

SLINK: An optimally efficient algorithm for the single-link cluster method

R. Sibson

King's College Research Centre, King's College, Cambridge, and Cambridge University
Statistical Laboratory

The SLINK algorithm carries out single-link (nearest-neighbour) cluster analysis on an arbitrary dissimilarity coefficient and provides a representation of the resultant dendrogram which can readily be converted into the usual tree-diagram. The algorithm achieves the theoretical order-of-magnitude bounds for both compactness of storage and speed of operation, and makes the application of the single-link method feasible for a number of OTU's well into the range 10^3 to 10^4 . The algorithm is easily programmable in a variety of languages including FORTRAN.

(Received January 1972)

1. Background

The *single-link*, or *nearest-neighbour*, cluster method is one of the oldest methods of cluster analysis; it was suggested by workers in Poland in 1951 (Florek *et al.*, 1951a, b) and independently by McQuitty (1957) and Sneath (1957). Its obvious disadvantage—the 'chaining' effect—has long been well known, and has prompted the invention of many other cluster methods of either a hierarchic or a non-hierarchic (overlapping) kind; see Lance and Williams (1967) and Jardine and Sibson (1971). These methods also have their disadvantages. The alternative hierarchic methods have been criticised by Jardine and Sibson for lack of continuity, which they regard as being a far more severe defect than the chaining effect in many applications; it is also difficult to see how most of these methods could be programmed for more than a few hundred OTU's. Many problems with a large set of OTU's turn out on inspection to be distribution-mixture problems, rather than cluster-analysis problems in the strict sense in which the OTU's do not constitute a random sample from some larger population. Nevertheless there are many problems for which a large-scale cluster method is needed: this paper shows that the single-link method can be programmed efficiently enough to meet this need, and since its defects are well-enough understood and of such a nature as to cause it to be misleading only rather rarely, the method itself should generally be acceptable; in fact Jardine and Sibson have proposed an axiomatic framework for cluster methods within which it is uniquely acceptable, and in that context its defects must be viewed as those of hierarchic classification itself. They suggest overlapping methods to supplement single-link although these are applicable only up to about 100 OTU's. Fisher and van Ness (1971) have explored just which conditions are satisfied by the various hierarchic methods, and although they do not rule out other methods they point out that single-link (nearest-neighbour in their terminology) has many advantages. The present paper provides an algorithm for carrying out the single-link method which achieves the theoretical order of magnitude bounds on speed and compactness, and the author believes this algorithm to be superior in these respects to other general-purpose single-link algorithms known to him which have appeared in the literature (see Gower and Ross, 1969; Lance and Williams, 1967; Wishart, 1969; van Rijsbergen, 1970); it enables single-link cluster analysis to be applied on an unprecedented scale, and also renders its application to smaller numbers of OTU's a trivial matter in terms of computer usage.

2. The single-link method

Following Jardine and Sibson (1971), we define a *dissimilarity coefficient* (DC) to be a symmetric non-negative function $d: P \times P \rightarrow \mathcal{R}$ where P is the set of OTU's, and where

$d(a, a) = 0$ for all $a \in P$. We also define a *dendrogram* to be a function $c: [0, \infty) \rightarrow E(P)$, where $E(P)$ is the set of equivalence relations on P , and c satisfies the conditions

$$\begin{aligned} h \leq h' \text{ implies } c(h) &\subseteq c(h') \\ c(h) &\text{ is eventually } P \times P \\ c(h + \delta) &= c(h) \text{ for all small enough } \delta > 0 \end{aligned}$$

Thus a dendrogram is a nested sequence of partitions with associated numerical levels, the partition at a high enough level being the whole set P . A dendrogram is usually represented as the familiar tree-diagram, but there is a great deal of freedom—freedom which can be misused—over the order in which the OTU's are disposed along the baseline; this order forms no part of the dendrogram as such. The single-link method of cluster analysis is defined very simply as follows. Let d be the dissimilarity coefficient. At a fixed level h consider the graph whose vertices are OTU's and whose edges link just those pairs of OTU's of dissimilarity at most h . Then $c(h)$ is the equivalence relation corresponding to the partition of P defined by the connected components of this graph. It is very easy to check that the $c(h)$ defined in this way for different values of h do in fact give a function c which satisfies the conditions for a dendrogram. The transformation $d \rightarrow c$ so defined is the single-link cluster method. Some authors have regarded the partition $c(h)$ at one level or at some small number of levels as constituting the result of applying the method; we shall take the more usual and simple point of view that it is the whole dendrogram which is the result of the method.

