DOCUMENT RESUME

ED 152 978 CE 015 071

AUTHOR	Gilpatrick, Eleanor
TITLE	The Health Services Mobility Study Method of Task Analysis and Curriculum Design. Research Report No. 11. Volume 3: Using the Computer to Develop Job Ladders.
INSTITUTION	Health Services Mobility Study, New York, N.Y.
SPONS AGENCY	City Coll. Research Foundation, New York, N.Y.; City Univ. of New York, N.Y. Hunter Coll. School of Health Sciences.; Employment and Training Administration (DOL), Washington, D.C.
PUB DATE	77
CONTRACT	82-34-69-34
NOTE	330p.; Not available in hard copy because of reproducibility problems; For related documents see ED 149 021, ED 149 022, and CE 015 072
EDRS PRICE	MF-$0.83 Plus Postage. HC Not Available from EDRS.
DESCRIPTORS	Administrator Guides; *Career Ladders; Classification; *Computer Programs; Curriculum Development; *Guidelines; *Health Occupations; Information Processing; Information Utilization; Job Analysis; *Job Skills; Manpower Development; Manuals; Occupational Mobility; Performance Criteria; Systems Analysis; *Task Analysis
IDENTIFIERS	Health Services Mobility Study

ABSTRACT

This document is volume 3 of a four-volume report which describes the components of the Health Services Mobility Study (HSMS) method of task analysis, job ladder design, and curriculum development. Divided into four chapters, volume 3 is a manual for using HSMS computer based statistical procedures to design job structures and job ladders. Chapter 1 is a manual for coding and preparing the HSMS task data for computer-based analysis. Chapter 2 describes how to use HSMS task data and computer programs to group tasks into interrelated families and hierarchies of tasks. Chapter 3 describes how to assign tasks to job levels, job structures, and job ladders. Finally, chapter 4 indicates how the analytic results can be used by an individual institution or department to make rational use of manpower, restructure jobs, assign tasks to jobs and titles at various levels, provide upward mobility, and/or evaluate task performance. The appendixes contain the five HSMS computer programs, instructions for their use, and various related materials. (The other three volumes of the HSMS study include the following information: the HSMS skill and knowledge scales and knowledge classification system [volume 1]; a companion document which describes the work carried out by the director of a task analysis project and its job analysts [volume 2]; and the HSMS curriculum design method [volume 4].) (BM)

THE HEALTH SERVICES MOBILITY STUDY METHOD OF TASK ANALYSIS AND CURRICULUM DESIGN

Research Report No. 11

Volume 3

USING THE COMPUTER TO DEVELOP JOB LADDERS

by
Eleanor Gilpatrick, Director
Health Services Mobility Study

Contract No. 82-34-69-34
EMPLOYMENT AND TRAINING ADMINISTRATION
U.S. Department of Labor

Sponsored by Hunter College and
The Research Foundation, City University of New York

ACKNOWLEDGEMENTS

The author gratefully acknowledges the funds and time provided the Health Services Mobility Study (HSMS) over ten years of continuous funding. This has allowed for design of the method, field testing, revision, and application in "real-world" settings. The major funding for the Health Services Mobility Study has come from the Manpower Administration, now the Employment and Training Administration of the U.S. Department of Labor (Contract No. 82-34-69-34). Our special thanks go to William Throckmorton, our Project Officer, who has cared for the project and nurtured it.

The project had its origin in 1967 with the Office of Health Affairs of the Office of Economic Opportunity (Grant No. CG 8783). Other funding has come from the Health Services and Mental Health Administration (Contract No. 110-69-256) and a Memorandum of Agreement with the Division of Allied Health Manpower, Bureau of Health Manpower Education, Department of Health, Education and Welfare. We also received funds to assist with the completion of the work from the Bureau of Radiologic Technology, New York State Department of Health.

HSMS is sponsored by the Hunter College School of Health Sciences, where the author is an associate professor, and the Research Foundation of the City University of New York. The support and encouragement given the project by Dr. Michael R. McGarvey, Vice President for Health Affairs at Hunter College, has made it possible to complete the work. The author is deeply grateful.

The methodological work represented in this document is the result of the collaboration of many individuals. I wish to thank, in particular, Dr. Earl E. Davis who was HSMS's chief consultant for methodology, Stephen Jasik, Edward Friedman and George Chaikin, our chief program consultants, Irene Seifer who made major inputs during the development of the scales, who led the field work, and who provided editorial review for this document, and Christina Gullion, who was in charge of data processing and helped in the preparation of the appendixes. Richard Preston did the typing.

We deeply appreciate the help of the administration and staff of several hospital centers and health facilities in New York City where we were allowed to conduct field tests, pilot tests, and full scale application of the method. These include The Montefiore Hospital and Medical Center, Dr. Martin Luther King, Jr. Health Center, Memorial (Sloan-Kettering) Hospital, the Mount Sinai Hospital and Medical Center, and LaGuardia Hospital.

It would be impossible to fully acknowledge all the other people who have been involved in the work. I thank everyone for their help and support. Any errors are solely the responsibility of the Health Services Mobility Study and the author.

Eleanor Gilpatrick

The research reported herein was conducted under a contract with the Employment and Training Administration, U.S. Department of Labor, under the authority of the Comprehensive Employment Training Act of 1973. Researchers are encouraged to express their own judgments freely. Interpretations or viewpoints stated in this document do not necessarily represent the official position or policy of the U.S. Department of Labor, the New York State Department of Health, or the City University of New York.

PREFACE

In September of 1967, the author became the director of the Health Services Mobility Study, a project funded by the Office of Economic Opportunity. The grant carried the charge that the project investigate the impediments to upward occupational mobility in New York City Municipal Hospitals and that it suggest means of overcoming obstacles to such mobility. It was a one-year grant.

Ten years later, the Health Services Mobility Study (HSMS) is ending its research and development activities. During that time, HSMS examined the occupational structure of New York City Municipal Hospitals and investigated the problems of skill shortages and credentialing.[1] It then undertook to design a method to promote occupational mobility by tying job requirements to curriculum design in a single system.

HSMS developed, field tested, and applied a new task analysis method to analyze work and design job ladders. It produced a method of curriculum design using task data that also makes it possible to design educational ladders to parallel job ladders. The HSMS method can be used to make job structures and curricula responsive to quality standards and the needs of consumers.

HSMS has made theoretical contributions to the fields of job analysis, curriculum development, and occupational testing. It has helped to promote the concepts of upward occupational and educational mobility, and has developed a design for a safe practice, quality assurance program in diagnostic radiology.

The HSMS method was pilot-tested in an ambulatory care community health center. It was given a full-scale application in diagnostic radiology. An abbreviated version of the method was applied to the technologist, technician and aide functions in radiation therapy and diagnostic ultrasound. A curriculum has been developed covering the aide, technician, and technologist levels in diagnostic radiology.

Although these applications have been in health services occupations, all of the components of the method are generic and can be applied to any work activity in any industry.

---

[1] Eleanor Gilpatrick and Paul Corliss, The Occupational Structure of New York City Municipal Hospitals, New York: Health Services Mobility Study and/or Praeger Publishers (Research Report No. 2), 1970.

Now the time has come to share the method so that it can be used by others. This research report offers all the components of the HSMS method of task analysis, job ladder design, and curriculum development for use as a system or in part. It is offered to any institution that wishes to expend time and resources to rationally structure work, utilize its labor force, evaluate its work performance, develop job ladders, design job-related education, or create work-related test instruments. This material is reported as follows:

Research Rpt. No. 11 — THE HEALTH SERVICES MOBILITY STUDY METHOD OF TASK ANALYSIS AND CURRICULUM DESIGN.

Vol. 1 — Basic Tools: The Concepts, Task Identification, Skill Scales and Knowledge System.

Vol. 2 — Writing Task Descriptions and Scaling Tasks for Skills and Knowledge: A Manual.

(Also contains an abbreviated version of the task description method.)

Vol. 3 — Using the Computer to Develop Job Ladders.

(Includes technical material, computer programs, scholarly review, and a mini-manual for performance evaluation.)

Vol. 4 — Developing Curriculum Objectives from Task Data: A Manual.

The reader is directed to other HSMS documents for additional information not contained in Research Report No. 11 as follows:

Technical Rpt. No. 11 — HEALTH SERVICES MOBILITY STUDY: FINAL REPORT FOR THE PERIOD OCTOBER 1967 THROUGH MARCH 1972.

(Contains a review of the literature in task analysis and the derivation of the HSMS task analysis method.)

Working Paper No. 11 — THE DESIGN OF CURRICULUM GUIDELINES FOR EDUCATIONAL LADDERS USING TASK DATA.

(Earlier version of HSMS curriculum design method. Contains a review of the literature in occupational curriculum design and behavioral objectives, and other related material.)

Research Rpt. No. 7	TASK DESCRIPTIONS IN DIAGNOSTIC RADIOLOGY.
Vol. 1	Medical Tasks: What the Radiologist Does.
Vol. 2	Radiologic Technologist Tasks Dealing With Patient Procedures.
Vol. 3	Machine-Related, Patient Care and Administrative Tasks: What Radiologists, Technologists, Nurses, and Physicists Do To Run Things and Look After Patients and Equipment.
Vol. 4	Index of Tasks by Code Number and Extended Name.
Research Rpt. No. 8	USING TASK DATA IN DIAGNOSTIC RADIOLOGY.
Vol. 1	Job Ladders: Assigning Tasks to Jobs.
Vol. 2	Curriculum Objectives for Radiologic Technology.
Research Rpt. No. 9	THE TECHNOLOGIST FUNCTION IN FIELDS RELATED TO RADIOLOGY: TASKS IN RADIATION THERAPY AND DIAGNOSTIC ULTRASOUND.
Research Rpt. No. 10	RELATING TECHNOLOGIST TASKS IN DIAGNOSTIC RADIOLOGY, ULTRASOUND AND RADIATION THERAPY.
Working Paper No. 12	USING TASK DATA FOR PERFORMANCE EVALUATION AND PROFICIENCY TESTING. (tentative title)
	(Theory of criterion-referenced and norm-referenced testing; use of task data as inputs to testing. The HSMS theoretical document on occupational proficiency tests and issues of validity.)

CONTENTS

FIGURES

CHAPTER 1

HSMS STATISTICAL ANALYSIS: INTRODUCTION

This third volume of Research Report No. 11 is a manual for using HSMS task data to design job structures and job ladders; it also describes the use of task data for institutional performance evaluation and manpower planning.

The four volumes of this report present the entire HSMS task analysis and curriculum design system. Volume 1 contains the HSMS skill and knowledge scales and the HSMS Knowledge Classification System. It is the companion document to Volume 2, which describes the work carried out by the director of a task analysis project and its job analysts and covers task identification, task description, skill scaling, and knowledge identification and scaling. Volume 4 presents the HSMS curriculum design method.

ABOUT VOLUME 3

This volume describes how HSMS uses computer-based statistical procedures to design job structures and job ladders. It presents the HSMS computer programs and describes how to use them to group tasks, arrange them into jobs and job ladders, and how to use the results for performance evaluation and manpower planning.

Chapter 1 is a manual for coding and preparing the HSMS task data for computer-based analysis. Chapter 2 describes how to use HSMS task data and computer programs to group tasks into interrelated families and hierarchies of tasks. Chapter 3 describes how to assign tasks to job levels, job structures, and job ladders.

Chapter 4 indicates how the analytic results can be used by an individual institution or department to make rational use of manpower, restructure jobs, assign tasks to jobs and titles at various levels, provide upward mobility, and/or evaluate task performance.

The five HSMS computer programs are presented in Appendixes B through F, along with instructions for their use. Appendix A presents a brief history of the HSMS analytic instruments, covering data on the definition of task, the HSMS scales, and the Knowledge Classification System. Appendix G is a scholars' review of the HSMS method.

CODING AND KEYPUNCHING

When the HSMS tasks have been identified, described, scaled for skill and knowledge requirements, reviewed, and approved, the final data forms are ready for coding and data processing. The task identification data appear on HSMS Task Description Sheets or Task Identification Summary Sheets. The skill scale data appear on HSMS Skill Scaling Sheets, and the knowledge category and scale data appear on HSMS Knowledge Identification Sheets. The data must then be transferred to punched computer cards. This is done by use of code sheets which represent the columns on computer cards. This section describes the code sheets and how they are used.

Overview

Each task which is to enter the data base is a basic unit for purposes of statistical analysis. Each is uniquely identified by its Code Number. All data cards for a given task must include the task's Code Number.

To facilitate information retrieval, we made it possible to identify a task by several criteria such as job title, institution, department, task frequency, and an abbreviated name. The latter makes it possible to easily interpret data listings and the analytic results which appear on computer print-outs. Each data card contains a fixed number of columns set aside for identification purposes; but only the first card includes the abbreviated task name.

The skill scale data posed no special problem. With sixteen skill scale values to record for each task, we decided that the second data card for a task would have a fixed format in which a specific two-column field would be assigned to each skill scale, and scale values would be punched excluding the decimal points. Thus, a task's scale value for a skill can be determined by the numbers punched in given columns on the card.

In the case of the knowledge categories, no such fixed assignment of category to column could be made because of the vast number of knowledge categories in the System. Even if we could know in advance which categories would be involved in each "run," a fixed column assignment for each category would require hundreds of cards and thousands of zero punches, since most categories are not needed for most tasks. A new format would be needed for each run. This problem was solved by the decision to use a format for knowledge data in which eight columns are assigned for entry of a knowledge category's code number, followed by two columns in which tc enter the scale value. For any given task there as many knowledge data cards as are needed to cover all the knowledge categories it requires at non-zero values. Each task's first data card tells the computer how many other data cards to expect for the given task.

Thus, any task must have at least two data cards. Tasks that require no knowledge categories above zero on the knowledge scale have only two cards. All other tasks have as many data cards as are required to record all the scale value data. The data cards for each task are numbered T00, T01, T02, and so on; each contains the Task Code Number, any optional identification information selected, and the data that the particular card has been designed to carry.

Figure 1 presents the HSMS T00 code sheet for a task. There are four banks which together represent the 80 columns of a computer card. The first line of each bank gives the column numbers. The second line indicates the column designations, i.e., the instructions to the coder on what to enter on the third line. The third line is what the keypunch operator punches, based on what has been entered by the coder.

Columns 1, 2, and 3 represent the card number for the given task. A ' is always punched in column 1 to designate (T)ask data.

Figure 1. HSMS CARD TOO SHEET: SUMMARY CARD

Col. No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Content	Task Card	Data Number		X	Task Identification Number						X	Job Title Code			X	Institution Code	X	Performer Code		
Code	T	0	0	X							X				X		X			

Col. No.	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Content	Department Code		Shift Code	X	Type of Task	X	Task Frequency	X	Number of Cards to Follow		X	Alpha - Numeric Name of Task								
Code				X		X		X			X									

Col. No.	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
Content	(Abbreviated Version) in Columns 32 to 78																			
Code																				

Col. No.	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
Content																			X	
Code																			X	

] Indicates blank. Coded by: ________ Checked by ________

This distinguishes the task data cards from other cards. Columns 5 through 27 are identical for a given task on all its cards. The task's Code (Identification) Number is always punched in columns 5 through 10, right justified in the field. Columns 12 through 27 contain optional data which are not needed for statistical analysis. HSMS enters an "N" in column 25 to designate a task that has been reviewed and is now in "normative" final form.

Columns 29 to 80 are unique to card T00. Columns 29 and 30 are coded after all the skill and knowledge scale values have been coded. The number entered tells the computer how many cards are to follow for the task. The figure corresponds to the number of the last knowledge card for the task.

Figure 2 is the code sheet for the T01 card; it is designed to include all the skill scale data for a task. There are column designations for all 16 skill scales. If a task has been scaled at zero for a skill, zeroes should be coded and punched in the columns assigned to the skill.

Figure 3 is the code sheet for all the knowledge data cards for the tasks. Columns 2 and 3 are coded in numerical order as the code sheets are filled out, beginning with T02. There is room for data for four knowledge categories per card. The coder enters a knowledge category's own 8-digit code number; this is followed by the nonzero scale value for the category (omitting the decimal point) in the two columns designated.

Figure 2. HSMS CARD T01 SHEET: SKILL SCALES

Col. No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Content	Task Data Card Number			X	Task Identification Number						X	Job Title Code			X	Institution Code	X	Performer Code		
Code	T	0	1	X							X				X		X			

Col. No.	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Content	Department Code		Shift Code	X	Type of Task	X	Task Frequency	X	X	X	X	Scale 2 Locomotion		X	Scale 3 Object Manipulation		X	Scale 4 Guiding or Steering		X
Code				X		X		X	X	X	X			X			X			X

Col. No.	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
Content	Scale 5 Human Interaction		X	Scale 6 Leadership		X	Scale 7 Oral Use of Language		X	Scale 8 Reading Use of Language		X	Scale 9 Written Use of Language		X	Scale 10 Decision Mk: Methods		X	Scale 11 Decision Mk:Quality	
Code			X			X			X			X			X			X		

Col. No.	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
Content	X	Scale 12 Figural Skills		X	Scale 13 Symbolic Skills		X	Scale 14 Taxonomic Skills		X	Scale 15 Implicative Skills		X	Scale 16 Financial Error Cons.		X	Scale 17 Cons. of Err. Humans		X	
Code	X			X			X			X			X			X			X	

X Indicates blank.

Coded by: ____________ Checked by: ____________

Figure 3. HSMS CARD T02 (Or Higher) SHEET: KNOWLEDGE

Col. No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Content	Task Card	Data	Number	[X]		Task	Identifi	cation	Number		[X]	Job	Title	Code	[X]	Institution Code	[X]	Perf	ormer	Code
Code	T			[X]							[X]				[X]		[X]			

Col. No.	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Content	Department Code		Shift Code	[X]	Type of Task	[X]	Task Frequency	[X]	[X]	Knowledge ID Number	(8 digits)	Classification		System	→			[X]	Scale Value Scale	18
Code				[X]		[X]		[X]	[X]									[X]		

Knowledge Categories Must Be Entered In Numerical Order

Col. No.	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
Content	[X]	[X]		Knowledge ID	Number	Classif (8 digits)	ication	System	→		[X]	Scale Value Scale	18	[X]	[X]	Knowledge ID Number		Classi- (8 digits)		
Code	[X]	[X]									[X]			[X]	[X]					

Col. No.	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
Content	fication	System	→	[X]	Scale Value Scale	18	[X]	[X]	Knowledge ID Number	(8 digits)	Classification			System	→		[X]	Scale Value Scale	18	(
Code				[X]			[X]	[X]									[X]			

[X] Indicates blank.

Coded by: ____________ Checked by ____________

1-8

The knowledge category columns must be filled out in order with no blank fields left until after the last knowledge category. This is because the HSMS "MATRIX" program reads a blank knowledge field as the end of the information for the particular task.[1]

The code sheets must be carefully checked at least once by someone other than the original coder. This is especially important in the case of the 8-digit knowledge category codes.

The Use of A Code Book

The code sheets are designed to be self-explanatory. A coder should be able to fill out each sheet by referring to a task's data sheets. However, the organization carrying out task analysis may wish to maintain a Code Book to cover the task data collected. In addition to a listing of task names and code numbers, the user may wish to code job titles, performer, departments, and/or shifts. If the user is part of a central office or a consortium arrangement, it may be necessary to give each individual institution a code number.

The following are rules for assigning code numbers:

1. Code numbers are assigned in consecutive order as new information is entered.
2. Each task, regardless of performer or institution, will have been assigned a unique Task Code Number. All overlap tasks should have the same Task Code Number. (The data for a given task enter HSMS sta-

---

[1] This means that if any categories are to be deleted after keypunching, this cannot be done merely by erasing the data. Any category that is eliminated must be replaced. It is easiest to replace the eliminated category by the last one entered on the last card, and then erasing the last entry. If an entire card is eliminated, the count in columns 29-30 on card T00 must be changed.

tistical analysis only once; i.e., a code number is included only once.)

3. Each job title, regardless of department or institution, is assigned a unique job title code number. All titles which are the same will have the same title code number.

4. Each institution is assigned a unique institution code number.

5. Each department name, regardless of institution, is assigned a unique department code number. All department names which are the same will have the same department code number.

6. Performers are assigned code numbers separately for each institution, starting from 001 in each case. A performer's code is preceded by the institution's code, and so a unique number results.

7. Coding for all cards: Identification Information.

Columns	Code	Instructions
1	T	Always a T to designate task data.
1-2	00 to 99	Card number as appropriate.
5-10	000001 to 999999	Task Code (Identification) Number. Right justify in field.
12-14	001 to 999	Job Title Code. Optional. Right justify.
16	1 to 9	Institutional Code. Optional.
18-20	001 to 999	Performer Code. Optional. Right justify.
21-22	01 to 99	Department Code. Optional. Right justify.
23	1 to 9	Shift Code. Optional.
25	___	Type of task as per designations used for analysis. HSMS uses "N" for normative tasks. Optional.
27	___	Frequency scale value for given performer or job title. Optional.
80	___	Special designation. Optional.

8. Coding unique to Card T00: Identification of task and card.

Columns	Code	Instructions
29-30	01 to 99	Number of cards to follow. Card T01 is 01; add all the cards needed for all the knowledge categories (4 to a card), from T02, on. Same as columns 2-3 on the task's last card. Right justify.
32-78	--	Abbreviated name of the task. Use words and abbreviations that unambiguously refer to the task's abbreviated task name on the Task Description Sheet or Summary Sheet.

9. Coding unique to Card T01: Skill Scales.[2]

Columns	Skill Scale
32-33	Scale value for Locomotion (Scale 2).
35-36	Scale value for Object Manipulation (Scale 3).
38-39	Scale value for Guiding or Steering (Scale 4).
41-42	Scale value for Human Interaction (Scale 5).
44-45	Scale value for Leadership (Scale 6).
47-48	Scale value for Oral Use of a Relevant Language (Scale 7).
50-51	Scale value for Reading Use of a Relevant Language (Scale 8).
53-54	Scale value for Written use of a Relevant Language (Scale 9).
56-57	Scale value for Decision Making on Methods (Scale 10).
59-60	Scale value for Decision Making on Quality (Scale 11).
62-63	Scale value for Figural Skills (Scale 12).
65-66	Scale value for Symbolic Skills (Scale 13).
68-69	Scale value for Taxonomic Skills (Scale 14).
71-72	Scale value for Implicative Skills (Scale 15).
74-75	Scale value for Financial Consequences of Error (Scale 16).
77-78	Scale value for Consequences of Error to Humans (Scale 17).

[2] Code scale value for each skill scale without decimal point in the two-column field designated. Code all zeroes; code a zero scale value as 00.

10. Coding unique to Card T02 and Higher: Knowledge Data.

Columns	Code	Instructions
30-37	8-digit code	First Knowledge Classification System category code.
39-40	--	Nonzero scale value for Levels of Knowledge (Scale 18) for preceding category, without decimal.
43-50	8-digit code	Next Knowledge Classification System category code.
52-53	--	Nonzero scale value for Levels of Knowledge (Scale 18) for preceding category, without decimal.
56-63	8-digit code	Next Knowledge Classification System category code.
65-66	--	Nonzero scale value for Levels of Knowledge (Scale 18) for preceding category, without decimal.
69-76	8-digit code	Next Knowledge Classification System category code.
78-79	--	Nonzero scale value for Levels of Knowledge (Scale 18) for preceding category, without decimal.

Keypunching

It is most efficient to arrange the code sheets in three groups:

1. A set of T00 sheets in numerical order by Task Code Number.

2. A set of T01 sheets in numerical order by Task Code Number.

3. A set of T02 and higher sheets in numerical order by Task Code Number, and within that, in numerical order by Code Number.

This arrangement makes it possible to use drum set-ups for keypunching, and provides listings which can be inspected visually for illegal punches in blank fields. The cards should be keypunched, verified, and listed. They should then be proofed against the original data sheets. This provides a double check against errors picked up in coding.

Once this is done, the task data card decks should be structured so that the cards appear in numerical order by Task Code Number, and within each task, in numerical card number order. At this point the task data are ready for submission to the computer for their first computer checks and analyses.

USE OF THE "EDIT" PROGRAM TO CHECK AND SELECT DATA

HSMS employes a multi-purpose computer program, "EDIT," which prepares the task data for use with its other analytic and statistical programs. "EDIT" is first used to check that the data cards are arranged properly for submission; it also performs several error checks.[3] We generally submit all the data cards to a check by EDIT before putting the data on magnetic tape in permanent form.

Checks

We have EDIT carry out a series of checks in the first computer run with a set of data by selecting the proper options in the EDIT program. The following are the checks carried out:

---

3 For a detailed description of EDIT, instructions for use, and a listing of the program, see Appendix A.

1. A check that each task (Task Code Number) appears only once in the set of data.

2. A check that the number of cards for a task is consistent with the number indicated on its card T00.

3. A check that all the cards for a task have the same Task Code Number.

4. A check that data cards appear in proper sequence (as indicated in columns 1-3).

5. A check that punches occur only in permissible columns.

6. A check that scale values all end in 0 or 5.

7. A check that all knowledge categories have a scale value above 00.

8. A check that knowledge categories appear only once for a given task.

The first EDIT submission is usually run to include an optional listing of the data cards. Thus, when there is an error message relating to the checks, or if the analyst finds an error, it is possible to refer to the data listing to find the problem. Having the data in card image form makes it relatively easy to plan and make corrections.

One of the functions of EDIT is to create a matrix of tasks by skill and knowledge categories; EDIT enters zeroes in this matrix whenever a category in the data base is not required by a task.

EDIT then orders the data base in successive matrix arrangements and lists the resulting information in various predetermined formats. In the first run with a set of data, some of the format listings are used by staff to check for "legal" and correct use of knowledge categories. The four format listings of data are as follows:

1. EDIT gives each skill and knowledge category an internal number. These are listed in the order in which they appear in the input (data card) file, together with the number of tasks in which each appears (frequency), and the Task Code Number and scale value of each task in which the variable is scaled above 00.

2. EDIT lists the information described in 1, above, in descending order of the skill or knowledge category frequency of occurrence in the tasks. The original internal numbers, the Task Code Numbers, and the scale values appear again in order of frequency.

3. EDIT again lists the skill and knowledge information described above, in order by skills first (in a preset order), followed by the knowledge categories in ascending numerical order of their 8-digit code numbers. The original internal numbers, frequency, Task Code Numbers and scale values appear again in this third order of presentation.

4. EDIT renumbers the skill and knowledge categories (variables) internally and lists these in order as in the third listing, with skills first, followed by the knowledge categories in ascending order of their 8-digit code numbers. The fourth listing omits the Task Code Numbers and scale values.

The fourth listing, described above, is then used as follows:

1. The person doing the checking should be someone familiar with the data and the knowledge identification and scaling.

2. The fourth listing is fairly compressed. The person who makes the check sets up a three column table by using this print-out. The skill codes and the knowledge category codes of the print-out become the left-hand column of the table.

   The checker then refers to the Knowledge Classification System and enters the abbreviated name of the category next to its code number in the right-hand column. The third column is the listing of each category's frequency of occurence across tasks.

3. A checker who has worked with the data can usually spot any inappropriate categories caused by coding or keypunching errors. When an error is found, the checker uses the third listing to find the tasks in

which the error appears. Then the data card listings are used to find the data errors and plan for the corrections.

4. An error is usually remedied by correcting a category code number. If an illegal number has been used it may have to be eliminated entirely. The user should remember that, in making corrections, an illegal knowledge category code and scale value cannot be removed by leaving blank fields (except for the last knowledge category for a task). Each must be replaced. See footnote 3.

5. After the errors have been corrected the EDIT run is repeated and the data are entered on magnetic tape.

Using EDIT to Select Variables

Once the data are corrected, the EDIT program is used again to arrange the raw data matrix for further analysis. In the data matrix created, the rows are the tasks; the columns are the skill and knowledge variables; and the entries are the scale values. The HSMS purpose is to go from this basic data matrix to the assignment of tasks to job levels and to recommendations on job ladders. The essential problem is to group a large number of tasks that require a large number of skill and knowledge variables at varying scale values into a meaningful smaller number of groupings so that the underlying association of skills and knowledges (variance) will be reflected in the groupings of the tasks.

To solve the problem HSMS employs a form of factor analysis called "principal components analysis" in a procedure we call "two-mode" factor analysis. It is used to assign tasks to groupings that require related skills and knowledges. The word "factor" means grouping. (The way skills and knowledge categories group together in factors determines the way tasks can be grouped together in factors.)

The HSMS method requires the use of several computer programs. Before they can be used it is necessary to make sure that each particular "run" with its tasks and skill and knowledge variables conforms to general statistical requirements, i.e., that there are substantially more tasks than variables, that variables with very low frequencies across tasks are eliminated, and that there are no more than 144 variables (the current limit of the HSMS "two-mode" programs). This requires selection.

The selection stage is composed of two parts:

1. Selection of the tasks to enter the statistical analysis for given computer "runs."
2. Selection of the skill and knowledge variables that have sufficient frequency to enter into the factor analysis for each run.

The first selection step is the choice of tasks for analysis. The user may be interested in the interrelationships among a large number of tasks, such as those covering one or more departments or services, and may also wish to examine the factor structure of a subset of tasks. If this is the case, the analysis is carried out separately for each set of data; when the input decks are structured for computer submissions the "runs" are given separate designations to avoid confusion.

After the error check, EDIT is used by itself to obtain corrected listings for each "run." The listings are then used to aid the user in deciding which variables to eliminate so that the number of skill and knolwedge variables is an acceptable quantity, based on program limits and/or statistical requirements.

The user works with the second listing, described earlier, which lists the variables in descending order of frequency. The table used for checking, described earlier, can be redone and used as a reference so that the user can consider the identity of the knowledge categories (which are listed by code number).

Working with the second EDIT listing, the user numbers the skill and knowledge variables from 1 to 144, in descending order of frequency and notes the frequency of the 144th variable. Generally, variables with lower frequencies are eliminated to bring the total to 144.

Even if the original number of variables is 144 or less, it is nonetheless advisable for statistical reasons to use a frequency cutoff of at least 4. Retaining variables with frequencies lower than 5 distorts the results as a consequence of the high number of zero scale values that appear in the raw data matrix. HSMS selects a minimum cutoff option of 4; the total number of variables is then determined by the number of variables retained after the cutoff is used.

If the 144th variable falls at a relatively high frequency, and the selection of this frequency as a cutoff figure will eliminate more variables than the number needed to arrive at 144, it is desirable to select a lower frequency cutoff. With large numbers of tasks in a run it is desirable to retain a full set of 144 variables when these all have relatively high frequencies.

HSMS usually selects a cutoff figure which results in several more than 144 variables; we then selectively eliminate additional variables to bring the total to 144. We may eliminate either broad or fine-level categories if both are identified for a given run. If the technologist level is of particular interest, we may eliminate some categories found only at the professional level. The selective elimination of variables permits the user to retain relevant data based on the needs of the particular analysis.

We record the cutoff and deletion decisions for each run. EDIT is then run with the statistical programs, as described in the next chapter. The EDIT listings described earlier are repeated in the subsequent print-outs, and have the following characteristics:

1. The fourth EDIT listing presents the variables selected; they are renumbered internally and listed in order with the skills first, followed by the knowledge categories in ascending order of their 8-digit code numbers. Frequency data are listed, but not task and scale value data. When DELETE and CUTOFF options have been selected, this listing does not include the eliminated categories.[4] (In the three prior listings the categories eliminated as a result of DELETE or CUTOFF options are listed and marked by asterisks.) This listing serves as a "Variable Description Dictionary."

---

[4] DELETE and CUTOFF are the EDIT options used to reduce the number of skill and knowledge variables which are copied to tape. DELETE is used to name specific skill and knowledge categories to be eliminated; CUTOFF is used to specify the frequency at or below which variables are automatically eliminated.

Eliminating variables with low frequencies for factor analysis does not mean the loss of information about such variables. Later in the analysis, when tasks are arranged in order of difficulty for assignment to job levels, and in the design of curricula, all the data are dealt with. At this stage variables of low frequency do not provide information for grouping tasks, and their elimination is not a loss.

2. EDIT provides a "Task Description Dictionary" which lists each task's internal (computer) number in numerical order and its Task Code Number, together with the abbreviated name of the task, as found on card T00 for the task.

3. The final (internal) numbers assigned to skill and knowledge variables and to tasks are the numerical references for observations (tasks) and variables (skill and knowledge categories) when PCVARIM, X2MOFA, and X2MFA2 (the HSMS statistical programs) are used.

Using EDIT to Prepare Variables for Analysis

When EDIT is run with the statistical programs, one additional option is used to prepare the variables for statistical analysis. The EDIT program performs a logarithmic transformation on the data to bring them to a closer approximation of linearity among variables.[5]

Thus, at this stage, there is a reduced data matrix for each "run" whose rows are the tasks selected for the given run, and whose columns are the 144 or less variables selected for that run; the entries are the scale values adjusted by EDIT.

---

5 The NORMALIZE option permits a logarithmic transformation of the data to adjust for a large number of zeroes in the data base. (Tasks which do not require knowledge categories required by any other task are scaled at zero.) The NORMALIZE option follows the formula:

X = SQRT (X + 0 5), where X is a scale value.

CHAPTER 2

ANALYTIC TECHNIQUES FOR GROUPING TASKS

This chapter describes the use of three computer programs to group HSMS tasks into interrelated families of tasks based on the skill and knowledge scale data. The first section presents a general discussion of the HSMS method; the second section describes the use of the HSMS PCVARIM program for grouping variables; and the third section describes the use of the "two-mode" factor analysis programs for grouping tasks. The two latter sections serve as a method manual. Details about the three programs, instructions for computer submissions, and program listings appear in Appendixes C, D, and E.

OVERVIEW

In order to arrive at a grouping of tasks, the HSMS method uses statistical procedures and techniques which are also used by scientists engaged in building predictive models. However, the HSMS use of these analytic techniques is a descriptive, applied use of statistics. The techniques serve only to organize and simplify the data; the results are suggestive, not predictive. The overall method for grouping tasks is as follows:

1. The basic data are the skill and knowledge scale values assigned to tasks. The "variables" are the skills and the knowledge categories. The "observations" or "subjects" are the tasks.

2. Principal component factor analysis is used to examine the factor structure (relationships) among the variables. A solution (number of factors) is selected which best describes the relationships among variables. This becomes the basis for grouping tasks.

3. "Two-mode" factor analysis is used to examine the factor structure of tasks based on the factor structure of variables and to assign each task to a factor.

4. The factors are named for the skill, knowledge, and work content that characterizes the variables or tasks that are assigned to factors.

5. The task factors are the basis for structuring jobs, designing job ladders, and designing curricula, because they represent groupings of tasks which have much in common with respect to skill and knowledge requirements.

Factor analysis is an analytic technique that is used when the statistical relationships among a large number of variables are of interest. The object is to replace the separate relationships of each variable with every other variable with a smaller number of interrelated variable groups (factors). Each factor is essentially a construct that expresses the interrelationships within a particular group.

The initial factor analysis technique used by HSMS is called "principal components analysis." It examines the correlation of each variable with every other variable across a given set of observations (tasks) and groups these to best account for all the variability among the variables. Using various criteria, the analyst selects a factor solution, i.e., the number of factors in which to group the variables. A factor accounts for the variance among several variables in a test space that is analagous to the way a regression line accounts for the variance between two variables in two-dimensional space.

The factor analysis program which provides the solutions for grouping the HSMS variables is called PCVARIM, an abbreviation for Principal Components Factor Analysis with Varimax Rotation. The "two-mode" factor analysis programs which produce the HSMS "task factors" are called Two-Mode Factor Analysis Part One (X2MOFA) and Part Two (X2MFA2).

To arrive at principal axis (PA) factors, the PCVARIM program and the two-mode programs use a principal components technique, with unities in the diagonal rather than communality estimates. HSMS uses a correlation matrix of variables (rather than a covariance or cross-products matrix), and varimax rotation of the PA factors, which produces an orthogonal (rather than oblique) factor solution.[1]

Unlike other factor analytic techniques which first reduce the total variability in a test space, the principal components technique summarizes the total variability in a test space into a smaller number of orthogonal components. The factors produced are maximally independent of one another.

In the HSMS two-mode programs the reduced data matrix is used to form two conceptually different but necessarily related correlation matrixes. One is the correlation of every variable with every other variable across all the tasks (as in the PCVARIM program); the other is the correlation of every task with every other task across all the variables.

The two-mode programs produce the principal components of the variable matrix as in the PCVARIM solution; in addition, the

---

[1] The use of correlation matrixes provides solutions that are not dependent on the standard deviations of the variables, as covariance matrixes are, nor on the means and standard deviations, as cross-product matrixes are.

programs produce a transfer of the principal components of the task correlation matrix based on the Eckert-Young theorem. This produces the task factors which reflect the variable factors.[2]

SELECTING THE VARIABLE FACTORS

HSMS uses the PCVARIM program to select the "solution" (number of factors) which best groups the skill and knowledge variables. PCVARIM is used with EDIT, as described in Appendix C. EDIT selects and lists the variables in the order to be used in the PCVARIM program, and "normalizes" the data.

PCVARIM makes it possible to examine a variety of factor solutions. Solutions with as many as twelve factors or as few as two factors may be examined, as well as all the others in between. An EDIT print-out can be ordered. The PCVARIM print-out includes means and standard deviations, the correlation matrix, and other information such as Eigenvalues, principal axis factors, and communalities.

Of chief interest are the "Varimax Factor Loadings." These are presented in arrays, starting with the largest factor solution requested, and ending with the solution containing the smallest number of factors requested. The user examines each factor solution and decides on the one that seems most appropriate.

---

2 After the "variable mode" is rotated to simple structure, the "task mode" is "counter rotated" by obtaining the transformed characteristic vectors of the observation mode induced by the varimax rotation of the variable mode following the Eckert-Young Theorem. See Appendix G for a discussion of the procedures and Appendixes D and E for the programs.

For each "solution" requested there is an array arranged as a matrix, in which the columns are numbered and stand for the factors in the given solution. The rows are numbered and stand for the variables, in the order determined by EDIT. At the bottom of each array are data on the variance accounted for by each factor.

The entries in the arrays are the varimax factor loadings. These are decimal numbers, of which none can be greater than .999. Every variable has a loading on every factor in a solution. Variables can load on factors within the range of ±.999. Variables which are positively interrelated on a factor will have the same sign. The + or - sign has no other intrinsic meaning. A loading of ±.400 or more is of interest. For a four-factor solution, the first five variables and the end of the array might appear as follows:

VAR. NO.	VARIMAX FACTOR LOADINGS 1	2	3	4
1	.429 x	.257	-.009	-.246
2	.578 x	.139	-.622 x	.039
3	.635 x	.360	-.218	-.113
4	-.054	-.034	.023	-.220
5	-.006	-.072	-.909 x	.015
etc.	etc.	etc.	etc.	etc.
VARIANCE	18.207	10.037	8.547	5.876
PCT VAR	.207	.114	.097	.067
CUM PCT	.207	.321	.418	.485

Note: An x denotes a high-loading variable and is entered by the analyst during inspection of the print-out.

The loadings reflect the extent to which a variable's scale value variations contribute to the variance accounted for by the factor in the given solution. In examining several factor solutions (for example, ten factors, nine factors, and so on, down to three factors), the analyst notes which variables have "high loadings" (.450 or higher, independent of sign) on each factor in each solution. These variables "determine" a factor; their identities give some sense of the underlying meaning of the factor. In the case of HSMS data, a factor may suggest the skills and knowledge categories needed for a work function, a specialty, a type of service, or a type of procedure.

The choice of an acceptable factor solution (that is, the choice of five factors rather than four or six factors) can be based on statistical criteria, on common sense, or on a combination of these.

HSMS eliminates solutions which have any factors with less than three variables with relatively high loadings (i.e., ±.450 or more), and solutions in which no underlying structure of interest is evident. The choice is limited to solutions which make sense, those which account for most variables, and in which few variables have high loadings on several factors. We look for stability of factor structures across several factor solutions, and choose that solution whose underlying structure is most easily understood in terms of what we know about the nature of the work being studied.

The specific analytic work with the PCVARIM output is as follows:

1. For each factor solution array, the analyst places a check mark, x, or some other indicator next to each loading at $\pm$.450 or higher.

2. The analyst eliminates all solutions in which one or more factors (columns) have no loadings of $\pm$.450 or higher.

3. The analyst prepares analysis sheets for the remaining solutions:

   a. For each factor in a solution the analyst lists the variables that load at $\pm$.450 or higher. This is done by listing the internal number of the variable as shown in the array and then writing in the name of the skill or the code number and the name of the knowledge category.

   b. The name of a skill or knowledge category is obtained by translating the PCVARIM variable number into the skill code or knowledge number using the fourth EDIT listing: "Final Correspondence of Variable Numbers to Variable Codes as Written on TAPE9." The Knowledge Classification System name is then entered next to its code number.

   c. A single master dictionary can then be made and duplicated; it can be cut up and used for all the analysis sheets.

4. The analyst examines the analysis sheets for each factor solution, and considers the "sense" of the skill and knowledge categories that load high on each factor in the solution.

   a. The analyst notes how the factors appear and are differentiated as the solutions change from the smallest number of factors to the largest number of factors being considered. The analyst notes what new specialties or functions are represented or expressed as new factors appear, and notes which factors seem to be stable from solution to solution.

   b. The analyst considers the number of variables that load high on several factors in a solution and the number of variables that do not load high on any factors. These should both be at a minimum.

   c. The analyst notes any variables that load high with a sign opposite the one that is characteristic of the other high-loading variables. A solution with such inverse loadings is difficult to interpret.

d. The analyst eliminates factor solutions with only two or three high-loading variables in a factor unless some specialty of interest is expressed by the factor.

5. The analyst selects the factor solution that makes the most sense, accounts for a substantial number of variables, has a minimum number of variables that determine more than one factor, and shows relatively stable factor structures.

   a. The number of factors (i.e., the solution) is recorded.

   b. The characteristic sign of each factor, i.e., whether the high-loading variables have plus signs (+) or minus signs (-) is recorded.

   c. The analyst tentatively names each factor based on what the association of skill and knowledge categories suggests with respect to work content or function.

6. In reporting the factor structure of variables, HSMS includes loadings of ±.40 or more. High loadings are reported as positive, regardless of the characteristic sign of the factor, except for inverse loadings, which are reported as negative loadings. This is easier for the reader to interpret. Figure 4 presents an example of how the factor structure of variables can be presented.

OBTAINING TASK FACTORS

In the HSMS method, two-mode factor analysis follows after the selection of a skill-and-knowledge variable factor solution. This determines the number of factors and the nature of the factors to which tasks will be assigned. The HSMS "two-mode" factor analysis programs are based on a procedure for factoring an individual differences matrix to obtain idealized subject types. The HSMS subject types are task groupings; the variables are the scaled skill and knowledge categories.

Figure 4. EXAMPLE OF TABLE SHOWING FACTOR STRUCTURE OF VARIABLES

SKILLS AND KNOWLEDGES IN RADIOLOGIC TECHNOLOGY, ULTRASOUND TECHNOLOGY, RADIATION THERAPY TECHNOLOGY AND RELATED FUNCTIONS[a]: FACTOR STRUCTURE OF VARIABLES

Skill, or Knowledge Category Number and Abbreviated Name[b]	Factor Loadings[c] Rad. Tech.	Ultrasd. Tech.	Radther. Tech.	Qual.Ass. Materials	Patient Care
Object Manipulation Skills	.43				
Guiding or Steering Skills	.57	.64			
Human Interaction Skills	.60				
Leadership Skills					
Oral Use of a Relevant Language	.62				
Reading Use of a Relevant Language	.61			.41	
Written Use of a Relevant Language	.47	.41		.51	
Decision Making on Methods	.40			.50	
Decision Making on Quality					
Figural Skills	.73				
Symbolic Skills	.64			.54	
Taxonomic Skills	.76				
Implicative Skills	.65	.41			
Financial Consequences of Error				.61	
Consequences of Error to Humans	.67				
11731000 Normal structure and function	.43	.42	.56		
11731100 Regional anatomy	.77				
11731200 Topographic anatomy	.80				
11731300 Hematopoietic system[d]					
11731400 Circulatory system					

[a] Refers to analysis of 296 tasks listed in Table 1 by abbreviated task names, and 127 skill and knowledge variables; 88 variables were included in factor analysis. See (d), below.

[b] See Table 9, Appendix D, for full names of knowledge categories and Appendix C for skill scales.

[c] Loadings of ± .40 or more are shown; blanks indicate lower loadings.

[d] Indicates category not included in factor analysis because frequency across tasks is less than 5.

"Part One" of the two-mode program (X2MOFA) is run interfaced with EDIT, as is the case with PCVARIM. (See Appendix D.) The output of Part One provides card inputs for "Part Two" (X2MFA2). Part Two is described in Appendix E.

The tasks and the skill and knowledge variables are represented in the X2MOFA and X2MFA2 print-outs in the order assigned by the EDIT program. Therefore, the EDIT "dictionaries" are used to interpret the two-mode outputs with the PCVARIM outputs.

To obtain task factors with the two-mode method we extract and rotate the number of variable factors chosen in the prior PCVARIM analysis. After counter rotation of the task mode, the task factors are shown in an array in which the factors are the columns and the tasks are the rows. The entries are the tasks' numerical loadings on the factors. Each task has a loading on each factor. To interpret the data one must understand what a task factor represents, the meaning of the loadings, and how a task's loading on a factor arises.

For every variable factor in the solution chosen there is a corresponding task factor. The skill and knowledge variables that determine a variable factor determine how tasks will load on the corresponding task factor. A task's loading on a factor reflects the skills and knowledge categories required for the task, the scale values at which they are required, and the loading of those particular variables on the corresponding variable factor.

For any given task, the more skill and knowledge categories it requires of those variables that determine a given variable factor, the higher the task's loading will be on the corresponding task factor. The higher the task's scale values for those variables, the higher the task's loading on the factor. (The influence of each variable can be estimated by noting the variable's loading on the variable factor.) Since a task will have some numerical loading on all task factors, a given task can load low or negatively (inversely) on factors that are determined by skill or knowledge categories not required for the task.

The characteristic sign of the variable factor determines the sign of the task factor. Within this, a task's loading can be greater than an integer and can range anywhere down to zero, through zero, to the range of values of the opposite sign. For a four-factor solution, the first five tasks in the array might appear as follows:

COUNTER ROTATION OF SECOND MODE

Task	1 (+)	2 (+)	3 (-)	4 (-)
1	.3757 x	-.0974	.0326	.0215
2	-.1645	-.0751	.0832	-.0347 x
3	.0057	.4333 x	.0287	.0405
4	.0230	.0065	-1.0979 x	.0082
5	.1692	.0600	-.0688	-.8189 x
etc.	etc.	etc.	etc.	etc.

Note: An x denotes the assignment of a task to a factor. Plus (+) and minus (-) signs at the top of each column denote the characteristic sign of the factor. These are entered by the analyst during inspection of the print-out.

The higher the numerical value of a loading that has a sign opposite the characteristic sign of the factor, the lower the value of the loading on the factor.

It is possible to examine a task's loading on all the factors and assign it to the factor on which it has its highest loading (within sign). A task loads highest on the factor with which it has most in common in terms of skills and knowledges.

Low-level tasks which require few skills and knowledge categories and low scale values will have low and/or inverse sign loadings on all factors. The differences in loadings on factors for low-level tasks is so insignificant, that a common-sense assignment of such tasks to factors is often preferable to a mechanical statistical rule for assignment of tasks to factors.

When low-level tasks such as Task 2 in the example listed above have ambiguous loadings, HSMS assigns each to a factor which includes other tasks with similar requirements and with similar loadings in a context where the task makes sense. Generally we used a people-oriented factor and a materials-oriented factor for such assignments.

The specific analytic work with the Two-Mode Part Two output is as follows:

1. The print-out of Two Mode Factor Analysis Part Two (X2MFA2) is used in the assignment of tasks to factors. The relevant portion of the print-out is the array entitled, "Counter Rotation of --- Variable Mode." A number is included in the space indicated in this heading title which corresponds to the number of tasks in the given run.[3]

---

[3] The words "counter rotation" and this number differentiate this array from the "Rotation of the Variable Mode" which corresponds to the array for the first mode, i.e., the skill and knowledge category mode, which is "rotated." For HSMS purposes, the counter rotated mode, the second mode, is the task mode.

2. The analyst obtains the "Task Description Dictionary" print-out which is part of the EDIT output. This lists the internal number of each task, each task's Code Number, and the abbreviated task name. These are cut and mounted on the print-out of the Counter Rotation Mode array so that the internal number of the task on the far left of the array corresponds to the internal numbering of the task in the EDIT dictionary. The result is a table in which one can read the identity of every task in the analysis and its loading on every task factor.

3. The analysts enters the characteristic sign and factor number of each factor at the top of each page of the array. (The number is listed only on the first page.) The sign is obtained by referring to the counterpart skill and knowledge variable factor.

4. The analyst reads across each row and marks the task's highest loading (within sign) in the row. In the example listed above the highest loading within sign for Task 2 is on Factor 4 (-.0347), because the other loadings, though higher numerically, ahve a sign opposite from the characteristic sign of the factor.

5. When all the tasks have been assigned to a factor the analyst reads the names of the tasks assigned to each factor. The work content of the factors should now emerge.

   a. A number of low-level tasks will seem inappropriate as assigned. These can be reassigned based on logic, as described earlier.

   b. A number of tasks may seem to be related to each other and inappropriate on any of the factors. These may reflect a relationship to a factor that did not emerge due to insufficient frequency of variables in the data base. These can be assigned to a separate non-factor grouping.

   c. A number of tasks may have very high loadings on two factors. These should be assigned in a manner that reflects the nature of the majority of tasks that load unambiguously on the factors.

   d. If there are any remaining ambiguities, final assignments can be made after the job level of tasks has been evaluated, as described in Chapter 3.

6. A list of tasks tentatively assigned to each factor should be prepared, listed in order by Task Code Number, with the factor loading included. In this listing and in subsequent reports we use the convention of presenting the loadings with positive characteristic signs; we use the negative sign to represent inverse loadings; thus, loadings with inverse signs are correctly interpreted as less than zero.

7. The factors are given final name and number designations which reflect the work specialty of the tasks that load on the factor. This may be an occupational reference and/or a functional reference.

It is possible for the analysis conducted up to this point to prove to be disappointing; i.e., the task factors may not "make sense" or may not be sufficiently differentiated to be useful. At this time it is in order to reconsider the PCVARIM factor solutions. In HSMS experience, the PCVARIM solutions sometimes make more sense after the consequences for task factors are seen. It then may be obvious that a larger or smaller number of variable factors can provide more interpretable or useful task factors. After a new factor solution is chosen, the two-mode programs are rerun, and the analysis steps are repeated.

It is the strength of the HSMS method that one is not "stuck" with any given solution. The one that most clearly accounts for the data and serves the ultimate purpose of the analysis is the best solution; and it is never too late to reconsider which solution is most appropriate to the needs of the user after trying several solutions.

CHAPTER 3

JOB STRUCTURES AND JOB LADDERS

This chapter describes the use of the HSMS MATRIX program for assigning tasks to job levels.[1] It is a manual for the structuring of jobs and the creation of job ladders based on the task factors and the job level analysis. The first section explains how MATRIX is used to assign an index of "difficulty" to tasks within factors based on skill and knowledge requirements. The second section describes the assignment of tasks to job levels. The third section discusses the design of job sequences or ladders and presents examples of HSMS recommendations. The sequence of events is as follows:

1. The tasks are arranged in rank order within factors by "point score" order of "difficulty," based on a count of all the skills and knowledges required for each task and the scale values at which each is required (including variables not part of the factor analysis).

2. Tasks are assigned to job levels within each factor based on their "point scores" and "profiles."

3. Once the tasks are assigned to job levels within factors the results are examined and job ladder and lattice recommendations are made.

DETERMINING RELATIVE TASK DIFFICULTY: POINT SCORES

The hierarchy of tasks in a factor is the arrangement of tasks by order of difficulty so that tasks can be assigned to job levels. The MATRIX program is used for this purpose.

---

[1] See Appendix F for a description and listing of MATRIX.

Factor loadings take account of the skill and knowledge variables that enter into the factor analysis; they do not reflect all the skill and knowledge requirements of tasks. For this purpose HSMS carries out a "point analysis" in which every skill or knowledge category needed for a task and the scale value at which each is required enters into a "point score" for each task. This is an index of educational requirements, or "difficulty."

The HSMS MATRIX program allows the user to examine tasks separately for each factor or in any grouping of interest. It presents the task data being examined in an array in which the tasks are arranged in columns from left to right in any order selected. The rows are all the skill and knowledge variables, listed from top to bottom in the order in which they appear in the tasks (as arranged from left to right), including <u>all</u> the variables. The entries are the original scale values. Regardless of the order in which the tasks are arranged, by reading across a row one can see the tasks for which a given skill or knowledge category is required, at what scale values.

Within a factor, some skills or knowledge categories are required at the same scale value for all tasks, and some are required at more than one scale value, depending on the tasks. We assign points to the tasks based on what is found in the MATRIX array. The total is the task's point score. This is calculated for all the variables and tasks in a factor or in a selected set of tasks.

The specific analytic work with the MATRIX program to arrive at point scores is as follows:

1. The user works with the tasks in one task factor at a time. If there is interest in comparing point scores across factors in order to decide on a factor solution, or to decide on a factor assignment for given tasks, tasks assigned to more than one factor may be included. A given MATRIX "run" is the set of tasks that are to appear in a given MATRIX array. The resulting point scores are automatically comparable across all the tasks in the "run."

2. For purposes of point score analysis the user enters the tasks in a given "run" in numerical order by Task Code Number. This makes it easy to locate the tasks in the array. Any other order, such as by factor loading, is acceptable.

3. The MATRIX print-out includes the following:

   a. A list of tasks in the array listed by Task Code Number and abbreviated name presented in the order entered. The tasks are given internal numbers, and the last number is the total number of tasks in the array.

   b. The actual MATRIX array. The column headings are the Task Code Numbers in the order requested, arrayed from left to right (from lowest to highest Task Code Number). The far right-hand column identifies the rows, which are the skill or knowledge category codes, listed in the order in which they appear in the tasks as listed. The entries are the scale values with the decimals omitted. (See example under 5, below.)

   c. A list of the skill and knowledge categories represented in the array, with the skills first, followed by the knowledge category code numbers in numerical order. The frequencies across the tasks in the array are also given.

4. When a given set of tasks is larger than the number that can be accommodated on the width of a computer sheet, the MATRIX row array is repeated, and the column headings are continued until all the tasks are listed. The analyst must cut the continuation sheets and mount them carefully to provide one con-

tinuous array. This is easily done by matching lines and headings and using glue or transparent tape.

5. Below is an example of a MATRIX array as described above:

EXAMPLE OF MATRIX ARRAY
(First 7 Tasks in Task Code Order)

TSK NO:	73	74	98	113	116	138	153	ETC	SKILL OR CATEGORY
	50	30	30	30	30	50	50		HUM INTR
	40	40	40	40	40	40	40		ORAL USE
	45		15	45	15	45			METHODS
	70	70		70	35	70	15		QUALITY
	20				10				IMPLICIT
	10	10	10	10	10	10	10		FINC ERR
	10	10	10	10	10	30	30		HUMN ERR
	25	25							12223000
		20	20		20		20		READ USE
			15				15		OBJ MANP
					20				WRIT USE
					25		15		11737300
					15				65620000
							25		11738000
	13	9	7	9	13	11	11		Point Score*

* Entered during analysis.
Figures were calculated for this sample.

Code: Circle = 1 point; Box = 2 points; Triangle = 3 points.

6. The analyst decides on a symbol code or a color code for calculating the point scores. The example above uses a symbol code because this report is printed in black and white. A set of colored felt tip pens is preferable for coding.

7. The analyst works with one row (variable) at a time. In the example above, the first row is HUM INTR (the Human Interaction Skill Scale). The analyst observes the lowest scale value appearing in the row (30). Using the code for the lowest value (1 point = circle), the analyst marks each scale value with the code for one point. Then, using the code for the next higher value (50), the analyst marks each second highest scale value with the code for two points (box). If there were a third value, it would be marked with the code for the next higher value (a triangle). QUALITY has the scale values of 15, 35, and 70, and all three codes.

8. When all the rows have been symbol-coded or circled with color-coded pens, the columns, which represent the tasks, are added. A circle counts as one point, a box as two points, and a triangle as three points, etc. A blank receives no points. The sums are entered at the bottom. These are the point scores for the tasks.

9. The analyst records the task's loading on the factor and its point score using the list of tasks printed as part of the output to MATRIX. This list can be duplicated, cut out, and arranged in point score order.[2]

ASSIGNING TASKS TO JOB LEVELS

The HSMS method assigns tasks to job levels by arranging the tasks in a factor so that tasks which require similar skills and knowledges at similar scale values are assigned to corresponding and appropriate job levels. The point scores are the basis for this assignment of tasks to levels within factors.

To assign tasks to job levels, the MATRIX program is run for each factor with the tasks arranged from left to right in order of their point scores. This ordering of the array produces a stepwise pattern because skills and knowledge categories that appear in low-level tasks continue to appear across the array, and, as higher-level tasks are added, indented new arrays appear for skills and knowledge categories not required for lower-level tasks. (See Figure 6, presented later in this chapter.)

---

[2] The order of point scores roughly approximates the order of factor loadings, but differences are inevitable. This is because variables enter into the point scores that were not among those selected for the factor analysis; and variables with little variance would not have high loadings on the variable factors.

We mark off on each row the first task at which a scale value changes to a higher value, and continue across the row, marking the first appearance of still higher scale values. We examine the array to note at which position (task) large numbers of skill and knowledge categories are required for the first time, and where the scale values first rise. These step-like demarcation points help us to assess when we have a change of job level.

Often there is a dramatic point of change. In cases where the array shows no dramatic breaks we supplement our analysis by determining point score ranges for job levels and by examining the names of the tasks, using common sense to determine the difference between levels.

It is usually possible to compare job levels across factors, and/or to use the point score of a task on several factors as a benchmark when determining job levels. However, this is only true when the knowledge categories involved have been "broken out" to comparable levels of detail. For example, in the case of the professional-level (5) radiation physicist which emerged on a quality control factor, it was impossible to compare point scores with other factors. The knowledge categories required for the physicist are listed in the Knowledge System solely as broad-level subject categories, and are not "broken out" into fine-level categories; as a result the tasks have relatively low point scores.

