SFB 649 Discussion Paper 2005-010

Working with the XQC

Wolfgang Härdle*
Heiko Lehmann*

* CASE - Center for Applied Statistics and Economics,
Humboldt-Universität zu Berlin, Germany

This research was supported by the Deutsche
Forschungsgemeinschaft through the SFB 649 "Economic Risk".

http://sfb649.wiwi.hu-berlin.de
ISSN 1860-5664

SFB 649, Humboldt-Universität zu Berlin
Spandauer Straße 1, D-10178 Berlin

1 Working with the XQC

Wolfgang Härdle, Heiko Lehmann

1.1 Introduction

An enormous number of statistical methods have been developed in quantitive finance during the last decades. Nonparametric methods, bootstrapping time series, wavelets, estimation of diffusion coefficients are now almost standard in statistical applications. To implement these new methods the method developer usually uses a programming environment he is familiar with. Thus, such methods are only available for preselected software packages, but not for widely used standard software packages like MS Excel. To apply these new methods to empirical data a potential user faces a number of problems or it may even be impossible for him to use the methods without rewriting them in a different programming language. Even if one wants to apply a newly developed method to simulated data in order to understand the methodology one is confronted with the drawbacks described above. A very similar problem occurs in teaching statistics at undergraduate level. Since students usually have their preferred software and often do not have access to the same statistical software packages as their teacher, illustrating examples have to be executable with standard tools.

In general, two statisticians are on either side of the distribution process of newly implemented methods, the provider (inventor) of a new technique (algorithm) and the user who wants to apply (understand) the new technique. The aim of the XploRe Quantlet client/server architecture is to bring these statisticians closer to each other. The XploRe Quantlet Client (XQC) represents the front end - the user interface (UI) of this architecture allowing to access the XploRe server and its methods and data. The XQC is fully programmed in Java not depending on a specific computer platform. It runs on Windows and Mac platforms as well as on Unix and Linux machines.

1.2 Application vs. Applet

The XploRe Quantlet Client can be initiated in two different ways. The way depends on whether the XQC is supposed to run as a standalone application or as an applet embedded within an HTML page. The XQC comes packed in a single Java Archive (JAR) file, which allows an easy usage. This JAR file allows for running the XQC as an application, as well as running it as an applet.

Running the XQC as an **application** does not require any programming skills. Provided that a Java Runtime Environment is installed on the computer the XQC is supposed to be executed on, the *xqc.jar* will automatically be recognized as an executable jar file that opens with the program *javaw*. If the XQC is embedded in a HTML page it runs as an **applet** and can be started right after showing the page.

1.3 Configuration

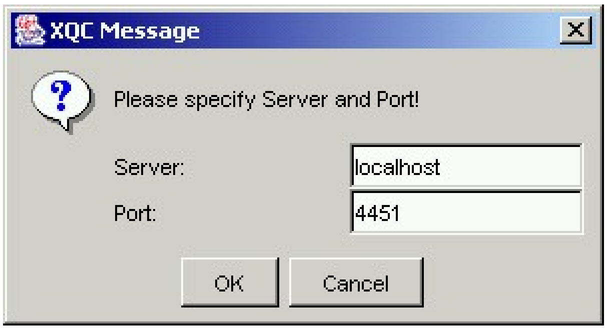
Property files allow configuring the XQC to meet special needs of the user. These files can be used to manage the appearance and behavior of the XQC. Any text editor can be used in editing the configuration files. Generally, the use of all information is optional. In its current version, the XQC works with three different configuration files. In its actual version the XQC works with three different configuration files. The *xqc.ini* file contains important information about the basic setup of the XploRe Quantlet Client, such as server and port information the client is supposed to connect to. It also contains information about the size of the client. This information can be maintained either relative to the actual size of the screen by using a factor or by stating its exact width and height. If this information is missing, the XQC begins by using its default values.