3. Order-of-magnitude limitations

A dendrogram on N OTU's can have up to $N - 1$ distinct *splitting levels*—levels at which $c(h)$ changes—and so at the very least storage of $O(N)$ is required for a dendrogram. There are in fact numerous ways of achieving this order-of-magnitude bound. A DC on N objects can take up to $\frac{1}{2}N(N - 1)$ distinct values, and most cluster methods, in particular the single-link method, can be affected by changes in any one of these, so a cluster method operating on a DC will have a time-dependence at least $O(N^2)$ because each DC value must be examined at least once. The DC is the starting-point for cluster analysis, but almost always it is obtained from data held separately for each OTU. If the DC is to be held in core storage for random access, $O(N^2)$ locations will be needed, whereas both the original data and the dendrogram only require $O(N)$ locations, although if there is much data the constant may be large. Thus we want to avoid holding the DC in core if possible, and this is a failing of most clustering algorithms, which require repeated random access to the DC, for example to sort the values into numerical order. The SLINK algorithm avoids this problem by using the DC values a part-row at a time—at stage n random

access is needed only to values of the form $d(i, n)$ for $i < n$ —and no sorting or rearrangement procedures are employed. The storage needed for a part-row is again $O(N)$, and so provided the DC values can be either generated on demand in the order 2-1; 3-1, 3-2; 4-1, 4-2, 4-3; 5-1, . . . or read in this order from an input stream or device, having been generated and written in this order to, for example, disc store, then the core store requirement is only $O(N)$ for the cluster method.


4. The pointer representation

Although the characterisation as a function $c: [0, \infty) \rightarrow E(P)$ certainly captures what is meant by a dendrogram, it is clearly not how the information would actually be kept. There are many ways of specifying a dendrogram on N objects in about $2N$ function values; we shall achieve it by means of two functions each defined on the set $1, \dots, N$. The pair of functions will be called a *pointer representation*. $\pi: 1, \dots, N \rightarrow 1, \dots, N$ and $\lambda: 1, \dots, N \rightarrow [0, \infty]$ constitute a pointer representation if the following conditions hold

$$\begin{aligned} \pi(N) &= N & \lambda(N) &= \infty \\ \pi(i) > i & \quad \lambda(\pi(i)) > \lambda(i) & \text{for } i < N \end{aligned}$$

We shall show that there is a natural 1-1 correspondence between pointer representations and dendrograms. Suppose first that c is a dendrogram. Define π, λ for $i < N$ by

$$\begin{aligned} \lambda(i) &= \inf \{h: \exists j > i \text{ with } (i, j) \in c(h)\} \\ \pi(i) &= \max \{j: (i, j) \in c(\lambda(i))\} \end{aligned}$$

 $\lambda(i)$ is the lowest level at which i is no longer the last object in its cluster, and $\pi(i)$ is the last object in the cluster which it then joins; we are, of course, regarding the OTU's in P as being labelled by the integers $1, \dots, N$. It is easy to see that π, λ so defined is a pointer representation. Now suppose that we are given a pointer representation π, λ . We define a function σ by taking $\sigma(i, h)$ to be the first element k in the sequence

$$i, \pi(i), \pi^2(i), \pi^3(i), \dots, N$$

for which $\lambda(k) > h$. Then define


$$c(h) = \{(i, j) : \sigma(i, h) = \sigma(j, h)\}$$

It is easy to check that c defined in this way is a dendrogram. We now prove that these two transformations are mutually inverse.

Lemma

The transformations $c \rightarrow \pi, \lambda$ and $\pi, \lambda \rightarrow c$ defined above are mutually inverse, and so constitute a 1-1 correspondence between dendrograms and pointer representations.

Proof

 We prove that $c \rightarrow \pi, \lambda \rightarrow c'$ in fact leads back to c , and that $\pi, \lambda \rightarrow c \rightarrow \pi', \lambda'$ leads back to π, λ . Consider first $c \rightarrow \pi, \lambda \rightarrow c'$. By definition $c'(h) = \{(i, j) : \sigma(i, h) = \sigma(j, h)\}$. Now $(i, \sigma(i, h)) \in c(h)$ and $(j, \sigma(j, h)) \in c(h)$, so if $\sigma(i, h) = \sigma(j, h)$ we have $(i, j) \in c(h)$, that is, $c'(h) \subseteq c(h)$. Conversely, if $(i, j) \in c(h)$ then $(\sigma(i, h), \sigma(j, h)) \in c(h)$. Suppose that these are not equal; without loss of generality $\sigma(i, h) < \sigma(j, h)$. Then $\lambda(\sigma(i, h)) \leq h$, a contradiction. We deduce that $c(h) \subseteq c'(h)$ and hence that $c(h) = c'(h)$, that is, $c = c'$. Now consider $\pi, \lambda \rightarrow c \rightarrow \pi', \lambda'$. λ' is defined by

$$\begin{aligned} \lambda'(i) &= \inf \{h: \exists j > i \text{ with } (i, j) \in c(h)\} \\ &= \inf \{h: \exists j > i \text{ with } \sigma(i, h) = \sigma(j, h)\} \end{aligned}$$

