מכון הג'וינט
לגרונטולוגיה והתפתחות אדם וחברה בישראל
על שם ברוקדייל

Brookdale Institute
GERONTOLOGY AND
ADULT HUMAN DEVELOPMENT IN ISRAEL
American Joint Distribution Committee

DISCUSSION PAPER

THE CROSS-SECTION PROCESSOR

CDC VERSION 1

by

MEIR KOHN

No. 13 - 76

c.3

Jerusalem, Givat-Ram, P.O.B. 13087 .ת.ד ,גבעת-רם ,ירושלים

THE INSTITUTE

is a national ... on and education in gerontology ... nd is funded by the American ... e assistance of the Brookdale ... he State of Israel. Its resear ... nd, primarily, of an applie

The In ... recommend alternat ... nd policies. It attempt ... l other public officials ... plementation.

BR-D-13-76 c.3

Kohn, Meir

The Cross-Section Processor:
CDC Version 1

DISCUS

are pro ... idents of the social a ... nted officials who ma

Their p ... e in order to promote ... policies and program

The finc ... the author or authors ... and of other persons

The Cross-Section Processor

CDC Version 1

Meir Kohn

March 1976

Development of this version was supported in part by

The Brookdale Institute of Gerontology and Human Development in Israel
The Central Research Fund of the Hebrew University
The Center for Research in Agricultural Economics

Thanks are due to Robert E. Hall for most of the original ideas and to Michael Gurantz for assistance in preparing the program. N. A. Barr, Z. Bodie and M.D. Hurd helped to prepare earlier versions of the program at Berkeley and M.I.T.

CONTENTS

1. INTRODUCTION

The Cross-Section Processor (CSP) is designed to facilitate the analysis of large bodies of cross-section data. Data-handling and processing are performed efficiently to minimize execution time and the kinds of analysis and specification most useful in cross-section work are made available and easy to use.

The features and capabilities of CSP are as follows:

1. Assembler routines for rapid input-output of compact data files;
2. Simple free-format control statements;
3. Setup of dummy variable and linear and bilinear spline specifications by simple commands;
4. The ability to integrate user-supplied subroutines in data processing;
5. Extensive checking of data processing;
6. Linear regression and two-stage least squares;
7. Multinominal and conditional logit analysis;
8. Creation and processing of several moment matrices simultaneously.

In the following it is assumed that the user is familiar with the FORTRAN language, with the SCOPE operating system and with the necessary statistics and econometrics.

Chapter 2 describes the use of the CSP input-output package in creating and reading standard CSP data files. Chapters 3, 4 and 5 explain the use of CSP statements and user-provided subroutines to carry out the various types of analysis. Chapter 6 concentrates descriptions of the various control statements; a glance at Chapter 6 is recommended before reading Chapters 3, 4, and 5.

2. PREPARING AND READING A STANDARD DATA FILE

2.1 Introduction

The cost of repeated input-output of large data sets and their physical size (with the associated difficulty of storage and access) make their statistical analysis a practical problem. CSP contains a package of assembler-coded subroutines to create and read standard data files in which the data is packed to reduce volume. Furthermore, the I-O is asynchronous to increase processing speed. Several files may be written or read simultaneously and both fixed-length and variable records are allowed.

2.2 The structure of a standard CSP file

The standard file begins with a header identifying the file and containing information necessary for its processing. The header is followed by any number of data records (called hereafter observations). Each observation consists of two vectors--one of reals (called hereafter scalars) and one of non-negative integers (called hereafter characteristics). Either vector may be null (of length zero). The file may consist of any number of observations whose number need be neither defined nor known in advance. The file may consist of observations of fixed length or of observations of variable length which is specified separately for each observation.

2.3 How to call the I-O routines

2.3.1 To open a file for writing

CALL OPENWRT (NSC, NCH, NMAX, TITLE, BUFFER, LFN)

NSC - the length of the vector of scalars for each observation.

NCH - the length of the vector of characteristics.
Either NSC or NCH may be zero. Both should be zero if a file of variable length records is to be written.

NMAX - the largest value attained by any of the characteristics.
If NMAX is zero the largest value is assumed to be 255.
NMAX may not be greater than 1023.

TITLE - the name of an 8-word array containing the title of the file

BUFFER - the name of a 1300-word array to be used in writing the file.

LFN - the name of the SCOPE file on which the file is to be written, e.g. 7LCSPFILE. Up to ten files may be written simultaneously.

2.3.2 <u>To add an observation to a file</u>

CALL PUTBLK (SC, JCHAR, NSC, NCH, LFN)

SC - the name of an NSC-word array containing the vector of scalars for the current observation.

JCHAR - the name of an NCH-word array containing the vector of characteristics for the current observation.

NSC - the length of the vector of scalars for the current observation.

NCH - the length of the vector of characteristics for the current observation. The parameters NSC and NCH need be specified only for a file of variable length records.

LFN - the name of the file to which the observation is to be added. Necessary only when several files are being written simultaneously.

2.3.3 <u>To add an observation after checking that the largest value of any characteristic does not exceed NMAX</u>

CALL PUTCHK (SC, JCHAR, NSC, NCH, LFN)

The parameters are as explained for PUTBLK. Obviously, the checking involves extra time and should not be done unnecessarily.

2.3.4 <u>To close a file after writing the last observation</u>

CALL ENDBLK (LFN)

LFN - as for PUTBLK.

2.3.5 <u>To rewind a file after writing and closing and before reading it</u>

CALL RWNDWRT (LFN)

LFN - as for PUTBLK.

2.3.6 To open a file for reading

CALL OPENRD (NSC, NCH, IOLD, TITLE, BUFFER, LFN)

NSC - on return equals the length of the vector of scalars for each observation (zero for variable-length-record files).

NCH - on return equals the length of the vector of characteristics for each observation (zero for variable-length-record files).

IOLD - set equal to 1 if the file was written during the execution of the current program, i.e. if the call to OPENRD was preceded by a call to RWNDWRT, otherwise set equal to 0.

TITLE - the name of an 8-word array which on return will contain the title of the file being read.

BUFFER - the name of a 1300-word array to be used in reading the file. If IOLD equals 1 then this should be the same array that was used in writing the file.

LFN - the name of the SCOPE file to be read, e.g. 6LTAPE12. Up to ten files may be read simultaneously.

2.3.7 To read the next observation from the file

CALL GETBLK (SC, JCHAR, IEOF, NSC, NCH, LFN)

SC - the name of an NSC-word array which on return will contain the vector of scalars for the next observation.

JCHAR - the name of an NCH-word array which on return will contain the vector of characteristics for the next observation.

When reading a variable-length-record file SC and JCHAR should be large enough to accomodate the largest observation.

IEOF - on return equals 1 if the previous observation read was the last on the file; otherwise equals 0 on return.

NSC - on return equals the length of the vector of scalars for the next observation (if the file being read consists of variable-length records).

NCH - on return equals the length of the vector of characteristics for the next observation (if the file being read consists of variable-length records).

LFN - the name of the file from which the next observation is to be taken. Necessary only when more than one file is being read simultaneously.

2.3.8 To rewind a file in order to read it again

CALL RWNDRD (LFN)

LFN - the name of the file to be rewound. The file is positioned after the header so that the next call should be to GETBLK (not to OPENRD).

2.4 Examples of use

Example 2.1 writing a CSP file to be read later by the CSP program.

Suppose that the original data is read from TAPE3 and each observation consists of 25 variables of which the first 18 match the definition of characteristics, the others not. The maximum value of any of the characteristics is 100. The title is read from card input. The file written will be known to SCOPE as CSPFILE.

```
       {SCOPE control Cards}     (see Section 8)
7/8/9
       PROGRAM CREATE (INPUT, OUTPUT, TAPE3)
       DIMENSION BUFFER (1300), TITLE (8), SC(7), JCHAR(18)
       READ 1, TITLE
     1 FORMAT (8A10)
       CALL OPENWRT (7, 18, 100, TITLE, BUFFER, 7LCSPFILE)
    10 READ (3,2) JCHAR, SC
     2 FORMAT (16I5/2I10, 5F10.0/2F5.0)
```

```
      IF (EØF,3) 20,30
   20 CALL PUTBLK (SC, JCHAR)
      GO TO 10
   30 CALL ENDBLK
      STOP
      END
7/8/9
      DATA FILE NUMBER 1 - CREATED 1/1/76
6/7/8/9/.
```

Example 2.2 Writing two files simultaneously and then reading one of the files.

With the same original data as in Example 2.1, two files are written. The first, is as before, the second consists of characteristics only. The first file is then rewound and read to calculate, say, the average of the scalars.

```
      .
      .
      .
      CALL OPENWRT (7,18,100, TITLE1, BUFF1, 5LFILE1)
      CALL OPENWRT (0,18,100, TITLE2, BUFF2, 5LFILE2)
   10 READ (3,2) JCHAR, SC
      IF (EOF,3) 20, 30
   20 CALL PUTBLK (SC, JCHAR, 0, 0, 5LFILE1)
      CALL PUTBLK (0, JCHAR, 0, 0, 5LFILE2)
      GO TO 10
   30 CALL ENDBLK (5LFILE1)
      CALL ENDBLK (5LFILE2)
      CALL RWNDWRT (5LFILE1)
      CALL OPENRD (NSC, NCH, 1, TITLE1, BUFF1, 5LFILE1)
   40 CALL GETBLK (SC, JCHAR, IEOF)
      IF (IEOF .EO .1) GO TO 50
      .
      .
      .
      GO TO 40
      .
      .
      .
   50
      STOP
      END
```

Example 2.3 Reading a file and rereading it.

