Fujiwara, Takeshi; Nakano, Junji; Yamamoto, Yoshikazu; Kobayashi, Ikunori

Working Paper

An implementation of a statistical language based on JAVA

SFB 373 Discussion Paper, No. 2001,72

Provided in Cooperation with:
Collaborative Research Center 373: Quantification and Simulation of Economic Processes, Humboldt University Berlin

Suggested Citation: Fujiwara, Takeshi; Nakano, Junji; Yamamoto, Yoshikazu; Kobayashi, Ikunori (2001) : An implementation of a statistical language based on JAVA, SFB 373 Discussion Paper, No. 2001,72, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin, https://nbn-resolving.de/urn:nbn:de:kobv:11-10050444

This Version is available at:
https://hdl.handle.net/10419/62691

AN IMPLEMENTATION OF A STATISTICAL LANGUAGE BASED ON JAVA

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

ABSTRACT

As computing technologies have been developed rapidly and continuously, statistical languages should be frequently improved by adopting them for realizing comfortable statistical environments. Although that work was a difficult task before, we can implement such languages relatively easily by modern software techniques such as the Java language, which unifies recent computer technologies widely and neatly.

In this paper, we describe the idea of a new Jasp (JAva based Statistical Processor) language, which is a function based and object oriented language, is suitable for interactive operation, and has flexibilities and diverse extendibilities. Considering required characteristics for a statistical language, we decide to implement the Jasp language by Java technologies through the Pnuts language, which is a script language written in Java, and a preprocessing approach. This implementation brings several advantages such as reduction of development costs, effective uses of existing resources, and enough reliabilities.

1. Introduction

One of the essential parts in a statistical system is its language, by which users describe their statistical works. Nowadays, we have many statistical systems which have rich set of statistical procedures, user friendly interfaces and high reliabilities. However, some of their languages can not use recent computing environments fully, partly because they are designed many years ago. For the past few decades, computing environment has progressed continuously and rapidly. For adopting new technologies efficiently, a statistical language needs to be newly designed.

In recent statistical systems, function based languages, for example, the S language (Chambers, 1998), are admitted to be flexible and intuitive to express formulae, functions and tentative programs. At the same time, object oriented languages such as the Java

*The Graduate University for Advanced Studies, 4-6-7 Minami-azabu, Minato-ku, Tokyo 106-8569, Japan (fuji@ism.ac.jp)

†The Institute of Statistical Mathematics, 4-6-7 Minami-azabu, Minato-ku, Tokyo 106-8569, Japan (nakanoj@ism.ac.jp)

‡Faculty of Engineering, Tokushima Bunri University, 1314-1 Shido, Okawa, Kagawa 769-2193, Japan (yamamoto@is.bunri-u.ac.jp)

§Faculty of Engineering, Tokushima Bunri University, 1314-1 Shido, Okawa, Kagawa 769-2193, Japan (ikunori@es.bunri-u.ac.jp)

Key words: Function based language; GUI; Java; Object oriented language; Preprocessing; Statistical System.

language are accepted to be good at arranging and reusing programs and well-organized knowledge as the form of encapsulated objects. An object oriented approach may also arrange relevant statistical methods and techniques as a hierarchy of classes, which is a kind of models to collect similar data structures and procedures. It is desirable that a new statistical language has both function based and object oriented characteristics.

Considering these requirements for a statistical language, we design and implement a new statistical language (The Jasp language) for the statistical system Jasp (Nakano *et al.*, 2000), which has various features. The Jasp language is designed and implemented using the Java language mainly based on the Pnuts language (Tomatsu, 2000). Java is a well-designed object oriented language which uses various recently developed abilities systematically, and Pnuts is a simple function based script language which is written in Java and can use Java classes directly. We extend Pnuts for having object oriented abilities and functions required for statistical works using a preprocessing approach. This paper describes basic ideas of language design and implementation techniques which can keep our programming works as little as possible.

2. Considerations on a modern Statistical Language

2.1. Indispensable Features for Statistical Languages

A statistical language is used to express and execute the process of statistical analyses which a user wants to perform. It is also the main means of the communication between a user and the system. For helping users effectively in these works, a statistical language of today needs to have several features.

