EXTENSIBILITIES OF A JAVA-BASED STATISTICAL SYSTEM

Ikunori Kobayashi*, Takeshi Fujiwara†, Junji Nakano‡, and Yoshikazu Yamamoto§

ABSTRACT

A statistical system needs to be able to work with other systems in a flexible way and be easily extensible, because no one statistical system can implement all the features required by a wide variety of users. Recently, new ways of extending statistical systems have been brought about by the rapid development of software technologies, including the Java language. In this paper, we propose the use of several Java language techniques for realizing extensions of Jasp, a Java-based statistical system, and demonstrate three practical features of these extensions.

The first practical feature is demonstrated in the ability to handle eXtensible Markup Language (XML) documents, which are the standard format for data transmission on the WWW. The second practical feature is the use of the Component Object Model (COM) interface of Microsoft Windows for communication, e.g., with Microsoft Excel. The third practical feature is the sharing of program resources from other statistical software products, and to perform this task we have extended Jasp by using a translator to execute functions written in the XploRe language.

1. Extensibilities of statistical systems

General-purpose statistical systems are expected to fully support the various stages of statistical analysis. When we acquire data, statistical systems can help us in designing experiments, or they can automatically sample data. To describe and show the characteristics of the data, we calculate descriptive statistics, execute exploratory data analysis techniques, and draw graphs. Considering the results from such operations, together with a working knowledge of the data, we construct mathematical or algorithmic models for the data generation mechanism. By using the obtained models, we predict future values or make decisions for practical purposes. For supporting this whole work process, recent comprehensive statistical systems that contain a large number of functions such as database management, drawing graphics, interactive operation environment, model description languages, and report writing have been developed. As each user has a different emphasis when working with each of these functions, it is impossible to make a complete system that is suitable for all users. A feature for customizing a system, and one that can add desired features is therefore indispensable for statistical systems.

*Tokushima Bunri University, 1314-1 Shido, Sanuki, Kagawa 769-2193, Japan
†Aoyama Gakuin University, 4-4-25 Shibuya, Shibuya-ku, Tokyo 150-8366, Japan
‡The Institute of Statistical Mathematics and The Graduate University for Advanced Studies, 4-6-7 Minami-Azabu, Minato-ku, Tokyo 106-8569, Japan
§Tokushima Bunri University, 1314-1 Shido, Sanuki, Kagawa 769-2193, Japan

Key words: COM; Statistical system; XML; XploRe.
Statistical systems also have to handle data from various inputs. Statistical theory and procedures encompass techniques that can be useful for many common purposes, and in statistical software products, data are stored and arranged systematically for ease of use. As we cannot prepare all the procedures required for all data when building a system, some procedures need to be added as and when they are required. Therefore, almost all statistical systems are designed to have extensible features.

An easy and effective way of extending a system is to write a program in the built-in statistical language, which is designed to be simple when written for statistical calculation procedures. However, it is usually difficult to change fundamental system-related features using its own language, and for this, we have to write complicated programs in a system construction language, for example, in C, to add new low-level functions, such as those needed for handling other data formats.

The most primitive way of adding functions to a statistical system is to use separate software products. For example, programmers have used independent editors to write programs in statistical languages for many years. Recently, however, statistical systems have had their own editor as part of the system. Advanced users sometimes use two or more statistical systems for analyzing a particular dataset to execute various required procedures. In this case, users have had to learn the different operations of the systems. This is often not an easy task, especially for inexperienced users.

Another way of extending a system is to use part of other software directly from inside a program, meaning that existing programs can be reused. If a program that executes the same functions required by a user has already been written, then it is natural to consider using it in new software. However, the realization of this task in practical statistical systems requires many complicated programming techniques, and previously, it was almost impossible for statisticians to perform this task.

Over the last decade, computing environments have developed rapidly and continuously. Important among these changes is the development of technologies for reusing software products. For example, object-oriented frameworks can provide a theoretical basis for building independent software parts. Widely used common platforms, such as Microsoft Windows, UNIX (including Linux), and Java, can run programs on a large number of computers. Mutual connections among computers via the Internet have brought about an immense quantity of available data and programs. We can also exchange and share information from many users at the same time through the Internet. These technologies help us to implement software reuse mechanisms more easily.

Considering the above situation, it is not wise to construct one, large statistical system that tries to include all functions within it from the beginning. Rather, it is desirable and more promising to design an extensible core statistical system that can efficiently cooperate with other independent software products.

For achieving such ends, recent statistical systems have encompassed some extensibility. For example, the S statistical system (Chambers and Hastie, 1992) can call on modules written in Fortran and C. Many statistical programs have been developed in Fortran and C over a long period, and they have been well tested. However, such traditional languages lack the mechanisms of highly modular written programs, and have some problems in encapsulation and error handling, when used with other systems.