The file written in Example 2.1 is to be read and listed and then rewound and read again for some calculations.

```
      PROGRAM READER (OUTPUT)
      DIMENSION BUFFER (1300), TITLE (8), SC(7), JCHAR(18)
      CALL OPENRD (NSC, NCH, 0, TITLE, BUFFER, 7LCSPFILE)
      PRINT 1, NSC, NCH, TITLE
    1 FORMAT (2I5, 10X, 8A10)
   10 CALL GETBLK (SC, JCHAR, IEOF)
      IF (IEOF.EQ.1) GO TO 20
      PRINT 2, JCHAR, SC
    2 FORMAT (16I5/2I10, 5F10.0/2F5.0)
      GO TO 10
   20 CALL RWNDRD
   30 CALL GETBLK (SC, JCHAR, IEOF)
      IF (IEOF . EO.1) STOP
      .
      .
      .

      END
```

3. DATA PROCESSING AND MODEL SPECIFICATION

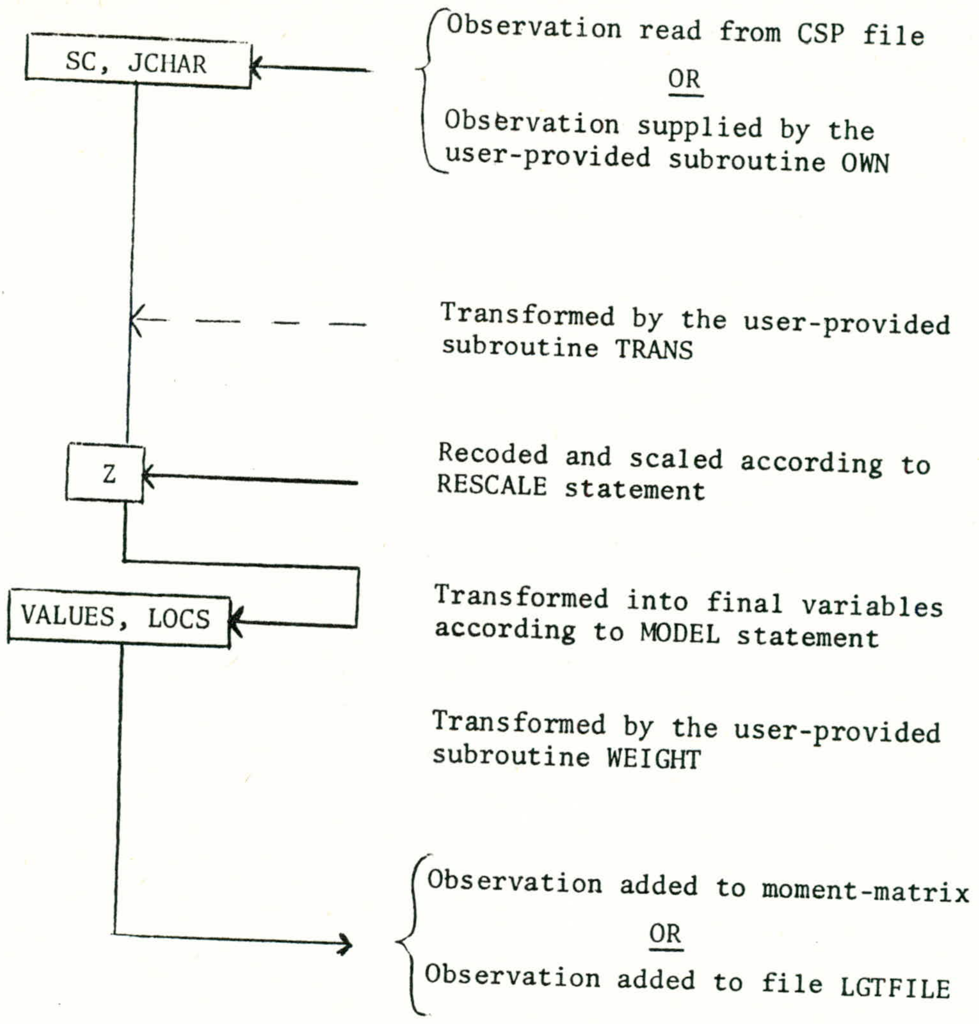
3.1 Introduction

The raw data is read and transformed by CSP under control of the control statements NUMBER, MODEL, RESCALE and PROCESS (see Section 6).

Each observation consists of a vector of scalars called SC and a vector of characteristics called JCHAR (see Section 2). The variables represented by these two vectors will be called the original variables. These variables undergo a series of transformations to become the final variables that enter the moment-matrix or file for logit analysis. For example, suppose that there is a single original variable, a characteristic, which may take on the values 1 through 6 for different observations. This variable is used as the basis of a group of six dummy variables: if the original variable takes on the value j, then the jth dummy variable is set equal to 1 and all the others to 0. The single original variable is transformed here into a group of six final variables.

The original variables in SC and JCHAR are transformed by the user-provided subroutine TRANS and recoded and scaled under control of the RESCALE statement and placed in the vector Z. In Z the characteristics are represented as real numbers (not integer as in JCHAR) and follow the scalars. The variables in Z are transformed under control of the MODEL statement to become the final variables. These may then be transformed by the user-provided subroutine WEIGHT before the observation is added to the moment-matrix or file.

3.2 Flowchart of data processing


SC, JCHAR
Observation read from CSP file
OR
Observation supplied by the user-provided subroutine OWN
Transformed by the user-provided subroutine TRANS
Z
Recoded and scaled according to RESCALE statement
VALUES, LOCS
Transformed into final variables according to MODEL statement
Transformed by the user-provided subroutine WEIGHT
Observation added to moment-matrix
OR
Observation added to file LGTFILE

3.3 Reading an observation

There are two alternative ways of reading an observation:

3.3.1 Use of subroutine OWN

If the keyword OWN appears on the PROCESS statement then the user-provided subroutine OWN will be called each time an additional observation is required.

Example 3.1   Reading observations directly from a data file using subroutine OWN.

```
      SUBROUTINE OWN (SC, JCHAR, IEØF)
      DIMENSION SC(1), JCHAR (1)
      IEOF=0
      READ(5,1) (SC(I),I=1,7),(JCHAR(I),I=1,18)
  1   FORMAT(7F5.0/(8I10))
      IF(EOF,5)10,20
 10   IEOF=1
 20   RETURN
      END
```

Note: a) The data must be read from FORTRAN logical unit 5. The SCOPE name of the data file is USERFIL or as specified on the FILES statement (see Section 7).

b) The subroutine parameters are SC and JCHAR, the names of the vector of scalars and the vector of characteristics respectively, and IEOF which should be set to 1 on the call after the last observation and to 0 otherwise.

c) See Section 8 on how to attach user-provided subroutines.

3.3.2 Reading from a standard CSP data file

If the keyword OWN does not appear then it is assumed that the data is available on a standard CSP file called CSPFILE.(See Section 2).

3.4 Tranforming the original variables

Example 3.2   Transformation using subroutine TRANS.

```
      SUBROUTINE TRANS (SC, JCHAR, IDROP)
      DIMENSION SC(1), JCHAR(1)
      IF(SC(2).LE.O.) GO TO 10
      SC(7)=ALOG(SC(2))
      RETURN
 10   IDROP=1
      RETURN
      END
```

Note: a) The subroutine TRANS will be called for each observation <u>only</u> if the keyward TRANS appears on the PROCESS statement.

b) The subroutine parameters are SC and JCHAR, the names of the vector of scalars and of the vector of characteristics respectively, and IDROP which should be set to 1 if the current observation is to be dropped.

c) See Section 9 on how to attach user-provided subroutines.

d) Linear transformations of variables for regression analysis (e.g. SC(6) = SC(1) + 0.5 * SC(2)) are more efficiently done by using the GENERATE statement (See Section 4,3).

3.5 Recoding and scaling with the RESCALE statement

Recoding and scaling are performed under control of the RESCALE statement. For example

```
RESCALE CH7 = (0, 1, 1, 2, 2) $
```

would cause the seventh characteristic to be recoded so that 0 remains 0, 1 and 2 become 1, 3 and 4 become 2.

Note: a) The first number in parenthesis is the new value for <u>zero</u> not for 1.

b) If <u>any</u> characteristic is recoded then the value zero will be considered a missing value for <u>all</u> characteristics. Hence, if any characteristic is zero for the observation after recoding, then the observation is dropped. If no characteristic is recoded then zero will be considered a legitimate value for all characteristics and will not cause the observation to be dropped.

c) If a characteristic is recoded, a value for that characteristic outside of the range covered by the recoding (e.g. greater than 4 for the above example) will cause the observation to be dropped.

It is sometimes necessary to scale a scalar variable (see, in particular, Section 3 on linear splines). This too is performed under control of the RESCALE statement:

```
RESCALE SC4 =(1., 10., 100.) $
```

would cause the fourth scalar to be scaled so that 1 become 1, 10 become 2 and 100 become 3.

Note: a) Intermediate values are scaled accordingly e.g. 5 becomes 1.55 and 55 becomes 2.5.

b) Values outside those indicated (less than 1 or greater than 100 for the above example) cause the observation to be dropped.

3.6 Creation of final variables under control of the MODEL statement

3.6.1 Specifying the number of original variables

The first item on the model statement is of the form

```
MODEL/i, j / ... $
```

where i is the number of scalars among the original variables and j is the number of characteristics (this information is necessary for the generation of the final variables).

The value of i should not be less than the value of NSC specified for the CSP file to be read although it may be greater--for instance if new variables are created by TRANS. Likewise for j and NCH.

