Casualty Actuarial Society
E-Forum, Spring 2010

The Spring 2010 CAS *E-Forum*

The Spring 2010 Edition of the CAS *E-Forum* is a cooperative effort between the Committee for the CAS *E-Forum* and various other CAS committees.

The Committee on Management of Data and Information presents for discussion five papers prepared in response to the 2010 call for papers on data management, data quality, and data technology topics.

The CAS extended the call for papers in conjunction with the Insurance Data Management Association. The purpose of this program is to develop a source of literature on data topics important to casualty actuaries. This *E-Forum* also includes one additional paper.

This call paper that will be discussed by the authors at the 2010 Risk and Product Management (RPM) Seminar scheduled for March 15-17, 2010, in Chicago, IL.

Committee on Management of Data and Information

Virginia R. Prevosto, *Chairperson*

Waswate Ayana
Jeremy Todd Benson
Suzanne E. Black
Peter T. Bothwell
Erich A. Brandt
Robert Neil Campbell
Houston Hau-Shing Cheng
Kirk Allen Conrad
Benedict M. Escoto
Mark A. Florenz
David Dennis Hudson
Joseph Marino Izzo
Weidong Wayne Jiang
Mary Jo Kannon
Gary W. Knoble
Ravi Kumar
William J. Lakins
Dennis T. McNeese
Raymond S. Nichols
James L. Norris
Thomas A. Nowak
Rudy A. Palenik
Ying Pan
Scott J. Roth
William Paige Rudolph
Richard H. Seward
John A. Stenmark
David A. Traugott
Linda M. Waite
Cheri Widowski, *Staff Liaison*

2010 Spring *E-Forum*

Table of Contents

2010 Data Management, Quality and Technology Call Papers

Text Mining Handbook
Louise Francis, FCAS, MAAA, and Matt Flynn, PhD 1-61

Data and Disaster: The Role of Data in the Financial Crisis
Louise Francis FCAS, MAAA, and Virginia R. Prevosto, FCAS, MAAA, ... 1-35

Data Mining and Predictive Modeling with Excel 2007
Spyridon Ganas 1-17

Duplicate FHA Single-Family Mortgage Records and Related Problems
Thomas N. Herzog, PhD, ASA 1-18

Very Large Calculation Systems: A Specialized Solution for the Complex Needs of Advanced Knowledge Workers
James Madison 1-40

E-Forum Committee

Glenn M. Walker, *Chairperson*

Mark A. Florenz
Karl Goring
Dennis L. Lange
Elizabeth A. Smith, *Staff Liaison*
John Sopkowicz
Zongli Sun
Windrie Wong
Yingjie Zhang

For information on submitting a paper to the *E-Forum*, visit http://www.casact.org/pubs/forum/.

Text Mining Handbook

Louise Francis, FCAS, MAAA, and Matt Flynn, PhD

Abstract

Motivation. Provide a guide to open source tools that can be used as a reference to do text mining

Method. We apply the text processing language Perl and the statistical language R to two text databases, an accident description database and a survey database.

Results. For the accident description data new variables are extracted from the free-form text data that are used to predict the likelihood of attorney involvement and the severity of claims. For the survey data, the text mining identified key themes in the responses. Thus, useful information that would not otherwise be available was derived from both databases.

Conclusion. Open source software can be used to apply text mining procedures to insurance text data.

Availability. The Perl and R programs along with the CAS survey data will be available on the CAS Web Site.

Keywords. Predictive modeling, data mining, text mining

1. INTRODUCTION

Text mining is an emerging technology that can be used to augment existing data in corporate databases by making unstructured text data available for analysis. An excellent introduction to text mining is provided by Weiss, et al. (2005). Francis (2006) provides a short introduction to text mining with a focus on insurance applications. One of the difficulties in getting started with text mining is acquiring the tools, i.e., the software for implementing text mining. Much of the software is expensive and/or difficult to use. For instance, some of the software requires purchase of expensive data mining suites. Other commercial software is more suited to large scale industrial strength applications such as cataloging and searching academic papers. One innovation that has made budget-friendly software available to text miners is open source software. Two of the very popular open source products that will be featured in this paper are R and Perl. R is a programming language that has wide acceptance for statistical and applied mathematical applications. Perl is a programming language that is particularly strong in text processing. Both languages can be easily downloaded from the Internet. Moreover, a large body of literature now exists to guide users through specialized applications such as text mining. This paper will rely heavily on information in the book *Practical Text Mining in Perl* by Roger Bilisoy (2008) when illustrating text mining applications in Perl.

It will also rely heavily on the R **tm** library. While this library is described by Feinerer, Hornik, and Meyer (2008), the applications are not insurance applications, and a significant amount of trial and error can be required to achieve successful application of the software to insurance problems.

We hope to make it easier for potential users to employ Perl and/or R for insurance text mining projects by illustrating their application to insurance problems with detailed information on the code and functions needed to perform the different text mining tasks.

In this paper we will illustrate the application of the text mining software using two applications: 1) analyzing a free-form accident description field in an insurance claims database and 2) analyzing free-form comments in a survey of actuaries. Both applications involve relatively small text fields, compared to those anticipated by many text mining products. We will also show how text mining can be used to generate new information from the unstructured text data.

1.1 Research Context

The field of text mining is rapidly evolving, but at this time is not yet widely used in insurance. Kolyshkina and Rooyen (2006) presented the results of an analysis that applied text mining on an insurance claims database. They applied text mining to a free-form claim comment field to derive "concepts" from the description. Francis (2006) presented an introduction to text mining that provided a simple illustration using a liability claim description field.

A number of texts introduce text mining to less technical audiences. Inmon and Nesavich (2007) provide a nontechnical overview that describes some of the history and applications of text mining. This book gives an overview of the procedures used to mine data, from preprocessing and integrating the text to using data mining procedures like self-organizing maps. Weiss, et al. (2005) provide a comprehensive introduction that includes pseudo code to implement the different parts of the process as well as access to a simple software tool that implements many of the procedures in the book. Weiss's software tool provides an inexpensive way to gain insights into text mining and to perform some text processing and pattern recognition procedures. However, the software has some limitations. It requires download and installation of Java along with the text mining software, can only work with data in xml format, produces output that requires further processing by another programming language before common analyses can be performed and has a steep learning curve. A more elaborate description of difficulties encountered when utilizing text mining software, both open source and commercial, is provided by Francis (2006). Francis suggests that text miners may want to learn one of the languages specifically oriented towards text processing such as Perl or Python.

The rapidly evolving field of text mining has seen advances in the open source tools available for text mining. These tools include 1) introductions to programming in Perl such as Bilisoly (2008),

which specifically focus on text mining, 2) interfaces to Perl in some of the popular statistical programming languages such as SAS, 3) new text mining functions in the popular analytical programming language R, and 4) interfaces to R in some of the popular programming languages. This paper will introduce text mining users to two of these open source tools and provide two detailed examples of their application. The two examples will be of particular interest to actuaries and insurance professionals.

1.2 Objective

The objective of this paper is to improve the accessibility and usability of text mining for insurance applications. By documenting two open-source software languages that can be used for text mining we hope to reduce the significant learning curve that can be associated with text mining software. We focus on two popular open-source languages that can be downloaded for free. This paper will describe in some detail two applications of text mining in insurance. Helpful references for those new to Perl and R will also be provided, as this paper is not a self-contained introduction to these languages.

1.3 Outline

The remainder of the paper proceeds as follows:

- Section 2 will discuss the first phase of text mining: preprocessing of text data and will present techniques that are programmed in Perl;
- Section 3 introduces the R text mining library and will apply it to the GL accident description data;
- Section 4 will apply the R text mining functions to the survey data.
- Both sections 3 and 4 will introduce the second or analytical phase of text mining along with their implementation using R statistical functions.
- Section 5 will summarize the analysis and present conclusions.

2. BACKGROUND AND METHODS

There are two main phases to text mining: 1) Preprocessing and integration of unstructured data, and 2) statistical analysis of the preprocessed data to extract content from the text. Section 2 will cover the preprocessing phase and show how Perl can be used for the preprocessing.

2.1 Data and Software

Two different data sets will be used to illustrate the open source tools: a claim description database and a survey response database. The claim description data is a field from a general liability (GL) database. Table 2.1 provides a sample of several records of the claim description data.

Table 2.1

Accident Description
Crew was hooking a jumper pipe between fire line and water line in order to perform a flush test. when the line was charged the jumper pipe slipped off causing water to spray on newly hung dry wall. Damage to one piece of dry wall and few pieces of i.
CAT 345 traveling under a guide wire when the back of the boom caught the wire. When the wire became taut it caused the power pole to break and wire to snap.
Insd was working and damaged buried service wires at a customer's residence.

The GL data also contains two additional variables: the claim's severity developed to ultimate and trended to a common point in time and an indicator variable encoding whether or not an attorney is involved in the claim. Both variables are potential dependent variables in a predictive model. That is, we may be interested in what the characteristics are of claims that tend to have attorney involvement, as the claims adjusters might be better able to manage claims if they had such information. In addition, the claim severity variable is a key outcome variable of the underwriting and claims adjustment process and may assist a company to better understand and manage this variable.

The survey data is from the Casualty Actuarial Society Quinquennial Survey conducted in 2008. Numerous questions on the survey allowed for write-in answers. One of the questions asked was: "What are the top two issues that will impact the CAS in the next five years?" Table 2.2 shows an example of some of the answers to the question.

Table 2.2

Survey Question: Top Two Issues Affecting CAS
A crisis that could affect our ability to "regulate" ourselves.
A need to deal more thoroughly with nontraditional risk management approaches.
Ability of members to prove they are more than just number crunchers.
Ability to convince noninsurance companies of the value/skills offered by CAS members.

The survey data does not contain any potential dependent variables. When analyzing free-form survey responses, text mining is used to group the unique response into categories of responses, as an aid in reducing the many individual responses to a smaller set that still faithfully represents the responses supplied.

2.1.1 The Software - Perl

Two software programs are featured in this paper: Perl and R. Each must be downloaded from the appropriate Web site and then installed, before it can be used.

The Perl Web Site is www.Perl.org. Figure 3 displays the screen for the home page of the Web site. By clicking on "Download" on the top left side, you will be redirected to a site where you can download Perl. Perl must then be installed by running the execute file. A commonly used alternative is ActivePerl from http://www.activestate.com/activeperl/. One common "gotchya" with either install (or, battling multiple Perl installs/versions on Windows platforms) is that the Windows command search must be correct for Windows to find the desired perl.exe file. This executable search path is governed by the search path for the commands. If things are amiss, one can check/edit the setting of one's Window's environmental "PATH" variable. The current setting can be examined from a cmd prompt by simply typing "PATH". The path variable itself can be easily set in standard Windows fashion from the desktop, right-click on "My Computer", "Properties", "Advanced", "Environmental Variables" button, scroll-down the tiny window to find the path variable and click on the "Edit" button, and adjusting the very long string in the tiny edit window.

While free documentation for using Perl is available on the Perl Web Site, there are many books that will guide the new user through key aspects of its use that may be difficult to understand solely from the online documentation. The book *Practical Text Mining with Perl* by Bilisoly (2008) is an excellent resource for text mining in Perl that requires no prior exposure to Perl. However the book does not cover a few introductory topics such as installation and header lines required at the beginning of Perl programs. *Perl for Dummies* (Hoffman, 2003) covers these topics in its first few chapters.

Key things to keep in mind when using Perl are:

- Perl must be run from DOS. One gets to DOS by finding the Command Prompt on the

Programs menu.[1]

- Before running Perl, switch to the Perl directory (i.e., if Perl was installed and it is in the folder named Perl, in DOS, type "cd C:\Perl").
- Programs need to be saved in text processing software. We recommend Notepad rather than Word, as some of the features of Word cause unexpected results when running a program. We recommend using the extension ".pl".
- The header line of a Perl program is dependent on the operating system. Appendix B provides examples of the header line that executes successfully on a Windows computer.
- To run a Perl program type the following at the command prompt:

  Perl program_name input_file_name, output_file_name[2]

  - The input and output files are only required if the program requires a file and the file name is not contained in the program itself.
- A few aspects of the syntax to help readers follow the Perl code are:
  - Each line ends with a semicolon.
  - The # is used for comments.
  - Scalar variable names begin with a $.
  - Vector variable names begin with the @.
  - A specific value of an array is accessed with brackets; $Array [1] accesses a specific value of an array. Note the use of the dollar sign when accessing a specific array value.

[1] The operating system used with Perl for this paper was Windows XP. The command prompt is found by clicking on "Programs" and then "Accessories." Placing a shortcut on the desktop will make it more efficient to get to DOS.
[2] The input and output file may be contained in the program code, as illustrated in some of our examples, rather than entered at runtime.

Figure 2.1

The Perl Web Site

2.2 Step 1: Preprocessing the Data

The data that most analysts are accustomed to working with is structured data. Text data is unstructured data. Unstructured data has the following characteristics:

- It has no specified format.
    - .txt, .html, .xml, .pdf, .doc are among the possible formats for text data.
- It has variable length.
    - One record might contain a phrase of a few words, a few sentences or an academic paper of many pages.

- It can have variable spelling.
    - Misspellings, abbreviations, and singular versus plural forms of words are among the reasons for variable spelling .
- Punctuation and other nonalphanumeric characters may be in the data.
    - Periods, question marks, dashes, equal signs, and quotation marks are among the characters that may be found.
- The contents are not predefined and do not have to adhere to a predefined set of values.
    - Even when restricted to one discipline such as actuarial science, a paper can be on a variety of topics such as ratemaking, reserving, reinsurance, enterprise risk management, or data mining. An accident description field may refer to the part injured, the injury, circumstances of the accident, etc.

To make text data useful, unstructured text data is converted into structured data for further processing. There are several steps involved in the preprocessing:

- Parse the data. That is, extract the words from the data, typically discarding spaces and punctuation.
- Eliminate articles and other words that convey little or no information.
- Replace words that are synonyms, and plural and other variants of words with a single term.
- Create the structured data, a table where each term in the text data becomes a variable with a numeric value for each record.

2.3 Parsing the Text Data

Parsing text involves identifying the spaces, punctuation, and other non alphanumeric characters found in text documents, and separating the words from these other characters. Most programming and statistical languages contain character procedures that can be used to parse the text data. For instance, if the text data contains no punctuation and the words are separated by a space the algorithm for parsing words from spaces is:

- Initialize an array to hold the words that are parsed.
- Search for the first space and record its position.

- Extract the string from the first position to the position before the next space. This is the first word.
- Find the next space and record its position.
- Use the positions of the previous and subsequent space to extract the second word.
- Repeat until all words are parsed.

In Perl a special text function helps one to quickly parse text data. The function is the *split* function, which splits words from spaces and other unwanted characters. The split function takes as its arguments a character string used to separate words (contained between slashes) and the text string to be parsed.

For instance, the following code (saved in the program Parse1.pl) can be used to parse the survey response, "Ability of members to prove they are more than just number crunchers."

$Response[3] = "Ability of members to prove they are more than just number crunchers";

@words =split (/ /, $Response);

When the print function is used (i.e., within a loop containing print "$words[4]\n";) the result is a printout of the word array with each entry on a different line as in Figure 2.2.

A natural question to ask is, "What happens if there are two spaces instead of one?" The following code parses a text string with more than one space separating the words where the string expression "\s+" is used to denote one or more spaces.

$Response = "Ability of members to prove they are more than just number crunchers";

@words =split (/ [\s+]/, $Response);

The notation "\s" denotes a single space and is the same as simply placing a space between the slashes in the split function. However, the notation "\s+" denotes one or more spaces, and can thus be used where there are a variable number of spaces. In addition, the split parameters "\s+" have been enclosed in brackets[5]. The code was saved in the program "Parse2.pl.

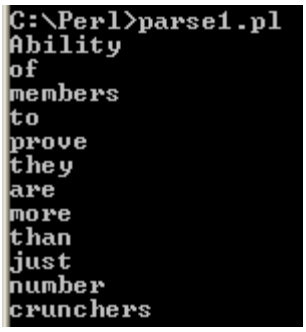
The expression "\s+" is referred known as "regular expression." Regular expressions can be used to 1) parse text containing more complicated structures such as multiple spaces and punctuation, 2)

[3] Note the use of quotation marks used when giving a string variable a value within a program
[4] $words is used to print an array element, rather than the entire array. See sample program Parse1.pl.
[5] Without a bracket around the "\s+", the code does not always work as expected.

search for specific strings of text in a large database, and 3) replace certain strings with other strings (or the empty string). Because regular expressions use special variables to specify character patterns, such as expressions that denote digits, letters, etc., we have included a section in Appendix A that contains a glossary defining the common variables.

Figure 2.2

Output of Pars1.pl shown on DOS screen



Regular expressions can be used to eliminate unwanted punctuation from text data. For instance, suppose you are parsing the expression "A crisis that could affect our ability to 'regulate' ourselves." This express contains quotes and a period that we want to remove. This can be accomplished by first using the split function then using an additional regular expression to retain only the alphanumeric characters, not punctuation. An example of this is supplied in the program Parse3.pl. In addition, the substitution function, **s**, can be used to find and replace unwanted text. For instance if the text is:

$Response="Desire/ability for participation from a wide spectrum of members."

Typically the substitution operator has the following form: s /string/replacement string/. If we want to replace all periods with a space we would use s/.//g. The g following the operator stands for 'global' and will replace all occurrences of the string. Additional modifiers may be required, i.e., when the character removed is the "/". We can remove the "/" with the following code which must use the escape symbol "\" before the "/".

Response=~ s/\///g;

More complicated expressions are used to remove all unwanted punctuation, and examples are provided in the sample Perl programs and appendices[6] supplied with this paper.

[6] Many of the Perl programs referenced in this paper are included in Appendicies.

More complicated regular expressions can also be used to extract strings *not* containing a specific set of characters, extract strings with pre-specified patterns of letters or numbers, such as a phone number or social security number, and to determine where sentences begin and end, etc. The reader who wishes to pursue this subject further should refer to Chapter 2 of Bilisoly (2008), Chapter 12 of Hoffman (2003) or the more extensive coverage in Frenz (2005).

The early days of text mining involved simple word searches for a key word in some text. For instance, suppose we noticed that a number of GL claims involved damage to homeowners and we wished to further investigate these claims. Perl could be used to search for the word "homeowner" in the accident description field. The following is the procedure for doing this:

- First, read in the accident description field.
- For each claim:
    - Read in each word.
    - If the lower case of the word is "homeowner" output a 1 for the new indicator variable, otherwise output a 0.

Code for this is displayed in the program SearchTarget.pl. Inputting data is covered briefly in Appendix A. Table 2.3 below shows that claims involving homeowners are more than twice as severe as other claims:

Table 2.3

Homeowner Claim	Mean Severity
No	2,376.6
Yes	6,221.1

2.4 Simple Text Statistics

One of the first things an analyst may want to do when examining text is to produce simple text statistics. For instance, what are the length statistics of words and the most common words in the text? In Perl the length function can be used to evaluate the length of words. A count variable is updated with the count of words of length *n*. Table 2.4 presents the length statistics for both the GL Claims data and the survey response data. The program length.pl contains the code for computing the length of words. The length of each word is tabulated within a loop. A key line of

code is:

```
$count[length($x)] +=1; #increment counter for words of this length.
```

(Note: in this code the variable $x holds the word after the line has been parsed.)

Table 2.4

Distribution of Length of Words

Length	GL Data	Survey Data
1	1,062	21
2	4,172	309
3	5,258	298
4	5,418	215
5	2,982	153
6	2,312	143
7	2,833	213
8	1,572	161
9	1,048	216
10	591	146
11	111	92
12	156	44
13	78	61
14	19	2
15	0	3
16	1	1
17	2	0
18	1	0
19	1	0

To compile length statistics one needs to create a list of every word in a database, loop through the database, and tabulate the number of times each word is found. To efficiently create a list of words, a hash structure is used in Perl. A hash is like an array, but can be distinguished from an array in a number of ways. An array is typically indexed with zero and integer values, while a hash can be indexed with a letter or word. Instead of an index the hash has a key that maps to a specific array value. For instance, while the first entry in a Perl array is $array[0], the first entry in a hash might be $hash{'a'} or $hash{'x'} or even $hash{'hello'}. (Note that the order is not relevant.) Because the time to locate a value on a hash table is independent of its size, hashes can be very efficient for processing large amounts of data (Hoffman 2003, Wikipedia, 2009).

In Perl a hash variable begins with a %. A hash holding a list of words might be denoted

%words. A hash holding the counts of words from a document might be %count, and the indices of the hash can be the words themselves. A specific value of a hash is referenced by using a dollar sign ($) in front of the hash variable name, and referencing the specific item with brackets. For example, the specific word homeowner is referenced as $words{'homeowner'}. Thus the indices of the hash are strings, not numbers. A counter for each word can be updated by referencing the word directly, rather than trying to locate its position in an array. This eliminates the need to loop through an array to find a specific word before updating the count for the word in a document. Suppose we wanted to update the count for the word "actuary". This could be accomplished using the code $count("actuary"} +=1; In general, to count words using a conventional array we would need to do the following:

- Loop through each response and each word in each response.
- Find all unique words, and number of unique words.
- Assign each word an index *I* in an array.
- Create an array to hold count for each word. Assign word from array to index.
- Loop through record of text. For each word in each record:
    - Find the array index *I* for the word.
    - Update count stored in counter(*I*) by 1.
- Continue until last word on last response is counted.

With hashes the process requires less looping. The procedure is:

- Loop through each record.
- Loop through each word for each record.
- The index for the word in the hash is the word, i.e., the index for the word actuary is 'actuary'. Update the counter by 1.
- Continue until last word on last response is counted.

Sample code that was used to tabulate the words in the CAS survey response is shown in Testhash.pl. Table 2.5 displays the frequency of each word in the GL data and in the survey response. Figure 2.3 shows that an approximate linear relationship holds between the log ranks and the log probabilities for the survey data, as would be expected under Zipf's law. [7]

[7] Zipf's Law provides a probability distribution for the frequency of words in text. It is like a discrete version of the Pareto distribution. A feature of Zipf's law is that a plot of the frequency of words versus the rank of the word on a log scale will be approximately linear. Perl can be used to tabulate the frequencies of words in a document or database to see if they follow Zipf's law.

Table 2.5

Word Frequencies for Survey Data

Rank	Word	Count	P(Rank=k)
1	of	102	0.05
2	the	80	0.04
3	to	57	0.03
4	and	53	0.03
5	in	42	0.02
6	actuaries	34	0.02
7	other	27	0.01
8	from	26	0.01
9	for	25	0.01
10	erm	24	0.01
726	alternative	1	0.00
727	thin	1	0.00
728	information	1	0.00
729	industries	1	0.00
730	retire	1	0.00

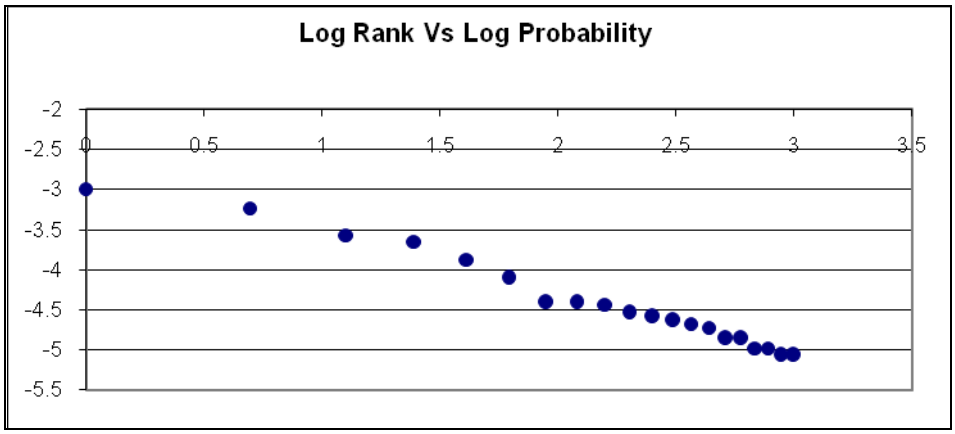


Figure 2.3

2.5 Term Data and Stop Words

In order to perform the statistical procedures that analyze text for content, it is necessary to set up a database that contains information about the "terms" in each response. Under this approach,[8] a "bag of words" is used to extract the important content from text. This means that the order of the words in the text is discarded and only the presence/absence of particular words is recorded. In

[8] Note that other approaches based on linguistics take sentence structure into account

its simplest form, the terms are the complete word collection on a particular record, but later we show adjustments that are made to reduce the number of terms. One could set up a database, where each row is a record and each column is a word in the text on the record. A simple database that can be constructed from text information is a term frequency database, where the frequency of each word in each record is recorded. An example of such a "term-document" matrix is shown below.

Table 2.6 Example of Term-Document Matrix – Survey Data

Ourselves	cas	Not	That	communicators/executive	our	approaches
1	0	0	1	0	1	0
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

To produce the matrix of term indicators it is necessary to 1) create a list of all words in the data base (which will be referred to as the grand dictionary), 2) check each record (either claim description or survey response) for the presence of each word, 3) create an indicator value of 1 if the word is present, otherwise a zero is recorded and 4) print the results to a file for further processing.

The code in TermDocMatrix.pl is used to perform this task. In this example, the use of hashes improves the efficiency of programming the term-document matrix.

Another preprocessing task involves the elimination of "stop" words. Stop words are very common words like the articles "the" and "a" that do not typically add to the informational content of a text string and are therefore are eliminated. Note that certain types of natural language processing techniques that evaluate grammatical structures may utilize such words but the simple statistical approaches we are illustrating eliminate these words.

A simple approach to the elimination of stop words is to use the substitution operator. Thus to eliminate the word "the", use the code

```
s/the//g,
```

which substitutes the null character for the word being eliminated. This is illustrated in the program "StopWords.pl" where separate substitutions are performed for the most common stop words

found in our text data. A more comprehensive approach uses a list of stop words from an external file and eliminates each word in the list. A further task that is performed in text mining is "stemming." When stemming a word, the "stem" or main root of the word replaces the numerous grammatical permutations of the word, such as plural versus singular, past and futures tenses, etc. As part of the process of stemming, synonyms are generally replaced with a single term. For instance, in the survey data, some respondents use the abbreviation ERM when referring to enterprise risk management. Since our two example databases are relatively small, we simply find the multiple versions of the same term and use the substitute operator to substitute a single term for the multiple versions. Many text mining programs reference a database of root terms and synonyms as part of the stemming process. We will illustrate the use of one such implementation in the section on text mining in R. It should be noted that most repositories of synonyms used for stemming will not contain some of the relationships that appear in a given unique database. This is particularly true when misspellings are considered. Thus the stemming part of preprocessing must necessarily be tailored to a given application and code may need to be manually developed to complete the task.

2.6 Search Techniques

One of the more common interactions that Internet users have with text mining is search technology. Though information retrieval (IR) technology is not a focus of this paper, a brief illustration of a simple search process will be provided. The search will utilize a similarity measure to compare a short text string and rank a database of text records with respect to how closely they match the search string. Suppose we are comparing the phrase "Credibility of the Profession" to the phrase "Credibility of the CAS," Table 2.7 shows the term-document matrix for these two expressions. One measure of similarity between the two rows is correlation, which can be measured by the Pearson product moment correlation. The correlation computed between the two rows below is (-.25), even though three out of the five words overlap. This is a consequence of the correlation coefficient measuring the strength of linear relationship, while the relationship between the two rows below is non-linear, as a result of the (1,0) and (0,1) cells.

Table 2.7

Credibility	of	The	profession	CAS
1	1	1	1	0
1	1	1	0	1

The text mining literature frequently uses the "cosine" measure as the similarity statistic. Its formula is:

(2.1) $$\cos(\theta) = \frac{\mathbf{A} * \mathbf{B}}{\|\mathbf{A}\| * \|\mathbf{B}\|}, \mathbf{A}, \mathbf{B} = \text{word frequencies}$$ [9]

.

It is customary to replace the simple word frequency with a weighted frequency before computing cosine and other statistics. The weighted frequency statistic is the TF-IDF (term frequency – inverse document frequency) statistic.

This statistic computes a weight for each term that reflects its importance. This statistic is more relevant when IR technology is applied to academic papers or Web pages where the "records" is an entire paper or the contents of one Web site. For such "records," many terms appear multiple times, whereas most terms in our two examples appear at most once in a given record. The relevance of the term to a particular document will depend on how many words are in the document. This is why the denominator of the TF-IDF is adjusted for the number of words in the document. It is also adjusted for the number of records (or documents) containing the word (i.e., terms that appear on many records are down-weighted).

The TF-IDF formula is:

(2) TF-IDF(i)= Frequency(i)*N/df(i),
where df is the frequency of word (i) in all documents
N is the number of words in the record/document

The TF-IDF can be computed for a phrase and for a database of records or documents. Then, the records can be ranked. The program matchline.pl in Appendix E searches a database (in this case the 10 line database in Top10.txt accompanying this paper). It finds the record most similar to the input phrase "Credibility of the CAS", using the cosine, measure, though other similarity measures could be used instead. The procedure used is:

- Read the database.
- Create a hash of all words on each record.

[9] The norm of a vector in the denominator of the formula is the square root of the sum of the squares of the elements. It appears that the cosine differs from the more familiar correlation coefficient in not centering the variables by subtracting their means.

- Create a hash of all words in the database.
- Compute the TF-IDF statistic for each term on each record of the database.
- Read the search string.
- Compute the TF-IDF for the search string.
- Compute the cosine correlation between the TF-IDF of the search string and each record in the database.
- Determine which record is the closest match and print it out.

2.7 Next Steps: Statistical Analysis to Derive Content

To derive content from text, techniques referred to as unsupervised learning are used. With unsupervised learning techniques, the statistical analysis procedure has no dependent variable to fit a model to. Instead the values of the variables in the data are used to group like records together. For instance, in the GL claims data, all records with the words indicating a vehicle accident might be grouped together. In the survey response data, all records invoking the word "credibility" might be grouped together. All responses using "ERM" or "Enterprise Risk Management" might be grouped together, but in a separate group from those with the word "credibility". A procedure referred to as clustering is used to perform the statistical analysis of the term-document matrix to group similar records. Bilisoly (2008) illustrates using the data from the Perl preprocessing (such as the term-document matrix) within an R program to perform the required statistical analysis. Bilisoly (2008), Francis (2006) and Weiss et al. (2005) provide a more detailed description of how to apply clustering and other unsupervised techniques to text mining applications. The next section of this paper will introduce R and its use in text mining. Though Perl can be used to preprocess the data and perform simple text analytics, we will introduce the R functions that read in and preprocess data as well as the functions that perform the statistical analysis of the data that is required for content analysis.

3 The Software – R and the GL Database

3.1 –Introduction to R

One of the most popular open source analytical languages is R. This language is widely used to perform statistical and data mining analyses. R can be downloaded and installed on one's computer from the CRAN Web Site: http://cran.r-project.org/. Figure 3.1 displays the screen for the home page of the Web site. By clicking on your operating system: Linux, Mac OS X, or Windows, under "Download and Install R" on the top center, you can download and install R. While free documentation for using R is available on the R Web Site, there are many books that the new user might enjoy and benefit from. We recommend *Introductory Statistics with R* by Peter Daalgard, *Modern Applied Statistics with S* by Venables and Ripley, *Data Analysis and Graphics Using R* by Maindonald and

Braun, and *Statistics, an Introduction using R* by Crawley.

The CRAN Web Site includes a wealth of material for learning R including FAQs, tutorials, introductory papers.

Of particular note is the link on the left side of the main page to Contributed Packages or Libraries of specialized functions. We will need to download and install the main text mining package **tm** as well as a number of helper packages. One of our motivations for introducing the R text mining procedures in this paper is that they can be confusing to figure out and produce useful results from, in the absence of guidance from those who have already struggled with these issues.

3.2 The tm package

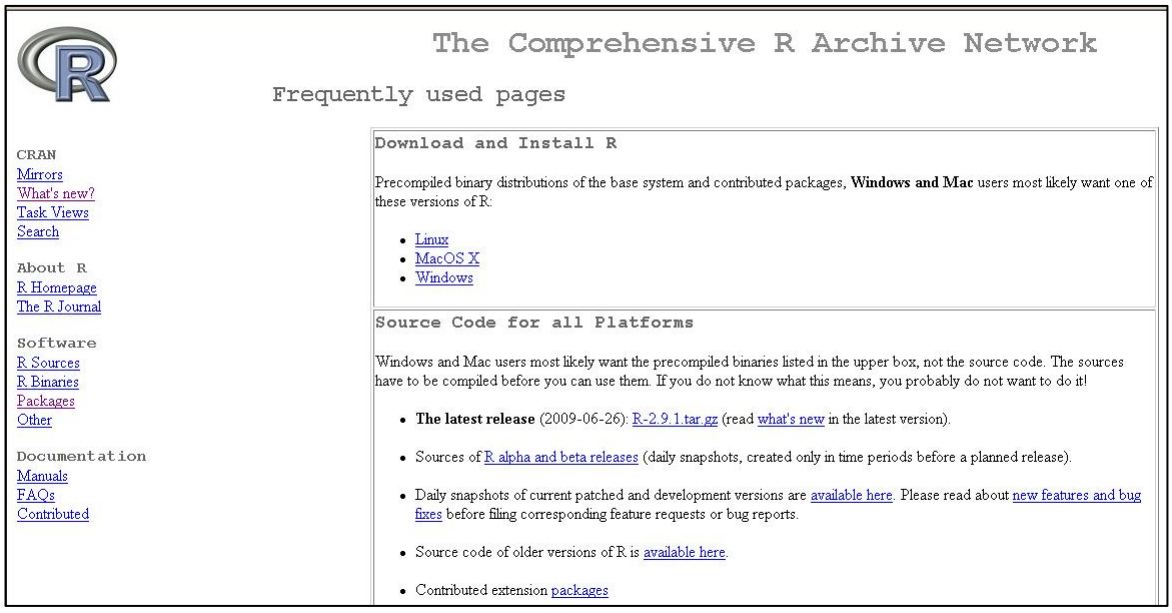
R is a statistical computing environment best known for its ability to analyze structured data. The key package that makes analysis of unstructured data available to the R user that will be used throughout this section is the text mining package **tm**. Feinerer et al. (2008) provides an introduction to this tool. However, the authors of this paper experienced minor difficulties in using **tm** based solely on Feinerer et al. and the **tm** help documentation. Note that while this paper was in progress, the **tm** package was updated and previous code had to be changed to conform to the current version. One objective of this paper is to make the process easier for others new to the R package **tm**.

One can install packages quite conveniently in R via the /Packages/Install Packages menu command. Currently, the CRAN package repository features over 1,800 packages! Several additional packages might be of interest to CAS members, including **actuar** by Vincent Goulet.[10]

[10] http://www.actuar-project.org/

Figure 3.1

CRAN – Comprehensive R Archive Network



Once required packages are downloaded and installed, we first will access the libraries of functions. Note that several of the packages used for specialized graphics in this paper cannot be found on the CRAN Web Site. For these, we provide documentation in our accompanying R programs about where to download these packages.[11] Once they are downloaded, the user must use the "Install Packages from Local Zip Files" option of the Packages menu. While R has some skeletal GUI functionality, the R programming language is the primary vehicle for using R. The R command line begins with the symbol ">". Examples of R code presented in this section will begin with a line that starts with ">" and use a different font.

The "library" command loads the bundled functions of the desired library. Help describing those functions is a short command away.

```
>library(tm)  #loads library
>help(package=tm)
```

The ability to replicate the examples in this paper is dependent on the user installing all packages that we reference in the text and code. For example, checking the results of the "help" command

[11] We use the Tinnr editor, but our code can be read in Word.

above, the tm package "Suggests: filehash, proxy, Rgraphviz, Rmpi, RWeka, snow, Snowball, XML", that is, **tm** depends on a number of additional packages for some of its functionality. Moreover, some of the packages are updated periodically; therefore the R user should occasionally check for and, if necessary, update using the update packages feature of the R packages menu. In addition, it is recommended that one install version 2.9.2 or later of R, as many of our illustrations do not work on earlier versions.

A number of text data sources are available for use within the R text mining package. One or more of the text sources is used to illustrate the application of data in some of the papers cited in the reference section (Weiss, 2005 Bilisoly 2008). In addition the documentation for the **tm** packages utilizes some of these databases (Feinerer et al., 2008). For an up-to-date list of available data sources and readers available within the **tm** package use the following command in R:

```
> getSources()
[1] "DataframeSource" "DirSource"       "GmaneSource"     "ReutersSource"
[5] "URISource"       "VectorSource"
```

Before the analyst can analyze data, whether structured data, such as a pricing database, or unstructured data, such as accident descriptions or responses to survey questions, the data must be read into R. The most common function used to read in data is the read.table function, which reads data that is tab-, comma-, or space-delimited, or uses a delimiter specified by the programmer. (It defaults to the space/tab delimiter). However, since data is commonly found in other formats than the classical text or ASCII files a variety of other readers are now available.

```
> getReaders()
 [1] "readDOC"                "readGmane"
 [3] "readHTML"               "readNewsgroup"
 [5] "readPDF"                "readReut21578XML"
 [7] "readReut21578XMLasPlain" "readPlain"
 [9] "readRCV1"               "readTabular"
```

One can utilize one of the available readers to read in our source data (which is in the form of a single ASCII file) with a single column of text data, laid out in one logical record per row. (Note the forward slashes in the file name). The read.csv function (which reads in comma-delimited data that was saved in csv format in Excel) is used for our data. Note that we want each row of data to be

read in its entirety as a single string, and if spaces are used as a delimiter, each word or string of characters not separated by a space will be read in as a separate variable value.

```
>txt.csv <- read.csv("c:/temp/GLAccDesc.csv"); # or

>txt.csv<-read.csv("c:/temp/Top2.csv" to read in as tab delimited)
```

In order to apply many of the capabilities of R, it is next necessary to convert the file that was read in to the format that the **tm** function requires. To do this the file must be converted to a corpus. A corpus, as understood in natural language processing, is a body or collection of text information (Manning and Schutze, 2003). Wikipedia defines a corpus as "a large and structured set of texts (now usually electronically stored and processed). [Corpus] are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules and a specific universe." (Wikipedia, 2009). For instance a collection of Edgar Allen Poe's short stories can be considered a corpus. In fact, the Poe corpus is utilized by Bilisoly (2008) in many of the text mining examples in his book. Another corpus might be a collection of news articles written and disseminated by Reuters. Within each corpus, each article or story is treated as a separate record by most text mining programs. Thus each record of this paper's claims and survey data is treated the same way as an entire Poe story by the software. We can define a Corpus as follows:

```
>txt <- Corpus(DataframeSource(txt.csv)),
```

which converts the text read by the read.csv function into a corpus. We can then summarize and inspect that collection. The summary function provides some basic information about the text data. For the 1617 rows of the GL data corpus, the summary function code and output are:

```
>summary(txt)

A corpus with 1617 text documents

The metadata consists of 2 tag-value pairs and a data frame

Available tags are:

 create_date creator

Available variables in the data frame are:

 MetaID
```

Metadata is data about data, i.e., a description of the types of variables, their functions, permissible values, etc. Certain types of data formats, such as html and xml contain tags and other

data structures that provide some metadata. For our simple text file there is very little descriptive information about the data in the file, hence the relatively brief and uninformative output of the summary function. The inspect function allows the text miner to print out and inspect a subset of the data. It is a good practice to inspect a sample of the data, both to detect data quality issues and to become familiar with the contents of the data. Below we show the code and output of two records from the GL claims data.

```
>inspect(txt[1:2])

[[1]] Crew was hooking a jumper pipe between fire line and water line in order to perform a flush test. When the line was charged the jumper pipe slipped off causing water to spray on newly hung dry wall. Damage to one piece of dry wall and few pieces of I…

[[2]] While exposing Verizon line - employee used hands to pull line from trench causing it to tear. …
```

As described in the Perl section of this paper, it is necessary to perform some preprocessing to prep the data for text mining. Example tasks include converting text to lower case, removing stop words, stemming and identifying synonyms. To see all available text transformations currently available within **tm** use:

```
>getTransformations()
```

Some applications of available transformations (output) are:

```
>txt <- tm_map(txt, tolower)

>txt <- gsub(Old Pattern, New Pattern, txt)for (j in 1:length(txt)) txt[[j]] <- gsub("[|&|-|/|\\|()|\\.]", " ", txt[[j]]);

# Replace enterprise risk management

>for (j in 1:length(txt)) txt[[j]] <- gsub("enterprise risk management", "erm",txt[[j]])

>for (j in 1:length(txt)) txt[[j]] <- gsub("off shoring", "offshoring", txt[[j]]);

# remove stopwords

>txt <- tm_map(txt, removeWords, stopwords("english"))

>txt <- tm_map(txt, removeNumbers)

>txt <- tm_map(txt, removePunctuation)
```

The transformation function **tm_map** can convert text to lowercase. That is, tm_map(txt, tolower) functions the same way as the Perl function **lc**. With Perl, the text data was preprocessed with the substitution operator to remove unwanted punctuation and patterns. With R, it is easiest to apply to transformations sequentially. To remove unwanted characters in the text data apply the **gsub** transformation. For instance, the following replaces the "/" separating some of the words in the survey data with a space.

```
>for (j in 1:length(txt)) txt[[j]] <- gsub("/", " ",txt[[j]])
```

Then apply the remove punctuation transformation. Without replacing the slash first, two words will be combined into one when punctuation is removed.

Next, remove the stopwords. The **tm** library comes with a list of stopwords in a number of different languages including English. As alternative to using the list, the user can supply a list of stopwords, i.e.:

```
>newstopwords <-c("and", "for", "the", "to", "in", "when", "then", "he", "she", "than")

>  txt <- tm_map(txt, removeWords, newstopwords)
```

The **tm** package stemming function uses an algorithm that removes common word endings for English words, such as "es", "ed" and "'s".

The output from the transformation code above is:

```
>inspect(txt[1:3])

[[1]] crew  hook  jumper pipe  fire line  water line perform flush test  line  charg  jumper pipe slip  caus water  spray  newli hung dri wall damag  piec  dri wall  piec

[[2]]  expos verizon line - employe  hand  pull line  trench caus  tear

[[3]] cat 345 travel  guid wire    boom caught  wire  wire  taut -  caus  power pole  break  wire  snap
```

When comparing this to the lines of unprocessed output on the last page it is apparent that the ending "ing" was removed from "hooking" and "charged" was replaced with "charg", etc.

The next set of tasks for text mining includes creating a Document by Term matrix (or DTM), identifying frequently occurring words, and removing sparse terms.

```
>dtm <- DocumentTermMatrix(txt)
```

Produces a matrix such as:

Table 3.1

a. Subset of Term-Document Matrix – GL Claim Data

alleg	caus	damag	Travel	truck
0	1	1	0	0
0	1	0	0	0
0	1	0	1	0
0	1	0	1	0
0	0	1	0	0

b. Subset of Term-Document Matrix – Survey Data

abil	account	actuari	Topic	Valu
1	0	0	0	0
0	0	0	0	0
1	0	0	0	0
1	0	0	0	1
1	0	0	0	0

Document by term matrices are generally quite sparse – they contain lots of zeros. As illustrated even in the tiny example above, many terms appear in only a few records. The function removeSparseTerms removes terms that occur infrequently in a database. Terms which occur only once or twice are likely to consume a lot of computational resources without adding anything useful to the analysis. We can greatly reduce the size of the dtm without losing much useful information. The following function:

```
>dtm3 <- removeSparseTerms(dtm, 0.94)
```

reduces the number of terms from over 2,200 to about 280 in the GL data. The sparsity parameter of 0.94 says to remove all terms from the dtm with zeros for the term count in 94% of the documents. When the remove sparse terms function is applied to the survey data, the number

of terms is reduced from approximately 400 to approximately 250.

