MARKETING EDUCATION AND APL

E. K. Valentin, Weber State University

ABSTRACT

APL is a very powerful interpreted programming language. Marketing educators will find it well-suited for disposing of innumerable clerical tasks and for bringing analytical concepts to life in the classroom. In this article, APL's distinctive features and applications are delineated.

INTRODUCTION

APL (a programming language), which is widely used by actuaries, engineers, scientists, and mathematicians, fills a gap left by other computer languages and packages, including spreadsheets and statistical software. This article provides an overview of APL's advantages and drawbacks and suggests how marketing educators can use APL in and out of the classroom.

DISTINCTIVE FEATURES

Advantages

Although it is possible to write programs in APL that resemble their BASIC, FORTRAN, or Pascal equivalents in structure, initially, it is best not think of APL as a computer language at all. Instead, APL should be viewed as a compact and powerful system of notation for writing expressions resembling matrix algebra that a computer can execute (Iverson 1962). By entering X←5 3.1 .7, for example, a three-element vector labelled X is created. (The arrow is a special APL character). Note that nothing about X, such as its dimensionality, was declared in advance. Moreover, integers and rational numbers can be mixed freely; scalars and multidimensional arrays can be created as readily as the vector X; and m-dimensional arrays are easily transformed into n-dimensional arrays.

If +/X is entered, the sum of the elements comprising the vector X is displayed; and if Y←+/X is entered, the result (i.e., the sum of X) is stored in a new variable, labelled Y. Accordingly, the APL operator +/ is equivalent to the familiar summation operator Σ. If X were a numerical matrix, entering +/X would display the sums of the rows; and +/+/X would display the sum of all values comprising X (i.e., APL would calculate the sum of each row and then add these subtotals). Other APL operators (also called predefined or primitive functions) can be used just as easily, for instance, to multiply matrices, transpose matrices, invert matrices, estimate least-squares coefficients, find the dimensions of arrays, re-dimension arrays, sort data, transform data (e.g., to logarithmic values), create arrays of random numbers, and generate factorials (Finnish APL Association 1982; Gilman and Rose 1984). A more extensive sketch of how APL's standard operators can be applied to numeric vectors and matrices, Boolean vectors, and character vectors and matrices is provided in the Appendix.

An APL statement may contain many operators and, therefore, perform many tasks. For example, in a single statement (which need not exceed seven characters), a matrix may be transposed, multiplied by another matrix, and inverted. Such statements can be typed and executed ad hoc in the so-called immediate mode or organized, saved, and used repeatedly much like Pascal units and objects, FORTRAN subroutines, and BASIC programs. However, Rubinstein and Lewis (1984, p. 230) drew the following parallel between APL and BASIC, which also applies to other conventional programming languages:

> From the perspective of APL, programming in BASIC is comparable to trying to speak English with a 100 word vocabulary. Even with this restriction, you could probably get along, but it would take a long time to make yourself understood.

APL is especially well-suited for non-repetitive analytical work because tasks can be coded compactly and quickly; and as shown above, APL operators actually can be applied to data arrays without writing programs, per se. Moreover, APL programs often can be developed faster than flow charts or pseudo-code, which makes APL well-suited for prototyping.

Drawbacks

APL is interpreted and, therefore, cannot produce compiled "stand-alone" programs. And although APL is the preferred language of many actuaries, engineers, mathematicians, and scientists, most business school graduates and managers favor electronic spreadsheets. Moreover, those who have had an introductory lesson in APL programming are much more likely to hate the language than to love it. Why? Whereas electronic spreadsheets mimic the manual counterpart with which nearly every business student and manager has some familiarity, APL resembles the mathematical hieroglyphics many business students deliberately avoid. In addition, because APL requires many non-standard symbols (see the Appendix), the keyboard is redefined to accommodate the special characters. Using the redefined keyboard is awkward and confusing at first, even with a template that shows where the APL characters are located. Moreover, APL aficionados typically prefer to impress novices by demonstrate the language's exotic and complex capabilities rather than the simple, but powerful, features, which are easily understood.