But $\sigma(i, h)$ is such a j if one exists, so

$$\begin{aligned} \lambda'(i) &= \inf \{h: \sigma(i, h) > i\} \\ &= \lambda(i) \end{aligned}$$

Now

$$\begin{aligned} \pi'(i) &= \max \{j: (i, j) \in c(\lambda'(i))\} \\ &= \max \{j: (i, j) \in c(\lambda(i))\} \end{aligned}$$

$$\begin{aligned} &= \max \{j: \sigma(i, \lambda(i)) = \sigma(j, \lambda(i))\} \\ &= \max \{j: \pi(i) = \sigma(j, \lambda(i))\} \\ &= \pi(i) \end{aligned}$$

So $\pi', \lambda' = \pi, \lambda$ and the proof is complete.

5. Recursive updating of the pointer representation

Our reason for considering the pointer representation of a dendrogram rather than any other comparably compact representation is that the pointer representation can be updated on the inclusion of a new OTU in a highly efficient way. We shall use the phrase 'the dendrogram on the first n OTU's' to mean the single-link dendrogram obtained from the restriction of the DC to the first n OTU's; this will in general be different from the restriction to the first n OTU's of the single-link dendrogram on all N OTU's, and the latter is a construct which we shall not use. Quantities relating to the dendrogram on the first n OTU's will be given subscript n , so the dendrogram is c_n and its pointer representation is π_n, λ_n .

For given n we define $\mu_n(i)$ recursively on i :

$$\mu_n(i) = \min \{d(i, n+1), \min_{\pi_n(j)=i} \max \{\mu_n(j), \lambda_n(i)\}\}$$

Thus $\mu_n(i)$ is defined for $i = 1, \dots, n$ and since

$$\mu_n(i) \leq d(i, n+1)$$

and d is a (finite) DC, $\mu_n(i)$ is finite for all i . We then define π, λ , which we shall prove to be the pointer representation of c_{n+1} , that is, π_{n+1}, λ_{n+1} , as follows.

$$\pi(n+1) = n+1 \quad \lambda(n+1) = \infty$$

$$\lambda(i) = \min \{\mu_n(i), \lambda_n(i)\} \text{ for } i < n+1$$

$$\pi(i) = \pi_n(i), \text{ except that if } \mu_n(i) \leq \lambda_n(i) \text{ or}$$

$$\mu_n(\pi_n(i)) \leq \lambda_n(i) \text{ then } \pi(i) = n+1, \text{ again for } i < n+1.$$

Lemma

$$\pi, \lambda = \pi_{n+1}, \lambda_{n+1}$$

Proof

We show first that π, λ is a pointer representation. Certainly $\pi(n+1) = n+1, \lambda(n+1) = \infty$, so consider $i < n+1$. $\pi(i) = \pi_n(i) > i$ or $= n+1 > i$ if $i < n$, and if $i = n$ then $\mu_n(n) < \infty = \lambda_n(n)$ so $\pi(n) = n+1 > n$. Thus in all cases $\pi(n) > i$ if $i < n+1$. If $\pi(i) = n+1$, and $i < n$, then $\lambda(i) < \infty = \lambda(n+1)$, and if $i = n$ then $\lambda(n) = \mu_n(n) < \infty = \lambda(n+1)$. If $\pi(i) = \pi_n(i)$ then $\mu_n(i) > \lambda_n(i)$ and $\mu_n(\pi_n(i)) > \lambda_n(i)$ so $\lambda(\pi(i)) = \lambda(\pi_n(i)) = \min \{\mu_n(\pi_n(i)), \lambda_n(\pi_n(i))\} > \lambda_n(i) = \lambda(i)$. In all cases we have $i < n+1$ implies $\lambda(i) < \lambda(\pi(i))$. Having established that π, λ is a pointer representation, we must now show that it in fact represents the right dendrogram.