The *xqc_language.ini* allows for 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. If no language file can be found, the client starts with its default setup, showing all menus and messages in English.

The *xqc_methodtree.ini* file finally contains information about the method tree that can be shown as part of the METHOD/DATA window, see section 1.7. A detailed description of the set up of the method tree will be part of section 1.8.

1.4 Getting connected

After starting the XQC the client attempts to access and read information from the configuration files. If no configuration file is used error messages will pop up. If server and port information cannot be found, a pop up appears and enables a manual input of server and port number, as displayed in figure 1.1.



Please specify Server and Port!

Server: localhost

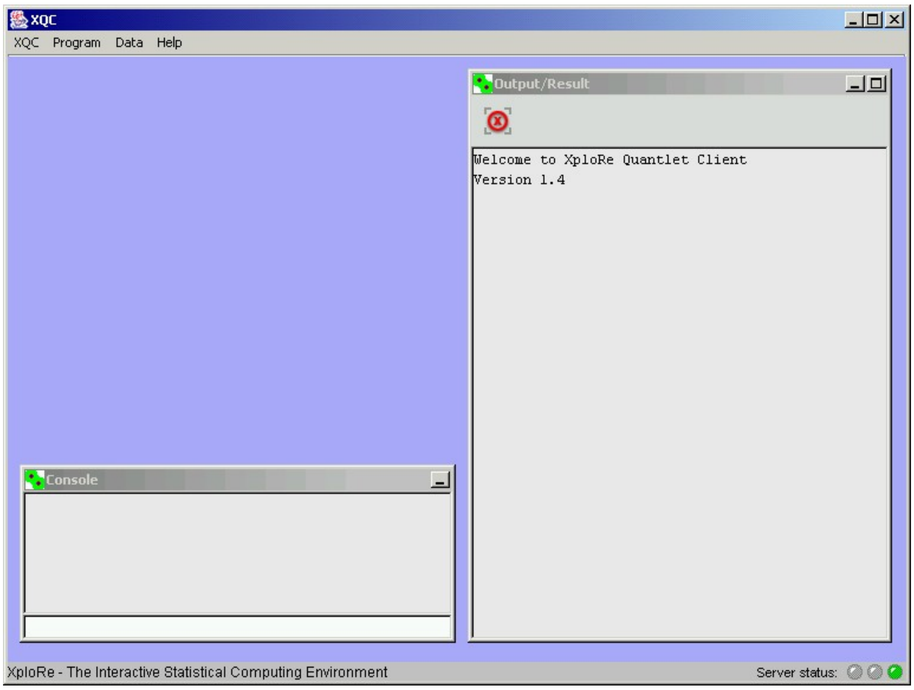
Port: 4451

Figure 1.1: Manual input for server and port number

The screen-shot in figure 1.2 shows the XQC after it has been started and connected to an XploRe server. A traffic light in the lower right corner of the screen indicates the actual status of the server. A green light means the client has successfully connected to the server and the server is ready to work. If the server is busy, computing previously received XploRe code, the traffic light will be set to yellow. A red light indicates that the XQC is not connected to the server.

1.5 Desktop

If no further restrictions or features are set in the configuration file (e.g. not showing any window or starting with executing a certain XploRe Quantlet) the XQC should look like shown in the screen shot. It opens with the two screen components CONSOLE and OUTPUT/RESULT window. The CONSOLE allows for the sending of single-line XploRe commands to the server to be executed immediately. It also offers a history of the last 20 commands sent to the server. To repeat a command from the history, all that is required is



```
Welcome to XploRe Quantlet Client
Version 1.4
```

Figure 1.2: XQC connected and ready to work

a mouse click on the command, and it will be copied to the command line. Pressing the ‘Return’ key on the keyboard executes the XploRe command.

