GUI and Command-line - Conflict or Synergy?

Article · January 2000

Source: CiteSeer

CITATIONS
11

READS
2,184

2 authors:

Antony Unwin
Universität Augsburg
172 PUBLICATIONS **2,950** CITATIONS
SEE PROFILE

Hofmann Heike
Iowa State University
122 PUBLICATIONS **2,014** CITATIONS
SEE PROFILE

GUI and Command-line – Conflict or Synergy?

Antony Unwin and Heike Hofmann

Department of

Computer-Oriented Statistics and Data Analysis

University of Augsburg, Germany

Interfaces rely on mutual understanding. Graphical interfaces tend to aid understanding and be easier to pick up again after a break. They are good for direct interaction without exact specification. Command-line interfaces are for specialists with deep understanding and offer powerful, flexible structures. They make it easy to store and repeat precise sequences. Ideally, we would like to have both forms of interface. To make progress we will have to understand better what statisticians and data analysts need and what the strengths (and weaknesses) of the two approaches are.

1 Introduction

Statistical software should be expressive, powerful, fast and reliable, but it should also be intuitive, easy to use, flexible, forgiving and consistent – and probably a few other things as well. Software development used to concentrate very much on technical issues, but it is gradually becoming accepted that the user-interface is very important as well. For statistical software that means, in crude terms, a choice between a command-line interface and a Graphical User Interface (GUI). Programming or batch interfaces can be more or less excluded because almost all statistical analysis can now be performed interactively. The interfaces of older packages were partly limited by the requirements of batch processing and partly by small main memory. They were designed for analyses which in

those days took a long time and which to-day can be carried out instantaneously.

The aim of this paper is not to push one interface approach over the other but to outline the differences between them, to discuss their respective strengths and weaknesses and to review the potential for combining them effectively. Command-line interfaces allow the precise, repeatable commands of the traditional computer science approach and are associated with displaying optimal results for well-defined questions. They need not, of course, be restricted only to command-line input and linear output. GUIs should allow direct, interactive manipulation, a more modern approach associated with searching for information and interesting structures without fully specified questions. The emphasis with GUIs should be on direct manipulation systems and not on menu systems.

Good interfaces are needed to use software to its full potential. This applies just as much for expert users as for beginners. A beginner faced with a poor interface will make a mess of their analysis and be unsure of their results. An expert with the same software will complete their analyses more successfully, but at a cost of having to worry about syntax and details instead of being free to concentrate on the statistical modelling. As Wirth (1985) puts it, writing about languages: "The real cost is hidden in the unseen efforts of the innumerable programmers trying desperately to understand them [large systems] and use them effectively."

One problem in assessing interfaces is that we tend to judge them on the basis of actual implementations we know and not on their potential. John Chambers describes a dire example from a GUI in Appendix B of his recent paper (Chambers 1999, though the appendices are only available in the full version from his webpage) and implies it is what you might expect from a GUI. Let's hope not! GUI's suffer from this kind of criticism at the moment because there are so many bad ones around, but criticisms of similar style may be applied just as well to command-line systems. We should always consider whether weaknesses identified are inherent in the approach or a failing of the implementation. Examples include GUI packages with windows cluttered with buttons or command-line packages with confusing syntax.

Another problem is disparaging software for individual faults. It is easy enough to find clumsy features or sloppy details in any piece of software, but what counts is the overall effectiveness. We try in this paper to discuss the potential more than the actual, though it is unavoidable that we offer some examples from actual implementations.

[It is impossible to give the feel and flavour of using different systems in a published paper, but there are many examples and illustrations on the webpages of the various packages. We urge readers to inspect these while considering what we have written. A partial list of URL's is given at the end of the paper.]

2 What is an interface?

In English the word interface can be used in two separate, but related, ways in computing. You can mean an interface between two systems or the interface between users and software. The latter may be distinguished by calling it the user-interface. In German there are two different words, Schnittstelle (literally, place of intersection) for the former and Oberfläche (literally, surface) for the latter. Chambers, for instance, uses the former meaning in his recent article [Chambers (1999)] and downplays the latter. Command-line structures are good for his kind of interface because they are written to be efficient and easy for connecting systems. But what about the human-computer interface? That is what we discuss in this paper.