Consider some fixed level h . The clusters for c_{n+1} at level h are related to those for c_n as follows: add a one-OTU cluster consisting just of $n+1$; unite with this each cluster containing an OTU i such that $d(i, n+1) \leq h$. Define $\kappa_n(i, h) = \{j: (i, j) \in c_n(h)\}$. Then we can express this process in terms of σ by saying that $\sigma_{n+1}(i, h) = \sigma_n(i, h)$ unless there exists $j \in \kappa_n(i, h)$ with $d(j, n+1) \leq h$, in which case $\sigma_{n+1}(i, h) = n+1$. To establish that $\pi, \lambda = \pi_{n+1}, \lambda_{n+1}$ it will be enough to show that σ defined in terms of π, λ has the property required of σ_{n+1} , since clearly $\pi, \lambda \rightarrow \sigma$ is 1-1. It is easy to see that if $\sigma(i, h) \neq \sigma_n(i, h)$ then $\sigma(i, h) = n+1$, so it is enough to check that $\sigma(i, h) = n+1$ if and only if there exists $j \in \kappa_n(i, h)$ with $d(j, n+1) \leq h$. Now

$$\mu_n(\sigma_n(i, h)) \leq h$$

if and only if

$$\text{either } d(\sigma_n(i, h), n+1) \leq h$$

$$\text{or for some } j \text{ such that } \pi_n(j) = \sigma_n(i, h)$$

$$\text{we have } \mu_n(j) \leq h \text{ and } \lambda_n(j) \leq h$$

i.e. if and only if

either $d(\sigma_n(i, h), n + 1) \leq h$
or for some $j \in \kappa_n(i, h)$ such that
 $\pi_n(j) = \sigma_n(i, h)$ we have $\mu_n(j) \leq h$

and so by an inductive argument we have

$\mu_n(\sigma_n(i, h)) \leq h$ if and only if there exists
 $j \in \kappa_n(i, h)$ such that $d(j, n + 1) \leq h$

Now $\sigma(i, h) = n + 1$ if and only if

either $\pi(j) = n + 1$ for some $j = i, \pi_n(i), \dots < \sigma_n(i, h)$
or $\mu_n(\sigma_n(i, h)) \leq h$

But if the first of these alternatives holds, we must have

$\lambda_n(j) \geq \mu_n(\pi_n(j))$ for some $j = i, \pi_n(i), \dots < \sigma_n(i, h)$
or $\lambda_n(j) \geq \mu_n(j)$ for some $j = i, \pi_n(i), \dots < \sigma_n(i, h)$

and since for such a j $\lambda_n(j) \leq h$, this implies that for some
 $j \in \kappa_n(i, h)$ we have $\mu_n(j) \leq h$ and hence $\mu_n(\sigma_n(i, h)) \leq h$. Thus
the first alternative implies the second, and

$\sigma(i, h) = n + 1$
if and only if $\mu_n(\sigma_n(i, h)) \leq h$.
if and only if there exists $j \in \kappa_n(i, h)$
with $d(j, n + 1) \leq h$
if and only if $\sigma_{n+1}(i, h) = n + 1$

and this completes the proof.

If we start with π_1, λ_1 , which must be given by $\pi_1(1) = 1$,
 $\lambda_1(1) = \infty$, then after $N - 1$ steps of the above recursive
process, we shall obtain π_N, λ_N which is the pointer represen-
tation of the single-link dendrogram on the whole set $P = 1$,
 \dots, N .

6. The SLINK algorithm

The SLINK algorithm is simply a convenient way of carrying
out the recursive process computationally. Three arrays of
dimension N are used, and we shall denote them by Π, A, M .
Suppose that Π, A contain π_n, λ_n in their first n locations. Then
the SLINK algorithm overwrites these to place π_{n+1}, λ_{n+1} in
the first $n + 1$ locations as follows:

1. Set $\Pi(n + 1)$ to $n + 1$, $A(n + 1)$ to ∞
2. Set $M(i)$ to $d(i, n + 1)$ for $i = 1, \dots, n$
3. For i increasing from 1 to n
 - if $A(i) \geq M(i)$
 - set $M(\Pi(i))$ to $\min \{M(\Pi(i)), A(i)\}$
 - set $A(i)$ to $M(i)$
 - set $\Pi(i)$ to $n + 1$
 - if $A(i) < M(i)$
 - set $M(\Pi(i))$ to $\min \{M(\Pi(i)), M(i)\}$
4. For i increasing from 1 to n
 - if $A(i) \geq A(\Pi(i))$
 - set $\Pi(i)$ to $n + 1$

The total space needed for this process, assuming that the DC
values are available in the correct order, is clearly $O(N)$ —in
fact $3N$ plus overheads—the number of operations needed
to find π_N, λ_N is $O(N^2)$, so, as claimed, the SLINK algorithm
constructs a representation of the single-link dendrogram in a
way which is optimally efficient in order-of-magnitude terms.
It is also clear that the amount of work done for each dissimi-
larity value is very small: generate or read it and load it into M ;
check it against the value in A and adjust values accordingly;
check A entries against one another. It seems unlikely that this
scheme of operations can be substantially reduced, and so it is
unlikely that any other algorithm can improve much on the
constant multiplying N^2 in any given language/machine context.

