Klinke, Sigbert

Working Paper

Statistical user interfaces

Papers, No. 2004,35

Provided in Cooperation with:
CASE - Center for Applied Statistics and Economics, Humboldt University Berlin

Suggested Citation: Klinke, Sigbert (2004) : Statistical user interfaces, Papers, No. 2004,35, Humboldt-Universität zu Berlin, Center for Applied Statistics and Economics (CASE), Berlin

This Version is available at:
https://hdl.handle.net/10419/22208

Statistical user interfaces

Sigbert Klinke

Humboldt-Universität zu Berlin, School of Business and Administration, Institute of Statistics and Econometrics, Spandauer Strasse 1, 10178 Berlin, Germany
sigbert@wiwi.hu-berlin.de

1 Introduction

A *statistical user interface* is an interface between a human user and a statistical software package. Whenever we use a statistical software package we want to solve a specific statistical problem. But very often at first it is necessary to learn specific things about the software package.

Everyone of us knows about the "religious wars" concerning the question which statistical software package/method is the best for a certain task; see Marron (1996) and Cleveland and Loader (1996) and related internet discussions. Experienced statisticians use a bunch of different statistical software packages rather than a single one; although all of the major companies (at least the marketing departments) tell us that we only need their software package.

Why do these wars, not only concerning statistical software packages, evolve? One of the reasons is that we need time to learn about the software package besides learning the statistical methodology. And the more time we need to learn to use the software package, the more we are defending "our" software package. But if we need to spend a lot of time for learning to use a statistical software package, then the question, whether this software package really has a good user interface, arises?

The basic problem is that the development of statistical software is started by experts of statistical methodology. Since they have to have a deep inside in their special fields, most of them have a very limited view to problems of other users. We generally do not consider the sex of the users, the ethnic or cultural background, the statistical knowledge or other factors when we *create* a user interface.

Thus the important questions we have to answer when we *design* a user interface are: What does the user want to do with this software package? And how can he do it effectively?

Fortunately, during years of development of software packages, we have collected a lot of experience about human behavior and specific differences

because of sex, ethnic or cultural background and so on. In the book of Shneiderman (1998) a lot of rules have been collected which should help the software developers to avoid the worst problems. But the best way for developing a user interface is a development cycle of designing, testing, redesigning, testing, redesigning, ... This will take a lot of time and money, but redesigning the basic components of a software package at late development will be much more expensive or even impossible.

In this chapter only a subset of all statistical software packages, namely DataDesk 6.0, GGobi 0.99, R 1.7.1, SPSS 11.0 (English version), SYSTAT 10, XploRe 4.6 and Mathematica 4.3 will be used for illustrating purposes (see also the section "Web references"). It covers a wide range of different statistical software packages.

In all statistical software packages we can find errors in the user interface design. User interface design is not a exact science as statistics, but it relies heavily on the way how we perceive information and how we react to it. That includes that in every design we will make errors before we can learn to avoid them in future. Therefore a lot of theories are available, partially explanatory, partially predicting, which should help us to design user interfaces.

2 The golden rules and the ISO norm 9241

Complex statistical tasks require more and more complex statistical programs. In consequence more complex user interfaces are needed to be developed. Software developers recognized that common rules exist in simplifing the use of software systems. Shneiderman (1998) published a summary of these rules known as the "golden rules" of user interface design:

1. *Achieve consistency*
   The first rule is the one which is most often violated, especially when teams of people work together. Users expect that in similar situations the software behaves similarly and requires the same actions from the user.
2. *Use shortcuts*
   Beginners need a comfortable clear structured way to accomplish their task, but power users of a software package want to do their work as quickly as possible.
3. *Give informative feedback*
   Users need to have a feedback on their actions. The amount of the feedback depends on the action and the user's experience. Frequent actions require only short answers whereas rare actions require more extended answers. Beginners need more feedback whereas power users may just need acknowlegdement that the task is finished.
4. *Design closed actions*
   Actions should have a clear structure with a start and a well-defined end. This holds especially for dialogs and forms.

5. *Offer error prevention and easy error handling*
   Software should not support erroneous input from the user and provide default values. The user should be able to recover easily from errors. If a user can revert his actions easily then this will increase his trustworthiness in the software package.
6. *Support user control*
   Users prefer to initiate actions in a software package. If a user believes that he only reacts to the system he will experience a control loss.
7. *Reduce memorization*
   Humans can only remember seven plus minus two information bits in their short term memory (Miller, 1956). Extensive memorization to handle a software package should be avoided.

A formalization of the rules can be found, partially in very detailed instruction, in the ISO (International Standardization Organization) norm 9241. The norm itself, which distinguishes between requirements and recommendations, consists of 17 parts:

1 General introduction	10 Dialogue principles
2 Guidance on task requirements	11 Usability statements
3 Visual display requirements	12 Presentation of information
4 Keyboard requirements	13 User guidance
5 Workstation layout and postural requirements	14 Menu dialogs
6 Environmental requirements	15 Command dialogs
7 Display requirements with reflections	16 Direct manipulation dialogs
8 Requirements for displayed colors	17 Form-filling dialogs
9 Requirements for non-keyboard input devices	

3 Development of statistical user interfaces

In the past we have seen the development of software according to new concepts in computer science. From the end of the 1960s / beginning of the 1970s when computers became available till now, we can distinguish several phases. In the beginning we had non-interactive, batch oriented software packages, e.g. SAS and SPSS. The idea of incremental programming and interaction lead to systems like PRIM-9 (Tukey et al., 1973, 1974) or programming languages like BASIC. Another paradigm was that the notation used should be compact, like in languages as in APL or C. The availability of personal computers with window systems introduced graphical user interface (GUI) in contrast to command line interfaces (CLI) also to statistical software packages. As mentioned in the interview with J.E. Gentle (Härdle, 2004) statistical software packages nowadays should come with both interfaces. During the last fifteen years we

saw that team programming, reusability of software, network/internet computing and access to databases had their impact on programming languages (ADA, C++, Java, SQL) as well as on statistical software packages like S/S-Plus, R, XploRe, Jasp, etc.

Before we start to develop a statistical software package and a user interface (GUI or CLI), we should think about the kind of problems a user may have:

1. A user could formulate an inadequate goal, e.g. using Excel for semi-parametric modeling.
2. A user could not find the right tool/method, since the developer uses inappropriate labels, e.g. the command `paf` in XploRe should better be named `selectrows`.
3. A user could not be able to find or execute a specific method, e.g. in a statistical programming language with a lot of commands and macros, he could loose the overview. For example, languages like XploRe or R consist of a lot of commands, macros and packages.
4. The feedback from the software package to a user action could be inappropriate or misleading, e.g. the error message `syntax error`.

The first problem can not be solved with a better interface design, but so can the latter three (Franzke, 1995). We have two ways to solve them: either we make a better user interface or the user has to spend a lot of time for learning about the interface. One of the most time consuming design error is a subtle inconsistency, for example if the parameter orders for nearly identical actions, either in formulas for GUIs or commands in CLIs, are different. The user will always loose time to look up these subtle differences.