What is a good interface? It depends on the user's needs and experience, on what they have to do, on what facilities they have (hardware, other software, availability of help and advice) and on their style of working. The term "user-friendliness" has been discredited by overuse and by systems that weren't, but also because it is not the right term. What is useful is help for the user, unfriendly help would be better than friendly non-help.

Many people recommend command-line interfaces because you can specify exactlyprecisely what you want to do, because you can check what you have done easily and because you can repeat complicated analyses either exactly or with specified changes of parameters. Objections to these interfaces are that:

i. they are good for regular users who can remember the wide range of commands and options but not for

occasional users;

ii. even experts misstype. (Huber [1994, p67] suggests that a third of lines might be erroneous, i.e. mistyped or otherwise in error);

iii. the commands aremay not be intuitive;

iv. they are suited for modelling and not for interactive graphics.

Such interfaces give the user the impression of working when in fact they are striving to remember syntax and not concentrating on the statistical task in hand. Tognazzini (1992, p181) describes this as abandoning cognitive process on the primary task (of statistical analysis) to give attention to the secondary task (of command syntax). It is the contrast between thinking about concepts and having to worry about technical details.

When computing took a long time and mistakes were costly, input precision was important. Few results were available and they had to be right. Now we can compute many results very quickly (which provides circumstantial checks) and many different approaches can be tried. Computers can now be forgiving and errors or false trails quickly forgotten.

The newer GUIs are recommended for being intuitive, easy to learn, requiring recognition rather than recall and being good for exploratory analyses and graphics. Objections are that:

i. they are not precise enough;

ii. it is difficult to retrace steps or to replicate analyses;

iii. they are not well suited for modelling;

iv. they are difficult to design well.

3 Users and software?

Users might be classified as

statistical researchers in a university

applied statisticians

regular users of the same statistical tools

occasional users of statistics

students

The first two groups are "power-users" who undoubtedly require powerful systems which offer them flexible structures to work with. Since they work daily with the software they like short-cuts, terse output and the ability to adapt systems to their varying needs.

The third group would like turnkey systems that fit the particular problem they have to deal with. They do not require flexibility and a good interface is probably not essential, since they always need the same commands. A former graduate now working with a major statistical software house told me that a common request from industrial clients was to "please design the software so that non-expert users can’t cause trouble with it".

People in the fourth group, which may be potentially the largest, need simple tools and software they can return to easily after not using it for a while. An intuitive and forgiving (i.e. safe) interface is invaluable. Occasional users will not invest time in packages with initially steep learning curves, as they then have to relearn every time they use the package. The increasing public availability of data (especially on the web) suggests that this group could become

much bigger, if suitable software was available. The history of statistics as a subject suggests that others will develop this area first.

Students need assistance in using the software and in the concepts behind the methods available. At one extreme they may only require a limited range of tools (e.g. business studies students) or, at the other extreme, they may go on to become statistical researchers and need much more specialised features.

The last three groups should not be given power to change the software. This is a major bone of contention. Huber (1994), for instance, bewails lack of control over software as a reversion to the 1960s and 1970s when the computer centres dictated what was good for the users. He writes of the "liberating period of the 1980's, when the people gained hands-on control of their own hardware". Yet limiting control is the only way to ensure a consistent integrated and stable interface. Car owners accept a much more limited control over the design of their cars than they may have had in the past, but get much more reliable vehicles in exchange. Specialist drivers and researchers have extra needs, but the vast majority do not.

An initial coarse classification of work to be done might be simply statistics and EDA. Exploratory work needs fast, flexible software while model-building and confirmatory testing require precise, replicable commands. EDA cannot readily be carried out with a command-line interface and there is as yet no GUI ideal for model-building. A further factor is the size of the data sets. Interfaces suitable for small data sets (such as spreadsheets with a few variables and not many cases) are unsuitable for large data sets (with many variables and thousands of cases). Spreadsheets are an interesting example. They have been used extensively in paper form for years, but were never implemented by "serious" computer users. Despite a poor interface for functional forms and for forcing tight structure, their interface offers many attractive features: they mirror directly what many people are accustomed to, they emphasise the numbers rather than the formulae, they permit adding notes easily, they have flexible graphics and they are very helpful for test calculations. Their weaknesses for programming and large scale problems are irrelevant to the vast majority of their users.