ILLUSTRATIVE APPLICATIONS

Statistical Analysis

Because APL symbols resemble matrix notation, statistical programs can be developed with comparative ease by translating matrix expositions found in statistics texts into APL code. For that reason, several commercially available statistical packages, including Statgraphics and Stat 1, were developed initially in APL. They were distributed with an APL interpreter that would run programs, but could not be used to write them.

Today, many excellent statistical packages are readily available; yet, APL remains an invaluable self-learning and teaching aid. Anyone who learns more from tracing numerical examples and working exercise problems than from abstract symbolic expositions, will find APL the perfect "number cruncher." Years ago, I wrote several statistical programs (e.g., regression, principal components, discriminant, and cluster analysis routines) because I did not have access to a good statistical package. Writing such programs enhanced my understanding of statistical analysis and provided me with several programs and utilities that are still better suited than commercial packages for demonstrating various aspects of statistical analysis in the classroom. Also, when fortified with prompts, they are virtually "idiot proof" and much easier to use than commercial packages, especially when the data set to be analyzed is not extremely large.

Data Entry, Editing, and Formatting

Data entry and interviewing screens can be constructed almost as easily in APL as in Ci2. Moreover, the same kinds of range checks and branching can be built in. Furthermore, APL's operators are well-suited for identifying likely data entry errors and outliers; and complex data transformations can be made much more easily using APL than using SPSS, for instance. After data are checked or transformed, they can be exported from APL work spaces as ASCII files, which can be read by almost any statistical package, electronic spreadsheet, and word processor. Outputs from statistical packages, too, can usually be exported as ASCII files, which are easily imported into APL work spaces. I have used APL to reformat SPSS output so as to meet the requirements of various journals, to conduct supplementary analyses, and to plot group means on semantic differential scales along with each (unabridged) questionnaire item.

Pedagogical Applications

Marketing educators will find APL invaluable in constructing illustrations when teaching marketing research, marketing management, and other courses rich in statistical and quantitative content. I have used APL not only to illuminate statistical principles, such as the Central Limit Theorem, but also to illustrate the diffusion of innovations, the relationship between cash flow and sales over the product life cycle, retail gravitation and store location principles, and brand switching behavior. Moreover, APL is often much handier than any spreadsheet for analyzing financial and other data found in cases and for developing quantitative handouts and exam problems.

Chores

APL is useful in disposing of countless office chores; e.g., recording exam scores, calculating and plotting distributions, determining final grades based on the best m of n weighted scores, converting letter-grades to their decimal equivalents, converting

numerical scores to letter grades, etc. Few, if any, grade book packages approach the flexibility that is easily programmed into an APL-based grade book.

CONCLUDING COMMENTS

This article provided a glimpse of APL's power and potential applications. APL's major drawbacks are that programs cannot be run without the interpreter, and it takes a while to get used to the APL character set and keyboard layout. A more detailed exposition of the language, its syntax, and applications can be found in the widely used manuals by Gilman and Rose (1984) by Turner (1984).

Unfortunately, sellers of APL packages (mainly STSC and IBM) do not seem to realize that, even though business students and faculty may learn to love APL (Valentin 1989), only engineers and mathematicians are likely to fall in love with APL immediately. Accordingly, STSC's APL*PLUS/PC is available to students for no less than $225 (plus shipping); and although STSC does sell Pocket APL for as little as $35 (plus shipping), the editing features of this stripped-down version of APL*PLUS/PC are so awkward that most Pocket APL customers surely run out of patience long before discovering APL's "magic."

APL2 is a recent extension of APL, which is even more powerful. For demonstration disks and further information regarding APL*PLUS/PC and APL2 contact STSC, Inc. (301-984-5000) or IBM (800-IBM-2468), respectively.

REFERENCES

Gilman, Leonard and Allen J. Rose (1984), APL: An Interactive Approach. New York: John Wiley and Sons.

Finnish APL Association (1982), Finn APL Pocket Idiom Library. Helsinki: Finnish APL Association.

Iverson, Kenneth E. (1962), A Programming Language. New York: John Wiley and Sons.

Rubinstein, Mark and Stephen D. Lewis (1984), "APL: A Language for Modern Times," PC Magazine, 3 (April), 229-39.

Turner, Jerry R. (1984), APL Is Easy! Rockville, MD: STSC, Inc.

Valentin, E. K. (1989), "APL: The Missing Link in Decision Support Systems?" Journal of Computer Information Systems, 29 (Summer), 26-29.

