Department of the Interior

U. S. Geological Survey

Open-File Report 78 - 550

MODEL, a computer program for calculating weights and roots of characteristic analysis models.

by

J. Thomas Hanley

1978

Contents

Scope and Purpose

Characteristic analysis is a method designed to treat various domains of geologic data (i.e., geologic, geochemical, geophysical, remote sensing) that have been transformed to Boolean form where "1" means a "favorable" value, "0" means an indeterminate value and "-1" means an "unfavorable" value (Botbol, et al., 1978). Favorability is defined by the condition that a measured value of a variable is higher than the immediately adjacent values.

MODEL is an integral part of characteristic analysis. It provides users with a way to familiarize themselves with the three methods of calculating variables weights external to CHARAN, the characteristic analysis program.

This report offers the user a guide for using MODEL and contains worked examples. It also provides a brief overview of the statistics and basis algorithm.

Acknowledgements

The trinomial probability equation was formulated by R. B. McCammon and D. H. Root. Other equations and the LROOT2 coding were modified from CHARAN (characteristic analysis program) written by J. M. Botbol, R. B. McCammon, R. W. Bowen and the author.

Computer Restrictions

This program was written for use interactively on the Honeywell Multics system in conjunction with a Tektronix 4014. The minimum that a user must know to use MODEL is the login procedure and how to link to the MODEL object segment (see Appendix B). The capabilities of the user are enhanced by knowledge of editors, which can be used to enter models prior to invocation of the program. The program will ask if a Tektronix is being used at the beginning of a session.

Program Limitations

The greatest limitation of this program is portability. MODEL was written for the Honeywell Multics and uses three machine-dependent subroutines: assoc, closer and filprnt. Assoc is a subroutine that attaches and opens a file, utilizing a given unit number and file type. Closer is an entry name in assoc and closes and detaches the data file given the unit number. Filprnt prints a segment, in this case help_model.

This program can be changed to batch mode easily but it loses its versatility to the user.

The number of cells and variables is limited to 10 each. This can be increased by changing all entries in common block /a/, other arrays dimensioned at 10 and any format statements that deal with reading or writing one of the above arrays.

Data Input

Input data consists of the number of cells, the number of variables and the model. The model is entered as the symbols -, 0 and + which correspond to -1, 0 and +1 respectively. These data may be entered into a character segment prior to invoking MODEL via a text editor. The setup of the segment is as follows:

line 1:	number of cells (ncell): i2
	number of variables (nvar): i2
lines 2-11:	each line consists of the symbols for each variable of a cell, up to 10 variables: 10a1

An example of an input segment can be found in Appendix A. The name of the segment must be no more than 8 characters.

Input data may alternatively be entered via the terminal. The program prompts for the data (see Apendix C). The user has the option to save a model entered via the terminal either when a new model is to be entered or at the end of a session.

Program design and usage

After linking to MODEL, upon execution, the program prompts for the transmission rate i.e., 30, 120, 960. A series of paragraphs describing the structure of each matrix follows.

Next is the option of model entry. There are two ways of entering data once within the program, either from an external segment or from the terminal.

Upon model entry, the program then calculates and displays the product, tally and probability matrices. The product matrix P is calculated as:

$$P = X'X.$$

The product matrix is a v x v matrix where v is the number of variables.

The tally matrix, T, is also a v x y matrix whose diagonal elements, T_{ii}, equal the number of positive occurrences of variable i. The upper triangular part of T, where $i < j$, is the number of positive-positive matches for variables i and j. The lower triangular part of T, where $i > j$, is the number of negative-negative matches of variables i and j.

