Vector thinking in APL, R and Mathematica

A. I. McLeod
University of Western Ontario

Abstract

Keywords: APL, Mathematica, R.

APL was first developed by Kenneth E. Iverson (http://en.wikipedia.org/wiki/Kenneth_E._Iverson), in a Ph.D. Thesis at Harvard and in subsequent graduate courses he taught there, as an executable mathematical notation (Iverson 1980) and this development was eventually recognized with a Turing Award (http://en.wikipedia.org/wiki/Turing_Award). The material from Iverson’s thesis and lectures was later published in book form (Iverson 1962) and then after Iverson joined IBM it was implemented on IBM/360 computers and this implementation is the subject of the first widely available manual on APL (Pakin 1968). A survey of APL and its extensions to APL2 and J is given in (http://en.wikipedia.org/wiki/APL_programming_language).

There are many similarities between APL and the S family of languages. At the most fundamental conceptual level these languages emphasize the elegance of vector thinking and the vectorization of computations (Chambers 2008, Ch. 6.4). For example, one method of generating 1000 NID(0, 1) random variables would be to generate a matrix of dimension 12×1000 of uniform (0,1) random numbers and then sum the columns and subtract 6. In R we use apply but in proglangAPL the reduction operator / plays the same role. Similarly in Mathematica, in this situation, we would use Map or /@. Specifically in R,

\[ R> X <- \text{apply}(\text{matrix}(\text{runif}(12\times1000), \text{nrow}=1000), \text{MARGIN}=1, \text{sum}) - 6 \]

whereas in APL,

\[ X \leftarrow -6 ++/ (10\times -6) \times 1000 12 \rho \times 10000 \rho 10 \ast 6 \]

and in Mathematica,

\[ X = \text{Total} /@ \text{Partition}[\text{Array}[\text{Random[]} &, 1000\times12], 12] - 6; \]

Vectorization of computations combined with an interactive programming environment, pioneered in APL, has proved to be a very effective way of getting results fast. In fact, Wolfram (2003, p. 1408) lists nine similar vector-thinking type functions in APL and their equivalents in Mathematica. Roughly speaking, for numerical computation, all nine of these vectorization functions have their builtin approximate equivalents in S as well as shown in Table 2.
Some basic vectorization functions in APL, Mathematica and S.

<table>
<thead>
<tr>
<th>Function (APL)</th>
<th>Function (Mathematica)</th>
<th>Function (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>catenate</td>
<td>Join</td>
<td>c, cbind, rbind</td>
</tr>
<tr>
<td>compress</td>
<td>Select</td>
<td>Index vectors*</td>
</tr>
<tr>
<td>grade</td>
<td>Ordering</td>
<td>order</td>
</tr>
<tr>
<td>grade</td>
<td>Sort</td>
<td>sort</td>
</tr>
<tr>
<td>iota operator</td>
<td>Range</td>
<td>1:n</td>
</tr>
<tr>
<td>ravel</td>
<td>Flatten</td>
<td>as.vector</td>
</tr>
<tr>
<td>reshape</td>
<td>Partition</td>
<td>dim</td>
</tr>
<tr>
<td>shape</td>
<td>Dimensions</td>
<td>dim</td>
</tr>
</tbody>
</table>

* Index vectors, see ?, Sections 2.7

The R workspace management package rwm contains functions loadws(...) savews(...) and clearws(...) that are loosely inspired by the APL commands )LOAD, )SAVE and )CLEAR (Polvika and Pakin 1975, Ch. 8), (Anscombe 1981, p. 103), (Grenander 1982, pp. 151–152), (Stiers, Goovaerts, and de Kerf 1983, Ch. 9).

Today there are many exciting developments in the use of APL for software re-engineering (Askoolum 2006). A new .Net application of APL has been implemented that is useful for internet applications utilizing Microsoft Office products - see,


An overview of APL with links to many other current implementations is available from Wikipedia,

http://en.wikipedia.org/wiki/APL_%28programming_language%29

References


**Affiliation:**

A.I. McLeod

University of Western Ontario

E-mail: aimcleod@uwo.ca