

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

2

NAVAL POSTGRADUATE SCHOOL Monterey, California

AD A137170



DTIC
ELECTE
JAN 24 1984
S B

THESIS

DTIC FILE COPY

NONPARAMETRIC STATISTICS TEST
SOFTWARE PACKAGE

by
Philip J. O'Brien, Jr.
September 1983

Thesis Advisor: Stephen Paek

Approved for Public Release; Distribution Unlimited

84 01 23 022

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
	AD-A137770	
4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED	
Nonparametric Statistics Test Software Package	Master's Thesis September 1983	
7. AUTHOR(s)	6. PERFORMING ORG. REPORT NUMBER	
Philip J. O'Brien, Jr.		
9. PERFORMING ORGANIZATION NAME AND ADDRESS	8. CONTRACT OR GRANT NUMBER(s)	
Naval Postgraduate School Monterey, California 93943		
11. CONTROLLING OFFICE NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
Naval Postgraduate School Monterey, California 93943		
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	12. REPORT DATE	13. NUMBER OF PAGES
	September 1983	249
	15. SECURITY CLASS. (of this report)	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
Approved for Public Release, Distribution Unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
Nonparametric Tests	Wilcoxon	
Binomial	Kolmogorov	
Quantile	Smirnov	
Distribution Free Tests		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
<p>This thesis provides the computer software to perform 24 of the most common nonparametric tests, and a text explaining how and why to use that software.</p> <p>Nonparametric tests are valuable to experimenters and operations analysts for three reasons: ease of explanation to non-statisticians, simplicity of computation, and applicability to data</p>		

sets which cannot be analyzed by parametric tests. Reasons for unsuitability of parametric test include data of nominal or ordinal level of measurement, lack of common variance, or lack of normal distribution of the underlying population.

There are two programs provided: Lochinvar, an interactive Pascal program, and Crunch, a Fortran program. Lochinvar is designed to prompt and screen a user's entry of data and options. Crunch performs calculations; it is suitable for use in whole or part as a subcomponent of other programs.

Accession to the	
NTIS	<input checked="" type="checkbox"/>
DTIC	<input type="checkbox"/>
Unann.	<input type="checkbox"/>
Just.	<input type="checkbox"/>
By _____	
Distribution/ _____	
Availability Codes	
Dist	Avail and/or Special
A-1	



Approved for Public Release, Distribution Unlimited

Nonparametric Statistics Test Software Package

by

Philip J. O'Brien, Jr.
Major, United States Marine Corps
A.B., Stanford University, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September, 1983

Author:

Philip J. O'Brien, Jr.

Approved by:


Steph J. Park THESIS ADVISOR

Charles W. Hutchins Jr. SECOND READER

Paul Walker
Chairman, Department of Operations Research

K.T. Marshall
Dean of Information and Policy Sciences

ABSTRACT



This thesis provides the computer software to perform 24 of the most common nonparametric tests, and a text explaining how and why to use that software.

Nonparametric tests are valuable to experimenters and operations analysts for three reasons: ease of explanation to non-statisticians, simplicity of computation, and applicability to data sets which cannot be analyzed by parametric tests. Reasons for unsuitability of parametric test include data of nominal or ordinal level of measurement, lack of common variance, or lack of normal distribution of the underlying population.

There are two programs provided: Lochinvar, an interactive Pascal program, and Crunch, a Fortran program. Lochinvar is designed to prompt and screen a user's entry of data and options. Crunch performs calculations; it is suitable for use in whole or part as a subcomponent of other programs.

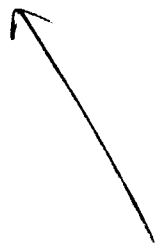


TABLE OF CONTENTS

I.	INTRODUCTION	8
II.	COMPARISON OF PARAMETRIC VERSUS NONPARAMETRIC TESTS	10
III.	THE COMPUTER PACKAGE	18
IV.	ROSTER OF AVAILABLE NONPARAMETERIC TESTS ...	26
V.	BINOMIAL TEST.....	31
VI.	QUANTILE TEST.....	36
VII.	MANN WHITNEY TEST.....	41
VIII.	COX STUART TEST FOR TREND	44
IX.	SIGN TEST	48
X.	MCNEHAR TEST FOR SIGNIFICANCE OF CHANGE	52
XI.	CHI-SQUARE TEST FOR INDEPENDENCE (R*C CONTINGENCY)	57
XII.	CHI-SQUARE GOODNESS OF FIT TEST	62
XIII.	WILCOXON SIGNED RANK TEST	66
XIV.	NONPARAMETRIC CORRELATION	70
XV.	NONPARAMETRIC REGRESSION	75
XVI.	MONOTONIC REGRESSION	80
XVII.	MEDIAN TEST	84
XVIII.	KRUSKAL WALLIS TEST	89
XIX.	SQUARED RANK TEST FOR EQUAL VARIANCE	93
XX.	HARTLEY TEST FOR EQUAL VARIANCE	97
XXI.	FRIEDMAN TEST AND DURBIN TEST	100
XXII.	QUADE TEST	105
XXIII.	COCHRAN TEST FOR RELATED OBSERVATIONS	109
XXIV.	KOLMOGOROV TEST	114
XXV.	LILLIEFORS TEST	119
XXVI.	SHAPIRO WILK TEST FOR NORMALITY	122
XXVII.	SMIRNOV TEST.....	125
XXXVIII.	CRAMER VON MISES TEST	128
	LIST OF REFERENCES	131

APPENDIX A	TUTORIAL: LINKING TO AND USING THE COMPUTER PACKAGE	132
APPENDIX B	THE PROGRAM LOCHINVAR	139
APPENDIX C	THE DATA FILE	144
APPENDIX D	THE OPTIONS FILE	149
APPENDIX E	THE PROGRAM CRUNCH	153
APPENDIX F	FORTRAN SOURCE CODE: CRUNCH	158
APPENDIX G	PASCAL SOURCE CODE: LOCHINVAR	211
INITIAL DISTRIBUTION LIST	248

ACKNOWLEDGEMENT

The greatest debt which I, and contemporary students of nonparametric statistics, owe is to W. J. CONOVER, who wrote a lucid and succinct text on the subject. This both encouraged and facilitated study of nonparametric tests; my thesis would have been immeasurably more difficult without his book.

My thesis advisor, LTC Stephen PAEK, USA, taught me the subject matter and channeled my efforts towards making a usable computer package. Both he and my second reader, CDR Charles HUTCHINS, USN, made numerous suggestions for improved syntax, sentence structure and spelling throughout this text. With cunning and perseverance, I have managed to sneak a few errors past their watchful eyes. I not only claim them as my own fault, but leave their correction to the student as an exercise.

My father began drilling me in mathematics before I went off to kindergarden. Both he and my mother made sacrifices to see that I was expensively educated. More importantly, they encouraged me to use at least some portion of my God-given talents.

I. INTRODUCTION

This thesis is intended to help non-statisticians use nonparametric tests to their fullest. The text explains when nonparametric tests are preferable to parametric tests, how to choose the most appropriate test, and how to use the nonparametric software package.

Parametric tests are often the preferred form of statistics because of their acceptance in the academic world, the availability of computer support, and flexibility in model building. Nonparametric tests provide a valuable alternative to parametrics, because they may be applied to data which, for any one of a number of reasons, cannot be analyzed with parametric tests. The simplicity of nonparametric tests often make the test statistics easier to compute and test results easier to explain. Section II gives a detailed comparison of parametric versus nonparametric tests.

When confronted with the same hypothesis, various parametric tests give consistent results because each test extracts the same information from the data set. Nonparametric tests may yield varying results with the same data, because different tests extract different information from the data set. To overcome this potential problem, the most appropriate nonparametric test should be chosen to analyze the data set. Section IV gives guidance on selecting the best test.

In order to make using nonparametric tests as easy as possible, two programs were written to perform the most common 24 nonparametric tests. A general description of how to use the programs is contained in section III. Sections V through XXVIII contain descriptions of each of the tests,

and detailed instructions on their use. The appendices contain more detailed information about the programs. Appendix A is a tutorial which depicts use of the programs to perform a test on a data set. Appendices B through E describe the two programs and two files. Appendix G contains the interactive Pascal program, Lochinvar, while Appendix F contains the Fortran program Crunch.

II. COMPARISON OF PARAMETRIC VERSUS NONPARAMETRIC TESTS

This section contrasts the nonparametric statistical test with parametric statistics. To do this, there is a short summary addressing various levels of measurement and a brief discussion on the relationship between the normal distribution and interval data. Finally there is a comparison of the advantages of the two classes of tests. This will help the reader decide whether to use the tests of this text.

A. LEVELS OF MEASUREMENT

Statistics manipulate raw data to extract whatever information that data contains. Level of measurement indicate how much information is contained in the data, and how that information may be manipulated. Although formally there are four levels of measurement, as defined by Stevens [Ref. 2], there are only three which matter when choosing the test: nominal, ordinal, and interval. As a consistent rule any information contained in data of a lower level of measurement is contained in data of a higher level; any manipulation permitted on a lower level is permitted on a higher level.

Nominal data elements contain very little information: only membership in a subset of the total data set can be determined. Manipulation of the nominal data sets is limited to counting the number of elements in each subset.

Ordinal data elements contain information relative to other elements: element A is either less than, greater than, or equal to each other element in the set. However, the question "greater than by how much" cannot be answered. The data set may be ranked.

Interval data elements contain more information. Not only can element A be compared to other elements, but inequalities can be quantified; e.g., $A > B$ implies that there exists a k such that $A = B + k$, and that the k can be found by $k = A - B$. The data set can be manipulated with normal arithmetic operations, such as addition, division, or raising to a power.

B. LEVEL OF MEASUREMENT AND DISTRIBUTION ASSUMPTIONS

The normal distribution is completely specified by two parameters: expected value and variance. In estimating those parameters from a data set, arithmetic operations are used, and intervals are treated as though they were meaningful. For example, the expected value is estimated by the sample mean, which is found by summing the data elements and dividing by the number of elements. The variance is found by summing the squares of the intervals between each data element and the sample mean, and dividing by the number of elements minus one. Because of the use of arithmetic operations and intervals in computing its parameters, the normal distribution seems to require interval data sets.

Parametric tests assume that a data set contains elements with an interval level of measurement drawn from a normally distributed population. Comparison of two data sets usually require the assumption of equal variance in the two populations from which the samples were drawn. There is a fair amount of literature concerning how valid parametric tests are when any of these assumptions are violated; a large, vocal group of statisticians would say that such tests produce valid results even when the assumptions are decidedly not met. But meeting all of the assumptions of parametric tests are the guarantee of validity when using such test.

Nonparametric tests make no assumptions about the underlying distribution of the population from which the data is drawn. Additionally, there are nonparametric tests to analyze data with nominal level of measurement or higher. Because some nonparametric tests require only nominal or ordinal data, and don't make assumptions about population distributions, the power of those tests may be less than that of parametric tests-- assuming that the data set justified use of parametrics in the first place.

C. ADVANTAGES AND DISADVANTAGES OF CLASSES OF TESTS

The advantages and disadvantages of parametric tests are the mirror image of those of nonparametric test. Therefore, the advantages of each class of tests is given.

1. Advantages of Parametric Tests

The general advantages of parametric tests are wide acceptance of methodology, consistency of results, and capability to handle multiple variables in systematic model building.

a. Wide Acceptance

The wide acceptance of parametric test has a number of advantages. Statistical software packages contain a very complete set of parametric tests, and only a small--- although increasing over time---number of nonparametric tests. Table of the theoretical distributions needed to judge test statistics (the normal, student's t, chi-square and F) are extensive and ubiquitous. The terms of the test are known to a majority of persons trained in the social or physical sciences; whether the terms are fully understood may not influence the value of their recognition.

b. Consistency of Results

Different parametric tests give the same conclusions, given the same null and alternative hypotheses. Parametric tests all extract the same information from the data, and merely rearrange it for different purposes. The parametric distributions are all derived from the normal.' With related test statistics and similar test distributions for critical values, it is not surprising that different techniques give consistent answers to the same question.

Nonparametrics may provide different results to the same question on the same data set, because different tests extract different types of information from the data set. For example, with paired data elements, the sign test considers only which member of the pair is larger; the signed rank test needs to know not only which is larger, but also the magnitude of that difference. Consistency of results is therefore a possible advantage for parametric tests. However, this advantage can be minimized by selecting the best (most powerful) nonparametric test possible, as discussed in a later section.

c. Versatility in Model Building

The final, and probably most important, advantage of parametrics is their versatility in explaining very complex interactions in data. Analysis of variance can test differences in treatment effects, multiple blocking effects, and interaction among treatments and blockings. Regression can be made to do analysis of variance, (by using nothing but dichotomous variables), but is even more flexible, by building models using variables with continuous values rather than only a fixed number of treatment levels.

' The t is a normal distribution in which the lack of knowledge of the population expected value is corrected for by lowered kurtosis; the chi-square distribution of degrees of freedom n is the sum of squares of n standard normal distributions; the F is the ratio of two chi-squares.

This versatility of parametrics is based on the very strong assumptions made about the populations from which the data is drawn: literature beyond the elementary statistics textbook level contains many ways to determine if the assumptions are met, and what to do when they are not met.

Nonparametric tests, on the other hand, assume nothing about the underlying distribution, and often work with nominal or ordinal data. Not surprisingly, only very straight-forward answers can be given for any one test. A clever application of a sequence of tests to the same data set may, however, yield answers to a number of different questions, and provide some inferences on interaction.

2. Advantages of Nonparametric Tests

The advantages of nonparametric tests are simplicity and applicability.

a. Simplicity

Nonparametric tests' simplicity allows for relatively easy calculation of the test statistic and critical value for a null hypothesis. This simplicity also allows nonparametric tests to be explained to, and defended in front of, persons who are not trained in statistics.

The test statistic for most nonparametric tests can be calculated by hand. For example, a set of 20 pairs of data elements (X, Y) can be compared for equality of expected value ($E(X) = E(Y)$) by either the t-test or the nonparametric sign test. The t-test needs sample means and variances, so some arithmetic operations are required: four sums of 20 terms each, 40 squaring of numbers, four divisions. In contrast, the sign test requires counting in how many pairs the X element is greater than the Y element. The contrast is not always so one sided: as the amount of information extracted from the data increases, so does complexity of computation. For example, rank-order nonparametric tests

require ranking of data elements, which can become tedious in large data sets. However, the process of ranking tends to be self-correcting, because an unranked or misranked data value will stand out in a simple sequential check of the ranks. Arithmetic mistakes in squaring, summing, et cetera, may not be so obvious unless the operation is redone. When computations are made by hand, therefore, nonparametric test statistics are easier to find.

Since hand-held calculators are even more ubiquitous than parametric tables, this advantage of nonparametric test is less important than it once was. If the number of keystroke errors is proportional to the number of numbers entered in the calculator, though, the simplicity of nonparametric tests may still be an advantage.

Simplicity could also be an advantage in the rare case when tables of distribution are not available. Many nonparametric test distributions, especially the simpler tests, rest on the binomial distribution. For small sample sizes, the binomial can be calculated by hand; with a small calculator, a distribution for sample size of 20 or 30 can easily be calculated. The normal, student's t, chi-square and F distributions lack closed forms for their cumulative distribution functions. Numerical integration--even if the density functions were remembered or available--would be challenging with a calculator (since all but the normal require one or more gamma functions) and probably is beyond the realm of paper and pencil mathematics.

Simplicity is also a virtue in explaining test results. Comparison and rankings are fairly easy to follow (although sometimes difficult to perform), while the arithmetic operations of parametric tests are less transparent. Most non-statisticians have had experience in tossing coins, rolling dice or some games of chance. Analogies can be made from those experiences to the evaluation of test results.

In the previous example of 20 pairs (X,Y), with the sign test producing a test statistic of 4 (i.e., in four pairs, X>Y), an analogy could be made to the odds that a coin were fair if it came up heads four times in 20. Explaining that same data set with the t-test would require mention of standard deviations and pooled variances, and perhaps drawing of overlapping bell-shaped curves. An audience predisposed to disagree with the test results might be hard to convince with parametric tests.

b. Applicability

Nonparametric tests can be used on a larger number of data set than can parametric tests, because of the lack of assumptions about the distribution of the underlying population, and the ability to handle ordinal or interval data. This advantage is more vital in the "real world" than in academia, for three general reasons:

(1) An analyst is sometimes handed data sets to do something with; for any one of a number of reasons, parametric tests may not be applicable. Most commonly, the form in which the data was collected may fail to meet the underlying requirements of parametrics, but meets the requirements of nonparametrics.

(2) The cost of gathering interval data is prohibitive, while ordinal or nominal data can be gathered at low cost. For example, on a field exercise, uncalibrated eyes can generally judge distances greater than or less than, but some sort of instrumentation must be used to determine the exact magnitude of the difference. The expensively equipped national test ranges are a response to the need for very accurate interval level of measurement data. Laboratories, too, are often equipped for fine metrics. But in field or operational tests, ordinal level of measurement may be the highest level available.

(3) The experiment was designed to gather interval data, from populations with common variance, but something went wrong. Instrumentation sometimes fails. Even if the measurement is accurate, variances may not be common or the data may not be transformable to look normal. Rather than giving up, or repeating the experiment, nonparametric tests may answer some questions originally to be answered by analysis of variance or regression.

III. THE COMPUTER PACKAGE

A. GENERAL

This section gives a brief description of the computer package. Tests are described in sections V through XXVIII, while the Appendices describe the programs and files in detail.

The computer package consists of two computer programs, Lochinvar and Crunch. Also involved are three types of CMS files: data files and an options file, written by Lochinvar to be read by Crunch; and output files, written by Crunch, containing test results.

B. LOCHINVAR

The program Lochinvar (LOad CHoices and INput VARIables) is the user's main contact with the programs. It is interactive, prompting and screening the user's entries. Its purpose is to write two types of files needed by the program Crunch: the data file, and the option file.

1. Data Entry

Lochinvar allows the user to input data in two separate ways. The user must decide which one to use before starting the program.

a. Method I: Data Values into a Data File

When the data is numerical, the user should strongly consider this method. Lochinvar prompts entry of data set and variable names, and of the data set values themselves. All that information is sent to a file on the user's A disk; the user chooses one of six files, "lochi a", ..., "lochi f", into which the data is written. Unless

erased or over-written, the data will be available for repeated use.

All 24 of the data tests in this text can compute test statistics from data files. However, a data file itself is not enough to perform a test; the option file is needed to reference the data file and communicate the choice of test and test parameters to Crunch. After a data file is written, Lochinvar prompts the writing of the options file, as described in subsection 2, below.

b. Method II: Cell Counts in the Options File

When the data is non-numerical, this is the user's only option; it may be more convenient even if the data is numerical. Four of the nonparametric tests handle nominal data--binomial, McNemar, chi-square test for independence, and Cochran. Instead of entering all the data values, the user may run one of these tests by entering the appropriate count of elements which fall into a test's subcategories. For example, the binomial test can be run on a large data set by simply entering two numbers: the count of "successes" and the count of "failures".

In this method of data entry, Lochinvar prompts entry of cell counts at the same time it is prompting the test selection and choice of parameter. All that information is written to the options file; no data file is written.

2. The Options File

The purpose of the options file is to tell Crunch everything it needs to know to perform a nonparametric test. In all cases, the options file contains the user's selection of which test to be performed, the parameters to be used in that test, and to which CNS output file the output should be written. If the data values are to be used (Method I), the option file states which data file Crunch should read. If cell counts are used (Method II), the options file contains those counts, as well as the data set name.

The options file is described in detail, with an example, in Appendix D.

3. Using Lochinvar

In CMS, type "LOCHI". The program will display a sequence of questions; the user's response will depend on the method of data entry and test selection.

a. Use of Previously Entered Data

The program asks "do you wish to perform a test on a previously entered data set?", i.e., one of the six CMS data files "lochi a", ..., "lochi f".

If the answer is yes, then Lochinvar asks which data set to use. The program reads the data set's name, and displays it to the user, and asks "Is this the right data set?" If yes, the user indicates so, and the test selection is next. If no, the user must choose another file.

If the answer to the question of previously entered data is no, then Lochinvar asks whether data entry is by data values (Method I) or cell counts (Method II). If the user wants to enter cell counts, the next step is to select a test. If the user wants to enter a data set, the program asks:

- (1) To which file the input data is to be written
- (2) What is the data set name, number of variables and variable names.

The program will then prompt entry of data values. If there are two variables in the data set, the user will be offered the choice of entering data pair by pair; this will only work if there are an equal number of elements in each of the two variables. Normally, all the data elements of the first variable will be entered together, then all of the second variable, et cetera. After entry of each variable, Lochinvar will offer the user the chance to check, change or add data elements to the values just entered.

After the data set's information has been entered, Lochinvar writes it to the specified file. Writing of the options file is next.

b. Selection of the Test to be Performed

After the choice of data entry has been given, Lochinvar offers a menu of tests from which the user may choose. If cell counts was the chosen method of data entry, four tests are offered. If a data file is to be used, the choice set of test offered is determined by:

- (1) the number of variables in the data set
- (2) whether the variables have an equal number of elements

Lochinvar will not offer a test unless the data set contains at least the minimum number of variables needed to perform the test.

Some nonparametric tests require that data elements of one variable be related to corresponding elements of other variables, i.e., that the data set be paired or blocked. Lochinvar will not offer such a test unless there are the same number of elements in each of the data set's variables. However, variables can be equal sized but not paired or blocked. The user should not choose an inappropriate test just because Lochinvar offers it.

c. Output File Selection

The user chooses among six possible output files, "u listing",... "z listing". The files will be written to user's A disk after execution of the program Crunch. Previous contents of the designated file will be overwritten after Crunch runs.

d. Test Parameter Entries

Each test has different entries required. See the section on the desired test, sections V through XXVIII for specifics.

C. CRUNCH

1. Executing the Nonparametric Test

After the options file has been written, the user executes the test by typing "CRUNCH" while in CMS.

2. Actions of the Program Crunch

Crunch reads the options file, and when necessary, the designated data file. It performs the nonparametric test requested, and reminds the user with a display on the screen to which output file the results have been written. Crunch actually writes three files:

a. The Output File

The output files are named "u listing", ..., "z listing". At the top of the file is the label, consisting of the name of the data set, the number of variables in the data set, the number of variables used in the test (if less than the number in the whole data set), and the names of the variables used in the test. Following the label is the name of the test performed. The rest of the output file's format varies, depending on the test chosen. In general, the output file contains only the most salient results of the computations. All output files contain either a test statistic, or a test statistic and the empirical level of significance for the desired null hypothesis.

b. The Problem Listing File

If there are any problems encountered by the computer that result in Fortran error or warning messages, such messages will be written to the CMS file "problem listing".

c. The Nptest Listing File

This file contains intermediate computations of the program as it works towards the test statistic. For example, the Smirncv test requires finding differences in probabilities of two empirical distributions. Since the

test statistic is simply the largest absolute difference, the output file will only list the largest positive and most negative difference, as well as the test statistic. "Nptest listing", among other things, will include each difference between the empirical cumulative distribution functions (cdf's). This listing file is not arranged to be attractive or suitable for submission, but it is labelled, and may be of some use in understanding the operations of Crunch.

D. ADDITIONAL FEATURES OF THE PACKAGE

To avoid overwriting input or output files, six of each were provided. If more are needed, the user can rename some of the existing files.

Lochinvar is a convenient way to enter data by hand. When data is generated from other programs or read from disk or tape, the user may want to create his own data file rather than use Lochinvar. Detailed instructions for this are contained in Appendix C.

Crunch is written in Fortran. Users who are experienced programmers may want to modify it to suit their own tastes, or adapt portions of it into their own programs. An explanation of the program is contained in Appendix E. The program is listed in Appendix G.

E. CONSIDERATIONS IN DATA ENTRY

Lochinvar is written in Pascal, a convenient language for interactive programs, but a language which may be unfamiliar to many users. This section provides sufficient insight to Lochinvar so that the user can efficiently communicate choices to the program.

1. Entry of Numbers

When entering numerical data, Lochinvar requires that the first character read is either a numeral or a minus sign followed by a numeral. Therefore, to enter a number between minus one and plus one, precede the decimal with a "0".

When entering more than one number at a time, leave a space (not a comma) between the numbers.

If Lochinvar expects a number and gets anything else, it will halt with an error. This is a convenient way to get out of the program quickly, but anything entered on that run of Lochinvar will be lost. The one exception is when entering a data set of data values and an options file in a single run. After the data set is entered, the program writes it to the designated file, and asks "now, do a test with this data?" Any subsequent errors will not effect the data file itself, just erase all additional entries.

2. Entry of Characters

When the program prompts for a single letter response, e.g., "enter Y or N", enter "Y" or "N" and press the enter key. The program expects and will read only the first letter. Single letter entries are contained in "repeat...until" loops which will not allow the user to procede until one of the offered choices is made.

Do not use the alternate character (apl) setting on the keyboard while using Lochinvar.

