APL2 as a specification language for statistics

by N. D. Thomson

APL has had a dedicated following for many years among some sections of the academic and industrial statistical communities. One of its greatest strengths is its value as a specification language. Not only can algorithms be described consistently and unambiguously, but also, given an appropriate interpreter, the specifications can be immediately executed. A group of academic and industrial statisticians in the United Kingdom recognized these capabilities and embarked on a project called ASL (APL Statistics Library) with the support of the British APL Association. ASL aims to provide a collection of coherent APL functions for widely used statistical calculations, thereby creating standards for the unambiguous expression of statistical algorithms. A natural consequence of this is that discussions of more complex algorithms and methods can occur without the need to revisit and redefine basic functions and the ways in which they interpret data.

Many statistical algorithms already exist in APL. The APL Statistics Library (ASL) is unique, however, in the way in which it uses APL2 as a specification language for statistical functions, which themselves define a statistical sublanguage with a high degree of consistency and extendability in its naming conventions. This allows users of ASL-based software to predict with greater accuracy the purpose and usage of a function from its name and argument names.

In software engineering, specification languages exist to allow programmers to evaluate programs and their correctness at all levels of detail. APL provides this facility for statistical algorithms but with the important additional property of *executability*. This means that ASL code used for the purpose of specification can be submitted to an APL2 interpreter and executed.

Further, by having algorithms defined at APL source level, potential users are given much greater control over their analyses than they would have using conventional packages. Alternative functions are available to perform operations such as matrix inversion, numerical integration, random number generation, and approximations for functions associated with distributions. Users can substitute their own functions at appropriate points in an algorithmic sequence.

This paper gives examples that (1) describe the philosophy of ASL code and documentation, and (2) illustrate the way in which it provides a medium for discussion of algorithms among statisticians.

ASL structure

ASL is structured into "volumes." The foundation volume is called the Basic Statistics Volume and has two key roles: first, that of specifying a core of algorithms that statistical practice and experience require as basic; and second, that of standardizing the statistical sublanguage, thus giving users of ASL a great general advantage in communicating with each other. Later volumes that cover more specialized areas such as regression, time series, and mul-

Table 1 Coding system for writing functions and naming arguments

Prefix	Algebraic Type	Rank	Suffixes
	b		
ne	n	s	
f	z	v	v
s	(r)	m	m...
	c	a	a
	g		c

Explanation of Codes	
ne	nonempty
f	frequency
s	shape
b	Boolean
n	non-negative integer
z	integer
r	real (default)
c	character
g	general, i.e., character or numeric
s	scalar
v	vector
m	matrix
a	array
c	continuous

tivariable analysis can utilize and build on the existing core of algorithms.

The Basic Statistics Volume is divided into a number of sections covering univariable statistics, distribution functions, elementary multivariable statistics, estimation and significance testing, nonparametric statistics, basic time series, analysis of variance, and combinations.

ASL function naming conventions

The function names used in the statistical sublanguage are inflectional, and the general pattern of a function name is

```
<root> <inflection..>
```

where the dots denote the possibility of multiple occurrences.

Sometimes alternative algorithms are given, for example, for generation of random normal variables, which differ in characteristics such as elegance, speed, and space requirements. These are given serial numbers as a further inflection.

Sometimes a function calls a succession of auxiliary functions. This may occur in a set of tightly bound recursive functions such as arise in combinatoric algorithms. The auxiliary functions are named by applying successive Δs as prefixes to the root or root + inflection, so that the full specification of a function name is

```
[Δ..]<root><inflection..><serial number>
```

The largest root-based groups are statistics and distribution functions. Examples of function names are the following:

MEAN	
MEANFM	mean of a frequency matrix
PCTILEFMC	percentile of a frequency matrix—continuous case
NORMQUANT	quantile of normal distribution
FTAIL	probability in right tail of F distribution
PERM	list of permutations of given order
Δ*PERM*	auxiliary to *PERM*
ΔΔ*PERM*	auxiliary Δ*PERM*

The following conventions are used in writing functions and naming arguments:

1. Function results are denoted by Z.
2. A left or right argument is specified either by a descriptive name such as *IXSET* standing for "index set," or by a composite "word" made up of not more than four parts, in lowercase letters, which describe the argument type using the coding system illustrated in Table 1. The type fm (frequency matrix) describes the special case of a numeric matrix with two columns, the first of which is to be interpreted as denoting class values in ascending order and the second as a vector of integers giving the number of items belonging to each class.
3. A reasonable degree of abbreviation in naming functions and arguments is employed to avoid excessively long names. Function names are never less than four characters in length.

Examples of functions from the Basic Statistics Volume

The meaning of a statistical function such as "mean" is data-dependent. If it is applied to a vec-

tor, i.e., a sequence of numbers, then the underlying calculation will be different from that performed if it were applied to a matrix and the result defined to mean a sequence of column means.

It has always been a fundamental part of the philosophy of APL to generalize function semantics with regard to data. In this spirit, it is possible to have the same APL expression realize both of the above interpretations of "mean," but the expression is also meaningful if applied to an array of three or more dimensions.

There is, however, a problem associated with this ability to generalize, namely that a data matrix is capable of several different interpretations. For example, each column of the matrix:

```
      DATA
0 1
1 4
2 3
3 2
```

may be regarded as a vector of values of a variable. On the other hand *DATA* can be interpreted as a frequency matrix as described in the previous section—that is, one item with value 0, four items with value 1, and so on. Another interpretation might be two items lying between 0 and 1, four items between 1 and 2, etc., with the items in each class spread uniformly throughout the class width. For example the four items in the second class would be spread to values at 1.125, 1.375, 1.625, and 1.875.

