uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

u Ottawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Yiwen Chen

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE / DEGREE

Systems Science

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Handling Metadata for Statistical and Other Databases

TITRE DE LA THÈSE / TITLE OF THESIS

Professor John Nash

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Professor Morad Benyoucef

Professor François Théberge

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

HANDLING METADATA FOR STATISTICAL

AND OTHER DATABASES

YIWEN CHEN

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the Msc. degree in Systems Science

School of Management

University of Ottawa

Your file Votre référence
ISBN: 978-0-494-25752-4
Our file Notre référence
ISBN: 978-0-494-25752-4

ABSTRACT

Metadata, data about data, provides information on various facets of our data. It facilitates survey design as well as data discovery, analysis and retrieval. Given the many requirements to incorporate and deal with metadata in statistical applications, metadata tools in commonly used statistical software would be very useful. R presents an opportunity for this. However, the common problems of metadata such as metadata capture and the diversity of difficulties of adding functions to R under Windows bring a big challenge for incorporating metadata functions into R.

This thesis aims to present the possibility for handling metadata in the statistical system, R. It provides an R package with functions to facilitate metadata management. Taking the advantage of the metadata levels and templates defined in the thesis, the functions attempt to implement (semi-) automatic metadata capture and updating. In addition, the functions provide the examples of programming with R especially for the novice. Furthermore, this thesis supplies a guideline of the future work, that is, a road map for the development of R metadata tools.

ACKNOWLEDGEMENTS

While writing this thesis, I have received help, support, and encourage from many people. On completing this thesis, I would like to take this opportunity to express my deepest gratitude to them.

First and foremost, I would like to thank my thesis supervisor Dr. John C. Nash for many insightful conversations during the development of the ideas in this thesis and for helpful comments on the text. Without his guidance, support, and encouragement I could not have finished this thesis so smoothly.

I note the encouragement of Glen Newton of the National Research Council and Joseph Potvin of the Treasury Board Secretariat, along with other members of the newly started Ottawa-Gatineau R User Group (OGRUG).

I also sincerely appreciate and thank you in advance to my examiners who have taken on the task of reading my work.

I extend heartfelt thanks to my family. Words alone cannot express the thanks I owe to my parents, my husband and my little son for their love, support and encouragement.

Last but not least, I would like to acknowledge my friends Helen Louie for her kindly proofreading this thesis and Wenjing Zhang for her encouragement and great help when I was in a difficult situation.

TABLE OF CONTENTS

TABLE OF FIGURE

Chapter 1 **Introduction**

1.1 Motive for the Research

The development of data analysis and data dissemination increases the requirement for metadata. Generally speaking, metadata is data about data. In the context of statistics, metadata is descriptive information or documentation about data. It helps data producers to acquire user requirements, objectives of surveys and documentation of production systems. It helps data users to discover and retrieve data with potential relevance to their problems. It may also help software tools to process displayed data and communicate with users appropriately. Therefore, it benefits not only users but also software.

Traditional databases do not contain information about stored data, metadata. This may lead to obstacles for data utilization, exchange and management. Here we present an example to make a comparison of two data sets stored in two different databases to show this problem.

In Table 1, there are two sample data sets which represent the grades in mathematics of 16 years olds leaving school in Scotland in the years 1981 and 1991 [Lamb and Smart, 2000]. To compare these data sets, the numbers shown in Table 1 are meaningless. They are not easy to understand by data users without metadata such as table headers, variable labels, units of measurements and the data source.

ID	Grade	...
1	2	...
2	5	...
3	4	...
4	3	...
...	...	...

ID	Grade	...
1	3	...
2	2	...
3	7	...
4	6	...
...	...	...

Table 1 Two database tables containing variables of 1981(left) and (1991)(right) grades

Furthermore, from Table 2 users can see that these two data sets were collected based on the two distinct grading standards. In the year 1981, the grades were classified on a scale of 1-5 while in 1991 a scale of 1-7 was used. To make the comparison, background information about data collection such as the definitions of grading standards should be provided to users. These are part of the metadata.

Global ontology	Grade (1981)	Grade (1991)
High Pass	1-2	1-2
Pass	3	3-4
Low Pass	4-5	5-7
No Award	No Award	No Award

Table 2 A global ontology for the both variables of 1981 an 1991 grades

From this simple example, we can see that it is not trivial to incorporate metadata into statistical and other databases. Nowadays, there are some software tools for handling metadata in various formats, as in the work of Bridge [Kent, 1999] and Idareasa [Fairgrieve and Brannen, 1998]. The former is an object-oriented metadata repository that was designed to develop a classification database. The latter is a metadata model, which aims to build a set of metadata computing principles. However, these tools may be particular to a few applications. Therefore, it would be very useful to deal with metadata in commonly used statistical software so we are able to meet increasing demands, either explicit or implicit, to integrate and handle metadata in statistical applications. R, an open source statistical language, presents an opportunity and challenge for this.

1.2 Brief Overview of the Research

R is a statistical system with functionality for many statistical procedures and a wide variety of graphical techniques. It is becoming a major tool for both academics and other statisticians. On the other hand, R is a functional programming language that

is highly extensible. It allows users to create packages and/or add functions to grow its functionality according to the users' requirements. In addition, with the development of database add-on packages, R seems compatible with an effort to incorporate and deal with metadata in statistical applications.

This thesis aims to present some possibilities for handling metadata in the R system. However, to achieve this target is neither easy nor smooth.

- The common problems of managing metadata and programming with R lead to many problems in collecting and updating metadata as well as in adding functions to and creating packages for R.

- Data creators are always too preoccupied with ongoing work to provide and/or keep updating the appropriate metadata.

- There is limited documentation to show how to program with R or build an R package. Few advanced tutorial, few-detailed example and a complex procedure to build a package.

Given these obstacles, we anticipate many difficulties. Indeed, we did not expect to achieve as much as we were able to do.

In order to find a feasible and effective approach to solve the issues posed by R and metadata, this research created an R package that defines metadata levels and templates and provides R functions to manage metadata.

We introduce a new type of data, metadata, into R. This bridges the gap between R and metadata management and is intended to be an aid for both data/metadata providers and end-users. We also hope that our code will benefit R developers, especially novices in programming with R, by providing example programming scripts that can be used as the basis for further development.

1.3 Reading Guide to the Thesis

- The next chapter introduces the definitions, importance and the beneficiaries of metadata from different points of view. It also discusses the concepts, common problems and the methodology in managing metadata.

- Chapter 3 introduces the main features of the statistical system R and explains the reasons we choose R for expressing our tools for handling metadata.

- Chapter 4 is a justification of this thesis work. It discusses the problems arising in the management of metadata in R. It states the key objectives of the thesis and presents why it is worthwhile to try to achieve them.

- Chapter 5 is the core part of the presentation of this research. It demonstrates the model design and the functionality of the R package Rmdset we have built. It introduces the metadata levels and templates defined to handle metadata in R. It also illustrates the usage of the functions provided by Rmdset with examples.

- Chapter 6 provides a road map for developing an integrated R-metadata system based on the proof-of-concept given by Rmdset. It discusses the functionality and the structure of such a system. It introduces R functions required in the future model as well as the mechanism to facilitate metadata storage, exchange and dissemination in XML.

- Chapter 7 summarizes the work and outlines its contributions.

Chapter 2 **A Brief View of Metadata**

This chapter provides a broad view of metadata, especially in the field of statistics. It introduces definitions of metadata from different views. It states the importance and the beneficiaries of metadata. It discusses the concepts and common problems of metadata management. Finally, it presents the methodology to handle metadata.

2.1 What is Metadata?

There exist many different definitions about metadata in the international forum. Here introduce some of them on the perspectives from general to specific.

Firstly, from a very general point of view, "Metadata is data about data" [Curran, 1999]. "Metadata answers who, what, when, where, why and how about every facet of the data" [NOAA, 2002].

> Who collected and who distributes the data?
>
> What is the subject, processing, projection[1] of the data?
>
> When was the data collected?
>
> Where the data was collected?
>
> Why the data was collected (What is the purpose)?
>
> How was the data collected? How should it be used?
>
> How much does it cost? [Metadata Education Project, 2002]

Secondly, from the data management point of view, when a database implements a data model, metadata can be regarded as the schema of the data model. In other words, metadata is its coded presentation [Ullman, 1982]. "From today's

---

[1] The act of projecting or the condition of being projected.[Answers.com, http://www.answers.com/topic/projection]

observable usage, metadata applies to various styles of symbol-encoded data description ranging from strictly formal (that is: "formatted", or typed) through semi-structured to ASCII coded free text, with marked-up text styles (HTML, XML) somewhere in between" [MetaNet WG2, 1999]. Therefore, in some extent, metadata present a data model in a formal format. Metadata explains the structure of how the different components of a data model are connected [Ullman, 1982].

Finally, from the statistical data management point of view, statistical metadata is a document of statistical data. It is the main theme of this thesis. The following two sections will state what it is, who needs it and why it is needed respectively.

2.2 Statistical Metadata

In statistics, Bo Sundgren is recognized as having coined the term "metadata". While there were parallel developments, they did not use the word [Kent, 1999]. In Sundgren's PHD dissertation paper, he describes "metadata" as "an Infological[2] Approach to Data Bases" [Sundgren 1973].

According to Sundgren (1997): "statistical metadata is descriptive information or documentation about statistical data, i.e. microdata and macrodata. Statistical metadata facilitates sharing, querying, and understanding of statistical data over the lifetime of the data." The two types of statistical data (electronic or otherwise) are described as follows (see Lenz, 1994):

> Microdata - data on the characteristics of units of a population, such as individuals, households, or establishments, collected by a census, survey, or experiment;

---

[2] The infological problem is described by Börje Langefors as the problem of how to define the information that should be provided by the system in order to satisfy user needs.

Macrodata - data derived from microdata by statistics on groups or aggregates, such as counts, means, or frequencies.

This thesis aims to present and solve the issues mainly posed by statistical metadata, for that reason, hereafter statistical metadata referred to as metadata.

2.3 Who Needs Metadata and Why?

One group that needs metadata is statistical data producers. Generally speaking, statistical data producers consist of statistical survey and statistical information system designers, input data providers and production statisticians. Metadata will inform designers of user's need or provide design experiences of similar surveys in the past and/or other statistical organizations or companies. Metadata will tell the data producers the objective of the survey, their benefits and the cost to conduct a statistical survey. For carrying the correct operations in a production procedure, metadata will provide production statisticians with checklists and other documentation of the production system.

Another group that needs metadata is statistical data users. Metadata will help them to search for statistical data with potential relevance to their problems. Metadata will assist them to retrieve the data that they identify as having potential interest. Metadata may also aid them to repeat or omit part of the analysis of retrieved data that are appropriate for segments of the data.

Software tools also can benefit from metadata as well. Metadata describes how data need to be processed in a formal way. Textual metadata enables software to display output data and communicate with users appropriately. Also, metadata may inform users how to use software tools properly to analyze statistical data. In addition, metadata permits the interpretation of the result of an analysis in a

responsible way [United Nation, 1995].

To conclude, the rapid development of technologies in communications and statistical data management increases the requirements for quality in statistical information exchange. In order to design effective and harmonized statistical information systems, good metadata and good metadata handling play significant roles. The major techniques used in handling metadata are discussed in the next section.

2.4 Art of Handling Metadata

2.4.1 Metadata Management

Metadata capture, metadata management, metadata dissemination and metadata discovery are the typical steps in the procedure of handling metadata. Generally, these procedures may include some or all of these steps; even the methodologies applied may be different [Pierre and Restivo, 1998]. The definitions of these steps follow:

- ***Metadata Capture***

Metadata capture is the procedure of gathering metadata into the metadata management stage. Metadata can be gathered either automatically or manually.

Modern or future statistical information systems should collect metadata with as few manual operations as possible. Metadata should be captured at one time if this is feasible [United Nations, 1995]. In order to increase the effectiveness and efficiency of metadata capture, metadata should be collected while data and metadata are being created or gathered if this is possible. This means metadata should be gathered at the same time as other activities in statistical data production are taking place. Metadata capture is as important and significant as other missions are.

Metadata should be a by-product of the statistical production process [Sundgren, 2003].

❋ ***Metadata Management***

Metadata management is the process of incorporating captured metadata into a metadata management system for further processing. Generally speaking, it may include part or all of the following manipulations: metadata updating, metadata filtering, metadata reviewing and metadata preparation for dissemination.

Metadata management consists of its administration and manipulation. A metadata repository is the solution to implement this procedure. The generic structure of a metadata repository will be introduced in the section 2.4.3. To build a metadata repository, metadata templates and a metadata standard are the two key concepts that need to be understood. Below we give their definitions.

Metadata standard --- A metadata standard is used to describe the information or documentation about data. For documenting statistical survey design, processing, analysis or data dissemination, it is proposed that this be designed as a comprehensive, hierarchical thesaurus of terms or an outline of all the concepts contained in it. The purpose of a metadata standard is to define entities in metadata models, define relationships with other standards and support these related standards [Laplant, Lestina, Gillman and Appel, 1996].

Metadata template --- A Metadata template is a specification of the construction and attributes of the metadata elements. The statistical metadata repository may allow a user to create a metadata template based on the metadata elements defined by a specific metadata standard. It should allow a user to specify the possible levels or values of the new elements when they are added to the metadata template. It also allows a user to define the attributes of the metadata elements. For example, a user can define the maximum number of times an element can occur, or define the value

of an element as a string, a date, an integer, a floating-point number (i.e., a real) or a Boolean variable [Pierre and Restivo, 1998].

- ***Metadata Dissemination***

Metadata dissemination provides ways to help users to find metadata. Metadata can be disseminated using different approaches with different formats such as posting on a server or printing in a magazine [Pierre and Restivo, 1998].

- ***Metadata Discovery***

The process of metadata discovery allows users to search for required metadata. In this process, users can issue a formal query (e.g. using SQL), browse the data or use other search methods [Pierre and Restivo, 1998].

This thesis focuses on the metadata capture and management.

2.4.2 MetaModel

According to Tannenbaum (2002), a metamodel can be defined as the following:

> "A metamodel is the graphic representation of an organized set of metadata requirements, typically organized by any of the following perspectives:
> The metadata's source (a tool, application, repository, etc)
> The metadata's classification (common, unique, or specific) and associated beneficiary category
> The metadata's usage (e.g., a particular development methodology)"

A common metamodel could be depicted as shown in Figure 1

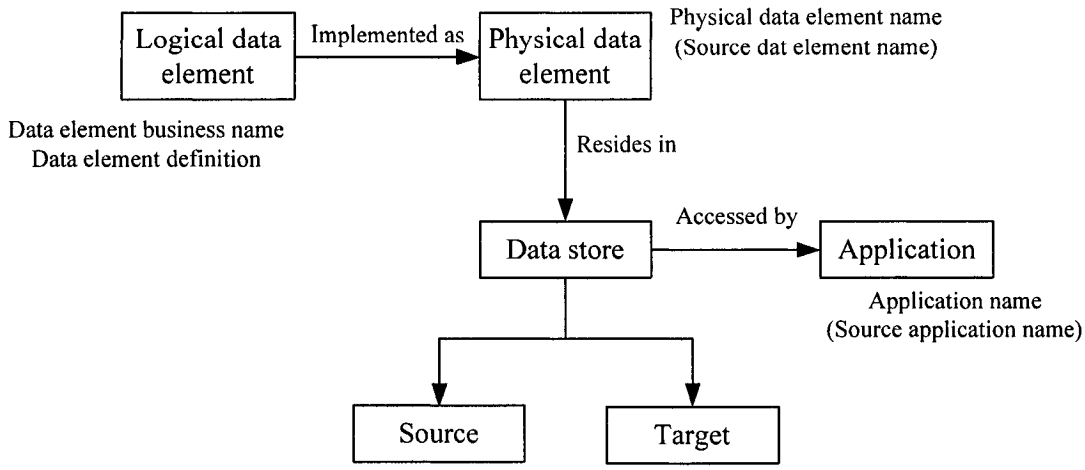

Figure 1 A common metamodel [Tannenbaum , 2002]

From the above definition, a metamodel is a model of a semantic structure. In statistics, semantics represents the meaning and behavior of the components of metadata. So a statistical metamodel is a graphical description of the structure and semantics of metadata elements. It provides a consistent specification for using metadata content. Note that there are other approaches for example that of the Object Management Group (shortened as OMG) which uses the Unified Modeling Language (UML) [Kent, 1999].

2.4.3 Metadata Repository

In the information system, a metadata repository is a store for holding metadata and/or pointers to external metadata in a distributed metadata environment. A true metadata repository is an integrated system that means metamodels of different repositories are inter-related to allow different metadata sources to be accessible and connected with each other [Tannenbaum, 2002]. "A statistical metadata repository (MDR) is a logically central statistical metadata repository that allows for the query, editing, and managing of metadata. Such a system provides a mechanism for looking up information about statistical products as well as their design, development, and analysis" [CODED, The Eurostat Concepts and Definitions Database, 2003].

❋ ***Metadata Repository Structure***

Generally speaking, a metadata repository consists of the repository software, the repository database that underlines this and the repository metamodel. Figure 2 shows the structure of a generic metadata repository [Tannenbaum, 2002].

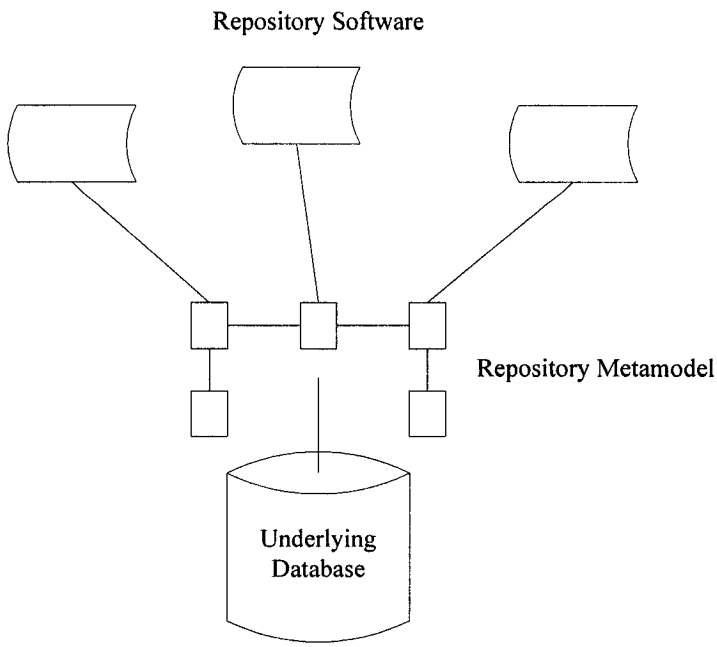

Figure 2 Generic metadata repository

❋ ***The Repository Software***

In general, repository software includes types of "Go-get-it routines," "repository policies," "repository templates," "repository utilities."