3.6.2 Picking a subset of the original variables

Suppose that the original variables consist of fifty scalars and a hundred characteristics for each observation and that we are interested in only a subset of these for the purpose of a specific regression. Then

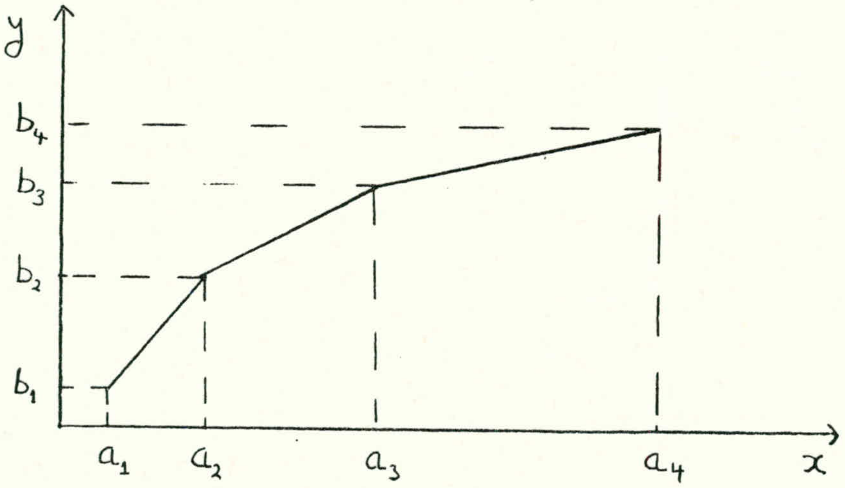
```
MODEL / 50, 100/ SC1, SC2, SC3, SC4, SC10, CH81 $
```

will produce a vector of final variables of length 6 such that

1st	final	variable	is	the 1st	scalar
2nd	"	"	"	" 2nd	"
3rd	"	"	"	" 3rd	"
4th	"	"	"	" 4th	"
5th	"	"	"	"10th	"
6th	"	"	"	"81st	characteristic.

3.6.3 Linear and bilinear splines

A linear or bilinear in variables functional form is frequently inappropriate or over-restrictive in cross-section work. Of the various alternatives (which include step functions and polynomials), there has been growing interest in the piecewise linear or bilinear function,also known as the linear or bilinear spline. For a description see D.J. Poirier, "On the Use of Bilinear Splines in Economics," Journal of Econometrics 3 (1975), 23- 34. For example, suppose we wanted to describe the graph of y and x as a broken line:



The points a_1, a_2, a_3, a_4 on the x-axis where the line is broken are called knots. Any given value of x can be represented as a convex linear conbination of the knot values:

$$x = x_1 a_1 + x_2 a_2 + x_3 a_3 + x_4 a_4,$$

where each of the weight variables x_1, x_2, x_3, x_4 takes on a value between zero and one. For example if $x = (a_2 + a_3)/2$ then $x_1=0$, $x_2=0.5$, $x_3=0.5$, $x_4=0$. For each knot there is an associated value of y--call it the knot coefficient. The value of y associated with any value of x is a linear conbination of the knot coefficients where the weights are precisely the variables x_1, x_2, x_3, x_4. For example for $x = (a_2 + a_3)/2$, $y = (b_2 + b_3)/2$ or $y = b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$.

If y is SC1 and x is SC2 then the knot coefficients could be estimated using CSP in the following way:

Example 3.3 Linear spline

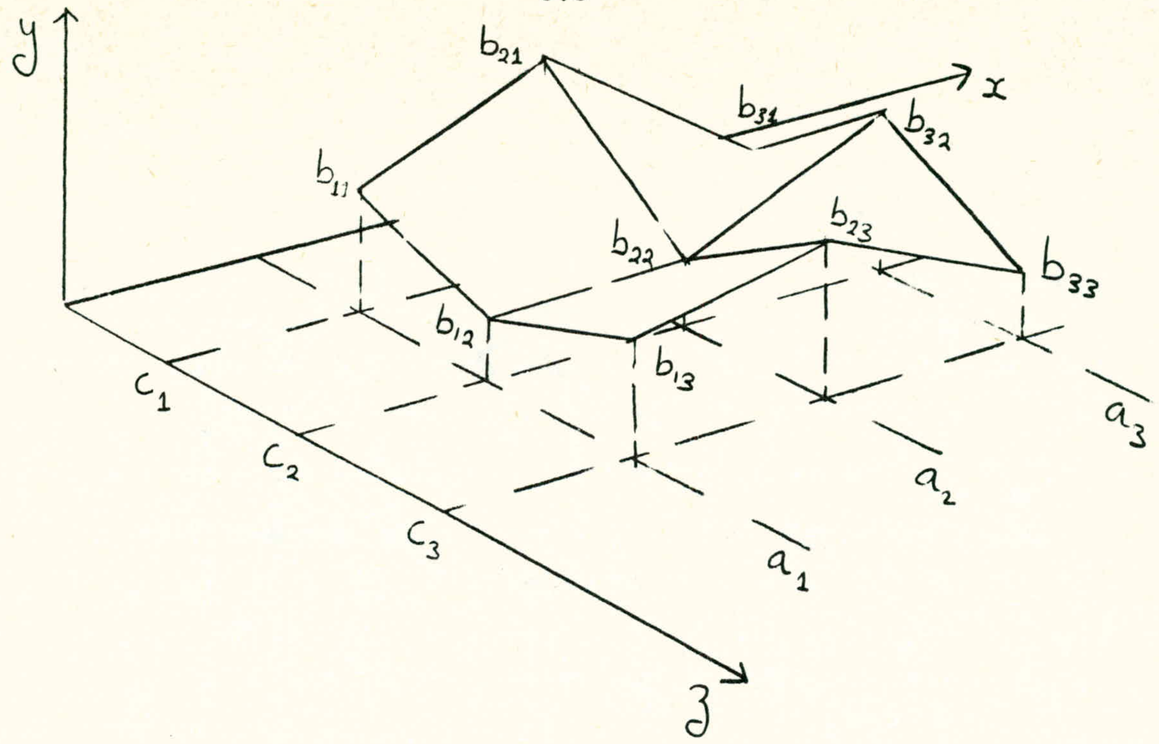
```
  .   .   .
MODEL / 2, 0/ SC1, (SC2(4) ) $
RESCALE SC2 = (a1, a2, a3, a4) $
  .   .   .
REG DEP = 1, INDEP = 2 3 4 5 $
  .   .   .
```

Note: a) The appearance of (SC2(4)) on the MODEL statement causes the generation of the four weight variables from the original variable.

b) The knot values are defined on a RESCALE statement (see Section 3.5).

c) The knot coefficients are the coefficients of the regression of final variable 1 (i.e.y) on the final variables 2 through 5.

The bilinear spline is a generalization of the linear spline to the case of two independent variables with interaction:



A mesh is defined on x and z with knots at (a_1, c_1) etc. Each value of (x, z) can be represented as a convex linear combination of the four adjacent knots. If we call the weight given to the knot (a_i, c_j) w_{ij} then the point $((a_1 + a_2)/2, (c_1 + c_2)/2)$ could be represented by

$$(x, z) = \Sigma\, w_{ij}\, (a_i, c_j)$$

where $w_{11} = w_{21} = w_{22} = 0.25$, $w_{ij} = 0$ otherwise. The y-value for the point (x, z) would be just

$$y = \Sigma\ w_{ij} b_{ij}$$

where b_{ij} is the knot coefficient associated with the knot (a_i, c_j). If SC1 is y, SC2 is x and SC3 is z then the knot coefficients could be estimated as follows:

Example 3.4 Bilinear spline

```
  .   .   .
MODEL/3, 0/SC1,(SC2(3), SC3(3) ) $
```
RESCALE SC2 = (a_1, a_2, a_3) SC3 = (C_1, C_2, C_3) \$
```
  .   .   .
REG DEP= 1 INDEP = 2 THRU 10 $
```

Note: a) The appearance of (SC2(3),SC3(3)) causes the generation of the nine weight variables from the two original variables.

b) The knot values are defined on a RESCALE statement (see Section 3.5).

c) The knot coefficients are the regression coefficients of final variable 1 on final variables 2 through 10.

Example 3.5 Bilinear spline with no interaction

```
  .   .   .
MODEL /3, 0/ SC1 (SC2(3) ) (SC3(3) ) $
```
RESCALE SC2 = (a_1, a_2, a_3) SC3 = (C_1, C_2, C_3) \$
```
  .   .   .
REG DEP = 1 INDEP = 2 THRU 7 $
  .   .   .
```

3.6.4 Dummy variables

Cross-section analysis typically involves heavy use of dummy variables (step functions, ANOVA). CSP provides for the creation of a set of final dummy variables from a given original characteristic (categorized or qualitative) variable. Furthermore,it provides for the creation of sets of dummy variables from the interaction of any number of characteristics.

Example 3.6 Simple set of dummy variables.

```
  .   .   .
MODEL /5, 10/ SC1 SC2 CH10/DUMMY/ ( CH2(4) ) $
  .   .   .
```

Note: a) Dummy variables appear <u>after</u> the other variables and are separated from them by the keyword /DUMMY/.

b) The appearance of (CH2(4)) on the MODEL statement causes the generation of a set of 4 dummy variables from the second characteristic of the original variables. If the characteristic takes on the value j = 1, 2, 3 or 4 for a given observation, then the jth dummy variable will be 1 and the others 0. For values of j less then 1 or greater than 4 the observation will be dropped.

c) The final variables generated by the above MODEL statement will be

1	SC1
2	SC2
3	CH10
4	4 Dummy variables generated from CH2.
5	
6	
7	

<u>Example 3.7</u> Dummy variables with interaction

```
  .   .   .
MODEL/0 , 10//DUMMY/ ( CH1(2), CH2(3), CH3(2) ) $
  .   .   .
```

Note: a) as for Example 3.6.

b) The appearance of (CH1 (2), CH2(3), CH3(2)) causes the generation of a set of 12 dummy variables from characteristics 1, 2 and 3. If the first characteristic takes on the value i = 1, or 2, the second j = 1, 2 or 3 and the third k = 1 or 2 then the ((k-1)*6 + (j-1) * 2 + i)th dummy will be 1 and the others 0. The vector is arranged as would be a 2 x 3 x 2 FORTRAN array, with the first index changing most rapidly. On the use of the NAMES statement to obtain a key to the final variables see Section 3.11.