7. Classifiability

Jardine and Sibson (1971) suggest the use of the quantity

$$\Delta_1 = \sum_{i < j} (d(i, j) - d^*(i, j)) / \sum_{i < j} d(i, j)$$

as a measure of classifiability, where $d^*(i, j)$ is the ultrametric
DC corresponding to the single-link dendrogram c and is

defined by

$$d^*(i, j) = \inf \{h : (i, j) \in c(h)\}.$$

The smaller Δ_1 is, the more amenable to single-link classi-
fication the data is. The calculation of Δ_1 can readily be
incorporated into an implementation of the SLINK algorithm,
and this is recommended.

8. Presentation of results

The user of a cluster method may reasonably expect to be
provided with output in a form which he can readily appreciate,
and this will usually take the form of numerical output from
which a tree-diagram can easily be drawn, possibly accom-
panied by the tree-diagram itself, either drawn on a plotter or
approximated on a line-printer. For most purposes the latter
is adequate. The pointer representation of a dendrogram is not
particularly helpful from the user's point of view, and it is
desirable to convert it into another representation called the
packed representation for output. The packed representation
consists of two functions τ, v defined as follows.

$$v(i) = \lambda(\tau(i))$$

$$\tau^{-1}(\pi(\tau(i))) > i \text{ if } i < n, \text{ and}$$

$$v(j) \leq v(i) \text{ if } i \leq j < \tau^{-1}(\pi(\tau(i)))$$

This in fact characterises the dendrogram uniquely, and it is not
difficult to convert the pointer representation to the packed
representation, the conversion taking time $O(N^2)$ with a very
small coefficient for N^2 . It is convenient to provide an extra
array of dimension N to facilitate the conversion, so the total
store size is $4N$ plus overheads. The packed form represen-
tation is a numerically coded form of a tree-diagram, which may
be constructed from it as follows: in positions $1, \dots, N$ along
the baseline insert OTU numbers, the number in position i
being $\tau(i)$; above this draw a vertical to height $v(i)$ above the
baseline; when all verticals have been drawn, draw a horizontal
to the right (that is, in the direction of increasing position
number) until it meets another vertical. This will give a tree-
diagram representing the dendrogram, but with all vertical
stems displaced to the extreme right of the clusters which they
represent. This form of tree-diagram can be produced extremely
easily from the packed form output on a line-printer, and this is
normally to be recommended. If a more conventional form of
tree-diagram is wanted, then either a more elaborate computer
graphics technique can be used, or the dendrogram can simply
be re-drawn by hand; this is easy because the OTU's are
presented by the packed representation in a suitable order for a
tree-diagram to be drawn on them.

Appendix

A FORTRAN SLINK PROGRAM

The program given here calculates the single-link dendrogram
from a DC read in value-by-value from an input stream. Much
of the main subroutine is special to this case, but the sub-
programs called from it are quite general and have been separ-
ated out to allow them to be used in calling programs designed,
for example, to work with an internally generated DC. The
calling program for the subroutine SLINK must declare NA,
NB as integer arrays and HA, HB as real arrays, all singly
subscripted and of the same dimension, and must set NMXOBJ
to their dimension and TOP to a large positive real value such
that $\text{TOP} - 1.0$ is larger than every DC value. It must also set
the stream numbers NRDATA, NWRECD, NPDEND as appro-
priate. Subroutine RCLOCK should be provided to set T
to the time in seconds (data type REAL) from some appro-
priate point in the calling program. Experience with this pro-
gram shows that it spends almost all its time reading DC values,
and this emphasises the desirability of using internally gener-
ated DC values, or at least of avoiding the FORTRAN I/O
package, for any substantial number of OTU's. The time taken