Obviously we can not develop one user interface for all users. The slogan *Know your user* (Hansen, 1971) in detail (statistical background knowledge, cultural background, etc.) is an important building block to the success of a (statistical) software package. We can distinguish three types of users: *novice users* who need to execute a small set of simple exercises and need an informative feedback from the software package. *Periodic users* who need support for forgotten execution sequences and need to know how to extend the current software package. But they usually need only a short feedback to an executed task. A *power user* is mainly interested in fast answers and needs only very limited feedback. Some statistical software packages, XploRe and R offer even multiple GUIs for different types of users.

However, basic guidelines for all user types are:

1. consistency in the appearance
2. effective information control by the user
3. minimal memorization and minimal data entry by the user
4. flexible adaption of the data display
5. compatibility between data entry and output

3.1 Graphical user interfaces

For novice users it is clear that they prefer software with GUIs (see Temple, Barker and Sloane, Inc., 1990), but for power users this is not quite clear, see Ulich et al. (1991). There are some general drawbacks of GUIs:

1. They need valuable screen space for menus, icons, windows etc.
2. There is no consistent set of menus, icons and windows. We have to relearn them for each software package.

A look at Fig. 1 shows the entry screens of different statistical software packages. Here we find all elements forming a modern GUI: menu bar, toolbar(s) and multiple windows. Note that a *statistical* user interface is more than a GUI: design of the programming language, help system, basically every aspect in the interface between a user and a statistical software package.

Some packages try to help a novice user with his next task. SPSS opens, after starting the program, a dialogue box to load a dataset (see Fig. 1(b)). For R, which can load all data objects from previous sessions automatically, such feature is not necessary.

3.2 Toolbars

Although toolbars play a more and more important role in software nowadays, we immediately notice the sparse toolbars (see Fig. 2), due to the fact that we have no common set of icons. For example GGobi and DataDesk do not offer any toolbar, XploRe, SPSS and R offer only toolbars related to standard tasks, like opening, saving and printing programs, images etc. and specialized software functions. Among the considered programs only SYSTAT offers toolbars for standard statistical graphics (histogram, pie chart, boxplot, scatterplot and scatterplot matrices) and tasks (descriptive statistics, two-way-tables, two-sample t-test, ANOVA, correlations, linear regression, clustering and non-linear modeling). But to learn the meaning of the icons may take some time.

3.3 Menus

The first parts of a software package we use are the menu bar, the menus and the menu items. Menus can give a clear structure to the methods/actions of a software package. Liebelt et al. (1982) have found that a proper organization of the menu reduces the error rate to about 50%. Most statistical software packages have a very clear separation of the tasks in the menu bar (see Fig. 2).

It might be distracting if the position of the menu items changes (Mitchell and Shneiderman, 1989). For unused menu items (not applicable tasks in the current situation) it is preferable if they are grayed out and do not vanish from the menu. Statistical software packages behave very differently. The menu bar

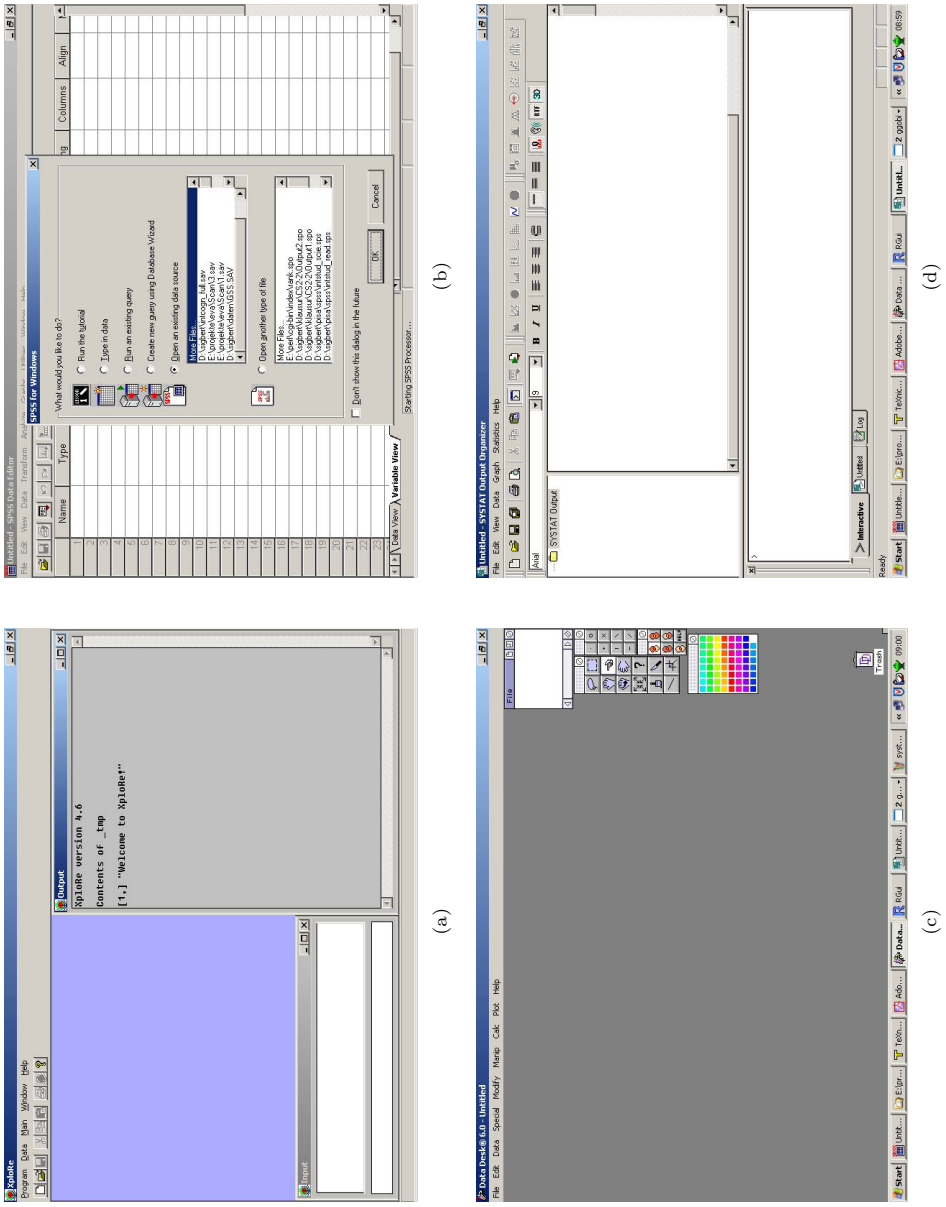


Fig. 1. Entry screens of the statistical software packages, (a) XploRe, (b) SPSS, (c) DataDesk and (d) SYSTAT.