Text output coming from the XploRe server will be shown in the OUTPUT/RESULT window. Any text that is displayed can be selected and copied for use in other applications - e.g. for presentation of results within a scientific article. At the top of the screen the XQC offers additional functions via a menu bar. These functions are grouped into four categories. The **XQC** menu contains the features *Connect*, *Disconnect*, *Reconnect* and *Quit*.

Depending on the actual server status not every feature is enabled - e.g. if the client is not connected (the server status is indicated by a red traffic light) it does not make sense to disconnect or reconnect, if the client is already connected (server status equals a green light) the connect feature is disabled.

1.6 XploRe Quantlet Editor

The **Program** menu contains the features *New Program*, *Open Program (local)...* and *Open Program (net)....* *New Program* opens a new and empty text editor window. This window enables the user to construct own XploRe Quantlets.

The feature *Open Program (local)* offers the possibility of accessing XploRe Quantlets stored on the local hard disk drive. It is only available if the XQC is running as an application or a certified applet. Due the Java sandbox restrictions running the XQC as an unsigned applet, it is not possible to access local programs.

If the user has access to the Internet the menu item *Open Program (net)* can be useful. This feature allows the opening of Quantlets stored on a remote Web server. All it needs is the filename and the URL address at which the file is located.

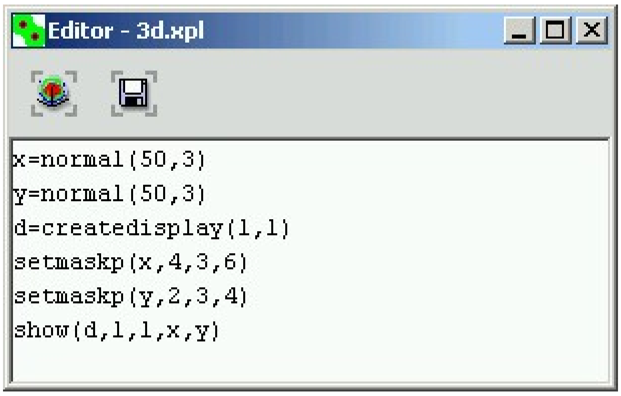
Figure 1.3 shows a screen shot of the editor window containing a simple XploRe Quantlet. Two icons offer actions on the XploRe code:

- - Represents the probably most important feature - it sends the XploRe Quantlet to the server for execution.
- 💾 - Saves the XploRe Quantlet to your local computer (not possible if running the XQC as an unsigned applet).

The Quantlet shown in figure 1.3 assigns two three-dimensional, standard normal distributions to the variables x and y. The generated data are formatted to a certain color, shape and size using the command `setmaskp`. The result is finally shown in a single display.

1.7 Data Editor

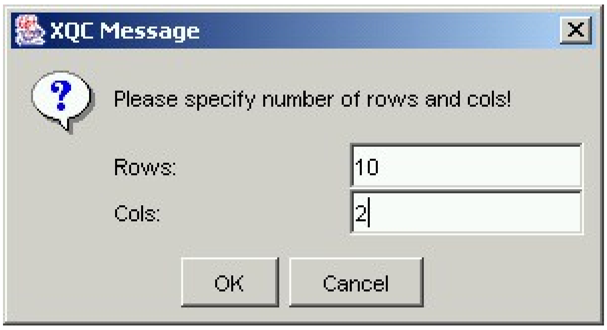
The **Data** menu contains the features *New Data...*, *Open Data (local)...*, *Open Data (net)...*, *Download DataSet from Server...* and *DataSets uploaded to Server*.



```
x=normal(50,3)
y=normal(50,3)
d=createdisplay(1,1)
setmaskp(x,4,3,6)
setmaskp(y,2,3,4)
show(d,1,1,x,y)
```

Figure 1.3: XploRe Editor window

New Data can be used to generate a new and empty data window. Before the data frame opens a pop-up window as shown in figure 1.4 appears, asking for the desired dimension - the number of rows and cols - of the new data set. The XQC needs this information to create the spreadsheet. This definition does not have to be the exact and final decision, it is possible to add and delete rows and columns later on.