Note that at this point the remainder of the text mining is the same whether the term-document matrix was created in Perl or in R. Thus if there was a document term matrix created in Perl named "dtm.csv", it would be necessary to use the read function to read it in.

```
>dtm.csv<-read.csv("c:/Directory/dtm.csv")
```

Now that the text data has been preprocessed into a more convenient and compact form we can begin to investigate relationships with simple statistics. In section 2, we illustrated how to compute word frequencies in Perl using hashes. In the R **tm** package, there are ready-made functions to perform simple statistical analyses of the term data.

In section 2 we illustrated how to compute correlations between a phrase and the records in a text database. A simple exploratory text mining statistic to compute is the correlation between a single term and all the other terms in a database. If the two terms always occur together, i.e., if both terms are always present or absent together, the correlation will be 1.0. If the two terms never occur together the correlation will be 0.0. Thus correlation is an indicator of how closely related two terms are. The R **tm** package has a function that finds terms that are related to a word of interest to the analyst. We can identify terms that are correlated with a target variable, in this example "travel". The following function:

```
>findAssocs(dtm2,  " travel ", 0.15)
```

finds all words with a correlation of at least 0.15 with the target term. In general, the terms correlated with "travel" appear to describe vehicular accidents.

Table 3.2 Words Correlated With The Word "Travel"

Term	Correlation
Travel	1.00
vehicl	0.34
windshield	0.28
north	0.27
south	0.27
Tire	0.24
lane	0.20
hit	0.19
onto	0.19
flew	0.18
near	0.18
rte	0.18
claimant	0.17
debri	0.15
east	0.15
front	0.15

The correlation measure is a "similarity" measure. That is, it measures how closely related two variables are. It is one of many measures of similarity. Other similarity measures are the Chi-Squared statistic, which measures how closely related two categorical variables are, and phi, which measures the correlation between binary categorical variables. Similarity measures can be applied across rows as well as across columns of a database. When applied across rows the similarity indicates how similar two records are. In the case of text mining, a similarity measure would indicate how many words the two rows have in common.

Complimentary statistics to the similarity measure are dissimilarity measures. Dissimilarity statistics measure how far apart or dissimilar two columns or rows of a database are. A common dissimilarity measure is the Euclidian distance measure, which measures the squared distance between each variable (i.e., for our text data each term) for record *i* and each term for record *j*.

The Euclidian Distance Formula:

$$d_{i,j} = \left(\sum_{k=1}^{m} (x_{i,k} - x_{j,k})^2\right)^{1/2} \quad i, j = \text{records}, m = \text{number of variables.} \tag{3.1}$$

A similar measure, which used absolute value differences rather than the squared difference is the taxicab or Manhattan distance. Kaufman (1990) and Francis (2006) provide a more comprehensive

background on the common dissimilarity measures.

The R proxy library can be used to define similarity and distance measures. Note that although the library uses the dissimilarity function, this function measures dissimilarity and similarity, depending on the measure used. Below is an example of code for computing the cosines between the rows of the GL data.

```
>library(proxy)

>dissimilarity(dtm3, method = "cosine")[1:10]
```

3.3 Clustering for content analysis

The dissimilarity measure is a key input into a statistical procedure that is commonly used in data mining to create content. The procedure is clustering. Clustering separates records into groups that are similar with respect to the terms contained in each record. The algorithm attempts to maximize the dissimilarity between the groups and minimize the dissimilarity within groups, hence it relies heavily on a dissimilarity measure to perform the grouping. Sanche (2006) describes how clustering can be used for variable reduction. Clustering is used to reduce a large number of variables to a smaller set that maintains the predictive information of the larger set. The clustering procedure finds groups of variables that are strongly related to each other, that is, they are highly correlated and tend to convey similar information. After clustering, the group is replaced by a single variable. For instance, in Sanche's example, the two variables, population density and car density, were replaced by one variable encoding the density information. A similar concept applies in text mining: where multiple words refer to the same concept, and therefore can be grouped together, as one variable representing the concept.

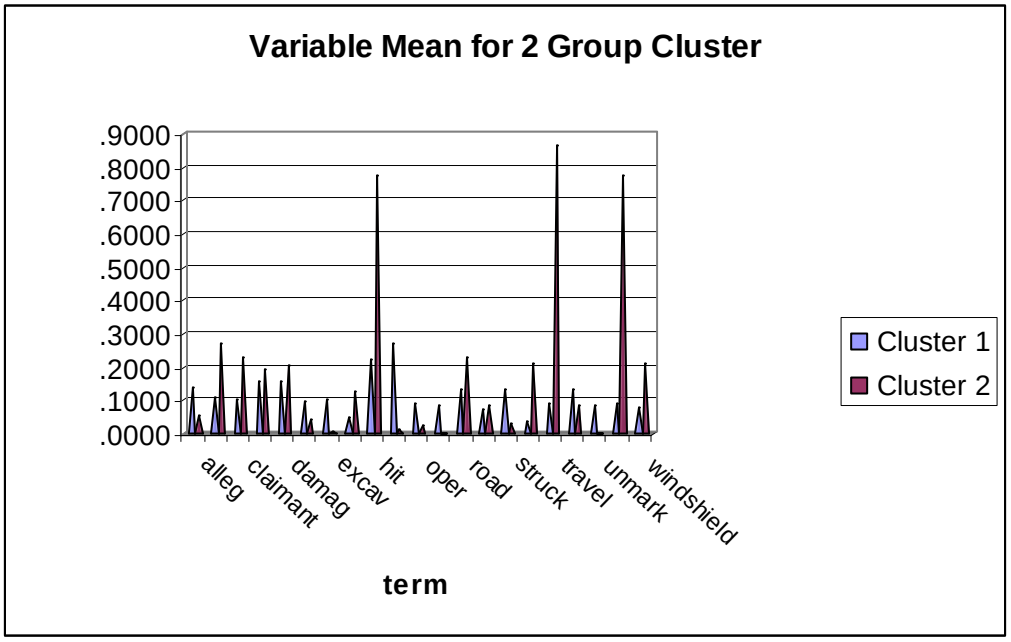
The R stats library contains functions that can be used to identify groups of related terms via the key clustering methods, hierarchal and k-means clustering. Load the stats library, then run the clustering procedure. Only one of the two procedures, k-means clustering, will be applied to the GL data. The hierarchical technique will be illustrated on the survey data in Section 4. To use either procedure, first load the stats library, then run the clustering procedure. The function for k-means clustering is **kmeans**.

The analyst may specify a distance measure or use the default Euclidian measure. To apply k-means to a document-term matrix (dtm3) use the following:

```
>library(stats)

>glKmeans <- kmeans(dtm3, 5)
```

K-means clustering "which aims to partition the points into k groups such that the sum of squares from points to the assigned cluster centers is minimized." (R Documentation for the "stats" package). The procedure is used to break out the term-document matrix into subsets based on clusters. With k-means clustering the user must specify "k", the number of clusters, as well as the distance measure. Francis (2006) discusses several approaches to determining the number of groups to use. We used the k-means procedure to develop several different-sized groupings. With the GL (as opposed to the survey) data, our ultimate objective is to extract information from the accident description that can be used in predictive modeling. First we note that the text miner usually attempts to understand the groupings by examining descriptive statistics, especially the average number of appearances for the terms in each group. Figure 3.2 displays the statistics for the two cluster grouping.

Figure 3.2 Average # Times Term Is Used for Two Clusters



From Figure 3.2, it is clear that the claims in cluster 2 tend to have a higher frequency of the words "hit" and "travel" (as well as vehicle) and thus appear to have a high representation of car

accidents (suggesting two groupings of vehicle accidents versus everything else). While not shown here, other cluster groupings display a different, but typically informative set of descriptive statistics.

3.4 Use of clusters in prediction

The clusters computed by the clustering procedure can be incorporated into modeling databases and used to predict a variable of interest. The decision tree procedure is a popular data mining procedure that can be implemented in R to predict a dependent variable of interest. Brieman et al. (1993), DeVille (2006), and Derrig and Francis (2008) provide detailed introductions to the tree modeling technique. Two reasons for the popularity of decision tree techniques are (1) the procedures are relatively straightforward to understand and explain and (2) the procedures address a number of data complexities, such as nonlinearities and interactions that commonly occur in real data.

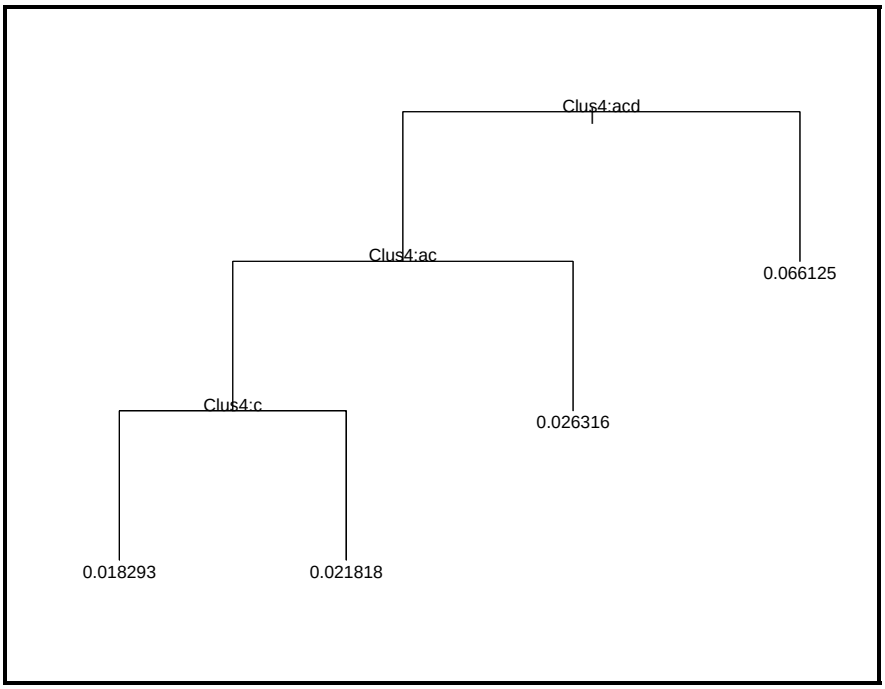
In the GL dataset used for this paper, besides the text data, we have two target variables of interest: an indicator of attorney involvement in the claim and the amount of the claim or its severity. An alternative that we will not explore in this paper is to directly apply the tree models to cluster the terms. A good paper discussing this topic is Himmel, Reincke¸ and Michelmann (2008).

We can fit a CART-style tree model with the **rpart** library. First we remerge the attorney and severity columns back with the newly formed document-term matrix and the fit a tree model using recursive partitioning.

```
>GLAccDesc.matrix <- as.matrix(dtm3)
>attorney <- txt.csv$attorney
>severity <- txt.csv$TrendSev
>GLAccDesc<- data.frame(GLAccDesc.matrix,attorney)
>GLAccDesc.rpart <- rpart(attorney ~ ClusterGroup[12]) #fit tree using clusters as predictors
```

[12] ClusterGroup denotes one of the new variables (cluster groupings) created from the clustering procedure.

Figure 3.3
RPART tree for attorney=0/1. Four Group Cluster is Used for Prediction



A convenient feature for the rpart procedure is that one can export the resulting tree model to pmml (perhaps for import into SAS EMiner™ or another environment for scoring.)

```
>library(pmml)
>rpart.pmml <- pmml(GLAccDesc.rpart)
>saveXML(pmml(GLAccDesc.rpart), file="C:/temp/rpart.xml")
```

The tree in Figure 3.3 graphically summarizes how the data was partitioned into groups, and the terminal branch or terminal node displays the predicted value for each partition.[13] Note that the tree procedure uses all four clusters (although alternative structures could have been found) in predicting attorney involvement and that there are significant differences between the groups in the propensity for attorney involvement. The tree summary (created using the summary function) indicated that the lowest probability of attorney involvement is associated with cluster 3 and highest probability is associated with cluster 2. Table 3 displays the top terms associated with each cluster based on descriptive output. This table indicates that words indicating a vehicular accident are in cluster 3

[13] To know the specific values of the cluster variable used for partitioning the data the summary function must be used.

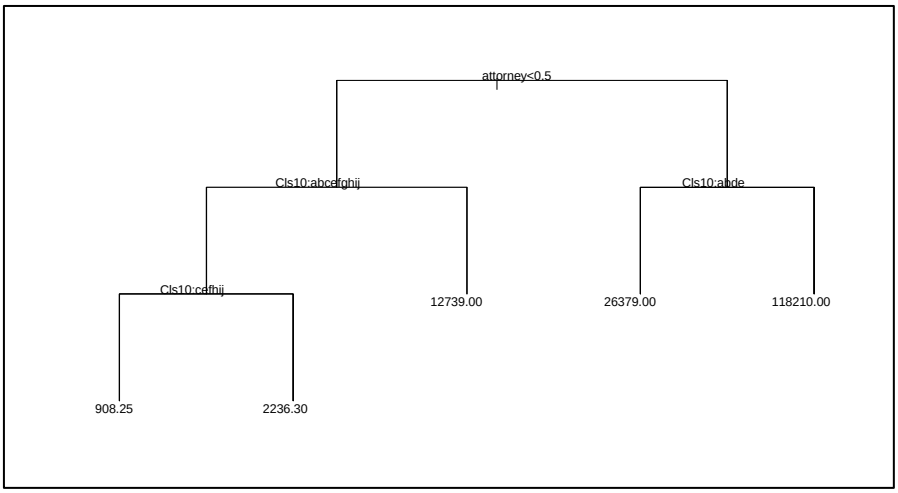
while cluster 2 appears to be an "all other" category.

Table 3. Cluster Content, 4 Group Cluster

Cluster	Key Words
1	excavate, operate, line
2	all other (moderate representation of many words)
3	hit, travel vehicle
4	allege

Next tree analysis is used to model claim severity. The tree in Figure 3.7 utilized the two, four, seven, and 10 group clusters computed with the k-means procedure, along with attorney involvement as potential predictors. The ten group cluster (but not the other clusters variables) is used in the fitted model to predict severity. This would suggest that the different-sized clusters extracted different content from the text data and that the 10 group cluster was the most predictive of claim severity. Note that the average severity for each terminal node varies from a low of $908 to a high of $26,379. The tree summary indicates the lowest severities are associated with no attorney and cluster 4 (of 10). Cluster 4 has a relatively low average content of most of the words, though it loaded higher on operator than other terms. After inspecting output we found that most of the other terminal nodes contained multiple clusters, with the lowest severities in the "no attorney" branches.

Figure 3.7 Severity Tree Using Several Cluster Variables as Predictors



4 Using R to Text Mine the 2008 CAS Quinquennial Membership Survey

Braithwaite (2009) provides a short overview of the survey results in the August 2009 *Actuarial Review*. The full report of the survey task force, Braithwaite et al. (2009) is available in the Summer 2009 CAS *E-Forum*. Survey responders provided extensive written comments. Now we will apply our text-mining R skills to summarizing the free-form text responses to the question, "What are the top two issues affecting actuaries and the CAS in the next five years?"

4.1 Preprocessing the Survey Data

First, we'll read in the text and create a document collection, or Corpus.

```
> txt.csv <- read.csv(file="C:/temp/Top 2 Issues.txt", header=FALSE)

>txt <- Corpus(DataframeSource(txt.csv), dbControl=list(useDb=TRUE, dbName="txtcsv",
dbType="DB1"))

>inspect(txt[1:2])

[[1]] A crisis that could affect our ability to regulate ourselves.

 [[2]] A need to deal more thoroughly with non-traditional risk management approaches
```

Next, we'll apply some transformations, converting to lower case and removing stopwords in our 336 sample documents. Notice from the output above that we'll have to do something about

removing punctuation from our survey written comments. The text is particularly free-form, including punctuation, etc., so we'll need to use the **gsub** function to take out slashes, backslashes, dashes, parentheses, periods, etc., to allow the words to stand alone. (Note that in R we must use a double-backslash to escape characters as the backslash is itself a reserved character.)

Next we'll create a document by term matrix, or dtm. The resulting matrix is quite sparse, with the 336 rows representing documents and the 489 columns for each term. We can tighten up our dtm by restricting ourselves to the more frequently occurring terms.

```
>dtm <- DocumentTermMatrix(txt)
```

Note that this matrix has 336 rows and 489 columns, so we want to remove sparse terms as follows:

```
>dtm3 <- removeSparseTerms(dtm, 0.99)

>nrow(dtm3); ncol(dtm3)

[1] 336

[1] 79
```

Before analyzing the data further we can export the term-document matrix (for possible use in other software) using R's write.csv function. It is also helpful for many analyses to convert the matrix into a data frame (an R database).

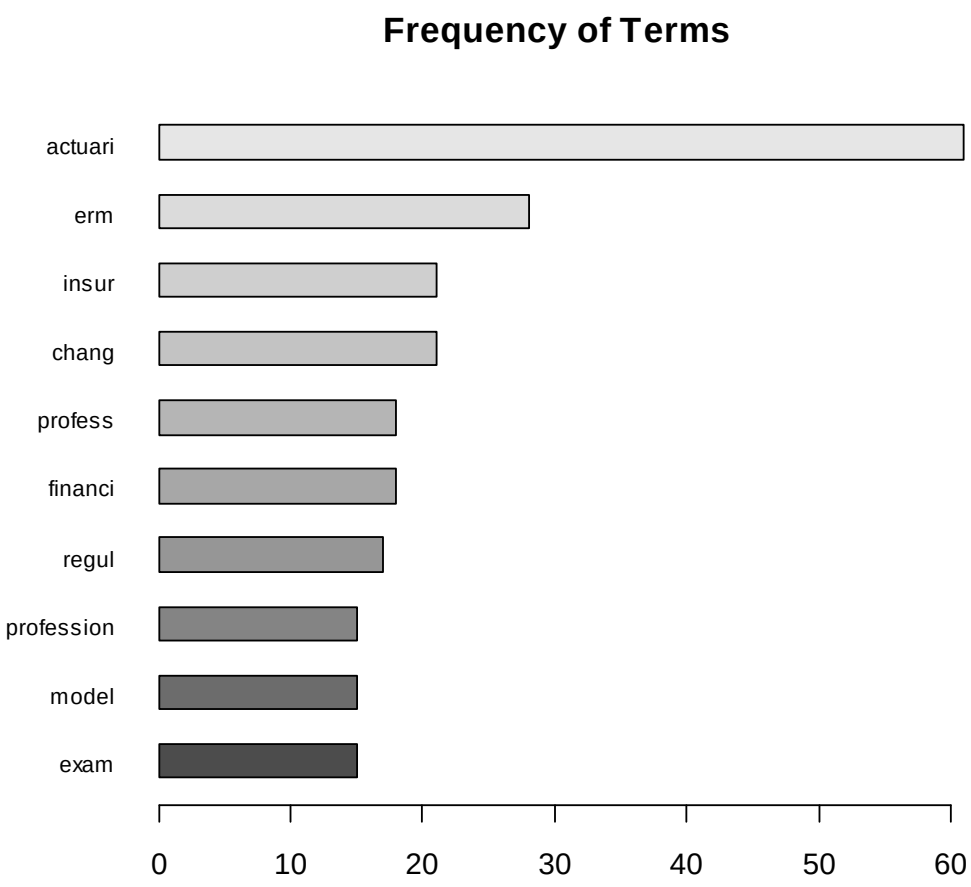
```
>dtm2<-data.frame(dtm)
```

4.2 Statistical Analyses of Survey Data

Next we examine the frequency of terms in the survey data. The **sapply** function is used to sum each column and count the number of responses containing the term. This is then sorted and the top 10 terms are plotted in Figure 4.1:

```
>numwords <- 30
>v <- as.matrix(sort(sapply(top2, doit),decreasing=TRUE)[1:numwords],
colnames=count);v[1:numwords]
>w <- rownames(v); length(w); w
>require(grDevices); # for colors
>x <- sort(v[1:10,], decreasing=FALSE)
>barplot(x, horiz=TRUE, cex.names=0.5, space=1, las=1, col=grey.colors(10), main="Frequency of
Terms")
```

Figure 4.1



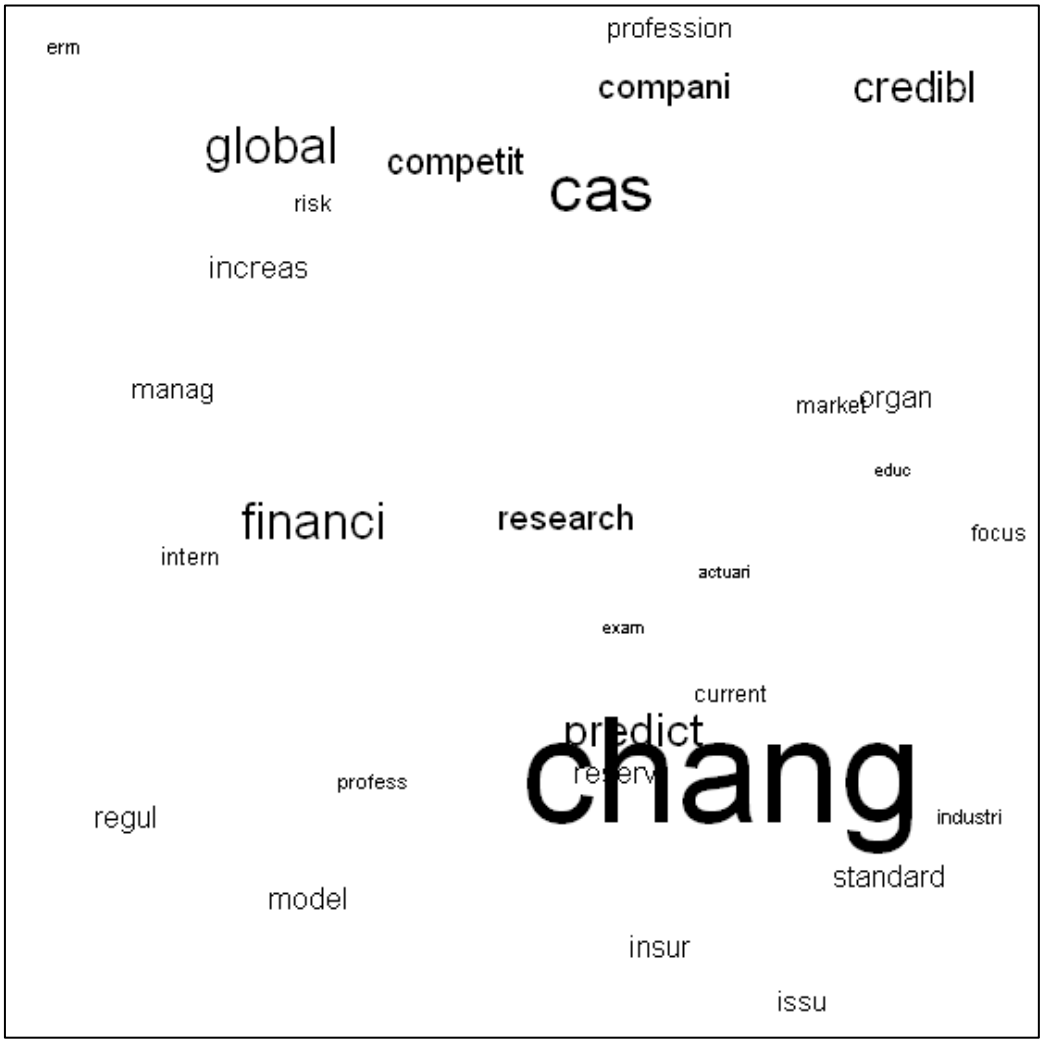
Another way to profile collections is to use a "tag cloud" or a "word cloud" or "weighed list," a list where the font size or color of the words represents the term frequency. Tag clouds were originally popular on photo collections to index the content of the photos. A second variation uses random placement of the words. A third jazzy variation in the "fun" package creates a spinning 3-D tag cloud for an html page using flash and javascript.

In Figure 4.2, the package "snippets" is used to randomly plot terms, with the font size representing the frequency of the term. A number of themes are revealed by the frequency graphs including change, ERM, regulation, credibility (of the profession), and prediction.

```
>require(snippets)

>set.seed(221)
```

```
>x=runif(length(w))

>y=runif(length(w))

>plot(x, y, type="n", axes=FALSE)
```

Figure 4.2 Cloud Map of Word Frequency in Survey



The most frequently occurring word in the survey response is "actuari". This is not surprising, as the respondents are actuaries commenting about their profession. It is common in text mining to drill down on some text by extracting "snippets" or some of the text associated with a word of interest. In R, the subset function can be used to find and print the text in responses containing the word actuary.

```
>rownames(subset(top2, actuari>=1)) #which rows have actauri>=1

>txt.actuari <- txt.csv[rownames(subset(top2, actuari>=1)),]

>txt.actuari; #print out records with actuary
```

Table 4.1 displays a sample of the text associated with the word "actuary." It is clear that a number of different topics are revealed, including professionalism/reputation of actuaries, relevance of actuaries, education, and the financial crisis.

Table 4.1 Text associated with term "actuary"

Actuarial Malpractice
Actuarial Malpractice
Actuarial Students of today are not good communicators/executive material.
Actuaries regain status as "masters of risk"
Relevance of actuaries as providers of solutions for various business problems.
Relevance to the practicing actuary
Relevance to the practicing actuary
Reputation issues arising from current financial crises
reputation of actuarial profession and professionals
Rise of other International actuarial organizations
The current financial crisis - actuaries need to protect their role in insurance and risk management
Training and educating actuaries

4.3 Unsupervised Methods for Content Analysis

We applied the k-means clustering to the GL claims data. In this section we will illustrate another clustering technique, hierarchical clustering. We will use the technique to identify common themes in the survey responses. In this illustration, hierarchical clustering will be applied to the terms or columns, rather than the records (responses). Note that the transpose function (for instance **x'**<- t(term_document_matrix)) must be applied to the data before clustering. Hierarchical clustering is a recursive procedure that sequentially groups terms into larger and larger groups, by combining together terms with the smallest dissimilarity. This will group words that tend to occur

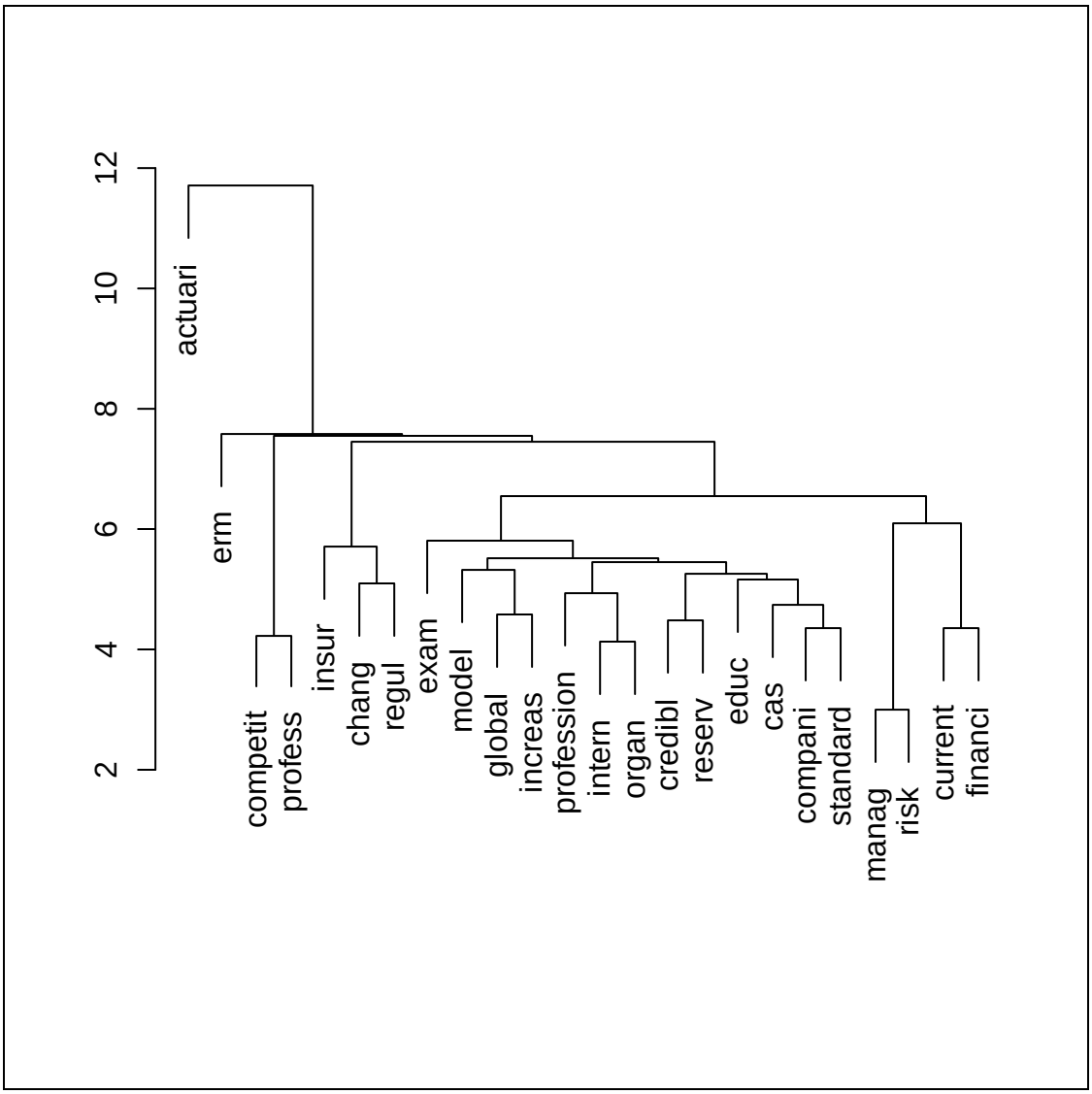
together into the same groups. Hierarchical clustering produces a graph that looks like a tree. Each level of the tree displays a step in the sequence of combining terms into larger(smaller) groupings. The largest group containing all the data is displayed at the top of the graph. At the bottom of the tree, each term is its own group.

We note that change and regulation (changes in regulation) group together as do current and research (research current topics). It appears that the term "actuari" seems to separate early from all other terms. In addition the term "increas" is associated with both education and research. We also note that "professio" and "profession" occur as separate words. This is an artifact of the stemming function where profession was stemmed to "profess", while professional was stemmed to "profession".

The word "intern" (international) is associated with the stemmed professional, while the word "competition" is associated with profession (i.e., competition from other professions). Other topics displayed on the clustering tree are exams, reserves, and modeling. The term "financial" also appeared frequently. Using the **findAssoc** function to find terms that occur frequently with "financial," we find financial crisis and financial analysis, including dynamic financial analysis are common themes.

The tree graph can guide the user in determining how many groupings to maintain. That is by examining the tree, the analyst can group together terms that logically seem to belong together. Alternatively, the number of groups can be subjectively determined. For instance, if one decides to use only two groups, we might split the data into responses that contains the word "actuary" versus everything else. Unfortunately the two grouping scenario would not be very informative, so a larger number of groups would be needed.

Figure 4.3 Hierarchical Cluster of Survey Words



The next technique we apply to the survey data is principal components. Principal components is an unsupervised method that is used for variable reduction. The original variables (in our case, after removing sparse terms, there are 35 terms remaining) are replaced by a smaller number of variables that capture the essential features of the original variables. For instance, the words "risk" and "manage" frequently occur together, and the two can be replaced by one term or "component" in principal components terminology representing the concept "risk management". A description of the methodology used in principal components techniques is outside the scope of this paper (see Kim, 1978 for more information).

Note that the correlation matrix between terms is a key input into the calculation and highly correlated terms tend to be associated with the same factor.

```
>require(stats)
>prcomp(top2, scale=TRUE)
>summary(prcomp(top2, scale=TRUE))
```

The key information for the analysis is the new "components" or variables estimated from the procedure and the "loadings," or coefficients of each term on the component. Each component is essentially a weighted average of some of the terms, and the terms that have high loadings are the key contributors to the component and can be used in identifying the concept underlying each component. For instance the words "research" and "practical" tend to have a high loading on the first component, suggesting that they represents the concept "practical research". To identify the concept associated with each factor, it is necessary to access the loading of each term on the top factors (we used output statistics to retain the top 15 components). The loading is part of the principal components object referenced with "PrincipalComponentsObject$rotation". The following code finds and prints out the top four terms associated with each factor. These are used to construct Table 4.3, the theme associated with each factor.

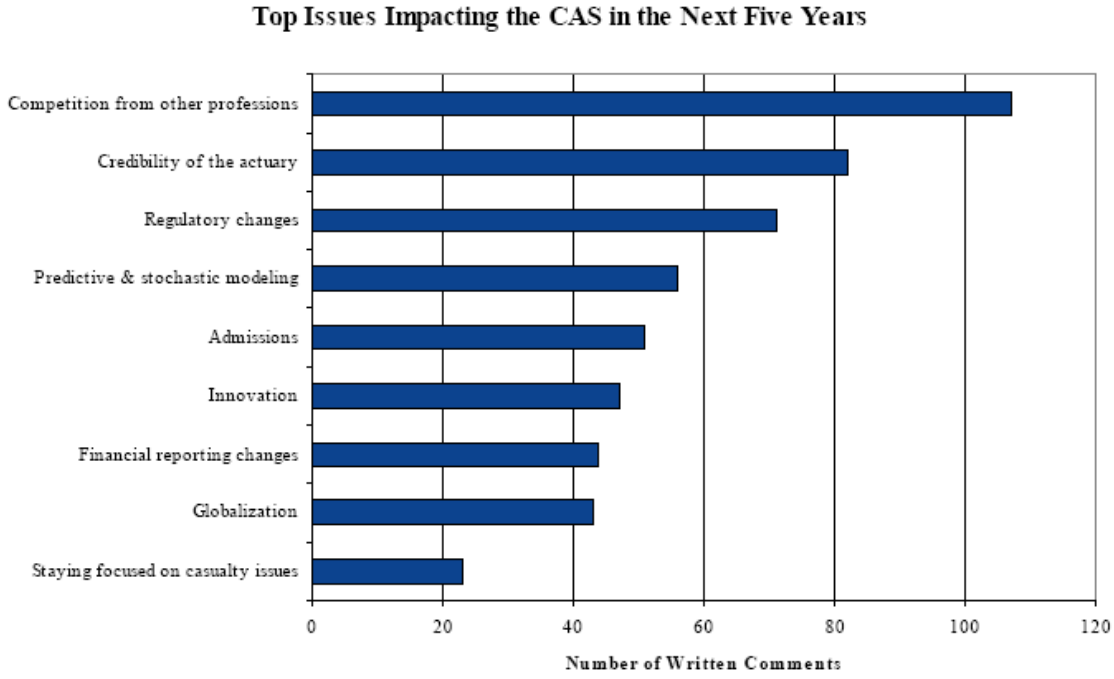
```
>.for (i in 1:15) {

 >  top4[[i]] <- sort(survey.prcomp$rotation[,i], decreasing=TRUE)[1:4]}

>top4.
```

Table 4.2: Concept Associated With Each Component

Component	Concept
1	practical research
2	predictive modeling
3	maintain standards
4	demand for actuarial skills
5	risk management
6	exam structure
7	actuarial organization
8	reputation/financial issues
9	economic capital
10	competition - other professions
11	opportunities for actuaries
12	capital
13	predictive modeling
14	globalism/competition
15	reserving and credibility

These concepts show a significant overlap with the top issues identified in the write-up of the survey published by the task force which are displayed in the graph below. In order to summarize the responses, a CAS volunteer examined classified and tabulated responses. While this review provides context that automated procedures does not provide, nonetheless, the text mining procedures were able to identify key themes in the survey response. In addition, text mining uncovered additional themes, such as economic capital and risk management.

Figure 4.4: Top Issues From Quinquenial Survey Summary



5 Conclusions

In the previous three sections, we provided an introduction to utilizing Perl and R for text mining, including brief examples of code that the programmer needs to successfully use these tools. Text files containing the code used in the paper will be posted with the paper. Using two sample text databases we showed that an analyst can:

- Extract information from an accident description field that can be use to model the propensity for attorneys to be involved in a claim and claim severity
- Analyze survey response data to efficiently identify major themes

A goal of this paper is to make it easier for those interested in text mining to use open source software in their work. This paper is not intended to replace reference books that introduce the two languages (which we encourage the reader to acquire). However, without specific guidance focused on text mining, these tools can be challenging to use. In Table 5.1 we compare the capabilities of the two open source languages for text mining. Table 5.1 presents the procedures we illustrated for each language.

Table 5.1 Features of Procedures Illustrated

Task	Perl	R
parse data	yes	yes
convert to lower case	yes	yes
eliminate punctuation	yes	yes
eliminate stopwords	yes	yes
stemming	yes	yes
compute frequencies	yes	yes
compute correlations	yes	yes
cluster terms	no	yes
apply multivariate models	no	yes
efficient for large databases	yes	no

Acknowledgment

The authors thank Vicky Tuite for her editorial and debugging assistance and Edward Vandenberg for his editorial assistance.

Supplementary Material

Survey datasets and R code will accompany this paper

Appendix A: Perl Procedures with Glossary of Perl Terms

This section is provided to assist the reader in understanding some of the unique structures and variables used in the Perl language. Additional online help for beginners can be found at http://www.perlmonks.org/index.pl?node=Tutorials#Getting-Started-with-Perl. The section on "strings" is particularly helpful with text manipulations, and the section on Pattern Matching, Regular Expressions, and Parsing is a good resource on parsing.

In the text below, Perl code is displayed in italics.

Input Files

The following code opens a file for printing and prints the data. The directory and file name must be specified. Note the use of the chomp function to remove end of line characters.

```
$TheFile="c:\perl\GLACC.txt";
open (INFILE, $Thefile);
while <INFILE> {
chomp; # eliminate end of line character
$Sentence=$_;  # read text into new variable, one line at a time
print "$Sentence\n"; } # print line
```

Files can also be input from the command line when running a program as in:

```
perl progname.pl <datafile> <outputfile>
```

To open the file supplied at the command line the following code is needed:

```
open(MYDATA, $ARGV[0]) or die("Error: cannot open file '$ARGV[0]'\n");
# opens data file
open(OUTP, ">$ARGV[1]") or die("Cannot open file '$ARGV[1]' for writing\n"); # opens output file
```

Printing

Many of the examples used in the paper print directly to the computer screen. The command to do this is "print". Thus

print "hello";
prints the word hello to the screen.

print "hello\n";
prints hello and a carriage return, so that the next printed line will be on a new line.

To print to a file, the programmer must first open a file for output. The character > before the file name denotes that the file is opened for output:
open(OUTP1, ">OutInd1.txt") or die("Cannot open file for writing\n");
will open the output file "OUTP1" or print an error message if the file cannot be opened.
print OUTP1 "hello\n";
will print the word hello along with a carriage return to the file.

When printing to an output file, the command "printf" rather that "print" can also be used.

Looping
The most common looping command in the code in this paper is the "while" command. Its structure is:

while (expression is true) {code to perform}.

Thus:

```
while($line = <MYDATA> ){
code to perform tasks
}
```

will read each line of data from the file MYDATA and assign each line, one at a time to the string variable $line. The code within braces will then (usually) perform other tasks with the data, such as print it out. The code will stop executing when the end of the file is reached.

Another common looping structure is "foreach".

```
foreach $word (@words) {
 code to perform tasks
        }
```

will loop through each word in the array @words and perform the tasks specified in the code within brackets, such as replace unwanted characters or print the word. Again the brackets are used as with the while structure to specify the code that is executed within the loop.

Regular Expressions - In the body of the paper, regular expressions were introduced as a way to pattern match a text string. Below is a guide to special variables and characters that are regularly used in Perl regular expressions:

/ /
The foreword slash is a delimiter used in pattern matching. Within it the characters defining the pattern are specified. Thus split (/ /, $Test) will split a string expression using the space as a separator.

\
A single backward slash is used as an escape operator. Thus /(\.)/ when used in a regular expression denotes the period. The \ must be used with some special characters, such as the period and the backwards slash itself.

[ ] brackets are used within expressions as parentheses, i.e., to subset. See the next example.

^ denotes not in a regular expression. Thus the code:

([^.]\.)*

denotes a text string that is not period, such as words or spaces, followed by a period. This could be used to find a regular sentence structure that ends in a period. Note that the * operator denotes zero or more of the character preceding it (i.e., of non-period characters).

$_
is a special variable that holds the current line of text read in from a file by the Perl.

$1
is a special match variable. It contains all text from the left most parenthesis in the end of the last match. The sample code following the special variable *g* matches the text up to the first period (and also illustrates the period and g modifier).

.
The period is a wildcard character. If one wanted to match the letter a followed by anything followed by d, one would use */a.d/*. For instance, the word "and" would be a match.

g is a modifier that stands for "global" and indicates a need to search the entire record of test. Below is an example using the previous two terms.

$Test = "A crisis that could affect our ability to 'regulate'. ourselves.";# a test string

```
while ( $Test =~  /([^.]*\.)/g )  {
 print "$1 \n"; }
```

It prints out:

A crisis that could affect our ability to 'regulate'.

=~
is similar the equal operator. It binds a pattern match on the right to a variable on the left.

\w
denotes one letter or number. Letters only, including both small and capital, are denoted */a-zA-Z/.*

\w(2)
denotes two alphanumeric characters.

\w(n)
denotes *n* alphanumeric characters.

\w+
denotes one or more letters or numbers.

\d
denotes one digit. It is the same as [0-9].

\d+
denotes one or more digits.

\s+
denotes one or more spaces.

\b
denotes a word boundary, such as the space or tab characters.

\D
denotes any non-digit character. It is the same as [^0-9].