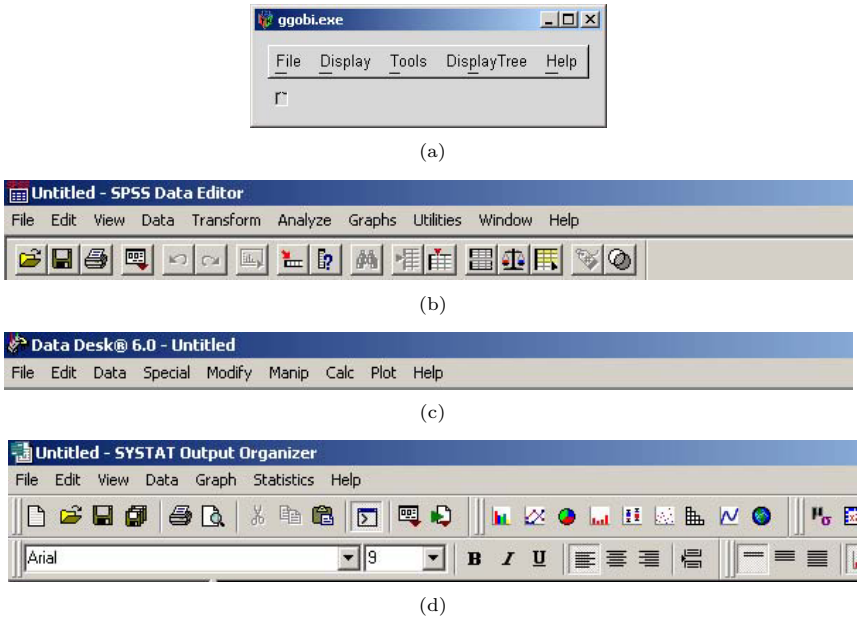


Fig. 2. Menu bars and toolbars of the main windows of (a) GGobi, (b) SPSS, (c) DataDesk and (d) SYSTAT.

in XploRe and R changes heavily depending on the active window which can be very disturbing to the user. It would have been better to attach an appropriate menu to each window. Also in GGobi the menu changes depending on the active window: compare Fig. 4(b) to Fig. 2(a). Nevertheless this is less disturbing to the user because additional menus appear only once in the menu bar and heavy changes take place in the items of the `Display` menu which are invisible to the user. The menu `Display` is empty after starting GGobi, but filled when a dataset is loaded.

If we create a menu hierarchy we basically have two possibilities to organize them: a small, deep hierarchy or a broad, flat hierarchy. Norman and Chin (1988) found that broad, flat hierarchies lead to a better user performance. Most software packages follow this concept intuitively, none of the software packages has a menu depth larger than four.

Several orders of menu items within menus are possible: by time, by numbering, by alphabet, by importance or by function. Card (1982) found that an alphabetical order of menu items is better than a functional order. McDonald et al. (1983) showed that the advantage of the alphabetical order is lost if each menu item consists of a one line definition rather than of one meaningful known word. Nowadays all statistical software packages prefer a functional or-

der by separating the menu items with horizontal lines into menu item groups. But within a menu item group the order is unclear.

To achieve consistency within a menu system, the same terms should be used. If we use one word items then they should be clearly separable, like "insert" and "delete". The exact difference between "change" and "correct" is unclear. Cyclic menus, which means we can achieve the same task by different ways through the menu hierarchy, should be avoided. Users become unsure where to find a specific action; the same problem is well known from the World Wide Web.

The approach to put the most used menu items at the top and suppress the others, may irritate the user. The irritation occurs not with the most used items, but with the items which are used less often. Their position appears to be more or less randomly. Thus, Sears and Shneiderman (1994) found that bringing only the most used items to the top of the menu is an effective technique.

For power users shortcuts, e.g. through keyboard sequences or toolbars, are very important. Often used menu items should get short shortcuts, e.g. Ctrl+O for open a data set, whereas rarely used shortcuts can have longer keyboard sequences. Most statistical software packages offer only the standard shortcuts coming from the Windows operating system; only GGobi offers shortcuts for different view modes. Unfortunately, we have no common sets of shortcuts for a (statistical) task. We have not even a common naming convention for menus, e.g. the statistical tasks are in the `Calc` menu in DataDesk, in the `Statistics` menu in SPSS and in the `Analyze` menu in SYSTAT.

For an effective menu system design it is helpful to log the user choices and to analyze them for improvements.

3.4 Forms and dialog boxes

The use of menus leads to another form of interaction with the user: forms and dialog boxes. Basically they should

- have a meaningful title
- use a consistent and for the user well known terminology
- group information in a consistent and meaningful way
- minimize mouse movement and jump from one item to another item in a useful way
- support error prevention
- allow for fast error correction
- provide default values, if possible
- indicate optional values clearly
- inform when the dialog or form has enough information

Here, a very good example is SPSS. See as example in Fig. 3 four out of six steps for reading ASCII data into SPSS. The six dialog boxes are grouped in information about:

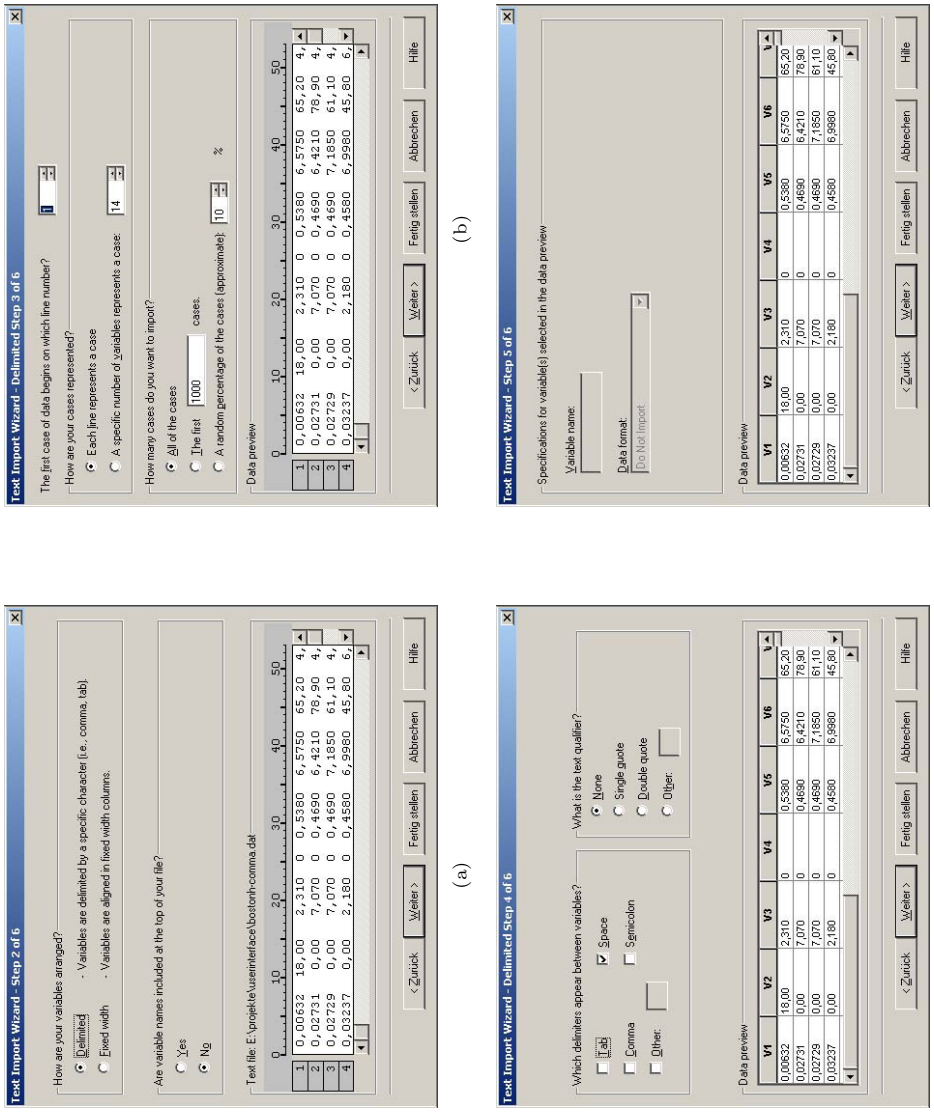


Fig. 3. Four out of six steps of reading ASCII data in SPSS. They provide a very clear, intuitive interface even for unexperienced users for reading data.

