Kansas Working Papers in Linguistics

Volume 8, Number 1

Edited by Letta Strantzali

The editor would like to express her thanks to the faculty and staff for their help in the preparation of this volume. Funding for this journal is provided by the Graduate Student Council from the Student Activity Fee.

© Linguistics Graduate Student Association
University of Kansas, 1983

Reprinted 1992

Kansas Working Papers in Linguistics

Volume 8, number 1, 1983

CONTENTS

COMPUTERIZED PERMUTATION OF PIKEAN FIELD MATRICES

Kenneth L. Miner and Barbara Lynn Taghva
University of Kansas

Present-day computer technology ought to encourage re-examination of pre-computer linguistic concepts, especially tools for analysis, since some earlier ideas, difficult to apply or work with when offered, may turn out to be useful if computerized. Here we give one example of such an investigation, that of Pike's field matrix permutation.

Pike 1962 explored the concept of grammatical systems, or more properly sub-systems, as matrices, with categories (syntactic, morphological, phonological, semantic) as parameters and formatives as intersections of parameters, or cells. Pike 1963 put forward the idea of matrix permutation as a tool in linguistic analysis. This technique, like many, perhaps all, good ones, was of course always implicit in linguistic work, and Pike's contribution was to show how it might be systematized. We will have nothing to say here about the implications of matrix permutation for Pike's theory of language, or indeed for any theory of language, but will regard it merely as a tool for organizing data and revealing generalities.

In our discussion we will use, as data, Taos verbal prefixes as described in Trager 1946.[1] One first sets up a matrix of the familiar sort, with rows and columns determined by an arbitrary arrangement of categories, as in Table 1.

Table 1. Initial Taos Matrix

SUBJ \ OBJ	A	B	C	1 sg	1 du	1 pl	2 sg	2 du	2 pl
2 sg	o	ku	i	may	may	may	X	X	X
3 sg	Ø	u	i	o	ąn	i	ą	mąn	mą
1 du	ąn	kąn	ąpęn	X	X	X	ą	mąpęn	mąpi
3 du	ąn	ąn	ąpęn	o	ąn	i	ą	mąn	mą
2 du	mąn	mąn	mąpęn	may	may	may	X	X	X
2 pl	mą	mąw	mąpi	may	may	may	X	X	X
1 pl	i	kiw	ipi	X	X	X	ą	mąpęn	mąpi
3 pl	i	iw	ipi	o	ąn	i	ą	mąn	mą
1 sg	ti	o	pi	X	X	X	ą	mąpęn	mąpi

Then successive permutations of rows and columns are made in order to bring similar formatives together. The goal is to obtain blocks (and other patterns of distribution; see below) so that generalizations can be made. For the data in the initial matrix of Table 1, a possible final matrix is given in Table 2.

Table 2. Final Taos Matrix.

SUBJ \ OBJ	2 sg	1 pl	1 sg	1 du	A	B	C	2 pl	2 du
3 sg	ą	i	o	ąn	Ø	u	i	mą	mąn
3 pl	ą	i	o	ąn	i	iw	ipi	mą	mąn
3 du	ą	i	o	ąn	ąn	ąn	ąpęn	mą	mąn
1 du	ą	X	X	X	ąn	kąn	ąpęn	mąpi	mąpęn
1 pl	ą	X	X	X	i	kiw	ipi	mąpi	mąpęn
1 sg	ą	X	X	X	ti	o	pi	mąpi	mąpęn
2 sg	X	may	may	may	o	ku	i	X	X
2 pl	X	may	may	may	mą	mąw	mąpi	X	X
2 du	X	may	may	may	mąn	mąn	mąpęn	X	X

It is revealing to line in the occurring patterns of distribution, as has been done here.

At least five generalizations can be tentatively made from Table 2:

i. ą- marks 2 sg OBJ
ii. mą- marks involvement of 2nd person
iii. -n- marks dual number
iv. -pi-, with alternant -pę- before -n, marks 1 acting on 2
v. may- marks 2 acting on 1, and takes precedence over, or in Pike's terms "outranks", ii. and iii.

Pike himself would no doubt carry the analysis further, but we wish here only to illustrate the method. Note that in the "final" matrix there is a residue involving i- and o-. A new matrix can sometimes be set up to examine, by the same method, such a remainder.

Although the exact pattern of distribution of formatives depends partly on how the parameters are set up, certain patterns tend to recur in a significant way:

1. A <u>linear</u> pattern, such as that for ą-, tends to reflect marking of one parameter on one axis, in this case, 2nd sg object.

2. A <u>block</u> pattern, such as that for may-, tends to reflect marking of a combination of parameters, one on each axis, in this case, 2nd person acting on 1st person.