In assigning tasks to job levels, HSMS uses a specific convention in which all job levels are identified by number, general name, and academic or functional requirements. These are presented in Figure 5, below:

Figure 5. THE HSMS JOB LEVELS

HSMS Job Levels	Academic or Functional Description
8. Specialized Advanced Professional	MD's with specialization such as attending radiologists, internists, surgeons; doctorates and post doctorates with experience.
7. Advanced Professional	MD's in residency or doctoral candidates carrying out advanced functions.
6. Specialized Professional	Masters-level occupational study and experience; supervision of professionals.
5. Professional	At least four years of academic and/or specialized occupational education; may include masters level.
4. Junior Professional; Supervisor	Supervision and/or instruction of students and/or staff at levels 1, 2, and/or 3; may be equivalent to baccalaureate degree-level specialty.
3. Technologist	Specialized technologist education; equivalent to associate degree level.
2. Technician	Specialized technical education; up to, including, and beyond high school, but less than associate degree level.
1. Aide	Entry level; up to and including high school.

Figure 6 is a hypothetical presentation of a MATRIX array in which the tasks are arranged in ascending point score order based on point score analysis. The assignment of tasks to levels is shown, and the stepwise pattern is apparent.

Figure 6. MODEL OF "MATRIX" ARRAY OF SKILLS AND KNOWLEDGES BY TASK AND JOB LEVEL

Skills and Knowledge Categories	FACTOR I LADDER								
	Level 1			Level 2			Level 3		
	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9
Skill 1	1.0	1.0	2.0	2.0	2.0	4.0	4.0	9.0	9.0
* Skill 2	1.0	2.0	2.0	2.0	2.0	2.0	4.0	7.0	9.0
Skill 3		2.0	2.0	4.5	4.5	4.5	2.0	7.0	7.0
* Knowledge 1				1.5	1.5	1.5	7.0	1.5	7.0
Knowledge 2				3.5	3.5	2.5	5.5	2.5	9.0
Skill 4				5.0	5.0	5.0	5.0		5.0
Knowledge 3							5.0	6.0	
* Knowledge 4							3.5	9.0	9.0
Knowledge 5							8.5	8.5	8.5
Knowledge 6							3.5	7.5	7.5
Knowledge 7							7.0	7.0	9.0
Knowledge 8							7.0		8.5
Point Score	2	4	5	10	10	10	19	21	28

* Asterisk denotes variables that determined the factor.

1. Tasks are listed from left to right in ascending order of point score.
2. Skills and knowledge categories are listed from top to bottom in order of appearance in the task array.
3. Tasks are assigned to levels based on increasing numbers of skills and knowledges required and their scale values.
4. Not every skill or knowledge appears in all subsequent, higher-level tasks.
5. Scale values do not necessarily rise from level to level.
6. Scale values may vary within a level.

Code: circle = 1; box = 2; triangle = 3; double circle = 4; double box = 5.

Below is an example of how the task content of factors can be presented by job level, Task Code Number, name, point score, and factor loading:

TASKS ASSIGNED TO QUALITY ASSURANCE, MATERIALS FACTOR BY JOB LEVEL, POINT SCORE AND FACTOR LOADING p. of

Task Code	Abbreviated Task Name and Job Level	Point Score	Factor Loading
	Level 5 (Professional)		
557	Collecting and presenting technical information about and/or recommending new diagnostic x-ray equipment.	56	.79
546	Designing, maintaining, evaluating radiation protection and monitoring programs in diagnostic radiology.	55	.72
	etc.		
	Level 2 (Technician)		
556	Calibrating diagnostic x-ray test, survey, or measuring instruments.	28	.28
533	Checking automatic exposure termination of diagnostic radiography equipment.	27	.07
	etc.		
	Level 1 (Aide)		
192	Inspecting, checking, preparing xeroradiography equipment for use.*	14	-.07

The following explanation is given to the reader in the form of footnotes or notes:

Note: Tasks are arranged in descending order within levels by point score (difficulty as reflected in the number of skill and knowledge categories required and the scale values at which the categories are required). Factor loadings run from high, positive values, through zero, to negative values (for lower-level tasks).

* Each task was assigned to the factor on which it has its highest loading. Task names marked with an asterisk (*) were assigned on the basis of logic.

DESIGNING JOB LADDERS AND LATTICES

The tasks assigned to any given level within a factor are likely to be representative of the central tasks of a job. Naturally, any job will also include certain peripheral tasks not on the factor which reflect the administrative or institutional idiosyncrasies, paper work, conferences, etc., usually associated with any job. In some cases a real job would combine the tasks in more than one factor, such as when an institution is not large enough to differentiate jobs. However, for the purposes of job or educational ladder design, the tasks at a given level within a factor suggest the most rational assignment of major duties to a job, since they require the maximum application of a minimum but adequate educational investment.

The factor results do not necessarily guarantee that job ladders can be designed within task factors. After tasks have been assigned to factors and levels it is possible to find that only one level is represented in a given factor. However, some factors that appear only at higher job levels can be logical higher-level steps for tasks at lower job levels in other factors. This happens when the higher-level factor combines the skill and knowledge requirements of one or more other factors. This becomes the basis for constructing job lattices.

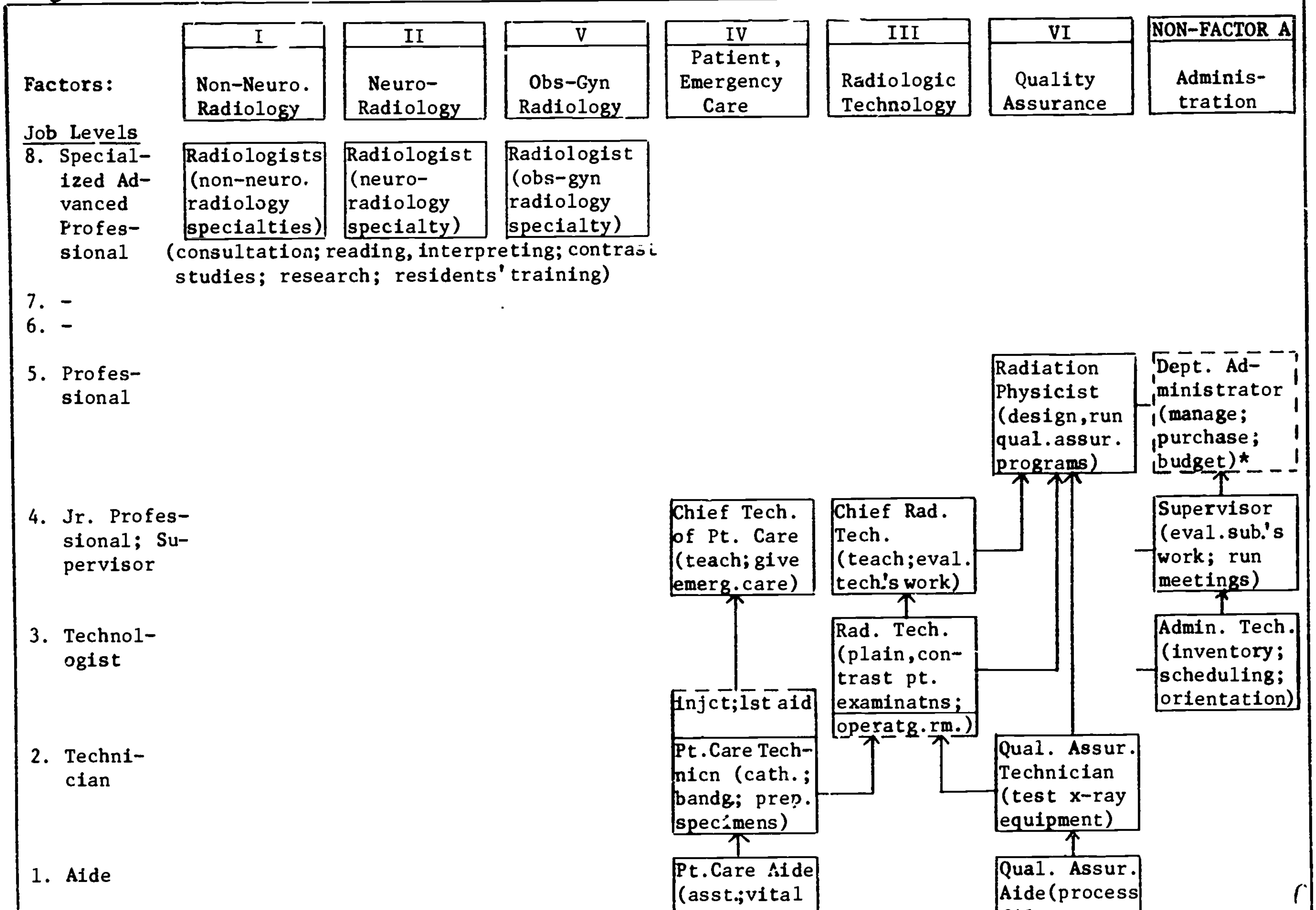
The HSMS job ladder and lattice recommendations are arrangements of jobs in promotional steps derived from the task factors. The jobs in a ladder or lattice require interrelated skill and knowledge categories based on given task factors. Job lattices allow for linkages across factors both horizontally and diagonally. This provides crossover options and a choice of promotional pathways based on the principle that the skills and knowledges required at a given level for a factor may serve as a basis for more than one specialty. A given specialty may build on more than one kind of prior preparation; the entry to specific professional jobs could thus be reached through more than one factor. Conversely, a given job level in a factor can be a step towards more than one specialty.

The analyst makes recommendations on job structures, job ladders, and job lattices based on the nature of the task factors which emerge and on the assignment of tasks to job levels. Once these are laid out and the skill and knowledge content is examined, logical recommendations usually emerge.

Past HSMS analyses are presented in Figures 7, 8, and 9. Figure 7 is an example of the factor structure that emerged in diagnostic radiology; Figure 8 shows the job ladder-lattice recommendations which resulted from the analysis. Figure 9 shows a similar set of recommendations including diagnostic ultrasound and radiation therapy.

These examples are offered as indications of HSMS recommendations, since it is impossible to provide more detailed instructions on how to go from the factor and point score data to the actual suggestions.

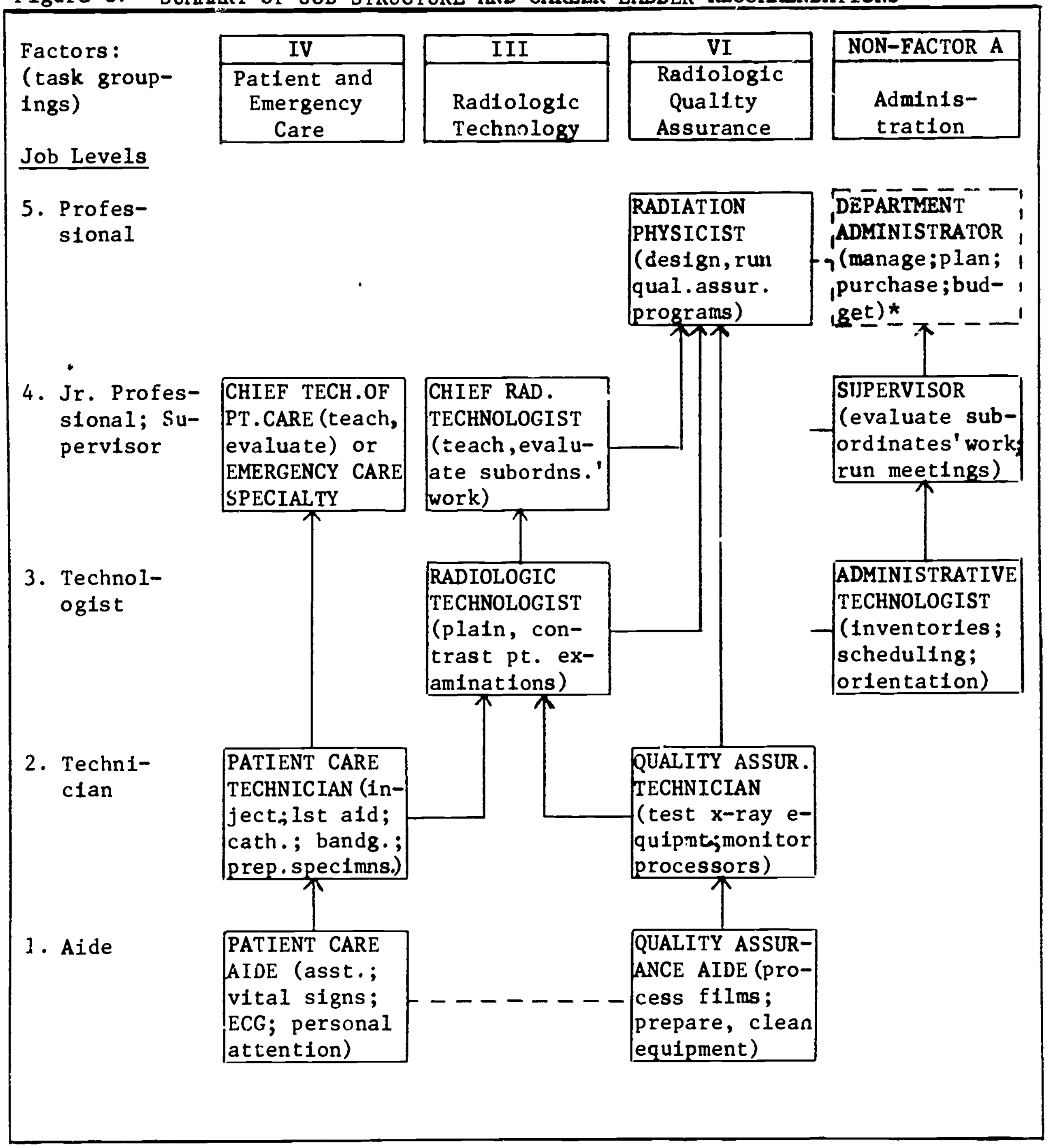
Figure 7. SUMMARY OF FACTOR STRUCTURE OF TASKS BY JOB LEVEL: DIAGNOSTIC RADIOLOGY



*Tasks in box with asterisk (*) not covered by data.

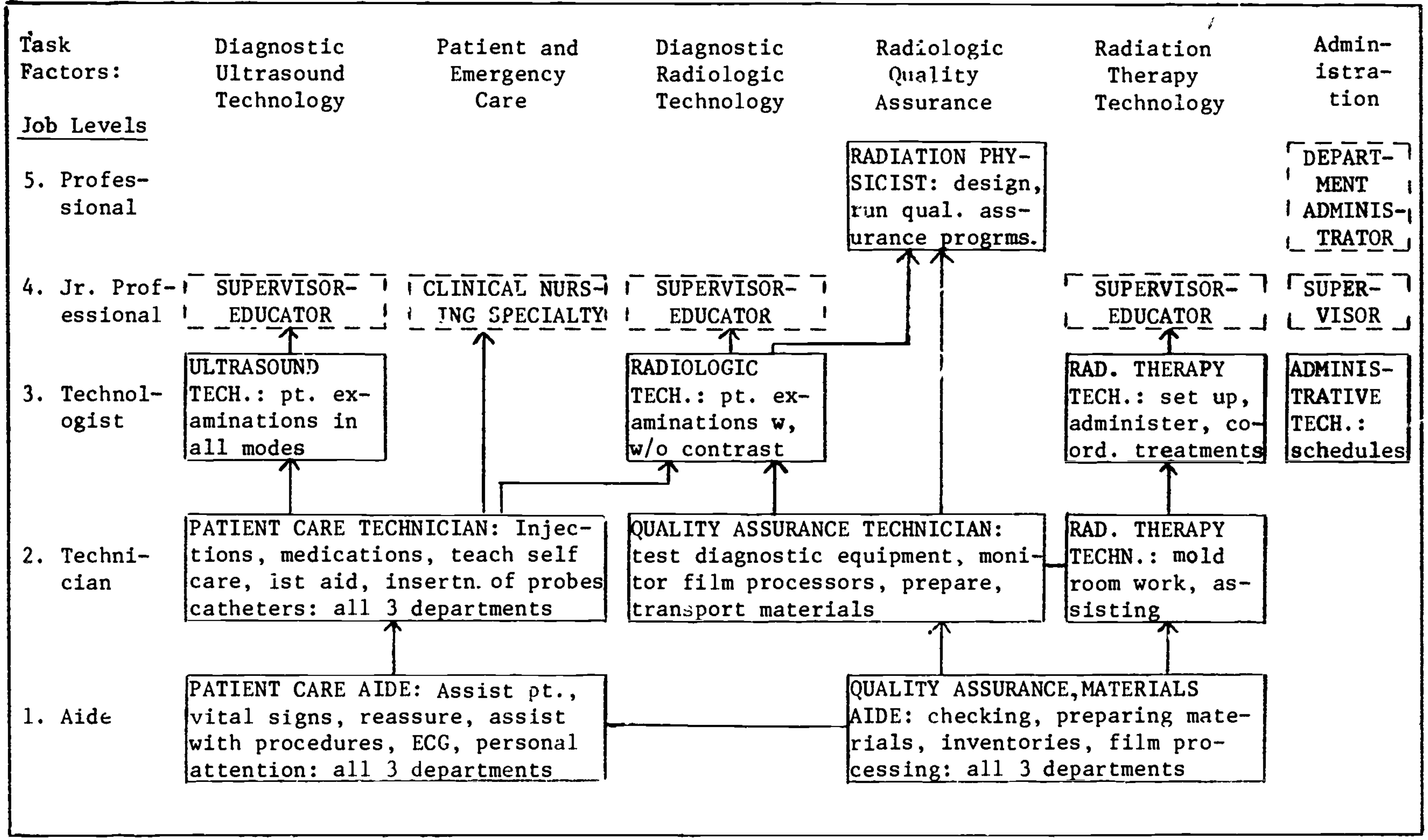
Source: Research Report No. 8.

Figure 8. SUMMARY OF JOB STRUCTURE AND CAREER LADDER RECOMMENDATIONS



* Tasks in box with asterisk (*) not covered by data.
Source: Research Report No. 8.

Figure 9. RADIOLOGIC TECHNOLOGY, ULTRASOUND AND RADIATION THERAPY: RECOMMENDED JOB STRUCTURES AND PROGRESSIONS BY TASK FACTOR AND JOB LEVEL



Broken boxes represent jobs whose tasks are not fully covered by data.
Source: Research Report No. 10.

Once the job structure and job ladder recommendations are made, the user is ready to prepare the data for use in curriculum outlines, curriculum objectives, and educational ladders to parallel the job ladders. This is discussed in Volume 4 of this report. Chapter 4, which follows, describes the use of the data for institutional evaluation and planning.

CHAPTER 4

USING TASK DATA TO MAKE RATIONAL USE OF MANPOWER [1]

Prior chapters described how statistical analysis of HSMS-type task data can be used to develop job structure and job ladder recommendations. This chapter shows the director or administrator of a local institution how to adapt the recommendations, based on nationwide practices, to the needs of an individual institution. The chapter is directed to the hospital or department administrator who wishes to implement job structure and career ladder recommendations resulting from task analysis, or who wishes to use task data for performance evaluation.

An institution cannot be expected to provide upward mobility to its employees unless it has practical reasons for doing so; it must make economic sense. The first section of this chapter argues that there are benefits to be derived from using HSMS-type data and recommendations. It discusses the economic rationales for job structuring, restructuring, and the development of job ladders using HSMS methods.

The second section describes how the administrator can use the data generated by HSMS methods to rationally structure or restruc-

[1] Much of the material presented in this chapter appeared as Chapter 3 in Research Report No. 8. (Eleanor Gilpatrick, Using Task Data in Diagnostic Radiology, Volume 1, "Job Ladders: Assigning Tasks to Jobs," New York: Health Services Mobility Study, 1977.)

ture jobs. It shows how to examine the allocation of tasks to levels, and describes the creation of new jobs. The third section discusses the development of a career ladder program, cost saving strategies, and trainee selection. The fourth section deals with the use of HSMS-type data to evaluate institutional performance. It provides a mini-manual for using HSMS-type task data to create performance evaluation instruments.

RATIONALES FOR JOB RESTRUCTURING AND CAREER LADDERS

The HSMS method makes it possible to analyze tasks in terms of their skill and knowledge requirements and their relationship to other tasks and job levels. It is therefore possible to assign tasks to job titles and to make optimum use of more highly trained and more expensive employees, and to make sure that the work is being carried out by staff who are properly trained to provide quality input.

The assignment of tasks to job titles is job structuring or restructuring. The arrangement of jobs into a promotional sequence from one level to another is job ladder construction. It is not always necessary to do job restructuring in order to design and implement job ladders; it is also possible to derive advantages from job structuring or restructuring without having to arrange jobs into a promotional ladder. This is discussed below.

The costs to consider in structuring or restructuring jobs are those of salary and education. Direct education or tuition costs can be borne by students, employers, or society; education costs, however, are also reflected in salary levels. The education time needed to prepare for jobs is highly correlated with salary levels. Reference to high-level staff or jobs implies high salaries, high skill and knowledge requirements, and long, expensive periods of educational preparation. Conversely, reference to low-level jobs implies low levels of educational preparation and low salary levels.

Job Structuring and Restructuring

Job structuring and restructuring can provide cost advantages if tasks are assigned to jobs so that the skill and knowledge levels required for tasks are compatible with the educational and salary levels of the jobs to which they are assigned. Allocation of low-level tasks to high-level jobs is wasteful of salary and education costs. If there are shortages of high-level personnel, the waste is felt as decreased services or other outputs.

It also makes economic sense to assign tasks to jobs so that the skill and knowledge requirements for all the tasks in a job are similar. Assigning tasks requiring different, non-overlapping skill and knowledge requirements to a single job increases the amount of educational preparation needed to do the job, even if all its tasks are at the same level. This prolongs the educational preparation time needed and probably inflates salary levels.

Job structuring and restructuring may be done to make jobs at lower levels less boring for workers in order to improve morale and thereby improve performance and/or reduce turnover costs. Such "job" enlargement" can be done economically by increasing the variety of task activities in a job while still assigning tasks which require the same basic investment in skill and knowledge training.

Job structuring is needed when the institution is to provide a new service or function, or is to utilize a new technology. It is then important to identify the tasks involved, to decide the job levels to which the tasks should be assigned, and to decide the existing job titles to which the tasks can be assigned to minimize the need for new educational preparation. It is important to clarify whether there is justification for creating entirely new jobs and/or whether the development of a job ladder is appropriate. The issues are discussed later in this chapter.

Job Ladders

Job or career ladders provide upward mobility for the in-house labor force of an institution. Promotional lines provide for a supply of new entrants into jobs as older incumbents retire, are dismissed, or as more staff are needed in a job title to handle increased volume.

The most powerful economic reason to have a career mobility program is to fill chronic vacancies at middle and upper job levels. In a field such as health services, most promotional lines require additional education as an individual goes from one job level to another. An economically desirable career mobility program would provide job ladder sequences that minimize the additional education needed between levels. If a job ladder starts from an entry level job with few vacancies, and progresses from one job level to another within interrelated task groupings to the level where shortages exist, the amount of educational investment required between each level is minimized; employees need to be trained only for the educational gap between one job level and the next.

There are other economic arguments in favor of job ladders. By selecting in-house incumbents in appropriate jobs to move up a job ladder, the institution can cut the costs that are incurred in orienting new employees. A program of upward mobility can also become an incentive for efficient performance if selection for upgrading is partly dependent on the quality of performance in the current job. Since trainees currently successful at one job level are likely to be successful at the next level (because of similar job content), the failure rate may be reduced. A career ladder program may also reduce the costs of turnover to the degree that high turnover reflects discouragement with "dead-end" jobs.

Actual salary costs may be lower with the use of upgrading programs than if staff are hired from the outside. The staff selected for upgrading will be at the top of their salary range when selected, but will enter the new jobs at the bottom of the salary range. They will be replaced in their former jobs by staff who are themselves newly upgraded and who will be entering at the bottom of the salary range for <u>their</u> new jobs.

Competition among institutions to attract and hire outside individuals whose training is in short supply creates an inflationary pressure on salary levels. An in-house career mobility program adds to the supply of scarce labor and reduces inflationary pressures. Strategies for job ladder construction are discussed later in this chapter.

When shortage jobs are at a high level, with no related jobs at intermediary levels, job structuring or restructuring may be needed to provide job ladders. If the educational distance from an entry job to a shortage job is a matter of several years, one cannot talk about a viable job ladder. For example, a one-step rise from the aide to the professional level would be unrealistic. But a ladder from the aide to the technician level, and from there to the technologist level can ultimately lead to the professional level in reasonable stages.[2]

To create a new job as an intermediary level on a ladder or to provide a new service or function is a form of specialization of labor that can be cost saving within limits. As different components of work are separated out and assigned to different jobs, the work can be done more efficiently and more economically. Lower-level tasks can be grouped into jobs at lower salaries. The limit to this approach is that the institution must be large enough to provide full-time work in each of the subdivided specialties. Short of this, workers would not be efficiently utilized. This question is discussed later in this chapter.

---

[2] It is important to note that a job ladder progression refers to the relationship among job titles. A given individual may not move up on all the rungs of a ladder. At any point in time incumbents at one level on a ladder are the population from which those who will be trained to go to the next level on the ladder are selected.

USING TASK DATA TO STRUCTURE JOBS

Assuming that the administrator of a department is interested in the rational structuring of jobs in the department, HSMS-type methods can provide the raw materials. The HSMS method produces task identifications and descriptions, groupings of tasks, assignments of tasks to job levels, and results in job structure and career ladder suggestions. The administrator can adapt data, analyses, and recommendations that reflect nationwide practice for use in his or her own department.

Analysis of Job Structures

Data Preparation

In adapting the recommendations the administrator first decides on the job titles to be examined; the second step is to identify the tasks being carried out in those titles; the third step is to analyze the pattern of distribution of the tasks in terms of task overlaps across jobs, the levels of tasks in jobs, and the groupings of tasks in jobs.

The administrator starts by selecting the job titles to be examined. These are placed on a reference list. The list should include all the in-house titles of interest along with the salary or salary range for each. Next, a HSMS job level should be assigned to each job on the list. Figure 5 in Chapter 3 presents the HSMS job levels. A way to check the appropriateness of job level designations is to note whether the rank order of the job titles by salary level is the same as the rank order of job titles by HSMS job level designation.

The next step is to determine which individual(s) are familiar with all the work being done by all the incumbents in the job titles on the list. This may be the administrator, or different supervisors may be familiar with different titles. These individuals are "respondents" or resource persons who can be asked to provide the basic information on the current allocation of tasks to titles.

The next major step in the analysis is to determine which of the tasks covered in the nationwide analysis are being carried out in job titles in the department or in titles related to the department. It may be best to get an overall sense of where the activities are being done before getting detailed information for each title.[3] The final list should include only tasks being done at the institution.

The next step is to find out in which job title or titles each task is done. For each in-house job the entire list is considered to ensure that all the tasks for a given title are covered. This means that a copy of the task inventory of all tasks should be prepared for each interview with each respondent. If it is decided that it is important to know which tasks are carried out by individual employees in a title, separate lists should be prepared for each; and each should be given the appropriate job level designation by title.

Each respondent is asked to indicate which tasks in the inventory are carried out in a given title. If a true representation of

---

[3] The HSMS extended task names provide good content references: they are sufficiently detailed to avoid confusion about what work activities are being referred to.

work assignments is of interest, an effort should be made to include out-of-title work, because this may be of major economic significance. Since this is an in-house analysis, security is not an issue.

At this stage there is a task list for each title and/or individual being studied. Each contains all the tasks done in that job. (The Task Code Numbers and abbreviated task names are useful references at this point.) Next to the name of each task, two additional pieces of information are needed. The first is the HSMS job level designation; the second is the name or number of the factor to which the task was assigned in the prior analysis.

A third piece of information may be of interest to the administrator. That is the frequency with which the task is carried out in a given job. This information will be helpful if there is interest in the relative importance of a task in the structure of a job. The basic information is obtainable from the respondents. (The use of the HSMS Task Frequency Scale (Scale 1) is discussed in Volume 2 of this report.)

If there is an interest in overall assessment of the manpower utilization pattern in the department, the next step is to create an array that contains the information of interest. This is done by arranging the job titles (and the names of individuals within titles) in columns, from left to right in descending order by HSMS job level and/or salary level. Within job levels the titles should be arranged by task factor. The factor for a job is determined by the most prominent factor showing on the task list collected for the title (or individual). The rows in the array are to be all the tasks found in the department, arranged from top to bottom in descending order by HSMS job level, and within job levels by the same factor order used for the columns.[4] The entries in the array are x's. Working with each list separately, one fills in a column at a time, placing an "x" in the appropriate column if a given task is found on the given job's list. Figure 10 is a hypothetical example of such an array in which HSMS data were used for the task numbers, levels, factors, and titles; the example assumes twelve incumbents, listed by number.

The array provides an overall view of the extent and location of task overlap and the appropriateness of current allocations of tasks to job titles by levels and factors. One examines the overlap of tasks across job titles or incumbents by reading across the array; one examines the mix of tasks in jobs by level and factor by reading down the columns. An ideal utilization pattern would be roughly in the shape of a diagonal, falling from left to right (as wide as the adjacent columns in a given factor; as high as the adjacent rows in a given factor within a level). Figure 10 shows this pattern, with the exception of Tasks 490, 74, and 275.

Task Overlap

Task overlap occurs when a task is carried out in more than one job title (or across more than one incumbent of a title if there

---

[4] To save space, abbreviations and Task Code Numbers can be used to designate columns and rows, since the entries will be x's.

Figure 10. HYPOTHETICAL ARRAY OF TASK ALLOCATIONS BY JOB TITLE

Job Levels[a]:			8		5	4		3		2		1		
Factors[b]:			I	II	VI	III	IV	III	A	IV	VI	IV	VI	
Job Titles[c]:			Non-neur. Rad.	Neuro. Rad.	Physicist	Rad. Tech. Supr.	Pt. Care Supr.	Rad. Tech.	Admin. Tech.	Pt. Care Tech.	Qual Ass. Aide	Pt. Care Aide	Qual. Ass. Aide	
Incumbents:			1	2	3	4	5	6	7	8	9	10	11	12
Level	Factor	Task Code												
8	I	441	X											
		329	X											
		448	X											
	II	404		X										
		397		X										
		430		X										
5	VI	528			X									
		546			X									
		542			X									
4	III	82				X								
		7				X								
	IV	158					X							
		305					X							
3	III	526						X						
		362						X						
		363						X						
	A	496						X						
		131							X					
		272							X					
2	IV	299								X				
		33								X				
		143								X				
		308								X				
		243								X				
	VI	535									X			
		548									X			
		549									X			
1	IV	290										X		
		190										X		
		193										X		
		490	X									X		
		74	X									X		
	VI	147											X	
		275						X					X	
		69												X
		552												X
		79											X	

a Level 8: specialized advanced professional; level 5: professional; level 4: educator, supervisor; level 3: technologist; level 2: technician; level 1: aide.

b Factor I: Non-neurologic Radiology; Factor II: Neuroradiology; Factor III: Radiologic Technology; Factor IV: Patient Care; Factor VI: Quality Assurance; Non-factor A: Administration.

c Fill in the in-house titles.

are different jobs within a title). Not all overlap is undesirable or avoidable. There are always overlap tasks to be done which are the mortar that holds the central tasks in a job together. There can be duplications that reflect the different locations or shifts in which the work is carried out. However, when there is duplication of the central work in a given department, this bears close examination; thus, the overlap data in the array should be given careful analysis.

The most important type of overlap to look for is where the same task is found in jobs that are at different levels. The allocation of low-level tasks to high-level titles is wasteful. If high-level tasks are allocated to low-level titles this may imply inadequate performance or wasteful training. Given acceptable performance of a task in the titles where it is currently overlapped, there is a _prima facie_ economic argument for downward assignment of an overlap task to the lowest level in which it is currently found. In Figure 10, Tasks 490, 74, and 275 are such tasks.

Sometimes the overlap reflects the case where supervisors fill in for absent staff. This may be a waste of expensive supervisory time. One solution might be to develop a "flying squad" for lower-level jobs. Such staff would be trained for several jobs at the aide level and would be on call to fill in for absentees. The squad(s) would provide a source of experienced manpower to cover staff absences at the aide level. By virtue of this experience, employees at the aide level could later make informed choices about the specialty in which they would like to rise. Management would be in a position to take account of especially gifted employees and encourage them. Rotation would permit job enrichment.

Sometimes overlap of tasks across job levels reflects the refusal of professional staff to delegate work. Some professionals prefer to carry out lower-level tasks when they are not comfortable about the quality of the performance of lower-level staff. Discovery of this kind of overlap actually pinpoints job performance and training inadequacies. The solution is to provide remedial training so that higher-level staff can rely on the quality of work assigned to lower-level staff.

The in-house analysis of the overlap data should result in the separation of necessary from unnecessary task overlaps, a design for the rational restructuring of jobs, and a list of steps needed to remedy the problems uncovered.

Job Structure By Task Level and Factor

At this stage it is possible to analyze whether the tasks in each job are at appropriate job levels. Allocation of tasks to jobs at comparable job levels is the economic objective. Since most jobs cannot be held together without one or two tasks that are essentially simple and/or administrative, percentage distributions and task frequency data are used to examine whether current allocations are sound.

The allocation of lower-level tasks to higher-level jobs suggests waste. It is also important to consider the presence of higher-level tasks in lower-level jobs. In a case where a task is rated at a level higher than the job in which it is found, the task may be beyond the reach of the incumbent's experience and training, and performance may be unsatisfactory. Alternatively, the staff in this job may be receiving training for the one task at levels that are beyond the needs for all the other tasks of the job, and this would be wasteful of training.

The analysis of the composition of jobs by task factor is similar to the analysis of the task levels. The array and the lists provide insights about the breadth of training required for the jobs. A job made up of tasks that cross several factors may require training in a larger number of subject areas than is economically warranted. For example, if the same staff member were administering medication and testing x-ray equipment, an investment in training in pharmacology subjects and in technological subjects would be required. With no transferability from one to the other, and no likelihood that this combination would be found in other lateral or higher jobs, we have a wasteful job structure.

Creation of New Jobs

A new job may need to be created as a result of the analysis of task allocations described above, or to provide an intermediary job between high and low-level jobs, or to provide for a new function, or to utilize a newly available technology. The key to structuring a new job is to know all the tasks to be covered, their job levels, and their factor designations. Frequency data reflecting the expected work loads to be assigned are also helpful.

The decision to institute a quality assurance program in diagnostic radiology provides a case to illustrate the type of analysis that might be considered in the creation of new jobs, given the principles already described. HSMS found that the tasks in this case fall into two job levels, those of the technician and those of the professional.

Quality assurance tasks are not found in every hospital, and not all quality assurance tasks are present in any one hospital. Those that are present can be found in a variety of job titles and job levels. Now that there is interest in instituting quality assurance programs, it would be sensible to decide on appropriate job structures to contain the technician and professional-level tasks. It would be cost-saving to remove technician-level tasks from their current locations in technologist and professional-level jobs and to create a quality assurance technician job. The remaining professional-level tasks would be assigned to the professional radiation physicist. However, this solution makes sense only if the institution is sufficiently large to benefit from a newly created quality assurance technician job.

In the absence of sufficient volume, the HSMS designation of level and factor for tasks can be used to decide the best allocation of the technician-level tasks among existing jobs. The decision can vary for different institutions depending on frequency data for current tasks; however, the following reasoning would apply.

The technician tasks can be taught to the aide and added to the aide's current duties. The new costs would be for training and a salary increase, because the aide job would now include technician-level tasks. It is also possible to teach the tasks to the technologist and add them to the technologist's current duties. The new costs would be those for training; there would also be hidden costs due to less than optimum use of the technologist's time when carrying out technician-level tasks. With the use of data on frequency and current work loads and flows, a sound economic decision can be reached.

What is inescapably apparent is that there is little justification for assigning the technician tasks to the professional, who is an expensive employee. If the reason for a job structure in which the professional is doing the technician-level tasks is that this is needed to provide full-time work for the professional physicist, the true function of the physicist may not be properly understood, and appropriate professional tasks may be missing. In addition, it may be sensible for a small institution not to hire the professional as a full-time employee, but to retain a consultant who will set up and run the quality assurance program as needed.

CAREER LADDERS AND COST SAVING STRATEGIES

An institution may decide to develop a program for upgrading staff in job ladder progressions because it is convinced that this approach is most efficient in the long run; it may have decided that this is the way to expand the services it provides, whether in sheer quantity when demand increases, or in the provision of new services or functions. The institution may have decided that this is the way to fill chronic vacancies. Its commitment to upward mobility may have been brought about through collective bargaining, with a portion of the wage package set aside for the upgrading-training of staff. In any of these circumstances there are basic decisions to be made that can affect costs and the success or failure of the program. This section brings together various insights gained by HSMS about the cost aspects of career mobility programs.

Overview

Unlike the situation where students gain their occupational preparation before they enter the labor force, an upward mobility program is concerned with the needs of working students, with employers, and with educators. The students are employed adults who very likely are the main source of support of themselves and their families. Employers may wish to provide their staff with occupational preparation, but also need to maintain the quantity and quality of their productive output

In health services occupations the jobs require instruction in formal disciplines. The subject matter must be imparted by teachers and learned in the classroom and in supervised clinical practice. Unlike many factory or civil service staff, the health worker cannot "pick up" what is needed in the higher-level job by simply observing other workers during the course of a work day in his current job. In addition, entry to health services jobs is often circumscribed by requirements such as licensure, certification, graduation from AMA-approved or otherwise accredited programs, and/or academic degrees. In most cases licensure and certification require graduation from accredited programs as well as passing examinations.

Given these considerations, an in-house upward mobility program must involve four basic types of costs. These are education costs, released-time costs, relief worker costs, and trainee failure costs; there are alternative ways of dealing with each.

Education costs cover classroom instruction and clinical practice. These would be faced by anyone entering study for an occupation. The options and choices about which this chapter comments are as follows:

1. There can be an in-house (hospital-based) program in which the institution runs the program; or there can be an academic program in which a student accumulates academic credits towards a degree at the associate, baccalaureate or masters level.

2. The program can be designed as an educational ladder with course work sequenced so that the whole program leads to the top of the ladder and shorter segments lead to lower-level jobs, so that students can exit and reenter the program at job-related intervals; or there can be discrete programs designed for each job.

3. Time schedules for instruction can be geared to full-time students and regular academic semesters; or they can be geared to the time requirements of employed students.

Released-time costs are payments to trainees while they are studying to permit them to maintain incomes. The options include finding outside assistance to pay employees, counting these costs as fringe benefits along with health insurance and passing them along as production costs to third-party payers, and/or having employees and/or educational institutions share in the costs.

Relief worker costs cover the salaries of employees who will provide the relief work while trainees are studying. Among the options are hiring temporary employees to provide the relief work for individual trainees, or using a staged approach in which the workers who will replace the trainees in their former jobs when the latter are upgraded are the ones to provide the relief work. A strategy for this is discussed below.

Trainee failure costs are incurred when trainees fail in their upgrading-training programs and are not able to fill the upper-level jobs. The selection criteria for trainees can affect success or failure. A set of alternatives is discussed below.

Education Costs

Sequential educational programs based on job ladders save education costs by eliminating redundant education and providing reinforcement and transferability of training. In addition, it is more economical in both the short and long run for health care institutions to give up the production of educational programs at technician and technologist levels. If they combine into consortia on a city-wide or system-wide basis they can purchase educational programs from academic institutions which can offer accredited programs and academic credits usable toward college degrees.

If large numbers of students are involved, the educational institutions could be persuaded to offer programs that are properly timed and sequenced to service the career ladder programs adopted by employers. The movement to work/study, continuing education, and work-oriented timing for course hours has been growing in colleges and universities since the late 1960's. Consortia can be created of employers in a system, such as in a municipal or voluntary system, or in a geographic area. Their function would be to adopt mutually acceptable job ladders and to purchase educational programs for a consortium's pool of trainees.

The alternative is having health care delivery institutions provide internal training for their manpower needs. The training produced is often so specific to the needs of the institution that the trainee finds it of little use for upward mobility or even for lateral movement in the job market. This is particularly true in the so-called "new career" titles. Since the institutions themselves are not permitted to provide academic credits, the training is of no help in the attainment of the degrees which are a part of the credential system and are needed for higher-level jobs.

The current time requirements for accredited occupational programs is a good argument for using the necessary time for students to accumulate degree credits as well as occupational certification.

Aide-level training should include remediation and be used to ready workers for later educational programs. It might be best to provide this in conjunction with programs leading to high school equivalency diplomas or college-level credits. Everyone at the aide level should have the chance to receive high school equivalency training, especially credits in the high school subjects required for entry to associate or baccalaureate-degree programs. Aides should be able to receive credit for their work experience when this is appropriate.

Given the number of trainees for upgrading programs that employer groups or consortia can offer, educational institutions could reduce per capita costs through the use of facilities and faculties in courses offered in the evening, on weekends, during vacations, and at other non-peak times. In health services, member hospitals would be natural affiliates for the clinical training.

A system-wide consortium approach could combine employers, educational institutions, and the relevant employee trade unions and professional associations. Consortia could make maximum use of federal, state, local, and foundation funding for its programs. It is a full-time job to locate the funds, write the proposals, and put the packages together. This can be done efficiently on a large, city-wide or system-wide basis.

Released-Time and Relief Costs

It is desirable to keep students employed and provide them with released-time training. The employer retains the services of current staff; the employee maintains an income source; and the educational institution may be able to use its plant at maximum efficiency. Released-time training costs could be passed on as direct costs of service, but, in addition, employees may wish to accelerate their training, and can be asked to contribute by studying without compensation on weekends, holidays, and during vacation time.

A strategy to minimize released-time and relief costs starts with an overall manpower planning program. With proper planning the upper-level "target" job on a ladder is one that will have openings for newly trained staff to fill. These openings could come about due to new or expanding services, due to turnover, retirement, or due to chronic current vacancies. The number of openings to be filled must be known before planning can take place. Planning is needed to make sure that money is in the budget for the job titles to be filled when the training ends and the trainees are ready to work in the titles.

It is also important that entry-level jobs on a ladder be able to be filled easily, because some relief workers must be hired at this level. This is not possible if there are no individuals available to be recruited and trained to fill the entry-level jobs.

If the entry level job is one in which employment may be reduced in the future, then the upgrading program solves the redundancy problem for staff that would otherwise be let go. New staff are not needed at entry levels, and the cost of upgrading is reduced by the amount that would be needed to recruit, train, and employ new replacements.

A multi-staged, coordinated system of training can fill vacancies and also provide for replacements. It involves half-time study and full-time income. It should include double-track staging to provide training at minimum cost with no loss in production. For the trainees, it can provide maintenance of income and job security, while guaranteeing maximum upward mobility. Double-track staging means that two educational programs run simultaneously. Each program is for half the trainees, and runs during the hours that the other half are working. The trainees work during non-overlapping time periods; study can overlap for weekends, holidays, and vacations.

The strategy for double-track programs is based on the following considerations. If trainees work half time and train half time, and if relief workers are to be used to maintain output, one relief worker can relieve two trainees, but only if the two trainees are in different time slots. Anything else is a waste of relief worker costs. Alternate months, weeks, or days for the tracks are better than alternate half days, because half days are wasteful of travel time and the warm-up time needed to refocus trainees' attention from study to work and back again.

The multi-stage strategy dovetails all the steps in a career ladder. With this approach released-time and relief costs can be kept below the cost of staffing the jobs whose vacancies are to be filled Figure 11 provides a hypothetical example. It shows how dovetailing of programs, maximum use of relief workers, and use of non-overlapping trainee work times can keep the costs to a minimum.

In this example the plan is to fill eight technologist jobs (at an institution that is part of a consortium) in the length of time needed to train new aides, to train aides to become technicians, and to train technicians to become technologists in a half-time, work/study

Figure 11. A MINIMUM COST STRATEGY FOR UPGRADING: STAGED SEQUENCES

Stage of Program and Jobs by Level	Vacancies	Employment by Function: Doing Normal Work	Relief Trainees[a]	Training for Entry Level Job	Upgrading Training[b]	Full-time Work Equivalent	Employment by Salary: Top of Range	Bottom of Range	Total	Average Monthly Wage Bill
0. Before program:										
Technologists(3) @ $1,000-$1,600mo.	8	12				12	12		12	$19,200
Technicians(2) @ $ 830-$ 900mo.		12				12	12		12	10,800
Aides(1) @ $ 660-$ 750mo.		12				12	12		12	9,000
(Budgeted vacancies @ top of range)	8									(12,800)
Total		36	0	0	0	36	36	0	36	$39,000
(Total including vacancies)							(44)		(44)	(51,000)
1. Hire and prepare 2 for aide jobs: (to free 2 aides to relieve 4 aides who will go into training to be technicians in Stage 2). New hires = 1/4 technologist vacancies to be filled. Time required: training for level 1.										
Technologists	8	12				12	12		12	$19,200
Technicians		12				12	12		12	10,800
Aides		12		2		12	12	2	14	10,320
Total	8	36	0	2	36	36	36	2	38	$40,320
2. Upgrading training of aides begins: a. Two aides relieve 4 aides selected for training to be technicians. Two-track program alternates work/study. Time required: training gap between levels 1 and 2. b. Halfway through period another 2 are hired, trained to be aides. (Time required overlaps with a.)										
Technologists	8	12				12	12		12	$19,200
Technicians		12				12	12		12	10,800
Aides		8	2	2[c]	4	12	12	4	16	10,980
Total	8	32	2	2[c]	4	36	36	4	40	$40,980

[a] Assumes that each relief worker relieves 2 employees who are each in half-time upgrading training.
[b] Assumes that upgrading trainees work half time and study half time at full-time salaries.
[c] For half the period.

Figure 11. A MINIMUM COST STRATEGY FOR UPGRADING: STAGED SEQUENCES (continued)

Stage of Program and Jobs by Level	Vacancies	Employment by Function: Doing Normal Work	Relief Trainees[a]	Training for Entry Level Job	Upgrading Training[b]	Full-time Work Equivalent	Employment by Salary: Top of Range	Bottom of Range	Total	Average Monthly Wage Bill
3. Upgrading-training of technicians begins:										
a. Four aides newly trained as technicians are upgraded.										
b. Four technicians relieve 8 technicians selected for training to be technologists. Time required: training gap between levels 2 and 3.										
c. Two new hires are trained to be aides one quarter way into the period (the time required overlaps).										
d. Halfway through the period 2 aides relieve 4 aides selected for training to be technicians (the time required overlaps).										
e. Three quarters of the way into the period another two are hired and trained as aides (unless fewer aides will now be needed than at start). (The time required overlaps so that total time is as in a.)										
Technologists	8	12				12	12		12	$19,200
Technicians		4	4		8	12	12	4	16	14,120
Aides		8	2	2[d] + 2[d]	4	12	8	8	16	9,960
Total	8	24	6	4	12	36	32	12	44	$43,280
4. Full cycle completed: Eight vacancies filled; 16 staff upgraded; 8 new hires.										
Technologists		20				20	12	8	20	$27,200
Technicians		12				12	4	8	12	10,240
Aides		12				12	4	8	12	8,280
Total	0	44				44	20	24	44	$45,720

[d] or a quarter of the period.

program. A ladder in quality assurance leading to radiologic technologist is an example of such a sequence.

The example assumes that trainees study half-time and receive full-time salaries. Current incumbents, including trainees for upgrading, are at maximum salaries for their lines and receive current wages until upgraded. New incumbents start at minimum rates.

Costs are reduced during the program by employing new staff only as needed in the staged sequences. All staff used for relief work are fully utilized and are retained at the end of the program to fill the slots vacated by the staff who have been upgraded. All staff needed for relief work at higher levels are provided from in-house staff. Output is kept constant. (See the column for full-time equivalent employment.)

If the 8 technologists were hired from the outside, the total salary cost of staffing 44 new employees for one month would be $51,000 (or $47,000, depending on whether new technologists would be recruited at the top or bottom of their salary range). At the end of the training cycle the same staffing of 44 employees would only cost $45,720 per month because the upgrading program reduces costs on every line where upgrading takes place. The additional cost savings from the reduction of training time by using an educational ladder to parallel the job ladder and from the elimination of orientation costs are not included in the figures given.

Stage 0 in Figure 11 shows current staffing and costs on a monthly basis. (The salary figures are illustrative.) In Stage 1 two individuals are hired (one quarter of the number of technologist vacancies). They are trained as aides.

In Stage 2 the first training step takes place. The new aides are able to provide released-time relief for four aides who now study to become technicians. Halfway through the period another two aides are hired and trained, so that a total of four aides can replace the four who become technicians at the end of the training in this period.

In Stage 3 the second training step takes place. The upgrading of four aides to be technicians makes it possible to relieve eight technicians to be trained to become technologists. At a point one quarter way into the period, Stage 1 is repeated, and then Stage 2, so that two new aides again relieve four aides for study. With an additional two hired and trained, four new aides are available to replace the four aides who are upgraded to be technicians at the end of the period. The training is dovetailed so that a total of eight new technicians are available through upgrading to replace the eight technicians who become technologists at the end of Stage 3.

At Stage 4 eight vacancies have been filled, sixteen workers have been upgraded, and eight new employees have been hired. Sixteen jobs formerly filled by staff at the top of their salary range are now filled by staff at the bottom of their range. At no time do the costs meet or exceed what costs would have been if the vacancies were filled from outside.

Trainee Failure Costs and Selection Criteria

Given the need to minimize the costs and time involved in training, there is some incentive for the institution to train those individuals who are most likely to succeed in the "target job" (the job for which the trainees are to be prepared). If, in addition, the existence of an upward mobility program can improve the quality of performance of individuals in current jobs, the net cost of upgrading programs can be substantially reduced. The HSMS approach provides two selection criteria that can be assumed to predict trainee success because they tie functioning in the current job to functioning in the related target job. Assuming that the job ladder reflects an association of tasks that require related skills and knowledges, we may conclude that the important tasks in jobs at varying levels on a job ladder are related. The HSMS criteria for trainee selection are as follows:

1. The current job title from which the trainees should be selected for a given target job is the one just below the target job on a job ladder.
2. The incumbents within the job ti . from which trainees are to be selected should be those with the best ratings for current performance.

If employees believe that the quality of their performance in the current job will be a factor in trainee selection, their current performance will be improved; at the same time, the most able trainees can be selected. The attractiveness of these criteria is that the first one is impersonal; it focuses on all the incumbents in a given job title; the second criterion is reasonable, since it rewards good performance. It also reduces the need for testing to performance testing or rating of a small group. If performance evaluation is ongoing, no additional testing is required.

Another important criterion is that of motivation. It is a concept which can best be handled indirectly, since it is subjective. For the purpose of trainee selection, self-selection for training is an acceptable indication of motivation, provided that all employees have had adequate access to information about the availability of the career mobility training program.

In any system of upgrading, especially if trade unions are involved, the criterion of seniority must also be considered. Seniority is a perfectly acceptable means of choosing between two otherwise equal candidates, and its use as one among several criteria is compatible with the HSMS approach.[5]

A Trainee Selection Strategy

Once the job title of the trainee population has been selected, the program can be announced. The potential trainee population

---

[5] A different sort of criterion is expressed in the practice called "creaming," which involves taking the most educated applicants regardless of their current job. "Creaming" is successful in the short run largely because educational levels are roughly related to job ladder sequences, and education provides intellectual skills. However, after creaming is over and the better educated are chosen, there is then no model for continued selection. Another criterion used to select trainees is scores on aptitude tests. The use of aptitude tests is no better than the validity of the test used (that is, the extent to which the test reflects job content and is free of cultural or educational bias). The HSMS approach bypasses the inadequacies of aptitude tests by going directly to work-related criteria.

would be those in the title who apply for the program, and this limited number of staff would be the ones whose current performances are evaluated as a basis for selection.

If a program of performance evaluation such as the one presented in the next section were underway, the available data might be sufficient to select trainees. Otherwise, performance evaluation would proceed as follows:

1. The tasks in the trainee population's job title would be identified as described earlier in this chapter. These would be designated by job level and factor.

2. Experts, such as supervisors, would select the most central tasks in the trainee population's job. These would be the reference tasks for the evaluation.

3. Supervisors familiar with the applicants' work performance would be selected as raters.

4. A performance rating instrument would be prepared:

   a. The extended task name for each task selected would be presented.

   b. For each task, the name of the employee to be rated and the rater would be entered.

   c. For each task, the rater would be instructed to consider the task and the criteria for evaluating the outputs of the task or performance of the task.

   d. For each task, the rater would be asked to compare the given employee's achievement of output or performance criteria with others regularly performing the task.

   e. The same scale would be used for each task and for each employee to be rated. The instructions and scale would read roughly as follows:

   Please compare this employee's performance of the task listed above with the performance of other persons regularly performing this task. Consider the criteria for the output of the task or for performance of the task, and consider to what degree the criteria are met by this employee and by others in the same job title. Please check the statement that best describes your comparison of this person with the others performing this task.

9...( )...Distinctly superior with respect to others in title.

8...( )...Considerably above average with respect to others in title.

7...( )...Moderately above average with respect to others in title.

6...( )...Slightly above average with respect to others in title.

5...( )...Average with respect to others in title.

4...( )...Slightly below average with respect to others in title.

3...( )...Moderately below average with respect to others in title.

2...( )...Considerably below average with respect to others in title.

1...( )...Distinctly inferior with respect to others in title.

5. The scores of each employee being rated would be calculated. If an employee is rated by more than one rater, scores would be averaged.

6. The seniority of applicants would be used to select from among applicants with otherwise equal scores.

Implementation

An institution committed to upward mobility as a continuous part of its manpower function must be aware that this requires planning and specification of the means for implementation. Such a program needs careful prior analysis and work if it is to be designed to suit the needs of the institution and the needs of individual staff members.

Implementation of a career mobility approach necessitates changes within the instituion such as the coordination of recruitment, training plans, and upgrading programs with the operations of the institution. Planning and a redirection of focus may be needed. No amount of commitment at high management levels can substitute for the involvement of middle and lower line personnel in the implementation of institutional change. The greatest enemy of a viable mobility program is staff ignorance of what is happening.

For this reason the issues of upward mobility should be discussed at every level in an organization and in cooperation with employee organizations where they exist. It should be noted that persons are less resistant to upward mobility for others when they have avenues open to themselves as well. Thus, a career ladder or lattices linking entry-level jobs through graduated sequences to the very highest professional and administrative jobs is most desirable if maximum support is to be enlisted.

EVALUATION OF INSTITUTIONAL PERFORMANCE

Evaluation is much in the minds of health services delivery administrators. There is pressure to review work as a means to greater efficiency; more importantly, there is pressure to review work as a way to promote quality. This section is a mini-manual for the use of HSMS task data in performance evaluation. It shows how the HSMS task descriptions or extended task names can be used to assess whether an institution is achieving its goals, to pinpoint the tasks being carried out below acceptable levels, or to evaluate an individual's performance.

Coverage

The institution must first decide what it wants to evaluate. Does it wish to learn whether the institution or department is accomplishing its goals? Is it to find out how the work in its most important functions is being carried out? Is it to find out how everyone in a given job title is carrying out the work assigned? Is it to find out how specific individuals, such as new employees, newly trained employees, or employees due for a review, are doing?

If the institution is interested in whether it is accomplishing its goals, a series of preliminary questions have to be answered at the outset. First, the goals themselves must be articulated. Then it must be determined how the goals should be manifested in work. The mere pronouncement of the objectives or goals of an institution is not enough to bring about the performance needed to attain the objectives. The institution must be able to point to the means of achieving the goals through their embodiment in tasks, elements within tasks, or standards of task performance.

If the institution is interested in the overall functioning in a department, it must first know what tasks are being carried out, and then it must decide which of the tasks it wishes to examine, and which performers of the tasks it wishes to review.

If the institution is interested in examining the work in a given job title it has to know which tasks are being carried out in the title and which tasks and performers it wishes to review. Even in reviewing the work of specific staff, it is necessary to know which tasks are being carried out by the performers who are to be reviewed.

Preparing For Performance Evaluation

To know whether there are task descriptions to cover all the tasks to be reviewed, the administrator would utilize task inventories in the manner described earlier in this chapter under the section titled "Using Task Data To Structure Jobs." That section describes the creation of task lists by job title and/or employee name.

The output of the first step is a set of tasks to be included in the review. For each task there should be a list of the names of the employees whose performance of the tasks are to be rated, and the name of one or more individuals who will rate the performers' work based on past experience or by current observation of the individuals at work.

The raters could be supervisors, co-workers, patients, customers, other appropriate persons, or a combination of these. In most cases the performer's supervisor is an appropriate person to evaluate a performer's outputs or task performance because of his or her experience or direct observation. However, patients or co-workers might be considered for health services tasks. If the output is directly consumed by the patient, such as when the task is to give personal care, the patient may be a reasonable judge of the output. In cases where the performer assists a senior co-worker who is not his or her supervisor, the co-worker may be the best rater of the task's outputs or performance.

The next step is to edit the task descriptions to reflect actual and/or desired in-house performance for evaluation purposes. Even though the task descriptions are already written from the point of view of approved procedures, the institution may wish to edit these to conform to actual practice at the institution and the objectives of the review. The institution may wish to address the following questions as a basis for refining the task list:

1. Are the tasks included the most appropriate to accomplish our goals?
2. In each task, is this the way we want to have the task done?
3. If there are choices of procedures, which do we prefer?
4. If there are choices of equipment, which do we prefer or have?
5. What should we be doing that we are not doing?

The output of this step is a set of task descriptions edited to describe the work as the institution requires it to be done.

Output and Performance Criteria

Each task description includes a statement naming the output of the task. (It appears in the upper left of the first page of the Task Description Sheets.) A task can have a tangible output, such as a set of radiographs taken during a particular examination. A task can have an intangible output, such as an explanation to the patient of how to prepare at home for an examination.

If a task has a tangible output, it should be possible to state concretely the criteria for evaluating the quality of the output. If these output criteria or standards can be stated explicitly, task performance can be evaluated objectively. If a task has intangible outputs, it may be hard to state output criteria. This would be the case when the output cannot be separated from the procedure, such as in giving reassurance, or when largely intellectual processes are involved, such as in diagnosis. In such cases it may be possible to state objective criteria for task performance rather than for the output *per se*. It may be crucial that all the steps in a task be done correctly in a proper sequence. The absence of a step may be as important as a wrong step. Such standards can be termed performance criteria.