There is a tendency to judge software by the most powerful tools they provide, (whether with a good interface or not), rather than by whether they do the simple things well. Consider this quotation from a recent paper: "To-day, most software products have intuitive, graphical interfaces that allow users to perform sophisticated actions without learning obscure command sequences."[Finholt and Olson (1997)]. Before racing ahead to enable "sophisticated actions", we should remember that there are still a great many simple things which are difficult to do in many packages and which packages:are valuable: e.g. reordering categories in a bar chart for better comparisons, changing the bin width in a histogram to explore the data distribution or moving variables in and out of linear models for finding the best model.

There is a huge HCI (Human Computer Interaction) literature, but it does not seem to have had much effect on statistical software (or indeed much directly on any kind of software, sadly). The practical problems of designing whole, growing systems outweigh the relevance of scientific studies on individual issues. There are some good books which outline general concepts such as those by Norman (1988, 1992, 1993) and Tognazzini (1992) and those edited by Laurel (1990) and Baeker et al (1995). Shneiderman's article in Baeker et al is particularly relevant here.

4 Some basic tasks for statistical software

Interfaces may be compared by the style of working they encourage and by how they perform typical data analysis tasks. Advertising for software packages often lists large numbers of tools, but as Norman (1988) points out, it is the tasks that have to be carried out that should be emphasised, not the tools used. It is hard to define typical or representative tasks because not only do people work differently, but their software influences how they work. Two people analysing the same data with different software will hopefully come to similar conclusions, but they will not go about their analyses in the same ways. Nevertheless, there are some tasks that will be performed by everyone and should include: calculating summary statistics on variables, plotting histograms, fitting models and selecting subgroups for further analysis.

4.1 Summary statistics

Taking S+ and Data Desk as examples of the two approaches, they both make it easy to get a variety of statistics on variables. In Data Desk it is simple to choose the variables, just by grabbing their icons. In S+ they can be typed (slow and error-prone) or, in the latest PC version 4.5, they can be selected from a list. Data Desk offers a dialog box to specify which statistics and in S+ you can set a range of options.

If the user decides that the statistics are not the ones they want, they can change the options and run again (S+ and Data Desk for several variables). If only one variable is involved, then in Data Desk it is possible to change the options in that output window and have the results update there. Both packages offer a variety of ways for producing statistics for subgroups, but this is a tricky issue, both for specifying which groups and for laying out the results.

For this task the GUI looks better, but there is not a lot in it. It is hard to believe there is nothing better on offer.

4.2 Plotting histograms

Many packages offer extensive ranges of options for drawing displays. Anyone who has used the Chart Wizard in Excel will be familiar with that! For exploring data what is needed are simple commands with sensible defaults and intuitive and flexible ways of changing them. GUI systems allow several histograms to be drawn at once. Bin-widths and other parameters may be amended interactively (e.g. as in MANET, Unwin et al [1996]) or via dialog boxes. It is easy to move the displays around or change their sizes for comparative purposes. It is easy to query the displays directly to get information.

GUIs are good for preparing presentation graphics as well, since you can click directly on the part of the graphic you want to change. The big advantage of command-line systems lies in preparing routines, which are to be run automatically many times, always using the same format.

GUIs definitely have the advantage in flexibility and ease of use with graphics. It would be valuable to combine this with the command-line strength of repeatability. This could be achieved if the options could be translated into commands in an efficient manner.

4.3 Fitting models

Command-line interfaces offer powerful commands for model-building (in the sense of Huber (1994): seemingly simple but sophisticated), which can be readily amended as part of a continuing model-building process. The command syntax reflects the model form. All outputs (whether intermediate or final) can be available as objects for further analysis, as exemplified in S+.

Both JMP and Data Desk provide GUI approaches to model-building and attempt to make clear what is available via hyperviews and buttons. This can lead to cluttered windows. Better implementations are probably possible, but much work remains to be done.

