APL and the Teaching of Statistics

Peter Naeve

1. INTRODUCTION

Undoubtedly the statistical community has become aware of the great possibilities of modern computer facilities. This might be seen by the ever increasing list of books and articles dealing with such themes as 'statistical computing', by the growing number of conferences and symposia devoted to the impact of computer on statistics and by numereous computer programs ready made for statisticians' use. As there are already many papers on the special topic 'computers in the teaching of statistics' too (Evans (1973), Naeve (1978)) justification must be given why one more paper is added.

With respect to teaching all this books, articles and conferences do not tell the true story. One gets an impression of a world that is wishful thinking compared with every days teaching of statistics. This is especially true for the role of APL in that behalf.

With respect to APL computer scientists and programmers are divided in two hostile parties. This attitude seems to have been carried over to statisticians without paying much attention if an argument that might be convincing from a computer scientist's point of view keeps its place when considered from a statistician's point of view. Or if there are other properties of APL which although of little value for a computer scientist make APL such a rich tool for a statistician.

The following lines do not intend to give an introduction to APL. The APL novice may take Gilman, Rose (1974), Pakin (1972) or Polivka (1975) as an advice. They are meant as a list of arguments why one should use it.

2. APL AS A NOTATIONAL LANGUAGE IN STATISTICS

The richness of build in primitive functions and their wide scope allows for almost one to one mapping of formules written in conventional mathematical notation and executable 'programs'. To give one example if one had to evaluate

$$SS = \hat{B}'C(C'GC)^{-1}C'\hat{B}$$

this could be done in APL as follows

```
SS ← CB +.×(⌹(⍉C)+.×G+.×C)+.×CB ← (⍉C)+.×B   .
```

As Evans (1973) points out 'the programming details are minimized and one can get on to the statistical implications.

The feature of user defined functions eases the development of an even more statistical tailored notation. Direct function definition even if not yet always implemented is of special value in this respect. The following example is an modified one from Rosenkrands (1974).

```
CORR  :  (SPD ⍵)÷SQRT (SSD ⍵)∘.×SSD ⍵
SSD   :  1 1⍉SPD ⍵
SPD   :  (⍉DEV ⍵)+.×DEV ⍵
DEV   :  ⍵-(⍴⍵)⍴MEANS ⍵
MEANS :  (+/[1]⍵)÷(,⍴⍵)[1]
```

It is easily seen that APL even allows for a top down approach - one of computer specialists' favorite. Having introduced this set of functions which altogether compute the correlation matrix for n variates in the sequel *CORR DATA* 1 or *CORR X* are easily understood notations.

3. APL IS STRUCTURED

This statement will be strongly contradicted by most computer scientists.

But it holds. The fact is that the meaning of 'structured' is almost totally linked to logic-structured programming as Metzger (1980) points out. With respect to while do , repeat until and other constructs APL surely is not structured at all. But if one takes the data on which an algorithm will act into consideration one soon notices that APL is structured with respect to data (Metzger (1980)). Take for instance a contingency table. As seen in figure 1 it is made up of four parts.

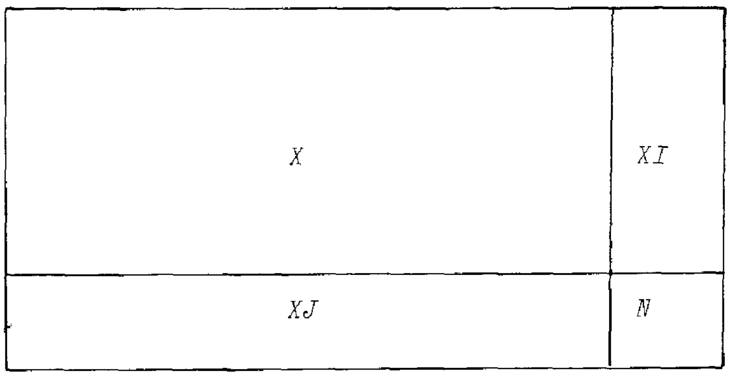


Fig. 1.: contingency table

The four parts are

X	observed cell frequencies
XI	row margins (row totals)
XJ	columns margin (columns totals)
N	number of observations (over all totals)