When Lochinvar prompts for the data set name or variable names, the user must enter from one to 48 characters. These names are one of the few things for which the program does not offer a chance to correct an error. The user must be careful, checking the names prior to hitting the "enter" key. If an error is made anyway, the data or options file can be corrected using Xedit in CMS.

3. Minimum Entry

If the program prompts for an entry, enter at least one numeral or character. The interactive read commands are in "do while not eoln " (end of line) loops; if the enter key is pressed with nothing to enter, Lochinvar will halt with an error, because it attempted to read beyond the end of the line. As with numerical entry errors, previous entries will be lost.

IV. ROSTER OF AVAILABLE NONPARAMETRIC TESTS

A. CRITERIA FOR SELECTING THE MOST APPROPRIATE TEST

In selecting a nonparametric test, the user must consider three criteria: purpose of the test, data set level of measurement, and whether the data elements are related (i.e., paired or blocked).

B. CRITERION I: PURPOSE OF TEST

Nonparametric tests can be successfully used to test for questions of location, independence/interaction, and distribution. Table 1 shows a matrix of the nonparametric test with level of measurement (LOM) on the horizontal axis and general purpose on the vertical.

Location usually refers to whether the expected value of one population is equal to that of another, based on sample variables. Other purposes are testing for proportion of successes or for value of a single population's quantile. Table 2 lists the location tests with their null hypotheses, to help the user narrow the choice set.

Independence of two samples refer to statistical independence, i.e., correlation of zero. Interaction may be a before/after effect, or prediction of one variable by another. Table 3 lists these tests by their specific null hypotheses.

Some tests for distribution check for equal variance among variables. Other tests check a single variable's empirical cumulative distribution function (ecdf) against a theoretical distribution, with either hypothesized or estimated parameters. Still other tests compare the empirical

TABLE 1
Nonparametric Tests

<u>Purpose,</u> <u>Minimum #</u> <u>Vars:</u>	<u>Levels of Measurements:</u>		
	<u>Interval</u>	<u>Ordinal</u>	<u>Nominal</u>
LOCATION:			
1 var		Quantile	Binomial
2 var	Wilcoxon*	Sign* Mann-Whitney	
≥2 var	Quade*	Median Friedman* Kruskal -Wallis	Cochran
INDEPENDENCE/ INTERACTION:			
1 var		Cox-Stuart	McNemar
2 var	Correlation NP Regression Monotone Regression	Cox-Stuart*	R*C Contingency (chi-square)*
DISTRIBUTION:			
1 var	Kolmogorov (N<31) Goodness of Fit (chi-square, N>14) Wilkes-Shapiro (N<31) Lilliefors (N<31)		
2 var		Smirnov Cramer-von Mises	
≥2 var	Squared Rank Hartley		
* indicates that variables must be paired or blocked			

cdf's of two variables against each other. Table 4 has a list of these tests, with their hypotheses, levels of measurement, and other considerations.

The decision matrix identifies tests suited to each of these purposes. The specific section on the test should be checked to be sure that the test will do what is desired.

TABLE 2
Nonparametric Tests for Location

<u>HYPOTHESIS</u>	<u>NAME</u>	<u># VARS</u>	<u>LOM</u>	<u>SEC</u>	<u>PAGE</u>
$P(\text{success}) = P^*$	Binomial	1	N	V	30
$P(X < x^*) = P^*$	Quantile	1	O	VI	35
$E(X) = E(Y)$	Sign	2*	O	IX	47
	Mann-Whitney	2	O	VII	40
	Wilcoxon	2*	I	XI	65
$E(W) = \dots = E(Z)$	Cochran	$\geq 2^*$	N	XXIII	108
	Median	≥ 2	O	XVII	83
	Kruskal-Wallis	≥ 2	O	XVIII	88
	Friedman	$\geq 2^*$	O	XXI	99
	Quade	$\geq 2^*$	I	XXII	104

* indicates that variables must be paired or blocked

TABLE 3
Nonparametric Tests for Independence/Interaction

<u>HYPOTHESIS</u>	<u>NAME</u>	<u># VARS</u>	<u>LOM</u>	<u>SEC</u>	<u>PAGE</u>
Trend (v. order)	Cox-Stuart	1	O	VII	43
Trend (X v. Y)	Cox-Stuart	2*	O	VII	43
Before/After	McNemar	2*	N	X	51
$\rho(X, Y) = 0$	R*C Contingency (chi-square test for independence)	2*	N	XI	56
	NP Correlation#	2	O	XIV	69
$E(Y X) = a + b * X$ $b \neq 0$	NP Regression	$\geq 2^*$	I	XV	74
	Monotone Regress	≥ 2	I	XVI	79

* indicates that variables must be paired or blocked
gives pearson's and Spearman's rho, Kendall's Tau

TABLE 4
Nonparametric Tests for Location

<u>HYPOTHESIS</u>	<u>NAME</u>	<u># VARS</u>	<u>LOM</u>	<u>SEC</u>	<u>PAGE</u>
var(W) = ...=var(Z)	Hartley Squared Rank	>2 >2	I I	XX XIX	96 92
F _x (X) = G*(x) (specified parameters)	Kolmogorov (N<31) # Goodness of Fit # (chisquare, N>14)	1 1	I I	XXIV XX	113 61
F _x (X) = G*(x) (estimated parameters)	Lilliefors (N<31) (normal, exponential) Goodness of Fit # Wilkes-Shapiro (normal, N<31)	1 1 1 1	I I I I	XXV XII XXVI	118 61 121
F _x (X) = G _x (X) F, G: empirical cdfs	Smirnov Cramer-von Mises	2 2	0 0	XXVII XXVIII	124 127

tests normal, uniform, exponential, weibull, erlang

C. CRITERION II: LEVELS OF MEASUREMENT

Levels of Measurement were treated in section II.

D. CRITERION III: RELATIONSHIP AMONG DATA ELEMENTS

Data set variables may be unrelated, or paired (if 2 variables) or blocked (if more than 2 variables). When unrelated, the order of variables in the first variable is not dependent on the order of all other variables; e.g., the data elements of a variable could be entered in any order. In data sets with unrelated variables, the number of elements do not have to be equal.

Paired or blocked data sets require sample sizes to be equal and the order of the elements of the variables matters. Examples of pairing might be responses of husbands and wives, the intelligence of twins, the condition of the same experimental subject before and after the experimental treatment.

E. SELECTING THE BEST TEST

As mentioned in section II, two different nonparametric tests applied to the same data set could conceivably give different results to the same hypothesis. The example was the sign test and the Wilcoxon signed rank test. The reason given in section II--that those two tests extract different information from the data--underscores the reason for the decision matrix. In general, the most appropriate nonparametric test is one that tests exactly what is desired, and uses all the available information contained in the data. Using all the available information implies selecting a test which requires the level of measurement of the data, and using a paired or blocked test if appropriate.

These rules require thought. For example, the user might want to know if procedure X produces better results than procedure Y. At first glance, that is a problem of location: is $E(X) = E(Y)$? However, it could also be considered a problem of distribution: is $F_X(z) = F_Y(z)$? The test for distribution would check the empirical distributions across the entire range of observations, and thus use more of the information of the data set, particularly important with small data sets.

The computer package, described in the previous section, makes running multiple tests on the same data set fairly easy. The user can explore the results of using different tests on the same data, to get insight into his problem.

V. BINOMIAL TEST

A. OVERVIEW

1. Purpose

To characterize p , a population's proportion of "successes", based on the sample.

2. Level of Measurement

Nominal

3. Assumptions

The sample is randomly drawn from the population. Data may be divided into any dicotomous groupings, i. e., two mutually exclusive and collectively exhaustive groups. Here, the groups are called "successes" and "failures".

4. Hypotheses

Let p^* be a specified number $0 \leq p^* \leq 1$.

a. Two tailed test

(1) Null hypothesis H_0 : $p = p^*$

(2) Alternative hypothesis H_1 : $p \neq p^*$

b. One tailed test

(1) Null hypothesis H_0 : $p \leq p^*$

(2) Alternative hypothesis H_1 : $p > p^*$

c. One tailed test

(1) Null hypothesis H_0 : $p \geq p^*$

(2) Alternative hypothesis H_1 : $p < p^*$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user specifies which treatment of the data set is used for the test, a hypothesized proportion of successes (p^*), and the desired level of significance (α^*). The

program uses the count of successes and failures; there are three ways to indicate that dichotomization:

a. The user enters cell counts. This is the only way to handle non-numerical data, but will work for any data.

b. The user dichotomizes the data set by a user specified list. This is the way to handle numerical, nominal data. Data values which are elements of the specified list are in one set; the rest of the data values are in the other set. The user indicates whether the list defines success or failure.

c. The data set is divided in two by a partition value. This is only possible if the data is numerical and at least ordinal. All elements above the partition value are in one set; the user defines whether that set is success or failure. The values below the partition are the other type.

2. Program Activation with Cell Counts (choice a above)

a. Initiation and test selection

In CMS, type LOCHI. Enter "N" for no to question about previously entered data. Enter "C" for cell counts. Enter "U", "V", ... "Z" for desired output file. When offered a choice of tests, enter 1.

b. Cell count entry

Lochinvar will ask the number of successes and number of failures--enter those cell counts.

c. Parameter entry

Lochinvar will ask for a hypothesized proportion of successes P^* , and a desired level of significance, α^* . When prompted, enter those numbers

3. Program Activation with Data Values (choice b or c)

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 1. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 1. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index of variable to be used in test. Enter whether dichotomizing by list or by partition value

(2) If by list, enter the number of elements in that list. Enter elements of list. Enter "Y", when prompted if elements on that list represent success.

(3) If by partition value, enter that value. Enter "Y" if data elements below that value are successes.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the data file to which the data has been sent, e.g., "U LISTING". The names of the data set and the variable used in the test are at the top of the output file. The empirical level of significance, α , and confidence interval limits, are listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Level of Significance and Intervals

a. Count number of successes. Divide by the total number of elements to give p -hat.

b. If $NS \geq 20$, generate a binomial probability mass function with parameters N and p -hat. Otherwise, use the

normal approximation to the binomial, with parameters $N\hat{p}$ and $N\hat{p}(1 - \hat{p})$.

c. Compute tail probabilities

(1) The upper tail probability is the likelihood that the number of successes is greater than or equal to the sample number of successes.

(2) The lower tail probability is the likelihood that the number of successes is less than or equal to the sample number of successes.

d. Compute the level of significance, α

(1) if $H_0: p \leq p^*$, α equals the upper tail probability.

(2) if $H_0: p \geq p^*$, α equals the lower tail probability

(3) if $H_0: p = p^*$, α equals two times the minimum of the upper and lower tail probabilities

e. Compute the interval, N less than 21

(1) Let y_1 = number of successes minus one, = index of the pmf of the lower tail probability. Vary p^* , $0 \leq p^* \leq \hat{p}$, until the lower tail probability equals $\alpha/2$. p^* is the lower confidence limit.

(2) Let y_2 = number of successes plus one, = index of the pmf of the upper tail probability. Vary p^* , $\hat{p} \leq p^* \leq 1$, until the upper tail probability equals $\alpha/2$. This p^* is the upper confidence limit.

f. Compute the interval, N greater than 20

(1) Compute the standard error of estimate by taking the square root of the quantity (number of successes minus the number of failures, divided by N to the third).

(2) Find the value z , which is the desired quantile, $1 - \alpha/2$, of the standard normal distribution

(3) The lower confidence limit is \hat{p} minus z times the standard error. The upper confidence limit is \hat{p} plus z times the standard error.

2. Program Operations

a. Determine the cell count

b. If N is less than or equal to 20

(1) Generate a vector of N choose k , $k=0,1,\dots,N$.

(2) Generate a binomial pmf, with parameters N and $p\text{-hat}=\text{number of successes divided by } N$.

(3) Compute the tail probabilities

(4) Compute the empirical level of significance

(5) Using the previously generated N choose k vector, generate partial binomial pmf's, with terms 0 to number of successes minus one, or number of successes plus one to N . A bisection search technique varies p , $0 \leq p \leq p\text{-hat}$ or $p\text{-hat} \leq p \leq 1$, until the sum of the terms of the partial pmf is within 0.0001 of the desired level of significance ($\alpha^*/2$, because the interval is two tailed).

c. If the sample contains more than twenty elements

Use the normal approximation to the binomial. The subroutine Muvar uses numerical integration to find tail probabilities; from which the empirical level of significance is estimated. Muvar also furnishes the z quantile for computing the confidence interval limits.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400, if the program computes the cell count from a data list; unlimited if the user enters the cell counts directly. Maximum number of variables: 1.

b. Subroutines/Procedures Used

(1) Lochinvar: Binome, Which_vec, Hyp

(2) Crunch: Bintest; Muvar, Cibi21 or Nchuzk, Binpmf, Cibi20, Bisex

(3) External: Shsort

VI. QUANTILE TEST

A. OVERVIEW

1. General Purpose

The quantile test addresses the question of what proportion of the population is less than or equal to a specified value. An example would be that a student might be concerned less with the average grade on a test than the proportion of students who received at least the minimum passing grade.

2. Level of Measurement

Ordinal

3. Assumption

The data set is a random sample from the population.

4. Hypotheses

Here, p^* is the specified population proportion, $0 \leq p^* \leq 1$, and x^* is a specified value within the range of the population.

a. Two tailed test

(1) Null hypothesis H_0 :

the p^* th quantile = x^*

(2) Alternative hypothesis H_1 :

the p^* th quantile $\neq x^*$

b. One tailed test

(1) Null hypothesis H_0 :

the p^* th quantile $\geq x^*$

(2) Alternative hypothesis H_1 :

the p^* th quantile $< x^*$

c. One tailed test

(1) Null hypothesis H_0 :

the p^* th quantile $\leq x^*$

- (2) Alternative hypothesis h_1 :
the p *th quantile $> x^*$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must specify the values for p^* and x^* , and the letter of the hypothesis desired.

If there is more than one variable in the designated data set, the user must specify which is to be used.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 2. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 2. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index of variables to be used in test, if there is more than one in the data set.

(2) Enter the values of p^* , x^* , the letter of the chosen hypothesis, and the hypothesized level of significance, α^* .

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the data file to which the data has been sent, e.g., "U LISTING". The names of the

data set and variables used are at the top of the output file. Also included are the test statistic (the number of data elements less than or equal to x^*), the level of significance, and confidence limits for the population's p^* th quantile.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Calculation of Level of Significance and Confidence Limits

a. Compute the Test Statistic

Crunch compare the data elements with x^* , counting the number which are less than or equal to x^* . If there are no ties, $T1 = T2 =$ the count. If some elements are equal to x^* , then $T1 =$ the count of data elements less than x^* , and $T2 = T1$ plus the number of ties.

b. Computation of the Level of Significance

If $T1 = T2$, the computation of the level of significance is exactly the same as the binomial test, with $T1$ as the number of successes, the total number of data elements as n , and p^* as the p^* of the binomial test. If $T1 < T2$, then the lower tail probability is calculated using $T1$; the upper tail probability uses $T2$; and the computation proceeds as with a normal binomial test. See section V for details about computation of level of significance in the binomial test.

c. Find the Confidence Limits

The data elements are ranked; the ranked values are called "order statistics", so that $X(1)$ is the smallest value, $X(k)$ is the largest values in a data set of k values. The confidence limits are found by computing the values r and s , which are indices of the order statistics; once they are found, $X(r)$ is the lower limit, $X(s)$ is the larger limit.

(1) With less than 21 data elements, a binomial pmf is generated, with parameters k = number of data elements, and $p = p^*$. R is found by summing terms of the pmf from the 0th term ($P(N=0)$) until the first term which causes the sum to be greater than the half of the hypothesized level of significance ($\alpha^*/2$). The index of that term is r . S is found by summing terms of the pmf from the k th term ($P(N=k)$) until the first term which causes the sum to be greater than $\alpha^*/2$. The index of that term is S .

This procedure is conservative on both bounds, and therefore may be very conservative overall. For example, with $k = 18$, $p^* = 0.5$, and $\alpha^* = 0.15$, Crunch would choose r of 5 and s of 13; the probability that in the population an x would fall between $X(5)$ and $X(13)$ is 0.904, while the user would have been satisfied with about 0.85. With the binomial tables, the user might choose indices 6 and 13, or 5 and 13, which would have only 0.0481 in one tail but 0.1151 in the other, so that the test would have almost exactly the α desired.

(2) With more than 20 data elements ($=k$), the hypothesized variance is $k(p^*(1-p^*))$; its square root is the sample standard deviation. The expected value is kp^* . R is the expected value plus the quantity standard deviation times the $(\alpha^*/2)$ quantile of the standard normal distribution. If r is not an integer, round up. S is the expected value plus the quantity standard deviation times the $(1-\alpha^*/2)$ quantile of the standard normal distribution. Again, round up if necessary to make s an integer.

2. Program Method of Operation

The subroutine `celler` calculates $T1$ and $T2$ by calling the subroutine `dicot`, which takes the values of the data set and find the number less than and the number less than or equal to the specified x^* . The subroutine `quantile` receives that information, then calls the subroutine `bintst` to perform the binomial test described above.

Quantl then calls subroutines Nchuzk and Binpmf to generate a binomial pmf to calculate the confidence limit indices r and s, if the total number of values is 20 or less. Otherwise, Quantl calls the subroutine Normv for values of the $(\alpha*/2)$ and $(1-\alpha*/2)$ quantile of the normal distribution to use in the calculation of r and s. The data values are sorted by the subroutine RNQ, and the X(r) and X(s) values dereferenced as the confidence limits.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Only one variable is used.

b. Subroutines/Procedures Used

- (1) Lochinvar: Quantile
- (2) Crunch: Quantl, Celler, Dicot, Bintst; Rnq, Nchuzk, Binpmf or Normv
- (3) External: Shsort

VII. MANN WHITNEY TEST

A. OVERVIEW

1. General Purpose

With two variables, test whether the populations from which the variables were drawn have the same mean.

2. Level of Measurement

Ordinal

3. Assumptions

- a. Each variable is a random sample.
- b. There is mutual independence between samples.

4. Hypotheses

a. Two tailed test

- (1) Null hypothesis $h_0: E(X) = E(Y)$
- (2) Alternative hypothesis $h_1: E(X) \neq E(Y)$

b. One tailed test

- (1) Null hypothesis $h_0: E(X) \geq E(Y)$
- (2) Alternative hypothesis $h_1: E(X) < E(Y)$

c. One tailed test

- (1) Null hypothesis $h_0: E(X) \leq E(Y)$
- (2) Alternative hypothesis $h_1: E(X) > E(Y)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must decide which two of the variables to use in the test if the data set contains more than two. The user specifies the level of significance.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 4. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 4. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test.

(2) Enter the level of significance.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The empirical level of significance and whether to reject the null hypothesis are stated.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic/Level of Significance

a. Sums of the Ranks of the First Variable

Rank the values of the two variables. Sum the ranks of each. The test statistic T is the sum of the ranks of the first variable.

b. Overall Test Statistic

The overall test statistic, if there are many ties, is T_1 . Let n be the number of elements in the first variable, m the number of elements in the second variable; N

is the sum of n and m . Sum the squared ranks of both variables to find R^2 . The numerator of the test statistic is T minus $n(N+1)/2$. The denominator is the square root of the difference of two terms: $R^2*nm/N(N-1)$, and $(nm(N+1)**2)/4(N-1)$.

2. Program Method of Operation

The subroutine Manwht calls the nonims1 subroutine Utest, to get the level of significance.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

- (1) Lochinvar: Manwhit
- (2) Crunch: Manwht
- (3) External: Utest

VIII. COX-STUART TEST FOR TREND

A. OVERVIEW

1. General Purpose

This is a specialized form of the sign test, to determine if the data elements in the second half of a sample are significantly greater or less than the corresponding values in the first half. With one variable, the order of the data elements determines how the data elements are paired for comparison.

With two variables, in which the data elements are paired, the ranks of the independent variable orders the values of the dependent. This yields a rough test of the interaction or dependence of the two variables.

2. Level of Measurement

Ordinal

3. Assumptions

- a. The data elements represent random draws
- b. Either there is no trend, or there is an underlying trend upward or downward throughout data.

4. Hypotheses

Where (+) indicates the second data element is greater than the first in the pair, otherwise (-), so that $P(+)=P(-)$ indicates no trend, $P(+)>P(-)$ indicates upward trend (1 variable) or vary directly (1st variable to 2d variable), and $P(+)<P(-)$ indicates downward trend or vary an inverse relationship.

a. Two tailed test

(1) Null hypothesis $h_0: P(+)=P(-)$

(2) Alternative hypothesis $h_1: P(+)\neq P(-)$

b. One tailed test

(1) Null hypothesis $h_0: P(+)\leq P(-)$

(2) Alternative hypothesis $h_1: P(+)>P(-)$

c. One tailed test

(1) Null hypothesis $h_0: P(+)\geq P(-)$

(2) Alternative hypothesis $h_1: P(+)<P(-)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If one variable is used, make sure data elements are in the order (often order of time) for the trend to be tested. If two variables are to be used, data elements must be paired. The user selects the type of hypothesis and the level of significance. If there are two variables, the user specifies which is dependent.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 3. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 3. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index or indices of variables to be used in test. Enter pairs to be excluded, if using two variables.

(2) Enter letter of hypothesis type and level of significance.

(2) If there are two variables, enter which is dependent.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Level of Significance

a. Form Pairs (X,Y) Significance

(1) If there is one variable, call it Y, and go to step (2). If two variables, let X be the independent variable. and Y be dependent. Find the order of the X. Use that ordering to re-order Y in ascending order of the values of X.

(2) Let k be the number of elements in the variable Y. If K is even then there will be $(k/2)$ pairs, if odd then $((k-1)/2)$ pairs. Let np equal the number of pairs.

b. Find Test Statistic

(1) For $i=1$ to np, compare $Y(i)$ to $Y(np-i+1)$. If $Y(i) > Y(np-i+1)$ then increment the counter for failures. If $Y(i) < Y(np-i+1)$ then increment the counter for successes. The test statistic is from the binomial test, with n equal the number of successes plus failures, p^* equal 0.5, and the number of success equal number of times the second half data elements are strictly greater than the corresponding first half elements.

2. Program Method of Operation

a. Ordering the Dependent Variable

The subroutine Coxstu dereferences the X values into the INDEP vector, sorts those values using Shsort, and

uses the indices returned in the vector KEY to dereference the Y values into a vector TLIST.

If there is only one variable, use a do for loop to load its values, in their current order, into TLIST.

b. Computing the Test Statistic

The subroutine Coxstu calls the subroutine Sigt to perform the sign test. See section IX.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

- (1) Lochinvar: Coxstu
- (2) Crunch: Coxstu, Sigt
- (3) External: Shsort

IX. SIGN TEST

A. OVERVIEW

1. General Purpose

With pairs of data (X_i, Y_i) , determine if the expected value of the first element, $E(X)$, is greater than the expected value of the second element, $E(Y)$.

2. Level of Measurement

Ordinal within each pair; that is, it can be determined that X_i is greater than Y_i (+), or that Y_i is greater than X_i (-), or that they are equal. The values of the pair do not have to be ordered with any values outside the pair.

3. Assumptions

a. Random Sample

X, Y are a random sample of the population pairs.

b. Level of Measurement

Ordinal within each pair, nominal among pairs.

c. Invariance of Probability that $X_i > Y_i$

The underlying population from which the sample is drawn is assumed to have a constant $P(X > Y)$.

4. Hypothesis

Here, (+) is the case that $X > Y$, (-) is the case that $Y > X$, where X, Y refer to population random variables.

a. Two tailed test

(1) Null hypothesis $h_0: P(+) = P(-)$

(2) Alternative hypothesis $h_1: P(+) \neq P(-)$

b. One tailed test

(1) Null hypothesis $h_0: P(+) \leq P(-)$

(2) Alternative hypothesis $h_1: P(+) > P(-)$

c. One tailed test

(1) Null hypothesis $h_0: P(+) \geq P(-)$

(2) Alternative hypothesis $h_1: P(+) < P(-)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

It is possible that the data is non-numerical, but ordinal within pairs, e.g., a list of winners of arm-wrestling contest between two platoons as a measure of arm strength. If so, then perform the test as a binomial test, entering cell counts, with number of successes as the number of times $X_i > Y_i$, n is the total number of untied pairs, and p^* as 0.5.

With numerical data, the program will make the comparisons. The user must enter the hypothesis letter and the hypothesized level of significance.

If there are more than two variables in the designated data set, the user must choose which are included in the test.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 5. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 5. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter blocks or pairs to be excluded, if any.

(2) Enter the letter of the hypothesis type and the hypothesized level of significance, α^* .

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic and the level of significance are printed, as well as whether to reject the null hypothesis.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Level of Significance

a. Calculation of the Test Statistic

For each pair (X_i, Y_i) , subtract Y_i from X_i . Count the number of times the difference is greater than 0: this is the test statistic. The total number of untied pairs is the count of times the difference is not zero.

b. Calculation of the Level of Significance

The level of significance is calculated by the binomial test, with the number of successes equal to the test statistic, the total sample size equal to the number of untied pairs, and the hypothesized proportion of success p^* equal to 0.5.