3.6.5 The constant

Example 3.8 Constant

```
  .   .   .
MODEL /5, 0 / SC1 SC2 / DUMMY / CONSTANT $
  .   .   .
```

Note: a) The appearance of the keyword CONSTANT will cause the generation of a final variable which always takes on the value 1.

b) The keyword CONSTANT must follow the keyward /DUMMY/.

c) The keyword CONSTANT must be the last item of a MODEL statement.

3.6.6 Switching variables

Example 3.9 Switching variables

```
  .   .   .
MODEL/ 5, 10/ SC3* (CH2(3) ) $
  .   .   .
```

Note: a) The appearance of SC3*(CH2(3)) will cause the generation of a set of 3 final variables. If characteristic two takes the value j = 1, 2 or 3 for a given observation then the jth final variable will be set equal to the value of the third scalar for that observation. The other final variables will be set to 0. The result is equivalent to multiplying a set of dummies generated from CH2 by SC3.

b) The specification of the type SC3*(SC2(3)) is recognized and results in the multiplication of the set of three weight variables generated from SC2 by SC3.

c) Specifications of these types should precede the /DUMMY/ keyword.

d) One degree of interaction within the parentheses (i.e. two variables) is allowed.

3.7 Storage of the final variables in VALUES and LOCS

In general, a large number of the final variables will be zero and several more (dummy variables) will be equal to one. Therefore, to save space and processing, the vector of final variables is not kept in its entirety. Rather, two vectors, called VALUES and LOCS are kept with the non-zero values of the non-dummy variables in VALUES and their indices in the first part of LOCS. There are ILOC such values. The remainder of LOCS contains the indices of those dummy variables equal to one. The total length of LOCS is JLOC.

Example 3.10 Generation of VALUES and LOCS.

Suppose that the MODEL statement is

```
MODEL/ 5, 10/ SC1 SC2 (SC3(4) )/DUMMY/(CH1(3) ) CONSTANT $
```

For a given observation the final variables generated might be

2.3	from SC1
10.5	from SC2
0.	the weights created from SC3
0.32	
0.68	
0.	
0.	the dummies created from CH1
1.	
0.	
1.	the constant.

VALUES would be 2.3 10.5 0.32 0.68 ;

LOCS would be 1 2 4 5 8 10 ;

ILOC would be 4 and JLOC would be 6.

3.8 Transformation using subroutine WEIGHT

The final variables may be transformed by the user-provided subroutine WEIGHT.

Example 3.11 Weighting observations with subroutine WEIGHT

```
      SUBROUTINE WEIGHT (Z, VALUES, LOCS, ILOC, JLOC, IDROP)
      DIMENSION Z(1), VALUES (1), LOCS (1)
      WGHT = Z(7)
      IF (WGHT. LE. 0.) GO TO 30
      DO 10 I = 1,ILOC
   10 VALUES (I) = WGHT * VALUES (I)
      I1 = ILOC + 1
      DO 20  I = I1, JLOC
   20 VALUES (I) = WGHT
      ILOC = JLOC
      RETURN
   30 IDROP = 1
      RETURN
      END
```

Note: a) The subroutine parameters are Z, VALUES, LOCS, ILOC, JLOC and IDROP. The vector Z consists of the scalars (elements 1 through NSC) and the characteristics (elements NSC + 1 through NSC + NCH), all in REAL mode, after transformation (see Sections 3.4 and 3.5). The observation is dropped if IDROP is set equal to 1. The other parameters are explained in Section 3.7).

b) In this example the implicit "1"'s represented by elements ILOC + 1 through JLOC of LOCS must be made explicit (be represented in VALUES) in order to be multiplied by the weight.

c) In this example the 7th variable in Z is used to weight the observation.

d) Subroutine WEIGHT will be called for each observation only if the keyword WEIGHT appears on the PROCESS statement.

e) See Section 8 on how to attach user-provided subroutines.

3.9 Use of the final variables

3.9.1 Addition to the moment-matrix for regression analysis

This is the default use of the vector of final variables (if no other use is specified). If there are n final variables then a moment-matrix will be created consisting of the n x n cross-products of the n variables. Since the matrix is symmetric only n(n + 1)/2 elements are actually calculated.

Upon completion of data processing the matrix is written on the file MATRIX. The file consists of two binary logical records: the first, of twelve words consists of the matrix sequence number (see 3.9.2), the order of the matrix, the number of observations, the number of words taken up by the matrix (the second records),and an 8-word title; the second consists of a vector containing the upper half of the matrix (i.e. a_{11}, a_{12}, a_{22}, a_{13}, a_{23}, a_{33}, a_{14} etc).

If the keyword PUNCH appears on the PROCESS statement then the matrix will also be written on the file PUNCH in coded format suitable for reading through the LOAD statement (see Section 4,1,2).

3.9.2 The simultaneous creation of several matrices

In cross-section work it is often useful (and economical) to create a number of moment-matrices on a single run through the data. For instance we might wish to create three matrices--one for all observations, one for males only and one for females only. (CSP is able to create up to twenty matrices at once. The program is able to use only one MODEL at a time so that the various matrices will all contain the same variables but for different subsamples of the data as specified by the user.

If n matrices are to be created(for n greater than 1) then NMAT = n should appear on the PROCESS statement.

The addition of observations to the various matrices is controlled using the vector LMAT. If LMAT(j) = 1 then the observation is added to matrix j; if LMAT (j) = 0 then the observation is not added to matrix j. The values of LMAT for a given observation may be set by any of the user-provided subroutines OWN, TRANS or WEIGHT (See Sections 3.3.1, 3.4 and 3.8 respectively).

<u>Example 3.12</u> Simultaneous creation of matrices

```
  .   .   .

PROCESS TRANS NMAT = 3 $

  .   .   .

      SUBROUTINE TRANS (SC, JCHAR, IDROP)
      DIMENSION SC(1), JCHAR(1)
      COMMON/UCOM/LMAT (20)
      DO 10 I = 1,3
   10 LMAT (I) = 0
      J = JCHAR(3)
      LMAT(J) = 1
      RETURN
      END
```

Note: a) The statement
COMMON/UCOM/LMAT(20)
must appear.

b) In this case each observation is added to one of three matrices according to the value of characteristic three.

c) The values of LMAT may also be set by subroutines OWN on WEIGHT.

When n matrices have been created all n will be written on to file MATRIX separated by ends-of-file. The keyword PUNCH on the PROCESS statement will result in all of the matrices being written on to PUNCH (see Section 3.9.1). When carrying out regression analysis it will be necessary to specify from which matrix to take the variables (see Section 4, 1, 4).

3.9.3 Creating a file for logit analysis

If the keyword LGTFILE appears on the PROCESS statement then the final variables for each observation will be saved on a standard CSP data file named LGTFILE. The file contains variable-length records. The first record has a null scalar vector and a characteristic vector of length 3. The first element is nonzero if there is a repetition factor, the second is equal to the number of final variables, and the third is equal to the number of categories. Following this is a variable-length record for each observation consisting of a scalar vector of length ILOC (the vector VALUES) or of length ILOC + 1 if there is a repetition factor--the ILOC+1st element being the value of the repetition factor--and a characteristic vector of length JLOC + 1 consisting of the vector LOCS (length JLOC) followed by the category of the current observation. The appearance of the keyword LGTFILE obliges the setting of the parameters DEP and NCAT where DEP is the original variable whose value is the category of the current observation and NCAT is the number of possible categories for the problem, i.e. the PROCESS statement must include at least the following

```
PROCESS LGTFILE DEP = CH3 NCAT = 4 $
```

If a repetition factor is used then the parameter REP should be set equal to the original variable whose value is the repetition factor for the current observation, e.g. REP = SC7.

No moment matrix is created.

3.9.4 Preventing the creation of a moment-matrix

The appearance of the keyword NOMAT on the PROCESS statement prevents creation of a moment-matrix.

3.10 Subroutines BEFORE and AFTER and the keyword READ

The appearance of the keyword BEFORE on the PROCESS statement causes the user-provided subroutine BEFORE to be called before the first observation is read. Likewise, the appearance of AFTER causes AFTER to be called after the last observation is read. See Section 8 on how to attach user-provided subroutines to CSP.

The appearance of the keyword READ causes the prevention of all processing other than reading the observation and calling TRANS (if this has been requested).

3.11 How to check data processing

The process of transforming and selecting the data can be quite complex and it is essential to keep track of what is going on. CSP has a number of features which help in this task.

3.11.1 Listing the various vectors of variables

A number of keywords on the PROCESS statement control the listing of the various vectors:

SC	- causes the vector of scalar original variables to be printed;
CH	- causes the vector of characteristic original variables to be printed
Z	- causes the vector Z (see Section 3.1) to be printed;
VAL	- causes the vector of final variables, as represented by the vectors VALUES and LOCS (see Section 3.7) to be printed.

The number of cases for which the indicated vectors are to be printed is determined by the parameter NPRINT. Its default value, if it does not appear on the PROCESS statement, is 100.

Example 3.13 Listing original and final variables for the first 50 observations

```
  .   .   .

PROCESS  SC  CH  VAL  NPRINT = 50 $

  .   .   .
```

3.11.2 Summary of observations dropped

On completion of data processing, CSP prints a breakdown of dropped observations. This indicates how many were dropped by each of TRANS, CODE (the subroutine performing coding and scaling), VALDO (the subroutine which creates the final variables according to the MODEL statement), and WEIGHT. For those observations dropped by CODE and VALDO there is a further breakdown of why they were dropped.

3.11.3 Use of subroutine COUNT

The user-provided subroutines can keep counts of what was done to which observation and why by calling subroutine COUNT.

```
CALL COUNT (J)
```

results in the Jth count register being incremented by 1. J is a positive integer less than 100. All count registers are set to zero before the first observation and the status of non-zero registers is reported at the end of data processing.