Object-oriented languages are designed to have sufficient modularity and reusability, e.g., the widely used Java language. Java has many good features for supporting advanced programming techniques, for example, constructing a graphical user interface, and the ability
to handle recently developed network technologies such as eXtensible Markup Language (XML). Because of these advanced design principles, we propose the use of the Java language in building statistical systems. Of course, Java has some defects. For example, it is slow compared with other compiler languages and it is still under development. Considering the history of the Java language, these deficiencies are being improved on steadily, and we expect that they will be fully solved in the future.

Therefore, we decided to develop a statistical system, a Java-based statistical processor (Jasp) using the Java language, taking into account its extensibilities (Nakano et al., 2000). Like other statistical systems, Jasp has its own language. The Jasp language is easy to use, and is sufficiently powerful because it is designed to have the characteristics of both a function-based language and an object-oriented language (Kobayashi et al., 2002). It can use Java classes directly from within. We adopted a scripting language, Pnuts (Tomatsu, 2001), as the basis of the Jasp language, used several Java libraries, and adopted a preprocessing approach to build it. This approach was effective for developing Jasp within a short period (Fujiwara et al., 2001).

In this paper, we describe three features of Jasp, which have been added by utilizing the extensibility of the Java and Jasp languages. The first feature is the use of XML documents. As XML is designed to be flexible for various uses, it is thought to be a promising intermediate form for data exchange among different software products on the WWW. We used DandD rule (Shibata, 2000) as the Document Type Definition (DTD) for this extension. The second feature is the use of an interface to a Component Object Model (COM) module. COM (Microsoft, 2000) is the basis of the Microsoft Windows family software, such as the spreadsheet application, Excel. Microsoft Windows is widely used, so that the Excel format is one of the de facto standards for data exchange. By using JCom (Watanabe, 2001), an open source Java-COM bridge library, we can realize the import and the export of data using the Excel format. The third extension is the execution of XploRe quantlets, which are quantitative procedures written in the XploRe language, and run in the XploRe computing environment (Härdle et al., 1999). By using the XploRe quantlet import mechanism, XploRe users can enjoy the advanced features of Jasp, and Jasp users can use the many XploRe quantlets in the Jasp system. We implemented this by adding a translator that converts the XploRe quantlets into Jasp programs, and so extending the Jasp interpreter.

2. Handling XML documents

Various data are described by the XML markup language (World Wide Web Consortium, 1997). Markup languages use tags to add information to the text by indicating the logical components of a document. XML is widely accepted as the standard intermediate data exchange format, and is useful for diverse software products, such as database systems and spreadsheets. XML is so powerful that we can express not only procedures but also various types of data in a uniform way. It may be used in statistics to both share and analyze data together. This function will be useful for future statistical environments.

In XML documents, we can use DTD to define in advance the regulations for using tags for a particular purpose. If an XML document conforms to the rules of a DTD, then it is easy to interpret its contents, and many users can share it more easily. Several DTDs have been proposed for describing statistical data, for example, DandD (Shibata, 2000) and the Data Documentation Initiative (DDI) (DDI committee, 2001). We used DandD to manage the XML documents in Jasp.

To use XML documents in programs, the Document Object Model (DOM) specifica-
tion was defined. DOM is an Application Programming Interface (API) used to access a document’s logical structure and contents, and it is independent of programming languages. Some realizations of DOM exist in Java, and we adopted the Xerces parser (Apache XML project, 2000) for parsing XML documents, because there are many Xerces related documents available, and many Java programmers use it. Thus, an XML document is imported as an object into Jasp, because Xerces can construct a Java class from an XML document, and Java classes can be used directly in Jasp. However, as all information in XML documents is treated as strings in Java, we had to convert some of these into real numbers. Some functions for this task were implemented in Jasp by programming the necessary Java class and Jasp functions for easier access.

As an example, the following commands can handle a dataset described by the DandD format.

```java
acid = JaspSimpleXML("acid.j.xml")
acid_tree = acid.printTree()
acid_matrix = importDataVector(acid, 3)
y = acid_matrix[,"rainfall"]
x = acid_matrix[,"sulfate"]
plot = plot_scatterdiagram(y, x, "rainfall vs. sulfate")
```

In the first line, the function “JaspSimpleXML()” reads an XML document written in a DandD DTD file named “acid.j.xml”, creates a DOM instance in Java, and assigns it to an object variable named “acid”. The structure of acid is displayed in the second line. The function “importDataVector()” in the third line creates a data matrix in the Jasp language from the variable acid that represents the DOM object. The second argument of this function expresses the number of the column. The last line produces a scatter diagram instance.