APPENDIX

Some Illustrative APL Examples

The listed data sets are used in the illustrative examples.

```
Numeric vectors:  Nv = 5 2 ¯3 1 7 ¯8      Ra = 1 0 1 0 1 1      Rb = 0 0 1 1 1 1

Numeric matrix:  Nm = 3 ¯2 8              Character vector:  Cv = ABCDEF
                      ¯6 7 9
                      4 6 ¯3

Character matrix:  Cm = ABCD
                        EFGH
                        IJKL
```

	Entry	Result displayed	Comments
1.	Z←6 3.4 0 ¯3	None	Assigns integers and/or decimals to Z
2.	Y←Z	None	Assigns contents of Z to Y
3.	Y	6 3.4 0 ¯3	Displays contents of Y
4.	ρNv	6	Shape of Nv, a 6-element vector
5.	ρCm	3 4	Shape of Cm, a 3×4 matrix
6.	2 3ρNv	5 2 ¯3 1 7 ¯8	Reshapes Nv into a 2×3 matrix
7.	6ρ'><'	><><><	Fills 6 spaces with "><"
8.	6+5	11	Adds scalars
9.	Ra+Rb	1 0 2 1 2 2	Adds vectors
10.	4+Nv	9 6 1 5 11 ¯4	Adds scalar to vector
11.	Nm+3 3ρNv	8 0 5 ¯5 14 1 9 8 ¯6	Reshapes Nv, then adds to Nm
12.	+\Nm	3 1 9 ¯6 1 10 4 10 7	Displays cumulative row totals
13.	+/Nm	9 10 7	Adds rows
14.	+⌿Nm	3 ¯2 8 ¯3 5 17 1 11 14	Displays cumulative column totals
15.	+⌿Nm	1 11 14	Adds columns
16.	-Nv	¯5 ¯2 3 ¯1 ¯7 8	Reverses signs
17.	Ra-Rb	1 0 0 ¯1 0 0	Subtracts vectors
18.	-/Ra	2	1-0+1-0+1-1

19. -\Ra	1 1 2 2 3 2	Cumulative results for 1-0+1-0+1-1
20. ×Nv	1 1 ¯1 1 1 ¯1	Shows signs of Nv elements
21. 5×Nv	25 10 ¯15 5 35 ¯40	Multiplies a scalar by a vector
22. ×\Nv	5 10 ¯30 ¯30 ¯210 1680	Displays sequential products
23. ×/Nv	1680	Displays final result of chain multiplication
24. Ra×Nv	5 0 ¯3 0 7 ¯8	Multiplies vectors
25. ÷2 4	0.5 0.25	Calculates reciprocals
26. ÷/100 4	25	Divides first element by second element
27. Nv÷2	2.5 1 ¯1.5 0.5 3.5 ¯4	Divides a vector by a scalar
28. *1 0 2	2.7 1 7.4	Raises e to the specified powers
29. Nm*2	9 4 64 36 49 81 16 36 9	Squares elements of Nm
30. ⌊2.77 2.33	2 2	Rounds down
31. 3⌊Nv	3 2 ¯3 1 3 ¯8	Selects the lesser of 3 and elements of Nv
32. Ra⌊Rb	0 0 1 0 1 1	Selects the lesser of Ra and Rb, in pairs
33. ⌊/Nv	¯8	Selects the smallest element of Nv
34. ⌈2.33 2.67	3 3	Rounds up
35. 3⌈Nv	5 3 3 3 7 3	Selects the greater of 3 and elements of Nv
36. Ra⌈Rb	1 0 1 1 1 1	Selects the greater of Ra and Rb, in pairs
37. ⌈/Nv	7	Selects the largest element of Nv
38. ⌈Nv	5 2 3 1 7 8	Finds absolute values of Nv
39. 3⌈Nv	2 2 0 1 1 1	Finds residuals of dividing Nv by 3
40. ⍟2.7183 1 10	1 0 2.3	Finds natural logs of specified values
41. 10⍟2 10 100	0.3 1 2	Finds base-10 logs
42. 2⍟2 4 6.06	1 2 2.6	Finds base-2 logs
43. !6	720	Finds 6-factorial
44. 2!6	15	Combinations of 6 items taken 2 at a time
45. ?2 3⍴5	3 1 4 3 4 5	Fills a 2×3 matrix with random numbers between 1 and 5
46. ○1 2	3.1 6.3	Calculates product of pi and specified data
47. 1○1 2 3	0.84 0.91 0.14	Calculates sines