by the main part of the program *excluding* the reading or generation of DC values is, on Cambridge University Computer Laboratory TITAN, approximately 100 seconds for $N = 1,000$, and increases as N^2 .

```

1 C
2 C
3 C FORTRAN SLINK 003 CREATED 15/12/71
4 C
5 C
6 C Calling program sets stream numbers, value of TOP (infinity)
7 C and value of NMXOBJ (dimension of arrays).
8 C
9 C SUBROUTINE SLINK1(NA,NB,HA,HB,NMXOBJ,TOP,NRDATA,NWRECD,NPDEND)
10 C DIMENSION NA(NMXOBJ),NB(NMXOBJ),HA(NMXOBJ),HB(NMXOBJ),REF(2),
11 C 1 TITLE(6)
12 C DATA AD,AS,AW,IHD,IHS,IHW/
13 C
14 C Read reference code, number of objects (OTUs), type (S for
15 C similarities, otherwise dissimilarities) and mode (W for whole
16 C matrix, D for subdiagonal with diagonal, otherwise strictly
17 C subdiagonal).
18 C
19 C READ(NRDATA,9001) REF(1),REF(2),NOBJ,ATYPE,AMODE
20 C
21 C Check that number of objects is within range, that type is not S,
22 C and that mode is not W or D.
23 C
24 C IF(NOBJ.LT.2.OR.NOBJ.GT.NMXOBJ) STOP 1
25 C IF(ATYPE.EQ.AS) STOP 2
26 C IF(AMODE.EQ.AW.OR.AMODE.EQ.AD) STOP 3
27 C
28 C Initialise for one object,
29 C
30 C NMIS3 = 0
31 C SIZE = 0.0
32 C NA(1) = 1
33 C HA(1) = TOP
34 C
35 C For each of the remaining objects set NA(I) to I and HA(I)
36 C to TOP, and read the current part-row into HB.
37 C
38 C DO 1 I = 2,NOBJ
39 C 1 I = I-1
40 C NA(I) = I
41 C HA(I) = TOP
42 C READ(NRDATA,9002) (HB(J), J = 1,I)
43 C
44 C Check for missing DC values, signalled by negative entry, and
45 C replace them by TOP-1.0. Update NMIS3, the number of missing
46 C DC values, and SIZE, the sum of the DC values.
47 C
48 C DO 2 J = 1,I
49 C IF(HB(J)) 3,2,2
50 C 3 HB(J) = TOP-1.0
51 C NMIS3 = NMIS3+1
52 C 2 SIZE = SIZE+HB(J)
53 C
54 C SLINK1 is a subroutine which carries out the rest of the SLINK
55 C algorithm to produce the pointer representation of the complete
56 C dendrogram in NA and HA.
57 C
58 C 1 CALL SLINK1(NA,HA,HB,I,1,NMXOBJ)
59 C
60 C SLINK2 is a subroutine which converts the pointer representation
61 C into the packed representation by a chain-building method.
62 C
63 C CALL SLINK2(NA,NB,HA,HB,NOBJ,NMXOBJ)
64 C
65 C Object labels are read into HB and finally a title for the DC
66 C is read. SCALE is calculated as the largest value in NA, and
67 C the packed representation and other information is written to a
68 C printer-type stream.
69 C
70 C DO 4 I = 1,NOBJ
71 C J = NR(I)
72 C 4 READ(NRDATA,9003) HB(J)
73 C READ(NRDATA,9004) (TITLE(J), J = 1,6)
74 C WRITE(NWRECD,9005) (TITLE(J), J = 1,6),REF(1),REF(2),NOBJ
75 C SCALE = 0.0
76 C DO 5 I = 1,NOBJ
77 C IF(HA(I)-TOP+1.0) 6,7,7
78 C 6 SCALE = AMAX1(SCALE,HA(I))
79 C WRITE(NWRECD,9006) HB(I),HA(I)
80 C GOTO 5
81 C 7 WRITE(NWRECD,9007) HB(I)
82 C 5 CONTINUE
83 C
84 C If there is missing data this is reported, otherwise the value
85 C of DELTA-ONE-HAT is calculated using the function SLINK3,
86 C which returns the sum of the values of the ultrametric DC
87 C corresponding to the dendrogram.
88 C
89 C IF(NMIS3) 999,8,9
90 C 9 NPAIR = NOBJ*(NOBJ-1)/2
91 C WRITE(NWRECD,9008) NMIS3,NPAIR
92 C GOTO 10
93 C 8 DELHAT = (SIZE-SLINK3(NA,HA,NOBJ,NMXOBJ))/SIZE
94 C WRITE(NWRECD,9009) DELHAT
95 C 10 DO 11 I = 1,NOBJ
96 C IF(HA(I)-TOP+1.0) 11,12,12
97 C 12 HA(I) = -1.0
98 C 11 CONTINUE
99 C
100 C The packed representation is written to a punch-type stream
101 C and finally the runtime in seconds is calculated.
102 C
103 C WRITE(NPDEND,9010) REF(1),NOBJ,SCALE,((HB(I),HA(I)), I = 1,NOBJ)
104 C WRITE(NPDEND,9004) (TITLE(J), J = 1,6)
105 C CALL RCLJCK(I)
106 C WRITE(NWRECD,9011) T
107 C RETURN
108 C
109 C Dummy label.