See section V for details about computation of levels of significance by the binomial test.

2. Program Method of Operation

The subroutine Sigt compares the values of the lower indexed variable with the corresponding values of the other variable. Values within epsilon ($=0.0001$ by default) are considered tied. If not tied, then the program checks whether $X_i > Y_i$ or $X_i < Y_i$, and increments the appropriate count, (+) or (-). Sigt then calls the subroutine Bintst to compute the level of significance.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum
number of pairs: 200. Required number of variables: 2.

b. Subroutines/Procedures Used

- (1) Lochinvar: Signtest
- (2) Crunch: Sgnt
- (3) External: Shsort

X. MCNEMAR TEST FOR SIGNIFICANCE OF CHANGE

A. OVERVIEW

1. General Purpose

With data in pairs (X_i, Y_i) , and x and y values each classifiable as successes or failures, to determine if a significant change has occurred.

2. Level of Measurement

Nominal

3. Assumptions

- a. The pairs are a random sample.
- b. X and Y can be divided into mutually exclusive and collectively exhaustive groups: success and failure.
- c. Each of the k pairs can be categorized into one of four cells: success/success $(1,1)$, success/failure $(1,0)$, failure/success $(0,1)$, and failure/failure $(0,0)$.

4. Hypothesis

a. Null hypothesis

$$H_0: P(0,1) = P(1,0) \text{ for all } i$$

b. Alternative hypothesis

$$H_1: P(0,1) \neq P(1,0) \text{ for all } i$$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If the data is nominal, the user must enter the counts of each of the four categories.

If the data is ordinal or above, the user may enter cell counts, or may specify partition values to dichotomize X and Y . Lochinvar will ask for the partition value for X , and whether successes are less than it. For example, if

17.5 is the partition value for X, and successes are greater than it, any X value greater than 17.5 will be a success; depending on its Y value, the pair will be counted in the success/success (1,1) cell, or success/failure (1,0) cell. After asking for the X partition value, Lochinvar will ask for the Y partition value and whether successes lie above or below.

If there are more than two variables in the designated data set, Lochinvar will ask which two are to be used for the test.

2. Program Activation when Using Cell Counts

a. Initiation and test selection

In CMS, type LOCHI. Enter "N" for no to question about previously entered data. Enter "C" for cell counts. Enter "U","V",..."Z" for desired output file. When offered a choice of tests, enter 6.

b. Cell count entry

Lochinvar will ask the cell counts for each of the four categories: success/success, success/failure, failure/success, and failure/failure.

Enter the hypothesized level of significance.

3. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 6. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 6. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of the two variables to be used in test. Enter blocks or pairs to be excluded, if any.

(2) Enter the partition value for the lower indexed variable and whether successes are less than that value. Enter the partition value for the variable with the higher index; indicate whether successes are less than that value.

(3) Enter the hypothesized level of significance.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file.

If there are 20 or less pairs in the data set which are classified as (1,0) or (0,1), then the test statistic is T_2 =number of pairs classified as (1,0); that test statistic and level of significance will be listed. Otherwise, the test statistic T_1 , explained below, and the level of significance will be given.

In either case, the program will compare the empirical level of significance with the hypothesized, and state whether the null hypothesis is rejected.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Level of Significance

a. Computation of Cell Counts

If cell counts were provided by the user, go to step b or c. If data values were used, Crunch examines each

pair (X_i, Y_i) to see the relation each data element has to its corresponding partition value. Crunch decides which cell the pair belongs to, $(1,1), \dots, (0,0)$, and increments the count for that cell.

b. If Count for Cells $(1,0)$ and $(0,1)$ less than 21

The test statistic is T_2 , which equals the count of cell $(1,0)$. The binomial test is invoked to find the level of significance. T_2 is the number of successes; the total count of cells $(1,0)$ and $(0,1)$ as n , the total sample size; the hypothesized proportion of success, p^* is 0.5. See section V for details of computation.

c. If Count for Cells $(1,0)$ and $(0,1)$ exceeds 20

The test statistic is T_1 , equal to the squared difference between the cell counts $(1,0)$ and $(0,1)$, divided by their sum. This test statistic is compared to the chi-square distribution with one degree of freedom to find the level of significance.

2. Program Method of Operation

a. Cell Count Calculation

The subroutine Celler compares each of the lower indexed values to the first partition value, then each of the other variable's values to the second partition value. At each comparison, a "1" is assigned to an integer vector if the value is a success, otherwise a "0" is assigned. The resulting integer vector, with the same number of elements as the original data set, is then used to find cell counts. With k pairs, the first through the k th elements of the integer vector are multiplied by two, then added to the corresponding $k+1$ st through $2k$ th elements. The resulting k element vector has values 0,1,2,3, corresponding to the four cells. The vector is then used to compute cell counts.

b. Calculation of the Test Statistic

Crunch figures if the cell counts of $(1,0)$ and $(0,1)$ total more than 20. If so, T_1 is computed, otherwise T_2 is used.

c. Calculation of the Level of Significance

If T1 is used, the subroutine `mcnemar` performs linear interpolation of tabled values of the chi-square distribution with one degree of freedom. If T2 is used, the binomial distribution subroutine `BinTst` is used to calculate the level of significance.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of pairs: 200 if program computes cell counts; no limit if the user enters cell counts.

b. Subroutines/Procedures Used

- (1) Lochinvar: `Cel_test`, `McNemar`
- (2) Crunch: `McNemr`, `Celler`, `McCount`
- (3) External: `Shsort`

XI. CHI-SQUARE TEST FOR INDEPENDENCE (R*C CONTINGENCY)

A. OVERVIEW

1. General Purpose

Used with a data set composed of paired data elements, to test whether the classification or value of the first element of the pair is statistically independent of the classification or value of the second.

2. Level of Measurement

Nominal

3. Assumptions

- a. The pairs is randomly drawn from the population.
- b. The first element of each pair can be placed in exactly one of r categories (rows), and the second element of each pair can be placed in exactly one of c categories (columns).

4. Hypothesis

a. Null hypothesis

$H_0: P(\text{pair is in row } i \text{ and col } j) \text{ equals } P(\text{first element in row } i) \text{ times } P(\text{second element in row } j)$

b. Alternative hypothesis

$H_1: P(\text{pair is in row } i \text{ and col } j) \text{ does not equal } P(\text{first element in row } i) \text{ times } P(\text{second element in row } j)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

a. Entry of cell counts

If the data is nominal, the user must decide the categories of the rows and columns, and enter the observed cell counts, O_{ij} , for each cell. The user may use this technique for data of a higher level of measurement.

b. Entry of fixed row and column totals

If the data is ordinal or above, and the categorization of the data points is by their order (e.g., not by whether their last digit is odd), the user may enter R_i 's and C_j 's. The sums of the R_i 's and C_j 's must exactly equal the number of pairs in the data set.

c. Entry of row and column partition values

If the data is ordinal or above, and the categorization is based on order, the user can specify $(r-1)$ values to divide the data elements into r rows, and $(c-1)$ values to divide the data elements into c columns. For example, a data pair would fall into cell $(1,2)$ if the first element of the pair were less than or equal to the first row partition value, and the second element of the pair were greater than the first column partition value but less than or equal to the second.

2. Program Activation when Using Cell Counts (choice a)

a. Initiation and test selection

In CMS, type LOCHI. Enter "N" for no to question about previously entered data. Enter "C" for cell counts. Enter "U","V",..."Z" for desired output file. When offered a choice of tests, enter 7.

b. Cell count entry

Lochinvar will ask the number of rows and columns. It will then prompt entry of the observed cell count for each cell.

3. Program Activation Using Data Values (choice b or c)

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 7. If data needs to be entered, enter "N" for first question, "V" for second; enter the

data; when offered a choice of tests, type 7. Enter "g", "v", ... "z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of the two variables to be used in test. Enter the number of rows and the number of columns. Enter whether using fixed row and column totals or partition values.

(2) If by fixed totals, enter the row totals when prompted, then the column totals.

(3) If by partition value, enter the row partition values when prompted, then the column partition values.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The data set and variable names are at the top of the output file. The test statistic T, the degrees of freedom, and the matrix of observed cell counts are listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Computation of Test Statistic

The program Crunch will:

a. Find the observed cell count

Specify the r rows and c columns, thus creating $r*c$ mutually exclusive cells. Count the number of pairs which fall into the categorization of each cell: that number is the observed cell count, O_{ij} , of cell ij .

b. Find the expected cell count

All the cell counts in row i are summed to find the row total R_i . All the cell counts in column j are summed to find the column total C_j . The expected cell count E_{ij} is found by multiplying R_i times C_j and dividing by the total number of pairs, N .

c. Find the test statistic T

For each cell, subtract O_{ij} from E_{ij} . Square each difference. Divide each squared difference by the corresponding E_{ij} . Sum those quotients to find the test statistic T , which can be compared to a chi-square distribution with $(r-1)*(c-1)$ degrees of freedom. The quantile of the chi-square distribution corresponding to T indicates the empirical confidence coefficient, or the complement of the empirical level of significance.

2. Program Operations

a. Computation of the cell counts

If the user supplied the cell counts, go to step b. If the user gave fixed row and column totals, rank all the first elements to find the row partition values, and rank all the second elements to find the column partition values. Compare the first element of each pair to the row partitions: assign to each pair the index of the largest partition value which the first element is less than; if the first element is larger than the largest partition value, it is assigned the integer r . Do the same with the second elements and the column partition values. Each pair now has two integers assigned, which designate membership in a row and in a column. Count the number of pairs falling in each cell: this is the observed cell count O_{ij} .

b. Compute the test statistic T

Find the row and column totals by summing O_{ij} 's. Compute the expected cell counts E_{ij} by R_i*C_j/N . Square the difference $(O_{ij}-E_{ij})$, divide by E_{ij} , sum over all cells.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400, if the program computes the cell counts; unlimited if the user supplies cell counts. Maximum number of variables/treatments: 2. Maximum total cells ($r*c$): 50.

b. Subroutines/Procedures Used

(1) Lochinvar: Cel_test, Rc_cont; or Which_test, Rc_cont, Set_cell, Which_vec, Exclud_b

(2) Crunch: Celler, Partv, Pc3 or Pc4, Rcctr, Rcont

(3) External: Shsort

XII. CHI-SQUARE GOODNESS OF FIT TEST

A. OVERVIEW

1. General Purpose

With a sample of 15 or more data elements, test if that sample could be drawn from a specified distribution.

2. Level of Measurement

Interval

3. Assumptions

The variable contains a random sample.

4. Hypothesis

Where $F^*(x)$ is hypothesized distribution function, and $F(x)$ is the distribution of the population.

a. Null hypothesis

$H_0: F(x) = F^*(x)$ for all x

b. Alternative hypothesis

$H_1: F(x) \neq F^*(x)$ for at least one x

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must decide among five distributions for which to test. Parameters may be specified by the user, or may be estimated by the program (using method of moments).

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 8. If data needs to be entered, enter "N" for first question, "V" for second; enter data,

and when offered a choice of tests, type 8. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter blocks or pairs to be excluded, if any.

(2) Enter theoretical distribution. Choose among: uniform, normal, exponential, erlang and weibull.

(3) Enter the parameters of the theoretical distribution, or request that the program estimate the parameters. In the normal and exponential the method of moments can be invoked to allow the program to estimate the maximum likelihood estimates of the parameters. In the case of the erlang, the user must specify the integer shape parameter n , but can allow the program to estimate the scale parameter λ . In the case of the uniform and the weibull, the user must specify the parameters

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic T and the degrees of freedom will be displayed. Finally, the matrix of cell counts, expected and observed, will be given.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculation of the Expected Cell Counts

Let k equal the number of data elements. If $15 \leq k \leq 50$, divide k by 5 and round to the nearest integer.

This will be C , the number of cells; e.g., $k=17$ implies $C=3$, $k=18$ implies $C=4$. In all cells except perhaps the first and last the expected cell count will be 5. If $C*5=k$, then the expected cell counts for all cells is 5. If $C*5=k+1$, the expected cell count of the first cell is 4; if $C*5=k+2$, the expected cell count of the first and last cells is 4. If $C*5=k-1$, the expected cell count of the first cell is 6; if $C*5=k-2$, the expected cell count of the first and last cells is 6. For example, if $k=17$, $C=3$; since $3*5=17-2$, the first and last cells have expected counts of 6, i.e., the expected cell counts are 6,5,6.

If $k > 50$, there will be 10 cells. Divide k by 10 and truncate; that will be the minimum expected cell count E . The remainder of the truncation is $rem=k-10*E$; add 1 to the cell count of the last rem cells.

The expected cell counts are stored in the vector EJ . This is the building block of the $F=P(X \leq x)$ values of the theoretical distribution. Scan the EJ vector to find the CUM vector, i.e., $CUM(1)=EJ(1)$, $CUM(2)=EJ(1)+EJ(2)$, ..., $CUM(j)=\text{sum of } EJ(i), i=1, \dots, j$. Divide each element of the CUM vector by the number of elements, k , to form the $CUMP$ vector.

b. The Partition Values

The usual form of theoretical distributions takes a value x and returns a value F , such that $F = P(X \leq x)$. In this case, the inverse form of the theoretical distributions is used to take an F , contained in the $CUMP$ vector, and return an x . The first $(C-1)$ values returned form the partition values which will divide those data elements into cells.

c. The Observed Cell Count

The data elements are compared to the partition values contained in the vector $PARTV$. An element falls in the i th cell if it is larger than $PARTV(i-1)$ but less than

or equal to PARTV(i). Elements less than PARTV(1) are in the first cell; elements larger than PARTV(C-1) are in the last cell. The observed cell counts are contained in the vector RCCT.

d. The Test Statistic

For each cell, subtract EJ(i) from RCCT(i), square that difference, and divide that square by EJ(i). Sum over all cells. This is the test statistic T, with degrees of freedom (C-1).

2. Program Method of Operation

The subroutine Gdfit calls the subroutine Muvar to find the sample mean and variance if parameters must be estimated. The subroutine Cpvgen figure the EJ, CUM, and CUMP vectors. The vector PARTV is figured in one of five subroutines, depending on the theoretical distribution specified: unid, nornd, expd, erld, weid. The observed cell counts RCCT are found in Celcnt. The test statistic T is computed in the subroutine Gdfit.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variable: 1.

b. Subroutines/Procedures Used

- (1) Lochinvar: Gd_fit_n, Gd_fit_e, Gd_fit_u, Gd_fit_er, Gd_fit_w, Gd_fit
- (2) Crunch: Gdfit, Cpvgen, Muvar, Celcnt, Unid, Nornd, Expd, Erld, Weid
- (3) External: Shsort

XIII. WILCOXON SIGNED RANK TEST

A. OVERVIEW

1. General Purpose

With two paired variables (X, Y) , the signed rank test determines whether the expected values of populations X and Y could be the same.

This requires interval data. If the data is only ordinal, use the sign test.

2. Level of Measurement Interval

3. Assumptions

The pairs (X_i, Y_i) are random samples from the population of all pairs.

4. Hypotheses

a. Two tailed test

(1) Null hypothesis $h_0: E(X) = E(Y)$

(2) Alternative hypothesis $h_1: E(X) \neq E(Y)$

b. One tailed test

(1) Null hypothesis $h_0: E(X) \leq E(Y)$

(2) Alternative hypothesis $h_1: E(X) > E(Y)$

c. One tailed test

(1) Null hypothesis $h_0: E(X) \geq E(Y)$

(2) Alternative hypothesis $h_1: E(X) < E(Y)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

To make the test more flexible, a constant term can be added to the higher indexed variable, so that the test becomes a question of the relationship of $E(X)$ and $E(Y) + b$.

By default, the value of b is 0; but the user can specify any real number.

Lochinvar also asks for the hypothesis type and level of significance.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 9. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 9. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter blocks or pairs to be excluded, if any.

(2) Enter whether a constant term is to be added to the value of the higher indexed variable, Y . By default, the constant is 0.

(3) Enter the letter of the type of hypothesis and the hypothesised level of significance.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file.

Crunch will provide the sum of negative ranks and sum of positive ranks, as well as the test statistic T in

all cases. If there are fewer than 20 pairs, table look up is required; otherwise, the program will compute the empirical level of significance.

When the number of untied pairs is less than 20, the output will also contain the vector Dif, the differences $Y_i - X_i$, in ranked order.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic/Level of Significance

a. Sums of the Negative and of the Positive Ranks

Two sums are calculated, one for the negative differences-- those pairs in which $X_i > Y_i$ --and one for the positive differences. Pairs in which $X_i = Y_i$ are not considered. Subtract X_i from Y_i , to form the vector Dif. Take the absolute value of Dif to form Adif. Rank the values of Adif to form the vector Rank. If $Y_i < X_i$, the i th value of Rank is added to the sum of negative ranks, then that i th value is multiplied by negative 1. If $Y_j > X_j$, the j th value of Rank is added to the sum of positive ranks.

b. Computation of Test Statistic

The vector Rank was originally the ranks of the absolute differences; now, the ranks for the negative differences have been multiplied by -1. Sum the vector Rank to find the numerator of the test statistic. Sum the squares of the elements of Rank to find the denominator of the test statistic. Divide numerator by denominator.

c. Level of Significance

If the number of untied pairs is 20 or less, use table A-13 [ref 1], using the sum of positive ranks. Otherwise, Crunch computes an approximation, using the test statistic T and the standard normal distribution.

2. Program Method of Operation

The subroutine Wilcox finds the Dif and Adif vectors, and calls the subroutine Ranque (which in turn calls Rnq) to find the vector Rank. Wilcox computes the sums of the positive and negative ranks, as well as the test statistic T. If the number of untied pairs is 20 or less, it will display the sums of ranks. Otherwise, it will call the subroutine Normv to find the approximate empirical level of significance, and display whether to reject the null hypothesis.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2; maximum number of pairs: 200.

b. Subroutines/Procedures Used

- (1) Lochinvar: Wilcoxon
- (2) Crunch: Wilcox, Ranque, Rnq, Normv
- (3) External: Shsort

XIV. NONPARAMETRIC CORRELATION

A. OVERVIEW

1. General Purpose

With a pairs of data elements, X and Y, test whether the value of X is statistically independent of the value of Y, or if it varies directly or inversely with Y.

Three different test statistics are available by running this one program: Pearson's rho, Spearman's rho, and Kendall's tau.

2. Level of Measurement

Interval for the Pearson's rho to have meaning.
Ordinal for Spearman's rho and Kendall's tau.

3. Assumptions

X,Y are a random sample of population pairs.

4. Hypothesis

Where rho is a measure of correlation, $-1 \leq \rho \leq 1$

a. Null hypothesis

$H_0: \rho(X,Y) = 0$, a necessary condition for
statistical independence

b. Alternative hypothesis

$H_1: \rho(X,Y) \neq 0$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user chooses which pair of variables from a data set are used in a test, and whether to exclude any of the pairs from the test. As described below, the computation of Kendall's tau uses an algorithm of complexity level n -squared (when comparing every pair with every other pair):

if there are more than 40 pairs Lochinvar offers the user the chance to skip the Kendall's tau test. If so, only Spearman's rho and Pearson's rho (which may have no meaning if the data is ordinal) will be computed. Otherwise, all three will be produced.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 10. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 10. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) If there are more than two variables in the data set, enter the indices of the two to be used. Enter pairs to be excluded, if any.

(2) If there are more than forty pairs, enter whether the Kendall's tau should be computed.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. Spearman's rho, Pearson's rho, and Kendall's tau are listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistics

a. Pearson's Rho

Pearson's rho is S_{XY} divided by square root of S_{XX} times S_{YY} . The deviations of X are found by subtracting the mean of X from each value of X; deviations of Y are Y_i minus mean of Y. The product of deviation of X_i times deviation of Y_i , summed over all pairs, is S_{XY} . The sum of squared deviations of X_i is S_{XX} ; the sum of squared deviations of Y_i is S_{YY} .

b. Spearman's Rho

Rank the values of X. Rank the values of Y. Those pairs of ranks ($R(X_i), R(Y_i)$), are then used to compute S_{XY} divided by square root of S_{XX} times S_{YY} . That is, Spearman's rho is Pearson's rho performed on the ranks of the values rather than the values themselves.

c. Kendall's Tau

Tau is roughly the measure of the number of concordant pairs minus discordant pairs divided by total number of possible pairing. Concordance implies positive correlation; e.g., if pair of the data set number 4 (X_4, Y_4) has an X value greater than, say, X_5 , and a Y value greater than Y_5 , then this is one small indication that there is positive correlation between X and Y. Therefore, the 4th and 5th pairs of the data set form to make one concordant pair. If X_6 is greater than X_4 , but Y_6 is less than X_4 , this is a little evidence of negative correlation; the 4th and 6th pairs of the set form to make one discordant pair. Ties in either X or Y mean that the comparison yields neither concordance nor discordance.

To compute tau, order the pairs (X_i, Y_i) in ascending order of X's. Compare the first pair (the one with the lowest X value) to every other pair. If X_1 is tied with

X_i or Y_i with Y_i , go to the next pair; if not, ($Y_i < Y_j$) implies concordance, ($Y_i > Y_j$) implies discordance; increment the relevant count. After the first pair is compared with all others, the second pair is compared with all succeeding pairs (3,4,...,k), incrementing counts as necessary. Each pair is compared to all succeeding pairs, until X_{k-1} is compared to X_k . Kendall's tau equals the number of concordant pairs minus the number of discordant pairs, all divided by the total possible number of pairings ($k(k-1)/2$). If all pairings are concordant, tau will equal 1; all discordant pairings will yield an -1.

2. Program Method of Operation

a. Pearson's and Spearman's Rhos

The subroutine Rhoer calculates S_{XY} , S_{XX} , S_{YY} for any input values, and returns--inter alia--rho. For Pearson's rho use the values of the data set. For Spearman's rho, use subroutine Rng to rank the values of the data set, and send those ranks to Rhoer.

b. Kendall's Tau

Since the values have already been ranked, use the indices of ranked X to order the pairs ($R(X_i), R(Y_i)$). Ranks work the same as values for finding concordance or discordance, but ties are easier to discern with ranks. With nested do for loops, compare the i th pair ($i = 1$ to $k-1$) with pairs $j = i+1$ to k ; $R(X_i) = R(X_j)$ or $R(Y_i) = R(Y_j)$ means to continue with looping. Otherwise, if $R(Y_i) < R(Y_j)$ then increment concordant counter, if $R(Y_i) > R(Y_j)$ then increment discordant counter. At the end of the nested loops, divide the difference of the concordant counter minus the discordant counter by the term $(k(k-1)/2)$.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of pairs: 200.

b. Subroutines/Procedures Used

- (1) Lochinvar: Correl8
- (2) Crunch: Correl, Rnq, Rhoer
- (3) External: Shsort

XV. NONPARAMETRIC REGRESSION

A. OVERVIEW

1. General Purpose

Regression tests the effect of the value of independent variable X has on the value of the independent variable Y. The linear regression model assumes that the effect of X on Y is linear, that is, $E(Y|X) = E(Y) + b_1(X - E(X))$. This test:

- (1) Finds estimates of b_1 .
- (2) Predicts values of Y given values of X
- (3) Test if the linear model is correct
- (4) Finds confidence limits for the value of b_1

2. Level of Measurement

Ordinal, to test the linear model

Interval, for the other purposes listed above

3. Assumptions

a. Random sampling of pairs. X may be specified, as long as the Y portion of the pair, given X, is independent.

b. The linear model is the correct model.

4. Hypothesis

Where b_1 is slope the parameter; b_1^* hypothesized.

a. Null hypothesis

$H_0: b_1 = b_1^*$

b. Alternative hypothesis

$H_1: b_1 \neq b_1^*$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

a. The user chooses the two variables to be used, and designates which one is dependent. The program will

automatically fit the least squares model, finding b_1 . The intercept parameter b_0 , $E(Y|X) = b_0 + b_1 * X$, is also computed.

b. The user can input values for X to predict values of Y based on the least squares parameters.

c. The user can specify a hypothesized b_1^* to test whether a linear model is appropriate. The default value is $b_1^* = 0$; this is the classical test of whether an independent variable predicts a dependent variable.

d. The user can request confidence intervals on the slope parameter b_1 . As described below, the algorithm to compute those intervals is of degree of complexity n squared, that is, for n pairs (x, y) , $n(n-1)/2$ comparisons must be made and stored to find the confidence intervals. For larger values of n , the user should consider not requesting this option.

If the user does choose to have the confidence intervals computed, the value W will be requested by Lochinvar. That value is explained by Conover [Ref. 1: p. 267], and contained in table A-12 of that reference.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 11. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 11. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter pairs to be excluded, if any.

(2) Enter values of X to compute $E(Y|X)$ based on least squares linear fit.

(3) Enter hypothesized slope parameter b_1^* .