Please specify number of rows and cols!

Rows: 10

Cols: 2

Figure 1.4: Dimension of the Data Set

The menu item *Open Data (local)* enables the user to open data sets stored on the local hard disk. Again, access to local resources of the user's computer is only possible if the XQC is running as an application or a certified applet. The file will be interpreted as a common text format file. Line breaks within the file are considered as new rows for the data set. To recognize data belonging to a certain column the single data in one line must be separated by either using a “;” or a “tab” (separating the data by just a “space” will force the XQC to open the complete line in just on cell).

Open Data (net) lets the user open a data set that is stored on a Web server by specifying the URL.

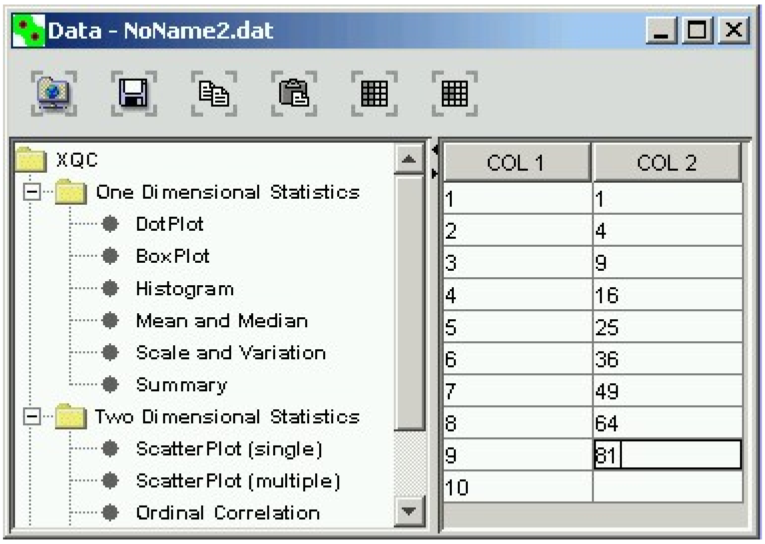
The menu item *Download DataSet from Server* offers the possibility to download data from the server. The data will automatically be opened in a new method and data window, offering all features of the method and data window (e.g. applying methods, saving, . . .) to them.

A helpful feature especially for research purposes is presented with the menu item *DataSets uploaded to Server*. This item opens a window that contains a list of objects uploaded to the server using the data window or the console. Changes of these objects are documented as an object history. Due to performance reasons only uploaded data and actions on data from the CONSOLE and the TABLE MODEL are recorded.

The appearance of the data window depends on the settings in the configuration file. If a method tree is defined and supposed to be shown, the window shows the method tree on the left part and data spreadsheet on the right part of the frame. If no method tree has been defined, only the spreadsheet will be shown. The method tree will be discussed in more detail in section 1.8. Figure 1.5 shows a screen shot of the combined data and method frame.

Icons on the upper part of the data and method window offer additional functionalities:

- 🖥 - If columns or cells are selected - this specific selection, otherwise the entire data set can be uploaded to the server with specifying a variable name.
- 💾 - Saves the data to your local computer (not possible if running the XQC as an unsigned applet).
- ⎘ / 📋 - Copy and paste.



COL 1	COL 2
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	

Figure 1.5: Combined Data and Method Window

- ▦ / ▦ - Switches the column or cell selection mode on and off. Selected columns/cells can be uploaded to the server or methods can be executed on them.