Command-line interfaces are currently far better here.

4.4 Selecting cases

The Command-line approach is good when we can define exactly which cases we want and for naming the resulting group. The GUI approach is good for making selections from graphics.

Sometimes we want to analyse a pre-defined subgroup of cases (e.g. all males), sometimes we want to analyse a subgroup that has attracted our attention in analysis (e.g. a specific group of points in a residual plot). Selecting points in graphics is always easier with a GUI, often it is the only way of making the selection. It is excellent for the fuzzy selecting that is interesting in exploring data, i.e. varying the selection to see the effect on results.

On the other hand, selecting a pre-defined group sounds like it would be easier with a command-line interface. But

consider the following example: select all males over 50 who are divorced or separated. This demands a complex statement (e.g. using SQL or "ifs") which is difficult to set up and difficult to read, though it can be checked in principle. It is too complicated a line for a command-line interface. A GUI approach for this, called selection sequences, has been implemented in MANET (Hofmann and Theus 1998). Selections are made from each of the relevant graphics windows (bar chart for gender, histogram for age, bar chart for marital status in the example), qualified by the appropriate selection mode (mostly intersection) and recorded in the windows. This method is easily grasped, but, like the command-line approach, is not easy to check.

A combination of the two approaches would be beneficial.

5 Principles for the design software interfaces

Interfaces are a subjective matter, not just because we have different skills, different experiences and different requirements, but because we have different tastes. Nevertheless it is possible to identify some interfaces as better than others in an objective way, by assessing them in terms of general principles. A good source is Baeker et al (1995). Guidlines for interfaces for statistical software would include the f ollowing.

5.1 Make low demands on users

Don’t make heavy demands on user’s memories (eg remembering names or syntax or where objects are) or on their typing skills. Systems need to be able to list and display objects in an organised fashion and to be able to find objects used in analyses or displays. Compare, for instance, typing full path names to opening files with using a dialog box. This is important in analysing data sets with many variables. Constructing functions of variables is easier and more reliable with "Drag and Drop" than by typing. As Tognazzini (1992, p133) puts it: "Make the application memorable by reducing the user’s need to memorise."

5.2 Consistency

Commands should have the same form and qualifiers should always operate in the same way. It is surprising how difficult it can be to maintain consistency, where you might think it was easier to reuse code e.g. in JMP: the Fit Y by X and the Fit Model dialogs differ in having Y in different positions and in the latter’s having no cancel button. Responses and output of all kinds should be consistent.

5.3 Implicit availability

What you would need in any situation should be available, but it does not have to be directly displayed. Ideally it should be intuitive from the styling on the window. MANET changes the form of the cursor, as it is moved across a window, to indicate further options. JMP and Data Desk both make use of standard buttons on their window-frames, which give access to a range of context-sensitive features. This is a huge improvement over old-style outputs of model-building results which gave you everything (SPSS in the past) or almost nothing (GLIM).

5.4 "Do it where it’s at" (Immediacy of space)

If an object is to be queried or changed then it is intuitive to do it by clicking on the object, not by using written commands and specifying the object by name.

If an object is queried or changed, then ideally the information should be shown on or beside the object. Systems which report values of points currently under the cursor at a standard position on screen could be said to be consistent, but they lose the association of the information with the point. (It may be done because the information obscures a part of the screen. This is only a valid reason if it is permanent, which happens with painting names for instance. Information, which disappears when you let go the mouse is not a problem.) Statisticians might want to query cases, variables, graphics or models. Sweller et al (1990) describes a teaching study which supports this idea. They refer to reducing cognitive load.

5.5 Immediacy (Immediacy of time)

Access – you should be able to do what you want immediately and not have to close one module and open another. Amongst other things, this implies that following several paths in parallel should be possible (see also the principle of multiple views below). Exploratory analyses may involved looking at a variety of histograms, dotplots and boxplots of the same data and inspecting a range of models. Single-window linear work methods are not good for this.

Response – if a result or feedback does not appear immediately, your current train of thought is broken. It is better to break the train of thought between results and commands than between commands and results (i.e. when you decide, not when the computer decides).