The next step in the evaluation process is to go over the tasks to be reviewed and separate those for which objective output criteria can be written from those for which performance criteria will be written. The criteria should then be discussed, written, and reviewed by appropriate expert staff members in the department.

For a task which requires output criteria, the eventual evaluation instrument will need to contain the task reference and the criteria. The extended task name or the output statement on page 1 of the Task Description Sheet is probably sufficient as the reference.

When a task has several outputs, criteria for all may be written or the most important output and its criteria can be used. For a task which requires performance criteria, the evaluation instrument will need the extended task name as the task reference; depending on the performance criteria, the entire task description or particular elements of the task may also be used to highlight performance standards.

A decision must be made at this stage whether to assess the performer's work over a past period of time or to have the raters observe the performer during an evaluation period. There are arguments for or against either approach. There are negative aspects to relying on memory, but there are negative aspects to reliance on a single example when the performer may have been nervous. The practicability of observation also has to be considered; some tasks take a great deal of time or require that the performer be alone with the patient. The approaches may be combined. The decision should be made by the institution to suit its particular needs.

Rating Instruments

Figure 12, presented later in this section, is a suggested performance rating instrument. There would be one such instrument for each task, and as many copies of each as there are raters and performers to be reviewed. The sections to be filled in to fit each task are indicated in brackets. The institution may wish to change the language used in this example; however, the instructions to the raters should make the following points:

1. The rater is to keep in mind only the task named, only the criteria mentioned, and only the person being evaluated.

2. For evaluation of past work, the rater's use of the scale involves the rater's judgment of whether the performer meets the criteria, how many criteria are met (if there are several), how often the criteria are met in the usual course of the performer's work, and the degree to which the criteria are complied with.

3. For evaluation of work being currently observed for the purpose of evaluation, all the above considerations must be eliminated; the work being currently evaluated is the only thing that can be considered.

4. In deciding on the ratings, the rater is to assess the performer's outputs or performance using the criteria as absolutes, and is not to compare the performance or outputs of one performer with those of another.

5. The rater checks a rating value on a nine-point scale whose ratings range from highly unacceptable to much better than acceptable.[6]

The Rating Data

The ratings provide data to describe the quality of the task performance in statistical terms. They can be expressed as distributions of superior or inferior performance around scale point 5, which is the minimum acceptable level. The distributions of ratings for each task tells the institution about its overall performance of each task. The distribution of ratings for each performer tells the institution

---

6 In the section on trainee selection presented earlier in this chapter a similar rating scale is presented. It differs with respect to the reference. The earlier one compares the performer with others in the title. The one here compares the performance with absolute criteria. The reason for the difference is that one must assume a normal distribution of ratings for predictive purposes when doing trainee selection. When absolute standards are involved, skewed distributions can be expected. For evaluation of institutional performance, the skewness of the distribution is of interest and, if it is in a positive direction, is even desirable.

Figure 12. <u>SAMPLE OUTPUT OR PERFORMANCE RATING INSTRUMENT</u>
p. 1 of 2

GENERAL INSTRUCTIONS

You are being asked to consider the work activities of one or more persons employed in this institution. Each work activity, called a task, will be summarized for you on one of the following pages, along with the name of the person whose work you are to consider. You may be asked to consider the work of more than one person doing the same task, and/or more than one task done by the same person. However, there is a place for you to indicate that you do not feel that you have enough information to rate the person or the work represented by the task named.

Your ratings will make an important contribution to determining the current general level of performance in the task being rated. These ratings can be used to help plan for improvement of work performance.

The task statements will each be accompanied by a statement of what qualities are considered desirable with respect to the outputs which result from the task (the task output criteria), or by a statement of what performance standards are desirable with respect to how the task is carried out (task performance criteria). Several criteria may be mentioned.

If you are asked to consider work carried out over a past period of time, please consider how many of the criteria are met, to what degree they are met, and how often they are met by the person named, over that period of time. You should then check off the statement that best describes your evaluation.

If you are asked to evaluate work you are currently observing, please consider how many of the criteria are met and the degree to which they are met only in the work you are currently observing. Then check off the statement that best describes your evaluation.

Try to be fair, objective, and impartial in your ratings. Base your ratings on the employee's attainment of the criteria for the task named and not on any personal characteristics which he or she may have; do not compare this person's performance or outputs with those of others. Please do not let your evaluation of this person's performance in one task affect your judgment of how another task is done by the same person.

Please fill in your name and title wherever it is called for.

Thank you very much for your cooperation.

Figure 12. SAMPLE OUTPUT OR PERFORMANCE RATING INSTRUMENT (continued)
p. 2 of 2

OUTPUT OR PERFORMANCE RATING SHEET

1. Please enter: Your Name ____________________ Title ____________

2. You are asked to consider the following task: (Code No. - - - - - - -)

[Institution fills in the task code number (above) and in this space enters the extended task name. Institution may also wish to name the task output(s).]

3. You are asked to consider employee: ____________________ (Name)
____________________ (Title)

4. Do you feel qualified to evaluate the quality of this person's work performance in this task? ... ( ) Yes; ... (No). If no, please go on to the next task. If yes, please go on to item 5.

5. Please consider the criteria to use to evaluate the task's output or the performance of this task:

[Institution fills in the output or performance criteria.]

6. You are being asked to evaluate the performance or output of this employee in one of the following ways:

( ) Consider the employee's work over a substantial and representative period of time; do not concentrate on very recent experiences or only on outstanding examples of achievement or failure.

( ) Observe the employee at work currently.

[Institution selects one or both.]

7. Please rate this employee according to your judgment of the degree to which he or she meets this task's output or performance criteria: How many criteria are met, and how well are they met? Please check the statement that best describes your evaluation.

9...( )...Distinctly superior with respect to criteria.
8...( )...Considerably above acceptable with respect to criteria.
7...( )...Moderately above acceptable with respect to criteria.
6...( )...Slightly above acceptable with respect to criteria.
5...( )...Acceptable with respect to criteria.
4...( )...Slightly below acceptable with respect to criteria.
3...( )...Moderately below acceptable with respect to criteria.
2...( )...Considerably below acceptable with respect to criteria.
1...( )...Distinctly inferior with respect to criteria.

about the competence of individuals. It is then possible to pinpoint problem tasks and problem performers and design remediation through training or reorganization.

When the Output or Performance Rating Sheets have been collected, these should be arranged in sets by task, and arranged within each task set by order of the scale value checked. The results can then be entered in a table similar to that presented in Figure 13. On each row a task's code number and a very abbreviated task name are entered. Column (1) is the total number of ratings for a task. This would be equal to the total number of performers being rated on the task. (If performers receive ratings by several raters it might be necessary to first average all the ratings for a given performer.) In column (2) there are sub-columns, one for each scale value. For each task, the number of ratings at each scale value is entered. (Their sum should equal the figure in column (1).) Column (3) is the percentage distribution of the scale values. It is necessary to do percentage distribution so that comparisons can be made from task to task. (The percentage distribution is obtained by dividing a given entry in a sub-column in (2) by the total figure in column (1), and multiplying by 100.)

A similar table can be made that shows the ratings for employees. In Figure 14, each row refers to an employee. Column (1) is the total number of tasks on which the employee has been rated. Columns (2) and (3) now refer to the distribution across employees.

The institution is now in a position to judge which tasks are being performed at acceptable levels, and to what extent. Ratings at

Figure 13. SAMPLE HEADING FOR TABLE OF OUTPUT OR PERFORMANCE RATINGS BY TASK

Task Name (Abbreviated)	Task Code No.	Total Task Ratings (1)	Ratings of Performance by Task																	
			Number of Output or Performance Ratings by Scale Value (2)									Percentage Distribution of Ratings by Scale Value (3)								
			9	8	7	6	5	4	3	2	1	9	8	7	6	5	4	3	2	1

Figure 14. SAMPLE HEADING FOR TABLE OF OUTPUT OR PERFORMANCE RATINGS BY EMPLOYEE

Employee Name and Job Title	Total Empl. Task Ratings (1)	Ratings of Performance by Employee																	
		Number of Output or Performance Ratings by Scale Value (2)									Percentage Distribution of Ratings by Scale Value. (3)								
		9	8	7	6	5	4	3	2	1	9	8	7	6	5	4	3	2	1

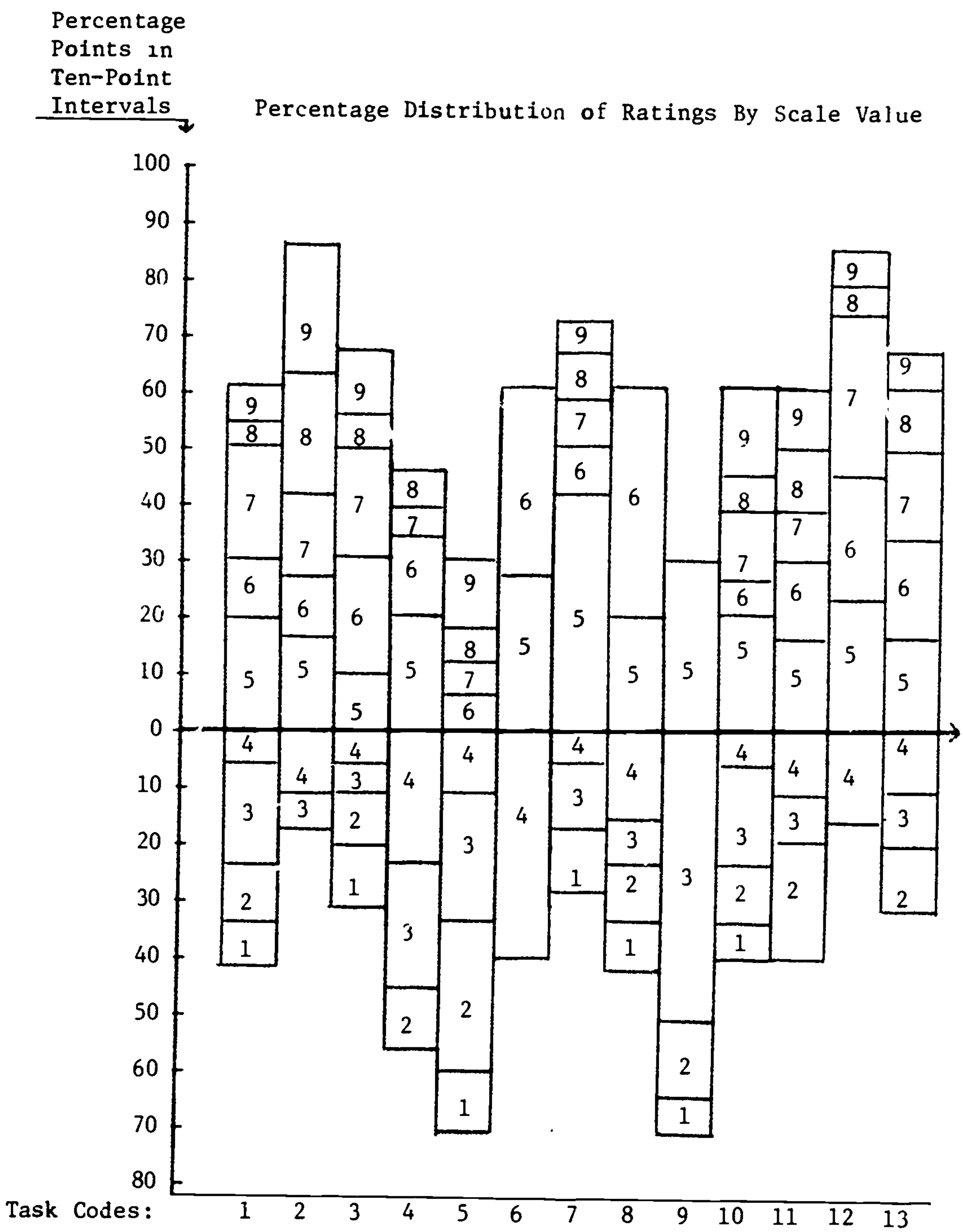
scale points 1 to 4 fall below acceptable levels. Ratings from 5 to 9 are at or above acceptable levels. The institution can now decide what level of achievement it wishes to attain, and what type of distribution for a task warrants the task being considered a "problem task."

Figure 15 represents a graphic portrayal of a hypothetical distribution based on column (3) of Figure 13. The distribution represents hypothetical data for thirteen tasks whose code numbers are listed along the bottom horizontal axis.

Each bar represents a task. Each vertical division on the left-hand vertical scale represents ten percentage points. Within each bar, the percentage distribution of the ratings for a task are laid off by scale value. Rating scale values starting from the value of 5 are laid out above the zero line and move up to 9; the scale value of 4 is laid out below the zero line; others follow down to 1. The number that appears in an area within a bar indicates the scale value represented by the area in which the number is found. Laid out this way, the area above the zero line shows the percentages of the tasks at acceptable ranges, and the area below the zero line shows the percentages of the tasks at unacceptable ranges.

The institution can now see that a task such as Task 6, while having a 40 percent distribution below the "acceptable" level, has none below the rating of 4, "slightly below acceptable." Any task with as much as ten percent of its ratings at 1, "distinctly inferior," might be in trouble. Task 5 is such a task. Tasks 6 and 8 have no ratings above 6, "slightly above acceptable," and that might be of

Figure 15. HYPOTHETICAL GRAPHIC REPRESENTATION OF DISTRIBUTION OF OUTPUT OR PERFORMANCE RATINGS BY TASK



Each bar represents distribution within one task. Numbers within bars represent the scale value which the area represents. Areas are percentages as read on the vertical scale.

concern. The clearest problem task is Task 9, with 70 percent of its outputs "moderately below acceptable" or worse, and none better than "acceptable."

A similar visual presentation can be prepared for the employee data. In such a case the bars would refer to employees rather than tasks, and the distributions would indicate performance ratings across the tasks of the performer's job. Once the "problem tasks" or "problem employees" are located, it becomes possible to diagnose what it is about the quality of the performance or of the output that has given rise to the inadequacy of the results. Then remediation can be planned.

APPENDIXES

APPENDIX A

TECHNICAL HISTORY OF THE HEALTH SERVICES MOBILITY STUDY INSTRUMENTS

The following pages are excerpts from an earlier method manual developed by the Health Services Mobility Study (HSMS) in 1971.[1] Since then HSMS has further developed and refined its instruments and methods. This appendix reports on the work that was done to develop the analytic instruments and to assure their reliability.

CONCEPTS

Reliability is the consistency with which an instrument yields the same results when a set of observations are made by different people or are repeated over time. An instrument such as a task definition is of limited usefulness if it yields different results when applied one day than it does when applied another, or if different results are obtained when two or more different analysts apply it to the same observations. The relevant aspect of reliability for task identification is inter-rater reliability. This refers to the consistency with which two or more individuals using the same definition identify the same tasks for the same performer.

---

[1] A Job Analysis Method For Developing Job Ladders And For Manpower Planning, Health Services Mobility Study, Research Report No. 3, Part A, Volume III, "Task Identification: Data." Part B, Volume III, "Skill Dimensions of Tasks: Data." and Part C, Volume III, "Knowledge Identification and Scaling of Tasks: Clustering Tasks." 1971. This document is no longer available for distribution. It has been replaced by Research Report No. 11.

Validity is a difficult concept with which to deal, both conceptually and practically. The validity of an instrument is concerned with what the instrument identifies and how well it identifies. It deals with the extent to which an instrument identifies what it is supposed to identify.

The relevant aspect of validity for task identification is content validity. This involves the systematic examination of the content of the data to determine whether they cover an accurate and representative sample of the activity to be identified.

THE DEFINITION OF TASK

HSMS revised its definition of "task" during a series of three field tests (referred to as pre-test, re-test, and pilot test). This section describes the HSMS field testing of its definition of "task" and the results.

Reliability

The HSMS field tests measured inter-rater reliability in task identification. The tests measured the extent to which several analysts, each exposed to the same training, the same situations, and the same experiences with performers, identified the same tasks. A given number of performers were observed and interviewed by a group of HSMS analysts who then independently filled out task identification data. Reliability was measured in terms of the analysts' agreement on the output, what is used, and the recipient, respondent, and co-workers for a task, with full agreement including all three components.

An overall reliability score and a score for each analyst provided information about the method itself as well as about the functioning of each analyst. We found that the reliability of the analyst improves with practice; thus the user can expect increasing reliability over time.

HSMS devised a Standard Reliability Score which provides a standardized measure that can be compared across test situations. The Standard Reliability Score for any analyst measures his agreement with other analysts. The formula for the Standard Reliability Score is as follows:

$$A_r = \frac{W}{(N-1)(T)}$$, where:

A_r = the Standard Reliability Score for an analyst, per performer;

W = the relevant summed agreement scores (the sum of the number of other analysts in agreement with a given analyst on the identification of each of the tasks identified by any and all of the analysts);[2]

N = the number of analysts involved in the test;

T = the total number of separate tasks identified by all the analysts in the team for the performer.

A_r is taken to two decimal places. Perfect inter-rater reliability for an analyst would be 1.00. For example, if five analysts in a test agreed on the same 20 tasks for a particular performer, the summed agreement score for any one analyst would be the sum of 20 tasks,

---

[2] The summed agreement scores cannot by themselves provide standard measures of reliability because their absolute values are affected by the number of analysts and the total number of tasks identified in a particular test.

times the agreement score of 4 for each, or 80. The Standard Reliability Score would be: $\frac{80}{(4)(20)}$, or 1.00. The analyst's inter-rater reliability is his average across performers; overall reliability is the average across analysts.

Reliability Test Results

The pilot test was conducted under very difficult circumstances. Because of time pressures and illnesses, the reliability data were collected for three performers, rather than for the five originally planned. In addition, the analysts went immediately from classroom training in the revised method to field testing without prior evaluation of their field performance. As a result, the testing for some of the analysts may have reflected the reliability of the definition as it was in the second test. (Misconceptions were clarified *after* the test data were collected.)

Statistical Data

Table 1 shows the analysts' reliability scores in the three tests. For the pilot test, separate scores are also shown for tasks of each of three performers. Column headings refer to the tests and to the performers, respectively, moving from left to right. There were five analysts in each test; only one analyst (Analyst 1) was involved in all three tests; one other analyst (Analyst 2) was involved in the last two tests. Any analyst involved in the first or second test was reported on only if he or she was present for the third test.

TABLE 1. COMPARISON OF TASK ID RELIABILITY SCORES

Analyst	Test Situations[b]			Pilot Test Performers[b]		
	Pre-test	Re-test	Pilot Test	EKG Tech.	Family Health Worker	Nurse Practitioner
(1) No. of Performers:	5	11	3			
(2) Average[a]	.55	.57	.53	.51	.53	.55
(3) (T) 1	.55	.55	.56	.54	.56	.59
(4) (S) 2	n.a.	.57	.55	.53	.55	.57
(5) (F) 3	n.a.	n.a.	.53	.55	.52	.53
(6) (M) 4	n.a.	n.a.	.50	.52	.47	.52
(7) (L) 5	n.a.	n.a.	.49	.40	.53	.55

n.a. Not applicable. Analyst was not on staff at the time.
[a] Average based on five analysts for each test; analysts varied from test to test.
[b] In order as studied, from left to right.

Line (1) indicates the number of performers studied in each test situation; there were five, eleven, and three, respectively, in the three tests. The three performers for the pilot test are listed in increasing order of job level. This was expected to provide a rising level of difficulty for the analysts, since difficulties in determining task boundaries generally increase as the job level rises. The performers were studied in the order shown. Since reliability tends to increase with practice, we expected an offsetting effect to the increasing difficulty.

Line (2) presents the overall reliability scores. There was some falling off from the second test. Pilot test reliability was .53. This is probably the result of the inexperience of the analysts in the test. As line (3) indicates, Analyst 1, present for all three tests, has scores that rise from test to test. Analyst 2's score dropped somewhat from the re-test to the pilot test. However, both of these analysts had prior experience and scored higher than the three without prior experience.

The effect of experience is also manifest when one examines the separate averages for the three performers of the pilot test (line 2). There is a rising trend, despite the rising level of complexity of the jobs.

Interpretation

The pilot test results indicate that careful training and field experience are needed to raise reliability above an average of .53. However, .53 is an acceptable figure for reliability when one considers that we are not dealing with probability samples. The figure suggests that, with a small degree of training, half the analysts will always agree, or that all the analysts will agree half the time -- with all the possible variations between.

Validity

In the first two tests HSMS used the concept of "accuracy" as an approximation of validity. We measured the individual analyst's agreement with an analyst on the team who was designated as the "expert analyst." The expert was the senior staff member who trained the others and who was presumed to be better able to use the method. The pilot test was to have departed from the use of the "expert"; the accuracy score was to have been a measure of congruity with the group judgment. It was assumed that the solutions arrived at through discussions in a team Task ID Conference would be better than those of a single analyst.

In the first two tests, the sum of "correct" identifications was not used as a measure of accuracy, since the sum could not also reflect excessive task identification as errors. Our Standard Accuracy Score adjusted for excess identifications and also permitted comparison of accuracy measures across test situations, even when the correct number of tasks varied. The formula for the Standard Accuracy Score is as follows:

$$A_c = \frac{C}{T}\text{, where:}$$

A_c = the Standard Accuracy Score for an analyst, per performer;

C = the number of relevant correct answers: the total number of tasks correctly identified, minus the total number of tasks identified by the analyst in excess of the correct total. (If the result is a negative number, it is entered as a zero.);

T = the total number of correct tasks (as per criterion).

On further reflection, HSMS concluded that a good statistical measure of validity was not available since, when the analysts are newly trained the conference or group decisions are no better than the group's overall mastery of the method, and the "expertise" of any given senior analyst cannot be measured objectively.

Rather than test for validity, HSMS built procedures into the method to ensure that the tasks identified would conform to the definition and adequately reflect the performer's work. These include the requirements that there be team agreement on task identification, the use of literature, and review by the performer, the director, and by a minimum of three expert reviewers. These procedures provide content validity which is verified by inspection.

SKILL SCALES

Development of Several Skill Concepts

In dealing with interpersonal interaction skills, the HSMS staff members provided several important insights. They suggested that therapeutic interaction is no different from non-therapeutic interaction with respect to skill, but, rather, differs only with respect to motivation. For that reason, four original interpersonal interaction skill scales (Therapeutic, Non-therapeutic, Co-worker Cooperation and Leadership) were combined into two skill scales (Human Interaction and Leadership). The Leadership skill reflects the general literature on the conditions of leadership, rather than the nature of leadership. Both of the interpersonal skills are described in terms of the conditions under which they are required to be exercised rather than their manner of expression. This allows for individual differences in the way in which the skills are manifested according to personality differences in the performers.

The content of the language skills was clarified once it was understood that the Knowledge System would deal with vocabulary and grammar. The three scales relate to precision in the choice or use of language. Originally, there were four language scales; testing disclosed a high correlation between Comprehension of Spoken Language and Oral Use of Language, and we decided to combine the scales. Such combination was not found to be warranted with Reading and Writing. It is logical to expect that these latter skills are needed independently.

HSMS had long considered the need to account for and measure intellectual skills as well as manual and interpersonal skills. We first conceived of one skill, General Intellectual Development. It was to contrast with General Educational Development, and refer to an intellectual capability that develops as knowledge is attained, but is independent of specific knowledge content. It soon became clear that we were dealing with a multidimensional skill concept.

HSMS designed a crash program to determine the nature and number of skills that would account for intellectual skills of a general nature that would be learnable, free of specific content, and scalable with respect to task requirements.

Using the work of J.P. Guilford as a departure point,[3] we worked with three broad areas of mental content: Semantic, Symbolic, and Figural, and, within each, three broad areas of mental activities: Cognition, Mental Search, and Evaluation. We also identified and included four specific mental relationships: Classification Principles, Changes in Mental Content, Organizational Principles, and Implications. The various combinations of these dimensions resulted in 36 scales.

To find how the scales might "work" in the field, we decided to sample a variety of jobs by function and by level of educational requirements. We created a set of six functional and six educational criteria to select performers and job titles. This resulted in a matrix of 36 job category cells. We then attempted to find two performers for each cell, one to be interviewed only and one to be observed and inter-

---

3 J.P. Guilford, The Nature of Human Intelligence, New York: McGraw-Hill Book Co., 1967.

viewed. The design called for identification and scaling of three tasks per performer, with the tasks selected to be representative of the job, without duplicating tasks. This gave us a set of 216 tasks.

We trained our analysts to scale for only those GIS skills required in task performance. The field manual described the GIS skills and presented a definition, a description of the principles to use in scaling, and a set of questions which could be asked aloud by the analysts or posed to themselves to aid in scaling. (We used a numerical scale.)

The scaling data were punched on cards, one task per card, with accompanying identifying information. The data were then evaluated in a series of runs using principal component factor analysis.

We found that content was the chief discriminator of intellectual skills. We concluded that different intellectual skills are probably involved when the material acted upon or worked with is semantic, symbolic, or figural in content.

Identifying the nature of the three GIS content skills was a problem, however. We could not deal with semantic, symbolic and figural skills without clarification. To be learnable through practice and scalable, the behavior involved must be more concretely specified. It had been difficult for the analysts to discriminate among the twelve content skills that clustered on each content factor; it was hard to scale independently for each.

We decided to separately factor the variables that clustered on each original factor; these were the variables for each of the three content areas (semantic, symbolic and figural) covering mental activities and relationships. We also reasoned that factor structures might be most differentiated if we factored only tasks collected from the upper half of the data base with respect to the educational level required for the jobs from which the tasks were drawn. We thus then used the data from 108 of the 216 tasks, running variables from each of the three content area factors separately. The results were as follows:

1. The factor structures within each of the three content areas were different. Only in semantic skills was the structure obtained in the various runs consistent and robust enough to suggest discrete variables within semantic skills.

2. In semantic skills there appeared to be four factors. When high inter-correlations were discounted, and relevance to use in task analysis was considered, two factors remained. One was related to implications and the other to class characteristics and organizing principles. The two which were discarded included one dealing with changes in semantic meaning and one dealing with cognition, neither of which was considered relevant to our needs.

3. Symbolic skills and figural skills had less stable, less consistent factor structures. We decided that a single skill could adequately cover each content area.

4. The factors dealing with semantic implications and semantic class characteristics and organizing principles covered mental relationships. We reasoned that, as such, they might apply across the three content areas. Thus, we arrived at four General Intellectual Skills: (1) Implicative Skills; (2) Taxonomic Skills (class characteristics and organizing principles); (3) Symbolic Skills; and (4) Figural Skills.

5. HSMS scaling principles for each skill were subsequently worked out by staff members.

Statistical Attributes

The attributes of scalability and unidimensionality were made possible for the scales by determining and describing the nature of each scale's quality, naming the scale appropriately, and identifying the scaling criteria. The scaling criteria are called scaling principles.

In the final revisions of the scales, scaling principle(s) were specified for each skill. In cases where more than one principle was necessary, a description of three levels for each principle was formulated. Then all combinations of the levels for the principles were examined to see whether they made sense; that is, we asked the question, "Can we conceive of a task displaying this combination of levels of the principles?" A set of tentative descriptors were then written.

Equal Appearing Intervals

The attribute of reliable scale values was achieved through the Thurstone technique of "equal appearing intervals."

The idea underlying the technique is that a scale can be made up of descriptors, but values cannot be arbitrarily assigned to these, because the real or perceived "distance" between the descriptors along the scale must be accurate for later applications of multivariate statistical techniques. The problem is resolved by asking independent raters (judges) to assign values to a set of descriptors arranged at random. Scale values are calculated using the median scale value of the judges' responses. The reliability of the median is assesed by determining the interquartile spread between the 25th percentile and the 75th percentile values. (The median is at the 50th percentile). Usually, a spread of 2 points or more is considered unreliable, and such a descriptor is discarded.

The formula for the median scale value is: $\frac{L + (50.00) - \sum P_b)}{P_w}$

where,

L = lower limit of interval in which the median falls (50th percentile).

P_w = percentage of cases that fall in the interval containing the median.

$\sum P_b$ = cumulative percentage in the interval below the one containing the median.

On a nine-point scale (which we used) there are ten intervals. The first contains the zero. The <u>frequency</u> of a descriptive item is calculated by counting the number of judges that placed the item in a given interval. The <u>percentage</u> referred to above is the frequency of an item in an interval, as a percentage of the number of judges whose responses are being used. The <u>cumulative percentage</u> referred to above is the percentages of the item in a given interval, plus the sum of the percentages of the item in all lower intervals. An example of our equal interval tests is presented in Figure 1.

The scales have gone through a number of revisions. Each time a scale was revised it was subjected to another equal interval test unless the revision was only a minor language change. Table 2 reports the results of the equal interval tests for the 18 HSMS scales in use [in 1971].

Figure 1. EQUAL INTERVAL TEST FOR SCALING LEVELS OF LOCOMOTION

Instructions to Judges

You are being asked to rate a set of statements which may be used to describe the levels of a particular skill. The skill is one which can be called for in the performance of job tasks. The skill to be considered here is Locomotion.

This skill refers to the degree of body coordination required of a performer in a task. The skill involves the movement of the performer's body, torso or limbs through space in order to achieve predetermined standards of body movement or position.

The level of the skill rises with the degree of body coordination required. This is determined by the complexity of the standards involved, or the complexity of the external circumstances which restrict motion.

The scale level is not determined by considerations of strength or the level of knowledge which may be involved.

On the following pages you will find a number of statements which could describe levels of the skill. Next to each statement is a series of numbers from 0 to 9; 0 is to represent the absence of Locomotion; 9 is to represent the highest possible level of Locomotion which a task can require. You are asked to check off a value for each statement.

Please read the entire set of statements first, keeping in mind that you are asked to rate each statement with a number from 0 to 9, according to the level of Locomotion which the statement represents to you. Make any notes that you need as a guide. The statements have been lettered at random.

You are asked to check a value for each statement so that each can later be put in ascending order. The statements which you consider to be at higher levels must be assigned values higher than statements which you consider to be at lower levels of the skill.

You need not assign all the numbers from 0 to 9. If you think that two statements are at an equal level they should be scored the same even if they do not have the same wording. Equal numerical differences should represent equal degrees of the skill.

Once you have sorted the statements to your satisfaction, please reread each statement and check off the scale number which, in your opinion, best represents its level of Locomotion. Remember that 0 is to be used when the skill is absent or irrelevant, and 9 represents the highest possible level of the skill.

Please see that you have properly checked the scale points which you have selected for each statement. Be sure that every statement has one and only one scale point checked. Please write your name below. Thank you for your cooperation.

Judge's Name ________________________________

	No Locomotion								Highest level of the skill
	0	1	2	3	4	5	6	7	8	9

Descriptive Statement — Scale: Locomotion

(a) The task requires the performer to move his body, torso or limbs through space so as to achieve somewhat complex, predetermined standards for body motion or position. A moderate degree of body coordination is called for.

(check one)

0	1	2	3	4	5	6	7	8	9
( )	( )	( )	( )	( )	( )	( )	( )	( )	( )

(b) The task requires the performer to move his body, torso or limbs through space so as to achieve extremely complex, predetermined standards for body motion or position. An extremely high degree of body coordination is called for.

(check one)

0	1	2	3	4	5	6	7	8	9
( )	( )	( )	( )	( )	( )	( )	( )	( )	( )

(c) The task does not require the performer to move his body, torso or limbs through space so as to achieve a predetermined standard for body motion or position.

(check one)

0	1	2	3	4	5	6	7	8	9
( )	( )	( )	( )	( )	( )	( )	( )	( )	( )

(d) The task requires the performer to move his body, torso or limbs through space so as to achieve simple, predetermined standard(s) for body motion or position. A small degree of body coordination is called for.

(check one)

0	1	2	3	4	5	6	7	8	9
( )	( )	( )	( )	( )	( )	( )	( )	( )	( )

(e) The task requires the performer to move his body, torso or limbs through space so as to achieve considerably complex, predetermined standards for body motion or position. A high degree of body coordination is called for.

(check one)

0	1	2	3	4	5	6	7	8	9
( )	( )	( )	( )	( )	( )	( )	( )	( )	( )

Table 2. DATA BASE FOR SCALE VALUES USING THURSTONE EQUAL INTERVAL SCALING
(p. 1 of 2)

Scale Name, Number; and Number of Judges[b]	Statistical Description	Items Listed in Order Presented to Judges									
		a	b	c	d	e	f	g	h	i	j
1. Frequency (15)	Median Scale Value[a]	5.0*	2.0*	8.0*	0.0*	3.0*	9.0*	1.0*	7.0*	4.0*	
	Interquartile Range	1.0	0.9	0.8	0.5	1.1	0.6	0.8	0.7	0.1	
2. Locomotion (15)	Median Scale Value[a]	5.0*	9.0*	0.0*	1.5*	7.0*					
	Interquartile Range	1.9	0.6	0.5	1.0	0.9					
3. Object Manipulation (15)	Median Scale Value[a]	1.5*	0.0*	5.0*	9.0*	6.5/	3.5*	7.5*	3.0/		
	Interquartile Range	1.0	0.5	0.9	0.6	1.4	1.2	1.0	0.9		
4. Guiding or Steering (15)	Median Scale Value[a]	1.5*	4.0/	9.0*	7.0*	0.0*	3.0*	5.5*	8.0/		
	Interquartile Range	1.1	1.2	0.4	1.0	0.5	1.4	1.0	0.9		
5. Human Interaction (18)	Median Scale Value[a]	5.0*	7.0*	3.0*	0.0*	9.0*	1.0*				
	Interquartile Range	1.1	1.0	1.5	0.0	0.6	0.8				
6. Leadership (22)	Median Scale Value[a]	4.5#	4.0/	1.0#	8.5#	5.5/	3.5/	6.5#	5.5/	3.0#	0.0*
	Interquartile Range	1.5	2.0	0.7	2.1	3.1	1.4	2.1	1.6	1.2	0.4
7. Oral Use of Language (18)	Median Scale Value[a]	4.0*	7.5*	2.0*	0.0*	9.0*					
	Interquartile Range	1.2	1.1	1.0	0.5	0.6					
8. Reading Use of Language (17)	Median Scale Value[a]	2.0*	7.0#	0.0*	9.0*	5.0*					
	Interquartile Range	1.2	0.9	0.5	0.8	1.6					
9. Written Use of Language (15)	Median Scale Value[a]	6.5*	5.0*	9.0*	2.0*	0.0*					
	Interquartile Range	1.1	1.1	0.8	0.8	0.5					

* Item was kept.
# Item was edited.
/ Item was eliminated.

[a] Rounded.
[b] Refers to number of judges in equal interval test.

Table 2. DATA BASE FOR SCALE VALUES USING THURSTONE EQUAL INTERVAL SCALING (continued)
(p. 2 of 2)

Scale Name, Number, and Number of Judges[b]	Statistical Description	Items Listed in Order Presented to Judges									
		a	b	c	d	e	f	g	h	i	j
10. Decision Making on Methods (17)	Median Scale Value[a]	6.0/	3.0*	4.5*	0.0*	7.0*	1.5*	9.0*	4.0/	2.5/	
	Interquartile Range	2.0	1.6	1.5	0.5	1.4	0.9	0.6	3.7	1.2	
11. Decision Making on Quality (16)	Median Scale Value[a]	7.0*	1.5*	9.0*	5.5*	0.0*	3.5*	6.5/	2.0*		
	Interquartile Range	1.1	1.3	0.4	1.4	0.5	1.5	1.2	0.9		
12. Figural Skills (15)	Median Scale Value[a]	5.0#	6.0/	9.0#	3.0/	0.0*	8.0/	3.5#	1.0#	7.0#	
	Interquartile Range	0.9	1.5	0.4	1.3	0.5	0.9	1.2	0.6	1.5	
13. Symbolic Skills (16)	Median Scale Value[a]	1.5*	5.0*	9.0*	7.0*	7.5/	4.0/	0.0*	3.5*		
	Interquartile Range	1.3	0.7	0.5	1.5	1.0	1.3	0.5	1.7		
14. Taxonomic Skills (15)	Median Scale Value[a]	7.0*	2.0*	9.0*	5.5*	0.0*					
	Interquartile Range	0.9	1.5	0.4	1.9	0.5					
15. Implicative Skills (15)	Median Scale Value[a]	3.5/	5.0*	2.0*	9.0*	6.5/	0.0*	1.0*	4.0*	8.0*	
	Interquartile Range	1.2	0.9	0.8	0.4	1.0	0.5	0.8	1.1	0.8	
16. Financial Conseq. of Error (15)	Median Scale Value[a]	6.0*	1.0*	9.0#	4.0*	0.0*	7.5*				
	Interquartile Range	0.7	0.8	1.1	1.1	0.5	1.1				
17. Conseq. of Error to Humans (15)	Median Scale Value[a]	5.5*	2.0*	0.0*	7.0*	9.0*	1.0*	8.0*	9.0*		
	Interquartile Range	1.0	0.7	0.5	0.6	0.4	0.8	0.5	0.9		
18. Levels of Knowledge (23)	Median Scale Value[a]	4.5/	1.5*	5.5*	3.5*	0.0*	7.0*	9.0*	2.5*	8.0*	
	Interquartile Range	1.4	1.0	1.4	1.4	0.6	0.7	0.6	1.6	0.8	

* Item was kept.
\# Item was edited.
/ Item was eliminated.

[a] Rounded.
[b] Refers to number of judges in equal interval test.

The column to the far left presents the scale name and number and indicates in parentheses the number of judges used. The next column from the left provides a line for the rounded median scale value for an item and a line for the interquartile range. Columns (a) through (j) represent the items in random order, as they were presented. A median scale value marked with an asterisk (*) indicates an item kept for the scale. A number sign (#) indicates an item kept but revised. A slash mark (/) indicates that an item was discarded.

Items were discarded for one of three reasons:

1. The item had an unacceptably high interquartile value.
2. The item was too close in value to another acceptable item to be considered to be distinguishable in the field.
3. Inclusion of the item would not permit the scale to be self-evidently cumulative.

Cumulativeness

Originally, we tested each scale to make sure that its items were cumulative. This was done by having judges indicate which other descriptors on a scale (presented at random) could be assumed to also be contained within the level represented by the given descriptor. The count for inclusions and exclusions was evaluated using probability theory to determine when there were significant indications of cumulativeness for a given descriptor. The latest scales, however, are based on selectively ascending descriptors. By eliminating items which drop on any scale principle as the scale is ascended, the cumulativeness of the scale becomes self-evident. (Since the Decision Making on Quality scale is an exception, it was tested statistically for cumulativeness.)

Reliability and Accuracy Testing

The skill scales have undergone several major revisions as a result of their being field tested for reliability and accuracy (conformity with a criterion for the correct scale values). There were three field tests.

In HSMS tests "reliability" is defined operationally. It means the extent to which several analysts, each exposed to the same training, the same situations, and the same experience with the performer, agree in their scaling of a given set of tasks. This is a measure of inter-rater reliability.[4] Our procedure involved calculation of an overall score for each scale and a score for each analyst. It thus provided information about each scale as well as about the functioning of each analyst.

The calculation was applied for each scale, using only those tasks for which at least one analyst chose a non-zero scale value. Thus, the scores do not reflect the cases of total agreement that a scale is not relevant to a task. This is therefore a more rigorous test than one including all the tasks, and an artificially high reliability or accuracy score is avoided.

---

[4] Inter-task reliability would reflect the degree of agreement on scale values for overlap tasks. That is, since the same task can appear in the job of more than one performer, it can be scaled separately each time. This calculation can be made if sufficient task overlap data are accumulated. HSMS did not test for inter-task reliability.

The Standard Reliability Score can be compared from one situation to another. The Standard Reliability Score for any analyst reflects his degree of agreement with the other analysts, in this case on a given scale. The formula for the Standard Reliability Score is as follows:

$$A_r = \frac{W}{(N-1)\ (T)}$$, where:

A_r = the Standard Reliability Score for an analyst, per scale;

W = the relevant summed agreement score (the sum of the number of other analysts who have the same scale value for a given task on a given scale);

N = the number of analysts involved in the test;

T = the total number of tasks involved for the given scale.

A_r is taken to two decimal places. Perfect inter-rater reliability for an analyst would be 1.00.

Accuracy

Our measure of accuracy was actually a measure of congruity with the group's judgment. We assumed that the scale values arrived at through discussion in the Skill Scaling Conference were more likely to be correct than any one analyst's, especially when the analysts worked independently. Using the group's scale values as the norm, we calculated the overall accuracy for each scale and the analysts' individual scores. The resulting measure is a form of reliability with respect to the group norm.

Since the accuracy measure is determined by conformity to a group judgment, a close relationship between this measure and the reliability measure could result. It is possible, however, to have high prior inter-rater agreement and then have the team members reverse their independently arrived-at judgments during the Skill Scaling Conference. In such cases the accuracy measure could be lower than the reliability measure. The most desirable result is when the accuracy measures are higher than the reliability measures for each analyst, since this indicates that the divergence among analysts is in the direction of the norm.

The Standard Accuracy Score compares the individual analyst's scale values with the group answers. The formula for the Standard Accuracy Score is as follows:

$$A_c = \frac{C}{T}, \text{ where:}$$

A_c = the Standard Accuracy Score for an analyst, per scale;

C = the number of relevant correct answers (as given by criterion);

T = the total number of tasks involved for the given scale.

A_c is taken to two decimal places. Perfect accuracy for an analyst would be 1.00.

TEST RESULTS

Table 3 presents the summary data for the HSMS pilot test. The scales are listed on the left in numerical order. The next column lists the number of tasks used as a data base (those of the 46 for which at least one analyst had a non-zero value for the scale). The Standard

Table 5. SUMMARY OF RELIABILITY AND ACCURACY DATA FOR SKILL SCALES

Brief Name of Scale	T[a]	S 1	F 2	L 3	M 4	T 5	I 6	7	Average[b]	S 1	F 2	L 3	M 4	T 5	I 6	7	Average[b]
		Analyst Name and Number: Standard Reliability Scores								Analyst Name and Number: Standard Accuracy Scores							
1. Frequency	46	.80	.81	.76	.78	.68	.79		.77	.93	.89	.85	.89	.78	.87		.87
2. Locomotion	4	.50	.50	.70	.40	.70	.60		.57	.75	.50	1.00	.50	1.00	.75		.75
3. Object Manip.	45	.44	.41	.48	.40	.35	.42		.42	.49	.29*	.51	.38*	.73	.82		.54
4. Guiding Steerng.	7	.74	.74	.69	.31	.37	.74		.60	1.00	1.00	.86	.43	.43	1.00		.79
5. Human Inter.	4[illegible]	.42	.48	.47	.54	.42	.47		.47	.52	.57	.63	.87	.61	.74		.66
6. Leadership	31	.36	.26	.23	.32	.31	.30		.30	.97	.16*	.26	.23*	.23*	.84		.45
7. Oral Lang.	46	[illegible]	.78	.78	.04	.74	.78		.65	1.00	1.00	1.00	.02*	.89	1.00		.82
8. Reading Lang.	46	.79	.70	.80	.75	.52	.78		.72	.93	.74	.96	.85	.57	.93		.83
9. Written Lang.	46	.91	.80	.91	.85	.86	.90		.87	1.00	.80	.98	.89	.89	.98		.92
10 Methods	46	.47	.27	.42	.37	.43	.48		.41	.67	.30	.54	.72	.67	.78		.61
11 Quality	46	.61	.62	.57	.35	.40	.52		.51	.87	.91	.72	.41	.46	.78		.69
12 Figural	45	.48	.16	.50	.47	.45	.48		.42	.89	.20	.53	.51	.47	.93		.59
13 Symbolic	38	.56	.26	.64	.63	.62	.62		.55	.66	.34	.82	.79	.74	.92		.71
14 Taxonomic	45	.55	.15	.48	.52	.55	.52		.46	.78	.20	.64	.69	.80	.78		.65
15 Implicative	46	.25	.33	.27	.29	.25	.30		.28	.50	.54	.39	.50	.46	[illegible]		.50
16 Finan. Err.Con	42	.71	.60	.69	.49	.60	.68		.63	.88	.81	.81	.52	.64	.93		.76
17 Human Err.Con	46	.61	.64	.54	.55	.60	.61		.59	.70	.83	.63	.61	.63	.89		.71
Col. Average		.59	.50	.58	.47	.52	.59			.80	.60	.71	.56	.65	.85		

[a] Total number of tasks for each scale. [b] Line average of analysts' scores. *Accuracy less than reliability.

Reliability Scores for each of the six analysts follow, and then the average across analysts. The Standard Accuracy Scores are presented the same way, on the right-hand side of the table. The analysts' overall averages are listed at the bottom.

Ten scales showed acceptable levels of reliability (.50 or higher). Revisions were then made for those scales requiring improvement. Given the fact of limited training time and the need to unlearn older versions of some of the scales, and given the tendency for reliability to increase over time, the results suggest that, with some revisions (made later), future results would be acceptable.

KNOWLEDGE

Generally, the "reliability" tested and measured by HSMS refers to the extent to which several analysts in a team, exposed to the same performer at the same time, will independently arrive at the same data results. This is a measure of inter-rater reliability. In the case of knowledge identification HSMS adopted this measure for preliminary testing. However, there are two limits to such a reliability test to bear in mind:

1. It is impossible to measure the extent to which analysts will be affected by other analysts' questioning of the performer about knowledge categories. Therefore, the independence of the raters is not totally assured.

2. The HSMS test of knowledge category identification applies only to those subjects in the Knowledge System which are relevant to the tasks covered in the test. Therefore, the reliability figures describe reliability for only a part of the Knowledge System.

The test for validity was translated to an accuracy measure, i.e., a measure of congruity with the group judgment, as was the case with skill scaling. We assumed that data arrived at through discussion in the team conference are more likely to be correct than any single analyst's, especially when the analysts work independently.

Knowledge Identification Reliability and Accuracy Tests

Because of time pressures, the tasks of only one performer were involved in the testing. Ten Nurse Practitioner's tasks were selected. The tasks cover a range of the performer's activities and a range of knowledge requirements. Of the ten tasks, three required over fifty knowledge categories (as determined in the team conference) and seven required fewer than seven categories.

Standard Reliability Scores were calculated for each task separately and for each analyst. Averages were then calculated across tasks and across analysts. In the case of knowledge identification, the Standard Reliability is as follows:

$$A_r = \frac{W}{(N-1)(K)}, \text{ where:}$$

A_r = the Standard Reliability Score for an analyst per task;

W = the relevant summed agreement score (the sum of the number of other analysts who identified (or did not identify) all of the categories identified by any of the analysts for a given task);

N = the number of analysts involved in the test;

K = the total number of separate knowledge categories identified by any and all of the analysts, per task.

A_r is taken to two decimal places. Perfect reliability for an analyst would be 1.00.

Standard Accuracy Scores were calculated for each task separately and for each analyst. Averages across tasks and across analysts were also calculated.

The formula for the Standard Accuracy Score in cases where each analyst may have identified a different number of knowledge categories per task is as follows:

$$A_c = \frac{C}{K}$$, where:

A_c = the Standard Accuracy Score for an analyst, per task;

C = the analyst's total correctly identified categories (as determined by the team conference), minus the total number of categories identified by the analyst in excess of the correct total. (If the result is a negative number, it is entered as a zero.);

K = the total number of correct knowledge categories (as per the criterion).

A_c is taken to two decimal places. Perfect accuracy for an analyst would be 1.00.

Results

Table 4 presents the results of the testing. The tasks are listed on the top row in order of the total number of categories identified by the team for the task, from left to right. The analysts are listed in the far column. The upper portion of the table refers to reliability, and the lower portion refers to accuracy.

Table 4. SUMMARY OF PILOT TEST RELIABILITY AND ACCURACY DATA FOR KNOWLEDGE IDENTIFICATION

Analysts	Tasks by Total No. of Correct Categories											
	61	52	51	9	5	5	5	Average[a]	3	3	2	Average[b]
	Standard Reliability Scores											
(S) 1	.66	.62	.65	.57	.77	.73	.60	.66	.60	.77	.69	.67
(F) 2	.61	.61	.55	.54	.43	.55	.60	.56	.52	.71	.47	.56
(L) 3	.55	.61	.54	.50	.64	.65	.70	.60	.68	.74	.60	.62
(M) 4	.66	.51	.59	.66	.77	.49	.70	.63	.28	.25	.69	.56
(T) 5	.52	.64	.61	.63	.71	.69	.70	.64	.68	.77	.60	.66
(I) 6	.58	.60	.66	.68	.81	.55	.60	.64	.68	.71	.67	.65
Average	.60	.60	.60	.60	.69	.61	.65	.62	.57	.66	.62	.62
Analysts	Standard Accuracy Scores											
(S) 1	.72	.75	.74	.33*	.80	.80	.80	.70	.67	1.00	.50*	.71
(F) 2	.61	.75	.55	.44*	.00*	.20*	.40*	.42*	.33*	.67*	.00*	.40*
(L) 3	.52*	.48*	.47*	.56	.60*	.60*	.60*	.55*	1.00	.67*	.00*	.55*
(M) 4	.59*	.42*	.33*	.56*	.80	.60	.60*	.56*	.00*	.00*	.50*	.44*
(T) 5	.43*	.44*	.63	.44*	.60*	.60*	.60*	.53*	.67*	.67*	.00*	.51*
(I) 6	.75	.73	.92	1.00	1.00	.60	.80	.83	1.00	.33*	1.00	.81
Average	.60	.59*	.61	.55*	.63*	.56*	.63*	.60	.61	.56*	.33*	.57*

* Accuracy lower than reliability.
[a] Average of 7 tasks.
[b] Average of 10 tasks.

The data averages are shown for seven tasks and for ten tasks because the tasks with as few as two or three categories show somewhat distorted results for Analyst 4. The overall reliability averages are not greatly changed, but the accuracy average is somewhat higher using seven tasks.

The results indicate generally high average reliability by analyst, with none lower than .56, and an overall average of .62. On the other hand, the accuracy data indicate some problems in either over- or under-identification with at least one or two analysts. The overall accuracy is somewhat below the reliability average (.60 for seven tasks and .57 for ten). This reflects the deviation from the group norm for Analyst 2 in particular, and Analyst 4 in the ten-task average. Analysts 1 and 6 show high accuracy. Analyst 6 was the trainer in this case, and influenced the team decisions in the direction of the content of the method. Analyst 1, the field team leader, showed high agreement with the correct identifications.

Since this was the first test of the method, the results are encouraging; however, it is clear that team agreements, careful training, and review are necessary components of reliability.

Knowledge Scale Reliability and Accuracy

For testing reliability in the use of the knowledge scale, the units of observation are the knowledge categories rather than the tasks. A small number of Nurse Practitioner tasks provided over 100 observations for use in the test of the knowledge scale. However, the full range of the scale could not be tested, since none of the categories identified were scaled above 5.5 on the knowledge scale.

The "correct" knowledge category identifications for a given task became the framework for testing the reliability of the scale. Unlike the case of knowledge category identification, a fixed number of observations were involved. We calculated inter-rater reliability and an accuracy measure based on agreement with the scale values determined at the team conference.

The formula for the Standard Reliability Score is:

$$A_r = \frac{W}{(N-1)(K)}$$, where:

A_r = the Standard Reliability Score for an analyst;

W = the relevant summed agreement score (the number of other analysts choosing the same scale value, for each category);

N = the number of analysts involved in the test;

K = the total number of categories involved in the entire test.

The Standard Reliability Score was calculated separately for each analyst, and an average was taken for the scale as a whole.

The formula for the Standard Accuracy Score is:

$$A_c = \frac{C}{K}$$, where:

A_c = the Standard Accuracy Score for an analyst;

C = the number of correct answers as determined by the team conference;

K = the total number of categories involved for the test.

The Standard Accuracy Score was calculated for each analyst separately, and an average was then taken for the scale as a whole.

Table 5 reports the test results. The scale shows a relatively lower average reliability (.47) and a much higher average accuracy. The reason is again due to analyst deviation. Without Analyst 4, overall reliability is .55. However, unlike the case with other low reliability scales, the analysts (other than Analyst 4) were never more than one scale point away from the value arrived at in the team conference. Since the reliability and accuracy measures treat <u>any</u> difference as equally wrong and magnify the errors, the results are more acceptable than may first appear. The lesson again is that training, team agreement, and review must be part of the method.

Table 5.

SUMMARY OF PILOT TEST RELIABILITY AND ACCURACY FOR KNOWLEDGE SCALE

Statistical Measure	Analyst Name and Number						
	S	F	L	M	T	I	Aver-
	1	2	3	4	5	6	age
Standard Reliability Score	.49	.53	.46	.38	.45	.52	.47
Standard Accuracy Score	.60	.73	.64	.38	.54	.88	.63

K = 113 separate knowledge categories
N = 6 analysts

APPENDIX B

THE HSMS "EDIT" PROGRAM

INTRODUCTION

The HSMS "EDIT" program was designed by Stephan Jasik and modified by Edward Friedman and George Chaikin to prepare the HSMS task data for use with the HSMS factor analysis programs PCVARIM and X2MOFA (Two-Mode Factor Analysis, Part One).

The HSMS method calls for an unforeseeable number of keypunched data cards for the given set of tasks being studied. The number of cards cannot be predicted because the number of knowledge categories needed for tasks varies, and the total number of knowledge categories identified for an entire set of tasks cannot be known ahead of time. Since it is impossible to predict the identity and number of knowledge categories, HSMS selected a data card format to handle required knowledge categories as they are identified for the task, without knowing which are needed for other tasks. The format is not predetermined in terms of preselected knowledge categories. Such a format could require thousands of cards, most of which would be blank to indicate zero scale requirements for categories. Instead, the EDIT program creates a matrix of tasks by skill and knowledge categories; it enters zeroes in this matrix whenever a category is not required by a task.

A data unit consists of all the cards for a given task, for which the ID name is the task's Code Number. For any given task, the task's Code Number appears on each card. Card TOO includes information on how many cards will follow for the given task and an abbreviated name of the task. Card T01 is in a fixed format, where fields refer to skill scales; it presents the task's scale values for each of the 16 skill scales. Any other cards are for knowledge categories. These are numbered from T02 to T0*n*, and are set up so that each 8-digit number of the knowledge categories identified for the particular task is punched, each followed by its scale value.

The EDIT program provides the user with information on the number, identity, and frequency of the skills and knowledge categories; it orders the categories, provides checks on the data, and makes possible selection and/or logarithmic transformation of the data for statistical use. EDIT therefore interfaces with PCVARIM and X2MOFA (Appendixes C and D), and is always run when they are run.

EDIT also provides dictionaries for the tasks and skill and knowledge categories in which the tasks' names appear in abbreviated form, and tasks and variables are shown with their internal numbers and Code Numbers. The other HSMS programs (except for MATRIX) refer to the data by their internal numbers.

DESCRIPTION

EDIT is written in FORTRAN IV and was used in the Control Data Corporation's (CDC) 6600 computer at the Courant Institute of Mathematical Sciences of New York University. The operating systems in use during HSMS analyses were KRONOS and NOS (Network Operating System). EDIT was stored on magnetic tape in compiled and loaded form (i.e., in binary object code in non-relocatable form) and in OLDPL form.

The HSMS task data are transferred to magnetic tape in the form of an OLDPL. The data were maintained in numerical Task Code order in the OLDPL. A utility program, UPDATE, is used to transfer data "decks" to tape, correct data, or to generate temporary local files in the form of a compile file, which is part of the input file for EDIT. The input file for UPDATE is magnetic tape in OLDPL form, and the output file is C=DATA. The input file for EDIT is DATA, and the output file is TAPE9 (with optional printed output).

The EDIT program is designed to receive the data decks in any order, but always with the cards for a given task <u>in numerical order by card number</u>, beginning with Card T00. EDIT then performs the functions described below.

<u>Error Checks and Listing of Data</u>

EDIT provides a variety of error checks and the option of listing or not listing the data cards as they appear in the input file. The listing is a default setting, and a NOLIST card is required to suppress it. Some of the checks are suppressed by selection of NOLIST; others are carried out regardless of the option selected. The error checks and whether they are suppressed with NOLIST are shown below. HSMS uses NOLIST when EDIT is used with PCVARIM and X2MOFA.

1. A check that the number of cards for the task is consistent with the number indicated on card T00. Suppressed by NOLIST.

2. A check that all the cards for a task have the same Task Code Number. Suppressed by NOLIST.

3. A check that data cards appear in proper sequence (as indicated in columns 1-3). Suppressed by NOLIST.

4. A check that each task (Task Code Number) appears only once in the set of data. Suppressed by NOLIST.

5. A check that all punches occur in permissible columns; error messages indicate that either a blank or a punch should appear in a given column. This check is always made.

6. A check that skill scale values all end in 0 or 5. This check is always made.

7. A check that all knowledge categories have a scale value above 00. Suppressed by NOLIST.

8. A check that knowledge categories appear only once for a given task. This check is always made.

Options to Reduce the Number of Variables

DELETE and CUTOFF are two options which make it possible to reduce the number of skill and knowledge variables which are copied to TAPE9. These options are used when EDIT interfaces with PCVARIM and X2MOFA (which are dimensioned for no more than 145 variables). DELETE is used to name specific knowledge and skill categories to be eliminated; CUTOFF is used to specify the frequency at or below which skill and knowledge categories are automatically eliminated.

Ordered Listings of the Data

EDIT provides successive rearrangements of the data base (which exists as a matrix in which the skill and knowledge categories are the columns and the tasks are the rows), and lists the resulting information in various formats. These arrangements are designed to help the user select a frequency for CUTOFF, categories for DELETE, and listings of the data in orders and forms useful for work with other programs. The data are structured as follows:

1. Each skill and knowledge category is given an internal number and is listed in the order in which it appears in the input file, together with its frequency, and the Task Code Number and scale value for each task in which it is scaled above 00. (This listing could be eliminated were the EDIT program to be revised.)

2. The skill and knowledge categories are listed in descending order of their frequency of occurrence in the tasks. The original internal numbers, the Task Code Numbers, and the scale values appear again.

3. The skill and knowledge categories are listed with the skills first (in a preset order) followed by the knowledge categories in ascending numerical order of their 8-digit code numbers. The original internal numbers, frequency, the Task Code Numbers and the scale values appear again. (This listing could be reduced to just the identification numbers and frequencies were the program to be revised.)

4. The variables copied to TAPE9 are renumbered internally and listed in order with the skills first, followed by the knowledge categories in ascending order of their 8-digit code numbers. Frequency data are listed, but no task and scale value data. If DELETE and CUTOFF options have been selected this listing does not include the eliminated categories. (In the previous three listings categories to be eliminated as a result of DELETE or CUTOFF options are indicated by asterisks.)