The probability matrix, M, is a v x v matrix with diagonal elements of 1.0. The upper triangular part of the matrix equals the sum of T_{ij} and T_{ji} which are the number of positive-positive matches and negative-negative matches, respectively for variables i and j. The lower triangular matrix consists of trinomial probabilities multiplied by 100 to yield percentages. The trinomial probability, r_{ij}, is calculated using the following equation:

let $k = T_{ij} + T_{ji}$

then

$$r_{ij} = \frac{\sum_{\mu=o}^{k-1} \sum_{\upsilon=o}^{\mu} \binom{T_{ii}}{\upsilon} \binom{P_{ii}-T_{ii}}{\mu-\upsilon} \sum_{\alpha=o}^{T_{jj}-\upsilon} \binom{n-P_{ii}}{\alpha} \binom{P_{ii}-T_{ii}-\mu+\upsilon}{T_{jj}-\upsilon-\alpha} \sum_{\beta=o}^{P_{jj}-T_{jj}-\mu+\upsilon} \binom{n-P_{ii}-\alpha}{\beta} \binom{T_{ii}-\upsilon}{P_{jj}-T_{jj}-\mu+\upsilon-\beta}}{\dfrac{n!}{T_{jj}!\,(n-P_{jj})!\,(P_{jj}-T_{jj})!}}$$

The trinomial probability, r_{ij}, is defined for use in the program model as the probability of observing up to k matches which is interpreted as not being due to chance for variables i and j.

Upon printing out the three matrices, the program prompts for the type of calculation required by the user. There are three methods for obtaining weights for the chosen variables: 1) sum of squares method, 2) first principal component of the product matrix and 3) first principal component of the probability matrix. The input for the sum of squares method is the product matrix, P. The weights obtained from the sum of squares method are calculated using:

$$w_i = (\sum_{j=1}^{v} P_{ij}^2)^{\frac{1}{2}} \quad \sum_{i=1}^{v} (\sum_{j=1}^{v} P_{ij}^2)^{\frac{1}{2}}$$

where w_i is the weight of variable i.

The input for the first principal component of the product matrix is also P. The weights are obtained by solving the equation

$$| P- \lambda I| =0$$

where λ is the largest characteristic root of the matrix. The eigenvector of P is obtained by solving

$$Pw = \lambda w$$

where $w'w = 1$, w is an v x 1 vector which provides the weights for the variables.

M is the input for the first principal component of the probability matrix. The weights are calculated as the eigenvector corresponding to the largest characteristic root of M.

Following calculation of the weights, the program then asks if a new model is to be entered, and if so, asks if the old model is to be saved as a segment? If not, it asks if another calculation is to be performed with the existing model. Only one set of weights are calculated on each pass of the program. The user has the option of saving each model as a segment if the model was entered via the terminal during program execution.

Appendix A, Input segment example

The input segment name must be 8 characters or less. The first line consists of the numbers of cells (ncell) and the number of variables (nvar) both with i2 formats.

Lines 2-11 consist of the symbols -, 0, and + which represent -1, 0, and +1 respectively. Each line consists of the symbols for each variable of a cell and is read with a 10a1 format.