Let *XC* denote the contingency table than one might construct *XC* with the following APL code starting from a given *X* .

```
XI ← + / X                  ⍝  row totals
XJ ← + / [1] X              ⍝  column totals
XJ ← + ⌿ X                  ⍝  equivalent expression
N  ← + / + / X              ⍝  over all total
N  ← + / + ⌿ X              ⍝  equivalent expression
XC ← XX,[1]+⌿ XX ← X,+/X    ⍝  XX  is matrix  X  augmented with
                               columns totals
```

One should contrast this with the following PASCAL code to do the job of calculating column margins.

```
for j := 1 to n do
begin
xj [j] := 0
for i := 1 to m do
     xj[j] := Xj[j] + x[i,j]
end
```

But remember there has to be a declaration part too and you are forced to provide all the necessary space in advance for PASCAL (like all the other 'high level well structured' languages) offers no 'catentation' like feature.

4. APL EASES THE TASK OF REWRITING ONE'S PROGRAMS

Even in situations where one did approach a statistical programming task according to the many rules of software engineering one might face many reasons afterwards why the program should be rewritten. The list starts with questions of optimization either computing time or space and ends with questions of better understanding (self explanatory programs). But as Abrams (1973) states: 'program rewriting is relatively rare in the 'real world' of traditional programming languages'. Due to the power of the build in functions APL programs are usually much shorter than programs in any other language doing the same job. This fact and the interactive nature of APL makes the task of rewriting an easy one. But rewriting one's program is more than just a technical matter. Rewriting means rethinking and reanalyzing the algorithm. Rewriting will give deeper insight into the problem and its solution. Abrams (1973) illustrates these findings in connection with the problem of calculating the critical path of a project network.

What follows is another exmaple done by an graduate student - which demonstrates the benefits one gets from rewriting. Reaching deeper insight clearly is a valuable goal for a student. The ease with respect to programming time and effort has not been measured in this example. The problem was to define a function to do the job of the Sweep Operator. The algorithm may be presented as follows :

```
§ data matrix to be swept, b result: swept matrix
  r index of row and column which are to be exchanged §
  $b_{rr} \leftarrow 1/data_{rr}$
  $b_{ir} \leftarrow data_{ir}/data_{rr}$                              $i \neq r$
  $b_{rj} \leftarrow data_{rj}/data_{rr}$                              $j \neq r$
  $b_{ij} \leftarrow data_{ij} - (data_{ir} \times data_{jr})/data_{rr}$          $i,j \neq r$
```

The first solution offered was:

```
      ∇ Z←K SWEEP A ; AS1 ; AR ; AC
[1]     H←ρA
[2]     N←H[1]
[3]     T←(ιN)-(ιN)
[4]     T[K]←1
[5]     TINV←(((ιN)-((ιN)-1))-T
[6]     AC←(TINV/[1](T/A))
[7]     AR←(TINV/(T/[1]A))
[8]     R←TINV/[1](TINV/A)
[9]     AS1←R-(AC+.×AR)÷A[K;K]
[10]    AS←(TINV\[1](TINV\AS1))+(T\(TINV\[1](AC÷(¯1×A[K;K]))))+(
        T\[1](TINV\AR÷A[K;K]))
[11]    AS[K;K]←1÷A[K;K]
      ∇
```

Although there are no loops in that function as would be in an equivalent FORTRAN, PASCAL or ALGOL program it is far away from a 'good' APL functions. But at the moment our concern are not problems like to many (unnecessary) brackets, to many lines - for instance one could easily combine line [1] and [2] to get *N*←(*H*←ρ*A*)[1] or even better *N*←(ρ*A*)[1] or equivalently *N*←1↑ρ*A* . The main concern is that at this stage the student has not been fully aware of the fundamental difference between APL and the traditional programming languages. In contrast to the latter APL calls for thinking in higher dimensional structures. Although the traditional programming languages allow for e.g. two dimensional arrays their syntax usually enforce upon a one dimensional (element by element) way of processing. Compare for instance matrix addition in APL

```
    C ← A + B
```

and in FORTRAN

```
          DO     10     I = 1,N
          DO     10     J = 1,N
    10    C(I,J) = A(I,J) + B(I,J)
```

