Lehmann, Heiko

Working Paper

XploRe Quantlet Client: Web Service for Mathematical and Statistical Computing

SFB 373 Discussion Paper, No. 2003,23

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

Suggested Citation: Lehmann, Heiko (2003) : XploRe Quantlet Client: Web Service for Mathematical and Statistical Computing, SFB 373 Discussion Paper, No. 2003,23, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin, https://nbn-resolving.de/urn:nbn:de:kobv:11-10050233

This Version is available at:
https://hdl.handle.net/10419/22238

XploRe Quantlet Client – Web Service for Mathematical and Statistical Computing

Heiko Lehmann

CASE, Humboldt-Universität zu Berlin, Spandauer Strasse 1, 10178 Berlin, Germany

Abstract. Many mathematical and statistical questions require the use of computational assistance. The proposed XploRe Quantlet Client (XQC) with it's combined CUI and GUI interface gives access to the XploRe software environment (XploRe Quantlet Server – XQS) with it's large number of available mathematical and statistical methods (Quantlets) via the Internet. To offer the client to a large community the Java language is used to implement the client's functionalities.

Keywords. Web services, User Interface, Net based mathematical and statistical computing, Client/Server architecture, Java, XploRe, www.xplore-stat.de/java/java.html

1 Introduction

Mathematical and statistical research and the use of computers and its calculating power has become an inseparable unit. A large number of mathematical and statistical methods have been developed during the last decades, e.g. nonparametric methods, bootstrapping time series, wavelets, estimation of diffusion coefficients. Most of these methods are only available for a certain software package or software environment, but not for widely used standard software packages. To apply these methods to empirical analyses a potential user may face a number of problems. It may often even be impossible for him to use the existing methods without rewriting them in a different programming language.

A similar problem occurs in teaching statistics. For students it is often difficult to relate statistical concepts presented by their teacher to real world situations and problems. Use of computers has proven itself as favorable in this area to illustrate the statistical content learned in class. Quite a few projects are currently working on this topic (e-Stat, WebStat, MM*Stat). Kinds of realization of these projects go from Java Applets to visualize statistical content to complete electronic books and tutorials containing interactive examples.

Statistical software has also become an important part of scientific research that is reflected in the publications of the research results. Publishing a mathematical theorem requires also the publication of the proof of this theorem. The result of a computation can be seen as the equivalent of a mathematical theorem. When publishing results of statistical research, it is desirable for an interested reader to be able to verify and regenerate these results. Even more desirable to an interested researcher is to be able to inspect the source code, modify it and produce variations of the results. Buckheit and Donoho (1995) outline the topic of *Reproducible Research* – "When we publish articles containing figures which were generated by computer, we also publish the complete software environment which generates the figures." They propose to publish research papers as electronic books and include the software environment the results where generated with to make them interactively accessible.

The approach presented in this paper is meant to solve the challenges stated above. This paper describes a client that offers access to the statistical computing environment XploRe with its broad range of available statistical methods (see Härdle et al. 1999). Due to compatibility reasons the clients is fully programmed in Java. It can run as a Java application as well as being started as a Java applet. Special configuration files allow for influencing appearance and behavior of the XQC.

2 XploRe Quantlet Client/Server Architecture

The general XploRe Quantlet Client/Server (XQC/XQS) architecture is based on a common three level client/server model as shown in figure 1. It consists of the main components server, middleware and client (see Kleinow and Lehmann 2001).

A **server** is offering services to one or more client(s). The server of the XQC/XQS architecture consists of the XploRe Quantlet Server (XQS) representing the powerful statistical computing engine written in C++ that provides a high-level statistical programming language. Running on a remote computer the XQS can offer a magnitude of computer power, which many users would not be able to access in other ways. Having access to the method- and database the XQS and the method- and database respectively is easily extendible by new statistical methods via XploRe programs (Quantlets) as well as native code methods, e.g. *-dll* and *-so*. For server side communication purposes the middleware MD*Serv is attached to the XQS. The Communication between MD*Serv and XploRe server is realized via standard I/O streams – the middleware reads from the server's standard input and writes to its standard output.

The server offers access to a **data**- and **method pool**, which contains a variety of methods and data. This easy extendible database ensures the possibility to add newly developed statistical methods and to use them via the client without any changes on the client side.