We also implemented a method to convert a Jasp matrix to a file written in XML. For example, a 2 x 3 matrix named `exMatrix` (with row and column labels)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

can be output to an XML document named `exMatrix.xml` by the command

```
exMatrix.writeXML("exMatrix.xml").
```

The following shows the content of the document.

```xml
<JaspMatrix>
  <observation>
    <x>1</x>
    <y>2</y>
    <z>3</z>
  </observation>
  <observation>
    <x>4</x>
    <y>5</y>
    <z>6</z>
  </observation>
</JaspMatrix>
```

In this case, this XML document has no DTD declarations.
3. The COM interface

Microsoft Windows is the most widely used computer platform. Almost all statistical systems that were originally developed on other platforms, such as mainframes and workstations, are available for Windows, and are now mostly used in Windows. In addition, Microsoft Excel, a Windows platform spreadsheet application, is convenient for handling data and for calculating basic statistics, and is widely used on personal computers for various daily activities. Therefore, huge amounts of data are stored in the Excel format.

In cases where statistical systems cannot directly read and write a file in the Excel format, we need to convert such data into either standard Comma Separated Values (CSV), or their original data formats. This is a troublesome job for most users. Therefore, for our Jasp system, we needed to implement functions that could read and write Excel files.

Recent Windows applications support COM technologies. COM is a software architecture for component-based development, which assembles application systems from pre-manufactured software components or parts. COM defines a language-independent, object-oriented, extensible, and binary interoperability standard that allows individual components to communicate with each other. Almost all recent Windows applications use COM technologies, including Microsoft Office software, such as Excel and Word. Because Excel can be used as a COM component, these communication capabilities are available from any COM component. We handle Excel data from Jasp using COM technologies.

Some software libraries for handling COM that use Java programs are open to the public via the Internet. We have adopted a Java-COM bridge, JCom (Watanabe, 2001) for Jasp, because JCom is a group of Java classes, that can be directly called from Jasp programs. We constructed wrapper classes for JCom in the Java language to customize it for statistical computing. We imported these wrapper classes into Jasp, and wrote easy-to-use wrapper functions in the Jasp language. Therefore, Jasp users need only handle Excel files using these functions.

The following program shows an example to use the wrapper functions.

```java
loadJasp("ExcelInterface")
x = readExcel("sample.xls", 1, 114, 1, 2)
x
autocorr = Autocorrelation(x)
writeExcel(autocorr.autcor, "sample1.xls", 1, 3, true)
```

The first command, `loadJasp("ExcelInterface")`, loads the functions (library) for manipulation of the Excel data files. The function `readExcel()` is one of the wrapper functions, and it reads a file in Excel format. The five arguments of this function express the input file name, the start row, the end row, the start column, and the end column of the cells where the data are kept. The `readExcel()` function returns an object of the matrix. In this example, the object is assigned to a variable, `x`. The command line that has only the variable's name shows the content of the assigned object in the GUI and CUI windows. The function `Autocorrelation(x)` calculates the autocorrelation of `x` and returns the result.

The function `writeExcel()` is used in the last command line to write the matrix data stored in Jasp into an Excel spreadsheet. The five arguments used are the matrix data, the output file name, the row and the column of a cell where the upper left corner of the data matrix is stored, and a flag for labeling. Figure 1 shows a snapshot of the Jasp session after the above example had been executed.
In this implementation, we focused on communicating with Excel, but other COM applications can be used as well using Jasp via JCom. However, the use of Jasp functions from Excel is not considered. Note that this feature works only with Windows platforms.

4. Executing XploRe quantlets

To extend the features of Jasp, we added the ability to execute XploRe quantlets by using the internal translator to translate the XploRe language to the Jasp language. XploRe is a powerful and practical statistical system that has several advanced features, such as implementation of many recent statistical procedures, it is an easy language, and it has advanced Internet capabilities. XploRe quantlets are procedures written in the XploRe language that run in an XploRe computing environment (Härdle et al., 1999). On the other hand, Jasp is an experimental system that realizes different advanced features, for example, use of a Graphical User Interface (GUI), and use in distributed computing. It is beneficial for users to use Jasp and XploRe together in order to enjoy the advantages of both systems.

We have chosen XploRe quantlets to extend Jasp’s abilities for three reasons. First, the grammar of the XploRe language is simple but comprehensible. Second, the XploRe language can handle data objects, although it lacks some object-oriented characteristics, such as an inheritance mechanism. Third, abundant procedures are available in XploRe, and many of these are written in the XploRe language, and not in C++ (from which the XploRe system is constructed). As a whole, there are many common features between the XploRe and Jasp languages.