5. EDIT provides a "Task Description Dictionary" which lists each task's observation number (its internal number in numerical order) and its actual Task Code Number, together with the abbreviated name of the task as found in card T00 for the task.

6. The final numbers assigned to skill and knowledge categories and to tasks are the numerical references for observations (tasks) and variables (skill and knowledge categories) in PCVARIM, X2MOFA, and X2MFA2.

Logarithmic Transformation of Data

When EDIT is used to interface with PCVARIM or X2MOFA, the NORMALIZE option permits a logarithmic transformation of the data to adjust for a large number of zeroes in the data base. (Tasks which do not require knowledge categories required by any other task are scaled at zero.) The NORMALIZE option follows the formula:

X = SQRT (X + 0.5), where X is a scale value.

COMMENTS

EDIT places heavy demands on a computer's core memory. The size of the matrix array of data (36000 cells) is such that it was necessary to compress the data so that 3 units are stored in a word. In its present form EDIT can be used only with a computer that has a 60-bit word. EDIT requires 200K of core memory to run, and is dimensioned to handle up to 700 tasks, assuming a task-by-category matrix of 36000 cells. Because there is a wide range in the number and frequencies of variables that can be associated with a set of tasks, it is virtually impossible to estimate beforehand the precise number of tasks which can actually be handled in a particular run: HSMS has successfully run EDIT with as many as 560 tasks. With a data base larger than this there may be a risk that a time limit may be reached before all of the functions of the program have been carried out, even though the printed output will be complete.

The EDIT program was designed early in the history of HSMS, and, therefore, includes procedures and options which we now see can be eliminated or which should be able to be suppressed. For example, once new data are checked and variables have been selected for use in PCVARIM or X2MOFA, the EDIT checks, restructurings, and most listings are unnecessary. The reader is encouraged to revise EDIT to make it more efficient. HSMS would have done so were it to have continued its operations.

SYSTEM SUBROUTINES CALLED BY EDIT

The EDIT program which is presented in this appendix calls for a series of subroutines. Among these, EXIT, TIME, and DATA have not been included in the listing, since they are systems-based. It is assumed that the user can utilize comparable routines after reading the descriptions which follow.[1]

EXIT

This subroutine terminates program execution and returns control to the operating system. A STOP statement may be preferable.

TIME(a)

This subroutine can be used as a function or subroutine. The value is returned via the argument and the normal function return. The subroutine returns the current reading of the system clock as the value of the argument a or of the function in the form 10Hbhh.mm.ss.b, where b denotes a blank, and hh, mm, and ss are the number of hours, minutes, and seconds, respectively. The value returned is Hollerith

---

[1] INPAGE is called and does appear in EDIT. It is an entry point appearing in PAGER.

data and can be output using an A format specification. The type of this format is real.

DATE(a)

This subroutine can be used as a function or subroutine. See CALL TIME(a), above. The current date is returned as the value of argument a or of the function in the form 10Hbmm/dd/yyb (unless it is changed at installation option), where b denotes a blank, mm is the number of the month, dd is the number of the day within the month, and yy is the year. The value returned is Hollerith data and can be output using an A format specification. The type of this function is real.

STRUCTURING THE INPUT FILE: EDIT WHEN USED ALONE

In the set-up presented below, the NOS system in use in 1977 at the Courant Institute with the CDC-6600 is assumed. It is also assumed that the reader can refer to the UPDATE Reference Manual or a counterpart program. In the following instructions b denotes a blank.

INPUT FILE FOR EDIT USED ALONE

Name	Keypunch Cards (one per box)	Instructions
Systems Cards	_ _ _ _ .EDIT ALONE	Identification number and other information depending on system.
	USER(_ _ _ _ _ _ _ _)	USER and user code.
	CHARGE(_ _ _ _ _ _ _ _ _ _ _ _ _ _)	Charge card.
	HEADER.EDIT ALONE	Optional to print out heading across a whole page.
	LABEL(OLDPL,VSN=T_ _ _)	Identifies OLDPL tape.
	UPDATE(Q,D,8,C=DATA) or UPDATE(F,D,8,C=DATA)	When selecting tasks using *COMPILE form of UPDATE. When using all data in OLDPL.
	UNLOAD(OLDPL)	Unless being used for more than one run per submission.
	LABEL(PROGRMS,VSN=T_ _ _)	Identifies program tape.
	COPYBF(PROGRMS,EDIT)	Copies program to local file.
	SETTL(1000)	Time limit.
	RFL(200000)	Defines field length.
	EDIT.	
	End of Record Card	
Input Cards for UP DATE	*COMPILEb _ _ _,_ _ _,_ _ _ etc., or *COMPILEb _ _ _,_ _ _._ _ _ (option)	Calls tasks from OLDPL. Each card starts: *COMPILE in Cols. 1-8; Column 9 blank, followed by Task Code Numbers separated by commas; last entry has no comma. Tasks can be in any order. Consecutive Code Numbers can be called as last entry on card after comma or as only entry on card by writing first Code Number, then a period, then last Code Number of the series. A new card must follow.
	End of Record Card	Needed even if no UPDATE cards

INPUT FILE FOR EDIT USED ALONE (continued)

Name	Keypunch Cards (one per box)	Instructions
Input Cards for EDIT	TITLEb b b b b _ _ _ _ _ _ _ _ etc.	Columns 1-5: punch TITLE; Columns 11-60: punch title of program, date, and any other special identification of this run.
	NOLIST (option)	Only if listing of data and some error checks are to be suppressed.
	DELETEb b b b _ _ _ _ _ _ _ _ (option)	One card for each individual skill or knowledge category to be deleted from input data (if not to be deleted by CUTOFF). Columns 1-6: punch DELETE; Columns 11-18: punch 8-character skill code or 8-digit knowledge category code number.
	CUTOFFb b b b _ _ (option)	To eliminate skill or knowledge categories at or below a selected frequency from input data. Columns 1-6: punch CUTOFF; Columns 11-12: punch selected frequency (usually 4 or more) left justified in field.
	NORMALIZE (option)	Not usually used when EDIT is used alone.
	End of File Card	

```
      OVERLAY(EDIT,0,0)                                                  EDITFA
      PROGRAM EDITFA(INPUT,OUTPUT,DATA,TAPE9,TAPE1=DATA,TAPE5=INPUT,     EDITFA
     1   TAPE6=OUTPUT)                                                   EDITFA
                                                                         EDCOM
      COMMON/TABLES/IBOT,ITTAB(700),KC(700),KFREQ(700),KPT(700),IW(700), ENLRG
     1 NXT,MTAB(36000),NTASK,MAXTAB,MAXKC                                ENLRG
                                                                         EDCOM
      COMMON/COLS/IT00(3,2),ITON(2,2),IT01(16,2),IT02(8,2)               EDCOM
                                                                         EDCOM
      COMMON/IO/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)           ENLRG
                                                                         EDCOM
      COMMON/SKILLS/KEYWD(16),KEYALB(16)                                 EDCOM
                                                                         EDCOM
      COMMON/DELIST/NDEL,LDEL(700),KUTOFF                                ENLRG
                                                                         EDCOM
      COMMON/LOG/TT1,LNORM,LPRINT                                        EDCOM
                                                                         EDCOM
      COMMON/TCARD/ITDCN,ITIN,NC                                         EDCOM
                                                                         EDCOM
                                                                         EDCOM
      LOGICAL TT1,LNORM,LPRINT                                           EDCOM
      LOGICAL SEARCH,LS,NEWTSK                                           EDCOM
                                                                         EDCOM
                                                                         EDCOM
      DIMENSION KCREAD(8)                                                EDCOM
      DIMENSION KCBUFF(700),ISKV(700)                                    ENLRG
      DIMENSION LIST(3),LISTR(2)                                         EDCOM
      DIMENSION PRLIST(700)                                              ENLRG
      DIMENSION IFMAT(3)                                                 EDCOM
                                                                         EDCOM
      EQUIVALENCE (LIST(1),ITDCN),(LISTPR(1),PRLIST(1))                  EDCOM
                                                                         EDCOM
      DATA NXT/1/,IBOT/0/,MAXTAB/36000/,MAXKC/700/                       ENLRG
                                                                         INTDTA
      DATA TT1/.TRUE./,LNORM/.FALSE./,LPRINT/.TRUE./                     INTDTA
```

```
                                                                         INTDTA
                                                                         INTDTA
                                                                         INTDTA
      DATA KEYWD/8RLOCOMOTN,8ROBJ MANP,8RGDG-STRG,8RHUM INTR,8RLEADSHIP, INTDTA
     28RORAL USE,8RREAD USE,8RWRIT USE,8RMETHODS ,8RQUALITY ,8RFIGURAL , INTDTA
     38RSYMBOLIC,8RTAXONOMC,8RIMPLICIT,8RFINC ERR,8RHUMN ERR/            INTDTA
                                                                         INTDTA
      DATA KEYALB/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16/                INTDTA
                                                                         INTDTA
                                                                         INTDTA
      DATA (IT00(I),I=1,3)/2,5,29/,                                      INTDTA
     2     (IT00(I),I=4,6)/3,10,30/                                      INTDTA
      DATA (ITON(I),I=1,2)/2,5/,                                         INTDTA
     2     (ITON(I),I=3,4)/3,10/                                         INTDTA
      DATA (IT01(I),I= 1,16)                                             INTDTA
     1 /32,35,38,41,44,47,50,53,56,59,62,65,68,71,74,77/,                INTDTA
     X     (IT01(I),I=17,32)                                             INTDTA
     2 /33,36,39,42,45,48,51,54,57,60,63,66,69,72,75,78/                 INTDTA
      DATA (IT02(I),I=1, 8) /30,39,43,52,56,65,69,76/,                   INTDTA
     2     (IT02(I),I=9,16) /37,40,50,53,63,66,76,79/                    INTDTA
                                                                         INTDTA
                                                                         INTDTA
      INPUT = 5                                                          INTDTA
      IOUT = 6                                                           EDITFA
      ITAPE = 9                                                          EDITFA
      CALL OVERLAY(4HEDIT,1,0)                                           EDITFA
      INPUT = 1                                                          EDITFA
      REWIND ITAPE                                                       EDITFA
      CALL OVERLAY(4HEDIT,2,0)                                           EDITFA
      REWIND ITAPE                                                       EDITFA
      CALL OVERLAY(4HEDIT,3,0)                                           EDITFA
      REWIND ITAPE                                                       EDITFA
      CALL EXIT                                                          EDITFA
      END                                                                EDITFA
                                                                         EDITFA
```

```
      SUBROUTINE PAGER(I)                                               PAGER
                                                                        PAGER
      COMMON/SKIP/LINSKP                                                PAGER
      COMMON/IO/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)          ENLRG
      COMMON/LOG/TT1,LNORM,LPRINT                                       PAGER
      LOGICAL LPRINT                                                    PAGER
      DIMENSION I(1),IHEAD(12),ISUBHD(11)                               PAGER
      DATA J/0/,K/0/,NPAGE/0/,IHEAD/12*10H                 /,           PAGER
     1ISUBHD/11*10H          /                                          PAGER
                                                                        PAGER
C   AT THIS ENTRY POINT THE PARAMETER I IS NOT USED                    PAGER
                                                                        PAGER
   10 NPAGE = NPAGE + 1                                                 PAGER
      WRITE(IOUT,1)IHEAD,NPAGE                                          PAGER
    1 FORMAT(1H1,12A10,*PAGE*,I4)                                       PAGER
      WRITE(IOUT,2) ISUBHD                                              PAGER
    2 FORMAT(1H0,11A10/)                                                PAGER
      LINCNT = 2                                                        PAGER
      LINSKP = 0                                                        PAGER
      RETURN                                                            PAGER
      ENTRY INPAGE                                                      PAGER
      K = K + 1                                                         PAGER
      IF (K.GT.1) GO TO 30                                              PAGER
      DO 20 L = 1,7                                                     PAGER
   20 IHEAD(L) = I(L)                                                   PAGER
      CALL TIME(IH)                                                     PAGER
      IHEAD(9) = IH                                                     PAGER
      CALL DATE(IH)                                                     PAGER
      IHEAD(11) = IH                                                    PAGER
      ISUBHD(1) = 10HCARD NO.                                           PAGER
      ISUBHD(2) = 10HINPUT DATA                                         PAGER
      GO TO 10                                                          PAGER
```

```
30 IF ((K.EQ.2).AND.LPRINT) GO TO 10                                  PAGER
   IF (K.EQ.2) RETURN                                                 PAGER
   L = K - 2                                                          PAGER
   GO TO (40,70,80,90,100,110)L                                       PAGER
40 ISUBHD( 1) = 10H INT    VAR                                        PAGER
   ISUBHD( 2) = 10H CODE    FR                                        PAGER
   ISUBHD( 3) = 10HEQ                                                 PAGER
   ISUBHD( 4) = 10HTASKS IN W                                         PAGER
   ISUBHD( 5) = 10HHICH VARIA                                         PAGER
   ISUBHD( 6) = 10HBLE APPEAR                                         PAGER
   ISUBHD( 7) = 10HED  -  PRI                                         PAGER
   ISUBHD( 8) = 10HNTED AS VA                                         PAGER
   ISUBHD( 9) = 10HRIABLE COD                                         PAGER
   ISUBHD(10) = 10HES APPEARE                                         PAGER
   ISUBHD(11) = 10HD IN INPUT                                         PAGER
50 NPAGE = NPAGE + 1                                                  PAGER
   IX = 0                                                             PAGER
   WRITE(IOUT,1)IHEAD,NPAGE                                           PAGER
   DO 60 L = 1,20                                                     PAGER
60 WRITE(IOUT,3)                                                      PAGER
 3 FORMAT(1H )                                                        PAGER
   WRITE(IOUT,4)(ISUBHD(L),L=4,11)                                    PAGER
 4 FORMAT(26X,8A10)                                                   PAGER
   WRITE(IOUT,3)                                                      PAGER
   WRITE(IOUT,5)I                                                     PAGER
 5 FORMAT(40X,I4,* DISTINCT VARIABLE CODES WERE FOUND IN THE INPUT DA PAGER
  1TA*////32X,*ASTERISKS MARK VARIABLES TO BE DELETED DUE TO CUTOFF ( PAGER
  2K DIRECTIVE*)                                                      PAGER
   GO TO 10                                                           PAGER
70 ISUBHD( 7) = 10HED  -  SOR                                         PAGER
   ISUBHD( 8) = 10HTED BY FRE                                         PAGER
   ISUBHD( 9) = 10HQUENCY OF                                          PAGER
   ISUBHD(10) = 10HVARIABLE C                                         PAGER
   ISUBHD(11) = 10HODES                                               PAGER
   GO TO 50                                                           PAGER
```

```
 80 ISUBHD( 8) = 10HTED BY ASC                                  PAGER
    ISUBHD( 9) = 10HENDING ORD                                  PAGER
    ISUBHD(10) = 10HER OF VARI                                  PAGER
    ISUBHD(11) = 10HABLE CODES                                  PAGER
    GO TO 50                                                    PAGER
 90 ISUBHD( 7) = 10HED  -  DEL                                  PAGER
    ISUBHD( 8) = 10HETED VARIA                                  PAGER
    ISUBHD( 9) = 10HBLES - NOT                                  PAGER
    ISUBHD(10) = 10H WRITTEN O                                  PAGER
    ISUBHD(11) = 10HNTO TAPE9                                   PAGE
    GO TO 50                                                    PAGER
100 ISUBHD( 1) = 10HVARIABLE N                                  PAGER
    ISUBHD( 2) = 10HO.  VAR CO                                  PAGER
    ISUBHD( 3) = 10HDE  FREQ                                    PAGER
    ISUBHD( 4) = 10H  FINAL CO                                  PAGER
    ISUBHD( 5) = 10HRRESPONDEN                                  PAGER
    ISUBHD( 6) = 10HCE OF VARI                                  PAGER
    ISUBHD( 7) = 10HABLE NUMBE                                  PAGER
    ISUBHD( 8) = 10HRS TO VARI                                  PAGER
    ISUBHD( 9) = 10HABLE CODES                                  PAGER
    ISUBHD(10) = 10H AS WRITTE                                  PAGER
    ISUBHD(11) = 10HN ON TAPE9                                  PAGER
    GO TO 50                                                    PAGER
110 ISUBHD( 1) = 10HOBSERVATIO                                  PAGER
    ISUBHD( 2) = 10HN NO.   TA                                  PAGER
    ISUBHD( 3) = 10HSK NO.                                      PAGER
    ISUBHD( 4) = 10HTASK DESCR                                  PAGER
    ISUBHD( 5) = 10HIPTION DIC                                  PAGER
    ISUBHD( 6) = 10HTIONARY                                     PAGER
    ISUBHD(7) = 10H                                             PAGER
    ISUBHD(8) = 10H                                             PAGER
    ISUBHD(9) = 10H                                             PAGER
    ISUBHD(10)= 10H                                             PAGER
    ISUBHD(11)= 10H                                             PAGER
    GO TO 10                                                    PAGER
    END                                                         PAGER
```

```
      LOGICAL FUNCTION SEARCH(ITEM,IND,LIST)                            SEARCH
      DIMENSION LIST(IND)                                               SEARCH
      SEARCH = .TRUE.                                                   SEARCH
      IF (ITEM.EQ.0) RETURN                                             SEARCH
      IF (IND.EQ.0) GO TO 20                                            SEARCH
      DO 10 I = 1,IND                                                   SEARCH
      IF (LIST(I).NE.ITEM) GO TO 10                                     SEARCH
      SEARCH = .FALSE.                                                  SEARCH
      RETURN                                                            SEARCH
   10 CONTINUE                                                          SEARCH
   20 IND = IND + 1                                                     SEARCH
      LIST(IND) = ITEM                                                  SEARCH
      RETURN                                                            SEARCH
      END                                                               SEARCH

      OVERLAY(EDIT,1,0)                                                 RODIR
      PROGRAM RODIR                                                     RODIR
                                                                        RODIR
      DIMENSION IDIR(6),IC(7),JC(10)                                    RODIR
                                                                        EDCOM
      COMMON/TABLES/IPCT,ITTAB(700),KC(700),KFREQ(700),KPT(700),IN(700), ENLRG
     1 NXT,MTAB(36000),NTASK,MAXTAB,MAXKC                               ENLRG
                                                                        EDCOM
      COMMON/COILS/IT00(3,2),ITOM(2,2),IT01(15,2),IT02(3,2)             EDCOM
                                                                        EDCOM
      COMMON/IO/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)          ENLRG
                                                                        EDCOM
      COMMON/SKILLS/KEYWD(16),KEYALB(16)                                EDCOM
                                                                        EDCLM
      COMMON/DELIST/NDEL,LDEL(700),KUTOFF                               ENLRG
                                                                        EDCOM
      COMMON/LOG/TT1,LNORM,LPRINT                                       EDCOM
                                                                        EDCOM
      COMMON/TCARD/ITDCN,ITIN,NC                                        EDCOM
```

```
                                                                    EDCOM
                                                                    EDCOM
      LOGICAL TT1,LNORM,LPRINT                                      EDCOM
      LOGICAL SEARCH,LS,NEWTSK                                      EDCOM
                                                                    EDCOM
                                                                    EDCOM
      DIMENSION KCREAD(8)                                           EDCOM
      DIMENSION KCBUFF(700),ISKV(700)                               ENLRG
      DIMENSION LIST(3),LISTR(2)                                    EDCOM
      DIMENSION PRLIST(700)                                         ENLRG
      DIMENSION IFMAT(3)                                            EDCOM
                                                                    EDCOM
      EQUIVALENCE (LIST(1),ITDCN),(LISTPR(1),PRLIST(1))             EDCOM
                                                                    EDCOM
      DATA IDIR/10HTITLE     ,10HDELETE    ,10HCUTOFF    ,10HNORMALIZE ,  RDDIR
     1 10HNOLIST    ,10H          /,IAR/10H↑↑↑↑↑↑↑↑↑↑/               RDDIR
      DATA IBL/10H          /                                       RDDIR
                                                                    RDDIR
                                                                    RDDIR
      NDEL = 0                                                      RDDIR
      KUTOFF = 0                                                    RDDIR
      NUM = 0                                                       RDDIR
   10 READ(INPUT,1)ID,IC                                            RDDIR
    1 FORMAT(8A10)                                                  RDDIR
      IF(EOF(INPUT))20,30                                           RDDIR
   20 RETURN                                                        RDDIR
   30 NUM = NUM + 1                                                 RDDIR
      DO 40 I=1,6                                                   RDDIR
      IF (ID.EQ.IDIR(I)) GO TO 50                                   RDDIR
   40 CONTINUE                                                      RDDIR
      LINCNT = LINCNT + 2                                           RDDIR
      IF (LINCNT.GT.55) CALL PAGER(56)                              RDDIR
      WRITE(IOUT,2)NUM,ID,IC                                        RDDIR
      WRITE(IOUT,3)                                                 RDDIR
```

```
    3 FORMAT(7X,8(1H↑),* - UNRECOGNIZABLE DIRECTIVE*)                    RDDIR
      GO TO 10                                                           RDDIR
   50 GO TO (100,200,300,400,500,1000)I                                  RDDIR
  100 IF (TT1) GO TO 110                                                 RDDIR
      LINCNT = LINCNT + 2                                                RDDIR
      IF (LINCNT.GT.55) CALL PAGER(56)                                   RDDIR
      WRITE(IOUT,2)NUM,ID,IC                                             RDDIR
      WRITE(IOUT,4)                                                      RDDIR
    4 FORMAT(7X,8(1H↑),* - A TITLE HAS ALREADY BEEN READ*)               RDDIR
      GO TO 10                                                           RDDIR
  110 CALL INPAGE(IC)                                                    RDDIR
      TT1 = .FALSE.                                                      RDDIR
      GO TO 1000                                                         RDDIR
  200 DECODE(10,9,IC)JC                                                  RDDIR
    9 FORMAT(10R1)                                                       RDDIR
      IF(JC(1)-1R0)210,230,230                                           RDDIR
  210 DECODE(8,11,IC)JC                                                  RDDIR
   11 FORMAT(R8)                                                         RDDIR
      DO 220 I=1,16                                                      RDDIR
      IF (JC.EQ.KEYWD(I)) GO TO 240                                      RDDIR
  220 CONTINUE                                                           RDDIR
      LINCNT = LINCNT + 2                                                RDDIR
      IF (LINCNT.GT.55) CALL PAGER(56)                                   RDDIR
      WRITE(IOUT,2)NUM,ID,IC                                             RDDIR
      WRITE(IOUT,5)                                                      RDDIR
    5 FORMAT(17X,8(1H↑),*- UNRECOGNIZABLE SKILL*)                        RDDIR
      GO TO 10                                                           RDDIR
  230 KCDEL = 0                                                          RDDIR
      DO 235 I=1,8                                                       RDDIR
      IF ((JC(I).LT.1R0).OR.(JC(I).GT.1R9)) GO TO 520                    RDDIR
  235 KCDEL = 10*KCDEL + (JC(I)-1R0)                                     RDDIR
      I = KCDEL                                                          RDDIR
```

```
240 LS = SEARCH(I,NDEL,LDEL)                                              RDDIR
    IF (LS) GO TO 1000                                                    RDDIR
    LINCNT = LINCNT + 2                                                   RDDIR
    IF (LINCNT.GT.55) CALL PAGER(56)                                      RDDIR
    WRITE(IOUT,2)NUM,ID,IC                                                RDDIR
    WRITE(IOUT,6)                                                         RDDIR
  6 FORMAT(17X,8(1H↑),* - THIS KC HAS ALREADY BEEN DELETED*)             RDDIR
    GO TO 10                                                              RDDIR
300 DECODE(10,9,IC)JC                                                     RDDIR
    I = 1                                                                 RDDIR
    IF (KUTOFF.EQ.0) GO TO 310                                            RDDIR
    LINCNT = LINCNT + 2                                                   RDDIR
    IF (LINCNT.GT.55) CALL PAGER(56)                                      RDDIR
    WRITE(IOUT,2)NUM,ID,IC                                                RDDIR
    WRITE(IOUT,8)                                                         RDDIR
  8 FORMAT(7X,6(1H↑),* - A CUTOFF HAS ALREADY BEEN READ, THIS CARD WIL   RDDIR
   1L BE IGNORED*)                                                        RDDIR
    GO TO 10                                                              RDDIR
310 J = JC(I) - 1R0                                                       RDDIR
    IF (J) 320,350,330                                                    RDDIR
320 LINCNT = LINCNT + 2                                                   RDDIR
    IF (LINCNT.GT.55) CALL PAGER(56)                                      RDDIR
    WRITE(IOUT,2)NUM,ID,IC                                                RDDIR
    WRITE(IOUT,7)(IBL,N=1,I),IAR,(IBL,M=I,8)                              RDDIR
  7 FORMAT(16X,9R1,* - DIGIT MUST APPEAR HERE*)                           RDDIR
    GO TO 10                                                              RDDIR
330 IF(JC(I).EQ.1R ) GO TO 1000                                           RDDIR
    IF (J-9) 350,350,320                                                  RDDIR
350 KUTOFF = 10*KUTOFF + J                                                RDDIR
    I = I+1                                                               RDDIR
    GO TO 310                                                             RDDIR
400 LNORM = .TRUE.                                                        RDDIR
    GO TO 1000                                                            RDDIR
```

```
  500 LPRINT = .FALSE.                                                    RDDIR
 1000 LINCNT = LINCNT + 1                                                 RDDIR
      IF (LINCNT.GT.55) CALL PAGER(56)                                    RDDIR
      WRITE(IOUT,2)NUM,IO,IC                                              RDDIR
    2 FORMAT(1X,I3,3X,8A10)                                               RDDIR
      GO TO 10                                                            RDDIR
      END                                                                 RDDIR


      OVERLAY(EDIT,2,0)                                                   RDTASK
      PROGRAM RDTASK                                                      RDTASK
                                                                          EDCOM
      COMMON/TABLES/IBUT,ITTAB(700),KC(700),KFREW(700),KPT(700),IW(700),  ENLRG
     1 NXT,MTAB(30000),NTASK,MAXTAB,MAXKC                                 ENLRG
                                                                          EDCOM
      COMMON/CODES/IT00(3,2),ITON(2,2),IT01(16,2),IT02(8,2)               EDCOM
                                                                          EDCOM
      COMMON/IO/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)            ENLRG
                                                                          EDCOM
      COMMON/SKILLS/KEYWD(16),KEYALB(16)                                  EDCOM
                                                                          EDCOM
      COMMON/DELIST/NDEL,LDEL(700),KUTOFF                                 ENLRG
                                                                          EDCOM
      COMMON/LOG/IT1,LNORM,LPRINT                                         EDCOM
                                                                          EDCOM
      COMMON/TCARD/ITDCN,ITIN,NC                                          EDCOM
                                                                          EDCOM
                                                                          EDCOM
      LOGICAL IT1,LNORM,LPRINT                                            EDCOM
      LOGICAL SEARCH,LS,NEWTSK                                            EDCOM
                                                                          EDCOM
                                                                          EDCOM
```

```
      DIMENSION KCREAD(8)                                                    EDCOM
      DIMENSION KCBUFF(700),ISKV(700)                                        ENLRG
      DIMENSION LIST(3),LISTR(2)                                             EDCOM
      DIMENSION PRLIST(700)                                                  ENLRG
      DIMENSION IFMAT(3)                                                     EDCOM
                                                                             EDCOM
      EQUIVALENCE (LIST(1),ITDCN),(LISTPR(1),PRLIST(1))                      EDCOM
                                                                             EDCOM
                                                                             RDTASK
                                                                             RDTASK
      CALL INPAGE(NUM)                                                       RDTASK
      NUM = 0                                                                RDTASK
      NTASK = 0                                                              RDTASK
                                                                             RDTASK
   10 READ(INPUT,1)ICARD                                                     RDTASK
    1 FORMAT(80R1)                                                           RDTASK
      IF (EOF(INPUT)) 20,30                                                  RDTASK
   15 WRITE(IOUT,6)                                                          RDTASK
    6 FORMAT(20(2H *),* ERROR - PREMATURE END OF FILE ON INPUT DATA*)        RDTASK
   20 RETURN                                                                 RDTASK
   30 NUM = NUM + 1                                                          RDTASK
      IF (.NOT.LPRINT) GO TO 35                                              RDTASK
      LINCNT = LINCNT + 2                                                    RDTASK
      IF (LINCNT.GT.55) CALL PAGER(56)                                       RDTASK
      WRITE(IOUT,2)                                                          RDTASK
    2 FORMAT(1H )                                                            RDTASK
      WRITE(IOUT,3)NUM,ICARD                                                 RDTASK
    3 FORMAT(5X,I4,5X,80R1)                                                  RDTASK
   35 CALL NUMBER(IT00,LIST,3)                                               RDTASK
      NEWTSK = SEARCH(ITIN,NTASK,ITTAB)                                      RDTASK
      IF (NEWTSK) WRITE(ITAPE,7)ITIN,(ICARD(I),I=31,80)                      RDTASK
    7 FORMAT(I5,50R1)                                                        RDTASK
      IF (.NOT.LPRINT) GO TO 40                                              RDTASK
      IF (ITDCN.NE.0) CALL PERROR(1)                                         RDTASK
```

```
      IF (NC.LE.0) CALL PERROR(3)
      IF (NC.LE.0) NC = 100
      IF (NEWTSK) GO TO 40
      LINCNT = LINCNT + 1
      IF (LINCNT.GT.55) CALL PAGER(56)
      WRITE(IOUT,9)
    9 FORMAT(10(2H *),* THIS TASK HAS ALREADY APPEARED IN INPUT - THIS O
     1CCURENCE WILL BE IGNORED*)
   40 READ(INPUT,1)ICARD
      IF (EOF(INPUT)) 15,50
   50 NUM = NUM + 1
      IF (.NOT.LPRINT) GO TO 80
      LINCNT = LINCNT + 1
      IF (LINCNT.GT.55) CALL PAGER(56)
      WRITE(IOUT,3)NUM,ICARD
      CALL NUMBER(ITON,LISTR,2)
   60 IF (LISTR(1).NE.(LIST(1)+1)) CALL PERROR(1)
      IF (LISTR(2).NE.ITIN) CALL PERROR(2)
      IF (LISTR(1).NE.0) GO TO 80
      LINCNT = LINCNT + 1
      IF (LINCNT.GT.55) CALL PAGER(56)
      WRITE(IOUT,8)
    8 FORMAT(* ABOVE CARD APPEARS TO BE START OF A NEW TASK - IT WILL BE
     1 REPRINTED BELOW AS IF IT WERE *)
      NUM = NUM - 1
      GO TO 30
   80 IND = 16
      CALL NUMBER(ITOI,ISKV,16)
      DO 82 I=1,16
      IF (MOD(ISKV(I),5).EQ.0) GO TO 82
      LINCNT = LINCNT + 1
      IF (LINCNT.GT.55) CALL PAGER(56)
      WRITE(IOUT,11)I
```

```
 11 FORMAT(* ERROR - - SKILL VALUE NO.*,I2,* DOES NOT END WITH A 0 OR  RDTASK
   1A 5*)                                                              RDTASK
 82 CONTINUE                                                           RDTASK
    IF (NEWTSK) CALL PUTLIST(IND,KEYAL8,ITIN,ISKV)                     RDTASK
    IF(NC.EQ.1) GO TO 10                                               RDTASK
    IND = 0                                                            RDTASK
    DO 200 I = 2,NC                                                    RDTASK
    READ(INPUT,1)ICARD                                                 RDTASK
    IF (EOF(INPUT)) 15,85                                              RDTASK
 85 NUM = NUM + 1                                                      RDTASK
    IF (.NOT.LPRINT) GO TO 100                                         RDTASK
    LINCNT = LINCNT + 1                                                RDTASK
    IF (LINCNT.GT.55) CALL PAGER(56)                                   RDTASK
    WRITE(IOUT,3)NUM,ICARD                                             RDTASK
    CALL NUMBER(ITON,LISTR,2)                                          RDTASK
    IF (LISTR(1).NE.I) CALL PERROR(1)                                  RDTASK
    IF (LISTR(2).NE.ITIN) CALL PERROR(2)                               RDTASK
    IF (LISTR(1).NE.0) GO TO 100                                       RDTASK
    LINCNT = LINCNT + 1                                                RDTASK
    IF (LINCNT.GT.55) CALL PAGER(56)                                   RDTASK
    WRITE(IOUT,8)                                                      RDTASK
    NUM = NUM - 1                                                      RDTASK
    GO TO 30                                                           RDTASK
100 CALL NUMBER(ITO2,KCREAD,8)                                         RDTASK
    DO 110 J=1,7,2                                                     RDTASK
    IF (KCREAD(J).EQ.0) GO TO 110                                      RDTASK
    IF (MOD(KCREAD(J+1),5).EQ.0) GO TO 102                             RDTASK
    LINCNT = LINCNT + 1                                                RDTASK
    IF (LINCNT.GT.55) CALL PAGER(56)                                   RDTASK
    K = (J+1)/2                                                        RDTASK
    WRITE(IOUT,11)K                                                    RDTASK
```

```
102 CONTINUE                                                          RDTASK
    IF (KCREAD(J+1).NE.0) GO TO 105                                   RDTASK
    IF (.NOT.LPRINT) GO TO 110                                        RDTASK
    LINCNT = LINCNT + 1                                               RDTASK
    IF (LINCNT.GT.55) CALL PAGER(56)                                  RDTASK
    WRITE(IOUT,5)KCREAD(J)                                            RDTASK
  5 FORMAT(* KNOWLEDGE CATEGORY *,I5,* HAS SKILL VALUE ZERO - THIS OCC RDTASK
   1URENCE WILL BE IGNORED*)                                          RDTASK
    GO TO 110                                                         RDTASK
105 CONTINUE                                                          RDTASK
    LS = SEARCH(KCREAD(J),IND,KCBUFF)                                 RDTASK
    IF (LS) ISKV(IND) = KCREAD(J+1 )                                  RDTASK
    IF (LS) GO TO 110                                                 RDTASK
    IF (.NOT.LPRINT) GO TO 110                                        RDTASK
    LINCNT = LINCNT + 1                                               RDTASK
    IF (LINCNT.GT.55) CALL PAGER(56)                                  RDTASK
    WRITE(IOUT,4)KCREAD(J)                                            RDTASK
  4 FORMAT(* KNOWLEDGE CATEGORY *,I8,* HAS ALREADY APPEARED IN THIS TA RDTASK
   1SK - THIS OCCURRENCE WILL BE IGNORED*)                            RDTASK
110 CONTINUE                                                          RDTASK
200 CONTINUE                                                          RDTASK
    IF (NEWTSK) CALL PUTLIST(IND,KCBUFF,ITIN,ISKV)                    RDTASK
    GO TO 10                                                          RDTASK
    END                                                               RDTASK

    SUBROUTINE NUMBER(ICOL,LIST,NLIST)                                NUMBER
    COMMON/IO/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)          ENLKG
    DIMENSION ICOL(NLIST,2),LIST(NLIST)                               NUMBER
    DATA IBL/10H          /,IAS/10H**********/,ID/10H----------/      NUMBER
    DO 100 I=1,NLIST                                                  NUMBER
    J = ICOL(I,1)                                                     NUMBER
    K = ICOL(I,2)                                                     NUMBER
```

```
      LIST(I) = 0                                                       NUMBER
      IF (J.EQ.2) GO TO 5                                               NUMBER
      L = J - 1                                                         NUMBER
      IF (ICARD(L).NE.1R ) GO TO 30                                     NUMBER
    5 L = K + 1                                                         NUMBER
      IF (ICARD(L).NE.1R ) GO TO 40                                     NUMBER
      DO 10 L=J,K                                                       NUMBER
      IN = ICARD(L)-1R0                                                 NUMBER
      IF (ICARD(L).EQ.1R ) IN = 0                                       NUMBER
      IF((IN.LT.0).OR.(IN.GT.9)) GO TO 20                               NUMBER
   10 LIST(I) = LIST(I)*10 + IN                                         NUMBER
      GO TO 100                                                         NUMBER
   20 LIST(I) = 0                                                       NUMBER
      L = L - 1                                                         NUMBER
      WRITE(IOUT,1)(IBL,M=1,L),IAR,(ID,N=L,79)                          NUMBER
    1 FORMAT(14X,81R1,* DIGIT MUST APPEAR HERE*)                        NUMBER
      LINCNT = LINCNT + 1                                               NUMBER
      IF (LINCNT.GT.55) CALL PAGER(56)                                  NUMBER
      GO TO 100                                                         NUMBER
   30 LIST(I) = 0                                                       NUMBER
      L = L - 1                                                         NUMBER
      WRITE(IOUT,2)(IBL,M=1,L),IAR,(ID,N=L,79)                          NUMBER
    2 FORMAT(14X,81R1,* BLANK MUST APPEAR HERE*)                        NUMBER
      LINCNT = LINCNT + 1                                               NUMBER
      IF (LINCNT.GT.55) CALL PAGER(56)                                  NUMBER
      GO TO 5                                                           NUMBER
   40 L = L - 1                                                         NUMBER
      WRITE(IOUT,2)(IBL,M=1,L),IAR,(ID,N=L,79)                          NUMBER
      LINCNT = LINCNT + 1                                               NUMBER
      IF (LINCNT.GT.55) CALL PAGER(56)                                  NUMBER
  100 CONTINUE                                                          NUMBER
      RETURN                                                            NUMBER
      END                                                               NUMBER
```

```
      SUBROUTINE PERROR(I)

      COMMON/TABLES/IBLT,ITTAB(700),KC(700),KFREQ(700),KPT(700),IW(700),
     1 NKT,MTAB(36000),NTASK,MAXTAB,MAXKC

      COMMON/COLS/ITOO(3,2),ITON(2,2),ITO1(16,2),ITO2(8,2)

      COMMON/IO/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)

      COMMON/SKILLS/KEYWD(16),KEYALB(16)

      COMMON/DELIST/NDEL,LDEL(700),KLTOFF

      COMMON/LOG/TT1,LNORM,LPRINT

      COMMON/TCARD/ITOCN,ITIN,NC


      LOGICAL TT1,LNORM,LPRINT
      LOGICAL SEARCH,LS,NEWTSK


      DIMENSION KCREAD(8)
      DIMENSION KCBUFF(700),ISAV(700)
      DIMENSION LIST(3),LISTR(2)
      DIMENSION PRLIST(700)
      DIMENSION IFMAT(3)

      EQUIVALENCE (LIST(1),ITOCN),(LISTPR(1),PRLIST(1))


      DATA IBL/1H /,IAR/1H*/,ID/1H-/,IENTRY/0/
```

```
      LINCNT = LINCNT + 2                                                 PERROR
      IF (LINCNT.GT.55) CALL PAGER(56)                                    PERROR
      GO TO (10,30,20)I                                                   PERROR
   30 WRITE(IOUT,1)                                                       PERROR
    1 FORMAT(18X,6H++++++,72(1H-),* WARNING*/                             PERROR
     1  65X,*THIS TASK NUMBER DIFFERS FROM ENTRY ON SUMMARY CARD*)        PERROR
      RETURN                                                              PERROR
   10 WRITE(IOUT,2)                                                       PERROR
    2 FORMAT(15X,2H++,80(1H-),* APPARENT CARD SEQUENCE ERROR*)            PERROR
      RETURN                                                              PERROR
   20 WRITE(IOUT,4)                                                       PERROR
    4 FORMAT(42X,2H++,* - - - ERROR - - - SUMMARY CARD SAYS NO DATA CARD PERROR
     1S FOLLOW - */,20X,* ALL SUBSEQUENT CARDS WILL BE TREATED AS DATA F PERROR
     2OR THIS TASK UNTIL A (T00) CARD IS FOUND*)                          PERROR
      RETURN                                                              PERROR
      END                                                                 PERROR
                                                                          PERROR

      SUBROUTINE PUTLIST(N,LIST,ITIN,ISKV)                                PUTL
      COMMON/TABLES/IBOT,ITTAB(700),KC(700),KFREQ(700),KPT(700),IW(700),  ENLRG
     1NEXT,MTAB(36000),NTASK,MAXTAB,MAXKC                                 ENLRG
      DIMENSION LIST(1),ISKV(1)                                           PUTL
      DATA MASK20/3777777B/,MASK40/17777777777777B/                       ENLRG
      IF (N.GT.MAXKC) CALL ABORT(20HISKV ARRAY OVERFLOW      )            ENLRG
      DO 100 I=1,N                                                        PUTL
      IF (ISKV(I).EQ.0) GO TO 100                                         PUTL
      IF (IBOT.EQ.0) GO TO 20                                             PUTL
      DO 10 J=1,IBOT                                                      PUTL
      IF (KC(J).EQ.LIST(I)) GO TO 50                                      PUTL
   10 CONTINUE                                                            PUTL
   20 IBOT = IBOT + 1                                                     PUTL
                                                                          PUTL
```

```
      IF (IBUT.GT.MAXKC)  CALL ABORT(20HKC ARRAY  OVERFLOW      )        ENLRG
      KC(IBUT) = LIST(I)                                                   PUTL
      J = IBUT                                                             PUTL
      ITOP = NEXT                                                          PUTL
      GO TO 60                                                             PUTL
   50 K = KPT(J)                                                           PUTL
      ITOP = AND(MASK20,SHIFT(MTAB(K),20))                                 PUTL
      MTAB(K) = OR(SHIFT(NEXT,40),AND(MTAB(K),MASK40))                     ENLRG
   60 KFREQ(J) = KFREQ(J) + 1                                              ENLRG
      KPT(J) = NEXT                                                        PUTL
      MTAB(NEXT) = OR(SHIFT(ITOP,40),SHIFT(ISKV(I),20),ITIN)               PUTL
      NEXT = NEXT + 1                                                      ENLRG
      IF (NEXT.GT.MAXTAB) CALL ABORT(20HTABLE ARRAY OVERFLOW    )          PUTL
  100 CONTINUE                                                             ENLRG
      RETURN                                                               PUTL
      END                                                                  PUTL
                                                                           PUTL

      OVERLAY(EDIT,3,0)
      PROGRAM SORT                                                         SORT
                                                                           SORT
                                                                           EDCOM
      COMMON/TABLES/IBUT,ITTAB(700),KC(700),KFREQ(700),KPT(700),IW(700),   ENLRG
     1 NXT,MTAB(36000),NTASK,MAXTAB,MAXKC                                  ENLRG
                                                                           EDCOM
      COMMON/COLS/IT00(3,2),IT0N(2,2),IT01(16,2),IT02(3,2)                 EDCOM
                                                                           EDCOM
      COMMON/IO/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)             ENLRG
                                                                           EDCOM
      COMMON/SKILLS/KEYWD(16),KEYALB(16)                                   EDCOM
                                                                           EDCOM
      COMMON/DELIST/NDEL,LDEL(700),KUTOFF                                  ENLRG
                                                                           EDCOM
      COMMON/LOG/ITI,LNORM,LPRINT                                          EDCOM
                                                                           EDCOM
      COMMON/ICARD/ITCON,ITIN,NC                                           EDCOM
```

```
                                                                        EDCOM
      LOGICAL TT1,LNORM,LPRINT                                          EDCOM
      LOGICAL SEARCH,LS,NEWTSK                                          EDCOM
                                                                        EDCOM
                                                                        EDCOM
      DIMENSION KCREAD(8)                                               EDCOM
      DIMENSION KCBUFF(700),ISKV(700)                                   EDCOM
      DIMENSION LIST(3),LISTR(2)                                        ENLRG
      DIMENSION PRLIST(700)                                             EDCOM
      DIMENSION IFMAT(3)                                                ENLRG
                                                                        EDCOM
                                                                        EDCOM
      EQUIVALENCE (LIST(1),ITDCN),(LISTPR(1),PRLIST(1))                 EDCOM
                                                                        EDCOM
      DATA MASK20/3777777B/                                             ENLRG
      CALL SHLSRT(KFREQ,IBOT,IW)                                        SORT
      DO 10 I=1,IBOT                                                    SORT
      J = IW(I)                                                         SORT
      IF (KFREQ(J).GT.KUTOFF) GO TO 20                                  SORT
   10 LS = SEARCH(KC(J),NDEL,LDEL)                                      SORT
   20 IF (NDEL.GT.1) CALL SHLSRT(LDEL,NDEL)                             SORT
      CALL INPAGE(IBOT)                                                 SORT
      DO 100 I=1,IBOT                                                   SORT
      CALL PKC(I)                                                       SORT
  100 CONTINUE                                                          SORT
      CALL INPAGE(IBOT)                                                 SORT
      DO 200 I=1,IBOT                                                   SORT
      K = (IBOT+1) - I                                                  SORT
      J = IW(K)                                                         SORT
      CALL PKC(J)                                                       SORT
  200 CONTINUE                                                          SORT
      CALL SHLSRT(KC,IBOT,IW)                                           SORT
      CALL INPAGE(IBOT)                                                 SORT
```

```
      DO 300 I=1,IBOT                                                    SORT
      J = IW(I)                                                          SORT
      CALL PKC(J)                                                        SORT
  300 CONTINUE                                                           SORT
      CALL INPAGE(IBOT)                                                  SORT
      ITOP = 1                                                           SORT
      IF (NDEL.EQ.0) GO TO 430                                           SORT
      DO 400 I=1,IBOT                                                    SORT
      J = IW(I)                                                          SORT
      KTEMP = KC(J)                                                      SORT
      DO 350 K=ITOP,NDEL                                                 SORT
      IF (KTEMP.NE.LDEL(K)) GO TO 350                                    SORT
      CALL PKC(J)                                                        SORT
      IW(I) = 0                                                          SORT
      LDEL(K) = 0                                                        SORT
      ITOP = K + 1                                                       SORT
      GO TO 400                                                          SORT
  350 CONTINUE                                                           SORT
  400 CONTINUE                                                           SORT
      IFLAG = 0                                                          SORT
      DO 420 I=1,NDEL                                                    SORT
      IF (LDEL(I).EQ.0) GO TO 420                                        SORT
      IF (IFLAG.NE.0) GO TO 410                                          SORT
      WRITE(IOUT,1)                                                      SORT
    1 FORMAT(1H0,* ERROR - - THE FOLLOWING KCS WERE TO BE DELETED, BUT W SORT
     1ERE NOT FOUND IN THE INPUT DATA*)                                  SORT
      IFLAG = 1                                                          SORT
  410 WRITE(IOUT,2)LDEL(I)                                               SORT
    2 FORMAT(20X,I8)                                                     SORT
  420 CONTINUE                                                           SORT
  430 CALL INPAGE(IBOT)                                                  SORT
      DO 500 I=1,IBOT                                                    SORT
      IF(IW(I).EQ.0) GO TO 500                                           SORT
```

```
      J = IW(I)                                                              SORT
      IND = IND + 1                                                          SORT
      IW(IND) = IW(I)                                                        SORT
      IF (MOD(IND,56).EQ.0) CALL PAGER(57)                                   SORT
      IF(KC(J).LE.16) GO TO 450                                              SORT
      WRITE(IOUT,3)IND,KC(J),KFREQ(J)                                        SORT
    3 FORMAT(8X,I4,3X,I8,2X,I4)                                              SORT
      GO TO 500                                                              SORT
  450 K = KC(J)                                                              SORT
      WRITE(IOUT,4)IND,KEYWD(K),KFREQ(J)                                     SORT
    4 FORMAT(8X,I4,3X,R8,2X.I4)                                              SORT
  500 CONTINUE                                                               SORT
      CALL INPAGE(IBOT)                                                      SORT
      DO 600 I=1,NTASK                                                       SORT
      READ(ITAPE,5)ITTAB(I),(LISTPR(J),J=1,5)                                SORT
    5 FORMAT(I5,5A10)                                                        SORT
    6 FORMAT(10X,I5,6X,I5,5X,5A10)                                           SORT
      IF (MOD(I,56).EQ.0) CALL PAGER(57)                                     SORT
  600 WRITE(IOUT,6)I,ITTAB(I),(LISTPR(J),J=1,5)                              SORT
      WRITE(IOUT,7)NTASK,IND                                                 SORT
    7 FORMAT(1H1,I10,* = NUMBER OF OBSERVATIONS*//1X,I10,* = NUMBER OF V     SORT
     1ARIBLES*//50X,*END OF JOB*)                                            SORT
                                                                             SORT
      REWIND ITAPE                                                           SORT
      IFMAT(1) = 10H(10X,16F4.                                               SORT
      IFMAT(2) = 10H1/(16F4.1)                                               SORT
      IFMAT(3) = 10H)                                                        SORT
      WRITE(ITAPE,8)IFMAT                                                    SORT
    8 FORMAT(3A10)                                                           SORT
      XNORM = 0.                                                             SORT
      LAST = NXT-1                                                           SORT
      IF (LNORM) XNORM = SQRT(0.5)                                           SORT
  650 DO 700 I=1,NTASK                                                       ENLRG
      DO 680 J=1,IND                                                         SORT
                                                                             SORT
```

```
      SV = XNORM                                                       SORT
      K = IW(J)                                                        SORT
      K = KPT(K)                                                       SORT
      N = SHIFT(MTAB(K),-40)                                           ENLRG
  660 K = SHIFT(MTAB(N),-40)                                           ENLRG
      SK = FLOAT(AND(MASK20,SHIFT(MTAB(N),-20))) * 0.1                 ENLRG
      IF (LNORM) SK = SQRT(SK + 0.5)                                   ENLRG
      ITN = AND(MTAB(N),MASK20)                                        ENLRG
      IF (ITN.EQ.ITTAB(I)) GO TO 670                                   ENLRG
      IF (K.LE.N) GO TO 680                                            SORT
      N = K                                                            SORT
      GO TO 660                                                        SORT
  670 SV = SK                                                          ENLRG
  680 PRLIST(J) = SV                                                   SORT
    9 FORMAT(3X,I7,16F4.1/(16F4.1))                                    SORT
  700 WRITE(ITAPE,9)ITTAB(I),(PRLIST(J),J=1,IND)                       SORT
      RETURN                                                           SORT
      END                                                              SORT

      SUBROUTINE PKC(I)                                                PKC
                                                                       PKC
      COMMON/SKIP/LINSKP                                               PKC
                                                                       EDCOM
      COMMON/TABLES/IBLT,ITTAB(700),KC(700),KFREQ(700),KPT(700),IW(700),  ENLRG
     1 NXT,MTAB(36000),NTASK,MAXTAB,MAXKC                              ENLRG
                                                                       EDCOM
      COMMON/COLS/ITOC(3,2),ITON(2,2),IT01(16,2),IT02(8,2)             EDCOM
                                                                       EDCOM
      COMMON/IO/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)         ENLRG
                                                                       EDCOM
      COMMON/SKILLS/KEYWD(16),KEYALB(16)                               EDCOM
```

```
                                                                        EDCOM
      COMMON/DELIST/NDEL,LDEL(700),KUTOFF                               ENLRG
                                                                        EDCOM
      COMMON/LOG/TT1,LNORM,LPRINT                                       EDCOM
                                                                        EDCOM
      COMMON/TCARD/ITDCN,ITIN,JC                                        EDCOM
                                                                        EDCOM
                                                                        EDCOM
      LOGICAL TT1,LNORM,LPRINT                                          EDCOM
      LOGICAL SEARCH,LS,NEWTSK                                          EDCOM
                                                                        EDCOM
                                                                        EDCOM
      DIMENSION KCREAD(8)                                               EDCOM
      DIMENSION KCBUFF(700),ISKV(700)                                   ENLRG
      DIMENSION LIST(3),LISTR(2)                                        EDCOM
      DIMENSION PRLIST(700)                                             ENLRG
      DIMENSION IFMAT(3)                                                EDCOM
                                                                        EDCOM
      EQUIVALENCE (LIST(1),ITDCN),(LISTPR(1),PRLIST(1))                 EDCOM
                                                                        EDCOM
                                                                        PKC
      ILF = 1H                                                          PKC
      IF (MOD(LINSKP,20).EQ.0) ILF = 1H+                                PKC
      WRITE(IOUT,3)ILF                                                  PKC
    3 FORMAT(A1)                                                        PKC
                                                                        PKC
      J = LINK(KPT(I))                                                  PKC
      K = ((J-1)/20)+2                                                  PKC
      LINCNT = LINCNT + K                                               PKC
      IF (LINCNT.GT.55) CALL PAGER(6)                                   PKC
      IF (LINCNT.EQ.2) LINCNT = K                                       PKC
      LINSKP = J                                                        PKC
      IASTER = 1H                                                       PKC
```

```
      IF (NDEL.EQ.0) GO TO 30                                         PKC
      DO 10 L = 1,NDEL                                                PKC
      IF (LDEL(L).EQ.KC(I)) IASTER = 1H*                              PKC
   10 CONTINUE                                                        PKC
   30 IF (KC(I).LE.16) GO TO 40                                       PKC
      WRITE(IOUT,1)I,IASTER,KC(I),KFREQ(I),(LISTPR(L),L=1,J)          PKC
    1 FORMAT(1X,I4,2X,A1,I8,2X,I4,10(2X,I4,1H/,F3.1)                  PKC
     1                      60(/22X,10(2X,I4,1H/,F3.1)))              PKC
      RETURN                                                          PKC
   40 K = KC(I)                                                       PKC
      WRITE(IOUT,2)I,IASTER,KEYWD(K),KFREQ(I),(LISTPR(L),L=1,J)       PKC
    2 FORMAT(1X,I4,2X,A1,A8,2X,I4,10(2X,I4,1H/,F3.1)                  PKC
     1                      60(/22X,10(2X,I4,1H/,F3.1)))              PKC
      RETURN                                                          PKC
      END                                                             PKC

      FUNCTION LINK(IEND)                                             LINK
      COMMON/TABLES/IBOT,ITTAB(700),KC(700),KFREQ(700),KPT(700),IW(700),EDCOM
     1 NXT,MTAB(36000),NTASK,MAXTAB,MAXKC                             ENLRG
                                                                      ENLRG
      COMMON/COLS/IT00(3,2),ITON(2,2),IT01(16,2),IT02(8,2)            EDCOM
                                                                      EDCOM
      COMMON/IT/INPUT,IOUT,ITAPE,ICARD(80),LINCNT,LISTPR(1400)        EDCOM
                                                                      ENLRG
      COMMON/SKILLS/KEYWD(16),KEYALB(16)                              EDCOM
                                                                      EDCOM
      COMMON/DFLIST/NDEL,LDEL(700),KUTOFF                             EDCOM
                                                                      ENLRG
      COMMON/LOG/TT1,LNORM,LPRINT                                     EDCOM
                                                                      EDCOM
      COMMON/TCARD/ITOCN,ITIN,NC                                      EDCOM
                                                                      EDCOM
```

```
                                                                        EDCOM
      LOGICAL TT1,LNORM,LPRINT                                          EDCOM
      LOGICAL S ARCH,LS,NEWTSK                                          EDCOM
                                                                        EDCOM
                                                                        EDCOM
      DIMENSION KCREAD(8)                                               EDCOM
      DIMENSION KCBUFF(700),ISKV(700)                                   EDCOM
      DIMENSION LIST(3),LISTR(2)                                        ENLRG
      DIMENSION PRLIST(700)                                             EDCOM
      DIMENSION IFMAT(3)                                                ENLRG
                                                                        EDCOM
      EQUIVALENCE (LIST(1),ITDCN),(LISTPR(1),PRLIST(1))                 EDCOM
                                                                        EDCOM
      DATA MASK20/3777777B/                                             EDCOM
                                                                        ENLRG
      I = 0                                                             LINK
      N = SHIFT(MTAB(IEND),-40)                                         LINK
   10 NEXT = SHIFT(MTAB(N),-40)                                         ENLRG
      I = I + 1                                                         ENLRG
      LISTPR(I) = AND(MASK20,MTAB(N))                                   LINK
      I = I + 1                                                         ENLRG
      PRLIST(I) = FLOAT(AND(SHIFT(MTAB(N),-20),MASK20)) / 10.0          LINK
      LINK = I                                                          ENLRG
      IF (NEXT.LE.N) RETURN                                             LINK
      N = NEXT                                                          LINK
      GO TO 10                                                          LINK
      END                                                               LINK
                                                                        LINK
```

```
         IDENT  SHLSRT$
                                                                          SHLSRT
***      SHLSRT - 1 WORD/ENTRY TABLE SORT
*                                                                         SHLSRT
*        CALL SHLSRT( TABLE , N )          - DIRECT ( IN PLACE ) SORT     SHLSRT
*        CALL SHLSRT( TABLE , N , PTR )    - INDIRECT SORT                SHLSRT
*                                                                         SHLSRT
*        ENTRY:                                                           SHLSRT
*              TABLE = ADDRESS OF TABLE ( ARRAY ) TO BE SORTED            SHLSRT
*              N = NUMBER OF WORDS IN TABLE                               SHLSRT
*              PTR = ADDRESS OF AN ARRAY TO BE USED AS A POINTER          SHLSRT
*              TABLE ( DIMENSIONED PTR(N) )                               SHLSRT
*                                                                         SHLSRT
*         IF THE THIRD ARGUMENT IS ABSENT THEN A DIRECT ( IN PLACE )      SHLSRT
*         SORT WILL BE PERFORMED, ELSE THE THIRD ARGUMENT WILL            SHLSRT
*         BE USED AS A POINTER TABLE AND AN INDIRECT ( POINTER )          SHLSRT
*         SORT WILL BE PERFORMED. ON EXIT THE POINTER TABLE WILL          SHLSRT
*         POINT THE THE ELEMENTS OF *TABLE* IN ASCENDING ORDER.           SHLSRT
                                                                          SHLSRT

         ENTRY  SHLSRT
STBA     BSS    2                                                         SHLSRT
STBD     SA5    STBA              RESTORE A0                              SHLSRT
         SA0    X5                                                        SHLSRT
SHLSRT   PS                                                               SHLSRT
         SB1    1                                                         SHLSRT
         SA2    A1+B1                                                     SHLSRT
         SX6    A0                                                        SHLSRT
         SA3    A2+B1                                                     SHLSRT
                                                                          SHLSRT
```

```
         SA2     X2
         SA0     X1-1       A0 = FWA-1                          SHLSRT
         SA6     STBA       SAVE A0                             SHLSRT
         SB7     X2                                             SHLSRT
         SB6     X2         N = N (LENGTH)                      SHLSRT
         NZ      X3,STBI    IF AN INDIRECT SORT REQUESTED       SHLSRT
                                                                SHLSRT
STB1     SX6     B5         M = M/2                             SHLSRT
         AX6     1                                              SHLSRT
         SB6     X6                                             SHLSRT
         SB3     B1         J = 1                               SHLSRT
         ZR      B6,STB0    RETURN IF M = 0                     SHLSRT
         SB4     B7-B6      K = N-M                             SHLSRT
         SB2     B3         I = J                               SHLSRT
STB2     SB5     B2+B6      L = I+M                             SHLSRT
         SA1     A0+B2      A(I)                                SHLSRT
         SA2     A0+B5      A(L)                                SHLSRT
         IX4     X2-X1                                          SHLSRT
         PL      X4,STB3    IF A(L) .G . A(I)                   SHLSRT
         BX6     X1                                             SHLSRT
         LX7     X2         INTERCHANGE A(L) AND A(I)           SHLSRT
         SA6     A2                                             SHLSRT
         SA7     A1                                             SHLSRT
         SB2     B2-B6      I = I-M                             SHLSRT
         GT      B2,STB2    IF I .GT. 0                         SHLSRT
STB3     SB3     B3+B1      J = J+1                             SHLSRT
         SB2     B3         I = J                               SHLSRT
         LE      B3,B4,STB2      IF J .LE. K                    SHLSRT
         EQ      STB1                                           SHLSRT
STBI     BX6     X1         SET UP POINTER TABLE                SHLSRT
         SB2     B6-B1                                          SHLSRT
         SA0     X3-1       A0 = FWA-1 OF POINTER TABLE         SHLSRT
                                                                SHLSRT
```

```
         SA6     A6+B1          SAVE ADDR OF A                    SHLSRT
         SA6     X3                                               SHLSRT
         SX6     X6+B1                                            SHLSRT
         SB2     B2-B1                                            SHLSRT
         SA6     A6+B1                                            SHLSRT
         NZ      B2,*-1                                           SHLSRT
                                                                  SHLSRT
STBI1    SX6     B6             M = M/2                           SHLSRT
         AX6     1                                                SHLSRT
         SB6     X6                                               SHLSRT
         SB3     B1             J = 1                             SHLSRT
         ZR      B6,STBI4       IF M = 0                          SHLSRT
         SB4     B7-B6          K = N-M                           SHLSRT
         SB2     B3             I = J                             SHLSRT
STBI2    SB5     B2+B6          L = I+M                           SHLSRT
         SA1     A0+B2          P(I)                              SHLSRT
         SA2     A0+B5          P(L)                              SHLSRT
         SA3     X1                                               SHLSRT
         SA4     X2                                               SHLSRT
         IX5     X4-X3                                            SHLSRT
         PL      X5,STBI3       IF  A(P(L)) .GT. A(P(I))          SHLSRT
         BX6     X1                                               SHLSRT
         LX7     X2             INTERCHANGE P(L) AND P(I)         SHLSRT
         SA6     A2                                               SHLSRT
         SA7     A1                                               SHLSRT
         SB2     B2-B6          I = I-M                           SHLSRT
         GT      B2,STBI2       IF I .LT. 0                       SHLSRT
STBI3    SB3     B3+B1          J = J+1                           SHLSRT
         SB2     B3             I = J                             SHLSRT
         LE      B3,B4,STBI2    IF J .LE. K                       SHLSRT
         EQ      STBI1                                            SHLSRT
                                                                  SHLSRT
```

B-38

```
STBI4    SA5    STBA                                                     SHLSRT
         SA1    A5+B1      ADDRESS OF A ( TABLE )                        SHLSRT
         SB2    A0+B1      P(1)                                          SHLSRT
         SB3    B2+B7      LWA+1 OF P                                    SHLSRT
         SA0    X5         RESTORE A0                                    SHLSRT
         SX1    X1-1                                                     SHLSRT
STBI5    SA2    B2         CHANGE POINTER ARRAY TO FORTRAN SUBSCRIPTS    SHLSRT
         SB2    B2+B1                                                    SHLSRT
         IX6    X2-X1                                                    SHLSRT
         SA6    A2                                                       SHLSRT
         LT     B2,B3,STBI5                                              SHLSRT
         EQ     SHLSRT                                                   SHLSRT
                                                                         SHLSRT
       END                                                               SHLSRT
```

```
      SUBROUTINE ABORT(MSG)                                            ABORT
      INTEGER MSG(1)                                                   ABORT
      CALL SYSTEM(52,12H USER ABORT.)                                  ABORT
      STOP                                                             ABORT
      ENTRY ABORTM                                                     ABORT
      CALL SYSTEM(52,MSG)                                              ABORT
      END                                                              ABORT
```

APPENDIX C

EDIT WITH THE PCVARIM PROGRAM

HSMS uses the PCVARIM program to select the "solution" (number of factors) which best groups the skill and knowledge variables. PCVARIM is an abbreviation for Principal Components Factor Analysis with Varimax Rotation.

