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

Working Paper

A mixed user interface for a statistical system

SFB 373 Discussion Paper, No. 2001,77

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

Suggested Citation: Yamamoto, Yoshikazu; Nakano, Junji; Fujiwara, Takeshi; Kobayashi, Ikunori (2001) : A mixed user interface for a statistical system, SFB 373 Discussion Paper, No. 2001,77, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin, http://nbn-resolving.de/urn:nbn:de:kobv:11-10050516

This Version is available at:
http://hdl.handle.net/10419/62737

A Mixed User Interface for a Statistical System

Yoshikazu Yamamoto[1], Junji Nakano[2], Takeshi Fujiwara[3] and Ikunori Kobayshi[1]

[1]Tokushima Bunri University, 1314-1 Shido, Kagawa 769-2193, JAPAN
[2]The Institute of Statistical Mathematics, 4-6-7 Minami-Azabu, Minato-ku, Tokyo 106-8569, JAPAN
[3]The Graduate University for Advanced Studies, 4-6-7 Minami-Azabu, Minato-ku, Tokyo 106-8569, JAPAN

Summary

A user interface is one of the most important factors for deciding the usefulness of a statistical system. Nowadays, a graphical user interface (GUI) is popular because it is easy and intuitive to use. A character user interface (CUI) is, however, still important for using full abilities of the system by writing statistical programs in order to perform complicated statistical analyses which have not been realized in the system. We propose a "mixed user interface" for utilizing a GUI and a CUI alternatively and seamlessly, and consider its required characteristics.
We also explain an implementation of the mixed user interface of the statistical system Jasp (Java based statistical processor), which is written in the Java language and adopts many recently developed computer technologies.

Keywords: Character user interface, Graphical user interface, History of a data analysis, Object-oriented approach

1 Introduction

The purpose of statistical system software is to support statistical analyses for users. In spite of a lot of efforts of making statistical "expert systems", it is still impossible for any statistical system to analyze data automatically, i.e., to judge values of results of an analysis and select statistical models as human experts can do. What we can expect for a statistical system is to perform heavy calculations for the data handling or the model fitting, and to show various graphs for visualizing data or models. Therefore, smooth collaboration between a user and a statistical system is important. A user wants to convey his or her ideas about analyses simply and accurately to the system, and to see results of data processing by the system in various ways clearly and intuitively. All these problems are related to the user interface of the system. It apparently decides the usefulness of the system heavily, although the total value of a statistical system, of course, is affected by many other factors such as supported statistical procedures, reliability and preciseness of calculation, possibility of collaboration with other software, etc. This paper focuses on the problem of a user interface of modern statistical systems.

As statistical models have been described mainly by mathematical expressions, it would be best for a statistical system to manipulate mathematical expressions directly on a graphical user interface (GUI). They are, however, still difficult to be handled by computers. Reading and writing mathematical expressions are not easy tasks even for modern computers. We need some training to write mathematical expressions in word processing at will. At present in almost all systems, mathematical expressions need to be written in statistical languages, which have been also used as communication means on character user interfaces (CUIs), in which a user writes programs by a editor, and executes them by using an interpreter or a compiler.

Recently developed statistical techniques, for example, non-parametric density estimation, bootstrap methods and neural networks, are expressed mainly by complex algorithms, not by simple mathematical expressions. Although mathematical considerations are given to them, they are less useful for interpreting real data analysis results, because of their tremendous complicated structures. Statistical algorithms can also have been fully expressed by programs written in statistical languages. In these senses, statistical languages and CUIs are still important.

A GUI is apparently easier to use than a CUI, as a GUI can offer the full screen to show various information, and a user does not need to memorize all the language grammar. Because a GUI can visualize ideas, it is easy to grasp the whole image of an analysis intuitively. Programming by a GUI may be intimate or friendly when we design icons and other GUI parts pleasantly.

A GUI, however, has some demerits. We often open too many windows on the screen and lose our analysis way, partly because they are so easily created by just one click of a mouse. To write programs and to show programming logic by the GUI are still difficult. It is also not suitable to perform very complicated analysis by the GUI.

As GUIs require much more computer resources both in hardware and software than CUIs, they became widely available just recently. Many GUIs of statistical systems have been designed by a "composite user interface" (Liu, Chan, Montgomery & Muller 1995) approach, which adds a separate GUI front-end program on the statistical system which had only a CUI. This approach was thought to be reasonable, because we had many established CUI statistical systems, which were widely used and enough reliable. For example, Nakano (1998) adopted this approach to add a GUI using Internet to the traditional statistical system SHAZAM (White 1997). However, Yamamoto and Nakano (2000) pointed out that this approach has the deficiency that the design of a GUI is strongly restricted by the original CUI. They also designed an experimental GUI for time series analysis, in which the history of the data analysis is recorded as a tree of data objects and the system can be used by simple mouse operations for them.