> Go-get-it routines --- The application program interfaces (APIs), remote procedure calls (RPCs), and external interfaces that allow the outside world to access repository contents.
>
> Repository polices --- Repository program code that is implemented, controlled, and executed by the repository; used to control repository access and repository update as well as other

repository functionality.

Repository templates --- Views that control and format the display of repository contents

Repository utilities --- Vendor-supplied functions that perform standard repository functions (back up, load, etc.) [Tannenbaum, 2002].

❋ ***The Repository Database***

The structure of a metadata repository database can be either open or closed. An open metadata repository database is designed and built in a database management system and can be accessed by query languages such as SQL. In contrast, a closed metadata repository database can only be accessed through the tools provided by its developer. It is not easy to say which one of these two kinds of database is better, since much depends on how the database is designed [Tannenbaum, 2002].

❋ ***The Repository Metamodel***

The repository metamodel is the logical centre of a metadata repository. It is logically connected with each part of a metadata repository: the repository database, the repository software as well as the metamodels of connected or related repositories. The repository metamodel correlates to the fundamental repository database directly. It records how the stored metadata, related metamodels and the repository software are linked each other It also performs tasks such as to control the security of repository and metadata [Tannenbaum, 2002].

Chapter 3 **R and Why We Use It**

At present, there are some software tools for handling metadata in various formats, but these may be particular to a few applications, such as metadata repository Bridge [Kent, 1999] and metadata model Idaresa [Fairgrieve and Brannen, 1998]. Hence, it would be very useful to deal with metadata in commonly used statistical software to deal with the increasing demands to integrate and handle metadata in statistical applications. We chose R, a powerful and emerging statistical package and programming system, as the example platform on which to base our research. An introduction of R and discussion of why we chose R are given in this chapter.

3.1 What is R?

R, a GNU project (www.gnu.org), is a computer language and a statistical environment. It was designed as a dialect of S[3] to supply a wide variety of statistical techniques and highly extensible graphics [R Home Page, 2003].

The initial authors of R were Robert Gentleman and Ross Ihaka of the University of Auckland in New Zealand. Partially derived from their first names and partly from a play on the name of the Bell Labs language 'S', R is literally named 'R'. From the "Comprehensive R Archive Network" (CRAN) users can download the sources, binaries and documentation for R [Hornik, 2005].

As an interpreted computer language, R provides interfaces and debugging facilities to other programming languages such as C, FORTRAN, Perl, Python, and Java. It also provides a support to many data technologies like XML, database

---

[3] Developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues.

integration and Simple Object Access Protocol (SOAP)[4] [Steffen Durinck, 2003].

As a statistical environment, R can increase its functionality by adding new functions defined by users. R also can extend itself by creating packages. R has many built-in and plug-in packages dealing with a very wide range of modern statistics. In addition, with its own LaTeX-like documentation format, the extensive documentation can be provided in a number of formats on-line or in hardcopy [Hornik, 2005].

3.2 Why We Use R

With the development of R, more and more academics or other statisticians learn, use and recommend it. R has been introduced in lectures of departments of mathematics and statistics in such institutions as Stanford University in the USA, Queen's University and Waterloo University in Canada, and Beijing University in China. Meanwhile, some large government departments and private companies are using it as a major tool for statistics and data processing.

To date, R has not been particularly good for handling extremely large data sets, because R stores and manipulates data in main memory (RAM), nor in supporting concurrent access to data. For such tasks, it makes sense to use database management systems (DBMs), in particular relational DBMs (RDBMs) to help to do these things well. On CRAN, there are several packages that allow R to communicate with DBMs. Most of these packages have functions to allow the data within a database to be selected by SQL queries and to be retrieved as a whole data frame or in pieces thereof as groups of rows [R Development Core Team, 2003].

As we have indicated, given the many requirements to incorporate and deal with

---

[4] http://www.w3.org/TR/soap/

metadata in statistical applications, widely applicable tools for handling metadata in commonly used statistical software would be very useful. Because of its open source nature and extensive capabilities for extension and modification, R presents an opportunity and challenge for developing statistical metadata tools.

Chapter 4 **The Research Question**

This chapter is a justification of this research. It first introduces problems posed by metadata management. Second, it presents the problems arising from using R to handle statistical metadata. These are the central issues that this research aims to address. Finally, this chapter discusses why it is worthwhile to solve those problems.

4.1 Problem Summary

With the feature of extensibility, R presents the possibility to build an R package and/or add functions to R to handle metadata. However, it is not always easy to achieve the above tasks. Moreover, with the problems inherent in handling metadata, developers may encounter many difficulties, which are enumerated in the following paragraphs.

❋ ***How to Get Metadata***

Experience has shown that metadata capture plays a crucial role in the success of statistical information systems. Tedium, financial cost and time demands of the procedure(s) of metadata capture as well as the disjunction between metadata providers and users are the major reasons for failure [United Nations, 1995].

Metadata should be provided by the data producers. Unfortunately, they usually will not supply the metadata for several reasons. Data providers often keep their knowledge about data in their heads because they do not need the explicit metadata. Furthermore, they are often too busy or too lazy to supply the metadata. It takes data suppliers a lot of effort to produce good metadata but with few rewards. By contrast, metadata users know the importance of metadata but cannot produce it by themselves. Moreover, metadata users may expect to get metadata with little or

no expense [United Nations, 1995]. Therefore, motivating data producers to provide metadata is a central theme of all metadata projects including this thesis research.

❋ ***How to Keep Metadata up to Date***

The contents of a dataset will vary over time. The process of survey design or data editing will result in changes in a data set. Sometimes corresponding metadata are not be updated together with the data. Metadata lose their value without accuracy, and the failure to update metadata devalues the data set. Clearly, what users need is the current version of metadata. Therefore, continuously and efficiently updating metadata should be a necessary feature of a metadata management system.

In order to guarantee the quality of metadata, a function to check the new or changed data in a data set and then to harvest corresponding metadata is thus a necessary feature of a metadata tool.

❋ ***How to Bring Metadata into R***

While there are many built-in and plug-in packages for handling statistical data in R, few of them are for metadata. Nowadays the growing role of statistical metadata brings an extensive amount of metadata for use in most statistical offices [Bethlehem, 2003]. R has powerful statistical functionality and various features for handling data, but without the tools for handling metadata, users may feel it is not convenient or suitable for large statistical offices or systems.

While there are a few theoretical treatments of statistical metadata, we have not found well-documented experience in handling metadata with R. Give that we could find no precedent for this work, the task was a lonely one. Nevertheless, some progress has been made.

❋ ***How to Program with R***

It is difficult to find explicit documentation or examples for programming with R at the detailed level needed for metadata handling. However, more and more R tutorials are being offered by educational or research organizations, some posted online though most of them are on the introductory level. These generally only give brief statements of the basic rules to write an R function with simple examples, and these are mostly for statistical or graphical rather than data management tasks. To add a useful function to R we still need more guidance.

Learning from examples of R source code is a method worth considering. Unfortunately, R is a large package, and for a new R developer there are many details to learn. The source code of one function may refer to one or more other functions that one needs to also comprehend. To understand the source, one is driven to learn much more than just one routine. Questions come forth in succession, and may overwhelm the reader.

As we will describe later, our Rmdset package requires other R functionality. Examples are the DBI package to provide a Database Interface and the RMySQL package that provides a link to the MySQL database system. The former package is one of those that is supplied on the collection of the Comprehensive R Archive Network (CRAN) mirrors, so is easy, indeed almost automatic, to install using R's excellent package installation system. However, RmySQL[5] was not so easily acquired and set up. While it is listed in CRAN web pages, it is not available for direct install.

❋ ***How to Build an R Package***

The R project provides a document, "Writing R Extensions" (R-exts), to help users to

---

[5] RMySQL 05-6 is available on http://stat.bell-labs.com/RS-DBI/download

create R add-on packages and write R documentation. Though it explains the process from the package structure, checking, and building packages to submitting a package to CRAN (The Comprehensive R Archive Network), users may still feel it is difficult to complete those tasks. For example, to build an R package under Windows, users need to set up a Unix-like environment. Also necessary will be the installation of some other software, such as Rtools. There is no information about this in the R-exts (R manual, Writing R Extensions). Therefore, users may have no idea where to start.

In order to build the package designed as a proof of concept of ideas within this thesis, I used a few documents found by searching the Internet, such as "Building R for Windows" (Sutherland, 2005) and "Creating R Packages" (Nunes, 2005). These articles fortunately give the instructions systematically. However, you cannot finish the job by depending only on one of these references. My experience was that it was necessary to

- compare and combine the advice in the documents;
- try repeatedly to select the appropriate recommended methods and settings;
- understand the problems encountered in the process.

Clearly, this was a process that was time consuming and frustrating.

4.2 The Implementation Objectives

The main purpose of this research is to show the possibility for handling metadata in a statistical system such as R. In another word, it aims to find appropriate approaches to solve the problems discussed in the above section. Therefore, this research designed and implemented an R package with the functionality, which is introduced below.

❋ ***Define Metadata Levels and Templates to Aid Metadata Capture and Maintenance***

To incorporate metadata handling into R, this research defined and designed metadata levels and templates to facilitate metadata capture, maintenance and management. The definitions of metadata levels and the elements of each metadata level help systems and users to explain, utilize, and store a data set. They provide a mechanism to capture metadata as a by-product of a data set. The metadata templates provide data structures to store and retrieve metadata. Metadata are allowed to be saved in the user's specified databases and to be invoked by R. In conceptual terms, this implements the objective to manipulate metadata within the R system.

❋ ***Build R Functions To Assist Metadata Management***

Besides the model of metadata templates, this research also added functions to R for managing metadata. These functions allow users to do the following tasks:

- to generate metadata automatically or semi-manually;
- to discover the new or changed data;
- to update and maintain metadata;
- to display appropriate metadata with data.

❋ ***Provide documentation for Package Installation and Usage***

This thesis provides a document to illustrate the installation and the usage of the R package with examples and sample datasets. In addition, accompanied with each function, there is a LaTeX-like format manual, which gives the users information about the function such as the function name, a brief statement of the usage, the required argument(s) and the acceptable value(s) of the argument(s). These manuals also demonstrate how to use the functions with one or more examples.

❋ ***Provide a Road Map for Developing An Integrated R-Metadata System***

At present, issues about dealing with metadata in R are still in the baby stage. Developers are facing a great deal of difficulty along the way to develop an R-metadata system. To avoid losing the direction, this thesis also aims to provide a road map for the future development.

4.3 Key Elements of the Research

This research should contribute to the development of both R and statistical metadata management. It benefits the metadata producers, the metadata users, and the R developers.

❋ ***Add a New Type of Data (Metadata) to R***

From the simplest definition "data about data," metadata is a kind of data. Among all the R built-in and plug-in packages, few have metadata as a data type. The package built with this thesis defined metadata templates and elements and stored them in the package. It also contains functions that can create, import and export metadata. It brings a new type of data, metadata, to R.

❋ ***Provide a Prototype to Show How to Build an R Metadata Repository***

A model of statistical metadata repository was designed and implemented to fill the bridge between R and metadata management. This model can be regarded as the prototype of an R metadata management system: it provides the basic functions of a metadata management system and it affords the feasible methods to handle statistical metadata with R. The structure and mechanism of this model may promote future development of incorporating statistical metadata management into R.

❋ ***Help to Decrease the Burden to the Metadata Providers***

The R package build with this thesis provides functions to reduce the workload of the

data producers. For instance, some functions were designed to generate some kinds of metadata automatically. That means users do not need to type all the required metadata manually. These functions also supply some metadata options to allow users to input the metadata by pressing only one key on the keyboard instead of typing characters and/or numbers. These features may greatly reduce the manual operations to the data producers. In addition, the metadata templates built with this thesis assist the users to create metadata in a consistent format.

- ***Aid to Increase the Benefit to Data Users***

On the one hand, the functions designed to generate metadata will remind the data producers to provide the metadata that must be supplied by them. This feature guarantees the users provide the metadata to some extent. On the other hand, the function designed to update metadata would help the data producers to afford the current version metadata to the data users. Furthermore, the metadata templates built with this thesis facilitate the users to identify and utilize metadata.

- ***Provide Documentation and Samples in Programming with R***

To incorporate metadata into R directly, all the functions supplied by this thesis were developed in the environment of R. Therefore, this thesis can provide suggestions and samples for other R programmers especially for novices. Additionally, this thesis contributes a comprehensive document for people who want to make an R package under Windows. It will greatly reduce their effort and save their time to do so.

As has been noted previously, this thesis aims to solve the problems discussed early in this chapter. The mechanism and solution will contribute to the development for R and the metadata management system as well as their users. The following chapter gives the detailed answer to how to deal with these problems.

Chapter 5 **The R Metadata/data Set Package – Rmdset**

This research designed and implemented an R package to present some possibilities for handling statistical metadata in R. The package built as described herein provides a feasible and effective approach to respond to the research questions mentioned in the last chapter. It consists of the definition of metadata levels and templates, R functions and documentation such as help pages. For better understanding of the package, usage examples are also provided in this chapter.

5.1 Package Introduction

5.1.1 Package Functionality

As discussed in the section 4.3, the R package built here intends to:

- Define metadata levels and templates to aid metadata capture and maintenance;
- Build R functions to assist metadata management;
- Provide documentation for the package installation and usage.

The metadata levels were defined based on the typology introduced by Kent and Schuerhoof (1997). Metadata elements of each level were selected based on some standards and projects about statistical metadata. The Dublin Core, IT standard 16.0.0[6] and the ICPSR[7] (Inter-university Consortium for Political and Social Research) project are the major references.

---

6 IT Standard 16.0.0: survey design and statistical methodology metadata, U.S. Census Bureau.

7 Supported by United States Department of Health and Human Services. Substance Abuse and Mental Health Services Administration. Office of Applied Studies

When users produce a data set, there is a function allows them to build templates for the metadata and capture the metadata as a by-product of the data set. When users modify a data set, there are functions to allow the user to update the metadata at the same time.

The resulting data sets will be saved with the corresponding metadata as {data, metadata} sets in the user's specified databases. Users can access, edit, and filter the data sets and metadata sets using SQL queries and updates

This package is easy to install. With the examples provided by the package, users can learn how to use it quickly. Users also can get help from the R documents attached to the package.

5.1.2 Analysis of the Package Design

❋ ***Package Name***

"Rmdset" stands for R metadata/data set. It was named because Rmdset will generate a data-metadata set, {data set, metadata set}, when requested to create a data set. The data-metadata set will include the new data set and its accompanying metadata set. Here we use "metadata set" because we classified metadata in three levels, operational, conceptual and documentary (see section 5.2.1 and 5.3), and we will present each level of metadata in a separate data frame. Then a metadata set will be a collection of all the levels of metadata of a data set. In this way, we may not only keep a data set and its metadata in an individual database, but also can manipulate the data set or any level of its metadata separately.

❋ ***Package Selection***

As we introduced in Chapter 3, powerful statistical functions and flexible graphical displays have led to R becoming widely used. Incorporating and dealing with metadata in statistical applications, design and build R metadata tools would be very

useful.

R is open source software that can extend its functionality by adding plug-in packages. R allows users to bind their functions and useful data into a package. Once users installed this package, these data and functions can be invoked by the R functions library() and data() at will.

As an interpreted computer language, R has interfaces to databases. Among the R data base packages, RMySQL is the most developed "back-end" package [R Development Core Team, 2003]. It works with the "front-end" package DBI to provide users the functionality to read, write and edit data in MySQL databases.

Moreover, MySQL database is the most popular open source database in the world. It is well known for its consistent fast performance, high reliability and ease of use. From large corporations to specialized embedded applications it has been applied in more than 10 million installations on every continent in the world [www.mysql.com].

Therefore, we created the package Rmdset based on R, RMySQL and DBI packages. Package Rmdset contains metadata templates and R functions to facilitate the capture and update of metadata. These functions will export metadata output to the users' specified MySQL databases. RMySQL functions also can retrieve the stored metadata into R for display or further editing. This thesis provides help documents to assist users to master Rmdset and RMySQL.

❋ ***Model Introduction***

According to the definition referred to in Chapter 2, package Rmdset acts as a prototype of an R metadata repository. First, it links to the database for holding metadata. Second, it allows users to access and retrieve metadata in the database. Third, it allows users to edit and manage metadata in the database. Figure 4 shows

how the package Rmdset handles metadata in R. Rmdset stands for R metadata/data set.