(4) Enter whether confidence limits are desired.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The least squares fit parameters will be listed, and the Spearman's rho statistic for the specified value of b_1 . If the user requested expected values of Y for some values of X, the values of X and Y will be listed. Finally, if confidence bounds were requested, those will be given.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic/Level of Significance

a. Find the Regression Parameters b_1, b_0

This step is identical to parametric simple regression. Find the means of X and Y, and the vectors of deviations ($X_i - \text{mean of X}$), ($Y_i - \text{mean of Y}$). Multiply the corresponding deviations of X and Y together, and sum the products to form S_{XY} . Square the deviations of X, then sum those squares, to form S_{XX} . Sum the squared deviations of Y to find S_{YY} . The slope parameter b_1 is S_{XY}/S_{XX} . The intercept parameter b_0 is the mean of Y minus b_1 times the mean of X. The expected value of Y given X is b_0 plus b_1 times X. Residuals are the difference between the expected and actual value of Y.

b. Test the Value of X as a Predictor of Y

In parametric regression, it is standard to test whether the slope parameter b_1 could be 0 in the entire population. In this package, any value b_1^* can be set as the null hypothesis. The test checks to see if the residuals of the fit using b_1^* are correlated with the values of X; if not, then the fit is considered adequate. The residuals are $(Y_i - b_1^* X_i)$, the vector of residuals and the independent variable X are compared using the Spearman's rho test. See section XIV for details of that test.

c. Predictions of Y based on Least Squares Fit

With the least squares parameters b_0 and b_1 already computed, Crunch multiplies each given X times b_1 , adds b_0 , and displays the predicted Y.

d. Confidence Bounds for the Slope Parameter b_1

Form the vector of slopes S. $S(1)$ is $(X(1) - X(2))$ divided by $(Y(1) - Y(2))$; $S(2)$ is $(X(1) - X(3)) / (Y(1) - Y(3))$, and so on. With k pairs (X, Y) , there are $N = k(k-1)/2$ elements of the slope vector, when comparing each i with each other j once.

Sort the vector S. The indices of the confidence limits are r ($0.5 * (N - W)$) and s ($0.5 * (N + W) + 1$), where N is the total number of slopes and W is the user entered value from table A-12 of Conover. If necessary, round r or s up to integers. The lower limit on b_1 is $S(r)$, the upper limit is $S(s)$.

2. Program Method of Operation

The subroutine Regret calls the subroutine Rhoer to find the least squares parameters b_1 and b_0 . Regret figures the residuals for the specified b_1^* , and sends the residuals to subroutine Correl to have Spearman's rho calculated. Regret figures the vector of slopes S; that vector is sorted by Sshort; Regret finds indices s and r and the confidence limits to b_1 .

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum
number of variables: 2.

b. Subroutines/Procedures Used

- (1) Lochinvar: Regret
- (2) Crunch: Regret, Rhoer, Correl
- (3) External: Shsort

XVI. MONOTONIC REGRESSION

A. OVERVIEW

1. General Purpose

With independent variable X , and dependent variable Y , find the $E(Y|X)$ equation, based on a least squares fit of the ranks of X and Y . Linear interpolation finds values from X to $E(\text{rank of } X)$ to $E(\text{rank of } Y)$ to $E(Y)$.

2. Level of Measurement

Interval

3. Assumptions

- a. The sample is a random sample
- b. The regression of Y on X is monotonic.

4. Hypothesis

This test does not produce a test statistic; the assumption is that the monotonic model is correct. Instead, it will predict expected values of Y for given values of X . Also, it can produce a trace of the monotone regression, which is analogous to the straight line of the least squares fit of linear regression.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must specify the index for the independent value X , and dependent variable Y . The user can include values of X to have predicted a corresponding value of Y . The user can request a trace of the monotone regression.

2. Program Activation

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 12. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 12. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables X and Y. Enter pairs to be excluded, if any.

(2) Enter whether to predict values of Y for specified values of X. If yes, enter values of X.

(3) Enter whether a trace is desired.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. In all cases, there will be listed the parameters b_0 and b_1 , the intercept and slope of the least squares fit on the ranks of X and Y. If the user specified X's to predict Y's, the predicted values will be listed. Finally, if a trace was requested, the pairs of X's and Y's will be given.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculations

a. Least Squares Fit on Ranks of X and Y

Sort and rank X and Y. With n pairs, $X(i)$ and $Y(j)$ are the order statistics, $R_X(i)$ and $R_Y(j)$ are the ranks, $i=1, \dots, n$, $h=1, \dots, n$. Compute the least squares fit

to find b_0 and b_1 , such that $E(RY|RX) = b_0 + b_1 * RX$. See section XV for details of least square computations.

b. Predication of Y Based on Values of X

Linear regression will extrapolate values of Y for a given value of X if that X is outside the range of the observed values $X(1), \dots, X(n)$. Monotone regression will not extrapolate; if the user specifies an X outside the range of the observations, that X will not yield an expected Y.

Let X_p be one of the values the user specified to predict a Y, Y_p . Find the pair of values of the order statistics of X which bracket X_p , $X(i) \leq X_p \leq X(i+1)$. The expected rank of X_p , RX_p , must fall between those order statistics' ranks, $RX(i) \leq RX_p \leq RX(i+1)$. Use linear interpolation to find RX_p : $RX_p = RX(i) + (X_p - X(i)) / (X(i+1) - X(i)) * (RX(i+1) - RX(i))$.

Use the parameters b_0 and b_1 to find RY_p , the rank of Y_p : $RY_p = b_0 + b_1 * RX_p$.

Find the ranks of Y which bracket RY_p , $RY(j) \leq RY_p \leq RY(j+1)$. The value of Y_p lies between $Y(j)$ and $Y(j+1)$. Use linear interpolation to find Y_p : $Y_p = Y(j) + (RY_p - RY(j)) / (RY(j+1) - RY(j)) * (Y(j+1) - Y(j))$.

c. Trace of Monotone Regression

The trace is a set of lines, between a sequence of points (X_i, Y_i) , which describe the monotone regression. When plotted, the trace can be used to predict a value of Y for an X. This is similar to using the straight line of linear regression, $Y = b_0 + b_1 * X$, for prediction.

To find the endpoints of the trace, use $X(1)$ and $X(n)$, and the procedure of part b, to find $E(Y|X(1))$ and $E(Y|X(n))$.

Intermediate points of the trace will use the values of Y to predict expected X. Each value of Y has a rank, RY , which is used with b_0 and b_1 to find RX^t (trace).

$RX_t = (RY - b_0) / b_1$. For each RX_t , find the pair of RX 's which bracket it. Use linear interpolation to find X_t .

2. Program Method of Operation

The subroutine Monore calls Rnq to rank and sort values. The subroutine Correl finds the least squares parameters b_0 and b_1 from the rank vectors RX and RY . The subroutine ESTY performs estimations of Y for given X 's, calling Rtov (which calls Interp) to do the linear interpolations. The subroutine Trace performs the trace, calling Vtor, Rtov and Interp.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

- (1) Lochinvar: Monoreg
- (2) Crunch: Monore, Rtov, Vtor, Esty, Interp, Rnq
- (3) External: Shsort

XXVII. MEDIAN TEST

A. OVERVIEW

1. General Purpose

With two or more samples, this test will determine if all the populations from which the samples were drawn could have the same median.

As described below, the package allows combining of data set variables into composite variables in order to test population groups for common median.

The median may be a surrogate for expected value.

2. Level of Measurement

Ordinal

3. Assumptions

The variables were randomly drawn from the respective populations, and are mutually independent.

4. Hypothesis

With k populations, $X(1), X(2), \dots, X(k)$, and the median denoted as M ,

a. Null hypothesis

$$H_0: M(X_1) = M(X_2) = \dots = M(X_k)$$

b. Alternative hypothesis

$$H_1: M(X_i) \neq M(X_j) \text{ for at least one of the pairs}$$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user chooses which variables of the data set to include in the test.

Aggregate variables may be formed by having Crunch combine the expected cell counts of several variables into

one or more aggregate variables. For example, let data set Farmland contain six variables, in order, A through F. Variable A is corn yield per acre of farm land treated with fertilizer and pesticide. B is corn yield with fertilizer but no pesticide. C is corn yield with pesticide but no fertilizer. D is corn yield per acre from land untreated by either fertilizer or pesticide. Variables E and F have other values. Variables A and C could be combined into aggregate variable 1, and B and D into 2, to test whether the median yield with pesticide is equal to the median yield without pesticide. Similarly, A and B, and C and D, could be combined into aggregate variables 1 and 2, to test the median of yields with and without fertilizer.

Proper planning and labeling of the variables will facilitate this aggregation.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 13. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 13. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter blocks or pairs to be excluded, if any. Enter whether to use aggregate variables.

(1) If aggregate variables are not used, enter indices of variables to be used in the test.

(2) If aggregates are used, Lochinvar will prompt for a choice on each data set variable, $i=1$ to the number of variables in the data set. If the i th data set variable is to be excluded from the test, enter 0. If it is to be included, enter the index of the aggregate variable into which it should be combined. For example, with the Farmland data set described above, Lochinvar would prompt user choices on data set variable $i= 1$ to 6. Let the test be a comparison of yields with or without pesticide, thus variables A and C into aggregate variable 1, B and D into 2, and E and F excluded. When prompted for the data set var 1, enter 1. For data set var 2, enter 2. For data set var 3, enter 1. For data set var 4, enter 2. For data set var 5 and 6, enter 0.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The program will print the test statistic and the degrees of freedom, as well as a two by number of variables contingency table, showing how many elements of each variable are less than or equal to the grand median of all variables, and how many are greater than.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculation of the Cell Counts of the M Matrix

The M matrix has two rows and a column for each variable in the test. The A row has the number of elements in each variable less than or equal to the grand median; the

B row has the number of elements greater than the grand median. The values of all the variables are copied into a vector, which is sorted; the median of all those values is the grand median. The counts of the cells of the matrix M are set to 0. Each element of the first variable is compared to the grand median; if less than or equal, the A row first column cell count is incremented, otherwise increment B row first column cell count. After all elements of the first variable are compared to the grand mean, do the second variable, incrementing the 2 column's cell count, et cetera.

b. Computing the Test Statistic

Sum all counts of the A row to find A_{tot} , and all counts of B to find B_{tot} . Sum A_{tot} and B_{tot} to find N . Sum the first column counts to find $NAV(1)$, the second column counts to find $NAV(2)$, et cetera. The expected cell count for cell a_j is $A_{tot} * NAV(j) / N$; for b_j , $B_{tot} * NAV(j) / N$. The test is in essence the chi-square test for independence ($r * c$ contingency table), with the test statistic as the sum over each cell of observed minus expected count squared, divided by the cell expected count. The actual equations used exploit the structure of the problem. The degrees of freedom are $(r-1)(c-1) = \text{number of variables minus one}$.

2. Program Method of Operation

a. Aggregation of Variable

The subroutine Media assumes that aggregate variables are always used. When not, the program uses k aggregate variables where k equals the number of data set variables included in the test; aggregate variable 1 is data set variable 1, ..., aggregate variable k is data set variable k . When aggregates are in fact specified, their values, contained in rp vector, will be used. In either case, the vector AGVARI will contain the indices relating each data set variable to an aggregate variable.

b. Cell counts

The matrix M is a fiction. The vector NA is the count of elements less than or equal to the grand median; it is arranged according to aggregate variable indices. NB is the count of elements greater than the grand median. In a do for loop, i=1 to the number of data set variables, k is set to AGVARI(i), and each element of the ith data set variable is compared to the grand median. If the element is less than or equal to the grand median, increment NA(k) (=NA(AGVARI(i))), otherwise, increment NB(k).

c. Compute the Test Statistic

NA(i) and NB(i) are summed to find NAV(i). All elements of NA are summed to find A, all elements of NB are summed to find B. A plus B equals N. The test statistic is computed using the specialized equations of Conover [ref 1, p. 172]. The degrees of freedom are the number of aggregate variables minus one.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables or aggregate variables: 15.

b. Subroutines/Procedures Used

- (1) Lochinvar: Median, Ag_med, Which_vec, None_ex
- (2) Crunch: Media
- (3) External: Shsort

XVIII. KRUSKAL WALLIS TEST

A. OVERVIEW

1. General Purpose

With a data set of more than two variables, X_1, X_2, \dots, X_k , test whether the expected values of all variables are equal.

If this test's null hypothesis is rejected, then it is possible to determine which of the pairs of variables have an unequal mean.

If the data set is blocked, consider instead of this test the Cochran (nominal), Friedman (ordinal) or Quade (interval) tests. If paired, consider the Sign (ordinal) or Wilcoxon (interval) tests.

2. Level of Measurement

Ordinal

3. Assumptions

a. Each variable is a random sample from within its population.

b. The sampling was mutually independent among the populations.

c. The population distributions are identical, or else some of the populations tend to have higher values than others.

4. Hypothesis

a. Null hypothesis

$$H_0: E(X_1) = E(X_2) = \dots = E(X_k)$$

b. Alternative hypothesis

$H_1: E(X_i) \neq E(X_j)$ for at least one pair of populations represented by the data set.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must specify which variables of a data set are included in the test.

After the test is run, Crunch will present the overall test statistic, and a series of comparisons between the variables. If, and only if, the null hypothesis has been rejected by the overall test statistic may the user infer any information by the comparisons between the pairs (X_i, X_j) of variables.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 14. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 14. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar, enter indices of variables to be used in test.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The overall test statistic will be given. Also listed will be comparisons of pairs of variables; these

comparisons only have meaning if the null hypothesis has been rejected. The paired comparison are essentially a t test run on the ranks of the two variables being compared, testing the hypothesis that their mean ranks are the same. Crunch supplies the absolute difference in their mean ranks and the pooled standard deviation; the user must find the appropriate value of t to conduct the t test.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Calculation of Test Statistic for the Null Hypothesis

a. Find the Sums of Variable Ranks R_i

Rank all values in the data set. Sum the ranks in the i th variable to get the value R_i . The number of data elements in the i th variable is n_i , so the average rank in the i th variable is R_i/n_i . The sum of n_i over all k variables is N .

b. Find the Test Statistic

The test statistic numerator is the sum over k treatments of $(R_i^2)/n_i$, minus $((N+1)^2)*N/4$. To find the test statistic denominator, S^2 , first sum over all data elements the square of the ranks. Subtract from this sum of squares the term $((N+1)^2)*N/4$. Divide that difference by $(N-1)$ to get S^2 . Divide the numerator by S^2 to find the test statistic T .

c. Make Comparisons of the Pairs of Variables

As mentioned above, this step is the student t test on the ranks of two variables to see if their population mean could be equal; it means nothing if the null hypothesis has not been rejected.

For each pair of variables X_i, X_j , find the absolute difference in mean ranks, $|R_i/n_i - R_j/n_j|$. Find the pooled variance, $(S^2*(N-1-T)/(N-k))$ times $(1/n_i + 1/n_j)$. The

pooled standard deviation is the square root of the pooled variance. If the absolute difference is greater than the pooled standard deviation times the desired value of t , reject the assertion that the mean of populations i and j are the same.

2. Program Method of Operation

The subroutine Kruwal calls Ranque to rank the values of the data set. In a nested do for loop, $i=1$ to k , $j=1$ to n_i , the intermediate terms of the statistics are computed. Kruwal prints the overall test statistic to the specified output file. It then makes pairwise listings of the absolute difference of mean ranks and the corresponding pooled standard deviations.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

- (1) Lochinvar: Kruwal
- (2) Crunch: Kruwal, Ranque
- (3) External: Shsort

XIX. SQUARED RANK TEST FOR EQUAL VARIANCE

A. OVERVIEW

1. General Purpose

With two or more treatments in a data set, test whether the population variances of those treatments are equal.

2. Level of Measurement

Interval

3. Assumptions:

Random sampling within each treatment population, and among treatment populations. Level of measurement is interval.

4. Hypotheses

a. Two tailed test, where X,Y are treatments

(1) Null hypothesis $h_0: \text{var}(X) = \text{var}(Y)$

(2) Alternative hypothesis $h_1: \text{var}(X) \neq \text{var}(Y)$

b. One tailed test, two treatments

(1) Null hypothesis $h_0: \text{var}(X) \geq \text{var}(Y)$

(2) Alternative hypothesis $h_1: \text{var}(X) < \text{var}(Y)$

c. One tailed test, two treatments

(1) Null hypothesis $h_0: \text{var}(X) \leq \text{var}(Y)$

(2) Alternative hypothesis $h_1: \text{var}(X) > \text{var}(Y)$

d. Number of treatments (X_i) are three or more

(1) Null hypothesis $h_0: \text{var}(X_1) = \text{var}(X_2) = \dots = \text{var}(X_k)$

(2) Alternative hypothesis $h_1: \text{var}(X_i) \neq \text{var}(X_j)$,
for some i, j , indices of the data set

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The program allows the user to choose which treatments to include in the test.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 15. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 15. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar, Enter indices of variables to be used in test.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables appear on the top of the output file. Also included are the test statistic and the number of treatments used to compute the statistic.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic (T1/T2)

a. Find the ranks of each absolute deviation:

Compute the sample mean for each of k treatments. Subtract the treatment mean from each of the treatment's elements. Take absolute values of each of those differences. Rank all N absolute differences (N=total elements in data set).

b. Complete intermediate statistics:

- (1) SUMR, the sum of all ranks.
- (2) SUMR2, the sum of the square of all ranks.
- (3) SUMR4, the sum of all the fourth power of each rank.
- (4) SUMRJ, the k element vector with the sum of the ranks for each treatment.
- (5) SUMR2J, the k element vector with the sum of squared ranks for each treatment.
- (6) R2MU, the average squared rank: $SUMR2/N$.
- (7) RMU, the average rank: $SUMR/N$.

c. For two treatments compute the statistic T1

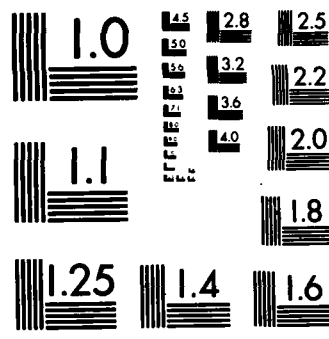
- (1) Numerator: $(SUMR2J(1) - n1 * R2MU)$
- (2) Denominator: $SUMR4 * n1 * n2 / (N * (N - 1))$, minus $(RMU ** 2) * n1 * n2 / (N - 1)$
- (3) $T1 = \text{numerator} / \text{denominator}$

d. For three or more treatments compute T2

- (1) Numerator: difference between these two terms, sum over k treatments of $(SUMR2J ** 2) / ni$, minus $(R2MU ** 2) * N$
- (2) Denominator: $(SUMR4 - N * (R2MU ** 2)) / (N - 1)$
- (3) $T2 = \text{numerator} / \text{denominator}$

2. Program Operations

The program uses subroutine MUVAR to find treatment means. The subroutine Squark finds the absolute deviations;



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

while the subroutine Rnq ranks them. The subroutine Squark Uses nest "do for" loops, $i=1$ to k (treatments) and $j=1$ to n_i (elements in treatment), to calculate the vectors SUMRJ and SUMR2J, and the sums SUMR, SUMR2, SUMR4. RMU and R2MU are found by dividing SUMR and SUMR2 by N . Depending on the number of treatments, T1 or T2 is found by simple arithmetic using the intermediate statistics.

3. Program Characteristics

a. Limitations

Maximum number of data elements is 400. Maximum number of treatments is 15.

b. Subroutines/Procedures Used

- (1) Lochinvar: Squarank, which_vec
- (2) Crunch: Squark, Muvar, Rnq

XX. HARTLEY TEST FOR EQUAL VARIANCE

A. OVERVIEW

1. General Purpose

Test for equal variance among the k underlying populations of the variables or treatments, reflected by the values for those treatments in the data set.

2. Level of Measurement

Interval

3. Assumptions

Random sampling within each treatment, independence of sampling among treatments. Interval level of measurement

4. Hypothesis

a. Null hypothesis

$H_0: \text{var}(X_1) = \text{var}(X_2) = \dots = \text{var}(X_k)$, where k = number of treatments

b. Alternative hypothesis

$H_1: \text{var}(X_i) \neq \text{var}(X_j)$ for at least one pair

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The program allows the user to choose which treatments to include in the test.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for initial choice details.

In CMS, type L0CHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 16. If data needs to be entered, enter "N" for first question, "V" for second; when data

entered, choose to do a test; when offered a choice of tests, type 16. Enter "U", "V", ... "Z" for desired output file.

b. **Variable and Parameter Choice**

When prompted by Lochiavar, Enter indices of variables to be used in test.

4. **Test Execution**

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. **Output Available to the User**

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. There are the maximum and minimum variances and indices of the associated treatments. Finally, there is the F statistic, to be compared to a table prepared for the Hartley test.

C. **ACTIONS PERFORMED BY THE PROGRAM CRUNCH**

1. **Description of Calculation of Test Statistic**

Crunch does the following: Compute the sample variance for each treatment. Find the smallest and largest variance. Divide the largest variance by the smallest variance to yield the test statistic F. Compare the test statistic to a special table to see if the null hypothesis should be rejected.

2. **Program Method of Operation**

The treatment variances are figured in the subroutine NUVAR. The subroutine HARTLE scans the variances to find the largest and the smallest, divides the former by the latter, and gives that number as the test statistic F.

3. Program Characteristics

a. Limitations

The maximum number of data elements is 400; the maximum number of treatments is 15.

b. Subroutine/Procedures Involved

- (1) Lochinvar: Hartley, Which_vec
- (2) Crunch: Hartle, Muvar
- (3) External: Shsort

XXI. FRIEDMAN TEST AND DURBIN TEST

A. OVERVIEW

1. General Purpose

To perform a treatment and block test, similar to ANOVA, using only ordinal data. In fact, the data may be rankable only within each block; i.e., no relationship between data values in different blocks is necessary.

The Friedman test is for data sets of complete block design, i.e., each treatment is applied to each block.

The Durbin test is a generalized Friedman test, for use with incomplete block design. With t treatments and b blocks, each treatment is applied to r blocks ($r < b$), and each block receives k treatments ($k < t$). The user must guarantee that the design is balanced, and insert a filler value in place of missing data values when entering the data set.

Each test will generate a set of differences between pairs of treatment summed ranks. If, and only if, the null hypothesis is rejected, these differences may be used, in conjunction with a standard deviation value and the student's t distribution, to determine if there is a significant difference between two treatments.

2. Level of Measurement

Ordinal within each block.

3. Assumptions

Values within blocks are independent among blocks.

4. Hypothesis

a. Null hypothesis

H_0 : Each possible ranking within the block is equally likely (that is, the treatment effects are equal)

b. Alternative hypothesis

H1: At least one of the treatments tends to get higher ranks than at least one other treatment.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must decide which treatments and blocks to include in the test. If a incomplete block design is to be used, the user must specifically enter the data set for this purpose: there will be t variables, and b elements entered for each variable. In those elements which have a no data value (a treatment which was not applied to some block), enter a filler value. In a data set, there can be only one filler value, and it must be different than any of the data values. For example, if all the data values are positive, -1 could be used as a filler.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 17. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 17. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter whether to use the Friedman or Durbin test.

(2) If the Friedman test is chosen, enter the treatments to be used and any blocks to be excluded.

(2) If the Durbin test is chosen, enter the treatments to be used, any blocks to be excluded, k = number of treatments applied to each block, r = number of blocks to which each treatment is applied, and the filler value.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The overall test statistic for the null hypothesis will be given. In addition, the difference between pairs of treatments' summed ranks will be listed; these are meaningful only if the null hypothesis is rejected.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Use of Filler, If Durbin Test

If the Durbin test is indicated, find the maximum value in the data set. Add 1 to that maximum value to form $MP1$. In a do for loop, substitute $MP1$ for the filler value, and create the integer vector $INTEST$. $INTEST(i)$ equals 1 if the i th element in the data set is not the filler value, 0 if the i th element is a filler.

b. Rank Values Block by Block

In both tests, find the ranks of the values of the data set block by block. Load those ranks into the R vector.

In the Durbin test, the filler elements will have a higher rank than those of data elements, since the fillers were loaded with $MP1$. Therefore, after all the ranking is complete, multiply the rank vector times $INTEST$; this sets the ranks of all filler elements to 0.

c. Sums of Treatment Ranks and Total Sums

Square each rank of the rank vector R , and sum those squares over all treatments and blocks to form $A2$. Sum the ranks of each treatment to form the vector RJ ; $RJ(j)$ contains the sum of ranks of the j th treatment.

d. Test Statistic for the Friedman Test

Square each element of the RJ matrix; sum those squares; divide that sum of squares by the number of blocks to find $B2$. The denominator of the test statistic is $A2 - B2$. Let b be the number of blocks, t the number of treatments. The numerator of the test statistic is $(b-1) * (B2 - 0.25bk * (k+1)**2)$. Divide the numerator by the denominator to get the test statistic $T2$. Compare $T2$ to the F distribution with $(k-1)$ and $(k-1)(b-1)$ degrees of freedom

e. Test Statistic for the Durbin Test

Sum the squares of RJ . Multiply that sum of squares time $12(t-1)$. Divide that product by $rt(k-1)(k+1)$, where r is the number of blocks to which each treatment is applied, and k is the number of treatments applied to each block. This quotient is the first term of the test statistic. The second term is $3r(t-1)(k+1)$ divided by $(k-1)$. Subtract the second term from the first term to find the test statistic T . T is compared to the chi-square distribution with $t-1$ degrees of freedom.

f. Paired Comparisons of Treatments

Crunch will print the following numbers in all cases, but they only have meaning when the null hypothesis is rejected. For each i, j of the treatments, the absolute difference between the sums of treatment ranks is listed. The number that is equivalent to the standard deviation is also computed. For absolute differences greater than the standard deviation times the t quantile $(1-\alpha/2)$, the treatments can be considered to have a different effect.

