```
integer array hist, route[1:n]; integer i, j, k, m;
for i := 1 step 1 until n do hist[i] := 0;
for i := 2 step 1 until n do hist[B[i]] := hist[B[i]] + 1;
route[1] := j := k := 1;

comment route[j] is the current end point and if hist[route[i]] ≠ 0
  a further line can be found. hist[1] must be non-zero initially
  because the tree is connected;

for i := 2 step 1 until n do
L1:
if hist[k] = 0 then
  begin
  j := j - 1; k := route[j];
  goto L1
  end
else
  begin
  hist[k] := hist[k] - 1;
  for m := 2 step 1 until n do
  if k = B[m] then goto L2;
L2:xprint(j, k, m, C[m]); j := j + 1;
  route[j] := k := m; B[m] := - B[m]
  end
end mintreeprint
```

Algorithm AS 15

Single Linkage Cluster Analysis

By G. J. S. Ross

Rothamsted Experimental Station

Language

Algol 60.

Description and Purpose

This procedure uses the minimum spanning tree to compute a single linkage cluster analysis as described by Gower and Ross (1969). Two forms of output are provided, (i) a list of the members of each group at each level of clustering and (ii) a dendrogram which summarizes the information in (i).

Structure

The procedure call is singlelinkage (*n, delta, B, C, dlarge, groupprint, topprint, sideprint, printx*).

Formal parameters

Input:

n	integer	The number of points.
delta	real	The amount by which the clustering threshold is raised at each iteration.
dlarge	real	As defined in AS 13.
B	integer array	
C	real array	

Auxiliary procedures

(i) Procedure *groupprint* (m, n, G, H).
This procedure prints the list, $G[1:n]$, of groups (whose last members are non-zero in $H[1:n]$) formed at level m, according to local output conventions. Something like

```
print (m);
for p := 1 step 1 until n do
    begin
    print (G[p]);
    if H[p] = 1 then write ('*')
    end
```

is suitable, together with carriage control characters as required.

(ii) Procedure *topprint* prints topheadings for the dendrogram.
(iii) Procedure *sideprint*(i) prints a new line and a side title for each line of the dendrogram.
(iv) Procedure *printx*(j) prints three characters for each element of the dendrogram. These should be as follows:

$j = 0$	three spaces,
1	two underlines, space,
2	two underlines, vertical bar,
3	three underlines,
4	two spaces, vertical bar,
5	two underlines, vertical bar.

Time

Time depends partly on $n \times$ the number of iterations required to form a single cluster and partly on n^2.

Restrictions

None, except that if *delta* is too small an unnecessary amount of printing is required. About 10–15 iterations give a satisfactory dendrogram. If more than 20 iterations are required the full dendrogram is not printed, excess portions on the right-hand side being omitted.

Comments

In algorithms AS 13, AS 14 and AS 15 much space and time can be saved by use of packing techniques.

I am indebted to Mr A. J. H. Walter of Atlas Computer Laboratory, Chilton, who tested these algorithms.