The Basic Statistics Volume provides pairs of functions that deal with the multiple-vector and frequency matrix interpretations. For the mean they are called *MEAN* and *MEANFM* respectively, so that the following results hold:

```
      MEAN DATA
1.5 2.5
      MEANFM DATA
1.6
```

The definition of the root (i.e., noninflected) function for *MEAN* applied to a nonempty array (nea) is:

```
[0]   Z←MEAN nea
[1]   Z←(+⌿nea)÷1⌷⍴nea
```

The symbol ⍴ means the "shape" of the array, e.g., `4 2` in the case of *DATA*. The symbol ⌷ means "index" so `1⌷⍴DATA` is the first item in `4 2`, namely `4`.

The symbol ⌿ means take sums along the first axis. That is, if *nea* is a vector, *MEAN* returns the mean of a sequence of numbers; if it is a matrix, it returns a vector of column means. If it is a three-dimensional array, then *MEAN* returns a matrix of the means of items occupying the same positions in the different planes, which is useful in dealing with replications of cross-tabulated data.

The function *MEAN* can readily be generalized to calculate other moments about the origin:

```
[0]   Z←n MOMENT nea
[1]   Z←(+⌿nea*n)÷1⌷⍴nea
```

The only difference is the addition of "$*n$" where $*$ denotes exponentiation.

```
      2 MOMENT DATA
3.5 7.5
```

The moment about the mean can now be specified as:

```
[0]   Z←n MOMENTM nea
[1]   nea←nea-(⍴nea)⍴MEAN nea
[2]   Z←n MOMENT nea
```

Using basic functions to discuss more advanced ones

The example chosen is that of the jack-knife.[1] Suppose that the reader were required to explain this concept to someone who had not met it before but had reasonable acquaintance with basic statistics. The first step might be to describe the process by which items are withdrawn one at a time from a vector v to form ρv samples each of size $(\rho v)-1$. Four additional symbols are needed, all but the first of which belong to APL2 but not to first-generation APL.

- ⍳ **Index of**—which is the vector of positive integers from `1` to N, e.g., `⍳3` is `1 2 3`
- ~ **Without**—in the sense that $A \sim B$ means the object A excluding those items which occur in the object B
- ⊂ **Enclose**—which means regard the array to its right as a single object (scalar)
- ¨ **Each**—an operator which directs that the function to its immediate left must be applied to each of the items in the argument on its right

Write n for ρv. The sample construction process for the jack-knife is described by first enclosing ιn so that it can be considered as a single unit, then the individual items of ιn are each excluded in turn. This is described by the following:

```
(⊂ιn)~¨ιn
```

and the process can be made into a function, as follows:

```
[0]   Z←JINDEX n
[1]   Z←(⊂ιn)~¨ιn

      JINDEX 3
 2 3  1 3  1 2
```

The result of *JINDEX* is a set of n vectors each of length $n-1$, which are the index sets that must be applied to v to select the samples required for the jack-knife. The process of selection is described by bracket indexing:

```
[0]   Z←IXSET SELECT v
[1]   Z←v[IXSET]

      2 3 SELECT 27 9 52
 9 52
```

This selection process must be applied for each of the index sets, which leads to the following function:

```
[0]   Z←JSAMPLES v
[1]   Z←(JINDEX ρv)SELECT¨⊂v

      JSAMPLES 27 9 52
 9 52  27 52  27 9
```

The **enclose** which is applied to v is necessary because each of the index sets given by *JINDEX* ρv is applied to the single object v which has to be (notionally) replicated once for each index set.

As an aside, the choice of v rather than *nev* for the argument indicates that the algorithm remains sound (although of trivial interest) in the case where v is an empty vector.

The above development may seem a little tedious because each APL2 symbol and function has been described at some length. To illustrate how rapidly this small learning investment pays off, consider how easy it is now to specify another quantity which arises early in jack-knife theory, namely the jack-knife root mean square:

```
[0]   Z←JACKRMSE nev
[1]   Z←((2 MOMENTM
           MEAN¨¨JSAMPLES nev)
             ×¯1+ρnev)*0.5
```

Conclusion

This paper has endeavored to argue the case for using APL2 within the professional statistics community as a language for standardizing and specifying algorithms. The paper illustrates how a set of such standard APL2 functions can be used as a basis for reasoning about and extending base algorithms. A group of statisticians in the United Kingdom is actively engaged in continuing to develop this work.

Cited reference

1. B. Efron, *Jack-knife, the Bootstrap and Other Resampling Plans*, Society of Industrial and Applied Mathematics, U.S. (1982).

Accepted for publication July 25, 1991.

Norman D. Thomson *IBM United Kingdom Laboratories, Hursley House, Hursley Park, Winchester, Hampshire S021 2JN, England.* Mr. Thomson joined IBM in 1969. He has earned an M.A. in mathematics at Cambridge and a B.Phil. in statistics and computing at St. Andrews. His interests have been in education and in the application of statistics and simulation in manufacturing. Dr. Thomson has been involved in many projects in collaboration with the IBM United Kingdom plants. He has been an ardent enthusiast for APL and is the author of *APL Programs for the Mathematics Classroom*, Springer-Verlag, 1988.

Reprint Order No. G321-5452.