The spreadsheet of the data and method window also offers a context menu containing the following items:

- Copy
- Paste
- No Selection Mode - Switches OFF the column or cell selection mode.
- Column Selection Mode - Switches ON the column selection mode.
- Cell Selection Mode - Switches ON the cell selection mode.
- Set Row as Header Line
- Set column Header

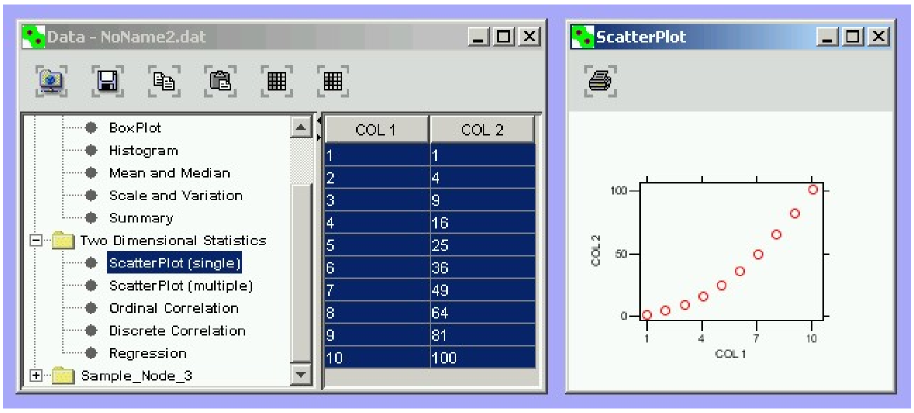
- Delete single Row
- Insert single Row
- Add single Row
- Delete single Column
- Add single Column

Most of the context menu items are self-explaining. However, there are two items that are worth taking a closer look at - *'Set Row as Header Line'* and *'Set column Header'*. The spreadsheet has the capability to specify a header for each column. This information can be used within XploRe Quantlets to name the axis within a plot, making it easier for the user to interpret graphics. A more detailed description is included in section 1.8. Default values for the headers are COL1, COL2, ... as shown in figure 1.6. Naming a single column can be performed using the menu item *'Set column Header'*. The name has to be maintained within the pop up window that appears right after choosing this menu item. It can also be used to change existing column headers. The spreadsheet also offers the possibility to set column headers all at once. If the data set already contains a row with header information - either coming from manual input or as part of an opened data set - these row can be set as header using the menu item *'Set Row as Header Line'*. The row with the cell that is active at that time will be cut out of the data set and pasted into the header line.

Setting the header is also possible while opening a data set. After choosing the data, a pop up asks whether or not the first row of the data set to be opened should be used as the header. Nevertheless, the context menu features just described above are of course still available, enabling the user to set or change headers afterwards.

Working with the XQC's method and data window does not require any XploRe programming knowledge. All it requires is a pointing device like the mouse. Applying, for example, the scatter-plot-method on the two columns would only mean to

- Switch on the column selection mode
- Mark both columns
- Mouse click on the method "Scatter Plot"



COL 1	COL 2
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

Figure 1.6: Working with the Data and Method Window

Result will be a plot as shown in figure 1.6. As stated above, the selected area can also be uploaded to the server using the icon in order to be used for further investigation. This new variable can be used within XploRe Quantlets written using the EDITOR window or manipulated via the CONSOLE.

1.8 Method Tree

The METHOD TREE represents a tool for accessing statistical methods in an easy way. Its setup does not require any Java programming skills. All it needs is the maintenance of two configuration files.

Settings maintained within the *xqc.ini* file tell to the XQC whether there will be a method tree to be shown or not and where to get the tree information from. The client also needs to know where the methods are stored at. The `MethodPath` contains this information. Path statements can either be absolute statements or relative to the directory the XQC has been started in. For relative path information the path must start with `XQCROOT`. The settings in the example below tell the client to generate a method tree by using the file *xqc_methodtree.ini* with the XploRe Quantlets stored in the relative subdirectory `xqc_quantlets/`.

```
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 children, the method to be executed and its 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.

Example

The following example shows how to set up a simple method tree. First of all, we choose XploRe Quantlets used within this e-book that we want to be part of the method tree. The aim of the Quantlet should be to generate graphics from selected data of the data spreadsheet or to just generate text output. Before being able to use the Quantlets within the method tree, they have to be 'wrapped' in a procedure. The name of the procedure - in our case for example 'STFstab08MT' - has to equal the name of the saved XploRe file. Our example Quantlet *STFstab08MT.xpl* is based on the original Quantlet *STFstab08.xpl* used in chapter **??**. The procedure must further have two parameters:

- `data` - Used for passing the selected data to the XploRe Quantlet.
- `names` - Contains the names of the selected columns taken from the header of the spreadsheet.

It might also be necessary to make some minor adjustments within the Quantlet in order to refer to the parameter handed over by the XQC. Those changes depend on the Quantlet itself.