Users who are satisfied with a GUI at first, often become to hope to perform complicated data handling which are not included in the system when they are accustomed to the system. Then programming works become required. Therefore, we need both a CUI and a GUI for statistical systems, and they should be used alternatively and seamlessly. We call the user interface which has these characteristics as a "mixed user interface", which will be considered in the next section in detail. As the example realization of a "mixed user interface", we will explain the user interface of our statistical system Jasp (Java based statistical processor) in section 3.

2 User interfaces for statistical systems

Statistical analysis generally consists from several steps. We need to acquire data, show visual displays for summarizing their characteristics, calculate basic statistics and transform them for better descriptions. After these preliminary data handling, we fit models for approximating the data generation process, diagnose them and seek better models. We repeat these processes again and again until we have satisfactory models. At the last stage, we use estimated statistical models to predict future values or simulate future movements. In these exploratory works, it is important to record the history what we have done before. When we succeed in our statistical analysis, we sometimes notice that developed procedures are meaningful statistical techniques also for other data. We definitely hope to store them for later use in

convenient ways. All these processes need to be supported by a user interface of a statistical system.

Historically, a CUI has been used mainly as an environment to write and execute programs. Programs are written in the statistical language, which is designed for programming statistical works easily and flexibly, and for systematically describing statistical knowledge in the system.

Programs at the first stage of a tentative analysis are generally short and simple, but full of various techniques. For these purposes, interpreter languages are more suitable than compiler languages, because interpreters can respond quickly. The function-based procedural language is thought to be a good solution by its easiness of simple modular programming. The type-less language is less restrictive than the typed language.

The grammar of a statistical language should be simple but enough powerful for vast range of statistical uses, as users of statistical systems are not professional programmers. Statistical languages should include frequently used functions, such as matrix calculations, maximization of complicated functions, and well known statistical procedures. For realizing these objects, statistical languages have been developed gradually from the simple subroutine call of Fortran to powerful function-based languages such as S (Chambers & Hastie 1992) and XploRe (Härdle, Klinke & Müller 1999) languages.

For arranging developed programs in the hierarchy of statistical techniques for later use, object-oriented programming is desirable, because it has mechanisms which are useful for a knowledge representation, such as an inheritance.

As we know these characteristics are hopeful for a statistical language, we tried to implement them as the Jasp language (Nakano, Fujiwara, Yamamoto & Kobayashi 2000). Details of the Jasp language are stated in the separate paper.

Besides a language, the environment for using it is also important for a statistical system. Even if the language is simple, we still have to study hard to master it and use it freely. As learning and using the language are not easy, we need an environment to support these processes. We are first interested in a CUI environment for them. An editor window for writing programs is clearly indispensable. It is natural to execute a part or whole of them from inside the editor. We also need an interactive command line window to execute one line or a few lines of commands given by the user directly. Results of them should appear just after the command input in the same window. Graphs, however, is not suitable for a character user interface, then they can be shown in a GUI window.

Recently, interactive develop environments (IDEs) for general purpose languages become popular for developing programs. IDEs help users for writing programs by using GUIs. Although a statistical system with a GUI looks as

same as the IDE, there may be differences: the IDE is a GUI environment for writing and testing programs, and a statistical system is not only for writing and testing statistical programs but also for supporting the whole statistical analysis processes. In other words, the IDE is a closed environment in the world of programming, and a statistical system is an open environment to consider the domain knowledge from which data were generated and the future usage of results in various ways. In data analysis processes, we sometimes do not know how to handle the data at all when we first face to them. We have to gather knowledge about the data by iterating some data handling processes and make clear what we can do about that data. This point is also different from usual programming works, in which the object of programming is clear from the beginning. It will never happen that a user who does not know programming at all tries to use the IDE. Not a few people who use statistical systems are naive for statistics. Statistical systems need to help such users efficiently.

A GUI may be desirable to display data and calculation results as icons which are arranged as a tree for expressing the history of the data analysis. Such records of the history are useful in a trial and error process to seek better ideas for data or models. As icons of data and statistics should represent one conceptual statistical object, we need to apply possible statistical procedures for them easily, for example, each icon has a pop-up menu for listing available procedures and they can be started by simple mouse operations such as click or drag and drop. Functions in the system should be displayed also as a tree which reflects the statistical theory systematically in order to find the procedure we want to use without much efforts.