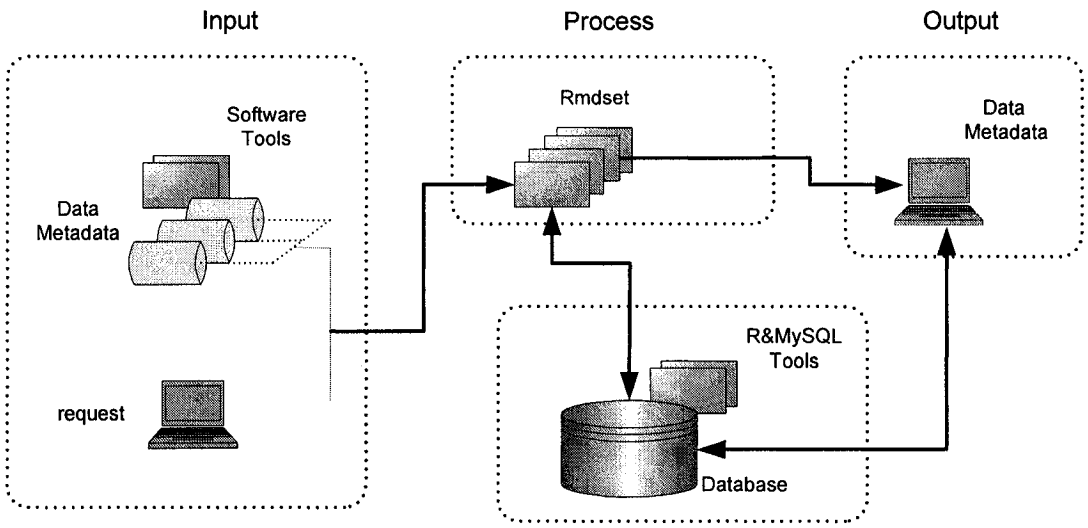

Figure 3 Procedure of handling metadata in R

The package permits users to enter data and metadata for creating data sets, metadata and metadata templates. It also accepts users' requests to produce, edit or retrieve data and metadata.

In the process stage, the package handles metadata based on the users' requirements. When users intend to create a data set, Rmdset will generate a data-metadata set with input data. When users alter the data, Rmdset will update the metadata automatically.

All the data, metadata and metadata templates imported or generated by Rmdset are allowed to be displayed on the screen directly or stored in a MySQL database. By SQL queries, stored data and metadata can be exported for screen display or further handling in R.

5.1.3 Essential Software for Building the Package

The R package was built under Windows XP with packages R 2.1.1, RMySQL 0.5-6, and MySQL 4.1. To run the package properly, users should install the above software with the same or later version. However, R and MySQL are available for other platforms such as Linux, though we will only discuss the Windows installation in this thesis. To save user effort, we introduce the websites where users can freely download these packages:

- R 2.1.1 --- http://cran.r-project.org/

  Note that this site leads visitors to local mirrors for the R software

- RMySQL 05-6 --- http://stat.bell-labs.com/RS-DBI/download

  Note that there is no Windows binary ".zip" file for RMySQL as for other packages in CRAN (The Comprehensive R Archive Network). That is why some Windows users feel it is difficult to find this package.

- MySQL 4.1 --- http://dev.mysql.com/downloads/mysql/4.1

To create an R package under Windows is also a challenge for novices. This is a common question posted in some R forums. One of problems is to search and install the essential software. Indeed, the process of creating the package Rmdset required a lot of time and effort. We repeat our advice to use the recommended software platform (R 2.1.1 and Window XP) when building Rmdset to avoid difficulties.

To build an R package under Windows, users will need Rtools, minGW, Perl, fptex and HtmlHelp. Below we give a brief description about each of these.

- Rtools --- A set of UNIX utilities, which was created by Brian Ripley and

Duncan Murdoch. The version used by this research was downloaded from http://www.murdoch-sutherland.com/Rtools/tools.zip

- minGW --- Minimalist GNU (www.gnu.org) for Windows. "A collection of freely available and freely distributable Windows specific header files and import libraries combined with GNU toolset that allow one to produce native Windows programs that do not rely on any 3rd-party C runtime DLLs". This is needed only if your package contains C/C++ /Fortran code. This research used Cygwin 5.0.0 which was download from http://www.mingw.org/download.shtml

- Perl --- The Windows port of Perl (a scripting language used by the R installer and checker). Version 5.8.4 or later are all suitable. The package Rmdset was build with Perl 5.8.7 Build 815 from http://www.activestate.com/Products/ActivePerl/Download.html

- fptex --- A tool to format the LaTeX version of documentation pages. Users will not be able to check for TeX errors generated in the creation of their documentation without fpTeX. It can be downloaded from http://ftp.ktug.or.kr/tex-archive/systems/win32/fptex/current/

- HtmlHelp --- The Microsoft html help compiler to make compiled html (.chm) files. The version used by this research was downloaded from http://msdn.microsoft.com/library/enus/htmlhelp/html/hwmicrosofthtmlhelp downloads.asp

Moreover, during the process of developing the package, the package JGR was adopted as a beneficial extra tool. JGR is a JAVA GUI for R. It won the 2005 Chambers Award for its significant contribution for R. The features supplied by JGR brought convenience and benefits for developing Rmdset. I believe it will also bring these advantages for users of Rmdset. Hence, it is worth giving an

introduction of these features, especially for novices.

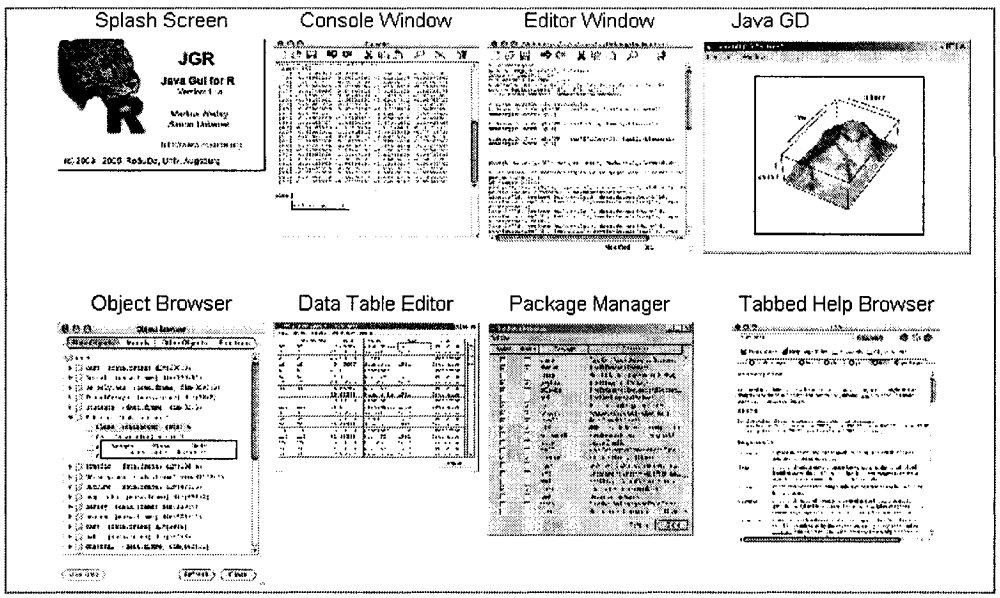

Figure 4 Features of JGR

Figure 4 shows the screen shots of the main features of JGR. It was copied from the home page of JGR.

- Console Window --- the input-output area with the features of syntax highlighting, multi-line history display and auto completion by pressing the TAB key.

- Editor --- a text editor with features of syntax highlighting, quick hints, brace matching and dragging and dropping command lines into console.

- SpreadSheet --- a data table editor with features of sorting on columns, inserting and deleting rows and columns and renaming columns. If a data table is completed and saved in the computer it could be imported in R by using the function read.table().

- Package Manager --- provides a list of R plug-in packages and a list of

loaded packages. User can install or uninstall and load or unload packages by clicking the name of the corresponding package in the list.

- Help System --- an online help system for R and all the loaded packages [Helbig, Urbanek and Theus, 2004].

5.1.4 Package Structure

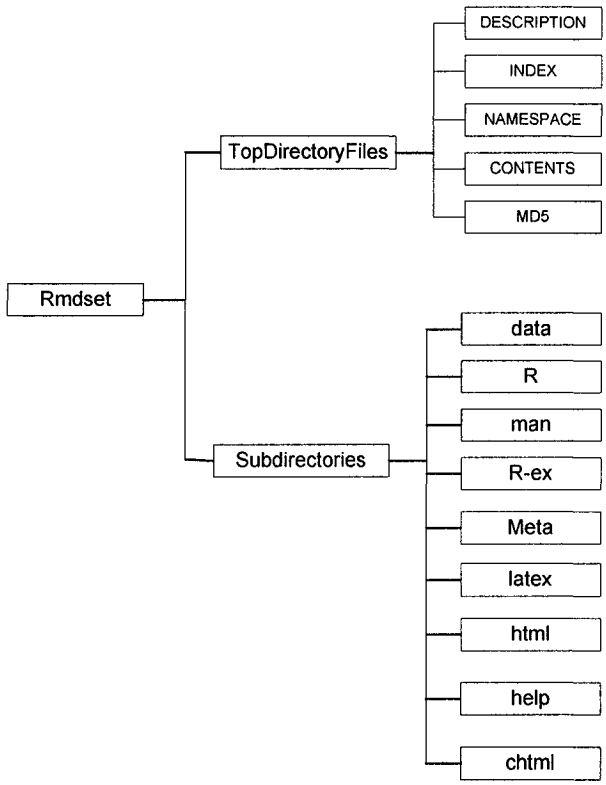

Figure 5 The structure of the package Rmdest

An R package is a collection of top directory files and subdirectories containing source code and documentation to facilitate system administration and utilization for users. Based on the objective and the structure of package design, these files may differ from one package to another. As shown in Figure 5, files in the package Rmdset are straightforward and necessary for developers, users and systems.

The "DESCRIPTION" file contains the basic information about the package written as ASCII text. This gives a brief description of what the package does as well as information about the package such as title, release date and which this package depends on. Figure 6 is a copy of the DESCRIPTION file of Rmdset.

```
Package: Rmdset
Type: Package
Title: R metadata set
Version: 1.0
Date: 2005-12-08
Author: Yiwen Chen <ychen079@uottawa.ca>
Maintainer: Yiwen Chen <ychen079@uottawa.ca>
Description: represents the possibility for handling metadata in R
Depends: R (>= 2.1.1), DBI (>= 0.1-4), RMySQL(>= 0.5-6)
License: GPL
Packaged: Fri Dec 08 07:10:18 2005; YIWEN CHEN
Built: R 2.1.1; 2005-12-08 07:10:19; Windows
```

Figure 6 The DESCRIPTION file in the R package Rmdset

The file 'INDEX' gives the name and description for each sufficiently interesting object in the package [R Development Core Team, 2005]. In terms of Rmdset, the interesting objects are the data and functions built for the package (Figure 7). "docmd" and "varMDSet " are two R data files. "md.addVar", "md.dataset" and "md.delVar" are three R functions. These will be introduced in the next two sections.

```
docmd              Documentary Level Metadata Template
md.addVar          Add a New Variable with Metadata to a Data Set
md.dataset         Create  a Data Set with Metadata
md.delVar          Delete a Variable with Metadata from a Data Set
varMDSet           Variable Metadata Set
```

Figure 7 The R package INDEX file

The file 'NAMESPACE' has three objectives: to hide internal functions and data, to prevent functions from breaking when a user (or other package writer) picks a name that clashes with one in the package and to provide a way to refer to an object within a particular package. The namespace file specifies the objects that our package exports and imports. It specifies which variables and functions are available to package users, and which variables and functions should be imported from other packages. Once a package has been loaded, the imported variables from other packages will cause these other packages to be loaded as well (unless they have already been loaded) [R Development Core Team, 2005].

Besides top directory files, the contents in the subdirectories are equally important for both users and the package. The two most important subdirectories "data" and "R" contain the data objects and the interpreted code of the package. "data" will be described in the section 5.2, metadata levels and templates, while "R" will be discussed in the section 5.3, the R functions.

In addition to "data" and "R", the Rmdset package also supplies subdirectories like man, chtml and R-ex. For a general user, having the knowledge of "man" and "R-ex" is enough for them to utilize the functionality of a package. "man" supplies manual pages for each object in the package and meanwhile "R-ex" keeps the usage examples for users. Moreover, from R help, the special feature of an R package, users will get demo scripts that demonstrate the functionality of the package. Specifically, the latex document format presents R help pages in a nice and unified display. Figure 8 and 9 show the part of help pages of Rmdset as examples.

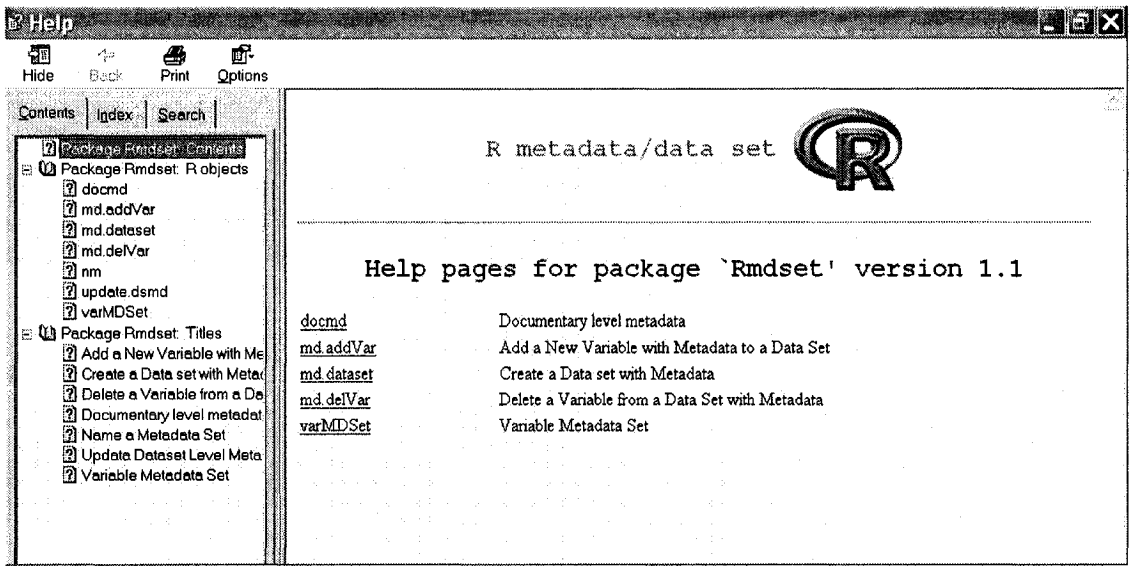

Figure 8 Help page for the R package Rmdset

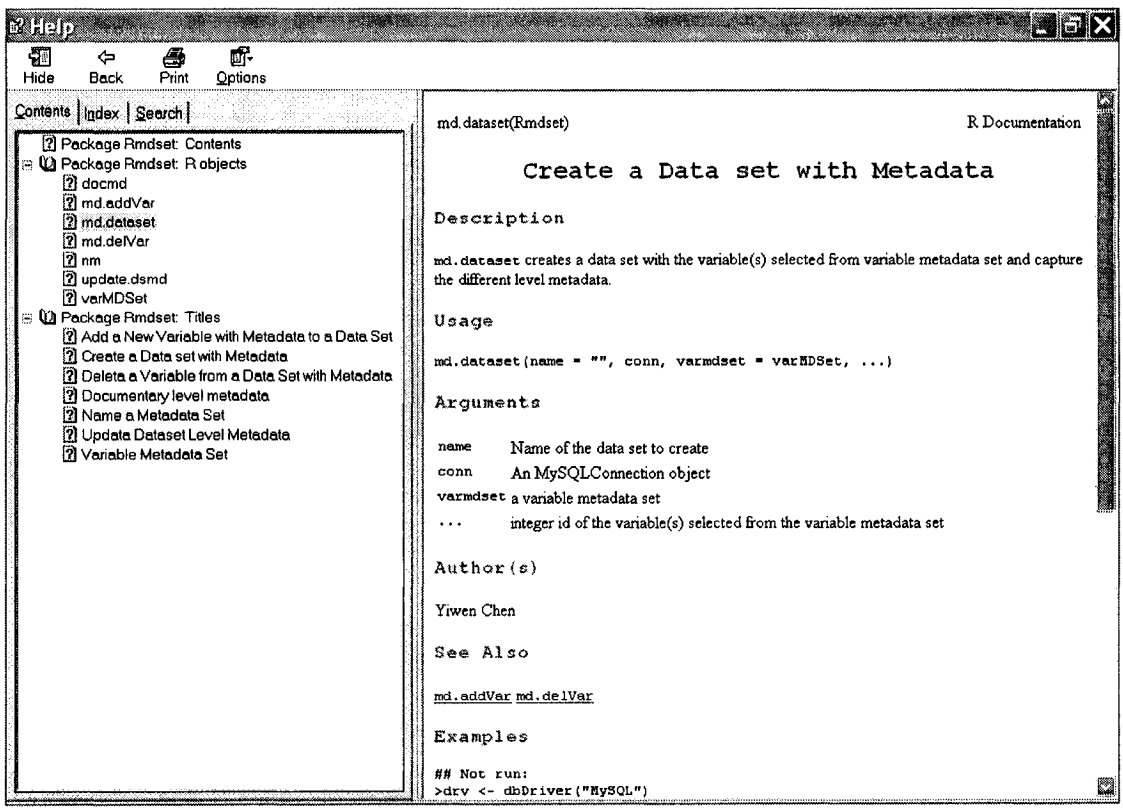

Figure 9 Help page for the function md.dataset

5.2 Metadata Levels and Templates

5.2.1 Metadata Levels

Based on the typology introduced by Kent and Schuerhoof in 1997, metadata were classified into five levels: operational, conceptual, logistic, documentary and process. This research focused on the first three levels of metadata and defined them as followings:

- Operational metadata --- metadata used to automate statistical activities. It includes data set level metadata and variable level metadata. For example, data set name, variables and value domain;
- Conceptual metadata --- metadata used to define and standardize statistical concepts such as subject matter concepts or special information;
- Documentary metadata --- metadata supplied for the end users. It includes information such as data source, storage, transformation and description. For example, file name, file location and storage format.

