



*DSC 2001 Proceedings of the 2nd International
Workshop on Distributed Statistical Computing
March 15-17, Vienna, Austria*
*<http://www.ci.tuwien.ac.at/Conferences/DSC-2001>
K. Hornik & F. Leisch (eds.) ISSN 1609-395X*

R objects, two interfaces! (R objects to interfaces?)

Antony Unwin*

1 Introduction

What do we mean by interfaces? John Chambers in his article in the *American Statistician* (Chambers, 1999, [1]) writes primarily about interfaces between different software packages, a very important issue and one which is well-known to all involved in R and Omegahat. I want to talk about interfaces in another sense, the interfaces between users and software.

The experts who use a software very often do not worry about the interfaces. They have developed their own way of doing things and are happy with what they have: otherwise they would not be using the software! If they are developing as well, then they tend to concentrate on tools and capabilities not on the HCI. The less demanding an interface, the less restrictions have to be placed on developers. Interfaces influence programme structures. It is quite different for occasional users. They may well be put off by a complex or clumsy interface. There are some (and Luke Tierney is on record as being one of them) who believe that the interface to a package should not be too easy, because there is then no learning effort needed when simple tasks are carried out. The logic is that people learn how to manipulate the software while they understand the methods and so can use the software effectively when the methods become more complicated. The mental effort that was initially needed for the syntax is then devoted to the methods. This line of argument implies that applying simple methods doesn't need thought (a dangerous assumption in statistical work) and that complicated methods require complex manipulations (maybe with a better interface this would not be such a problem).

*University of Augsburg, Germany, unwin@math.uni-augsburg.de

Criticising software because of its interface implies that the software is good enough in the first place that it is worth improving the interface(s). There is bad software around with good interfaces (some games, for instance) but if the software is bad, no interface can save it, no matter how good. Good interfaces make the software easier to use for experts, make it more accessible to others and make it more transparent. In academic circles these are not always goals which are commonly aspired to. Experts want to remain experts, they want to remain a select group and they certainly do not want others to know what they are doing. That's all very well for some esoteric theoretical areas, but it seems a great pity to apply the same criteria to such impressive and practical software as R.

2 R's interface

Where does R stand on interfaces? Originally, it adopted what S did and so it has a very good command-line interface: very good for 20 years ago, that is. S+ has recently provided a spreadsheet and GUI type interface in some versions, but I have not used that and cannot therefore not comment seriously. Unseriously, I suspect that the spreadsheet features added to S+ are as good a spreadsheet compared to Excel as the statistics added to Excel are as good a statistics package as S+.

The R interface has many attractive features:

- commands can be powerful and flexible
- copy and paste is easy
- macros can be written straightforwardly
- output is terse, but expandable

and, of course,

- (almost) everything is an object and can be accessed.

So what more could anyone want? Here is one (subjective) wishlist:

- interactive querying of objects
- interactive opening of objects
- hotlinked objects
- better graphics
- interactive graphics and controls
- restoring of previous states
- multiple paths and multiple output windows

3 Interface Wishlist

3.1 Interactive querying

This has been around for a long time, but has hardly developed in many systems. The ideal should be that you can query every object: whether in a graphic or text representation. Typing in commands to ask R about its objects is useful, but hardly intuitive. (We will ignore querying of graphics in R for obvious reasons.) There are two separate components, firstly the capability of querying at all, which is very much a GUI feature, and secondly the capability of making a range of context sensitive queries of any particular object. Given R's object structure, the latter is pretty well already in place. Flexible querying is implemented in MANET (Hofmann, 2000, [2]) in a hierarchical way. Standard queries (option-clicking an object) provide default information (e.g. querying a continuous variable gives the range and the number of different values) while extended queries (shift-option-click) go into more detail (some additional statistics for a continuous variable). Current plans are to experiment with a third level (a histogram for a continuous variable?). Note that this is querying, not labelling: the results are shown only while the mouse button is down.

The ability to click on object names is something commonplace in surfing the web, but querying is different. In surfing you are taken on somewhere else (and it may not always be clear why), while in querying you get a temporary insight into greater detail, a kind of temporary logical zoom. Having the extra information implicitly available is reassuring and supportive, hardly what experts need, but excellent for occasional users.

3.2 Interactive opening

Opening objects is a more powerful idea, you not only find out what is contained in an object, you can access and manipulate it. If a data set had been loaded in R, then a query would tell you what variables are included (without having to have the list written out to the output window), while opening it would make the variables available for operations, such as dragging and dropping them into a model formula. This is a key example for discussing the relative merits of command-line and GUI interfaces. Drag and drop is intuitive in that the required action reflects closely what you want to do, the action reinforces the thought. There is no need to worry about typing errors and long, informative variable names can be used. All this is a considerable help in exploratory and initial analyses. For repetitive or routine analyses it can be more of an irritant than a help. Then the action should be neutral, efficient and require little thought (though informative variable names are still a help).