So when introduced to this point of view the student delivered after some thought the following function :

```
        ∇ AS←K SWEEP DATA
    [1]   F←(N←(⍴DATA)[1])⍴1
    [2]   F[K]←¯1×DATA[K;K]
    [3]   PR←(TE←K=⍳N)/[1](FID←DATA÷F∘.×|F)
    [4]   PCOLD←TE/DATA
    [5]   PR[1;K]←PCOLD[K;1]←0
    [6]   AS←FID+PCOLD+.×PR
    [7]   AS[K;K]←|AS[K;K]
        ∇
```

Still not perfect but much more in line with APL-thinking. And besides a more concise function the student surely has reached a level of deeper understanding of the whole procedure. Without the process of rewriting this insight scarcely would have been gained.

5. APL EVOKES CREATIVITY

Recently some students judged statistics to be boring compared to probability theory for in the latter there were usually several lines of attack to find a solution to a problem. Although one might find some strong arguments against their opinion seen from a novice point of view there is some truth in it.

But when one incorporates APL in the process of learning the before mentioned opinion turns out to be completely wrong. Take for instance the problem of calculating the ranks for a given vector *X* of observations when ties are possible.
Smillie (1974) gives the following solution :

```
(P+.×V)÷+/P←V∘.=V←X[⍋X]   .
```

Whereas Snyder (1973) comes up with this line of code :

```
.5×(⍋⍋X)+⍒⍋⌽X   .
```

Let us take another simple problem: how to evaluate a polynomial

$$P(x) = \sum_{i=0}^{n} c_i x^i$$

Here is a list of some functions to do the job.

```
POL1  :  +/α×ω*¯1+ιρ,α
POL2  :  (ω*¯1+ιρ,α)+.×α
POL3  :  (×\¯1↓1,(ρ,α)ρω)+.×α
POL4  :  ω⊥⌽α
POL5  :  (ω∘.*¯1+ιρ,α)+.×α
POL6  :  (×\0 ¯1↓1,⍉((ρ,α),ρ,ω)ρω)+.×α
POL7  :  (,ω)[((ρ,ω),1)ριρ,ω]⊥⌽α
```

Let *C* denote the vector of coefficients and *X* the argument for which the polynomial shall be evaluated then the result is given by calling

*C POL*i *X* .

The last three functions are also capable of handling vector arguments i.e. the polynomial with coefficient vector *C* is evaluated at several points x_j . *POLi*,i= 1,2,5 are adapted from Iverson (1972). APL and efficient statistical computing are not contradictory. There is always the temptation to turn APL down because it is an interpretative language and therefore a natural second best compared to compiler languages with their run time superiority.

But first of all we must be aware that this is a computer oriented type of argument. When a computer specialist's main concern lies in effective use of his computer with respective to optimization (minimization) of CPU-time and memory space this cannot necessarily be a statistician's point of view. It is easy to produce with highly efficient code very fast a lot of garbage. So a second thought will show that it is not the time of the computer but our time that matters as Dennis Evans emphazised during the discussion following the presentation of his paper (Evans (1980)) at COMSTAT 1980.

Statistics calls for insight in the process evoked by a statistical procedure i.e. the link between data and algorithm, the sequence of patterns which are build up and so on.And here APL is a useful tool as hopefully is shown by the foregoing arguments. So statisticians should hold their own interpretation of efficiency against that of the computer scientist when negotiations about implementing APL at the computer center are held.
This does not mean that the other interpretation of efficiency shall be completely denied. As Kennedy and Gentle (1980) again and again point out: 'the purpose ... is to present selected computational methods, algorithms, and other subject matter which is important to the effective use of the given methods and algorithms in the computer'.

As might be seen in the before mentioned example by Abrams (1973) one might combine those questions of efficiency in the sense of Kennedy and Gentle with the aim to get deeper insight in the problem and the applied methods to reach a really efficient (fast and clear) solution.

To write efficient programs in any high level language demands some knowledge of the structure and properties of the host computer and some insight in how the compiler does his job. But there is nothing different with APL. As Sykes (1973) states the essential point is 'know your operators'. After some trial and error and some thought the results given in the following table concerning the time for functions POLi, i=5,6,7 on an IBM 5110 will become clear.