3.11.4 Limiting the number of observations processed

Normally observations are read from the input file until an end of file is encountered. For purposes of debugging it is useful to be able to halt processing after a given number of observations. This can be done by writing NMAX = n on the process statement. This will cause a maximum of n observations to be processed.

3.12 Variable names and titles

3.12.1 The NAMES statement

The NAMES statement enables the user to assign a name to a given variable.

The original variables are given the names SCi or CHj (for scalars and characteristics respectively). The appearance of SCi = <u>name</u> on the NAMES statement will cause SCi to be replaced by <u>name</u> in all output from the program. The name may be of up to ten characters, the first of which must be alphabetic (no special characters are allowed). The appearance of the NAMES statement will cause the creation of a vector of names for the final variables. These names are manufactered from the names of the original variables according to the specification of the MODEL statement. This vector is written on to the file NAMES as a single binary logical record: the first word gives the length of the vector (five times the number of final variables) and this is followed by the vector of names, each of five words. If there is no NAMES statement and no file NAMES is attached, then the final variables will be referred to by their numbers alone. A specific name may be assigned to final variable i by writing VRi = <u>name</u> on the NAMES statement.

The presence of the keyword PRINT on the NAMES statement will cause the vector of final variable names to be listed. <u>This is a handy way of checking on the creation of final variables from original variables.</u>

<u>Example 3.14</u> The NAMES statement.

```
  .  .  .
      MODEL / 5, 10/ SC2 (SC3(3) )/DUMMY/(CH2(2) ) CONSTANT $
  .  .  .
      NAMES SC3 = EDUCATION VR5 = MALE VR6 = FEMALE PRINT $
  .  .  .
```

The following output would result:

```
1   SC2
2   EDUCATION (1)
3   EDUCATION (2)
4   EDUCATION (3)
5   MALE
6   FEMALE
7   CONSTANT.
```

3.12.2 The TITLE statement

The statement TITLE $ is assumed to precede a card image containing a title which is to be printed in the subsequent output. More then one title may be used in a run.

Example 3.15 Use of the TITLE statement.

```
. . .
TITLE $
        MATRIX OF SURVEY DATA
PROCESS  . . . $
TITLE $
        REGRESSION OF INCOME ON EDUCATION
REG . . .   $
TITLE $
        REGRESSION OF BRUSHING ON EDUCATION
REG . . .$
```

Note: a) The header of the matrix written on to file MATRIX (see Section 3.9.1) will contain the title
MATRIX OF SURVEY DATA

b) The output of the first regression will carry the title
REGRESSION OF INCOME ON EDUCATION
and the second the title
REGRESSION OF BRUSHING ON EDUCATION.

3.13 The order of statements for data processing

The MODEL statement must appear before the PROCESS statement and before the RESCALE and NAMES statements if either appears.

4. REGRESSION ANALYSIS

CSP performs linear regression analysis by ordinary or two-stage least squares. The analysis is controlled principally through the REG statement.Any number of regressions may be performed on the same run.

4.1 Providing a moment matrix

The input to regression analysis is not the original data but a moment-matrix including the cross products of all the final variables to enter the equation. The matrix may be made available in a number of ways.

4.1.1 Attaching the file MATRIX

Upon completion of data processing (see Section 3.9.1) the moment matrix is written on the file MATRIX. This file may be saved (on tape or as a permanent file) and then made available on a subsequent run (see Section 7).

4.1.2 Reading a matrix with LOAD

If the matrix has been punched onto cards (or card images) by CSP (see Section 3.9.1) or by another program it may be read and placed on the MATRIX file by using the LOAD statement. If the matrix was punched by CSP then the first card will be a header carrying a title and the matrix parameters. In this case the statement

```
LOAD HEADER $
```

should be followed immediately by the header and then the matrix. If no header card is present the statement should take the form

```
    LOAD IMAT = imat NOV = nov  NOBS = nobs $
```

where imat is the sequence number of the matrix, nov is the order of the matrix and nobs is the number of observations. The first card of the matrix should follow immediately.

Note that more than one matrix may be read and that the matrices will be written on to MATRIX in the same order as they are read (not according to IMAT if this differs from the order read--which it should not).

4.1.3 The matrix was created on the current run

If the matrix was created on the present run (i.e. one of the control statements was a PROCESS statement) then it will be available for further processing on the file MATRIX and no additional action is required of the user.

4.1.4 Choosing one of several matrices

If several matrices are present on the file MATRIX then the user must indicate which matrix is to be processed. In the absence of explicit instructions the first matrix will always be used.

Example 4.1 Processing several matrices. Five matrices are present on file MATRIX and it is desired to run a regression on the first, third and fifth and also to print the means, etc., of the third matrix:

```
  .   .   .
  REG DEP = 1 INDEP = 2 THRU 10 $
  MATRIX 3 MEANVAR $
  REG DEP = 1 INDEP = 2 THRU 10 $
  MATRIX 5 $
  REG DEP = 1 INDEP = 2 THRU 10 $
  .   .   .
```

Note: a) That nothing need be done to acess the first matrix since this is automatically available.

b) For other matrices the REG statement must be preceded by a MATRIX statement specifying which matrix and indicating what information is to be printed.

4.2 Printing information about the moment-matrix

This is controlled through the MATRIX statement.The following options are available.

4.2.1 Printing the whole matrix

If the keyword PRINT appears on the MATRIX statement the whole matrix will be printed, or rather its lower half.

4.2.2 Means, variances and standard errors

The appearance of the keyword MEANVAR causes the mean, variance and standard error to be printed for each variable in the matrix.

4.2.3 Correlations

The appearance of the keyword CORR causes the printing of the correlation matrix of all variables in the matrix.

4.2.4 Specifying which matrix

If file MATRIX contains more than one matrix, then the MATRIX statement should indicate which matrix is required (the default is the first).

4.3 Linear transformation on the matrix

Linear transformations of the final variables, e.g. creating a new variable which is the sum of two existing variables, can be performed most economically by linear operations on the moment-matrix.

Example 4.2 Use of the GENERATE statement

```
    . . .
    GENERATE 10 = 5 + 2 * 3 - 1.5 * 7 $
    . . .
```

Note: a) The variable number to the left of the equals sign is the number of the new variable created. If it is the number of an existing variable then it replaces the existing variable.

b) The right hand side may consist of any linear combination of existing variables which may be premultiplied by constant.

c) The file MATRIX is normally left unchanged by the transformation which only affects the matrix currently in core. If it is desired to save the new transformed matrix, this may be done with the

```
MATRIX SAVE $
```

statement which causes the matrix to be saved on file SAVE (see Section

4.4 Ordinary least squares

Ordinary-least-squares regression is performed under the control of the REG statement:

```
    REG DEP = 5 INDEP = 6, 2, 9, 11 $
```

The 5th final variable is the dependent variable and the 6th, 2nd, 9th and 11th final variables are the independent variables. Several dependent variables may be run against the same group of dependent variables using a single REG statement as follows:

```
    REG DEP = 4, 5, 3  INDEP = 6, 2, 9, 11 $
```

This is useful, for instance, in the estimation of linear probability models.

Lists of variables can be uritten more economically by using the keyword THRU:

```
REG DEP = 21   INDEP = 1 THRU 20, 22 THRU 30 $
```

The independent variables here will be 1 through 20 and 22 through 30.

4.5 Two-stage least squares

Two-stage least squares regression is performed under control of the REG statement:

```
REG DEP = 5  ENDOG = 6  EXOG = 7  INST = 8, 9 $
```

The 5th final variable is the dependent variable; the 6th final variable is an endogenous independent variable; the 7th final variable is an exogenous independent variable; the 8th and 9th final variables are to be used as instrumental variables (together with those defined as exogenous independent variables). Only a single dependent variable is allowed on each REG statement for two stage least squares.

The number of instrumental variables must be at least as great as the number of endogenous independent variables (the order condition for identification).

4.6 Multicollinearity and nomalization

CSP automatically excludes the last of a group of mutually linearly dependent explanatory variables. In the printout of the regression results the coefficient, standard error and t-ratio are all zero for such a variable. The tolerance level (roughly speaking the percentage of the variation not dependent on preceding variables) for linear dependence is fixed at 0.00005. It may be set to any value by the user through the TOL parameter, e.g.

```
REG ... TOL = .1E-6 $
```

The user can choose which of a group of linearly dependent variables is to be excluded by use of the NORM keyword.

Example 4.3 Use of the NORM parameter

```
    . . .
    MODEL/10, 20/SC1/DUMMY/(CH2(5) ) CONSTANT $
    . . .
    REG DEP = 1  INDEP = 7, 2 THRU 6 NORM = 4 $
    . . .
```

Note: a) There is an obvious linear dependence between the group of dummy variables 2 through 6 and the constant.

b) Variable 4 will be excluded from the regression. That is, it will appear in the output with coefficient, standard error and t-ratio zero. If NORM = 4 did not appear, then CSP would automatically exclude the last linearly dependent variable i.e. variable 6.

c) Of course, variable 4 could also be excluded by simply not including it in the list of independent variables. However, in this case it would be absent from the output and this is often inconvenient

d) A list of variables may follow NORM = and not only a single variable.

e) For two stage least squares only variables in the EXOG list may be excluded using NORM.

4.7 Controlling output

The output of the regression processor normally consists of a list of coefficients, standard errors and t-ratios for the independent variables. Printed in addition are the sum of squares of the dependent variable, the sum of squared residuals, the standard error of the regression, the number of observations, the number of variables and the number of degrees of freedom.

If the first independent variable is the constant then R^2 is also printed. Since all the numerical errors are concentrated in the calculation of R^2 the value printed is not very reliable--particularly for large samples.

Additional output may be obtained by using the following keywords:

MATRIX - the moment-matrix of variables included in the regression is printed.

INVERSE - the inverse of the moment-matrix of independent variables is printed.