Example:

```
 4 6
+0--++
0++++-
---+-0
-+-+0+
```

Appendix B, Link to Model

```
lk >udd>ORERES>THanley>charan_model>model
```

```
    model
 tektronix?

no
 do you want to see discription of program?

yes
     This program expects as input the number of cells(ncell) and variables
(nvar) in the model up to a maximum of 10 each.  The model itself
consists of -1, 0, and +1.  In the ouptut of the model matrix these
are represented by -, 0, and + respectively.  The product matrix
(nvar x nvar) is the model matrix premultiplied by its transpose.
This matrix is used in the 'primitive' or sum of squares method.
     Along the diagonal of the tally matrix (nvar x nvar) are the total
number of positive occurences of each variable.  The upper triangular
matrix consists of the positive-positive matches of the variables
and the lower triangular matrix consists of the negative-negative
matches.  This matrix is used in the 1st principal component
calculation.
     The diagonal of the probability matrix consists of 1.0.
The upper triangular matrix consists of the positive-positive and
negative-negative matches obtained from the tally matrix and the lower
triangular matrix consists of the trinomial probabilities.


 do you want to enter model via terminal?

yes
 enter number of cells

4
 enter number of variables

6
 enter the model using - for -1, 0 for 0, and + for +1
   each line consisting of all variables for each cell

+0--++
0++++-
-00+-0
-+-+0+
 model matrix

+0--++
0++++-
-00+-0
-+-+0+
product matrix

   3  -1   0  -3   2   0
  -1   2   0   2   1   0
   0   0   3   1   0  -3
  -3   2   1   4  -1  -1
   2   1   0  -1   3   0
   0   0  -3  -1   0   3
```

```
tally matrix

   1   0   0   0   1   1
   0   2   1   2   1   1
   1   0   1   1   1   0
   0   0   1   3   1   1
   1   0   0   0   2   1
   0   0   0   0   0   2
 probability matrix

   1   0   1   0   2   1
   0   1   1   2   1   1
  17  50   1   2   1   0
   0  50  50   1   1   1
  67  17  33   0   1   1
  33  17   0   0  17   1
 enter type of calculation required
    1: primitive type
    2: 1st principal component
    3: probability without replacement

1
variable  characteristic  rank
 number        weight
    1            0.183        2
    2            0.121        5
    3            0.166        3
    4            0.216        1
    5            0.148        4
    6            0.166        3
do you want to enter a new model?

no
do you want to run another calculation on same model?

yes
 enter type of calculation required
    1: primitive type
    2: 1st principal component
    3: probability without replacement

3
variable  characteristic  rank
 number        weight
    1            0.365        5
    2            0.460        2
    3            0.514        1
    4            0.396        4
    5            0.431        3
    6            0.221        6
 characteristic root=  0.22e+01

do you want to enter a new model?

yes
```

```
---
 do you want to save this model in a segment?

yes
 enter name of output file

model1
 do you want to enter model via terminal?

no
 enter name of segment containing model (a8)

md3
 model matrix

+++00
+++00
+00++
0+0++
00+++
product matrix

   3   2   2   1   1
   2   3   2   1   1
   2   2   3   1   1
   1   1   1   3   3
   1   1   1   3   3
tally matrix

   3   2   2   1   1
   0   3   2   1   1
   0   0   3   1   1
   0   0   0   3   3
   0   0   0   0   3
 probability matrix

   1   2   2   1   1
  30   1   2   1   1
  30  30   1   1   1
   0   0   0   1   3
   0   0   0  90   1
 enter type of calculation required
    1: primitive type
    2: 1st principal component
    3: probability without replacement

1
variable  characteristic  rank
 number        weight
    1           0.196       2
    2           0.196       2
    3           0.196       2
    4           0.206       1
    5           0.206       1
do you want to enter a new model?

no
```

```
do you want to run another calculation on same model?

yes
 enter type of calculation required
    1: primitive type
    2: 1st principal component
    3: probability without replacement

2
variable  characteristic  rank
 number       weight
    1          0.447        1
    2          0.447        1
    3          0.447        1
    4          0.447        1
    5          0.447        1
 characteristic root=  0.90e+01

do you want to enter a new model?

no
do you want to run another calculation on same model?

no

STOP

fortran_io_: Close files?   yes

r 1007 3.115 37.218 804
```