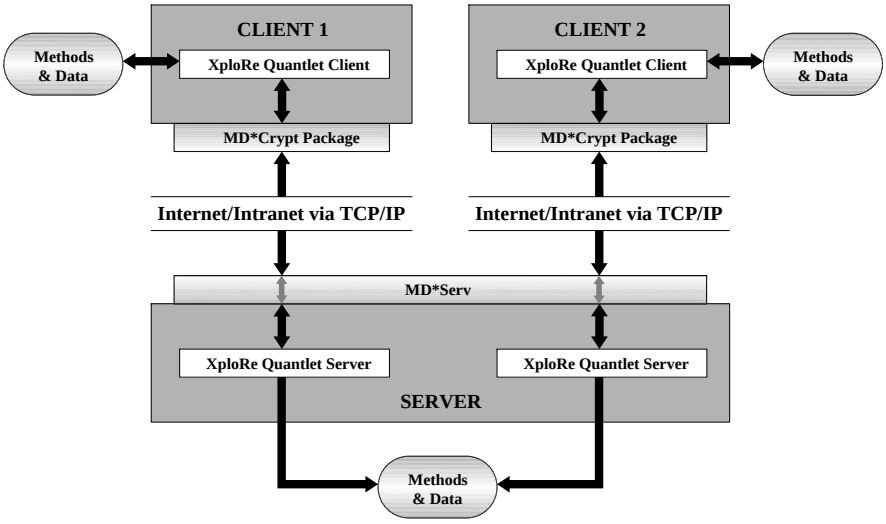

Figure 1: XQS/XQC architecture

The **client** is the part of the architecture requesting a service. Using the client the user is able to access the statistical methods, data and computing power offered by the server. The XploRe Quantlet Client (XQC), responsible for presenting the statistical results, represents the client of

the XQC/XQS architecture. For client side communication purposes the MD*Crypt package is attached to the client. The MD*Crypt package supports the client side communication between client and server in the XQC/XQS model. It is implemented in a single Java package to make it available to different clients, e.g. XQC and GraphFitI (Blauth, 2000), without double programming effort. The MD*Crypt package gets in and keeps contact to MD*Serv using the MD*Crypt protocol. Via TCP/IP incoming data are prepared to be available to the client in an easy accessible way. On the other hand MD*Crypt package offers methods that allow to send commands to the XploRe server (see Feuerhake 2001). Running as an application or a certified applet the XQC can access its own local data pool containing statistical methods and data.

3 Aims of the XQC

The XploRe Quantlet Client (XQC) represents the front end - the user interface (UI) of the XQC/XQS architecture. The XQC is fully programmed in Java. Using a pure Java solution the XQC does not depend on a certain computer platform. It can run on Windows and Mac platforms as well as on Unix and Linux machines. Running as an application or a certified applet the XQC can access resources of the computer it is running on. Because of Java's sandbox principle the XQC underlies restrictions of accessing the local system while running as an applet. Unfortunately these restrictions also restrict its functionality of accessing local methods and data. Figure 2 shows a screen shot of the XQC running as an application.

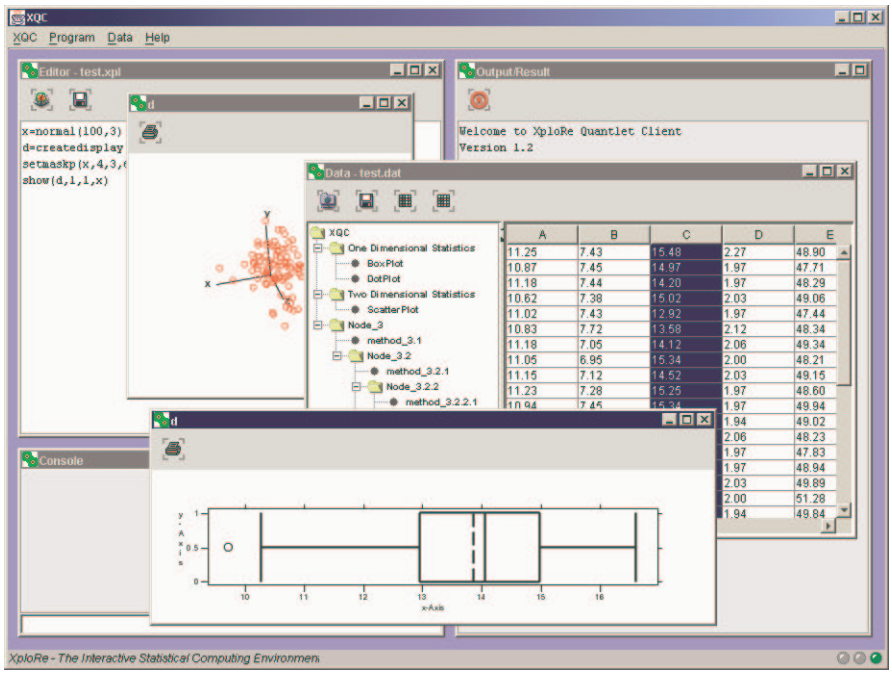

Figure 2: XQC in action

The appearance of the XQC is based on the MS Windows XploRe version (see Härdle et al. 1999) to ensure an easy handling and a familiar look. In difference to MS Windows version that only offers a CUI the XQC combines a CUI with GUI functionality. The CUI (Character-UI) functionality is realized by a console that offers direct command line access to the XploRe server and an editor-window with a text area for writing and executing more than just a single line command. To make use of this functionality the potential user needs to be familiar with the XploRe programming language. The GUI (Graphical-UI) functionality is realized by a combined

data- and method-window. It enables the user to explore data with given methods without having to know the XploRe language itself. Like the name implies the data- and method-window consists of two parts - an excel-like table for editing the data and a tree with methods for the use on the data.