PUNCH - the regression coefficients are written on to file PUNCH. The first card image is a header consisting of the first 70 characters of the title, the sequence number of the regression and the number of the dependent variable. Then follow the coefficients, five to a card.

EXP - the antilog of the coefficient is printed together with the coefficient itself.

4.8 Transforming regression results with subroutine ENDREG

The user-provided subroutine ENDREG may be used for further processing of the regression results. If the keyword ENDREG appears on the REG statement then the subroutine will be called after the regression results have been printed.

Example 4.4 Use of ENDREG to calculate F-statistic for a linear hypothesis

```
      . . .
      REG  . . .  ENDREG $
      REG  . . .  ENDREG $
      . . .
      SUBROUTINE ENDREG (NV, AINV, B, SE, NAMES, SSQ, DNAME)
      DIMENSION AINV (NV, NV), B(1), SE(1), NAMES(5,1), DNAME(5)
      COMMON/REGCOM/SSR, NOB, SEREG, NINC, DF, IREG, TOL,
     *ILHAND, TITLE(8), YMEAN, RSQ
      IF (IREG.EO.2) GO TO 10
      DF1 = DF $ SSR1 = SSR
      RETURN
   10 Q = DF1 - DF
      F = (SSR1 - SSR) * DF/(SSR * Q)
      PRINT 1, F, Q, DF
    1 FORMAT(*0F-STATISTIC IS*, G15.6,* WITH*, 2F4.0,*DF*)
      RETURN
      END
```

Note: a) The first regression embodies the linear hypothesis.

b) Information is made available to ENDREG as subroutine parameters and via the COMMON block REGCOM. The following are available:

NV	-	the number of independent variables specified for the regression (i.e. the number following INDEP or both ENDOG and EXOG).
AINV	-	the NV by NV generalized inverse of the moment-matrix of independent variables. Rows and columns for excluded variables are zero.
B	-	The NV -vector of coefficients
SE	-	the NV -vector of standard errors
NAMES	-	the 5 by NV matrix of names of the independent variables. Each name occupies five words.
DNAME	-	the 5-vector name of the dependent variable
SSQ	-	the sum of squares of the dependent variable
SSR	-	the sum of squared residuals
NOB	-	the number of observations
SEREG	-	the standard error of the regression
NINC	-	the number of included variables
DF	-	the number of degrees of freedom (REAL)
IREG	-	the sequence number of the REG statement being executed
TOL	-	the toterance level for linear dependence
ILHAND	-	the sequence number of the current dependent variable among those appearing on the current REG statement.
TITLE	-	an 8-vector title.

c) See Section on how to attach a user-provided subroutine to CSP.

YMEAN	-	the mean of the dependent variable.
RSQ	-	R^2

4.9 Use of CSP to calculate predicted values or residuals

Example 4.5 Calculation of residuals.

Suppose that a regression has been calculated and the coefficients written on to file TAPE20:

```
     FILES PUNCH = TAPE20 $
     MODEL/ 5, 10/SC1, SC2,SC3,(SC4(11), SC5/DUMMY/(CH5(7) CONSTANT $
     PROCESS $
     REG DEP = 15  INDEP = 23, 1 THRU 14, 16 THRU 22 PUNCH $
     . . .
```

Now the coefficients are to be read and used to calculate the residuals which are to be written on to file TAPE15. The following subprograms need to be attached (See Section 8):

```
     OVERLAY (CSP, 0, 0)
     PROGRAM FIRST         ( TAPE20, TAPE15)
     COMMON/CNTRL/NSPACE, X(162)
     COMMON SPACE (6000)
     NSPACE = 6000
     CALL OVERLAY (3HCSP, 2, 0)
     CALL OVERLAY (3HCSP, 1, 0)
     STOP
     END
     SUBROUTINE BEFORE
     COMMON/BCOM/COEFF(23)
     READ (20,1) COEFF(23),(COEFF(I),I=1,14)(COEFF(I)I=16,22)
   1 FORMAT(/(5G16.8))
     COEFF(15) = 0 .
     RETURN
     END
     SUBROUTINE WEIGHT (Z,VALUES, LOCS, ILOC, JLOC, IDROP)
     DIMENSION Z(1), VALUES (1), LOCS(1)
     COMMON/BCOM/COEFF(23)
```

```
      CALL VMULT (COEFF, VALUES, LOCS, ILOC, JLOC, YHAT)
      Y = VALUES (ILOC)
      RES = Y - YHAT
      WRITE (15) RES, Y, YHAT
      RETURN
      END
```

And the following CSP control statements would be used:

```
      MODEL/5, 10/SC1, SC2, SC3,(SC4(11)), SC5/DUMMY/(CH5(7)) CONSTANT $
      PROCESS BEFORE WEIGHT NOMAT $
      STOP $
```

Note: a) Files TAPE20, TAPE15 must be declared on the PROGRAM statement.

b) The coefficients are read by subroutine BEFORE before data processing begins. The order of the coefficients is that of the variables in the INDEP list and not the sequential order; they are read into COEFF accordingly. The "coefficient" of the dependent variable is zero (as would be those of any other variables not included as independent variables in the regression).

c) The CSP subroutine VMULT is used to get the dot product of the vector COEFF and the vector of final variables (as represented by VALUES and LOCS). The result is returned as the sixth subroutine parameter.

d) The same MODEL statement is used as in the original run.

e) The keywords BEFORE and WEIGHT appear to ensure that these subroutines are called (See Sections 3.8 and 3.10) and NOMAT appers to prevent creation of a moment-matrix (Section 3.9.4).

5. LOGIT ANALYSIS

CSP performs multinomial and conditional logit analysis by a Newton-Raphson maximum likelihood procedure. The analysis is controlled through the LOGIT statement.

5.1 A description of the multinomial logit model

Let there be T observations on the occurrence of one of M mutually exclusive and exhaustive events. The probability, P_{it}, that the ith event $(1 \leq i \leq M)$ occurs for observation t $(1 \leq t \leq T)$ is a function of K explanatory variables of the form

$$P_{it} = \exp(X_t'\beta_i) / \sum_{j=1}^{M} \exp(X_t'\beta_j),$$

where X_t is the K-vector of values of the K explanatory variables for the tth observation and the vectors β_i $(i = 1, \ldots, M)$ are vectors of coefficients.

CSP estimates the coefficient vectors by maximizing the sample likelihood function using an iterative Newton-Raphson procedure.

From the definition of P_{it} above it should be clear that multiplying all of the coefficients by some scalar would not affect the probabilities. Hence, some sort of normalization is needed. The normalization used is to set the coefficient vector β_M equal to zero. Thus, the result will consist of M-1 vectors of coefficients.

5.2 A description of the conditional logit model

Let there be T observations on the choice of one and only one of $M_t (1 \leq t \leq T)$ alternatives. The probability, P_{it}, that the ith alternative $(1 \leq i \leq M_t)$ is chosen from among the M_t alternatives for observation t is a function of the values of K explanatory variables for all of the alternatives:

$$P_{it} = \exp(X'_{it}\beta) / \sum_{j=1}^{M_t} \exp(X'_{jt}\beta) ,$$

where X_{it} is the K-vector of values of the K explanatory variables for the ith alternative $(1 \leq i \leq M_t)$ for the tth observation and β is a K-vector of coefficients. CSP estimates the coefficient vector by maximizing the sample likelihood function using an iterative Newton-Raphson procedure.

5.3 Providing the data for logit analysis

Since the estimation of a logit model involves an iterative procedure, the logit processor works from a compact file of final variables on the file LGTFILE. On how to create this file see Section 3.9.3.

The following points should be noted:

5.3.1 The explanatory and "dependent" variables

The MODEL statement should specify the explanatory variables only. The PROCESS statement must contain the keyword LGTFILE and specify the "dependent variable" using the DEP parameter as well as the number of categories using the NCAT parameter.

5.3.2 The "dependent" variable for multinomial logit

For multinomial logit, the "dependent" variable should be one of the original variables (scalar or characteristic) the value of which is the index of the event which occurred for the given observation. For example, if M equals 3 then NCAT = 3 should appear on the process statement; if SC1 is the "dependent" variable and the 2nd event occured for a given observation, then SC1 should equal 2 for that observation.

5.3.3 Organization of data for conditional logit

For conditional logit each alternative will appear as an "observation" on LGTFILE so that there will be $\sum_{t=1}^{T} M_t$ "observations" on the file in all (rather than T as for a multinomial logit problem). The "dependent" variable should be one of the original variables which takes on the value 2 for the chosen alternative and the value 1 otherwise. Furthermore, the chosen alternative should always appear last for every observation. For example, suppose there are three observations and two explanatory variables

X_1	X_2	Alternative	Observation
.1	5	1 (chosen)	1
.3	1	2	
.05	-3	3	
.0	4	1	2
.7	-3	2 (chosen)	
.1	2	1	3
.2	-1	2	
.0	7	3 (chosen)	
.4	5	4	

then the "observations" on LGTFILE should be

X_1	X_2	Value of dependent variable	"Observation"	Observation
.3	1	1	1	1
.05	-3	1	2	
.1	5	2	3	
.0	4	1	4	2
.7	-3	2	5	
.1	2	1	6	3
.2	-1	1	7	
.4	5	1	8	
.0	7	2	9	

For conditional logit NCAT = 2 should appear on the PROCESS statement.

5.3.4 Repetition factor

It is often necessary to weight the observations (e.g. when identical observations are repeated many times it is more economical to enter them once with a weight equal to the number of times the observation is repeated). This may be done by setting one of the original variables (scalar or characteristic) equal to the weight or repetition factor and by specifying REP = SCi or REP = CHj on the PROCESS statement. Note that for conditional logit the value of this variable for the "observation" consisting of the chosen alternative is the value used.

5.3.5 LGTFILE created on previous run

The file LGTFILE need not be created on the current run. It can be created on a separate run or saved from one run to another and then attached for the current run (see Section 7).