1. reuse of old format
2. information about the arrangement of the variables
3. information about the arrangement of the cases
4. separation between single data
5. variable formats and names
6. saving the defined format

The forms always provide default values, show the consequence of changing a value in the bottom and allow easy navigation between forms forward and backward. We can cancel the whole operation or finish it at any time. The last form gives a clear signal when we are finished. The SPSS developers have designed all these forms and dialogs very carefully.

However, we have to keep in mind that pure statistical programming languages, like R or XploRe, will have to incorporate forms and dialog boxes in their programming language. This turns out to be a difficult task.

3.5 Windows

Usually statistical software packages use different windows to visualize data and output. Fig. 4 shows a scatterplot of the variables "percentage of lower status people" and "median houseprice" of the Boston Housing data (Harrison and Rubinfeld, 1978). We easily find that the software packages have different methods to handle output. SPSS and SYSTAT have a special independent output window for text and graphic output. DataDesk, R (by default) and XploRe use the multiple document interface (MDI) coming with the Windows operating system. Actually R allows to switch between different types of handling windows (MDI/SDI). GGobi creates a complete set of independent windows.

In GGobi and DataDesk the data in the windows is linked (see Fig. 5(a)). Thus interactive brushing is very easy.

A problem of some statistical software packages is that the user can easily create a lot of windows, see in Fig. 4 and even worse in Fig. 5 for DataDesk. But a large number of windows can easily irritate the user. Statistical software packages have tried to overcome this problem with different approaches: having separate graphic types, for example the scatterplotmatrix in SPSS or trellis displays in R; XploRe has a datatype for a graphical display which consists of single plots. The idea is always the same: statistical information that belongs together should be in one window. Another strategy is a virtual desktops (see the software package VanGogh in Keller, 2003) as we find them under Linux GUIs.

Power users prefer full-screen views (Bury et al., 1985, see). Note that in Fig. 5 we tried to maximize the size of the graphics in R, SPSS and XploRe. SPSS and SYSTAT follow partially such a strategy with separating clearly between spreadsheet presentation of data and variables and output results. But

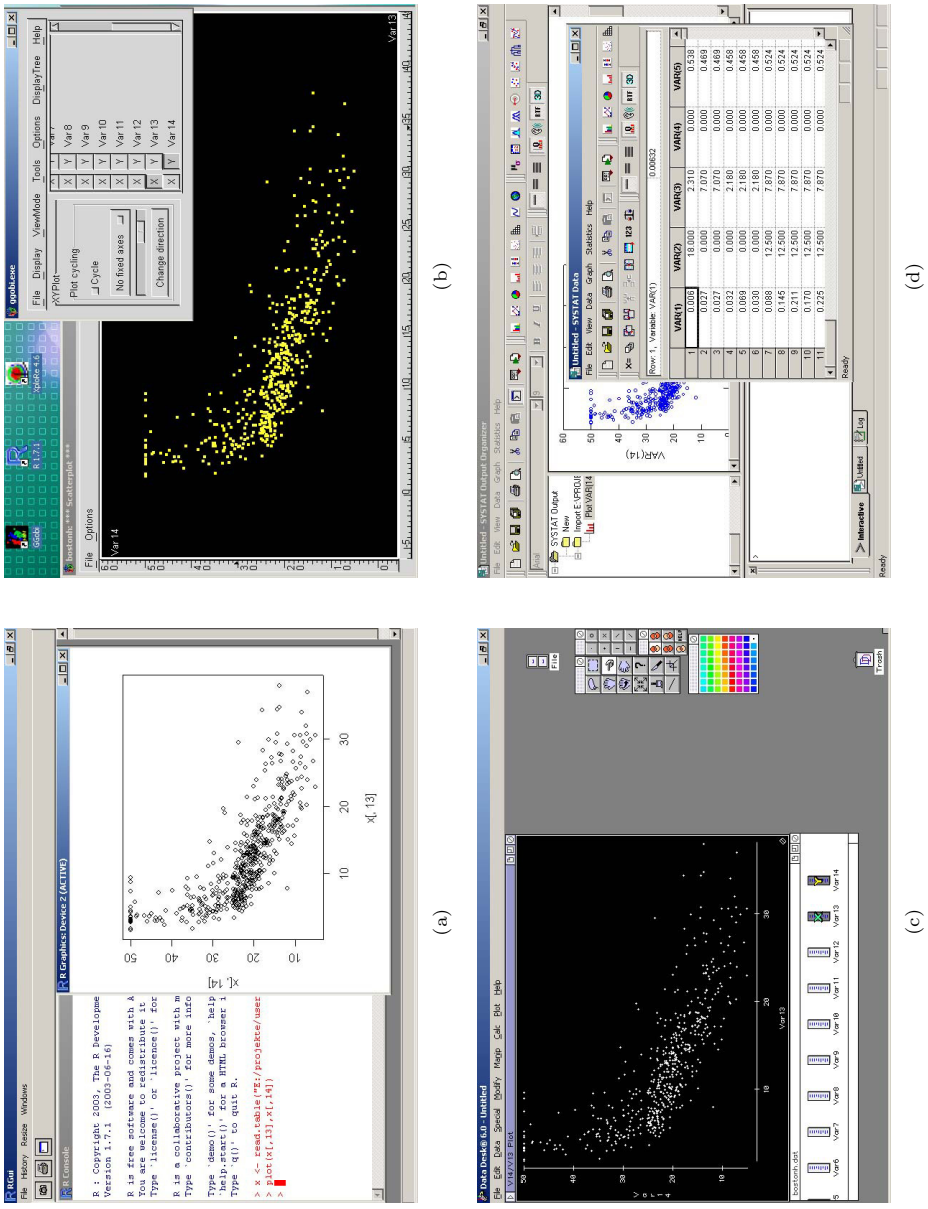


Fig. 4. Scatterplot of the variables "percentage of lower status people" and "median houseprice" of the Boston Housing data in (a) R, (b) GGobi, (c) DataDesk and (d) SYSTAT.

Staggers (1993) has shown that users work faster with compact information on one screen rather than to scroll.

The grouping of information in a window plays an important role in GUI design. Fitts (1954) developed an effective forecasting model for the time T for a movement over a distance D to an object with width W

$$T = C_1 + C_2 \log 2(2D/W)$$

with device dependent constants C_1 and C_2.

Maybe approaches like "The CAVE" (Cruz-Neira et al., 1993), a virtual reality environment, will lead to more screen space.

The question of the contents of the windows is related to showing windows. Tufte (1983, 1990) has shown proper and improper use of statistical graphics (see also Chapter **??**"). Modern statistical techniques, like data mining, but also exploratory data analysis, has lead to principles of analysis like *get an overview*, *zoom*, *select* and *look to details*.

3.6 Response times

The productivity of a user depends heavily on the response time of a software package to a user action. To achieve a good productivity we have to balance between the speed of working and the error rate. Fast response times (<1 sec) lead to faster user work. Fast working increases the error rate because the user does not think much about the current action since he is concentrated on the responses from the software package. Slow response times (>15 sec) lead to slower user work. Slow working decreases the error rate because the user has time to think about the next action. But if he makes a mistake he will loose time.