Property files allow for customizing the XQC to meet special needs and thus to manage its appearance and behavior. Because of its pure Java implementation, the resulting platform independence and the customizability via property files the XQC recommends itself for the integration into HTML and PDF contents (e.g. electronic books) for visualizing statistical and mathematical coherences.

4 Structure of the XQC

Due to the characteristic of the Java programming language all components of the XQC are capsulated in it's own class object respectively. This object-oriented characteristic of the Java language offers the possibility to reuse certain objects in other projects as shown for the XQC plot classes.

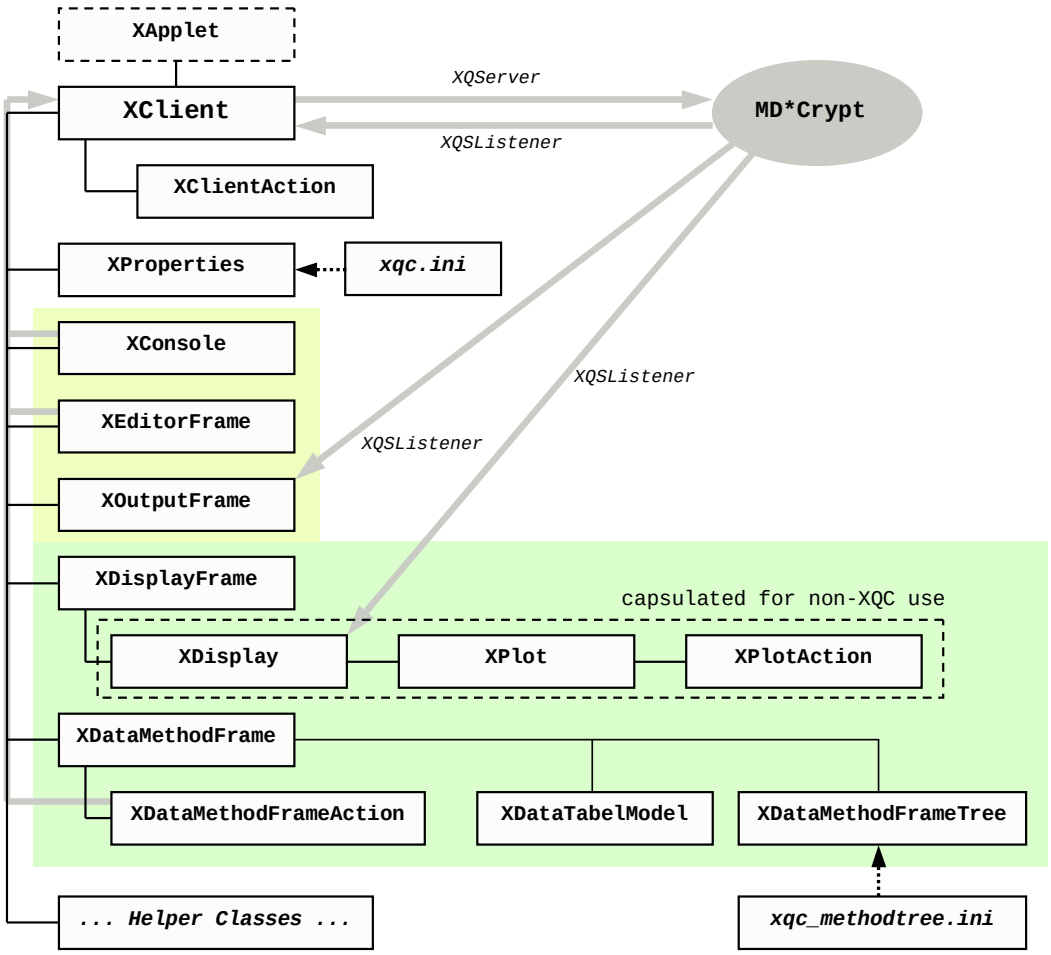

Figure 3: XQC - Structure

Figure 3 shows the general structure of the XQC and it's main components. Each square implies a single Java class. There also exist some – I would call them – "helper classes" used by other classes within the process but not shown in figure 3. Examples of these "helper classes" are a dialog class for message and error dialogs, classes to handle local files and Internet files (data and methods).