model.fortran

```
c     main program for determining weights of characteristic-analysis models
c
      common /a/ mstore(10,10),model1(10,10),ic(10,10),iprob(10,10),exp(10,10)
      dimension ssq(10),rank(10,2),mat(10,10),itrans(10,10),xmat(10,10)
      double precision ifile
      data yes/"y"/
c
c     tektronix calls and program explanation
c
      print 1
 1    format(1x," tektronix?"/)
      call ans(pz)
      if(pz.ne.yes) go to 7
      print 5
 5    format(1x," enter transmission rate"/)
      read,ibaud
      call initt(ibaud)
      call newpag
 7    print 8
 8    format(1x," do you want to see discription of program?"/)
      call ans(pauz)
      if(pauz.eq.yes) call filprnt("help_model")
c
c     entry of model
c
 10   print 15
 15   format(1x," do you want to enter model via terminal?"/)
      call ans(pz2)
      if(pz2.eq.yes) go to 45
      print 20
 20   format(1x," enter name of segment containing model (a8)"/)
      read(5,25) ifile
 25   format(a8)
      call assoc(10,ifile,"si  ")
      read(10,30) ncell,nvar
 30   format(2i2)
      do 40 i=1,ncell
      read(10,35) (mstore(i,j),j=1,nvar)
 35   format(10a1)
 40   continue
      call closer(10)
      go to 80
 45   print 50
 50   format(1x," enter number of cells "/)
      read,ncell
      print 55
 55   format(1x," enter number of variables "/)
      read,nvar
      print 60
 60   format(1x," enter the model using - for -1, 0 for 0, and + for +1"/"    ea
\cch line consisting of all variables for each cell "/)
      do 70 i=1,ncell
      read(5,65) (mstore(i,j),j=1,nvar)
 65   format(10a1)
 70   continue
c
c     calculation of product, tally and probability matrices
```

```
c
 80   call translat(ncell,nvar,$140)
      if(pz.eq.yes) call newpag
      call iccalc(ncell,nvar)
      call transpos(ncell,nvar,itrans)
      call mult(ncell,nvar,itrans,mat)
      call coprob(ncell,nvar,mat)
      call writem(ncell,nvar,mat)
c
c     calculation of weights
c
 90   print 95
 95   format(1x,' enter type of calculation required'/5x,'1: primitive type'/5x,
\c'2: 1st principal component'/5x,'3: probability without replacement'/)
      read,itype
      go to(100,110,120) itype
c
c     calculation of weights using primitive method
c
 100  call prim(ncell,nvar,rank,mat)
      root=1.0
      iswit=1
      go to 130
c
c     calculation of weights using first principal component
c
 110  do 115 i=1,ncell
      do 115 j=1,nvar
      xmat(i,j)=float(mat(i,j))
 115  continue
      call lroot2(xmat,nvar,root,rank,ncell,$140)
      iswit=2
      go to 130
c
c     calculation of weights using probability without replacement
c
 120  call lroot2(exp,nvar,root,rank,ncell,$140)
      iswit=3
c
c     ranking and display of weights
c
 130  call rankem(rank,nvar)
      call displa(rank,nvar,root,iswit)
c
c
 140  print 150
 150  format(1x,'do you want to enter a new model?'/)
      call ans(pauz2)
      if(pauz2.eq.yes.and.pz2.eq.yes) go to 165
      if(pauz2.eq.yes) go to 10
      print 160
 160  format(1x,'do you want to run another calculation on same model?'/)
      call ans(pauz)
      if(pauz.eq.yes) go to 90
      if(pz2.ne.yes) go to 220
 165  print 170
 170  format(1x,' do you want to save this model in a segment?'/)
```

```
      if(pauz.ne.yes) go to 210
      print 180
180   format(1x," enter name of output file"/)
      read(5,25) ifile
      call assoc(11,ifile,"so  ")
      write(11,30) ncell,nvar
      do 200 i=1,ncell
      write(11,190) (mstore(i,j),j=1,nvar)
190   format(10a1)
200   continue
      call closer(11)
210   if(pauz2.eq.yes) go to 10
220   stop
      end
```

ans.fortran

```
        subroutine ans(pauz)
        data yes/'y'/
        data sno/'n'/
        data blk/'    '/
        resp=blk
305     read(5,306) resp
306     format(a1)
        if((resp.eq.yes).or.(resp.eq.sno)) go to 308
        print 310
310     format(' please enter yes or no'/)
        go to 305
308     pauz=sno
        if(resp.eq.yes) pauz=yes
        resp=blk
        return
        end
```

help_model

This program expects as input the number of cells(ncell) and variables (nvar) in the model up to a maximum of 10 each. The model itself consists of -1, 0, and +1. In the ouptut of the model matrix these are represented by -, 0, and + respectively. The product matrix (nvar x nvar) is the model matrix premultiplied by its transpose. This matrix is used in the "primitive" or sum of squares method.