The right amount of the response time depends also on user experiences, e.g. if he sees that one software package reads a large dataset in 30 seconds whereas another software package needs 3 minutes for the same dataset then he will assume something has gone wrong. A power user is generally more impatient than a novice user. A partial solution to the problem of slow response times is a progress bar which shows how much time it will take till the end of the action.

Generally a user expects that simple actions, e.g. reading a small dataset, are done fast and complex actions, e.g. building a model from a large dataset, can take much longer time. The study of Martin and Corl (1986) found that the response time for a complex statistical task does not matter much for productivity, whereas the response time for a simple task (entering data) is linearly related to productivity. A variation in response times ($\pm 50\%$) does not matter much. In a set of mixed tasks the user balances out: he thinks about the task when the response time is slow and works fast if the response time is fast.

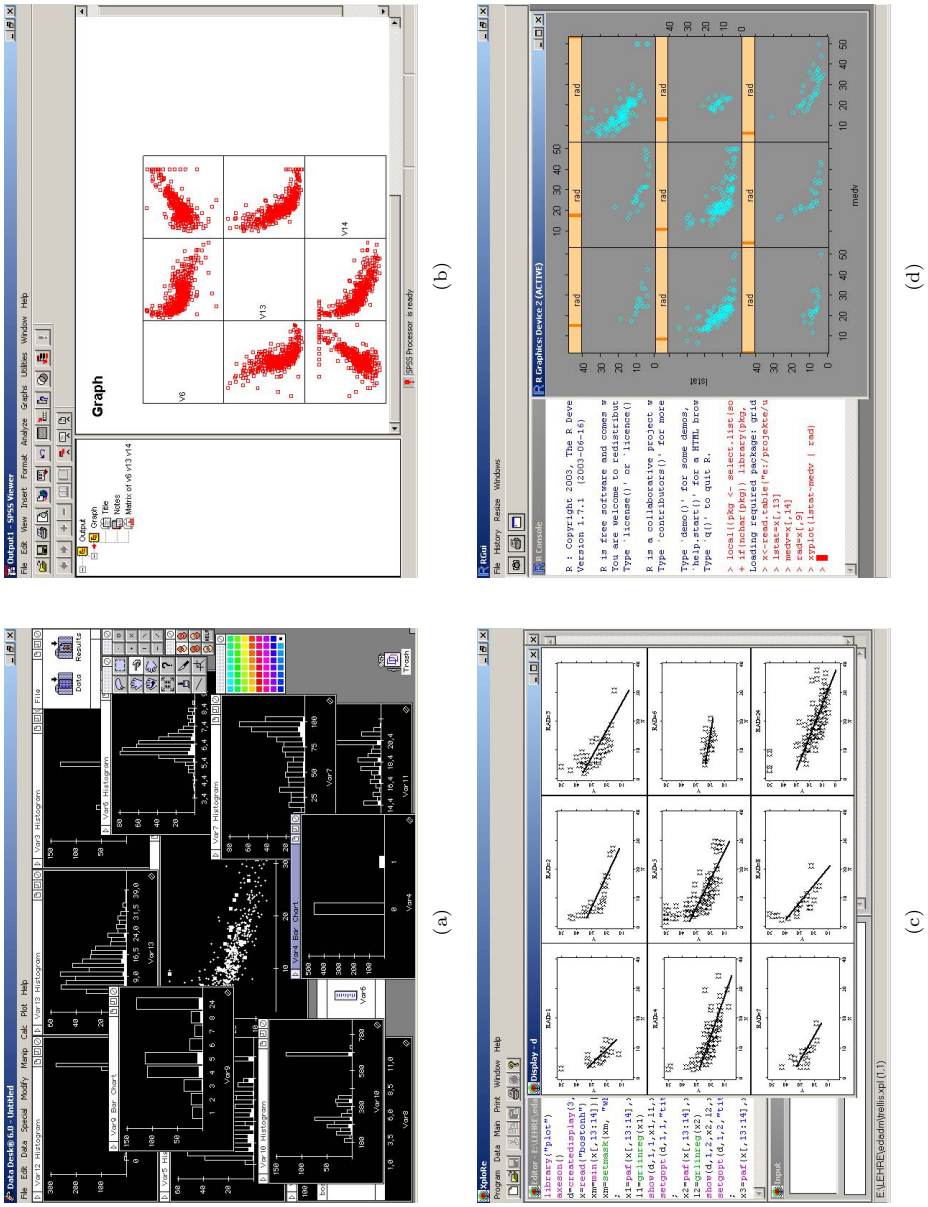


Fig. 5. (a) Linked scatterplot, histograms and barcharts in DataDesk. (b) Scatterplotmatrix of three variables "average number of rooms", "percentage of lower status people" and "median houseprice" in SPSS. (c) Trellis display of the variables "percentage of lower status people" and "median houseprice" conditioned on the variable "index of accessibility to radial highways" in XploRe. (d) Trellis display of the same variables in R.

3.7 Catching the user attention

In Fig. 4(d) we see that in SYSTAT the data editor stays on top, although we just created a scatterplot in the underlying output window. But the user attention is still directed to the data editor. Similar problems can be observed in other software.

Another point in GUI design we should consider is the way how we catch the attention of the user. In statistical graphics Tufte (1983, 1990) has shown how the user's attention can be redirected from the data. In the same manner a too colorful GUI may distract the user. Wickens (1992) analyzed how to catch the users attention and gave some hints:

- use 3 different fonts with 4 different sizes in 2 intensities
- use up to 4 standard colors, more colors have to be used with care
- use soft sounds when everything is okay, use hard sounds for warnings and errors

Nowadays operating systems offer a large variety of true-type fonts, nevertheless most people use only a few fonts in their documents.

Especially the use of colors may create special problems. First, different user may combine different reactions to the same color (cultural background); second, it is known that in Europe and North America 8% of the population have problems in recognizing a color correctly. The largest problem here is the red-green blindness, both colors appear grey to such people.

The use of sound should only be an additional option. During teaching or when working in PC-Pools it will distract other users.

3.8 Command line interfaces and programming languages

In the beginning of the computer age all programs only had CLIs. One of the largest statistical software packages which has survived from these times, SPSS, still has a CLI. But it is hidden by a GUI and we can reach it via SPSS syntax editor. Statistical programming languages, like R and XploRe, are more like CLIs embedded in a GUI. Only statistical software packages like GGobi and DataDesk are real GUI software, but even DataDesk has a (visual) programming language.

In the recent past we observed that statistical packages like R or XploRe have a tendency to be split up between a GUI and a CLI. In fact on the R-Project page we find more than one GUI for R.

CLI provides some advantages compared to a pure GUI. Some manipulations, for example arithmetic transformation of data, can be done much faster with the keyboard than with the mouse.

With a programming language we can achieve a precise and compact way to manipulate and analyze data. We should be able to easily learn, read and write the programming language. Some problems that can arise are

- the design has too many objects and actions. A hierarchical approach like organizing objects and actions in libraries may help here. However, R and XploRe suffer both from an overwhelming number of packages, commands and programs.
- sometimes the names chosen for an action are to close to computer science and not to statistics. Do we *load*, *read*, *open* or *get* a dataset (see also Table 1)?
- inconsistent order of parameters for operations.