\W
denotes any nonalphanumeric character.

split(/Regular expression/,text string or variable)
is used to split a longer string, such as a sentence or paragraph into its component word. It is common to use the space or one or more spaces (specified as [\s+]) as the delimiter. An example is

$StringVar="Ability of members to prove they are more than just number crunchers";
split(/ /, $StringVar)

Appendix B

Example Code Using Perl Regular Expressions

I Simple parsing programs

Parse1.pl

```
#!perl -w
# Program Parse1.pl
# This program provides a simple example of parsing
# It uses the space as the separator in parsing words in the string
# Set scalar variable to text string
$Response = "Ability of members to prove they are more than just number crunchers";
@words =split (/ /, $Response); # parse words and put in array
foreach $word (@words) {
print "$word\n";
}  # print to screen, one word per line from array
```

Parse2.pl

```
#!perl -w
# Parse2.pl
# Program to parse text string using one or more spaces as separator
$Response = "Ability  of members  to prove they are more than just number crunchers";
@words =split (/\s+/, $Response); #parse words in string
# Loop through words in word array and print them
foreach $word (@words) {
print "$word\n";
}
```

Parse3.pl

```
#!perl -w
# Parse3.pl
# Program to parse a sentence and remove punctuation
$Test = "A crisis that could affect our ability to 'regulate' ourselves.";# a test string with punctuation
@words =split (/[\s+]/, $Test); # parse the string using spaces
# Loop through words to find non punctuation characters
foreach $word (@words) {
while ( $word =~  /(\w+)/g )  {
# match by 1 or more alphanumeric characters.  These will be the words excluding punctuation
print "$1 \n"; #print the first match which will be the word of alphanumeric characters
}
}
```

Simple Search Program

SearchTarget.pl

```
$target = "(homeowner)";
# initialize file variable containing file with text data
$TheFile ="GLACC1.txt";
open(INFILE, $TheFile) or die "File not found"; # open the file
# initialize identifier variables used when search is successful
$i=0;
$flag=0;
# read each line
while(<INFILE>) {
 chomp;
++$i;
# put input line into new variable
$Sentence = $_;
# parse line of text
@words = split(/[\s+]/,$Sentence);
$flag=0;
foreach $x (@words) {
if (lc($x) =~ /$target/) {
$flag=1;
 }
}
# print lines with target variable to screen
print "$i $flag $Sentence \n";
}
```

Appendix C

Code for Simple Text Statistics

I Frequency of Word Lengths

Length.pl

```
#!perl -w
# Enter file name with text data here
$TheFile ="Top2Iss.txt";
# open the file
open(INFILE, $TheFile) or die "File not found";
# read in one line at a time
while(<INFILE>) {
 chomp; # eliminate end of line character
 s/[.?!"()'{},&;]//g; # replace punctuation with null
 s/\// /g; # replace slash with space
  s/\-//g; #replace  dash with null
  s/^ //g; #replace beginning of line space
 print "$_\n"; # print cleaned line out
 @word=split(/[\s+]/);  # parse line
# count length of each word in array @count
            foreach $x  (@word) {
             $count[length($x)] +=1 ;}
  }
$mxcount=$#count;
# print out largest word size and frequency of each count
```

```
print "Count $mxcount\n";
for ($i = 0; $i <= $#count; ) {
# does word of that size exist?
if ( exists($count[$i])) {
print "There are $count[$i] words of length $i\n";
}
$i += 1;  # increment loop counter
}
```

II Zipf's Law – Word Frequencies

Testhash.pl

```
#!perl -w
# Testhash.pl
# Usage:  testhash.pl <datafile> <outputfile>
# input datafile must be present and a command line arg such as Top2Iss.txt
open(MYDATA, $ARGV[0]) or die("Error: cannot open file '$ARGV[0]'\n");
# output datafile must be present and a cmd line arg
open(OUTP, ">$ARGV[1]") or die("Cannot open file '$ARGV[1]' for writing\n");

print OUTP "Output results for ".$ARGV[0]."\n";

# read in the file, get rid of newline and punctuation chars
while( $line = <MYDATA> ){
            chomp($line);
# eliminate punctuation
            $line =~ s/[-.?!"()'{}&;]//g;
            $line =~ s/\s+/ /g;
            @words = split(/ /,$line);

            foreach $word (@words) {
        ++$counts{lc($word)};
            }
}
# sort by value (lowest to highest using counts for the key)
# and write the output file and screen

foreach $value (sort {$counts{$a} cmp $counts{$b} }
          keys %counts)
{
# print the word and the count for the word
            print "$value $counts{$value} \n";
            print OUTP "$value $counts{$value} \n"
}


# close the files
close MYDATA;
close OUTP;
```

Appendix D: Term-Document Matrix and Stop Words

Program to Create Term-Document Matrix.
TermDocMatrix.pl

```
#!perl -w
#
# Program TermDocData.pl
# This program computes the term-document matrix
# a key part is to tabulate the indicator/count of every term - usually a word
# it may then be used to find groupings of words that create content
# This would be done in a separate program
# Usage:  termdata.pl <datafile> <outputfile>
$TheFile = "Top9.txt";
#$Outp1 = "OutInd1.txt";
# open input file with text data
open(MYDATA, $TheFile ) or die("Error: cannot open file");
# open first output file
open(OUTP1, ">OutInd1.txt") or die("Cannot open file for writing\n");
# open second output file
open(OUTP2, ">OutTerms.txt") or die("Cannot open file for writing\n");
# read in the file each line and create hash of words
# create grand dictionary of all words
# initialize line counter
  $i=0;
# loop through data and convert to lower case and add to dictionary using hash
while (<MYDATA> ){
              chomp($_);
              $_ =~ lc($_);
              s/[-.?!"()'{}&;]//g;
              s/\s+/ /g;
                        @words = split(/ /);
                         foreach $word (@words) {
                          ++$response[$i]{lc($word)}; # get freq of each word on line
                          ++$granddict{lc($word)};
                         }
                           ++$i;
}
# record no of lines in file
$nlines = $i-1;
print " no of lines is $nlines\n";
# print statitics to screen
for ($j=0; $j<= $nlines; ++$j ) {
        print "$j ";
        foreach $word (keys %{$response[$j]} )
        {  print "$word, ${response[$j]{$word}},"; }
    print "\n";
}
# compute term-document matrix
# if term exists on record count frequency, else record gets a zero for the ter,
for $i (0..$nlines) {
      foreach $word (keys %granddict) {
                if (exists($response[$i]{$word}))
```

```
                {
                ++$ indicator[$i]{$word}; }
                 else
                {
                 $indicator[$i]{$word}=0;
                      }
print  OUTP1 "$indicator[$i]{$word},";
                 }
   print OUTP1 "\n";
}
# print stats to file
foreach $word (keys %granddict) {
          print OUTP2 "$word,$granddict{$word}\n";
          }
# close the files
close MYDATA;
close OUTP1;
close OUTP2;
```

Program to Eliminate Stop Words When Creating Term-Document Matrix. Stopwords.pl

```
#!perl -w
#
# StopWords.pl
# This program eliminates stop words and  computes the term-document matrix
# a key part is to tabulate the indicator/count of every term - usually a word
# it may then be used to find groupings of words that create content
# This would be done in a separate program
# Usage:  termdata.pl <datafile> <outputfile>
$TheFile = "Top2Iss.txt";
#$Outp1 = "OutInd1.txt";
open(MYDATA, $TheFile ) or die("Error: cannot open file");
open(OUTP1, ">OutInd1.txt") or die("Cannot open file for writing\n");
open(OUTP2, ">OutTerms.txt") or die("Cannot open file for writing\n");
# read in the file each line and create hash of words
# create grand dictionary of all words
# initialize line counter
  $i=0;
while (<MYDATA> ){
               chomp($_);
               s/[-.?!"()'{}&;]//g;
               s/^ //g;
               s/,//g;
               s/\d/ /g;
               s/(\sof\s)/ /g;
              s/(\sto\s)/ /g;
              s/(\sthe\s)/ /g;
              s/(\sand\s)/ /g;
              s/(\sin\s)/ /g;
              s/(The\s)/ /g;
              s/(\sfor\s)/ /g;
```

```
            s/(\as\s)/ /g;
            s/(A\s)/ /g;
            s/(\sin\s)/ /g;
            s/(\swith\s)/ /g;
            s/(\san\s)/ /g;
            s/(\swith\s)/ /g;
            s/(\sare\s)/ /g;
            s/(\sthey\s)/ /g;
            s/(\sthan\s)/ /g;
            s/(\sas\s)/ /g;
            s/(\sby\s)/ /g;
             s/\s+/ /g;
            if (not /^$/) { #ignore empty lines
                        @words = split(/ /);
                        foreach $word (@words) {
                        ++$response[$i]{lc($word)};
                        ++$granddict{lc($word)};
                        }
                          ++$i;
              }
}
$nlines = $i-1;
for $i (0..$nlines) {
     foreach $word (keys %granddict) {
                if (exists($response[$i]{$word}))
                {
                ++$ indicator[$i]{$word}; }
                 else
                {
                 $indicator[$i]{$word}=0;
                      }
print  OUTP1 "$indicator[$i]{$word},";
                 }
  print OUTP1 "\n";
}
foreach $word (keys %granddict) {
          print OUTP2 "$word,$granddict{$word}\n";
          }
# close the files
close MYDATA;
close OUTP1;
close OUTP2;
```

APPENDIX E

MATCHLINE.PL

#PROGRAM MATCHLINE.TXT TO SEARCH FOR THE PHRASE THAT MOST CLOSELY MATCHES A PHRASE.

```
#!perl -w
# matchline.pl
# Usage:  matchline.pl <datafile> <in phrase file > <outputfile>
# datafile must be present and a cmd line arg
# create a dictionary of all words in a file and alphabetize them
open(MYDATA, $ARGV[0]) or die("Error: cannot open file '$ARGV[0]'\n");
open(INPH, $ARGV[1]) or die("Error: cannot open file '$ARGV[1]'\n");
open(OUTP, ">$ARGV[2]") or die("Cannot open file '$ARGV[2]' for writing\n");

print OUTP "#Output results for ".$ARGV[0]."\n";

$nphr = 0;

# read in the file, get rid of newline and punctuation chars
while( $line = <MYDATA> ){
        chomp($line);
        $line =~ s/[-.?!"()'{}]//g;
        @words = split(/ /,$line);

        foreach $word (@words) {
        ++$granddict{lc($word)};  # this is the hash assignment  lc is lowercase
        ++$tf[$nphr]{lc($word)};
        }
        $nphr++;
}

# Read in the input phrase from file
$linecnt = 0;
while( $line = <INPH> ){
        print OUTP "Input Phrase: " . $line . "\n";
        chomp($line);
        $line =~ s/[-.?!"()'{}]//g;
        @words = split(/ /,$line);
        $linecnt++;
}
# FIXME if ($linecnt != 1 ) die("Input phrase file must contain only 1 line");
print "inph linecount ". $linecnt . "\n";

foreach $word (@words){
        ++$inph{lc($word)};
```

```
}

#print %tf[0];
# compute document frequencies
foreach $word (sort (keys(%granddict))){
	$sum = 0;
	for $i (0 .. $nphr) {
		#print $word . "\n";
		if ( exists $tf[$i]{$word}  ) {
			++$sum;
		}
	}
	$df{$word} = $sum;
}

# Step 3 Compute tf-idf weights
$n = $nphr + 1;
foreach $word (sort keys %granddict) {
	for $i ( 0 .. $nphr) {

		if (exists $tf[$i]{$word} ) {
			$tf_val = $tf[$i]{$word};
		}
		else {
			$tf_val = 0;
		}
		#print OUTP "Word ". $word. " " . $tf_val . " df: " . $df{$word}. "\n";
		$weight[$i]{$word} = $tf_val * log($n / $df{$word}) / log(2);
		#print "Weight ". $weight[$i]{$word}. " " . "\n";
	}
}

# Compute weight of input phrase
foreach $word (sort keys %granddict) {
	if (exists $inph{$word} ){
		$tf_val = $inph{$word};
	}
	else {
		$tf_val = 0;
	}

	$inph_weight{$word} = $tf_val * log($n / $df{$word}) / log(2);
}

# Step 4 Normalize the column of weights
for $i ( 0 .. $nphr - 1){
	$len2 = 0;
	foreach $word ( sort keys %granddict){
		$len2 += $weight[$i]{$word}**2;
		#print $word . " len2 " . $len2 . "\n";
	}
	$len = sqrt($len2);
	foreach $word (sort keys %granddict ){
```

```
                $unit[$i]{$word} = $weight[$i]{$word}/$len;
        }

}

# Normalize input weight so it can be compared with the others
foreach $word (sort keys %granddict){
        $len2 += $inph_weight{$word};
        #print "inph ". $word . " len2 " . $len2 . "\n";
}

$len = sqrt($len2);
foreach $word (sort keys %granddict ){
        $inph_unit{$word} = $inph_weight{$word}/$len;
}
#Step 5 Compute cosine simularities between input phrase and other phrases
$best = 0;
$best_idx = 0;
for $i ( 0 .. $nphr-1 ){
        $sum = 0;
        foreach $word (sort keys %granddict) {
                $sum += $unit[$i]{$word} * $inph_unit{$word};
        }
        $inph_cosine[$i] = $sum;
        printf "INPH %d  %.5f",$i, $inph_cosine[$i];
        printf OUTP "INPH  %d %.5f \n", $i, $inph_cosine[$i];
        if ($inph_cosine[$i] > $best){
                $best = $inph_cosine[$i];
                $best_idx = $i;
        }
}
printf "\nBest Match %.5f, %d\n", $best, $best_idx;
printf OUTP "\nBest Match %.5f, %d\n", $best, $best_idx;

# reopen the data file to get the best line since we didn't store it to save memory
open(MYDATA, $ARGV[0]) or die("Error: cannot open file '$ARGV[0]'\n");

$linecnt = 0;
while( $line = <MYDATA> ){
        print $line . " linecount: ". $linecnt . "\n";
        if( $linecnt == $best_idx ){
                #print $line;
                print OUTP "Best Line: " . $line . "\n";
                $linecnt++;
                last;
        }
        else {
                $linecnt++;
        }
}
# close the files
close MYDATA;
close INPH;
```

```
close OUTP;
```

MATCHLINE PROGRAM OUTPUT

Table E-1

Input Phrase: Credibility of the CAS	
0	0%
1	1%
2	27%
3	0%
4	0%
5	1%
6	0%
7	35%
8	31%
9	13%

Best Match 0.34764, 7
Best Line: Credibility of the profession

Table D-2 Program Input

Input Data for Matchline.pl Program

A need to deal more thoroughly with non-traditional risk management approaches
Ability of members to prove they are more than just number crunchers
ability to convince non-insurance companies of the value/skills offered by CAS members.
Ability to help sort out property pricing problems
Actuarial Malpractice (2)
Actuarial Students of today are not good communicators/executive material.
Admission requirements and Membership value in an increasingly globalized socio-economic network
Credibility of the profession
Credibility of the actuarial profession
Credibility of reserve estimates

APPENDIX F

R Resources for Beginners

There are a great number of online resources for learning R. The first place to begin is at the CRAN Web Site: http://cran.r-project.org/. From the main page, lower left, Documentation, there are links to the main R FAQ documents (Hornick, 2009) (see http://cran.r-project.org/doc/FAQ/R-FAQ.html) and R MAC OS X FAQ and R Windows FAQ. We also recommend, for those new to R, that you download the "R-Intro.pdf" file ("Introduction to R" by Venables et al.

There are several mailing discussion lists including: http://www.nabble.com/R-f13819.html, https://stat.ethz.ch/mailman/listinfo/r-sig-mac, and the R Manuals edited by the Core R Development team are a great place to start. The R Manuals include topics such as: An Introduction to R, R Data Import/Export, Extending R, etc.

Contributed Documentation also includes such titles as Using R for Data Analysis and Graphics - Introduction, Examples and Commentary (Maindonald), Simple R (Verzani), R for Beginners (Paradis), R for Windows Users (Wang), Fitting Distributions with R (Ricci), R Reference Card (Baron) etc.

Additional Web sites of interest include:

Learn R at the USGS by (Geissler and Philippi)

http://www.fort.usgs.gov/brdscience/learnR08.htm

Quick-R (Kabacoff)

http://www.statmethods.net/index.html

R Graph Gallery (Francois)

http://addictedtor.free.fr/graphiques/

UCLA Stat Computing Resources to help you learn R:

http://www.ats.ucla.edu/stat/r/

CRAN Main page – Task Views provide an introduction and guidance based on topic areas of

using R in such areas as Econometrics, Graphics, Spatial data, Survival Analysis, or Time Series.

R tutorials Web sites:

http://www.cyclismo.org/tutorial/R/

http://www.math.ilstu.edu/dhkim/Rstuff/Rtutor.html

http://www.stat.pitt.edu/stoffer/tsa2/R_time_series_quick_fix.htm

http://www.econ.uiuc.edu/~econ472/tutorial2.html

http://faculty.washington.edu/tlumley/Rcourse/

The Casualty Actuarial Society's Web Site also has a number of R resources. Among them is the Glenn Meyers article, "The R Programming Language—My 'Go To' Computational Software" (*Actuarial Review,* November 2006). Since programming in the R language has been featured at several seminars and meetings, such as the November 2008 Annual Meeting, handouts on R are also available on the CAS Web Site.

APPENDIX G

Python for Text Mining

A popular alternative to Perl for text processing is the open source language Python. Many professional programmers prefer Python to Perl. Our primary consideration in using Perl in this paper is that the authors had some experience with Perl. In addition, the recently published book, *Practical Text Mining with Perl*, by Bilisoly was an excellent aid to those new to using this language for text mining. This reference was invaluable. A recently published book now provides a comprehensive reference for text mining in Python (Bird et al., 2009).

Python can be downloaded from www.python.org. According to the Python Web Site, "You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs." Python is reputed to be easier to learn than Perl, yet is also a particularly handy tool for processing text data. According to Python's downloadable documentation "While Python has some dark corners that can lead to obscure code, there are relatively few such corners, and proper design can isolate their use to only a few classes or modules." On the other hand, Perl has some special variables and language constructs that can be considered "obscure."

Those wishing to learn more about using Python for text mining can visit the Python Web Site.

A brief overview of using Python in Natural Language Processing is provided by Madnani (2009). This paper introduces Python's natural language toolkit, a Python library that performs many text mining tasks. Finally, the book *Natural Language Processing with Python* (Bird, Klein & Loper, 2009) provides a more detailed guide to text mining in Python. Note that natural language processing libraries will generally perform many of the tasks described in this paper as text mining and natural language processing are similar disciplines with a significant overlap. However, natural language processing tools can perform functions not addressed in this paper such as part of speech tagging.

Both Perl and Python are considered especially useful for text processing. Thus, we do not recommend one over the other, but wish to make the reader aware of Python.

5. REFERENCES

[1.] Bilisoly, R., *Practical Text Mining with Perl*, Wiley Publishing, 2008.

[2.] Brender, A., "The Use of Internal Models for Determining Liabilities and Capital Requirements," *North American Actuarial Journal,* 6:2, 2002, pp. 1-10.

[3.] Bird, Steven, Ewan Klein, Edward Loper, *Natural Language Processing with Python*, O'Reilly Media, 2009.

[4.] Brieman, L., J. Freidman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Chapman & Hal/CRC Pressl, 1993.

[5.] Brieman, L., "Random Forests," *Machine Learning*, 45:1, 2001, pp. 5–32.

[6.] Braithwaite, Nancy, "CAS Quinquennial Membership Survey Results Show Members Seek Practical Focus," *Actuarial Review*, August 2009, http://www.casact.org/newsletter/index.cfm?fa=viewart&id=5804.

[7.] Braithwaite, Nancy, et al., "Report of the 2008 CAS Quinquennial Membership Survey Task Force," CAS 2009 Summer *E-Forum*, http://www.casact.org/pubs/forum/09sumforum/04_Quin_Survey_Report.pdf.

[8.] Crawley, Michael J., *Statistics: An Introduction Using R*Wiley Publishing, 2005.

[9.] Dalgaard, Peter, *Introductory Statistics with R*, Springer, 2008.

[10.] Davi, A., et al., "A Review of Two Text Mining Packages: SAS Text Mining and WordStat," *The American Statistician*, Feb. 2005.

[11.] De Ville, Barry, *Decision Trees for Business Intelligence and Data Mining: Using SAS Enterprise Miner*, SAS Press, 2006.

[12.] Francis, Louise A., "Taming Text: An Introduction to Text Mining", Casualty Actuarial Society *Forum*, Winter 2006.

[13.] Feinerer, I., K. Hornik, and D. Meyer, "Text mining infrastructure in R," *Journal of Statistical Software* 25:5, Mar 2008, http://www.jstatsoft.org/v25/i05.

[14.] Frenz, C., *Pro Perl Parsing*, Apress 2005.

[15.] Himmel, W., U. Reincke, and H. W. Michelmann, "Using Text Mining to Classify Lay Requests to a Medical Expert Forum and to Prepare Semiautomatic Answers," SAS Global Forum, 2008.

[16.] Hoffman. P., *Perl for Dummies,* 4th edition, For Dummies, 2003.

[17.] Inmon, W. and A. Nesavich, *Tapping into unstructured data: integrating unstructured data and textual analytics into business intelligence,* First edition, Prentice Hall Press, 2007.

[18.] Kaufman, L., and P. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, 9th edition, Wiley-Interscience, 1990.

[19.] Kim, J., *Introduction to Factor Analysis*, Sage Publications, 1978

[20.] Kolyshkina, I. and M. van Rooyen, "Text Mining For Insurance Claim Cost Prediction," Presented at the XVth General Insurance Seminar Institute of the Actuaries of Australia, October 2005.

[21.] Manning, C. and H. Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

[22.] Madnani, Nitin. "Getting Started on Natural Language Processing with Python," http://www.umiacs.umd.edu/~nmadnani/pdf/crossroads.pdf (this version revised slightly from the original article in *ACM Crossroads* 13:4, 2007).
[23.] Maindonald, J. H. and J. Braun, *Data analysis and graphics using R*, Cambridge University Press, 2007.
[24.] Mildenhall, S., "Minimum Bias and Generalized Linear Models," *Proceedings of the Casualty Actuarial Society* 1999, Vol. LXXVI, 393-487.
[25.] Weiss, S.M., N. et al., *Predictive Methods for Analyzing Unstructured Information*, Springer, 2005.
[26.] Venables, W.N., and B.D. Ripley, *Modern Applied Statistics with S,* 4th Ed., Springer, 2002.
[27.] Wikipedia, "Text Corpus," Accessed Sep 9, 2009.

Biography(ies) of the Author(s)

Louise Francis is a consulting principal at Francis Analytics and Actuarial Data Mining, Inc. She is involved in data mining projects as well as conventional actuarial analyses. She has a BA degree from William Smith College and an MS in Health Sciences from SUNY at Stony Brook. She is a Fellow of the CAS and a Member of the American Academy of Actuaries. She is the 2008-2011 CAS Vice President-Research & Development. She is a five-time winner of the CAS Management Data, and Information Prize.

Matthew Flynn, PhD is a senior statistician in claim research at Travelers Insurance in Hartford, CT. He is responsible for leading and developing advanced claim analytics at Travelers. He has a degree in finance from Purdue University. He is a frequent presenter at industry symposia and SAS user group meetings.

Data and Disaster: The Role of Data in the Financial Crisis

Louise Francis and Virginia R. Prevosto

Abstract:

Motivation. Since 2007 a global financial crisis has been unfolding. The crisis was initially caused by defaults on subprime loans, aided and abetted by pools of asset-backed securities and credit derivatives, but corporate defaults, such as that of Lehman Brothers, and outright fraud have also contributed to the crisis. Little research has been published investigating the role of data issues in various aspects of the financial crisis. In this paper we illustrate how data that was available to underwriters, credit agencies, the Securities and Exchange Commission (SEC), and fund managers could have been used to detect the problems that led to the financial crisis.

Method. In this paper we show that data quality played a significant role in the mispricing and business intelligence errors that caused the crisis. We utilize a number of relatively simple statistics to illustrate the due diligence that should have, but was not performed. We use the Madoff fraud and the mortgage meltdown as data quality case studies. We apply simple exploratory procedures to illustrate simple techniques that could have been used to detect problems. We also illustrate some modeling methods that could have been used to help underwrite mortgages and find indications of fraud.

Results. In both the Madoff fraud and the mortgage crisis a number of statistical tests could have been applied to uncover fraud and to provide a warning of the deterioration of the quality of mortgages underwritten.

Conclusions. Data quality issues made a significant contribution to the global financial crisis.

Keywords. Data, data quality, financial crisis

I. INTRODUCTION

In the eBook *Risk Management: The Current Financial Crisis, Lessons Learned and Future Implications*, published by the Joint Risk Management Section, a North American actuarial risk management organization, it was stated that, "The current financial crisis presents a case study of a financial tsunami…on what can go wrong. Its ramifications are far reaching and the lessons learned will be embedded in risk management practices for years to come."[1] In this publication a number of factors causing the financial crisis were identified: bubble behavior and mentality, liquidity problems, incentives (i.e., compensation), accounting disclosure issues, insufficient commitment to ERM, and flawed models. In addressing the issues raised with how models were used to price and rate securities based on mortgages, it was pointed out that there were data issues that had an impact on underlying assumptions embedded in the models. For example, the models assumed that housing prices would not decline over time. The following was quoted from a December 2005 report "the risk of national decline in home prices appears remote. The annual decline in HPA (i.e., housing price appreciation) has never been negative in the United States going back to 1992."[2]

[1] From the introduction of JRMS, 2008

[2] Schoolman, 2008

In addition to bankruptcies, forced buyouts, stock market, and bond value declines caused by the subprime meltdown, and the liquidity crunch caused by the crisis brought to light a number of large fraud schemes. One of the most prominent of these was the Ponzi scheme run by Bernard Madoff. What was most interesting was that a whistleblower, after examining data from one of Madoff's clients, first warned the Securities and Exchange Commission (SEC) about the fraud in 2000 and numerous times afterwards (Markopolos 2009).

Though it has not received a lot of publicity, poor data quality played a significant role in the global financial crisis that began to unfold in 2007. In this paper we will examine the role of data in the global financial crisis and in high-profile financial frauds. We will relate specific quality issues to the Actuarial Standard Board's ASOP No. 23 on Data Quality.

This paper will feature two data quality case studies:

1. An evaluation of data from a Madoff feeder fund[3] to illustrate a number of techniques that could have been used to determine that it was fake data.
2. A review of mortgage data—how it was used, and how it could have been used to properly evaluate the riskiness of mortgage loans and the derivatives based on them. The following sources of information will be used in our review:
    a. Demographic data for the U.S. housing market from the Home Mortgage Disclosure Act
    b. Aggregate data on foreclosure rates published in *GH Bank Housing Journa*l (Barth et al.)
    c. The Case-Shiller Home Price Index
    d. A fraud index developed by Interthinx, an ISO business

I. Case Study #1: The Madoff Data

In December 2008, the nation was stunned when Bernard Madoff was arrested for perpetrating one of the world's largest Ponzi schemes. What was remarkable about this fraud was that more than eight years earlier the SEC had been alerted to the fraud by Harry Markopolos. Markopolos, a securities industry executive, testified: "As early as May 2000, I provided evidence to the SEC's Boston Regional Office that should have caused an investigation of Madoff. I resubmitted this evidence with additional support several times between 2000 and 2008."

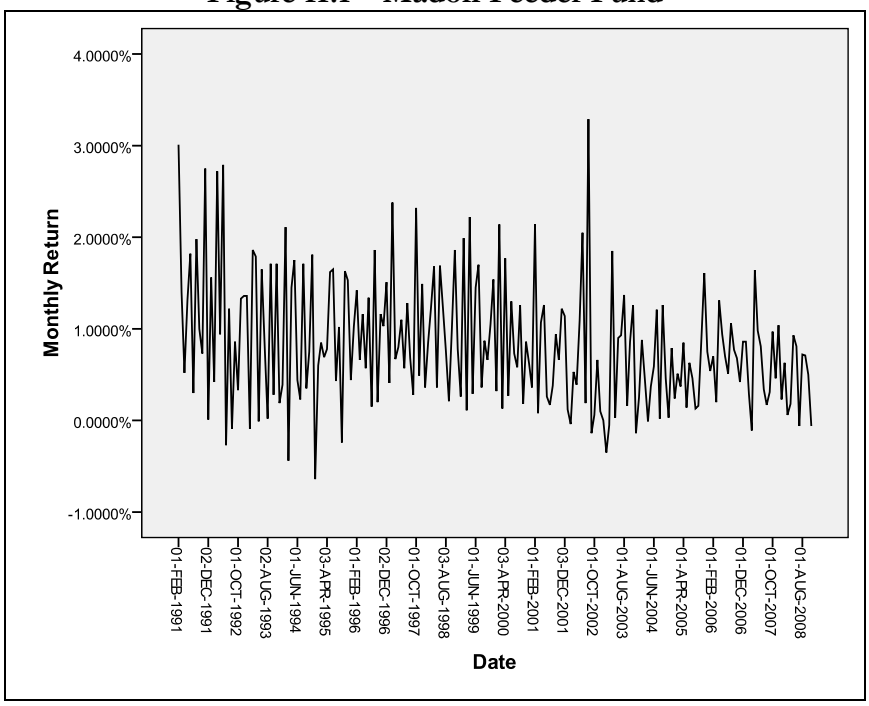
According to ASOP No. 23, "A review of data may not always reveal existing defects. Nevertheless, whether the actuary prepared the data or received the data from others, the actuary should review the data for reasonableness and consistency." The Markopolos testimony, along with

[3] The data is from the hedge funds Fairfield Sentry, Ltd.

the statements of others (Arvedlund 2008), suggest that due diligence was not performed by the "feeder funds" or fund managers, who provided many of the client funds "invested" in the Madoff Ponzi scheme. It also appears that the SEC was lax and did not follow through on what now appears to be obvious clues to the fraud.

Much of our analysis of data will be motivated by the analyses described in the Markopolos testimony (Markopolos 2009). The analysis will be applied to published returns from one Madoff feeder fund. The fund was sold by the Fairfield Greenwich Group. The information was downloaded from the Internet in early 2009. The data is the monthly return for the fund from January 1991 through October 2008. **Figure II.1** displays a graph of the monthly return for this fund.

Figure II.1 – Madoff Feeder Fund



In the testimony, Markopolos described a number of approaches he used to determine that Madoff return data was fraudulent. According to Markopolos, the returns on Madoff's investments were too good to be true. He believed that the "split-strike conversion" strategy that Madoff claimed to use would probably not beat T-Bill returns, especially after expenses are factored in.

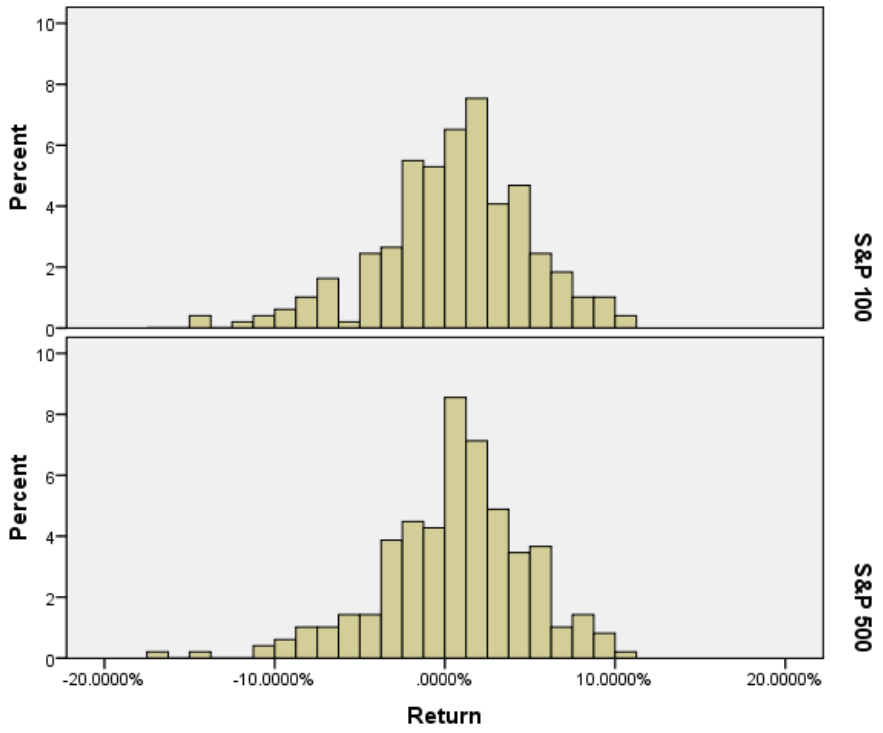
In the analyses in this section we will show that a number of simple graphical and descriptive statistics should have provided red flags or warnings that the Madoff data was fraudulent. These tests include histograms, descriptive statistics and a scatter plot. We also introduce a statistical test, Benford's Law, which is frequently used to detect fraudulent transactions by forensic accountants.

Madoff claimed to have purchased a basket of 30-35 stocks whose returns closely followed the returns of the S&P 100 index. Because it had fewer stocks, this portfolio could be expected to have higher volatility than the index it is tracking. Absent other aspects of the investing strategy to dampen volatility such as the split-strike strategy, which we will address later, Madoff's returns would be expected to share many characteristics with the S&P 100 index.

Test 1: Histogram of Returns[4]

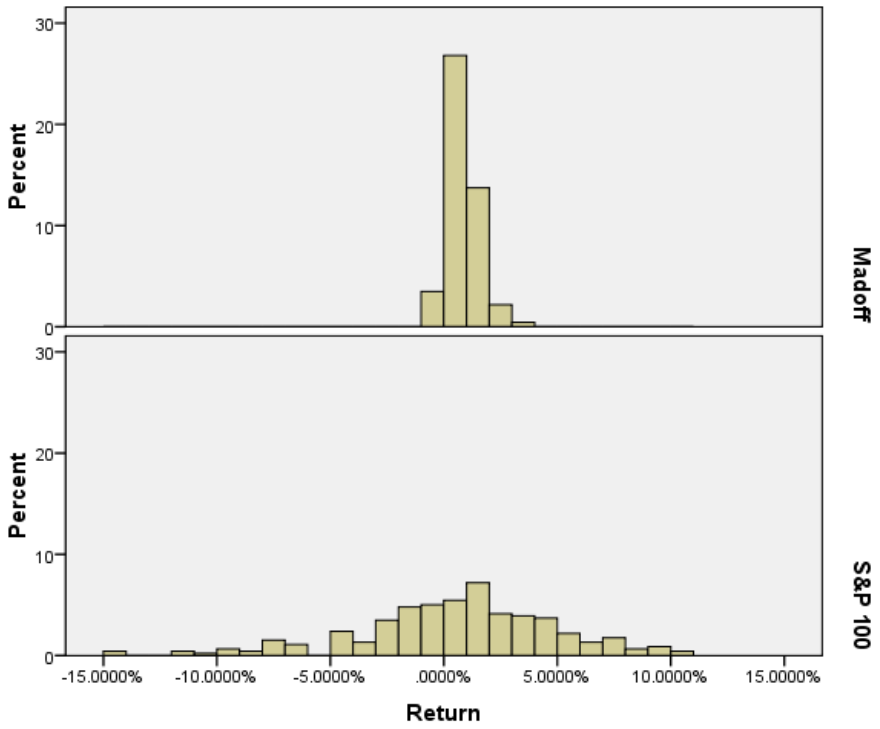
As **Figure II.2** shows, the histograms of the returns for the S&P 500 and its subset of 100 stocks ("S&P 100") are very similar and with an almost bell-shaped distribution. Now compare the histogram of the returns for the S&P 100 and the Madoff Feeder Fund, which was composed of the largest 30-35 of S&P 100 companies (see **Figure II.3**). The Madoff and S&P 100 histograms are dramatically different. The Madoff data is much less dispersed, i.e., it has a much higher peak and much shorter tail than the S&P 100 returns, and has virtually no left tail, suggesting the two are from very different distributions.

Figure II.2 – Histogram of S&P 500 and S&P 100



[4] For Figures II.2 and II.3, the x-axis is the monthly return for the fund shown and the y-axis is the percentage of months with said return. Bins are computed by SPSS software using an automatic statistical rule.

Figure II.3 – Histogram of S&P 100 and Madoff Feeder Fund



Test 2: Descriptive Statistics

Another test Markopolos performed was an examination of simple statistics for a Markopolos fund. We perform a similar analysis below. For comparative purposes, we have provided returns of several stock indices and large mutual funds including:

- The S&P 500. Because it contains 500 stocks instead of the 30-35 in Madoff's fund, it should have lower volatility (standard deviation) than the Madoff data.
- The S&P 100. This is the index Madoff claimed to track. Because it contains 100 stocks instead of the 30-35 in Madoff's fund, it should have lower volatility (standard deviation) than the Madoff data.
- A Balanced Fund that contains a mixture of equities and income producing investments and could be expected to have lower volatility than an equity index (S&P 100). A motivation for investing in a balanced fund is to reduce exposure to risk. A balanced fund is expected to be significantly less volatile and to have somewhat fewer extreme values than a stock-based fund.
- A long-term bond fund that should have the lowest volatility of all the assets as it is composed entirely of bonds.

In producing the following tables, data were limited to returns subsequent to June 1996 because the return series from our comparative mutual funds begins in mid-1996. A surprising result is that the Madoff Feeder Fund has a lower standard deviation than even the bond fund (see **Table II.1**). Indeed, its standard deviation is about 25% of that of the next most volatile asset category. It can also be noted that the Madoff data has a relatively large positive skewness while all the other assets have negative skewness.

Table II.1 – Return Statistics for Different Assets

Asset	Mean	Std. Deviation	Skewness	Kurtosis	N
Balanced	0.46%	2.84%	(0.89)	1.69	149
Long Bond	0.60%	2.40%	(0.36)	2.24	149
S&P 100	0.31%	4.77%	(0.47)	0.47	149
S&P 500	0.30%	4.61%	(0.70)	1.02	149
Madoff	**0.75%**	**0.62%**	**1.01**	**1.25**	**149**

Another statistic that Markopolos commented on was the small number of negative return months for Madoff investments. **Table II.2** displays the negative return statistics for each of the assets. The Madoff fund has a far lower percentage of negative returns than any of the other assets, including a bond fund. According to Markopolos, the probability of such a low percentage of negative return months with a real invested asset is virtually nil.

Table II.2 – Percent of Months With Negative Returns

Asset	Percent of Months
Balanced	39%
Long Bond	37%
S&P 100	44%
S&P 500	42%
Madoff	**6%**
Total	33%

Another point that Markopolos made was that, given a portfolio of 30 to 35 stocks, at least one stock would experience a significant loss[5] during at least one month that would result in a negative return of at least 3% for the portfolio. We see in **Table II.3** that the minimum return for the Madoff fund was a negative 0.6%, well above that of any of the other assets. For the entire 18 years

[5] Significant loss means the stock's value approaches zero.

(as opposed to the 12 years in Table II.2 and II.3[6]) of the Madoff data, the Madoff minimum return was -0.64% versus -14.6% for the S&P 100 for the same period.

Table II.3 – Investment Returns for Different Assets[7]

Asset	Median	Minimum	Maximum
Balanced	0.8%	-11.6%	5.7%
Long Bond	0.9%	-8.7%	11.4%
S&P 100	1.0%	-14.6%	10.8%
Madoff	**0.7%**	**-0.6%**	**3.3%**

Test 3: Benford's Law

Benford's Law is a little known statistical procedure that is used to detect accounting fraud. It is particularly useful in detecting "fake" data. The test is based on the distribution of the first digits of numbers. For instance, Triola (2002) notes that the first digits of amounts on checks tend to follow Benford's law. **Table II.4** below displays the theoretical distribution.

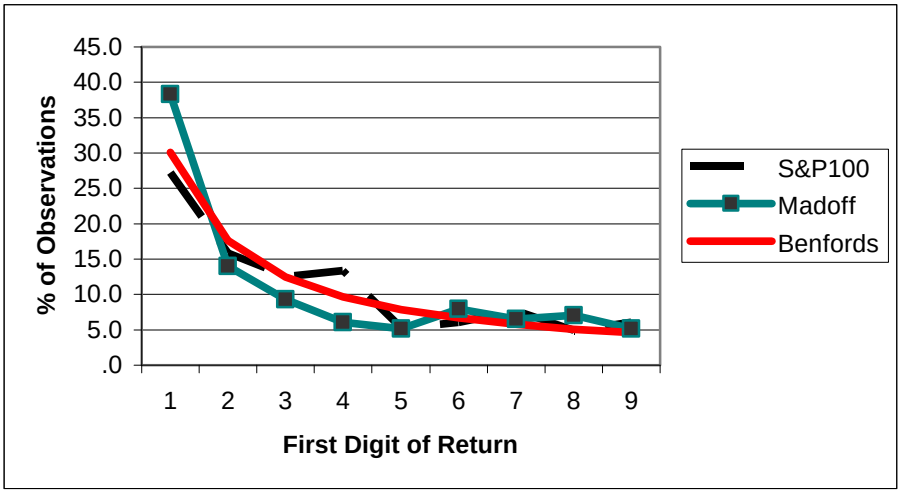
Table II.4 – Benford's Law

First Digit	Proportion
1	30.1%
2	17.6%
3	12.5%
4	9.7%
5	7.9%
6	6.7%
7	5.8%
8	5.1%
9	4.6%

A common mistake of people committing fraud is to assume that digits are uniformly distributed. Thus, the perpetrators of fake data tend to fabricate returns whose first digits fluctuate randomly around a discrete uniform distribution. **Figure II.4** displays the distribution of the first digit of the Madoff Feeder Fund's returns compared to the theoretical Benford's Law distribution. For comparison purposes, the distribution of S&P 100 returns are displayed, as we assume these represent the distribution of digits of a "true" random return series.

[6] As noted above, some of our mutual fund data used in the comparisons began in 1996, therefore for these tables we used the same time periods from the Madoff data, thus excluding Madoff returns from 1991 through June 1996.
[7] Percents shown are returns as a percent of assets.

Figure II.4 – Distribution of First Digit of Return



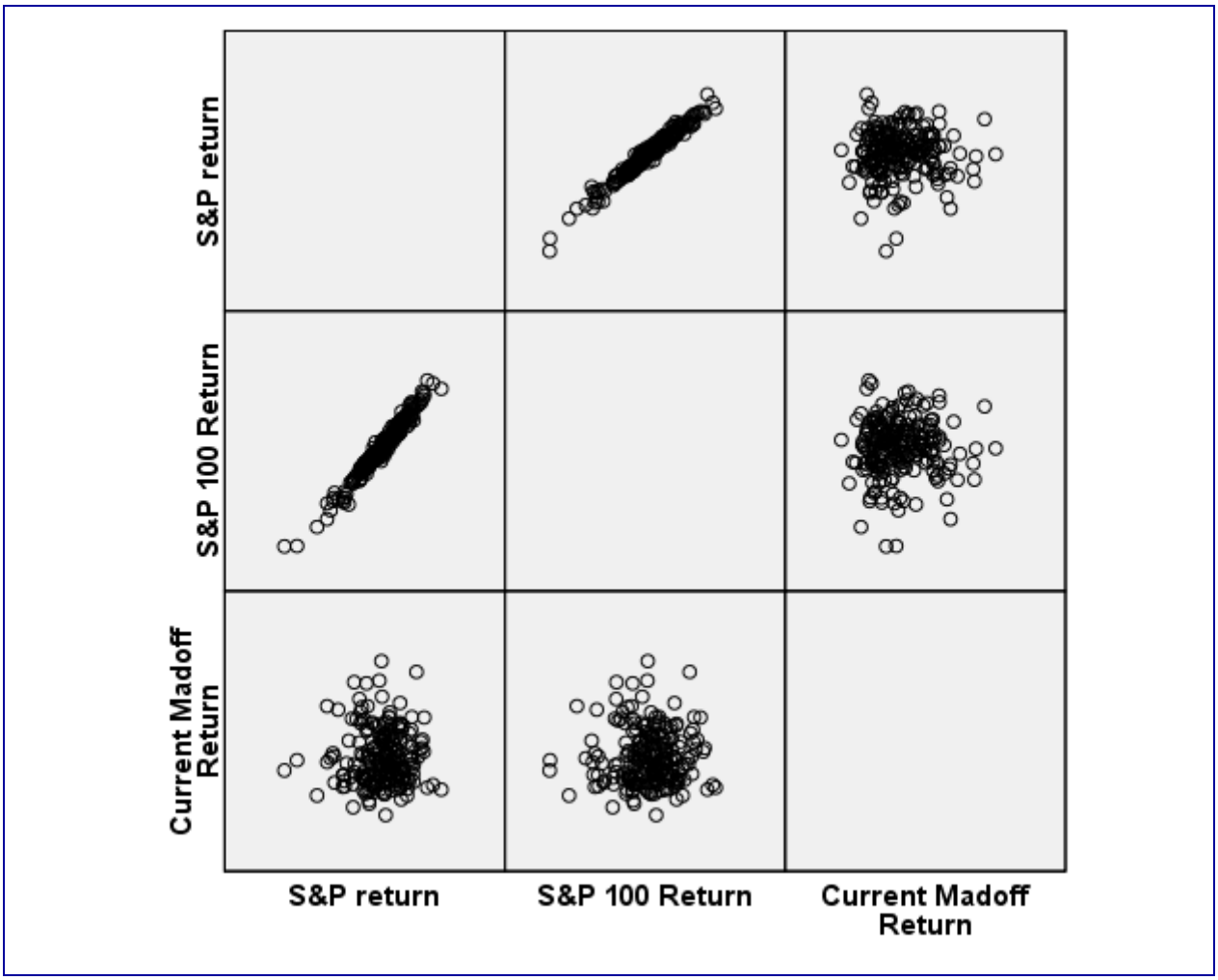
Note that the most obvious discrepancy between the Madoff data and Benford's Law is that the Madoff data displays an excess occurrence of the number 1. The well-known Chi-Square test can be used to compare the actual and expected frequencies of the digits to assess the overall significance of departures of actual from expected.[8] When this test was applied to the Madoff fund data the difference between actual and theoretical frequencies was not significant at the 5% level, although the p-value of 5.9% was close to significant. For comparison, when the Chi-squared test was applied to the S&P 100 data, its p-value was 13.5%. Other researchers have compared the Madoff data to the Benford's Law expectations and have been surprised by the findings. Kedroski opined, "It is interesting to see that any fraud here was sufficiently sophisticated such that the proffered performance numbers were credible from a distributional point of view." Thus, a test that is frequently used to detect fraud, in the case of the Madoff scheme, does not appear to provide compelling evidence.

