Chapter

2

Data Management, Analysis Tools, and Analysis Mechanics

This chapter explores different tools and techniques for handling data for research purposes. This chapter assumes that a research problem statement has been formulated, research hypotheses have been stated, data collection planning has been conducted, and data have been collected from various sources (see Volume I for information and details on these phases of research). This chapter discusses how to combine and manage data streams, and how to use data management tools to produce analytical results that are error free and reproducible, once useful data have been obtained to accomplish the overall research goals and objectives.

Purpose of Data Management

Proper data handling and management is crucial to the success and reproducibility of a statistical analysis. Selection of the appropriate tools and efficient use of these tools can save the researcher numerous hours, and allow other researchers to leverage the products of their work. In addition, as the size of databases in transportation continue to grow, it is becoming increasingly important to invest resources into the management of these data.

There are a number of ancillary steps that need to be performed both before and after statistical analysis of data. For example, a database composed of different data streams needs to be matched and integrated into a single database for analysis. In addition, in some cases data must be transformed into the preferred electronic format for a variety of statistical packages. Sometimes, data obtained from “the field” must be cleaned and debugged for input and measurement errors, and reformatted.

The following sections discuss considerations for developing an overall data collection, handling, and management plan, and tools necessary for successful implementation of that plan.

The Data Collection, Handling, and Management Plan

The data collection, handling, and management plan plays an important role within a research project. The plan provides a roadmap documenting the flow of data through the sequential phases of collection, storage, cleaning, reduction, analysis, and finally to archiving. Further, the management plan documents the relationships between all of the software tools and programs necessary to guide the data through this research life cycle.

The data handling and management plan needs to be developed before a research project begins. The plan, however, can evolve as the researcher learns more about the data, and as new avenues of data exploration are revealed.

Considerations

The data collection, handling, and management plan addresses three major areas of concern*: Data Input, Storage, Retrieval*, *Preparation; Analysis Techniques and Tools*; *and Analysis Mechanics*. These concerns are not independent, and have synergistic impacts on the plan.

Provided below is a list of questions that must be considered when formulating a data collection, handling, and management plan. Although the questions are organized into three major categories, many questions raised will affect decisions made in two or more categories.

Data Input, Storage, Retrieval, and Preparation

Are the data "clean?" The data input process oftentimes introduces typos, miscodes, and errors into the data. (These errors are distinctly different from random or measurement errors introduced in the measurement process). Different storage strategies support differing levels of data editing. *[Do the data need to be edited? Do edits need to be tracked?]*

Are the data static, or will updates be available through the lifetime of the analysis? Many data streams are updated periodically. For example, transportation safety data can be updated on an annual basis. Pavement data may be updated on a route-by-route basis as field observations are recorded. *[Will new data constantly be added? Will new data be in the same format? Will new fields be added?]*

Are the data obtained from a variety of sources or from a single source? As data are drawn from a greater number of sources, the need for transforming the data to a common format becomes more critical and challenging. Further, one must always consider whether the different sources use the same definitions for common variables. For example, does one source define "average delay" in the same manner as another? If not, can a simple transformation between definitions be established?

How much data will be managed and stored? Different storage strategies support varying record sizes and large numbers of records. *[Will storage space be a problem? Will access time be a problem?]*

Will all data be used in the analysis, or will subsets of the data be analyzed? To speed analysis, the researcher will sometimes want to work with a subset of fields rather than all database fields within a record at once. In other cases, only a subset of records will be analyzed. For example, a research may investigate traffic flow and speed relationships for workdays only. *[Are sophisticated query and data sub-setting features needed?]*

Do records in the database share common, duplicated information? Duplicative information wastes storage space, and in some cases creates database problems. In a safety analysis, for instance, geometric information may be duplicated if accidents occur at

the same location. Techniques to reduce duplication of information (and thus reduce the overall size of the database) are available. *[Are tables needed?]*

Analysis Techniques and Tools

Is the analysis well defined, or is it more exploratory in nature? For example, are you simply testing statistical differences, or are you looking for unknown relationships? Exploratory analysis and data familiarity requires graphical tools to help visualize relationships between variables, and requires the researcher to guide the exploration. *[Are graphical capabilities needed?]*

Is the technique well defined, or is it experimental? Well-defined analysis can be performed by most statistical packages. Specialized software packages and even custom applications developed using programming languages may be required to perform less well known statistical tests or comparisons. *[Does the software perform the necessary test or compute the necessary statistics, or will it need to be developed?]*

Analysis Mechanics

Is this a one-time analysis or will this analysis be repeated? How often will the analysis be repeated? The more times an analysis is repeated, the more important it becomes to automate the analysis process. This becomes even more important if the analysis requires that manual transformations be applied to the data. All statistical experiments should be reproducible, not only by other researchers, but by the original researcher also. *[Automation: Is a macro or script (small program code that performs a repetitive process) need to be created? Is a custom program needed?]*

Will the analysis be repeated on different data? Will the data be provided from the new site or device in the same format as the original experiment? *[Is a script needed? Is a monolithic application, or a series of filter/transformations needed?]*

Will multiple researchers access the data, or will only one researcher be performing the analysis? As more people become involved in the analysis, documentation of the analysis procedure becomes more important. A tradeoff between level of training and level of automation is revealed as more people become involved in the analysis process. *[Will data security be an issue? Will record locking be required? Are varying levels of secure access necessary?]*

How long does the analysis take? A computational procedure may require several hours or days to complete. What happens if the procedure is interrupted? Can it be restarted from where it was interrupted, or must the analysis be restarted from the beginning?