Along the diagonal of the tally matrix (nvar x nvar) are the total number of positive occurences of each variable. The upper triangular matrix consists of the positive-positive matches of the variables and the lower triangular matrix consists of the negative-negative matches. This matrix is used in the 1st principal component calculation.

The diagonal of the probability matrix consists of 1.0. The upper triangular matrix consists of the positive-positive and negative-negative matches obtained from the tally matrix and the lower triangular matrix consists of the trinomial probabilities.

translat.fortran

```
      subroutine translat(ncell,nvar,*)
c
c     subroutine to translate the symbols -,0,+ into the integers -1,0,+1 respec
\ctively
c
      common /a/ mstore(10,10),modell(10,10),ic(10,10),iprob(10,10),exp(10,10)
      dimension isym(3)
      data isym/'-','0','+'/
      do 20 j=1,nvar
      do 20 i=1,ncell
      do 10 k=1,3
      if(mstore(i,j).eq.isym(k)) go to 20
 10   continue
      write(6,15) mstore(i,j),i,j
 15   format(1x,'bad data ',a1,' in position ',i2,1x,i2)
      return 1
 20   modell(i,j)=k-2
      return
      end
```

iccalc.fortran

```
      subroutine iccalc(ncell,nvar)
c
c     subroutine to calculate the tally matrix
c
      common /a/ mstore(10,10),modell(10,10),ic(10,10),iprob(10,10),exp(10,10)
      do 10 i=1,nvar
      do 10 j=1,nvar
      ic(i,j)=0
 10   continue
      do 50 i=1,nvar
      do 40 j=1,ncell
      if(modell(j,i).eq.0) go to 40
      if(modell(j,i).gt.0) ic(i,i)=ic(i,i)+1
      l=i+1
      if(l.gt.nvar) go to 40
      do 30 k=l,nvar
      if(modell(j,k).gt.0.and.modell(j,i).gt.0) ic(i,k)=ic(i,k)+1
      if(modell(j,k).lt.0.and.modell(j,i).lt.0) ic(k,i)=ic(k,i)+1
 30   continue
 40   continue
 50   continue
      return
      end
```

transpos.fortran

```
      subroutine transpos(ncell,nvar,itrans)
c
c     subroutine to calculate the transpose of model matrix
c
      common /a/ mstore(10,10),modell(10,10),ic(10,10),iprob(10,10),exp(10,10)
      dimension itrans(10,10)
      do 10 i=1,ncell
      do 10 j=1,nvar
      itrans(j,i)=modell(i,j)
 10   continue
      return
      end
```

mult.fortran

```
      subroutine mult(ncell,nvar,itrans,mat)
c
c     subroutine to premultiply the model matrix by its transpose
c
      common /a/ mstore(10,10),modell(10,10),ic(10,10),iprob(10,10),exp(10,10)
      dimension itrans(10,10),mat(10,10)
      do 10 i=1,nvar
      do 10 j=1,nvar
      mat(i,j)=0
      do 10 k=1,ncell
      mat(i,j)=mat(i,j)+itrans(i,k)*modell(k,j)
 10   continue
      return
      end
```

coprob.fortran

```
      subroutine coprob(ncell,nvar,mat)
c
c     subroutine to calculate the probability matrix
c
      common /a/ mstore(10,10),modell(10,10),ic(10,10),iprob(10,10),exp(10,10)
      dimension mat(10,10)
      do 40 i=1,nvar
      do 40 j=1,nvar
      if(i-j) 10,20,30
 10   iprob(i,j)=ic(j,i)+ic(i,j)
      go to 40
 20   exp(i,j)=1.0
      iprob(i,j)=1
      go to 40
 30   k=ic(i,j)+ic(j,i)
      it=ic(i,i)
      jt=ic(j,j)
      ip=mat(i,i)
      jp=mat(j,j)
      call prob(ncell,it,jt,ip,jp,k,ipro)
      iprob(i,j)=ipro
      exp(i,j)=float(ipro)/100.
      exp(j,i)=float(ipro)/100.
 40   continue
      return
      end
```