Because the number, identity, and frequency of the skill and knowledge variables cannot be ascertained beforehand, because of the size limitations of the PCVARIM program, and because EDIT provides a logarithmic transformation of the data, PCVARIM is always used by HSMS interfaced with EDIT.[1]

DESCRIPTION

PCVARIM is written in FORTRAN IV and was used in the Control Data Corporation's (CDC) 6600 computer at the Courant Institute of Mathematical Sciences of New York University. The operating systems in use during HSMS analyses were KRONOS and NOS (Network Operating System). PCVARIM was stored on magnetic tape in compiled and loaded form (i.e, in binary object code in non-relocatable form) and in OLDPL form.

The HSMS task data are transferred to magnetic tape in the form of an OLDPL. A utility program, UPDATE, is used to transfer data to tape, correct data, or to generate temporary local files in the form of a compile file, which is part of the input file for EDIT. The

---

[1] The EDIT program appears in Appendix B.

input file for UPDATE is magnetic tape in OLDPL form, and the output file is C=DATA. The input file for EDIT is DATA, and the output file is TAPE9 (with optional printed output). The input file for PCVARIM is TAPE9, and the output file can be printed or punched.

The most attractive feature of PCVARIM is that all the factor solutions (i.e., the number of factors to be rotated) one wishes to examine can be requested with a single submission. As long as the number of principal axis factors to be extracted is as large as the largest number to be rotated, any and all smaller number solutions can be called with little additional cost. Therefore, the user has all the solutions to be examined available at one time

The tasks and the skill and knowledge variables enter PCVARIM in the order of the final internal numbering assigned by the EDIT program. Therefore, the EDIT "dictionaries" are used to interpret the PCVARIM output.

HSMS makes an inspection of the output as an error check as follows:

1. Check that means and standard deviations seem appropriate for the data.
2. Check that all rotations called for are present.
3. Check that no number in the correlation matrix exceeds 1.0, that no factor loading exceeds .999, and that matrix, correlation and factor loadings are not all zeroes.

Note: As of September, 1977, the comment cards included in the program had not been updated to include modifications of the PCVARIM Rotation Option Card. The user should follow the instructions for structuring the input file which follows, rather than the comment cards in the program file (which are not reproduced here).

STRUCTURING THE INPUT FILE: EDIT WITH PCVARIM

In the set-up presented below, the NOS system in use in 1977 at the Courant Institute's CDC 6600 is assumed. It is also assumed that the reader can refer to the UPDATE Reference Manual or a counterpart program. In the following instructions b denotes a blank.

INPUT FILE FOR EDIT WITH PCVARIM

Name	Keypunch Cards (one per box)	Instructions
Systems Cards	_ _ _ _ .EDIT WITH PCVARIM	Identification number and other information depending on system.
	USER(_ _ _ _ _ _ _ _)	USER and user code.
	CHARGE(_ _ _ _ _ _ _ _ _ _ _ _ _ _)	Charge card.
	HEADER.EDIT WITH PCVARIM	Optional to print out heading across a whole page.
	LABEL(OLDPL,VSN=T_ _ _)	Identifies OLDPL tape.
	UPDATE(Q,D,8,C=DATA) or UPDATE(F,D,8,C=DATA)	When selecting tasks using *COMPILE form of UPDATE. When using all data in OLDPL.
	UNLOAD(OLDPL)	
	LABEL(PROGRMS,VSN=T_ _ _)	Identifies program tape.
	COPYBF(PROGRMS,EDIT)	Copies program to local file.
	COPYBF(PROGRMS,PCVARIM)	Copies program to local file.
	UNLOAD(PROGRMS)	
	SETTL(1000)	Time limit.
	RFL(200000)	Defines field length.
	EDIT.	
	REWIND(OUTPUT)	Use only if no printed EDIT output is desired. Usually it is desirable to have EDIT output.
	RFL(105000)	
	PCVARIM.	
	End of Record Card	

INPUT FILE FOR EDIT WITH PCVARIM (continued)

Name	Keypunch Cards (one per box)	Instructions
Input Cards for UP DATE	*COMPILEb _ _ _ _, _ _ _ _, _ _ _ etc., or *COMPILEb _ _ _ _, _ _ _ _. _ _ _ (option)	Calls tasks from OLDPL. Each card starts: *COMPILE in Cols. 1-8; Column 9 blank, followed by Task Code Numbers separated by commas; last entry has no comma. Tasks can be in any order. Consecutive Code Numbers can be called as last entry on card after comma or as only entry or card by writing first Code Number, then a period, then last Code Number of the series. A new card must follow.
	End of Record Card	Needed even if no UPDATE cards.
Input Cards for EDIT	TITLEb b b b b _ _ _ _ _ _ _ _ etc.	Columns 1-5: punch TITLE; Columns 11-60: punch title of program, date, and any other special identification of this run.
	NOLIST (option)	If listing of data and some error checks are to be suppressed. Usually selected with PCVARIM.
	DELETEb b b b _ _ _ _ _ _ _ _ _ _ (option)	One card for each individual skill or knowledge category to be deleted from input data (if not to be deleted by CUTOFF). Columns 1-6: punch DELETE; Columns 11-18: punch 8-character skill code or 8-digit knowledge category code number.
	CUTOFFb b b b _ _ (option)	To eliminate skill or knowledge categories at or below a selected frequency from input data. Columns 1-6: punch CUTOFF; Columns 11-12: punch selected frequency (usually 4 or more) left justified in field.
	NORMALIZE (option)	Usually used when EDIT is used with PCVARIM.
	End of Record Card	

INPUT FILE FOR EDIT WITH PCVARIM (continued)

Name	Keypunch Cards (one per box)	Instructions
Input Cards For PCVARIM	_ etc.	Title Card. Cols. 1-40 read as one line; Cols 41-80 read as second line. Punch title of program, date, and any other special identification of this run.
Parameter Card	Cols. 4-6 _ _ _	Right justify in all fields unless otherwise indicated. Number of skill and knowledge categories to be treated as variables; up to 145. (HSMS uses up to 144.)
	Cols. 8-12 _ _ _ _ _	Number of tasks to be treated as observations; up to 99,999.
	Cols. 17-18 _ _	Number of principal axis factors to be extracted; up to 12. (Select the largest number to be rotated.)
	Cols. 23-24 _ _	Number of varimax factors to be rotated. (Select largest number desired, but no greater than Cols. 17-18.
	Col. 36 _	Punch Ø if no additional rotations. Punch 1 if additional rotations are wanted.
	Cols. 61-67 T A P E 9 _ _	Name of alternate input file when EDIT is used (left justified).
Rotation Option Card	Cols. 2-3 _ _ 5-6 _ _ 8-9 _ _ 11-12 _ _ 14-15 _ _ 17-18 _ _ 20-21 _ _ 23-24 _ _ 26-27 _ _ 29-30 _ _	Use if parameter card selected additional factor rotations (1 in Col. 36). Enter number of varimax factors to be rotated and listed; as many as are of interest, in descending order in the fields indicated (from 11 down to 2 is possible), so that each is less than the one before. Right justify.
	End of File Card	

PCVARIM

```
      OVERLAY(PCVARIM,0,0)                                                    KRONOS
      PROGRAM PCVARIM(TAPE9,INPUT,OUTPUT,PUNCH,TAPE5=INPUT,TAPE6=OUTPUT)      KRONOS
                                                                              PCVARIM
*         A - CORRELATION MATRIX                                              PCV-COM
*         S - VARIMAX ROTATED FACTORS                                         PCV-COM
          COMMON// A(144,144),S(144,12),H(145),V(145)                         PCV-COM
          COMMON // Q2(145),Q3(145),Q4(145)                                   PCVARIM
          COMMON // TIT(14),FMT(14)                                           PCVARIM
      INTEGER MNR(11)                                                         PCVARIM
      INTEGER IN,OUT                                                          KRONOS
      DATA MNR/11*0/                                                          KRONOS
C                                                                             KRONOS
      IN = 5                                                                  KRONOS
      OUT = 6                                                                 KRONOS
   10 READ(IN,500) TIT                                                        KRONOS
      IF (EOF(IN).NE.0) GO TO 350                                             KRONOS
      READ(IN,370)NVAR,NPEOP,NPAXFA,NROTFA,NVPCH,IMOR,CRIT,ILFN               KRONOS
      IF (IMOR.EQ.0) GO TO 15                                                 KRONOS
      READ(IN,16)(MNR(I),I=1,10)                                              KRONOS
 16     FORMAT(10I3)                                                          PCVARIM
          IMOR = 1                                                            PCVARIM
                                                                              PCVARIM
   15 IF (ILFN.NE.7L        ) IN = 9                                          KRONOS
      READ(IN,500) FMT                                                        KRONOS
      WRITE(OUT,390) TIT                                                      KRONOS
      WRITE(OUT,370)                                                          KRONOS
     1              NVAR,NPEOP,NPAXFA,NROTFA,NVPCH,IMOR,CRIT,ILFN             KRONOS
      WRITE(OUT,400) FMT                                                      KRONOS
      IF (NPEOP.LE.0) GO TO 80                                                PCVARIM
C                                                                             PCVARIM
```

```
C     CORRELATIONS                                                      PCVARIM
C     RAW DATA INPUT                                                    PCVARIM
C                                                                       PCVARIM
      DO 20 I = 1,NVAR                                                  PCVARIM
      S(I,1) = 0.0                                                      PCVARIM
      S(I,2) = 0.0                                                      PCVARIM
      DO 20 J = 1,NVAR                                                  PCVARIM
   20 A(I,J) = 0.0                                                      PCVARIM
      DO 40 K = 1,NPEOP                                                 PCVARIM
      READ(IN,FMT)(S(I,3),I=1,NVAR)                                     KRONOS
      DO 30 I = 1,NVAR                                                  PCVARIM
      S(I,1) = S(I,1)+S(I,3)                                            PCVARIM
      S(I,2) = S(I,2)+S(I,3)**2                                         PCVARIM
      DO 30 J = 1,I                                                     PCVARIM
   30 A(I,J) = A(I,J)+S(I,3)*S(J,3)                                     PCVARIM
   40 CONTINUE                                                          PCVARIM
      PEOPL = NPEOP                                                     PCVARIM
      DO 50 I = 1,NVAR                                                  PCVARIM
      S(I,4) = S(I,1)/PEOPL                                             PCVARIM
   50 S(I,5) = SQRT((S(I,2)-PEOPL*S(I,4)**2)/PEOPL)                     PCVARIM
      DO 60 I = 1,NVAR                                                  PCVARIM
      DO 60 J = 1,I                                                     PCVARIM
   60 A(I,J) = (A(I,J)-PEOPL*S(I,4)*S(J,4))/(PEOPL*S(I,5)*S(J,5))       PCVARIM
      DO 70 I = 1,NVAR                                                  PCVARIM
      DO 70 J = 1,I                                                     PCVARIM
   70 A(J,I) = A(I,J)                                                   PCVARIM
      WRITE(OUT,360) TIT                                                KRONOS
      WRITE(OUT,410)(I,S(I,4),S(I,5),I=1,NVAR)                          KRONOS
      GO TO 100                                                         PCVARIM
C                                                                       PCVARIM
 80   DO 85 I = 1,NVAR                                                  PCVARIM
   85 READ(IN,FMT) (A(I,J),J=1,I)                                       KRONOS
      DO 90 I = 1,NVAR                                                  PCVARIM
      DO 90 J = 1,I                                                     PCVARIM
   90 A(J,I) = A(I,J)                                                   PCVARIM
C                                                                       PCVARIM
```

```
C       CORRELATION OUTPUT                                               PCVARIM
C                                                                        PCVARIM
  100 ASSIGN 110 TO KKTV                                                 PCVARIM
      LL = 1                                                             PCVARIM
  110 IF ((NVAR-LL)-9) 120,120,130                                       PCVARIM
  120 ASSIGN 160 TO KKTV                                                 PCVARIM
      LS = NVAR                                                          PCVARIM
      GO TO 140                                                          PCVARIM
  130 LS = LL+9                                                          PCVARIM
  140 WRITE(OUT,430) TIT,(J,J=LL,LS)                                     KRONOS
      DO 150 I = 1,NVAR                                                  PCVARIM
  150 WRITE(OUT,420)(I,(A(I,J),J=LL,LS))                                 KRONOS
      LL = LL+10                                                         PCVARIM
      GO TO KKTV, (110,160)                                              PCVARIM
                                                                         PCVARIM
  160 SUM = 0.0                                                          PCVARIM
      DO 170 I = 1,NVAR                                                  PCVARIM
  170 SUM = SUM+A(I,I)                                                   PCVARIM
      WRITE(OUT,480) SUM                                                 KRONOS
C                                                                        PCVARIM
C       PRINCIPAL AXIS FACTORS                                           PCVARIM
C                                                                        PCVARIM
      IF (NPAXFA.EQ.0) GO TO 10                                          PCVARIM
      CALL HOW (NVAR,144,NPAXFA,A,V,S,H,W2,W3,W4)                        PCVARIM
      KP = NPAXFA                                                        PCVARIM
      DO 180 J = 1,KP                                                    PCVARIM
      R = SQRT(ABS(V(J)))                                                PCVARIM
      DO 180 I = 1,NVAR                                                  PCVARIM
  180 A(I,J) = R*S(I,J)                                                  PCVARIM
      WRITE(OUT,440) TIT                                                 KRONOS
```

```
      WRITE(OUT,450)(I,V(I),I=1,NVAR)                                  KRONOS
      SUM = 0.0                                                        PCVARIM
      DO 190 I = 1,KP                                                  PCVARIM
  190 SUM = SUM+V(I)                                                   PCVARIM
      WRITE(OUT,460) KP,SUM                                            KRONOS
      ASSIGN 200 TO KK                                                 PCVARIM
      LL = 1                                                           PCVARIM
  200 IF (KP-LL-9) 210,210,220                                         PCVARIM
  210 ASSIGN 250 TO KK                                                 PCVARIM
      LS = KP                                                          PCVARIM
      GO TO 230                                                        PCVARIM
  220 LS = LL+9                                                        PCVARIM
  230 WRITE(OUT,470)TIT,(J,J=LL,LS)                                    KRONOS
      DO 240 I=1,NVAR                                                  KRONOS
  240 WRITE(OUT,420)(I,(A(I,J),J=LL,LS))                               KRONOS
      LL = LL+10                                                       PCVARIM
      GO TO KK, (200,250)                                              PCVARIM
C                                                                      PCVARIM
C     VARIMAX FACTORS                                                  PCVARIM
C                                                                      PCVARIM
  250 IF (NROTFA) 260,10,290                                           PCVARIM
  260 IF (NPAXFA.LE.0) GO TO 10                                        PCVARIM
      DO 270 I = 1,NPAXFA                                              PCVARIM
      IF (V(I).LE.CRIT) GO TO 280                                      PCVARIM
  270 CONTINUE                                                         PCVARIM
  280 NR = I-1                                                         PCVARIM
      IF (NR.LE.0) GO TO 10                                            PCVARIM
  290 IF (NROTFA.GT.NPAXFA) GO TO 10                                   PCVARIM
      NR = NROTFA                                                      PCVARIM
      IF (NR-1) 10,300,310                                             PCVARIM
  300 WRITE(OUT,380)                                                   KRONOS
      GO TO 10                                                         PCVARIM
  310 DO 320 I = 1,NVAR                                                PCVARIM
      DO 320 J = 1,NR                                                  PCVARIM
  320 S(I,J) = A(I,J)                                                  PCVARIM
```

```
325  DO 330 I = 1,NVAR                                                   PCVARIM
     DO 330 J = 1,NR                                                     PCVARIM
 330 A(I,J) = S(I,J)                                                     PCVARIM
     DO 340 I = 1,NVAR                                                   PCVARIM
     H(I) = 0.0                                                          PCVARIM
     DO 340 J = 1,NR                                                     PCVARIM
 340 H(I) = H(I)+A(I,J)**2                                               PCVARIM
     CALL VARIM (NVAR,NR,NVPCH)                                          PCVARIM
         IF( MNR(IMOR) .EQ. 0 ) GO TO 10                                 PCVARIM
         NR = MNR(IMOR)                                                  PCVARIM
         IMOR = IMOR+1                                                   PCVARIM
     GO TO 325                                                           PCVARIM
                                                                         PCVARIM
 360 FORMAT (1H1,20X,13A6,A2)                                            PCVARIM
 370 FORMAT(6I6,F12.6,,T61,A7)                                           PCVARIM
 380 FORMAT (1H1,55HROTATION NOT DONE SINCE ONLY ONE FACTOR TO BE ROTAT PCVARIM
    1ED  )                                                               PCVARIM
 390 FORMAT (1H1,11HINPUT CARDS//,13A6,A2)                               PCVARIM
 400 FORMAT (1H ,13A6,A2)                                                PCVARIM
 410 FORMAT (1H0,16X,4HMEAN,16X,4HS.D.//(1H ,I6,2F18.6))                 PCVARIM
 420 FORMAT (1H ,I3,10F11.4)                                             PCVARIM
 430 FORMAT (1H1,2X,12HCORRELATIONS,10X,13A6,A2//1H ,2X,10I11//)         PCVARIM
 440 FORMAT (1H1,20X,13A6,A2//1H ,11HEIGENVALUES//)                      PCVARIM
 450 FORMAT (1H ,5X,I3,5X,F12.6)                                         PCVARIM
 460 FORMAT (1H0,12HSUM OF FIRSTI3,15H EIGENVALUES = F12.6)              PCVARIM
 470 FORMAT (1H1,2X,22HPRINCIPAL AXIS FACTORS,2X,13A6,A2//1H ,2X,10I11/ PCVARIM
    1/)                                                                  PCVARIM
 480 FORMAT (1H0,12HTHE TRACE IS,F18.6)                                  PCVARIM
 490 FORMAT (I3,I1)                                                      PCVARIM
 500 FORMAT (13A6,A2)                                                    PCVARIM
                                                                         PCVARIM
 350 CONTINUE                                                            PCVARIM
     END                                                                 PCVARIM
```

```
      SUBROUTINE VARIM(NROWS,NCOLS,NVPCH)                               VARIMP
                                                                        PCV-COM
*     A - CORRELATION MATRIX                                            PCV-COM
*     S - VARIMAX ROTATED FACTORS                                       PCV-COM
      COMMON// A(144,144),S(144,12),H(145),V(145)                       PCV-COM
      REAL ROOT(10),VAR(10),CUMV(10)                                    VARIM2
C                                                                       VARIM2
C     VARIMAX FACTOR ROTATION                                           VARIM2
C                                                                       VARIM2
C     OUTPUT COMMUNALITIES AND NORMALIZE FACTORS                        VARIM2
C                                                                       VARIM2
      SUM = 0.0                                                         VARIM2
      DO 10 I = 1,NCOLS                                                 VARIM2
   10 SUM = SUM+V(I)                                                    VARIM2
      WRITE (6,260) NCOLS,SUM                                           VARIM2
      WRITE (6,270)                                                     VARIM2
      SUM = 0.0                                                         VARIM2
      DO 20 I = 1,NROWS                                                 VARIM2
      WRITE (6,280) I,H(I)                                              VARIM2
      SUM = SUM+H(I)                                                    VARIM2
   20 H(I) = SQRT(H(I))                                                 VARIM2
      WRITE (6,290) SUM                                                 VARIM2
      DO 30 J = 1,NCOLS                                                 VARIM2
      DO 30 I = 1,NROWS                                                 VARIM2
   30 A(I,J) = A(I,J)/H(I)                                              VARIM2
      KROT = (NCOLS*(NCOLS-1))/2                                        VARIM2
   40 KX = 1                                                            VARIM2
      KZCT = 0                                                          VARIM2
C                                                                       VARIM2
C     COMPUTE SUMS A,B,C AND D                                          VARIM2
C                                                                       VARIM2
```

```
   50 KY = KX+1                                                    VARIM2
   60 CONA = 0.                                                    VARIM2
      CONB = 0.                                                    VARIM2
      CONC = 0.                                                    VARIM2
      COND = 0.                                                    VARIM2
      DO 70 I = 1,NROWS                                            VARIM2
      U = (A(I,KX)+A(I,KY))*(A(I,KX)-A(I,KY))                      VARIM2
      V = 2.0*A(I,KX)*A(I,KY)                                      VARIM2
      CONA = CONA+U                                                VARIM2
      CONB = CONB+V                                                VARIM2
      CONC = CONC+(U+V)*(U-V)                                      VARIM2
   70 COND = COND+2.0*U*V                                          VARIM2
      FLN = FLOAT(NROWS)                                           VARIM2
      TNUM = COND-(2.0*CONA*CONB)/FLN                              VARIM2
      TDEN = CONC-((CONA+CONB)*(CONA-CONB))/FLN                    VARIM2
      ANUM = ABS(TNUM)                                             VARIM2
      ADEN = ABS(TDEN)                                             VARIM2
      IF (ANUM-ADEN) 90,80,100                                     VARIM2
   80 IF (ANUM.EQ.0) GO TO 180                                     VARIM2
      GO TO 105                                                    VARIM2
C                                                                  VARIM2
C     NUM LESS THAN DEN                                            VARIM2
C                                                                  VARIM2
   90 TAN4T = ANUM/ADEN                                            VARIM2
      IF (TAN4T.LT.0.06993) GO TO 190                              VARIM2
      COS4T = 1.0/SQRT(1.0+TAN4T**2)                               VARIM2
      SIN4T = TAN4T*COS4T                                          VARIM2
      GO TO 110                                                    VARIM2
C                                                                  VARIM2
C     NUM GREATER THAN DEN                                         VARIM2
C                                                                  VARIM2
```

```
  100 CTN4T = ADEN/ANUM                                                VARIM2
      IF (CTN4T.LT.0.06993) GO TO 200                                  VARIM2
      SIN4T = 1.0/SQRT(1.0+CTN4T**2)                                   VARIM2
      COS4T = CTN4T*SIN4T                                              VARIM2
      GO TO 110                                                        VARIM2
C                                                                      VARIM2
C     NUM EQUAL DEN, BOTH NON-ZERO                                     VARIM2
C                                                                      VARIM2
C                                                                      VARIM2
  105 SIN4T = 0.707107                                                 VARIM2
      COS4T = SIN4T                                                    VARIM2
C                                                                      VARIM2
C     COMPUTE ANGLE OF ROTATION                                        VARIM2
C                                                                      VARIM2
  110 COS2T = SQRT(0.5+0.5*COS4T)                                      VARIM2
      SIN2T = SIN4T/(2.0*COS2T)                                        VARIM2
      COST = SQRT(0.5+0.5*COS2T)                                       VARIM2
      SINT = SIN2T/(2.0*COST)                                          VARIM2
      IF (TDEN.LT.0) GO TO 120                                         VARIM2
      COSPHI = COST                                                    VARIM2
      SINPHI = SINT                                                    VARIM2
      GO TO 130                                                        VARIM2
  120 COSPHI = 0.707107*(COST+SINT)                                    VARIM2
      SINPHI = ABS( 0.707107*(COST-SINT) )                             VARIM2
  130 IF (TNUM.LT.0) SINPHI = -SINPHI                                  VARIM2
C                                                                      VARIM2
C     ROTATE FACTORS                                                   VARIM2
C                                                                      VARIM2
  140 DO 150 I = 1,NROWS                                               VARIM2
      U = A(I,KX)*COSPHI+A(I,KY)*SINPHI                                VARIM2
      A(I,KY) = -SINPHI*A(I,KX)+A(I,KY)*COSPHI                         VARIM2
  150 A(I,KX) = U                                                      VARIM2
  160 IF (KY.EQ.NCOLS) GO TO 170                                       VARIM2
      KY = KY+1                                                        VARIM2
      GO TO 60                                                         VARIM2
```

```
  170 KX = KX+1                                              VARIM2
      IF (KX.NE.NCOLS) GO TO 50                              VARIM2
      IF (KZCT.NE.KROT) GO TO 40                             VARIM2
      GO TO 205                                              VARIM2
C                                                            VARIM2
C     NUM AND DEN EQUAL ZERO                                 VARIM2
C                                                            VARIM2
  180 KZCT = KZCT+1                                          VARIM2
      GO TO 160                                              VARIM2
  190 IF (TDEN.GE.0) GO TO 180                               VARIM2
      COSPHI = 0.707107                                      VARIM2
      SINPHI = COSPHI                                        VARIM2
      GO TO 140                                              VARIM2
C                                                            VARIM2
C     DEN / NUM LESS THAN E                                  VARIM2
C                                                            VARIM2
  200 COS4T = 0.0                                            VARIM2
      SIN4T = 1.0                                            VARIM2
      GO TO 110                                              VARIM2
C                                                            VARIM2
C     OUTPUT FACTORS                                         VARIM2
C                                                            VARIM2
 205  DO 220 J = 1,NCOLS                                     VARIM2
      ROOT(J) = 0.0                                          VARIM2
      DO 210 I = 1,NROWS                                     VARIM2
      A(I,J) = A(I,J)*H(I)                                   VARIM2
  210 ROOT(J) = ROOT(J)+A(I,J)**2                            VARIM2
  220 VAR(J) = ROOT(J)/FLN                                   VARIM2
      WRITE (6,300) (I,I = 1,NCOLS)                          VARIM2
      DO 230 I = 1,NROWS                                     VARIM2
  230 WRITE (6,310) I,(A(I,J),J = 1,NCOLS)                   VARIM2
      WRITE (6,320) (ROOT(J),J = 1,NCOLS)                    VARIM2
```

```
      WRITE (6,330) (VAR(J),J = 1,NCOLS)                                VARIM2
      CUMV(1) = VAR(1)                                                  VARIM2
      DO 240 I = 2,NCOLS                                                VARIM2
  240 CUMV(I) = VAR(I)+CUMV(I-1)                                        VARIM2
      WRITE (6,340) (CUMV(J),J = 1,NCOLS)                               VARIM2
      IF (NVPCH.LE.0) RETURN                                            VARIM2
      DO 250 I = 1,NROWS                                                VARIM2
 250  PUNCH 350, I,(A(I,J),J=1,NCOLS)                                   VARIM2
      RETURN                                                            VARIM2
C                                                                       VARIM2
C     FORMAT STATEMENTS                                                 VARIM2
C                                                                       VARIM2
  260 FORMAT (21H-THE SUM OF THE FIRST,I3,9H ROOTS IS,F14.4///)         VARIM2
  270 FORMAT (26H0   NO.      COMMUNALITIES,//)                         VARIM2
  280 FORMAT (1H ,I6,F14.4)                                             VARIM2
  290 FORMAT (28H0THE SUM OF COMMUNALITIES IS,F14.4)                    VARIM2
  300 FORMAT(*1 VAR*,13X,*VARIMAX FACTOR LOADINGS*/3X,*NO.*,I8,9I11)    VARIM2
  310 FORMAT (1X,I5,3X,10F11.3)                                         VARIM2
  320 FORMAT (1H-,8HVARIANCE,10F11.3/)                                  VARIM2
  330 FORMAT (1X,8HPCT VAR ,10F11.3/)                                   VARIM2
  340 FORMAT (1X,8HCUM PCT ,10F11.3)                                    VARIM2
  350 FORMAT(I7,9F8.5/(10F8.5))                                         VARIM2
      END                                                               VARIM2

      SUBROUTINE HOW (LP,NM,M,R,E,V,A,B,W1,W2)                          HOW
      DIMENSION R(1), E(1), V(1), A(1), B(1), W1(1), W2(1)              HOW
      IF (LP-1) 70,60,10                                                HOW
C                                                                       HOW
C  TRI-DIAGONALIZE MATRIX                                               HOW
C                                                                       HOW
   10 CALL TRIDI (LP,NM,R,A,B,W1,W2)                                    HOW
C                                                                       HOW
C  FIND EIGEN VALUES                                                    HOW
C                                                                       HOW
```

```
      CALL EIGVAL (LP,NM,M,E,A,B,W1,W2)                                 HOW
      IF (M.EQ.0) GO TO 30                                              HOW
C                                                                       HOW
C   FIND EIGEN VECTORS                                                  HOW
C                                                                       HOW
      K = IABS(M)                                                       HOW
      J = 1                                                             HOW
      DO 20 I = 1,K                                                     HOW
      CALL EIGVEC (LP,NM,R,A,B,E(I),V(J),W1,W2)                         HOW
   20 J = J+NM                                                          HOW
C                                                                       HOW
C   RESTORE INPUT MATRIX                                                HOW
C                                                                       HOW
   30 NM1 = NM+1                                                        HOW
      JJ = NM1                                                          HOW
      LP2 = LP*NM                                                       HOW
      DO 50 I = 2,LP2,NM1                                               HOW
      K = I                                                             HOW
      DO 40 J = JJ,LP2,NM                                               HOW
      R(K) = R(J)                                                       HOW
   40 K = K+1                                                           HOW
   50 JJ = JJ+NM1                                                       HOW
      GO TO 70                                                          HOW
C                                                                       HOW
C   SOLLTION FOR ORDER 1                                                HOW
C                                                                       HOW
   60 E(1) = R(1)                                                       HOW
      V(1) = 1.0                                                        HOW
      A(1) = R(1)                                                       HOW
      B(1) = 0.0                                                        HOW
   70 RETURN                                                            HOW
      END                                                               HOW
```

```
      SUBROUTINE EIGVAL (LP,NM,M,E,A,B,F,W)                             EIGVAL
C                                                                       EIGVAL
C   EIGEN VALUE SUBROUTINE FOR TRI-DIAGONAL MATRICES     DWM     1517-UB EIGVAL
C                                                                       EIGVAL
      DIMENSION E(1), A(1), B(1), F(1), W(1)                            EIGVAL
      EQUIVALENCE (S1,IS1), (S2,IS2)                                    EIGVAL
C                                                                       EIGVAL
C   FIND UPPER AND LOWER BOUNDS AND NORMALIZE INPUT                     EIGVAL
C                                                                       EIGVAL
      BD = ABS(A(1))                                                    EIGVAL
      DO 10 I = 2,LP                                                    EIGVAL
   10 BD = AMAX1(BD,ABS(A(I))+B(I)**2)                                  EIGVAL
      BD = BD+1.                                                        EIGVAL
      BDR = 1./BD                                                       EIGVAL
      DO 20 I = 1,LP                                                    EIGVAL
      A(I) = A(I)*BDR                                                   EIGVAL
      B(I) = B(I)*BDR                                                   EIGVAL
      W(I) = 1.                                                         EIGVAL
   20 E(I) = -1.                                                        EIGVAL
      DO 150 K = 1,LP                                                   EIGVAL
   30 IF ((W(K)-E(K))/AMAX1(ABS(W(K)),ABS(E(K)),1.0E-9).LE.1.0E-7) GO TO EIGVAL
     1 150                                                              EIGVAL
      X = (W(K)+E(K))*.5                                                EIGVAL
C                                                                       EIGVAL
C   FIND NUMBER OF EIGEN VALUES ,N, GREATER THAN OR EQUAL TO X          EIGVAL
C                                                                       EIGVAL
      IS2 = 1                                                           EIGVAL
      F(1) = A(1)-X                                                     EIGVAL
      IF (F(1).GE.0) GO TO 40                                           EIGVAL
      IS1 = -1                                                          EIGVAL
      N = 0                                                             EIGVAL
      GO TO 50                                                          EIGVAL
```

```
   40 IS1 = 1                                                         EIGVAL
      N = 1                                                           EIGVAL
   50 DO 110 I = 2,LP                                                 EIGVAL
      IF (B(I).EQ.0) GO TO 70                                         EIGVAL
      IF (B(I-1).EQ.0) GO TO 80                                       EIGVAL
      IF (ABS(F(I-1))+ABS(F(I-2)).GE.1.0E-15) GO TO 60                EIGVAL
      F(I-1) = F(I-1)*1.0E15                                          EIGVAL
      F(I-2) = F(I-2)*1.0E15                                          EIGVAL
   60 F(I) = (A(I)-X)*F(I-1)-B(I)**2*F(I-2)                           EIGVAL
      GO TO 90                                                        EIGVAL
   70 F(I) = (A(I)-X)*SIGN(1.,S1)                                     EIGVAL
      GO TO 90                                                        EIGVAL
   80 F(I) = (A(I)-X)*F(I-1)-SIGN(B(I)**2,S2)                         EIGVAL
   90 S2 = S1                                                         EIGVAL
      IF (F(I).EQ.0) GO TO 100                                        EIGVAL
      S1 = SIGN(S1,F(I))                                              EIGVAL
      IF (IS1.EQ.-IS2) GO TO 110                                      EIGVAL
  100 N = N+1                                                         EIGVAL
  110 CONTINUE                                                        EIGVAL
C                                                                     EIGVAL
C  TRAP EIGEN VALUES IN SMALLER AND SMALLER BOUNDS                    EIGVAL
C                                                                     EIGVAL
      N = LP-N                                                        EIGVAL
      IF (N.LT.K) GO TO 130                                           EIGVAL
      DO 120 J = K,N                                                  EIGVAL
  120 W(J) = X                                                        EIGVAL
  130 N = N+1                                                         EIGVAL
      IF (LP.LT.N) GO TO 30                                           EIGVAL
      DO 140 J = N,LP                                                 EIGVAL
      IF (X.LE.E(J)) GO TO 30                                         EIGVAL
  140 E(J) = X                                                        EIGVAL
      GO TO 30                                                        EIGVAL
```

```
  150 CONTINUE                                                        EIGVAL
C                                                                     EIGVAL
C  RESTORE INPUT AND ORDER EIGEN VALUES                               EIGVAL
C                                                                     EIGVAL
      DO 160 I = 1,LP                                                 EIGVAL
      A(I) = A(I)*BD                                                  EIGVAL
      B(I) = B(I)*BD                                                  EIGVAL
  160 F(I) = (W(I)+E(I))*BD*.5                                        EIGVAL
      J = LP                                                          EIGVAL
      K = 1                                                           EIGVAL
      DO 180 I = 1,LP                                                 EIGVAL
      IF (ABS(F(K)).GT.ABS(F(J))) GO TO 170                           EIGVAL
      E(I) = F(J)                                                     EIGVAL
      J = J-1                                                         EIGVAL
      GO TO 180                                                       EIGVAL
  170 E(I) = F(K)                                                     EIGVAL
      K = K+1                                                         EIGVAL
  180 CONTINUE                                                        EIGVAL
      IF (ISIGN(1,M).GE.0) GO TO 210                                  EIGVAL
      DO 190 I = 1,LP                                                 EIGVAL
  190 F(I) = E(I)                                                     EIGVAL
      J = LP                                                          EIGVAL
      DO 200 I = 1,LP                                                 EIGVAL
      E(I) = F(J)                                                     EIGVAL
  200 J = J-1                                                         EIGVAL
  210 CONTINUE                                                        EIGVAL
      RETURN                                                          EIGVAL
      END                                                             EIGVAL
```

```
      SUBROUTINE EIGVEC (LP,NM,R,A,B,E,V,P,Q)                           EIGVEC
      DIMENSION R(1), A(1), B(1), V(1), P(1), Q(1)                      EIGVEC
C                                                                       EIGVEC
C   SET UP SIMULTANEOUS EQUATIONS FOR EIGEN VECTOR WITH EIGEN VALUE E   EIGVEC
C                                                                       EIGVEC
      X = A(1)-E                                                        EIGVEC
      Y = B(2)                                                          EIGVEC
      LP1 = LP-1                                                        EIGVEC
      DO 40 I = 1,LP1                                                   EIGVEC
      IF (ABS(X)-ABS(B(I+1))) 10,20,30                                  EIGVEC
   10 P(I) = B(I+1)                                                     EIGVEC
      Q(I) = A(I+1)-E                                                   EIGVEC
      V(I) = B(I+2)                                                     EIGVEC
      Z = -X/P(I)                                                       EIGVEC
      X = Z*Q(I)+Y                                                      EIGVEC
      IF (LP1.EQ.I) GO TO 40                                            EIGVEC
      Y = Z*V(I)                                                        EIGVEC
      GO TO 40                                                          EIGVEC
   20 IF (X.EQ.0) X = 1.0E-10                                           EIGVEC
   30 P(I) = X                                                          EIGVEC
      Q(I) = Y                                                          EIGVEC
      V(I) = 0.                                                         EIGVEC
      X = A(I+1)-(B(I+1)/X*Y+E)                                         EIGVEC
      Y = B(I+2)                                                        EIGVEC
   40 CONTINUE                                                          EIGVEC
C                                                                       EIGVEC
C   SOLVE SIMULTANEOUS EQUATIONS FOR EIGEN VECTOR OF TRI-DIAGONAL MATRIX EIGVEC
C                                                                       EIGVEC
      IF (X.EQ.0) GO TO 70                                              EIGVEC
      V(LP) = 1./X                                                      EIGVEC
   50 I = LP1                                                           EIGVEC
      V(I) = (1.-Q(I)*V(LP))/P(I)                                       EIGVEC
      X = V(LP)**2+V(I)**2                                              EIGVEC
```

```
   60 I = I-1                                                    EIGVEC
      IF (I.EQ.0) GO TO 80                                       EIGVEC
      V(I) = (1.-(Q(I)*V(I+1)+V(I)*V(I+2)))/P(I)                 EIGVEC
      X = X+V(I)**2                                              EIGVEC
      GO TO 60                                                   EIGVEC
   70 V(LP) = 1.0E10                                             EIGVEC
      GO TO 50                                                   EIGVEC
   80 X = SQRT(X)                                                EIGVEC
      XR = 1./X                                                  EIGVEC
      DO 90 I = 1,LP                                             EIGVEC
   90 V(I) = V(I)*XR                                             EIGVEC
C                                                                EIGVEC
C     TRANSFORM EIGEN VECTOR TO SOLUTION OF ORIGINAL MATRIX      EIGVEC
C                                                                EIGVEC
      J = LP1*NM-NM                                              EIGVEC
      K = LP                                                     EIGVEC
      GO TO 130                                                  EIGVEC
  100 K = K-1                                                    EIGVEC
      J = J-NM                                                   EIGVEC
      Y = 0.0                                                    EIGVEC
      DO 110 I = K,LP                                            EIGVEC
      L = J+I                                                    EIGVEC
  110 Y = Y+V(I)*R(L)                                            EIGVEC
      DO 120 I = K,LP                                            EIGVEC
      L = J+I                                                    EIGVEC
  120 V(I) = V(I)-Y*R(L)                                         EIGVEC
  130 IF (J.NE.0) GO TO 100                                      EIGVEC
      RETURN                                                     EIGVEC
      END                                                        EIGVEC
```

```
      SUBROUTINE TRIDI (LP,NM,R,A,B,W,Q)                                TRIDI
C                                                                       TRIDI
C   TRI-DIAGONALIZATION SUBROUTINE      DWM       1517-J3               TRIDI
C   RETURN ORIGNAL R IN UPPER TRIANGULAR HALF INCLUDING DIAGONAL        TRIDI
C   RETURN MODIFIED W MATRICES IN LOWER HALF OF R MATRIX                TRIDI
C   RETURN NEW DIAGONAL IN A                                            TRIDI
C   RETURN NEW FIRST OFF DIAGONAL IN B                                  TRIDI
C                                                                       TRIDI
      DIMENSION R(1), A(1), B(1), Q(1), W(1)                            TRIDI
      LP1 = LP-1                                                        TRIDI
      LP2 = LP1*NM+LP                                                   TRIDI
      LPP = LP2-NM                                                      TRIDI
      NM1 = NM+1                                                        TRIDI
C                                                                       TRIDI
C   STORE ORIGINAL DIAGONAL                                             TRIDI
C                                                                       TRIDI
      L = 0                                                             TRIDI
      DO 10 I = 1,LP2,NM1                                               TRIDI
      L = L+1                                                           TRIDI
   10 A(L) = R(I)                                                       TRIDI
      B(1) = 0.                                                         TRIDI
      IF (LP-2) 130,120,20                                              TRIDI
   20 KK = 0                                                            TRIDI
      DO 100 K = 2,LP1                                                  TRIDI
      KL = KK+K                                                         TRIDI
      KU = KK+LP                                                        TRIDI
      KJ = K+1                                                          TRIDI
C                                                                       TRIDI
C   CALCULATE AND STORE MODIFIED COLUMN MATRIX W                        TRIDI
C                                                                       TRIDI
      SUM = 0.0                                                         TRIDI
      DO 30 J = KL,KU                                                   TRIDI
```

```
   30 SUM = SUM+R(J)**2                                                        TRIDI
      S = SQRT(SUM)                                                            TRIDI
      B(K) = SIGN(S,-R(KL))                                                    TRIDI
      S = 1./S                                                                 TRIDI
      W(K) = SQRT(ABS(R(KL))*S+1.)                                             TRIDI
      X = SIGN(S/W(K),R(KL))                                                   TRIDI
      R(KL) = W(K)                                                             TRIDI
      DO 40 I = KJ,LP                                                          TRIDI
      JJ = I+KK                                                                TRIDI
      W(I) = X*R(JJ)                                                           TRIDI
   40 R(JJ) = W(I)                                                             TRIDI
C                                                                              TRIDI
C  CALCULATE NEW R MATRIX WITH ROW K-1 NOW HAVING ZEROS OFF 2ND DIAGONAL      TRIDI
C                                                                              TRIDI
      DO 60 J = K,LP                                                           TRIDI
      JJ = J+1                                                                 TRIDI
      Q(J) = 0.0                                                               TRIDI
      L = KK+J                                                                 TRIDI
      DO 50 I = K,J                                                            TRIDI
      L = L+NM                                                                 TRIDI
   50 Q(J) = Q(J)+R(L)*W(I)                                                    TRIDI
      IF (JJ.GT.LP) GO TO 70                                                   TRIDI
      DO 60 I = JJ,LP                                                          TRIDI
      L = L+1                                                                  TRIDI
   60 Q(J) = Q(J)+R(L)*W(I)                                                    TRIDI
   70 X = 0.0                                                                  TRIDI
      DO 80 J = K,LP                                                           TRIDI
   80 X = X+W(J)*Q(J)                                                          TRIDI
      X = .5*X                                                                 TRIDI
      DO 90 I = K,LP                                                           TRIDI
```

```
   90 Q(I) = X*W(I)-Q(I)                                                TRIDI
      LL = KK                                                           TRIDI
      KK = KK+NM                                                        TRIDI
      DO 100 I = K,LP                                                   TRIDI
      LL = LL+NM                                                        TRIDI
      DO 100 J = I,LP                                                   TRIDI
      L = LL+J                                                          TRIDI
  100 R(L) = R(L)+Q(I)*W(J)+Q(J)*W(I)                                   TRIDI
C                                                                       TRIDI
C  SORT OUTPUT                                                          TRIDI
C                                                                       TRIDI
      L = 1                                                             TRIDI
      DO 110 I = 1,LP                                                   TRIDI
      X = A(I)                                                          TRIDI
      A(I) = R(L)                                                       TRIDI
      R(L) = X                                                          TRIDI
  110 L = L+NM1                                                         TRIDI
  120 B(LP) = R(LPP)                                                    TRIDI
  130 RETURN                                                            TRIDI
      END                                                               TRIDI
```

APPENDIX D

EDIT WITH TWO-MODE FACTOR ANALYSIS PART ONE (X2MOFA)

The HSMS "two-mode" factor analysis program X2MOFA is based on the Tucker-Messick procedure for factoring an individual differences matrix to obtain idealized subject types.[1] In the HSMS application, the subject types are task groupings, and the variables are the scaled skills and knowledge categories. HSMS uses the X2MOFA program after it has utilized PCVARIM to select the number of principal axis factors to rotate.[2]

The HSMS "two-mode" program is run in two parts to overcome the problem presented by the great amount of core storage required for the program. The output of "Part One" (X2MOFA) provides card inputs for "Part Two" (X2MFA2).

X2MOFA is always used by HSMS interfaced with EDIT [3] because of the nature of the task data inputs and the need to utilize EDIT to provide a logarithmic transformation of the data.

---

1 The program presented in this appendix was developed at the University of Illinois. It is a modification of the original Tucker-Messick technique. X2MOFA was programmed by Sharon Wolf and Ping Kao under the direction of L.R. Tucker, H.C. Triandis, and E.E. Davis in the period 1964 to 1966. It was adpated for HSMS for use with the Control Data Corporation (CDC) 6600 computer by Katherine Kurtz in 1968. Aspects of the technique are described in Appendix G of this report, where further references are cited.

2 The PCVARIM program appears in Appendix C.

3 The EDIT program appears in Appendix B.

DESCRIPTION

X2MOFA is written in FORTRAN IV, and was used in the Control Data Corporation's (CDC) 6600 computer at the Courant Institute of Mathematical Sciences of New York University. The operating systems in use during HSMS analyses were KRONOS and NOS (Network Operating System). X2MOFA was stored on magnetic tape in compiled and loaded form (i.e., in binary object code in non-relocatable form) and in OLDPL form.

The HSMS task data are transferred to magnetic tape in the form of an OLDPL. A utility program, UPDATE, is used to transfer data to tape, correct data, or to generate temporary local files in the form of a compile file, which is part of the input file for EDIT. The input file for UPDATE is magnetic tape in OLDPL form, and the output file is C=DATA. The input file for EDIT is DATA, and the output file is TAPE9 (with optional printed output). The input file for X2MOFA is TAPE9, and the output file is printed and punched. The punched output is part of the input file for X2MFA2.

The tasks and the skill and knowledge variables enter X2MOFA and X2MFA2 in the order of the final internal numbering assigned by the EDIT program. Therefore, the EDIT "dictionaries" are used to interpret the two-mode outputs.

Options

The two-mode programs allow the user to select whether the observation "mode" or the variable "mode" will be the first to be rotated. As originally used in Illinois, the observation mode was the first to be rotated. For HSMS, since we wish the factor structure of tasks (observations) to reflect the factor structure of variables (skills and knowledges), the variable mode is rotated first. (Thus, once the number of factors is selected based on PCVARIM analysis, the mode one (variable) factors in the X2MFA2 output duplicate those for that particular solution in the PCVARIM output.)

Because it is possible to confuse language references, the following should be borne in mind:

Rows = observations; program allows for more of these than variables.
Columns = variables; program allows for fewer of these than observations.
Mode 1 = whichever is to be rotated (variables or observations).
Mode 2 = whichever is to be counter-rotated (observations or variables).

The X2MOFA program permits the user to select a column-by-column or row-by-row correlation matrix, a column-by-column or row-by-row covariance matrix, or a cross-products matrix. If a cross-products matrix is selected, the user can select a floating point constant to be subtracted from the raw data.

HEDLIN: A System Subroutine

The X2MOFA program which is presented in this appendix calls a series of subroutines. Among these, HEDLIN, which is a systems-based routine, has not been included in the listing. It is assumed that the user can utilize a comparable routine after reading the description to follow.

HEDLIN is a routine to label listings by block letters (FORTRAN callable). It prints a string of up to 50 characters in large block letters (10 columns wide and 10 rows high). Five lines are printed, ten characters per line. The date and time can be included in the message.

Checks

HSMS makes an inspection of the output as an error check as follows:

1. Check that the means and standard deviations seem appropriate to the data.
2. Check that none of the printout listings show zeroes everywhere.
3. Check tha. the diagonals in the correlation matrix (if listed) do not exceed 1.00 and are near .99.

Note: As of September, 1977, the comment cards included in the program had not been updated to correct language references to rotation modes and dimension modifications. The user should follow the instructions for structuring the input file which follows, rather than the comment cards in the program file (which are not reproduced here).

STRUCTURING THE INPUT FILE: EDIT WITH X2MOFA

In the set-up presented below, the NOS system in use in 1977 at the Courant Institute's CDC 6600 is assumed. It is also assumed that the reader can refer to the UPDATE Reference Manual or a counterpart program. In the following instructions b denotes a blank.

INPUT FILE FOR EDIT WITH X2MOFA

Name	Keypunch Cards (one per box)	Instructions
Systems Cards	_ _ _ _ .EDIT WITH TWO MODE ONE	Identification number and other information depending on system.
	USER(_ _ _ _ _ _ _ _)	USER and user code.
	CHARGE(_ _ _ _ _ _ _ _ _ _ _ _ _ _)	Charge card.
	LABEL(OLDPL,VSN=T_ _ _)	Identifies OLDPL tape.
	UPDATE(Q,D,8,C=DATA) or UPDATE(F,D,8,C=DATA)	When selecting tasks using *COMPILE form of UPDATE. When using all data in OLDPL.
	UNLOAD(OLDPL)	
	LABEL(PROGRMS,VSN=T_ _ _)	Identifies program tape.
	COPYBF(PROGRMS,EDIT)	Copies program to local file.
	SKIPF(PROGRMS,1)	Skips over PCVARIM program on tape.
	COPYBF(PROGRMS,X2MOFA)	Copies program to local file.
	UNLOAD(PROGRMS)	
	SETTL(1000)	Time limit.
	RFL(200000)	Defines field length.
	EDIT.	
	REWIND(OUTPUT)	Use only if no printed EDIT output is desired. Usually not needed at this stage.
	HEADER.EDIT WITH TWO MODE ONE	If REWIND(OUTPUT) is used, this is not needed. Prints out heading across a whole page.
	RFL(120000)	
	X2MOFA.	
	End of Record Card	

INPUT FILE FOR EDIT WITH X2MOFA (continued)

Name	Keypunch Cards (one per box)	Instructions
Input Cards for UP DATE	*COMPILEb _ _ _,_ _ _,_ _ _ etc., or *COMPILEb _ _ _,_ _ _._ _ _ (option)	Calls tasks from OLDPL. Each card starts: *COMPILE in Cols. 1-8; Column 9 blank, followed by Task Code Numbers separated by commas; last entry has no comma. Tasks can be in any order. Consecutive Code Numbers can be called as last entry on card after comma or as only entry on card by writing first Code Number, then a period, the last Code Number of the series. A new card must follow.
	End of Record Card.	Needed even if no UPDATE cards.
Input Cards for EDIT	TITLEb b b b b _ _ _ _ _ _ _ etc.	Columns 1-5: punch TITLE; Colums 11-60: punch title of program, date, and any other special identification of this run.
	NOLIST (option)	If listing of data and some error checks are to be suppressed. Usually selected with X2MOFA.
	DELETEb b b b _ _ _ _ _ _ _ _ (option)	One card for each individual skill or knowledge category to be deleted from input data (if not to be deleted by CUTOFF). Columns 1-6: punch DELETE; Columns 11-18: punch 8-character skill code or 8-digit knowledge category code number.
	CUTOFFb b b b _ _ (option)	To eliminate skill or knowledge categories at or below a selected frequency from input data. Columns 1-6: punch CUTOFF; Columns 11-12: punch selected frequency (usually 4 or more) left justified in field.
	NORMALIZE (option)	Usually used when EDIT is used with X2MOFA.
	End of Record Card	

Note: In the following pages options appear which, if selected, could produce results different than those obtained using the options selected by HSMS. In such cases the HSMS choice is indicated.