2. Program Method of Operation

The subroutine Friedm performs most of the computations. The subroutine Ranque ranks the values block by block. The subroutine Parcom does the comparisons of treatment sums of ranks.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

- (1) Lochinvar: Friedman
- (2) Crunch: Friedm, Ranque, Rnq, Parcom
- (3) External: Shsort

XXII. QUADE TEST

A. OVERVIEW

1. General Purpose

With interval data organized into t treatments and b blocks, test whether all treatments have the same effect. If not, treatments can be compared for differences in pairs.

2. Level of Measurement

Interval

3. Assumptions

a. The values of each blocks are independent of the values of the other blocks.

b. The sampling within each block can yield a range of the elements; that range must be rankable among blocks

4. Hypothesis

a. Null hypothesis

H_0 : Each possible ranking within the block is equally likely (that is, the treatment effects are equal)

b. Alternative hypothesis

H_1 : At least one of the treatments tends to get higher ranks than at least one other treatment.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must specify the treatments to be included in the test, and whether any blocks are to be excluded. Crunch will provide the absolute difference between pairs of treatments; if and only if the null hypothesis is rejected, these comparisons can be used to test whether the effects of the pairs are equal.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 18. If data needs to be entered, enter "N" for first question, "V" for second; enter the data; when offered a choice of tests, type 18. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar: Enter indices of variables to be used in test.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic T and its degrees of freedom are listed. Finally, the absolute difference between the pairs of sums of treatment ranks are given. These differences, when used with a standard deviation computed by Crunch and the t distribution, can be used to check for equal effect between pairs of treatments.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Rankings within and among Blocks

Rank each element in the block; assign the elements to the matrix of ranks R. Subtract the largest element from the smallest element; that becomes the block's

range, stored in the vector R. After all blocks have had their elements ranked, rank the vector R; those ranks form the vector Q.

b. The Size Matrix S

With t equal to the number of treatments, the average rank of R is $(t+1)/2$. Subtract that average rank from each element of R. Multiply the reduced ranks of the first block by Q(1), the reduced ranks of the second block by Q(2), et cetera, to form the matrix S.

c. Sums of the Treatment Elements of S

Sum the elements of the jth treatment of S over all blocks to form the jth element of the vector SJ. Square each element of SJ, sum those squares over all treatment, and divide by b, the number of blocks, to find B1. Square all elements of S, and sum all those squares to find A1.

d. Test Statistic

The test statistic T1 is $(b-1)$ time B1 divided by the quantity A1 minus B1. Compare T1 to an F statistic with degrees of freedom $(t-1)$ and $(b-1)(t-1)$.

e. Pairwise Comparisons

These values have meaning only if the null hypothesis is rejected. Compute the absolute difference between each pair of the vector SJ. If that difference is greater than the student's t quantile times the computed standard deviation, the pair of treatments can be said to have a different effect. The standard deviation is found by taking the square root of $(2b(A1-B1)/(b-1)(t-1))$.

2. Program Method of Operation

The subroutine Quade calls Ranque to find the matrix R and the vector Q. The test statistic is computed in Quade, as is the standard deviation. The subroutine Parcom makes the pairwise comparison.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum
number of variables/treatments: 15.

b. Subroutines/Procedures Used

- (1) Lochinvar: Quade
- (2) Crunch: Quade, Ranque, Rnq, Parcom
- (3) External: Shsort

XXIII. COCHRAN TEST FOR RELATED OBSERVATIONS

A. OVERVIEW

1. General Purpose

With a data set of b blocks, to which k treatments have been applied, test whether the treatments have been equally effective. "Effective" must be reducible to two categories, mutually exclusive and collectively exhaustive. Here, these categories will be called success and failure.

If the data is not reducible in this fashion, the Friedman (ordinal) or Quade (interval) tests are available.

2. Level of Measurement

Nominal

3. Assumptions

The blocks are a random sample from among all blocks

4. Hypothesis

a. Null hypothesis

H_0 : The treatments are equally effective

b. Alternative hypothesis

H_1 : There is a difference in effectiveness among the groups.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must decide the standard for success or failure of each treatment. If the data is nominal, the user should enter cell counts. See paragraph 2, below. If the data is ordinal, the user may enter a partition value for each treatment of a data set; see paragraph 3. If this latter method is used, Crunch will classify each value of an

entered data set as a success or failure by its relationship to the partition value for its treatment.

If partition values are used in conjunction with a data set, the user may specify which variables of the data set are to be included in the test, and may designate blocks to be excluded.

2. Program Activation when Using Cell Counts

a. Initiation and test selection

In CMS, type LOCHI. Enter "N" for no to question about previously entered data. Enter "C" for cell counts. Enter "U", "V", ... "Z" for desired output file. When offered a choice of tests, enter 19.

b. Cell count entry

Lochinvar will ask how many treatments and how many blocks are included in the test. After those numbers are entered, Lochinvar will ask whether block 1, treatment 1 was a success. Enter yes or no. The program will prompt for each treatment in block 1, then move to block 2, and so forth, through the b*k choices. The matrix of 1's (successes) and 0's (failures) will be displayed, and the user has the chance to correct the values for any block (the program will prompt for entries of each treatment of a specified block).

3. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 19. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 19. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

Lochinvar will ask if treatment 1 is in the test. If yes, then it will then ask the partition value for treatment 1, whether successes lie above or below that value. The same questions will be asked for each treatment. After each treatment is either excluded or given a partition value, Lochinvar will offer the user the chance to exclude blocks from the test.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic T will be listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculation of Block and Treatment Totals

If the user chooses the method of cell counts (paragraph 2, above), Lochinvar will sum the successes over each of b blocks and over each of k treatments. These totals will be in the cells vector of line 3a of the options file (see Appendix D for details of the options file), to be read by Crunch.

If the user chooses to enter partition values (paragraph 3 above), Crunch uses that information to compute the block and treatment totals.

b. Computing the Test Statistic

The numerator of the test statistic involves the sums of successes for each treatment. The total number of successes N is found by summing each treatment number of

successes C_i ; N divided by the number of treatments k is the average number of successes per treatment. Find the sum of squared deviations by summing over k treatments the square of $(C_i - N/k)$. Multiply that sum of squares by $k(k-1)$ to get the test statistic numerator.

The denominator of the test statistic is the sum over all b blocks of the number of successes in block i times the number of failures in block i .

Divide the numerator by the denominator.

2. Program Method of Operation

a. When Using Cell Counts

In the procedure Cochran, Lochinvar prompts the answer of yes or no to the question of whether block i , treatment j , is a success. If yes, it increments the total of successes in the count for block i and treatment j . The total for block i is contained in the i th element of the vector CELLS; the total for the j th treatment is contained in the $(j+b)$ th element of CELLS, where b is the number of blocks. The number of blocks (NROW) and number of treatments (NCOL) are written in Line 3 of the options file; the vector CELLS is written in Line 3a. The Crunch subroutine Celler reads those values, and sends them to the subroutine Cochran, which computes the test statistic.

b. When using Partition Values

In the procedure Cochran, Lochinvar asks the partition value of each treatment included in the test; those values are entered in the odd numbered elements of vector PART_V, that is, $1, 3, \dots, (2k-1)$, where k is the number of treatments. In the even numbered elements of PART_V is a $+1.0$ if successes are below the preceding partition value, or -1.0 if the successes are above it. For example, PART_V of 3.0 -1.0 7.0 1.0 means that there are two treatments; in the first treatment, successes are above 3.0, in the second, successes are below 7.0. The number of

blocks (NROW) and the number of treatments (NCOL) are written in the Line 3 of the options file; PART_V is written into line 3a.

The Crunch subroutine Caller calls the subroutine Coch to use PART_V and the data elements values to find the totals of successes for the blocks and treatments. It places those counts (in the same format as CELLS, describe in subparagraph a, above) into the vector RCCT. That vector is used by the subroutine Cochrn to compute the test statistic.

3. Program Characteristics

a. Limitations

When the user enters cell counts, the sum of the treatments and blocks cannot exceed 50 ($b+k$), and the total number of cells ($b*k$) cannot exceed 400.

When the program computes the treatment and block totals, the maximum number of data points is 400; the maximum number of treatments is 15. The sum of treatments and blocks ($b+k$) cannot exceed 50.

b. Subroutines/Procedures Used

- (1) Lochinvar: Cochran, Cel_test
- (2) Crunch: Cochrn, Caller
- (3) External: Shsort

XXIV. KOLMOGOROV TEST

A. OVERVIEW

1. General Purpose

With a sample of 30 or less data points, test to see whether the population from which the sample is drawn is of a specified distribution with specified parameters. The choice of hypothesised distributions is uniform, normal, exponential, weibull, and erlang (gamma with an integer shape parameter).

The Lilliefors test is identical to the Kolmogorov test except that it estimates parameters from the sample; normal and exponential distributions can be hypothesised.

2. Level of Measurement

Interval

3. Assumptions

The sample is a random sample from the population.

4. Hypotheses

With $F^*(x)$ as the theoretical distribution with specified parameters, and $S(x)$ as the population's underlying distribution function, the hypotheses are:

a. Two tailed test

(1) Null hypothesis h_0 : $F^*(x) = S(x)$ for all x

(2) Alternative hypothesis h_1 : $F^*(x) \neq S(x)$ for

at least one x in range of sample

b. One tailed test

(1) Null hypothesis h_0 : $F^*(x) \leq S(x)$ for all x

(2) Alternative hypothesis h_1 : $F^*(x) > S(x)$

c. One tailed test

(1) Null hypothesis h_0 : $F^*(x) \geq S(x)$

(2) Alternative hypothesis h_1 : $F^*(x) < S(x)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If a data set has more than one variable, the user must specify which is used. If the chosen variable has more than 30 elements, the chi-square goodness of fit test will automatically be given. Otherwise, the user must specify the distribution to use, and all its parameters, from among the following:

- a. Normal (specify μ , σ^2)
- b. Exponential (specify λ =rate, scale)
- c. Uniform (specify a =lower bound, b =upper bound)
- d. Erlang (specify n =integer, shape; λ =scale)
- e. Weibull (specify α =shape, λ =scale)

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 20. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 20. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

- (1) Enter index of variable to be used in test.
- (2) Enter number of theoretical distribution.
- (3) Enter parameters as prompted.
- (4) Enter letter of type hypothesis.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The values of the largest difference $F^*(x) - S(x)$, the smallest (most negative) difference, and the greatest absolute difference are displayed. Depending on the hypothesis, Crunch selects the proper value of these three as the test statistic, and displays that value.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Empirical Cumulative Distribution Function (CDF)

Sort the data elements, giving order statistics $X(1), \dots, X(n)$, where n is the number of data elements. The empirical cdf equals 0 for values of X lower than the smallest X , $X(1)$. At $X(1)$, the cdf jumps to $1/n$; the cdf increases by $1/n$ at each new order statistic. At $X(n)$, the empirical cdf equals 1, and remains 1 for all values of x greater than the sample. The empirical cdf is stored in the vector S .

b. Points on the Theoretical CDF

With a specified distribution and parameters, find the value of the cdf for each value of the order statistics $X(1), \dots, X(n)$. For example, with exponential ($\lambda=2$) specified and $X=1.5$, the value of the theoretical cdf is $1.0 - \exp(-(2*1.5)) = 0.9502$. Store these values in the vector F .

c. Difference between Empirical and Theoretical CDF's

With a sample of size n , there are $2n$ differences between S and F . For example, at $X(1)$, S jumps from 0 to $1/n$, so that $F(1)$ is compared to both values. At each order statistic value, the theoretical distribution $F(i)$ is compared to $S(i-1)$ and $S(i)$. Crunch keeps track of the t_{min} (most negative difference, or if all are positive, the smallest positive), and t_{max} (most positive, or if all are negative, the closest to zero).

d. Test Statistics

For a two tailed test (hypothesis A), the test statistic is the larger of the absolute values of t_{max} and t_{min} . For hypothesis B, t_{max} is the test statistic. For hypothesis C, t_{min} is the test statistic.

2. Program Method of Operation

The subroutine Kolmog compute the empirical cdf, by a do for loop $i=1$ to n , with $S(i)=i/n$. Shsort orders the sample values to form order statistics. The theoretical cdf's are found in one of the following subroutines: normf, expf, unif, erlf, and weibf.

The subroutine Findt finds the differences between S and F for each value of X . For $i=1$, F is compared to 0 and $S(1)$. Then, in a do for loop $i=2$ to n , $F(i)$ is compared to $S(i)$ and $S(i-1)$. These $2n$ difference are stored in the vector T . Another do for loop, $j=1$ to $2n$, finds t_{max} and t_{min} . From those two values, the greatest absolute difference is found.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

(1) Lochinvar: Kolmog, Upar, Wpar, Erpar, Npar,
Expar

(2) Crunch: Kolmog, Findt; Normf, Unif, Expf,
Erlf, Weibf

(3) External: Shsort

XXV. LILLIEFORS TEST

A. OVERVIEW

1. General Purpose

With a sample of less than 31 elements, this tests whether the sample came from a specified distribution, either a normal or exponential. The parameters used for the test are the maximum likelihood estimates (MLE), derived from the sample mean and variance. Those two values are computed and displayed in the output file; however, Lilliefors tests only the distribution type, not the parameter value. (Any value other than the MLE would have a higher probability of rejecting the null hypothesis).

If specific parameters must be tested for, use the Kolmogorov test.

2. Level of Measurement

Interval

3. Assumptions

The data values are a random sample of the population.

4. Hypothesis

With $F^*(x)$ as the hypothesised distribution function, with MLE parameters, and $S(x)$ as the actual population distribution

a. Null hypothesis

$H_0: F^*(x) = S(x)$ for all x

b. Alternative hypothesis

$H_1: F^*(x) \neq S(x)$ for at least one x

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If more than one variable is in the data set, the user must specify which one to utilize for the test. If that variable has more than 31 elements, the goodness of fit test will be offered automatically. Otherwise, choose whether to test for the normal or the exponential distribution.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 21. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 21. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index of variable to be used in test, if there is more than one in the data set.

(2) Enter whether to test against the best fitting normal or exponential distribution.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The sample mean and variance are listed, as well as the test statistic.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Finding the MLE Parameters

The sample mean and variance are computed. If the test is for normality, the theoretical distribution's parameters μ and σ^2 are estimated by the sample mean and variance respectively. If the exponential is the theoretical distribution, its parameter λ is the reciprocal of the sample mean.

b. Calculation of the Test Statistic

The remainder of the test is exactly the same as the two-tailed Kolmogorov test. The empirical distribution function is generated using $(1/n)$ increments for each of the n sample data elements. The difference between the empirical and theoretical distribution is computed at each value of x contained in the sample. The greatest absolute difference is the test statistic.

2. Program Method of Operation

The subroutine Lillief calls Muvar to find the sample mean and variance, then assigns MLE values to the rp vector. The subroutines Norf or Expf, as appropriate, are called to generate values of the theoretical distribution at the values of $X(1), \dots, X(n)$. The subroutine Findt finds the largest, smallest and greatest absolute difference; that last value is the test statistic.

3. Program Characteristics

a. Limitations

Maximum number of data points: 30, in one variable.

b. Subroutines/Procedures Used

- (1) Lochinvar: Lillief
- (2) Crunch: Lillief, Muvar, Expf, Norf, Findt
- (3) External: Shsort

XXVI. SHAPIRO WILK TEST FOR NORMALITY

A. OVERVIEW

1. General Purpose

With a sample of 30 or less data points, test to see if the sample is from a normal distribution of unspecified parameters. Although Crunch computes the sample mean and variance, they are not displayed. No statement about the population parameters is made by this test.

2. Level of Measurement

Interval

3. Assumptions

The sample is randomly drawn from the population.

4. Hypothesis

With $F^*(x)$ as a normal distribution of unspecified parameters, and $S(x)$ as the distribution function of the population,

a. Null hypothesis

$H_0: F^*(x) = S(x)$ for all x

b. Alternative hypothesis

$H_1: F^*(x) \neq S(x)$ for at least one x

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If there is more than one variable in the test, the user must specify which one is to be utilized. If that variable has more than 30 elements, the goodness of fit test will be automatically offered. If not, no further choices are necessary.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 22. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 22. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

Enter index of variable to be used in test, if there is more than one in the data set.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic will also be included.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic/Level of Significance

a. Compute the Sum of Squared Deviations

The sum of squared deviations is the sum over n of the square of the difference between $X(i)$ and the mean of X . The program actually figures the sample variance and multiplies by $(n-1)$. This is the denominator of the test statistic.

b. Compute the Sum of the Weighted Differences

The sample is sorted into the order statistics, $X(1), \dots, X(n)$. Subtract values with the same depth from the end points, e.g., subtract $X(n) - X(1)$, $X(n-1) - X(2), \dots, X(n-i+1) - X(i)$. The n is even, there are $n/2$ differences; if n is odd, there are $(n-1)/2$ differences, with the median excluded. Multiply these differences by $a(i)$ weights, provided by Lochinvar. Sum the weighted differences. This is the numerator of the test statistic.

c. Compute the Test Statistic

Divide the sum of weighted difference by the sum of squared deviations.

2. Program Method of Operation

The subroutine finds the sample variance by calling Muvar. Shsort is used to order the sample into order statistics $X(1), \dots, X(n)$. A do for loop, $i=1$ to $\text{trunc}(n/2)$, is used to calculate the sum of $a(i)*(X(n-i+1)-x(i))$; the $a(i)$'s are contained in the rp vector.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

- (1) Lochinvar: Wilkes
- (2) Crunch: Wilksh, Muvar
- (3) External: Shsort

XXVII. SMIRNOV TEST

A. OVERVIEW

1. General Purpose

With two samples, X and Y , each drawn from a different population, test whether the underlying population distributions $F(x)$ and $G(y)$ could be equal over the entire range of X and Y .

The Smirnov and Cramer-von Mises test differ only in the computation of the test statistic, and therefore, the choice of hypotheses.

2. Level of Measurement

Ordinal

3. Assumptions

The samples are drawn randomly from their populations, and are rankable against each other as well as within samples.

4. Hypotheses

a. Two tailed test

(1) Null hypothesis $h_0: F(x) = g(x)$ for all x

(2) Alternative hypothesis $h_1: F(x) \neq g(x)$ for at least one x in range of sample

b. One tailed test

(1) Null hypothesis $h_0: F(x) \leq g(x)$ for all x

(2) Alternative hypothesis $h_1: F(x) > g(x)$

c. One tailed test

(1) Null hypothesis $h_0: F(x) \geq g(x)$

(2) Alternative hypothesis $h_1: F(x) < g(x)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must designate which of two variables are used, if the data set contains more than two variables. The user specifies the type of hypothesis.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 23. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 23. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar: Enter indices of variables to be used in test.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic will also be listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculate the Difference Vector S

X and Y are sorted into order statistics, $X(1), \dots, X(m)$, and $Y(1), \dots, Y(n)$. The counter for X is i, the counter for Y is j. Compare $X(i)$ with $Y(j)$. If $X(i) < Y(j)$ then increment i; if $X(i) > Y(j)$ then increment j. The difference between the empirical distribution functions is $(i-1)/m$ minus $(j-1)/n$. After each comparison of $X(i)$ and $Y(j)$ the appropriate counter is incremented and the difference is calculated and stored in the vector S. The counter i is incremented until it reaches $(m+1)$; j can reach $(n+1)$. If $X(i) = Y(j)$, increment i, calculate and store the difference, then increment j, and compute and store the difference. There will be $(m+n)$ elements in S.

b. Calculate the Test Statistic

For a two tailed test (hypothesis A), the greatest absolute difference is the test statistic. For hypothesis B, the largest difference is the test statistic. For hypothesis C, the smallest difference is the test statistic.

2. Program Method of Operation

The subroutine Smirn calls Shsort to rank X, then rank Y. It calls the subroutine Finds to calculate the difference vector S, then chooses the test statistic based on the hypothesis specified.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

- (1) Lochinvar: Smirnov
- (2) Crunch: Smirn, Finds
- (3) External: Shsort

XXVIII. CRAMER-VON MISES TEST

A. OVERVIEW

1. General Purpose

With two samples, X and Y, test whether the underlying population distributions $F(x)$ and $G(y)$ could be equal over the entire range of X and Y.

The Smirnov and Cramer-von Mises test differ only in the computation of the test statistic, and therefore, the choice of hypotheses.

2. Level of Measurement

Interval

3. Assumptions

The samples are randomly drawn from the respective populations, and are independent of each other.

4. Hypothesis

a. Null hypothesis

$H_0: F(x) = G(x)$ for all x

b. Alternative hypothesis

$H_1: F(x) \neq G(x)$ for at least one x

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If there are more than two variables in the data set, specify which two are in the test.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name;

when offered a test, type 28. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 28. Enter "U", "V", ... "Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

Enter indices of variables to be used in test., if the data set has more than two.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. In addition, the test statistic is listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculate the Difference Vector S

This step is exactly the same (using the same subroutine) as in the Smirnov test. See section XXVII for details.

b. Calculating the Test Statistic

Find the sum of the squared elements of the S vector. With m as the number of elements of X, and n as the number of elements of Y, multiply the sum of squared S(i) times $(m*n)/(m+n)**2$. This is the test statistic.

2. Program Method of Operation

This subroutine Crmvon calls Shsort to order X and order Y into order statistics. The subroutine Finds finds the vector S. In a do for loop, $i=1$ to $(m+n)$, Crmvon

squares and sums $S(i)$. That sum is then multiplied by $(m*n)/(m+n)**2$.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

- (1) Lochinvar: Cramvon
- (2) Crunch: Cramvon, Finds
- (3) External: Shsort

LIST OF REFERENCES

1. Conover, C., Practical Nonparametric Statistics, John Wiley & Sons, New York, 1971
2. Stevens, S., "On the Theory of Scales of Measurement", Science, 103: 677-680, 1946
3. Zaks, Rodney, Introduction to Pascal, Sybex, Berkeley, Ca., 1981

APPENDIX A

TUTORIAL: LINKING TO AND USING THE COMPUTER PACKAGE

A. PURPOSE

This appendix contains three things: instructions on how to link to the class disk to perform nonparametric tests; instructions on how to copy the programs onto a user's A disk; and an example of how to use the program.

B. LINKING TO THE CLASS DISK

To perform the test, a student can link to user number 2266p, read only password OBRIEN, and use the programs there to perform nonparametric tests. The steps are:

1. Linking to the Class Disk

In CMS, type "CP LINK 2266P 191 192 RR". The computer will type back ENTER READ ONLY PASSWORD. Enter "OBRIEN". After receiving back the "R" message, enter ACC 192 B/A.

2. Using the Programs

An example of the use of Lochinvar and Crunch is given below. Detailed instructions for use are given in section III and the sections on specific tests.

3. Releasing the Class Disk

After you are done, type RELEASE 192.

C. COPYING THE PACKAGE

Three files are needed to run nonparametric tests: "LOCHI MODULE", "CRUNCH TEXT", and "CRUNCH EXEC". The first two are compilations of the source code, contained in "LOCHI PASCAL" and "CRUNCH FORTRAN". To copy any of these programs:

1. Link to the Class Disk
2. Copy the File: Type XEDIT <filename> <filetype>. Type SAVE = = A in the space marked ==> at the bottom of the screen, then FILE. The file will be on the A disk.
3. Release the Class Disk

D. AN EXAMPLE OF TEST USE

To fully understand this example, read or review sections III and XXI of this text. The data set used is from Conover [Ref. 1: p.319]; the test used is the Friedman test. The example shows the program's prompting in capital letters, and the user's response in lower case, with a <== to the right.

