copy of the formal edit session documentation should be placed in a folder associated with that file or project.

6. Printing the file, if required - Optionally, the final version of the edited file is printed if that has been the standard procedure for the project. Archiving the file(s) and documentation. A copy of the final version of the file, copies of the edit information, and copies of all other documentation associated with the file **are also archived in safe, off-site storage.**

7. **Only one copy of the file is to be used for analysis and distribution - Hopefully the file is stored on a network drive, which offers access to multiple users and is backed up frequently.**

8. Updating master record(s) listing current data files - Any master records that note the current version of data sets should be updated. This may include a notebook to which all users have access to check on the current status of their data or a computerized database (i.e., the data set catalog) used for the same purpose. Any place where the file version and date are recorded must be immediately updated with the new information.

Figure 1. Sample Data Edit Report.

Name: Steve Tessler
Date: October 7, 1993 (completion date)
File(s) Edited: AQINS.DBF from Aquatic Macroinvertebrate LTEMS project
Reason for Edit: Taxonomic code changes required to the TAXA field as per correspondence with Steve Hiner/Reese Voshell @ VPI. Changes needed and taxa-change categories are fully outlined in the correspondence and Change Sheets created for this edit, and include changing certain generic ID's to species, eliminating terrestrials and impossible taxa (for Virginia) in the data, and regrouping *Baetis* and *Pseudocloeon* as *Baetis* complex.
Program(s) Used: FoxPro 2.5 and DBBrowse for .DBF files; TSE pre-release 1.0 and QEdit 2.15 (both Semware) for ASCII files.
Original File Information: AQINS.DBF, Version *n*, 13,779 records, contains data from 1986 through 1992. Full error-checking pending.
Final File Information: AQINS.DBF is now temporary version 7 of *this* edit, dated 10/07/93, time 10:15:23.03a, with 13,731 records.
Editing Details:
1. Created AQINS.1 as a tab and delimited ASCII version of the original .DBF. Files with a number extension are ASCII.
2. Did global delete of terrestrials and impossibles. 33 records removed; new number of records = 13,746. Saved as AQINS.2.
3. Found error while previewing for *Peltoperla* changes. 15 lines were duplicates in 3L301 2nd Qtr 1988; confirmed by checking paper records; they were deleted. Saved as AQINS.3, #Rec now = 13,731.
4. Made unusual, one-time-only changes as per Change Sheets. 19 records were changed; # rec still 13,731. Saved as AQINS.4.
5. Changed all UNID to straight taxon code (i.e., removed X's from the code). 138 X's removed, no rec# change. Saved as AQINS.5.
6. Changed genus to species for monotypic genera and *Perlesta* to *P. placida* group as per Change Sheets. 395 changes made to 15 taxa. Saved as AQINS.6, still with 13,731 records.
7. Changed names up one taxon level globally as per Change Sheets. 358 changes made, still 13,731 records. Saved as AQINS.7; loaded into AQINS7.DBF for checking and sorting. Re-saved as AQINS.DBF, Version *n+1*.

8. Data set documentation and archiving

Numerous references to data set documentation and archiving are given throughout these guidelines. This section addresses the overall strategies and concerns for documenting and archiving data sets. Although some of the documentation procedures and metadata standards are currently evolving, good preparation and comprehensive documentation of data sets ease the transition to a more rigorous system.

Figure 2. Sample Data set Documentation.

Resource Study/Data Collection Plan
- Project Title
- Problem Statement
- Description of Study/Action
- Description of Data (types, ranges, etc.)
- Inception/Duration Dates
- Stipulated standard operating procedures (accuracy, precision, etc.)
- Project Originator/Manager
- Long-Term Data Manager
- Project Data Management Plan

Implementation
- Modifications to standard operating procedures
- Data Entry, Verification, and Validation
- Considerations/Modifications
- Data Edit Report(s)
- Data set Version(s)

Archiving
- Compilation of all Analog and Digital Documentation
- Location of Master Data and backups
- Data Dissemination Contact(s)
- Data Dissemination Plan and Records
- Data Distribution Arrangements
- Access Restrictions
- Metadata Document

8.1 Documentation

Documentation of a data set should begin at the conceptual stage for which data will be collected. Notes about the purpose of the study, need for the data, monitoring goals,

and so on are important metadata considerations. Good sampling design, standard operating procedures, comprehensive plans for error checking, validation, archiving, application, and dissemination should be recorded.

