An Introduction to a Presentation to the Department of Statistics of the University of Bologna, March 15th 2019. The IDEOLOG Program.

Progamming Languages and Programming Methodologies

Yesterday in parliament, I was checking one of the programs that I intend to run today. I was in the process of removing bugs, when the division bell rang and I had to leave my desk in a state of confusion.

I happened to explain the reason for my confused state of mind to one of my parliamentary colleagues; and I asked him whether he had ever had the experience of programming a computer. He told me that he had; and that he had relied on punched cards. This took me back to the distant past; and it reminded me of my first experiences of programming in the late 1960?s.

The cards were punched by the massive Hollerith machines that were weighty and robust so that the perfect alignment of the punched holes could be maintained. Woe betide anyone who dropped their deck of cards and had not taken the precaution of numbering them. To restore them to their proper order would have been a considerable task.

When I began computing, I was given the choice between ATLAS Autocode and the new IBM FORTRAN programming language. Regrettably, I chose FORTRAN. Manchester ATLAS Autocode, which was the progenitor of ALGOL and the grandfather of Pascal, was far superior. It had all the modern control structures such as *if-then-else-if, do while, repeat until,* and so on.

FORTRAN had none of these. It relied heavily on the notorious *GoTo* command that gave rise to the spaghetti programming that was much despised in programs written in the early versions of BASIC

Programs of that era were linear and sequential, passing in a straight line from beginning to end. Nothing much changed in this respect until the advent of the microcomputer, when programs became capable of interacting instantly with their users during their execution.

The first modern micro-computer operating system was that of the Apple Macintosh; and the glory was that it was programmed in Pascal. (later it was converted to C and C++)

Programs were now implemented in Units. From the central main loop of the program, various functions and procedures were accessed that were gathered into such units. The paradigm of an interactive program was the word processor, wherein every function was meant to be accessible from the main page, primary via pull-down menus.

The sort of programming that I have pursued has been one in which the interactions are via interrogations and command lines that are mediated by a text console. Much of the output is via the graphics console. In this way, I have managed to avoid the fancy stuff of commercially orientated programs.

Unfortunately, I have had to forego this methodology—at least in so far as I intend to produce programs that are available to Windows users. The reason for this has been the failure of the authors of the Free Pascal compiler to maintain the legacy graphics unit on which I have been relying.

They have not produced a version that works adequately on the 64-bit computer with which I am currently working—this one. Therefore, I have had to resort to the Lazarus/Delphi compiler that has been built on the basis of Free Pascal.

The consequence is that there are now two versions of my IDEOLOG program that demonstrates the methods of linear filtering. The older version relies on command lines, whereas the newer version employs a more modern or commercial style of user interface. The latter version is still under development; and it lacks some of the essential features.

Delphi and Lazarus adopt an object-orientated methodology that is common to modern programming environments. The object-oriented philosophy with its features of inheritance and polymorphism is what lies behind the graphical objects of a modern user interface.

As far as the ordinary programmer is concerned, these features are not an issue. The challenge arises from the structure of the interface, which depends on a succession of forms that contain the numerous widgets and gadgets by which the user interacts with the program. There is much circumstantial detail that accompanies these devices, which has to be learned.

The devices include list boxes, edit boxes, radio buttons and captions and text displays that change in consequence of the options selected by the user. They consume large amounts of computer memory. Care must be taken, therefore, to dismiss from memory the forms that are not in immediate use.

In my opinion, a statistical program, in contrast to a word processor, should not attempt to create a modeless environment that gives immediate access to all of the available functions. Instead, it is best to drive the users down narrow, predefined routes toward whatever may be their ultimate objectives. This explains the absence from my program of the usual panoply of pull-down menus.

The Motives of the IDEOLOG Program

The IDEOLOG program has been designed to illustrate a wide variety of procedures in linear filtering. A common opinion of econometricians is that linear filters should be based on dynamic stochastic models that can be fitted to the data. Such models typically comprise a combination of statistically independent components that represent trends, cycles, seasonal fluctuations and noise.

For any such model, there will be filters that represent the optimal means of extracting the various components. A difficulty can arise when the model does not adequately represent the essential features of the data. It may even prove to be impossible to fit a model to the data because of the evolving nature of the underlying processes.

I prefer to avoid such difficulties by adopting a nonparametric approach that is guided by the salient features of the data rather than by the properties of a fitted model. I am guided principally by the spectral structure of the data, which is the frequency-domain counterpart of it autocovariance structure.

These structures are relevant only to data that are devoid of global trends. Therefore, to pursue a spectral analysis, any trend that is present must be removed from the data. The data should also have a reasonable homogeneity over time. The amplitudes of their variations should be roughly constant.

The increasing amplitudes of the cyclical and the seasonal variations that characterise many econometric data sequences can often be overcome simply by taking logarithms of the data. The trend in the data can be removed by taking differences or by interpolating a polynomial function and by working with the residual deviations.

If the differences are taken, then the trend can be restored to the filtered data by a process of anti-differencing or summation that requires a careful determination of the initial conditions. Alternatively, if a polynomial trend has been extracted, then it can be added back to the filtered sequence of the residual deviations. The program pursues both techniques.

Amongst the procedures that are illustrated by the program are some new ones that perform the necessary computations within the frequency domain by working with the Fourier ordinates of the data. Also included are some time-domain procedures that are used frequently by econometricians. I am very critical of some of these procedures. Their inclusion is intended to illustrate the errors that can arise from disregarding the spectral structure of the data.