1. Entering the Data File

a. Preliminary Steps

There are four initial steps to perform before entering a data file. The first is to activate Lochinvar, then to request not to use a previously entered file, then to enter the file by data values, and finally to write the information to data file "lochi a".

```
lochi <== (1)
EXECUTION BEGINS...
DO YOU WANT TO PERFORM A TEST ON A PREVIOUSLY ENTERED DATA SET?
ENTER Y OR N
n <== (2)
```

```
YOU HAVE A CHOICE OF TWO METHODS TO ENTER DATA:
1. ENTER EACH VALUE OF DATA SET. VALUES MUST BE NUMERICAL, AND
MEASUREMENT AT LEAST ORDINAL. ALL TESTS MAY BE PERFORMED THIS WAY.
2. ENTER ONLY THE COUNT OF SUCCESSES/FAILURES OR ROW/COLUMN CELL
COUNT. DATA VALUES AND LEVEL OF MEASUREMENT MAY BE NOMINAL, BUT
COUNTS MUST BE NON-NEGATIVE INTEGERS. THE FOLLOWING TESTS MAY BE
PERFORMED: BINOMIAL, MCNEMAR, R*C CONTINGENCY, CHI-SQUARE
GOODNESS OF FIT.
ENTER V FOR VALUE OR C FOR COUNT
v <== (3)
```

```
ENTER FILETYPE (A-F) TO WHICH DATA GOES
a <== (4)
```

b. Enter Information of Label of Data Set

The label needs three types of information: the data set name, the number of variables, and the variable

names. For this example, we are using the data of exercise 1, page 309 in Conover, with four variables, named the names of the seasons:

```

ENTER NAME OF DATA SET
  prob 1,p 309, number of births by hospital <== (1)
ENTER NUMBER OF VARIABLES OR TREATMENTS
  4 <== (2)
ENTER NAME OF VARIABLE NR          1
  winter <== (3a)
ENTER NAME OF VARIABLE NR          2
  spring <== (3b)
ENTER NAME OF VARIABLE NR          3
  summer <== (3c)
ENTER NAME OF VARIABLE NR          4
  fall <== (3d)

```

c. Enter Data Elements

Next, enter the data when prompted. Enter either integers or real numbers, i.e., with or without decimal points; the program will read them as reals. Leave a space (not a comma) in between numbers. Enter all elements of a variable on one line if possible; if not, there will be a chance to add more elements after the "ENTER" key is pressed. There will also be a chance to check or correct entries. In the following example, a mistake is made, then corrected, in entry of variable one. In variable two the data is checked and found correct; in variable three it is not checked. In variable four, the first four data elements are listed, the "ENTER" key is pressed, the last three data elements are added, and the data checked.

(1) Enter, Check, and Correct the Values of Variable One:

```

WHEN ENTERING DATA ELEMENTS <1, START WITH 0.
ENTER VALUES FOR VARIABLE          1
  92 9 98 19 21 58 43 <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A (ADD), C (CHECK) OR G (GO ON)
  C <==
NUMBER OF ELEMS ENTERED              7
NUMBER OF ELEMS IN THIS VAR          7
92.000   9.000  98.000  19.000  21.000  58.000  43.000

```

(The last entry should have been '42' vice '43',
so re-enter variable:)

```
WANT TO RE-ENTER THIS VARIABLE?
ENTER Y OR N
y <==
ENTER VALUES FOR VARIABLE 1
92 9 98 19 21 58 42
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A (ADD), C (CHECK) OR G (GO ON)
c <===
NUMBER OF ELEMS ENTERED 7
NUMBER OF ELEMS IN THIS VAR 7

92.000 9.000 98.000 19.000 21.000 58.000 42.000
WANT TO RE-ENTER THIS VARIABLE?
ENTER Y OR N
n <==
```

(2) Enter, Check Values for Second Variable:

```
ENTER VALUES FOR VARIABLE 2
112 11 109 26 22 71 49 <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A (ADD), C (CHECK) OR G (GO ON)
c <===
NUMBER OF ELEMS ENTERED 14
NUMBER OF ELEMS IN THIS VAR 7

112.000 11.000 109.000 26.000 22.000 71.000 49.000
```

(This time, it is right the first time,
so no re-entry:)

```
WANT TO RE-ENTER THIS VARIABLE?
ENTER Y OR N
n <==
```

(3) Enter Values for Third Variable:

```
ENTER VALUES FOR VARIABLE 3
99 10 52 19 23 51 44 <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A (ADD), C (CHECK) OR G (GO ON)
g <===
```

Variable: (4) Enter, Add to, Check Values for Fourth

```
ENTER VALUES FOR VARIABLE 4
77 12 81 18 <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A (ADD), C (CHECK) OR G (GO ON)
a <===
ENTER VALUES FOR VARIABLE 4
24 62 41 <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A (ADD), C (CHECK) OR G (GO ON)
c <===
NUMBER OF ELEMS ENTERED 28
NUMBER OF ELEMS IN THIS VAR 7

77.000 12.000 81.000 18.000 24.000 62.000 41.000
WANT TO RE-ENTER THIS VARIABLE?
```

ENTER Y OR N
n <==

(5) Lochinvar offers a chance to re-enter the entire data set

YOU HAVE ENTERED 4 VARIABLES,
WITH THE FOLLOWING NUMBER OF ELEMENTS IN EACH
VAR NR NR OF ELEMENTS

1	7
2	7
3	7
4	7

DO YOU WISH TO RE-ENTER THE ENTIRE DATA?
ENTER Y OR N
n <==

(6) Lochinvar Determines the Variables are of equal size:

THE VARIABLES/TREATMENTS ARE OF EQUAL SIZE:
PAIRED COMPARISON TESTS ARE POSSIBLE

(7) Choice to do a Test with this DATA

NOW, DO A TEST WITH THIS DATA SET?
ENTER Y OR N
y <==

2. Choice of Test, Test Parameters, Output File

a. Choice of Output Files

The relevant choices are 'U LISTING', 'V LISTING', ..., 'Z LISTING'. Anything that is already in the file will be written over.

WHICH NAME FOR OUTPUT FILE, U-Z?
u <==

b. Choice of Test

Lochinvar will offer test according to the number of variables in the data set and whether the variables are of equal length. Test number 17 is desired; the Friedman rather than the Durbin is chosen.

PICK AMONG THE FOLLOWING NON-PARAMETRIC TESTS:

1. BINOMIAL
2. QUANTILE
3. COX-STUART TEST FOR TREND <2 VARS, PAIR>
4. MANN-WHITNEY
5. SIGN <PAIR>
6. MCNEMAR <PAIR>
7. R*C CONTINGENCY TABLE (CHI-SQUARE) <PAIR>
8. GOODNESS OF FIT, N>14 (CHI-SQUARE)
9. WILCOXON SIGNED RANK <PAIR>
10. RANK CORRELATION (KENDALL & SPEARMAN) <PAIR>
11. NON-PARAMETRIC LINEAR REGRESSION <PAIR>
12. NON-PARAMETRIC MONOTONE REGRESSION <PAIR>

13. MEDIAN
 14. KRUSKAL/WALLIS
 15. SQUARED RANK TEST FOR EQUAL VARIANCE
 16. HARTLEY TEST FOR EQUAL VARIANCE
 17. FREIDMAN OR DURBIN <BLOCKED>
 18. QUADE <BLOCKED>
 19. COCHRAN <BLOCKED>
 20. KOLMOGOROV (ONE SAMPLE: PARAMETERS SPECIFIED)
 21. LILLIEFORS (ONE SAMPLE: NORMAL OR EXPONENTIAL)
 22. WILKES-SHAPIRO TEST FOR NORMALITY
 23. SMIRNOV (2-SAMPLE)
 24. CRAMER-VON MISES (2-SAMPLE)
 DO YOU WANT TO SEE THE LIST AGAIN?
 ENTER Y OR N
 n <==
 ENTER THE NUMBER OF A TEST LISTED ABOVE
 17 <==
 THIS WILL ALLOW THE FRIEDMAN OR THE DURBIN TEST
 <DURBIN IS INCOMPETE BLOCK DESIGN: NEEDS FILLER>
 DO YOU WANT TO PERFORM THE FRIEDMAN TEST?
 ENTER Y OR N
 y <==

c. Choice of Test Parameters

For this example, only the first, second and fourth variables are used in the test, and the third block of the data set (i.e., the third element of each variable) is excluded from the test. The final test parameter entered is the level of significance.

(1) Variables Included in Test

WHEN ENTERING PARAMETERS <1, START WITH 0.
 DO YOU WANT ALL 4 INCLUDED IN TEST
 ENTER Y OR N
 n <==
 HOW MANY VARIABLES ARE INCLUDED IN TEST?
 3 <==
 ENTER INDICES OF VARIABLES TO BE IN TEST
 ENTER INDEX OF INCLUDED VARIABLE 1
 1 <==
 ENTER INDEX OF INCLUDED VARIABLE 2
 2 <==
 ENTER INDEX OF INCLUDED VARIABLE 3
 4 <==

(2) Exclude Third Block

DO YOU WISH TO EXCLUDE ANY BLOCKS FROM TEST
 ENTER Y OR N
 y <==
 HOW MANY BLOCKS DO YOU WISH TO EXCLUDE?
 UP TO 7 BLOCKS
 1 <==
 IN ASCENDING ORDER, ENTER INDICES, 1 - 7
 ENTER INDEX FOR EXCLUDED BLOCK NR 1
 3 <==

(3) Specify Level of Significance

ENTER LEVEL OF SIGNIFICANCE, ALPHA
 ENTER NUMBER >0, IF <1, START WITH 0.
 0.05

3. Run Test with Program Crunch

In CMS, type CRUNCH

4. Check Output File

Crunch will compute the test statistic, then display:

THE RESULTS ARE LISTED IN FILE
U LISTING

To check the results, type XEDIT U LISTING

APPENDIX B
THE PROGRAM LOCHINVAR

A. Purpose

This appendix is intended to assist in modifying the program Lochinvar, or in implementing it on a computer system other than the IBM-3033 at the Naval Postgraduate School. The last part of this appendix, general description of program operation, may be of interest to a user who is experienced in the use of Lochinvar.

Section III of the text describes the general use of the program, while Sections V through XXVIII give specific instructions on responding to Lochinvar's promptings for the various tests. Appendices C and D explain in detail Lochinvar's two output files, the data file and the options file, respectively.

The source code is contained in Appendix H.

B. DESCRIPTION OF PROGRAM CHARACTERISTICS

1. Language

Lochinvar was originally written and tested in Waterloo Pascal; it is presently in a form compatible with the IBM virtual system Pascal compiler, PASCALVS. The program's syntax is consistent with the Pascal described by Zaks [Ref. 3].

2. Input and Output Files

In Lochinvar, input or output files are referred to by one of nine text definitions. "Qin" refers to terminal input; "qout" refers to terminal output; "op" refers to the options file, "lochi op". Text "a" through "f" refer to data files "lochi a" through "lochi f"; these data files are used both as input and output.

Prior to using a file for output, Lochinvar must open that file by the "rewrite" command. For the data files, this occurs in the procedure New_data; for the options file, it occurs in the procedure File_io. All seven of those files are explicitly defined as fixed format, logical record length 80. The form of the rewrite command, here illustrated for "lochi a", is:

```
REWRITE(A,'NAME = LOCHI.A.A,RECFM=F,LRECL=80')
```

The terminal output is opened in the main program, using the command "TERMOUT(QOUT)".

Prior to reading a file as input, Lochinvar must open that file for reading with the "reset" command. For the data files, this occurs in the procedure Check_file. The terminal input file must be reset prior to each time data is entered from the keyboard. Rather than having reset commands scattered throughout the program, the procedure Rst is invoked. Rst resets the terminal for input with the command "TERMIN(QIN)".

3. Dimensions of Data Structures

At the beginning of Lochinvar are six constants. Tot_data defines the total number of data elements contained in the data set; it is currently set at 400. Tot_vars is the maximum number of variables in a data set plus one; it is currently set at 16 (for 15 variables). L_names is the length of variable or data set names; it is currently set at 50. L_r_par, L_i_par, and L_e_b are the number of real parameters, the number of integer parameters, and number of excluded blocks, respectively; each is currently set at 50.

Changing the setting of any one of those numbers will change all subsequent dimensioning within the program. It is not recommended that L_names be increased beyond 80, since this would place names on more than one line (card image) of the data or options file.

4. Type and Variable Declarations

Below the constant declarations, the type declarations define arrays, sets and types. Below that are variable declarations. All parameters are defined globally.

5. Description of Variables, Arrays, Sets

Below the variable declarations, a comment section gives brief descriptions of each variable, set or array passed as a parameter.

6. Description of Procedures and Functions

Each subprogram element contains a brief comment section describing its purpose.

C. DESCRIPTION OF PROGRAM OPERATION

The main program offers the user three choices: use a previously enter data set to perform a test, enter a new data set, or enter cell counts. See Section III of the text for a complete explanation of these choices. Depending on the user's choice, the main program will call one of three master subprograms to prompt the user through the data and parameter entries:

1. New Data

The master subprogram `New_Data` prompts entry of a new data set and offers the user the chance to perform a test with the newly entered data. Steps in this process:

a. Data Set File Type

`New_data` asks the filetype for the data, "a" through "f".

b. Names and Number of Variables

`New_data` calls `Namer`, which asks the data set name, the number of variables, and the variable names.

c. Data Element Values

If there are two variables, `New_data` offers the choice of entering data pair by pair. If the user chooses to

do this, New_data calls E_pairs. Otherwise, the procedure E_by_var prompts entry of data variable by variable.

d. Writing of Data File

New_data writes the data set file.

e. Test Choice

New_data asks if a test should be performed with this data. If yes, File_io and Which_test are called; these procedures are described in the next paragraph.

2. Old_data

If the user indicates that a previously entered data set is to be used, the procedure Old_data is called. It performs the following:

a. Data File Choice

Old_data asks the user's choice of file type of old data, "lochi a" through "lochi f". Old_data calls Check_file, which read and displays the name of the indicated data set. If it is the correct data set, it also reads the number of variables and number of elements per variable.

b. Output File Choice

The procedure File_io asks to which output file the test results should be sent. It writes the number of the data set file (NIF) and the number of the test results file (NOF) to line one of the options file.

c. Test Selection and Parameter Choice

Old_data calls the procedure Which_test. This procedure offers some or all of the nonparametric tests, depending on the number of variables in the data set and whether the number of elements per variable are equal. Once the choice is made, Which_test writes that choice to line 2 of the options file, along with whether the test involves cell counts. Which_test then calls the procedure for the chosen test.

d. Test Variable and Parameter Selection

Each test procedure is somewhat different, however, there are common points. If the test involves cells, the test procedure prompts cell related parameters, then calls `W_cells` to write those parameters to line 3 of the options file (see Appendix D for details of that file). If the data set contains more variables than the minimum needed for the test, `Which_vec` will prompt the user to enter which variables are included in the test. If the test involves paired or blocked data, the procedure `Exclud_b` offers the user the chance to exclude pairs or blocks from the test; in other cases, `None_ex` writes a "0" to line 5 of the options file. The test procedure then prompts entry of various integer and real parameters, and calls the procedure `W_param` to write those parameters to lines 6 and 7 of the options file.

3. Cel_test

If the user specified to enter cell counts, the procedure `Cel_test` is called by the main program. It calls `File_io` to write the NIF and NOF to the data file. It calls `Namer` to get the data set and variable names. The choice test dictates which procedure prompts the cell count entry. `Cel_test` writes the data set name and variable names to line 8 of the options file.

APPENDIX C
THE DATA FILE

A. PURPOSE

This appendix explains the structure of the data files, which contain the data set and variable names, the number of variables and number of elements per variable, and the data elements of the variables. This will aid the user in understanding the computer package. More importantly, it will allow the user to test data generated by another program, or read from cards or tape, without having to load the data into Lochinvar by hand.

B. METHOD OF WRITING DATA FILES

1. Formats of Data File Components

All names are written in character fields of length 48, from the first to the forty-eighth position on a line. All integers are written in a 16I5 format, i.e., with the least significant digit of the integer on position 5,10,...,80. All data elements are read as real numbers; they are written to be read in an 8F10.3 format. This means that there are eight fields for real numbers per line, 1-10,11-20,...,71-80, and the real numbers have three digits to the right of the decimal point, e.g., 789.123. Lochinvar includes a decimal point in all real numbers; if this is not possible, the least significant digit of an entry (the 0.001 digit) must be right justified in the field, i.e., aligned on column 10,20,...,80.

If the user has a data set whose data elements are all integers, there is a way to have the data file written without converting all the integers to real numbers. The least significant digit of the integers can be aligned on

columns 7,17,...,77, and the program Crunch will read them as real numbers with the same value as original integer data.

If the data elements of the data set are either all very large or all very small, they may not fit into the F10.3 format. A quick fix to that problem is to rescale the values, e.g., multiply very small numbers by 1000, or divide large numbers by that amount. The validity of the statistics will not be effected by that transformation.

If there are both very large and very small numbers in the data set, there will be numerical errors in single precision Fortran for any tests which use an interval level of measurement. For tests which use ordinal or nominal level of measurement, any monotonic transformation, such as $y = \ln x$, could be used to allow both very large and very small data values to fit into the F10.3 format.

2. Layout of the Data File

a. Data Set Name

The data set name occupies the first line of the data file. The name is from 1 to 48 characters long. On the same line as the name, in the forty-ninth position, there is the character "|", which on IBM keyboards is the upper case "1". For example, a data set file name:

This is a data set name |

b. Number of Variables and Elements per Variable

On the line following the data set name, there is a line of integers. The first integer is the number of variables in the data set. The second integer is the number of elements in the first variable; the third integer is the number of elements in the second variable, and so on. There must be non-zero integers in the first two positions. Lochinvar writes zero where there are no values, but that is not necessary. As mentioned above, integers are right justified on position 5, 10, ..., 80.

Let the data set consist of two variables. The first variable has two data elements, the second has four. the first two lines would appear as:

```

This is a data set name      |
 2      2      4      0      0      0      0      0...

```

c. Data elements

The data elements are listed variable by variable, i.e., all the elements of the first variable are listed, then all of the second, et cetera. As mentioned above, the data elements are read as real numbers in the F10.3 format.

Let the elements of the first variable be 27.5 and 31; let the elements of the second variable be 13, 17, 25 and 35. The first three lines of the data file would appear as:

```

This is a data set name      |
 2      2      4      0      0      0      0      0      0...
27.500  31.000  13.000  17.000  25.000  35.000

```

The zeros to the right of the decimal point are written by Lochinvar but are not necessary. The number could have been written without decimal points, with proper alignment, as shown below. However, when numbers are written with decimal points, a mistake in alignment is no problem; without decimal points, alignment is crucial. Using only alignment for data elements:

```

This is a data set name      |
 2      2      4      0      0      0      0      0      0...
 275    31    13    17    25    35

```

d. Variable names

The variable names begin on the line directly following the last data element line. Each variable name occupies a line; there must be the same number of lines for variable names as there are number of variables indicated in line two.

Let freshman be the name of the first variable and senior be the name of the second.

```
This is a data set name
  2       2       4       0       0       0       1       0
27.500  31.000  13.000  17.000  25.000  35.000
freshman
senior
```

C. LOCATION ON THE USER'S A DISK

1. Default Location

As mentioned repeatedly throughout the text, by default the data files are named "lochi a",...,"lochi f". Lochinvar writes files to those six locations; Lochinvar will not be easily modified unless the user is experienced in programming Pascal.

2. Reading Files from other than Default Locations

The program Crunch reads "lochi a",...,"lochi f" based on the file definitions in the file "CRUNCH EXEC" and the number of input file (NIP) of the options file. "CRUNCH EXEC" is explained in Appendix E, while the input file is explained in Appendix D. To have Crunch read from a properly written data file named anything other than "lochi a",...,"lochi f", the user will have to do the following:

a. Write an options file for the data (see section III and Appendix D), either manually (not recommended), or else using Lochinvar, specifying a data set with the same general characteristics as the test data set. For example, if the test data set is named QUEUE DATA, with four variables of equal length, tell Lochinvar to run the desired test on data set "lochi a", given that "lochi a" has four variables of equal length

b. Since the number of input file written in the options file is "1", corresponding to the default location "lochi a", change the file definition in the file "CRUNCH EXEC". The origin file "CRUNCH EXEC",

&TRACE
&ERROR &EXIT 99
FILEDEF 01 DISK LOCHI A

LÖAD CRUNCH (START

becomes,

&TRACE
&ERROR &EXIT 99
FILEDEF 01 DISK QUEUE DATA

LÖAD CRUNCH (START

c. Alternate Method

It would be easier, in the example above, to rename the current file "lochi a" to some other name, and rename "QUEUE DATA" to "lochi a".

APPENDIX D
THE OPTIONS FILE

A. PURPOSE

This appendix describes the options file. The twenty nonparametric tests which require ordinal or interval data level of measurement all have options files with similar format. The four tests which can handle nominal data have slightly idiosyncratic formats, which contain all the information of other test plus something additional.

This appendix is included for completeness of documentation of this text. It is not recommended that a user by-pass Lochinvar to write the options file by hand. If the computer package is implemented on an operating system which cannot run Pascal, though, that may be necessary.

B. GENERAL INFORMATION ABOUT THE OPTIONS FILE

1. Name

Lochinvar writes the options file to "lochi op" on the user's A disk.

2. General Format Rules

As in the data file, all integers are written in the format 16I5, all real numbers are written as 8F10.3, all names are in 48 character lines. See Appendix C, paragraph B, for all the different ways to have real numbers represented for the program Crunch to read.

3. Explanation of the Options File Format

a. List of Line Numbers

There are 14 line numbers: 1, 2, 3, 3a, 4, 4a, 5, 5a, 6, 6a, 7, 7a, 8, 8a.

b. Line Numbers in the Option File

Each of the 14 line numbers occupies at least one line (or card image) in the options file, unless that line is omitted. There is not a blank card for omitted line numbers; rather, the previous line is followed by the next. For example, if line numbers 3 and 3a are omitted, line 4 immediately follows line 2.

The lines with "a" in their number may occupy more than one line (or card image) in an actual file, because they may contain more information than would fit on a single 80 column card.

c. Rules for Inclusion of Lines in Options File

Lines 1 and 2 are included in every options file.

Lines 3 and 3a are omitted unless the test is one of the four which handle nominal data.

Lines 4 and 4a will be included in every test except when the user has entered cell counts (in one of the four tests which use nominal data). Line 4 is a single positive integer value which tells how many values to read in line 4a.

Line 5 equals "0" and line 5a is omitted unless the user has excluded blocks of the data set from being tested. In that case, line 5 indicates the number of elements to be read in line 5a.

Lines 6 and 7 are included in every test, and will contain single integer values. If a value of "0" is in one of those lines, then the corresponding "a" line will be omitted. For example, if line 6 has a "0", then line 6a will be omitted. If the single value in lines 4, 5, 6, and 7 is not "0", it must be a positive integer, representing the number of elements to be read in the corresponding "a" line. For example, if the single value in line 5 is "3", there will be three elements to read in line 5a.

The "a" lines may occupy as many actual lines in the data file to contain all the values to be read.

Lines 8 and 8a are used only when the user has entered cell counts for a test involving nominal data.

d. Content of the Lines

(1) Line 1: the number of input file (NIF), and number of output file (NOF), both integers

(2) Line 2: the number of the requested test (NTEST), and whether the test involves cell counts (CELLS), both integers, CELLS=1 if the test is one of the four which can use nominal data.

(3) Line 3: the cell counter option (TYPCEL), the number of rows in the cell matrix (NROW), the number of columns in that matrix (NCOL), and the number of elements in line 3a (NTORD), all positive integers. Depending on the TYPCEL option, line 3a will contain either integers or reals.

(4) Line 4: the number of treatments to be included in this test (NTREAT), a positive integer. Line 4a contains the indices of the variables of the data file which are included in the data set to be tested.

(5) Line 5: the number of blocks excluded from the test (NEXB). Line 5a contains the indices of excluded blocks.

(6) Line 6: the number of integer parameters, a non-negative integer. If line 6 is greater than zero, line 6a contains the integer parameters for the test.

(7) Line 7: the number of real parameters, a non-negative integer. If line 7 is greater than zero, line 7a contains the real parameters for the test.

(8) Line 8: data set name. Line 8a contains the names of data set variables.

C. EXAMPLE OF OPTIONS FILE FOR ORDINAL OR INTERVAL TEST

Table 5 contains a representative data file, with the line numbers of the options file for the twenty tests which require ordinal or interval data. This example is for the test to perform the problem contained in Appendix A, i.e., problem 1, page 309 of Conover [ref 1]. In this example, three of four variables were used, and one block excluded. Line numbers are enclosed by < >. Comments are enclosed by (*-*); the comments refer to the names of variables, as mentioned in the previous section.

TABLE 5
Options

<1>	1	11	(*NIF=1,NOF=11*)
<2>	17	0	(*NTEST=17,CELLS=0*)
<4>	3		(*NTREAT=3*)
<4a>	1	2	4 (*indices of treatments*)
<5>	1		(*NEXB*)
<5a>	3		(*index of the block excluded*)
<6>	0		(*number of integer parameters=0*)
<7>	1		(*number of real parameters=1*)
<7a>	0.95		(*real parameter=0.95*)

APPENDIX E
THE PROGRAM CRUNCH

A. PURPOSE

This appendix is intended to assist implementing Crunch on systems other than the IBM-3033 at the Naval Postgraduate School. It will also assist in modifying the program, or adapting portions of it for use in other programs.