3. An <u>L-shaped</u> pattern, such as that for -n-, tends to reflect marking of one parameter on both axes ("involvement" marking), as in this case, involvement of dual, either as SUBJ or OBJ.[2]

4. A <u>stairway</u> pattern, such as that for the Koryak prefix ne- vs. its absence (Comrie 1980) shown in Table 3, tends to reflect hierarchical marking, as in this case, where the hierarchy is 1p > 2p > 3 sg > 3 pl: ne- occurs when the subject is lower on the (animacy) hierarchy than the object. For Algonquian examples of this see Morgan 1966.

Table 3. Koryak prefix ne-.

OBJ \ SUBJ	1p	2p	3p sg	3p pl
1p		ne-	ne-	ne-
2p			ne-	ne-
3 sg				ne-
3 pl				ne-

(adapted from Comrie 1980: Table 3, p. 68)

The matrix permutation technique was applied to several languages in Erikson 1962 and 1965 and Pike and Erikson 1964. Hockett's negative response to the Potawatomi applications (Hockett 1966), which became quite well-known especially among Indianists, may be one reason for the subsequent lack of interest in the technique, at least as far as the literature reveals (but for a recent re-statement see Pike 1975). This criticism, however, had much more to do with the handling of the Potawatomi data than with the technique of analysis.

Another reason for subsequent neglect of the technique may well be thatit is quite laborious. There seems to be no alternative to drawing a new matrix after each permutation or two, increasing the likelihood of error each time. It should be obvious that, in principle, very complex data could be sorted out quite easily with a computer program and a CRT display.

Our program is written in FORTRAN, which, although it was not designed for character manipulation or list processing, seemed best for the initial experiment. Among its initial advantages are its portability (it can be run on many types of equipment) and its usability by untrained investigators. More important, the type of formatting required for matrix permutation turns out to be difficult to handle with other presently available programming languages.

An actual run of our program follows, except that since thirteen matrices are involved (our program will only carry out one permutation at a time) we will spare the reader all but the first and last; and we will spare him/her these also, since they are only slightly coded versions of Tables 1 and 2 respectively.

After the run, the program itself is given.

Finally, we close with a brief discussion of problems and limitations.

```
*FTN -NWRN /HOMEWORK/MATRIX1



NO.OF ROWS YOU NEED
=10
NO.OF COLUMNS YOU NEED
=10
TYPE IN ROW  1 SEPARATED BY COMMA'S
= ,A,B,C,1sg,1dl,1pl,2sg,2dl,2pl
TYPE IN ROW  2 SEPARATED BY COMMA'S
=2psg,o,ku,i,may,may,may,x,x,x
TYPE IN ROW  3 SEPARATED BY COMMA'S
=3psg,0,u,i,o,An,i,A,mAn,mA
TYPE IN ROW  4 SEPARATED BY COMMA'S
=1pdl,An,kAn,Apn,x,x,x,A,mApn,mApi
TYPE IN ROW  5 SEPARATED BY COMMA'S
=3pdl,An,An,Apn,o,An,i,A,mAn,mA
TYPE IN ROW  6 SEPARATED BY COMMA'S
=2pdl,mAn,mAn,mApn,may,may,may,x,x,x
TYPE IN ROW  7 SEPARATED BY COMMA'S
=2ppl,mA,mAw,mApi,may,may,may,x,x,x
TYPE IN ROW  8 SEPARATED BY COMMA'S
=1ppl,i,kiw,ipi,x,x,x,A,mApn,mApi
TYPE IN ROW  9 SEPARATED BY COMMA'S
=3ppl,i,iw,ipi,o,An,i,A,mAn,mA
TYPE IN ROW 10 SEPARATED BY COMMA'S
=1psg,ti,o,pi,x,x,x,A,mApn,mApi
YOUR MATRIX IS
```

[coded version of Table 1]

```
TYPE COMMAND
=C 10 5
YOUR MATRIX IS...

TYPE COMMAND
=C 10 6
YOUR MATRIX IS...

TYPE COMMAND
=R 2 8
YOUR MATRIX IS...

TYPE COMMAND
=R 5 10
YOUR MATRIX IS...

TYPE COMMAND
=R 5 9
YOUR MATRIX IS...

TYPE COMMAND
=R 7 3
YOUR MATRIX IS...

TYPE COMMAND
=R 4 5
YOUR MATRIX IS...

TYPE COMMAND
=R 7 8
YOUR MATRIX IS...

TYPE COMMAND
=C 10 2
YOUR MATRIX IS...

TYPE COMMAND
=C 10 3
YOUR MATRIX IS...

TYPE COMMAND
=C 10 4
YOUR MATRIX IS...

TYPE COMMAND
=C 10 4
YOUR MATRIX IS
```