5.3.6 Modifying data on LGTFILE with subroutine ALTER

If the keyword ALTER appears on the LOGIT statement then the user-provided subroutine will be called after each observation is read from LGTFILE but before the observation is processed.

Example 5.1 Use of ALTER to select observations

```
        . . .
        LOGIT ALTER $
        . . .
            SUBROUTINE ALTER (VALUES, LOCS, JLOC, ICAT, RFACT, IDROP)
            DIMENSION VALUES (1),LOCS(1)
            IF (VALUES (1), LE.O.)IDROP = 1
            RETURN
            END
        . . .
```

Note: a) The subroutine parameters include the vectors VALUES and LOCS. These are as described in Section 3.7 except that elements of LOCS corresponding to dummy variables have the corresponding element of VALUES set to 1 (hence, the vectors are of the same length--JLOC). ICAT is the value of the "dependent" variable for the current observation and RFACT the repetition factor. An observation may be dropped by setting IDROP equal to 1.

b) For this example an observation is dropped if VALUES (1) is not positive.

c) See Section 8 on how to attach user-provided subroutines.

5.3.7 Excluding variables

If the keyword NORM = i, j, k appears on the LOGIT statement, then the variables i, j, k will be excluded from the estimation. These variables will appear on the output with coefficients and standard errors zero. As in regression analysis (see Section 4.6) the last of a group of multicollinear variables will be excluded automatically and such a variable too will appear in the output with coefficient and standard error zero.

5.4 Specifying multinomial or conditional logit

If the keyword CONDIT appears on the LOGIT statement, conditional logit analysis will be performed. If not, multinomial logit analysis will be performed.

5.5 Initial values of the coefficients

Since the estimation procedure is iterative, there is need for an initial point, that is for initial values of the coefficients. There are a number of ways of providing such initial values:

5.5.1 Initial values zero

If no values are provided by the user, the initial values of the coefficients are taken to be zero.

5.5.2 Attaching the file COEFF

CSP automatically writes the vector of coefficients for the last iteration onto file COEFF. This may be saved by the user and attached for a new run (see Section 7), say for a number of additional iterations. The program is notified of the availability of the coefficients by writing the keyword OLDB on the LOGIT statement.

5.5.3 Input of coefficients from cards

The appearance of the keyword READB on the LOGIT statement causes CSP to read the coefficients from cards (card images) immediately following the LOGIT statement according to the format (5G16.8). No header is read. The parameter NB = n must appear on the LOGIT statement where n is the number of coefficients to be read.

5.5.4 Coefficients on the LOGIT statement

The coefficients will be read from the LOGIT statement if they appear in the following form:

```
        LOGIT          B = 1  .02  7.3  2  .05  3.4  $
```

where there are six coefficients which follow the keyword B = .

5.6 Convergence

5.6.1 Criteria for convergence

The maximum number of iterations to be performed (less if convergence is acheived) is set to n by writing NITER = n on the LOGIT statement. If this does not appear, one iteration is performed. There are two criteria for convergence--absolute and percentage. A coefficient value will be considered to have converged in absolute value if the absolute change from the previous iteration is less than .01. The criterion value may be changed to x by writing CONABS = x on the LOGIT statement.

A coefficient value will be considered to have converged in percentage terms, if the change from the previous iteration divided by the current value of the coefficient is less than .01. The criterion value may be changed to x by writing CONPC = x on the LOGIT statement.

No further iterations will be performed if all of the coefficient value have converged according to at least one of the criteria.

5.6.2 The problem of extreme cell frequencies

When the explanatory variables include dummy variables and one of the dummy variables is always 1 or always 0 for one of the categories (multinomial logit) or for the chosen alternative (conditional logit) then the maximum likelihood value of the corresponding coefficients will be plus or minus infinity. For obvious reasons such coefficients will not converge.

5.6.3 The problem of bad initial values

For multinomial logit convergence is not assured for all initial values of the coefficients. In particular, zero values may not be a good starting point. A solution that often works is to take estimates from a linear probability model and transform these for use as initial values.

5.7 Output from the logit processor

After each iteration CSP prints the vector of coefficient increments (change from the last iteration), standard errors and coefficient values together with the log-likelihood of the previous iteration. After the final iteration a table of coefficients and standard errors is printed and the coefficients are written on to the file COEFF as a single logical record. If the keyword PUNCH appears on the LOGIT statement, the coefficients are also written on to file PUNCH, 5 to a card (5G16.8), preceded by a card with the current title.

5.8 Using CSP to predict probabilities

Example 5.2 Predicting probabilities within the sample--multinomial logit. Suppose that coefficients have been calculated for a multinomial logit model:

```
MODEL/5, 10/ SC1, SC2/ DUMMY/(CH1(4) ) CONSTANT $
PROCESS LGTFILE DEP = CH2 NCAT = 4 $
LOGIT NORM = 5 NITER = 10 $
```

The files LGTFILE and COEFF have been saved by the user and are attached for the prediction run below (see Section   ). Now the user wishes to calculate the probabilities of occurrence of the various categories for each observation. The control statement should be of the form

```
LOGIT OLDB PROBS $
```

and the user should attach the user-provided subroutine PROBS (see Section 8)

```
. . .
SUBROUTINE PROBS (P, RFACT)
DIMENSION P(1)
. . .
```

The subroutine is called once for each observation and the vector P contains the probabilities of the NCAT categories. RFACT is the repetition factor.

Example 5.3 Predicting probabilities within the sample --
conditional logit

Suppose that coefficients have been calculated for a conditional logit model:

```
MODEL/3, 4/SC1, SC2 $
PROCESS LGTFILE DEP = CH1 NCAT = 2 $
LOGIT CONDIT NITER = 5 $
```

The files LGTFILE and COEFF have been saved by the user and are attached for the prediction run below (see Section 7). Now the user wishes to calculate the probabilities of choice for the various alternatives for each observation. The control statement should be of the form

```
        LOGIT OLDB CONDIT PROBS $
```

and the user should attach the user provided subroutine PROBS (see Section 8).

```
        SUBROUTINE PROBS (P, N, RFACT)
        DIMENSION P(1)
        .  .  .
```

The subroutine is called once for each observation (each group of alternatives). The vector P contains the probabilities of the N alternatives.

Example 5.3 Simulating changes in the data. Basically the same procedure should be followed as in Examples 5.2 and 5.3. The data might be altered by calling ALTER, in which case the control statement for the prediction run would be, e.g.,

```
        LOGIT ALTER OLDB PROBS $
```

Alternatively a new LGTFILE could be created to embody the changes. E.g., for Example 5.2,

```
        MODEL/5, 10/ SC1, SC2/DUMMY/(CH1(4) ) CONSTANT $
        RESCALE CH1 = (0  1  1  2  2 ) $
        PROCESS LGTFILE DEP = CH2  NCAT = 4 $
        LOGIT OLDB PROBS $
```

6. CSP CONTROL STATEMENTS

6.1 Format of CSP control statements

The statements are written in free format: the position of items and their order on a given statement are not important.

Each statement begins with a control statement identifier (e.g. MODEL, PROCESS, LOGIT) and ends with the symbol $. A statement may be continued on to any number of cards (card images) and any number of statements may be written on a single card (except that TITLE and LOAD must be the last statements on a card as must LOGIT if the keyword READB is used).

Keywords and list items must be separated from one another by commas or blanks or by special characters (*, /,),(, =) where these are appropriate.

The last statement should be

STOP $

if it is not present , the end-of-record will be interpreted as a STOP statement.

6.2 The CSP control statements and their keywords

In the following, numbers in parentheses will refer to the section where the use of the statement or keyword is discussed.

6.2.1 EDIT

The appearance of this statement anywhere among the control statements will prevent all processing other than reading and interpreting the CSP control statements and setting up the parameters for further processing.

All files and subroutines required should be available. The space required at each stage will be printed.

6.2.2 FILES

FILES sname = newname $

sname - the standard name for the file is sname
newname - the name to be used is newname
If it appears this should be the first statement.

6.2.3 GENERATE (4.3)

GENERATE var = expression $

var - the number of a final variable
expression linear combination of final variables
e.g. 2 + 3. * 4
2 and 4 are numbers of final variables, 3. is a constant coefficient.

6.2.4 LOAD (4.1.2)

LOAD HEADER $

or

LOAD IMAT = imat NOV = nov NOBS = nobs $

HEADER - indicates that the following card image is a header for the matrix to be read, which follows the header
IMAT = imat - the sequence number of the matrix is imat
NOV = nov - the dimension of the matrix is nov
NOBS = nobs - the number of observations is nobs

6.2.5 LOGIT (5)

LOGIT keywords $

The following keywords are recognized:

CONDIT - perform conditional logit analysis (5.2, 5.4)
ALTER - subroutine ALTER to be called (5.3.6)
PROBS - subroutine PROBS to be called (5.8)
PUNCH - write coefficients on to file PUNCH (5.7)

OLDB - coefficients available on file COEFF (5.5.2)

READB - coefficients follow (5.5.3)

NB = n- n coefficients are to be read

B = $\underline{\text{coeff}}_1, \ldots, \underline{\text{coeff}}_n$, the coefficients $\underline{\text{coeff}}_1$ to $\underline{\text{coeff}}_n$ are to be used (5.5.4)

NORM = $\underline{\text{var}}_1, \underline{\text{var}}_2 \ldots$ the variables $\underline{\text{var}}_1, \underline{\text{var}}_2, \ldots$ are to be excluded(5.3.7)

NITER = $\underline{n}$ at most $\underline{n}$ iterations to be performed (5.6.1). (default is $\underline{1}$)

CONABS = $\underline{x}$ absolute criterion for convergence is $\underline{x}$ (5.6.1) (default value is .01)

CONPC = $\underline{x}$ percentage criterion for convergence is $\underline{x}$ (5.6.1) (default value is .01)

6.2.6 MATRIX (4.2)

MATRIX keywords $

The following keywords are recognized:

PRINT	print the moment-matrix (4.2.1)
MEANVAR	print the means, variances and standard errors (4.2.2)
CORR	print the correlation matrix (4.2.3)
$\underline{n}$	process matrix $\underline{n}$ (4.2.4) (default value is 1)
SAVE	the matrix currently in core is saved on file SAVE (4.3)

6.2.7 MODEL (3.6)

MODEL/$\underline{\text{nsc}}$, $\underline{\text{nch}}$/$\underline{e}_1, \ldots, \underline{e}_i$/DUMMY/$\underline{d}_1, \ldots, \underline{d}_j$ CONSTANT $

$\underline{\text{nsc}}$	the number of scalar original variables (3.6.1)
$\underline{\text{nch}}$	the number of characteristic original variables (3.6.1)
$\underline{e}_1, \ldots, \underline{e}_i$	expression of the form:

SCk, CHl original variables (3.6.2)

(SCk(m)) linear spline (3.6.3)

(SCk(m), SCl(n))bilinear spline (3.6.3)

/DUMMY/ expressions after this generate dummy variables (3.6.4)

$\underline{d}_1$, ..., $\underline{d}_2$	expressions of the form (CHk(m)) simple set of dummies (CHk(m), CHl(n)) interactions.
CONSTANT	generate constant (3.6.5). Must come last and after /DUMMY/.