4.1 XClient – The Starting Point

The *XClient.class* works as the "Starting point" or "Main class" of the XQC. Calling its main-method a new instance of this class is created. The desktop frame with its menu bar represents this object visually. First activity of the XClient-object is to create an instance of the *XProperties.class* in order to access the xqc.ini file. Among other information this file contains information about the server and port to connect to. Property files will be discussed in detail later on. Using the given information the XQC tries to establish a TCP/IP connection to the server. For this purpose it uses MD*Crypt's services and creates an instance of the *XQServer.class* (see MD*Crypt package). This *XQServer.class* contains different methods that allow communication with the server.

4.2 User Interfaces

A user interface is a component that allows for interaction with another computer or like in our case server respectively. Two different types of interfaces can be distinguished – a Character User Interface (CUI) and a Graphical User Interface (GUI). A Character User Interface presents information to the user as text. It requires the user to type commands (known as command lines) to run programs. Unix and MS DOS are examples of CUIs. A Graphical User Interface on the other hand is an interface consisting of graphical elements such as windows, icons and as with the XQC of trees with underlying options. The user can select and activate these options by using pointing devices like a mouse. Since the user does not have to type in certain code in order to use options or to start programs or functions the GUI is much easier to use.

4.2.1 Character User Interface – CUI

The XQC's CUI is primarily meant for experienced XploRe users. It requires the user to be familiar with the XploRe programming language.

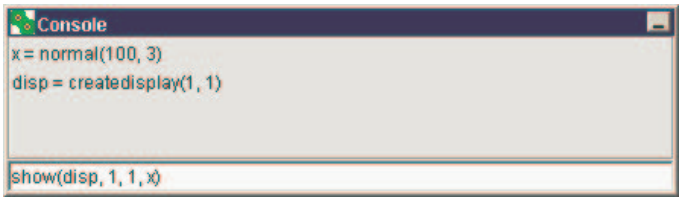

Figure 4: Console

The XQC's component **CONSOLE** represents one part of the Character User Interface (CUI). It consists of a command line window that allows entering single-line commands. These commands can be sent to the server for direct processing. Console itself does not have implemented an XQSListener (see MD*Crypt package) and can consequently not receive any results coming from

the XploRe server. A history of the last 20 commands sent to the server helps to keep an overview. They can easily be selected and executed again.

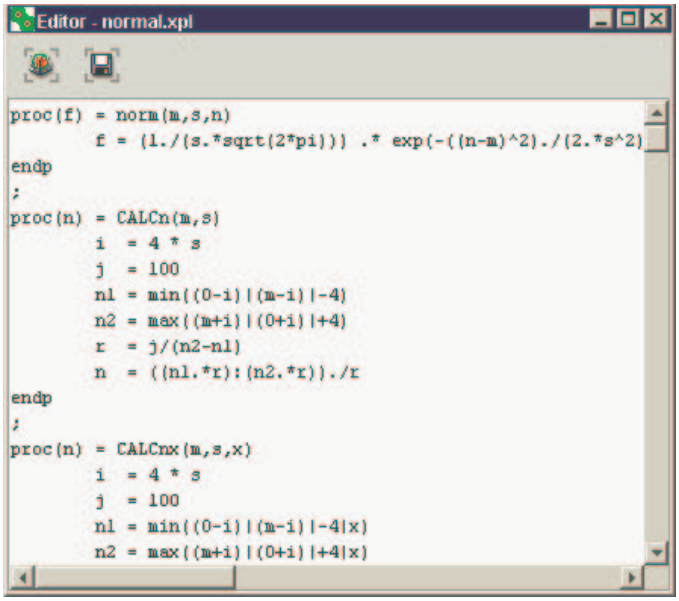

Figure 5: Editor frame

The **EDITOR FRAME** represents an extension of the single-line Console. This component is also part of the Character User Interface (CUI). It allows for writing and editing complete XploRe programs (XploRe Quantlets). In difference to the Console this XploRe code will not be send to and executed by the server after typing in a single line. Instead the execution is triggered for the complete XploRe program by clicking the according 'Execute' icon.

Both components do not receive any results coming from server. Depending on its kind result is either shown as text within the Output/Result frame or presented graphically in a display (part of the GUI).

As mentioned above, main task of the **OUTPUT FRAME** is the presentation of text output coming from the server. Therefore this component implements an XQSListener of the MD*Crypt package. Received content is not converted in any kind.

4.2.2 Graphical User Interface – GUI

The GUI is primarily thought for users that are not familiar with the statistical programming language XploRe and/or users that want to use existing statistical methods on their own data sets. Basic feature of a GUI is a surface that can be controlled intuitionally. A simple mouse-click can trigger functions of the underlying program without the need for the user to type a line of code.

The XQC's graphical user interface consists of several components. Right after starting the client a **DESKTOP FRAME** appears on the screen. Its menu bar allows for example to connect and disconnect with an XploRe server, open and save programs and data sets or to download data objects from server. Due to the sandbox concept of the Java programming language it must be distinguished between running the XQC as an application, a certified or a pure applet. Pure applets are limited in their functionality. They are not allowed to access local files or to use the

copy and paste functionality of the underlying operation system. For using the full functionality of the XQC it should be started as an application or a certified applet.