```
library ("graphic")
proc() = STFstab08MT(data, names)
     ...
endp
```

Figure 1.7: STFstab08MT.xpl

The XploRe coding within the procedure statement is not subject to any further needs or restrictions.

Once we have programmed the Quantlet it needs to be integrated into a method tree. For this purpose we define our own configuration file - *xqc_methodtree_STF* - with the following content shown in figure 1.8.

```
Node_1 = Stable Distribution

     Node_1.1 = Estimation
        Child_1.1.1 = stabreg.xpl|Stabreg
        Child_1.1.2 = stabcull.xpl|Stabcull
        Child_1.1.3 = stabmom.xpl|Stabmom

     Node_1.2 = Examples
        Child_1.2.1 = STFstab08.xpl|STFstab08
        Child_1.2.2 = STFstab09.xpl|STFstab09
        Child_1.2.3 = STFstab10.xpl|STFstab10
```

Figure 1.8: sample_tree.ini

We create a node calling it ‘Estimation’. Below this first node we set up the Quantlets *stabreg.xpl*, *stabcull.xpl* and *stabmom.xpl*. A second node - ‘Examples’ contains the Quantlets *STFstab08.xpl*, *STFstab09.xpl* and *STFstab10.xpl*. The text stated right beside each Quantlet (separated by the ‘|’) represents the text we would like to be shown in the method tree.

Now that we have programmed the XploRe Quantlet(s) and set up the method tree we still need to tell the XQC to show our method tree upon opening data sets.

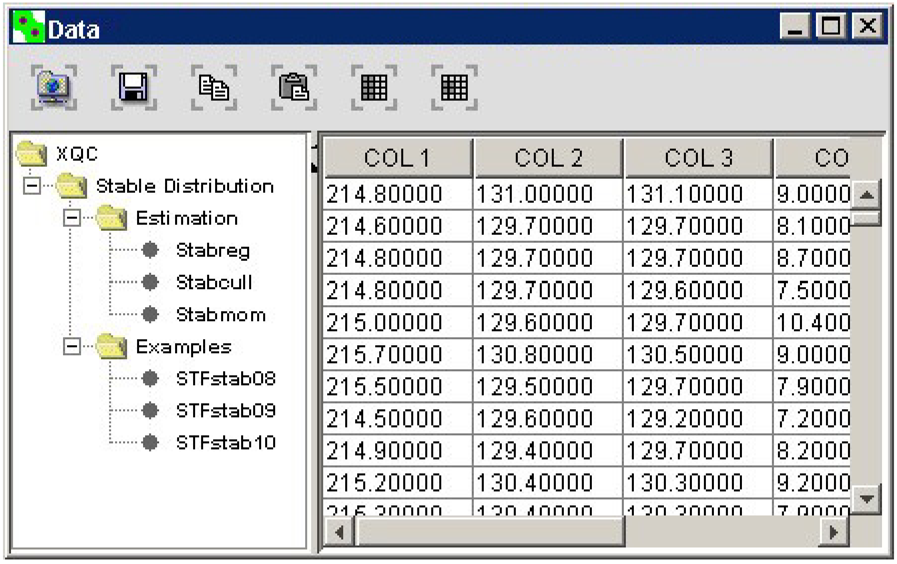
The settings as shown in figure 1.9 tell the XQC to show the method tree that is set up in our *xqc_methodtree_STF.ini* file and to use our XploRe Quantlet

```
...