Test 4: Scatter Plot of Returns vs. a Related Return Series

Figure II.5 displays a matrix scatter plot of the S&P 500 return versus that of the S&P 100 along with scatter plots of the Madoff returns versus both of those indices.

[8] Use the well-known formula $X^2 = \sum_k \frac{(Observed - Expected)^2}{Expected}$ with the Chi-Squared distribution and degrees of freedom of k-1

Figure II.5 – Scatter Plot Matrix of S&P 500, S&P 100, and Madoff Returns

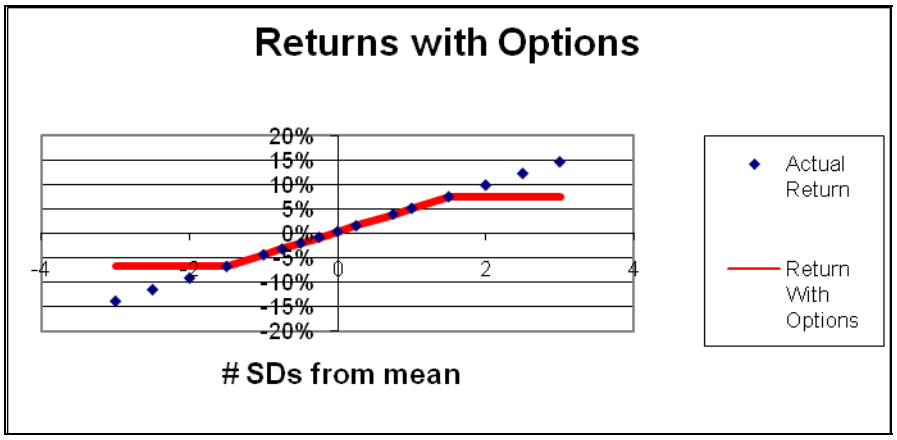


As one would expect, the scatter plot indicates a very high correlation between the S&P 500 and its relative, the S&P 100. However there is no apparent correlation between the S&P 100 and the Madoff fund, even though the Madoff Fund purportedly consists of S&P 100 stocks.

Test 5: The Split-Strike Strategy

In interpreting the graphs and tables, it is necessary to consider the "split-strike" options strategy that Madoff claimed to use. Forray described the strategy and evaluated their likely impact on the descriptive statistics and graphs such as those in this section. As described by Forray (2009), the split-strike strategy involved buying put options to limit downside volatility, while at the same time selling out-of-the-money call options to fund the purchase of the put options. **Figure II.6** is provided to illustrate the split strike options strategy.

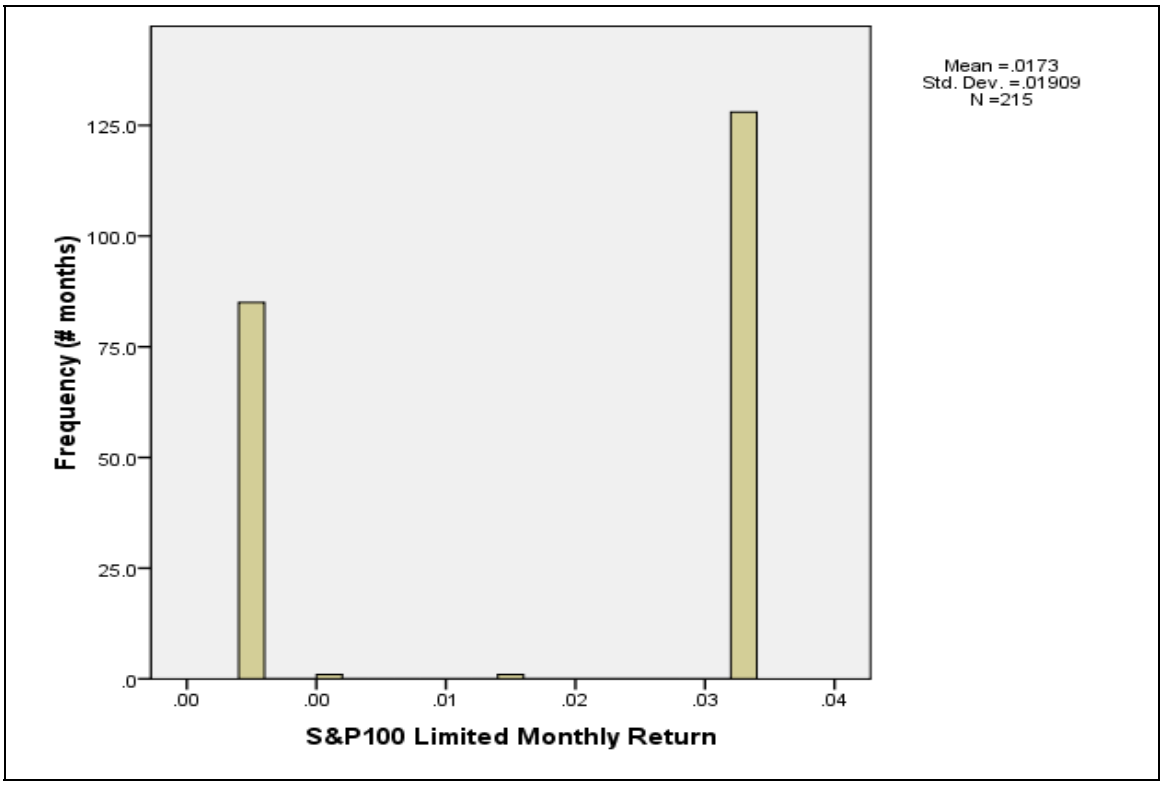
Figure II.6 – Effect of Options on Return



The use of options censors the return of stock funds. The sale of the call options limits the upside potential for the fund. On the down side the purchase of put options limits the downside negative return. For example, if a fund manager buys put options with a strike price of 1.5 standard deviations below the mean (for the S&P 100, approximately -7%), the fund's monthly return will never drop below the strike price (i.e., -7% is the minimum possible return). Alternatively, if returns exceed the strike price of the call options sold by the fund, the funds return is limited by the strike price (say to approximately +7.5%, if the strike price is at +1.5 standard deviations). We could expect the use of the options to reduce the correlations (especially in the tails) between the Madoff Fund and the S&P 100, but that it should still be high. Moreover, transaction costs are incurred to buy and sell options, and these costs depress the returns of the fund. Forray (2009) estimated the net cost of the option transactions to be 0.5% per month, suggesting that the excess return of the Madoff fund compared to the S&P 100 is much more suspect.

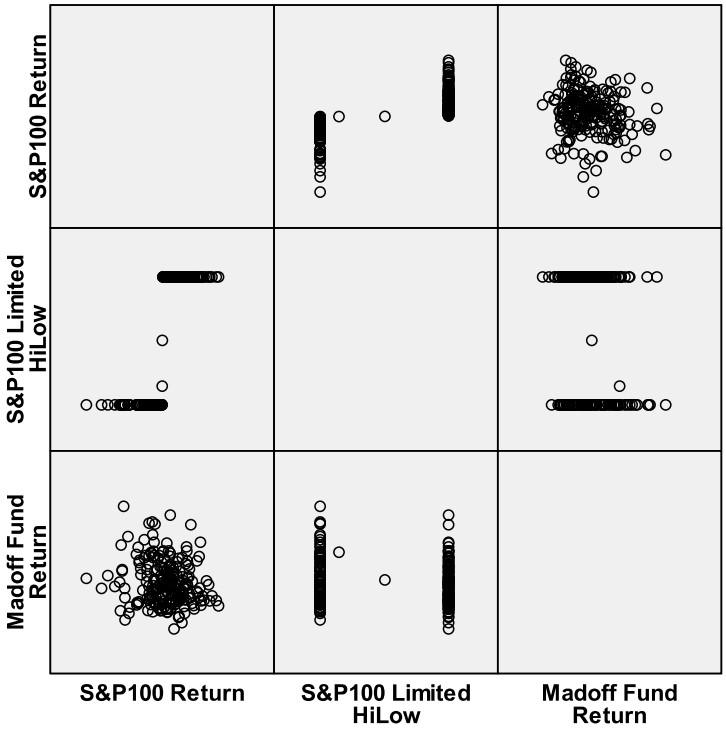
To provide a rough assessment of the impact of limiting returns with options, we capped our S&P 100 returns below at -0.6% and above at +3.3%, approximating the 18-year minimum and maximum in the Madoff fund data. Below we display the histogram of the limited data and note that it is quite different from the histogram in Figure II.3. In particular, the limiting of the returns causes two spikes in the histogram at the lower and upper limit values.

Figure II.7 – Histogram of S&P 100 Returns Limited Above and Below



We also provide a scatter plot matrix displaying the S&P 100, the limited S&P 100 and the Madoff returns. Note that there is still virtually no correlation between the Madoff data and the limited S&P 100 series, while there is still a meaningful correlation (the measured correlation was 0.775) between the S&P 100 and the limited S&P 100.

Figure II.8 – Scatter Plot Matrix, S&P 100, S&P 100 Limited and Madoff Fund



All Tests Recap

We evaluated the Madoff fund data using simple graphical and statistical procedures. We compared the histogram of the Madoff returns to that of a purportedly related return series (if the claim to the split-strike, S&P 100 investment is believed) and found stark differences. We produced a scatter plot of the Madoff returns and found no meaningful correlation, even though there should have been one. We also computed simple descriptive statistics for the Madoff returns and four other comparative return series. The Madoff returns (1) had a much higher mean and (2) a much lower standard deviation than all comparison categories, including a balanced and long term bond fund whose return volatility should be tempered compare to an equity fund. We also found that the Madoff data was positively skewed compared to the negative skewness of the other series.

A more sophisticated test, Benford's Law, was applied to he Madoff returns and failed to provide evidence of fraud. While Benford's Law is a handy tool used by fraud experts to uncover "fake" accounting transactions, it would not likely be known to the typical fund manager screening potential investments for clients. It is possible that Madoff was familiar with the rule and prepared reports to investors that would not trigger an investigation by those applying the rule. Nonetheless,

the dramatic departure of the Madoff returns from reasonably expected statistical patterns should have triggered a high level of concern on the part of fund managers and the SEC.

II. CASE STUDY #2: MORTGAGE DATA

The underwriters of mortgage loans had available to them an abundance of loan level data that could have been used to monitor the overall quality of their book of loans. In addition to their internal data, a number of external data sources were available. One of the popular external databases was LoanPerformance.[9] This database is cited frequently in the financial crisis literature (for instance, Demyanyk and Hemert 2008). As it is a commercial database we do not use it in our analysis, but will cite the results of others who had access to it.

Another database that has been used to monitor the performance of the mortgage industry is the Home Mortgage Disclosure Act (HMDA) database. Enacted by Congress in 1975 and implemented by the Federal Reserve Board's Regulation C, the HMDA requires lending institutions to report public loan data. This data is publically available from a U.S. government Web site.[10] The authors analyzed publicly available HMDA data for loans originating in 2007.[11] As we were limited in the number of years that were available, we chose a cross-sectional comparison, though, as we will show in the next section, time series statistics from another source portrays the unfolding of the crisis over time.

Our cross-sectional comparison uses data for six states – Alabama, Arizona, Florida, Michigan, Nebraska, and Nevada. According to Realty Trac® in its May 2009 U.S. Foreclosure Market Report™, released June 11, 2009,[12] the relative rank of foreclosures (shown in parentheses) for these six states are shown in **Table III.1**.

Table III.1

High Rates of Foreclosure	Low Rates of Foreclosure
Arizona (4)	Alabama (30)
Florida (3)	Nebraska (45)
Michigan (6)	
Nevada (1)	

[9] See www.loanperformance.com for information about their data.
[10] Currently it can be obtained from http://www.ffiec.gov/hmda/.
[11] Data for other years was not available in data set format from the HMDA site.
[12] http://www.realtytrac.com/gateway_co.asp?accnt=137300

Variables in the Data

There are 36 variables in the HMDA data. Most of these are related to the nature of the loan, but geographic and demographic information is also available. **Table III.2** is a list of some variables that could be useful in assessing the riskiness of loans. Some of the variables, such as census tract number, could be useful in incorporating external data not contained in the HMDA database.

Table III.2

Variables in HMDA Data
Agency Code
Loan Purpose
Loan Type
Property Type
Occupancy
Loan Amount(000s)
Preapproval
Action Type
County
Census Tract Number
Applicant Income (000s)
Purchaser_Type
Denial Reason
Rate Spread
Lien Status
Rate Spread
Lien Status

Table III.3 provides descriptive statistics for three of the variables that should be useful in assessing the riskiness of loans – loan amount, applicant's income, and rate spread (spread of loan interest rate to index interest rate, often LIBOR[13] or other internationally recognized benchmark). Note the heavy tails and skewness of the distributions, reflecting the presence of extreme values for the variables. Note also that some extreme values (i.e., loan amounts of $99.999 million) were assumed to be suspect and were not included in the analysis.

[13] LIBOR is the London Interbank Offered Rate. LIBOR is an interest rate at which banks can borrow funds, in marketable size, from other banks in the London interbank market. LIBOR is fixed on a daily basis by the British Bankers' Association.

Table III.3 – Descriptive Statistics for HMDA Florida Data[14]

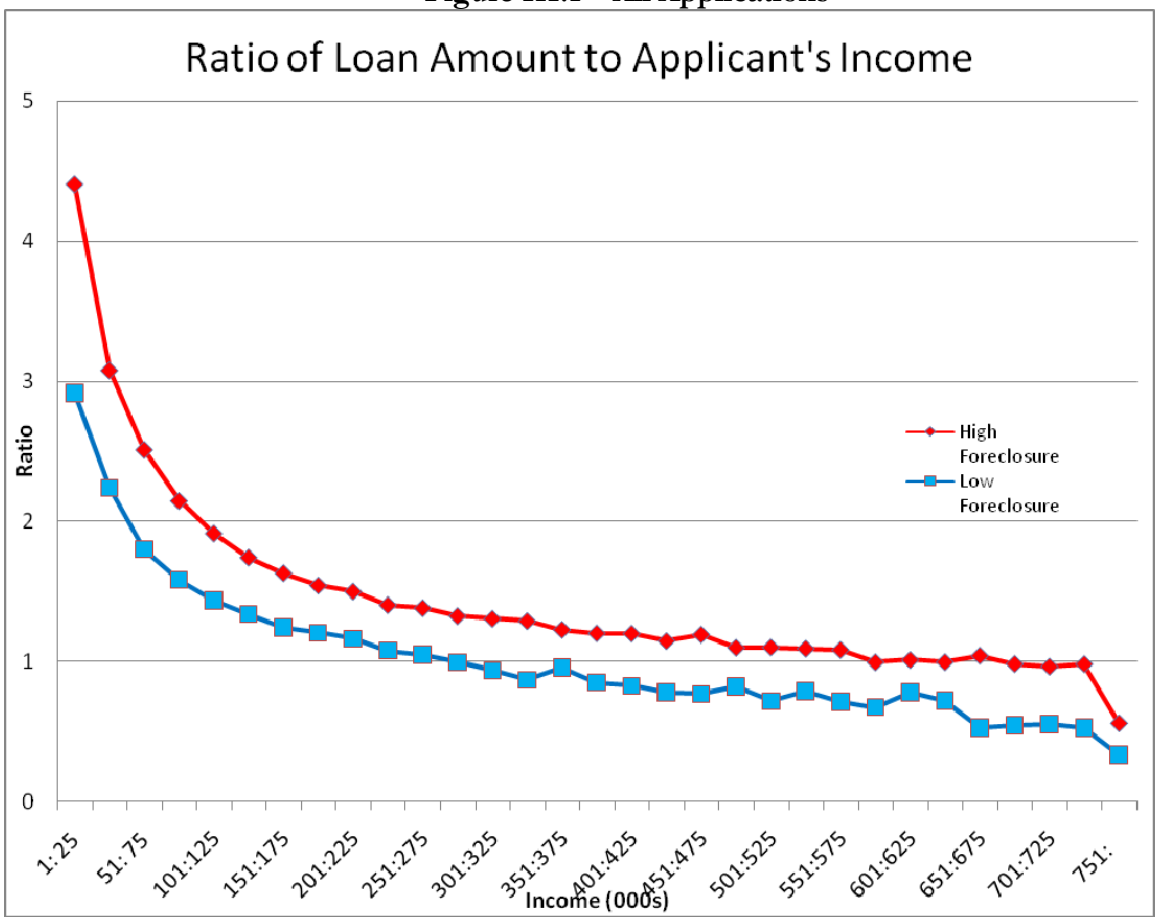
		Loan_Amount_000s	Applicant_Income_000s	Ratespread
N	Valid	1773450	1773450	159203
	Missing	0	0	1614247
Mean		206.52	114.20	5.0495
Median		171.00	75.00	4.7400
Skewness		18.549	16.011	.827
Std. Error of Skewness		.002	.002	.006
Kurtosis		1817.752	473.308	.775
Std. Error of Kurtosis		.004	.004	.012
Minimum		2	2	3.00
Maximum		45500	9981	30.36
Percentiles	5	31.00	28.00	3.0800
	10	50.00	35.00	3.1700
	20	90.00	45.00	3.3800
	30	120.00	54.00	3.6800
	40	147.00	64.00	4.0900
	50	171.00	75.00	4.7400
	60	198.00	88.00	5.4100
	70	229.00	105.00	5.9800
	80	275.00	136.00	6.5600
	90	364.00	204.00	7.3600
	95	468.00	300.00	8.0500

The loan and income information were used to compute a loan-to-value ratio, as loan amounts that were excessive relative to the income of the applicants were considered one of the key risk factors in the mortgage crisis.

Figure III.1 shows the ratio of Loan Amount to Applicant's Income for the six states with the data aggregated into two groups – high-foreclosure states versus low-foreclosure states. For this analysis, we have removed records from the database that have no income reported. The states with a higher incidence of foreclosures show a consistently higher ratio of Loan Amount to Applicant's Income across all income buckets.

[14] For our analysis for rate spread in this section, the percentiles and other descriptive statistics ignored records with missing values, i.e., coded as "NA."

Figure III.1 – All Applications



The logical question is: Was the higher riskiness of the loans based on the loan-to-income ratio reflected in the rate spread[15] for the loans? While **Table III.4** does not show a dramatic difference in the distribution of loan applications by rate spread, the states with a higher foreclosure rate have a slightly higher percentage of loan applications with a zero basis point difference. One would expect the opposite to be the case. Additional examination of the data showed that non-owner occupied houses and refinanced mortgages, as opposed to original mortgages, had lower rate spreads on average.

A question that arises is: does this reflect a major problem with the data reported into the HMDA database? That is, we observed that 90% of the records may have been incomplete with respect to this field, as the rate spread value was reported as "NA." In the computations underlying Table III.4, the NAs were treated as if the rate spread was zero.

[15] Spread of loan interest rate to index interest rate, often LIBOR or other internationally recognized benchmark.

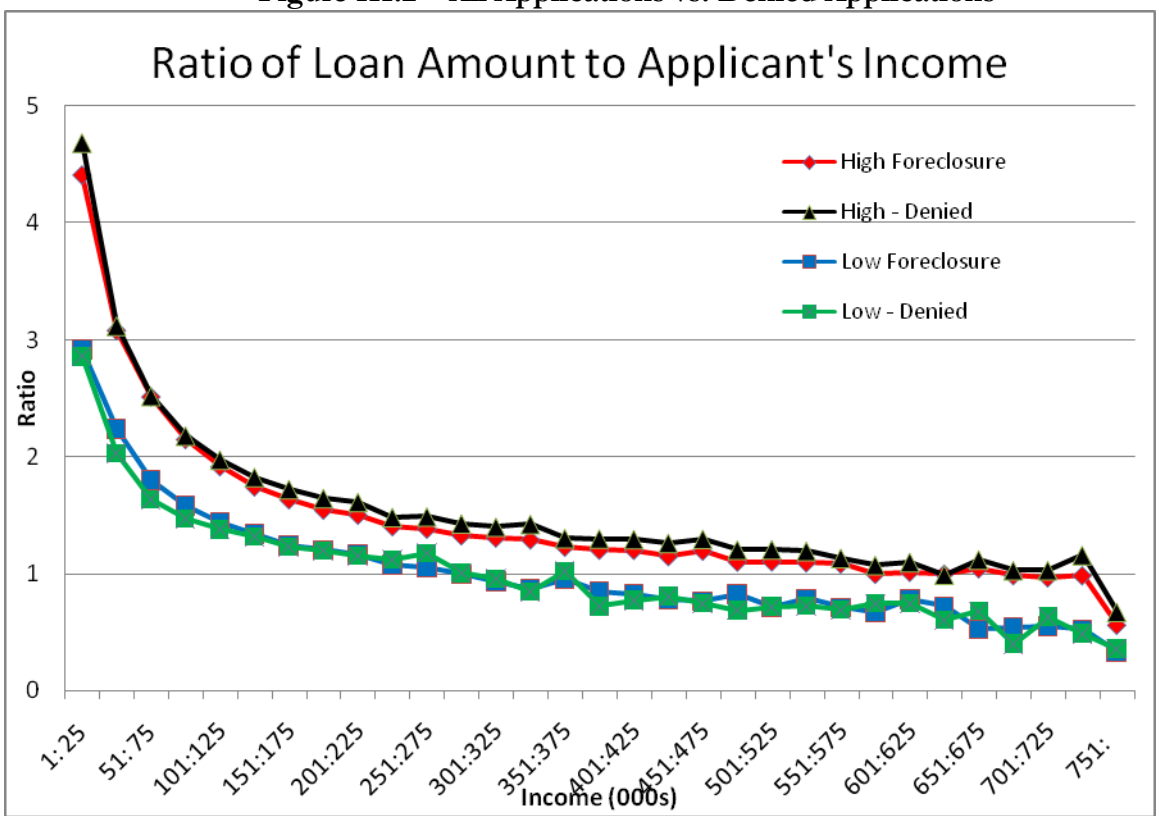
Table III.4

	Loan-to-Income Ratio		Distr of Rate Spread	
Rate Spread	Low Foreclosure States	High Foreclosure States	Low Foreclosure States	High Foreclosure States
None[16]	1.56	1.82	90.0%	91.7%
3.00: 3.99	1.51	1.92	3.2%	3.1%
4.00: 4.99	1.47	1.95	1.9%	1.2%
5.00: 5.99	1.22	1.56	1.9%	1.5%
6.00: 6.99	1.36	1.73	1.3%	1.3%
7.00: 7.99	1.26	1.33	0.9%	0.7%
8.00: 8.99	0.71	0.65	0.3%	0.3%
9.00: 9.99	0.50	0.46	0.2%	0.1%
10.00+	0.41	0.39	0.2%	0.1%
Total	1.54	1.81	100.0%	100.0%

Figure III.2 shows the ratio of loan amount to applicant's income for the six states again but showing all loan applications versus those applications that were denied for the two groups – high-foreclosure states versus low-foreclosure states. In this case, we see no discernable difference in this ratio between all applications and those that were denied.

[16] In this Table, records with a rate spread of NA were assumed to be zero.

Figure III.2 – All Applications vs. Denied Applications



Time Series Patterns: Demyanyk and Helmert (2008) Analysis Using Loan Level Data

Evolution of some of the key risk variables, such as loan-to-value ratios, can provide valuable insights into the riskiness of the loan portfolio and how it changed over time. Using data unavailable to us, Demyanyk and Helmert performed such an analysis, based on a history of loan level data from LoanPerformance.com. **Figure III.3** displays a time series of the loan-to-value ratio and shows a decline in loan-to-value ratio in 2002, followed by a steady increase through 2006.

Figure III.3 – Time Series of Loan-to-Value

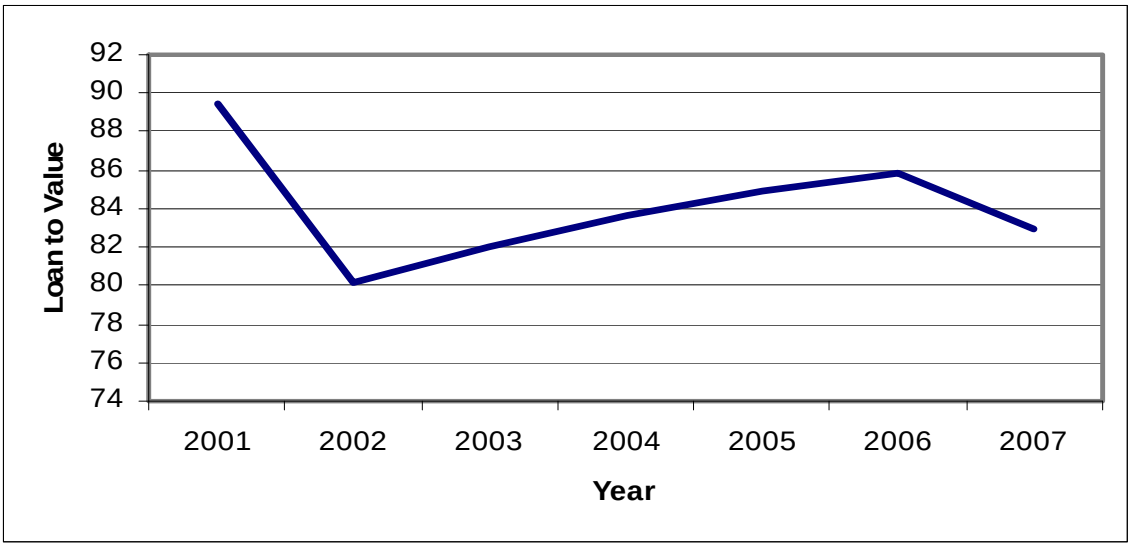


Figure III.4 displays the number and size of subprime loans from 2001 to 2007. It is clear that both the volume and size of such of these loans increased at extraordinary rates. Through 2005 the number of subprime loans increased more than five times and the dollar-value of subprime loans increased by almost a factor of eight. It should be noted overly aggressive growth strategies are a key factor in the development financial bubbles (Ferris, 2009). In addition, an analysis by Carson and Dastrup (2009) indicated that the large percentage of subprime loans (compared to total loans) was a key factor in the mortgage crisis.

Figure III.4 – Subprime Loans

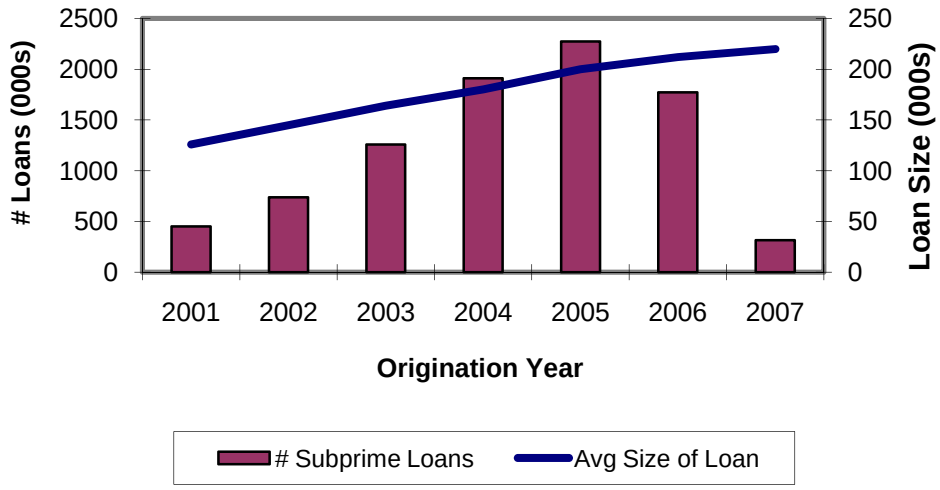


Figure III.5 – Balloon Payments and Documentation Percent

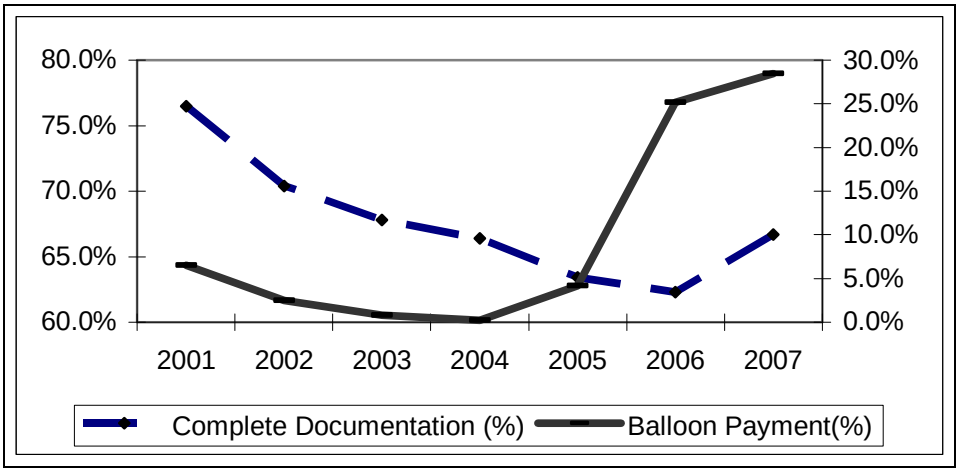


Figure III.5 presents the percentage of files with complete documentation and the percent with balloon payments. The graph indicates that over time documentation standards deteriorated. This is consistent with the appearance of categories of loans referred to as "liar loans" or NINJA (no income, no job, no assets) loans. In addition, Figure III.5 indicates that the percentage of loans with balloon payments ballooned as the mortgage crisis reached its peak. Such loans require the borrower to make a balloon payment upon maturity of the loan, or in the case of some subprime loans, several years into the loan. Without an infusion of income such as from an inheritance or from a litigation settlement, such funding could be hard to produce. In the case of subprime loans, this might force the borrower into another round of refinancing.[17]

Taken together, these descriptive statistics indicate that simple descriptive statistics derived from loan level data indicated the development of significant risks in loan portfolios. When reviewed over time, the statistics indicated deterioration in the quality of loans. When the data was viewed from a cross-sectional perspective, the statistics differentiated problematic from non-problematic states.

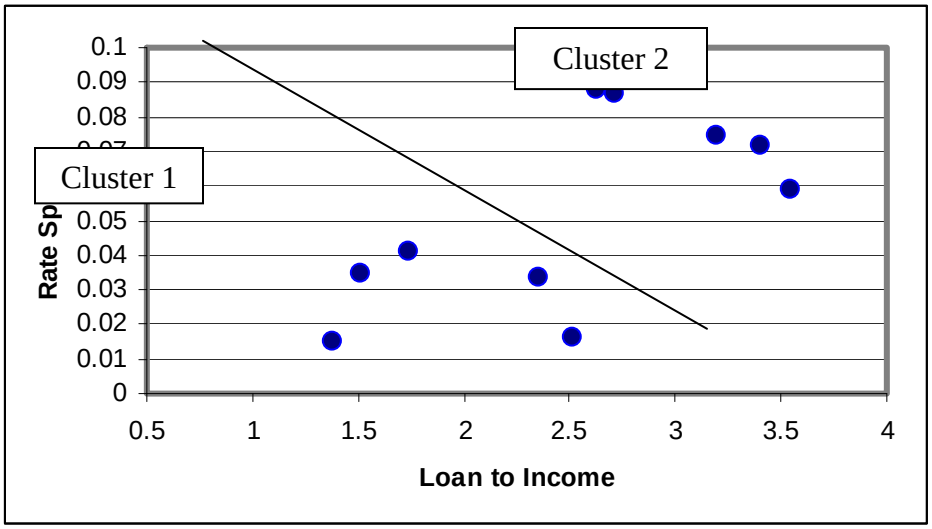
Can Loan Level Data be used to Model Foreclosures?

Can the loan level data be used to model foreclosures as an aid to underwriting loans? Though limited in the number of potential predictive variables compared to a commercial loan database, the HMDA data can be used to illustrate modeling techniques that can be applied to model foreclosures.

[17] The literature suggests that was the intent but foreclosure was also a possible outcome.

Since that data has no dependent variable, that is, we do not know which loans have already nor will default subsequent to being underwritten, we illustrate the unsupervised learning technique of clustering. Clustering can be used to group records, in this case, loan applicants with similar characteristics together. A simple illustration using hypothetical simulated data is supplied in **Figure III.6**. Suppose that data consisted of two loan characteristics. When we graph the values for the two variables the records seem to fall into two groups (separated on the graph by a line), one with low values on both variables and another with high values on both variables. One of these groups, also referred to as clusters, may be correlated to a variable of interest to us such as propensity to default. The clustering methodology applies a statistical procedure that maximizes the differences between clusters on values of the clustering variables. Some variables will inevitably be more significant in distinguishing groups from each other. The methodology used to cluster data is introduced by Francis (Francis 2006) and Sanche (Sanche 2006) and is widely described in statistics text books and is outside the scope of this paper.

Figure III.6 – Loan-to-Income Clusters



In order to perform clustering, data from the HMDA database for Florida were aggregated to the ZIP code level. The loan's census tract number was used to associate a ZIP code with each loan application record in the HMDA database. Then ZIP code level statistics were computed from predictive variables such as income, loan amount, and loan-to-value ratio.[18] These statistics included means, medians, and the percent of loans exceeding a high threshold (such as the 90th percentile value for the variable). For assessment of the effectiveness of clusters in grouping high-risk versus

[18] Data used in this analysis were from the state of Florida.

low-risk loans, data from an external source was incorporated into the data. The data is from LISC Research.[19] A model was used to score each ZIP code on the extent of its foreclosure, delinquency, and subprime problem to assist in identifying areas of greatest need for stabilization. In developing the score, information from the Mortgage Bankers Association Delinquency Survey and reports from McDash Analytics, a vendor of loan information, as well as other sources, was used. Note that a limitation of the data is that the foreclosure scores are a calendar period statistic while the HMDA data is organized by origination year. We believe that this mismatch will tend to temper the impact of relationships between HMDA loan characteristics and the LISC scores.

[19] Obtained from www.housingpolicy.org.

Table III.5 displays the mean of the ZIP code level characteristics for each of the three clusters assigned by the clustering procedure.

Table III.5 – Means On Variables[20]

	Cluster		
	1	**2**	**3**
Avg Loan Amount	297.23	566.96	163.80
Average Income	165.71	356.66	87.26
Mean LTV[21] Ratio	2.53	2.38	2.48
Rate Spread - mean	4.84	4.54	5.05
Median LTV Ratio	2.29	2.09	2.31
Median Rate Spread	4.40	3.95	4.67
Percent Applicants High LTV	4.4	3.8	4.5
Pct Applicants High Rate Spread	4.7	4.5	5.6
Percent Manufactured, Multi Family Houses	1.9	.4	6.1
Pct Home Improvement	57.8	56.5	65.6
Percent Refinance	52.4	52.5	57.3
Pct Owner Occupied	18.1	28.4	13.5

It can be seen that the clusters differ significantly on income level, loan amount, and the percentage of homes that are owner-occupied, but less so on loan-to-value and rate spread. Based on the mean statistics for the various loan risk variables, it seems that cluster 2 has the highest incomes and percent owner-occupied, while having the lowest loan-to-value ratio and rate spreads. It might be expected this cluster would have the fewest problem loans. Cluster 1 has intermediate values on income, loan amount, rate spread, median (but not mean) loan-to-value ratio, and the percent owner-occupied. This suggests these ZIP codes may have a less severe, but non-negligible problem than cluster 3.

Table III.6 displays the average score for each cluster for the foreclosure, delinquency and subprime scores. Keeping in mind that a high value indicates a severe problem, it appears that on all three problem-mortgage measures, cluster 2 has the least severe problem, followed by cluster 1, and then cluster 3 has the most severe problem.

While many limitations affect this exercise, we believe it illustrates that loan level data available to lenders during the period when the housing bubble was reaching its extreme could have been used to develop models for use in underwriting and in avoiding high-risk mortgages.

[20] Statistics for rate spread variable based only on non-zero records.
[21] LTV = Loan to value

Table III.6			
Cluster Grouping	**Intrastate Subprime Component Score**	**Intrastate Foreclosure Component Score**	**Intrastate Delinquency Component Score**
1	3.059	2.315	7.325
2	0.751	1.074	2.702
3	7.830	3.119	12.482
Total	6.774	2.920	11.288

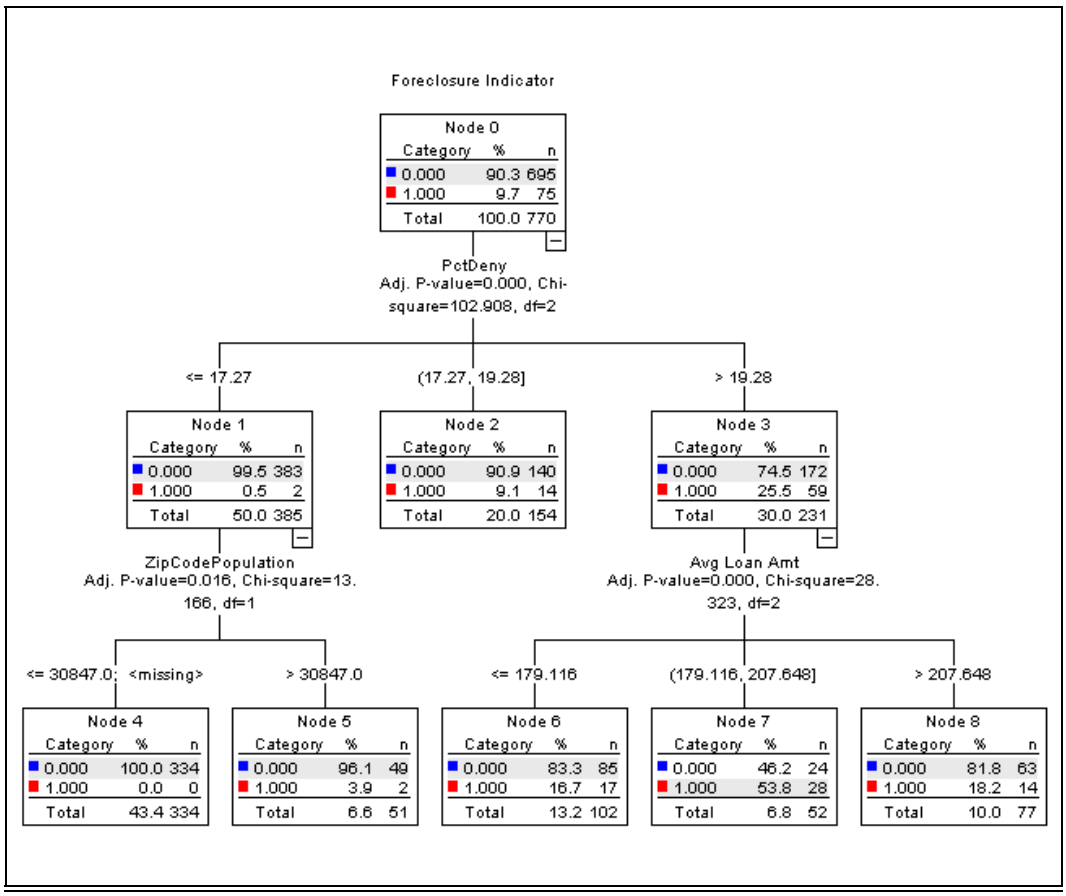
When a dependent variable is present, say loan data dating back to the late 80s and early 90s when the last housing bubble popped, a data mining method can be used to predict foreclosure problems. For illustrative purposes, one data mining method was applied to the ZIP code level data used in the previous section for clustering.[22] For this illustration, the dependent variables are the foreclosure and subprime scores from the LISC data. Of course, the values for these variables would not have been available at the time the mortgages were underwritten, however, we believe that other relevant data not available to us, such as historic default and foreclosure rates, should have been available to mortgage modelers.

The data mining method used was decision trees. This method allows the analyst to quickly search for important relationships in the data. The procedure allows the incorporation into the model of complex relationships, such as interactions and nonlinearities. In addition, many of the software tools used for tree modeling rank the predictive variables in importance. Derrig and Francis (2008) introduce a number of the tree procedures and we recommend this as well as DeVille (2006) to the interested reader seeking background information about trees.

The result of applying trees to model foreclosure problems is presented in **Figure III.7**.

[22] Only one state, Florida was used in this analysis. LISC sores are relevant only within a state, not across different states.

Figure III.7 – Tree-Based Foreclosure Model



In **Table III.7** the ranking of variables from a tree modeling procedure is presented. Note that the top two variables in predicting foreclosures are derived from the denial rates in the ZIP code. We suspect that this reflects the realization, partway through the 2007 year, of the extent of the mortgage problem, along with at least a partial return to traditional underwriting standards. The use of the house (single versus multifamily and manufactured) and the rate spread were also important predictors.