Opening result objects is more complex. In general what would be useful is not just a door to each object which can be opened to reveal what is inside it (basically equivalent to querying) but a hyperdoor which gives us access to the object components and tools to work with them.

Manipulating objects by transforming them, combining them and modelling with

them demands that we understand what they represent. There ought to be links to the procedures and inputs from which they came. In a linear system like R you can always fall back on the argument that a rerunning of commands will remind you where the objects came from. It is a reasonable argument, but not a strong one. In a non-linear system, like Data Desk for instance (Velleman, 1997, [6]), the software does generate the necessary links, but it gets very messy, very quickly. This problem will become larger rather than smaller. Consider as an example fitting a succession of different regressions and then working with the coefficients of the models as a data set. Each ‘variable’ value is now the result of a complex analysis itself and it would be important to keep track of this information. Statistical software allows such analyses but does it support them?

3.3 Hotlinking

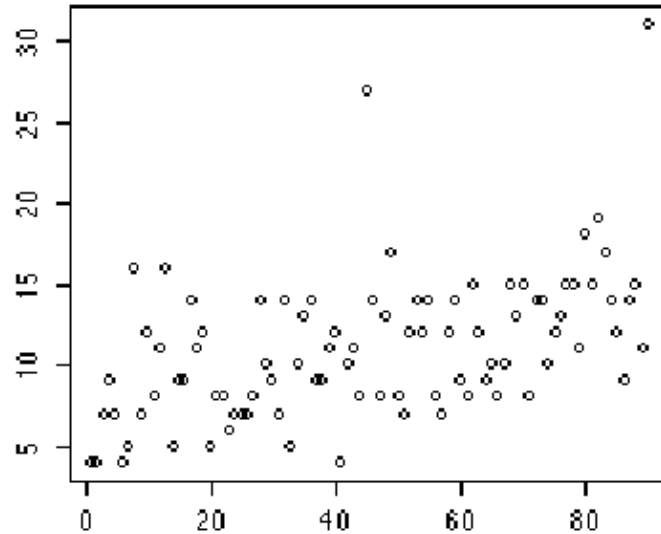
Paul Velleman uses cold, warm and hot linking in Data Desk (Velleman, 1997, [6]). If objects are cold-linked then a calculation once carried out on one to give the other remains unchanged even if the first object is changed. This is like pasting values rather than formulae in Excel and is the situation in R. With warm-linked objects the user has the choice of whether they want to update the second object when the first is changed. Hot-linking means that the change takes place automatically (as with formulae in Excel if recalculation is on). As always increased flexibility means increased care is necessary, but in general this particular flexibility has great potential. What-if and conditional analyses can be carried out exploratively without requiring recoding and new outputs. The power of hotlinking can be seen most impressively in the hotselection facility of Data Desk: redrawing graphs and recalculating models continually according to whatever is the current selection.

3.4 ‘Better’ graphics

These are partly a matter of taste and it is not appropriate to get into aesthetic discussions here, but, for example, is this scatterplot really the best R can do?

The open circles and the resulting overlapping effects are not good even with this small data set. The plot can be enhanced in a variety of different ways, but shouldn’t the default be better? Another example is the barplot command. Bar charts are very useful ways of summarising categorical variables, especially for large data sets. The default in R is to produce individual bars for each case and the command does not work for categorical variables with text categories. The fact that the plot command then works well (though it also produces a case plot when the categories are numerical) is neither here nor there. In considering interfaces what matters primarily is how straightforward and intuitive the commands are to use, given that they are there, not whether they are there or not.

It’s not that R’s graphics are particularly bad (there are far worse examples in statistical software) but that R is good enough to deserve better. The R group disagrees about the standard of R graphics as this quotation from www.R-project.org / underlines: ‘One of R’s strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where



needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.’ This is a strength for presentation graphics, but is not much consolation for exploratory graphics. On graphics in general we had better agree to differ.

3.5 Interactive graphics and controls







R’s graphics are not the sophisticated objects that R’s analytic objects are. Graphics need to be broken down into their statistical components (e.g. it is the columns in a histogram that are relevant statistically, not the bounding lines. It is irritating, though understandable, to copy a picture from some statistics package into a graphics package and discover that it is composed of graphical elements rather than of statistical elements.)

The interactive tools which software should offer are an issue of a different kind and are discussed in (Unwin, 1999, [4]). Principles of GUI design relating to statistical software are outlined in (Unwin et al, 1999, [5]). John Chambers has pointed out how awful GUIs can be (Chambers, 1999, [1]) but many bad examples should not discourage us from aspiring to do better. A major criterion should be whether the control action to carry out a task reflects the task or is orthogonal to it. Consistency of this kind makes for intuitive commands. If you want to reorder an individual bar in a bar chart in MANET you pick it up and move it. (But what is the intuitive way to reorder all bars? Adding a menu item or a button is a non-design solution.) Important from the point of view of implementation is that making static graphics better could be done without major system surgery, while making the graphics

interactive would require changes at a lower system level.