B. DESCRIPTION OF THE PROGRAM STRUCTURE

1. General

While Lochinvar is designed to be used only as an integral unit, Crunch is separable. Therefore, many things which are centralized in the main program of Lochinvar--such as dimensioning and variable declaration--are done in each subroutine of Crunch. Explanations and documentation are also at the subprogram level.

2. Subprogram Groups

Crunch can be thought of as a system of five groups, each made up of several subroutines and functions.

a. Main Program

The main program, together with the subroutines Fname and Vname, form one group; they read the data set and variable names, call the correct data formation group, then call the correct test subroutine, and remind the user which file contains the output.

b. Data Formation Group for Cell Counts

If the test involves cell count (Binomial, Quantile, McNemar, Test for Independence, Cochran), the data formation group Celler is called by the main program. Celler and the subroutines it calls produce the cell counts used as

the raw material for calculations in the test subroutine. Those cell counts are contained in the vector RCCT, which is dimensioned by the number of rows NROW and the number of columns NCOL.

c. Data Formation Group for Data Values

For all other tests, the master reader group is called. Mstrdr and its subroutines read the specified data file and the options file. They produce a list of the data elements which are relevant for the test, i.e., Mstrdr includes only the data set variables indicated, and excludes any blocks or pairs specified. These data values are contained in the vector LIST. LIST is dimensioned by the pointer vector LPTR. LPTR(1) is always 1. LPTR(i) is the index of the first element of the ith variable in LIST. With k variables, LPTR(k+1) is one more than the last element of LIST. For example, if LIST contains three variables, with two, four and seven elements each, LPTR will have four elements, (1,3,7,14). In dereferencing variable values, the ith variable will have elements LIST(LPTR(i)) through LIST(LPTR(i+1)-1). NTREAT is the number of variables included in the test.

d. Utility Subroutines and Functions

Many subroutines are called by more than one test subroutine. These versatile subroutines are grouped together.

e. Test Subroutine Groups

Each of the 24 tests has a principle subroutine called by the main program. In general, these subroutines perform the bulk of the computations for the test statistic or level of significance. However, some of the computation is done by calling other subroutines. Subroutines which are called only by one test are listed contiguous to that test's subroutine. Subroutines which are called by more than one test are listed in the utility subroutine group.

The test subroutines make use of either LIST or RCCT in their computations.

3. Language

Crunch is written to be compatible with the IBM Fortran H compiler. Only the most rudimentary syntax was used. The only loop statements are the do for loops of the form,

```
DO 10 I=LL,UL
```

where 10 is a line label for a continue statement, LL is the lower limit and UL is the upper limit of the loop. All other loops or branching is done with if statements. This very primitive, unstructured Fortran should be compatible with virtually any computer system.

4. Input/Output Files

The IBM file definitions are contained in the file "CRUNCH EXEC". All files except 08 (terminal) are implicitly fixed format, logical record length 80. Files 01 through 05 refer to "lochi a" through "lochi e"; file 06 is "problem data"; file 07 is "lochi f"; file 10 is "lochi cp"; files 11 through 16 are "u listing" through "z listing".

The number of input file (NIF) and number of output file (NOF) are listed in line one of the options file, and read by Crunch's main program. Files 01, ..., 05, 07, and 10 are candidates for NIF; files 11, ..., 16 are candidates for NOF. File 08 is used only to remind the user of which output file contains the test result. File 06 contains system error messages, while file 09 contains intermediate computations of Crunch.

The main program reads lines one and two of the options file. Either Celler or Mstrdr read most of the rest of the data and options file. Fname reads the data set name and writes it to the output file; Vname does the same for the variable names.

The test subroutines write all of the test results files "u listing" through "z listing", except the data set and variable names.

5. Dimensions and Declarations

At the beginning of the main program, and of each subroutine, the variables used are declared and the arrays dimensioned. All declarations and dimensioning is explicit. Although arrays could be dimensioned by variables in subroutines, this is not done. For example, with TOTELM as the integer representing the total data elements of the test, LIST could be dimensioned in a test subroutine as LIST(TOTELM) vice LIST(400). This was not done; as explained above, Crunch is intended to be separable. To change the capacity of the program, a global change command may be issued, if the installation has a good text editor. Otherwise, the dimensions must be changed one by one.

There are a very few arrays which consume inordinate amounts of storage for what information they provide. They have the dimension of 20,000; that dimension allows storage of a comparison of each of 400 data elements with each other data element. On the IBM-3033, these large arrays require the user to request 1 megabyte of storage. On smaller systems, these arrays may be redimensioned down to exactly the size necessary (if k =the number of elements, $k(k-1)/2$). An alternative is to eliminate them altogether, and take out test procedures which use them.

6. Description of Variables

At the beginning of the program, there is a brief description of all variables and arrays which are used or passed by the main program. After the subprogram title line, a similar list defines the variables used exclusively among subroutines of that subprogram.

Above the variable declarations in each test subroutine, there is a description of each of the integer and real parameters used in that test.

C. EXTERNAL SUPPORT

The non-INSL subroutine Shsort is essential to the operation of the program Crunch. Its documentation is available to any user at the Naval Postgraduate School. Any system sort routine can be used to support Crunch, which is currently configured to pass a real vector of values to be sorted, an integer vector $(1, \dots, n)$ to receive the order of the original data values, and the number of elements n .

The non-INSL Utest was used for the Mann-Whitney test. Its value is that it not only computes the test statistic, but also the level of significance. Modifications to Crunch could easily allow computation of the test statistic.

APPENDIX F
FORTRAN SOURCE CODE: CRUNCH

```

C THIS PROGRAM TAKES DATA FROM TEST A-F FILE, INFORMATION
C FROM TEST OF FILE TO PERFORM NON-PARAMETRIC TESTS
C
C SUBPROGRAM GROUPS
C
C MAIN CONTAINS THE MAIN PROGRAM, FNAME, VNAME, REMIND
C READS TOP 2 LINES OF OPTIONS FILE, WRITES THE
C TCF OF OUTPUT FILE, CALLS CELLER OR MSTRDR,
C TEST SUBROUTINE; REMINDS WHICH IS OUTPUT FILE
C
C CELLER CONTAINS CELLER, COCH, TC3, TC3LN, DICOT, TC4, TC4LN,
C PAIRST, MCOUNT. FINDS THE CELL COUNTS FOR FIVE
C NP TESTS: BINOMIAL, QUANTILE, MCNEMAR, COCHARAN,
C AND RC CONTINGENCY. CELL COUNTS TO RCCT;
C DIMENSIONED BY NROW AND NCOL
C
C MSTRDR CONTAINS MSTRDR, SUBLIST, NULIST,
C READS THE DATA FILE NUMBER OF VARIABLES, #
C ELEMENTS/VARIABLE, DATA ELEMENTS. INCLUDES
C VAR'S CF TEST INTO LIST, EXCLUDED PAIRS OR
C BLOCKS AS REQ. YIELDS LIST(DATA ELMS),
C LPTR(PCINTER FOR LIST), NTREAT(# OF VAR IN
C TEST), TOTELM (# ELMS IN LIST)
C
C UTILITY CONTAINS GENERAL SUPPORT SUBROUTINES/FUNCTIONS
C SUBROUTINES: EXPF, FINDS, FINDT, MUVAR, NORMF, NORMV,
C PARCOM, RANQUE, RHOER, RNO, RPARAM
C
C TESTS CONTAINS THE 24 NONPARAMETRIC TESTS: USE EITHER
C EITHER LIST OR RCCT; COMPUTE TEST STATISTICS
C OR LEVEL OF SIGNIFICANCE; WRITES TC OUTPUT
C FILE
C THE ORDER IN THE PROGRAM IS THE SAME AS THE
C NUMBER OF THE TEST, NTEST
C
C VARS NUMBER 1-6 INDICATING FILE TEST A-F FOR DATA
C NIF INT VECTOR(16) GENERAL PURPOSE HOLDER
C IHOLD INT NUMBER OF VARS OR TREATMENT IN DATA SET
C NVAR S INT NUMBER OF VAR/TREATS TEST ACTUALLY USES
C NTREAT INT VECTOR(16) INDICES OF TREATS OR VARS
C ACTUALLY USED
C
C NELMS INT VECTOR(16) # ELEMENTS OF DATA IN EACH
C VAR/TREAT; IF SOME BLOCKS EXCLUDED,
C NUMBER REDUCED
C
C TOTELM INT TCTAL # OF DATA ELEMENTS IN DATA SET;
C IF SOME BLOCKS EXCLUDED, NUMBER REDUCED
C
C NTEST INT INDEX OF NON-PARAMETRIC TEST TO BE USED
C
C EXB INT VECTOR(200) INDICES OF BLOCKS TO BE
C EXCLUDED; MAY BE EMPTY
C
C NEXB INT NUMBER OF ELEMENTS OF EXB
C
C IP INT VECTOR(50) INTEGER PARAMETERS
C
C IP1 INTEGER VAR FOR CELLER; MAYBE NROW OR SOME
C OTHER VALUE DEPENDING ON TYPE OF CELL
C DEFINITION (SEE TYPCEL)
C
C NIP INT NUMBER OF ELEMENTS IN IP
C
C RP REAL VECTOR(10) REAL PARAMETERS
C
C NRP INT NUMBER OF ELEMENTS IN RP
C
C NIF INPUT FILE INDEX
C
C NOF OUTPUT FILE INDEX
C
C LIST REAL VECTOR(400) DATA ELEMENTS
C
C LPTR INT VECTOR(17) POINTS TO THE FIRST ELEMENT
C OF EACH VAR OR TREATMENT'S DATA IN LIST;
C THE LAST ELEMENT OF LPTR=TCTELM+1
C
C RNVAR S REDUCED NUMBER OF VARIABLES
C
C RNELMS VECTOR OF REDUCED # OF ELEMENTS PER VARIABLE
C
C DICOP =-1 IF DICOTOMIZE BY PARTITION, ELSE=NUMBER

```

CCCCCCCCCCCCCCCC

CF VALUES OF SMALLER SUBSET CF DATA,
 C IF DATA NOT DICOTOMIZED
 DIVAL REAL PARTITION VALUE OR LIST CF SUBSET VALUES
 NSUCC IF DICOTOMIZED DATA, HOW MANY OF ONE SUBSET
 CELLS INT = 1 IF TEST USES CELL COUNTS, ELSE = 0
 TYPEL INTEGER METHOD OF DEFINING CELLS:
 1-DICOT PARTITION,
 2-DICOT LIST, 3-R*C PARTITION,
 4-FIXED ROW/COL
 COUNT, 5-ENUMERATED CELL COUNT
 RCCT RCW, CCL CELL COUNT, BY RCWS (RE FIRST VAR),
 E.G., ALL COLS OF ROW 1, THEN ALL COLS OF ROW 2
 ROWNR VECTOR OF ROWS TO WHICH 1ST VAR ELEMS BELONG
 COLNR VECTOR OF COLS TO WHICH 2D VAR ELEMS BELONG
 RORC INTEGER LOGICAL 1=ROW 0=COL
 NTORD INTEGER # OF VALUES TO READ (W/I CELLER)
 CIV INTEGER VECTOR CELL INPUT VALUES
 CRP REAL VECTOR CELL INPUT VALUES
 TIEV INTEGER VECTOR OF TIES: TIEV(1).LE.TIEV(2).GT;
 BOTH WITHIN EPSILN OF PARTITION VALUE

101
102
201
202

```

INTEGER NIF, NFILE, IHOLD(16), NVAR S, TOTELM, NTEST,
* NTREAT, IP(50), NRP, LPTR(17), NELMS(16), RNELMS(16),
* DICOP, NSUCC, CELLS, NOF, CIV(50), NTCRD, RCRC, ROWNR(50),
* RCCT(50), TYPEL, TIEV(2), EXB(200), TREAT(16), RNVAR S,
* REDELM, CCLNR(50)
REAL RP(50), LIST(400), DIVAL(200), CRP(50)
FORMAT(16I5)
FORMAT(8F10.3)
FORMAT(1X,16I5)
FORMAT(1X,8F10.3)
READ(10,101) NIF, NOF
READ(10,101) NTEST, CELLS
WRITE(9,201) NIF, NOF, NTEST
TYPEL=0
IF (CELLS.EQ.1) CALL CELLER(NIF, NOF, NTEST, NROW, NCOL,
* RCCT, LIST, LPTR, NVAR S, NTREAT, NELMS, RP, IP,
* TREAT, TIEV, TYPEL)
IF (CELLS.EQ.C) CALL MSTRDR(NVAR S, NELMS, LIST,
* LPTR, TOTELM, NTREAT, NIP, IP, NRP, RP, NIF, NOF, TREAT)
IF((TYPEL.NE.62).AND.(TYPEL.NE.72))
* CALL VNAME(NIF, NOF, NVAR S, NTREAT, TREAT)
IF(NTEST.EQ.1) CALL BINTST
(NOF, RCCT, RP, IP, ALPHAT, CILL, CIUL)
IF(NTEST.EQ.2) CALL QUANTL(NCF, RCCT, RP, IP, ALPHAT,
* CILL, CIUL, LIST, LPTR, TIEV, TYPEL)
IF(NTEST.EQ.3) CALL COXSTU(LIST, LPTR, IP, RP, NOF, NTEST)
IF(NTEST.EQ.4) CALL MANWHT(LPTR, LIST, NCF, RP)
IF(NTEST.EQ.5) CALL SGNT(LIST, LPTR, NOF, TOTELM, RP, IP,
* NTEST)
IF(NTEST.EQ.6) CALL MCNEMR(RCCT, RP, IP, NOF)
IF(NTEST.EQ.7) CALL RCONT(RCCT, NROW, NCOL, NOF)
IF(NTEST.EQ.8) CALL GDFIT(LIST, LPTR, IP, RP, NOF)
IF(NTEST.EQ.9) CALL WILCOX(LIST, LPTR, IP, RP, NOF)
IF(NTEST.EQ.10) CALL CORREL
(LIST, LPTR, NTREAT, TREAT, NCF, IP, RP)
* IF(NTEST.EQ.11) CALL REGRET(LIST, LPTR, IP, RP, TREAT, NOF)
IF(NTEST.EQ.12) CALL MCNOFE(LIST, LPTR, IP, RP, TREAT, NOF)
IF(NTEST.EQ.13) CALL MEDIA(NTREAT, LIST, LPTR, NIF, IP, NOF)
IF(NTEST.EQ.14) CALL KRUAL
(LIST, LPTR, IP, RP, NOF, NTREAT, TREAT)
* IF(NTEST.EQ.15) CALL SQUARK
(LIST, LPTR, NTREAT, TREAT, IP, RP, NOF)
IF(NTEST.EQ.16) CALL HARTLE
(LIST, LPTR, NTREAT, TREAT, IP, RP, NOF)
* IF(NTEST.EQ.17) CALL FRIEDM
(NTREAT, LPTR, LIST, IP, RP, NOF, TOTELM)

```

```

      IF(NTTEST.EQ.1E)CALL QUADE
      * (NTREAT,LPTR,LIST,NCF,TCTELM)
      IF(NTTEST.EQ.19)CALL CCCHRNI(NROW,NCOL,RCCT,IP,RP,NCF)
      IF(NTTEST.EQ.20)CALL KOLMGG(LIST,LPTR,TREAT,IP,RP,NCF)
      IF(NTTEST.EQ.21)CALL LILIEF(LIST,LPTR,TREAT,IP,RP,NCF)
      IF(NTTEST.EQ.22)CALL WILKSH(LIST,LPTR,TREAT,IP,RP,NCF)
      IF(NTTEST.EQ.23)CALL SMIRNZ(LIST,LPTR,TREAT,IP,RP,NCF)
      IF(NTTEST.EQ.24)CALL CRVONM(LIST,LPTR,TREAT,IP,RP,NCF)
      CALL REMIND(NCF)
      STOP
      END

```

```

C-----
      SUBROUTINE REMIND(NCF)
      INTEGER NCF
      WRITE(8,1C1)
101  FORMAT(1X,' THE RESULTS ARE LISTED IN FILE')
      IF (NOF.EC.11) WRITE(8,11)
11   FORMAT(1X,' U LISTING')
      IF (NOF.EC.12) WRITE(8,12)
12   FORMAT(1X,' V LISTING')
      IF (NOF.EC.13) WRITE(8,13)
13   FORMAT(1X,' W LISTING')
      IF (NOF.EC.14) WRITE(8,14)
14   FORMAT(1X,' X LISTING')
      IF (NOF.EC.15) WRITE(8,15)
15   FORMAT(1X,' Y LISTING')
      IF (NOF.EC.16) WRITE(8,16)
16   FORMAT(1X,' Z LISTING')
      RETURN
      END

```

```

C-----
      SUBROUTINE VNAME(NIF,NCF,NVARS,NTREAT,TREAT)
C THIS READS EACH VARIABLE NAME, AND IF THE VARIABLE
C IS IN THE TEST, WRITE THE NAME INTO THE OUTPUT FILE
      INTEGER NIF,NCF,NTREAT,TREAT(16),NVARS,I,J
      J=1
      IF (NVARS.EQ.NTREAT) WRITE(NCF,402) NVARS
402  FORMAT(1X,'ALL',I5,
      * ' VARIABLES OF DATA SET USED IN THIS TEST')
      IF (NVARS.NE.NTREAT) WRITE(NCF,403) NVARS,NTREAT
403  FORMAT(1X,'OF',I5,' VARIABLES IN DATA SET,',/5X,I5,
      * ' USED IN THIS TEST')
      DO 10 I=1,NVARS
      READ (NIF,20C)
      IF (I.NE.TREAT(J)) GO TO 10
      WRITE(NCF,4C1) TREAT(J)
401  FORMAT(1X,' THE NAME OF VARIABLE/TREATMENT #',I4)
      J=J+1
      WRITE(NCF,2C0)
10   CONTINUE
200  FORMAT(1X,
      *50HAAAAABBBBCC(CDDCDEEEFFFFGGGG0000IIIIJJJKKKLLLLMM)
99   CONTINUE
      RETURN
      END

```

```

C-----
      SUBROUTINE FNAME (RORW,IF,OF)
C THIS READS THE FILE NAME FROM FILE "IF", IF RORW IS 1,
C AND WRITES THE FILENAME TO OF IF RORW IS 2
      INTEGER RORW,IF,OF
      IF (RORW.GT.1) GO TO 10
      READ(IF,20C)
      GC TO 99
10   WRITE (OF,122)
122  FORMAT (10X,' NAME OF DATA SET:')
      WRITE (OF,200)
200  FFORMAT (1X,

```

```

99 * 48HAAAABERBC CCC DDDD EEEEE FFF FGGGG GGGG OOO IIII JJJJ KKKK LLLL )
CONTINUE
RETURN
END
C*****
SUBROUTINE CELLER(NIF,NOF,NTTEST,NROW,NCOL,RCCT,
*LIST,LPTR,NVAFS,NTREAT,NELMS,RP,IP,TREAT,TIEV,TYPCEL)
C THIS SUBPROGRAM GROUP FINDS THE CELL COUNTS FOR ONE OF
C FIVE NONPARAMETRIC TEST--BINOMIAL, QUANTILE, COCHARAN,
C MCNEMAR, RC CCNTINGECY (CHISQAURE TEST FOR INDEP)-- AND
C PLACES THOSE CELL COUNTS INTO THE VECTOR RCCT, WHICH IS
C DIMENSIONED BY NROW AND NCOL. UNLESS THE USER HAS
C ENTERED THE CELL COUNTS, THE SUBROUTINE CALLS SUBROUTINE
C MSTRCTR TO READ THE DATA SET INDICATED BY NIF. THE DATA
C VALUES ARE IN LIST, WHICH IS DIMENSIONED BY LPTR.
C SUBROUTINES:
C DICOT PERFORMS THE DICOTOMIZATION FOR THE BINOMIAL AND
C QUANTILE, GETTING # SUCCESS=RCCT(1) AND # FAIL
C =RCCT(2) TYPCEL=2 IF BY LIST, =1 IF BY PARTVAL
C COCH GETS THE BLOCK AND TREATMENT SUMS OF SUCCESSES
C BY PARTVAL; BLOCK SUCCESSES INTO RCCT(1),...,
C RCCT(8); TREAT SUCCESS INTO RCCT(8+1),....
C MCOUNT GETS THE FOUR CELL COUNTS FOR THE MCNEMAR TEST.
C RCCT(1) IS SUCCESS/SUCCESS, RCCT(2) SUCCESS/
C FAILURE, ETC. TYPCEL=
C TC3 DIVIDES PAIRS INTO CELLS OF AN RC CONTINGENCY
C TABLE, BASED ON PARTITION VALUES. RCCT(1)
C IS FIRST ROW FIRST COL, RCCT(2) IS FIRST ROW
C SECCND COL, ETC. TYPCEL=3. CALLS PAIRST,
C TC3LN, AND RCCTR
C PAIRST SORTS THE VALUES OF THE TWO VARIABLES, AND KEEPS
C AN INDEX KEY ON EACH, IN ORDER TO RE-PAIR
C THE PAIRS; KEY1=ROW, KEY2=COL
C TC3LN CALLED BY TC3. ASSIGNS AN INTEGER CORRESPONDING
C TO THE ROW IN WHICH AN ELEMENT OF THE FIRST
C VARIABLE FALLS, OR THE COL IN WHICH AN ELEMENT
C OF SECOND VARIABLE FALLS, BASED ON PARTITION
C VALLES CONTAINED IN VECTOR CRV. INTEGER VECTOR
C RORCNR CONTAINS THE INDICES OF ROW OR COL
C ASSIGNED FOR EACH VALUE
C RCCTR TAKES THE RORCNR VECTORS FOUND BY TC3LN OR TC4LN
C AND FORMS CELL COUNTS
C TC4 DIVIDE THE DATA PAIRS INTO THE CELLS OF AN RC
C CONTINGENCY TABLE BASED ON FIXED ROW AND COL
C TOTALS. TYPCEL=4. CALLS PAIRST, TC4LN, AND
C RCCTR
C TC4LN ASSIGNS A VALUE TO A ROW OR COLUMN, BASED ON
C FIXED ROW OR COL TOTALS; THE VECTOR OF ASSIGN
C IS RORCNR
C INTEGER NIF,NFILE,IHCLD(16),NVAR,S,TOTELM,NTTEST,
*NTREAT,IP(50),NRP,LPTR(17),NELMS(16),RNELMS(16),
*DICOP,NSUCC,CELLS,NOF,CIV(50),NTORD,RCRC,ROWNR(50),
*RCCT(50),TYPCEL,KEY1,KEY2,K,KK,I,J,JJ,IP1,NROW,NCOL,
*EXB(200),TREAT(16),RNVAR,S,REDELM,COLNR(50),T,LL,UL,
*TIEV(2)
REAL RP(50),LIST(400),DIVAL(200),CRV(50),VAL1(200),
*VAL2(200)
READ(10,101) TYPCEL,IP1,NCOL,NTORD
101 FORMAT(1E15)
102 FORMAT(8F10.3)
201 FORMAT(1X,1E15)
202 FORMAT(1X,8F10.3)
NROW=1
IF (TYPCEL.GT.2) NROW=IP1
WRITE(9,209) TYPCEL,NROW,NCOL,NTORD
209 FORMAT(1X,'TYFECEL,NRCW,NCOL,NTORD,' ,4I5)

