Neville Marchant *nevillem@herb.sid*

USER-FRIENDLY NUMBER CRUNCHING PROGRAMMING LANGUAGES

Have you ever wanted to do some calculations or statistics, but you were unable do so because your existing software was inadequate? Have you ever wanted to do some simple modelling or simulations? Have you ever thought of applying computer-intensive statistics in your work?

In the early big-iron-dinosaur days (read mainframes), the first new zillion-dollar computer on campus caused much excitement. It was cool to sign up for computer course and to be seen at the computing centre. Almost everybody had a flirt with Fortran, but only the desperate persisted, as Fortran and the mainframe environment soon sorted out the dedicated from the boys & girls.

Fortunately the need to hang around computing centres and to learn Fortran or Basic was never an issue for me. Back in Canada at the Uni of Alberta. a new IBM (of course) 360 was purchased and each department could have a remote IBM Selectric terminal with a fantastic golf-ball printing head — real cutting edge stuff.
There was a new mysterious language so strange and weird, it looked like it was a greek dialect. It was called APL — A Programming Language — great for crunching numbers.

APL was created by Ken Iverson, a mathematician who originally devised the language as a new system of math notation. IBM decided to implement his system as a computer language, and U of Alberta was one the first to support it on their shiny new blue mainframe.

The language required one to learn a new symbol set which was daunting and this requirement continues to scare people away from APL. But when I cleared this hurdle, the simplicity of the language was captivating. Every thing was an array or matrix; control structures for looping were seldom needed; execution was from right to left and data typing was minimal being either literal or numeric.

The mathematical similarities coupled with vector-matrix operations makes APL programs extraordinarily succinct. Programs in Basic that require 40 lines of code, can sometimes be done in 2 or 3 lines. Learning APL can also be damaging because once you use it, you are forever spoiled. To this day, I shudder when I look at a bloated Basic program; the mind rebels at the thought of learning another language for it seems such a backward step.

An endearing feature of APL comes from the fact that it is an interpreted language. This allows one to use APL as a powerful calculator. Thus, it is easy to translate

complex mathematical formulae into APL and get an immediate result.

APL is therefore great for statistics. Milliken & Johnson, authors of the two volume *Analysis of Messy Data* provide computational-intensive methods for dealing with incomplete data sets and the lack of replication. Standard stats packages do not fully support many of their procedures and the authors point to APL as a possible remedy.

APL has been ported to microcomputers for sometime now, and I have used two versions. They are powerful and complete implementations of APL, but my major complaint is the upgrade policies. The move to the 32 bit window environment is desirable, but the upgrade treadmill is prohibitively expensive, presumably because of the small niche market occupied by APL coupled with the philosophy that greed is good.

Fortunately, an inexpensive alternative to APL has emerged; Iverson and associates have produced a new dialect of APL which they call "J." Why J? "Because its easy to type" says one of the authors. Fair enough! J retains the flavour of APL with extensions that improve the language. The Greek symbols have gone having been replaced by standard ASCII characters thus removing a major learning obstacle.

J, Ver. 2.6 for Windows, can be purchased from an Australian Distributor for a mere $85.00 plus the cost of manuals for about the same price. The manuals are not crash hot as the Iverson's style can be rather dense, but adequate nonetheless. A bonus stats pack/tutorial can be down-loaded on the net for free. Its about a Mbyte Postcript file which needs a special viewer.

Version 3.0 is under development; it is 32 bit and should be a enhanced screamer. If you enjoy an rewarding intellectual challenge then try J. It is a ultra-powerful, convenient, easily programmable language/calculator that produces hard copy, provides unlimited storage and talks to data sets held in spreadsheets and databases.(Sorry HP; put your toys away).

To sum up: there is no mucking about with J; the language gets down business right away. It even hooks into Visual Basic (ugh!). This allows one to put a pretty face on J. Needless to say, I haven't bothered to look in to this feature.

jackk@wood.calm.wa.gov.au

(The above s/ware & tools can be downloaded from J homepage: *http://www.jsoftware.com/head.html* -Ed).