Modern statistical programming languages implement matrix algebra since we can easily transfer expressions, e.g. for computing the coefficients of a multiple linear regression, like $(X^\top X)^{-1}(X^\top Y)$ into a program (in XploRe: `inv(x'*x)*(x'*y)`). This allows for fast error correction and fast learning.

Table 1. Reading ASCII file with the Boston Housing data.

Software	Reading ASCII data
R	`x <- read.table("c:/data/bostonh.dat", header=FALSE)`

SPSS	`GET DATA /TYPE = TXT`
	`/FILE = 'c:databostonh.dat'`
	`/DELCASE = LINE`
	`/DELIMITERS = " "`
	`/ARRANGEMENT = DELIMITED`
	`/FIRSTCASE = 1`
	`/IMPORTCASE = ALL`
	`/VARIABLES = CRIM F7.2 ... MEDV F5.2 .`

SYSTAT	`IMPORT "c:/data/bostonh.dat.dat" / TYPE=ASCII`
XploRe	`x = read ("bostonh")`

Caroll (1982) found that hierarchical (verb-object-qualifier) and symmetric command sequences, like in Table 3 for linear regression, lead to the best user performance and can be easily learned and remembered. The reality in software packages is shown in the Table 2.

Again power users prefer rather short names whereas novice users can find actions with long names more informative. It is the best to have both available, like `DoLinearRegression` and `DoLinReg` or even `dlr`. Ehrenreich and Porcu (1982) suggest rules to make (automatic) abbreviations and Schneider (1984) proposes possible abbreviation methods :

- use a simple rule to create abbreviations
  - truncation (most preferred by users)
  - deletion of vocals (`DLnrRgrssn`)

Table 2. Simple linear regression with intercept between the variable "percentage of lower status people" (lstat) and the dependent variable "median houseprice" (medv) of the Boston Housing data in different statistical programming languages.

Software	Linear regression commands
R	`res <- lm (medv ~ lstat)`

SPSS	`REGRESSION`
	`  /MISSING LISTWISE`
	`  /STATISTICS COEFF OUTS R ANOVA`
	`  /CRITERIA=PIN(.05) POUT(.10)`
	`  /NOORIGIN`
	`  /DEPENDENT lstat`
	`  /METHOD=ENTER medv .`

SYSTAT	`REGRESS`
	`  USE "c:/data/bostonh.dat"`
	`  MODEL MEDV = CONSTANT + LSTAT`
	`  ESTIMATE`

XploRe	`res = linreg (lstat, medv)`

Table 3. Example of a hierarchical and symmetric command sequences in the context of linear regression.

```
do linear regression
do linear regression stepwise
do linear regression forward
do linear regression backward
plot linear regression line
plot linear regression residuals
```

  - use last and/or first letter
  - standard abbreviation, e.g. `QTY` for `Quantity`
  - phonetical abbreviation, e.g. `XQT` for `Execute`
- use a (simple) second rule for conflicts
- apply the second rule very rarely
- abbreviations with result from the second rule should have a special symbol included
- user should know both rules
- abbreviations should have a fixed length
- the software package should always use the long name, e.g. in error messages

Modern editors, e.g. the Visual Basic editor in Microsoft Office, support the writing of programs with semi-automatic command completion.

Table 4. Error messages in different software packages.

Software	Example	Error message

XploRe	`proc()=test(x)`
	`  if(x=1)`
	`  "true"`
	`  else`
	`  "false"`
	`  endif`
	`endp`
	`test(0)`
		Syntax Error
		in test line: 2
		Parse Error
		on position 5 in line 2

R	`if (x=1) "true" else "false"`	Error: syntax error
SPSS	as in Table 2, just `/DEPENDENT` changed to `/INDEPENDENT`	Warning Unrecognized text appears on the REGRESSION command. The only recognized subcommands are: Global options: DESCRIPTIVES MATRIX MISSING WIDTH; Case selection/weight: REGWGT SELECT; Variable list: VARIABLES; Equation options: CRITERIA NOORIGIN ORIGIN STATISTICS; Dependent variable(s): DEPENDENT; Equ. methods: METHOD BACKWARD ENTER FORWARD REMOVE STEPWISE TEST; Residuals: RESIDUAL CASEWISE PARTIALPLOT SAVE SCATTERPLOT OUTFILE. Text found: INDEPENDENT This command is not executed *WARNING* REGRESSION syntax scan continues. Further diagnostics from this command may be misleading - interpret with care Misplaced REGRESSION METHOD subcommand–The METHOD subcommand must follow a DEPENDENT subcommand or another METHOD subcommand. It cannot follow any other subcommand. Check for a missing DEPENDENT subcommand. Text found: METHOD

A future dream is that (statistical) software understands natural language. What has proven to be valuable to the user is the generation of a report of results in natural language.

3.9 Error messages

The most crucial response for a user is an error or warning message from the system. Error messages can be not very helpful, e.g. in XploRe or R `syntax error` in Table 4. A better solution would be to tell the user what the problem exactly was (use `x==1` instead `x=1`). But SPSS tells the user too much and the problem disappears behind the text. However, the ability of SPSS for abbreviating is impressive. From the linear regression example in Table 2 the

parameter `/NOORIGIN` can be shortened to `/NOO`. Further shortening to `/NO` produces an error message.

The language in an error message and warning should be positive, constructive, meaningful and precise. Shneiderman (1982) found in a study that the error rate could be reduced by 28% with well constructed error messages.

Again it is a good idea to log the error message to see which ones are needed to be improved and which parts of the software package has to be improved.

3.10 Help system

Nowadays software is always accompanied with online help systems and tutorials, mostly HTML-based. A help system should give the user quick access to the information he needs. Depending on the type of users, they have different approaches to use a help system. Reference or alphabetical guides are useful for power users, but novice users learn most from a lot of examples. Consequently the help system of modern statistical software is mostly composed of several parts: reference/alphabetical guide, introductory tutorials, indices and a search engine.

In Fig. 6 we see the entry page of help systems. The variation in the software packages is large, from very sparse help systems upto detailed explanations how to use the help system.

Finding information in the help system is a crucial task. Thus good navigation, indices and search are essential for any help system. Help systems based on the windows help system, e.g. used by DataDesk, bring already the capabilities for an index and searching. Creating a good index, for a book or a help system is not easy. Especially since the developers of statistical algorithms mostly do not care much about the documentation. The quality of the help system depends heavily on the contributors to it. Maybe automated ways of analyzing tutorials and descriptions to create a hierarchy and an index can improve the help system quality.

One of useful help systems that we have seen is the help system of Mathematica which is an inherent part of it. At the top of Fig. 7(d) we see the detailed navigation, we know always where we are. Mathematica separates the information well: red background for Mathematica commands and programs and their descriptions, white background for explanations.

Generally, help systems and tutorials should have a simple language, short sentences (Roemer and Chapanis, 1982) and a consistent terminology. This has been proven more helpful to the users and most help systems follow that suggestion. It is even more important since most help systems and tutorials are written in English and the majority of the statisticians do not speak English as native language.