prob.fortran

```
      subroutine prob(n,ti,tj,pi,pj,k,p)
c
c     subroutine to calculate trinomial probability
c
      integer ti,tj,pi,pj,p,alpha,beta
      fact=f(n)/(f(tj)*f(n-pj)*f(pj-tj))
      k2=k
      if(k2.le.0) go to 105
      sum=0.0
      do 100 i=1,k2
      mu=i-1
      do 100 j=1,i
      nu=j-1
      it1=tj-nu+1
      if(it1.lt.0) go to 15
      sum34=0.0
      do 50 l=1,it1
      alpha=l-1
 15   it2=pj-tj-mu+nu+1
      if(it2.lt.0.and.it1.lt.0) go to 75
      if(it2.lt.0) go to 30
      sum12=0.0
      do 25 m=1,it2
      beta=m-1
      c1=b((n-pi-alpha),beta)
      if(c1.eq.0.0) go to 30
      c2=b((ti-nu),(pj-tj-mu+nu-beta))
      sum12=sum12+c1*c2
 25   continue
 30   c3=b((n-pi),alpha)
      if(c3.eq.0.0) go to 75
      c4=b((pi-ti-mu+nu),(tj-nu-alpha))
      sum34=sum34+c3*c4*sum12
 50   continue
 75   c5=b(ti,nu)
      if(c5.eq.0.0) go to 100
      c6=b((pi-ti),(mu-nu))
      sum=sum+c5*c6*sum34
 100  continue
      temp=(sum/fact)*100.
      p=ifix(temp)
      if((temp-p).gt..5) p=p+1
      go to 110
 105  p=0
 110  return
      end
```

f.fortran

```
      real function f(n)
c     real function that calculates the factorial of n
c
      f=1
      if(n.le.1) return
      do 10 i=1,n
 10   f=f*i
      return
      end
```

b.fortran

```
      real function b(n,r)
c     real function that calculates the combinatorial of !n!
c                                                        !r!
c
      integer r
      b=0
      if((r.gt.n).or.(r.lt.0)) return
      nr=n-r
      b=f(n)/(f(r)*f(nr))
      return
      end
```

writem.fortran

```
      subroutine writem(ncell,nvar,mat)
c
c     subroutine for writing the product, tally and probability matrices and dis
\cplaying the ranked weights
c
      common /a/ mstore(10,10),modell(10,10),ic(10,10),iprob(10,10),exp(10,10)
      dimension rank(10,2),mat(10,10)
      print 5
 5    format(1x," model matrix"/)
      do 25 i=1,ncell
      write(6,20) (mstore(i,j),j=1,nvar)
 20   format(1x,10a1)
 25   continue
      print 30
 30   format(1x,"product matrix"/)
      do 40 i=1,nvar
      write(6,35) (mat(i,j),j=1,nvar)
 35   format(1x,10i4)
 40   continue
      print 50
 50   format(1x,"tally matrix"/)
      do 65 i=1,nvar
      write(6,35) (ic(i,j),j=1,nvar)
 65   continue
      print 70
 70   format(1x," probability matrix"/)
      do 80 i=1,nvar
      write(6,35) (iprob(i,j),j=1,nvar)
 80   continue
      go to 110
c
c     entry point displa : for displaying ranked weights
c
      entry displa(rank,nvar,root,iswit)
      print 85
 85   format(1x,"variable",2x,"characteristic",2x,"rank"/2x,"number",7x,"weight"
\c)
      do 95 i=1,nvar
      ir=ifix(rank(i,2))
      write(6,90) i,rank(i,1),ir
 90   format(4x,i2,9x,f6.3,7x,i2)
 95   continue
      if(iswit.lt.2) go to 110
      write(6,100) root
 100  format(1x," characteristic root= ",e9.2/)
 110  return
      end
```