In a GUI, we need a window for showing graphs or tables as results of the statistical analysis. As such displays easily become large amount, functions to hide unnecessary objects tentatively are required.

It would be good if all functions of the system which are available from the CUI can be used also in the GUI. It is, however, difficult because visual programming is still on the research stage. What we can realize at most is that almost all operations, except defining new functions and new statistical object classes, can be performed through the GUI window by mouse operations more easily than through the CUI window.

As was stated in the previous section, we often have to use a CUI and a GUI alternatively and seamlessly, i.e., we want to switch from one to another easily. We call such a user interface as a "mixed user interface". A CUI and a GUI in a mixed user interface have the same importance and there is no one way dependence between them. We have to design a good GUI without thinking about the language, and we have to design a good CUI or language without thinking about the GUI, at the beginning. Then, the user who uses a CUI only and the user who uses a GUI only will be both satisfied. In this

sense, a CUI and a GUI should be independent. However, a CUI window and a GUI window of the mixed user interface are completely dependent when they are used, because they communicate each other continuously, and almost all operations done in one window are also recorded in the other window. In realizing a mixed use interface, we have to implement a CUI or a language and a GUI to satisfy the above stated design principle as much as possible by adjusting them together at the same time. Object-oriented programming is thought to be useful for realizing this kind of interface.

3 Jasp user interface

An example of the "mixed user interface" is implemented in the statistical system Jasp. As Jasp is written in Java, it runs on many platforms supported by Java virtual machines. Jasp system consists of server and client programs. This structure is adopted for realizing distributed computing mechanism, and is also useful for implementing the mixed user interface.

The Jasp language is based on Pnuts, a scripting language written in Java (Tomatsu 2000). Jasp language is a function-based and a type-less interpreter language with the object-oriented framework.

The Jasp user interface runs as the client program. It can be invoked both as an application and as a Java applet from Web browsers. Jasp application client can read and write local files, but Jasp applet can read data from files specified by universal resource locators (URLs). The Jasp client program opens both a CUI and a GUI window separately.

3.1 CUI window

The CUI window has the upper editor area and the lower input/output area (Figure 1). In the upper area, we can read a program file written in the Jasp language, edit it and evaluate a part or whole of it. These operations can be performed by a key operation or a pull-down menu of the CUI window. We can also save the content of the editor or the input/output areas into files.

We can feed our commands directly to the lower input/output area. They are interpreted immediately, and calculation results are displayed successively in the window. We can copy the typed command to the upper editor area by the pull-down menu. This is useful for recording and modifying commands.

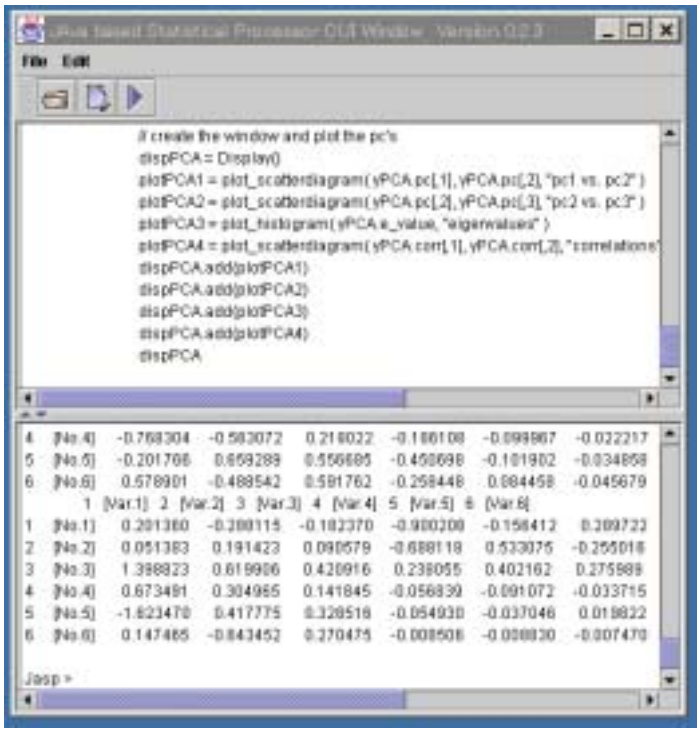

Figure 1: Jasp CUI window

3.2 GUI window

The GUI window consists of three areas (Figure 2). The upper left area shows the analysis history as a tree form, whose leaves are icons of objects or variables. The lower left area shows available functions and constructors of statistical objects also as the tree form.