[coded version of Table 2]

```
TYPE COMMAND
=$ 0 0

*JOURN OFF
```

```
  *Program name    :   Matrix Permutation
  *
  *Written by      :   Barbara Lynn Taghva
  *
  *Input           :   This program will read in the rows of an n by m matrix
  *                      ( for 0 =< n <= 10 and 0 =< m <=10 ). Each entry in
  *                    the matrix can be a string of up to four characters. The
  *                    entries shoud be separated by commas.
  *
  *                    Three kinds of commands can be given to the program,as
  *                    follows:
  *
  *                    (1)  R  i  j
  *                    (2)  C  i  j
  *                    (3)  $  0  0
  *
  *                    The first command will move row i to row j, while the
  *                    rest of the rows are displaced in a circular fashion.
  *
  *                    The second command will move column i to column j,
  *                    while the rest of the columns are displaced in a
  *                    circular fashion.
  *
  *                    The last command will stop the program.
  *
  *Output          :   The original matrix and all of the subsequent matrices
  *                    will be displayed.
  *
  *Errors          :   The error "illegal input string---- ignored" is the
  *                    only error message which will be displayed. This error
  *                    indicates that the user is not following the correct
  *                    format of the commands. In the case of error, the
  *                    Error Routine will issue this message and ignore
  *                    the command.
  *
  *
  CHARACTER     *1 CHAR
  CHARACTER     *4 A(10, 10)
  INTEGER       M, N, X, Y
  PRINT, 'NO.OF ROWS YOU NEED'
  READ, M
  PRINT, 'NO. OF COLUMNS YOU NEED'
  READ, N
  DO 20 I = 1, M
    WRITE (42, 10) I
10  FORMAT (1X, 'TYPE IN ROW ' , I2, 'SEPARATED BY COMMA''S')
    READ, (A(I, J), J = 1, N)
20 CONTINUE
  CALL          OUT (A, M, N)
30 PRINT, 'TYPE COMMAND'
  READ, CHAR, X, Y
  IF ((CHAR .NE. 'R') .AND. (CHAR .NE. 'C') .AND. (CHAR .NE. '$'))
 &          GO TO 40
```

```
   IF ((CHAR .EQ. 'R') .AND. ((X .GT. M) .OR. (Y .GT. M))) GO TO 40
   IF ((CHAR .EQ. 'C') .AND. ((X .GT. N) .OR. (Y .GT. N))) GO TO 40
   IF (CHAR .EQ. '$') GO TO 50
   CALL         MOVE (A, M, N, CHAR, X, Y)
   CALL         OUT (A, M, N)
   GO TO 30
   GO TO 50
40 PRINT, 'ILLEGAL INPUT STRING --- IGNORED'
   GO TO 30
50 STOP
   END
   SUBROUTINE   OUT (A, M, N)
   CHARACTER    *4 A(10, 10)
   INTEGER      M, N, I, J
   CHARACTER    *3 COL(10), ROW(10)
   DO 10 I = 1, 10
     COL(I) = 'COL'
     ROW(I) = 'ROW'
10 CONTINUE
   PRINT, 'YOUR MATRIX IS'
   PRINT, ' '
   PRINT, ' '
   WRITE (42, 20)((COL(I), I), I = 1, N)
20 FORMAT ( ' ' , 6X, 10(A3, I2, 1X))
   PRINT, ' '
   DO 40 I = 1, M
     WRITE (42, 30) ROW(I), I, (A(I, J), J = 1, N)
30 FORMAT ( ' ' , A3, I2, 1X, 10(A4, 2X))
   PRINT, ' '
40 CONTINUE
   RETURN
   END
   SUBROUTINE   MOVE (A, M, N, CHAR, X, Y)
   CHARACTER    *4 A(10, 10), TEMP
   CHARACTER    *1  CHAR
   INTEGER      M, N, X, Y, I, J, L
   IF (CHAR .EQ. 'C') GO TO 60
   IF (X .LT. Y) GO TO 30
   DO 20 K = 1, X-Y
     I = X-K
     DO 10 J = 1, N
       TEMP = A(I, J)
       A(I, J) = A(I+1, J)
       A (I+1, J) = TEMP
10   CONTINUE
20 CONTINUE
   GO TO 120
30 DO 50 I = X, Y-1
     DO 40 J = 1, N
       TEMP = A(I, J)
       A(I, J) = A(I+1, J)
       A)I+1, J) = TEMP
```

```
 40   CONTINUE
 50 CONTINUE
    GO TO 120
 60 IF (X .LT. Y) GO TO 90
    DO 80 K = 1, X-Y
      J = X-K
      DO 70 I = 1, M
        TEMP = A(I, J)
        A(I, J) = A(I, J+1)
        A(I, J+1) = TEMP
 70   CONTINUE
 80 CONTINUE
    GO TO 120
 90 DO 110 J = X, Y-1
      DO 100 I = 1, M
        TEMP = A(I, J)
        A(I, J) = A(I, J+1)
        A(I, J+1) = TEMP
100   CONTINUE
110 CONTINUE
120 CONTINUE
    RETURN
    END
```