Most important GUI component for communication with the XploRe server is the **METHOD** and **DATA FRAME**. As shown in figure 6 this component consists of a method tree and a table model – a spreadsheet. Following its function as an input component of the GUI the method and data frame sends commands (methods and data) to the server via the MD*Crypt protocol. Server results are either shown within the Output/Result frame or presented graphically in a display.

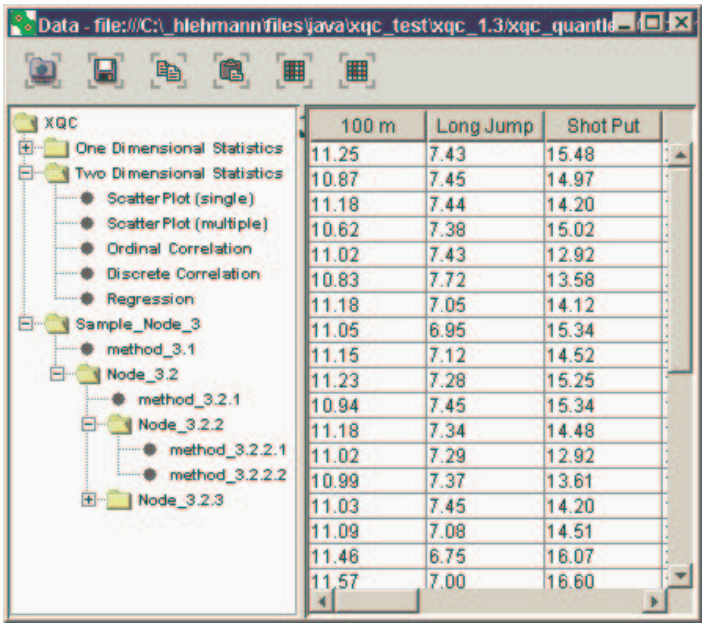

Figure 6: Method and data frame

The structure of the **TABLE** model is similar to a common spreadsheet. It holds the data that are object of research. To execute methods on the data, parts of the data set can be selected column wise or cell wise. Selection mode is chosen via the icons or the spreadsheet's context menu. In addition the table model allows maintaining column headers. These headers can either be set automatically while opening a dataset or manually via the spreadsheet's context menu. Headers are used for axis description within plots. This makes an interpretation of graphical output much easier. It is also possible to upload data to the server and to use those uploaded data within CUI components, e.g. programs written in the editor frame. Manipulation of the data set like deleting and inserting new rows is possible via the spreadsheet's context menu.

The **METHOD TREE** is a collection of executable methods thematically grouped. Navigation within the tree is very similar to a common file explorer (MS Windows Explorer). Analogous to the common file explorer the tree can consists of nodes and Childs on several levels. In the actual version of the XQC (version 1.4) the number of levels is limited to four levels. Childs represent statistical methods that can be used on the data. Nodes can be used to group methods. Using a special property file, that is discussed in detail later on, the method tree can be set up individually. This feature makes it possible to configure the XQC for different statistical or mathematical purposes, e.g. ‘Basic Statistics’ or ‘Multivariate Statistics’. The executable methods are XploRe programs that can either be stored at client side or at server side. As a defined strategy the XQC first tries to find a method on the client’s computer. On failure the XQC looks for the method on the server’s method pool (see figure 1).

To keep track of data uploaded to the server the XQC offers a **HISTORY**. This history contains a list of variable names of the uploaded data. Selecting a variable shows the source of this variable, changes performed on the variable and it's actual content. Due to performance reasons only uploaded data and actions on data from the CUI component CONSOLE and the GUI component TABLE MODEL are recorded. Parsing executed XploRe programs for uploaded variable information is not (yet) realized – this would slow down the speed of program's execution. For a future release it is planned to make the user able to record those data uploads and actions as well, adjustable via the property file.

Output within the GUI is realized throughout the **PLOT CLASSES**. These classes are responsible to present server results graphically. To receive results from the server plot classes have implemented MD*Crypt's XQSListener (see figure 3). Plot classes consist of different components – XDisplayFrame, XDisplay, XPlot and XPlotAction. One reason is the clarity in programming. Another even more important reason is an encapsulation of the pure plot classes (XDisplay, XPlot, XPlotAction) to make its functionality useable outside the XQC, useable for other client that want to use the XploRe server (see Mori). Only function of XDisplayFrame is the implementation of a desktop frame. This frame holds the XDisplay – a panel that can easily be integrated into other common Java components. The XDisplay itself holds the single plots realized by the XPlot class. This class converts server results according to the received information about type, shape and color to graphical output. Three-dimensional plots can be rotated using a pointing device or the cursor keys. This feature helps to discover structures in data sets, e.g. clusters. A context menu offers additional functionality. Java tool tips are used to show the coordinates of point within the plot.