Data storage: databases and data warehouses

The following sections discuss the range of data storage alternatives. Like programming languages, these database solutions differ not so much by what they make possible, but in what they make efficient. Of course, as databases grow in size, efficiency becomes increasingly important.

The researcher may use one or more of the following solutions in the overall management and handling of data. The key is to recognize that there is not a one-size-fits-all solution, and so the best solution for the particular task at hand must be carefully selected.

Small-scale database solutions

Text files and spreadsheets are categorized as "small" databases.

In text files, often called "flat files," all records related to a particular analysis are stored in consecutive lines in the file. Text files are the "least common denominator" of files and are generally used when transferring data from one statistical application to another.

From the research perspective, small-scale databases offer several advantages. First, they are simple to understand. All records in the database are the same, and all the necessary information is contained in each record (i.e., the researcher need not look up additional information in a separate table.) Second, they are simple to use. In general, the researcher will use all the information in the database, rather then selecting a portion of the data. Third, flat-file or spreadsheet databases are simple to update, edit, change, or append.

On the other hand, small-scale databases offer several disadvantages. First, they are not designed for "querying" or locating specific records (e.g., all the accidents from a specific site.) This becomes a greater problem as the number of records in the database increases, and queries become important. Second, duplicate information across records can increase storage requirements, and result in a propagation of errors through the analysis. For example, suppose that records in an accident database contained site-specific data. All accidents at a specific site would contain duplicate information. If the researcher found it necessary to update the site-data at this site, all records in the database would have to be updated simultaneously. Third, only one person (or application) can use the database at a time. When edits or changes are being made, the entire database is locked, and no one else can be provided access.

For many research projects, the small-scale solution (e.g., flat-file or spreadsheet) is appropriate. Other solutions should be examined when multiple users require access to the data, when the amount of data is large, or when the data is constantly being modified, queried, or appended.

Medium-scale database solutions

Medium-scale data solutions include the desktop relational database management systems (RDBMS). These systems store data in the form of related tables. Relational databases are powerful because they require few assumptions about how data is related or how it will be extracted from the database. Consequently, the same database can be viewed in many different ways. An important feature of relational systems is that a single database can be distributed across several tables. This differs from flat-file databases, in which each database is contained in a single table. The internal organization of data can affect how quickly and flexibly the analyst can extract information.

Requests for information from a relational database are made in the form of a query, or stylized question. For example, the query

SELECT ALL WHERE NAME = "SMITH" AND AGE > 35

requests all records in which the NAME field is SMITH and the AGE field is greater than 35. The set of rules for constructing queries is known as a query language. Different DBMSs support different query languages, although there is a semi-standardized query language called SQL (structured query language). Sophisticated languages for managing database systems are called fourth-generation languages, or 4GLs for short.

The information contained in a database can be presented in a variety of formats. Most DBMSs include a report writer program that enables you to output data in the form of a report. Many DBMSs also include a graphics component that enables you to output information in the form of graphs and charts.

Medium-scale solutions tend to focus on the single-user. Performance of these systems tends to degrade rapidly as the number of users accessing the data at any given time increases.

Examples of medium-scale database management solutions include FoxPro, Paradox, DBase, and Microsoft Access.

Medium-scale solutions are appropriate for a large number of research applications. Other solutions should be examined when the size of the database exceeds around 500 megabytes (MB), when the data is constantly updated (e.g., real-time data collection), or when the number of concurrent users and/or analysts is large.

Large-scale database solutions

While large-scale databases offer many of the same features as medium-scale databases, they differ in several important aspects. First, they are generally designed to handle large numbers of records (e.g., millions and greater.) As such, the overhead of such a system is significantly greater than that of a medium-scale database, but this is offset by the speed at which the data can be queried. Second, large-scale databases include some level of transaction logging. This feature tracks changes to the database, and allows the system manager to “rollback” or “undo” modifications if they are found to be in error. Third, large-scale databases are designed to provide multiple users with concurrent access to the data. It is common to have several hundred users concurrently querying a large-scale database.

Large-scale databases tend to require a significant investment in the computer hardware, and generally need expertise and resources available to manage, upgrade, and maintain the system.

Large-scale databases include MS SQL Server, Oracle, Informix, Interbase, and Sybase.

Large-scale solutions are appropriate for special research applications. Traffic management systems for a city, county, or state, for example, can manage their huge databases with large-scale database management solutions.

Working with data warehouses

A data warehouse differs from a regular database in several aspects: 1) A data warehouse is read-only; 2) A data warehouse contains data from disparate sources that

don't easily share data; and 3) A data warehouse allows different applications and software to make use of the same information.