Table III.7

Independent Variable	Importance	Normalized Importance
Denial Percent	.027	100.0%
Mean Denial Score	.027	99.9%
PctApprove	.024	88.5%
ZipCodePopulation	.020	72.6%
PctManuMultiFamily	.019	69.5%
Median Rate Spread	.017	61.6%
LoanPurpose	.016	60.5%
HouseholdsPerZipcode	.015	56.1%
Mean LTV Ratio	.014	52.7%

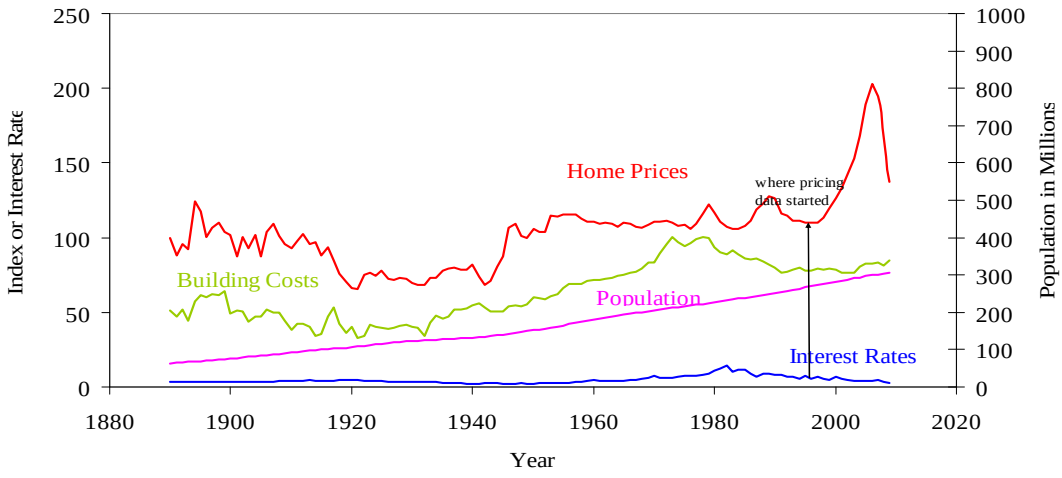
We believe this exercise with the HMDA data justifies the belief that well-known modeling techniques could have been applied to loan level data to predict the loans with a high likelihood of default.

Do Housing Prices Go Down?

A key and virtually universal belief underlying the pricing of many mortgage products and the credit analysis of the products was that housing prices never decline, especially when averaged over large geographic areas, such as the entire U.S. What is most surprising about this belief is that many of us may remember experiencing such a decline in the late 80s and early 90s (depending on where one lived).

A simple check of publically available data would have exposed this fallacy. In **Figure III.8**, we present the widely cited Case-Shiller Home Price index, along with some comparative indices.

Figure III.8 – Housing Price Time Series[23]



Mortgage Fraud: Could We Have Detected the Problem Sooner?

During the recent economic turndown, the incidence of mortgage foreclosures has risen substantially. This has caused an unprecedented drain on the United States economy as the federal government shores up the various financial institutions involved with the mortgages that are now defaulting. But could we have foreseen the escalation in foreclosures? Were there indicators to what was coming? Fraud is believed to have played a significant role in the subprime crisis. In his book *Confessions of a Subprime Lender* (2008), Richard Bittner stated that at the time he exited the business because of his perception of the irrationality of the mortgage market; approximately 70% of the loan applications sent to him contained some level of fraud/misrepresentation.

The authors reviewed Interthinx Fraud Risk Index data, which tracks the risk of mortgage fraud throughout the United States. The Fraud Risk Indices are calculated based on the frequency with which indicators of fraudulent activity are detected in mortgage applications processed by the Interthinx FraudGUARD® system.[24] The Fraud Risk Indices are based on detailed transaction data

[23] Data for Figure III.8 in Robert J. Shiller, *Irrational Exuberance*, 2nd. Edition, Princeton University Press, 2005, Broadway Books 2006, also Subprime Solution, 2008, as updated by author. Graph and data underlying it was obtained from www.iii.org. A publicly available housing price index, the Case-Shiller index, is used in the graph, and is available for download from the Standard and Poor's web site..

[24] FraudGUARD®system is a leading loan-level fraud detection tool available to lenders and investors from Interthinx, an ISO business.

from loan applications supplied by lenders. While the Interthinx FraudGUARD® system provides a check on over 300 items per application, these indices focus on a sub-set of the checks that are most indicative of the type of fraud represented by the indices—property valuation, identity, occupancy, and employment/income. The indices provide a means of comparing geographic regions or time periods—it is the relative values of the indices that are important and not the absolute value of a particular index. The definitions of the fraud risk indices are provided in Appendix A.

For this section of the analysis, we reviewed the indices for Alabama, Arizona, Florida, Michigan, Nebraska, and Nevada, which are the same states we used in the cross-sectional analysis. To develop the index for the "grouped" states (to simplify the presentation), we took the simple average of the state indexes for the states in each group.

While hindsight is always 20-20, the Overall Risk Index (**Figure III.9**) during 2004 may have been a leading indicator of the trouble that was brewing in the mortgage industry. The four states with high foreclosure rates in 2009 have consistently had an index over 100. The index is also rising again in Nevada. In contrast the index for Alabama and Nebraska were consistently lower. **Figure III.10**, the Property Value Fraud Risk Index, shows the same pattern of high risk for fraud during 2004.

Figure III.9 – Overall Fraud Risk Index

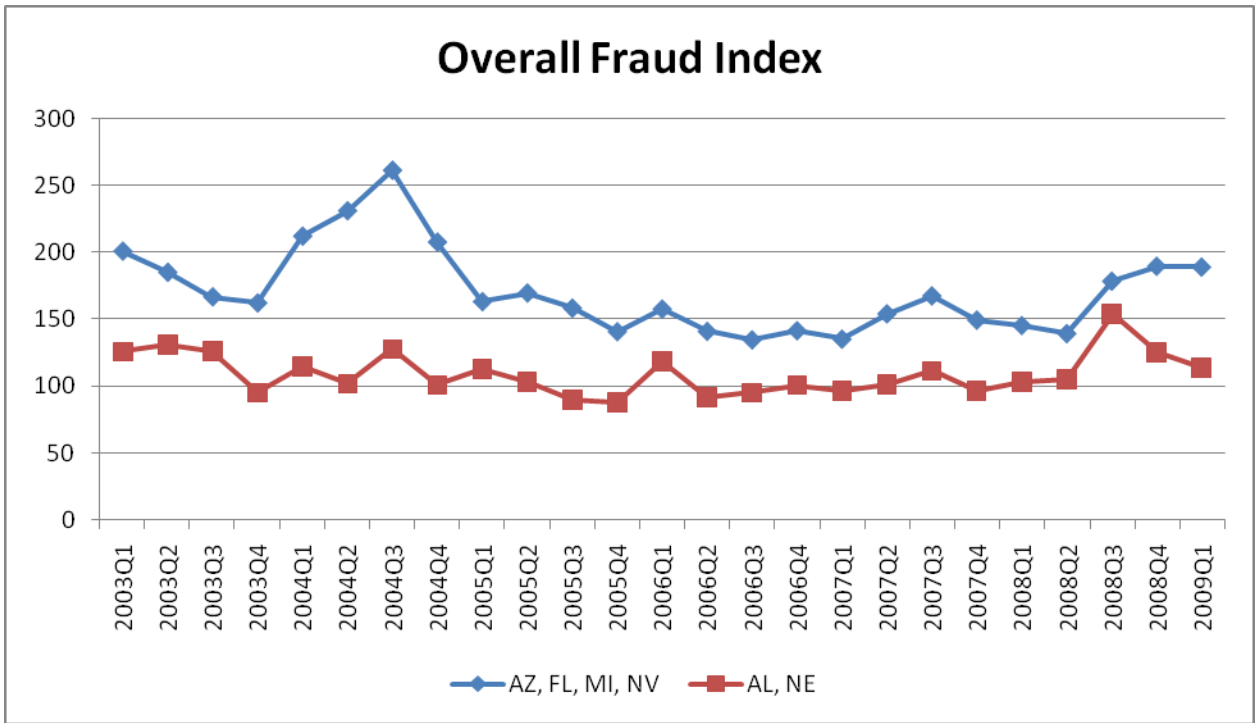
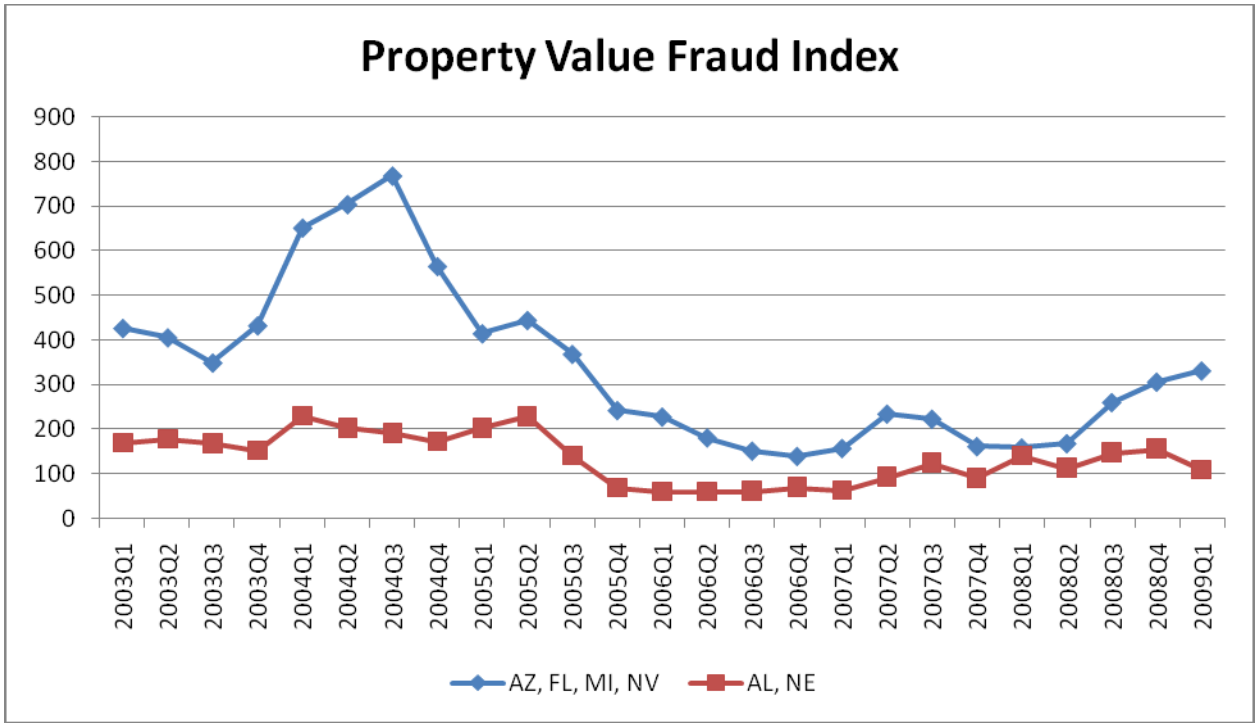


Figure III.10 – Property Value Fraud Risk Index



Mortgage Data Conclusions

ASOP No. 23 prescribes that actuaries use data that are complete and that are appropriate for the analysis. In ignoring the information prior to 1994 that housing prices declined historically on several occasions, the credit rating agencies used data that was incomplete to the point of absuridity. The analysts also failed to consider the limitations of their data (and perhaps to adjust subjectively for it) and to disclose those limitations.

The foregoing analyses also leads the authors to believe that:

- Simple descriptive statistics and predictive models may have helped avoid the crisis that developed in late 2007.

- Monitoring the make-up and quality of the loan portfolio is imperative and provides valuable insights into the riskiness of the portfolio. This is a continuous process as loans are underwritten and added to the portfolio. This is analgous to property/casualty insurance companies monitoring their probable maximum loss (PML) as they underwrite coastal risks in the United States.

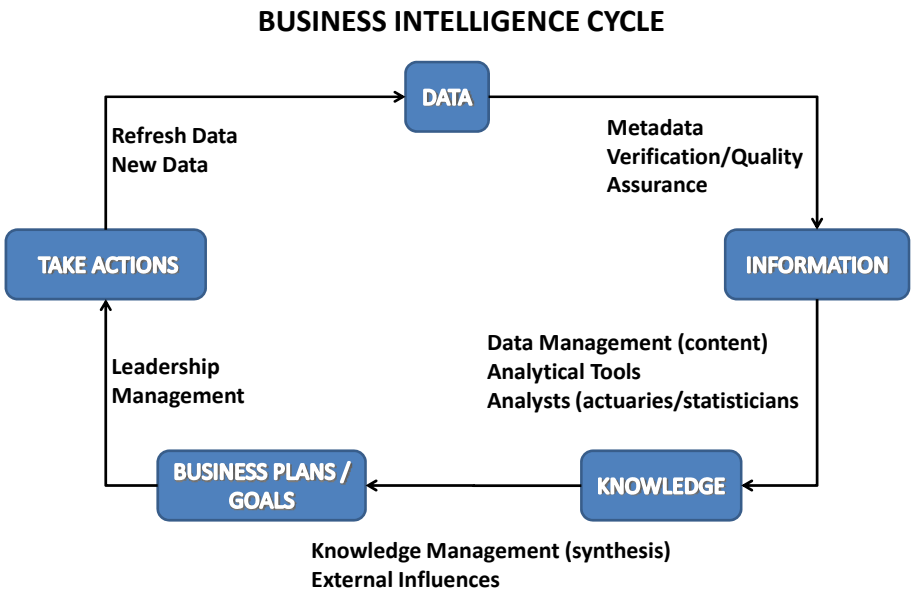
- Loan applications should be reviewed for not only quality of the risk, but also for completeness and accuracy of the application to judge its risk quality.

III. CONCLUSION: COULD BUSINESS INTELLIGENCE HAVE HELPED AVOID THE CRISIS?

Howard Dresner of The Gartner Group defined business intelligence as "a set of concepts and methodologies to improve decision making in business through the use of facts and fact-based systems." The authors believe that the signals were there in the data if the investors and those involved in the mortgage process had done the following:

- Developed systems to collect quality data
- Routinely reviewed simple descriptive statistics from such loan data as were available
- Used publicly available aggregate statistics to validate crucial model assumptions
- Used various analytical tools to convert the data into information to be used in the business process
- Applied leadership and businss aucumen to act on the analytics

Business intelligence coupled with the data management concepts espoused in the recent paper *Actuarial I.Q. (Information Quality)* may help us avert these types of problems in the future. We believe that the financial crisis is the poster child for the failure to apply these business intelligence concepts as visualized in the following figure.



In the case of the Bernard Madoff fraud, a simple review of published return data would have uncovered the unreasonableness of the data. In the case of the mortgage crisis, a review of loan level statistics should have provided an early warning that the quality of mortgages was deteriorating. Moreover, publicly available data would have indicated that housing prices have declined a number of times in the past and that it was extremely dangerous to use models that assume housing prices, in the aggregate, never decline.

Would more training of the principals involved in the financial crisis on data quality and business intelligence have prevented some of the damage? Was the crisis a result of ignorance, or were the financial incentive so overwhelming that business fundamentals were ignored? News sources indicated that inexperienced people were given significant responsibilities, both in the Madoff case (Scheer 2009) and and in pricing mortgage products (Bestiani 2009). Nevertheless, according to Bloomberg Markets (Helyar et al. 2009), some fund managers provided client funds to Madoff, even though they suspected that Madoff was involved in a kind of fraud referred to as "front-running."[25] The implication is that they were willing to go along with an illegal scheme if it brought extra returns to their funds and if they thought the risk to themselves and their invesors, should the fraud be detected, was low.

[25] Front-running is a type of insider trading where brokers trade ahead of the markets.

There are also indications that some data used in pricing and underwriting mortgages was intentionally ignored. Muolo et al. (2008) cite an executive from the compliance firm Clayton Holdings, who informed a rating agency about a number of "exceptions" (i.e., from underwriting standards) in a loan protfolio and was told, "We do not want to see it. It does not fit our model."[26] Bestiani (2009) notes that the rating firms with the most optimistic evaluations were given the rating business by lenders and investment firms. In a November 2, 2008 article titled "Was There a Loan it Did Not Like?" *New York Times* reporter Morgenson describes the travails of a senior underwriter at Washington Mutual who at the height of the bubble was pressured to approve loans that she felt were obviously flawed, and in some cases blatantly fraudulent. At this point, we are not privy to the motivations of those who contributed to the financial crisis. Nonetheless, both inexperience and "the moral hazard problem" appear to have contributed to the poor use of data and failure to apply the concepts of business intelligence.

[26] Muolo et al., 2008, , p. 283.

APPENDIX A: INTERTHINX FRAUD INDEX DEFINITIONS[27]

The Interthinx Fraud Risk Indices consist of the Mortgage Fraud Risk Index, which measures the overall risk of mortgage fraud, and the property valuation, identity, occupancy and employment/income indices, which measure the risk of these specific types of fraudulent activity.

The Mortgage Fraud Risk Index considers 40+ indicators of fraudulent activity including property misvaluation; identity, occupancy and employment/income misrepresentation; non-arms-length transactions; property flipping; straw-buyers; "silent seconds;" and concurrent closing schemes. The four type-specific indices are based on the subset of indicators that are relevant to each type of fraudulent activity.

Each index is calibrated so that a value of 100 represents a nominal level of fraud, a value calculated from the occurrence of fraudulent indicators between 2003 and 2007 in states with low foreclosure levels. For all five indices, a high-value indicates an elevated risk of mortgage fraud and each index is linear to simplify comparison across time and location.

The Interthinx Indices are leading indicators based predominantly on the analysis of current loan originations. FBI and FinCEN reports are lagging indicators because they are derived primarily from suspicious activity reports (SARs), the majority of which are filed after the loans have closed. The time lag between origination and the SAR can be several years. For this reason, the Interthinx Fraud Risk Indices' top fraud geographies and type-specific findings may differ from FBI and FinCEN fraud reports.

[27] *Mortgage Fraud Risk Report*, Interthinx, Inc, August, 2009.

IV. REFERENCES

[1.] Arvedlund, Erin, "Innocence Lost," Portfolio.com, December 17, 2008, http://www.portfolio.com/news-markets/top-5/2008/12/17/madoff-barrons/.

[2.] Barth J., et al., "A Short History of the Subprime Mortgage Market Meltdown," *GH Bank Housing Journal,* Milken Institute, January 2008, http://www.ghb.co.th/en/Journal/Vol2/04.pdf.

[3.] Bestiani, R., "The Perfect Financial Storm," Working Paper #44, 2009.

[4.] Bitner, R., *Confessions of a Subprime Lender*, Wiley, 2008.

[5.] Carson, R. and S. Dastrup, "The Housing Price Bubble: Data Mining to Explain The Housing Price Fall Across Metropolitan Areas," presented at Salford Data Mining Conference, August, 2009.

[6.] CAS Committee on Management Data and Information, "Data Quality White Paper," CAS *Forum,* Winter 1997, pp. 145-168, http://www.casact.org/pubs/forum/97wforum/97wf145.pdf.

[7.] CAS Data Management Educational Materials Working Party, "Actuarial I.Q. (Information Quality)," CAS *Forum*, Winter 2008, pp 136-195, http://www.casact.org/pubs/forum/08wforum/actuarialIQ.pdf.

[8.] Demyanyk, Y. and O. Van Hemert, "Understanding the Subprime Mortgage Crisis," SSRN paper, (December 5, 2008). Available at SSRN: http://ssrn.com/abstract=1020396.

[9.] Derrig, R. and L. Francis, "Distinguishing the Forest from the TREES: A Comparison of Tree-Based Data Mining Methods," *Variance* 2:2, 2008, pp. 184-208.

[10.] De Ville, B., *Decision Trees for Business Intelligence and Data Mining: Using SAS Enterprise Miner*, SAS Publishing, 2006.

[11.] Ferris, Shauna, "How to Destabalize a Financial System: A Beginner's Guide," presented at the Biennial convention of the Institute of Actuaries of Australia, April 19-22, 2009.

[12.] Forray, S. "The Rat without a Tail," *Contingencies*, September/October 2009, pp. 32-38.

[13.] Gorton, G., "The Subprime Panic," National Bureau of Economic Research Working Paper, October 2008.

[14.] Helyar J., K. Burton, V. Silver, "Madoff Enabelers Winked at Suspected Front-Running," Bloomberg, Jan 27, 2009, http://www.bloomberg.com/apps/news?sid=au4Y7Cudw2Xo&pid=20601109.

[15.] Joint Risk Management Section (JRMS), *Risk Management: The Current Financial Crisis, Lessons Learned and Future Implications*, published as an e-book, December 2008, see http://www.casact.org/cms/pdf/Essays_on_Financial_Crisis.pdf.

[16.] Kedroski, Paul, "Berrnie vs. Benford's Law: Madoff Wasn't That Dumb," December 19, 2008, http://paul.kedrosky.com/archives/2008/12/19/bernie_vs_benfo.html.

[17.] LISC Research and Assessment, "Zip Codes Foreclosure Needs Methodology Appendix," October 2008, http://www.housingpolicy.org/assets/foreclosure-response/zipmethodology.pdf.

[18.] Markopolos, H., "Testimony of Harry Markopolos, CFA, CFE, Before the U.S. House of Representatives Committee on Financial Services," Wednesday, February 4, 2009, 9:30 AM, http://ncc.gmu.edu/events/09_MOT-markopols/markopolos_020409.pdf.

[19.] Morgenson, Gretchen, "Was There A Loan It Didn't Like?" *New York Times*, November 1, 2008, http://www.nytimes.com/2008/11/02/business/02gret.html.

[20.] Muolo, P. and M. Padilla, *Chain of Blame: How Wall Street Caused the Mortgage and Credit Crisis*, John Wiley and Sons, 2008.

[21.] Sanche, R., Kevin F. Lonergan, "Variable Reduction for Predictive Modeling with Clustering," CAS *Forum*, Winter 2006, pp. 89-100, http://www.casact.org/pubs/forum/06wforum/06w93.pdf.

[22.] Scheer, D., "SEC Never Did Competent Probe," Bloomberg News, September 2, 2009, http://www.bloomberg.com/apps/news?pid=20601087&sid=agBw9n2hZi5U.

[23.] Schoolman, P., "Credit Crisis Lessons for Modelers," in *Risk Management: The Current Financial Crisis, Lessons Learned and Future Implications*, 2008, http://www.soa.org/library/essays/rm-essay-2008-schoolman.pdf.

[24.] Triola, M., *Elementary Statistics*, 9th Ed., Addison-Wesley, 2003.

Biographies of the Authors

Louise Francis is a Consulting Principal at Francis Analytics and Actuarial Data Mining, Inc. She is involved in data mining projects as well as conventional actuarial analyses. She has a BA degree from William Smith College and an MS in Health Sciences from SUNY at Stony Brook. She is a Fellow of the CAS and a Member of the American Academy of Actuaries. She is the 2009-11 CAS Vice President-Research & Development. She has won the Data Quality, Management and Technology call paper prize five times. She co-authored the paper "Actuarial I.Q. (Information

Quality)." She has also co-authored other papers on the subject of actuarial data management and data quality, as well as data mining.

Virginia Prevosto is a Vice President at Insurance Services Office, Inc. Ms. Prevosto has more than 30 years of actuarial and data management experience. Ms. Prevosto is a member of the American Academy of Actuaries, a Fellow of the Casualty Actuarial Society (CAS), and a member of the Insurance Data Management Association (IDMA) and the International Association for Information & Data Quality. Currently, she is chair of the CAS Committee on Management Data and Information, vice chairperson of the Joint Accreditation Committee, and a member of the Task Force on Basic Education Internet Modules. In the past, Ms. Prevosto also served as General Officer of the CAS Examination Committee; as chair of the CAS Candidate Liaison Committee; and as a member of the GIRO Data Quality Working Party and other professional committees of the CAS and AAA. Virginia has spoken at various venues on data management and data quality issues. Ms. Prevosto authored the paper "Study Note: ISO Statistical Plans," and co-authored the prize-winning paper "Dirty Data on Both Sides of the Pond" and the paper "Actuarial I.Q. (Information Quality)." She has also co-authored other papers on the subject of actuarial data management and data quality.

Acknowledgments

We wish to acknowledge Joseph Ng and Jason R. Smith, both who work for Insurance Services Office, Inc., for their invaluable assistance in data acquisition and preparation.

Data Mining and Predictive Modeling with Excel 2007

Spyridon Ganas

Abstract

With the release of Excel 2007 and SQL Server 2008, Microsoft has provided actuaries with a powerful and easy to use predictive modeling platform. This paper provides a brief overview of the SQL Server system. It then discusses the "Data Mining Client for Excel 2007" and explains how actuaries can use Excel to build predictive models, with little or no knowledge of the underlying SQL Server system.

Keywords: predictive modeling, data mining, exploratory data analysis, neural networks, regression modeling

1. INTRODUCTION

Microsoft Excel is the data analysis tool most frequently used by members of the actuarial community. Spreadsheets are ideal for basic actuarial tasks, such as analyzing loss triangles or building actuarial indications. More advanced actuarial tasks, such as building predictive models and conducting advanced statistical analysis, have historically required specialized software, such as SAS, S-Plus or SPSS.

With the release of Excel 2007 and SQL Server 2008, it is now possible to build complex statistical models directly in Excel. This paper begins by describing the SQL Server Business Intelligence platform. It then discusses the "Data Mining Client for Excel 2007" and shows how actuaries can take advantage of SQL Server's advanced capabilities without leaving the familiar Excel interface.

2. SQL SERVER

SQL Server is Microsoft's enterprise-class database solution. It consists of four components: the Database Engine, Integration Services, Analysis Services, and Reporting Services. These four components work together to create business intelligence. Creating business intelligence is the goal of all actuarial work; business intelligence is knowledge that is extracted from data and delivered to the individuals who can use it to improve the company's bottom-line.

The Database Engine is essentially a grown-up version of Microsoft Access. Like Access, it stores data in tables and allows users to analyze the data using queries written in the SQL language. Unlike Access, it is a true enterprise-class database system; its capabilities are limited only by the available hardware. From a business intelligence perspective, the primary function of the Database

Engine is to store the raw data. Terms such as "data mart" and "data warehouse" refer to large collections of raw data that are stored and managed by the Database Engine. It is not uncommon for SQL Server databases to store billions of rows or hundreds of gigabytes of data.

Although data warehouses and data marts have become increasingly common, most companies continue to store data in a variety of formats. Integration Services is an extract, transform and load (ETL) tool. Its primary purpose is to transfer data between different storage formats. For example, Integration Services can pull data out of an Excel file and load it into a SQL Server table. Integration Services is also the primary data-cleansing tool. Dirty data makes it difficult to develop valid statistical models. Integration Services includes a number of data cleaning techniques, such as fuzzy logic, that can be used to clean data by identifying or removing suspicious values.

Analysis Services allows users to analyze the raw data using Online Analytical Processing (OLAP) cubes and data mining algorithms. An OLAP cube is essentially a pre-calculated pivot table. It resides on the server and stores the raw data, along with pre-calculated summarized data, in a multi-dimensional format. The data in an OLAP cube is usually viewed using an Excel pivot table. OLAP cubes are valuable because they allow users to "slice and dice" data sets that would otherwise be too large for Excel.

Analysis Services also contains the tools used to build data mining models. SQL Server 2008 includes seven data mining algorithms, such as neural networks and time series models. These algorithms allow actuaries to extract the valuable knowledge hidden within the massive amounts of raw data.

The final component of the SQL Server business intelligence platform is Reporting Services. Reporting Services is a web-based reporting tool. Essentially, it creates a web page where users can view reports that were built using the data in the SQL Server tables. It also includes a web application, known as Report Builder, which allows users to create ad hoc reports without knowing the SQL language. Reporting Services is the easiest way to distribute business intelligence to a large number of users.

The four components of the SQL Server system are tightly integrated and designed to produce enterprise-class business intelligence. While SQL Server is an extraordinarily powerful system, it is also quite complex. In 2007, the average salary for a Microsoft-certified business intelligence developer was $132,000[1]; this is the best evidence that developing enterprise-class business intelligence requires a great deal of education and experience.

3. THE DATA MINING CLIENT FOR EXCEL 2007

The complexity of the SQL Server system has discouraged actuaries from using it as a business intelligence platform. Most actuarial business intelligence is built and distributed using Microsoft Excel. Excel can store moderately large amounts of data (over one million rows per worksheet in Excel 2007). Tools such as VBA macros and the "Remove Duplicates" button provide a means of cleaning the data, while pivot tables and a respectable set of built-in functions allow actuaries to perform a wide range of analysis. Perhaps the most common method of distributing actuarial business intelligence is the simple act of e-mailing an Excel spreadsheet to the end-users.

Although Excel is the business intelligence platform of choice for most actuaries, its statistical modeling capabilities are limited. Neural networks, classification and regression trees, and other data mining algorithms simply are not available in standard Excel installations. There is a good reason for this; most data mining algorithms require fast processors and large amounts of memory, which are typically available only on servers.

The "Data Mining Client for Excel 2007" allows users to build complex statistical models in Excel, while processing those models on a high-performance server running Analysis Services. This greatly reduces the time and effort required to extract information from the raw data. It also allows actuaries with limited statistical programming experience to analyze their data using the most powerful algorithms currently available.

3.1 Installing the Data Mining Client for Excel 2007

The Data Mining Client acts as a link between Excel (which is typically installed on a laptop or desktop computer) and a server running Analysis Services. This client-server architecture allows multiple users, each working on their own desktop or laptop computer, to share the computational power of a single Analysis Services Server.

As a result, both the server and the client computers must be set up. Installing the Data Mining Client on each of the client machines and telling the client the name of the server running Analysis Services is relatively straightforward; a wizard guides users through the installation process. It is important to note that the Data Mining Client requires Excel 2007; a trial version of Excel 2007 is currently available [2]. Also noteworthy is the fact that the data mining feature is not installed by default; it must be selected from the list of installable features. The Data Mining Client is available as a free download on the Microsoft website [3].

Setting up the server can be more difficult. Analysis Services must be installed and running on

the server [4], and a user with administrator privileges must set up an Analysis Services database. When the Data Mining Client is installed, a tool called the "Server Configuration Utility" is also installed [5]. This is a wizard that allows a user with administrator privileges to set up an Analysis Services database for use with the Data Mining Client. This wizard is also used to grant users of the Data Mining Client access to the Analysis Services Database. Since many users can use a single Analysis Services database at the same time, the server only needs to be set up once.

When the server has been set up and the Data Mining Client has been installed, users can begin building data mining models. Models can be selected from either the "Data Mining" or "Table Tools" menus. These menus allow users to access the advanced data mining capability of the Analysis Services server.

3.2 Analyzing Tables using the Table Tools

Excel 2007 introduces "Tables", a new feature than allows users to quickly sort, filter and format data. Clicking the "Table" button on the insert menu creates a table. The process of creating, formatting and using tables is described in the article "Working with Tables in Excel 2007" [6].

When a cell in a table is selected, the "Table Tools" menu appears at the top of the screen. This menu contains two submenus, "Design" and "Analyze". The design menu is primarily used to format the table. One menu item of interest to actuaries is the "Remove Duplicate Records" wizard. This tool can be used to clean data by identifying and removing identical rows.

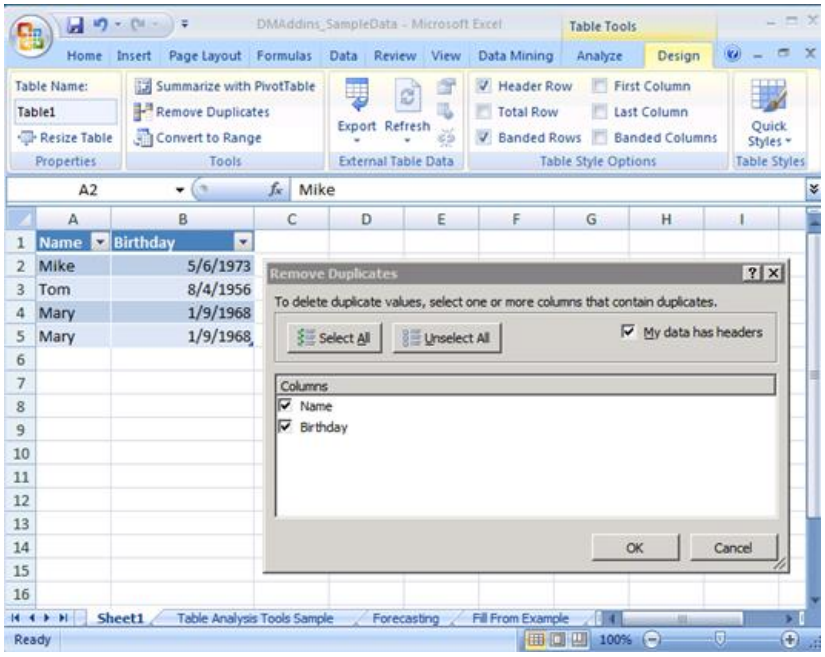


Figure 1. Removing duplicate records from a table.

The "Analyze" menu allows the user to perform eight task-oriented data mining functions. These are essentially "Black Box" functions, which allow a user to perform a task without forcing them to know anything about the underlying data mining algorithms. As seen in figure 2, these tasks are:

- Analyze Key Influences
- Detect Categories
- Fill From Example
- Forecast
- Highlight Exceptions
- Scenario Analysis
- Prediction Calculator
- Shopping Basket Analysis

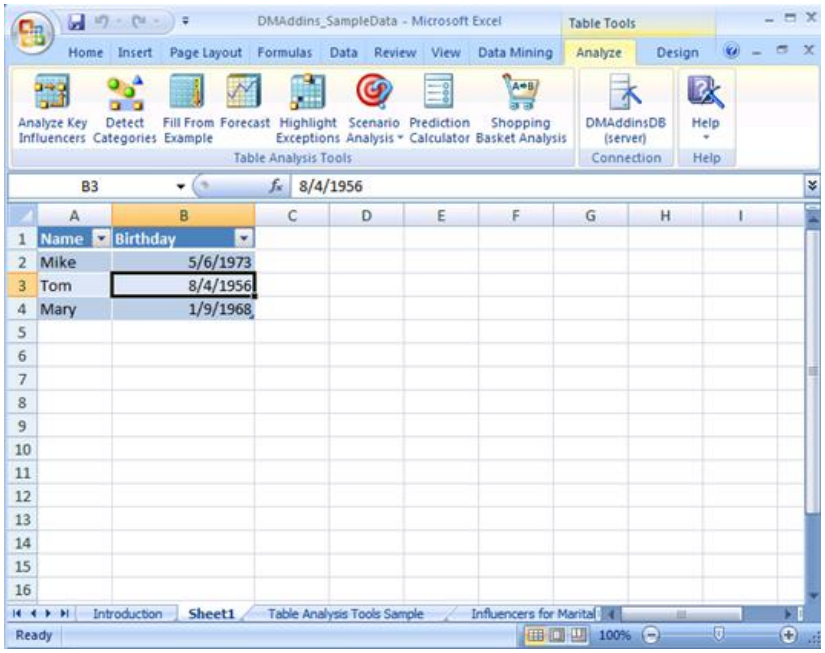


Figure 2. The Analyze menu.

3.2.1 Analyze Key Influencers

The Analyze Key Influencers button allows users to select one column in the table and then shows how the other columns are related to that column.

This can be used to predict zero-claim status for personal automobile insurance customer. For example, a table can be created that shows age, gender, marital status and if the customer had zero claims in a given time period [7]. The Analyze Key Influences tool will create a report that shows how strongly age, gender and marital status affect the likelihood of the customer having zero claims.

The report shown in Figure 3 shows the output of the Analyze Key Influences tool. In the data that was fed into this tool, married customers who are at least 51 years old are likely to have had zero claims. Single customers, especially those younger than 35, are more likely to have had a claim. This report shows that gender has no impact on zero-claim status. It also shows that age only predicts zero-claim status if the customer is younger than 36 or older than 50.

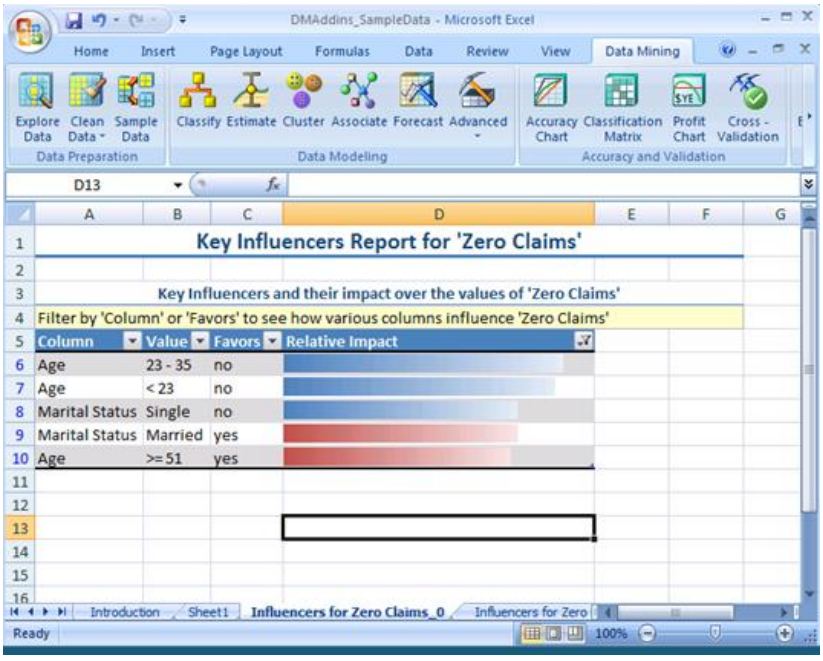


Figure 3. The output of the Analyze Key Influencers tool.

3.2.2 Detect Categories

The Detect Categories tool uses a clustering algorithm. Clustering algorithms are typically used to segment customers for marketing purposes. It identifies rows in the table that are similar and assigns the similar rows to a category. The final number of categories will depend on the number of identifiable groups of similar rows. Once the categories have been detected, a column will be appended to the table of data. The value in this column will show which category the row was assigned to.

A report containing three sections will also be created. The first section shows the number of categories and the number of rows assigned to each category. The second section of the report is similar to the Key Influencers Report. It allows a user to understand why a row was assigned to a given category. The third section of the report allows users to compare categories.

From the sample data, the Detect Categories tool found four categories in the zero-claim dataset. Looking at the key influencers and comparing the categories using the chart in the third section of the report can help explain these categories. For example, the first category contains mostly men whose age is "high" or "very high". Rows assigned to this category are clearly different from those in the second category (which primarily contains younger women).

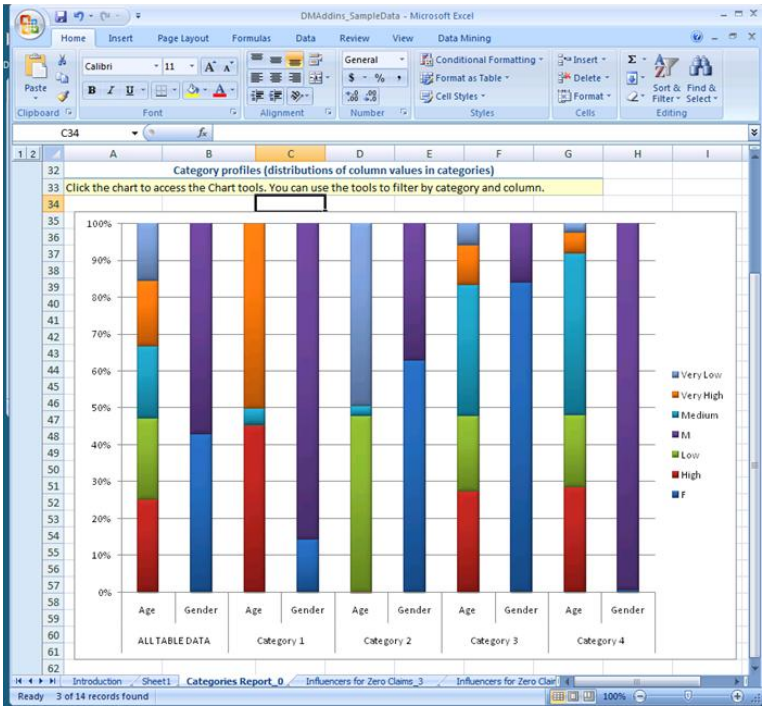


Figure 4. Comparing categories based on age and gender.

The Detect Categories tool may be a useful risk classification tool. Customers can be categorized using current or potential rating variables. Once customers are assigned to a category, the historical loss ratio of that category can be determined. If the loss ratios vary a great deal between categories, there is a strong chance that the current risk classification methodology needs to be reviewed.

3.2.3 Fill From Example

The Fill From Example tool provides a means of imputing missing data. If several rows of the

zero-claim sample dataset had no value in the married column, this tool could be used to predict the marital status of those records. A column containing the predicted values is appended to the table and a report similar to the Key Influencers report is created.

3.2.4 Forecast

The Forecast button uses a time series model to predict future values based on historical data. The algorithm automatically detects the periodicity of the data. This tool can be used by actuaries to predict the new business premium for the next quarter or the medical trend for the next year. The tool adds the predicted values to the bottom of the table and produces a line chart that shows the historical data and the predicted future values.

3.2.5 Highlight Exceptions

The Highlight Exceptions tool identifies anomalies in the data. Rows in the table that do not match the data's general patterns are highlighted. Often these anomalies are caused by data entry errors, making the Highlight Exceptions tool a powerful data-cleaning tool. In other cases, the tool identifies valid values that are simply outliers. These outliers often require further analysis before they feed into other data mining algorithms.

Outlier detection algorithms are often used to identify fraudulent claims. The technique can also be used to identify policies that are over/under priced, or policies that are rated using an inaccurate rating variables.

3.2.6 Scenario Analysis

The Scenario Analysis tools are primarily used for sensitivity analysis; the "Goal Seek" and "What-if" tools show how a model changes when the raw data is altered.

Goal Seek is used to determine which input variables need to be changed in order to reach a desired output. For example, a young customer who is judged to be a bad risk may become a good risk in a few years. The Goal Seek tool will determine how old the customer needs to be before the model judges them to be a good risk.

The What-If tool allows the user to see how the model changes if one of the variables changes. If all of the customers suddenly married, the model would be unable to use marital status to predict zero-claim status. The What-If tool allows the user to see the impact of this change on the model and its predictions.

3.2.7 Prediction Calculator

Since the cost of having an underwriter review a policy can be high, automated underwriting systems have become common. Policies that meet predetermined criteria are automatically issued, while other policies are forwarded to underwriters for review.

The Prediction Calculator tool can be used to create an automated underwriting system. If the average cost of a bad customer and the average profit from a good customer are known, the predictive model can be used to maximize estimated profits.

This is presented to the users as a series of drop-down boxes, which allow the users to select different values for each of the rating variables. Once the rating variable values are entered into the spreadsheet, the Prediction Calculator determines the odds of a true positive, a false positive, a true negative and a false negative. These odds are used to weight the costs and profits associated with each of the four possible outcomes. If the weighted average of the costs and profits is positive, the policy should be issued.

The Prediction Calculator is one example of an end-user tool that integrates data mining technology. Perhaps its greatest value to an actuary is that it clearly demonstrates how data mining can be used to improve a company's profitability.

3.2.8 Shopping Basket Analysis

The "Association Rules" data mining algorithm is used to conduct Shopping Basket Analysis. The algorithm identifies goods or services that are frequently purchased together. The output of the algorithm is a set of if-then statements along with a measure of accuracy; e.g. if a customer purchases Product A and Product B, then X% of the time they will also purchase Product C.

Shopping Basket Analysis is most frequently used in the retail industry, but it does have other applications. Healthcare organizations have used it to identify patients who are likely to have undiagnosed illnesses. Property and casualty insurers often use Shopping Basket Analysis to identify cross-selling opportunities (for example, customers who have a homeowners policy and an automobile policy are much more likely to purchase a policy for a recreational vehicle). In general, Shopping Basket Analysis provides an efficient method of analysis related transactional data.