In the upper left area, data, statistics and graphs are displayed as icons with the name of variables to which they are assigned. Temporary variables, whose name begins by underscore (_), are not displayed. We can show all variables including temporary variables by a menu operation. Icon trees can be folded by clicking + and changing it into −, and can be unfolded by the reverse operation.

These icons are manipulated directly by the mouse. When we select an icon by the mouse, a pop-up menu appears. This menu is a list of available methods for the object. For example, when we specify basic variables, we

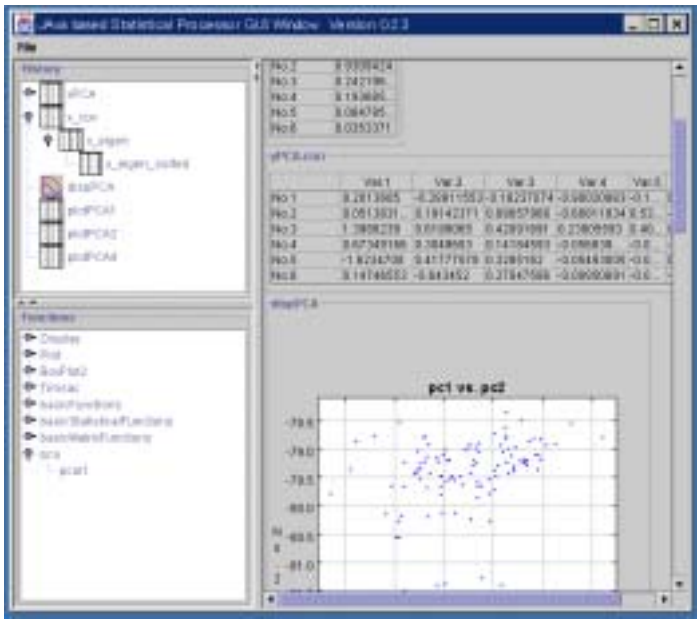

Figure 2: Jasp GUI Window

have a menu whose items are `Show value` for displaying its value in the right output window, `Delete` for deleting it. Icons which represent instances defined by Jasp class definitions will show the menu whose items are list of available methods when they are specified. In case arguments are required for executing functions and constructors, new input window will open, and we have to specify them by the drag and drop operation or typing variable names. All operations given in the GUI window are recorded as program executions in the CUI window.

The history of the data analysis is shown as an icon tree in the upper left area. New data or statistics object created by an operation is expressed as an icon of the sub tree under the original object.

In the lower left area, function and constructor names are shown as a statistical method menu in the tree form. When we click them, a selected function or constructor is invoked. A group of functions are stored in the "library" file. The name of library is displayed as the top level leaf of the tree, and their functions are shown as its sub tree. Constructors are arranged by the class definitions, i.e., an inheritance hierarchy of classes is displayed as the tree form. A class which does not have specified super class is a top level leaf

of the tree. This mechanism is useful for the menu display which reflects the structure of programs or the underlying statistical theory.

As we want to keep the simplicity of the Jasp language, it does not have any syntax for supporting a GUI. Users can write Jasp programs without thinking about the GUI. However, if we hope to use the program also on the GUI, we need additional information for the GUI. For example, there are functions and constructors which are not for the public use but for the internal use. They should not appear on the menu for public usage. We want to distinguish such functions from more important functions. Another example is the type declaration of arguments of functions. Although the Jasp language has no type declaration, it would be useful to check the appropriateness of the input variable at the GUI operation. Such checks are easily available if types of input arguments are specified in some ways.

For these purposes, we decide to include additional information as comments of programs in special formats. At present, we can write types and default values of input arguments, and explanations of functions and constructors. Functions and constructors without these comments are not shown as menu items in the lower left area. When a function or a constructor is specified, additional information in comments are shown in the input window to help users.

3.3 Mixed user interface

New users of Jasp probably start from the GUI operation. When they are familiar with the grammar of the Jasp language and the structure of Jasp functions and constructors, they move to the CUI. In our mixed user interface, however, we can always use the CUI and the GUI windows alternatively and seamlessly.

All operations using the mouse in the GUI window are also displayed in the input/output area in the CUI window. We can see the record of GUI operations, and learn commands, functions and constructors of Jasp in the CUI window. It is possible to write a program by combining recorded operations in the upper editor area of the CUI window.

When we execute commands by typing in the input/output area or evaluating a Jasp program in the editor area, results are shown in the input/output area, and at the same time, in the GUI window as icons which represent tables and graphs of data or statistics. Thus, the history of the data analysis is shown as the tree form. We note that it is difficult to show the relation of variables and loaded data, and the analysis process in the CUI window only.