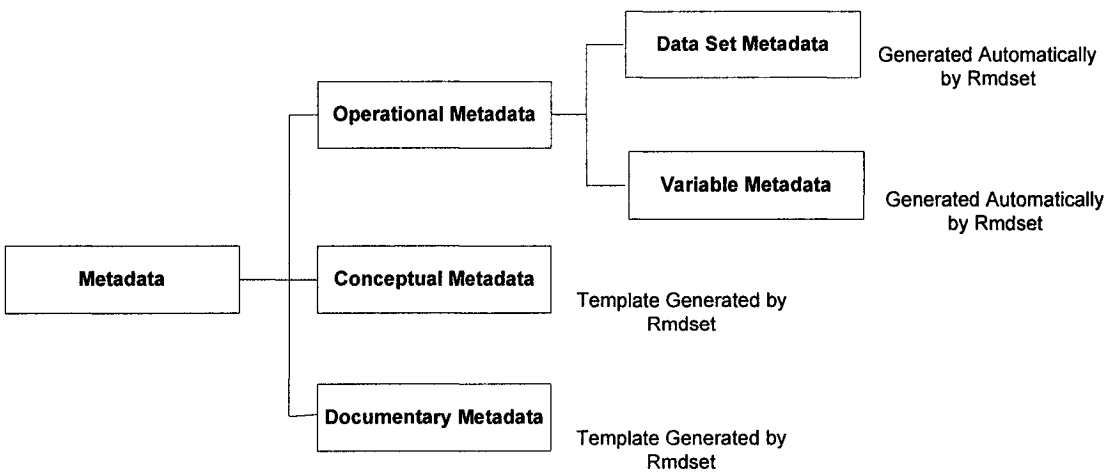

Figure 10 Metadata levels

Rmdset provides functions that will generate metadata (semi-) automatically. This means operational level metadata (data set level and variable level) will be created while building a new data set. The conceptual and documentary metadata still need to be provided by users. However, the package functions will create templates to assist metadata capture (section 5.2.2). Moreover, when users insert or remove a variable from a data set, Rmdset will update the operational level metadata of this variable automatically (section 5.3).

5.2.2 Metadata Elements and Templates

To facilitate metadata management we defined a metadata template and elements for each metadata level. The elements of each level were selected based on the Dublin Core, IT standard 16.0.0[8], the ICPSR[9] and some other metadata standards and projects.

In the current design, each set will only contain the primary or necessary elements to produce a data set. Because this research just try to show the possibility of using R to handle statistical metadata. With the further development of Rmdset, more and more elements will be added to each level. Users can obtain them by installing and loading a newer version Rmdset package. Like most other R packages, the procedure of installing and loading Rmdset can be done in a few minutes.

❋ ***Operational Level Metadata Elements and Templates***

The operation level metadata has two sublevels: data set level and variable level.

---

8 IT Standard 16.0.0: survey design and statistical methodology metadata, U.S. Census Bureau.

9 Inter-university Consortium for Political and Social Research supported by United States Department of Health and Human Services. Substance Abuse and Mental Health Services Administration. Office of Applied Studies

Metadata	Description
DatasetName	the name of a data set
CreatedAt	the date and time that the data set was created
LastModifiedAt	the last modification date and time of a data set
TimeZone	time zone of the recorded time
Variables	lists all the variables in a data set

Table 3 Data set level metadata elements

Metadata	Description
Variable	name of a variable
Unit	measure unit of a variable including default and all acceptable units
DataType	data type of a variable, e.g. Integer(4),Character(20), DATE
ValueDomain	acceptable value or value range
Comment	a brief comment of a variable such as "Created", "Added", or "Deleted"
At	the last modification date and time of a variable

Table 4 Variable level metadata elements

The actual database tables defining the elements given above in Table 3 or Table 4 will have a very similar structure to these tables. However, the values of metadata may differ depend on the different data sets. Figure 11 shows an example data set level metadata of a data set.

	Metadata	Value
1	DatesetName	test
2	CreatedAt	Thu Dec 08 09:42:43 2005
3	LastModifiedAt	Thu Dec 08 09:42:43 2005
4	TimeZone	Eastern Standard Time
5	VariableName	ID,Age,Treatment

Figure 11 An example data set level metadata of a data set

❋ ***Conceptual Level Metadata Elements and Template***

Metadata	Description
ConceptName	subjection matter concept
Definition	definition of a concept

Table 5 Conceptual level metadata elements

As previously introduced, conceptual metadata is data about subject matter concepts or special information. This kind of information is not as easy to capture as operational metadata. On the one hand, different data sets are built on different research themes or targets, which will lead to defining different concepts based on the subjects. For example, a surgical trial of intracerebral haemorrhage should give a definition or an introduction about the experimental surgery method. On the other hand, the subject matter concepts are always new concepts created by the research. Therefore, the conceptual level metadata should be supplied by the data producers. However, Rmdset provides a function md.dataset() that will generate a template to assist the data producers to provide metadata.

Notes: 'NA' = Not Available

Data has not yet to be entered in the table.

ConceptName	**Definition**
NA	NA

Table 6 Conceptual level metadata template

❋ ***Documentary Level Metadata Elements and Template***

Metadata	**Description**
DocumentName	name of document
FileName	filename of document
Author	author name of document
Language	language version of document, default English
Dataset ID	a unique identifier for the data set
Population	the entire aggregation of items from which samples can be drawn
Sample	survey sampling
Location	information to help a user to find the data set
Description	to help users to judge relevance to their research
Purpose	purpose of research/survey/clinical trial
Software	software used to create the data set
Quality	quality meets needs
Usage	meaning of data and how use the data set
Transfer Format	format for transfer the data
Storage Format	format for storage the data

Table 7 Documentary level metadata elements

Documentary metadata are metadata for the data end-users. Metadata such as a description of a data set is always saved in a separate file. In this case, metadata will be the location of the file. This infomation must be provided by the data producer. Similarly as for conceptual metadata, function md.dataset() will create a template for documentary metadata when building a data set.

Besides the definitions of metadata elements and templates, this research also designed a variable-metadata set to facilitate metadata capture. It was built and saved as an R data file in Rmdset with a name of varMDSet.

❋ ***Variable- Metadata Set***

The variable-metadata set is a data set that contains candidate variables for creating a new data set and the corresponding metadata of these variables. It was designed to capture variable level metadata automatically. Rmdset allow users to specify a variable-metadata set when creating a data set. The default is the package built-in variable metadata set varMDSet (Table 8).

In order to demonstrate managing metadata in the R system, this research picked common variables as the basic elements of varMDSet. These elements are primary or necessary variables to build a statistical data set. More elements will append to varMDSet with the development of the Rmdset in the future. It is also possible to increase metadata for further description about variables. For example, future work will consider incorporating "Relationship", the relationship between two variables or units (e.g. a formula). This exhibits the active feature of metadata, which will be introduced in Chapter 6.

Notes: 'NA' = Not Applicable

'INT (2)' = Integer of length 2 digits

'ENUM ('1', '2') = Categorical variable that can take values '1' and '2'

'CHAR (40)' = Character string of maximum length 40 characters

'seq(auto+)' = automatic increment sequence

ID	Variable	Unit	DataType	ValueDomain
1	Age	Year	INT(2)	0-100
2	Sex	NA	ENUM('1','2')	1(m) 2(f)
3	ID	NA	INT(4)	seq(auto+)
4	Treatment	NA	CHAR(1)	c(conservative) s(surgery)
5	Name	NA	CHAR(20)	NA
6	Education	Year	Int(1)	1(<=8yrs) 2(9-11) 3(12) 4(13-15) 5(>=16)
7	Address	NA	CHAR(40)	NA
8	PostCode	NA	CHAR(6)	NA
9	TelNumber	NA	CHAR(12)	NA
10	IncomeSource	NA	ENUM('1','2','3','4','5')	1(Salary) 2(PublicAssistance) 3(Retirement/Pension/Disability) 4(Other) 5(None)
11	Employ	NA	ENUM('1','2','3')	1(FullTime) 2(PartTime) 3(Unemployed)
12	Unemploy	NA	ENUM('1','2','3','4')	1(Homemaker) 2(Student) 3(Retired/Disabled)
13	MaritalStatus	NA	ENUM('1','2','3','4')	1(NeverMarried) 2(NowMarried) 3(Separated) 4(Divorced/Windowed)
14	DrugResponse	NA	ENUM('1','-1')	1(positive) -1(negative)
15	PreMedicalHistory	NA	CHAR(40)	NA

Table 8 Variable-metadata set varMDSet

In summary, all the above metadata templates as well as the variable-metadata set were prepared to assist the package functions to handle metadata. The next section, package functions, will illustrate how these data sets are applied by the functions with usage examples.

5.3 Package Functions

This research designed and built three functions for the package Rmdset, md.dataset(), md.addVar() and md.delVar(). Integrated with the R data files docmd and varMDSet build for the package, these functions implement capturing and updating metadata semi-automatically.

5.3.1 Create a Data Set With Metadata Set---md.dataset

❋ ***Function Introduction***

As discussed previously, metadata capture is a major theme of metadata management. For many reasons, data set producers provide data sets without metadata. By contrast, data users are unable to get a good understanding of the data nor utilize the data set well when lacking metadata. To improve the quality and quantity of data communication, creating a tool to facilitate metadata capture is an important task

The function md.dataset() was designed to address this issue. It allows the user to produce a data set and its metadata set simultaneously. As we defined in the section 5.2.1, Rmdset output is a data-metadata set that consist of the data set, operational (dataset and variable levels) metadata, and conceptual and documentary metadata templates. Rmdset will export and save these data-metadata sets to the user's MySQL database(s). Users then can query the data and/or metadata using SQL or tools designed as interfaces to make such queries. Figure 12 shows the flow chart of the function md.dataset.

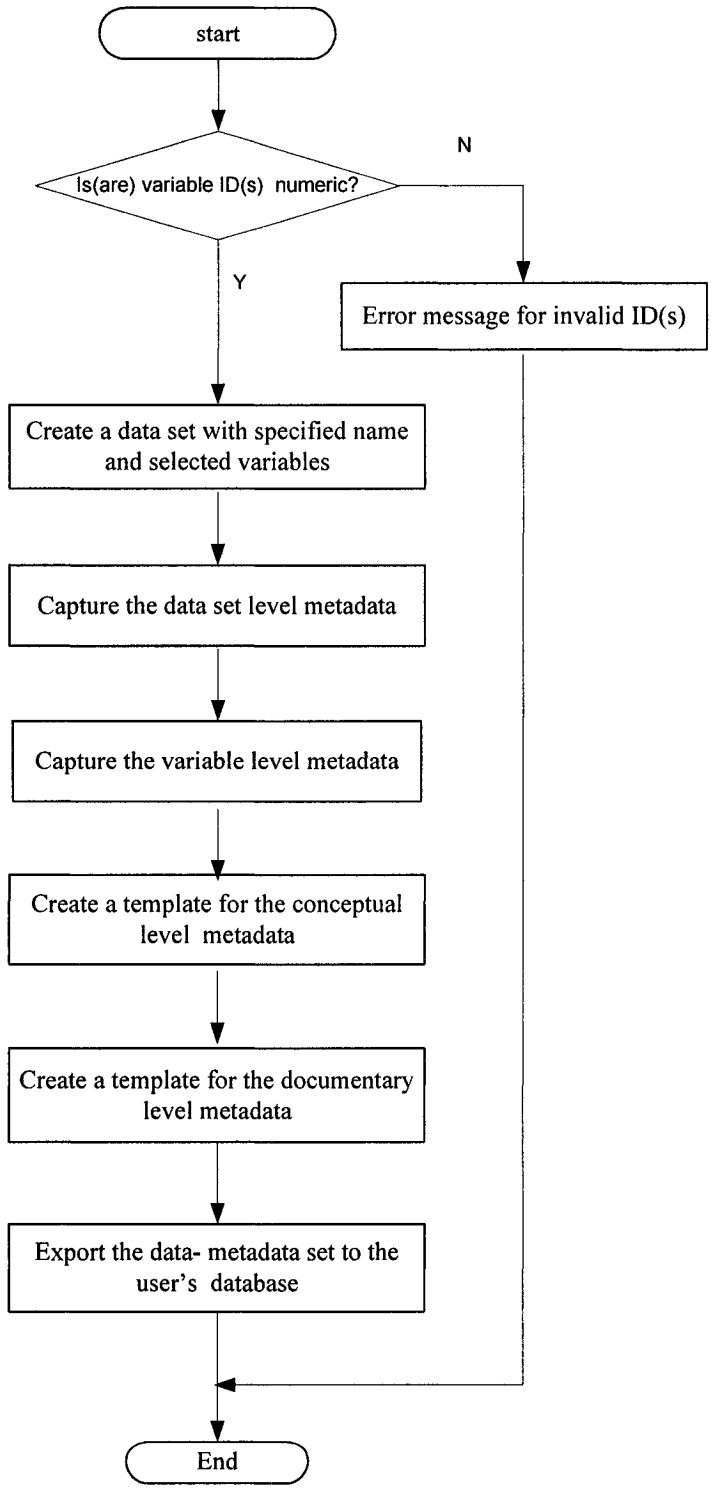

Figure 12 Flow chart of the function md.dataset

❋ ***Usage Examples***

The usage syntax of md.dataset is:

```
> md.dataset(name, conn, varmdset, ...)
```

There four arguments:

- name --- the name of the data set to be produced;
- conn --- a MySQL connection to the user's database;
- varmdset --- a variable-metadata set which has been introduced in the section 5.2. It contains candidate variables and corresponding metadata.
- ... --- IDs of the variables selected from varmdset.

As described in the previous section the ID of a variable is numeric. Therefore, users are required to provide a numeric value for the argument "...". Otherwise, an error message will display for the invalid input (see figure 13 as an example).

```
> md.dataset("test1",con,varMDSet,"a")
Error in md.dataset("test1", con, varMDSet, "a") :
    invalid id value
```

Figure 13 Invalid argument value

To use the function md.dataset to create a data-metadata set, a user can implement it in the following way.

First, load package Rmdset and specify a MySQL database. Figure 14 shows an example.

```
#Loading package Rmdest
> library(Rmdset)
Loading required package: DBI
Loading required package: RMySQL
Attaching package: 'Rmdset'


       The following object(s) are masked _by_ .GlobalEnv :
        md.addVar md.dataset md.delVar


#Specify a MySQL database
> drv <- dbDriver("MySQL")
> con <- dbConnect(drv, group="rtest")
```

Figure 14 Load Rmdset and specify a MySQL database

Second, load "varMDSet", the package built-in variable-metadata set to select variables for the data set. Once the user installed and loaded the package Rmdset, he or she can use the R function data() to load any data set saved in the package.

```
# Load "varMDSet"
> data(varMDSet)
> varMDSet
   ID      VariableName Unit              DataType                                                                                ValueDomain
1   1               Age Year                INT(3)                                                                                  0-150\r
2   2               Sex   NA         ENUM('1','2')                                                                                1(m) 2(f)\r
3   3                ID   NA                INT(4)                                                                               seq(auto+)\r
4   4         Treatment   NA               CHAR(1)                                                                 c(conservative) s(surgery)\r
5   5              Name   NA              CHAR(20)                                                                                     N\r
6   6         Education Year                Int(1)                                                  1(<=8yrs) 2(9-11) 3(12) 4(13-15) 5(>=16)\r
7   7           Address   NA              CHAR(40)                                                                                     N\r
8   8          PostCode   NA               CHAR(6)                                                                                     N\r
9   9         TelNumber   NA              CHAR(12)                                                                                     N\r
10 10      IncomeSource   NA ENUM('1','2','3','4') 1(Salary) 2(PublicAssistance) 3(Retirement/Pension/Disability) 4(Other) 5(None)\r
11 11            Employ   NA     ENUM('1','2','3')                                                  1(FullTime) 2(PartTime) 3(Unemployed)\r
12 12          Unemploy   NA ENUM('1','2','3','4')                                          1(Homemaker) 2(Student) 3(Retired/Disabled)\r
13 13     MaritalStatus   NA ENUM('1','2','3','4')                1(NeverMarried) 2(NowMarried) 3(Separated) 4(Divorced/Windowed)\r
14 14      DrugResponse   NA        ENUM('1','-1')                                                                  1(positive) -1(negative)\r
15 15 PreMedicalHistory   NA              CHAR(40)                                                                                     N\r
```

Figure 15 Loading varMDSet

Note: when R imports a dataset from a MySQL database, there is often "\r" displayed at the end of a line if the type of the last column is character. Sometimes an "NA" at the end of a line will become "N\r". Figure 15, "N\r"="NA"="Not

applicable". We believe that this behaviour is due to the data transfer between the database and R, which we are trying to understand and correct.

Third, create the data-metadata set using md.dataset(). For example, to create a sample medical trial "test" which has variables "ID", "Age" and "Treatment". The function md.dataset was designed to require the user entering the integer identifier numbers instead of typing characters, much convenient. As shown in Figure 15, the "ID" s of the selected variables in this example are "3", "1" and "4".

From Figure 16, we can see that md.dataset() produced a data-metadata set of "test", {test, test_dsmd, test_varmd, test_conmd, test_docmd}. "test_dsmd", "test_varmd", "test_conmd" and "test_docmd" represent data set, variable, conceptual and documentary level metadata of "test" respectively. R function dbListTables() will list all the tables saved in a database.

```
# Create a data-metadata set (a data set with its metadata set)
> md.dataset("test",con,varMDSet,3,1,4)

# Show tables created and saved in the user's MySQL database
> dbListTables(con)
[1] "test"       "test_conmd" "test_docmd" "test_dsmd"  "test_varmd"
```

Figure 16 The data-metadata set of "test"

The above figure shows that md.dataset() will name each table that holds metadata with a prefix representing the name of the data set and a postfix representing the level of metadata. For example, "test_conmd", "test" is the name of the data set created by md.dataset() and "conmd" means that table "test_conmd" contains the conceptual level metadata of "test". To do this will not only help users to identify different levels of metadata of a data set but also help them to distinguish different data-metadata sets save in the same database. Therefore, it supplies the

great flexibility for the user to store a data-metadata set. Users can save each data-metadata set in an independent database or save all the data-metadata set in one database. The latter case is suitable for a project with more than one data set.