3.3 The Data Mining Menu

The Data Mining menu is designed for users who are familiar with data mining concepts. The tools on the data mining menu are split into five sections: Data Preparation, Data Modeling, Accuracy and Verification, Model Usage, and Management. These tools allow users to complete every step in the data mining lifecycle without leaving the familiar Excel interface.

The tools available on the Data Mining menu provide full access to all the capabilities available in SQL Server Analysis Services. For example, the classify button allows users to categorize data using a CART algorithm, logistic regression, a naïve Bayes model, or a neural network. Users can also adjust algorithm parameters, such as the minimum support for a leaf in a decision tree or the number of nodes in the hidden layer of a neural network. The tools and options available in the data mining menu are much more flexible, and thus more powerful, than those available on the Table Tools menu discussed in the previous section.

3.3.1 Data Preparation

Data preparation is often the most time consuming part of the data mining process. A common rule of thumb is that 70% of a data mining project is spent cleaning data. The Data Mining Client provides Excel with some limited data cleaning capabilities. While theses tools are not as powerful as those including in a true ETL tool (such as SQL Server Integration Services), they may allow users to avoid the more complex data cleaning tools.

Traditionally, the first step in the modeling process is to look at the distribution of the data and its basic characteristics. The Explore Data button simply creates a histogram for a single column of data. This provides an easy means of identify data from a non-normal distribution.

The Clean Data button provides limited data cleansing capabilities. It allows users to identify outliers within a single column. Non-robust algorithms, such as linear regression, can be greatly influenced by outliers. The Clean Data button also allows users to re-label discrete values. For example, teenagers and twenty-year-olds could be re-labeled to have a single value for an age variable.

Sampling is the final step in the data preparation step of the data mining lifecycle. The Sample Data button can be used to select a random subset of the raw data. It can also be used to "oversample" the data. Oversampling is used when the raw data contains important, but rare, rows. Oversampling ensures that these important rows are not drowned out by less important data. This is especially important when developing models to predict rare events, such as large losses or fraudulent claims.

3.3.2 Classification Algorithms

Classification algorithms are used to separate data into distinct groups. Personal automobile clients can be classified as good drivers or bad drivers, or current customers could be classified as likely to renew their policy or likely to switch to a different company (this is known as churn modeling). The Classification button allows users to classify data using one of four algorithms: the naïve Bayes algorithm, logistic regression, decision trees, or neural networks.

The naïve Bayes algorithm is the fastest classification algorithm, but it often misses complex patterns that can be found by the other algorithms. The algorithm calculates conditional probabilities and then uses Bayes Theorem to predict the data's class. Since this involves relatively simple calculations, the naïve Bayes algorithm is often used on very large data sets or when a model needs to be trained very quickly.

Logistic regression is used when then classification is binary (i.e. true vs. false, yes vs. no, good vs. bad). Logistic regression is a type of generalized linear model. The output of a logistic regression is a number between zero and one, which represents the probability that the data falls into the "True" category.

Decision trees are often the preferred classification algorithm. Decision trees can capture more complex relationships than the naïve Bayes algorithm, and are not limited to binary classes. They are also not "black box" algorithms; the output of a decision tree algorithm is a set of rules that any individual can easily understand and use to classify data. These rules are often presented in the form of a tree-like graph, which give the algorithm its name.

Neural networks are the most advanced classification algorithm available. Neural networks can find complex nonlinear patterns within the data. Neural networks will often provide the best predictions. However, they can take a great deal of time to train and are "black box" algorithms. Explaining the reason behind a neural networks prediction can be difficult or impossible. This has prevented neural networks from being used in highly regulated insurance environments.

There are countless insurance applications of classification models. They can be used to identify fraudulent claims, or used to identify insureds who are likely to leave for another company. One popular use of classification models is to automatically sort potential customers into high risk or low risk groups. A new policy that is classified as high risk can be referred to an underwriter for review, while the low risk policies can be issued automatically.

3.3.3 Estimation Algorithms

Estimation algorithms are used to predict continuous values. They can be used to predict the lost

cost of a specific workers compensation policy or the number of accidents a driver will get into during the next year. SQL Server uses four estimation algorithms: decision trees, linear regression, logistic regression and neural networks.

Decision trees, linear regression and logistic regression are all special cases of the Classification and Regression Tree (CART) algorithm. The decision tree algorithm is the most general form of the CART algorithm and is also used as a classification algorithm. When a decision tree does not contain any "branches" (i.e. binary splits, such as separating the data into male and female subsets), the decision tree algorithm produces a liner regression model[8].

When estimating a binary value (true/false, yes/no, etc.), the desired output is a probability that the value is true/yes, etc. This probability is measured as a percent or as a value between 0 and 1. Logistic regression is often used to model binomially distributed data[9]. A logistic regression model is essentially a linear regression model, where the dependent variable is transformed using the logit transformation, logit=Ln(p/1-p). This transformation causes the model to output a value between 0 and 1, making it possible to model the probability of a true/yes outcome.

The same features that make neural networks powerful classification algorithms allow them to accurately estimate continuous data. However, the previously discussed limitations of neural networks limit their use as an estimation algorithm. In spite of their "black box" nature, neural networks can be used to confirm the results of other algorithms or to solve problems where the results, rather than the logic behind the results, are all that matters.

3.3.4 Clustering Algorithms

Classification and estimation algorithms are two classes of supervised algorithms. The term "supervised" is used to describe data mining algorithms that model a pre-selected dependant variable. "Unsupervised" algorithms, such as clustering algorithms, look at all of the available data in order to identify patterns. All patterns, rather than just those affecting the dependant variable, are reviewed and analyzed.

Clustering algorithms try to split records into similar groups. For example, clustering a video store's rental database might show three major groups of customers: those who enjoy action films, those who like romance movies, and customers who rent cartoons. This type of analysis can allow businesses to understand who their clients are and how to better serve them.

The Data Mining Client for Excel make it easy to interpret the groups identified by the clustering algorithm. The Browse window offers four view of the clusters. The first view is a cluster diagram,

shown in Figure 5.

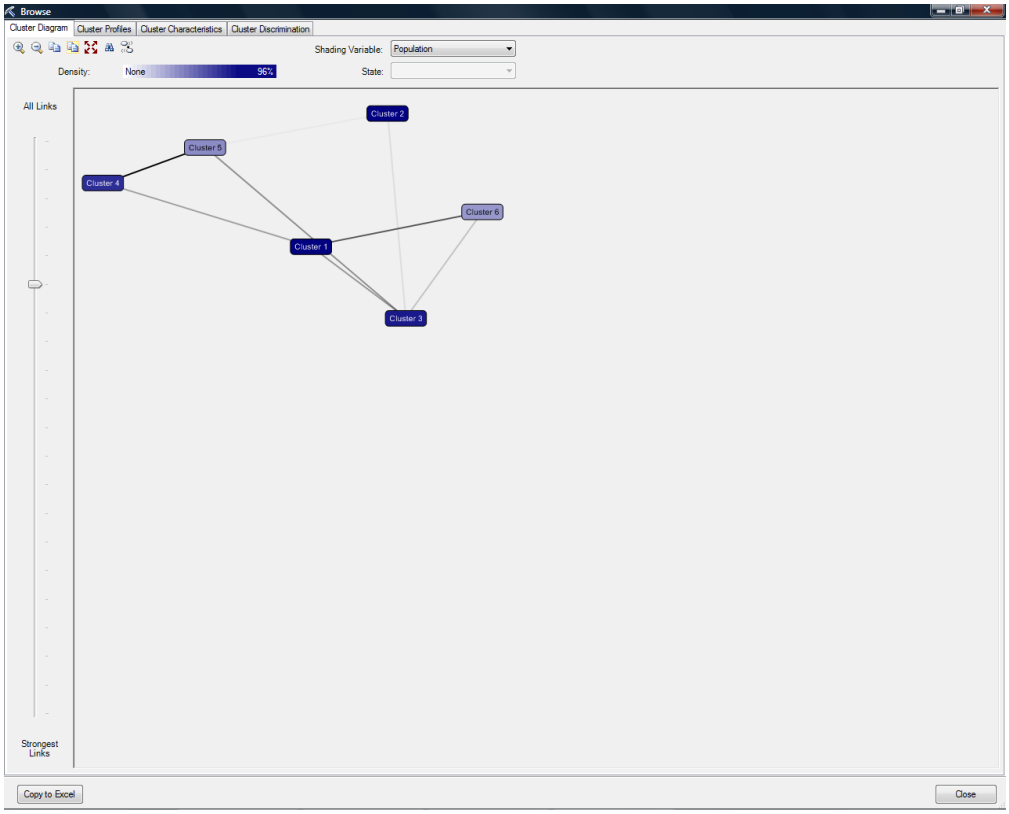


Figure 5. A Cluster Diagram.

A cluster diagram allows the analyst to see the relationship between various clusters, based on different attributes. The darkest line in Figure 5 shows that clusters 4 and 5 are most similar, while the various shades of color show how the cluster are related with respect to the "population" variable.

The second tab in the cluster viewer is shows "cluster profiles", which are essentially a univariate statistical analysis of each of the variables, for both the overall population and each cluster individually. This view makes it possible to identify the differences between the clusters.

The third tab, called "cluster characteristics", makes it possible for an analyst to understand the nature of a given cluster. Characteristics are values of a given variable that help distinguish one cluster from another. For example, if one cluster contains only young men while another contains women of any age, an analyst may consider gender to be the "most interesting" characteristic. The final tab, "cluster discriminants", uses these cluster characteristics to show the differences between two chosen clusters.

Clustering is a valuable data mining technique. While it is most commonly used by marketers to

segment a business's customers, it is important to remember that clustering can be used to pre-process data. By reducing a large number of variables into a single variable (the cluster names), clustering can make computationally intractable data mining problems solvable.

3.3.5 Time Series

Time series are statistical models that represent data that is collected over time. These models usually decompose data into a series of periodic components. For example, the daily sales at a pizzeria will depend on the day of the week (Fridays are busy, Mondays are slow), the day of the month (customers tend to eat out on the 1st and the 15th of the month, when they get paid), the season (summer is busy, winter is slow) and the year (inflation increases the dollar amount of the sales). A time series analysis will identify patterns like these and allow the analyst to forecast future data points.

3.3.6 Association Algorithms

The Associate button is used to create a shopping basket analysis. It performs the same analysis as the "Shopping Basket Analysis" button described in section 3.2.8, but allows the user to change more of the algorithms parameters.

3.3.7 Accuracy and Validation

The four buttons in the Accuracy and Validation section of the data mining toolbar allow analysts to evaluate the quality of a data mining model. Models can be evaluated using accuracy charts, a classification matrix, profit charts or the cross-validation method.

Figure 6 shows an accuracy chart. Also known as a lift chart, this graph shows how a chosen model compares to a perfect model and a model based on random guessing. Figure 6 was produced by applying the decision tree algorithm to the "Bike Buyer" data in the sample file that is installed with the Excel Data Mining Add-In. This lift chart shows how accurate the data mining model is; in this example, the 5% of individuals whom the model selects as most likely to buy a bicycle represent 22.91% of the total population of individuals who actually buy a bicycle.

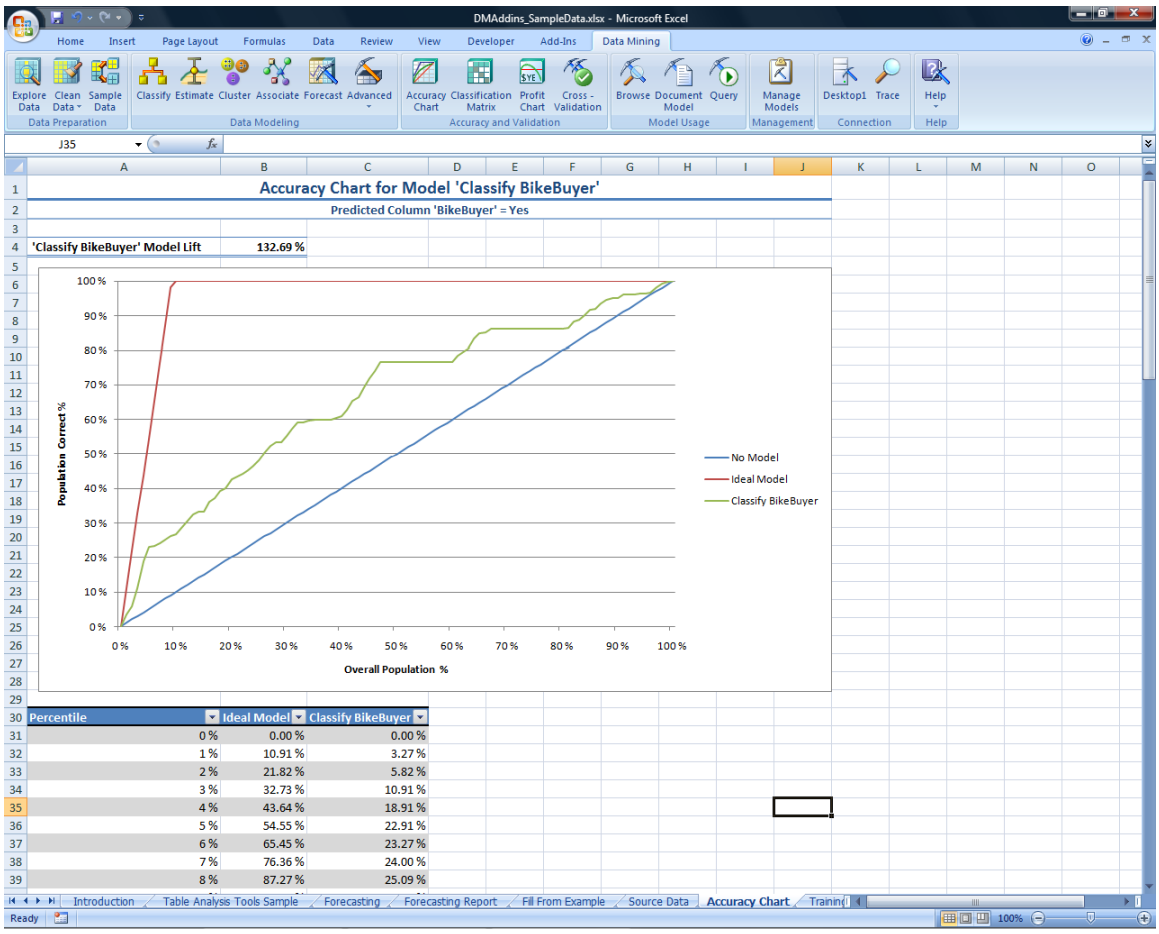


Figure 6. Using an Accuracy Chart to Evaluate a Data Mining Model

The Classification Matrix button produces a pivot table which shows the number of records that are correctly or incorrectly classified. For example, the profit chart button can be used to assist marketers planning a mailing campaign. The profit chart would ask for the cost of sending out a mailing and the profit from a successful mailing. It then uses the data mining model to determine the optimal number of mailing to send out.

The Cross-Validation button is the final means of evaluating a data mining model. Cross-validation randomly splits the data into disparate training and testing subsets. The training data is used to parameterize the mining model, while the testing data is used to measure the accuracy of the model. This process is repeated a number of times and the results of all of the models can then be reviewed. Variance among the outputs of these models is a sign that the models are over-fitting the data, while similar outputs are a sign that the model is an accurate depiction of the patterns found within the data.

These Accuracy and Validation tools often show that a data mining model does not meet the user's needs. An analyst may need to try other modeling algorithms, or conduct more data cleaning, or even add additional data in order to reach the desired level of accuracy and applicability.

3.3.8 Managing the Data Mining Models

Models that are built using Excel can be temporary or permanent. Temporary models disappear when Excel is closed. Permanent models are saved in the Analysis Services database.

The Manage Models button is used to delete, export or reprocess permanent models. A data mining model can only be accurate if the data used to train it is similar to the data it is trying to predict. If new or better data becomes available, the model should be reprocessed.

The model viewer that opens after a model is created can be reopened using the Browse button. This tool allows users to understand how the model is making it's predictions. The Document Model button copies some of the information presented in the model viewer into an easy-to-read Excel spreadsheet. It also lists the algorithm's parameters and basic information about the columns of data used in the model.

3.3.9 Using a Model to Make Predictions

The final step in the data mining process is applying the model to new data in order to make predictions. The Query button opens a wizard that allows users to select the model, new data and output for a prediction query. This wizard creates a prediction query, which runs new data through the trained model. The results of a prediction query include the prediction, the probability that the prediction is correct, and the prediction's support (the number of rows in the training data set that suggest this prediction is correct).

The query button is important because it allows Excel users to easily run new data through a data mining model on the SQL Server Analysis Services system,. Allowing end users to "score" their own data is a valid and effective means of deploying a data mining model. The Data Mining Extensions (DMX) code that is produced by the Query button can also be used for other purposes, such as building a Visual Basic application that returns data from the mining model.

4. CONCLUSION

Predictive modeling has become an essential actuarial skill. Since the value of a model is determined by both its quality and the amount of time required to build it, the ability to build models quickly may become a sustainable competitive advantage for some insurance companies.

The Data Mining Client for Excel 2007 allows actuaries to build a variety of powerful models very quickly. While specialized statistical packages will continue to serve as the primary data mining platforms at most companies, Excel 2007 and SQL Server Analysis Services certainly deserve a place in an actuary's toolbox.

5. REFERENCES

[1] Michael Domingo, RedmondMag.com 2007 Salary Survey, http://redmondmag.com/salarysurveys/
[2] An Excel 2007 trial edition is available at: http://us1.trymicrosoftoffice.com/product.aspx?sku=3203819&culture=en-US
[3] The SQL Server 2008 version of the data mining client for Excel 2007 is available at: http://www.microsoft.com/downloads/details.aspx?familyid=896A493A-2502-4795-94AE-E00632BA6DE7&displaylang=en
[4] A trial edition of SQL Server 2008 is available at: http://www.microsoft.com/sqlserver/2008/en/us/trial-software.aspx
[5] This video explains the process of setting up a SQL Server Analysis Services database using the Server Configuration Utility: http://www.microsoft.com/sql/technologies/dm/DMAddinConfigure/DMAddinConfigure.html
[6] JKP Application Development Services, "Working with Tables in Excel 2007", visited on 9/8/2008 http://www.jkp-ads.com/articles/excel2007tables.asp
[7] The author created the zero-claim dataset. It is not based on any real-world data and therefore the sample "results" are completely fictional.
[8] ZhaoHui Tang & Jamie MacLennan, *Data Mining with SQL Server 2005*, page 153
[9] Fred L. Ramsey & Daniel W. Schafer, *The Statistical Sleuth - A Course in Methods of Data Analysis* , page 614

Abbreviations and notations

CART, classification and regression tree
ETL, extract, transform and load
OLAP, online analytical processing
GLM, generalized linear models

Biography of the Author

Spyridon Ganas is an actuarial analyst at a non-profit healthcare organization. His work focuses on building actuarial business intelligence tools using the SQL Server platform. He has a bachelor's degree from Babson College and read applied statistics at St. Cross College, Oxford. He can be contacted at spyridon.ganas@stx.oxon.org

Duplicate FHA Single-Family Mortgage Records and Related Problems*

Thomas N. Herzog, PhD, ASA
Office of Evaluation
U.S. Department of Housing and Urban Development

*This work represents the personal views of the author rather than those of HUD and/or the government of the United States. A number of the examples used in this work have been modified to protect individual privacy and/or to simplify the exposition. In particular, none of the borrower names and addresses is intended to represent those of actual FHA-insured mortgages.

Abstract

The Federal Housing Administration (FHA) insures mortgages against the risk of foreclosure. Since its inception in 1934, FHA has insured over 37 million mortgages on single-family homes. This requires FHA to store data on a large number of mortgages. Because of the large number of mergers/acquisitions among mortgage lenders, it is sometimes difficult for the surviving lenders to maintain accurate databases. As a consequence, such lenders do not always transmit accurate/timely data to FHA on the termination of FHA-insured single-family mortgages that they are servicing. This, in turn, means that FHA's database has many mortgages listed as "active" that have in fact been terminated. In addition, FHA made a decision many years ago to omit the property address of insured mortgages from its databases on single-family mortgages because of the high cost of computer storage at that time. Although this decision was later reversed as such costs declined, even in the year 2008, FHA had a substantial number of "active" mortgage records without a corresponding property address.

In order to improve the quality of FHA's databases in these two aspects, we have applied a number of internal consistency checks and record linkage techniques. The first approach was to use a variety of internal consistency checks to identify "active" mortgage records whose underlying mortgages had in fact terminated. The second approach involved matching FHA records with corresponding records of the Government National Mortgage Association (GNMA). This second approach allowed us to (1) obtain property addresses from the GNMA database and add them to the FHA database as well as (2) identify additional "active" FHA mortgage records whose underlying mortgages had terminated.

We have employed a variety of internal consistency checks to identify and subsequently remove "duplicate" mortgage records from this database.

1. INTRODUCTION

1.1 Background

The Federal Housing Administration (FHA), an agency within the U.S. Department of Housing and Urban Development (HUD), insures mortgages against the risk that the borrower, for whatever reason, will be unable to continue making payments on his/her mortgage. The FHA's mortgage

*This work represents the personal views of the author rather than those of HUD and/or the government of the United States. A number of the examples used in this work have been modified to protect individual privacy and/or to simplify the exposition. In particular, none of the borrower names and addresses is intended to represent those of actual FHA-insured mortgages.

guarantee insurance programs are partitioned into four separate insurance funds. FHA's Mutual Mortgage Insurance Fund (MMIF) is its largest fund. The MMIF insures mortgages on single-family homes consisting primarily of single-family detached houses and townhouses. FHA reported that as of June 30, 2009, the MMIF had 4.9 million mortgages insured with an aggregate face amount of $604 billion. FHA's General Insurance Fund (GIF) also insures mortgages on single-family homes. In addition, the GIF insures loans on individual condominium units as well as on apartment buildings, nursing homes, hospitals, and mobile homes. The Cooperative Management Housing Insurance Fund (CMHIF) insures mortgages on cooperative apartment buildings. The Special Risk Insurance Fund (SRIF) insures mortgages on single-family homes, excluding condominiums, and apartment buildings.

Since its inception in 1934, the FHA has insured over 37 million mortgages on single-family homes. The bulk of these mortgages have been insured under FHA's Mutual Mortgage Insurance Fund (MMIF). The mutuality feature of this fund means that dividends (also known as distributive shares) may be paid to certain borrowers when they terminate their insurance. The amount of the dividend depends on the mortgage amount, the year the mortgage began amortizing, and the amortization plan of the mortgage. On September 1, 1983, FHA instituted a one-time premium collection system whereby the entire premium is paid in advance, and unearned premium refunds are paid to those who successfully terminate their loans prior to maturity.

1.2 Purpose

When a borrower refinances or prepays an FHA-insured single-family mortgage, the lender servicing that mortgage is supposed to notify FHA, and FHA is supposed to make the appropriate changes to its databases. Unfortunately, this process does not always work as intended. As a consequence, FHA has hundreds of thousands of mortgage records in its single-family data warehouse with a termination status of "active" when in fact the underlying mortgage has been refinanced, paid in full, or terminated for some other reason. Because (1) FHA only insures "first" mortgages, i.e., those with a primary lien on the underlying property, and (2) no condominium units[1] are supposed to be insured under the MMIF, it follows that there should be at most one "active" MMIF-insured mortgage per property address.

[1] For purposes of this paper, it is useful to assume that FHA-insured condominium units have never been insured under FHA's MMIF. In actuality, this was only true through September 30 2008, after which newly-originated mortgages on condominium units were insured under the MMIF.

In addition, a relatively small number of additional mortgages have been entered into FHA's single-family data warehouse under two or more identification numbers,[2] usually with only slight differences between the identification numbers and few, if any, differences in the case data. When such records are displayed together, it is usually apparent that both represent the same mortgage; however, because the identification numbers are different, the data warehouse treats both records as unique, individual mortgages. Our intent here is to describe some of the data problems we have investigated as well as the efforts we have made to improve the accuracy of the affected records. The consequences of these data problems will be discussed in Section 3.4.

1.3 FHA Single-family Data Warehouse

Currently, FHA's primary single-family database is known as the FHA single-family data warehouse. While the vast majority of its records do not have serious data problems, we have identified some with severe problems. Because so many cases are involved here, an error rate of only one or two percent could result in hundreds of thousands of incorrect records.

1.4 Outline of Paper

In the next section we discuss the concept of an FHA case number in order to facilitate the rest of the discussion of the paper. In Section 3, we discuss the problem of the duplicate mortgage records. In Section 4, we discuss the problem of the mortgage records with incorrect termination statuses. In both Sections 3 and 4, we begin by illustrating the nature of the problem, we then describe the procedures we used to identify and correct these problems, and finally we list some of the consequences of these errors. In Section 5, we summarize a scheme used to obtain property addresses on FHA-insured single-family mortgages from a database of mortgages maintained by the Government National Mortgage Association (GNMA).Finally, in Section 6, we describe schemes for estimating the number of mortgages records having the problems considered here.

2. FHA CASE NUMBERS ON SINGLE-FAMILY MORTGAGES

Beginning in January 1962, all FHA case numbers consisted of a 3-digit State and Office code prefix (SSO-) followed by a dash, a six digit serial number, and a final check digit (SSO-DDDDDDC). Within each HUD field office, the serial numbers are assigned chronologically. The use of the check-digit was

[2] These identification numbers are known as "FHA case numbers" and are discussed in detail in Section 2.

intended to improve the accuracy of the FHA case number; unfortunately, it was easy to circumvent the protective function of the check digit by appending an "X" to the end of a questionable FHA case number. This caused FHA's previous single-family computer system to calculate and insert the appropriate check digit for any number entered, thus passing over the incorrect or missing check digit. This allowed the processing of the invalid FHA case to continue and even gave its case number an aura of legitimacy because it would then have a "correct" check digit attached from that point on. Fortunately, this problem was eliminated with the implementation of HUD's Computerized Homes Underwriting Management System[3] (CHUMS) in the mid-1980s. The CHUMS assigns FHA case numbers automatically, leaving no chance for manual error.

3. THE PROBLEM WITH THE DUPLICATE MORTGAGE RECORDS

3.1 Statement of Problem

The first problem we considered was the existence of thousands of mortgage records entered into HUD's single-family data warehouse under two or more FHA case numbers, usually with only slight differences between the FHA case numbers and few, if any, differences in the case data.

[3] CHUMS is still in use today.

3.2 Procedures for Identifying Duplicate Mortgage Records

In Section 3.3, we present six examples of (potentially) duplicate case records within FHA's single-family data warehouse. One scheme identified cases whose FHA case number was not consistent with the zip code on its property address. This worked well on cases having a zip code and turned up about 1,000 duplicates. Unfortunately, it was not more useful because millions of case records do not have a zip code in the data warehouse. Another scheme focused on cases with the highest serial numbers and the lowest serial numbers for each office. A related scheme made use of the fact that serial numbers are assigned in chronological order. Both of these are types of "range tests." A final scheme used deterministic record linkage techniques to identify pairs of records with identical serial numbers and mortgages amounts. Such record pairs were then investigated manually using an information-retrieval system to identify and delete duplicate case records.[4]

3.3 Examples of Problem Cases Found in the Data Warehouse

We now discuss some representative examples that illustrate the problems discussed in Section 3.1:

Example 3.1:

Consider the following case record.

FHA Case Number	Street Address	City	State	Zip Code	Name(s) of Borrower
441-1451573	704 Hand Ave	Cincinnati	PA	45232	Collins, Larry & Juanita

Here we have a record with a Philadelphia, Pennsylvania office code of "441" but a Cincinnati, Ohio zip code. Further research showed that the office code should have been entered as "411" so that the actual record should have been:

FHA Case Number	Street Address	City	State	Zip Code	Name(s) of Borrower
411-1451573	704 Hand Ave	Cincinnati	OH	45232	Collins, Larry & Juanita

Example 3.2:

Consider the following case record.

FHA Case Number	Street Address	City	State	Zip Code	Name(s) of Borrower

[4] Section 5.4 of Herzog, Scheuren, and Winkler [2007] contains a general discussion of deterministic tests used in data quality work while Section 8.3 describes deterministic record linkage techniques of the type employed here.

441-2760279	1309 Oakland Road	Richmond	PA	23231	McNelly, Robert

Here we have a record with a Philadelphia, Pennsylvania office code of "441" but a Richmond, Virginia zip code. Further research showed that the office code should have been entered as "541" so that the actual record should have been:

FHA Case Number	Street Address	City	State	Zip Code	Name(s) of Borrower
541-2760279	1309 Oakland Road	Richmond	VA	23231	McNelly, Robert

Examples 3.1 and 3.2 are examples in which internal consistency checks were used to identify erroneous entries within mortgage records that led to the identification of mortgage records as duplicate records.

Example 3.3:

We next consider the following six contiguous case records extracted from FHA's data warehouse in ascending order of their FHA case numbers:

FHA Case Number	Initial Mortgage Amount	Begin Amortization Date	Contract Interest Rate	Name(s) of Borrower
131-5132066	$70,100	Aug-1987	10.5%	Bopp, Jeffrey S.
131-5132095	$47,750	Oct-1987	11.0%	Fudge, Robert W.
131-5132116	$76,500	Aug-1987	10.5%	Woods, Sherrie D.
131-5132151	$68,800	Jun-1984	14.5%	O'Hara Edward J Kim A
131-5132180	$47,600	Aug-1987	10.5%	Epps, Sherri A.
131-5132197	$43,250	Oct-1987	10.5%	Collins, Richard M.

In the third column of the table, we observe that the fourth record appears to be out of sequence. The year that this loan began to amortize was 1984 versus 1987 for the five other loans shown. Moreover, its contract interest rate of 14.5% is much higher than those of the other five loans. Further examination revealed that the correct FHA case number for this mortgage was 131-3807331 and so this record was a duplicate entry that needed to be deleted from the FHA single-family data warehouse. In a sense then, the record on FHA case number 131-5132151 has an inconsistency between its case number and both (1) its begin amortization date and (2) its interest rate.

Example 3.4:

In this example, we examine, in ascending order of the FHA case number, the first six case records on the FHA single-family data warehouse within office "372" having a begin amortization date on or after January 1, 1975.

FHA Case Number	Property Address	Begin Amortization Date	Contract Interest Rate	Name(s) of Borrower
372-0101064	295 Kings Highway Amhurst, NY	Dec-1979	11.5%	Bruce, Ronald H
372-0113867	160 Rand St Rochester, NY 14615	Apr-1983	12.0%	Stuart P D
372-0116726	538 Spencer Road Rochester, NY 14609	Jun-1983	12.0%	Laudico M J
372-0707605	256 Hazelwood Ave Buffalo, NY 14215	Jan-1975	9.5%	Richardson AF L
372-0707736	52 Ackerman St Rochester, NY 14609	Jan-1975	9.0%	Bowers John P
372-0708494	22 Worcester Place Buffalo, NY 14215	Jan-1975	9.0%	McDowell A L

We note here that the first three records listed appear to be out of sequence – or, as Naus [1975] says, out of range. Further research revealed that the FHA case numbers corresponding to these three records should have been 372-101064x, 372-113867x, and 372-116726x, respectively. This is an example of what Naus [1975] calls a range test.

The approaches just described were relatively naïve. We developed a more sophisticated approach with the assistance of the staff in HUD's Office of Information Policy Systems. The idea was to use computerized (deterministic) record linkage techniques to match case records. The first scheme we ran successfully involved finding records with identical serial numbers and identical mortgage amounts. At the time this work was done initially, the database used had around 10 million records. The result was a file consisting of nearly 200,000 matches from which we identified about 5,000 duplicate records by doing clerical follow-up/review. In this process, we focused primarily on comparing the property addresses of the matching pairs of records. We might have been able to do this more efficiently if we

had had software that standardized and parsed our property addresses. At the time we were unaware of the existence of commercial software that performs these tasks although some of this software is expensive.[5]

We consider two examples to illustrate this work.

Example 3.5:

FHA Case Number	Property Address	Initial Mortgage Amount	Begin Amortization Date	Interest Rate	Name(s) of Borrower
372-1132617	10998 Mill Rd Bethany, NY 14054	$33,000	Feb-1983	12.5%	Brown David O
374-1132613	10998 Mill Road Bethany, NY 14054	$33,000	Feb-1983	12.5%	Brown JR

Here, we have two records that we matched on the serial portion of their FHA case numbers – 113261-—as well as on their initial mortgage amounts—$33,000. It is clear that they also match on their begin amortization dates, interest rates, property addresses, and borrower name(s). We note the variation in the spelling of "road" in the property address field as well as the variation of the names in the borrower name field.

Example 3.6:

FHA Case Number	Property Address	Initial Mortgage Amount	Interest Rate	Name(s) of Borrower
371-0912411	1111 Stratford Ave Bronx, NY 10464	$54,950	15.5%	Bynum William Randolph
374-0912416	1111 Stratford Ave Bronx, NY 10464	$54,950	15.5%	Bynum William Randolph

Here again, we have two records that we matched on the serial portion of their FHA case numbers—091241—as well as on their initial mortgage amounts --$54,950.

[5] See Section 19.2 of Herzog, Scheuren, and Winkler [2007] for more details about such software.

3.4 Consequences of Problem of Duplicate Mortgage Records

Some of the current consequences of these errors are as follows:

- Such errors have a deleterious effect on these important databases and adversely affect their use for analytical (e.g., statistical and actuarial) studies. In particular, it is hard enough to construct accurate claim and prepayment models for the MMIF, even when the data are 100% accurate. Such errors also lead to an overestimate of the amount of insurance-in-force and a corresponding underestimate of the fund's capital ratio.
- Some valid FHA cases may not be entered onto the single-family data warehouse because another case had previously been entered onto the system with that FHA case number.
- Some claim payment requests may be delayed because the data on the case cannot be found on the single-family data warehouse under the correct FHA case number. This might occur if the FHA case number is not entered onto the system correctly. Such delays can be expensive as they increase HUD's interest costs.

Some consequences of these errors, encountered in the recent past, but not of consequence today include:

- There was potential for problems with unearned premium refunds. For example, HUD could have paid both an insurance claim and an unearned premium refund on the same FHA-insured mortgage, or HUD could have paid two or more unearned premium refunds on the same mortgage.
- These problems could have lead to fraud as unscrupulous tracers[6] pressured borrowers to accept multiple unearned premium refund payments, or as HUD employees or contractors attempted to take advantage of the situation.
- These errors could have contributed to the unfavorable publicity HUD receives for not finding borrowers who are supposedly owed money by HUD. This is in contrast to the vigorous efforts that other U.S. government agencies (e.g., the Internal Revenue Service) make to collect money owed the U.S. government.
- These errors had a deleterious effect on the financial condition of the Mutual Mortgage Insurance Fund. Even when a check was not sent to a borrower for a refund payment, the fund was nevertheless debited when an unearned premium was declared on a mortgage record in the data warehouse.

[6] "Often referred to as a third-party tracer (because the government is not directly involved with the refund process), a tracer can be defined as an information broker who works to locate individuals due a refund, notify them of unclaimed monies owed to them, help them obtain it from HUD/FHA, and receive a percentage of that refund as a fee in exchange for these services." Source: http://www.webspawner.com/users/mrsauz/.

4. MORTGAGE RECORDS WITH AN INCORRECT TERMINATION STATUS

4.1 Statement of Problem

The second problem we considered was the existence of hundreds of thousands of mortgage records residing on the single-family data warehouse and having a termination status of "active" when the underlying mortgage has actually terminated, usually by prepayment.

4.2 Procedures Used to Identify Mortgage Records with Incorrect Termination Status

The first scheme we used was a naive internal consistency check within the data warehouse to identify addresses as identical. We simply paired records that agreed on both (1) the first 10 alphanumeric characters of the street address of the insured property and (2) the first four digits of the zip code of the property address of the insured property. (As discussed in Chapter 8 of Herzog, Scheuren, and Winkler [2007], this is a type of record linkage.) Thus far, this process has worked reasonably well in that it has generated tens of thousands of pairs of records to examine and the termination status of a vast majority of these records was in fact in need of correction. Because the FHA single-family data warehouse has over 37 million records and the computer[7] we are using does not have the ability to process that many cases at once, we have blocked the data by office code. This means that we had to partition our data into four or five groups according to office code in order to process all of the cases.

We used a second internal consistency type of record-linkage scheme to identify mortgage records whose termination status was "active" but that were in fact prepayments as follows. From all of the mortgage records that had an entry in the field "old FHA case number" we extracted the old FHA case number. We then created a file consisting of all of the records with the old FHA case numbers whose termination status was "active." This enabled us to identify slightly over 8,000 mortgage records whose termination status needed to be changed from "active" to "terminated" by prepayment. The hope here was that these changes could be done in batch via an automated process rather than manually on a case-by-case basis.

A third scheme we used matched FHA records to records in a database maintained by the Government National Mortgage Association (GNMA). GNMA, like FHA, is an agency within HUD.

[7] We did our computing using IBM's APL2 Version 2. This has a maximum workspace size of 2 gigabytes.

GNMA's main task is to package FHA and VA mortgages into mortgage-backed securities and to sell these to investors. This is a more typical type of record linkage in that it involved two distinct databases. Here we were able to identify over 32,000 pairs of mortgage records in which the loan was listed as "active" on the record in the FHA data warehouse but the matching record was listed as "terminated by prepayment" in the GNMA database. The matching process was aided by the presence of unique identification numbers – namely, FHA case numbers – in both databases. To complicate matters slightly, the FHA case numbers in the GNMA database were frequently in formats other than the one required. Specifically, the field for the FHA case number in the GNMA database is supposed to consist of 15 digits. The first two digits are each supposed to be zero; the next ten digits are supposed to be the actual FHA case number including the check digit; and the last three digits are supposed to represent the FHA "ADP Section-of-the-Act" code. Frequently, this field in the GNMA database consisted of the three-digit FHA State/Office Code, followed by a dash, and ending with the six digit serial number and the check digit. Less frequently, the field had additional leading zeros. Finally, in a number of instances, the field had non-numeric characters (besides the dash in the fourth position and blank spaces at the end). To deal with these problems, we did separate analyses for those records that had a dash in the fourth position. Otherwise, we decided to delete from our analyses all case records whose FHA case number fields had any non-numeric characters.

4.3 Examples of Mortgages with Incorrect Termination Status

Example 4.1:

FHA Case Number	Property Address	Begin Amortization Date	Status of Loan	Interest Rate	Name(s) of Borrower
371-1019310	109-07 211th Place Queens Village, NY 11429	Mar-1982	Active	16.5%	Smith John Paulette
374-4413730	109-07 211th Place Queens Village, NY 11429	Sep-2004	Active	5.5%	Smith, Paulette

In this example, the data warehouse lists two active mortgages on a property located in Queens Village, New York, where in reality the first mortgage that was originated in March of 1982 at an interest rate of 16.5% has been refinanced at least once, i.e., during September 2004.

Example 4.2:

FHA Case Number	Property Address	Begin Amortization Date	Status of Loan	Interest Rate	Name(s) of Borrower
372-1221854	323 HIGHGATE AVE BUFFALO, NY 14215	Apr-1984	Active	13%	J. & K. Falkides
372-1519223	323 HIGHGATE AVE BUFFALO, NY 14215	Jan-1987	Prepaid	9%	Falkides, John P

In this example, the data warehouse lists one active mortgage and a later mortgage terminated by prepayment (denoted by "T") for a property in Buffalo, New York. It appears that the first mortgage was originated during April of 1984 at an interest rate of 13% and refinanced during January of 1987 at an interest rate of 9%.

Example 4.3:

FHA Case Number	Property Address	Begin Amortization Date	Status of Loan	Interest Rate	Name(s) of Borrower
131-5109339	14104 RTE 1750 COAL VALLEY, IL 61240	Oct-1986	Active	9.0%	Wangel, Dale J
131-7200510	14104 RTE 1750 COAL VALLEY, IL 61240	Jul-1994	Claimed	7.5%	Wangel, Dale E

In this example, the data warehouse lists one active mortgage and a later mortgage terminated by insurance claim for a property in Coal Valley, Illinois. Again, it appears that the first mortgage was originated during October of 1986 at an interest rate of 9% and refinanced during July of 1994 at an interest rate of 7.5%. The second loan eventually resulted in an insurance claim being paid by HUD.

4.4 Consequences of Problem of Incorrect Termination Status

We list below some of the consequences of this problem.

- The amount of insurance in force for the Mutual Mortgage Insurance Fund, the General Insurance Fund, and the Special Risk Insurance Fund are all overstated.
- Some lenders are paying periodic mortgage insurance premiums on mortgages that have already terminated.
- Some borrowers have not been paid unearned premium refunds or distributive shares to which they are (or were at one time) entitled.
- Such errors have a deleterious effect on these important databases and adversely affect their use for analytical (e.g., statistical and actuarial) studies. In particular, it is hard enough to construct accurate claim and prepayment models for the MMIF, even when the data are 100% accurate.

5. MORTGAGE RECORDS WITHOUT A PROPERTY ADDRESS

5.1 Statement of Problem

A number of years ago, some HUD staff made a decision not to have a field for property address on the HUD database of FHA-insured single-family mortgages. This decision has since been reversed, but HUD recently still had about 25,000 insured single-family mortgages on its single-family data warehouse

that lacked a property address – a critical data element. Although this is a large number of loans, it represents less than one percent of the FHA-insured single-family mortgages currently in-force.

5.2 Procedures Used to Find Property Addresses

We used the record linkage scheme of Section 4, in which we linked FHA mortgage records with GNMA records to append the property address from the GNMA record to the corresponding (i.e., matched) FHA record. Thus far, we have thereby obtained addresses for about 5,600 of the 25,000 FHA mortgage records.

5.3 Consequences of Missing Addresses

The following are some of the consequences of missing addresses:

- FHA could pay an insurance claim on a mortgage that it had not insured.
- FHA could pay a premium refund and/or a dividend on a mortgage that it had not insured.

6. ESTIMATING THE NUMBER OF PROBLEM MORTGAGE RECORDS

One scheme for estimating the number of duplicate records on the FHA single-family data warehouse involves the use of a procedure known as "capture-recapture." This is described in a number of texts, e.g., Bishop, Fienberg, and Holland [1975]. This involves drawing two or more independent samples from the data warehouse. If two samples—A and B—are used, for example, then this procedure entails identifying the number of duplicate records found (1) in both samples, (2) in sample A but not in sample B, and (3) in sample B but not in sample A.

At first glance, one might think that capture-recapture methods might work well for estimating the number of mortgages with the wrong termination status. However, this is not the case. We could only use capture-recapture methods here to estimate the number of mortgage records we could potentially correct using the scheme described in Section 4. This is primarily because some borrowers might terminate their mortgages either without refinancing with FHA or by selling their property to someone who does not take out an FHA-insured mortgage. So, we are faced with coming up with a different approach. One possible approach is to examine the contract interest rates of the "active" mortgages as well as the mortgages that are listed as "active" but have not paid any required periodic mortgage premiums during the last five years. The later type of mortgages can be identified because the servicing

lender identification number of such mortgages is given by "99995". The following table can be used to take a first, albeit naïve, approach to this problem.