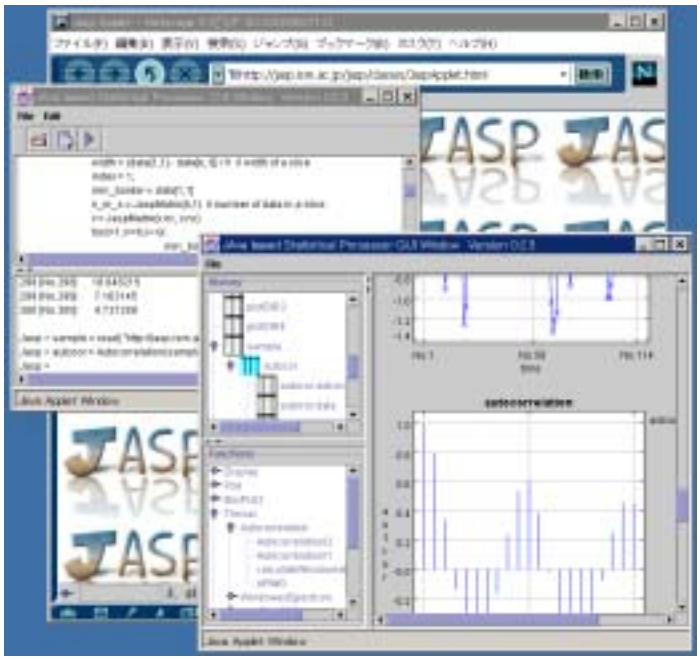

Figure 3: An example of Jasp applet

3.4 An example of the Jasp user interface

Figure 3 is an example of the whole Jasp user interface, working as an applet of Netscape 6 on the Microsoft Windows 98. We select `Open URL` button and read an example program included in the Jasp distribution file. The program is shown in the upper area of the CUI window. We select this entire program by pushing `Select All` button or by mouse dragging. When we push `Evaluate` button, the selected program is executed. Results are shown in the lower area of the CUI window by characters and in the GUI window by icons, graphs and tables.

When we use the GUI , we usually start from `File` menu of the GUI window, then select `Load URL data` and read data from the URL. The data can be shown as an icon in the upper left area of the GUI window. For example, if we want to calculate the autocorrelation function of the time series data, we select `Timsac` in the lower left area and select `Autocorrelation/1` menu item. This constructor needs the name of variable to which calculated results are assigned. It should be specified in the pop-up input window. Arguments

of constructors or functions can be given by drag and drop operations. All these operations are recorded in the CUI window.

4 Conclusion

As a statistical system is often used by users who are not expert statisticians or professional programmers, a friendly user interface is one of the most important factors of it. For these users, an easy and simple GUI is preferred when they start their statistical works. Even such users will hope to perform complicated statistical methods for their data by using a CUI when they proceed their works earnestly.

Therefore, we propose a "mixed user interface" in which users can use a GUI and a CUI alternatively and seamlessly. We insist that a GUI and a CUI need to be independently designed at the beginning, and operations on one interface should also be reflected on the other. We consider the importance of the recording the analysis history on the GUI in an exploratory statistical analysis.

As a realization of the "mixed user interface", we explain the user interface of our statistical system Jasp (Java based statistical processor). Jasp is available freely from our Web site (`http://jasp.ism.ac.jp/`).

References

Chambers, J.M. & Hastie, T.J. (ed.) (1992), *Statistical Models in S.*, Pacific Grove: Wadsworth.

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

Liu, L.-M., Chan, K.-K., Montgomery, A. L. & Muller, M. E. (1995), A system-independent graphical user interface for statistical software, *Computational Statistics & Data Analysis*, **19**(1), 23–44.

Nakano, J. (1998), Graphical User Interface for Statistical Software Using Internet, in *COMPSTAT 1998 Proceedings in Computational Statistics* (ed. Payne, R. & Green, P.), 407–412. Heidelberg: Physica-Verlag.

Nakano, J., Fujiwara, T., Yamamoto, Y. & Kobayashi, I. (2000), In: *COMPSTAT2000 Proceedings in Computational Statistics*, 361-366. Heidelberg: Physica-Verlag.

White K.J. (1997), *SHAZAM – User's Reference Manual Version 8.0*, New York: McGraw-Hill. (`http://shazam.econ.ubc.ca/`)

Tomatsu, T. (2000), Pnuts, (`http://javacenter.sun.co.jp/pnuts/`)

Yamamoto, Y. & Nakano, J. (2000), *A time series analysis system using visual operations*, SFB 373 Discussion Paper No. 32, 2000. (`http://sfb.wiwi.hu-berlin.de/`)