First, a statistical language must be easy and intuitive to use. Most users of a statistical system are not professional programmers. They do not want to use much time for studying another command language. This is the reason that a function based language is preferred as a statistical language, because functions are easy and suitable to express statistical procedures. Writing short functions for a particular small part of the statistical analysis is rather simple and we can perform whole statistical works by using these small functions. In addition, a language without type declaration is preferred. Type declaration is important for a rigid compiler language, but is too formal for tentative statistical works.

Although a function based language is easy to use, it is not good at arranging many functions, because it does not have the mechanism to encapsulate related functions. This can be essentially realized by an object oriented programming approach. Well arranged objects reflect statistical notions naturally, and are intuitive. However, designing objects at the beginning of the statistical analysis is a difficult task. In this stage, we have to try to perform many statistical techniques in order to grasp the clear image of data and appropriate statistical procedures. After executing such trial and error, we come to the stage of designing objects. Thus, a statistical language requires both features of a function based language and an object oriented language.

Second, a statistical language needs to be suitable for interactive operations. In general, a statistical analysis is a process of operating various statistical procedures one by one with checking results of each operation. Even when we use one particular statistical procedure, it may be performed repeatedly for checking the results and trying to variously transformed data. Results of each calculation will be used as information for deciding the next possible statistical procedures, or may be used as inputs of other procedures. If a language is not fit for interactive use, these operations are so complicated and cumbersome. It is clear that an

interpreter language is better for interactive operations than a compiler language, because the former responds more quickly than the latter.

Third, a statistical language needs to be extendible. Developers of a system can not realize all the vast requirements of users. In addition, computing technologies and environments have proceeded rapidly, and it is necessary to follow them continuously. Therefore, a statistical language needs to be extended easily by not only developers but also users.

2.2. Implementation of Statistical Languages

For implementing statistical languages, two approaches have been mainly used; one approach is to implement a new language from scratch, and the other is to use an existing language as a base of statistical language.

Many statistical languages, for example, S and XploRe (Härdle *et al.*, 1999), adopt the first approach. It has the advantage that the language can be designed unconstraintly for statistical purposes. However, the cost for developing and maintaining the system are expensive. Enhancing and changing the language specification which was once implemented requires professional skill for programming and is difficult.

On the other hand, for example, XlispStat (Tierney, 1990) adopts the second approach. It uses the Lisp language as a base, and functions for statistical works are added on it. This approach can reduce the programming cost remarkably by using the functions of the base language, but, at the same time, language features are greatly affected by the base language. Although Lisp is a flexible and powerful language, the Lisp style programming of XlispStat is not easy and intuitive to use for many data analysts.

3. The Jasp Language

3.1. Design principles of the language

Although we have many statistical languages, we are not satisfied by them because of lack of some features that we consider in the previous section. We decide to design and implement new Jasp language for realizing such features.

First, we design Jasp language as a function based language basically for easiness and intuitiveness. All the abilities can be executed as function calls. At the same time, an object oriented mechanism is implemented to bundle related functions without changing original function programs. Constructor methods of objects are used as almost same as function calls. It is recommended that beginners of statistics and programming or users who perform basic analyses write programs by functions, and users who consider future reuse of programs write programs by classes. It also possible that at the initial stage of the analysis, users mainly use functions, and that at the next stage, they collect and package related functions as a class form. This feature of the Jasp language has advantage to other statistical languages.

Second, we design Jasp language as a script interpreter language largely based on Pnuts, because it is suitable for executing functions interactively. We can also perform commands in a file at a stretch, that is, batch processing. The Jasp CUI (Character User Interface) embeds the Jasp interpreter and helps users to execute these operations from environments such as a program editor and a command line editor.

Third, the user can extend Jasp system using another languages. For developers or advanced users, efficient implementations of new functions are required to be written in the system construction language, that is, Java for Jasp. This task is easy in Jasp, because

the Pnuts language can call Java classes directly. This is one of the main reason we choose Pnuts as the base of the Jasp language. At present, many programmers have developed various Java applications as classes. They are all available from Jasp with little efforts. Some advanced users require complicated statistical calculations, and need to write Java programs by themselves, which can also be used from Jasp.