Once a data-metadata set has been created, users are allowed to assign values for the data set by using the functions provided by MySQL or R. Figure 17 presents an example of applying the R function read.csv() and dbWriteTable() to assign and export data to the sample data set "test".

```
> #Import data into R
> vl <- read.csv("d:/nn-data.csv",FALSE)
>
> #Assign values to "test" and export it to the user's database
> dbWriteTable(con,"test",vl,row.names =0, overwrite =FALSE,append = TRUE)
>
> #Display the data set "test"
[1] TRUE
> dbReadTable(con,"test")
       ID Age Treatment
1       11  52         s
2       12  56         s
3       13  70         c
```

Figure 17 Display data set "test"

```
# Display the data set level metadata of "test"
> dbReadTable(con,"test_dsmd")
        Metadata                          Value
1    datesetName                        test\r
2      CreatedAt Thu Dec 08 09:42:43 2005\r
3 LastModifiedAt Thu Dec 08 09:42:43 2005\r
4       TimeZone     Eastern Standard Time\r
5   VariableName          ID,Age,Treatment\r
```

Figure 18 The data set level metadata of " test "

Figure 18 shows the data set level metadata of "test"captured automatically by the function md.dataset. Since the data set "test" has never been modified, the "CreatedAt" is same as the "LastModifiedAt".

Figure 19, shows the variable level metadata of "test". It gives users the information about the variables of the data set "test". From "test_varmd", variable level metadata of test, users are informed that variable "ID" and "Treatment" do not have unit where "NA" means "Not Applicable". "Age" is recorded by year. Moreover, "ID" is an automatic increment sequence("seq(auto+)"). Variable "Treatment" has two valid values "c" and "s" represent "conservative" and "surgery" respectively.

```
# Display the variable level metadata of "test"
> dbReadTable(con,"test_varmd")
VariableName  Unit  DataType                    ValueDomain  Comment                          At
1          ID  <NA>     INT(4)                 seq(auto+)\r created Thu Dec 08 09:42:43 2005\r
2         Age  Year     INT(3)                      0-150\r created Thu Dec 08 09:42:43 2005\r
3   Treatment  <NA>    CHAR(1) c(conservative) s(surgery)\r created Thu Dec 08 09:42:43 2005\r
```

Figure 19 The variable level metadata of "test"

In current design, md.dataset() only supply templates for conceptual level and documentary level metadata (Figure 20 and 21). Values of metadata still need to be supplied by users.

```
# Display the template of the conceptual level metadata of "test"
> res1 <- dbSendQuery(con, "describe test_conmd")
> fetch(res1)
        Field          Type Null Key Default Extra
1 ConceptName  varchar(20)  YES         <NA>
2  Definition varchar(140)  YES         <NA>
```

Figure 20 Structure of the conceptual level metadata template of "test"

Note: In the Figures 20 and 21, "NA" = Not available. Values of metadata have not yet to be entered in the table.

```
# Display the template of the documentary level metadata of "test"
> dbReadTable(con,"test_docmd")
         Metadata Value
1    DocumentName  NA\r
2        FileName  NA\r
3          Author  NA\r
4        Language  NA\r
5       DataSetID  NA\r
6       Structure  NA\r
7      Population  NA\r
8          Sample  NA\r
9        Location  NA\r
10    Description  NA\r
11        Purpose  NA\r
12       Software  NA\r
13        Quality  NA\r
14          Usage  NA\r
15         Sender  NA\r
16        Message  NA\r
17 TransferFormat  NA\r
18  StorageFormat  NA\r
```

Figure 21 The template of documentary level metadata of "test"

Because of the data-metadata set was build in R and was stored in a MySQL database, users can read the data either in R or MySQL (Figure 22)

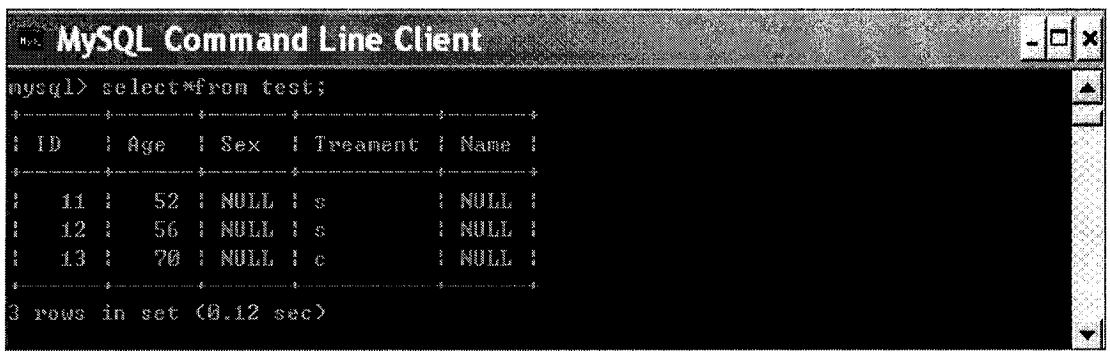

Figure 22 Data set "test" displayed in the MySQL command window

Due to the limit of the width of the MySQL command window, sometimes the

data at the end of a line will be pushed to the next line. Therefore, it is recommended to read the data in R for a nice display.

❋ ***Features Summary***

To summarize the above discussion, function md.dataset() provides users the following features:

- Produce metadata as the by-product of a data set;
- Keep a data set and the corresponding metadata set as an integrated and independent data-metadata set, {data, metadata};
- Provide flexibility of a data-metadata set storage;
- Allow the user to specify a variable-metadata set;
- Enter integer numbers instead of typing words when the user input the variable to build a data set.

5.3.2 Update Metadata While Adding a Variable---md.Addvar

❋ ***Function Introduction***

It is not unusual to modify a data set after creating it. Data producers will edit or modify the data set. They may insert a new variable or remove an existing variable for some reasons. However, there is a danger that they may not update the metadata as well. In contrast, data users always need the current version of the metadata. This makes keeping metadata up to date as important as metadata capture

To solve this problem, this research designed a function, md.addVar(), to update the metadata while adding a new variable to a data set. This function takes data set level and variable level metadata as samples to present the mechanism of automatic

metadata updating. Figure 23 shows the flow chart of md.addVar().

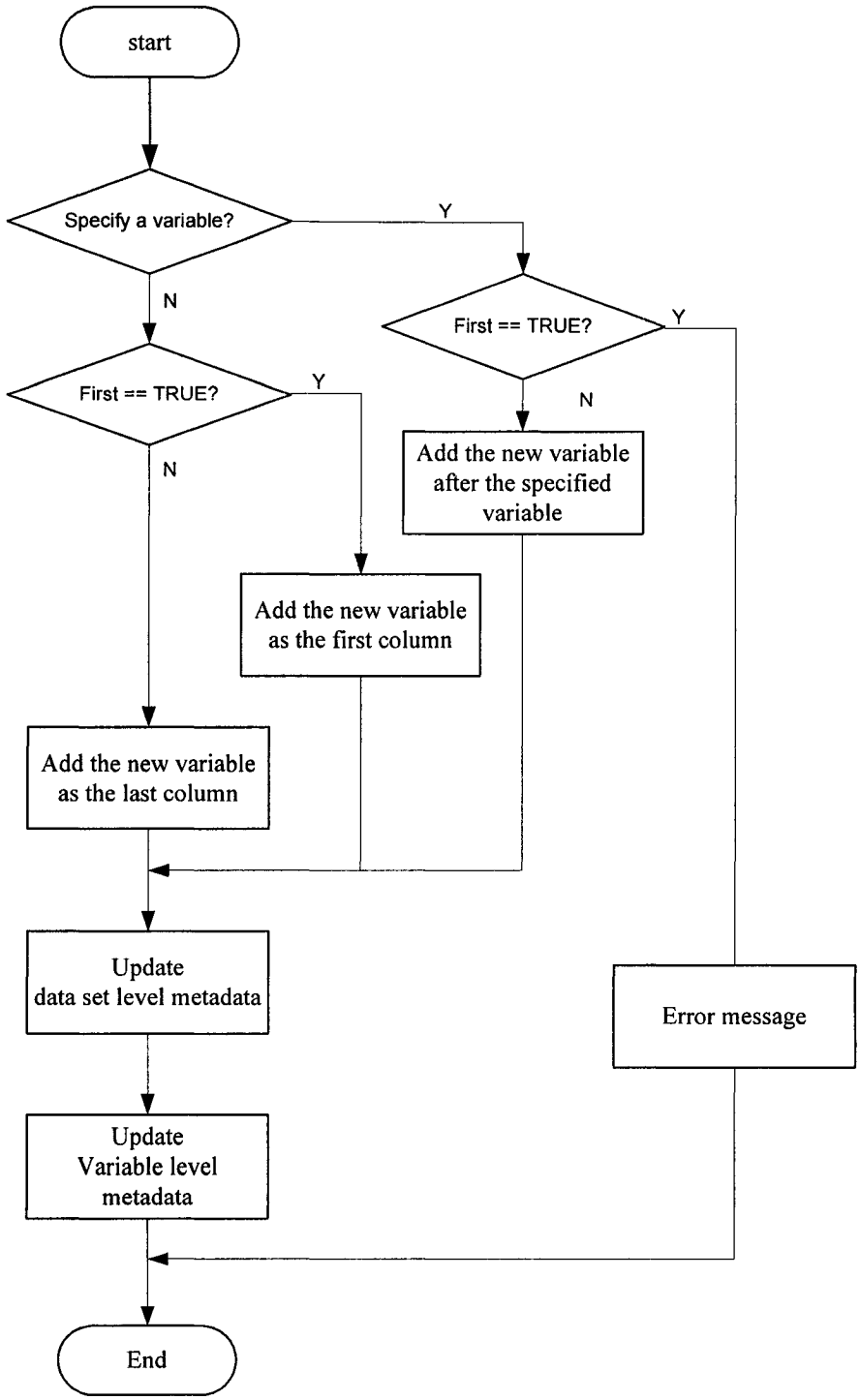

Figure 23 Flow chart of the function md.addvar

There are six arguments designed for the function md.addVar():

- conn --- a MySQL connection to the user's MySQL database;
- name --- the name of the data set that a new variable need to be added to;
- varmdset --- a variable-metadata set;
- id --- the ID of the new variable in the variable-metadata set.
- variable --- the name of the variable after which the new variable will be inserted.
- first --- a logical argument with default value of FALSE. The new variable will be added as the first column if it is TRUE.

The function md.addVar() allows users to specify the (column or ordering) position to add a new variable. The new variable can be added after an existing viable in the data set or be inserted as the first or the last column of the data set. Below we introduces these three cases and the usage of md.addVar() by editing the example data set "test" created previously.

❋ ***Usage Examples***

The usage syntax of md.addVar() is:

```
> md.addVar(conn,name,varmdset=varMDSet,id,variable,first=FALSE)
```

Example 1 illustrates how to insert a new variable "Sex" and place it after the variable "Age". Users can do this as shown in Figure 24. Since the function md.addVar() only altered the structure of the data set without assigning values, all the values of "Sex" are not available, "NA". Users can assign these values later by applying the method introduced in Figure 17.

```
# Insert the new variable "Sex" after "Age"
# "2" is the id of "Sex" in the varMDSet
md.addVar(con,"test",varMDSet,2,"Age")
[1] TRUE
>
# Show the data set "test"
> dbReadTable(con,"test")
  ID Age  Sex Treatment
1 11  52   NA         s
2 12  56   NA         s
3 13  70   NA         c
```

Figure 24 Insert the variable "Sex" after "Age"

The data set level and variable level metadata of "test" was updated automatically (Figures 25 and 26).

```
# Show the data set level metadata of "test"
> dbReadTable(con,"test_dsmd")
        Metadata                        Value
1    DatesetName                      test\r
2      CreatedAt  Thu Dec 08 09:42:43 2005\r
3 LastModifiedAt    Thu Dec 08 09:48:53 2005
4       TimeZone   Eastern Standard Time\r
5   VariableName       ID,Age,Sex,Treatment
```

Figure 25 Update the data set level metadata while adding a new variable

```
# Show the variable level metadata of "test"
> dbReadTable(con,"test_varmd")
 VariableName Unit     DataType                ValueDomain Comment                          At
1           ID   NA        INT(4)               seq(auto+)\r created Thu Dec 08 09:42:43 2005\r
2          Age Year        INT(3)                   0-150\r created Thu Dec 08 09:42:43 2005\r
3    Treatment   NA      CHAR(1) c(conservative)s(surgery)\r created Thu Dec 08 09:42:43 2005\r
4          Sex   NA ENUM('1','2')                1(m)2(f)\r   added Thu Dec 08 09:48:53 2005\r
```

Figure 26 Update the variable level metadata while adding a new variable

From Figure 26, a data user will know the variable "Sex" does not have a unit. It has two valid values "1" and "2" where "1" represents "male" and "2" represents "female". It was added at 09:48:53, December 8, 2005.

Example 2, to insert a new variable "Education" and placed it as the first column the data set "test".

```
>#  Add "Education" as the first column of data set test
> md.addVar(con,"test",varMDSet,6,first=TRUE)
[1] TRUE
>
>#  Show the data set "test" after adding "Education"
> dbReadTable(con,"test")
  Education ID Age   Sex Treatment
1        NA 11   52 <NA>         s
2        NA 12   56 <NA>         s
3        NA 13   70 <NA>         c
```

Figure 27 Add the variable "Education" as the first column of "test"

```
>#Show the data set level metadata of "test" after adding "Education"
> dbReadTable(con,"test_dsmd")
         Metadata                                Value
1     DatesetName                                test\r
2       CreatedAt          Thu Dec 08 09:42:43 2005\r
3  LastModifiedAt            Thu Dec 08 10:33:23 2005
4        TimeZone             Eastern Standard Time\r
5   VariableName     Education,ID,Age,Sex,Treatment
```

Figure 28 Update the data set level metadata while adding "Education"

```
># Show the variable level metadata of "test" after adding "Education"
> dbReadTable(con,"test_varmd")
  VariableName Unit      DataType                                ValueDomain Comment                  At
1           ID   NA       INT(4)                                    seq(auto+)\r created Thu Dec 08 09:42:43 2005\r
2          Age Year       INT(3)                                      0-150\r created Thu Dec 08 09:42:43 2005\r
3    Treatment   NA       CHAR(1)                 c(conservative) s(surgery)\r created Thu Dec 08 09:42:43 2005\r
4          Sex   NA ENUM('1','2')                                  1(m) 2(f)\r   added Thu Dec 08 09:48:53 2005\r
5    Education Year      int(1) 1(<=8yrs) 2(9-11) 3(12) 4(13-15) 5(>=16)\r   added Thu Dec 08 10:33:23 2005\r
```

Figure 29 Update the variable level metadata while adding "Education"

Example 3, to insert a new variable "Name" and placed it as the last column of the data set "test" .

```
> # Add "Name" as the last column of "test"
> md.addVar(con,"test",varMDSet,5)
[1] TRUE
>
># Show "test" after adding "Name"
> dbReadTable(con,"test")
  Education ID Age  Sex  Treatment Name
1        NA 11  52 <NA>         s <NA>
2        NA 12  56 <NA>         s <NA>
3        NA 13  70 <NA>         c <NA>
```

Figure 30 Add the variable "Name" as the last column of "test"

```
> # Show the data set level metadata of "test" after adding "Name"
> dbReadTable(con,"test_dsmd")
        Metadata                                  Value
1    DatesetName                                test\r
2      CreatedAt           Thu Dec 08 09:42:43 2005\r
3 LastModifiedAt             Thu Dec 08 10:45:38 2005
4       TimeZone               Eastern Standard Time\r
5   VariableName  Education,ID,Age,Sex,Treatment,Name
```

Figure 31 Update the data set level metadata while adding "Name"

```
>#  Show the variable level metadata of "test" after adding "Name"
> dbReadTable(con,"test_varmd")
  VariableName Unit      DataType                                ValueDomain Comment                 At
1           ID   NA       INT(4)                                  seq(auto+)\r created Thu Dec 08 09:42:43 2005\r
2          Age Year       INT(3)                                      0-150\r created Thu Dec 08 09:42:43 2005\r
3    Treatment   NA       CHAR(1)                 c(conservative) s(surgery)\r created Thu Dec 08 09:42:43 2005\r
4          Sex   NA ENUM('1','2')                                    1(m) 2(f)\r   added Thu Dec 08 09:48:53 2005\r
5    Education Year         Int(1) 1(<=8yrs) 2(9-11) 3(12) 4(13-15) 5(>=16)\r   added Thu Dec 08 10:33:23 2005\r
6         Name   NA     CHAR(20)                                              N\   added Thu Dec 08 10:45:38 2005\r
```

Figure 32 Update the variable level metadata while adding "Name"

If a variable was assigned to two positions, an error message will display on the screen .

```
> md.addVar(con,"test",varMDSet,2,"Age",TRUE)
Error in md.addVar(con, "test", varMDSet, 2, "Age", TRUE) :
    can not assign two positions for a variable
```

Figure 33 Assign two positions for a variable

An error message will also be displayed if we try to add a variable that already exists in the data set .

```
#Show the error message for duplicate variables
> md.addVar(con,"test",varMDSet,5)
Error in mysqlExecStatement(conn, statement, ...) :
    RS-DBI driver: (could not run statement: Duplicate column name 'Name')
```

Figure 34 Show the error message for duplicate variables

❋ ***Summary***

To summarize the above, the function add.mdVar() allows users

- To update data set level and variable level metadata while adding a new variable to a date set
- To specify where to insert the new variable in a data set
- To record the date and time that the new variable was added into the data set

5.3.3 Update Metadata While Deleting a Variable---md.Delvar

❋ ***Function Introduction***

Comparing with md.addVar(), function md.delVar() was designed to update the metadata when users delete an existing variable from a data set. Figure 31 shows the flow chart of the function md.delVar().

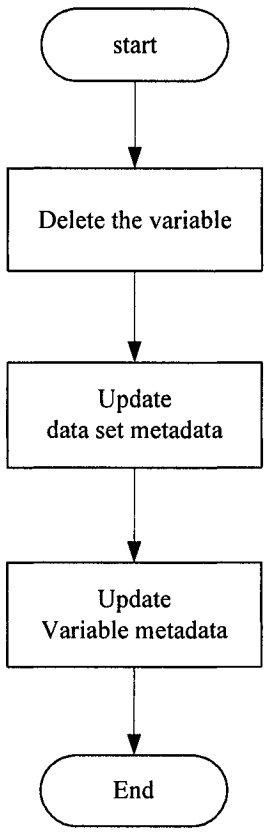

Figure 35 Flow chart of the function del.mdVar

The function md.addVar only has three arguments:

- conn --- a MySQL connection to the user's MySQL database;
- name --- the name of the data set that one of its variables will be removed;
- variable --- name of the variable which will be removed.