6.2.8 NAMES (3.12)

NAMES keywords $

The following keywords are recognized:

SC_i = <u>name</u>	scalar i receives the name <u>name</u>
CH_i = <u>name</u>	characteristic i receives the name <u>name</u>
VR_i = <u>name</u>	final variable i receives the name <u>name</u>
PRINT	print the names of the final variables

Note: a) <u>name</u> must be no longer than 10 characters and consist of alphanumeric characters only with no embedded blanks.

b) This statement must be preceded by a MODEL statement.

6.2.9 PROCESS

PROCESS keywords $

The following keywords are recognized:

SC	print the vector of scalars for the first NPRINT observations (3.11.1)
CH	print the vector of characteristics for the first NPRINT observations (3.11.1)
Z	print the vector Z for the first NPRINT observations (3.11.1)
VAL	print the vector of final variables for the first NPRINT observations (3.7, 3.11.1)
OWN	subroutine OWN to be called (3.3.1)
TRANS	subroutine TRANS to be called (3.4)
WEIGHT	subroutine WEIGHT to be called (3.8)

BEFORE	subroutine BEFORE to be called (3.10)
AFTER	subroutine AFTER to be called (3.10)
READ	no processing beyond TRANS (3.10)
PUNCH	punch matrix(es) (3.9.1)
NOMAT	no matrix to be created (3.9.4)
LGTFILE	create file for logit analysis (3.9.3)

NMAX = n read no more than n observations (default value is ∞) (3.11.4)

NPRINT = n print no more than n observations (default value is 100) (3.11.1)

DEP = vname dependent variable for logit file (3.9.3)

vname is of the form CHi or SCj

REP = vname repetition factor (3.9.3)

NCAT = n number of categories (3.9.3)

NMAT = n create n matrices (3.9.2) (default value is 1)

Note: a) If LGTFILE appears so must DEP = vname and NCAT = n

b) This statement must be preceded by a MODEL statement.

6.2.10 REG (4)

REG keywords $

The following keywords are recognized:

DEP = varlist	the variables varlist are to be dependent variables (4.4, 4.5)
INDEP = varlist	the variables varlist are to be independent variables for an ordinary least squares regression (4.4)
ENDOG = varlist	the variables varlist are to be endogenous explanatory variables for two-stage-least-squares regression (4.5)
EXOG = varlist	the variables varlist are to be exogenous explanatory variables for two-stage-least-squares regression (4.5)
INST = varlist	the variables varlist are to be instrumental variables for two-stage-least-squares regression (4.5)

NORM varlist	the variables varlist are to be excluded from the regression (4.6)

Note: varlist is a list of numbers of final variables e.g. 10,11,12,6 or 1 THRU 7,9,16,12 THRU 14

MATRIX	print the moment-matrix of independent variables (4.7)
INVERSE	print the inverse (4.7)
PUNCH	write coefficients on to file PUNCH (4.7)
EXP	print the antilog of the coefficients (4.7)
ENDREG	subroutine ENDREG to be called (4.8)
TOL = x	set the inversion tolerance to x (4.6) (default value is .00005).

6.2.11 RESCALE (3.5)

RESCALE keywords $

The following keywords are recognized:

CHi = (j_0, j_1, ..., j_n) the characteristic original variable is to be recoded , 0 to j_0, 1 to j_1, etc.

SCi = (a_1, a_2, ..., a_n) the scalar original variable is to be scaled, a_1 to 1, a_2 to 2 etc.

Note: This statement must be preceded by a MODEL statement,

6.2.12 STOP

The final statement.

6.2.13 TITLE (3.12.2)

To be followed immediately by the title to be entered.

7. FILES USED BY CSP

The following files are defined and used by CSP:

Name under SCOPE	Logical Unit	Purpose
INPUT	1	Control statements
NAMES	2	Names of transformed variables
TAPE3	3	Scratch file
MATRIX	4	Regression moment matrices
USERFIL	5	User data file
COEFF	6	Logit coefficients
TAPE7	7	Scratch file
PUNCH	8	Punch output
SAVE	9	Saved transformed matrices
CSPFILE	-	Standard data file
VALFILE	-	Data file for logit iterations
OUTPUT	-	

The SCOPE file names of logical units 2 through 9 may be redefined using the FILES statement (6.2.2). The SCOPE file names of INPUT and OUTPUT may be redefined by placing the alternative names as parameters on the load-and-go statement (see Section 8), e.g.

```
CSP(MYINP,MYOUT)
CSPOBJ(,MYOUT)
```

8. ACCESS TO CSP AND USER-PROVIDED SUBROUTINES

To use the standard version of CSP (with 5640 words of blank common) the following SCOPE statements are required:

```
ATTACH, CSP.
CSP.
```

The standard version requires 32000_8 words of central memory. This suffices for the creation of moment matrices of up to (approx.) 90 variables and the calculation of regressions of up to (approx.) 70. The exact memory requirement (words of blank common) is printed at each stage and if the available memory is insufficient, processing stops.

In order to increase the size of blank common or to include user-provided subroutines in the program the following procedure should be followed.

CSP is organized as a set of overlays. Subroutines must be added to the zero-level overlay, which is now also provided by the user in the following form:

```
OVERLAY(CSP,0,0)
PROGRAM MAIN
COMMON/CNTRL/NSPACE,X(170)
COMMØN SPACE (n)
NSPACE = n
CALL OVERLAY(3HCSP,2,0)
CALL OVERLAY(3HCSP,1,0)
STOP
END

{User-provided subroutines       }
{(e.g. OWN,TRANS,ENDREG,ALTER)}
```

n denotes the size of blank common desired by the user.

To run CSP in this way the following set-up should be used:

```
        .
        .
        .
        ATTACH, CSPOBJ.
        RUN(S).
        LOAD(LGO)
        CSPOBJ.
        .
        .
        .
    7/8/9
              OVERLAY(CSP,0,0)
              etc
    7/8/9
              CSP Control Statements
6/7/8/9.
```

To attach subroutines for CSPFILE creation :

```
         .

         .
         ATTACH,CSPLIB.
         RUN(S)
         LDSET(LIB=CSPLIB)
         LGO.
         .
         .
         .

     7/8/9
              { User's Program }
   6/7/8/9
```

DISCUSSION PAPERS PUBLISHED IN HEBREW:

דפי דיון

2-75 "הפצרי על ההתיקרות בגין מדיניות החרום הכלכלית", יעקב חביב ורוברט לרמן.

2-76 "שרותים לעולים קשישים בישראל", יחיאל ערן וגרדה פרידהיים.

6-76 "מוגבלות ושיקום - שלב בהתפתחות האדם: מודל מתימטי", משה נורדהיים.

8-76 "תכניות לקשישים במרכזים קהילתיים בישראל", סוזי לינדר-פלץ.

10-76 "טיפול בית - קווים מנחים לגיבוש מדיניות וארגון הספקת השרות", נסים ברוך.

12-76 "סקירה בקורתית של מדיניות השכון בישראל", רוברט לרמן.

14-76 "שמוש בפנאי של גילאי 70+ - סקר ראשוני", חנה ויול ואירית ברמן-אשכנזי.

16-77 "לברור המונח 'זקן' במקרא", משה א. קורץ.

18-77 "סקירה ספרותית בנושא הכנה לפרישה מהעבודה", נורית ניראל.

20-77 "הקשישים בירושלים: תמורות ותחזיות דמוגרפיות", מרים שטרקשל ועמירם גונן.

22-77 "דיור תומך קבוצתי לקשישים (מעונות ודיור מוגן): מספר מקומות ושטחי הקרקע לאוכלוסיה היהודית בירושלים, 1992-1977", עמירם גונן, מרים שטרקשל וישראל קמחי.

24-78 "סיבות פניה למעונות לקשישים עצמאיים", חנה ויול ואילה ביבר.

26-78 "תמורות במערכת הפנסיה", יעקב חביב וחיים פקטור.

28-78 "מערכת הפנסיות מעבודה בישראל - הכדאיות לפרט", חיים פקטור.

30-78 "נטיה לניידות בקרב שכירים בישראל", יעקב ויסברג.

REPRINT SERIES

רשימת תדפיסים

R-1-78 "The Hazard of Persistent Cigarette Smoking in Later Life," J.H. Abramson.

R-2-78 "The Effect on Poverty Status in Israel of Considering Wealth and Variability of Income," Jack Habib, Meir Kohn, and Robert Lerman.