3.6 Restoring of previous states

You can restore a state in R by just running the same commands in order again and stopping at the appropriate point. This is a roundabout way of achieving your objective and diverts attention from the reason for doing it. Possibly you want to scrap everything since and start from this point again, but possibly you just want to quickly check a fit or a value that arose before. Stephan Lauer's TURNER software for analysing categorical data (Lauer, 2000, [3]) did something like this in automatically generating a window which included all contingency table tests carried out. Since combining categories and editing data were allowed there was a respawn button for each test which reproduced the corresponding analysed table. (Interestingly enough, the original idea of storing all tables proved to be much more complicated and less efficient than just crudely recalculating as necessary.)

	df =	<input type="text" value="12"/>	ChiSq=	273.07	<input type="button" value="Respawn"/>
	df =	<input type="text" value="9"/>	ChiSq=	173.79	<input type="button" value="Respawn"/>
	df =	<input type="text" value="6"/>	ChiSq=	174.01	<input type="button" value="Respawn"/>
	df =	<input type="text" value="8"/>	ChiSq=	271.63	<input type="button" value="Respawn"/>
	df =	<input type="text" value="6"/>	ChiSq=	271.49	<input type="button" value="Respawn"/>
	df =	<input type="text" value="3"/>	ChiSq=	1.0526	<input type="button" value="Respawn"/>

More recently Sylvia Winkler's CASSATT (Winkler, 2000, [7]) automatically edits selection sequences to find more efficient descriptions of selected subgroups, but while the tasks are related the aims are not the same: reaching a particular point in the analysis on the one hand and generating a clear subgroup description on the other.

3.7 Multiple paths and multiple output windows

Fitting many models and comparing results is fairly easy in R and a commendable feature, but more control could be provided. The models have to be calculated sequentially (command-line implies command-linear) and neither is the set of results automatically summarised nor are the individual models stored in a list (although lists are an elegant way of storing results in R, you just have to do it by hand). Modern computing power encourages the calculation of multiple models and software should provide the facilities to compare, contrast and combine the results. Developments like Exploratory Modelling Analysis and Bayesian Model Averaging demand ways of working in model space as distinct from in data space. Data Mining analyses would also benefit. The tools required are not so much statistical as

organisational, but that does not mean it should be left to the computer scientists to decide what should be designed.

4 Summary

This paper has been about R and the Human Computer Interface, not about R and its interfaces to other packages. The latter kind of interface has received a lot of attention, the former kind of interface has not. Perhaps this is because it is a difficult task with results that can not readily be objectively measured. If people have been prepared to put up with the standard controls on video recorders for so long, they must be ready to put up with any kind of interface for computer software. It is easy to be a critic and it is not easy to develop complex systems. R has been an extraordinary success but could be even better in a future incarnation. It is unlikely that the current version could be given a fully-fledged GUI and superficial GUIs are worse than no GUI at all. That is one of the interpretations of the subtitle of the paper: if R had a voice, it would quite properly object to having an additional interface. (We might also argue for the interpretation that R would object to being said to have any interface at all given its simple, functional one.)

It has not been the intention of this paper to press for a GUI interface rather than a command-line one. There is a place for both, and it would be interesting to see software that could successfully handle both. That is doubtless a lot more difficult to achieve that we might think. Having two complementary interfaces to the objects in R would not make the system more powerful, but it would empower the users and that is surely the ultimate aim in providing software.

References

- [1] Chambers, J. (1999), "Computing with Data: Concepts and Challenges", *American Statistician*, 53(1), 73-84.
- [2] Hofmann, H. (2000), "MANET", www1.math.uni-augsburg.de/Manet/, Augsburg: Rosuda.
- [3] Lauer, S. (2000), "Turning the Tables with Turner", *SCGN*, 11(1), 5-9.
- [4] Unwin, A.R. (1999), "Requirements for Interactive Graphics Software for Exploratory Data Analysis", *Computational Statistics*, 14, 7-22
- [5] Unwin, A. R., Hofmann, H. (1999), "GUI and Command-line: Conflict or Synergy?", In K. Berk Pourahmadi, M. (Ed.), *Computing Science and Statistics, Proceedings of the 31st Symposium on the Interface*, 31 (pp. 246-253). Chicago: Interface Foundation.
- [6] Velleman, P. (1997), "Data Desk", www.datadesk.com. Ithaca: Data Description.

- [7] Winkler, S. (2000), "CASSATT", www1.math.uni-augsburg.de/Cassatt/, Augsburg: Rosuda.