INPUT FILE FOR EDIT WITH X2MOFA (continued)

Name	Keypunch Cards (one per box)	Instructions
In-put Cards for X2 MOFA	_ etc.	Title Card. Cols. 1-40 read as one line. Cols. 41-80 read as second line. Punch title of program, date, and any other special identification of this run.
Para-Meter Card	Cols. 1-3 _ _ _	Number of observations for rows of input matrix. (For HSMS this is tasks treated as Mode 2.) Up to 700. Right justify in field.
	Cols. 4-6 _ _ _	Number of variables for columns of input matrix. (For HSMS this is skills and knowledge categories treated as Mode 1.) Up to 144. (Redimensioned to 150 but never used.) Right justify.
	Cols. 8-9 _ _	Number of factors to be extracted, up to 12. (For HSMS this is the number of principal axis factors chosen using PCVARIM output.) Right justify.
	Col. 10 <u>1</u> (option)	Punch 1 for column-by-column correlation matrix (always used by HSMS). Punch 2 for column-by-column covariance matrix. Punch 3 for corss-products matrix. See Column 13. Punch 4 for row-by-row correlation matrix. Punch 5 for row-by-row covariance matrix.
	Col. 11 _	Punch 1 for printout of matrix. Punch 0 to suppress printout.
	Col. 12 <u>1</u>	Punch 1 for punched output (always used by HSMS).
	Cols. 13-19 _ _ _ _ _ _ _	If 3 in Column 10: punch floating point constant to be subtracted from raw data. (Right justify.)
	Cols. 61-67 <u>T A P E 9</u> _ _	Name of alternate input file when EDIT is used (left justify).
	End of File Card	

OPT=1 FTN 4.6+428 77/09/15

TWO-MODE PART ONE

```
      OVERLAY(X2MOFA,0,0)                                              KRONOS
      PROGRAM X2MOFA(TAPE9,INPUT,OUTPUT,PUNCH,TAPE1,TAPE2,TAPE3,       KRONOS
     + TAPE7=INPUT,TAPE6=OUTPUT)                                       KRONOS
      DIMENSION FMT(40),T(12,700),Y(150,150),Z(150,12),R1(150),A(150), 11/19/76
     .    B(150),WRK1(150),WRK2(150)                                   X2MOFA
      COMMON // FMT,Y,Z,R1,A,B,WRK1,WRK2,LAST                          X2MOFA
      EQUIVALENCE (T,Y)                                                X2MOFA
           MAX NUMBER OF COLUMNS IN MATRIX ***                         X2MOFA
      DATA MAXC / 150 /                                                X2MOFA
                                                                       X2MOFA
      ZERO BLANK COMMON REGION                                         X2MOFA
                                                                       X2MOFA
      FMT(1) = 10H START                                               HOWNOWA
      FMT(2) = 10H X2MOFA                                              HOWNOWA
      FMT(3) = 10H PART 1                                              HOWNOWA
      CALL HEDLIN(FMT)                                                 HOWNOWA
      WRITE(6,1)                                                       HOWNOWA
    1 FORMAT(1H1)                                                      HOWNOWA
      LF = LOCF(FMT)                                                   HOWNOWA
      LL = LOCF(LAST)                                                  HOWNOWA
      I = 0                                                            HOWNOWA
      DO 10 L=LF,LL                                                    HOWNOWA
      I = I + 1                                                        HOWNOWA
   10 FMT(I) = 0                                                       X2MOFA
      IN = 7                                                           KRONOS
      II = 40                                                          KRONOS
      READ(IN,530)NI,NJ,NROT,NSTD,NPRT,IPCH,SUBT,INTAP,ILFN            KRONOS
      WRITE (6,531) NI,NJ,NROT,NSTD,NPRT,IPCH,SUBT,INTAP,ILFN          HOWNOWA
```

```
  531 FORMAT(1X,3I3,3I1,F7.0,I2,T61,A7)                                   HOWNOWA
      FNI = FLOAT(NI)                                                     X2MOFA
      FNJ = FLOAT(NJ)                                                     X2MOFA
      SFNI = SQRT(FNI)                                                    X2MOFA
      SFNJ = SQRT(FNJ)                                                    X2MOFA
      IF (ILFN.EQ.7H          ) GO TO 15                                  KRONOS2
      IN = 9                                                              KRONOS
C  THIS VALUE IS SET FOR THE LENGTH OF FORMATS CREATED BY EDITFA          HOWNOWA
      II = 3                                                              HOWNOWA
   15 READ(IN,540)(FMT(I),I=1,II)                                         KRONOS
      WRITE (6,541) (FMT(I),I=1,II)                                       HOWNOWA
  541 FORMAT(1X,8A10)                                                     HOWNOWA
      REWIND 3                                                            X2MOFA
      GO TO ( 30,20,40,50,60) , NSTD                                      X2MOFA
   20 WRITE (6,650)                                                       X2MOFA
      GO TO 180                                                           X2MOFA
   30 WRITE (6,670)                                                       X2MOFA
      GO TO 180                                                           X2MOFA
   40 WRITE (6,680) SUBT                                                  X2MOFA
      GO TO 80                                                            X2MOFA
   50 WRITE (6,690)                                                       X2MOFA
      GO TO 70                                                            X2MOFA
   60 WRITE (6,700)                                                       X2MOFA
   70 WRITE (6,690)                                                       X2MOFA
   80 CONTINUE                                                            HOWNOWA
      IF (INTAP.NE.0) IN = INTAP                                          HOWNOWA
      DO 170 I = 1,NI                                                     HOWNOWA
      WRK1(1) = 0.0                                                       HOWNOWA
      WRK2(1) = 0.0                                                       HOWNOWA
   90 READ(IN,FMT)(A(J),J=1,NJ)                                           KRONOS
  100 DO 110 J = 1,NJ                                                     X2MOFA
      WRK2(1) = WRK2(1)+A(J)**2                                           X2MOFA
```

```
  110 WRK1(1) = WRK1(1) + A(J)                            H3WNOWA
      WRK1(1) = WRK1(1) / FNJ                             H3WNOWA
      WRK2(1) = WRK2(1)-FNJ*(WRK1(1)**2)                  X2MOFA
      WRK2(1) = SQRT(WRK2(1))/SFNJ                        X2MOFA
      WRITE (6,550) I,WRK1(1),WRK2(1)                     X2MOFA
      DO 150 J = 1,NJ                                     X2MOFA
      IF (NSTD-4) 120,130,140                             X2MOFA
  120 A(J) = A(J)-SUBT                                    X2MOFA
      GO TO 150                                           X2MOFA
  130 A(J) = (A(J)-WRK1(1))/(SFNJ*WRK2(1))                X2MOFA
      GO TO 150                                           X2MOFA
  140 A(J) = (A(J)-WRK1(1))/SFNJ                          X2MOFA
  150 CONTINUE                                            X2MOFA
      DO 160 J = 1,NJ                                     X2MOFA
      DO 160 K = 1,NJ                                     X2MOFA
  160 Y(K,J) = Y(K,J)+A(J)*A(K)                           X2MOFA
C                                                         X2MOFA
  170 WRITE (3) (A(J),J = 1,NJ)                           X2MOFA
      GO TO 310                                           H3WNOWA
C                                                         X2MOFA
C     NSTD = 1 OR 2                                       X2MOFA
C                                                         X2MOFA
  180 REWIND 2                                            X2MOFA
      DO 200 I = 1,NI                                     X2MOFA
      READ(IN,FMT)(A(J),J=1,NJ)                           KRUNOS
      DO 190 J = 1,NJ                                     X2MOFA
      WRK2(J) = WRK2(J)+A(J)**2                           X2MOFA
  190 WRK1(J) = WRK1(J)+A(J)/FNI                          X2MOFA
  200 WRITE (2) (A(J),J = 1,NJ)                           X2MOFA
C                                                         X2MOFA
C     NOW HAVE X MATRIX ON TAPE 2                         X2MOFA
C                                                         X2MOFA
```

```
      WRITE (6,660)                                                     X2MOFA
      DO 210 J = 1,NJ                                                   X2MOFA
      WRK2(J) = WRK2(J)-FNI*((WRK1(J))**2)                              X2MOFA
      WRK2(J) = SQRT(WRK2(J))/SFNI                                      X2MOFA
  210 WRITE (6,550) J,WRK1(J),WRK2(J)                                   X2MOFA
      REWIND 2                                                          X2MOFA
      DO 280 I = 1,NI                                                   X2MOFA
      READ (2) (A(J),J = 1,NJ)                                          X2MOFA
      IF (NSTD-2) 220,240,520                                           X2MOFA
  220 DO 230 J = 1,NJ                                                   X2MOFA
  230 A(J) = (A(J)-WRK1(J))/(SFNI*WRK2(J))                              X2MOFA
      GO TO 260                                                         X2MOFA
  240 DO 250 J = 1,NJ                                                   X2MOFA
  250 A(J) = (A(J)-WRK1(J))/SFNI                                        X2MOFA
  260 DO 270 J = 1,NJ                                                   X2MOFA
      DO 270 K = 1,NJ                                                   X2MOFA
  270 Y(K,J) = Y(K,J)+A(J)*A(K)                                         X2MOFA
  280 WRITE (3) (A(J),J = 1,NJ)                                         X2MOFA
C                                                                       X2MOFA
C      NOW HAVE X*X IN Y (COVS, CORR, OR XPRODS) AND STD VARS ON TAPE 3 X2MOFA
C                                                                       X2MOFA
      IF (NPRT.LE.0) GO TO 310                                          X2MOFA
      L2 = 0                                                            X2MOFA
  290 L1 = L2+1                                                         X2MOFA
      L2 = MIN0(L2+16,NJ)                                               X2MOFA
      WRITE (6,570) (I,I = L1,L2)                                       X2MOFA
      DO 300 I = 1,NJ                                                   X2MOFA
  300 WRITE (6,550) I,(Y(I,J),J = L1,L2)                                X2MOFA
      IF (L2.LT.NJ) GO TO 290                                           X2MOFA
C                                                                       X2MOFA
C                                                                       X2MOFA
 310  CALL HOW( NJ,MAXC,NROT,Y,R1,Z,A,B,WRK1,WRK2)                      X2MOFA
C                                                                       X2MOFA
C     EIGEN VECTORS NOW IN Z(NI BY NROT), AND ROOTS ARE IN R1(NROT)     X2MOFA
C     NOW CALCULATE MATRIX OF FACTOR SCORES, Z* * X                     X2MOFA
C     THE FACTOR SCORES WILL END UP AS PA FACTORS IN T(NROT, NI)        X2MOFA
C                                                                       X2MOFA
```

```
      REWIND 3                                                          X2MOFA
      DO 320 I = 1,NROT                                                 X2MOFA
      DO 320 J = 1,NI                                                   X2MOFA
  320 T(I,J) = 0.                                                       X2MOFA
      DO 330 J = 1,NI                                                   X2MOFA
      READ (3) (A(JJ),JJ = 1,NJ)                                        X2MOFA
      DO 330 I = 1,NROT                                                 X2MOFA
      DO 330 K = 1,NJ                                                   X2MOFA
  330 T(I,J) = Z(K,I)*A(K)+T(I,J)                                       X2MOFA
C                                                                       X2MOFA
C     NOW OUTPUT ROOTS, PERCENT VARIANCE, ETC.                          X2MOFA
C                                                                       X2MOFA
      CUMVAR = 0.                                                       X2MOFA
      TOTVAR = 0.                                                       X2MOFA
      DO 340 J = 1,NJ                                                   X2MOFA
  340 TOTVAR = TOTVAR+R1(J)                                             X2MOFA
      WRITE (6,580) TOTVAR                                              X2MOFA
      WRITE (6,560)                                                     X2MOFA
      DO 350 J = 1,NJ                                                   X2MOFA
      PCT = (R1(J)/TOTVAR)*100.                                         X2MOFA
      CUMVAR = CUMVAR+PCT                                               X2MOFA
  350 WRITE (6,590) J,R1(J),PCT,CUMVAR                                  X2MOFA
C                                                                       X2MOFA
C     PRINT CHARACTERISTIC VECTORS FOR MODE 1                           X2MOFA
C                                                                       X2MOFA
      WRITE (6,640) NJ                                                  X2MOFA
      WRITE (6,570) (K,K = 1,NROT)                                      X2MOFA
      DO 360 J = 1,NJ                                                   X2MOFA
  360 WRITE (6,550) J,(T(J,K),K = 1,NROT)                               X2MOFA
C                                                                       X2MOFA
C                                                                       X2MOFA
C     NOW COMPUTE PA FACTORS FOR MODE 1                                 X2MOFA
C                                                                       X2MOFA
```

```
      DO 390 K = 1,NROT                                                X2MOFA
      IF (R1(K).GT..001) GO TO 370                                     X2MOFA
      R1(K) = 0.                                                       X2MOFA
      GO TO 380                                                        X2MOFA
  370 R1(K) = SQRT(R1(K))                                              X2MOFA
  380 DO 390 J = 1,NJ                                                  X2MOFA
  390 Z(J,K) = Z(J,K)*R1(K)                                            X2MOFA
      WRITE (6,600) NJ                                                 X2MOFA
      WRITE (6,570) (K,K = 1,NROT)                                     X2MOFA
      DO 400 J = 1,NJ                                                  X2MOFA
  400 WRITE (6,550) J,(Z(J,K),K = 1,NROT)                              X2MOFA
      IF (IPCH.LE.0) GO TO 420                                         X2MOFA
C                                                                      X2MOFA
C     PUNCH MODE ONE                                                   X2MOFA
C                                                                      X2MOFA
      L = 1                                                            X2MOFA
      LFT2 = MINO(12,NROT)                                             X2MOFA
      DO 410 J = 1,NJ                                                  X2MOFA
  410 PUNCH 620, J,L,(Z(J,K),K = 1,LFT2)                               X2MOFA
C                                                                      X2MOFA
C                   1/6/70                  $$$$$$$$$$$ $$$$$$$$$      X2MOFA
C                                                                      X2MOFA
C     NOW PRINT PA FACTORS FOR MODE TWO                                X2MOFA
C                                                                      X2MOFA
  420 WRITE (6,600) NI                                                 X2MOFA
      WRITE (6,570) (K,K = 1,NROT)                                     X2MOFA
      DO 430 I = 1,NI                                                  X2MOFA
  430 WRITE (6,550) I,(T(K,I),K = 1,NROT)                              X2MOFA
      IF (IPCH.LE.0) GO TO 450                                         X2MOFA
C                                                                      X2MOFA
C     PUNCH MODE TWO                                                   X2MOFA
C                                                                      X2MOFA
```

```
      L = 2                                                               X2MOFA
      LFT2 = MIN0(12,NROT)                                                X2MOFA
      DO 440 I = 1,NI                                                     X2MOFA
  440 PUNCH 620, I,L,(T(K,I),K = 1,LFT2)                                  X2MOFA
C                                                                         X2MOFA
C     PRINT CHARACTERISTIC VECTORS FOR MODE 2                             X2MOFA
C                                                                         X2MOFA
  450 WRITE (6,640) NI                                                    X2MOFA
      DO 480 K = 1,NROT                                                   X2MOFA
      IF (R1(K).GT..001) GO TO 460                                        X2MOFA
      R1(K) = 0.                                                          X2MOFA
      GO TO 470                                                           X2MOFA
  460 R1(K) = 1.0/R1(K)                                                   X2MOFA
  470 DO 480 I = 1,NI                                                     X2MOFA
  480 T(K,I) = T(K,I)*R1(K)                                               X2MOFA
      WRITE (6,570) (K,K = 1,NROT)                                        X2MOFA
      DO 490 I = 1,NI                                                     X2MOFA
  490 WRITE (6,550) I,(T(K,I),K = 1,NROT)                                 X2MOFA
C                                                                         X2MOFA
C     NOW CALCULATE AND PRINT CORE MATRIX                                 X2MOFA
C                                                                         X2MOFA
      WRITE (6,710)                                                       X2MOFA
      DO 500 I = 1,NROT                                                   X2MOFA
  500 WRITE (6,610) I,R1(I)                                               X2MOFA
      IF (IPCH.LE.0) GO TO 520                                            X2MOFA
C                                                                         X2MOFA
C     PUNCH CORE MATRIX                                                   X2MOFA
C                                                                         X2MOFA
      DO 510 I = 1,NROT                                                   X2MOFA
  510 PUNCH 630, I,R1(I)                                                  X2MOFA
C                                                                         X2MOFA
  520 CONTINUE                                                            X2MOFA
C                                                                         X2MOFA
```

```
C     FORMAT STATEMENTS                                                   X2MOFA
C                                                                         X2MOFA
  530 FORMAT(3I3,3I1,F7.0,I2,T61,A7)                                      X2MOFA
  540 FORMAT(8A10)                                                        KRONOS2
  550 FORMAT (1H ,I3,16F8.3)                                              X2MOFA
  560 FORMAT (58H     CHARACTERISTIC ROOTS  PERCENT VARIANCE  CUM. PERC   X2MOFA
     1ENT)                                                                X2MOFA
  570 FORMAT (/////3H   ,16I8/)                                           FIXFMT
  580 FORMAT (/////18H TOTAL VARIANCE = ,F12.3/)                          FIXFMT
  590 FORMAT (1H I3,5X,F12.3,6X,F12.2,1X,F12.2)                           X2MOFA
  600 FORMAT(/////27H PRINCIPAL AXES FACTORS FOR,I4,14H-VARIABLE MODE)    FIXFMT
  610 FORMAT (1H I3,F12.3)                                                X2MOFA
  620 FORMAT (I3,I1,12F6.2)                                               X2MOFA
  630 FORMAT (I3,F12.7)                                                   X2MOFA
  640 FORMAT(/////44H CHARACTERISTIC VECTORS (FACTOR SCORES) FOR ,I4,     FIXFMT
     P 14H-VARIABLE MODE)                                                 FIXFMT
  650 FORMAT(/////30H  COLUMN BY COLUMN COVARIANCES  ////)                FIXFMT
  660 FORMAT (31H1MEANS AND STANDARD DEVIATIONS ///)                      X2MOFA
  670 FORMAT(/////30H COLUMN BY COLUMN CORRELATIONS  ////)                FIXFMT
  680 FORMAT(/////16H CROSS PRODUCTS.,F8.0,33H WAS SUBTRACTED FROM EACH   FIXFMT
     +ENTRY.   ////)                                                      FIXFMT
  690 FORMAT(/////24H ROW BY ROW CORRELATIONS   ///)                      FIXFMT
  700 FORMAT(/////23H ROW BY ROW COVARIANCES   ////)                      FIXFMT
  710 FORMAT(/////36H CORE MATRIX, EQUALS 1.0/SQRT(ROCT), /56H  SO IS CO  FIXFMT
     +RE MATRIX FOR THE PA FACTORS OF THE TWO MODES.   ///)               FIXFMT
      END                                                                 X2MOFA
```

```
      SUBROUTINE HOW (LP,NM,M,R,E,V,A,B,W1,W2)                          HOW
      DIMENSION R(1), E(1), V(1), A(1), B(1), W1(1), W2(1)              HOW
      IF (LP-1) 70,60,10                                                HOW
C                                                                       HOW
C   TRI-DIAGONALIZE MATRIX                                              HOW
C                                                                       HOW
   10 CALL TRIDI (LP,NM,R,A,B,W1,W2)                                    HOW
C                                                                       HOW
C   FIND EIGEN VALUES                                                   HOW
C                                                                       HOW
      CALL EIGVAL (LP,NM,M,E,A,B,W1,W2)                                 HOW
      IF (M.EQ.0) GO TO 30                                              HOW
C                                                                       HOW
C   FIND EIGEN VECTORS                                                  HOW
C                                                                       HOW
      K = IABS(M)                                                       HOW
      J = 1                                                             HOW
      DO 20 I = 1,K                                                     HOW
      CALL EIGVEC (LP,NM,R,A,B,E(I),V(J),W1,W2)                         HOW
   20 J = J+NM                                                          HOW
C                                                                       HOW
C   RESTORE INPUT MATRIX                                                HOW
C                                                                       HOW
   30 NM1 = NM+1                                                        HOW
      JJ = NM1                                                          HOW
      LP2 = LP*NM                                                       HOW
      DO 50 I = 2,LP2,NM1                                               HOW
      K = I                                                             HOW
      DO 40 J = JJ,LP2,NM                                               HOW
      R(K) = R(J)                                                       HOW
   40 K = K+1                                                           HOW
```

```
   50 JJ = JJ+NM1                                                          HOW
      GO TO 70                                                             HOW
C                                                                          HOW
C  SOLUTION FOR ORDER 1                                                    HOW
C                                                                          HOW
   60 E(1) = R(1)                                                          HOW
      V(1) = 1.0                                                           HOW
      A(1) = R(1)                                                          HOW
      B(1) = 0.0                                                           HOW
   70 RETURN                                                               HOW
      END                                                                  HOW

      SUBROUTINE EIGVAL (LP,NM,M,E,A,B,F,W)                                EIGVAL
C                                                                          EIGVAL
C  EIGEN VALUE SUBROUTINE FOR TRI-DIAGONAL MATRICES     DWM     1517-UB    EIGVAL
C                                                                          EIGVAL
      DIMENSION E(1), A(1), B(1), F(1), W(1)                               EIGVAL
      EQUIVALENCE (S1,IS1), (S2,IS2)                                       EIGVAL
C                                                                          EIGVAL
C  FIND UPPER AND LOWER BOUNDS AND NORMALIZE INPUT                         EIGVAL
C                                                                          EIGVAL
      BD = ABS(A(1))                                                       EIGVAL
      DO 10 I = 2,LP                                                       EIGVAL
   10 BD = AMAX1(BD,ABS(A(I))+B(I)**2)                                     EIGVAL
      BD = BD+1.                                                           EIGVAL
      BDR = 1./BD                                                          EIGVAL
      DO 20 I = 1,LP                                                       EIGVAL
      A(I) = A(I)*BDR                                                      EIGVAL
      B(I) = B(I)*BDR                                                      EIGVAL
      W(I) = 1.                                                            EIGVAL
   20 E(I) = -1.                                                           EIGVAL
      DO 150 K = 1,LP                                                      EIGVAL
   30 IF ((W(K)-E(K))/AMAX1(ABS(W(K)),ABS(E(K)),1.0E-9).LE.1.0E-7) GO TO EIGVAL
     1 150                                                                 EIGVAL
      X = (W(K)+E(K))*.5                                                   EIGVAL
```

```
C                                                                       EIGVAL
C     FIND NUMBER OF EIGEN VALUES ,N, GREATER THAN OR EQUAL TO X        EIGVAL
C                                                                       EIGVAL
      IS2 = 1                                                           EIGVAL
      F(1) = A(1)-X                                                     EIGVAL
      IF (F(1).GE.0) GO TO 40                                           EIGVAL
      IS1 = -1                                                          EIGVAL
      N = 0                                                             EIGVAL
      GO TO 50                                                          EIGVAL
   40 IS1 = 1                                                           EIGVAL
      N = 1                                                             EIGVAL
   50 DO 110 I = 2,LP                                                   EIGVAL
      IF (B(I).EQ.0) GO TO 70                                           EIGVAL
      IF (B(I-1).EQ.0) GO TO 80                                         EIGVAL
      IF (ABS(F(I-1))+ABS(F(I-2)).GE.1.0E-15) GO TO 60                  EIGVAL
      F(I-1) = F(I-1)*1.0E15                                            EIGVAL
      F(I-2) = F(I-2)*1.0E15                                            EIGVAL
   60 F(I) = (A(I)-X)*F(I-1)-B(I)**2*F(I-2)                             EIGVAL
      GO TO 90                                                          EIGVAL
   70 F(I) = (A(I)-X)*SIGN(1.,S1)                                       EIGVAL
      GO TO 90                                                          EIGVAL
   80 F(I) = (A(I)-X)*F(I-1)-SIGN(B(I)**2,S2)                           EIGVAL
   90 S2 = S1                                                           EIGVAL
      IF (F(I).EQ.0) GO TO 100                                          EIGVAL
      S1 = SIGN(S1,F(I))                                                EIGVAL
      IF (IS1.EQ.-IS2) GO TO 110                                        EIGVAL
  100 N = N+1                                                           EIGVAL
  110 CONTINUE                                                          EIGVAL
C                                                                       EIGVAL
C     TRAP EIGEN VALUES IN SMALLER AND SMALLER BOUNDS                   EIGVAL
C                                                                       EIGVAL
      N = LP-N                                                          EIGVAL
      IF (N.LT.K) GO TO 130                                             EIGVAL
      DO 120 J = K,N                                                    EIGVAL
```

```
  120 W(J) = X                                                  EIGVAL
  130 N = N+1                                                   EIGVAL
      IF (LP.LT.N) GO TO 30                                     EIGVAL
      DO 140 J = N,LP                                           EIGVAL
      IF (X.LE.E(J)) GO TO 30                                   EIGVAL
  140 E(J) = X                                                  EIGVAL
      GO TO 30                                                  EIGVAL
  150 CONTINUE                                                  EIGVAL
C                                                               EIGVAL
C  RESTORE INPUT AND ORDER EIGEN VALUES                         EIGVAL
C                                                               EIGVAL
      DO 160 I = 1,LP                                           EIGVAL
      A(I) = A(I)*BD                                            EIGVAL
      B(I) = B(I)*BD                                            EIGVAL
  160 F(I) = (W(I)+E(I))*BD*.5                                  EIGVAL
      J = LP                                                    EIGVAL
      K = 1                                                     EIGVAL
      DO 180 I = 1,LP                                           EIGVAL
      IF (ABS(F(K)).GT.ABS(F(J))) GO TO 170                     EIGVAL
      E(I) = F(J)                                               EIGVAL
      J = J-1                                                   EIGVAL
      GO TO 180                                                 EIGVAL
  170 E(I) = F(K)                                               EIGVAL
      K = K+1                                                   EIGVAL
  180 CONTINUE                                                  EIGVAL
      IF (ISIGN(1,M).GE.0) GO TO 210                            EIGVAL
      DO 190 I = 1,LP                                           EIGVAL
  190 F(I) = E(I)                                               EIGVAL
      J = LP                                                    EIGVAL
      DO 200 I = 1,LP                                           EIGVAL
      E(I) = F(J)                                               EIGVAL
  200 J = J-1                                                   EIGVAL
  210 CONTINUE                                                  EIGVAL
      RETURN                                                    EIGVAL
      END                                                       EIGVAL
```

```
      SUBROUTINE EIGVEC (LP,NM,R,A,B,E,V,P,Q)                           EIGVEC
      DIMENSION R(1), A(1), B(1), V(1), P(1), Q(1)                      EIGVEC
C                                                                       EIGVEC
C   SET UP SIMULTANEOUS EQUATIONS FOR EIGEN VECTOR WITH EIGEN VALUE E   EIGVEC
C                                                                       EIGVEC
      X = A(1)-E                                                        EIGVEC
      Y = B(2)                                                          EIGVEC
      LP1 = LP-1                                                        EIGVEC
      DO 40 I = 1,LP1                                                   EIGVEC
      IF (ABS(X)-ABS(B(I+1))) 10,20,30                                  EIGVEC
   10 P(I) = B(I+1)                                                     EIGVEC
      Q(I) = A(I+1)-E                                                   EIGVEC
      V(I) = B(I+2)                                                     EIGVEC
      Z = -X/P(I)                                                       EIGVEC
      X = Z*Q(I)+Y                                                      EIGVEC
      IF (LP1.EQ.I) GO TO 40                                            EIGVEC
      Y = Z*V(I)                                                        EIGVEC
      GO TO 40                                                          EIGVEC
   20 IF (X.EQ.0) X = 1.0E-10                                           EIGVEC
   30 P(I) = X                                                          EIGVEC
      Q(I) = Y                                                          EIGVEC
      V(I) = 0.                                                         EIGVEC
      X = A(I+1)-(B(I+1)/X*Y+E)                                         EIGVEC
      Y = B(I+2)                                                        EIGVEC
   40 CONTINUE                                                          EIGVEC
C                                                                       EIGVEC
C   SOLVE SIMULTANEOUS EQUATIONS FOR EIGEN VECTOR OF TRI-DIAGONAL MATRIX EIGVEC
C                                                                       EIGVEC
      IF (X.EQ.0) GO TO 70                                              EIGVEC
      V(LP) = 1./X                                                      EIGVEC
   50 I = LP1                                                           EIGVEC
      V(I) = (1.-Q(I)*V(LP))/P(I)                                       EIGVEC
      X = V(LP)**2+V(I)**2                                              EIGVEC
```

```
   60 I = I-1                                                           EIGVEC
      IF (I.EQ.0) GO TO 80                                              EIGVEC
      V(I) = (1.-(Q(I)*V(I+1)+V(I)*V(I+2)))/P(I)                        EIGVEC
      X = X+V(I)**2                                                     EIGVEC
      GO TO 60                                                          EIGVEC
   70 V(LP) = 1.0E10                                                    EIGVEC
      GO TO 50                                                          EIGVEC
   80 X = SQRT(X)                                                       EIGVEC
      XR = 1./X                                                         EIGVEC
      DO 90 I = 1,LP                                                    EIGVEC
   90 V(I) = V(I)*XR                                                    EIGVEC
C                                                                       EIGVEC
C  TRANSFORM EIGEN VECTOR TO SOLUTION OF ORIGINAL MATRIX                EIGVEC
C                                                                       EIGVEC
      J = LP1*NM-NM                                                     EIGVEC
      K = LP                                                            EIGVEC
      GO TO 130                                                         EIGVEC
  100 K = K-1                                                           EIGVEC
      J = J-NM                                                          EIGVEC
      Y = 0.0                                                           EIGVEC
      DO 110 I = K,LP                                                   EIGVEC
      L = J+I                                                           EIGVEC
  110 Y = Y+V(I)*R(L)                                                   EIGVEC
      DO 120 I = K,LP                                                   EIGVEC
      L = J+I                                                           EIGVEC
  120 V(I) = V(I)-Y*R(L)                                                EIGVEC
  130 IF (J.NE.0) GO TO 100                                             EIGVEC
      RETURN                                                            EIGVEC
      END                                                               EIGVEC
```

```
      SUBROUTINE TRIDI (LP,NM,R,A,B,W,Q)                                TRIDI
C                                                                       TRIDI
C  TRI-DIAGONALIZATION SUBROUTINE      DWM        1517-UB               TRIDI
C  RETURN ORIGNAL R IN UPPER TRIANGULAR HALF INCLUDING DIAGONAL         TRIDI
C  RETURN MODIFIED W MATRICES IN LOWER HALF OF R MATRIX                 TRIDI
C  RETURN NEW DIAGONAL IN A                                             TRIDI
C  RETURN NEW FIRST OFF DIAGONAL IN B                                   TRIDI
C                                                                       TRIDI
      DIMENSION R(1), A(1), B(1), Q(1), W(1)                            TRIDI
      LP1 = LP-1                                                        TRIDI
      LP2 = LP1*NM+LP                                                   TRIDI
      LPP = LP2-NM                                                      TRIDI
      NP1 = NM+1                                                        TRIDI
C                                                                       TRIDI
C  STORE ORIGINAL DIAGONAL                                              TRIDI
C                                                                       TRIDI
      L = 0                                                             TRIDI
      DO 10 I = 1,LP2,NP1                                               TRIDI
      L = L+1                                                           TRIDI
   10 A(L) = R(I)                                                       TRIDI
      B(1) = 0.                                                         TRIDI
      IF (LP-2) 130,120,20                                              TRIDI
   20 KK = 0                                                            TRIDI
      DO 100 K = 2,LP1                                                  TRIDI
      KL = KK+K                                                         TRIDI
      KU = KK+LP                                                        TRIDI
      KJ = K+1                                                          TRIDI
C                                                                       TRIDI
C  CALCULATE AND STORE MODIFIED COLUMN MATRIX W                         TRIDI
C                                                                       TRIDI
      SUM = 0.0                                                         TRIDI
      DO 30 J = KL,KU                                                   TRIDI
```

```
   30 SUM = SUM+R(J)**2                                                    TRIDI
      S = SQRT(SUM)                                                        TRIDI
      B(K) = SIGN(S,-R(KL))                                                TRIDI
      S = 1./S                                                             TRIDI
      W(K) = SQRT(ABS(R(KL))*S+1.)                                         TRIDI
      X = SIGN(S/W(K),R(KL))                                               TRIDI
      R(KL) = W(K)                                                         TRIDI
      DO 40 I = KJ,LP                                                      TRIDI
      JJ = I+KK                                                            TRIDI
      W(I) = X*R(JJ)                                                       TRIDI
   40 R(JJ) = W(I)                                                         TRIDI
C                                                                          TRIDI
C  CALCULATE NEW R MATRIX WITH ROW K-1 NOW HAVING ZEROS OFF 2ND DIAGONAL   TRIDI
C                                                                          TRIDI
      DO 60 J = K,LP                                                       TRIDI
      JJ = J+1                                                             TRIDI
      Q(J) = 0.0                                                           TRIDI
      L = KK+J                                                             TRIDI
      DO 50 I = K,J                                                        TRIDI
      L = L+NM                                                             TRIDI
   50 Q(J) = Q(J)+R(L)*W(I)                                                TRIDI
      IF (JJ.GT.LP) GO TO 70                                               TRIDI
      DO 60 I = JJ,LP                                                      TRIDI
      L = L+1                                                              TRIDI
   60 Q(J) = Q(J)+R(L)*W(I)                                                TRIDI
   70 X = 0.0                                                              TRIDI
      DO 80 J = K,LP                                                       TRIDI
   80 X = X+W(J)*Q(J)                                                      TRIDI
      X = .5*X                                                             TRIDI
      DO 90 I = K,LP                                                       TRIDI
```

```
   90 C(I) = X*W(I)-Q(I)                                                TRIDI
      LL = KK                                                           TRIDI
      KK = KK+NM                                                        TRIDI
      DO 100 I = K,LP                                                   TRIDI
      LL = LL+NM                                                        TRIDI
      DO 100 J = I,LP                                                   TRIDI
      L = LL+J                                                          TRIDI
  100 R(L) = R(L)+Q(I)*W(J)+Q(J)*W(I)                                   TRIDI
C                                                                       TRIDI
C  SORT OUTPUT                                                          TRIDI
C                                                                       TRIDI
      L = 1                                                             TRIDI
      DO 110 I = 1,LP                                                   TRIDI
      X = A(I)                                                          TRIDI
      A(I) = R(L)                                                       TRIDI
      R(L) = X                                                          TRIDI
  110 L = I+NM1                                                         TRIDI
  120 B(LP) = R(LPP)                                                    TRIDI
  130 RETURN                                                            TRIDI
      END                                                               TRIDI
```

APPENDIX E

TWO-MODE FACTOR ANALYSIS PART TWO (X2MFA2)

This appendix presents the second half of the HSMS "two-mode" factor analysis program.[1] Part Two (X2MFA2) utilizes the punched output from Part One (X2MOFA). The description to follow indicates how the cards are structured to become the input file for X2MFA2.

DESCRIPTION

X2MFA2 is written in FORTRAN IV and was used in the Control Data Corporation's (CDC) 6600 computer at the Courant Institute of Mathematical Sciences of New York University. The operating systems in use during HSMS analyses were KRONOS and NOS (Network Operating System). X2MFA2 was stored on magnetic tape in compiled and loaded form (i.e., in binary object code in non-relocatable form) and in OLDPL form.

The X2MFA2 program permits the user to select whether the observation "mode" or the variable "mode" will be the one rotated, and which will be counter-rotated.[2] The maximum number of factors is preselected in X2MOFA, but can be fewer than the number selected at that time.

The tasks and the skill and knowledge variables enter X2MOFA and X2MFA2 in the order of the final internal numbering assigned by

---

[1] See Appendix D for a description of Part One (X2MOFA).

[2] Mode 1 = whichever is to be rotated (variables or observations).
Mode 2 = whichever is to be counter-rotated (observations or variables).

the HSMS EDIT program. Therefore, the EDIT "dictionaries" are used to interpret the two-mode outputs.[3]

CHECKS

If the factor structure obtained with the PCVARIM program is for the same mode that is rotated in this program, then the first factor matrix in the printed output should be equivalent, except for rounding errors, to the factor structure obtained with PCVARIM for the given number of factors.[4] For example, HSMS selects a column-by-column correlation matrix to obtain the factor structure of variables. This is the mode rotated in X2MFA2; the factor matrix of PCVARIM for the factor solution chosen (number of rotations) approximates the first and second factor matrixes in the X2MFA2 printout.

As another check, the first and second factor matrixes should be the same except for rounding.

STRUCTURING THE INPUT FILE FOR X2MFA2

The punched output from Part One (X2MOFA) must be separated and arranged in appropriate order for submission. The order for submission reflects the choice of modes to be rotated and counter-rotated. This need not correspond to the order in which the punched output appears; in the case of HSMS usage, the punched output must be rearranged.

---

3 EDIT is described in Appendix B.

4 PCVARIM is described in Appendix C.

The punched output is really three separate decks. Each deck is individually numbered in columns 1 through 4. Column 4 is used to identify all the cards of the same deck; i.e., is blank or is punched with a 1 or a 2. Columns 1 through 3 number the cards for the given deck.

The cards come out in the following order, and are rearranged for submission as indicated in the instructions which follow.

Deck 1: Principal axis factors for the variable mode (HSMS's Mode 1 column mode, the one to be rotated. As many cards as the number of skill and knowledge variables).

Deck 2: Principal axis factors for the observation mode (HSMS's Mode 2 row mode, the one to be counterrotated. As many cards as the number of task observations).

Deck 3: Core matrix cards. As many cards as the number of factors extracted in X2MOFA.

INPUT FILE FOR X2MFA2

Name	Keypunch Cards (one per box)	Instructions
Systems Cards	_ _ _ _ . TWO MODE PART TWO	Identification number and other information depending on system.
	USER(_ _ _ _ _ _ _)	USER and user code.
	CHARGE(_ _ _ _ _ _ _ _ _ _ _ _)	Charge card.
	LABEL(PROGRMS,VSN=T_ _ _)	Identifies OLDPL tape.
	SKIPF(PROGRMS,3)	Skips over EDIT, PCVARIM and X2MOFA on tape.
	COPYBF(PROGRMS,X2MFA2)	Copies program to local file.
	UNLOAD(PROGRMS)	
	RFL(120000)	Defines field length.
	HEADER.TWO MODE 2_ _ _ etc.	Prints out heading across a whole page.
	X2MFA2.	
	End of Record Card	
Input Cards For X2MFA2	Cols. 1-3 _ _ _	Right justify in all fields. Number of units in mode to be rotated. For HSMS this is skill and knowledge variable mode. Up to 700; for HSMS, up to 144. This is Mode 1.
Parameter Card	Cols. 5-6 _ _	Number of factors to be rotated. Up to the number selected for Part One. For HSMS same as factor solution chosen. This is the core matrix.
	Cols. 7-9 _ _ _	Number of units in mode to be counter-rotated. For HSMS this is task mode. Up to 700. This is Mode 2.
	Col. 10 _	Punch 0 for no punched output; Punch 1 for punched output.
	Col. 11 0	Punch 0 to rotate Mode 1 and counter-rotate Mode 2 HSMS always uses this. Punch 1 to rotate each mode separately and counter-rotate the core matrix.

INPUT FILE FOR X2MFA2 (continued)

Name	Keypunch Cards (one per box)	Instructions
Input Cards For X2MFA2 (continued)	Cols. 1-10 (3X,F12.7)	Format card for core matrix deck. For HSMS, punch as shown. (Can be up to 14 columns.)
Core Matrix Deck	Insert core matrix deck next. As many cards as the number of factors to be rotated and counter-rotated.	If the number of factors extracted in Part One was greater than number desired now, use the proper number of cards in the order found in the punched output. Set the remaining cards aside for possible later use.
Mode 1 Deck	Cols. 1-7 (I3,1X, Cols. 8-9 _ _ Cols. 10-14 F6.2)	Format card for Mode 1 deck. For HSMS skill and knowledge variable mode; punch as shown. Punch the number of factors to be rotated. Right justify. (Same as cols. 5-6 in parameter card.) For HSMS punch as shown.
	Insert Mode 1 deck next. As many cards as the number of units in Mode 1.	For HSMS this is the skill and knowledge variable mode deck.
Mode 2 Deck	Cols. 1-7 (I3,1X, Cols. 8-9 _ _ Cols. 10-14 F6.2)	Format card for Mode 2 deck. For HSMS task observation mode; punch as shown. Punch the number of factors to be counter-rotated. Right justify. (Same as cols. 5-6 in parameter card.) For HSMS punch as shown.
	Insert Mode 2 deck next. As many cards as the number of units in Mode 2.	For HSMS this is the task observation mode deck.
	End of File Card	

TWO-MODE PART TWO

FTN 4.6+42?   77/09/15

```
      OVERLAY(X2MFA2,0,0)                                               KRUNOS
      PROGRAM X2MFA2(INPUT,OUTPUT,PUNCH, TAPE7=INPUT,TAPE6=OUTPUT )     X2MFA2
*                                                                       X2MFA2
C$$$  PART II OF 2-MODE FA --- ROTATION                                 X2MFA2
C                                                                       X2MFA2
C     PARATERMETER CARDS                                                X2MFA2
C     1. I3 NR1 = NO. OF VARIABLES OF FIRST MODE FOR ROTATION, TO 700   EXTEND
C        I3 NF = NO. OF FACTOR TO BE ROTATED                            X2MFA2
C        I3 NR2 = DIMENSION OF OTHER MODE, TO 700                       EXTEND
C        I1 NPCH = 1 IF WANT PUNCH OUTPUT, OTHERWISE 0                  X2MFA2
C        I1 LN = OPTION    0---ROTATE FIRST MODE AND COUNTER ROTATE     X2MFA2
C                              OTHER MODE                               X2MFA2
C                          1---ROTATE EACH MODE SEPARATELY,COUNTER ROTAT X2MFA2
C                              THE CORE MATRIX                          X2MFA2
C     2. FORMAT FOR CORE MATRIX                                         X2MFA2
C     3. CORE MATRIX DATA (AS MANY AS NO. OF FACTOR)                    X2MFA2
C     4. FORMAT OF PA FACTOR FOR FIRST MODE                             X2MFA2
C     5. DATA OF PA FACTOR OF FIRST MODE                                X2MFA2
C     6. FORMAT OF PA FACTOR OF OTHER MODE                              X2MFA2
C     7. DATA OF PA FACTOR OF OTHER MODE                                X2MFA2
C                                                                       X2MFA2
         COMMON // A(700,10),R(10),H(700),FMT(14),V(10),S(700,10)       EXTEND
     1   , T(10,10) , TT(10,10)                                         X2M-COM
         REAL VEC(10)                                                   X2MFA2
C                                                                       X2MFA2
C                                                                       X2MFA2
      READ (7,210) NR1,NF,NR2,NPCH,LN                                   X2MFA2
C                                                                       X2MFA2
C     V CONTAIN ROOTS,R CONTAIN 1.0/ROOT                                X2MFA2
C                                                                       X2MFA2
      READ (7,220) FMT                                                  X2MFA2
      READ (7,FMT) (R(I),I = 1,NF)                                      X2MFA2
      DO 10 I = 1,NF                                                    X2MFA2
      V(I) = 1./R(I)**2                                                 X2MFA2
   10 VEC(I) = V(I)                                                     X2MFA2
```

```
C                                                                       X2MFA2
C     READ IN FIRST MODE --PA FACTOR                                    X2MFA2
C                                                                       X2MFA2
      MODE = 1                                                          X2MFA2
      NR = NR1                                                          X2MFA2
   20 READ (7,220) FMT                                                  X2MFA2
      DO 30 I = 1,NR                                                    X2MFA2
      READ (7,FMT) KC,( A(I,J) , J = 1,NF )                             X2MFA2
      IF( KC .NE. I ) CALL ABORT( 32L ** ERROR IN INPUT SET 2         ) X2MFA2
      DO 30 K = 1,NF                                                    X2MFA2
   30 S(I,K) = A(I,K)                                                   X2MFA2
      DO 40 I = 1,NR                                                    X2MFA2
      H(I) = 0.0                                                        X2MFA2
      DO 40 J = 1,NF                                                    X2MFA2
   40 H(I) = H(I)+A(I,J)**2                                             X2MFA2
      WRITE (6,230) NR                                                  X2MFA2
          CALL VARIMX(NR,NF,NPCH)                                       X2MFA2
C                                                                       X2MFA2
C     F* IN A(700,10), F IN S(700,10)                                   EXTEND
C     GET F*F* , THEN PREMULTIPLY BY ROOT**-1                           X2MFA2
C                                                                       X2MFA2
      CALL MSATB (S,NR,NF,A,NR,NF,T,700,700,10)                         EXTEND
      WRITE (6,50)                                                      X2MFA2
   50 FORMAT (40H-FOR SOME CHECK,THERE ARE THE EIGN ROOTS)              X2MFA2
      DO 60 I = 1,NF                                                    X2MFA2
      WRITE (6,250) I,VEC(I)                                            X2MFA2
      DO 60 J = 1,NF                                                    X2MFA2
   60 T(I,J) = T(I,J)/VEC(I)                                            X2MFA2
      WRITE (6,240)                                                     X2MFA2
      DO 70 I = 1,NF                                                    X2MFA2
   70 WRITE (6,250) I,(T(I,J),J = 1,NF)                                 X2MFA2
      WRITE (6,80)                                                      X2MFA2
```

```
   80 FORMAT (16H-FOR CHECK,SS*=I)                                    X2MFA2
      CALL MSATB (T,NF,NF,T,NF,NF,A,10,10,700)                        EXTEND
      DO 90 I = 1,NF                                                  X2MFA2
   90 WRITE (6,250) I,(A(I,J),J = 1,NF)                               X2MFA2
      WRITE (6,100)                                                   X2MFA2
  100 FORMAT (16H-FOR CHECK,FT=F*)                                    X2MFA2
      CALL MSAB (S,NR,NF,T,NF,NF,A,700,10,700)                        EXTEND
      DO 110 I = 1,NR                                                 X2MFA2
  110 WRITE (6,250) I,(A(I,J),J = 1,NF)                               X2MFA2
      IF (LN.GE.1) GO TO 140                                          X2MFA2
C                                                                     X2MFA2
C     COUNTER ROTATION OF OTHER MODE                                  X2MFA2
C                                                                     X2MFA2
      READ (7,220) FMT                                                X2MFA2
      DO 120 I = 1,NR2                                                X2MFA2
      READ (7,FMT) KC,( A(I,J),J = 1,NF)                              X2MFA2
      IF( KC .NE. I ) CALL ABORT( 33L ** ERROR IN INPUT SET 3       ) X2MFA2
 120  CONTINUE                                                        X2MFA2
                                                                      X2MFA2
      CALL MSAB (A,NR2,NF,T,NF,NF,S,700,10,700)                       EXTEND
      WRITE (6,260) NR2,(II,II = 1,NF)                                X2MFA2
      DO 130 I = 1,NR2                                                X2MFA2
  130 WRITE (6,250) I,(S(I,J),J = 1,NF)                               X2MFA2
C                                                                     X2MFA2
C     ROTATE CORE MATRIX                                              X2MFA2
C                                                                     X2MFA2
  140 DO 150 I = 1,NF                                                 X2MFA2
      DO 150 J = 1,NF                                                 X2MFA2
  150 TT(I,J) = T(I,J)*R(J)                                           X2MFA2
      IF (LN.GE.1) GO TO 170                                          X2MFA2
      WRITE (6,270)                                                   X2MFA2
      WRITE (6,280) (II,II = 1,NF)                                    X2MFA2
      CALL MSABT (TT,NF,NF,T,NF,NF,A,10,10,700)                       EXTEND
      DO 160 I = 1,NF                                                 X2MFA2
```

```
  160 WRITE (6,250) I,(A(I,J),J = 1,NF)                                    X2MFA2
      GO TO 200                                                            X2MFA2
C                                                                          X2MFA2
C     ROTATE OTHER MODE                                                    X2MFA2
C                                                                          X2MFA2
  170 IF (MODE.NE.1) GO TO 180                                             X2MFA2
      NR = NR2                                                             X2MFA2
      MODE = 2                                                             X2MFA2
      GO TO 20                                                             X2MFA2
  180 WRITE (6,270)                                                        X2MFA2
      WRITE (6,290) (II,II = 1,NF)                                         X2MFA2
      CALL MSAB (TT,NF,NF,T,NF,NF,A,10,10,700)                             EXTEND
      DO 190 I = 1,NF                                                      X2MFA2
  190 WRITE (6,250) I,(A(I,J),J = 1,NF)                                    X2MFA2
  200 CONTINUE                                                             X2MFA2
C                                                                          X2MFA2
C     FORMAT STATEMENT                                                     X2MFA2
C                                                                          X2MFA2
  210 FORMAT (3I3,2I1)                                                     X2MFA2
  220 FORMAT (13A6,A2)                                                     X2MFA2
  230 FORMAT (1H1,11HVARIMAX FOR,I3,14H VARIABLE MODE//)                   X2MFA2
  240 FORMAT (1H-,16HTRANSFORM MATRIX//)                                   X2MFA2
  250 FORMAT (1X,I3,10F11.4)                                               X2MFA2
  260 FORMAT (1H1,19HCOUNTER ROTATION OF,I3,14H VARIABLE MODE//10X,I1,9I   X2MFA2
     111/)                                                                 X2MFA2
  270 FORMAT (1H-,34HROTATED CORE MATRIX G (NF1 BY NF2)//)                 X2MFA2
  280 FORMAT (1X,38HDATA=(F*)G(FJ) , WHERE FJ IS PA FACTOR//10X,I1,9I11/   X2MFA2
     1)                                                                    X2MFA2
  290 FORMAT (2X,89HDATA=(F1*)G(F2*),  (F1*) AND (F2*) ARE VARIMAX OF FI   X2MFA2
     1RST MODE AND SECOND MODE RESPECTIVELY//10X,I1,9I11//)                X2MFA2
      END                                                                  X2MFA2
```

```
      SUBROUTINE VARIMX(NROWS,NCOLS,NVPCH)                              VARIMX
      COMMON // A(700,10),R(10),H(700),FMT(14),V(10),S(700,10)          EXTEND
     1    , T(10,10) , TT(10,10)                                        X24-CCM
      REAL ROOT(10),VAR(10),CUMV(10)                                    VARIM2
C                                                                       VARIM2
C     VARIMAX FACTOR ROTATION                                           VARIM2
C                                                                       VARIM2
C     OUTPUT COMMUNALITIES AND NORMALIZE FACTORS                        VARIM2
C                                                                       VARIM2
      SUM = 0.0                                                         VARIM2
      DO 10 I = 1,NCOLS                                                 VARIM2
   10 SUM = SUM+V(I)                                                    VARIM2
      WRITE (6,260) NCOLS,SUM                                           VARIM2
      WRITE (6,270)                                                     VARIM2
      SUM = 0.0                                                         VARIM2
      DO 20 I = 1,NROWS                                                 VARIM2
      WRITE (6,280) I,H(I)                                              VARIM2
      SUM = SUM+H(I)                                                    VARIM2
   20 H(I) = SQRT(H(I))                                                 VARIM2
      WRITE (6,290) SUM                                                 VARIM2
      DO 30 J = 1,NCOLS                                                 VARIM2
      DO 30 I = 1,NROWS                                                 VARIM2
   30 A(I,J) = A(I,J)/H(I)                                              VARIM2
      KROT = (NCOLS*(NCOLS-1))/2                                        VARIM2
   40 KX = 1                                                            VARIM2
      KZCT = 0                                                          VARIM2
C                                                                       VARIM2
C     COMPUTE SUMS A,B,C AND D                                          VARIM2
C                                                                       VARIM2
   50 KY = KX+1                                                         VARIM2
   60 CONA = 0.                                                         VARIM2
      CONB = 0.                                                         VARIM2
      CONC = 0.                                                         VARIM2
      COND = 0.                                                         VARIM2
```

```
      DO 70 I = 1,NROWS
      U = (A(I,KX)+A(I,KY))*(A(I,KX)-A(I,KY))                         VARIM2
      V = 2.0*A(I,KX)*A(I,KY)                                          VARIM2
      CONA = CONA+U                                                    VARIM2
      CONB = CONB+V                                                    VARIM2
      CONC = CONC+(U+V)*(U-V)                                          VARIM2
   70 COND = COND+2.0*U*V                                              VARIM2
      FLN = FLOAT(NROWS)                                               VARIM2
      TNUM = COND-(2.0*CONA*CONB)/FLN                                  VARIM2
      TDEN = CONC-((CONA+CONB)*(CONA-CONB))/FLN                        VARIM2
      ANUM = ABS(TNUM)                                                 VARIM2
      ADEN = ABS(TDEN)                                                 VARIM2
      IF (ANUM-ADEN) 90,80,100                                         VARIM2
   80 IF (ANUM.EQ.0) GO TO 180                                         VARIM2
      GO TO 105                                                        VARIM2
C                                                                      VARIM2
C     NUM LESS THAN DEN                                                VARIM2
C                                                                      VARIM2
   90 TAN4T = ANUM/ADEN                                                VARIM2
      IF (TAN4T.LT.0.06993) GO TO 190                                  VARIM2
      COS4T = 1.0/SQRT(1.0+TAN4T**2)                                   VARIM2
      SIN4T = TAN4T*COS4T                                              VARIM2
      GO TO 110                                                        VARIM2
C                                                                      VARIM2
C     NUM GREATER THAN DEN                                             VARIM2
C                                                                      VARIM2
  100 CTN4T = ADEN/ANUM                                                VARIM2
      IF (CTN4T.LT.0.06993) GO TO 200                                  VARIM2
      SIN4T = 1.0/SQRT(1.0+CTN4T**2)                                   VARIM2
      COS4T = CTN4T*SIN4T                                              VARIM2
      GO TO 110                                                        VARIM2
C                                                                      VARIM2
C     NUM EQUAL DEN, BOTH NON-ZERO                                     VARIM2
C                                                                      VARIM2
                                                                       VARIM2
```

```
 105  SIN4T = 0.707107                                         VARIM2
      COS4T = SIN4T                                            VARIM2
C                                                              VARIM2
C     COMPUTE ANGLE OF ROTATION                                VARIM2
C                                                              VARIM2
  110 COS2T = SQRT(0.5+0.5*COS4T)                              VARIM2
      SIN2T = SIN4T/(2.0*COS2T)                                VARIM2
      COST = SQRT(0.5+0.5*COS2T)                               VARIM2
      SINT = SIN2T/(2.0*COST)                                  VARIM2
      IF (TDEN.LT.0) GO TO 120                                 VARIM2
      COSPHI = COST                                            VARIM2
      SINPHI = SINT                                            VARIM2
      GO TO 130                                                VARIM2
  120 COSPHI = 0.707107*(COST+SINT)                            VARIM2
      SINPHI = ABS( 0.707107*(COST-SINT) )                     VARIM2
  130 IF (TNUM.LT.0) SINPHI = -SINPHI                          VARIM2
C                                                              VARIM2
C     ROTATE FACTORS                                           VARIM2
C                                                              VARIM2
  140 DO 150 I = 1,NROWS                                       VARIM2
      U = A(I,KX)*COSPHI+A(I,KY)*SINPHI                        VARIM2
      A(I,KY) = -SINPHI*A(I,KX)+A(I,KY)*COSPHI                 VARIM2
  150 A(I,KX) = U                                              VARIM2
  160 IF (KY.EQ.NCOLS) GO TO 170                               VARIM2
      KY = KY+1                                                VARIM2
      GO TO 60                                                 VARIM2
  170 KX = KX+1                                                VARIM2
      IF (KX.NE.NCOLS) GO TO 50                                VARIM2
      IF (KZCT.NE.KRLT) GO TO 40                               VARIM2
      GO TO 205                                                VARIM2
C                                                              VARIM2
C     NUM AND DEN EQUAL ZERO                                   VARIM2
C                                                              VARIM2
```

```
  180 KZCT = KZCT+1                                                    VARIM2
      GO TO 160                                                        VARIM2
  190 IF (TDEN.GE.0) GO TO 180                                         VARIM2
      COSPHI = 0.707107                                                VARIM2
      SINPHI = COSPHI                                                  VARIM2
      GO TO 140                                                        VARIM2
C                                                                      VARIM2
C     DEN / NUM LESS THAN E                                            VARIM2
C                                                                      VARIM2
  200 COS4T = 0.0                                                      VARIM2
      SIN4T = 1.0                                                      VARIM2
      GO TO 110                                                        VARIM2
C                                                                      VARIM2
C     OUTPUT FACTORS                                                   VARIM2
C                                                                      VARIM2
C                                                                      VARIM2
  205 DO 220 J = 1,NCOLS                                               VARIM2
      ROOT(J) = 0.0                                                    VARIM2
      DO 210 I = 1,NROWS                                               VARIM2
      A(I,J) = A(I,J)*H(I)                                             VARIM2
  210 ROOT(J) = ROOT(J)+A(I,J)**2                                      VARIM2
  220 VAR(J) = ROOT(J)/FLN                                             VARIM2
      WRITE (6,300) (I,I = 1,NCOLS)                                    VARIM2
      DO 230 I = 1,NROWS                                               VARIM2
  230 WRITE (6,310) I,(A(I,J),J = 1,NCOLS)                             VARIM2
      WRITE (6,320) (ROOT(J),J = 1,NCOLS)                              VARIM2
      WRITE (6,330) (VAR(J),J = 1,NCOLS)                               VARIM2
      CUMV(1) = VAR(1)                                                 VARIM2
      DO 240 I = 2,NCOLS                                               VARIM2
  240 CUMV(I) = VAR(I)+CUMV(I-1)                                       VARIM2
      WRITE (6,340) (CUMV(J),J = 1,NCOLS)                              VARIM2
      IF (NVPCH.LE.0) RETURN                                           VARIM2
      DO 250 I = 1,NROWS                                               VARIM2
  250 PUNCH 350, I,(A(I,J),J=1,NCOLS)                                  VARIM2
      RETURN                                                           VARIM2
```

```
C                                                                       VARIM2
C     FORMAT STATEMENTS                                                 VARIM2
C                                                                       VARIM2
  260 FORMAT (21H-THE SUM OF THE FIRST,I3,9H ROOTS IS,F14.4///)         VARIM2
  270 FORMAT (26H0   NO.        COMMUNALITIES,//)                       VARIM2
  280 FORMAT (1H ,I6,F14.4)                                             VARIM2
  290 FORMAT (28H0THE SUM OF COMMUNALITIES IS,F14.4)                    VARIM2
  300 FORMAT(*1 VAR*,13X,*VARIMAX FACTOR LOADINGS*/3X,*NO.*,I8,9I11)    VARIM2
  310 FORMAT (1X,I5,3X,10F11.3)                                         VARIM2
  320 FORMAT (1H-,8HVARIANCE,10F11.3/)                                  VARIM2
  330 FORMAT (1X,8HPCT VAR ,10F11.3/)                                   VARIM2
  340 FORMAT (1X,8HCUM PCT ,10F11.3)                                    VARIM2
  350 FORMAT(I7,9F8.5/(10F8.5))                                         VARIM2
      END                                                               VARIM2


      SUBROUTINE ABORT(MSG)                                             ABORT
      INTEGER MSG(1)                                                    ABORT
      CALL SYSTEM(52,12H USER ABORT.)                                   ABORT
      STOP                                                              ABORT
      ENTRY ABORTM                                                      ABORT
      CALL SYSTEM(52,MSG)                                               ABORT
      END                                                               ABORT

      SUBROUTINE IPRNT ( NAME )

      PRINT 1,NAME
      FORMAT(*0  *,A7,* - YOU CAN NOT MULTIPLY THESE MATRICES*)
      CALL EXIT

      END
```

```
      SUBROUTINE MSAB(A,NRA,NCA,B,NRB,NCB,C,NNRA,NNRB,NNRC)

      REAL A(NNRA,1), B(NNRB,1), C(NNRC,1)

C          MSAB - C = A*B

C          S.I. JASIK - 3/10/71 - DO 2 ROWS AT A TIME

           IF( NCA .NE. NRB ) CALL IPRNT( 4HMSAB )


           IF( NRA .EQ. 1 ) GO TO 25
           N2 = 2*(NRA/2)

      DO 20 J = 1,NCB
      DO 20 I = 1,N2,2

      S1 = 0.    $    S2 = 0.
      DO 10 K = 1,NCA
      S1 = S1 + A(I,K)*B(K,J)
   10 S2 = S2 + A(I+1,K)*B(K,J)
      C(I,J) = S1
   20 C(I+1,J) = S2

      IF( NRA .EQ. N2 ) RETURN

   25 DO 40 J = 1,NCB
      S1 = 0.
      DO 30 K = 1,NCA
   30 S1 = S1 + A(NRA,K)*B(K,J)
   40 C(NRA,J) = S1
      RETURN

      END
```

```
      SUBROUTINE MSATB (A,NRA,NCA,B,NRB,NCB,C,NNRA,NNRB,NNRC)

      REAL A(NNRA,NCA) , B(NNRB,NCB) , C(NNRC,NCB)

*     MSATB - C = TRANSPOSE(A)*B

      IF(NRA.EQ.NRB) GO TO 2
      CALL IPRNT( 5HMSATB )

 2    DO 40 I = 1,NCA
      DO 40 J=1,NCB
        S = 0
        DO 30 K = 1,NRA
 30     S = S+A(K,I)*B(K,J)
 40     C(I,J) = S

      RETURN

      END

      SUBROUTINE MSABT (A,NRA,NCA,B,NRB,NCB,C,NNRA,NNRB,NNRC)

      REAL A(NNRA,NCA) , B(NNRB,NCB) , C(NNRC,NCB)

      IF( NCA .EQ. NCB ) GO TO 6
      CALL IPRNT( 5HMSABT )

 6    DO 60 I = 1,NRA
      DO 60 J=1,NRB
        S = 0
        DO 50 K = 1,NCA
 50     S = S+A(I,K)*B(J,K)
 60   C(I,J) = S

      RETURN

      END
```

APPENDIX F

THE HSMS "MATRIX" PROGRAM

The HSMS MATRIX program was designed by Edward Friedman to make it possible to create printed displays of selected task data. These are used to aid in the assignment of tasks to job levels and to design curricula.