prim.fortran

```
      subroutine prim(ncell,nvar,rank,mat)
c
c     subroutine to calculate weights using the sum of squares method
c
      dimension rank(10,2),itrans(10,10),mat(10,10),ssq(10)
      stot=0.0
      do 10 i=1,nvar
      ssq(i)=0.0
      do 5 j=1,nvar
      ssq(i)=ssq(i)+(mat(i,j)**2)
 5    continue
      ssq(i)=sqrt(ssq(i))
      stot=stot+ssq(i)
 10   continue
      do 15 i=1,nvar
      rank(i,1)=ssq(i)/stot
 15   continue
      return
      end
      subroutine lroot2(s,n,root,vect,ncell,*)
c SUBROUTINE TO CALCULATE WEIGHTS OF CHARACTERISTICS BY USE
c OF EIGENVALUES
      dimension vect(10,2),v(10),v1(10),t(10),s(10,10)
      test=.00001
      it=0
      k=200
      vmax=0.
      do 10 j=1,n
 10   vmax=vmax+abs(s(1,j))
      nmax=1
      do 25 j=1,n
      prod=0
      do 20 jj=1,n
      prod=prod+abs(s(j,jj))
 20   continue
      if(prod.lt.vmax) go to 25
      vmax=prod
      nmax=j
 25   continue
      do 30 j=1,n
      v(j)=1
      if(s(nmax,j).lt.0.) v(j)=-1
      if(s(j,j).eq.0.) v(j)=0
 30   continue
 40   it=it+1
      do 50 i=1,n
      v1(i)=0
      do 50 j=1,n
 50   v1(i)=v1(i)+s(i,j)*v(j)
      root=v1(nmax)
      if(root.le.0.) go to 70
      sumt=0
      do 60 i=1,n
      vect(i,1)=v1(i)/v1(nmax)
      t(i)=abs(v(i)-vect(i,1))
      sumt=sumt+t(i)
 60   v(i)=vect(i,1)
      k=k-1
      if(k)65,65,90
 65   if(sumt1-sumt)70,90,90
 70   print 75
 75   format(' not converging'///)
      do 80 i=1,n
 80   vect(i,1)=0
      root=0
      return 1
 90   sumt1=sumt
      if(sumt-test)100,100,40
 100  sumv=0
      do 110 i=1,n
 110  sumv=sumv+vect(i,1)*vect(i,1)
      den=sqrt(sumv)
      do 120 i=1,n
 120  vect(i,1)=vect(i,1)/den
      return
      end
```

rankem.fortran

```
      subroutine rankem(rank,nvar)
c
c     subroutine to rank the weights in descending order
c
      dimension rank(10,2),temp(10),itemp(10)
      do 10 i=1,nvar
      temp(i)=rank(i,1)
      itemp(i)=i
 10   continue
      do 30 i=1,nvar
      k=nvar-i
      do 20 j=1,k
      if(temp(j).ge.temp(j+1)) go to 20
      store=temp(j)
      istore=itemp(j)
      temp(j)=temp(j+1)
      itemp(j)=itemp(j+1)
      temp(j+1)=store
      itemp(j+1)=istore
 20   continue
 30   continue
      icnt=1
      rank(itemp(1),2)=1
      do 40 i=2,nvar
      it1=ifix(1000.*temp(i))
      it2=ifix(1000.*temp(i-1))
      if(it1.ne.it2) go to 35
      rank(itemp(i),2)=icnt
      go to 40
 35   icnt=icnt+1
      rank(itemp(i),2)=icnt
 40   continue
      return
      end
```

References cited

Botbol, J. M., Sinding-Larsen, R., McCammon, R. B., and Gott, G.B., 1978, A regionalized multivariate approach to target selection in geochemical exploration: Economic Geology, in press.