48. 2○1 2 3	0.54 ¯0.42 ¯0.99	Calculates cosines
49. Nv=2	0 1 0 0 0 0	Determines equivalence
50. Rb=Nv	0 0 0 1 0 0	Determines pair-wise equivalence
51. Nv≠2	1 0 1 1 1 1	Performs indicated logical checks
52. Rb≠Nv	1 1 1 0 1 1	Performs indicated locical checks
53. 4<Nv	1 0 0 0 1 0	Performs indicated logical checks
54. Ra<Nv	1 1 0 1 1 0	Performs indicated logical checks
55. Ra∧Rb	0 0 1 0 1 1	Performs indicated logical checks ("and")
56. Ra∨Rb	1 0 1 1 1 1	Performs indicated logical check ("or")
57. Ra⍲Rb	1 1 0 1 0 0	Performs indicated logical check ("nand")
58. Ra⍱Rb	0 1 0 0 0 0	Performs indicated logical check ("nor")
59. ~Ra	0 1 0 1 0 0	Displays "not" Ra
60. 'BME'∈Cv	1 0 1	Indicates whether BME are elements of Cv
61. Cv∈'BE'	0 1 0 0 1 0	Indicates where C or E are located within Cv
62. ⍋Nv	6 3 4 2 1 5	Ascending order of elements in Nv
63. ⍒Nv	5 1 2 4 3 6	Descending order of elements in Nv
64. Nv[⍋Nv]	¯8 ¯3 1 2 5 7	Reorders elements of Nv from low to high
65. Nv[⍒Nv]	7 5 2 1 ¯3 ¯8	Reorders elements of Nv from high to low
66. ⍳5	1 2 3 4 5	Generates integers from 1 through 5
67. Nv⍳7	5	Indicates the position of 7 within Nv
68. Cv⍳'D'	4	Indicates the position of D within Cv
69. Cv[1+⍳3]	BCD	Selects elements 2 through 4 from Cv
70. ⌽Nv	¯8 7 1 ¯3 2 5	Reverses the order of Nv
71. ¯2⌽Cv	EFABCD	Moves the last two elements of Cv to the front
72. ⍉Nm	3 ¯6 4 ¯2 7 6 8 9 ¯3	Transposes Nm
73. ⌹Nm	0.097 ¯0.054 0.096 ¯0.023 0.053 0.097 0.083 0.034 ¯0.012	Calculates the inverse of Nm
74. Ra⌹Rb	0.75	Least-squares estimate of α for Ra=αRb
75. ⍎'5 ¯3 2'	5 ¯3 2	Turns alphabetic data into numeric data

76. Z←'⍳4'⋄⍎Z	1 2 3 4	Assigns "⍳4" to Z; then executes Z
77. 5 1⍕5 2	5.0 2.0	Formats the stated numeric values
78. Cv,'ω1'	ABCDEFω1	Catenates Cv to "ω1"
79. ,Cm	ABCDEFGHIJKL	Ravels Cm
80. 2↑Cv	AB	Takes first two elements of Cv
81. ¯4↓Cv	AB	Drops last 4 elements of Cv
82. 2 2 2⊥1 0 1	5	101 in base 2 to its equivalent in base 10
83. 2 2 2⊤5	1 0 1	5 in base 10 to its equivalent in base 2
84. Ra\⍳4	1 0 2 0 3 4	Inserts 0 where 0 appears in Ra
85. Ra/Nv	5 ¯3 7 ¯8	Drops values that correspond to 0 in Ra
86. Nm+.×Nm	53 28 ¯18 ¯24 115 ¯12 ¯36 16 95	Calculates inner product of Nm and Nm
87. (⍳3)∘.=⍳3	1 0 0 0 1 0 0 0 1	Creates a 3×3 identity matrix; (⍳n)∘.=⍳n creates an n×n identity matrix
88. →(N=1)/L3	None	Goes to label L3 if N=1