Jasp is implemented using a preprocessor that converts the original Jasp programs
Extensibilities of a Java-Based Statistical System

into Pnuts programs (Tomatsu, 2001) and extracts any GUI environment information before executing the programs (Fujiwara et al., 2001). Figure 2 shows the structure of the interpretation of Jasp programs.

The translator converts libraries and programs written in the XploRe language into the Jasp language. The Jasp language is also preprocessed and is translated into pure Pnuts programs. There is another way of implementation to convert the XploRe language directly into the Pnuts language, and to execute it. However, we did not adopt this approach because once XploRe programs are converted into Jasp programs, then users can modify and edit the programs themselves, and they are helpful for users who are accustomed to the Jasp language.

The translator is implemented using a Java Compiler Compiler (JavaCC) (Sun Microsystems, 2000), which is a tool used to generate the Java program source using lexical analyses and parsing, such as Yacc and Lex for the C language (Kernighan and Pike, 1984).

Even when using JavaCC, several difficulties exist in the realization of the XploRe translator. The first difficulty arises from the two possible access methods for the return value of an XploRe object. The return value can be set to a variable as a whole, and each field can be extracted using the dot notation, or return values can be set to two or more variables separately at the same time when the calculations are complete. We solved this problem by writing a Java class named JaspList. This uses a feature of Pnuts in which the reference (or the substitution) of the member of the instance of the class that implements the pnuts.util.Property interface is automatically rewritten by calling the get (set) method. Some XploRe functions can be realized using the C++ language, and we have written Java programs for implementing them in Jasp. Examples of these are graphics functions. Although we have designed Jasp’s graphics capabilities on the basis of XploRe’s graphics capability, there are some differences. We have written new XploRe-compatible graphics classes in Java to solve this problem, and have replaced a formerly-used plot library with these classes. Some XploRe quantlets cannot work in Jasp because of the above differences with Jasp.

Figure 3 is an example program that uses a sir quantlet in XploRe. The sir quantlet is one from the XploRe library metrics, and is used to fit a multiple index model using the Sliced Inverse Regression method (Härdle et al., 1999). Figure 4 shows a snapshot of the above program being executed in Jasp. The statistical graphs generated in Figure 4 are displayed by graph drawing functions, which have been omitted in the above program.
loadJasp("Xpl2Jasp")
loadXploRe("XploRe_lib/sir.xpl") // load XploRe quantlet

nSIR = 200 // number of data
xSIR = JaspMatrixNormal(nSIR, 3, 0.0, 1.0) // generate random numbers
eSIR = JaspMatrixNormal(nSIR, 1, 0.0, 1.0)
bSIR2 = JaspMatrix([1.0, -1.0, -1.0], 3, 1)
bSIR1 = JaspMatrix([1.0, 1.0, 1.0], 3, 1)
ySIR = xSIR*bSIR1 + pow(xSIR*bSIR1, 3.0)
       + 4*(pow(xSIR*bSIR2, 2.0)) + eSIR

sir_result = sir(xSIR, ySIR, 20) // call the sir function
edr = sir_result.edr // refer the result
eigen = sir_result.eigen // refer the result

Fig. 3: A program to refer an XploRe quantlet in Jasp

Fig. 4: A snapshot of the execution of Figure 3
5. Concluding remarks

Because statistical systems must support all statistical work as a whole, traditional systems have become very large through the inclusion of the many features necessary for statistical evaluation including, for example, an editor or a database. By using recently developed software technologies, we have devised another way to construct an extensible general-purpose statistical system that can work with other software products.

Java is a language that assembles many advanced technologies in a unified way. Because Jasp is written in Java, Jasp benefits from the many excellent extensibilities of Java. Three features have been demonstrated in this paper. They were selected because they are widely used, and are well recognized as de facto standards. These technologies are still in continuous development, and are subject to gradual change. As it is important to adapt rapidly to these changes for easy collaboration, the easy and seamless extensibilities demonstrated in this paper are both desirable and indispensable for statistical systems. However, we have to note that our implementation here is still in the experimental stage, and requires considerable improvement for practical use.

Acknowledgements

The authors are grateful to the editor and referee for their helpful comments and suggestions for improving this paper.

Appendix : WWW addresses

We have used many programs which are open to the public on the Internet. Most of them are announced only as documents on the WWW, and their locations may be changed without notification. Therefore, we have listed current locations of such documents referenced in this paper as an appendix.

http://xml.apache.org/xerces-j/.

http://www.icpsr.umich.edu/DDI/.


Shibata, R. (2000). DandD.

Sun Microsystems. (2000). JavaCC.

http://www.pnuts.org/.

http://www.w3.org/XML/.

REFERENCES