POL	*C*←ι51 *X*←ι20	*C*←.5+ι51 *X*←ι20	*C*←.5+ι51 *X*←.5+ι20	*C*←ι26 *X*←.5+ι50
5	36	31.4	31.7	41.1
6	23.3	19.4	18.6	24.9
7	12	9.6	8.8	12.9

Fig. 2: Computing time (in sec)

6. WHAT KIND OF STATISTICS SHOULD WE TEACH ?

It seems to be wise to stop for a moment in talking about the advantages of APL. The phrase 'the teaching of statistics' rises two questions.

Firstly :	What should be taught, i.e. what kind of statistics ?
Secondly :	How could the goals be achieved, i.e. in our context : can APL help in reaching these goals ?

Although the answers to both questions cannot be given independently, it appears as if the enthusiasm over that powerful tool computer often made people skip the first question. As a result much effort and money (computer time, programming cost etc.) has been wasted on projects which rank low in an overall list of teaching goals and connected means.

The present author shares the point of view expressend by Tukey (1980). If one agrees that statistics is both confirmatory and exploratory this calls for a more subtile analysis of the role the computer may play in teaching statistics.

7. APL AND CONFIRMATORY DATA ANALYSIS

As Tukey puts it: 'confirmatory data analysis, especially as sanctification, is a routine relatively easy to teach and, hence a routine easy to computerize'. Confirmatory analysis uses a body of well established statistical methods and procedures. Tukey is not to be misunderstood. Implementing any such method on a computer as a dependable, efficient and useful computer program is not an easy task. Many things have to be considered among which the choice of the programming language is only one. APL is here just one out of several competitors.

In passing let us notice that the body of methods used in comfirmatory analysis has been widened tremendously through the existence of computers. Many more methods became computable with real data which had to be abondened before in the framework of pencil and paper calculations.

Confirmatory analysis cannot be done without understanding the methods which will be used. This obvious fact can be made more specific by breaking it into three parts.

Understanding the methods means

firstly	knowing how to interprete the results, knowing what kind of decisions and conclusions might be drawn from the result.
secondly	understanding the algorithm within the statistical procedure.
thirdly	knowing the way algorithm and data are linked together, understanding the dynamic properties of the procedure.

As has been shown elsewhere (Naeve (1978)) the computer can be used in the process of teaching statistics on three different levels - named black box, glass box and open box - which roughly correspond to the before mentioned aspects of understanding. Due to lack of space this point will not be elaborated here any further, the interested reader is asked to see the cited paper for more details.

At the moment there is some discussion whether one should rely on an interactive or batch mode in using the computer (Klensin and Dawson (1980), Cooper and Emerson (1980)). This truely is an important question and he who is free in the choice of his tools should spend considerable time in wheighing the pros and cons. But all the other users have to take their computer as it stands i.e. bound to batch or capable of interactive usage. But independently of the computer at hand all designers of statistical software have to answer the question as to where the statistician should be placed - inside the computer or in front of it.

To make this question more transparent two examples of computer output are presented. The first is part of a terminal session with the regression analysis modul of the IBM product APL Statistical Library.

```
                    :
THE SIGNIFICANCE OF ABOVE F = 0.9787
THIS FIT IS NOT CORRECT AT THE SIGNIFICANCE LEVEL:0.95
TRY A HIGHER ORDER OR DIFFERENT MODEL
```

The objections are twofold. Although for an experienced statistician there is nothing wrong with the first two lines things change when one imagines a statistical novice or nonexpert sitting at that terminal. Far too often he will not understand the information given in the first line and so blindly is caught by the message.

```
THIS FIT IS NOT CORRECT AT THE SIGNIFICANCE LEVEL:0.95
```

As most users of this type he only knows two significance levels i.e. 0.95 or 0.99 and applies them without giving much thought to the meaning of the concept of significance level. A program should not enforce such an attitude. A prompt for the significance level at the start of the program - as in this case - is no cure. One might say with respect to understanding : garbage in, garbage out.

The advice given in the third line seems to be a nice feature. But is this really so. In case there is no linear model at all which would fit to the data. Is this incorporated in *OR DIFFERENT MODEL* or is the meaning of these words: try another linear model i.e. take other independent variables into consideration ? So either the advice might be misleading or is of the kind: try something else.