❋ ***Usage example***

The usage syntax of md.addVar is:

```
> md.addVar(conn,name,variable)
```

For example, to delete the variable "Education" from the data set "test".

```
# Delete the variable "Education" from "test"
> md.delVar(con,"test","Education")
<MySQLResult:(3068,0,23)>

# Show the data set "test" after deleting "Education"
> dbReadTable(con,"test")
  ID Age Sex Treatment Name
1 11  52  NA         s <NA>
2 12  56  NA         s <NA>
3 13  70  NA         c <NA>
```

Figure 36 Delete the variable "Education " from the data set test

The data set level metadata was updated automatically by the function md.delVar() (Figure 37).

```
># Show the data set level metadata of "test" after deleting the variable
"Education"
> dbReadTable(con,"test_dsmd")
        Metadata                          Value
1     DatesetName                       test\r
2       CreatedAt Thu Dec 08 09:42:43 2005\r
3  LastModifiedAt    Fri Dec 09 11:35:45 2005
4        TimeZone     Eastern Standard Time\r
5   VariableName   ID,Age,Sex,Treatment,Name
```

Figure 37 Updating data set level metadata while deleting "Education"

The record of the deleted variable will not be removed from the variable level metadata. It will be marked as "deleted" to tell the data user if it ever existed in the data set. Function md.delVar() will also record when this variable was deleted.

```
># Show the variable level metedata of "test" after adding "Name"
> dbReadTable(con,"test_varmd")
  VariableName Unit     DataType                                ValueDomain Comment                          At
1           ID   NA      INT(4)                                  seq(auto+)\r created Thu Dec 08 09:42:43 2005\r
2          Age Year      INT(3)                                      0-150\r created Thu Dec 08 09:42:43 2005\r
3    Treatment   NA      CHAR(1)                 c(conservative) s(surgery)\r created Thu Dec 08 09:42:43 2005\r
4          Sex   NA ENUM('1','2')                                 1(m) 2(f)\r   added Thu Dec 08 09:48:53 2005\r
5    Education Year        Int(1) 1(<=8yrs) 2(9-11) 3(12) 4(13-15) 5(>=16)\r Deleted Fri Dec 09 11:35:45 2005
6         Name   NA   CHAR(20)                                            N\  added Thu Dec 08 10:45:38 2005\r
```

Figure 38 Updating variable level metadata while deleting "Education"

Therefore md.delVar() provides users with a function to remove a variable without losing the important metadata of the data set.

Chapter 6 **Road Map for Development**

The main targets of this thesis are to present an overview of the issues posed by metadata in statistical contexts and to review the possible mechanisms that could be used to build metadata tools using R as an illustrative platform. The preceding chapter has already illustrated how the above objectives were achieved by the package Rmdset. In the long term, an integrated R-metadata system should be built and the package Rmdset can be regarded as an embryo of such a system.

To build an integrated R-metadata system based on the R package Rmdset, much work still needs to be done. This chapter introduces the proposed model of an integrated R-metadata system and discusses the mechanism to implement each important task to handle metadata. From metadata capture, storage and exchange to metadata dissemination, from the updating of the package Rmdset and incorporation of XML techniques to a Web-based GUI, this chapter provides a road map for developing an integrated R-metadata system.

6.1 An Integrated R-Metadata System

Taking the long view, a system that combines all aspects of metadata is desirable. That is, we want an integrated metadata system which covers not only data but also processes.

6.1.1 Features

Based on Kent and Schuerhoof (1997), an integrated metadata system should have following features:

- ***All Aspects of Metadata Coverage***

It is difficult to define in advance of real-world usage what metadata will be included

in or left out of the system. Metadata should be considered in a broad way. A metadata system should allow for the integration of as many aspects of metadata as possible, conceptual, operational, logistic and documentary.

- ***Defined As Operational***

Metadata should be designed for multiple usages. Besides documentation, metadata should facilitate the implementation of other operations, some of which may not be clear at the design stage.

- ***Object-Oriented***

Object-oriented design should play an important role in metadata management. Here the object-oriented does not mean the application of an object-oriented language or object-oriented database management system. Rather, we want to define metadata objects that combine information and behaviour to represent the resources, data and processes in a system.

- ***Combining Systems and Tools***

It is not easy to cover all aspects of metadata in one system. The final design should be a loose combination of other systems and tools that are built using a consistent set of metadata definitions or principles.

From the view of this research, the ultimate design should integrate several R-metadata packages with other metadata systems and/or tools that can run in the R system. These packages and tools constitute an integrated R-metadata tools suite. We now introduce a proposed design of the tools suite based upon the model provided by Rmdset.

6.1.2 Suggested Model

The structure of an integrated R-metadata system could be designed as the model

shown in Figure 39 This model also shows the work flow of the whole procedure for managing metadata. Each part of work will be performed by at least one metadata tool according to the uniform metadata definition.

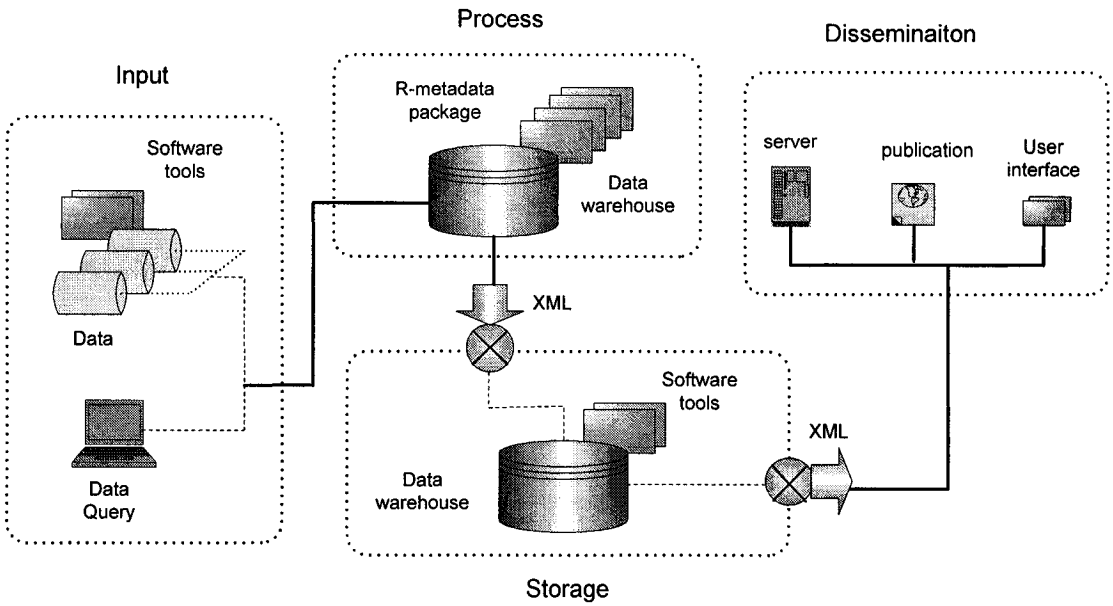

Figure 39 Model of an integrated R-metadata system

❋ ***The Input Model***

In the above system, users are required to provide the necessary data information to produce a data set and its corresponding metadata. This metadata could be a data file imported from an external system or the required arguments might be input by users. The system should also accept users' queries for data and/or metadata. Comparing this with the package Rmdset, we may no longer need users to provide (all the) metadata. Indeed, in some applications, we may be able to capture all levels of the metadata automatically using tools in our system.

❋ ***The Process Model***

The process model is the core part of the system. It provides an environment to perform data set and metadata production, storage and transformation. It manages

all the metadata in the system including system product i.e., output, metadata, system relationship metadata and system activity metadata. The functionality of the model is implemented by an R-metadata package; that is, we wish to create an R package based upon Rmdset.

The process model is also the logical centre of the system. It uses the input model to implement input validation and accept valid values of data and metadata. Furthermore, it links to the storage model to transfer metadata in XML format for data storage. Finally, it uses an output model to prepare data for dissemination via the storage model.

The data in the process model include data prepared for software tools, data and metadata generated in the production process and metadata for system administration. Among these only the data intended for dissemination can be accessed and retrieved by regular users. However, the "owner" of the R-metadata system will be able to query system internals.

- ***The Storage Model***

The storage model tells us how to store data and metadata and how to prepare it for dissemination. All the metadata in this model have the same type as in the process model so that these two models can work seamlessly together to form a coherent metadata management and storage system. All the data in the storage model are in the format of XML and can be transformed to several data formats as required for dissemination. The mechanism to store and transform metadata will be discussed in the section 6.2 "XML and Metadata".

- ***The Dissemination Model***

A modern metadata system should facilitate metadata publication and metadata discovery. The process of publishing metadata may include activities such as posting information to a server or printing data in a magazine. Users may be able to

find metadata by local query or web browsing.

❋ ***Graphical User Interface***

R also allows communication with a web server by invoking batch mode scripts. In research by Professor Nash and his student Wang (2004), the Common Gateway Interface (CGI) was applied to implement the communication mechanism.

This thesis does not provide such a user interface to the package Rmdset because Rmdset was created to establish the possibility for handling metadata in R, the main goal of the research, and the web-interface would demand a considerable body of additional program code. Indeed, future development will require many functions and linkages to be added to the package. Once created, however, applying these new functions to the package should be very simple, since the user only needs to install a newer version of the package, a task that is generally trivial in R because there is already an install-package function. Thus, if we build a GUI for Rmdset, and by necessity update the functionality of the interface, we require only that the user install and learn the usage of the new interface, much like installing and using any other R function.

However, the first stage is to construct the complete R-metadata system discussed in this chapter. Thereafter, a web-based GUI could be built to simplify the usage of the system.

6.1.3 Generic Functionality

According to the required features of a metadata system and the structure of R system, an integrated R-metadata system should be designed and built with following functionality:

- to capture and update metadata automatically;

- to perform advanced metadata editing;
- be able to compare proposed new metadata with that already entered;
- be able to avoid duplicate metadata definitions;
- be able to approve the reuse of existing metadata;
- allow the storage of different versions of metadata;
- allow queries of any version of metadata;
- allow queries of any level of metadata;
- to transmit metadata from the process system for storage or dissemination.

6.2 Improving Rmdset

6.2.1 Active Metadata

The next version of Rmdset should take active metadata into account. That is, metadata should not only supply information to users but also support activities to control statistical processes. An active feature of metadata will help the system to do things such as:

- record all the possible units of a variable in the data set;
- convert the variable values automatically when the measure scale is altered;
- record the formula if a variable is the result of a computation using other variable(s) in the data set;
- use recorded formula(s) and given variable(s) to a yield new column;
- use computed formula(s) to produce a unit for the resulting variable;

- adapt scale factors automatically if the units of one of variables in the computation is changed [Kent and Schuerhoof, 1997].

The active feature of metadata should be considered as an important part of future metadata system design. This means metadata should be defined as operational.

6.2.2 New Functions

To implement the functionality of the process model of the proposed R-metadata system, an R-metadata package should be created by modifying and improving Rmdset as follows.

First, create a dictionary of documentary level metadata based on the statistical metadata standards. This will contain the metadata elements and definitions. These elements are divided into two classes, mandatory and optional.

Second, in order to collect conceptual and documentary metadata automatically comes true write functions:

- addDocmd() to allow the user to add optional documentary metadata selected from the metadata dictionary;
- editDocmd() to allow the user to edit the documentary level metadata of the data set;
- addConmd() and editConmd() to allow the user to add and edit conceptual level metadata.

Third, add function md.editVar() to allow the user to edit a variable and update the corresponding metadata of a data set.

Finally, modify function md.dataset() to allow users to:

- create templates for conceptual and documentary metadata with appropriate formats using built-in templates;
- choose built-in or custom user templates for conceptual and documentary metadata.

R is not static. Along with the development of R, there are also R plug-in and related tools and packages like XML, StatDataML, and our own R-metadata system, all of which will need to be modified and improved to provide continuing utility for the R-metadata package, especially in relation to its process model.

6.3 XML and Metadata

The R-metadata system discussed in the preceding section proposes taking the advantage of XML for metadata storage and exchange. This section gives the introduction, analysis and architecture of the mechanism.

6.3.1 Why XML?

❋ ***What is XML ?***

From the XML 1.0 specification:

> "XML--- Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents."

In an XML text, each data item is "wrapped" by start and end tags. These tags distinguish the text in a document as data and metadata. With its specific structure and the way to describe the structure of data, XML is well-suited for metadata storage and metadata exchange.

❋ ***XML for Metadata Storage and Exchange***

In an XML document complex structures are represented as linear text. This provides a simple solution to overcome the critical obstacle to exchange metadata effectively [Westlake and Nelson, 1999]. Note that the Rmdset functions output text, so that they could, in principle, be easily adapted for XML output.

The following are several of the reasons why XML can offers advantages for metadata storage and exchange:

- XML has a naturally hierarchical structure which allows it to describe any set of structured data;
- XML can exchange data items along XML tags that describe their meaning usage and its relationships;
- An XML representation can be visible on screen or transformed to other format by the Extensible Stylesheet Language (XSL)[www.w3.org/TR/xsl/];
- Applications can deal with XML in memory with the Document Object Model (DOM) [www.w3.org/DOM/] or in sequence with the Simple API for XML (SAX) [www.saxproject.org/]. They are both free Application Programming Interfaces (APIs);
- An XML-based metadata exchange architecture can be led naturally by XML processing [Westlake and Nelson, 1999].

6.3.2 Generic Architecture

An XML-based metadata exchange architecture should include the following:

- Metadata storage --- holds all the metadata for exchange;
- Extract processes --- preparation procedures for metadata translations. Usually XML tags will be added to data items according to a specific metamodel of an XML exchange tool;
- A translation processes --- translate metadata from extracted processes to compatible with the common metamodel of a general metadata system;
- A metadata staging area --- gets metadata ready for exchange. It stores and manages metadata in a queue with uniform format;
- holds uniform, delimited metadata in a common tagged order and format and passes it to the metadata exchange mechanism;
- XML parser --- makes metadata available for an application or a programming language by removing the XML tags from the translated metadata. For example, Xerces XML Parser [xerces.apache.org/xerces-j/] and MSXML Parser [msdn.miscrosoft.com/xml/].

Figure 40 shows an example of a metadata exchange architecture, which uses XML as a metadata exchange mechanism. In this example, metadata from diverse sources will be extracted and then translated to XML delimited files which conforms to a metamodel. These files have the standard XML tags and instances. XML tags will be removed when passing the XML parser. The untagged metadata will be delivered for storage or retrieving [Tannenbaum, 2002].

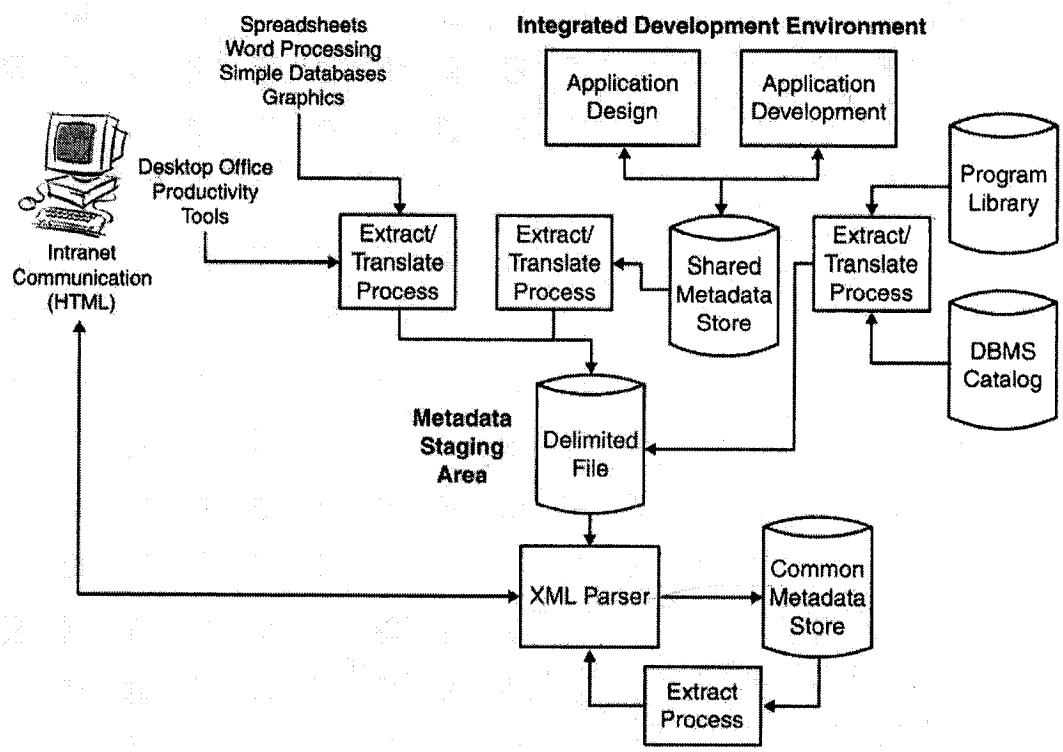

Figure 40 An example XML-based metadata exchange architecture

6.3.3 R-XML Packages

There are two XML-related packages available for users on CRAN, XML and StatDataML.

XML was developed by the Omegahat project (www.omegahat.org) with the functionality to help R to:

* parse XML files, URLs and strings, using either the DOM (Document Object Model)/tree-based approach, or the event-driven SAX (Simple API for XML) mechanism;
* generate XML content to buffers, files, URLs, and internal XML trees;
* read DTDs as R objects [Debian-r-cran-xml].