ShowMethodTree     = yes
MethodTreeIniFile = xqc_methodtree_STF.ini
MethodPath         = XQCROOT/xqc_quantlets/

...
```

Figure 1.9: Extract of the xqc.ini

stored in a subdirectory of the XQC itself.

Our method tree is now ready for finally being tested. Figure 1.10 shows a screen-shot of the final result - the method tree, set up above.



COL 1	COL 2	COL 3	CO
214.80000	131.00000	131.10000	9.0000
214.60000	129.70000	129.70000	8.1000
214.80000	129.70000	129.70000	8.7000
214.80000	129.70000	129.60000	7.5000
215.00000	129.60000	129.70000	10.400
215.70000	130.80000	130.50000	9.0000
215.50000	129.50000	129.70000	7.9000
214.50000	129.60000	129.20000	7.2000
214.90000	129.40000	129.70000	8.2000
215.20000	130.40000	130.30000	9.2000

Figure 1.10: Final result of our tree example

1.9 Graphical Output

The previous sections contain some examples of graphical output shown within a display. The XQC's displays do not show only the graphical results received from the XploRe server. Besides the possibility to print out the graphic it offers additional features that can be helpful for investigating data - especially for three-dimensional plots. Those features can be accessed via the display's context menu. Figure 1.11 shows three-dimensional scatter plots for three characteristics (lower inner frame vs. diagonal vs. upper inner frame) of a collection of 200 Swiss bank notes - containing 100 notes that are actually counterfeits.

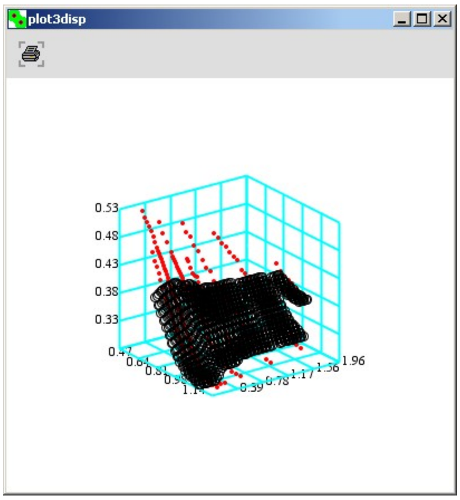


Figure 1.11: Scatter plot for characteristics of Swiss bank notes

The scatter plot shows genuine and counterfeit bank notes using a different look and color. As a default setting the x-y-dimension will be shown. The scatter plot suggests the existence of two clusters - genuine and counterfeits, although some bank notes are overlapping when only the characteristics 'lower inner frame' and 'diagonal' are considered. For a more detailed inspection three-dimensional plots can be rotated by using a pointing device such as a mouse (with the left mouse-button pressed) or by using the keyboards arrow-keys. Figure 1.12 shows the same plot as before - it has just been rotated by some degrees. Now, also considering the characteristic ‘upper inner frame’, the clusters of the data are even better to identify. Nevertheless, there still is one data point (represented by a circle) that lies among the data of the other cluster (represented by stars). For further research it would be helpful to know which data point it is. Of course the user could compare the characteristics of all data points to find the one. Here the XQC’s display offers a feature to show the point’s coordinates. This feature can be accessed via the display’s context menu. ‘Showing coordinates’ is not the only option. The user could also switch between the three dimensions - ‘Show X$\tilde{\text{Y}}$’, ‘Show X$\tilde{\text{Z}}$’ and ‘Show Y$\tilde{\text{Z}}$’.