8.2 Archiving

Formal archiving of master data sets and associated documentation is done to protect and maintain the physical and informational integrity of the data through time. Because archiving includes the physical storage of master data sets at separate locations or buildings, each project must evaluate and implement its own archiving procedures and include them in the data management plan.

The data and programs from a research project must be archived in such a way that they are safe and can be accessed by a subsequent user. The media used for the archive might be diskettes, tapes, or CDs - similar to that used for back-ups. Although the copying of data to the archive comes at the end of the project, the way the information will be transferred to the archive should be planned from the outset. Careful planning will be helpful throughout the project, because it helps to promote a **consistent directory structure and naming convention for computer files**, and also encourages the recording of all steps in the project.

The archive is more than a permanent storage place for the files used for the analysis. It must give access to all the information from the experiment or project. During the operational phase of a project, the information about the research is partly in the computer, partly on paper and other media (such as photographs) and partly in the minds of the research team. The archive need not all be computerised, but it must include all the relevant, non-ephemeral information that is in the minds of the research team. Where data cannot be archived electronically, the sources of information should still be recorded in the archive.

In the absence of a proper archiving scheme, the usual outcome is that the researchers leave, carrying with them the only copy of their part of the data, and hoping that the analysis and write-up will be continued later. Eventually the hope dwindles and the datasets become effectively lost to further research. To avoid this outcome, we believe that (i) at least one full copy of the archive should be left locally, and (ii) the final report should detail the structure of the archive and the steps taken to ensure its safekeeping.

9. Recommendations on data organisation

Here is an example of what NOT to do:

- We start with the raw data in an Access database file, called *exp971.mdb*.

- We transfer the data to an Excel spreadsheet file, transform some variables and save them into *exp972.xls*.

- A mistake in the data is noticed and is corrected in the spreadsheet file.

- We next transfer some of the values to a statistics package and save them in a new file *exp97.gsh*.

- A second mistake is corrected in this third file.

There are now three different "versions" of the data.

Unless we are extremely meticulous in keeping a full record of the changes made at each stage in the analysis, it is impossible to know which version correspond to the original. The data integrity is thus compromised, and it is no longer clear which file should be stored in the data archive.

This scenario is a familiar nightmare when, in the final year of a research project, a combined analysis is attempted of a set of experiments done over several years. One solution is as follows. **ALL corrections should be made to the values in the initial file, i.e. in *exp971.mdb* in the example above.** However when the transformations are made the first time, the commands to perform the transformation are saved.

The fact that the commands corresponding to these transformations have been kept, means that this command file can be run again to derive the second file, i.e. *exp972.xls*. The commands to derive *exp97.gsh* in the statistics package are then run and the analysis continues. The audit trail includes the names of the two command files and the archive contains the original data file, plus these two command files.

The potential nightmare scenario outlined above was for a simple situation, where the raw data are in a single rectangle or flat file. Adherence to standard database principles becomes even more important when the data are in multiple files. For example, in a survey, there may be information on villages in one data file and farmers within the villages in another. With the data in a database package, these data are can be merged based on the values in a linking field, which here is the village code.

For example:

Village Data

Village code	Name	Population	Chief
1	Ntsi	52	Kgatala
2	Matela	84	Molapo
3	Peete	62	Dinare

Farmer Data

Village Code	Farmer Name	Date of birth
1	Pitso,M	14/03/42
1	Jama,H	21/09/49
2	Phiri,J	11/02/45
3	Sello,C	04/10/39
3	Letsie,J	15/04/44

The merged data appears as:

Village Code	Name	Population	Chief	Farmer Name	Date of Birth
1	Ntsi	52	Kgatala	Pitso,M	14/03/42
1	Ntsi	52	Kgatala	Jama,H	21/09/49
2	Matela	84	Molapo	Phiri,J	11/02/45
3	Peete	62	Dinare	Sello,C	04/10/39
3	Peete	62	Dinare	Letsie,J	15/04/44

In a database package, the farmer and village data are stored separately, but may be viewed in one table, as above. In this case although the data for each village appear to be repeated when merged, they are not repeated in the database. Changing the village information while in the database requires just one change. For example if the population of village 1 were to increase to 55 we would only have to make that change once in the database, not twice. However, this is only the case while the data remain in the database. If the merged data are exported to a different package for analysis, the village information in this exported dataset is repeated. If you make changes to the population values at that stage you will have to make the same change for as many records as there are farmers in the village. It is easy to make mistakes in a large file and data integrity is compromised.

The solution outlined above is to keep a definitive version of the database, to which all changes are made, and from which sub-sets of the data are taken for analysis. Although organising this may take longer initially, it is safer and will save time in the long term.