```

```

110 C
111 C 999 STOP 0
112 C
113 C FORMAT statements.
114 C
115 9001 FORMAT(4X,2A4/15,2A1)
116 9002 FORMAT(F10.4)
117 9003 FORMAT(A4)
118 9004 FORMAT(3A4/3A4)
119 9005 FORMAT(1H1//1H6,34HFORTRAN SLINK 003 CREATED 15/12/71//1H0,
120 1 6HDC IS ,6A4/1H0,10HREFERENCE ,2A4/1H0,3HON ,14,8H OBJECTS//
121 2 1H9,25HDENDROGRAM IN PACKED FORM/1H0,14HOBJECT LEVEL/)
122 9006 FORMAT(1H ,1X,A4,1X,F10.4)
123 9007 FORMAT(1H ,1X,A4,6X,1H-)
124 9008 FORMAT(/1H0,1C,26H DC VALUES MISSING OUT OF ,18)
125 9009 FORMAT(/1H0,17HDELTA-ONE-HAT IS ,F6.4)
126 9010 FORMAT(4HDATA,A4,4H7777/15,1X,F10.4/(A4,1X,F10.4))
127 9011 FORMAT(/1H0,11HJCB RAN IN ,F7.2,5H SECS)
128 END
129 C
130 C
131 C
132 SUBROUTINE SLINK1(NA,HA,HB,I,1,NMXOBJ)
133 DIMENSION NA(NMXOBJ),HB(NMXOBJ),HA(NMXOBJ),HB(NMXOBJ)
134 DO 1 J = 1,I
135 NEXT = NA(J)
136 IF(HA(J)-HB(J)) 2,3,3
137 2 H = HB(J)
138 IF(HB(NEXT)-H) 1,1,4
139 3 H = H(J)
140 NA(J) = I+1
141 HA(J) = HB(J)
142 IF(HB(NEXT)-H) 1,1,4
143 HB(NEXT) = H
144 1 CONTINUE
145 DO 5 J = 1,I
146 NEXT = NA(J)
147 IF(HA(J)-HA(NEXT)) 5,6
148 6 NA(J) = I+1
149 5 CONTINUE
150 RETURN
151 END
152 C
153 C
154 C
155 SUBROUTINE SLINK2(NA,NB,HA,HB,NOBJ,NMXOBJ)
156 DIMENSION NA(NMXOBJ),NB(NMXOBJ),HA(NMXOBJ),HB(NMXOBJ)
157 NB(NOBJ) = NOBJ
158 DO 1 N = 2,NOBJ
159 H = HA(NOBJ+1-N)
160 NEXT = CBJ
161 NOW = NEXT
162 NEXT = NB(NOW)
163 IF(H-HA(NEXT)) 3,2,2
164 3 NB(NOW) = NOBJ+1-N
165 1 NB(NOBJ+1-N) = NEXT
166 NEXT = NB(NOBJ)
167 NOW = NEXT
168 N = NA(NOW)
169 IF(NB(NOW)) 5,999,6
170 6 NEXT = NB(NOW)
171 IF(NB(N)) 7,999,8
172 8 NB(NOW) = NB(N)
173 NB(N) = -NOW
174 IF(NEXT-NOBJ) 4,11,999
175 7 NOSE = -NB(N)
176 NB(NOW) = NB(NOSE)
177 NB(N) = -NOW
178 NA(NOW) = NOSE
179 IF(NEXT-NOBJ) 4,11,999
180 5 NOSE = -NB(NOW)
181 NEXT = NB(NOSE)
182 IF(NB(N)) 9,999,10
183 10 NB(NOSE) = NB(N)
184 NB(N) = -NOSE
185 IF(NEXT-NOBJ) 4,11,999
186 9 NA(NOW) = -NB(N)
187 NB(N) = -NOSE
188 N = NA(NOW)
189 NB(NOSE) = NB(N)
190 IF(NEXT-NOBJ) 4,11,999
191 11 NEXT = -NB(NOBJ)
192 DO 12 N = 1,NOBJ
193 HB(N) = HA(N)
194 NB(N) = NEXT
195 12 NEXT = NA(NEXT)
196 DO 13 N = 1,NOBJ
197 NOW = NB(N)
198 NA(N) = NOW
199 HA(N) = HB(NOW)
200 13 DO 14 N = 1,NOBJ
201 NOW = NA(N)
202 14 NB(NOW) = N
203 RETURN
204 999 STOP 0
205 END
206 C
207 C
208 C
209 FUNCTION SLINK3(NA,HA,NOBJ,NMXOBJ)
210 DIMENSION NA(NMXOBJ),HA(NMXOBJ)
211 NOBJ1 = NOBJ-1
212 SLINK3 = 0.0
213 DO 1 I = 1,NOBJ1
214 DO 2 J = 1,I
215 N1 = I+1-J
216 IF(HA(N1)-HA(I)) 2,2,3
217 2 CONTINUE
218 N1 = 0
219 3 I1 = I+1
220 DO 4 J = I1,NOBJ
221 IF(HA(J)-HA(I1)) 4,1,1
222 4 CONTINUE
223 1 SLINK3 = SLINK3+FLOAT((I1-N1)*(J-I1))*HA(I)
224 RETURN
225 END