Data warehouses also contain "metadata." Metadata refers to data about other data. Because a data warehouse contains data from disparate sources, the warehouse must also contain information about the source itself (e.g., who owns it, where it is located, how the data are collected, how often the data are updated, etc.) Prior to the use of data warehouses, this information was typically stored in a manual.

As more state departments of transportation implement data warehouses, knowledge of their content and access will be important to researchers.

From a data management perspective, the researcher would either access the data warehouse directly, or create a new, local database (e.g., MS Access based), downloading content from the data warehouse as needed.

Useful Data Handing Tools

Two tools are critical to the efficient and reproducible handling of statistical data: *make* and *perl*. Taken together, *perl* and *make* offer an unbeatable combination for handling data and managing large, complex statistical analyses.

Make

Make is one of several utilities originally developed for Unix programmers to aid in the management of large software programming projects. The usefulness of *make* should not be understated–it is an invaluable tool for the handling, management, and analysis of statistical data. *Make* is available for a variety of computer operating systems and will run on small laptops, desktops, workstations, and servers.

Make builds on the fact that every computer file has a time stamp associated with the last time it was changed, edited, or updated. *Make* uses this time stamp to decide which files are out of date with respect to each other. If *make* determines that two or more files are out of date with respect to each other, make will automatically run the necessary programs to get the files synchronized with each other.

Example: In its simplest form, the statistical analysis process includes the following steps: "Load Data," "Crunch Numbers," then "Generate output." One can think of these steps as a "transformation" of "inputs" into "outputs."

The "Crunch Numbers" activity needs to occur whenever the data is changed (e.g. new data arrives, or existing data is edited), or the "Crunch Numbers" program itself is modified (e.g., a new statistical test is added, or a new statistic is desired.)

The researcher often writes a "batch" file containing the necessary instructions to perform the desired transformation, and executes these commands whenever necessary.

Make allows the researcher to store the "batch" commands within a "makefile." Then, the utility will execute these commands whenever the time stamp of the input or the program is more recent than the time stamp of the output file.

Large statistical analyses often require that: 1) data from several sources be combined; 2)

several programs be executed on these data in a specific sequence with the outputs of one program becoming the inputs to the next; and 3) reports be generated during the process.

Large-scale analyses do not simply "appear." They are performed incrementally, with the researcher focusing on one small aspect of the problem, and when completed, turning her attention to the next aspect of the analysis. In many cases, the results of one analysis (or data transformation) are used as inputs to the next analysis.

Make allows the researcher to encapsulate the manipulations employed during each step of the analysis into a single set of computer instructions. This helps ensure that other researchers on other computers can repeat the overall analysis in the future.

Further, *make* helps the researcher operate more efficiently. For example, suppose that one step of a complex analysis takes several hours to complete, and suppose that the outputs of this analysis are used by subsequent programs for additional analysis. If the researcher simply used a "batch" file to store the necessary instructions, the long analysis step would have to be performed each time the researcher made a change to one of the smaller, subsequent programs. By using *make*, only the necessary programs would be executed, eliminating the need to constantly run the long analysis each time.

Perl

Perl is another of the Unix utilities that can save the researcher hours of time when handling statistical data, and performing statistical analysis. Like *make*, *perl* is available for most operating systems and will run on laptops, desktops, workstations, and servers.

*Per*l stands for "Practical Extraction and Reporting Language." *Perl* was designed as a tool for manipulating large data files, and as a "glue" language between different software packages. *Perl* makes it easy to manipulate numbers and text, files and directories, computers and networks, and especially other programs. With *perl*, it is easy to run other programs, then scan their output files for specific results, and then send these specific results off to other programs for additional analysis. *Perl* code is easy to develop, modify, and maintain. *Perl* is portable, and the same *perl* program can run on a variety of computer platforms without changes. *Perl* programs are text files and can easily be shared with other researchers.

Example: The researcher spends a surprising amount of time transforming data from one format to another. For example, many data collection devices generate text files with the individual data elements separated by commas with each record containing a variable number of characters and fields.

Many statistical analysis software packages require that each input record contain the same number of characters and fields. In order for these software packages to analyze data from data collection devices, the output of the data collection device must be cleaned (e.g., all "funny" characters must be removed), and reformatted into an electronic format that can be loaded by the statistical software.

For small amounts of data, many researchers perform this cleaning and reformatting task by hand, or within a general-purpose software package like a spreadsheet. However, this approach is prone to errors, and can result in irreproducible results. It also becomes very cumbersome and time consuming when large data files, or large numbers of files are transformed.

***Perl* is the perfect tool for these types of jobs, and it renders the analysis repeatable, which is an important aspect of the scientific method.**

References

1. Oran, A. and S. Talbott (1991.) Managing Projects with Make. O'Reilly and Associates, Inc. Sebastopol, CA., ISBN 0-937175-90-0, 149p.

2. Singh, H. (1998.) Data Warehousing: Concepts, Technologies, Implementations and Management. Prentice Hall, New Jersey, ISBN 0-13-591793-X, 332p.

3. Wall, L., T. Christiansen, and R. Schwartz (1996.) Programming Perl, 2nd Edition. O'Reilly and Associates, Sebastopol, CA, ISBN 1-56592-149-6, 646p.