This program is limited in certain ways. Because of the size of the customary CRT field, the attainable matrix size is too small. If one allows four character spaces per cell and two spaces between columns, as we have done in the foregoing, we have a maximum of eleven columns. Thus it was necessary to adapt even this data, which was deliberately chosen to be managable. In the run, mApn represents mąpęn-, Apn represents ąpęn-. The program can be modified in this regard by changing lines 2, 30, 32, 41, 45 and 51. For example, if a user decides to represent the formatives with arbitrary symbols -- using, say, one character per cell -- the number of possoble columns increases to 36. This would of course necessitate constant decoding between permutations. CRTs with larger capacity are however now beginning to become available.

Another limitation on this particular program is that the numbering of rows and columns remains constant, as illustrated below. Suppose the command is: move row one to row three. The change will be, e.g.,

$$\begin{array}{c|ccc} & C1 & C2 & C3 \\ \hline R1 & a & b & c \\ R2 & d & e & f \\ R3 & a & b & c \end{array} \longrightarrow \begin{array}{c|ccc} & C1 & C2 & C3 \\ \hline R1 & d & e & f \\ R2 & a & b & c \\ R3 & a & b & c \end{array}$$

rather than:

$$\begin{array}{c|ccc} & \mathrm{C1} & \mathrm{C2} & \mathrm{C3} \\ \hline \mathrm{R2} & d & e & f \\ \mathrm{R3} & a & b & c \\ \mathrm{R1} & a & b & c \end{array}$$

This could be a serious problem: the formatives in a given row or column remain constant while their arrangement changes, but if two rows or two columns happened to have the same formatives, one could lose the distinction between them during the procedure. This problem is solvable in FORTRAN, but the resulting extra storage needed would have drastically increased the cost of running the program. For this reason, row 1 and column 1 were used for labelling at the time of input.

There are two further limitations which are caused by the nature of the language FORTRAN. First, each time a user implements the program it is necessary to type in the data. That is, if I work on Taos three times I must type in the data each time. Second, it is impossible to obtain a copy of only the last, or final, matrix. In a different programming language, these problems would be solvable.

It is obviously desirable to find or develop a more suitable programming language for matrix permutation. In addition it is desirable to allow more than one permutation at a time, to use more than two parameters (perhaps), and to be able to store and retrieve a portion of a given matrix as a separate matrix. In spite of these initial difficulties, in principle Pike's technique can be rather easily computerized. No doubt there are other such techniques, perhaps some due to this same highly original linguist, which would lend themselves usefully to the computer.

FOOTNOTES

1 A, B, and C are special classes of 3rd person singular object. Abbreviations: sg = singular; pl = plural; du = dual; 1, 2, 3 = person indices; p = person; OBJ = object; SUBJ = subject.

2 These results are <u>not</u> intended as a contribution to the study of Tiwa, but as part of a demonstration. We emphasize this in the hope of avoiding a confrontation under some such heading as "What Tanoan is really like"!

REFERENCES

Comrie, Bernard, 1980. Inverse verb forms in Siberia: evidence from Chukchee, Koryak, and Kamchadal. Folia Linguistica Historica I:1.61-74.

Erikson, Barbara E., 1962. Application of matrix theory to subject-object reference in Potawatomi verbs. Mimeographed charts, University of Michigan.

-------------------, 1965. Patterns of person-number reference in Potawatomi. International Journal of American Linguistics 31. 226-236.

Hockett, Charles F., 1966. What Algonquian is really like. International Journal of American Linguistics 32:1.

Kennedy, M., and M. B. Solomon, 1965. Ten Statement FORTRAN Plus FORTRAN Four. Prentice-Hall: Englewood Cliffs, New Jersey.

Morgan, James O., 1966. A comparison of the transitive animate verb in eight Algonquian languages. Anthropological Linguistics 8:6.1-16.

Pike, Kenneth L., 1962. Dimensions of grammatical constructions. Language 38.221-323.

----------------, 1963. Theoretical implications of matrix permutation in Fore (New Guinea). Anthropological Linguistics 5.1-23.

----------------, 1975. On describing languages. In Robert Austerlitz, ed., The Scope of American Linguistics. Peter de Ridder Press: Lisse (=Papers of the First Golden Anniversary Symposium of the LSA) pp. 9-38.

----------------, and Barbara E. Erikson, 1964. Conflated field structures in Potawatomi and Arabic. International Journal of American Linguistcs 30.201-212.

Trager, George L., 1946. An outline of Taos grammar. In Linguistic Structures of Native America, Harry Hoijer, ed. (=Viking Fund Publications in Anthropology 6) pp. 184-221. Johnson Reprint 1967.