The mechanism of parsing XML is either the Document Object Model (DOM) which is tree-based, or the Simple API for XML (SAX), which is event-based. The tree-based parser reads the whole XML document and creates a tree structure in memory. To process the elements of the XML document, an application will navigate this tree and create a user-level representation of the nodes. The event-based will not construct a tree. Instead, it reads the XML elements serially and provides handler to deal with different elements.

Package XML offers two functions for these two types of XML parsers. Function xmlTreeParse(), supports tree-based parsing. It maps XML document into an internal tree and processes it and returns it as a list of nodes to R. It commonly uses for standard situations and easy to understand. Function xmlEventParse() supports event-driven based parsing. It is usually used to handle very large data files [http://www.omegahat.org/RSXML].

The package StatDataML is intended as a standard tool for statistical data exchange. It uses and supports an XML-based markup language. As such, it defines the XML rules and elements which cover the needs of most statisticians' data. StatDataML files are XML files which can be described with a Document Type Definition (DTD[10]) [Meyer, Leisch, Hothorn and Hornik, 2002].

[10] http://www.w3schools.com/dtd/default.asp

```
R : Copyright 2003, The R Development Core Team
Version 1.6.2  (2003-01-10)

## load the StatDataML package
> library(StatDataML)
Loading required package: XML

## load the iris data
> data(iris)

## show the first observation
> iris[1,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa

## write StatDataML file
> writeSDML(iris, file = "iris.sdml")
```

Figure 41 Write a StatDataML file

```
                        < M A T L A B >
                Copyright 1984-2000 The MathWorks, Inc.
                     Version 6.0.0.88 Release 12

>> path(path, 'StatDataML')
>> x = readsdml('iris.sdml', 'list.as.structarray')

>> x(1)

ans =

    Sepal.Length: 5.1000
     Sepal.Width: 3.5000
    Petal.Length: 1.4000
     Petal.Width: 0.2000
         Species: {'setosa'}

>> load durer
>> whos X, c = cellstr(caption)

  Name      Size            Bytes  Class
  X       648x509         2638656  double array

c =

'Albrecht Durer's Melancolia.'
'Can you find the matrix?'

>> writesdml(X, 'durer.sdml', 'TITLE', c{1}, 'TEXTDATA', 'TRUE')
```

Figure 42 Read a StatDataML file in MATLAB

In the present design, StatDataML has two user functions readSDML() and writeSDML() that are designed to read and write a StatDataML file (Figure 41 and 42). The file is given with a filename extension .sdml. Currently, StatDataML supports exchanging data between R and MATLAB (Figure 42) [StatDataML R Document, 2003]. Moreover, the package is still under development, and it is expected that more functions (for example to describe data with an XML Schema, XML based alternative to DTD [www.w3schools.com/xml/xml_dtd.asp]) will be created in the future to facilitate metadata exchange between R and other systems.

6.4 Metadata Dissemination

Metadata dissemination is the process of making metadata available to others. Metadata could be discovered by searching, browsing or importing based on different published approaches.

- ***Searching Metadata***

The metadata system provides a server application that enables users to search and retrieve metadata over a network, either a localhost, an intranet or the Internet. The implementation of such a server requires a network protocol, metadata standards and a search engine.

- ***Browsing Metadata***

The metadata system generates a series of Web pages (e.g. in HTML format) to help users to navigate metadata records of specific interest. To organize metadata records, a hierarchical categorization is often a useful solution.

- ***Exporting Metadata***

Metadata will be available to users in a number of formats by being exported based on particular metadata standards defined or applied by the metadata system. XML based metadata publication is one of the long-term solutions. The XML files can be

converted into other formats like HTML, or PDF for websites or printing.

PubStat of OECD (Organisation for Economic Co-operation and Development) and CoSSI of Statistical Finland are two XML based metadata dissemination models under construction. With the development of R and XML, users will expect to incorporate such kind of system into the integrate R-metadata tools suite.

6.5 Summary

According to the previous discussion, an integrated R-metadata system can be developed in steps:

- further exploration and expansion of the package Rmdset;
- continued development of the R-XML packages;
- incorporation of metadata dissemination tools into R.

These steps are independent and without a specific ordering. Packages and tools could be developed at the same time and independently. The R-metadata tools suite will be a loose combination of these packages and tools, sharing the same metadata principles.

Chapter 7 **Conclusions and Contributions**

7.1 Conclusions

The key issue for this research is handling metadata in statistical and other databases. We attempted to bring metadata into a popular and powerful statistical system to facilitate data analysis and metadata management. Therefore, we took R as the example system to create metadata tools for statistical and other kinds of data. Our aim was not to build a complete metadata system, which would be a very large enterprise, but instead to present the general ideas and to explore a feasible approach for metadata management.

Incorporating the diverse themes of metadata into the complex R system has not been easy. On several occasions, it was not clear if success in a proof-of-concept would be achieved. However, by reviewing common problems posed by metadata and creating a prototype R-metadata package, we learned and summarized some metadata principles that should be helpful for further research. Note, however, that the prototype system already has some usable functionality. The metadata principles that we wish to emphasize are:

- Capture metadata while creating a data set as much as possible

The variable-metadata set varMDSet (that is, the set of data and metadata for our variables) is a specific data set created using this research. It contains candidate variables and their corresponding metadata. If users use the package function md.dataset() to select one or more variables from varMDSet to build a data set, they will get the variable level metadata captured to company the data set. Moreover, the package Rmdset allows users to define all their variable-metadata set based on the same data structure as varMDSet. As a result, varMDSet may aid future work

in automatic metadata collection.

- Capture and/or update metadata with minimal manual intervention and only one time if possible

Metadata templates and functions designed and implemented for the package Rmdset assist the production and updating of metadata. These will help users capture or update some levels of metadata automatically and/or create templates for metadata. Metadata creation and updating are no longer extra work for data providers.

- Observe metadata from as broad a perspective as possible

To cover all the aspects of metadata, a future design will involve the active feature of the operational level metadata. Active metadata contains information such as the relationship between two columns in a data set (e.g. a formula). Taking advantage of active metadata allows us to control some aspects of the statistical processing of data automatically.

- Define metadata as an object that combines data information and behaviour

To apply the object-oriented approach in the future designs, we need to define metadata as an object, which will not only provide documentation to users but also perform some tasks such as computations, selections and making decisions (i.e. methods as well as data).

- Integrate tools and systems built based on the same metadata principles instead of intending to cover all metadata facet in one system.

Further development of Rmdset and integration with other existing R metadata-related packages may shorten and smooth the effort for constructing a comprehensive R metadata system.

7.2 Contributions of This Research

- Bridges the gap between R and metadata

This research not only adds a new type of data (metadata) to R but also provides an embryo of an R-metadata system. We defined metadata in different levels and created templates to help produce metadata as the by-product of data production.

- Benefits both data producers and data users

The package Rmdset achieves metadata capture and updating (semi-) automatically. This decreases the burden on the metadata providers and increases the benefit to data users. The classification of metadata and the definition of metadata elements and templates facilitate standardization in handling metadata.

- Provides documentation and examples in programming with R

R documents such as description files, source code, usage examples and help pages included in the package provide examples for adding functions to R. In addition, documentation about building R package under WindowsXP will let other developers, especially those with limited knowledge of R, avoid some of the frustrations of building metadata capabilities. Note that the package should run on other platforms such as Linux or Macintosh, since R is a cross-platform system. However, there are often subtle differences, and we have not tested the tools on other than the Windows platform.

- Provides guidelines for developing an integrated R-metadata system

This thesis offers the structure and methodology to build an integrated R-metadata tools suite. It indicates directions to keep research into handling metadata in R moving forward.

References

Apache XML project (2005), *Xerces Java Parser Readme*, viewed at 2006-07-11 from http://xerces.apache.org/xerces-j/

Auth, G. and Maur, E. V. (2002), A Software Architecture for XML-Based Metadata Interchange in Data Warehouse Systems, EDBT2002 Workshop Revised Paper

Bethlehem, J. (2003), *A Typological Introduction to Metadata Models,* viewed at 2003-10-09 from http://www.epros.ed.ac.uk/metanet/deliverables

Bi, Y.and Murtagh, F. (1998), *The Roles of Statistical Metadata and XML in Structuring and Retrieving Statistical Information,* viewed at 2003-09-28 from http://europa.eu.int/en/comm/eurostat/research/conferences

Chen, T. (2005), *Build R package for Win2000/XP,* viewed at 2005-11-19 from http://www.stat.nctu.edu.tw/MISG/SUmmer_Course/C_language/Ch14/BuildR

Curran, M. (1999), *Metadata an Introduction and Overview,* University of Ottawa CASLIS Ottawa Chapter Meeting

Debian-r-cran-xml (2006), *Package: r-cran-xml (0.99-7-1),* initially viewed in an earlier version at 2003-10-09 from http://packages.debian.org/stable/math/r-cran-xml

Durinck, S. Gentlman, R. and Dudoit, S. (2003), *BioConductor,* viewed at 2005-06-13 from http://www.bioinformatics.org/NETTAB

Elroi, D. (1998), *Can You Live without Metadata,* viewed at 2003-06-06 from http://www.elroi.com/ppts%20eletronic%20permit%2097

Fairgrieve J. and Brannen K.(1998), *Idaresa-a Tool for Construction, Description and Use of Harmonised Datasets from National Surveys,* Proceedings in Computational

Statistics, 13th Symposium held in Bristol, Great Britain, 1998, Physica-Verlag Heidelburg New York.

Gillman, D.W. and Appel, M. V. (1997), *The Statistical Metadata Repository:An Electronic Catalog of Survey Descriptions,* http://iassistdata.org/publications/iq, viewed at 2003-10-20

Helbig, M., Urbanek, S. and Theus, M. (2004), *JGR Presentation: A Unified Interface to R, h*ttp://stats.math.uni-augsburg.de/JGR/, viewed at 2005-05-10

Hornik, K. (2005), *Frequently Asked Questions on R,* http://cran.r-project.org/doc, viewed at 2005-11-01

IES(1995), *The Enterprise Newsletter Issue No 5-Metadata and XML for Business,* http://members.ozemail.com.au/~ieinfo, viewed at 2003-10-20

Inter-university consortium for political and social research (2003), *Treatment Episode Data Set (TEDS),* http://search.icpsr.umich.edu/SAMHDA/, viewed at 2005-11-12

Jeffery, K. G. (1999) *METADATA: the Future of Information Systems,* viewed at 2003-04-12 from http://www.wmo.ch/web/www/WDM/ET-IDM

Kent, J.P., Schuerhoff, M. (1997), *Some Thoughts about a Metadata Management System,* Proceedings of the Ninth International Conference on Scientific and Statistical Database Systems, Olympia, Washington, also at http://csdl2.computer.org/persagen/DLAbs-Toc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/ssdbm/1997/7952/00/7952toc.xml&DOI=10.1109/SSDM.1997.621184

Kent, J., Westlake, A. and Nelson, C. (1999), *MetaNet Deliverable D4, Metadata Work Package 1: Methodologh and Tools, MetaNet,* viewed at 2003-10-05 from http://www.epros.ed.ac.uk/metanet/deliverables

Lamb, J.M. and Smart, C. (2000), *Simultaneous Analysis of Heterogeneous Databases on the Web: The ADDSIA Project in Recent Developments and Applications in Decision Making,* http://webfarm.jrc.cec.eu.int/ETK-NTTS/Papers/final_papers/en187.pdf, view at 2005-09-02

Lang, D. (2005), *R & SPlus XML Parsers,* initially viewed in an earlier at 2003-10-09 from http://www.omegahat.org/RSXML,

Laplant,W., Lestina, G., Gillman, D. and Appel, M. (1996), *Proposal for a Statistical Metadata Standard*, http://www.census.gov/prod/2/gen/96arc, viewed at 2005-02-19

Lenz, H.-J. (1994), *The Conceptual Schema and External Schemata of Metadatabases*, Charlottesville, VA

Linnerud, J. (2003), *A Training Manual for the Adoption of Metadata Systems and Standards,* http://www.epros.ed.ac.uk/metanet/deliverables, viewed at 2005-02-10

Metadata Education Project (2002), *What is Metadata*, viewed at 2003-09-06 from http://www.sdvc.uwyo.edu/metadata/what.html#mat

MetaNet WG2 (1999), *The concept of metadata: A report on the nature of metadata and how these concepts can be used in practice*, viewed at 2003-10-20 from http://www.epros.ed.ac.uk/metanet/deliverables

Meyer D, Leisch, F., Hothorn, T. and Hornik, K. (2002), *StatDataML–An XML Format for Statistical Data*, http://www.wu-wien.ac.at/, viewed at 2003-10-09

MSDN, XML Developer Center (2006), http://msdn.microsoft.com/xml/default.aspx, viewed at 2006-07-11

NOAA (2002), *What is Metadata*, http://www.csc.noaa.govmetadata, viewed at 2003-04-06

NOAA (2002), *Why Metadata is Important*, http://www.sdvc.uwyo.edu/metdata, viewed at 2003-04-06

Nunes, M. (2005), *Creating R Packages*, http://www.maths.bris.ac.uk/~maman, viewed at 2005-08-10

Pierre, M. St. and Restivo, J. (1998), *An Integrated Tool Suite for Managing Metadata*, 1998: Proceedings of the Earth Observation&Geo-Spatial Web and Internet Workshop'98

R Development Core Team (2003), *An Introduction to R*, also at http://www.r-project.org/

R Development Core Team (2003), *R Data Import/Export*, also at http://www.r-project.org/

R Development Core Team (2003), R Installation and Administration, also at http://www.r-project.org/

R Development Core Team (2005), *R Language Definition (version 2.1.1)*, also at http://www.r-project.org/

R Development Core Team (2005), *Writing R Extensions (version 2.1.1)*, also at http://www.r-project.org/

R Home Page (2003), *The R Project for Statistical Computing*, viewed at 2003-04-03 from http://www.r-project.org/

R Home Page (2003): *The R Project for Statistical Computing,* viewed at 2003-04-20 from http://www.r-project.org/

Rossi, P. (2005), *Making R Packages Under Windows: A Tutorial*, viewed at 2005-08 from http://gsbwww.uchicago.edu/fac/peter.rossi/research

R package StatDataML Document (2003), *StatDataML: An XML Format for Statistical Data,* also at http://www.r-project.org/

SAX (2006), *About SAX*, http://www.saxproject.org/, viewed at 2006-07-11

Sundgren, B. (1973), A*n Infological Approach to Data Bases*, Stockholm 1973

Sundgren, B. (1973), *An Information Systems Architecture for National and International Statistical Organizations,* Methodological report, draft

Sundgren, B. (2003), *Developing and Implementing Statistical Metadata Systems,* viewed at 2003-10-25 from http://www.epros.ed.ac.uk/metanet/deliverables

Sutherland, M. (2005), *Building R for Windows*, viewed at 2005-11-19 from http://www.murdoch-sutherland.com/Rtools/

Sutherland, M. (2005), *Using MiKTeX with R for Windows,* viewed at 2005-11-19 from http://www.murdoch-sutherland.com/Rtools/miktex.html

Tannenbaum, A. (2002), *Metadata Solutions: Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand*, Addison-Wesley

The GNU Operating System (2006), http://www.gnu.org/ viewed at 2006-06-15

The World Wide Web Consortium (W3C), *Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation 04 February 2004,* viewed at 2005-01-06 from http://www.w3.org/TR/REC-xml/

Ullman J.D. (1982), *Principles of Database Systems, 2nd edition*, Rockville.Md.: Computer Science Press.

United Nation (1995), *Guidelines for the Modeling of Statistical Data and Metadat*a, http://www.unece.org/stats/publications, viewed at 2005-02-01

W3C, Architecture domain (2005), *Document Object Model (DOM)*, viewed at 2006-07-11 from http://www.w3.org/DOM

W3C Recommendation (2001), *Extensible Stylesheet Language (XSL) Version 1.0,* viewed at 2006-06-12 from http://www.w3.org/TR/xsl/

W 3 Schools (2006), *DTD Tutorial*, http://www.w3schools.com/dtd/default.asp, viewed at 2006-07-11

W 3 Schools (2000), *Simple Object Access Protocol (SOAP) 1.1*, viewed at 2005- 01-16 from http://www.w3.org/TR/soap/

W 3 Schools (2006), *XML Validation*, http:// www.w3schools.com/xml/xml_dtd.asp, viewed at 2006-07-11

Woodley, M. S. (2003), *DCMI (Dublin Core Metadata Initiative) Glossary,* viewed at 2003-11-12 from http://dublincore.org/documents

Yan, J., Rossini, A.J. (2003), *Building Microsoft Windows Version of R and R Packages under Intel Linux*, R News 2003

Appendices

A. Package Rmdset Configuration and Installation

Following lists the detail of the configuration when we developed the package Rmdset:

- Operating System: Windows XP
- Statistical System: R 2.1.1
- Database Server: MySQL 4.1.
- R Database Packages: DBI 0.1-6
  RMySQL 0.5-6

To install package Rmdset, these steps should be followed:

1. Install R system, DBI and RMySQL packages

Package Rmdset requires the R plug-in packages DBI 0.1-6 to provide a DataBase Interface and RMySQL 0.5-6 to provide a link to the MySQL database system.

You can free download these two packages from:

R 2.1.1 --- http://cran.r-project.org/ (The Comprehensive R Archive Network, CRAN)

DBI 0.1-6 --- http://cran.r-project.org/

RMySQL 0.5-6 --- http://stat.bell-labs.com/RS-DBI/download (Note: it is not on CRAN)

2. Install database server MySQL

From http://dev.mysql.com/downloads/mysql/4.1 you can free download MySQL 4.1

3. Install Rmdset

You can install Rmdset in either of these two ways:

First Way:

Copy Rmset_1.0.zip file to your computer.

i. In the RGui Window, select "Install package(s) from local zip files„„" from "Packages" menu.