This point or view may look a bit dogmatic. But one has to be somewhat strict in the beginning if one wants to implement statistical thinking in the user's head and not in his computer.
The following sample from the output produced by a similiar program are more along the recommended line of approach. The program was designed by a student of mine.

```
KONFIDENZINTERVALLE FUER DIE REGRESSIONSKOEFFIZIENTEN

Konfidenzintervall für den Niveaukoeffizienten:
P(362.07+-T(1-α/2;8)×13.906) = 1-α
Konfidenzintervall für den Regressionskoeffizienten:
P(53.314+-T(1-α/2;8)×3.627) = 1-α
```

8. APL AND EXPLORATORY ANALYSIS

Exploratory analysis is an ever growing body of helpful techniques. Although many of them can be applied on the pencil and paper level (perhaps with the need to use multi-color-pencils as Tukey (1977) would say)computer assistance might be welcomed. Several software products are available see Dawson, Klensin and Yntema (1980) or McNeil (1977). As might be judged by the latter APL can contribute substantially in this respect.

But exploratory is more than just a bundle of useful techniques and procedures, it is an attitude as Tukey (1980) puts it. Exploratory analysis in the long run cannot be done with fixed tools and routine lines of approach. Creativity ranks high and should not be hindered by provided programs. Although at a first glance the non-routine data analytic applications use many of the standard procedures and data processing tools there is a distinguishing characteristic - the necessity to allow for human thought and intervention (Heiberger (1979)).

There are many reasons why APL is suited best for the requirements of exploratory. Exploratory analysis calls for rethinking and reconstructing one's tools. As has been shown this task is eased with APL as the programming language. The workspace oriented concept of APL, the dinstinctions between global and local variables and the concept of a function quite easily offer the opportunity to enlarge the bundle of tools. The possibility to process in deskcalculator mode every global variable gives way to all kind of ad hoc manipulations with the data. The build-in function ⍎ (activate) allows for user intervention with software products. This might be seen by the following example.

Let us consider univariate Box Jenkins time series analysis. The time series data are kept in a variable *TDATA* . Usually the program would offer (for instance using a menu technique) some options to transform the data. One might bet that even the most careful planed program will face a user who is not satisfied by the options provided. Here the activate feature may help. The following line of code is a first simple solution

```
    TDATA ← ⍎ ⍞    .
```

The user might for instance answer the prompt by typing

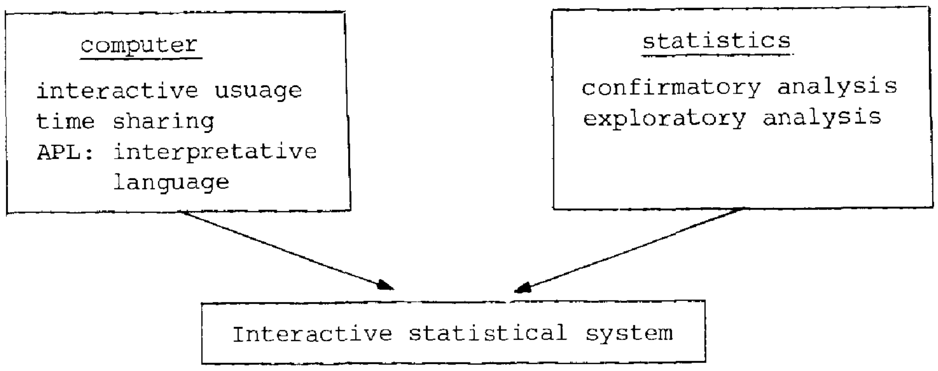
```
    4×TDATA - 1↓TDATA , 0
```

or whatever kind of transformation he wants. This line of approach surely can be widened to incorporate syntax checking (avoiding ⍎ *ERROR*) or taking input via shared variables into consideration.

Last but not least if analysis is an attitude with creativity as a vital content then APL seems to be the right host language for it supports creativity.

9. LEARNING BY DOING

To reach an attitude is a very difficult task. One promissing approach could be this. What has been said about computers and statistics can be combined in the following display.