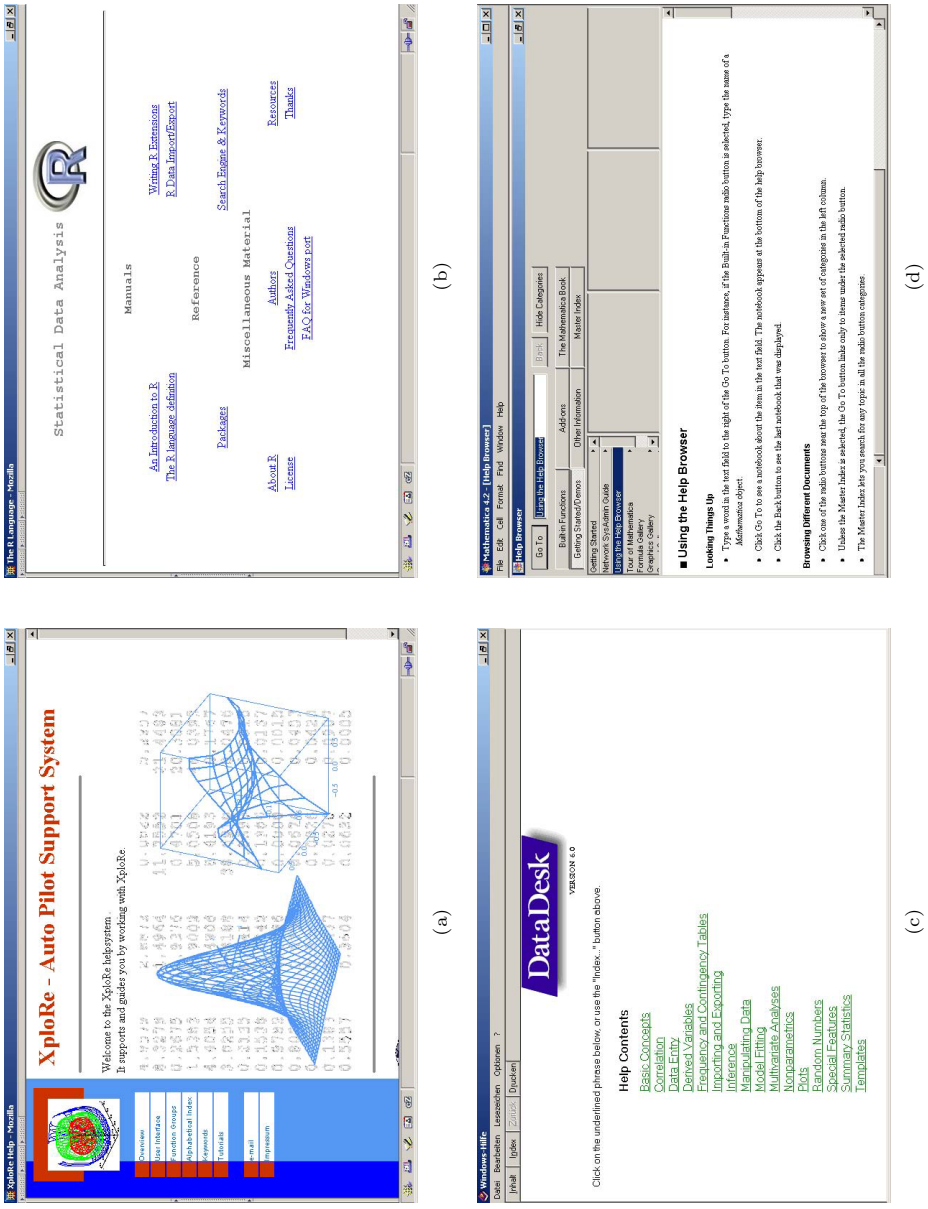
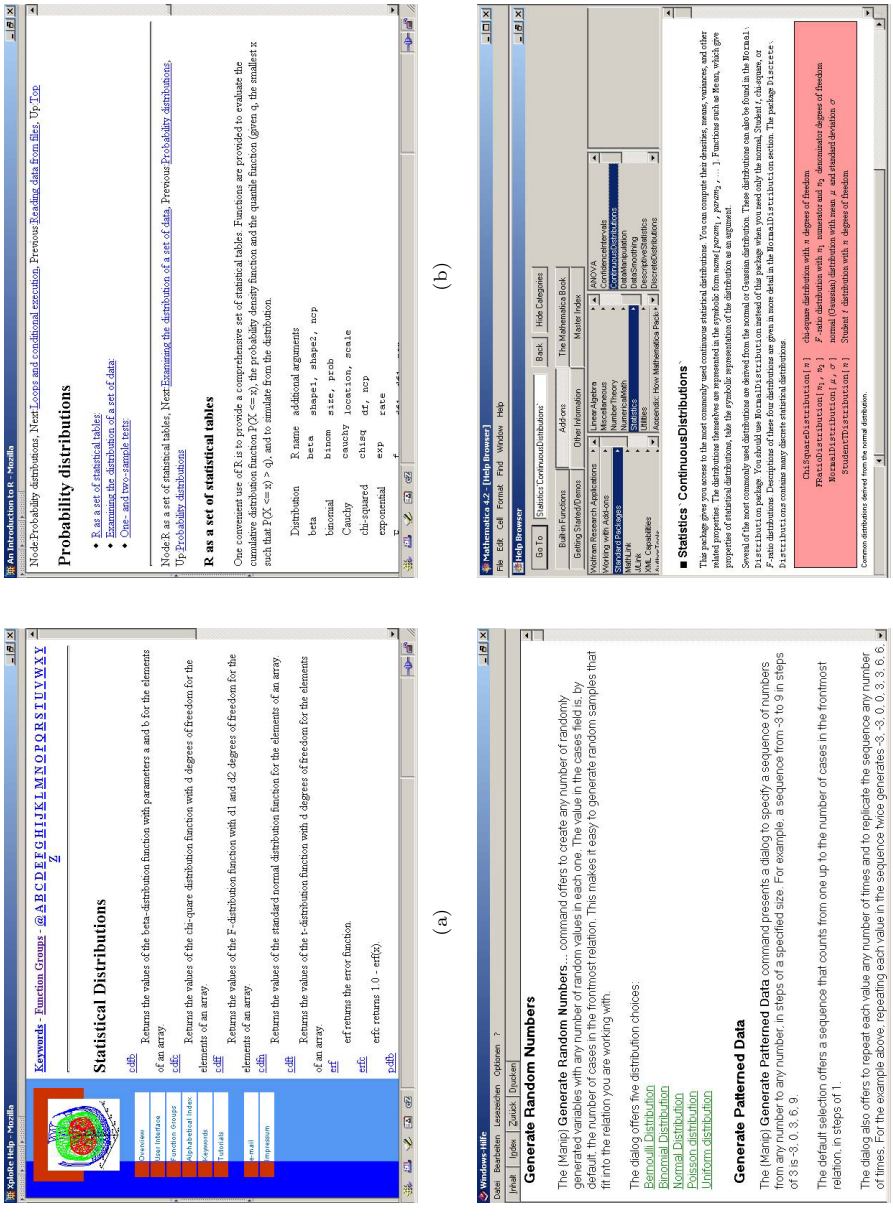


Fig. 6. Entry screens of the help systems in (a) XploRe, (b) R, (c) DataDesk and (d) Mathematica.



(a) (b) (c) (d)

Fig. 7. Help system entry for statistical distributions of statistical software packages, (a) XploRe, (b) R, (c) DataDesk and (d) Mathematica.