9.1 Data Documentation

List of files -All file names, dates, sizes, and directories and subdirectories that are to be transferred are listed. Brief, informative descriptions of each item are written.

File relations - Relations between data files are described. For example, if the data are relational and fully normalized, the primary and foreign keys in individual files used for linkages are described. A text diagram of the relations is included whenever possible. For each data file, a table of the data file structure is included that includes:

- the total number of records
- the size of each record
- the number of fields per record
- the names of fields, in record order
- field type, size, etc.
- a description of the field
- codes for missing values.

Description of the full data set, including limitations - The data set is described in a paragraph and a disclaimer if necessary. **Separate comments must be made about each file. Known problems with the data set--such as different protocols followed over different years, changes in equipment, detection limits or resolution, etc. must be explained.** A sample data format document is shown below (Figure 3).

Figure 3. Sample Data set Format Documentation

Contact Information:
Name, Position, Address, Phone, etc.
Data set Title: LTEMS Aquatic Macroinvertebrates Database
(see accompanying Data set catalog Report)
AQINS.HDR -- description of AQINS.TXT (ASCII form of AQINS.DBF)
AQINS.DBF and AQINS.TXT contain 13,731 records from 8/7/86 to 9/28/92
This is a brief header definition and description of the AQINS.TXT file containing the fixed-column ASCII version of AQINS.DBF. There are 8 fields (variables) in the file we are sending for each record (row or line). There are a total of 13,731 records in the file. Each record is based on the actual presence of a single taxon in a specific sample. The following notes detail specifics of the AQINS.TXT file and describe how it differs from the original dBASE file.
Description of Fields in the original AQINS.DBF dBASE file (see LTEMS manual).
NAME TYPE LEN DESCRIPTION
1. SITE Character 5 LTEMS Site code
2. QUARTER Character 1 Annual quarter (1-4)
3. SMPDATE Date 8 Sample Date
4. ASMPL Character 1 Sample # for a method
5. SMPMETH Character 3 Sample Method
6. TAXA Character 7 Taxon code
7. INSCNT Character 4 Taxon count in the sample
8. METERCNT Character 5 Calculated #/sq.meter
9. WEIGHT Character 6 Individual weight
10. STAGE Character 1 Individual stage
11. FUNCGRP Character 2 Taxon functional group
12. REARED Logical (T/F) 1 Individual reared or not
First, the fields METERCNT, WEIGHT, STAGE, and REARED were dropped from the ASCII version of the file. These were either virtually always empty or redundant with other data. They are still available in the .DBF file, or from me by request. Second, all empty or null fields have been filled with an X to represent missing data. Any analytical processing of the data should properly identify the X as missing data.
For quantitative methods (SMPMETH = PIB or SUR) the actual count of individuals of that taxon in the sample is given in both the TXT and .DBF files; so in these records a 1 represents only one individual in the sample. Most of the counts for the other sampling method types are null (SMPMETH = QUD, AER or RED); the mere existence of the record in the file indicates the presence of a specific taxon in that sample.

Figure 3. Continued

Other fields in the data set were empty and are now filled with X -- reflecting the absence of corresponding data for that record. FUNCGRP is not available for some taxa, so its analytical value at all is highly questionable. A single X has been placed in each such field that is truly null.
The AQINS.TXT file is sorted (ascending) by the following fields, in this order: SMPDATE, SITE, ASMPL (which follows SMPMETH), and TAXA. This sort gives a taxa-by-sample#-by -site-by-date which was useful for checking items in the file. The data occupy the first 45 characters in the text file, including double spaces between actual data columns. A short description of data field identity, column location, maximum length, and some notes on the data are given below. The range of Columns occupied for each field should be consulted for importing the data using a fixed column format into other programs.
Description of fields format in the AQINS.TXT file S.Tessler 7/93 SNP
Name Columns MaxLngth Notes
1. SITE 1-5 5 alpha-numeric LTEM site code
2. QUARTER 8 1 range is 1 to 4
3. SMPDATE 11-18 8 date format is numeric YYYYMMDD
4. ASMPL 21 1 range is 1 to 6 per SMPMETH
5. SMPMETH 24-26 3 PIB, SUR, QUD, AER, RED
6. TAXA 29-35 7 codes refer to TAXADICT.DBF
7. INSCNT 38-41 4 for PIB and SUR 1 is real count
8. FUNCGRP 44-45 2 see LTEMS manual; some null (X)