Reference

See AS 13.

```
procedure singlelinkage(n, delta, B, C, dlarge, groupprint,
    topprint, sideprint, printx);

comment Algorithm AS 15, J.R.statist.Soc. C, (1969), Vol.18, No.1;

value n, delta, dlarge, C; integer n; real delta, dlarge;
integer array B; array C;
procedure groupprint, topprint, sideprint, printx;
  begin

  comment Single linkage clustering defines, for any given distance
    threshold 'level', clusters such that any two points of a cluster
    can be connected by a chain of links each of length less than
    'level', a property not possessed by any two points not in the
    same cluster. Each iteration raises the threshold by delta and
    amalgamates any clusters joined by a single link, thus forming
    a hierarchical structure which can be represented by a dendrogram.
    The process terminates when all points lie within a single cluster.
    Information is supplied by procedure Primtree, in arrays B and C.
          Points are listed in sorted order in array G, and the
    corresponding array H marks the last member of each cluster with
    a 1, otherwise the entry is 0. The array X stores code numbers
    for output of the dendrogram;

  integer i, j, k, m, p, q, r, s, t, u, v, w; real dmin, level;
  integer array G, H[1:n], X[1:20 X n], W1. w2[0:n];

  comment Clustering starts at the first integral multiple of delta
    which is greater than dmin, the shortest link of the minimum
    spanning tree;

  dmin := C[2];
  for i := 3 step 1 until n do
  if dmin > C[i] then dmin := C[i];
  for i := 1 step 1 until n do
    begin
    G[i] := i; H[i] := 1;
    X[i] := 3
    end ;
  p := 0;
  for level := delta X (1 + entier(dmin / delta)),
  level + delta while k ≠ 1 do
    begin

    comment For each link in array C that is shorter than level,
      two clusters are amalgamated. The amalgamation involves a
      reordering of arrays G and H and removal of the end-of-cluster
      marker from the earlier cluster. Links once used are increased
      by dlarge to prevent re-use;

    p := p + 1;
    for i := 2 step 1 until n do
    if C[i] < level then
      begin
      j := B[i]; C[i] := C[i] + dlarge;
      k := i;
      for m := 1 step 1 until n do
        begin
        if G[m] = j then q := m;
        if G[m] = k then r := m
        end ;
      if q > r then
        begin
        m := r; r := q;
        q := m
        end ;
      for s := q step 1 until n do
      if H[s] ≠ 0 then goto NEXT1;
```

```
NEXT1:for t := r - 1 step - 1 until 1 do
      if H[t] ≠ 0 then goto NEXT2;
NEXT2:t := t + 1; H[s] := 0;
      for r := t step 1 until n do
        begin
        W1[r - t] := G[r]; W2[r - t] := H[r];
        if H[r] ≠ 0 then goto NEXT3
        end ;
NEXT3:w := s + 1; u := r - t + 1;
      for m := t - 1 step - 1 until w do
        begin
        G[m + u] := G[m]; H[m + u] := H[m]
        end ;
      u := r - t;
      for m := 0 step 1 until u do
        begin
        G[m + w] := W1[m]; H[m + w] := W2[m]
        end
      end ;
    groupprint(level, n, G, H); w := n X p;
    u := v := k := 0;

    comment Dendrogram indicators are now compiled and stored in X.
      Points are examined in the order defined by the array G. s is
      the corresponding entry on the previous iteration, u = 0 for
      the first member of a cluster, v = 1 when amalgamations
      occur, from the last member of the first component cluster
      until the last member of the amalgamated cluster. k is the
      total number of clusters;

    for i := 2 step 1 until n do k := k + H[i];
    if p < 20 then
    for i := 1 step 1 until n do
      begin
      j := G[i]; s := X[j + w - n];
      if u = 0 then
        begin
        if H[i] = 1 then t := 3 else
        if s = 3 then t := u := v := 1 else
          begin
          t := 0; u := 1
          end
        end
      else
      if H[i] = 1 then
        begin
        if v = 0 then
          begin
          t := 3; u := 0
          end
        else
          begin
          t := 2; u := v := 0
          end
        end
      else
      if s = 2 ∨ s = 3 then
        begin
        if v = 0 then t := u := v := 1 else
          begin
          t := 5; u := 1
          end
        end
      else
      if v = 0 then
        begin
        t := 0; u := 1
        end
```

```
      else t := 4;
      X[j + w] := t
      end
    end ;
topprint;
for i := 1 step 1 until n do
  begin
  j := G[i]; sideprint(j);
  if p > 19 then p := 19;
  for m := 0 step 1 until p do printx(X[m X n + j])
  end

comment Print newlines to separate the dendrogram. Packing in
  the X store saves considerable space and allows non-significant
  blank spaces before newline to be recognised and ignored.
  Packing G and H together simplifies the reordering process

end singlelinkage
```

Algorithm AS 16

Maximum Likelihood Estimation from Grouped and Censored Normal Data

By A. V. Swan

MRC Clinical Research Centre, London

Language

Algol 60.

Description and Purpose

The Algol procedure "Maxlicens" is designed to obtain the maximum likelihood estimates of the mean and standard deviation from a Normal sample, which may be censored or grouped in any way. At the same time approximations to their variances and covariance are obtained.

The theoretical basis of the procedure is described in Swan (1969).

Structure

The procedure is called as: Maxlicens (*n*, *x*, *P*, *mean*, *sigma*, *e*1, *e*2, *Maxits*, *cov*, *nobs*, *RMills*, *k*, *ifault*);

Formal parameters

n	integer	The number of observations.
x	real array	*Bounds* [1 : *n*, 1 : 2] containing the observations, excluding missing values.
P	integer array	*Bounds* [1 : *n*] containing a code describing the observations in *x*.