5.6 Immediacy of the data

It should be possible to identify cases and get variable values for individual cases or groups of cases quickly and easily. Many results are heavily influenced by outliers or by particular groups of points. Being able to identify them helps to assess the validity of results.

Displaying the raw data can sometimes reveal peculiarities, which explain the results of analyses. Being able to check the raw data, even for large data sets, in an efficient way can be very useful. In this respect Data Desk's design of giving each variable its own column seems a better solution for many variables compared to the spreadsheet approach. Some kind of tabling is essential, however.

5.7 Return

To get the most from an analysis you must be able to return to previous stages easily. Being able to undo the last command(s) is useful (and is important for graphical exploration where onscreen commands may not be exact). Retracing steps is vital for recovering from false moves, otherwise you have to go back to the beginning. Those are negative reasons for being able to return easily. More positive reasons are wanting to be able to compare results of different models. It is one thing to know they are somewhere in memory, it can be another to retrieve them readily and in a form which may easily be compared with others. In TURNER [Lauer (1998)] a table of loglinear models already fitted is automatically generated. Comparisons between models can be made and full results recovered.

5.8 Multiple views

Different displays and different models give alternative points of view. This implies the need for a multi-windowing system with multiple graphical displays and building of several models simultaneously. It ought to be possible to follow several ideas at once, whether different versions of the same type of model or different models. Methods complement each other.

5.9 Helpful help

Help may be provided in a wide variety of ways: printed manuals, help files, balloon help, "assistants", online help, hotlines and so on. Looking up manuals or similarly structured help files for explanations of a long list of parameters is not supportive of analyses. The ideal would be immediately available context-sensitive support to guide use. We are still far from that, but there are some promising examples e.g. in JMP.

5.10 Style

When in doubt do it simply, elegantly and beautifully: clear, well-arranged dialogues and sharp, readable outputs. Unfortunately it is much easier to see that an interface is ugly than to successfully design a good one.

6 Integration and extensibility

It is clear that some tasks are better handled by one approach rather than the other and that some can be more or less handled by both. It also seems reasonable that different users will have different preferences for the kind of interface they want. For any closed system we could imagine using both interfaces as alternatives and possibly combining the

best of each. Potential improvements for Command-line interfaces include more use of floating dialog boxes and of drag and drop. There are many interesting possibilities for GUIs for model-building using graphical displays of the model structure, which can be queried and manipulated. Open systems are more difficult.

Many older statistical software systems had a strong modular structure, which enabled new methods to be added. These methods were then only weakly linked to the system and had to be used more or less independently (sometimes to the extent of operating as separate programmes with only a common data format). More modern systems use an object-oriented approach which also permits extending the software, but potentially in a more integrated way. The degree of integration demanded influences a system's extensibility greatly.

Systems with a direct manipulation GUI are ideally closely integrated and have very tight rules about new modules for how the objects may be queried, moved and manipulated. For instance, how should an object from a new module behave if it is dragged into the window of an existing module? Combinations of independent software are hard. Even the Mac OS with its well thought-out rules (Apple Human Interface Guidelines (1987)) is unable to stop programmes carrying out quite basic tasks in very different ways and is unable to guarantee that all programmes can exchange information.

New modules for systems with command-line interfaces must make their objects accessible, but can more easily put limits on how they may be accessed. Any greater degree of integration is correspondingly difficult. Several systems from different areas (e.g. statistical software and Geographic Information Systems) have made much of achieving quite basic interfaces between each other.

Ideally it should be possible to separate much of the basic software in a system from the statistical components. The development environment would then guarantee the integration requirements. We are still some way away from that but making progress all the time. The ease with which data and files of all kinds can be moved between software and systems is steadily improving. So it becomes a matter of goals. If we wish to be able to use any software out there in conjunction with our own, then we require flexible, but very limited interfaces between systems. If we wish for more integration, then it will be at the expense of combining with other packages. Personal choices may depend on experience. Anyone who remembers passing punched cards into a computer centre and getting the output two days later will find instantaneous calculation of complex linear models wonderful. Youngsters who surf the net with several windows running in parallel will find it old-fashioned that statistical software doesn't encourage parallel working and doesn't offer efficient ways of collating and summarising outputs.