TABLE 1

Number of "Active" MMIF Mortgages as of September 30, 2005

Contract Interest Rate	Servicing Lender Number		Total
	99995	Not 99995	
≥ 10%	64,650	271,695	336,345
≥8% but <10%	26,544	861,912	888,456
< 8%	4,583	2,704,051	2,708,634
Total	95,777	3,837,658	3,933,435

According to the table above, as of September 30, 2005, the data warehouse listed over one million mortgages as "active" with an annual contract interest rate of at least 8% even though such interest rates recently were as low as about 5%. Because of all of the work we have done using our naïve matching schemes, we suspected that a large proportion of these mortgages had, in fact, been prepaid. Moreover, the mortgages whose servicing lender has been assigned the number "99995" have not paid any required periodic premiums during the last ten years or so. Based on our extensive experience at examining these mortgage records, we felt brave enough to make the following estimate, albeit highly subjective, of the number of these nearly four million "active" mortgage records that we thought were in actuality "terminated."

TABLE 2

Estimated Number of "Active" MMIF Mortgages as of September 30, 2005,
That Have Actually Been "Terminated"

Mortgage Characteristics		Number of "Active" Mortgages	Estimated Percentage of Mortgages that should have "terminated"	Estimated Number of Mortgages that should have "terminated"
Servicing Lender Number	Contract Interest Rate			
99995	Any	95,777	95	92,000
Not 99995	≥ 10%	271,695	90	240,000
Not 99995	≥8% but <10%	861,912	50	430,000
Not 99995	< 8%	2,704,051	2	54,000
			TOTAL	816,000

With more recent data, we have taken another look at this situation.

TABLE 3

Number of "Active" MMIF Mortgages as of July 31, 2009

Contract Interest Rate	Servicing Lender Number		Total
	99995	Not 99995	
≥ 10%	39,060	99,358	138,418
≥8% but <10%	3,387	310,043	313,430
< 8%	2,801	4,598,217	4,601,018
Total	45,248	5,007,618	5,052,866

Again, we felt brave enough to make the following revised estimate, albeit highly subjective, of the number of these roughly five million "active" mortgage records that we thought were in actuality "terminated."

TABLE 4

Estimated Number of "Active" MMIF Mortgages as of July 31, 2009
That Have Actually Been "Terminated"

Mortgage Characteristics		Number of "Active" Mortgages	Estimated Percentage of Mortgages that should have "terminated"	Estimated Number of Mortgages that should have "terminated"
Servicing Lender Number	Contract Interest Rate			
99995	Any	45,248	95	43,000
Not 99995	≥ 10%	99,358	90	90,000
Not 99995	≥8% but <10%	310,043	50	155,000
Not 99995	< 8%	4,598,217	1	46,000
			TOTAL	334,000

This revised estimate of 334,000 case records is a vast improvement over our previous estimate of 831,000. Something is going right. At the moment, we can only speculate as to what that is. Perhaps, the reason is that with the recent slowdown in the number of mortgages originated that began during the summer of 2007, lenders are finally getting a chance to catch up on their back-office tasks.

We note here that the "rule-of-thumb" in effect until about 10 years ago was to refinance one's mortgage when the current mortgage interest rate is 2% (i.e., 200 basis points) below the annual contract interest rate on one's current mortgage. We realize that some people do not take advantage of this opportunity because their outstanding balance is low and the costs of refinancing outweigh the savings from the lower interest rates. (On the other hand, FHA offers a "streamlined" refinancing option with minimal costs to the borrower.) Other people do not refinance because they are financially naïve or are undergoing major personal problems. So, some keen observers of the mortgage market may think that our estimates are too high while perhaps others may conclude that they are too low. In

any case, we hope that working with our colleagues at HUD, we can keep this process moving and continue to correct these data fields as expeditiously as possible without introducing any additional errors in the process.

7. REFERENCES

[1] Bishop, Y., S. Fienberg, and P. Holland, *Discrete Multivariate Analysis*, MIT Press, Cambridge, Mass., 1975.

[2] Herzog, T.N., F.J. Scheuren, and W.E. Winkler, *Data Quality and Record Linkage Techniques*, Springer, New York, NY, 2007.

[3] Naus, J., *Data Quality Control and Editing*, Marcel Dekker, New York, 1975.

Acknowledgements

The author would like to thank William J. Eilerman, David A. Middaugh, Teri Hines, and Zenora Hines for their kind assistance with this work.

Biography of the Author

Thomas N. Herzog, Ph.D., A.S.A., is the chief actuary of the Federal Housing Administration in Washington, D.C. He has a Sc.B. in applied mathematics from Brown University and a Ph.D. in mathematics (with a major in statistics) from the University of Maryland. He is a Fellow of the American Statistical Association and an Associate of the Society of Actuaries. Dr. Herzog is the author or co-author of four books: *Introduction to Credibility Theory*, *Applications of Monte Carlo Methods to Finance and Insurance* (written with Prof. Graham Lord of Princeton University), *Models for Quantifying Risk*, (written with Robin Cunningham and Dick London) and *Data Quality and Record Linkage Techniques* (written with Fritz Scheuren and William Winkler). He is the winner of the 1990 AERF Practitioner's Award for a paper on *Home Equity Conversion Mortgages* (written with Theresa R. DiVenti).

Very Large Calculation Systems

A Specialized Solution for the Complex Needs of Advanced Knowledge Workers

James Madison

Abstract

Motivation. Advanced calculations on large data sets provide important business insights. Such calculations must be flexible enough for the dynamic nature of advanced analytics done by actuaries and other high-skill users, yet must also leverage the power and stability of large-scale IT systems. The system design detailed in this paper balances these concerns, delivering the optimal combination of both.

Method. The design is based on a pattern the author observed repeatedly during a decade of building systems for actuaries and researchers: data is provided using traditional data warehousing, the calculations are implemented by IT in a manner that solidifies stable elements yet externalizes change-prone elements, system operation is exposed through self-service interfaces that allow users to configure and run the calculations against large data volumes, and the output is delivered by standard business intelligence tools and customized approaches.

Results. Systems with this design: optimize the users' time by moving the majority of the lower-value work to IT, have solid auditability and legal compliance, and provide high computing power and storage capacity, but do require users to give up some amount of control and flexibility.

Conclusions. Organizations should use this design to give their most advanced knowledge workers high power, localized control, and optimal efficiency.

Keywords. Data warehousing, exploratory data analysis, actuarial systems, ratemaking, modeling, simulation.

1. Introduction 3
1.1 Research Context 4
1.2 Objective 5
1.2.1 Points of Clarity Before Starting 5
2. Top-Level Architecture 6
3. Actuarial View 8
3.1 Routine Analytic Components 8
3.1.1 Data Warehouse 9
3.1.2 Standard Business Intelligence Tools 9
3.2 Flexible Analytic Components 10
3.2.1 High-Power Data Access 10
3.2.2 Exploration Area 11
3.2.3 Calculation Experiments 13
3.3 Justifying the VLCS 13
3.4 VLCS Components 14
3.4.1 Parameter Interface 15
3.4.2 Execution Interface 16
3.4.3 Generated Actuarial Data 18
3.5 Summary by Example: Data Mining 19
4. IT View 23
4.1 Operational Systems 23
4.2 Data Warehouse 24
4.3 Parameter Interface 25
4.3.1 Parameters by Screens 25
4.3.2 Parameters by Spreadsheets 26
4.3.3 Parameters by SQL 27
4.3.4 Parameters by Code Modules 27
4.4 Loaded Parameters 28
4.5 Execution Interface 28
4.6 Stable Calculations 28
4.7 High-Power Data Access & Overall Performance 31
4.7.1 Hardware Capacity 31
4.7.2 Partitioning and Parallelism 32
4.7.3 Networking 32
4.7.4 Workload Management 32
4.8 Standard BI Tools 33
5. Results and Discussion 34
5.1 A Complete and Complimentary Solution Suite 34
5.2 Reference and Reality 34
5.3 A Targeted and Challenging Undertaking 34
5.4 Governance and Maintenance Processes 35
5.5 Corporate Standards and the Center of Excellence 35
5.6 Build Versus Buy 35
5.7 Higher-End Work with Lower-End Skills 36
5.8 Agile Software Development 36
6. Conclusion 37
7. References 37

1. INTRODUCTION

Creative knowledge work and solid IT systems are necessarily at odds—the former requires high flexibility, the latter requires high stability. Reconciling their natures is difficult, so many organizations leave them separate. This can leave important knowledge work disconnected from mainstream systems and relegate mainstream systems to lower-value analysis work. Organizations seeking to maximize every analytical opportunity must bridge this gap by maximizing the efficiency of each area, then fully integrating them. This paper describes how to do it.

Flexibility is a dominant characteristic of creative knowledge work. Actuaries may operate from a strong mathematical foundation, but new and emerging opportunities start as little more than vague ideas. Turning ideas into actionable information requires looking at data—often lots of it—manipulating it, observing the outcome, getting more data, manipulating it more, observing it again, and so on through many iterations, often with several serendipitous epiphanies along the way.

Stability is a dominant characteristic of IT systems. During development, systems must be built to a fixed timeline and budget, multiple IT skill sets must converge on a single outcome, developers must learn business logic, and changes to code once it is written can have large propagation effects. During operations, systems must meet service level agreements (SLAs), be supported by IT staff of lesser skill, exist in the corporate systems environment for many years, rarely crash, and have their changes coordinated with multiple business units.

Little about these two worlds works well together. The actuary rightly asserts, "I don't know what I want," and IT says, "We can only build it if you specify it." When something finally does get produced, the actuary sees it for the first time and immediately recognizes a way to do it better. The change may take only minutes to observe, but then requires hours or days to explain to IT, and weeks or months to implement. If getting it through IT is painful enough, the actuary may just solve the problem in her favorite desktop analytical tool where the valuable logic stays until the day she leaves—with no one else knowing how it works, thus leaving the organization in a precarious position.

Faced with this conflict, most organizations silo the two needs. Not intentionally, but de facto. Actuaries do some of their most important work in ways that few others understand. IT builds comparatively low-value systems that are quite useful for standard business intelligence (BI) but miss out on the greatest analytical opportunities. The solution is to build a middle ground that serves the needs of both areas, bringing together the value that each provides while also maximizing the

efficiency of each area individually. When done properly, organizations can be confident that the full spectrum of analytic work is efficiently supported in their organization—from empowering the most creative analytic work, to codifying all the best ideas in long-term IT systems that protect and deliver the organization's most critical competitive analytical knowledge.

1.1 Research Context

Industry literature elaborates extensively on systems and data that are useful to the majority of the business community. These are shown in yellow in Figure 1.

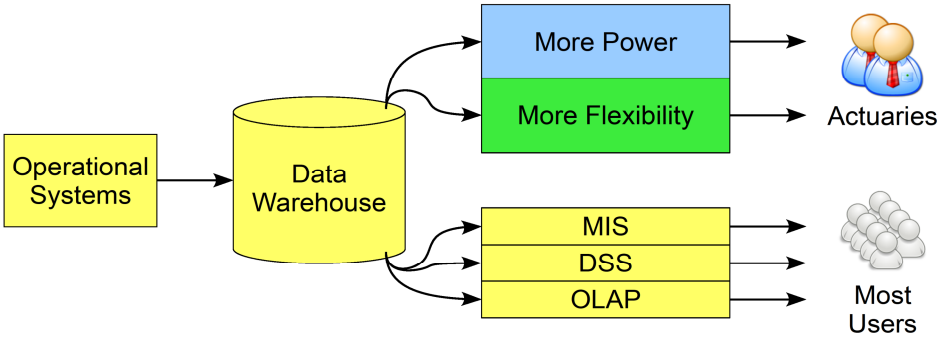


Figure 1 – Standard analytical systems serve most users—actuaries need more.

Operational systems, such as those in call centers and on consumer Web sites, generate transactional data that flows into a data warehouse (DW). The DW integrates the data into various standard analytical systems including management information systems (MIS) for executives, decision support systems (DSS) for intermediate managers, and on-line analytical processing (OLAP) systems for analysts.

Trouble is, a small but important minority of the business community needs more from their systems and data than standard analytical systems can deliver. Actuaries, statisticians, engineers, researchers and those in similar roles need more than the reporting or drilling that standard analytical systems provide. They need systems with the high flexibility required by the creative nature of advanced knowledge work. They need systems that perform advanced calculations—statistical analysis, simulations, data mining, conditional logic, predictive analytics, optimization modeling, emerging algorithms based on the cutting-edge of industry knowledge, and the many tried-and-true insurance algorithms that are inherently advanced such as rerating, loss development, risk analysis, reserving, and indications. They need to perform these calculations against large amounts of data—terabytes of data, collected from many sources around the organization, enhanced with external data,

preserved across years, even decades, with potentially many intermediate what-if versions. They need significant system resources—the raw computing power necessary to run all these calculations against all this data, yet still have answers come back at the speed of thought, lest important insights get lost while waiting for information to return. These needs demand a level of power, arbitrarily defined system behavior, and data use that is difficult to predict in advance. These needs are shown categorically in blue and green in Figure 1 and are the main focus of this paper.

MIS, DSS, OLAP, and similar mainstream analytical systems simply cannot handle these extreme needs for high-power and high-flexibility, nor were they ever intended to. Specialized vendor tools may attempt to address such needs, but vendor tools necessarily make generic assumptions that serve a wide customer base, thus making it difficult for any one customer to incorporate the uniqueness that creates competitive advantage—the very thing that most high-end analysis is intended to find!

1.2 Objective

The very large calculation system (VLCS) handles these extreme needs. The VLCS is a system design pattern that organizations can follow to create systems that provide calculation-intensive processing, large-volume data handling, and high user flexibility, while also addressing corporate IT processes, legal compliance, and similar non-technical constraints.

The objective of this paper is to address the major components of the VLCS as they are seen from the view of both actuarial and IT. The actuarial view details the functionality provided, what is possible with good system engineering, and what actuarial can reasonably demand from IT. The IT view elaborates on important back-end details, but in language meant to be understandable by technically savvy actuaries in joint conversations with IT.

1.2.1 Points of Clarity Before Starting

Nearly every aspect of the VLCS design already exists in some form in most organizations and has been written about somewhere in industry literature—this is one of its greatest strengths—no single aspect should be new or peculiar to actuaries or IT professionals with a few years of experience in a complex environment. The unique value presented here is that of providing a unified architecture, common language, and solid roadmap that all stakeholders can use to explicitly define and design such systems, and organize and justify the projects that build them.

Actuaries are not the only ones who need VLCSs, so non-actuaries are invited to interpret all references to *actuaries* and the *actuarial department* as their own role and group. Likewise, not all actuarial work demands the extreme power of a VLCS, so *actuarial work* as it is used here means the high end of the actuarial work continuum, where a VLCS might be applicable.

In any sufficiently advanced computing system, many system components interact across multiple layers of hardware, software, networking, and overall system design to deliver the simple outcome seen on the screen of the user. This paper uses the term *tool* to mean the simple thing the user sees on the screen and uses to get his job done. See the Definitions section for more.

2. TOP-LEVEL ARCHITECTURE

The top-level architecture of the VLCS is shown in Figure 2.

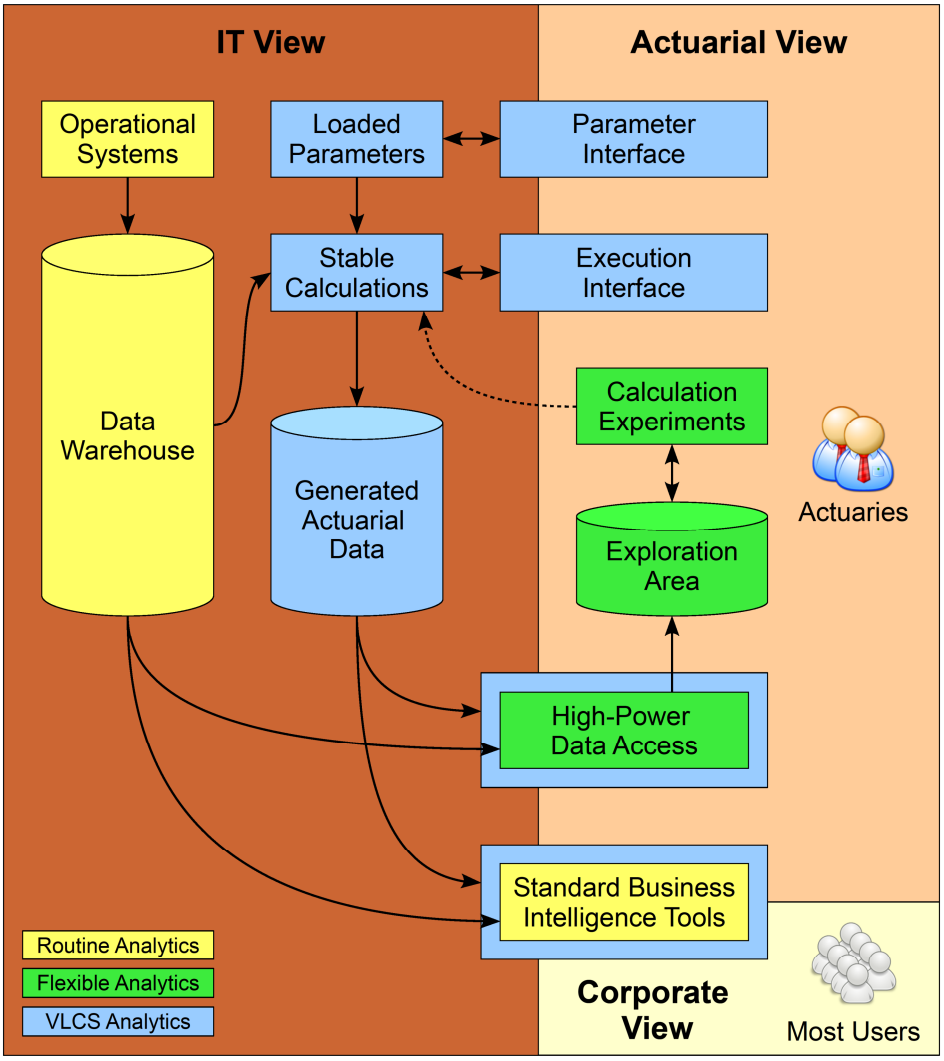


Figure 2 – Top-level architecture of the VLCS

The diagram reads as follows: *Operational systems* feed into a *data warehouse* that cleanses and integrates the data and preserves its history. *Standard business intelligence tools* deliver the data to *most users*, meeting most of their analytical needs most of the time. *Actuaries* tend to have more advanced needs, so actuaries access the data through *high-power data access* in addition to the standard tools. Actuaries keep this data and various working data sets in an *exploration area* that is fairly large and minimally governed. Actuaries perform data manipulations, what-if analyses, data mining, and other *calculation experiments* requiring a high level of flexibility. Some fraction of that work eventually yields important and clearly defined insights that are converted to *stable calculations* that IT builds into the organization's mainstream systems. Stable as they are, such calculations still vary across such things

as time, geography, customer traits, line of business, product, and similar grouping criteria, so a *parameter interface* is provided to give actuaries direct control over the calculations with no IT involvement. Running the system is done either by IT or directly by actuaries using an *execution interface*. The execution of the stable calculations using the *loaded parameters* results in *generated actuarial data*. This new data is fed back into the same flows as the regular DW data. Feeding it into the standard BI tools gives the larger business community well-defined and governed access to actuarial's most advanced calculations. Feeding it into the high-power data access gives actuarial a closed loop that allows large-scale what-if analysis using the full power and deep data of the DW platform.

Systems with this architecture can directly address the most data- and calculation-intensive questions a business can ask. Among the most advanced applications personally encountered by the author are rerated earned premium at present rates, loss development, reserving, geographic risk concentration, and rate indications. Other good candidates include any business question requiring years or decades of fine-grain data to be manipulated with highly detailed and varying parameters.

3. ACTUARIAL VIEW

The components of the actuarial view fall into three categories: 1) Routine analytic components: those that IT delivers to any team with repeatable work needing routine analytics. 2) Flexible analytic components: those delivered to any team with open-ended work needing flexible analytics. 3) VLCS components: those delivered as part of controlling the VLCS and utilizing its output. These categories are shown in yellow, green, and blue respectively in the top-level architecture.

3.1 Routine Analytic Components

Before attempting a solution as advanced as a VLCS, an organization needs to have succeeded at the basics. This includes, at a minimum, having a DW and using standard BI tools. The DW is required because it is the source of data that goes into the VLCS, and because it stores the VLCS output for integration with the larger organization. The BI tools are required because they answer basic analytical questions that establish the context for VLCS answers, and because they are the channel for delivery to the larger organization. DW and BI tools are covered at length in industry literature, both in general and with an actuarial focus[1], so they are reviewed here only enough to show their important foundational effect as it relates to the VLCS design.

3.1.1 Data Warehouse

A DW is a large repository of data that is historical, integrated, granular, and subject-oriented[2], referred to here as HIGSO form. Before building a VLCS, ensure that a large data repository can be found that has as many HIGSO characteristics as possible: many years of history (at least five years typically, up to 20 or more for asbestos, agent orange, mold, and other long-tail concerns), integration of many sources (policy writing system, claims system, etc.), deep granularity (vehicle identification number, not just year, make, and model; 9-digit zip, not just 5-digit zip; latitude/longitude, not just 9-digit zip; etc.), and diverse subjects regardless of which source it comes from (risks, coverages, vehicles, geography, service requests, demographic characteristics, etc.). Such an environment is a prerequisite to the VLCS because it provides the deep data that the VLCS must process.

3.1.2 Standard Business Intelligence Tools

The majority of access to the DW is done by commercial BI tools. At their simplest, these tools provide predefined reports. At their most robust, they provide OLAP and basic predictive analytics. For our purposes, the robust end of this spectrum can be considered in place when users can do the following with delivered data:

1) Drill down/up – Moving to lower levels of detail and higher levels of summary. For example, state down to county, down to city; agency up to territory, up to region.
2) Drill across – Moving among different collections of metrics. For example, from premiums across to losses to find loss ratio across to expenses to find combined ratio.
3) Slice & dice – Narrowing the metrics to a specific scope of interest. For example, only the male slice of customers in the urban-and-coastal dice of geography.
4) Statistics & trends – Performing basic analytics across time. For example, show standard deviation to find catastrophic outliers and do this for the past 36 months.

Achieving these four fundamental BI activities is a function of both the DW and the BI tools, with the data model of the DW being the more important of the two. A DW that facilitates these activities is known as dimensionally modeled[3]. It is the dimensions of the data that the user is drilling, slicing, and trending. Any organization that is serious about using its DW for driving its organization with analytics must model its data dimensionally. BI tools must then present this dimensional model in its most usable form. The dimensional form has a number of variations and

names such as OLAP, star schemas, pivot tables, and cubes; in practice, these are all just variations on achieving the same four activities.

The four fundamental BI activities combined with the four HIGSO characteristics of a DW constitute the majority of analysis that most business users need to do, even actuaries. If actuaries can drill up, drill down, drill across, slice and dice, apply all major statistical functions, and trend across any amount of history, data sourcing, depth of grain, and subject orientation, how much is left? All that is left are special situations and hard cases, which together lead us down the paths of flexibility and the VLCS.

3.2 Flexible Analytic Components

The DW and BI tools together should prove to be a valuable foundation for any organization's analytical needs, but when they also prove too rigid for certain dynamic needs, more flexibility must be introduced. The components of flexibility are access more powerful than standard BI tools, a very large storage area to hold data of arbitrarily defined data sets, and enough computing power to run complex analysis.

3.2.1 High-Power Data Access

High-power access results from powerful tools and almost unlimited authority to read the data. Such tools and authority contrasts with standard BI tools, which are designed to provide ease-of-use for routine tasks, and whose authority model is one of limiting users to only well-defined and highly refined data—a good value proposition most of the time, but potentially limiting for advanced work.

From the actuaries' view, high-power access essentially means having tools that allow the direct input of SQL and the capability to run it directly on the DW platform. Direct SQL input should be in addition to simpler graphical functionality that continues to keep routine work simple. The tools are usually the easy part—most good standard BI tools have a "back door" that can be easily enabled by IT to allow arbitrary SQL to be input and run on the underlying platform, and specialized data access tools almost certainly have such behavior.

The harder part is that IT is often concerned that actuaries may consume too much system power or may misunderstand the data. The power-consumption risk exists at several places throughout the architecture and will be addressed in the *workload management* topic toward the end of the paper. The misunderstanding risk is a serious one and is best addressed with the following policy: for a fairly high percentage of the data, actuaries might be on the cutting edge of the

company's understanding of its data, thus largely on their own. For example, a company's policy writing system might be 30 years old and capture 20,000+ data elements, but the DW may only officially define and support 500 of them in full dimensionalized HIGSO form. The benefit of virtually unlimited access is that important data elements could be discovered that would never see the light of day if actuaries could not dig into them. The risk is that actuaries may find themselves weeding through useless data that could get misinterpreted or find its way into institutional reporting contexts. The tipping point between the benefit and risk cannot be predefined, but the goal is to empower actuaries with SQL against all possible data so that their judgment of what is and is not valuable can be brought to bear, rather than IT filtering such judgments via typically slow and expensive processes.

3.2.2 Exploration Area

High-power data access reaches into the DW in a read-only manner, but actuaries frequently need write access for such things as storing results when query q_{n+1} requires as input the output of q_n, developing ideas by reviewing intermediate results, and storing particularly interesting outcomes for later review. These needs are met by having a large area with write access and high freedom known in the industry as an exploration area[4] or sandbox[5]. This area is a logical concept that IT can physicalize on any major database and server product suite—from large, generic database systems like Oracle and Microsoft, to specialized players like Teradata, Netezza, and SAS. This area should be as connected as possible to the main DW, provide significant storage capacity, have a liberal usage policy, offer some level of monitoring and maintenance, but restrict institutional reporting.

The optimal connection between the exploration area and the DW is achieved by collocating them on the same hardware, thus providing the maximum data transfer rate and the sharing of common tools. If collocation is not possible, a high-speed connection must be used between the areas so that actuaries can move data from the DW to the exploration area as efficiently as possible.

The storage capacity of the exploration area must be large enough to accommodate the base data pulled from the DW as well as intermediate result sets produced by the actuaries. As a starting estimate: determine the largest DW table the actuaries will use, multiply that by two to accommodate all the small tables that will be joined to it, multiply that by three for stored intermediate results, and multiply that by the number of actuaries doing exploration work. This usually results in a financially non-trivial number; constrain it as needed based on budget realities. When allocating this space,

avoid the temptation to have shared accounts since the lack of accountability can cause things to grow out of control. Instead, allocate the space to individual accounts and have IT train the actuaries in cross-account sharing, which they should find quite straightforward once they know it.

The usage policy on the individual accounts must be extremely liberal, the only real restriction being the total storage capacity allocation. Avoid or minimize other restrictions such as limits on the number of queries, types of queries, loading of third-party data, and similar activities.

Monitoring and maintenance of the exploration area should be done by IT to help the actuarial community understand what is going on in the environment: tracking data sets by who created them, who read them, and the operations performed against them; backing them up, recovering them if something goes wrong; having the dates for all these activities. These can be very useful to actuaries and are relatively easy for IT to establish using functionality built into the platform itself. Such value-adds are one of the main reasons the organization should build a centralized exploration area, have IT manage it, and encourage actuaries to use it. More advanced monitoring in the exploration area, such as verifying the quality of any particular data set, tracking the lineage of which data sets were used to produce which other data sets, or certifying data sets for their legal compliance, cannot be done in the exploration area by using built-in platform functionality and instead require the kind of customized programming that can only be done as part of building an IT system. This more advanced monitoring would also require having to institute restrictions on what actuaries can and cannot do, which begins to undermine the primary purpose of the exploration area.

Because quality, lineage, certification, and similarly advanced monitoring is challenging, the exploration area should produce little to no institutional reporting, or if it does, it should be clearly disclaimed and done at the risk of actuarial management. Institutional reporting is reporting that goes beyond the actuarial department, including senior leadership for important decisions, other departments for their operations, government agencies such as state insurance offices, and external organizations where SOX, GLBA, HIPAA, and similar laws may apply. Given that the cost of productionalizing exploration work can be as high as nine times the cost of developing the exploration work itself[6], it can be quite tempting to do institutional reporting from the exploration area, but this is a very slippery slope that must be actively avoided. Institutional reporting that requires the output of advanced calculations strongly indicates moving to a VLCS since IT is long-accustomed to handling compliance.

3.2.3 Calculation Experiments

With access to unlimited corporate data and ample working space, actuaries can now perform calculation experiments that explore the ideas they are attempting to turn into actionable information. Such experiments necessarily require extensive querying, joining, binning, filtering, redefining, translating of values, and storing of intermediate result sets—actions that generally result in some form of computer code, the size and complexity of which can become significant. The intensity of the work and the extensiveness of the code require that IT provides sufficient computing power, the freedom to install new or specialized software, and the right to use official corporate software in a highly flexible manner.

Sufficient computing power, like high-power data access, pushes in the direction of collocation of advanced actuarial work with the main DW. The DW will generally have more power than any secondary platform that could be created. Keeping the experimental work on this platform leverages this power and prevents having to justify and maintain stand-alone hardware.

Software for calculation experiments must balance two competing forces. It must include any software that actuaries need to work efficiently, yet it should consider the possibility of promotion to an IT system in the future. The need for efficiency requires that IT policies allow software installation in individual user accounts with minimal barriers. It also requires IT to help install such software when doing so requires system-level changes that are not possible with individual user authority alone. The need for future promotion asks that actuaries consider tools used by IT as the first suite of actuarial tools. If actuaries are open to using IT-sanctioned tools from the outset, even if they may not be their favorite or their first choice, it will make promotion of the stable calculations into a formal IT system much simpler and cheaper in the future. For example, actuaries might be inclined to pull data out of the DW and into Microsoft Access to do exploration, but if they work in an organization that uses Microsoft SQL Server as the DW platform, the actuaries might find that working directly in an account on the SQL Server is not much harder than working in Access, yet it is much easier to have IT productionalize SQL Server work than Access work should the work grow beyond just exploration.

3.3 Justifying the VLCS

The DW, BI tools, and exploration area together constitute a legitimate service model from IT to business areas. It may be quite reasonable to stop at this point and declare the IT service model for data delivery complete. Stopping here could also present several challenges. These are briefly

described in the "Consider a VLCS if…" column of Table 1. These challenges are directly addressed by creating a VLCS. However, creating a VLCS presents its own challenges. These are detailed in the "Defer a VLCS if…" column of Table 1. The situation presents a classic trade-off. Determining which path is right for a particular organization, department, or team is entirely up to those involved. The decision makers in such a situation are invited to use Table 1 as a guide during the decision making process.

Criteria	Defer a VLCS if…	Consider a VLCS if…
Routine work	**Integrated** – Routine work is tightly coupled with creative knowledge work	**Compartmentalized** – Routine work can be separated and handed to IT for automation
Structural stability	**Dynamic** – The actual nature of the analysis work changes as discoveries are made	**Stable** – The nature of the work does not change, only the data and parameters going into the work
Legal compliance	**Localized** – Decisions based on analysis do not go beyond well-defined internal use	**Constrained** – Legal or internal compliance require auditability and similar standard value-adds from IT
Corporate integration	**Limited** – The users of the actuarial work are actuarial itself or a small community they can manage	**Shared** – The work of actuarial has an organizational benefit realized by connecting it to the main DW flow
Bureaucracy tolerance	**Eager** – The turnaround time required for changes is short even for fairly stable elements of the system	**Patient** – The natural latency of IT is bearable when changes are requested for components under IT control
Staff change	**Stable** – Turnover is low and advanced-but-recurring work is well understood by several actuarial staff	**Churn** – The joke "no one's quite sure how it works" is more ominous than humorous when actuaries leave
Processing intensity	**Moderate** – Processing can be done within required time windows using hardware under actuarial control	**High** – Processing intensity demands the kind of hardware that actuarial cannot support on its own
Data volume	**Low** – Exploration area flexibility is valuable enough to justify having to manage the data and code in it	**High** – Working the data in its native environment provides storage and processing efficiency worth having

Table 1 – Criteria to weigh when considering a VLCS.

3.4 VLCS Components

So that the actuarial view can be discussed in its entirety, assume that a VLCS is justified, has been built, and has been delivered according to the details discussed later in the IT view. As part of the delivery, actuaries must have interface points that give them a high level of control, and simple yet powerful access to the output for themselves and the larger organization.

3.4.1 Parameter Interface

The parameter interface parameterizes business rules and the data generation process, putting both under the direct control of actuaries. Parameters are inputs to business rules that cause outputs to vary. Standard BI tools have some level of parameterization, the VLCS has more. BI tool parameters include such things as pick lists, user-defined hierarchies, pre-defined aggregates, value banding, and other inputs that influence the output, but not the business rules nor the state of the underlying data. VLCS parameters influence output, but they also vary business rules and change the data that gets generated.

As examples: A traditional BI tool may allow actuaries to group losses by range where some range of values is classified as catastrophic, but a VLCS might allow actuaries to provide catastrophe thresholds that cause different rating variables to be used to generate rerated earned premium. A traditional BI tool may allow trending of losses by geographic characteristics, but a VLCS may allow the calculation of potential future losses based on geographic proximity to geographic risks such as fault lines or nuclear power plants as defined by actuaries. A traditional BI tool may allow analysis of losses across time, but a VLCS may generate reserves for losses projected to ultimate based on assumptions provided by actuaries.

Notice that for each of these, variations in input (e.g., catastrophe thresholds and rating variables) cause variation in business rules (e.g., which thresholds cause which rating variables to be used) or generated data (e.g., the earned premium based catastrophe-stratified rates)—variations that cannot be solved from source data alone. What constitutes a catastrophe? Which rating variables are to be used for such catastrophe thresholds? How are exposures to be rerated? What trending assumptions are to be used to define the future? How close does a location have to be to a risk to be worrisome? Which locations are considered risky sites? How conservative or liberal should the reserving policy be? What are the assumptions on ultimates? These questions have three important characteristics to observe:

- They cannot be answered solely from basic DW/BI functionality—the downstream actuaries are the authorities on the question, and there is no realistic way to operationalize this into something like a call-center question, so actuaries must have an input path for these drivers of business rules and data generation.
- They are *not* stable enough to define for a one-time IT build. As soon as actuaries see, for example, the rating changes based on changed catastrophe thresholds, they will want to change them again. What-if loops frequently become a core VLCS behavior.

- They *are* stable enough to be given some basic structure that the IT system can consume. As a basic guideline: are the parameters that vary at least stable enough to be put into a spreadsheet of a few tabs with a few columns each? If so, the VLCS can be fed this information as input.

3.4.2 Execution Interface

With parameters loaded, actuaries run the system using the execution interface. The execution interface provides both self-service invocation of the VLCS and the status of all other invocations of the VLCS, both current and historical. The invocation portion of the interface is little more than a "Run!" button that starts a run of the VLCS. More important is the information provided by the interface about the other invocations of the VLCS—information that allows actuaries to be rational about the use of system resources and to be aware of previous runs for reuse. At a minimum, the interface should show the total system load, the number of processes the VLCS is currently running, their most significant parameters, who invoked them, and the approximate completion time. The goal is to allow the users to use the VLCS to its full capacity without overwhelming the system. The assumption is that power users with enough years of experience understand the nature of sharing large systems and will do so rationally if properly informed. Just in case such an assumption of rationality is a bit too optimistic, IT can put in controls discussed later. The interface should also provide historical invocation information so that previous result sets can be reused. For example, if a state was just rerated with the latest rate tables last week, that should be shown so no one else runs it again needlessly. The parameter and execution interfaces are often provided via the same application. An example of an interface providing both parameterization and execution functionality is shown in Figure 3.

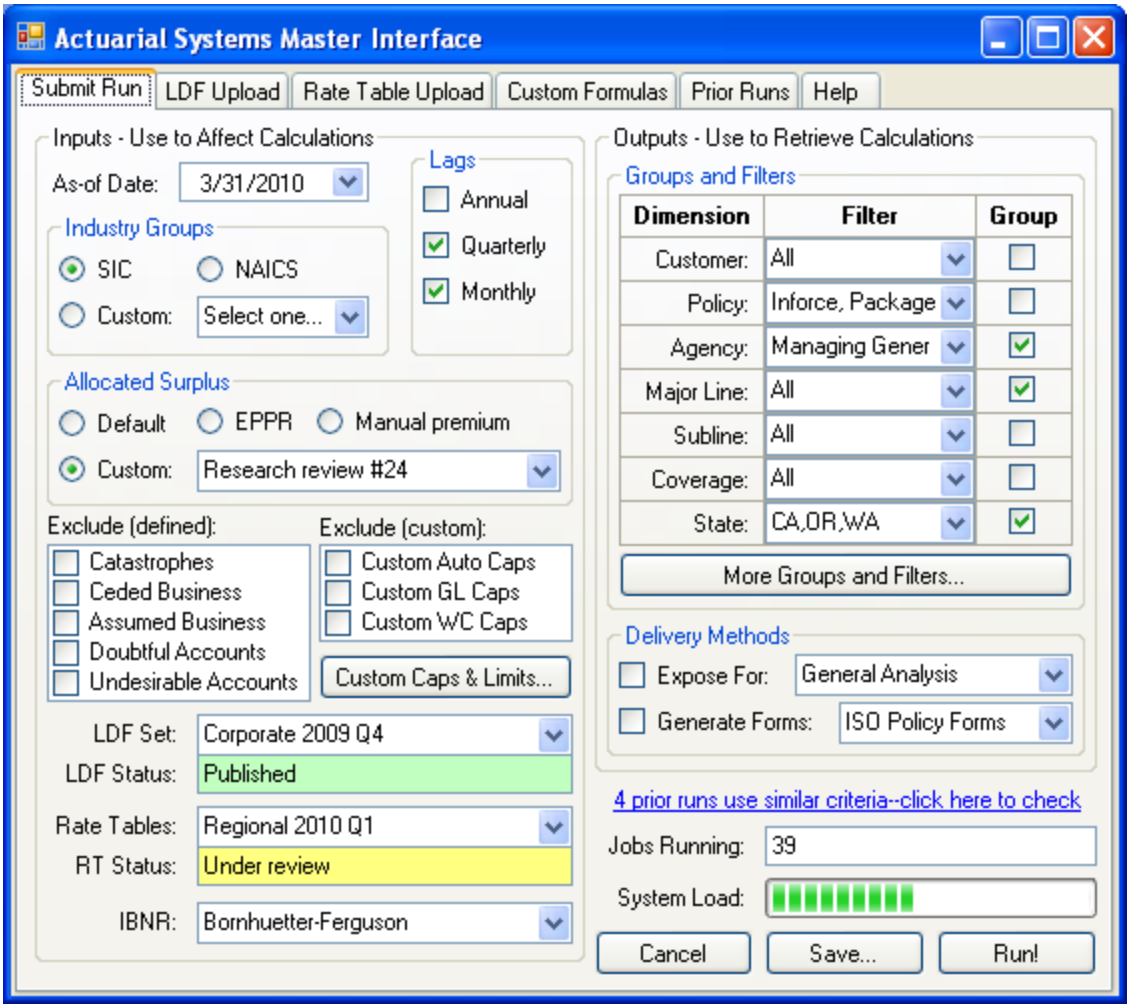


Figure 3 – An example of an interface with parameter and execution functionality.

The figure is contrived to facilitate our discussion here; an actual interface would be many screens, provide even more functionality, and be much more usable; but the figure is representative, and allows us to have a discussion that holds in reality even if the actual figure does not. Some things to observe from the figure follow. There are four major regions: items that are inputs to calculations are on the left, items that affect reporting outputs are on the top right, items that indicate system state are on the lower right, and items that have complexity unto themselves are on the tabs across the top. The calculation input items include things that will change the behavior of the system processing such as which as-of date to read from the DW; which industry grouping to use; what to include or exclude; what LDFs, rate tables, and IBNR algorithm to use; and the

certification status of items that are themselves data sets. The reporting output items include such things as how to group and filter; who to expose the output to, such as everyone for general use, internal research only, or other groups; and whether to generate pre-defined formats such as ISO forms or other corporate and industry templates. The reporting output should be somewhat limited since most general purpose reporting should go through the standard BI tools, but some customized and immediately generated predefined reports are typically useful. The system status items give a general indication of the load on the system to facilitate efficient sharing and alert the actuary if there are similar jobs worth reviewing to possibly avoid doing redundant work. The tabs provide extended interfaces for specific parameter sets of high complexity; for example, rate tables, which typically vary by state, line of business, and product, and may have many exceptions, variations, and historical versions.

3.4.3 Generated Actuarial Data

A run of the system generates actuarial data, raising the issues of storing it, certifying it, integrating it, and delivering it both to actuaries and the larger business community.

Storage initially should be within the DW. Collocation with the main DW data allows for simplified access and high-performance when compared to storage on platforms outside the DW. If the data will never undergo certification (such as with runs used entirely for what-if and similar research purposes), it should be copied to the exploration area and deleted from the DW.

Certification of the data is an organizational process involving review by key individuals or a governing body and the labeling of data according to some continuum, such as:

- Personal – the data is for one person doing what-if analysis; never to be promoted
- Uncertified – recently generated with the intent to promote
- Under review – being reviewed for promotion by the appropriate authority
- Certified – approved by the appropriate authority
- Published – sent to other organizations, often resulting in legal/archiving restrictions
- Obsolete – not used, may be a candidate for elimination or long-term archive

The certification will likely cover only a very small amount of the generated data since much of what the VLCS will be used for is what-if analysis that enhances the understanding of actuaries trying to discover something.

Integration is a standard DW practice that can be done with VLCS-generated just as it can with any data source. The integration is possible because the *process of generating* the data may require a VLCS, but *the end result* is little more than numbers that need to be drilled, sliced, diced, filtered, grouped, and treated like any other dimensional data. For example, rerated earned premium is a very difficult number to produce, and generally requires some form of a VLCS, but "rerated earned premium by agency, state, and class of vehicle, by month, trended over the last two years" is simple to provide via standard BI tools—once the rerated earned premium has been generated by the VLCS. Note carefully: collocation means putting the data in the same place physically, providing system performance and positioning it for possible integration. Integration means putting the data together logically, providing a unified business view—a proposition both more valuable and harder to do than just collocating.

Delivery of the generated data before it is certified and integrated should be through the high-power data access flow. This allows actuaries to review the data before expending organizational effort on certification, expending system effort on integration, or exposing the larger business community to immature data. Delivery of the generated data after it is certified and integrated should be through the standard BI tool flow, a simple proposition once integration has occurred.

3.5 Summary by Example: Data Mining

Data mining provides a great example of a problem space that is valuable for actuarial work[7], has product solutions with many of the characteristics of a good VLCS out-of-the-box, and yet still requires supplementation by IT to create a full VLCS solution. Data mining in its most basic form has the steps shown in Figure 4.