In the current version of the XQC there is no communication (sending of information) from GUI components back to the XploRe server. A challenge for future work is to offer the possibility to manipulate data via GUI components and send those manipulations to the server. An example would be to mark outliers within plots and have them automatically marked in the spreadsheet as well.

4.3 Configuration of the XQC

Property files allow for configuration of the XQC to meet special needs of the user. These files can be used to manage the appearance and behavior of the XQC. As ordinary ASCII files any text editor can edit the configuration files. Generally all information is optional to use.

The ***xqc.ini*** file contains the basic set-up of the XQC.

```
Server    = localhost
Port      = 4451

Size      = 0.9
Width     = 800
Height    = 600
```

Up on start the XQC tries to connect to the server and port given in this file. If this information is missing the client starts with a dialog box to enter server and port manually. Size statements allow for adjusting the size the XQC takes on the screen – either by using a factor or by stating

it's exact width and height. To influence the behavior of the XQC the following property statements can be used:

```
ShowOutputWindow      = yes
ShowCommandWindow     = yes
ExecuteCommands       = normal(10,2)
ExecuteProgram        = file:///C:/.../normal.xpl
OpenInEditor          = file:///C:/.../test1.xpl
OpenData              = XQCROOT/decathlon.dat
```

Using these settings it is possible to easily embed the XQC into multimedia contents. It could be started with executing a certain Quantlet stated in the file without displaying console or output frame. In this case the XQC behaves like a Java applet programmed for a particular task. It could also be started with just opening a dataset with certain predefined and customized methods to execute the methods on the dataset. Path statements are possible as absolute paths – locally (file:///…) or URL (http://…) – as well as relative to the directory the XQC has been started in (XQCROOT/…).

```
ShowMethodTree        = yes
MethodTreeIniFile     = xqc_methodtree.ini
MethodPath            = XQCROOT/xqc_quantlets/
```

The actual method tree is set up in a separate configuration file that is given by the property of MethodTreeIniFile. This file contains a systematic structure of the tree – nodes and childs, the method to be executed and it's description to be shown within the tree frame.

```
Node_1 = path name
        Child_1.1 = method|description
        Child_1.2 = method|description
        Child_1.3 = method|description
    Node_2 = path name
        Node_2.1 = path name
            Child_2.1.1 = method|description
```

The name of the method has to be identical to the name of the XploRe program (Quantlet). The Quantlet itself has to have a procedure with the same name as the method. This procedure is called by the XQC on execution within the method tree.

A third property file allows setting up the XQC's language. This file contains all texts used within the XQC. To localize the client the texts have to be translated.

5 XQC in practice

Important features of the XQC are its realization in the programming language Java and its configuration possibilities via the property files. Due to these features the client recommends itself for a broad range of use. As described in a previous chapter Java makes a platform independent use possible. The client can be used as an applet integrated into HTML content as well as a standalone application.

The XploRe Quantlet Client is meant to address the following target groups:

- Statistical and mathematical newcomer
  By using the GUI functionality – e.g. a method tree set up for a certain topic for use on his/her own data
- Experienced statisticians and mathematicians with knowledge of the XploRe language
  By using the CUI functionality – XploRe console, XploRe editor
- Educational purposes
  XQC configured for a certain statistical and mathematical task using the configuration files
- Electronic books
  Integration of the XQC in multimedia content (pdf, html) to publish statistical and mathematical results with the possibility of reproducible research

Lets take for example a teacher who teaches his students basic statistics as happens in different areas. Not only a few students think of statistics as being far from reality and abstract. By using the XQC theoretical content can be filled with living examples. By using the property files the teacher is able to configure the XQC according to the subject of an ongoing lecture. A pre-configured client for the topic 'One dimensional Statistics' could consist of executable XploRe programs illustrating the presentation of ordinal data with a histogram and the effect of changing the bandwidth. Students are able to verify content learned in class without the need of purchasing extensive statistical software packages. While using the XQC as an application or a signed applet the student would even be able to use the methods on his/her own data.