4 Outlook

There is a wishlist for the statistical user interface of a statistical software package:

- well-known icons for statistical tasks for useful toolbars
- a consistent and unified terminology for menu bars and items
- well designed dialog boxes and forms
- good editors for statistical programming languages
- a well constructed programming language
- a well designed HTML-based help system with clear structures
- a unique data format for exchanging data with other (statistical) software packages

In the past we have observed that statistical software packages got various GUIs. Even SPSS offers a web-based interface now, like the R Web server or the XploRe Java client version. Currently we observe that statistical software packages are embedded via direct calls (Excel: XploRe with MD*ReX, R in the RDCOM server or DataDesk /XL) or via CORBA (R: Omega Hat project) in other software. In the future statistical software packages will use the GUI of the "host" software, but the problems we will encounter are the same.

Since the user interface design depends heavily on our perception and behavior, there is still a lot of experimental research necessary to find answers to the problems that occur.

Web references

DataDesk	http://www.datadesk.com
GGobi	http://www.ggobi.org
Jasp	http://jasp.ism.ac.jp
Mathematica	http://www.wolfram.com
PRIM-9 video	http://cm.bell-labs.com/cm/cms/departments/sia/video-library/prim9.html
R	http://www.r-project.org
– GUIs	http://www.sciviews.org/_rgui/
– Omega hat/RDCOM	http://www.omegahat.org
– Web	http://www.math.montana.edu/Rweb
S/S-Plus	http://www.insightful.com
SAS	http://www.sas.com
SPSS	http://www.spss.com
SYSTAT	http://www.systat.com
VanGogh	http://stats.math.uni-augsburg.de/VanGogh/
XploRe	http://www.xplore-stat.de
– Java client	http://www.xplore-stat.de/java/java.html
– MD*ReX	http://www.md-rex.com

References

Bury, K., Davies, S., and Darnell, M. (1985). Window management: A review of issues and some results from user testing. Technical report, IBM Human factors Center Report HFC-53, San Jose, CA.

Card, S. (1982). User perceptual mechanism in the search of computer command menus. In *Proc. Human Factors in Computer Systems*, pages 190–196, Washington D.C.

Caroll, J. (1982). Learning, using and designing command paradigmas. *Human Learning*, 1(1):31–62.

Cleveland, W. and Loader, C. (1996). Smoothing by local regression: Principles and methods. In Härdle, W. and Schimek, M., editors, *Statistical Theory and Computational Aspects of Smoothing*, pages 10–49. Physika Verlag, Heidelberg, Germany.

Cruz-Neira, C., Sandin, D., and DeFanti, T. (1993). Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proc SIGGRAPH'93 Conference*, pages 135–142, New York. ACM.

Ehrenreich, S. and Porcu, T. (1982). Abbreviations for automated systems: teaching operators and rules. In Badre, A. and Shneiderman, B., editors, *Directions in Human-Computer Interaction*, pages 111–136. Ablex, Norwood, NJ.

Fitts, P. (1954). The information capacity of the human motor system in controlling amplitude of movement. *Journal of experimental psychology*, 47:381–391.

Franzke, M. (1995). Turning research into practice: Characteristcs of display-based interaction. In *Proc. CHI'95 Conference: Human Factors in Computing Systems*, pages 421–428. ACM.

Hansen, W. (1971). User engineering principles for interactive systems. In *Proc. Fall Joint Computer Conference*, volume 39, pages 523–532, Montvale, NJ. AFIPS Press.

Härdle, W. (2004). Interview with James E. Gentle. *Computational Statistics*, 19(1): 1–4. Physika Verlag, Heidelberg, Germany.

Harrison, D. and Rubinfeld, D.L. (1978). Hedonic prices and the demand for clean air. *J. Environ. Economics and Management*, 5: 81-102.

Keller, R. (2003). Visualizing augsburg traffic data with vangogh. Presentation at 'Workshop on Statistical Inference, Visualizing for Graphs' at Stanford University, CA., University of Augsburg, Dept. of Computer Oriented Statistics and Data Analysis.

Liebelt, L.-S., McDonald, J., Stone, J., and Karat, J. (1982). The effect of organization on learning menu access. In *Proc. Human Factors Society*, Twenty-Sixth Annual Meeting, pages 546–550, CA.

Marron, J. (1996). A personal view of smoothing and statistics. In Härdle, W. and Schimek, M., editors, *Statistical Theory and Computational Aspects of Smoothing*, pages 1–9. Physika Verlag, Heidelberg, Germany.

Martin, G. and Corl, K. (1986). System response time effects on user productivity. *Behaviour and Information technology*, 5(1):3–13.

McDonald, J., Stone, J., and Liebelt, L. (1983). Searching for items in menus: the effects of organization and type of target. In *Proc. Human Factors Society*, Twenty-Seventh Annual Meeting, pages 834–837, Santa Monica, CA.

Miller, G. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychologocial science*, 63:81–97.

Mitchell, J. and Shneiderman, B. (1989). Dynamic vs. static menus: An experimental comparison. *ACM SIGCHI Bulletin*, 20(4):33–36.

Norman, K. and Chin, J. (1988). The effect of tree structure on search in a hierarchical menu selection system. *Behaviour and Information Technology*, 8(2):25–134.

Roemer, J. and Chapanis, A. (1982). Learning performance and attitudes as a function of the reading grade level of a computer-presented tutorial. In *Proc. Conference on Human Factors in Computer Systems*, pages 239–244, Washington D.C. ACM.

Schneider, M. (1984). Ergonomic considerations in the design of text editors. In Vassiliou, Y., editor, *Human Factors and Interactive Computer Systems*, pages 141–161. Ablex, Norword, NJ.

Sears, A. and Shneiderman, B. (1994). Split menus: effectively using selection frequency organize menus. *ACM Transaction on Computer-Human Interaction*, 1(1):27–51.

Shneiderman, B. (1982). System message design: Guidelines and experimental results. In Badre, A. and Shneiderman, B., editors, *Directions in Human/Computer Interactions*, pages 55–78. Ablex, Norword, NJ.

Shneiderman, B. (1998). *Designing the User Interface*. Addison Wesley Longman, Inc.

Staggers, N. (1993). Impact of screen density on clinical nurses' computer task performance and subjective screen statisfaction. *International journal of Man-Machine Studies*, 39(5):775–792.

Temple, Barker and Sloane, Inc. (1990). The benefits of the graphical user interface. *Multimedia Review*, pages 10–17.

Tufte, E. (1983). *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT.

Tufte, E. (1990). *Envisioning Information*. Graphics Press, Cheshire, CT.

Tukey, J., Friedman, J., and Fishkeller, M. (1973). Prim–9: An interactive multidimensional data display and analysis system. Video. ASA Statistical Graphics Video Lending Library.

Tukey, J., Friedman, J., and Fishkeller, M. (1974). Prim–9: An interactive multidimensional data display and analysis system. Technical Report SLAC-PUB-1408, Stanford Linear Accelerator Center, Stanford, CA.

Ulich, E., Rautenberg, M., Moll, T., Greutmann, T., and Strohm, O. (1991). Task orientation and user-oriented dialogue design. *International journal of human-computer interaction*, 3(2):117–144.

Wickens, C. (1992). *Engineering psychology and human performance*. Harpercollins publisher.

Index