An interactive statistical system is something like the synthesis of the arguments and facts given before. It is a statistician oriented (speaking and understanding his language) system which does not pin the user to think in terms of computer science. Instead of that it allows for human intervention. The idea of such a system has been brought up by Mustonen (1977, 1980). He too showed that it can be realized even on a desk computer. It is evident that the teaching of statistics would benifit if such a system is at hand for the teacher.

If such a system is not available so start and implement one. All what is needed is APL. If the students are allowed to participate in building moduls for such a statistical system one will find out that they learn much more than APL (best learned on the job) and certain statistical procedures. They develop step by step that kind of attitude Tukey calls for. This process benefits from the ease of rewriting programs (this means rethinking in the statistical field) as given by APL.

The interpretative nature of APL allows for merging the usually separated phases of design, coding and checking (syntax, logic and acceptance). Leading princibles in the design such as, the user is a statistician and not a computer specialist, the statistical decisions should be made in front and not inside the computer quite easily can be pushed down onto the different levels of application during the process of program development.

This is something like a princible guided trial and error process. And students participating in that work will benefit for they get an ever growing understanding what statistics as an attitude really means. This is not wishful thinking. At the moment the author is working on such a statistical system to be implemented on a IBM 5110. APL as the reader may guess is the chosen programming language. Several students participate in this work. And they really demonstrate that the before mentioned effect is achieved. They all show a shift in their understanding about the nature of statistics in that direction Tukey wants us to teach.

References

Abrams Ph.S. (1973), Program writing, rewriting and style, APL Congress 73 Gerløv P.,Helms H.J., Nielsen J..

Cooper B.E., Emerson M.J., (1980) Interactive or batch, Compstat 1980, proceedings in computational statistics, Ed. Barrit M.M.,Wishart D..

Dawson R., Klensin J.C., Yntema D.B. (1980), The consistent system, The American Statistician vol 34 p 169 f..

Evans D.A. (1973), The influence of the computer on the teaching of statistics, J.R.Stat.Soc. A 136 p 135 f..

Evans D.A. (1980), APL 84 - an interactive, APL based, statistical computing package, Compstat 80 proceedings in computational statistics, Ed. Barrit M.M., Wishart D..

Gilman L., Rose A.J. (1974), APL - an interactive approach, 2.ed, J.Wiley, New York.

Heiberger R.M. (1979),Software for statistical theory and practice, Technical report 42, Department of Statistics, The Wharton School, University of Pennsylvania.

Iverson K.E. (1972), Algebra, an algorithmic treatment, Addison Wesley, Reading.

Klensin J.C., Dawson R. (1980), Interactive computing versus computing in an interactive environment, Compstat 1980, proceedings in computational statistics, Ed. Barrit M.M., Wishart D..

McNeil D.R. (1977), Interactive data analysis, Wiley New York.

Metzger R.C. (1980), Extended direct definition of APL functions, APL 80, Ed. v.d. Linden G..

Mustonen S. (1977), Survo 76, a statistical data processing system, Research report 6, Department of Statistics, University of Helsinki.

Mustonen S. (1980), Interactive analysis in Survo 76, Compstat 1980, proceedings in computational statistics, Ed. Barrit M.M., Wishart D..

Naeve P. (1978), CAI and computational statistics, Compstat Lectures I, Ed. Skarabis H., Sint P.P..

Naeve P. (1979), Some aspects of the teaching of statistics, discussion paper 60, University of Bielefeld, Department of Economics.

Pakin S. (1972), APL\360 Reference manual, Science research Associates.

Polivka R.P. (1975), APL : the language and its usage, Prentice Hall Englewood Cliff.

Rosenkrands B. (1974), APL as a notational language in statistics, Compstat 1974, proceedings in computational statistics, Ed. Bruckmann G., Ferschl F., Schmetterer L..

Smillie K.W. (1974), APL\360 with statistical application, Addison Wesley, Reading.

Snyder M. (1973), Interactive data analysis and nonparametric statistics, APL Congress 73, Ed. Gerløv P.,Helms H.J., Nielsen J..

Sykes R.A. (1973), Use and misuse of APL, efficient coding techniques, Scientific Time Sharing Corporation, Share XL.

Tukey J.W. (1977), Exploratory data analysis, Addison Wesley, Reading.

Tukey J.W. (1980), We need both exploratory and confirmatory, The American Statistician vol 34 p 23 f..