DESCRIPTION

MATRIX's main function is to array the scale values for skills and knowledges as rows, in the selected order in which they appear in tasks (which are the columns). This permits inspection of "profiles" of task requirements and comparison of task requirements.

In the array the tasks are the columns, listed by Task Code Number. The skill and knowledge categories, listed by their 8-character or 8-digit code identifications are the rows. The tasks are listed from right to left in the order designated by *C UPDATE cards. The skill and knowledge categories are listed in the order in which they appear in the tasks in the array.

The entries in the matrix are the non-zero scale values of the variables required by the tasks. Zero values are not shown, and the decimal points are omitted. Possible HSMS entries are 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, and 90. The arrays can be ordered in expanded or compressed form and in multiple copies for various uses.

The MATRIX program also provides data "dictionaries." One lists the tasks in their order in the array by internal number and by Task Code Number along with an abbreviated task name. Another lists the skill and knowledge categories with the skills first, followed by the knowledge category 8-digit codes in ascending numerical order. It also shows the frequency across the tasks in the display.

MATRIX is written in FORTRAN IV and was used in the Control Data Corporation's (CDC) 6600 computer at the Courant Institute of Mathematical Sciences of New York University. The operating systems in use during HSMS analyses were KRONOS and NOS (Network Operating System). MATRIX was stored on magnetic tape in compiled and loaded form (i.e., in binary object code in non-relocatable form) and in OLDPL form. The HSMS task data are transferred to magnetic tape in the form of an OLDPL. HSMS maintained the decks in numerical order in the OLDPL.

A utility program, UPDATE, is used to generate temporary local files in the form of a *C file which is part of the input file for MATRIX. The input file for UPDATE is magnetic tape in OLDPL form, and the output file is K=TAPE2. The input file for MATRIX is TAPE2, and the output file is printed.

SYSTEM SUBROUTINES CALLED BY MATRIX

The MATRIX program presented in this appendix calls a series of subroutines. Among these, EXIT, TIME, and DATE have not been included in the listing, since they are systems-based. It is assumed that the user can utilize comparable routines after reading the descriptions which follow.

EXIT

This subroutine terminates program execution and returns control to the operating system. A STOP statement may be preferable.

TIME(a)

This subroutine can be used as a function or subroutine. The value is returned via the argument and the normal function return. The subroutine returns the current reading of the system clock as the value of the argument a or of the function in the form 10Hbhh.mm.ss.b, where b denotes a blank, and hh, mm, and ss are the numbers of hours, minutes, and seconds, respectively. The value returned is Hollerith data and can be output using an A format specification. The type of this function is real.

DATE(a)

This subroutine can be used as a function subroutine. See CALL TIME(a), above. The current date is returned as the value of argument a or of the function in the form 10Hbmm/dd/yyb (unless it is changed at installation option), where b denotes a blank, mm is the number of the month, dd is the number of the day within the month, and yy is the year. The value returned is Hollerith data and can be output using an A format specification. The type of this function is real.

STRUCTURING THE INPUT FILE FOR MATRIX

In the set-up presented below, the NOS stem in use in 1977 at the Courant Institute's CDC 6600 is assumed. It is also assumed that the reader can refer to the UPDATE Reference Manual or a counterpart program.

INPUT FILE FOR MATRIX

Name	Keypunch Cards (one per box)	Instructions	
Systems Cards	_ _ _ _ .MATRIX	Identification number and other information depending on system.	
	USER(_ _ _ _ _ _ _ _)	USER and user code.	
	CHARGE(_ _ _ _ _ _ _ _ _ _ _ _ _ _)	Charge card.	
	RESOURC(MT=2)	Only if more than one set of task data for MATRIX array is being run.	
	HEADER.MATRIX_ _ _ _ _ _ _ _ etc.	Optional to print out heading across a whole page.	
	LABEL(OLDPL,VSN=T_ _ _)	Identifies OLDPL tape.	
	UPDATE(Q,D,8,K=TAPE2)	For selecting tasks using *C form of UPDATE.	
	UNLOAD(OLDPL)	Only if just one set of task data is being run.	
	LABEL(PROGRMS,VSN=T_ _ _)	Identifies program tape.	
	SKIPF(PROGRMS,4)	Skips over EDIT, PCVARIM, X2MOFA, and X2MFA2.	
	COPYBF(PROGRMS,MATRIX)	Copies program to local file.	
	UNLOAD(PROGRMS)	Only if just one set of task data is being run.	
	RFL(200000)	Defines field length.	
	SETTL(1000)	Time limit.	
	MATRIX.		
	End of Record Card	Only if just one set of task data is being run.	
	REWIND(OLDPL,TAPE2,MATRIX)	With second set of task data.	*
	UPDATE(Q,D,8,K=TAPE2)	With second set of task data.	*
	MATRIX.	With second set of task data.	*
		*These three cards are repeated again in this order as many times as there are sets of data beyond two.	
	End of Record Card	After last systems card.	

INPUT FILE FOR MATRIX (continued)

Name	Keypunch Cards (one per box)	Instructions
Input Cards For UP DATE	*C,_ _ _,_ _ _,_ _ _ and/or *C,_ _ _ *C,_ _ _ etc.	Calls the first set of tasks to be placed in the first array <u>in the order selected</u>; i.e., HSMS uses point score order, Task Code Number order, or factor loading order. Col. 1 is always asterisk; Col. 2 is always C; Col. 3 is always comma; then follows one or more Task Code Numbers separated by commas, with no comma after the last entry, up to Col. 80. It is convenient to place tasks on cards so that they can be rearranged for various arrays.
	End of Record Card	
Input Cards for MA TRIX	MATRIX_ _ _ _ _ _ _ _ etc. _ _ _ _ _ _ _ _ _ _ _ _ etc.	Title card. Cols. 1-40 read as one line; Cols. 41-80 read as second line. Use to identify the salient features of this first array of task data.
Parameter Card		Right justify in fields except as otherwise indicated.
	Cols. 3-5 _ _ _	Number of tasks in this set; up to 500.
	Col. 10 _	Punch 1 for long, expanded array. Punch 2 for short array. Punch 3 for condensed array (best for analysis). Punch 4 for all of the above.
	Col. 15 _	Number of copies of output.
	Cols. 16-20 _ _ _ _ _	Leave blank to call input data from UPDATE TAPE2 or enter name of alternate file. Left justify.
	End of Record Card	
Repeat the set-up on this page for second set of tasks, and repeat again for any additional sets of task data, one set-up per set.		
	End of file card comes last of all.	

OPT=1 FTN 4.6+428 77/09/15

MATRIX

```
      OVERLAY(MATRIX,0,0)                                                KRONOS
      PROGRAM MATRIX(TAPE1,TAPE2,INPUT,OUTPUT,TAPE5=INPUT,TAPE6,         MATRIX
     + TAPE7=OUTPUT)                                                     MATRIX
                                                                         MATRIX
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                    COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                               COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                            EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP              COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500), EXTEND
     + FREQ(500),PTR(500)                                                EXTEND
      COMMON MAT(40000)                                                  EXTEND
                                                                         MATRIX
      CALL DATE (TODAY)                                                  MATRIX
      CALL TIME(TIM)                                                     MATRIX
    5 CALL INPUT                                                         MATRIX
      CALL HEAD                                                          MATRIX
                                                                         MATRIX
      ISRT=0,  TASK DATA SORTED AS DESIRED AND IS ON UNIT 2              MATRIX
      ISRT≠0,  DATA MUST BE EXTRACTED FROM MASTER FILE ON UNIT 1         MATRIX
               AND WILL BE WRITTEN ON UNIT 2.                            MATRIX
                                                                         MATRIX
      IF(ISRT.NE.0) CALL SELECT                                          MATRIX
      CALL GET                                                           MATRIX
      CALL FORM                                                          MATRIX
      CALL OUT                                                           MATRIX
      GO TO 5                                                            MATRIX
      END                                                                MATRIX
```

```
      SUBROUTINE INPUT                                                  INPUT
      INTEGER TASK,STABLE,A,TID,TABLE,CHCICE,FREQ,PTR                   COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                              COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                           EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP             COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500), EXTEND
     + FREQ(500),PTR(500)                                               EXTEND
      COMMON MAT(40000)                                                 EXTEND
C     INITIALIZE MAT WITH BLANKS                                        INPUT
      CALL SETRAY(MAT,40000,1H )                                        EXTEND
      NPG=0                                                             INPUT
C                                                                       INPUT
C     READ IN TITLE CARD                                                INPUT
C                                                                       INPUT
    5 READ(5,500)TITLE                                                  INPUT
      IF(EOF(5))10,20                                                   INPUT
   10 CALL EXIT                                                         INPUT
C                                                                       INPUT
C     READ IN PARAMETER CARD                                            INPUT
C                                                                       INPUT
   20 READ(5,510)NTASK,CHOICE,NCOP,ISRT                                 INPUT
      N=NTASK                                                           INPUT
      IF(ISRT.EQ.0)   GO TO 30                                          INPUT
C     READ IN TASK NOS.                                                 INPUT
C     IF DATA IS TO BE EXTRACTED FROM MASTER FILE ON UNIT 1             INPUT
      READ(5,520)(TASK(I),I=1,N)                                        INPUT
   30 RETURN                                                            INPUT
C     INPUT FORMAT STATEMENTS                                           INPUT
  500 FORMAT(8A10)                                                      INPUT
  510 FORMAT(16I5)                                                      INPUT
  520 FORMAT(5X15I5)                                                    INPUT
      END                                                               INPUT
```

```
      SUBROUTINE SELECT                                                  SELECT
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                    COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                               COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                            EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCCP              COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500), EXTEND
     + FREQ(500),PTR(500)                                                EXTEND
      COMMON MAI(40000)                                                  EXTEND
C SELECT ONLY THOSE TASK CARDS DESIRED                                   SELECT
C IN ORDER SPECIFIED ON INPUT TASK(S) CARDS                              SELECT
      DO  100  I=1,N                                                     SELECT
      REWIND 1                                                           SELECT
   25 READ(1,1000) TID,NCD,A                                             SELECT
      IF(EOF(1))30,40                                                    SELECT
C  DATA EXHAUSTED AND TASK  NOT  FOUND                                   SELECT
   30 CALL ERROR(1)                                                      SELECT
   40 IF(TASK(I).EQ.TID) GO TO 60                                        SELECT
C  SKIP ADDITIONAL CARDS                                                 SELECT
      DO 50  J=1,NCD                                                     SELECT
      READ(1,1010)                                                       SELECT
   50 CONTINUE                                                           SELECT
      GO TO 25                                                           SELECT
C WRITE SELECTED DATA ONTO UNIT 2                                        SELECT
   60 WRITE(2,1020)A                                                     SELECT
      DO  70  J=1,NCD                                                    SELECT
      READ (1,1020) A                                                    SELECT
      WRITE(2,1020) A                                                    SELECT
   70 CONTINUE                                                           SELECT
  100 CONTINUE                                                           SELECT
C                                                                        SELECT
C     UNIT 2 CONTAINS DATA IN ORDER REQUIRED                             SELECT
C                                                                        SELECT
```

```
      ENDFILE 2
      REWIND 2                                                          SELECT
      RETURN                                                            SELECT
C     FORMAT CARDS FOR TASK DATA                                        SELECT
 1000 FORMAT(T5,I6,T29,I2,T1,8A10)                                      SELECT
 1010 FORMAT(1X)                                                        SELECT
 1020 FORMAT(8A10)                                                      SELECT
      END                                                               SELECT
                                                                        SELECT

      SUBROUTINE GET                                                    GET
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                   COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                              COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                           EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP             COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500), EXTEND
     + FREQ(500),PTR(500)                                               EXTEND
      COMMON MAT(40000)                                                 EXTEND
      INTEGER BCD(5)                                                    GET
      REWIND 2                                                          GET
C GET DATA FROM UNIT 2 AND FORM MATRIX MAT                              GET
  110 DO  200  I=1,N                                                    GET
      READ(2,2000)TID,NCD,BCD,TASK(I)                                   GET
      IF(EOF(2).EQ.0)GO TO 115                                          EXTEND
      WRITE(7,1)                                                        EXTEND
    1 FORMAT(*INPUT ERROR: NO. O  TASKS ON TAPE LESS THAN NO. OF TASKS O EXTEND
     +N PARAMETER CARD*)                                                EXTEND
      CALL EXIT                                                         EXTEND
  115 CONTINUE                                                          EXTEND
      IF((MOD(I,25)-1).NE.0) GO TO 120                                  GET
      CALL TOP                                                          GET
      WRITE(6,6000)                                                     GET
```

```
  120 WRITE(6,6100)I,TID,BCO                                                GET
      K=1                                                                   GET
      L=NSV                                                                 GET
      READ(2,2010)IT,(KV(J),J=K,L)                                          GET
      IF(IT.NE.3RT01) STOP                                                  GET
      IF(NCD.EQ.1) GD TO 180                                                GET
C     ONLY CARD TOC AND T01 EXIST                                           GET
C     OBTAIN KC INFO                                                        GET
  130 DO  150  II=2,NCD                                                     GET
      K=L+1                                                                 GET
      L=L+4                                                                 GET
      READ(2,2020)(KC(J),KV(J),J=K,L)                                       GET
  150 CONTINUE                                                              GET
  180 NKC=L                                                                 GET
  190 CALL UPDATE(I)                                                        GET
  200 CONTINUE                                                              GET
      CALL TOP                                                              GET
      WRITE(6,6200)   N,NR                                                  GET
      RETURN                                                                GET
 2000 FORMAT(T5,R6,T29,I2,T32,4A10,A7,T8,I3)                                GET
 2010 FORMAT(T1,R3,T31,16(1X,R2))                                           GET
 2020 FORMAT(T28,4(2XR8,1XR2))                                              GET
 6000 FORMAT(/10X*I*,7X*TASK NO.*,5X*TASK IDENTIFICATION*,                  GET
     X /1X,8(10H----------)/)                                               GET
 6100 FORMAT(/ 6XI5,9XR6,3X4A10,A7)                                         GET
 6200 FORMAT(1H0,5X*NO. TASKS = *,I6,5X*NO. OF CATEGORIES = *,I6)           GET
       END                                                                  GET
```

```
      SUBROUTINE UPDATE(J)                                                UPDATE
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                     COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                                COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                             EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCUP               COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500),  EXTEND
     + FREQ(500),PTR(500)                                                 EXTEND
      COMMON MAT(40000)                                                   EXTEND
      INTEGER SABLE(500),NAT(500)                                         EXTEND
      INTEGER DZ,B8                                                       UPDATE
      DATA DZ/2R00/                                                       UPDATE
      DATA B8/8R        /                                                 UPDATE
      IF(J.NE.1) GO TO 150                                                UPDATE
C                                                                         UPDATE
C     INITIALIZATION ENTRY FOR FIRST TASK ONLY                            UPDATE
C                                                                         UPDATE
      CALL ZERO(FREQ,NTASK)                                               UPDATE
      ISV=0                                                               UPDATE
      IKC=0                                                               UPDATE
      DO 100  I=1,NKC                                                     UPDATE
      IF(I.LE.NSV) GO TO 90                                               UPDATE
      IF(KC(I).EQ.B8) GO TO 100                                           UPDATE
      IF(KV(I).EQ.DZ) KV(I)=2R                                            UPDATE
      IKC=IKC+1                                                           UPDATE
      SABLE(IKC)=KC(I)                                                    UPDATE
      NAT(IKC)=KV(I)                                                      UPDATE
      GO TO 100                                                           UPDATE
C  SKILL CATEGORY AND VALUE                                               UPDATE
   90 IF(KV(I).EQ.DZ) GO TO 100                                           UPDATE
      ISV=ISV+1                                                           UPDATE
      TABLE(ISV)=STABLE(I)                                                UPDATE
      FREQ(ISV)=1                                                         UPDATE
      MAT(ISV)=KV(I)                                                      UPDATE
```

```
  100 CONTINUE                                                           UPDATE
      K=ISV+IKC                                                          UPDATE
      LC(J)=K                                                            UPDATE
      NR=K                                                               UPDATE
      KT=K                                                               UPDATE
C SORT IN ASCENDING ORDER KNOWLEDGE CATEGORIES FOR FIRST TASK             UPDATE
      IF(IKC.EQ.0 ) GO TO 120                                            UPDATE
      PTR(1)=1                                                           UPDATE
      IF(IKC.EQ.1) GO TO 105                                             UPDATE
      CALL SHLSRT(SABLE,IKC,PTR)                                         UPDATE
  105 DO 110 I=1,IKC                                                     UPDATE
      TABLE(I+ISV)=SABLE(PTR(I))                                         UPDATE
      MAT(I+ISV)=NAT(PTR(I))                                             UPDATE
      FREQ(I+ISV)=1                                                      UPDATE
  110 CONTINUE                                                           UPDATE
  120 RETURN                                                             UPDATE
C                                                                        UPDATE
C ENTRY FOR ALL       TASKS EXCEPT THE FIRST                             UPDATE
C                                                                        UPDATE
  150 KO=NR                                                              UPDATE
      M=0                                                                UPDATE
      K=KT+NR                                                            UPDATE
C                                                                        UPDATE
      DO 200  I=1,NKC                                                    UPDATE
      IF((I.LE.NSV).AND.(KV(I).EQ.DZ))   GO TO 200                       UPDATE
      IF(I.LE. NSV) KC(I)=STABLE(I)                                      UPDATE
      IF(KC(I).EQ.B8          ) GO TO 202                                UPDATE
      IF(KV(I).EQ.DZ) KV(I)=ZR                                           UPDATE
C                                                                        UPDATE
      DO 160  L=1,KO                                                     UPDATE
      IF(KC(I).EQ.TABLE(L)) GO TO 170                                    UPDATE
  160 CONTINUE                                                           UPDATE
C                                                                        UPDATE
```

```
C  NO ENTRY EXISTED IN TABLE BEFORE ADD ONE
      M=M+1                                                           UPDATE
      SABLE(M )=KC(I)                                                 UPDATE
      NAT(M )=KV(I)                                                   UPDATE
      GO TO 200                                                       UPDATE
C                                                                     UPDATE
C  ENTRY EXISTED IN TABLE BEFORE UPDATE APPROPIATE ROW FOR THIS TASK  UPDATE
C                                                                     UPDATE
  170 CONTINUE                                                        UPDATE
      KTPL = KT + L                                                   EXTEND
      IF (KTPL.GE.40000)  GO TO 220                                   EXTEND
      MAT(KTPL) = KV(I)                                               EXTEND
      FREQ(L)=FREQ(L)+1                                               EXTEND
  200 CONTINUE                                                        UPDATE
C                                                                     UPDATE
  202 IF(M.EQ.0) GO TO 210                                            UPDATE
      PTR(1)=1                                                        UPDATE
      IF(M.EQ.1) GO TO 204                                            UPDATE
C                                                                     UPDATE
C SORT IN ASCENDING ORDER NEW CATEGORIES FOR THIS TASK                UPDATE
      CALL SHLSRT(SABLE,M,PTR)                                        UPDATE
C                                                                     UPDATE
  204 DO 205 I=1,M                                                    UPDATE
      NR=NR+1                                                         UPDATE
      TABLE(NR  )= SABLE(PTR(I))                                      UPDATE
      FREQ(NR)=1                                                      UPDATE
      K=K+1                                                           UPDATE
      MAT(K)=NAT(PTR(I))                                              UPDATE
  205 CONTINUE                                                        UPDATE
  210 LC(J)=NR                                                        UPDATE
      KT=K                                                            UPDATE
      RETURN                                                          UPDATE
  220 WRITE(7,230) TID,J                                              UPDATE
                          CALL ABORT(10HOVERFLOW      )               EXTEND
  230 FORMAT(5X* LAST TASK READ *,R6, * OF  *,I10,*TASKS READ*/)      EXTEND
      END                                                             EXTEND
                                                                      UPDATE
```

```
      SUBROUTINE FORM                                                    FORM
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                    COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                               COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                            EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP              COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500), EXTEND
     + FREQ(500),PTR(500)                                                EXTEND
      COMMON MAT(40000)                                                  EXTEND
      LCS(1)=1                                                           FORM
      NALL=NTASK+1                                                       FORM
      DO 100  I=2,NALL                                                   FORM
      LCS(I)=LCS(I-1)+LC(I-1)                                            FORM
  100 CONTINUE                                                           FORM
  150 RETURN                                                             FORM
      END                                                                FORM


      SUBROUTINE OUT                                                     OUT
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                    COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                               COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                            EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP              COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500), EXTEND
     + FREQ(500),PTR(500)                                                EXTEND
      COMMON MAT(40000)                                                  EXTEND
      INTEGER LINE(14)                                                   OUT
      DO 10 I=1,NTASK                                                    OUT
      ENCODE(3,6010,TASK(I)) TASK(I)                                     OUT
      DECODE(3,6020,TASK(I)) TASK(I)                                     OUT
```

```
   10 CONTINUE
      IF((CHOICE.LE.0).O.(CHOICE.GT.4))GO TO 300
      GO TO(100,200,300,400) CHOICE
  100 CALL MATOUT
      GO TO 900
  200 CALL MATSHT
      GO TO 900
  300 CALL MATCND
      GO TO 900
  400 CALL MATOUT
      CALL MATSHT
      CALL MATCND
C     PRINT FREQUENCIES FOR SKILL SCALES AND KNOWLEDGE CATEGORIES
C     SORTED IN ASCENDING ORDER .
  900 CALL KCFREQ
      CALL TIME(TIM)
      CALL HEAD
      ENDFILE 6
      DO 1000 ICOP=1,NCOP
      REWIND 6
  950 READ(6,6000) LINE
      IF(EOF(6).NE.0) GO TO 1000
      WRITE(7,6000)LINE
      GO TO 950
 1000 CONTINUE
      REWIND 6
      RETURN
C
 6000 FORMAT(13A10,A6)
 6010 FORMAT(I3)
 6020 FORMAT(R3)
      END
```

```
      SUBROUTINE MATOUT                                                      MATOUT
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                        COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                                   COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                                EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP                  COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500),     EXTEND
     + FREQ(500),PTR(500)                                                    EXTEND
      COMMON MAT(40000)                                                      EXTEND
      INTEGER BLANK(25),LINE(25)                                             MATOUT
      M=25                                                                   MATOUT
      CALL SETRAY(BLANK,M,3R    )                                            MATOUT
      DO 1000 K=1,NTASK,25                                                   MATOUT
      M=MIN0(M,NTASK)                                                        MATOUT
      MMK=24-(M-K)                                                           MATOUT
      DO 900  L=1,NK                                                         MATOUT
      IF((MOD(L,25)-1).NE.0) GO TO 100                                       MATOUT
      CALL TOP                                                               MATOUT
      IF(MMK.EQ.0) GO TO 60                                                  MATOUT
      WRITE(6,6000) (TASK(I),I=K,M),(BLANK(J) ,J=1,MMK)                      MATOUT
      GO TO 70                                                               MATOUT
   60 WRITE(6,6000)(TASK(I),I=K,M)                                           MATOUT
   70 WRITE(6,6200)                                                          MATOUT
  100 CALL SETRAY(LINE,25,2R  )                                              MATOUT
      DO 150 I=K,M                                                           MATOUT
      II=I-K+1                                                               MATOUT
  120 J=LCS(I)+L-1                                                           MATOUT
  130 IF(J.GE.LCS(I).AND.J.LT.LCS(I+1)) LINE(II)=MAT(J)                      MATOUT
  150 CONTINUE                                                               MATOUT
  200 WRITE(6,6100)TABLE(L), LINE,TABLE(L)                                   MATOUT
  300 WRITE(6,6200)                                                          MATOUT
  900 CONTINUE                                                               MATOUT
      M=M+25                                                                 MATOUT
```

```
 1000 CONTINUE
      RETURN                                                        MATOUT
C FORMAT STATEMENTS                                                 MATOUT
 6000 FORMAT(1X*CATEGORY/*25(R3*/*),2X*CATEGORY*)                   MATOUT
 6100 FORMAT(1XR8,* *,25(1XR2* *),2XR8)                             MATOUT
 6200 FORMAT(1H ,13(10H----------))                                 MATOUT
      END                                                           MATOUT
                                                                    MATOUT


      SUBROUTINE MATSHT
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR               MATSHT
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                          COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                       COMMON
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP         EXTEND
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500), COMMON
     + FREQ(500),PTR(500)                                           EXTEND
      COMMON MAT(40000)                                             EXTEND
      INTEGER BLANK(25),LINE(25)                                    EXTEND
      M=25                                                          MATSHT
      CALL SETRAY(BLANK,M,3R   )                                    MATSHT
      DO 1000 K=1,NTASK,25                                          MATSHT
      M=MIN0(M,NTASK)                                               MATSHT
      MMK=24-(M-K)                                                  MATSHT
      DO 900  L=1,NR                                                MATSHT
      IF((MOD(L,50)-1).NE.0) GO TO 100                              MATSHT
      CALL TOP                                                      MATSHT
      IF(MMK.EQ.0) GO TO 60                                         MATSHT
      WRITE(6,6000) (TASK(I),I=K,M),(BLANK(J) ,J=1,MMK)             MATSHT
      GO TO 70                                                      MATSHT
   60 WRITE(6,6000)(TASK(I),I=K,M)                                  MATSHT
   70 WRITE(6,6200)                                                 MATSHT
                                                                    MATSHT
```

```
  100 CALL SETRAY(LINE,25,2R   )                                          MATSHT
      DO 150 I=K,M                                                        MATSHT
      II=I-K+1                                                            MATSHT
  120 J=LCS(I)+L-1                                                        MATSHT
  130 IF(J.GE.LCS(I).AND.J.LT.LCS(I+1)) LINE(II)=MAT(J)                   MATSHT
  150 CONTINUE                                                            MATSHT
  200 WRITE(6,6100)TABLE(L), LINE,TABLE(L)                                MATSHT
  800 CONTINUE                                                            MATSHT
  900 CONTINUE                                                            MATSHT
      M=M+25                                                              MATSHT
 1000 CONTINUE                                                            MATSHT
      RETURN                                                              MATSHT
C                                                                         MATSHT
C FORMAT STATEMENTS                                                       MATSHT
C                                                                         MATSHT
 6000 FORMAT(1X*CATEGORY/*25(R3*/*),2X*CATEGORY*)                         MATSHT
 6100 FORMAT(1XR8,* *,25(1XR2* *),2XR8)                                   MATSHT
 6200 FORMAT(1H ,13(10H----------))                                       MATSHT
      END                                                                 MATSHT
```

```
      SUBROUTINE MATCND                                                 MATCND
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                   COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                              COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                           EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP             COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500), EXTEND
     + FREQ(500),PTR(500)                                               EXTEND
      COMMON MAT(40000)                                                 XTEND
      INTEGER BLANK(42),LINE(42),KOOL(3,200)                            MATCND
      CALL ZERO(KOOL,600)                                               MATCND
      M=42                                                              MATCND
      MM=42                                                             MATCND
      CALL SETRAY(BLANK,MM,3R   )                                       MATCND
      DO 50 K=1,NTASK                                                   MATCND
      DO 50 L=1,3                                                       MATCND
      LL=42+L*6                                                         MATCND
      KOOL(L,K)=((.N.MASK(54)).A.SHIFT(TASK(K),LL)).O.KOOL(L,K)         MATCND
   50 CONTINUE                                                          MATCND
      DO 1000 K=1,NTASK,MM                                              MATCND
      M=MIN0(M,NTASK)                                                   MATCND
      MMK=(MM-1)-(M-K)                                                  MATCND
      DO 900  L=1,NR                                                    MATCND
      IF(L.NE.1) GO TO 100                                              MATCND
      CALL MOP                                                          MATCND
      IF(MMK.EQ.0) GO TO 60                                             MATCND
      WRITE(6,6000)(((KOOL(LL,I),I=K,M),(BLANK(J) ,J=1,MMK)),LL=1,3)    MATCND
      GO TO 70                                                          MATCND
   60 WRITE(6,6000) ((KOOL(LL,I),I=K,M) ,LL=1,3)                        MATCND
   70 WRITE(6,6200)                                                     MATCND
  100 CALL SETRAY(LINE,MM,2R   )                                        MATCND
      DO 150 I=K,M                                                      MATCND
      II=I-K+1                                                          MATCND
```

```
  120 J=LCS(I)+L-1                                                    MATCND
  130 IF(J.GE.LCS(I).AND.J.LT.LCS(I+1)) LINE(1I)=MAT(J)               MATCND
  150 CONTINUE                                                        MATCND
  200 WRITE(6,6100) LINE,TABLE(L)                                     MATCND
  800 CONTINUE                                                        MATCND
  900 CONTINUE                                                        MATCND
      M=M+MM                                                          MATCND
C                                                                     MATCND
C GET OUT OF GRAPH MODE                                               MATCND
C                                                                     MATCND
      WRITE(6,6400)                                                   MATCND
 1000 CONTINUE                                                        MATCND
      RETURN                                                          MATCND
C                                                                     MATCND
C FORMAT STATEMENTS                                                   MATCND
C                                                                     MATCND
 6000 FORMAT(1X42(* *,R1,*+*),1X* SKILL   *,/                         MATCND
     X        1X42(* *,R1,*+*),1X*   CR    *,/                        MATCND
     X        1X42(* *,R1,*+*),1X*CATEGORY*)                          MATCND
 6100 FORMAT(1X42(1XR2),1XR8)                                         MATCND
 6200 FORMAT(1H ,13(10H----------),*------*)                          MATCND
 6300 FORMAT(*_*)                                                     MATCND
 6400 FORMAT(*R*)                                                     MATCND
      END                                                             MATCND
```

```
      SUBROUTINE KCFREQ                                                         KCFREQ
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                           COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                                      COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                                   EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP                     COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500),        EXTEND
     + FREQ(500),PTR(500)                                                       EXTEND
      COMMON MAT(40000)                                                         EXTEND
      CALL SHLSRT(TABLE,NR,PTR)                                                 KCFREQ
      DO 100 L=1,NR                                                             KCFREQ
      IF(MOD(L,50)-1.NE.0) GO TO 50                                             KCFREQ
      CALL TOP                                                                  KCFREQ
      WRITE(6,6000)                                                             KCFREQ
      WRITE(6,6010)                                                             KCFREQ
   50 WRITE(6,6100) L,TABLE(PTR(L)),FREQ(PTR(L))                                KCFREQ
  100 CONTINUE                                                                  KCFREQ
 6000 FORMAT(10X*SORT BY SKILL SCALE AND/OR KNOWLEDGE CATEGORY*,/)              KCFREQ
 6010 FORMAT(5X*L*, 8X*CATEGORY*,10X*FREQUENCY*,/,                              KCFREQ
     X5X*-*    ,8X*--------*,10X*---------*,/)                                  KCFREQ
 6100 FORMAT(1XI5,8XR8,14XI5)                                                   KCFREQ
      RETURN                                                                    KCFREQ
      END                                                                       KCFREQ
```

```
      SUBROUTINE HEAD                                                     HEAD
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                     COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                                COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                             EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP               COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500),  EXTEND
     + FREQ(500),PTR(500)                                                 EXTEND
      COMMON MAT(40000)                                                   EXTEND
      WRITE(6,6000)                                                       HEAD
      WRITE(6,6100)                                                       HEAD
      WRITE(6,6200) TITLE,TODAY,TIM                                       HEAD
      WRITE(6,6100)                                                       HEAD
      RETURN                                                              HEAD
 6000 FORMAT(1H1,10(/1H0))                                                HEAD
 6100 FORMAT(30X,5(10H**********))                                        HEAD
 6200 FORMAT(/1H031X4A10,/1H031X,4A10,/1H0,49XA10,/1H0,49XA10,/)          HEAD
      END                                                                 HEAD

      SUBROUTINE TOP                                                      TOP
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                     COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                                COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                             EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP               COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500),  EXTEND
     + FREQ(500),PTR(500)                                                 EXTEND
      COMMON MAT(40000)                                                   EXTEND
      NPG=NPG+1                                                           TOP
      CALL TIME(TIM)                                                      TOP
      WRITE(6,600)TITLE,TODAY,TIM,NPG                                     TOP
      RETURN                                                              TOP
  600 FORMAT(1H1,2X8A10,2(1XA10),2X*PAGE*,I5,/)                           TOP
      END                                                                 TOP
```

```
      SUBROUTINE MOP                                                    MOP
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                   COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                              COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                           EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOF             COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500),EXTEND
     + FREQ(500),PTR(500)                                               EXTEND
      COMMON MAT(40000)                                                 EXTEND
      NPG=NPG+1                                                         MOP
      CALL TIME(TIM)                                                    MOP
C                                                                       MOP
C  ENTER GRAPH MODE                                                     MOP
C                                                                       MOP
      WRITE(6,600)                                                      MOP
      WRITE(6,610)TITLE,TODAY,TIM,NPG                                   MOP
      RETURN                                                            MOP
  600 FORMAT(1HQ)                                                       MOP
  610 FORMAT(1H ,2X8A10,2(1XA10),2X*PAGE*,I5,/)                         MOP
      END                                                               MOP




      SUBROUTINE SETRAY(A,N,B)                                          SETRAY
      INTEGER A,B                                                       SETRAY
      DIMENSION A(N)                                                    SETRAY
      DO 100 I=1,N                                                      SETRAY
      A(I)=B                                                            SETRAY
  100 CONTINUE                                                          SETRAY
      RETURN                                                            SETRAY
      END                                                               SETRAY
```

```
      SUBROUTINE ERROR(I)                                                  ERROR
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                      COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                                 COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                              EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP                COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500),   EXTEND
     + FREQ(500),PTR(500)                                                  EXTEND
      COMMON MAT(40000)                                                    EXTEND
      IF(I.LE.0.OR.I.GT.1)STOP                                             EXTEND
      STOP 1                                                               ERROR
      END                                                                  ERROR
                                                                           ERROR
```

```
      BLOCK DATA                                                           BLOCK
      INTEGER TASK,STABLE,A,TID,TABLE,CHOICE,FREQ,PTR                      COMMON
      COMMON /GIL0/ TITLE(8),TODAY,NPG,TIM                                 COMMON
      COMMON /GIL1/ TASK(500),STABLE(16),A(8)                              EXTEND
      COMMON /GIL2/ NTASK,ISRT,NSV,N,NR,NKC,TID,CHOICE,NCOP                COMMON
      COMMON /GIL3/ KC(500),KV(500),TABLE(500),LC(500),LCS(501),NB(500),   EXTEND
     + FREQ(500),PTR(500)                                                  EXTEND
      COMMON MAT(40000)                                                    EXTEND
      DATA NSV/16/                                                         BLOCK
      DATA STABLE/8RLOCOMOTN, 8ROBJ MANP,8RGDG-STRG,8RHUM INTR,            BLOCK
     X            8RLEADSHIP,8RORAL USE,8RREAD USE,8RWRIT USE,             BLOCK
     X            8RMETHODS ,8RQUALITY ,8RFIGURAL ,8RSYMBOLIC,             BLOCK
     X            8RTAXONOMC,8RIMPLICIT,8RFINC ERR,8RHUMN ERR/             BLOCK
      END                                                                  BLOCK
```

```
         IDENT  SHLSRT$                                                        SHLSRT

***      SHLSRT - 1 WORD/ENTRY TABLE SORT                                      SHLSRT
*                                                                              SHLSRT
*        CALL SHLSRT( TABLE , N )           - DIRECT ( IN PLACE ) SORT         SHLSRT
*        CALL SHLSRT( TABLE , N , PTR )     - INDIRECT SORT                    SHLSRT
*                                                                              SHLSRT
*        ENTRY:                                                                SHLSRT
*              TABLE = ADDRESS OF TABLE ( ARRAY ) TO BE SORTED                 SHLSRT
*              N = NUMBER OF WORDS IN TABLE                                    SHLSRT
*              PTR = ADDRESS OF AN ARRAY TO BE USED AS A POINTER               SHLSRT
*              TABLE ( DIMENSIONED PTR(N) )                                    SHLSRT
*                                                                              SHLSRT
*         IF THE THIRD ARGUMENT IS ABSENT THEN A DIRECT ( IN PLACE )           SHLSRT
*         SORT WILL BE PERFORMED, ELSE THE THIRD ARGUMENT WILL                 SHLSRT
*         BE USED AS A POINTER TABLE AND AN INDIRECT ( POINTER )               SHLSRT
*         SORT WILL BE PERFORMED. ON EXIT THE POINTER TABLE WILL               SHLSRT
*         POINT THE THE ELEMENTS OF *TABLE* IN ASCENDING ORDER.                SHLSRT

         ENTRY  SHLSRT                                                         SHLSRT
STBA     BSS    2                                                              SHLSRT
         SA5    STBA          RESTORE AC                                       SHLSRT
         SA0    X5                                                             SHLSRT
         PS                                                                    SHLSRT
         SB1    1                                                              SHLSRT
         SA2    A1+B1                                                          SHLSRT
         SX6    A0                                                             SHLSRT
         SA3    A2+B1                                                          SHLSRT
```

```
SA2     X2                                              SHLSRT
SA0     X1-1        A0 = FWA-1                          SHLSRT
SA6     STBA        SAVE A0                             SHLSRT
SB7     X2                                              SHLSRT
SB6     X2          N = N (LENGTH)                      SHLSRT
NZ      X3,STB1     IF AN INDIRECT SORT REQUESTED       SHLSRT
                                                        SHLSRT
SX6     B6          M = M/2                             SHLSRT
AX6     1                                               SHLSRT
SB6     X6                                              SHLSRT
SB3     B1          J = 1                               SHLSRT
ZR      B6,STB0     RETURN IF M = 0                     SHLSRT
SB4     B7-B6       K = N-M                             SHLSRT
SB2     B3          I = J                               SHLSRT
SB5     B2+B6       L = I+M                             SHLSRT
SA1     A0+B2       A(I)                                SHLSRT
SA2     A0+B5       A(L)                                SHLSRT
IX4     X2-X1                                           SHLSRT
PL      X4,STB3     IF A(L) .GT. A(I)                   SHLSRT
BX6     X1                                              SHLSRT
LX7     X2          INTERCHANGE A(L) AND A(I)           SHLSRT
SA6     A2                                              SHLSRT
SA7     A1                                              SHLSRT
SB2     B2-B6       I = I-M                             SHLSRT
GT      B2,STB2     IF I .GT. 0                         SHLSRT
SB3     B3+B1       J = J+1                             SHLSRT
SB2     B3          I = J                               SHLSRT
LE      B3,B4,STB2      IF J .LE. K                     SHLSRT
EQ      STB1                                            SHLSRT
```

```
STBI     BX6   X1             SET UP POINTER TABLE                     SHLSRT
         SB2   B6-B1                                                   SHLSRT
         SA0   X3-1           A0 = FWA-1 OF POINTER TABLE              SHLSRT
         SA6   A6+B1          SAVE ADDR OF A                           SHLSRT
         SA6   X3                                                      SHLSRT
         SX6   X6+B1                                                   SHLSRT
         SB2   B2-B1                                                   SHLSRT
         SA6   A6+B1                                                   SHLSRT
         NZ    B2,*-1                                                  SHLSRT
                                                                       SHLSRT
STBI1    SX6   B6             M = M/2                                  SHLSRT
         AX6   1                                                       SHLSRT
         SB6   X6                                                      SHLSRT
         SB3   B1             J = 1                                    SHLSRT
         ZR    B6,STBI4       IF M = 0                                 SHLSRT
         SB4   B7-B6          K = N-M                                  SHLSRT
         SB2   B3             I = J                                    SHLSRT
STBI2    SB5   B2+B6          L = I+M                                  SHLSRT
         SA1   A0+B2          P(I)                                     SHLSRT
         SA2   A0+B5          P(L)                                     SHLSRT
         SA3   X1                                                      SHLSRT
         SA4   X2                                                      SHLSRT
         IX5   X4-X3                                                   SHLSRT
         PL    X5,STBI3       IF  A(P(L)) .GT. A(P(I))                 SHLSRT
         BX6   X1                                                      SHLSRT
         LX7   X2             INTERCHANGE P(L) AND P(I)                SHLSRT
         SA6   A2                                                      SHLSRT
         SA7   A1                                                      SHLSRT
         SB2   B2-B6          I = I-M                                  SHLSRT
         GT    B2,STBI2       IF I .LT. 0                              SHLSRT
STBI3    SB3   B3+B1          J = J+1                                  SHLSRT
         SB2   B3             I = J                                    SHLSRT
         LE    B3,B4,STBI2    IF J .LE. K                              SHLSRT
         EQ    STBI1                                                   SHLSRT
```

```
                                                                        SHLSRT
STBI4    SA5    STBA                                                    SHLSRT
         SA1    A5+B1      ADDRESS OF A ( TABLE )                       SHLSRT
         SB2    A0+B1      P(1)                                         SHLSRT
         SB3    B2+B7      LWA+1 OF P                                   SHLSRT
         SA0    X5         RESTORE A0                                   SHLSRT
         SX1    X1-1                                                    SHLSRT
STBI5    SA2    B2         CHANGE POINTER ARRAY TO FORTRAN SUBSCRIPTS   SHLSRT
         SB2    B2+B1                                                   SHLSRT
         IX6    X2-X1                                                   SHLSRT
         SA6    A2                                                      SHLSRT
         LT     B2,B3,STBI5                                             SHLSRT
         EQ     SHLSRT                                                  SHLSRT
                                                                        SHLSRT
        END                                                             SHLSRT
```

```
      SUBROUTINE ZERO(A,N)
C
C     THIS SUBROUTINE ZEROS OUT AN ARRAY
C     FROM THE FIRST TO THE LAST WORD IN THE ARRAY.
C
      DIMENSION A(N)
C
      DO 100 I=1,N
      A(I)=0
  100 CONTINUE
C
      RETURN
      END
```

APPENDIX G

A REVIEW OF THE HEALTH SERVICES MOBILITY STUDY METHODOLOGY *

* The three papers presented in this appendix were written in 1976 in response to requests from the Director of HSMS and its funding agency. The materials reviewed did not include the full-scale application of HSMS methodology in diagnostic radiology and the development of curriculum guidelines, which were reported in late 1976 and 1977.

SOME NOTES ON THE STATISTICAL METHOD UTILIZED BY THE HEALTH SERVICES MOBILITY STUDY

By Earl E. Davis[1]

INTRODUCTION

The purpose of this paper is to provide a rationale for the statistical procedures utilized by the Health Services Mobility Study (HSMS) in its task analysis method. The basic statistical technique used by HSMS to analyze its task data is that of factor analysis. HSMS use of factor analysis has been reported briefly in some of its documents [10, 11, 12, 13].[2] This paper proposes to provide a technical description and explanation of the techniques used and statistical decisions made in connection with the HSMS method. We hope that this will facilitate replication of the HSMS statistical analyses and will provide a concrete basis in terms of which to discuss the rationale for the techniques adopted, to thus make possible critical review.

We first present a brief description of the HSMS data base and the HSMS use of factor analysis. Then some of the issues and some of the literature of the field are discussed in connection with the particular

---

1 This document was prepared by Dr. Davis in January, 1976, at the request of the Health Services Mobility Study (HSMS). It has been edited by Eleanor Gilpatrick, Director of HSMS. Dr. Davis was formerly Chief Consultant of HSMS. He is currently Research Professor and Chairman of the Department of Social Psychology and Sociology of The Economic and Social Research Institute, Dublin, and Adjunct Professor, Department of Statistics, Trinity College, University of Dublin.

2 Numbers in brackets refer to references listed at the end of this paper.

procedures utilized in the HSMS method. The issues relate to (1) the appropriateness of HSMS data for factor analysis, i.e., the basic assumptions underlying the use of parametric statistics in general, and factor analysis in particular, (2) the question of whether communality estimates or unities in the diagonal are preferable in the factor analytic procedure, and (3) the choice among different rotational procedures. These are considered in the light of HSMS objectives and the nature of the HSMS data.

Terms

A possible source of confusion in a discussion of factor analysis is various authors' use of terms. For example, terms such as "principal components," "principal axes," "principal factors," "principal axis factors," and others are used somewhat differently (and sometimes a bit loosely) by different writers. In a recent criticism of an article by Timmermans and Sternbach [45] one writer, reflecting Harman's usage [16], suggests that "the distinguishing characteristics of factor analysis in contrast to principal components analysis is the substitution of estimates of communalities in the diagonals in place of unities..." [36, p. 861]. Harman distinguishes between principal components and principal factors [16, p. 100] (i.e., component analysis and classical factor analysis [16, p. 346]). In this paper we use the term factor analysis to cover a variety of methods including the principal components method, unless distinguishing one method from another.

We also prefer to use the term "principal axis (PA) factors" instead of "principal factors" to avoid confusion with the so-called "principal-factor" technique, which is one of several factor analytic techniques for extracting principal axis factors in order to reduce an m x m matrix to an m x k matrix of PA factors (k < m).

HSMS Data Base

This paper assumes that the reader is acquainted with the general features, objectives, and underlying data base used in the HSMS method of task analysis as presented in relatively nontechnical terms in various HSMS documents [10, 11, 12, 13].

The goals of the HSMS method are to design job ladders and curricula based on the skill and knowledge requirements of tasks. This requires a descriptive use of statistics.

One feature of the HSMS method is a carefully conceptualized definition of the basic unit of observation, the task. The interrater reliability of the definition, used to identify tasks in the field, was tested and showed a satisfactory degree of reliability.

The basic variables developed in the HSMS method are sixteen skill dimensions and the knowledge categories of the HSMS Knowledge Classification System. These variables are applied to the task observations and are assigned values by use of the scales developed for each of the skills and a scale used for all the knowledge categories. Each of these seventeen scales was developed in a complex and carefully applied procedure using the Thurstone equal-interval scaling technique [9, 44], which gives these variables the appropriate statistical properties. Each scale has its lowest value at zero and its highest at 9.0. Some scales have as many as eight descriptors. No scale has less than five.

The HSMS method incorporates preparation of the HSMS task data for analysis using the EDIT program designed for HSMS. Simple and "two-mode" factor analyses are then used to cluster the variables and tasks.

In its initial preparation of the data, the EDIT program deletes variables from consideration that have a frequency across tasks below a selected minimum. This feature partly solves the problem of variables which take on a value of zero for an excessive number of observations. It also permits the user to reduce the number of variables to an appropriate number in relation to the number of observations, which is an important statistical consideration. The EDIT program is then used to "normalize" the data. That is, a nonlinear transformation is performed on the data to bring it into a closer approximation of linearity among variables.

HSMS Use of PCVARIM

The initial factor analysis program used by HSMS for determining the number of factors in the solution and for clustering the variables has the name PCVARIM, which is an abbreviation for Principal Components Factor Analysis with Varimax Rotation. This particular program has certain features that incorporate decisions about some of the procedures subsumed under the general term "factor analysis." The program uses a principal components technique for arriving at principal axis factors ("unities" in the diagonal rather than communality estimates), and it uses orthogonal varimax rotations. The matrix of variables is a correlation matrix rather than a covariance or a cross-products matrix.

The PCVARIM program was originated at the University of Illinois. It was initially programmed by Paul Herzberg [17] under the supervision of Henry Kaiser in the late 1950's. The factor analytic adaptation of the original principal components method consists essentially of an application of a form of triangular decomposition, which is described in some detail by Harman [16, pp. 101-103]. This is incorporated in a subroutine of the program referred to as TRIDI (a technique for tri-diagonalization of matrices). Subsequently, an extremely useful, widely used subroutine was developed by Householder, Ortega and Wilkinson, with the appropriate name of HOW. This subroutine involves a highly sophisticated set of procedures for the numerical analysis of eigenstructures, and calls, in course, the other subroutines TRIDI, EIGVEC (eigenvector), and EIGVAL (eigenvalue). This subroutine was originally programmed by David W. Matula, under the direction of William Meredith at the University of California at Berkeley's Computation Center. (A more complete account of the mathematics of the subroutine HOW may be obtained from the chapters written by the originators of the techniques [see in 37].)

The resulting PA factors obtained by PCVARIM are then rotated to simple structure using the varimax procedure for orthogonal rotation developed originally by Kaiser [20]. The history and mathematics of the development of this procedure for orthogonal rotation is described in complete detail by Harman [16, Chapter 14]. We have come to call this set of procedures "simple" factor analysis.[3]

HSMS Use of Two-Mode Factor Analysis

While, in principle, one can apply simple factor analysis to any set of data, simple factor analysis is usually used to arrive at some structure and a parsimonious explanation of a set of variables ,i.e., observed variations (or variates) which occur over a number of observed cases. In psychometrics, the cases, or observations, would usually be individuals In the HSMS research, the observations or sources of observed variations are the tasks involved.

While there is interest in the factor structure of the variables, i.e., skills and knowledge categories, the ultimate HSMS aim is to discover the underlying structure of the observations, i.e., tasks. The reason is that a goal of HSMS is the construction of job ladders based on the tasks. In other words, we wish to see how job tasks would cluster with each other. In principle, we could have used simple factor analysis of the type described above, or any variation thereof, to cluster the tasks. The problem was that we then could not know what the bases were on which the tasks clustered. We wished to inspect the factor structure of the skill and knowledge variables which comprise the HSMS data base as the basis for clustering the tasks. What was clearly needed was

---

3 "Simple" factor analysis is described here in fairly general terms, since the sources to which we refer contain the explicit mathematical formulations which permit complete replication.

the capability of factoring in more than one mode simultaneously, in such a way as to establish a unique relationship among the multiple reference axes of the modes involved, i.e., variables and observations.

The procedure used for clustering the HSMS task observations is a modified version of the Tucker-Messick procedure for factoring an individual differences matrix [7, 49]. The program permits the extraction of principal axis factors for both observations and variables (two modes), based on a covariance or a correlation matrix of variables. It is then possible to rotate one mode to "simple structure" by a varimax method, and to "counter-rotate" the second mode. For HSMS the first mode is the skills and knowledges variables; the second mode is the task observations. Counter rotation is done by obtaining the transformed characteristic vectors of the second (observations) mode induced by the varimax rotations of the first (variables) mode. [See 47 and 48.] We refer to this use of factor analysis as "two-mode factor analysis."

A problem similar to the HSMS problem had been studied in psychometrics over some time, namely, how to establish "idealized subject types." In 1964, E. E. Davis and H. C. Triandis, working at the University of Illinois, read the seminal article by Tucker and Messick [49] describing a procedure for factoring an individual differences matrix, and were aware of the continuing efforts by Tucker in the direction of multi-mode factor analysis [e.g., 47, 48]. They were also quite fortunate in having a good working relationship with Professor Tucker, who was in the same department at that time. They were thus able to "interrupt" Professor Tucker in his concentration on perfecting three-mode (and n-mode) factor analysis to apply the principles involved to two-mode factor analysis.

Davis and Triandis were dealing with social attitudes and interpersonal perceptions, but the principle behind the technique utilized is precisely what was called for to meet the aims of the HSMS method.[4]

In general terms, the two-mode factor analysis program performs the following functions. It permits the derivation of two principal axis factor matrices, one for variables (skills and knowledges), and one for observations (tasks), through a principal components technique, using tri-diagonalization, modeled on the PCVARIM program described above. The critical element, however, is the utilization of the Eckert-Young theorem [8] for approximating one matrix from another in such a way that it becomes possible to rotate one matrix to simple structure and subsequently "counter-rotate" (to use a rather loose terminology) the other matrix in such a manner as to obtain the isomorphic relationship between the two sets of reference axes that is desired.

In the case of the HSMS method, the first set of principal axis factors is rotated to simple structure by means of the varimax criterion. This is the first mode. Then the transformed characteristic vectors of the second mode are obtained by "counter-rotation" of the corresponding

---

4 Utilizing this technique, Davis and Triandis [7] were able to demonstrate empirically the validity of a model which provided the resolution of a long-standing conflict as to the determinants of social acceptance or rejection. This model and the studies and controversies leading up to its formulation and proof are perhaps best summarized in a replication study carried out by Goldstein and Davis [14].

second mode vectors. The transformed characteristic vectors of the second mode are thus induced by the varimax rotation of the first mode.

Either the observation mode or the variable mode may be used as the first mode and the other as the second mode, depending on the nature of the data. Davis and Triandis obtained simple structure of the observation mode first. In the case of HSMS, we obtain the simple structure of the variable mode first. The transformed characteristic vectors or factor loadings of the HSMS task observation mode correspond to, and thus can be interpreted in terms of, the corresponding skill and knowledge variable mode factors. The simple structure of variables helps to interpret the corresponding structure of tasks.

The matrix operations can be briefly conceptualized as follows:[5]

m = number of variables

n = number of observations

k = number of factors

Y = raw data matrix with elements $y_{i,j}$, $i = 1,\ldots,n$;

$j = 1,\ldots,m$

X = rescaled matrix with elements $x_{i,j} = (y_{i,j} - \bar{y}_j)/(s_j\sqrt{n})$

Where s_j = standard deviation of $y_{.j}$.

From the Eckert-Young theorem, we can say that:

$$ {}_nX_m \cong {}_nU_k * \lambda_k * {}_kV'_m $$

[5] In this presentation the observation mode is first rotated to simple structure. In the case of HSMS application, the variable mode is first rotated to simple structure.

where

U = characteristic vectors of XX'

V = characteristic vectors of $X'X$

λ = diagonal matrix of the square roots of the characteristic roots of <u>both</u> ${}_mX' * X_m$ <u>and</u> ${}_nX * X'_n$.

Also, ${}_nU_k * \lambda_k = {}_nP_k$

= principal axis factors for observations.

Now, let ${}_kT_k$ be the transformation matrix such that

$${}_nP_k * {}_kT_k = {}_nR_k$$

where R corresponds to the varimax factors for observations.

Then,

$${}_nX_m = {}_mP_k * {}_kT_k * {}_kT^{-1}{}_k * {}_kV'_m$$

$$= {}_nR_k * {}_kT^{-1}{}_k * {}_kV'_m$$

Let $S = T^{-1} * V'$; then,

$${}_nX_m = {}_nR_k * {}_kS_m$$

${}_nR_k$: varimax factor loadings for observations.

${}_kS_m$: transformed characteristic vectors for variables.

The set of operations above is an abbreviated version of the entire matrix operations, designed primarily to illustrate the operation of the Eckert-Young theorem. Steps such as the generation of the original and rescaled matrices and their conversion into PA factor matrices are omitted.

Davis and Triandis originally used a covariance matrix as the input to the factoring [7]. More recently, Osgood and associates employed a cross-products matrix as the input to the multi-mode factor analysis procedure [e.g. 33, 50]. Factor analysis is a generic technique which can be applied to any type of matrix. Historically, factor analysis has used a matrix of correlations among all the variables of an original raw data matrix in the initial step. However, it is equally possible to convert the raw data matrix to a covariance matrix, a cross-products matrix, or some other form of matrix, and to factor the resulting transformed matrix. The cross-products matrix is particularly applicable to the Osgood Semantic-Differential type of scales [33] because of their symmetry. We determined that the HSMS data do not appear to lend themselves to the use of either a cross-products or a covariance matrix, and a conventional correlation matrix is used.

NOTES ON SOME STATISTICAL ISSUES IN FACTOR ANALYSIS

Aside from the frequently heard (and often quite justified) criticism that factor analysis is used indiscriminately by people who do not know what they are doing or why they are doing it, there are basic statistical problems involved with the use of parametric statistics generally and factor analysis in particular. Obviously, any technique, no matter what its intrinsic value, can be misused or used indiscriminately.

Controversies concerning the legitimacy of factor analysis as a technique and concerning the best method of factor analysis characterized the development of the technique in the thirties and forties. Cureton describes this in a rather witty and sarcastic way [6]. More recently, such authors as Harman [16], Cooley and Lohnes [5] and others have dealt with the subject. Harman states:

> The many papers that appeared during the thirties and forties urging "this method" rather than "that method" had their place in the growth of the subject. However, with a fuller understanding of the salient features of each method, and with the increased efficiency of computations, the differences among the various methods no longer loom so ominously, and the followers of a particular approach are much more tolerant of the adherents of an alternative scheme. [16, p. 10]

Harman indicates that "the heated and inspired controversies about the 'best' method of factor analysis are over" [16, p. 9]. As Cooley and Lohnes have put it, "only recently have students of factor analysis begun to see that the different procedures are suitable for different purposes..." [5, p. 129].

Two of the major statistical questions relating to factor analysis as a technique have to do with the use of communality estimates or unities in the principal diagonal,and with procedures for rotation. Even more fundamental, however, are questions concerning the robustness of factor analysis. This section first discusses the issue of robustness, then the rationale for the use of unities in the diagonal of the correlation matrix as input to the factoring procedure, and, finally, the selection of a rotation technique in the HSMS application of factor analysis.

The Data, Factor Analysis and Robustness

In discussing preconditions for the use of factor analysis, Harman indicates that "all observed variables must be linearly related to one another" [16, p. 374]. He relaxes this requirement to include relationships that are monotonic. Another condition is that each observed variable be normally distributed or at least not distinctly non-normal [16, p. 374].

In this section we first suggest that the variables involved in the HSMS method are more amenable to factor analysis than might be initially assumed, and that it is appropriate to use factor analysis techniques in connection with them. Second, we suggest that modern statistical thinking and research findings indicate that factor analysis techniques are more robust than was previously thought to be the case, even with greater deviations from normality or linearity than was previously thought tolerable.