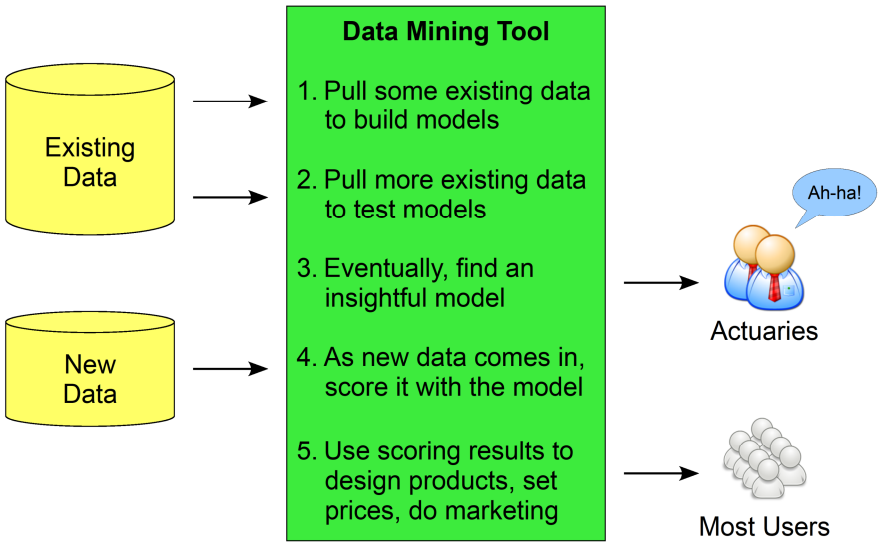


Figure 4 – Basic data mining: valuable functionality, on a small scale, managed by actuaries and business units

If the organization's goal is to have a handful of smart actuaries construct a small number of models for well-defined user communities, then Figure 4 is a cost-effective, manageable approach. The mining tool can be installed in the exploration area, the data pulled from the DW, and the deployment to users managed as part of actuarial operations. However, if the organization finds that the approach causes valuable actuaries to do low-value recurring tasks, worries the legal department due to the questionable audit trail, and creates similar forces that push from the left side of Table 1 to the right, the work can be built into a VLCS as shown in Figure 5.

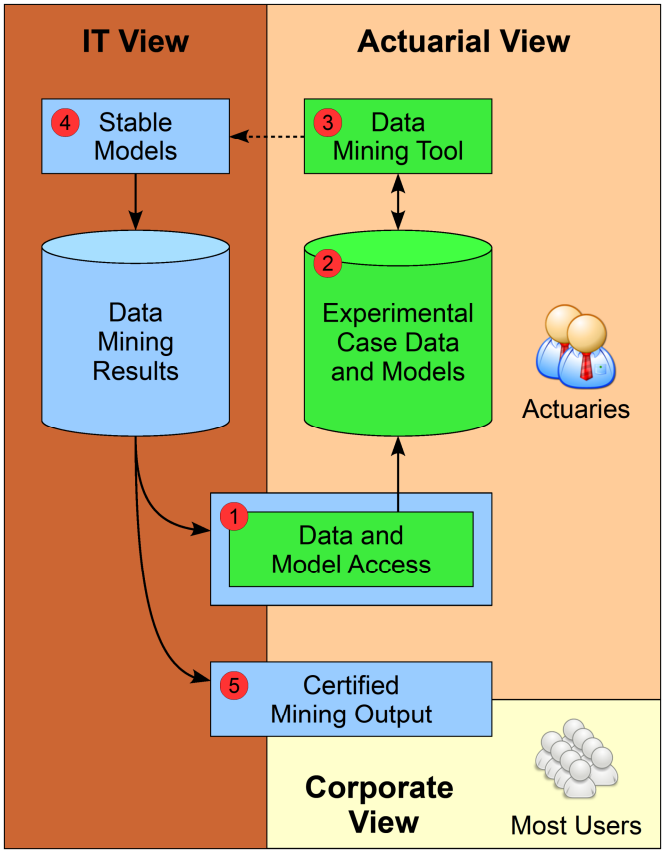


Figure 5 – VLCS data mining: enhanced functionality, on a large scale, managed as a formal corporate asset by IT

The figure reuses the top-level architecture, with the components related to a mining VLCS relabeled to show their purpose, and the components that are unchanged removed for simplicity, though they still apply. Elaborating on the five basic steps of data mining, but stating them in VLCS terms:

1) Actuaries pull data from a number of tables in the DW into the exploration area and join them in SQL into case form to be used for model learning, which is a form of calculation experiment.

2) Some of that data also tests the model. These learning and testing activities likely never become stable calculations since they will always involve creative thought.

3) Actuaries find a model with long-term usefulness to the larger organization, so the deployment step is done as part of an IT release process that includes both the code needed to get the data in case form and the model itself.

4) Each time IT has new DW data, it runs the new data against the model. No parameterization is needed for initial deployment—IT simply runs all the models on a fixed cycle.

5) IT integrates the predicted values into the standard BI tools for both actuaries and the larger user community to consume.

The organization now has a VLCS that codifies its mining work as a permanent corporate asset, but it is only a rudimentary VLCS. The organization can push the VLCS to an even higher level of value by iteratively layering on a variety of higher VLCS functionalities as need and budget allow, including:

1a) Define a variety of case tables that IT can generate as part of the VLCS and allow the selection of which case form to use in the parameter interface.

2a) Have the VLCS test the models as part of its run and store the test scores in separate tables that can be used when judging the output for certification.

3a) Define multiple models and predicted values that mine a wider array of business needs and allow selection in the parameter interface.

3b) Ensure both actuaries and IT use a common tool set so that mining algorithms move from the exploration area to the DW without modification rather than being recoded from the actuaries' language into the IT language.

3c) Expose parts of the model deployment process so actuaries do not have to wait for IT release cycles—that is, parameterize and self-service-enable *the deployment process itself*—a very advanced interpretation of VLCS principles.

4a) Add the ability to specify which mining algorithms to run and the parameters to those algorithms, such as confidence levels, windsorizing factors, decision tree depths, and the hundreds of other parameters that data mining allows.

4b) Add the ability to execute the mining algorithms on-demand after the parameters have been adjusted.

5a) Add status codes so the user community knows what is certified and what is experimental.

4. IT VIEW

Making the actuarial view work efficiently requires critical design and implementation decisions by IT behind the scenes. These decisions are discussed here in a way that makes the direction clear but leaves the details to each particular IT shop since every shop has its own unique tools and processes and the VLCS architecture is agnostic to them.

4.1 Operational Systems

Policy writing systems, claims payment systems, and billing systems are examples of operational systems. A large insurance company will typically have dozens if not hundreds of them. Known more formally as on-line transaction processing (OLTP) systems, such systems generate *operational* data which must be converted into HIGSO form and dimensionalized into *analytical* data to provide maximum analytical efficiency. Such conversion is complex and time-consuming work that a DW is specifically positioned to handle, so in the long-term, the goal should be to move all operational data into the DW and deliver it in dimensionalized HIGSO form. In the short-term, if the need to move a particular operational system's data through the DW is unclear, or a large-scale DW project cannot be funded or will take too long, the data should be placed in the exploration area on an interim basis for R&D purposes. Once established, this exploration use of operational data has the tremendous value of allowing preliminary analysis of the data without the overhead of a major DW project, and of allowing actuaries to learn about the data so they can later specify more precisely how they need it handled by the DW in dimensionalized HIGSO form. Such use must avoid long-term retention of the operational data since this will cause the environment to grow to a size and complexity not easily managed by non-IT areas. As with all data in the exploration area, institutional reporting on this data must be avoided.

Two approaches can be taken to get operational data into the exploration area. The aggressive approach is to point the high-power data access tools at the operational systems and read the data directly into the exploration area. This has the benefit of being easy to set up, but puts a processing burden on the actuaries and challenges operational system access policies, which typically restrict access to well-defined uses by a small number of administrative authorities. A more conservative approach is to employ IT to move the data into the DW and expose it for use largely as-is rather than converting it to dimensionalized HIGSO form. This has the benefit of leveraging the DW's existing tools, processes, and access policies related to acquiring operational data, but does require accepting the time and cost of a small DW project and giving up the tremendous power of

dimensionalized HIGSO form. Such DW work is not throw-away in the long-term (the way the more aggressive approach is), yet it is relatively cost-effective—loading sources into a DW is a small fraction of total DW costs; it is the integration, subject-orientation, and dimensionalization that are expensive. By using the more conservative approach, a balance is struck between moving quickly and positioning for the long-term.

4.2 Data Warehouse

Using a DW to support exploration requires IT to expose the internal business rules of the DW more than usual, using a DW to support a VLCS requires IT to know the proper location for a VLCS in the DW flow.

The exposure of DW business rules is something that the larger user community often wants to avoid. They assume that the normal data warehousing process gathered the appropriate business rules, built them into the systems, is verifying the business rules as part of routine operations, and is certifying the data accordingly—meeting these assumptions is a core DW value, thus a safe assumption. Actuaries engaging in exploration cannot operate on such black-box assumptions. Exploration necessarily looks for new patterns and connections, requiring the current patterns and connections to be understood. How data is grouped, which source "wins" when more than one is providing different data for the same field, how different code sets map to each other, the many variances based on which state of the U.S. is in question—these business-rule-driven manipulations of data in the DW must be published by IT in an easily understood format that is actively maintained.

The proper location for the VLCS in the DW is the location that balances the need for stable data coming in and flexible control going out. There are competing schools of thought about the "data staging area"[8] or "corporate information factory"[9] work that ultimately generates DW output, but for VLCS purposes, the three blocks shown in Figure 6 suffice. "Staging" takes in operational data and transforms it into HIGSO form. "Atomic" holds the HIGSO data, in a manner optimized for system use, but not efficient for BI tool or human use. "Delivery" restructures the HIGSO data into dimensional form so that users and BI tools can efficiently use it for analytics. These areas constitute the basic DW.

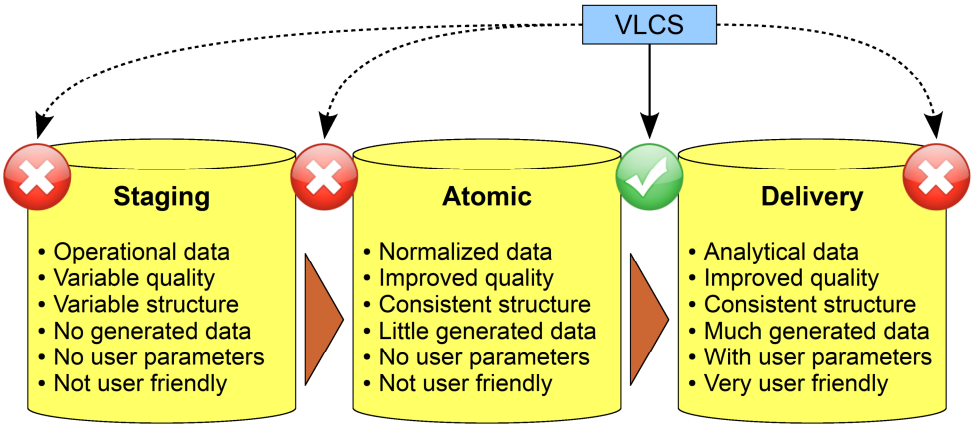


Figure 6 – Data warehouse layers with the proper location for the VLCS

The proper location for the VLCS is after atomic but before delivery. Locating the VLCS before staging is impossible because the data is not yet cleaned up and properly structured, and the staging intake is generally too slow and too controlled for the dynamic, self-service requirements of the VLCS. Locating it before atomic is possible, but this causes the VLCS calculations to take place at the same time as quality and structure improvements, resulting in a level of complexity that is difficult to manage; and atomic work is also still too slow and controlled for VLCS requirements. Locating it after delivery would require doing the calculations against data in dimensional form. While the dimensional form is powerful for BI tools and human thought, it is poorly suited for the computer-processing nature needed for the VLCS. Locating the VLCS between atomic and delivery is ideal—the source data is cleaned up and structured for efficient processing, the parameter loading can be kept dynamic, and the generated data can be efficiently stored and integrated into the dimensional structures. This location is the point of optimal balance between the stable data on the left and the flexibility needed on the right.

4.3 Parameter Interface

The parameter interface allows actuaries to express complex input with enough structure to allow parameter consumption by computer code. There are dozens of solution patterns, but they generally fall into the four categories discussed here.

4.3.1 Parameters by Screens

Data entry screens work well for needs that are well defined and seldom change, such as entering name, address, and other personal information in a Web page. They are rather poor for such things

as rate tables, loss development factors, arbitrary value banding, and other parameter sets that by their nature have lots of categories, ranges, factors, comparative logic, and other points of variance. Proper use of data entry screens for VLCS design tends to occur when the *categories* of parameters are stable but the *values* within those categories need to be arbitrarily selected. For example, if a report is always accessed by coverage, deductibles, and limits, these can each get a component on the interface, but the values in these components would be populated dynamically based either on the data itself, or on reference tables uploaded using one of the more robust techniques below. Screens may also be useful if the formulas are sufficiently simple. In such cases, a basic formula entry screen that looks something like a robust calculator can be written. The major weakness of screens is that they require a comparatively heavy IT engagement, both up front and when things change. The major strength of screens is that they can be constrained to prevent user error, thus allowing the parameterized work to be pushed to junior skill levels.

4.3.2 Parameters by Spreadsheets

Spreadsheets are often the optimal approach for passing parameters and can produce extremely robust parameterization[10]. Formulas and cell referencing provide high expressive power within the spreadsheet itself. The sheet/column/row/cell structure makes spreadsheets unambiguously navigable in computer code. Naming various sections within the spreadsheet makes the computer code human-readable. The ubiquity of code libraries that can process spreadsheets allows IT to choose among many programming languages. That nearly every IT professional and actuary has worked with spreadsheets throughout their career makes spreadsheets a common tool with which everyone is immediately comfortable.

To be done effectively, actuaries and IT must agree on the structure of the spreadsheet, where to upload it, and what to do when it is uploaded. The structure is the most important part. For example, if the computer code expects the names of coverages to be in column C and the limits on the coverages in column D, they had better be there. If the code will assume the first blank cell in a column to be the end of the list, there better be no embedded empty cells, etc. Actuaries must be able to specify a very robust structure for the spreadsheet, and once it is done, they must stick to it. IT should have the VLCS scan the spreadsheet's structure for basic compliance to the agreed format, but at some level, there is a garbage-in/garbage-out (GIGO) risk that is unavoidable and must be addressed through caution by actuaries using the spreadsheet and by IT staff monitoring the processes that consume the spreadsheets for anomalous behavior. Naming of cells and ranges should be used extensively as this greatly enhances human readability and system navigability.

Spreadsheets meeting these requirements are placed on designated locations such as shared drives where an IT process is watching for its arrival. Seeing the parameter file, the process may invoke the appropriate back-end activity including such things as running calculations, reporting run status, and notifying the user when milestones are complete.

4.3.3 Parameters by SQL

Eventually spreadsheets reach a limit because they are not designed for general-purpose data handling, at which point SQL can be used (pre-relational technologies, such as hierarchal databases, are not addressed here, but the concepts are the same). The SQL is written by actuaries based on the tables coming into that point in the processing, and the tables going out. This use of SQL is not interactive the way it is when querying data, but instead is done in advance and uploaded, even if "in advance" is immediately before running the calculations. The interface itself can be in two forms. One is truly free-formed where actuaries can put in literally any SQL needed to achieve their desired outcome. The other is constrained to the major clauses with some filtering to prevent issues. For example, the screen may have an area to list the columns to put in the SELECT clauses, the tables to put in the FROM clause, and the conditions to put in the WHERE clause. The code then puts these together and scans them for undesirable things such as cross joins, non-key joins, and other SQL constructs that are hard on the system and/or logically invalid. As with spreadsheets, there must be agreement about the GIGO risk and the appropriate mitigations should be in place.

4.3.4 Parameters by Code Modules

Anything that can be done in a relational database can be specified and executed in SQL, but some desired outcomes are terribly difficult to do in pure SQL and are much more readily accomplished using a combination of SQL and general purpose programming languages. Such languages come in two major forms from the view of actuaries. The lesser form is macros in office automation tools such as spreadsheets. Since using spreadsheets for parameterization is likely already part of the plan (per the previous section) it is also possible for actuaries to add macros that execute very advanced logic. These macros would be named and then called at run-time by the VLCS. The greater form is major statistical tools such as R, SAS, or SPSS. For these and similar general-purpose tools, actuaries will write the code modules to do whatever logic is needed. Many actuaries likely learned to do such coding in college, and the more adventurous are likely to now do it somewhat routinely in the exploration area. Such code modules, instead of just running on actuaries' local accounts, are enhanced and deployed by IT in a way that makes the code callable by the VLCS. The

enhancements are of a technical nature and do not change the core business logic, which remains the responsibility of the actuaries. Enhancements may include multithreading, error handling and recovery, monitoring and metrics, and other well-defined yet business-agnostic characteristics of IT systems. Callability is facilitated by enterprise integration middleware based on such things as REST, WS-* protocols, and message queuing technologies. Again, GIGO applies.

4.4 Loaded Parameters

The primary responsibility of IT once the parameters have been loaded is to stamp them with user, version, date, and status attributes and hold them for historical reference. Actuaries must be able to go back in time and see what parameters were used and correlate them to the output produced. If the output is eventually promoted into the main data warehouse for use in standard BI tools, the organization must be able to audit the parameters as part of the full lineage of all data under compliance.

4.5 Execution Interface

The execution interface provides self-service invocation of the VLCS on the assumption that actuaries will be rational in the use of the system if properly informed, but as a safeguard, IT must put in back-end components that put constraints on the maximum amount of power actuaries can consume. Database and operating system workload management tools should be used (as discussed shortly), but these generally need to be supplemented because a well-designed VLCS can spawn any number of parallel processes, thus overwhelming any generalized workload management tool. To prevent this, execution should use a customized back-end process that caps and queues submitted jobs and the number of processes per job; for example, a custom-coded UNIX daemon that listens on a TCP/IP port for invocation requests, forking the appropriate process if the cap is not reached, but queuing the request if the cap has already been reached. Such behaviors by the daemon should be table-driven, using tables common to the daemon and the execution interface so that the caps, queue, and current running state are easily read by the execution interface for presentation to the actuaries. Such tables should retain the information indefinitely, providing the historical view also desirable on the execution interface.

4.6 Stable Calculations

Stable calculations are those that are no longer dynamic enough to need the knowledgeable intervention of actuaries and are being done repeatedly enough to match the strengths of IT, thus

making it logical to implement them as formalized business rules in an IT-supported system. Separating these stable calculations from those that still require flexibility is one of the greatest challenges of VLCS design. In the data mining example earlier, the separation was fairly clear—the very advanced mining algorithms are stable across the industry and hidden within the mining tools, but parameters to those algorithms are externalized for the user to adjust. A much harder challenge occurs when the core calculations are to be developed in-house, requiring actuaries and IT to work together to determine what is flexible, what is stable, and how to separate them. The challenge is met by modularizing the steps in the process, clearly quantifying the information going into and out of each step, and outlining the basic logic of each step. An extremely simplified example of the modularized steps and critical information for a loss development process might look like Table 2. The design and build of the VLCS proceeds around this structure.

Modularized Step	Type	Critical Information
Acquire operational data	Stable	Inputs: operational data from around the enterprise. Outputs: operational data in the DW in HIGSO form. Logic: HIGSO processing. Rationale: Normal DW work.
Convert data to actuarial codes	Stable	Inputs: DW HIGSO data. Outputs: HIGSO data with codes that actuarial uses. Logic: Standard data transformation. Rationale: Actuarial needs data coded to their rules E.g. loss bin A = $1-$499, B = $500-$1,249, …
Capture base balancing numbers	Stable	Inputs: DW HIGSO data. Outputs: Aggregated counts and sums. Logic: Counting and summing. Rationale: Capture numbers at an aggregate level now so they can be used for balancing later.
Load loss development factors (LDFs)	Flexible	Inputs: Actuarial spreadsheet with LDFs. Outputs: LDFs in the database, stamped with user, data, and version information. Logic: Basic upload. Rationale: LDFs change based on the knowledge of actuaries and must be under their control.
Load catastrophe identifiers	Flexible	Inputs: Actuarial spreadsheet with date, geographic, and other identifiers of events to be considered catastrophic. Outputs: Identifiers in the database, stamped as above. Logic: Basic upload. Rationale: Actuarial is charged by the organization with classifying catastrophes based on data characteristics.
Develop & store losses in raw form	Stable	Inputs: DW HIGSO data and all uploaded parameters (LDFs and cat. IDs in this example). Outputs: Developed losses in the database. Logic: Calculations based on the standard rules of developing losses. Rationale: Well-defined algorithm for which all inputs have been received in the DW.
Balance against base numbers	Stable	Inputs: Developed losses and base balancing numbers. Outputs: Stratified accuracy percentages. Logic: Basic math. Rationale: Manipulated data often cannot be balanced at a fine grain, but in aggregate, basic "sanity checks" can be made.
Filter categories in/out as needed	Flexible	Inputs: All data from above and a parameter screen. Outputs: Run parameters in the database. Logic: Capture user preference for the run they are requesting. Rationale: Large amounts of loss development data are in the database, but the user needs only a targeted amount. E.g. non-catastrophic losses in Texas for the previous 12 months for auto coverages grouped by coverage limits.
Connect developed losses to BI tools	Stable	Inputs: Developed losses. Outputs: Developed losses presented side-by-side with main DW data in standard BI tools. Logic: Integration logic that matches the developed losses to their original HIGSO subject areas. Rationale: The larger organization may have an interest in actuarial's developed loss work—such as product or regional managers who control what kinds of business to go after.
Generate reports in triangle form	Stable	Inputs: Developed losses and user parameters. Outputs: A triangle-structured report holding just the data the user specified in the parameter screen. Logic: Filtering logic embedded in the SELECT and WHERE clauses used to retrieve the developed loss data; a bit of creative transposition coding to get it to look like a loss triangle. Rationale: Standard format for presenting developed losses.

Table 2 – Modularized steps of loss development and their inputs, outputs, logic, and rationale

4.7 High-Power Data Access & Overall Performance

The *high-power* aspect of data access is determined *far* more from the back-end tooling than by the software the actuaries see on their desktops. As discussed, the only real requirement for high-power data access from the actuarial view is that the tools allow them to run arbitrarily complex SQL. The challenge for IT is to ensure that when such queries get submitted, they run fast. This need for speed and the solutions about to be discussed also apply to some to degree to other contexts, such as making the exploration area and standard BI tools run fast, so this section will address topics that IT should keep in mind whenever they need a fast platform for any user community. The need for speed is satisfied by having sufficient hardware, utilizing it with partitioning and parallelism, connecting it together with high-bandwidth networks, and keeping it all under control with workload management tools.

4.7.1 Hardware Capacity

At some level, computing power comes down to the old racing adage, "speed costs money, how fast do you want to go?" No design, implementation, usage policy or other non-hardware factor will make up for a lack of the raw power that comes from sufficient hardware—but *non-hardware* factors can still produce bad results *despite available hardware*, such as running serially instead of in parallel, not partitioning, hopping repeatedly across the network instead of clustering processing steps on one server, etc. Thus a goal in good VLCS design is to do nothing that limits full and proper utilization of hardware, but sufficient hardware must also exist.

IT must then ensure that the hardware is configured in a manner known as *balanced*. A balanced hardware configuration is one that ensures that all hardware sub-components are operating at the same speed. Disks, disk controllers, motherboards, HBA cards, NIC cards, interconnect switches—these are the small hardware pieces that add up to the total hardware solution. Don't worry about what they all are—only know this: just as a chain is only as strong as its weakest link, the platform hardware is only as fast as its slowest subcomponent. Balancing ensures that all subcomponents are at the same speed. Balancing a large platform is so hard that the IT team within the organization often should not attempt it, instead outsourcing it to the vendor of the platform itself. The vendor is far more likely to be able to balance their platform than any insurance company's internal IT staff. Often such a service is provided for free or at low cost as part of the purchase of the platform, but IT should be sure to write it into the contract specifically.

4.7.2 Partitioning and Parallelism

The *very large* in the VLCS comes in two forms: very large data and very large processing. Very large data requires partitioned data; very large processing requires parallel processing. Partitioned data is data stored on disk in a way that allows the system to retrieve only the data that is used in any particular query. For example, a database may have 10 years of data, but the actuaries are only analyzing the last six months, so only the last six months are retrieved by the database, thus getting a 20x performance boost. Parallel processing is processing done using many processes all at once, each doing a small slice of the same big task. For example, the last six months can be done with six processes each going after just one month, thus getting a 6x performance boost. Overall, the example is 120x faster than one process on all the data. Factors consistently above 100x are not unreasonable on a VLCS built with proper partitioning and parallelism (P&P).

P&P are critical success factors to the VLCS[11]. They are achieved by using tools that can *potentially* handle the very large scale, and then implementing a design that *actually* handles the very large scale. During the tool selection process, vendors must show that every aspect of their suite supports P&P. During system design, actuarial must explain to IT how the data is used (which should already be happening per the same need when separating stability and flexibility). During implementation work, IT must build specific constructs that facilitate P&P. Actuarial must be patient when IT is building P&P since the work will consume time yet not be adding any new business rules, only system performance and scalability.

4.7.3 Networking

Network bandwidth must be high because actuaries will need large result sets or even large bodies of data to be pulled into their exploration area. Bandwidth should be 1-gigabit or 10-gigabit Ethernet to the desktop and 10-gigabit Ethernet or InfiniBand between the DW servers and exploration area servers. Better yet, put the exploration area on the DW server and encourage actuaries to pick tools that run directly on the DW platform, thus eliminating the network hops.

4.7.4 Workload Management

High-power data access, calculation experiments, the execution interface, the freedom of the exploration area, the many queries from standard BI tools being run by a large corporate community, and the repeated admonition to collocate as much as possible for maximum efficiency—these result in a serious risk of overwhelming the platform. Even with sufficient hardware overall, there is still a risk of overwhelming the platform at key times such as right after the

DW publishes the monthly refresh, right before lunch, at the end of the business day, when the boss e-mails a question to the department and a dozen people try to generate the answer independently, and other times when human behavior tends to cluster.

Keeping the platform from getting overwhelmed is the purpose of workload management (WM) tools. As noted, the execution interface benefits from having its own custom-written WM behavior, but in addition to this, WM tools in the operating system (OS) or database management system (DBMS) should also be utilized so that the full collection of processes competing for system resources can be managed holistically. Most OSs and DBMSs have WM tools, and they typically provide the ability to allocate system power based on both statically and dynamically definable characteristics. Static characteristics include such things as username, machine name, or the connection protocol. Dynamic characteristics include such things as how much power a job is consuming, how long a job has been running, or how many jobs are on the system.

IT must implement WM tools as part of platform construction, allocate permanent staff to ensuring WM operates efficiently, and provide easy-to-understand mechanisms for all users of the shared platform to see what is happening on the platform. Actuaries must be realistic about having to share power among the various forms of processing, especially since theirs are the process most likely to consume the most power. As with the execution interface, the hope is that smart users who are properly informed will be realistic about overall organizational priorities and their share of the power.

4.8 Standard BI Tools

The same techniques that allow the DW to connect dimensionalized HIGSO data to standard BI tools also allow generated actuarial data to be connected to standard BI tools. Kimball provides an example[12] so robust that it cannot be improved upon here, so the interested reader is encouraged to review it. In short, the example shows how to connect one of the most advanced forms of calculation work—data mining—to one of the more complex dimensional constructs—aggregated attributes as facts[13]. With such an advanced example established, the feasibility follows for connecting simpler calculations, such as earned premium, to basic dimensional constructs, such as products, geography, and customer.

5. RESULTS AND DISCUSSION

5.1 A Complete and Complimentary Solution Suite

The set of solutions provided here solve any conceivable analytical data problem in a mutually interconnected manner. As a result, there is no analytical data challenge that cannot be handled. *Simple and stable* analytical needs are handled through traditional DW/BI solutions. *Complex and changing* analytical needs are handled through the exploration area. *Complex and stable* analytical needs are handled through VLCS solutions. *Simple and changing* analytical needs are typically done by desktop tools, which were not discussed here since it is a straightforward solution space.

5.2 Reference and Reality

Nearly every aspect of this paper is based on an actual experience of the author—I have worked on data warehouses and exploration areas with dozens of terabytes, calculation engines that could consume every CPU available, interfaces with so many parameters they looked like a DJ's mixing board, rate table spreadsheets with thousands of active cells, process steps that jumped outside the firewall to a third-party vendor and back, an execution interface back-end so advanced it almost became its own little operating system, output that went to the CEO, output that became published rates with state insurance offices, and lots of what-if output that went nowhere—but I have never been on any *single* system that implemented every aspect of the VLCS as detailed here. Likewise, any particular VLCS in any real-world situation will likely not have every aspect. In part this is due to actuarial not needing every aspect and due to time and funding realities when building systems. Rather than being a monolithic goal, the architecture and approach provided here should be used as a broad reference and a large collection of ideas to draw from then narrow down to the needs of a particular situation.

5.3 A Targeted and Challenging Undertaking

The VLCS is a niche solution not to be undertaken lightly. If possible, solve the business problem through some lesser means. Can static reports solve the problem? Can general-purpose slice-and-dice OLAP meet the need? Is there a vendor tool that we can just buy? Can it be done directly from the exploration area, even though audit and legal get nervous about end-user reporting? Odds are, the VLCS will take longer to build and cost more than most people think at the start. Be cautious. That said, do not attempt to meet a VLCS need without a VLCS solution. The VLCS delivers extremely advanced analytical power better than any other pattern. If an organization

is confident that the effort is worth the undertaking, nothing can replace a well-built, highly-customized VLCS.

5.4 Governance and Maintenance Processes

Even the most stable business rules in a VLCS may change from time to time, and new calculations will stabilize in the exploration area and need promoting. These changes should be handled by a small governance team with both actuarial and IT membership. Determining the course of action requires a more open dialog than with traditional IT systems. Traditionally, the business states a need, IT estimates a cost, and the business makes a decision to execute or not. With a VLCS, there must be more of a back-and-forth discussion that explores where the functionality truly belongs. If everyone is comfortable with the stabilization and parameterization that can be provided, move it to IT. If everyone is comfortable with the localized control and do-it-yourself overhead, leave it in actuarial.

5.5 Corporate Standards and the Center of Excellence

The organization has a powerful opportunity for efficiency if it encourages actuarial to use official IT tools well before a promotion discussion occurs. If actuaries use IT servers for their high-power exploration work, IT languages for the code they write, and IT vendor packages for their major solution suites, then moving these over to IT if they prove to be stable will be vastly easier than if IT has to rewrite logic, approve new tools, and staff new skill sets. In return, IT must expand its suite of tools to include those preferred by actuarial and native to the nature of their work. IT must standardize and publish the proper use of those tools, with actuarial being a significant contributor to the standards and IT adding value primarily through formalization and stewardship. IT must staff a small group of technical experts, typically known as a "center of excellence," whose job is to actively support actuaries in the use of tools and standards, as well as perform the monitoring and maintenance of the exploration area previously discussed.

5.6 Build Versus Buy

As noted at the start, the generic nature of vendor tools makes them poor candidates for deepening the competitive advantage of differentiating work—a core effect of the VLCS. But a number of tools in the insurance industry follow the VLCS pattern, whether they call it that or not, and buying a VLCS may make more sense than building one if the packaged solution matches the business need. Verify that the product's overall architecture matches that of the VLCS architecture

in Figure 2, and pay particular attention to the flexibility of the parameter interface, parallelism, partitioning, high-power data access, connecting data to the DW, and integrating data into standard BI tools. If the product seems favorable, give more weight than usual to buying—building a VLCS is hard, and if a vendor has truly done it, buy it from them. If there are any clear faults in the product, give more weight than usual to building—getting locked into a limited VLCS destroys creative knowledge work.

5.7 Higher-End Work with Lower-End Skills

It was noted previously that VLCSs tend to apply to the upper end of actuarial work continuum. Notice, however, that if the calculations are codified into an IT system, the parameter and execution interfaces are sufficiently simple, and the data is hooked into standard BI tools, it is possible to assign more junior actuaries to the use of the VLCS, having them draw in more senior actuaries on an exception basis only. This ability to commoditize advanced work down to a more junior level is one of the most important value-adds of a VLCS because it allows the most senior actuaries to continually drive to the top of the actuarial work continuum. Very often, the core of the business case for a VLCS comes down to the hard dollar cost of building an advanced IT system versus the soft opportunity cost of not using the best actuaries on the hardest problem because they are bogged down in work that should be getting commoditized.

5.8 Agile Software Development

Systems are built using some combination of two basic approaches: waterfall and agile[14]. Waterfall is a sequential process that requires actuarial to create detailed specifications. IT builds to the specifications. Changes along the way are viewed negatively and rigidly controlled. Agile is an iterative process that requires actuarial to specify a small increment of what it needs[1]. IT builds the increment. Both learn from the increment, embrace changes discovered, build the next increment, and repeat until the system is complete. VLCS development should use the most agile approach possible, and it tends to lend itself efficiently to doing so. For example, notice how the data mining example earlier had a basic level then a more advanced level, and how the more advanced level had line items that could be discretely built and delivered to actuarial. It was presented in this form precisely to show how it could be iteratively delivered using agile rather than having to be a massive commitment under waterfall.

[1] Agile is a set of principles. Several methodologies implement these principles. Among the better ones is Scrum[15].

6. CONCLUSION

The VLCS is the most advanced analytical system an organization can build, with a risk/reward profile to match—it will push IT staff to the limits of their skills and IT systems to the limits of their power, and it will require actuaries to have truly unique and useful ideas that the VLCS can help them explore and deliver. But it will also provide a level of value not achievable through more humble means, including optimization of the actuaries' time, maximum reasonable flexibility, legal compliance and auditing, tolerance of staff turnover, the power of a large-scale IT system, deep competitive advantage, and maybe even a little fun—there's just something cool about having direct control over a powerful system running complex calculations at top speed against massive amounts of data!

Acknowledgments

The author is grateful to his many technical peers and business partners who have helped him understand how to engineer a VLCS and utilize it for maximum business value. Special thanks to Doug Anderson who first taught me the pattern, to Ian Penman whose engineering talent showed me the power of a VLCS in its most advanced form, and to Bob Allen who patiently explained the nuances of actuarial logic in terms that even an IT guy like me can understand.

7. REFERENCES

[1] Bukhbinder, George, Michael Krumenaker, and Abraham Phillips, "Insurance Industry Decision Support: Data Marts, OLAP, and Predictive Analytics," *Casualty Actuarial Society Forum*, Winter **2005**.
[2] Inmon, William H., "Building the Data Warehouse," 4th Ed., John Wiley & Sons Inc, **2005**, pp. 29-33.
[3] Kimball, Ralph and Margy Ross, "The Data Warehouse Toolkit: the Complete Guide to Dimensional Modeling," 2nd Ed., John Wiley & Sons Inc, **2002**, pp. 16-24.
[4] Inmon, William H., Robert H. Terdeman, and Claudia Imhoff, "Exploration Warehousing: Turning Business Information into Business Opportunity," John Wiley & Sons Inc, **2000**.
[5] Hewlett Packard, "Rationalizing R&D Sandbox Environments," **2009**.
[6] Brooks, Frederick P., "The Mythical Man-Month," Anniversary Edition, Addison-Wesley, **1995**, pp. 4-6.
[7] Guo, Lijia, "Applying Data Mining Techniques in Property/Casualty Insurance," *Casualty Actuarial Society Forum*, Winter **2003**.
[8] Kimball and Ross, pp 8-10.
[9] Inmon, pp. 402-409.
[10] Krumenaker, Michael, and Jit Bhattacharya, "User Implementation and Revision of Business Rules Without Hard Coding: Macro-Generated SAS® Code," *Proceedings of the 16th Annual Northeast SAS User Group Conference*, **2003**.
[11] Kenealy, Bill, "Insurers Attacking Problems in Parallel," Insurance Networking News, May **2009**.
[12] Kimball, Ralph and Joe Caserta, "The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data," John Wiley & Sons Inc, **2004**, pp. 204-206.
[13] Kimball and Ross, p. 152.
[14] Poppendieck, Mary and Tom Poppendieck, "Lean Software Development: An Agile Toolkit," Addison-Wesley, **2003**, pp 24-25.
[15] Schwaber, Ken, "Agile Project Management with Scrum," Microsoft Press, **2004**.

Definitions

Aggregated	Data that is not granular. For example, if the lowest grain of premium is the earned premium calculated on a daily basis, earned premium at a monthly level is an aggregated form of earned premium.
BI	Business intelligence: processes and tools that enable the business to make informed decisions based on available data.
Bin	A range of data treated as a unit. For example: drivers 18 to 25 years old are in the "Youthful Driver" bin; quartiled information has four bins. Also known as value banding.
Corporate information factory	A conceptual model for how a data warehouse should function. Defined by Bill Inmon, et al. The model advocates treating the intake, processing, and delivery of information in a DW as a factory-like process. It places primary importance on the quality of the data and its structure in the atomic area shown in Figure 6. Competes with data staging area.
CPU	Central processing unit: the computer chip inside every computer that is the primary area of processing. DW servers will have dozens if not hundreds of them.
Cube	An alternative term for a dimensionally modeled data structure. The term "cube" comes from the visual representation showing numerical values in the cells of a cube whose sides are defined by the many dimensions used for drilling around the cells. Visualization beyond three dimensions is difficult. Having more than three dimensions is common for any non-trivial analytical work and is known as a hypercube.
Daemon	A computer process running on a server. Usually runs at all times when the server is running, waiting for requests to come to it through some type of network communication.
Data Mining	The process of discovering previously unknown patterns in large data sets using automated tools guided by users with a reasonable knowledge of the data being mined.
Data staging area	A conceptual model for how a data warehouse prepares its data for delivery. Defined by Ralph Kimball, et al. The model advocates analogizing the intake and processing of information in a DW to a kitchen in a restaurant—it must be clean, efficient, and focused on the real objective—efficient usability in the delivery area shown in Figure 6. Competes with corporate information factory.
Dimensional	A way of structuring, or modeling, data. All data is classified as either a measure or an attribute by which to group measures. For example, "earned premium" is a measure that can be grouped "by state." Attributes are put together in logical groups called dimensions—from which the technique draws its name. For example, all attributes related to vehicles are together in the "vehicle" dimension.
DSS	Decision support systems: systems that provide a concise view of data for business decision makers.
DW	Data warehouse: any large collection of data in HIGSO form. "Large" is relative to the organization and its needs and cannot be precisely quantified generically.
Ethernet	A network communication infrastructure. It is a way of connecting computers so they can communicate. Generally used in 1-gigabit and 10-gigabit speeds. Think of the former as "slow" and the latter as "fast." Competes with InfiniBand.
GIGO	Garbage in/garbage out: An informal term used to summarize a class of problems caused by giving bad input to a system, resulting in bad output. The implication is that the fault lies with the input, not the system. While some garbage-in can be discovered and handled by a system, other input cannot. Garbage input not discoverable by the system is generally the type of input meant when using the GIGO term.
Granular	The decomposition level of data. In general, data should be stored in a data warehouse in the most granular form reasonable. For example, the vehicle dimension should be at the VIN granularity level, not just the year, make, and model granularity level. See also, aggregated.

HIGSO	Historical, integrated, granular, subject-oriented: the primary characteristics of good data warehouse data. Although Bill Inmon provides the elements of this definition, the HIGSO acronym was coined by this paper for simplicity of reference. To prevent acronym-overload, "integrated" may be used in conversation since it is the dominate consideration of the four characteristics.
InfiniBand	A network communication infrastructure. It is a way of connecting computers so they can communicate. Competes with Ethernet. Think of it as "very fast" when comparing to Ethernet.
Integrated	Data that is composed from more than one source, converging the sources on the same definition. For example: a Web site captures marital status as M, S, D, W, O. A call center system captures it as 0, 1, 2, 3, 4. The warehouse integrates and transforms these into Married, Single, Divorced, Widowed, Other.
IT	Information technology: components such as computers, systems, and technologies as used in business contexts; name of the department whose primary responsibility is the implementation and maintenance of such components.
Join	A way of connecting data with other data. It can be thought of as a side-by-side connection, such as joining driver, vehicle, and loss information to see how much youthful drivers in red cars are costing the company.
Message queuing	A design pattern that allows computers to talk to each other. One computer will send messages to another, and the messages will queue-up until the receiving computer can get to them. By loose analogy: an e-mail inbox is a message queue system for human beings.
MIS	Management information system: a system that provides a very simplified view of information to more senior-level managers.
OLAP	On-line analytical processing: systems that facilitate data analysis on large amounts of data with fast response time.
OLTP	On-line transaction processing: operational systems that bring in data one transaction at a time
OS	Operating system: The software that allows computer hardware to function. Microsoft Windows is the operating system familiar to most people.
Pivot table	A simplified dimensional presentation of data. A pivot table provides a very useful subset of the operations associated with dimensionally modeled data. It is generally not as full-featured as a standard BI tool, but it is easier to create, for example in Microsoft Excel with just a few minutes of effort.
Predictive Analytics	A form of analysis stronger than the usual drilling, slicing, dicing, trending, and statistical analysis done on dimensional data, but weaker than data mining; predictive analytics falls somewhere in the middle. The dimensionally presented HIGSO data is utilized, but the user is given the opportunity to apply basic data mining with such restrictions as using only mining algorithms that do not require specialized data preparation, having only default parameters used, applying only one mining algorithm rather than a chain of them, etc.
REST	Representational state transfer: a mechanism of computer-to-computer communication both advocated and criticized for its simplicity. Competes with WS-*.
Server	A computer that does work for other computers. Typically, a high-power computer in an IT data center designed to do work other than that directly related to user interaction. Often runs the UNIX operating system, though Windows can also run on servers.
SLA	Service level agreement: an agreement between IT and a business area that quantifies the operational characteristics of a system, such as how often it runs and how long it takes to complete when it does.

SQL	Structured query language: the language used to access data on nearly all modern database systems in a business environment. Older systems still exist that use non-SQL data access languages, but these will continue to fade over time.
Star schema	An alternative term for a dimensionally modeled data structure. The term "star" comes from the visual representation that comes from showing a table of numerical values in the middle and the many dimensions used for drilling around the outside like points in a star.
Subject-oriented	Data configured to the entities, concepts, and language understood by the business when doing analysis. Contrasts most sharply with the source-oriented form in which most operational data is received by the data warehouse.
TCP/IP	A network communication protocol. It allows computers to communicate with each other in a very consistent manner. It is the protocol that underlies the Internet as well as most internal corporate networks. Runs on top of Ethernet and InfiniBand.
Tool	A deliberately loose term used to encompass any combination of software, hardware, vendor applications, custom code, queries, data, algorithms, processes, and other solution elements gathered together to solve a problem. "Tool" is used heavily in this paper because complex solutions often require a diverse combination of specific component types; to state any one is often inaccurate; to keep listing all the components is cumbersome. Classic example: the performance a user sees in "standard BI tools" rarely has to do with the BI software (i.e., visible software); it is actually due primarily to the ability of the database (i.e., more software) to stripe the data across many drive arrays (i.e., hardware) according to the partitioning setup by IT (i.e., design), as well as the interconnects (i.e., networking) between database server nodes (i.e., more hardware) that facilitate data movement; thus it is better to say "BI tools" unless "BI software" is specifically what is meant.
UNIX	An operating system common on servers in IT data centers. Linux is generally regarded as a form of UNIX, though purists might argue otherwise.
VLCS	Very-large calculation system: the term coined for an IT system that fits the design pattern detailed in this paper. To prevent acronym-overload, "big calculator" may be used in conversation since this term captures the essence of what the VLCS is.
Windsorize	An outlier handling technique where outliers are placed into the bin closest to their extreme. E.g., data points above the 95th percentile are placed in the 95th percentile bucket and data points below the 5th percentile are placed in the 5th.
WS-*	Web service standards: a collection of dozens of standards produced by the World Wide Web consortium (W3C) for communication among computers. Competes with REST.

Biography of the Author

James Madison is a senior information architect at a large insurance company. He has managed, designed, and built various VLCS systems during more than a decade of insurance industry work. He is an adjunct professor of computer science at Rensselaer Polytechnic Institute. He can be contacted at madjim (at) bigfoot.com.