In the statistical community, we have many programs which have enough reliability and speed and were written in traditional languages such as Fortran or C. We sometimes want to use them from Jasp. The JNI (Java Native Interface), which is an interface between Java and other languages, makes this possible. As an example, TIMSAC (TIMe Series Analysis and Control) package (Akaike and Nakagawa, 1989), which is a set of Fortran programs for executing various time series analysis techniques, are embedded in Jasp by this mechanism.

3.2. Implementation of the Language

For implementing the Jasp language, we adopt the second approach in the section 2.2; Jasp is built on Pnuts. As we want to extend Pnuts for statistical purposes as we hope, just adding functions are not enough. For extending the Pnuts language, two approaches may be available: to change the source code of the Pnuts language, or to write the preprocessing program for translating the Jasp language into the Pnuts language. The first approach needs almost as same development and maintenance cost as that for implementing a language from scratch, because it requires complete understanding of the base language implementation. The second one has a weak point, i.e. the execution speed is slow because the process of execution needs two steps. This approach, on the other hand, has some advantages. The cost of development and maintenance can be greatly reduced, because the real execution works are performed by the base language. Usually, the base language is used widely, tested well, and maintained continuously, so the resulted language is more reliable than the newly implemented language by changing the source code of the base language. We implement the Jasp language by adopting the preprocessing approach to use the features of the Java and the Pnuts languages.

3.2.1. Java and Pnuts for Jasp

In recent computing environment, the Java language attracts considerable attention. Java adopts recent new computing technologies in a unified way. It has various advanced features such as platform independency, object oriented programming and substantial libraries. Java programs can be executed on various operating systems such as Windows, Linux and Solaris without any modifications. Because of the object oriented design of the Java language, we can reuse resources efficiently by using features such as the encapsulation and inheritance mechanism. In the Java language specification, many libraries for GUI (Graphical User Interface) and Network usages are prepared as standard libraries.

It is, however, not easy to learn and use the Java language. Object oriented programming is not suitable for tentative works in statistical analyses, because it is a difficult task to encapsulate statistical data and methods as a class at the beginning of the analysis. As Java programs need to be compiled, interactive operations are difficult. As results, although Java has many good abilities, it is not suitable for statistical analyses as it is.

To overcome these demerits of the Java language, we use Pnuts, a script language for the Java environment. Pnuts is easier to use than Java because it is a typeless interpreter function based language. As Pnuts is based on the Java language, it has high affinity with Java. For example, we can use all Java libraries directly from Pnuts.

As Pnuts is a general purpose language, it is possible to perform statistical analyses

in it. However, it is extremely cumbersome, because basic statistical functions are not included in Pnuts. As a statistical language, basic statistical libraries must be equipped at least for matrix computations, graphics, statistical distributions. We can implement them by using and modifying existing Java libraries, because the Pnuts language can use Java classes directly and dynamically. Even if we add these functions to Pnuts, the resulting system is still not easy to use because of the lack of operations which are required for statistical works. The built-in extendible abilities of Pnuts are useful to simplify the syntax for statistical computations. Some complicated functions, however, are difficult to realize by them, for example, the connectivity with a GUI. We employ an approach to translate small part of the Jasp program to the Pnuts program before processing by the Pnuts interpreter. This means that the core part of the Jasp language is as same as the Pnuts language, and has additional functions for statistical works (Figure 1).

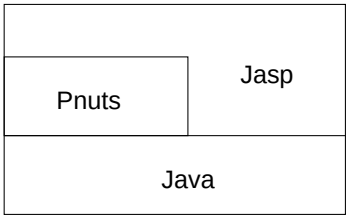

Fig. 1. Pnuts and Jasp

3.2.2. Using existing Java Libraries

We import the Jampack (Stewart, 2000) library for matrix computations, the Ptplot (Lee and Hylands, 2000) library for drawing 2D graphs and the Colt (Hoschek, 2000) library for calculating statistical distributions and the random number generation as built-in functions of the Jasp language. Because the Java language is an object oriented language and each library is implemented as classes, they are well encapsulated and can be reused effectively from other Java programs.

Pnuts, the base of Jasp, is a Java application and designed to use Java classes as simple as possible; it needs only two steps. First we import a library, then use its classes in the almost same way in the Java language, except following two small differences.