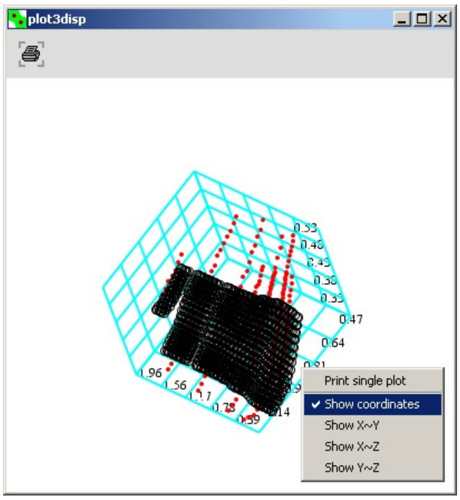
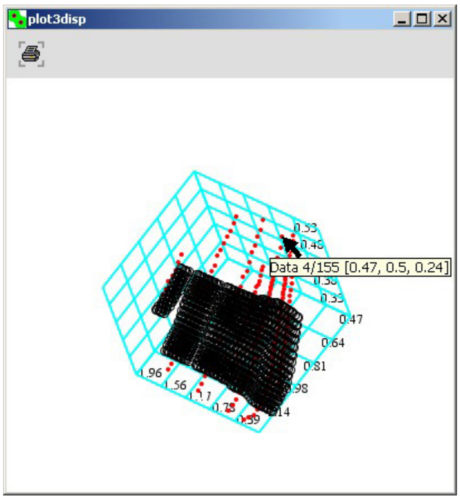


Figure 1.12: Rotating scatter plot showing the context menu

After the ‘Showing coordinates’ has been chosen all it needs is to point the mouse arrow on a certain data point in order to get the information. Figure 1.12 shows the details “1/70 [8.0, 11.2, 139.6]” for the data point that seems to be part of the ‘wrong’ cluster. “1/70” implies that we deal with data point number 70 of the first data set printed within the display. The information “first” is important since there could be more than just one data set shown in the display. The numbers within the brackets “[. . .]” are the actual characteristics of that data point.

The possibility to configure the XploRe Quantlet Client for special purposes as well as its platform independence are features that recommends itself for



Data 4/155 [0.47, 0.5, 0.24]

Figure 1.13: Showing the coordinates of a data point

the integration into HTML and PDF contents for visualizing statistical and mathematical coherences as already shown in this book.

SFB 649 Discussion Paper Series

For a complete list of Discussion Papers published by the SFB 649, please visit http://sfb649.wiwi.hu-berlin.de.

001 "Nonparametric Risk Management with Generalized Hyperbolic Distributions" by Ying Chen, Wolfgang Härdle and Seok-Oh Jeong, January 2005.
002 "Selecting Comparables for the Valuation of the European Firms" by Ingolf Dittmann and Christian Weiner, February 2005.
003 "Competitive Risk Sharing Contracts with One-sided Commitment" by Dirk Krueger and Harald Uhlig, February 2005.
004 "Value-at-Risk Calculations with Time Varying Copulae" by Enzo Giacomini and Wolfgang Härdle, February 2005.
005 "An Optimal Stopping Problem in a Diffusion-type Model with Delay" by Pavel V. Gapeev and Markus Reiß, February 2005.
006 "Conditional and Dynamic Convex Risk Measures" by Kai Detlefsen and Giacomo Scandolo, February 2005.
007 "Implied Trinomial Trees" by Pavel Čížek and Karel Komorád, February 2005.
008 "Stable Distributions" by Szymon Borak, Wolfgang Härdle and Rafal Weron, February 2005.
009 "Predicting Bankruptcy with Support Vector Machines" by Wolfgang Härdle, Rouslan A. Moro and Dorothea Schäfer, February 2005.
010 "Working with the XQC" by Wolfgang Härdle and Heiko Lehmann, February 2005.

SFB 649, Spandauer Straße 1, D-10178 Berlin
http://sfb649.wiwi.hu-berlin.de

This research was supported by the Deutsche Forschungsgemeinschaft through the SFB 649 "Economic Risk".