Open extensible systems must be highly modular to avoid clashes, but this reduces possible interactions between the different parts. A totally unrestricted system lacks consistency and does not look well. A tightly constrained system inhibits inventiveness, but is consistent, which makes it easier to use, and highly integrated, which makes it fast.

7 Conclusions

Better implementations than are available at the moment are possible for both command-line interfaces and GUIs. We may look forward to considerable improvements in both. Which we prefer and use will depend on our needs and on our style of working. It would be nice to think that we can expect systems, which have all the advantages of the two approaches, but this will take time and will depend on how much openness we demand. Researchers must have open systems they can experiment with and improve, but shouldn't standard users (whatever that may mean) have closed, integrated systems which are consistent and intuitive?

Note

Opinions are influenced by experience and no one can have used all possible software to an expert-user level. Our experience is currently biased towards GUI systems (Data Desk, MANET and, to a lesser extent, JMP) but we have used modern command-line systems such as S+, LispStat and R, and we have past experience of SAS, SPSS, BMDP, Minitab, Systat and Statistica.

Software References

Data Desk	www.datadesk.com
JMP	www.sas.com/otherprods/jmp/
MANET	www1.math.uni-augsburg.de/Manet/
R	www.ci.tuwien.ac.at/R/
S-Plus	www.mathsoft.com/splus/
LispStat	www.stat.umn.edu/~luke/xls/xlsinfo/
SAS	www.sas.com
SPSS	www.spss.com/software/spss/
Minitab	www.minitab.com
Systat	www.spss.com/software/science/systat/
Statistica	www.statsoft.com

Printed References

AppleComputer (1987) Human Interface Guidelines: The Apple Desktop Interface.

Baeker, R. M., Grudin, J., Buxton, W.A.S., Greenber, S. (Ed.). (1995). Human-Computer Interaction: Toward the Year 2000 (2nd ed.). San Francisco: Morgan Kaufmann.

Chambers, J. (1999). Computing with Data: Concepts and Challenges. American Statistician, 53(1), 73-84.

Finholt, O. (1997). From Laboratories to Collaboratories. Pysch Science, 8, 28-36.

Hofmann, H., Theus, M. (1998). Selection Sequences in MANET. Computational Statistics, 13(1), 77-87.

Huber, P. J. (1994). Languages for Data Analysis. In P. Dirschedl and Ostermann, R. (Eds.), Computational Statistics Heidelberg: Physica.

Lauer, S. (1998). Modelling Discrete Data: Interactive Loglinear Models. Computational Statistics, 13(1), 89-99.

Laurel, B. (Ed.). (1990). The Art of Human Computer Interface Design. New York: Adison-Wesley.

Norman, D. A. (1988). The Design of Everyday Things. New York: Doubleday.

Norman, D. A. (1992). Turn SIgnals are the Facial Expressions of Automobiles. New York: Adison Wesley.

Norman, D. A. (1993). Things that Make Us Smart. New York: Adison Wesley.

Shneiderman, B. (1995). A Taxonomy and Rule Base for the Selection of Interaction Styles. In R. M. Baeker Grudin, J., Buxton, W.A.S., Greenber, S. (Eds.), Human-Computer Interaction: Toward the Year 2000 (pp. 401-410). San Francisco: Morgan Kaufmann.

Sweller, J., Chandler, P., Tierney, P., Cooper, M. (1990). Cognitive Load as a Factor in the Structuring of Technical Material. J of Exp Psychology General, 119(2), 176-192.

Tognazzini, B. (1992). TOG on Interface. New York: Adison-Wesley.

Unwin, A. R. (1995). Interaktive Statistische Grafik – eine Übersicht? In J. Frohn et al. (Eds.), Applied Statistics –

Recent Developments (pp. 177-183). Dortmund: Vandenhoeck & Ruprecht: Göttingen

Unwin, A. R., Hawkins, G., Hofmann, H., and Siegl, B. (1996). Interactive Graphics for Data Sets with Missing Values - MANET. Journal of Computational and Graphical Statistics, 5(2), 113-122.