A theme which runs through the work of Kendall and Stuart [22, 23] is that, when n becomes large enough, the deviations from the "normal" distribution become trivial and the application of parametric statistical analysis becomes justified, despite the reservations of earlier "pre-classical" statisticians. The HSMS sample sizes are in practice large enough to meet the criterion. The n's which have been encountered thus far have been well over 200 task observations.[6]

The EDIT computer program, which is used to prepare the HSMS data for analysis, linearizes the data by a non-linear (logarithmic) transformation. In Kendall and Stuart's chapter on Canonical Variables [24,

---

6 Sources in Kendall and Stuart [22, 23, 24] were referred to the present author by Professor Kendall in a discussion in which the legitimacy (from the viewpoint of a statistician) of applying a factor analytic model to the HSMS data base was discussed in some detail. Professor Maurice G. Kendall, Personal Communication, Dublin, October 22, 1973. Confirmation of the validity of using factor analysis with data of the HSMS type has also been derived in discussions of the present author with R. C. Geary. Dr. R. C. Geary, Personal Communication, June 17, 1975.

Chapter 43], the authors show that logarithmic transformation can be used to linearize otherwise non-linear data. The authors, in discussing transformations, state:

> Consider a transformation to new variables ξ given by
>
> $$\boldsymbol{\xi} = \mathbf{ax} \qquad (43.4)$$
>
> where $\mathbf{a}$ is a matrix of coefficients. We confine our attention to linear transformations of this kind—non-linear situations are much more difficult to handle, and if they are suspected to exist an attempt should be made to linearize the data beforehand, for example, by a logarithmic transformation. [24, p. 286; emphasis added]

Recently, Kruskal and Shepard pointed out (in a paper on "non-metric linear factor analysis" [28]) that, "the standard methods, though presupposing linearity, are generally quite robust in the face of both random error and all but the most severe monotone departures from linearity" [28, p. 130]. These authors also point out the extreme expense (in computational time) of carrying out the rather strict procedure to achieve monotonicity and approximations to linearity which their program involves. They state that, "it is still doubtful whether there are any commonly occurring circumstances in which it is worthwhile to resort to the much more costly computation required," and conclude, citing Shepard and Carroll [39], that:

> It now appears that, in order to achieve an extension of (two-way) factor analysis of appreciable practical power, it may not be sufficient merely to weaken the metric assumption of linearity. It may be necessary to abandon even the assumption of monotonicity. [28, p. 153]

It would seem that, even with rather severe assumptions about departures from linearity in the data, the procedures which we have selected for the HSMS method could not have led to any significant distortion of the results. Given the time and the cost of the computational factors involved in alternative techniques, the gain which could be accrued from the alternative techniques would not offset the costs.

In other words, even if appreciable departures from linearity were the case, such departures would not significantly distort the results. Modern factor analytic theory suggests that the results to be gained by painstaking procedures to convert the data to follow the rigorous conventions of linearity do not yield results appreciably different from those obtained without transforming the data.

With regard to distribution of the data, we find that the variables which form the data base of the HSMS method do not grossly appear to depart from assumptions of "normal" distribution. Further, the requirement of normality is relaxed when descriptive rather than inferential uses of statistics are involved.

If we assumed that we were faced with the worst possible situation, in which each of the HSMS variables has a value of zero for close to half the observations, and all other values are positive, non-imaginary, and definite, we would be facing a situation in which we were seeking to factor analyze dichotomized variables. This is the limiting case for the HSMS data developed thus far.

For such a limiting case, Christoffersson [4, p. 5] has shown that the use of factor analysis with dichotomized variables is entirely feasible. He reviews the difficulty involved in factor analyzing a matrix of tetrachoric correlations, since such methods usually require (among other things) that the correlation matrix be Gramian (i.e., positive,semi-definite,and symmetrical,with unities in the principal diagonal). He goes on to demonstrate how the basic factor analytic model can be adapted to dichotomized variables.

Christoffersson uses two different maximum likelihood approaches which take into account variables that are dichotomized [4, p. 5], namely, (1) the conditional maximum likelihood method [31], and, (2) the unconditional maximum likelihood method [3]. The latter is an extension of the normal factor analytic model based on an estimation of parameters of the basic factor model. It uses the generalized least squares principle (the GLS-Estimator), and involves the tetrachoric expansion put forth by Kendall [21].[7]

Whelan [51] uses a Monte Carlo approach to an examination of factor analysis.[8] His findings appear to verify those of Christoffersson concerning the possibility of factor analyzing dichotomous variables. We may also infer the acceptability of the HSMS data from the following:

7 Christoffersson and his associates Anderson and Muthen [1] report that they have developed a computer program for carrying out factor analysis of dichotomized variables, which is available on request. They point out that, so far, they have run into difficulties with computational time. It is probably true to say that the two-mode program used by HSMS is one of the few computer programs available which not only carries out a factor analysis meeting the needs of HSMS, but also is within the bounds of reasonable computer costs.

8 Monte Carlo approaches have been used extensively in recent years in non-metric scaling (e.g. Klahr [25], Sherman [40]),as well as in the investigations of metric problems such as those posed by factor analysis.

> Quite frequently in economic and social research important variables can only be measured at a dichotomous or polychotomous level. The inclusion of such variables violates the assumption of normally distributed, continuous variables on which factor analysis is based. A related, though less serious, difficulty arises when variables are measured on a scale containing only three, five or seven points. The larger the number of points on such a scale the more closely the variables conform to the assumption of continuity. Any number of points greater than four is generally assumed to provide a sufficiently close approximation to continuity for the purposes of factor analysis. [51, p. 16]

To investigate the problem of dichotomous variables Whelan took the data which he had earlier analyzed using several factor analytic programs and transformed them into dichotomized variables by setting each negative value equal to -1/2, and each positive value equal to +1/2. This set of dichotomized data was then factor analyzed by means of one of the standard programs.

The author concludes that:

> It may be seen that the actual and estimated structures are quite similar, the highest correlation observed between estimates and actual being over 0.9....The relatively small deterioration in the quality of the estimates obtained when dichotomized data are used is therefore quite striking. [51, pp 16-17]

In a further analysis of the data, following an even greater asymptotic transformation in the distribution of the data, Whelan tested out a hypothesis put forth by Raven, Ritchie, and Baxter [38], suggesting that factor analytic results may be an artifact of the factor analysis algorithm in those cases where high proportions of the sample tend to have either very low or very high scores. This is analogous to the extreme case with HSMS data where there are "too many zeros."

In order to test the hypothesis, Whelan assigned artificially extreme values to the set of variables on which he had a known factor structure and found "no distinct pattern in the loadings." He concluded that the hypothesis found no confirmation. Whelan says that, "we see that even in the case of unequal endorsement rates [scores], the estimates of the factors derived by the programme are still quite good, since all the correlation coefficients are greater than 0.75" [51, p. 18].

Communality Estimates or Unities in the Diagonal

In describing the distinction between the principal components method and other factor analytic techniques, Harman refers to R as a matrix of observed correlations among variables. He goes on to state:

> A set of n variables can be analyzed either (a) in terms of common factors only, by inserting unities in the diagonal of R; or (b) in terms of common and unique factors, by inserting communalities in the diagonal of R. These two approaches, of course, correspond to the component analysis and the classical factor analysis models, respectively....In the first instance R is a Gramian matrix, generally of rank n, and the factor solution
>
> (16.1) $z = Af$
>
> is in terms of n common factors. Since A is a square non-singular matrix, in this instance, it will have an inverse. Then the required factor measurements are given simply by:
>
> (16.2) $f = A^{-1}z$.
>
> This solution is determined exactly, is unique, and involves no "estimation."

> However, when the factor model involves common and unique factors the solution is not so simple. Then the total number of factors exceeds the number of variables, and an inverse does not exist for the factor matrix M. The generally accepted procedure, in this case, is to resort to the "best fit" in the least squares sense [i.e., communality estimates]. [16, p. 346]

In an early paper [35] Karl Pearson set forth the "method of principal axes" which has formed the basis for the method of principal components. The method of principal components outlined by Pearson is no longer used in its original form, but rather in the form of specific adaptations to factor analysis of this technique,such as those made by Hotelling [19] and later scholars.

Cooley and Lohnes state that one of the major uses of factor analysis is "to find ways of identifying fundamental and meaningful dimensions of a multi-variate domain." They then go on to say that:

> this "construct-seeking" task of factor analysis is most frequently accomplished today by first conducting a principal-components analysis, and by then using the resulting principal factors as a set of reference axes for determining the simplest structure, or most easily interpretable set of factors, for the domain in question. [5, p. 131]

Whether one uses a principal components method with unities in the diagonal or other factor analytic techniques with communality estimates in the diagonal is actually a choice regarding the extraction of variance or reproduction of the observed correlations. According to Harman:

> ...An important property of [the principal components] method, insofar as the summarization of

> data is concerned, is that each component, in turn, makes a maximum contribution to the sum of the variances of the _n_ variables. For a practical problem only a few components may be retained, especially if they account for a large percentage of the total variance. However, all the components are required to reproduce the correlations among the variables.
>
> In contrast to the maximum variance approach, the classical factor analysis model is designed to maximally reproduce the correlations....
>
> ...each of the _n_ observed variables is described linearly in terms of _m_ (usually much smaller than _n_) common factors and a unique factor. The common factors account for the correlations among the variables, while each unique factor accounts for the remaining variance (including error) of that variable.[16, p. 15]

The principal components approach was selected for the HSMS data since it appears to be a logical use for an essentially descriptive undertaking. However, we did carry out an analysis to see whether the use of communality estimates in the diagonal would make an appreciable difference.

A number of methods have been proposed for estimating communalities. As Harman states, "As a matter of fact none of the methods has been demonstrated to lead to minimal rank of the correlation matrix" [16, p. 83]. Harman goes on to say that:

> As a saving grace, there is much evidence in the literature that for all but very small sets of variables, the resulting factorial solutions are little affected by the particular choice of "communalities" in the principal diagonal of the correlation matrix. [16, p. 83]

Before collecting main test data we compared factor solutions on pretest data obtained by the PCVARIM principal components technique (including use of unities in the diagonal) with ones resulting from squared multiple correlations as communality estimates in the diagonal and iteration by refactoring. (We used the BMD package from UCLA which has this option in it.) We then systematically compared the resulting factor structures by means of the coefficient of congruence developed by Tucker [46] and Wrigley and Neuheus [52]. In comparisons between these two methods, involving two different sets of data, we consistently obtained coefficients of congruence well in the .90's, and concluded that it made no significant difference which method was used with the HSMS data.

In a later run with 273 observations and 144 variables we compared our PCVARIM 6-factor solution (selected after inspecting all solutions from two factors to ten factors) with the BMD 6-factor solution. We again found great similarities in the loadings.

These results supported our decision to stay with the type of analysis which we originally selected as best for our needs, namely, two-mode factor analysis, which in the program available to us at the time incorporated a form of principal components technique.

The Question of Rotational Technique

Another question which has occupied the attention of factor analysts concerns techniques for rotating the initial principal axis factors. The PA factors, extracted by whatever technique, are initially unrotated and usually not very interpretable in their original form. Subsequently, any one of several rotational techniques is applied in order to achieve "simple structure" [42] in the form of a set of rotated factors which are generally more interpretable. A variety of techniques ranging from the early hand-rotational techniques of Thurstone [43] to later developments involve various analytical solutions for orthogonal or oblique rotations.

As mentioned earlier, we chose to use orthogonal rotation with our data, using Kaiser's varimax criterion [20]. We chose a solution which would not have correlated factors because the ultimate objective is to separate tasks into separate job ladders.

Since there has been a great deal of discussion about the various advantages and disadvantages of orthogonal versus oblique rotations, we decided early in the selection process for the HSMS method to conduct an empirical test as to whether the particular method of rotation made any significant difference with our test data. We used the BMD factor analysis program which contains a variety of options. We subjected sets of pretest data to both varimax orthogonal rotations and oblique rotations of the oblimax and oblimin types by varying the magnitude of beta in the basic rotational equation. We again used analytical comparisons involving coefficients of congruence, as described above. We consistently found the comparisons between the factor solutions yielding coefficients well into the .90's.

It may well be the case that for certain psychological variables different factor solutions are obtained depending upon whether one decides on this or that orthogonal or oblique rotational method. Our empirical evidence suggested, however, that with the HSMS data, no significant difference is found when one changes from one technique to the other. This suggestion of the basically orthogonal factor structure underlying the HSMS data and the robustness of factor analytic techniques applied to them seemed to support our choice of the two-mode program and the decisions it implies.

Conclusion

The HSMS method has been carefully developed from a statistical point of view. Decisions made at every step along the way were made consciously. Review of the more recent factor analytic literature suggests that our original decisions were taken on sound grounds and tend to corroborate the initial decisions which we made.

REFERENCES

1. Andersson, C.G., Christoffersson, A., and Muthen, B. FADIV: A Computer Program for Factor Analysis of Dichotomized Variables. Research Report 74-1, Uppsala University, Statistics Department, 1974.

2. Barlow, R.E., Bartholomew, D.J., Bremner, J.M., and Brunk, H.D. Statistical Inference Under Order Restrictions. New York: John Wiley and Sons, Inc., 1972.

3. Bock, R.D., and Lieberman, M. Fitting a response model for n dichotomously scored items. Psychometrika, 35, 1970, 179-197.

4. Christoffersson, A. Factor analysis of dichotomized variables. Psychometrika, 40, 1975, 5-32.

5. Cooley, William W., and Lohnes, Paul R. Multivariate Data Analysis. New York: John Wiley and Sons, Inc., 1971.

6. Cureton, Edward E. The principal compulsions of factor-analysis. Harvard Educational Review, 9, 1939, 287-95.

7. Davis, E.E., and Triandis, H.C. An Exploratory Study of Intercultural Negotiations. Technical Report #26, ONR Contract #177-472, Nonr-1834 (36). Urbana: University of Illinois, 1965.

8. Eckart, C., and Young, G. The approximation of one matrix by another of lower rank. Psychometrika, 1, 1936, 211-18.

9. Edwards, A.L. Techniques of Attitude Scale Construction. New York: Appleton-Century-Crofts, 1957.

10. Gilpatrick, Eleanor. Final Report for the Period October, 1967 Through March, 1972. Technical Report No. 11. New York: Health Services Mobility Study, 1972.

11. Gilpatrick, Eleanor. Suggestions For Job and Curriculum Ladders in Health Center Ambulatory Care: A Pilot Test of the Health Services Mobility Study Methodology. Research Report Nos. 4 and 5. New York: Health Services Mobility Study, 1972.

12. Gilpatrick, Eleanor. First Progress Report For Phase Four: April 1, 1972 to March 15, 1973. Technical Report No. 12. New York: Health Services Mobility Study, 1973.

13. Gilpatrick, Eleanor. An Introduction To The Work of The Health Services Mobility Study as of April, 1975. Technical Report No. 13. New York: Health Services Mobility Study, 1976.

14. Goldstein, M., and Davis, E.E. Race and belief: A further analysis of the social determinants of behavioral intentions. Journal of Personality and Social Psychology, 22, 1972, 346-355.

15. Harman, Harry H. Factor analysis. In H.S. Wilf and A. Ralston (Eds.), Mathematical Methods for Digital Computers. New York: John Wiley and Sons, Inc., 1960, 204-12.

16. Harman, Harry, H. Modern Factor Analysis. (2nd Ed., Rev.) Chicago: University of Chicago Press, 1967.

17. Herzberg, P.A. The Parameters of Cross-Validation. Urbana: University of Illinois, Department of Psychology, 1967.

18. Hoeffding, W. The large-sample power of tests based on permutations of observations. Annals of Mathematical Statistics, 23, 1952, 169.

19. Hotelling, Harold. Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24, 1933, 417-41, 498-520.

20. Kaiser, H.F. The varimax criterion for analytic rotation in factor analysis. Psychometrika, 23, 1958, 187-200.

21. Kendall, M.G. Relations connected with the tetrachoric series and its generalisations. Biometrika, 32, 1941, 196.

22. Kendall, M.G., and Stuart, A. The Advanced Theory of Statistics. Vol. I (3rd. Ed.) London: Charles Griffin and Co.,Ltd., 1969.

23. Kendall, M.G., and Stuart, A. The Advanced Theory of Statistics. Vol. II (2nd. Ed.) London: Charles Griffin and Co.,Ltd., 1967.

24. Kendall, M.G., and Stuart, A. The Advanced Theory of Statistics. Vol. III (1st. Ed.) London: Charles Griffin and Co.,Ltd., 1966.

25. Klahr, D. A Monte Carlo investigation of the statistical significance of Kruskal's nonmetric scaling procedure. Psychometrika, 34, 1969, 319-330.

26. Kruskal, J.B. Nonmetric multidimensional scaling: A numerical method. Psychometrika, 29, 1964, 28-42.

27. Kruskal, J.B. Monotone regression: Continuity and differentiability properties. Psychometrika, 36, 1971, 57-62.

28. Kruskal, J.B., and Shepard, R.N. A nonmetric variety of linear factor analysis. Psychometrika, 39, 1974, 123-157.

29. Ledermann, W. On the rank of the reduced correlational matrix in multiple factor analysis. Psychometrika, 2, 1937, 85.

30. Lingoes, J.C., and Guttman, L. Nonmetric factor analysis: A rank reducing alternative to linear factor analysis. Multivariate Behavioral Research, 2, 1967, 485-505.

31. Lord, F.M. An analysis of the verbal scholastic aptitude test using Birnbaum's three-parameter logistic model. Educational and Psychological Measurement, 28, 1968, 989-1020.

32. McDonald, R.P. A general approach to nonlinear factor analysis. Psychometrika, 27, 1962, 397-415.

33. Osgood, C.E. Exploration in semantic space: A personal diary. The Journal of Social Studies, 27, 1971, 5-64.

34. Osgood, C.E., Suci, G.J., and Tannenbaum, P.H. The Measurement of Meaning. Urbana: University of Illinois Press, 1957.

35. Pearson, K. On lines and planes of closest fit to systems of points in space. Philosophy Magazine, 6, 1901, 559-72.

36. Posavac, E.J. Analysis of human chronic pain. Science, 187, 1975, 860-61.

37. Ralston, A., and Wilf, H.S. (Eds.). Mathematical Methods for Digital Computers. Vol. II. New York: John Wiley and Sons, Inc., 1967.

38. Raven, J., Ritchie, J., and Baxter, D. Factor analysis and cluster analysis. Economic and Social Review, 2, 1971.

39. Shepard, R.N., and Carroll, J.D. Parametric representation of nonlinear data structures. In P.R. Krishnaiah (Ed.), Multivariate Analysis. New York: Academic Press, 1966, 561-592.

40. Sherman, C.R. Nonmetric Multidimensional Scaling: An Empirical and Theoretical Investigation. Unpublished Ph.D. dissertation. Toronto: University of Toronto, 1970.

41. Spearman, Charles. General intelligence, objectively determined and measured. American Journal of Psychology, 15, 1904, 201-93.

42. Thurstone, L.L. Multiple factor analysis. Psychological Review, 38, 1931, 406-27.

43. Thurstone, L.L. Multiple Factor Analysis. Chicago: University of Chicago Press, 1947.

44. Thurstone, L.L., and Chave, E.J. The Measurement of Attitude. Chicago: University of Chicago Press, 1929.

45. Timmermans, G., and Sternbach, R.A. Analysis of human chronic pain. Science, 184, 1974, 806.

46. Tucker, L.R. A method for synthesis of factor analysis studies. Personnel Research Section Report No. 984. Washington, D.C.: Department of the Army, 1951.

47. Tucker, L.R. The extension of factor analysis to three-dimensional matrices. In Fredericksen (Ed.), Contributions to Mathematical Psychology. New York: Holt, Rinehart, and Winston, 1964.

48. Tucker, L.R. Some mathematical notes on three-mode factor analysis. Psychometrika, 31, 1966, 279-311.

49. Tucker, L.R., and Messick, S. An individual differences model for multidimensional scaling. Psychometrika, 28, 1963, 333-367.

50. Tzeng, O. Differentiation of Affective and Denotative Meaning Systems in Personality Ratings Via Three-Mode Factor Analysis. Unpublished Ph.D. dissertation. Urbana: University of Illinois, 1972.

51. Whelan, B.J. Factor fiction: A Monte Carlo approach to factor analysis. Seminar paper delivered at The Economic and Social Research Institute, Dublin, June, 1975.

52. Wrigley, C., and Neuhaus, J.A. The matching of two sets of factors. Contract Memorandum Report, A. 32, Urbana: University of Illinois, 1955.

AN EVALUATION OF THE HEALTH SERVICES MOBILITY STUDY METHODOLOGY[1]

By Philip R. Merrifield[2]

INTRODUCTION

This report is presented in four major sections, each of which contains a brief summary and an evaluation, with primary emphasis on methodological issues. It is unavoidable that certain substantive issues arise; it is hoped that any ignorance on this evaluator's part of the complex area of health services may be forgiven, and that such deficiencies have small, if any, effect on the methodological comments. The four sections are:

Goals: importance; specificity; attainability.

Strategy: appropriateness for stated goals; applicability given presumed resources; awareness of options; inferred familiarity with tools and methods proposed.

Performance: analysis of career and educational ladders and lattices; task identification; skill and knowledge identification; scaling; reliability and validity of scales and derived measures; interrelations among tasks, skills, and aspects of knowledge; documentation.

Overall summary and suggestions.

---

1 This document was completed by Dr. Merrifield in July, 1976,and is in response to a request for review of the Health Services Mobility Study (HSMS) method. Documents supplied for review purposes included the Davis paper [1] (which precedes this), listings of the HSMS computer programs requested by the author, early HSMS documents not currently available [2], Research Report Numbers 4 and 5 [3], Technical Report No. 13 [4], and current HSMS scales not yet published. Dr. Merrifield also discussed aspects of the methodology with Eleanor Gilpatrick, Director of HSMS. Numbers in brackets refer to references listed at the end of this paper.

2 Dr. Merrifield is Professor of Educational Psychology at New York University and is engaged in a number of ongoing research and evaluation projects in the behavioral sciences and on learning. He is also an educational consultant.

GOALS

On first encounter, the goals of this project are awesome. To bound, dimensionalize, and map such an area--not only topologically but with a great deal of quantification--is surely an effort worthy of a great explorer. But affect aside, the importance of the effort is clearly and convincingly stated. Obviously, the work needed doing. It is to be hoped that the results will not now be set aside due to lack of funds during the present crisis. Decision-makers must be made forcefully aware of its value as well as in the domain of educational and career development that it directly concerns.

I note with approval the level of specificity and objectivity in the statements of goals and of anticipated end results from the project effort. It is a measure of confidence and competence that such specificity is introduced early in the documentation, for in so doing one runs great risk from later evaluators. A vague objective can be interpreted favorably or unfavorably: in these reports there are definite commitments, which I applaud. I shall attempt to make my comments equally specific, both as to apparent difficulties, successes, and suggested actions.

With all their importance and specificity, are these goals attainable? Given the state of the art of job analysis and psychological measurement, is it possible to prepare the extensive materials needed for such a broad program? In this domain, new knowledge and new tools develop rapidly, and a certain tolerance for Sisyphean labor must characterize those leading the project. I think it a decision wisely made to begin with a prototype system, with the aim of verifying strategy and method rather than obtaining new substantive knowledge. As a spin-off, however, it would appear that some very interesting results were obtained, and might provide a basis for planning the curriculum-building and selection-placement aspects of the general problem. In reporting results of the HSMS pilot test [3], Gilpatrick quite properly suggests a modicum of caution in generalizing from these results; although the outcomes are satisfying in terms of what was expected, the limitation of the pilot study to a single institution and the relatively small ratio of replications to variables warrant a concern for further data.

With regard to Goals, then, it appears to me that the staff of this project knew in quite specific terms where it wanted to go, knew where their objectives fit into the larger scheme, and made very reasonable initiating decisions to reach those goals.

STRATEGY

The basic scheme for the collection of data seems to involve three major sources of information: jobs, as carried out by performers; tasks, as components of jobs; and requisite skills and knowledge, presumed to have been already developed and/or learned by incumbent performers of jobs. Obviously, the skills and knowledge dimensions could form a framework for efficient training for new employees. Similarities of tasks across job categories would allow for horizontal transfers of performers from one job ladder to another, without the need to re-establish their possession of skills and knowledge in the new job setting.

Some orientation in the new setting might be required to maximize the transfer of skills to the new job.

It was believed that a system could be developed to "score" each task on each of the skills and knowledge components; from this score matrix, the technique of factor analysis could disclose groupings of skill and knowledge components (hereinafter SK) which would be conceptually simpler and perhaps highly related in action. For example, if two skills were both required at a higher level by one subset of tasks than they were required by the subset of the remaining tasks, the two skills would be correlated and would "load" on the same factor; the factor, in turn, would be defined by the skill and knowledge components that loaded on it. If the number of groupings of SK variables is suitable for describing the differences and similarities among the tasks, then factor analysis in the classic factor-score model or in what Davis [1] presents as two-mode factor analysis may be used.

At this point, a trade-off that may be important occurs. In order to use two-mode factor analysis, it is necessary to compute factors from the cross-products (or covariance, or correlations) among tasks, as well as among SK variables. This computation requires that each task be represented only once, to permit the computation of the inverse of the appropriate matrix. However, if a number of tasks appear common to a number of jobs, should they not be represented more frequently in the total space being analyzed? It seems to me that concerns for representativeness would indicate such inclusion, which might well change the values of the correlations among SK variables and thus have some effect on the factors. I should like to recommend some exploration of this possibility, and assert that the factor-score model would handle the problem quite well.[3]

In this connection, I must point out that the Davis paper is not sufficiently explicit regarding the source of data for the two-mode analysis: specifically, the computation of the core matrix should be included. My reading of the program by which the data reported were obtained assures me that the procedures themselves were sound, but the reporting is incomplete, at least for unsophisticated users.

The question of using correlations or cross-products requires more discussion, especially concerning the information about differences between means of tasks that is lost when correlations are used in two-mode factor analysis. The existing program provides options that could be explored in future data analyses, although it is appropriate to take the procedure used in this report as a first priority.

As for other strategic decisions regarding the factor analysis, my preference would be for principal factors, with communalities in the diagonal cells, rather than principal components. In the latter method, the number of factors is sometimes too large and sometimes too small, depending on the magnitude and pattern of the correlations. However,

---

[3] A factor score is defined as the sum of products of factor loadings of variables and standard scores of the task on the variables, weighted by the inverse of the correlation matrix.

considering the large number of variables per factor in the present study, this concern is more a matter of preference than of criticism. It would be interesting to see the congruence coefficients for the factor matrices resulting from different combinations of correlations, cross-products, principal components and principal factors.

To ask a specific question of the data, is the correlation between the knowledge categories "Drug excretion" and "Biochemistry of nutrients" [reported in 3, p. 3-10, Figure 7] actually near .80? One can estimate this by multiplying their loadings on Factor I (.92 and .86). Is there really that much consistency between tasks requiring both kinds of knowledge (or not requiring either)? If the correlation, in fact, is not near the value computed from the loadings on a factor, then some further explanation is due. One possibility is that the high loadings are artifacts, resulting from the use of 1.00 rather than communality in the diagonal cells.[4]

In the footnote to Figure 7[5] the wording is more dramatic than warranted ("partake," and "tend to rise in an interrelated manner" are particularly jargon-y). A factor has no life of its own, nor does a variable. A factor is a collection of variables with regard to which the tasks are differentiated in the same--or nearly the same way. For example, with

[4] Editor's note: The actual correlation coefficient is .94.

[5] Footnote reads as follows: "Note: Factor loadings represent the degree to which skill and knowledge variables partake of the factor. Loadings are standardized and range from $\pm.00$ to $\pm.99$. Loadings of .41 or higher are shown. These variables tend to rise in an interrelated manner."

regard to Factor I reported in Figure 7 [3, pp. 3-10 through 3-14], the 272 tasks are differentiated from each other very strongly by the amount of information they require about drugs, and to a lesser extent by the amount of information they require about physiological and anatomical systems. The variables do not "rise and fall together." They are somewhat alike in the ways they serve to differentiate one task from another, in this specific set of tasks.

Considering the obvious relations among skills and the probable relations among aspects of knowledge, the choice of orthogonal over oblique rotation seems open to discussion. While the desirability of simple structure need not be re-emphasized, we should remember that Thurstone's search for simple structure among the many factors in the domain of aptitudes and achievement led him to invent oblique rotation. Here arises another tradeoff. If the purpose is to define job ladders so that they are as unlike as possible, with a minimum of lattice-relations, then orthogonal rotation is more appropriate; if one wishes to maximize the mobility by searching for as many lattice-relations as possible, then oblique rotation, especially if tasks are replicated in some representative fashion, as referred to earlier herein, would be more useful. At the present time, there are no technical restraints on the choice of rotational option; when the decision was made, computing capability was much less flexible than currently.

Regarding the determination of jobs as clusters of tasks, the grouping procedure based on loadings that is used in this report is logically sound. Again, so rapidly does technology grow, statistically oriented clustering techniques are now available which would simplify the work but probably not change the outcome. A further elaboration of the SK-task-job relation could be evolved using multiple-regression models. Ratings could be made of the involvement of each SK factor in each job; these ratings could be used as weights and multipled by the task-factor scores (or loadings) to develop a task-job index. The foregoing suggestion is based on my interest and should not be interpreted as critical of the procedure used in the study being reviewed.

PERFORMANCE

The selection of putative skills and areas of knowledge seem to me to be consistent with the job analysis literature. With regard to skills, Guilford's tri-partite model was referred to but not adopted completely, so that one reads of figural skills and classification skills which represent a rather large collection of Guilford's hypothesized factors. In this instance, however, the reduction in the set of constructs has been made on the basis of expert judgment in the health field and is thus defensible; one should expect, however, that the skills factors might be more related here than in Guilford's model.

Specifications for both domains, skills and knowledge, are usable and relevant. The coding system for knowledge is interesting and will no doubt have further payoff as curricula are developed.

The choice to use equal interval scales is sound. There is, however, a substantial difference between equal-appearing scales based on judgments and equal-interval scales resulting from intensive analysis of many specific judgments. The choice to use equal-appearing intervals in this study seems to put a large burden on whether judges can follow the plea to assign values at equal intervals. It is not clear from the report whether judges were given any orientation regarding the difference between numerical intervals and psychological distances. In contrast, Thurstone's simplest empirical method provides for computing the average of the standard deviations for two adjacent items: this average becomes the unit of distance between those items, and similarly for other pairs of items. The mean or median of the item serves to determine its place in the sequence.

For example, let us use the data presented in Gilpatrick's Table 1 (reproduced herewith). Table 2, below, presents new scale values arrived at for two of the scales using the following method: we may use the medians to array the items, and the interquartile range as an estimate of the standard deviation (within a constant); if we assume a normal distribution of judgments, and look for an overlap of distributions of about 10% between adjacent items, it turns out that we can use the sum of the two interquartile ranges (twice the average) as the distance between adjacent items.

In Table 2, the scale values are, of course, in the same sequence. For "Guiding or Steering" the major increase in interval occurs between positions 2 and 3, and between 4 and 5, with a decrease between positions 5 and 6. (Compare column (5) with column (8).) An examination of the descriptors [2, Part B, Vol. III, p. 2-20] suggests that the differences relate to degree of precision and the distinction between small number of stimuli and extremely complex external arrays of stimuli. The scaled dif-

Table 1. DATA BASE FOR SCALE VALUES USING THURSTONE EQUAL INTERVAL SCALING

Scale Name, Number, and Number of Judges[b]	Statistical Description	Items Listed in Order Presented to Judges									
		a	b	c	d	e	f	g	h	i	j
1. Frequency (15)	Median Scale Value[a]	[illegible].0*	2.0*	8.0*	0.0*	3.0*	9.0*	1.0*	7.0*	4.0*	
	Interquartile Range	1.0	0.9	0.8	0.5	1.1	0.6	0.8	0.7	0.1	
2. Locomotion (15)	Median Scale Value[a]	5.0*	9.0*	0.0*	1.5*	7.0*					
	Interquartile Range	1.9	0.6	0.5	1.0	0.9					
3. Object Manipulation (15)	Median Scale Value[a]	1.5*	0.0#	5.0*	9.0*	6.5/	3.5*	7.5*	3.0/		
	Interquartile Range	1.0	0.5	0.9	0.6	1.4	1.2	1.0	0.9		
4. Guiding or Steering (15)	Median Scale Value[a]	1.5*	4.0/	9.0*	7.0*	0.0*	[illegible].0*	5.5*	8.0/		
	Interquartile Range	1.1	1.2	0.4	1.0	0.5	1.4	1.0	0.9		
5. Human Interaction (18)	Median Scale Value[a]	5.0*	7.0*	3.0*	0.0*	9.0*	1.0*				
	Interquartile Range	1.1	1.0	1.5	0.0	0.6	0.8				
6. Leadership (22)	Median Scale Value[a]	4.5#	4.0/	1.0#	8.5#	5.5/	3.5/	6.5#	5.5/	3.0#	0.0*
	Interquartile Range	1.5	2.0	0.7	2.1	3.1	1.4	2.1	1.6	1.2	0.4
7. Oral Use of Language (18)	Median Scale Value[a]	4.0*	7.5*	2.0*	0.0*	9.0*					
	Interquartile Range	1.2	1.1	1.0	0.5	0.6					
8. Reading Use of Language (17)	Median Scale Value[a]	2.0*	7.0/	0.0*	9.0[illegible]	5.0*					
	Interquartile Range	1.2	0.9	0.5	0.[illegible]	1.6					
9. Written Use of Language (15)	Median Scale Value[a]	6.5*	5.0*	9.0*	2.0*	0.0*					
	Interquartile Range	1.1	1.1	0.8	0.8	0.5					

* Item was kept.
\# Item was edited.
/ Item was eliminated.

[a] Rounded
[b] Refers to number of judges in equal interval test.

Table 2. NEW SCALE VALUES USING INTERQUARTILE RANGES

Descriptor Item	Interquartile Ranges For Item	Lower of Pair	Higher of Pair	Distance Sum of (3) + (4)	New Scale Value	HSMS Scale Value	HSMS Scale Distance	Scale Difference (6),(7)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
				Locomotion Skill Scale				
c	0.5	0.5	1.0	1.5	0.0	0.0	1.5	0.0
d	1.0	1.0	1.9	2.9	1.5	1.5	3.5	0.0
a	1.9	1.9	0.9	2.8	4.4	5.0	2.0	0.6
e	0.9	0.9	0.6	1.5	7.2	7.0	2.0	0.2
b	0.6	-	-	-	8.7	9.0	-	0.3
				Guiding or Steering Skill Scale				
e	0.5	0.5	1.1	1.6	0.0	0.0	1.5	0.0
a	1.1	1.1	1.4	2.5	1.6	1.5	1.5	0.1
f	1.4	1.4	1.0	2.4	4.1	3.0	2.5	1.1
g	1.0	1.0	1.0	2.0	6.5	5.5	1.5	1.0
d	1.0	1.0	0.4	1.4	8.5	7.0	2.0	1.5
c	0.4	-	-	-	9.9	9.0	-	0.8

Note: From values like these, a linear transformation may be made to produce equal midpoints and ranges for all scales; this, of course, throws away some useful data regarding differences between scales.

ference in the new scale values is somewhat larger than in the procedure followed in the study. It is possible that the smaller increment between the last two positions in the initial scaling is due to "end effect," which would tend to suppress the median value more than the standard deviation.

For "Locomotion," in contrast, a slight shortening of the scale seems to occur, mostly between the second and third descriptors, but this is compensated by the increase between the third and fourth. From the descriptor content [2, Part B, Vol. III, p. 2-18] the issue is the placement of "moderate" between "low" and "high" as adjectives used in the descriptors with regard to degree of body coordination. The differences are interesting, from a psychometric point of view, but it is doubtful whether the new scale values would make any major change at the level of correlation or factor loading. If new scales are developed, however, it should be kept in mind that the proposed method of equal-interval scaling uses much more of the relevant data and, given current computing technology, would take no more effort than the simpler method of "equal-appearing" intervals.

The HSMS indexes used to assess the scales for reliability and validity are logical,although strange to me. It would be helpful to have a reference to their derivation.or,if they are original,to show their relation to Lambda or the Contingency Coefficient. Considering that the ratings are ordinal, if not completely equal-interval, the opportunity exists to use an analysis-of-variance model for reliability and also for content validity (accuracy). As reliability coefficients are notorious for instability across samples and situations, it would be prudent to check routinely for each job analyzed.

There is a confusion in a footnote of the document reporting on the pilot test[6] between variance, in the usual sense related to range, and

---

[6] Footnote reads as follows: "The six-factor solution accounts for 73 percent of the variance. (Variance refers to a statistical measure which reflects the different scale values of each of the variables as found in the tasks. The greater the range and distribution for the scales in the task data, the greater the variance.) The fact that the large number of variables are accounted for by a small number of factors with as much as 73 percent of the variance accounted for is considered statistically very satisfactory." [3, p. 3-7].

variance in the factor-analytic sense, which really means covariance. Each factor can be used to reproduce a partial correlation matrix; in this case, one could obtain six partial-r-matrices. If these were added cell by cell, the resulting sum would closely approximate the correlation matrix between the variables initially computed from the task-by-SK matrix. The amount of "variance" in each of the partial matrices is the sum of squares of the loadings of variables on that factor. The total sum, called the trace, is equal to the number of variables when principal components are used; otherwise it is the sum of the communalities of the variables. The proportion of variance is thus the trace of the partial matrix divided by the trace of the initial matrix.

Davis' [1] discussion of robustness can be considered reassuring, given that sample size is interpreted as degrees of freedom. In the HSMS test data, degrees of freedom, computed as the number of tasks (replications) less 2, less number of variables, less number of derived parameters (factors) was 121. One can be reasonably confident about the eigenvalues here, but somewhat less so about the specific values of loadings. In my reading of the psychometric literature, the lowest ratio of replicates to variables I have come across is 5 to 1, a good bit larger than in the present study. However, it is doubtful that the clusters, in the large, would differ much if the number of replications were increased or the number of variables decreased, if that were done proportional to the present number and representativeness of variables and tasks.

On the other hand, were one to increase the number of tasks as noted earlier, to reflect the relative frequency of tasks across jobs, and if one were to reduce the number of knowledge categories by consolidating rationally at some more general level of description, say the level of 5 or 6 digits in the coding system, then one could anticipate greater confidence in the results, with perhaps a few differences which would be interesting. Again, I hasten to say that I am suggesting a trade-off, and not making a specific criticism. The results, even though from a single institution, are--to say the least--exciting and hope for the applicability of the method in other similar studies is well warranted.

I must note in passing that the factor analysis of dichotomous data has long been practiced and studied in the psychological literature. Phi is, after all, a product-moment coefficient and, if the dichotomies are made near the medians of the distributions, it is no great effort to presume an underlying bivariate normality for each pair of variables.

It is discouraging to note that the "loadings" of tasks on factors are not comparable from one factor to another, as noted in footnote c, Figure 16.[7] This situation would not arise with factor scores.

There is a frequent error of interpretation of factor loadings--frequent among many practicioners in the field--to which this study seems to fall prey. This error is to interpret a loading as one would a mean, e.g., to infer that a high loading of an SK variable on a factor implies

[7] Footnote reads as follows: "Loadings represent the degree to which task partakes of factor. Loadings are not standardized, and sign has no intrinsic meaning except for change from high on one, passing through zero, to opposite sign, as continuous hierarchy." [3, p. 3-31].

that a high degree of what the variable refers to is required, in some absolute sense. Not so: loadings are based on correlations, which are based essentially on rank order similarities among variables, i.e., the variables that are correlated differentiate among the tasks in the same way. Relative standing is all one gets from correlations (one can get closer to absolute values in the analysis of cross-products), and it is necessary to go beyond the loadings to factor scores to see anything about actual levels on whatever scale is being used. In the present study, it is likely that the second-mode values, relating tasks to factors, are not too different from factor scores and that their interpretation is appropriately made. However, the specification that correlations were used, and the lack of sufficient detail about the two-mode computations (were correlations used to compute the mode 2 factors as well as mode 1?) leaves a residual ambiguity that is nettling,but not sufficiently disturbing to rouse a strong criticism. Again, it is difficult to argue with the results.

Here is an example of the interpretation problem. If a variable has a high degree of involvement with <u>all</u> tasks, but its variance is small relative to error of measurement (lack of reliability), then it will probably have low correlations with other variables, and thus low loadings on all factors. Similarly, a variable with small involvement with <u>all</u> variiables will have low loadings on all factors. Obviously, given only that a variable has low loadings on factors, one cannot tell which is the case with regard to involvement level, but only that there is little difference in level of involvement from one task to another.

To look at the question of proficiency, one must have two inputs: factor loading of variable, and score of task on variables. In general, if a task has the same score on two variables, its relation to the factor(s) involved depends on the size of the loading of the variable(s) on the factor(s). If two variables have the same value for loading, then the task-factor relation depends on the differences in the task-variable score. A positive factor score means only that both loading and score on variable had the same sign (+ or -); a negative factor score means only that the sign of the loading differed from the sign of the task on variable score. The sign situation is further complicated by the factor rotation rule which, essentially, assigns a + or - sign to loadings on a factor depending upon whether the last rotation was to the left or to the right. Because this direction does not affect the structure, in that the correlation matrix can be reproduced with either sign for a factor, it is permissible to change all signs of all variables on a single factor; +'s become -'s and -'s become +'s. This change is often convenient when one wishes to interpret factors as ways of differentiating observations, or, in HSMS context, when one wants to describe SK factors as ways of differentiating tasks. When one wishes to interpret factor scores, or "loadings" from two-mode analysis, corresponding changes may be made. With factor scores, the change is direct, simply the changing of signs of factor scores for all factors for which the variable signs were changed. With two-mode analysis, because of the inverse transformation, it is not so apparent that a simple change will suffice. This point needs empirical study.

To comment directly on the data, it appears to me that the task hierarchies developed for the factors are reasonable [3, Figure 16, p. 3-31], in that certain groups of tasks seem related to certain skills and knowledge groups. Whether this is the optimal definition of a "job" may be another question. Again the question of comparability of tasks across factors becomes critical for interpretation of the total array.

Despite the issues raised above, I am struck by the consistency and apparent substantive coherence of the results,especially the task hierarchies. What might be revealed when the few obscurations are removed?

SUMMARY

I sense that there is a great effort underway, for which sound strategy at the highest levels has been laid out. My major concern is that the subordinate decisions, and the execution of the strategy, particularly in some of the more elaborate statistical procedures, seems to hav proceeded by fiat, for convenience. On the other hand, when "it works," a method certainly deserves further application.

In my critique, I pointed to three trade-offs:

1. Inclusion of tasks on a more representative basis rather than just once, at the cost of using factor scores instead of two-mode factor analysis.

2. Considering oblique rotations rather than orthogonal, at little cost but with the implication that the goal is to maximize rather than minimize lateral relations among tasks across jobs.

3. Reducing the number of SK variables through a rational consolidation, losing some specificity but gaining greater confidence in the stability of the statistical estimates, due to a larger ratio of replicates to parameters.

In addition, I suggested ways in which the scaling might be made more precise and the relation of jobs to skills and knowledge made more formal, using a multiple-regression model.

Technology moves rapidly, and it must be said that the decisions, made some five to seven years ago, were consistent with the technology as it was then. Current options allow more refined approaches to these problems, but no basic change in strategy is indicated.

Finally, I would be remiss not to report my feelings of excitement and challenge at this new vista in job analysis; my comments are to be interpreted as my attempt to make a really good thing a little better.

REFERENCES

1. Davis, Earl E. "Some Notes on the Statistical Method Utilized by the Health Services Mobility Study." (Edited by E. Gilpatrick), 1976, (contained herein).

2. Gilpatrick, Eleanor. A Job Analysis Method For Developing Job Ladders and for Manpower Planning. Research Report No. 3, Parts A, Volumes I and III; Part B, Volumes I and III; Part C, Volumes I and III. New York: Health Services Mobility Study, 1971. (Documents not currently available; to be revised.)

3. Gilpatrick, Eleanor. Suggestions For Job and Curriculum Ladders in Health Center Ambulatory Care: A Pilot Test of the Health Services Mobility Study Methodology. Research Report Nos. 4 and 5. New York: Health Services Mobility Study, 1972.

4. Gilpatrick, Eleanor. An Introduction To The Work of The Health Services Mobility Study as of April, 1975. Technical Report No. 13. New York: Health Services Mobility Study, 1976.

5. Harman, Harry, H. Modern Factor Analysis. (2nd Ed., Rev.) Chicago: University of Chicago Press, 1967.

A CRITICAL REVIEW OF THE METHODOLOGY AND STATISTICAL TREATMENT OF DATA IN THE TASK ANALYSIS AND CAREER LADDER DESIGN OF THE HEALTH SERVICES MOBILITY STUDY[1]

By Mark I. Appelbaum[2]

INTRODUCTORY COMMENTS

Any critical review of statistical and methodological procedures must proceed from the context in which the techniques were applied. Statistical and methodological techniques are rarely in and of themselves either correct or incorrect, but rather depend upon their particular usage. This fact, which is commonly recognized in the physical sciences (e.g., the differential methods of chemistry vs. chemical engineering), is often ignored in the behavioral and social domain where arguments abound concerning the appropriateness of certain techniques in absolute terms.

It is, perhaps, unfortunate that workers in the social sciences have failed to recognize the distinction between pure and applied research and hence have used the same critical standards for both classes of work without regard to the appropriateness of such a decision.

---

1 This document was completed by Dr. Appelbaum in July, 1976, and is in response to a request for review of the Health Services Mobility Study (HSMS) method. Documents supplied for review purposes included the Davis paper [2] (which precedes this), listings of the HSMS computer programs requested by the author, early HSMS documents not currently available [3], Research Report Numbers 4 and 5 [4], Technical Report No. 13 [5], and current HSMS scales not yet published. Dr. Appelbaum discussed aspects of the methodology with Eleanor Gilpatrick, Director of HSMS. Numbers in brackets refer to references listed at the end of this paper.

2 Dr. Appelbaum is Associate Professor of Psychology and Associate Dean of the Graduate School, University of North Carolina at Chapel Hill. He is also involved in research on National Assessment of Educational Progress, and a consultant for National Science Foundation and Behavioral Technology Consultants.

It is often the case, however, that the highly precise methods of the laboratory are too finely tuned for the applied piece of research, just as the highly sensitive torsion balance of the chemist is an inappropriate instrument for the chemical engineer. Conversely, the cruder methods which might be appropriate for applied research are not necessarily sensitive enough for tasks of the pure researcher. It is therefore mandatory that the research instruments be judged in the context of their use.

SCOPE AND ORGANIZATION OF THE HEALTH SERVICES MOBILITY STUDY

At the onset it must be noted that the scope and purpose of the Health Services Mobility Study (HSMS) is nothing short of monumental. Even with the limited classes or areas included, the task of organizing, measuring, and interrelating the many specific tasks, skills, and knowledge requirements is a vast undertaking. Particularly is this the case in light of the very limited set of techniques available for evaluation research and the even smaller number of well executed studies to serve as models. Taken as a whole I find the technical and methodological portions of the study indeed well conceived and executed.

EVALUATION OF THE METHODOLOGICAL COMPONENTS OF THE STUDY

There are three major aspects of the study which afford the possibility of a methodological evaluation. These are (1) the initial selection and organization of the task, skill, and knowledge components; (2) the measurement of these components; and, finally, (3) the interrelating of the components in order to identify a simplifying organization and to allow the development of job and curriculum ladders. The first of these is outside of the area of competency of this reviewer and shall, consequently, be left untouched.

Measurement of the Various Task, Skill and Knowledge Components

In the present study the process of measurement is fundamentally one of scale construction. The methods employed to develop the various scales are, perhaps, the most extensively documented in the entire report. In general, well known and highly accepted classical methods were employed--namely, constructions using equal interval methods. The quality of the resulting scales (and their consequent utility) depends, of course, upon the ability of the judges to construct truly equal interval scales. Being unfamiliar with the judges or their degree of training, it is difficult to assess the degree to which they were able to accomplish this task. Alternative methods of scale construction do not seem to be applicable in this study--the technical requirements being untenable (e.g., methods based upon paired comparisons).

In terms of the reliability of the scales, I was unfamiliar with the methods employed, but saw nothing which appeared to be, on the surface, inappropriate.

Interrelating of the Components In Order To Identify a Simplifying Organization

This section deals largely with the utilization of the "Two Mode" Factor Analysis procedure and issues attendant thereto. Discussion of the

Davis paper [2] [which appears in this document] is also included.

"Two Mode" Factor Analysis

Among the difficulties involved in understanding the "Two Mode" approach, is that (a) there is little existing documentation of the approach and (b) the bulk of the procedure is defined by the program which is used to produce the results. Perhaps a few words about the procedure as applied to the Health Services Mobility Study would be of some help. It is first necessary to understand that the basic input data are unlike those usually employed in factor analytic studies. Rather, what is employed is a matrix which has as its rows the various tasks (T) and as its columns the various skill/knowledge measures (S). The data entries are the judgments (actually a single "average" or "consensus" judgment) as to "how much" of a particular skill or knowledge is required for a particular task. The judgments are (as described in several of the reports) the expert ratings of trained judges for a rather idealized task, but one which corresponds to actually occurring tasks.

From this single basic data matrix two conceptually different but necessarily related "correlation" matrices are formed; one is obtained by intercorrelating tasks over the various skill/knowledge measures, the other by intercorrelating the skill/knowledge measures over tasks. (In the more usual application of this procedure, usually referred to as a Tucker-Messick Points of View Analysis, these would correspond to the correlation of individuals over variables (also referred to by

Cattell [1] as the Q technique), and the usual correlation of variables over individuals (Cattel's R technique [1]).

These matrices can be thought of as representing tasks in a skills/knowledge space and skills/knowledge in a task space. By using the results of the well known Eckert-Young theorem it is possible to approximate tne original rating matrix as the product of two conceptually different sets of relations based upon these two spaces. Specifically, Eckert-Young states:

$$ {}_{n}X_{m} = {}_{n}U_{k} * \lambda_{k} * {}_{k}V'_{m} $$

where ${}_{n}X_{m}$ is the original matrix of ratings; ${}_{n}U_{k}$ are the k characteristic vectors of X'X (the skills/knowledge measures correlated over tasks); ${}_{k}V'_{m}$ are the k characteristic vectors of XX' (the tasks intercorrelated over skills/knowledge measures), and λ is the common diagonal matrix of the characteristic roots of X'X and XX' which must be identical.

The first phase of the "Two Mode" program is to then extract from the ${}_{n}X_{m}$ matrix, the three matrices ${}_{n}U_{k}$, λ , and ${}_{k}V'_{m}$.

Having these three "basic" working matrices and noting that ${}_{n}P_{k} = {}_{n}U_{k}\lambda$ are the principal components of the skills/knowledge correlation matrix, one may then proceed to find a particular representation, there being infinitely many with respect to the reproduction of ${}_{n}X_{m}$ (the factorial invariance problem). Thus one may choose, for instance, to use the varimax representation of the principal components of the skills/knowledge correlation matrix as a starting point for a "nice" representation.

Let T represent the $k \times k$ non-singular transformation which achieves the varimax rotation. (We eliminate subscripts in this presentation.) Thus:

$$R = PT$$

where R is the varimax rotated solution. (T is analytically determined by a varimax program.) Since the Varimax Transformation Matrix is non-singular, T^{-1} exists.

We may now write:

$$\begin{aligned} X &= U \lambda V' && \text{(Eckert-Young)} \\ &= PV' && \text{(definition of principal components)} \\ &= PTT^{-1}V' && \text{(since } TT^{-1} = I\text{)} \\ &= (PT)(T^{-1}V') && \\ &= R(T^{-1}V') && \text{where R is the varimax representation.} \end{aligned}$$

Thus, if we wish to use the varimax representation of the principal components of the skills/knowledge correlation matrix, we must also transfer the principal components of the task correlation matrix in order for the equality to hold. As can be easily seen above, the required transformation is T^{-1} (this is the so-called, and mysterious, counter-rotation). It is the second phase of the program which achieves this rotation and "counter rotation."

The PCVARIM and "Two-Mode" Programs

The complete computer programs as supplied by HSMS were reviewed by visual inspection. While it would have been preferable to actually make the programs operational and run test data on them, this option was judged impractical in terms of the time and personnel involved.

Much of the "Two Mode" program listing is simply system overlay and input/output routines which do not concern us in this evaluation. The functional portion of the program is actually a fairly standard common factor analysis program (or principal components program, depending on how the communalities problem is handled). The major computational routine, the eigenvalue/eigenvector routine, is the rather dated and out-of-fashion "HOW" routine using tridiagonalization. While this routine is known to produce certain problems under rather unusual conditions (linearly independent but nonorthogonal vectors when there are roots of multiplicity greater than one), it is highly unlikely that this situation would occur in this particular application.

The second mode of operation, the rotation and counter-rotation of the basic components matrix, while unusual in factor analysis programs, is based upon well known methods of rotation and should cause no unusual problems. This method, of course, offers no solution to the basic invariance of factor score problem--but, on the other hand, adds no additional problems which cannot be said of many other routines.

In general, one can have reasonable confidence in the numerical results of the program. Given sufficient time and funds, however, one might wish to introduce the newer and less problematic numerical solution to the eigen problem.

Alternative Methodological Strategies

Given the basic task at hand, one might ask if alternative methodologies are available which might rather have been employed. Taking as an oversimplified statement that the goal was to organize the task-skill components of selected jobs within a particular health delivery unit, several alternative methodologies suggest themselves. These may be divided into two general classes: (a) factor analytic, (b) multidimensional scaling.

Factor Analytic Techniques

While there are literally hundreds of variations on the basic "common factor analysis," all of these have certain common features which make them different from the Principal Components analysis actually employed. From the onset the goal of Principal Components was the summarization of the total variance in a test space into a smaller number of orthogonal components. The goal was simply to summarize. All of the factor analytic techniques, however, have a goal of first reducing the total variability into a smaller "common space" variability (the communality problem) and to then find a representation in that reduced space which maximally reproduces the intercorrelations among the manifest variables. Many of the problems currently discussed by factor analysts center around those problems.

In terms of the goals of the HSMS prcject, the Principal Components approach seems, to this reviewer, the more straightforward approach and the one that most nearly will achieve the desired end product. It might also be mentioned that Principal Components are far easier to work with and interpret. Save the problem of whether to analyze cross-products, variance-covariance, or correlations, the results are completely determinate--Principal Components being the solution to a statistical maximization problem. While it would not be valid to select a method based simply upon ease, given the fulfillment of the basic project goals by Principal Components it seems a wise selection.

Multidimensional Scaling

Major advances have recently been made which allow considerable flexibility in creating spatial mappings of both stimuli and individual "points of view." The basic data requirements of this approach are, however, such that it would seem inappropriate to consider this methodology. To use this method it would be necessary to have paired judgments of "task similarity" for all tasks in all jobs. This would require judges who had familiarity with all jobs and all tasks within jobs as well as the ability to assess their similarity. Given the breadth of the jobs included and their complexity, it seems highly unreasonable that such judges would be available.

Comments on the Davis Paper [2]

The presentation given in the Davis paper [2] (presented herein) gives an adequate but not detailed review of the Two-Mode Factor procedure--one not well documented elsewhere. It would be helpful if more detail were added. It should be emphasized, however, that the unique portion of the technique is not so much a part of the classical factor problem as one of representation (actually related to several classical problems; namely those of factor scores and congruences). While apparently appropriate for the Health Services Mobility Study, the technique should not be used, in general, without great care and understanding.

There are a few issues raised which deserve some comments, although they have little direct impact on the Health Services Mobility Study. The first of these concerns is the selection of the correlation matrices as the unit of analysis. While this discussion is rather casually presented in the Davis paper, and while the decision is, most probably, the correct one, it is still worth noting that there is a rational decision. Since the scales which are being inter-correlated are themselves rather arbitrarily constructed scales (in the sense that adding a constant or using a constant multiplier would do no violence to the scales per se), one would not wish the resulting solution to be dependent upon the means or standard deviations of the scales. Had the cross-products matrices been used, the solution would have been dependent upon both the means and standard deviations; had the variance-covariance matrices been used, the solution would have been dependent upon the standard deviations. Since neither of these were desired, the choice of the correlation matrix was indeed appropriate--if not defended.

The second issue of concern deals with the rather oblique treatment of factor analysis versus principal components. While it might be of some comfort to note that these two conceptually different techniques have similar results, I do not find it pleasing to base the "justification" of using principal components on that fact. It seems to me that principal components, on a purely theoretical level, does exactly what was desired and should thus be the analysis of choice on that basis exclusively. The goal was to summarize the total variation in the system; there was no concept of common space variance. On that basis alone one is perfectly justified in using components--no additional justification is needed.

The final comment pertains to oblique versus orthogonal representation. To this, two comments are relevant. First, principal components are, by definition, orthogonal. Second, a representational basis is completely a matter of taste.

SUMMARY

Given the orientation stated in the Introductory Comments, it is the belief of this reviewer that, but for a very few technical points, the basic methodological approach taken in the Health Services Mobility Study is sound. While other investigators may have chosen other approaches, there are no problems, beyond those of personal taste, which would force the conclusion that some other approach should be preferred. It is particularly important to note that when dealing with an area so complex as this, the most one can reasonably hope for is that a few basic and important findings should result. Certainly this has been achieved. It is perhaps equally important to note that there are years of additional work (and large sums of additional funds) required before anything approaching completeness could be hoped for.

REFERENCES

1. Cattell, R. B. Handbook of Multivariate Experimental Psychology. Chicago: Rand McNally, 1965.

2. Davis, Earl E. "Some Notes on the Statistical Method Utilized by the Health Services Mobility Study." (Edited by E. Gilpatrick), 1976, (contained herein).

3. Gilpatrick, Eleanor. A Job Analysis Method For Developing Job Ladders and For Manpower Planning. Research Report No. 3, Parts A, Volumes I and III; Part B, Volumes I and III: Part C, Volumes I and III. New York: Health Services Mobility Study, 1971. (Documents not currently available; to be revised.)

4. Gilpatrick, Eleanor. Suggestions For Job and Curriculum Ladders in Health Center Ambulatory Care: A Pilot Test of the Health Services Mobility Study Methodology. Research Report Nos. 4 and 5. New York: Health Services Mobility Study, 1972.

5. Gilpatrick, Eleanor. An Introduction to the Work of The Health Services Mobility Study as of April, 1975. Technical Report No. 13. New York: Health Services Mobility Study, 1976.