In the Java language, we use `new` command to generate an instance. For example, when we create an instance of a normal distribution with zero mean and unit standard deviation, we have to use the sentence

```
Normal normal = new Normal(0.0, 1.0, new Drand(new Date()));
```

where `Normal` is the normal distribution class which is imported from the Colt library. In Jasp, the same work is performed by the simpler sentence

```
normal = Normal(0.0, 1.0, Drand(Date()))
```

without using type declarations.

Another difference is the invocation of static methods. For example, the `staticNextDouble` method for returning a random number from the normal distribution with the given mean and standard deviation are used by the sentence

```
x = Normal.staticNextDouble(0.0, 1.0);
```

in Java, and by the sentence

```
x = Normal::staticNextDouble(0.0, 1.0)
```

in Pnuts. Java libraries written by users can be used from Jasp in the same way.

3.2.3. Wrapping existing Java Libraries

Just using existing Java libraries as it is sometimes inconvenient for a statistician. One example is matrix computations using the Jampack library. If we add `a` and `b` by Jampack, where `a` and `b` are appropriate instance objects of `Zmat` class of Jampack, we have to use the sentence

```
c = Plus::o(a,b)
```

by using the static method `o` of `Plus` class. This syntax, however, are not simple and intuitive at all. Natural syntax `c = a + b` using the add operator `+` should be used in this situation. We can use built-in extendible abilities of Pnuts for realizing it. We use the translation function of arithmetical operations: an object (Java class) which can be an operand of an operation such as `+` and −, and implements `pnus.util.Numeric` interface, invokes the corresponding method automatically. For instance, if you use the expression `a + b`, Pnuts automatically execute `a.add(b)` method. In Jasp, as matrix calculations are performed by using the JaspMatrix class, which inherits the Jampack library and implements `pnus.util.Numeric` interface, we can also use same simple syntax for matrix computations.

3.2.4. Preprocessing Approach

We have shown the ways to implement the Jasp language by using Java libraries and an extendible ability of Pnuts. Some required complicated syntax, however, can not be simplified by these techniques. We may have two possibilities to resolve them: to analyze and modify the internal implementation of Pnuts for our purposes, or to preprocess and translate original programs into Pnuts programs. In the first approach, we have to understand the inside of the Pnuts implementation more throughly and need more programming works than in the second approach. This will cause the difficulty to follow a version up of Pnuts itself. Therefore, we decide to use the second way to implement the Jasp language. As the Pnuts language is a simple function based language and the large part of the syntax is also appropriate for statistical programming, we add and modify only small parts of it for realizing the Jasp language. Jasp programs are firstly preprocessed and translated into regular Pnuts programs, then they are executed by the Pnuts interpreter, as is shown in Figure 2. For implementing the translator, we use Java CC (Java Compiler Compiler) (Sun Microsystems, 2000), which is as same as the Yacc and Lex (Kernighan and Pike, 1984) for the C language.

About matrix operations, for instance, if we want to substitute the submatrix which consists of rows from the second one to the third one of `3x3` matrix `x` to a variable `y`, we can write

```
y = x[2:3,]
```

in the Jasp Language. This sentence is translated into the Pnuts sentence

```
y = x.get([2,3],[1,2,3])
```

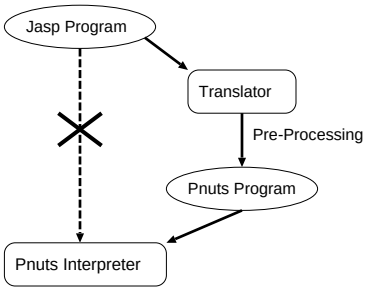

Fig. 2. Preprocessing Approach

by the preprocessor.

In Jasp, CUI and GUI operations are unified, that is, all of CUI operations are recorded as corresponding GUI operations. For example, if we create a variable by substituting a value in the CUI, the corresponding icon of the variable will appear in the GUI. In order to realize it, it is required to send information from the CUI to the GUI. We implement such a connectivity in the Jasp language by the translator. A simple Jasp sentence `x = 1` is replaced into the Pnuts sentences

```
x = 1
srv.addNode("x")
```

where the augmented command means to add a new icon which represents a variable `x` in the object list area of the GUI window (see Figure 3). In addition, this icon has the layered structure, and the user can refer to an analytical history by the structure. For instance, when mean value `m` and variance value `s2` are estimated from `x`, these icons are displayed in the hierarchy under the icon of `x`. Moreover, when the user click the icon, the operation list of variable will be displayed. The user can execute the operation easily with selecting item of list. It is also possible that the user operates a part of analysis in the GUI, and then operates another part of analysis in the CUI.