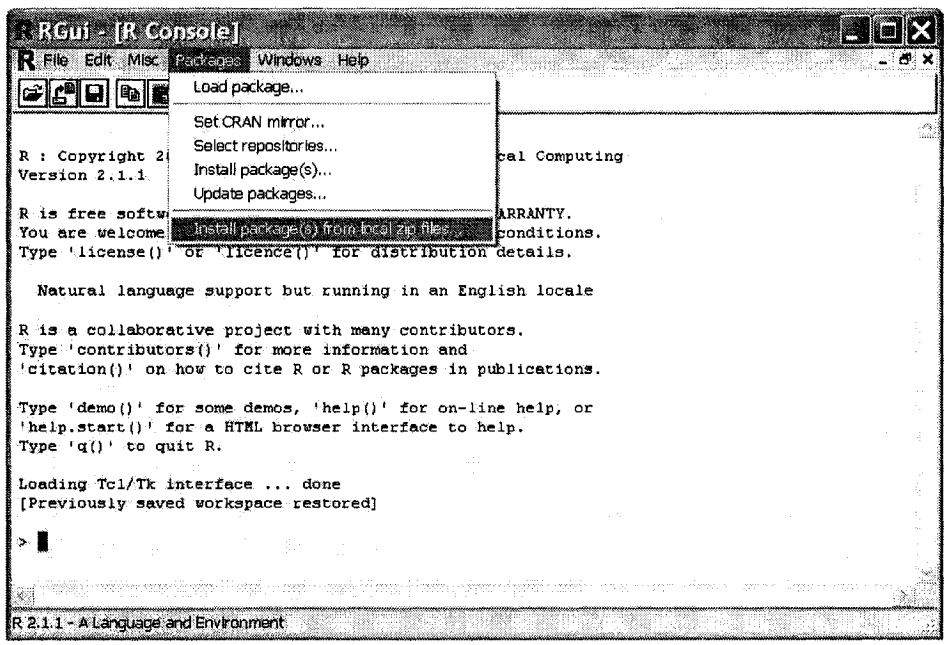

Figure 43 Install Package(s) From Local Zip Files

ii. Browse the Rmdset_1.0.zip file saved in you computer to install.

Figure 44 Brows Rmdset_1.0.zip

Second Way:

Copy Rmdest_1.0.zip file to the "library" directory of the R package installed in you computer directly

B. Rmdset Usage Examples

```
#load package "Rmdset"
>library(Rmdset)
# specify a MySQL database
>drv <- dbDriver("MySQL")
>con <- dbConnect(drv,group = 'rtest')

#load the default variable-metadata set "varMDSet"
>data(varMDSet)

---------------------------md.dataset()-------------------------

# create a data set named "test" with variables selected
from "varMDSet"
>md.dataset("test",con,varMDSet,3,1,4)
# show the structure of "test"
>res <- dbSendQuery(con, "describe test")
#retrieve the result from MySQL database
>fetch(res)
# assign values to "test"
>value <- read.csv("test-data.csv",FALSE)
>dbWriteTable(con,"test",value,row.names=0,overwrite=FALSE,
append = TRUE)
[1] TRUE
# retrieve "test"
>dbReadTable(con,"test")
# query the data set level metadata of "test"
>dbReadTable(con,"test_dsmd")
# query the variable levle metadata of "test"
>dbReadTable(con,"test_varmd")
# query the conceptual level metadata of "test"
>res <- dbSendQuery(con,"describe test_conmd")
>fetch(res)
# query the docmentary level metadata of "test"
> dbReadTable(con,"test_docmd")

---------------------------md.addVar()--------------------------

#1. insert a new variable after the variable "Age" in "test"
>md.addVar(con,"test",varMDSet,2,"Age")
# query the data set level metadata of "test"
```

```
>dbReadTable(con,"test_dsmd")
# query the variable level metadata of "test"
>dbReadTable(con,"test_varmd")
# query the docmentary level metadata of "test"
>dbReadTable(con,"test_docmd")

#2. add a new variable as the first column of "test"
>md.addVar(con,"test",varMDSet,6,first=TRUE)
# query the data set level metadata of "test"
>dbReadTable(con,"test_dsmd")
# query the variable level metadata of "test"
>dbReadTable(con,"test_varmd")
# query the docmentary level metadata of "test"
>dbReadTable(con,"test_docmd")

#3. add a new variable as the last column of "test"
>md.addVar(con,"test",varMDSet,5)
# query the data set level metadata of "test"
>dbReadTable(con,"test_dsmd")
# query the variable level metadata of "test"
>dbReadTable(con,"test_varmd")
# query the docmentary level metadata of "test"
>dbReadTable(con,"test_docmd")

---------------------------md.delVar()--------------------------

# remove the variable "Education" from "test"
>md.delVar(con,"test","Education")
<MySQLResult:(3068,0,23)>
>dbReadTable(con,"test")
# query the data set level metadata of "test"
>dbReadTable(con,"test_dsmd")
# query the variable level metadata of "test"
>dbReadTable(con,"test_varmd")
# query the docmentary level metadata of "test"
>dbReadTable(con,"test_docmd")
```

C. Build an R Package Under WindowsXP

This document takes R 2.1.1 as an example to illustrate how to build an R package under Windows XP.

Step One: Download and Install Essential Software Tools

1. UNIX tools: Rtools, cygwin and hhc.exe

   You can get these tools freely from:

   http://www.murdoch-sutherland.com/Rtools/tools.zip

   http://www.maths.bris.ac.uk/~maman/computerstuff/Rhelp/cygwin.zip

   http://www.maths.bris.ac.uk/~maman/computerstuff/Rhelp/hhc.exe

   We recommend unzipping and saving Rtoos and cygwin in the C directory "c: /".

   You may encounter a "Can't build chm files" problem which is because "R CMD build" does not know where hhc.exe is. So we recommend to install hhc.exe in "c:/cygwin".

2. C compiler: minGW 5.0.0

   Free download minGW5.0.0 from: http://www.mingw.org/download.shtml#hdr6 and install it in the C directory "c: /".

3. Download and Install Perl

   From http://www.activestate.com/Products/ActivePerl/Download.html down load Perl freely and save it in "c: /"

Perl 5.8.4 or later are all suitable

4. MixTex or fpTex for building the documentation

You can find MixTex from: http://www.miktex.org/. We recommend to install it in "c: /". Alternatively, to download fpTex from

http://ftp.ktug.or.kr/tex-archive/systems/win32/fptex/current/ and run it.

In the window of "TeXLive Setup Wizard":

i. Under "Source Directory" check "Local Directory/ZIP files"

ii. Under "Enable Internet Download" check "use IE5 settings"

iii. Under "CDROM/Local depot for files" type "C:\DOCUME~1\YIWENCHEN\LOCALS~1\Temp\"

iv. Under "URL for Internet files" type "ftp://ftp.ktug.or.kr/texarchive/systems/win32/fptex/current/"

v. Follow the default options till finished.

This procedure may take an hour or so.

5. Html compiler for building compiled Windows html help files

Download the Html Help from:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/hwMicrosoftHTMLHelpDownloads.asp and save it "c:/program files"

Step Two: Edit System Variables PATH

1. Open the "System Prosperties" window

To do so, you can from the "Start" menu to select "Control Panel" then "System" OR

From "Start", right click "My Compute" then select "Properties".

2. Edit "System variables" path

Under "Advanced" tab, click "Environment Variables"

Under "System variables" select "path" then "edit" to edit your path.

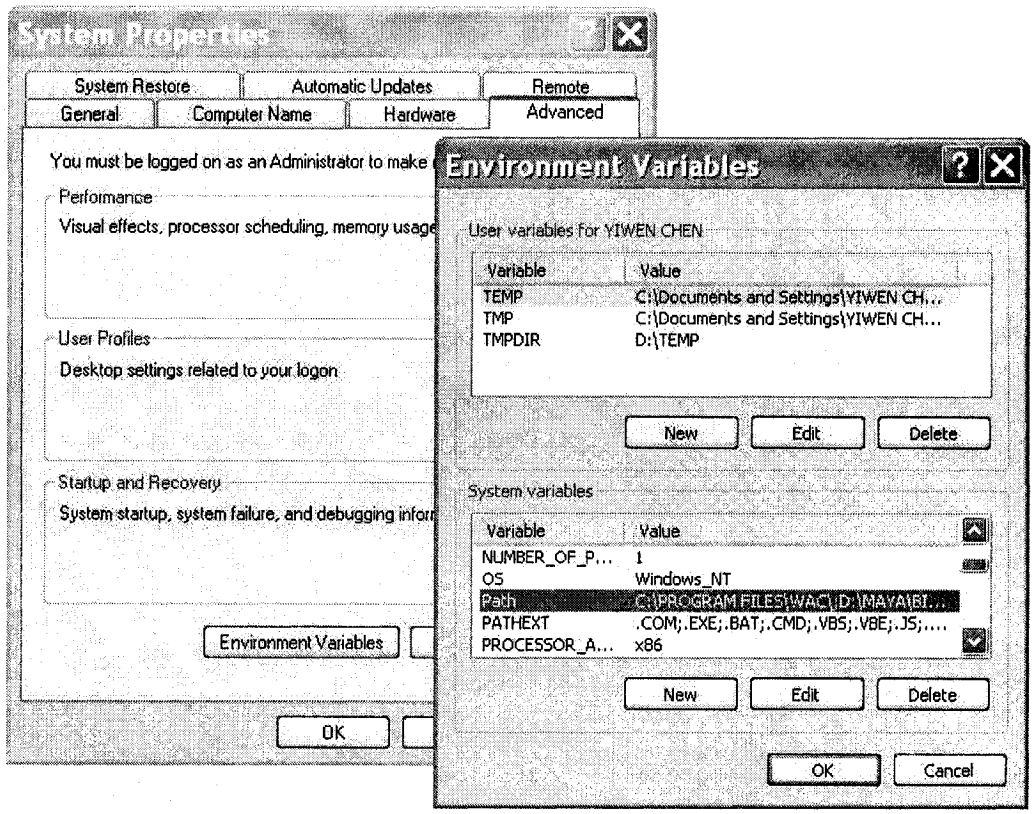

Figure 45 Set "System variables" Path

You should set your path to have Rtools, Perl, minGW, MikTeX or fpTex, R and the html help compiler at the beginning (see an example showed in Figure 4).

Notes:

- There is no space after semicolons
- path designations are NOT case-sensitive under Windows
- Rtools, Perl,minGW and MikTex have the binary or executable (.exe)files stored in the "bin" directories. Don't forget have these "bin" subdirectories in your path (HTML Help is an exception).

```
C:\Rtools\tools\bin;C:\Perl\bin\;c:\MinGW\bin;C:\cygwin;C:\texmf\miktex\bin;C:
\programfiles\R\rw2011\bin\;C:PROGRA~1\HTMLHE~1\;...
```

Figure 46 Example Path

3. Check "System variables" path setting and software installation

   i. Open the command window, change to the C directory and type "path". Check carefully whether your path has been set correctly. If you find an error, go back to "System variables" to edit your path again. After checking close the command window.

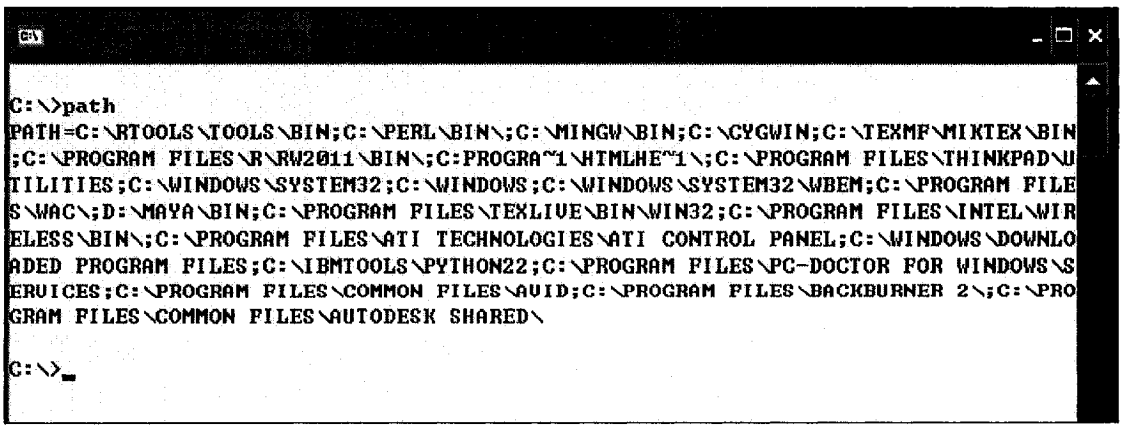

Figure 47 Check "System variables" path Setting

ii. Restart the command window and type "R" to invoke "Rterm", the command prompt version of R. If you see the usual R welcome message and the prompt ">", the "R.exe" file has been set correctly is your path. Otherwise, you may see "R is not a recognized as an internal or external command ...". Then quit R by typing "quit" or "q" at ">".

iii. At the command window type

```
gcc -help
perl --help'(perl)
Tex --help'(Tex)
dos2unix --help
```

You should get a long list for each of above commands. If you get a fail for any of these checks, go to check your path.

Step Three: Set a Valid Temporary Directory

At the command window, type "set TMPDIR=c:\Windows\temp". If you want to set "TMPDIR" rather than in the C directory, for example in the D directory, you can do it as below:

Open "Environment Variables" window as before.

i. Under "User variables for YIWENCHEN" click "NEW"

ii. Set "Varible name:TMPDIR" and "Variable value:D:\TEMP"

iii. Under "System variables" click "NEW" and repeat step ii.

iv. Create a new folder named "TEMP" or "Temp" in "d: /".

Once your path and "TEMPDIR" have been set correctly, you are ready to test and build your package.

Step Four: Construct Your Package

1. Use package.skeleton() to create a directory tree

package.skeleton(name = "anRpackage", list, environment = .GlobalEnv,
path = ".", force = FALSE)

The directory tree will include top directory files:'DESCRIPTION' and 'README' and subdirectories:'data','man','R' and'src'.

Note: if you don't have C or Fortan code in your package, the "src" directory is empty. Do remember to delete it otherwise you will get a WARNING about subdirectories when run "R CMD check".

Example (copied from R Help page of package.skeleton):

```
f <- function(x,y) x+y
g <- function(x,y) x-y
d <- data.frame(a=1, b=2)
e <- rnorm(1000)
package.skeleton(list=c("f","g","d","e"),name="MyPackage",path="
d:/mypkg/R/f.R")
```

Note: If you do not specify a path, the package will save into the local directory "Document and Settings-YIWENCHEN"

2. Edit "DESCRIPTION" file

Example:

```
Package: MyPackage
Type: Package
Title: An example package
Version: 1.0
Date: 2005-12-08
Author: Yiwen Chen <ychen079@uottawa.ca>
Maintainer: Yiwen Chen <ychen079@uottawa.ca>
```

```
Description: An example package to show how to build an R package
under WinXP.
Depends: R (>= 2.1.1)
License: GPL
Packaged: Fri Dec 23 10:09:53 2005; YIWEN CHEN
Built: R 2.1.1; 2005-12-23 10:09:54; Windows
```

You can edit your "DESCRIPTION" file in a text editor such asVIM which can be free download from http://www.vim.org.

3. Modify "NAMESPACE" file

The "NAMESPACE" file specifies which variable(s) and function(s) are available to package users, and which variable(s) and function(s) should be imported from other packages. For package "MyPackage", there are only two functions f and g will be public. So the "NAMESPACE" file is simple as this:

```
export(f,g)
```

4. Check Your R Functions

To check functions f and g at Rterm:

```
# invoke function f
source("d:/mypkg/R/f.R")
f
function(x,y) x+y
#test function f
f(1,3)
[1] 4
f(1,-2)
[1] -1
...
# invoke function g
source("d:/mypkg/R/g.R")
#test function g
g(0,142)
[1] -142
...
```

5. Check Your Source Code if you have any

6. Create and edit documentation files

   i. Create an Rd file Template for each function

```
prompt(f,file='f.Rd')
prompt(G,file='g.Rd')
```

   .Rd file will be created and saved in "c:\Documents~1\USENAME"

   ii. Edit Rd files and move them to the "man" directory.

   Example:

```
\name{f}
\alias{f}
\title{An Example Function}
\description{
  \code{f} To calculate the sum of two numbers}
\usage{
 f()
}
\arguments{

}
\author{ Yiwen Chen }
\seealso{
 \code{\link{g}}
}
\examples{\dontrun{
f(1,3)
[1] 4
}
}
\keyword{methods}
```

Step Five: Check Rd Files and Create HTML Files

1. To avoid R CMD check errors, use "R CMD Rd2txt" to check the Rd files first.

At command window, change to the directory which contains the Rd files and check your Rd files.

Example:

```
D:\MyPackage\man>R CMD Rd2txt f.Rd
```

R will translate and display function f in the command window.

2. Create a html file for each .Rd file

Example:

```
D:\R files\MyPackage\man>R CMD Rdconv -t=html -o=f.html f.Rd
```

Step Six: Check the Package Using R CMD Check

At the command window, change to the directory level above the start of the package. Type "R CMD check packagename" to check your package.

Example:

```
D:\> R CMD check MyPackage
```

Step Seven: Build the Package Using R CMD Build

At the command window type "R CMD build –binary –use-zip packagename "

Example:

```
D:\>R CMD build --binary --use-zip MyPackage
```

Then will find MyPackage_1.0.zip was created in the D directory " d:/".

Summary of Main Steps of package construction

1. Create Directory Tree

```
e.g. package.skeleton(list=c("f","g","d","e"), name="MyPackage")
```

2. Modify 'DESCRIPTION FILE'

3. Modify 'NAMESPACE FILE'

4. Check Your R Function

```
source("d:/MyPackage/R/f.R")
f
```

5. Modify .Rd files

6. R CMD Rd2txt

```
e.g. D:\MyPackage\man>R CMD Rd2txt f.Rd
```

7. R CMD Rdconv

e.g. D:\ MyPackage\man>R CMD Rdconv -t=html -o=f.html f.Rd

8. R CMD check

```
e.g. D:\> R CMD check
```

9. R CMD build

```
e.g. D:\R files>R CMD build --binary --use-zip MyPackage
```