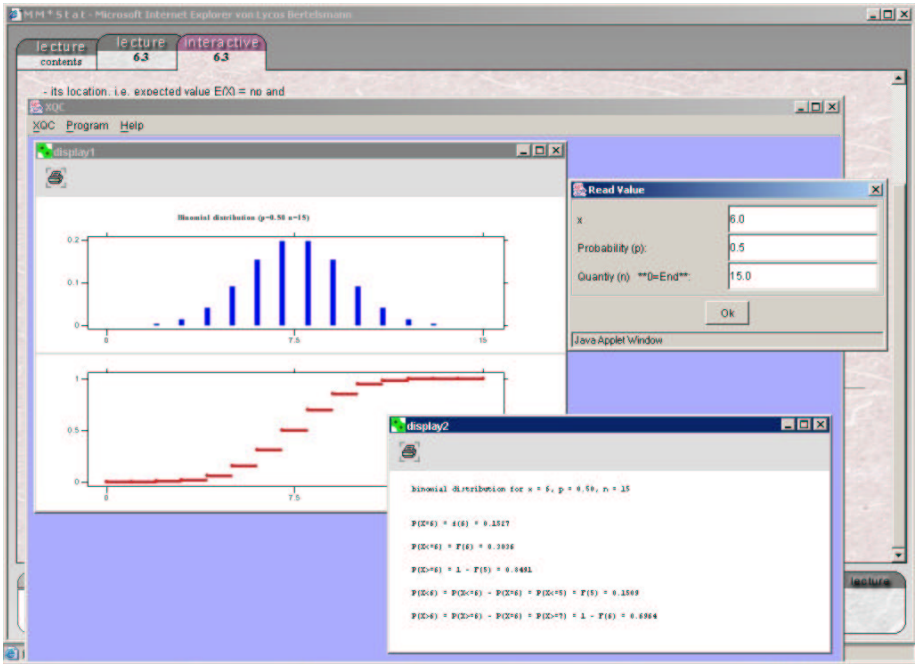

Figure 7: XQC running as part of MM*Stat

The MM*Stat project (Rönz, 2001) is an example of the XQC being integrated into an e-learning environment. MM*Stat is a HTML-based multimedia tool for supporting teaching and learning statistics in its basics. This tool is either available online via the Internet (http://www.md-stat.com) or as a CD version. Beside lecture unit's different kind of examples are offered to help to understand the theoretical content. The XQC offers its features within the interactive examples. These examples allow users to perform examples repeatedly with different data sets or variables or with changed parameters of the statistical method. Figure 7 shows an interactive MM*Stat

example – a Binomial distribution offering the possibility to change the parameter x, probability p and quantity n. As a result the distribution is shown graphically as well as the calculations for the given information.

As mentioned in the introduction of this paper a basic tenet of scientific research is the requirement that research results should be reproducible by other scientists. "Taking care to preserve and document the devilish details of computer programs pays dividends not only in the communication with other scientists, but also for the person conducting the research." (Gentle) Claerbaout et al (2000) give rules and examples of publishing documents allowing for reproducible research. They publish CD-ROMs that contain the text of their work along with a special viewer that makes those research results interactive documents. While reading those documents it is possible to open separate windows for figures in the document that allow for interaction with the code that actually generated the figure.

The XQC offers similar functionality. To realize desired reproducibility in scientific publications the XQC could easily be integrated into PDF or HTML content. Examples of these kinds of documents are available at http://www.xplore-stat.de/ebooks/ebooks.html. Most of the graphical figures contain a link that starts the XQC. Depending on the configuration the client behaves different. It either just executes the underlying XploRe method and rebuilds the figure or it additionally opens the editor frame allowing the reader to verify, change and execute the coding. Publication is not limited to the use of the Internet. A CD-ROM that contains the interactive document could also contain all parts of the XQC/XQS architecture. In this case client and server are running locally after started via an interactive link. In an even more extended version the CD-ROM could contain a Java Runtime Environment (JRE) avoiding the need of a locally installed JRE.

6 XQC among other web based statistical/mathematical solutions

Searching the Internet for web based statistical/mathematical solutions leads to three different approaches:

1. CGI techniques
2. "Standalone" Java applets
3. Java based distributed computing

Using CGI techniques the user enters data or the location of a data file via a CGI interface. A statistical program on the server side calculates and sends back the results to the user. The user will get the results either right away – shown in the browser window or the result will be sent to the user by e-mail. Examples are given by Inoue et al (2001), the MMM project (Günther et al, http://macke.wiwi.hu-berlin.de/mmm/) the Rweb project (http://www.math.montana.edu/Rweb/). The advantage of the CGI approach is the use of an architecture that is similar to the client/server architecture. With the statistical program running on the server side the user can access resources of a powerful computing system as offered by our XQC/XQS approach. The disadvantage of the CGI is the lack of interactivity. A CGI program works as a new individual process against a HTTP request without any communication that takes place between different processes.

Interactivity is an advantage offered by the use of Java applets. Most statistical (standalone) applets available via the World Wide Web have two things in common - they are completely programmed in Java and integrated in one single applet. Therefore, the user has to download the whole program containing the computation algorithm as well as routines for presenting the

results. Examples of such statistical applets are the Internet Statistical Computing Center (http://www.statlets.com) and the WebStat project of the University of South Carolina (http://www.stat.sc.edu/webstat/). The XQC/XQS architecture on the other hand splits the load between the client and the server. The Java client only has to present the output computed by the server. Therefore it can be a relatively slim application.