Usually in a statistical system, a language interpreter and a UI (User Interface) are independently designed, mainly because a UI is changed and improved more often than a language. To embed UI features in a language interpreter may make the language unstable. This preprocessing approach, however, brings the simple implementation of the UI features in the Jasp language, and we can change them without much programming works.

4. Conclusion

The Jasp language is designed for executing statistical calculations conveniently, especially in order to realize easiness, interactive operation environments and extendibilities. As the Jasp language is function based, a user can write tentative small functions easily. And at the same time, they can be systematically arranged as objects by the object oriented framework for later uses. As the Jasp language is simple and typeless interpreter language, and the interpreter can communicate with users through both a CUI and a GUI, users can operate Jasp interactively. Jasp can be extended by Jasp programs, Java programs and foreign language programs such as Fortran or C without much efforts.

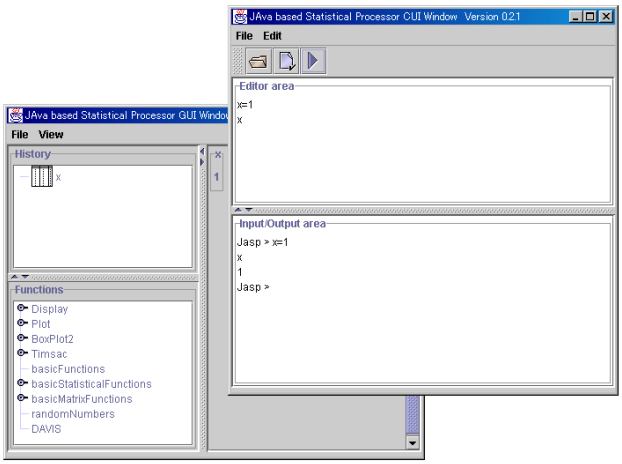

Fig. 3. The Jasp User Interface

For implementing these abilities, we use the Java language and the Pnuts language as a base, and adopt the preprocessing approach. This approach is a powerful solution for building a new statistical language, and is useful for reducing development and maintenance costs, and keeping the reliability of the system.

REFERENCES

Akaike, H.and Nakagawa, T. (1989). *Statistical Analysis and Control of Dynamic Systems*. Dordrecht: Kluwer Academic Publishers.

Chambers, J. M. (1998). *PROGRAMING WITH DATA – A Guide to the S Language*. New York: Springer.

Härdle, W., Klinke, S. and Müller, M. (1999). *XploRe – Learning Guide*. Berlin: Springer. (`http://www.xplore-stat.de/`)

Hoschek, W. (2000) Colt.
(`http://nicewww.cern.ch/~hoschek/colt/`)

Kernighan, B. W. and Pike, R. (1984). *The UNIX Programming Environment*. New Jersey: Prentice-Hall.

Lee, E. A. and Hylands, C. (2000). Ptplot.
(`http://ptolemy.eecs.berkeley.edu/java/ptplot/`)

Nakano, J., Fujiwara, T., Yamamoto, Y. and Kobayashi, I. (2000). A statistical package based on Pnuts. In: *COMPSTAT2000 Proceedings in Computational Statistics*, 361–366. Heidelberg: Physica-Verlag.

Stewart, P. (2000). Jampack.
(`ftp://math.nist.gov/pub/Jampack/Jampack/AboutJampack.html`)

Sun Microsystems. (2000). Java CC.
(`http://www.webgain.com/products/metamata/java_doc.html`)

Tierney, L. (1990). *LISP–STAT*. New York: John Wiley & Sons,Inc.
(`http://www.stat.umn.edu/~luke/xls/xlsinfo/xlsinfo.html`)
Tomatsu, T. (2000). Pnuts.
(`http://javacenter.sun.co.jp/pnuts/`)