```

References

- FISHER, L., and VAN NESS, J. W. (1971). Admissible clustering procedures, *Biometrika*, Vol. 58, pp. 91-104.
- FLOREK, K., ŁUKASZEWICZ, J., PERKAL, J., STEINHAUS, H., and ZUBRZYCKI, S. (1951a). Sur la liaison et la division des points d'un ensemble fini, *Colloquium Math.*, Vol. 2, pp. 282-285.
- FLOREK, K., ŁUKASZEWICZ, J., PERKAL, J., STEINHAUS, H., and ZUBRZYCKI, S. (1951b). Taksonomia Wroclawska, *Przegl. antrop.*, Vol. 17, pp. 93-207 (in Polish with English summary).
- GOWER, J. C., and ROSS, G. J. S. (1969). Minimum spanning trees and single-linkage cluster analysis, *Appl. Statist.*, Vol. 18, pp. 54-64.

- JARDINE, N., and SIBSON, R. (1971). *Mathematical Taxonomy*, J. Wiley and Sons Ltd., London and New York.
- LANCE, G. N., and WILLIAMS, W. T. (1967). A general theory of classificatory sorting strategies, I. Hierarchical Systems, *The Computer Journal*, Vol. 9, pp. 373-380.
- MCQUITT, L. L. (1957). Elementary linkage analysis for isolating orthogonal and oblique types and typal relevancies, *Educ. Psychol. Measmt.*, Vol. 17, pp. 207-222.
- SNEATH, P. H. A. (1957). The application of computers to taxonomy, *J. gen. Microbiol.* Vol. 17, pp. 201-226.
- VAN RIJSBERGEN, C. J. (1970). A fast hierarchic clustering algorithm. *The Computer Journal*, Vol. 13, pp. 324-326.
- WISHART, D. (1969). An algorithm for hierarchical classifications, *Biometrics*, Vol. 25, pp. 165-170.

Book review

Understanding Natural Language, by Terry Winograd, 1972; 195 pages. (Edinburgh University Press, £4-00)

This is a reprint in book form of an article that recently filled an entire issue of the journal *Cognitive Psychology*.

Mr Winograd is to be congratulated on a most impressive piece of work. He has an imaginary robot called SHRDLU (I did not find any explanation of this name) which operates on a 'world' consisting of five cuboids of various shapes, colours and sizes, three pyramids and a box, all sitting on a table top. This 'world' does not in fact exist, but can be seen on a television screen. The robot has an arm that can lift these objects, move them elsewhere within the limits of the table top, and set them down again.

The robot can be asked questions, and be given instructions to perform removal and building operations. The book includes a fairly long example to demonstrate the sort of conversation and operations that are possible. While this example looks remarkable, one is not told what one would really like to know, namely

1. are all the author's conversations with the machine as good as this, or was the best one picked for the book?
2. what happens when someone other than the author gives the instructions?
3. what happens if the user, while using correct English, is deliberately perverse in trying to fool the machine?

The discussion of disentangling the syntax of English in general, and also trying to take the meaning into account within the limited

world of SHRDLU's experience, is detailed and thoughtful. Yet many questions and difficulties arise that the book does not discuss at all.

Two examples must suffice:

In a section on 'Analysis of Word Endings' it is shown how, given a word that is not in the dictionary, it may be modified to try for a more basic word. If you use the word 'babies' it will correctly try 'baby', but the flow-diagram given will also try 'ty' if 'ties' is not in the dictionary, without thinking of trying 'tie'.

In describing the definition facility it is said that if we say 'A "marb" is a red block which is behind a box', the system recognises that we are defining a new word If we then talk about 'two big marbs', the system will build a description exactly like the one for 'two big red blocks which are behind a box'.

This seems to lead us to the situation that if we define a train as 'an engine pulling a set of coaches' then two long trains must be 'two long engines pulling a set of coaches'.

But I do not wish to be too critical in face of such a fine effort. I admire not only the programming, but also the excellent work that has gone into producing such an informative and readable book. What a pity that it should have been given a front cover of so juvenile an appearance.

I. D. HILL (London)

[Note: SHRDLU is the top line of characters on a linotype machine, corresponding to QWERTYUIOP on a typewriter.]

Book Review Editor]