For calculating just a simple histogram of a small dataset the use of a single Java applet would be an appropriate way. But the more complex a statistical algorithm gets and the larger the datasets are the more difficult is a pure Java implementation of these algorithms and the more load lies on the client computer. Computing a nonparametric time series process that contains about a thousand observations within a single Java applet is hardly possible. The computational load of the XQC/XQS model lies on the server side that can take advantage of a powerful underlying computer architecture. This speeds up the computational process significantly. Due to the communication process between client and server which takes place the XQC might take a little longer for calculating simple statistical problems compared to a pure Java applet. But with increasing complexity the time saved using the server power exceeds the time needed for the communication process.

Extending an existing Java applet with a new statistical method implies a high effort for reprogramming the method. The new method, usually developed using a statistical software package, would have to be reprogrammed in Java. Extending the XQC/XQS model just means to add the new method programmed in XploRe to the server's or to the client's method pool without any reprogramming effort on the client or server side.

Besides our XQC/XQS project there exist other projects using client/server approaches for statistical computing via the Internet. One example is the Jasp project (http://jasp.ism.ac.jp/index-e.html), see Kobayashi et al (2001). Jasp is a statistical system whose language is based on Pnuts (http://javacenter.sun.co.jp/pnuts/). Like the XQC/XQS model the Jasp approach uses the advantages of Java and Java applets respectively to implement a user interface. The user interface - a mixed user interface consisting of a CUI as well as a GUI - is an advantage of the JASP project over the current version of the XploRe Quantlet Client. It only offers a CUI, where knowledge of the programming language XploRe is required to perform statistical computing. But the XQC is not only meant to work like a "conventional" program – the advantage of the XQC is the possibility to customize its behavior via a configuration file. This characteristic offers a way to extend the features of electronically enhanced books (e-books) towards interactive examples. The Jasp approach allows for distributed computing on several servers whereas in the XQC/XQS model a client chooses a certain server that computes the data of this (and possible other) client(s) during the entire session.

Pure web-based client/server approaches suffer from well-known problems of the Internet – the security of data transferred via the Internet, the stability and the speed of the network/modem connection that may represent the bottleneck of the client/server architecture. Encrypting the data could solve the security problem. To take advantage of the server's speed a fast and stable network/modem connection is required for the transport of data and results. The quite fast technical development in this area should help to solve this problem in the future.

References

Blauth, A. (2000), GraphFitI – A program for model selection in graphical chain models, available online at http://www.stat.uni-muenchen.de/~blauth/GraphFitI/graphFitI.html

Buckheit, J., Donoho, D. (1995), WaveLab and Reproducible Research, in Wavelets and Statistics, A. Antoniadis, ed., Springer-Verlag, Berlin, New York.

Clearbout, J., Karrenbach, N., Schwab, M. (2000), Making scientific computations reproducible. Computing in Science & Engineering, Vol. 2, Issue 6, Nov.-Dec. 2000, p.61-67, available online at http://sep.stanford.edu/research/redoc/cip.html

Clearbout, J., Schwab, M., Reproducible electronic documents, available online at http://sep.stanford.edu/research/redoc/

Feuerhake, J., (2001), MD*CRYPT – the XQS/XQC protocol, available online at http://www.md-crypt.com

Gentle, J., Computational Statistics, available online at http://www.galaxy.gmu.edu/~jgentle/cmpstbk/

Günther, O., Müller, R., Krishnan, R., Bhargava, H., Schmidt, P. (1997), MMM: A Web-Based System for Sharing Statistical Computing Modules, 59-68, IEEE Internet Computing

Härdle, W., Klinke, S., Müller, M. (1999), XploRe -The Statistical Computing Environment, Springer-Verlag, New York.

Inoue, T., Asahi, Y., Yamamoto, Y., Yadohisa, H. (2001), A prototype of Data Representation System, Proc. of the ISM symposium "Statistical software in the Internet age", 85-90, Institute of Statistical Mathematics, Tokyo.

Kleinow, T., Lehmann, H. (2001), Client/Server based Statistical Computing, Proc. of the ISM symposium "Statistical software in the Internet age", 1-8, Institute of Statistical Mathematics, Tokyo.

Kobayashi, I., Fujiwara, T., Yamamoto, Y., Nakano, J. (2001), The Language and the Extendibility of the Statistical System Jasp, Proc. of the ISM symposium "Statistical software in the Internet age", 65-73, Institute of Statistical Mathematics, Tokyo.

Mori, Y., Information online at http://www.soci.ous.ac.jp/~mori/topE.htm

Lang, Duncan Temple, (2002), Calling R from Java, available online at http://www.omegahat.org/RSJava/RFromJava.pdf

Rönz, B. (2001), The multimedia-project MM*Stat for teaching statistics, Proc. of the ISM symposium "Statistical software in the Internet age", 27-31, Institute of Statistical Mathematics, Tokyo.