```

```

T=TYPCEL
IF ((T.NE.5).AND.(T.NE.61).AND.(T.NE.62).AND.
*(T.NE.71).AND.(T.NE.72)) GO TO 10
  READ(10,101) (RCCT(I),I=1,NTORD)
  IF ((T.EQ.62).GR.(T.EQ.72.))
*   CALL FNAME(1,NIF,NCF)
  LIST(1)=0.C
  LPTR(1)=1
  NVAR5=IP1
  NTREAT=IP1
  IF(T.EC.71) NTREAT=NCOL
  IF(T.EC.71) NVARS=NCOL
  DO 5 KK=1,NTREAT
    TREAT(KK)=KK
5   CONTINUE
  NELMS(1)=0
  NTREAT=NCOL
C   READ(10,101) NTRETE
C   WRITE(9,401) NTRETE
C401  FORMAT(1X,'NUMBER OF TREATMENTS',I5,' INDICES')
C   READ(10,101) (TREAT(I),I=1,NTRETE)
C   WRITE(9,201) (TREAT(I),I=1,NTRETE)
C   READ(10,101) NEXB
C   WRITE(9,302) NEXB
C302  FORMAT(1X,'NUMBER OF EX BLOCKS',I5,' INDICES')
C   IF (NEXB.GE.1) READ(10,101) (EXB(I),I=1,NEXB)
C   IF (NEXB.GE.1) WRITE(9,201) (EXB(I),I=1,NEXB)
  CALL RFARAF(NIP,NRP,IP,RP)
  IF ((T.NE.62).AND.(T.NE.72.))
*   CALL FNAME(1,10,NOF)
  CALL FNAME(2,9,9)
  CALL FNAME(2,10,NOF)
  GO TO 99
10  IF ((TYPCEL.GT.3).AND.(TYPCEL.LT.6)) GO TO 12
  READ(10,102) (CRV(I),I=1,NTORD)
  WRITE(9,202) (CRV(I),I=1,NTORD)
  GO TO 14
12  READ(10,101) (CIV(I),I=1,NTORD)
  WRITE(9,201) (CIV(I),I=1,NTORD)
14  CALL MSTRCR(NVAR5,NELMS,LIST,LPTR,TOTELM,NTREAT,NIP,
*   IP,NRP,RP,NIF,NCF,TREAT)
*   IF(TYPCEL.LE.2) CALL DICCT(NOF,NTORD,TYPCEL,IF1,CRV,
*   LIST,LPTR,TOTELM,RCCT,TIEV)
*   IF(TYPCEL.EQ.3) CALL TC3(NTREAT,LIST,LPTR,CRV,NTORD,
*   IP1,NCOL,RCCT)
*   IF(TYPCEL.EQ.4) CALL TC4(NTREAT,LIST,LPTR,CIV,
*   NTORD,IP1,NCOL,RCCT)
*   IF(TYPCEL.EQ.63) CALL MCGUNT(LIST,LPTR,CRV,RCCT,
*   TREAT)
*   IF(TYPCEL.EQ.73) CALL CGCH
*   (NTREAT,NOF,LIST,LPTR,RP,NTORD,CRV,NROW,NCOL,RCCT)
  IF(TYPCEL.EQ.73) GO TO 99
  WRITE(9,301)
  WRITE(NOF,301)
C 301  FORMAT(1X,' CELL MATRIX:')
  DO 80 I=1,NROW
    LL=I+(I-1)*NCOL
    UL=I*NCOL
    WRITE(9,201) (RCCT(J),J=LL,UL)
    WRITE(NOF,201) (RCCT(J),J=LL,UL)
C 80  CONTINUE
C 99  RETURN
  END
-----
C  SUBROUTINE COCH
C * (NTREAT,NCF,LIST,LPTR,RP,NTORD,CRV,NROW,NCOL,RCCT)
C   THIS FINDS WHICH DATA ELEMENTS ARE SUCCESSES,

```

```

C      BY COMPARING LIST WITH PT, COUNTS THE # OF SUCCESSES
C      PER RCW & COL, SENDS TOTS INTO RCCT, INDEX 1-NROW=
C      SUCCESS PER RCW(BLOCK),NROW+1 TO NROW+NCCL=SUCCESS
C      PER CCL(TREAT)
REAL LIST(400),RP(50),CRV(50),PT,MULT,V
INTEGER NTREAT,LPTR(17),NROW,NCCL,RCCT(50),N,IC,IR,
* LL,UL,NT
N=IFIX(FLCAT(NTORD)/2)
IF (N.NE.NTREAT) WRITE (NOF,101) N,NTREAT
101  FORMAT(1X,
* 'EXPECTED NUMBER OF TREATMENTS BASES ON NUMBER',
* '/OF PARTITION VALUES WAS',I4,
* ' WHICH DOES NOT AGREE WITH THE ',
* '/LISTED NUMBER OF TREATMENTS,',I4)
NT=NROW+NCCL
DO 5 I=1,NT
RCCT(I)=0
5    CONTINUE
DO 10 I=1,N
MULT=CRV((2*I))
PT=CRV((2*I)-1)*MULT
IC=I+NRCW
LL=LPTR(I)
UL=LPTR(I+1)-1
DO 20 J=LL,UL
IR=J-LL+1
V=LIST(J)*MULT
IF(V.GT.PT) GO TO 20
RCCT(IR)=RCCT(IR)+1
RCCT(IC)=RCCT(IC)+1
20  CONTINUE
10  CONTINUE
NROWP1=NRCW+1
WRITE (NOF,201) (RCCT(I),I=1,NROW)
WRITE (NOF,202) (RCCT(J),J=NROWP1,NT)
201  FORMAT(1X,'ROW TOTALS: '/20I3)
202  FORMAT(1X,'COLUMN TOTALS: '/20I3)
RETURN
END

```

```

C-----
SUBROUTINE TC3
* (NTREAT,LIST,LPTR,CRV,NTORD,NROW,NCCL,RCCT)
INTEGER NTREAT,LPTR(17),NTORD,NRCW,RCCT(50),CIV(50),
*KEY2(200),NPAIR,RCWNR(50),COLNR(50),NCOL,KEY1(200)
REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
WRITE(9,101)
101  FORMAT(1X,'INTO TC3')
CALL PAIRST(VAL1,VAL2,KEY1,KEY2,LIST,LPTR,NPAIR)
CALL TC3LN(1,CRV,VAL1,KEY1,NPAIR,NROW,NCCL,RCWNR)
CALL TC3LN(0,CRV,VAL2,KEY2,NPAIR,NROW,NCCL,COLNR)
CALL RCCTR(NRCW,NCOL,NPAIR,RCWNR,COLNR,RCCT)
RETURN
END

```

```

C-----
SUBROUTINE TC4
* (NTREAT,LIST,LPTR,CIV,NTORD,NROW,NCCL,RCCT)
INTEGER NTREAT,LPTR(17),NTORD,NRCW,NCCL,RCCT(50),
*KEY2(200),NPAIR,RCWNR(50),COLNR(50),CIV(50),KEY1(200)
REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
CALL PAIRST(VAL1,VAL2,KEY1,KEY2,LIST,LPTR,NPAIR)
CALL TC4LN(1,CIV,KEY1,NPAIR,NROW,NCOL,RCWNR)
CALL TC4LN(0,CIV,KEY2,NPAIR,NROW,NCOL,COLNR)
CALL RCCTR(NRCW,NCOL,NPAIR,RCWNR,COLNR,RCCT)
RETURN
END

```

```

C-----
SUBROUTINE PAIRST

```



```

* (VAL1,VAL2,KEY1,KEY2,LIST,LPTR,NPAIR)
INTEGER NTREAT,LPTR(17),NTORD,NROW,NCCL,RCCT(50),
*KEY2(200),NPAIR,RCWNR(50),COLNR(50),CIV(50),
*NRCCLM,NTCP,ICIV,ICRV,NIN,I,J,JK,RCRC,KEY1(200)
REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
WRITE(9,101)
101 FORMAT(1X,'INTO PAIRST')
NPAIR=LPTR(2)-LPTR(1)
DO 10 I=1,NPAIR
  J=I+NPAIR
  VAL1(I)=LIST(I)
  VAL2(I)=LIST(J)
  KEY1(I)=I
  KEY2(I)=I
10 CONTINUE
WRITE(9,305)
305 FORMAT(1X,' VALUES OF VAL1,VAL2 BEFORE SORT ')
WRITE(9,301) (VAL1(I),I=1,NPAIR)
WRITE(9,301) (VAL2(I),I=1,NPAIR)
CALL SHSORT(VAL1,KEY1,NPAIR)
CALL SHSORT(VAL2,KEY2,NPAIR)
WRITE(9,302)
302 FORMAT(1X,' VALUES OF VAL1,VAL2 AFTER SORT ')
WRITE(9,301) (VAL1(I),I=1,NPAIR)
WRITE(9,301) (VAL2(I),I=1,NPAIR)
301 FORMAT(7F10.3)
RETURN
END
-----
C
SUBROUTINE TC3LN
* (RORC,CRV,VAL,KEY,NPAIR,NROW,NCOL,RCRCNR)
INTEGER NTREAT,LPTR(17),NTORD,NROW,NCCL,RCCT(50),
*KEY2(200),NPAIR,RCWNR(50),COLNR(50),CIV(50),IK,
*NRCCLM,NTCP,ICIV,ICRV,NIN,I,J,JK,RCRC,RCRCNR(50),
*KEY1(200),KEY(200)
REAL LIST(400),VAL(200),CRV(50)
WRITE(9,101)NPAIR
C101 FORMAT(1X,' INTO TC3LN ,NPAIR:',15)
I=1
IF(RCRC.LT.1) GO TO 10
ICRV=1
K=0
NTCP=NFCW-1
GC TO 2C
10 CONTINUE
ICRV=NFCW
K=NROW-1
NTOP=NROW+NCOL-2
20 CONTINUE
WRITE(9,203) IK,ICRV,K,NTOP,VAL(IK),CRV(ICRV)
203 FORMAT(1X,' IK,ICRV,K,NTOP:',4I5,
* ' VAL(K),CRV(ICRV)',2F10.3)
* IF ((ICRV.GT.NTOP).OR.(VAL(IK).LE.CRV(ICRV)))
* GC TO 12
ICRV=ICRV+1
GO TO 20
12 RCRCNR(KEY(IK))=ICRV-K
WRITE(9,204) KEY(IK),RCRCNR(KEY(IK))
204 FORMAT(1X,' KEY(IK)',15,' RCRCNR(KEY(IK))',15)
IK=IK+1
IF(IK.GT.NPAIR) GO TO 99
GC TO 20
99 RETURN
END
-----
C
SUBROUTINE RCCTR(NROW,NCCL,NPAIR,ROWNR,CCLNR,RCCT)
C

```

```

      INTEGER NTREAT,LPTR(17),NTORD,NROW,NCCL,RCCT(50),
*KEY2(200),NPAIR,RCWNR(50),COLNR(50),CIV(50),CELLNR,
*NRCELM,NTCP,ICIV,ICRV,NIN,I,J,JJ,K,KEY1(200)
      REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
      WRITE(9,101)
101  FORMAT(1X,' INTO RCCTR')
      NRCELM=NRW*NCOL
      DO 10 I=1,NRCELM
10   RCCT(I)=0
      CONTINUE
      DO 20 I=1,NPAIR
          CELLNR=(ROWNR(I)-1)*NCOL+COLNR(I)
          RCCT(CELLNR)=RCCT(CELLNR)+1
20  CONTINUE
      RETURN
      END
C-----
      SUBROUTINE TC4LN(RORC,CIV,KEY,NPAIR,NROW,NCOL,RCRCNR)
      INTEGER NTREAT,LPTR(17),NTORD,NRCW,NCCL,RCCT(50),
*KEY2(200),NPAIR,RCWNR(50),COLNR(50),CIV(50),
*NRCELM,NTCP,ICIV,ICRV,NIN,I,J,JJ,K,RCRC,RCRCNR(50),
*KEY1(200),KEY(200)
      REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
      NRCELM=NRW*NCOL
      IF (RORC.LT.1) GO TO 10
          NTCP=NRCW
          ICIV=1
          K=0
          GO TO 20
10  NTOP=NROW+NCOL
          ICIV=NRCW+1
          K=NROW
20  J=0
          DO 30 I=ICIV,NTOP
              NIN=CIV(I)
              IF (NIN.EQ.0) GO TO 30
              DO 15 JJ=1,NIN
                  J=J+1
                  RCRCNR(KEY(J))=I-K
15  CONTINUE
30  CONTINUE
      RETURN
      END
C-----
      SUBROUTINE DICOT(NOF,NTORD,TYPCEL,IP1,CRV,LIST,LPTR,
* TOTELM,RCCT,TIEV)
      INTEGER DICOP,LPTR(17),UL,I,NSUCC,L,D,KEY1(400),
* NTORD,TYPCEL,NPAIR,IP1,RCCT(50),NFAIL,I,TIEV(2),
* KEY2(400),TCTELM
      REAL DIVAL(200),CRV(200),LIST(TOTELM),PT,DIF,ADIF,
* EPSILN
      NSUCC=0
      TIEV(1)=0
      TIEV(2)=0
      EPSILN=0.0001
      NPAIR=LPTR(2)-1
      WRITE(9,206) (CRV(I),I=1,NTORD)
206  FORMAT(1X,'CRV',7F10.3)
C-----THIS SECTION IF DICOTOMIZE BY PARTITION----
      IF (TYPCEL.EQ.2) GO TO 40
          PT=CRV(I)
          DO 10 I=1,NPAIR
              IF (LIST(I).GT.PT) NSUCC=NSUCC+1
              DIF=AES(LIST(I)-PT)
              IF (DIF.GT.EPSILN) GO TO 8
                  IF (LIST(I).LE.PT) TIEV(1)=TIEV(1)+1
                  IF (LIST(I).GT.PT) TIEV(2)=TIEV(2)+1
          10 CONTINUE
      8  CONTINUE
      END

```

```

8          WRITE (9,207) LIST(I),PT,NSUCC
207        FCFORMAT(1X,'LIST(I),PT,NSUCC',2F10.3,15)
10        CONTINUE
          GC TO 59
C-----THE FOLLOWING SECTION IF DICOTOMIZE BY LIST---
40        CONTINUE
          DO 41 I=1,NPAIR
            KEY1(I)=I
41        CONTINUE
          DO 42 I=1,NTORD
            KEY2(I)=I
            DIVAL(I)=CFV(I)
42        CONTINUE
          CALL SHSCRT(LIST,KEY1,NPAIR)
          CALL SHSCRT(DIVAL,KEY2,NTORD)
          L=1
          O=1
50        IF((L.GT.NPAIR).OR.(O.GT.NTORD)) GO TO 99
            DIF=LIST(L)-DIVAL(O)
            ADIF=ABS(DIF)
            WRITE (9,209) LIST(L),DIVAL(O),DIF,NSUCC
209        FORMAT(1X,'LIST,DIVAL,DIF,NSUCC',3F10.5,15)
            IF (ADIF.GT.EPSLN) GO TO 60
            NSUCC=NSUCC+1
            L=L+1
            GO TO 50
60        IF(DIF.GT.C) D=D+1
            IF(DIF.LT.C) L=L+1
            GO TO 50
99        WRITE (9,101) NSUCC,NPAIR
101       FORMAT(1X,' # OF STANDARD SUCCESSES ',15,' OF ',15)
          IF (IP1.EQ.0) NSUCC=NPAIR-NSUCC
C***REFLECT THE # OF SUCCESSES IF LIST CONTAINS FAILURES***
C          OR VALUES ABOVE PARTITION ARE FAILURES
          WRITE (9,105) NSUCC,NPAIR
          WRITE (NCF,105) NSUCC,NPAIR
105      FORMAT(1X,' NLMBER OF SUCCESSES ',15,' OF ',15)
          NFAIL=NPAIR-NSUCC
          RCCT(1)=NSUCC
          RCCT(2)=NFAIL
          RETURN
          END
C-----
SUBROUTINE MCCUNT (LIST,LPTR,CRV,RCCT,TREAT)
C THIS GETS CCUNTS FOR FOUR CELLS OF THE MCNEMAR TEST:
C 2 SUCCESS,2 FAILURE, SUCCESS-FAILURE,FAILURE-SUCCESS
C FBS=1.0 IF SUCCESS IN FIRST TREATMENT IS BELCW CRV(1),
C =-1.0 IF SUCCESS IS ABOVE CRV(1)
REAL LIST(400),CRV(50),FBS,SBS,FPV,SPV,F,S
INTEGER LPTR(17),RCCT(4),N,K,I,LF,LS,TREAT(16)
N=LPTR(2)-1
FBS=CRV(3)
SBS=CRV(4)
FPV=CRV(1)*FBS
SPV=CRV(2)*SBS
LF=1+IFIX(CRV(5))
LS=2-IFIX(CRV(5))
DO 5 I=1,4
  RCCT(I)=0
5 CONTINUE
DO 10 I=1,N
  K=1
  F=LIST(I)*FBS
  S=LIST(I+N)*SBS
  IF(F.GT.FPV) K=K+LF
  IF(S.GT.SPV) K=K+LS
  RCCT(K)=RCCT(K)+1

```

```

10 CONTINUE
   IF (CRV(5).LT.0.5) GO TO 20
   WRITE (NCF,111) TREAT(1),TREAT(2)
   GO TO 30
20 WRITE (NCF,111) TREAT(2),TREAT(1)
111 FORMAT(1X,'TREATMENTS: '/ ' BEFORE',I5,' AFTER',I5)
30 RETURN
END
C*****
SUBROUTINE MSTRDR(NVARS,NELMS,LIST,LPTR,TOTELM,
* NTREAT,NIP,IF,NRP,RP,NIF,NOF,TREAT)
C THIS SUBPROGRAM GROUP READS ALL OF THE DATA FILE EXCEPT
C THE NAMES, AND LINES 4-7 OF THE CPTIGNS FILE. IT CHOOSES
C THE DATA ELEMENTS OF THOSE VARIABLES INCLUDED IN THE TEST,
C AND PUTS THEM INTO THE VECTOR LIST. IT THEN REMOVES THE
C DATA ELEMENTS IF THE USER INDICATED TO EXCLUDE PAIRS OR
C BLOCKS. THE INTEGER VECTOR LPTR DIMENSIONS LIST; LPTR(1)
C = 1; LPTR(I) = 1ST ELEM OF ITH VAR IN LIST. NTREAT =
C # TREATMENTS/VARS IN TEST; TOTELM IS TOTAL ELEM IN LIST
C LPTR(NTREAT+1) = TOTELM+1. NELMS IS INT VECTOR OF ELEM
C PER VAR AFTER BLOCKS/PAIRS EXCLUDED; TREAT IS INT VECTOR
C OF INDICES OF ORIGINAL DATA SET WHICH ARE INCLUDED IN TEST
C IP IS INT VECTOR OF INT PARAMS; RP IS REAL PARAMS

SUBROUTINES
MSTRDR READS THE FILES
SUBLST PREPARES DATA ELEMENTS FOR REDUCTION
NULIST INCLUDES ONLY TREAT SPECIFIED INTO LIST
EXBLCC EXCLUDES PAIRS OR BLOCKS AS REQ
INTEGER NIF,NFILE,IHOLD(16),NVARS,TOTELM,EXB(200),
* NTREAT,IP(50),LPTR(17),NELMS(16),RNELMS(16),RNVARS,
* DICOP,NSLCC,CELLS,NTEST,TREAT(16),REDELM,NRP
REAL RP(10),LIST(400),DIVAL(200)
101 FORMAT(16I5)
102 FORMAT(8F10.3)
201 FORMAT(1X,16I5)
202 FORMAT(1X,8F10.3)
DICOP=0
CALL FNAME (1,NIF,NOF)
CALL FNAME (2,NIF,9)
CALL FNAME (2,NIF,NOF)
READ (NIF,101) (IHOLD(I),I=1,16)
NVARS=IHOLD(1)
LPTR(1)=1
TOTELM=0
DO 10 I=1,NVARS
  IP1=I+1
  NELMS(I)=IHOLD(IP1)
  LPTR(IP1)=LPTR(I)+NELMS(I)
  TOTELM=TOTELM+NELMS(I)
10 CONTINUE
READ (NIF,102) (LIST(I),I=1,TOTELM)
CALL SUBLST(LIST,LPTR,NVARS,NTREAT,TOTELM,TREAT)
CALL RPARAM(NIP,NRP,IP,RP)
WRITE (9,407)
407 FORMAT(1X,' LISTPTR VECTOR AND LIST AFTER REDUCTION' )
WRITE(9,201) (LPTR(I),I=1,17)
WRITE(9,202) (LIST(I),I=1,TOTELM)
WRITE(9,203)
303 FORMAT(1X,' NVARS,TOTELM,NTEST,NTREAT,NEXB,NIF,NRP' )
WRITE(9,201) NVARS,TOTELM,NTEST,NTREAT,NEXB,NIF,NRP
RETURN
END
C-----
SUBROUTINE SUELST(LIST,LPTR,NVARS,NTREAT,TOTELM,TREAT)
C THIS SUBROUTINE FEADS THE VALUES FOR THE INCLUDED TREATS,
C AND EXCLUDED BLOCKS, IF ANY, AND REDUCES THE ORIGINAL

```

C DATA SET OF THE DATA FILE INTO THE DATA LIST FOR THE TEST
 C SUBROUTINE NULIST INCLUDES TREATMENTS, EXBLOC EXCLUDES

```

C
  REAL LIST(400)
  INTEGER LPTR(17), NVAR, NTREAT, TOTELM, TREAT(16), NEXB,
  * RNELMS(16), RNVAR, EXB(200)
101  FORMAT(16I5)
102  FORMAT(8F10.3)
201  FORMAT(1X,16I5)
202  FORMAT(1X,8F10.3)
  DO 8 I=1, NVAR
    TREAT(I)=0
  8  CONTINUE
  READ(10,101) NTREAT
  WRITE(9,201) NTREAT
301  FORMAT(1X,'NUMBER OF TREATMENTS',I5,' INDICES')
  READ(10,101) (TREAT(I),I=1,NTREAT)
  WRITE(9,201) (TREAT(I),I=1,NTREAT)
  READ(10,101) NEXB
  WRITE(9,202) NEXB
302  FORMAT(1X,'NUMBER OF EX BLOCKS',I5,' INDICES')
  IF (NEXB.GE.1) READ(10,101) (EXB(I),I=1,NEXB)
  IF (NEXB.GE.1) WRITE(9,201) (EXB(I),I=1,NEXB)
  CALL NULIST
  * (NELMS,TOTELM,LIST,NTREAT,TREAT,LPTR,RNVAR)
  * IF (NEXB.GE.1) CALL EXBLOC(LPTR,TOTELM,LIST,NTREAT,
  * NELMS,NEXB,EXB,RNELMS)
  RETURN
  END
  
```

```

-----
C
  SUBROUTINE NULIST
  * (NELMS,TOTELM,LIST,NTREAT,TREAT,LPTR,RNVAR)
  INTEGER NIF,NFILE,IHCLD(16),TOTELM,NTEST,EXB(200),
  * NTREAT,IP(50),NRP,LPTR(17),NELMS(16),NULPTR(17),KP1,
  * RNVAR,NCF,I1,UL,LL,NVAR,TREAT(16)
  REAL RP(10),LIST(400)
  IF (NVAR.EQ.NTREAT) GO TO 98
  DO 5 K=1,17
    NULPTR(K)=0
  5  CONTINUE
  I=0
  NULPTR(1)=1
  DO 10 II=1,NTREAT
    IIP1=II+1
    K=TREAT(II)
    KP1=K+1
    LL=LPTR(K)
    UL=LPTR(KP1)-1
    DC 15 J=LL,UL
      I=I+1
      LIST(I)=LIST(J)
  15  CONTINUE
  NULPTR(IIP1)=LL+1-LL+NULPTR(II)
  10  CONTINUE
  TOTELM=I
  WRITE(9,408) (NULPTR(I),I=1,17)
  408  FORMAT(1X,'NULPTR',/16I5)
  LPTR(1)=1
  DO 20 K=1,17
    LPTR(K)=NULPTR(K)
  20  CONTINUE
  98  RNVAR=NTREAT
  RETURN
  END
  
```

```

-----
C
  SUBROUTINE EXELOC
  * (LPTR,TOTELM,LIST,NTREAT,NELMS,NEXB,EXB,RNELMS)
  
```

```

      INTEGER NIF,NFILE,IHOLD(16),TOTE LM,NTEST,EXB(200),I,
      *NTREAT,IP(50),NRP,LPTR(17),NELMS(16),EXI(200),NEXI,K,
      *RNELMS(16),TEP1, NCF, LL,UL,NVARS,TREAT(16),EXTFIS,J
      REAL RP(10),LIST(400)
      IF (NEXB.EC.0) GO TO 97
      K=0
C GET INDICES OF LIST ELEMENTS TO BE EXCLUDED
      LPTR(1)=1
      DO 10 I=1,NTREAT
        DO 12 J=1,NEXB
          K=K+1
          EXI(K)=LPTR(I)+EXB(J)-1
        12 CONTINUE
      10 CONTINUE
      NEXI=NK+1
      EXI(NEXI)=17
      K=1
      J=0
      TEP1=TOTE LM+1
C RELOAD LIST, EXCLUDING ELEMENTS AS INDICATED BY EXI
      DO 20 I=1,TEP1
        EXTHIS=EXI(K)
        IF (I.LT.EXTHIS) J=J+1
        IF (I.LT.EXTHIS) LIST(J)=LIST(I)
        IF (I.GE.EXTHIS) K=K+1
      20 CONTINUE
C CHANGE LIST PCINTER, NUMBER OF ELEMENTS
      LPTR(1)=1
      TOTE LM=TCTE LM-NEXB*NTREAT
      DO 25 I=1,NTREAT
        RNELMS(I)=NELMS(TREAT(I))-NEXB
      25 CONTINUE
      NTP1=NTREAT+1
      DO 30 I=2,NTP1
        LPTR(I)=LPTR(I)-(I-1)*NEXB
      30 CONTINUE
      LL=NTREAT+2
      DO 40 K=LL,17
        LPTR(K)=0
      40 CONTINUE
      GO TC 99
      97 DO 98 I=1,NTREAT
        RNELMS(I)=NELMS(TREAT(I))
      98 CONTINUE
      99 RETURN
      END
C*****
C THIS SECTION CONTAINS THE SUBROUTINES CALLED BY MCRE THAN
C ONE TEST SUBROUTINE. THEY APPEAR IN ALPHABETICAL CRDER:
C EXPF, FINDS, FINDT, MUVAR, NCRMF, NORMV, PARCCM, RANQUE
C RHOER, RNQ, RPARAM
C-----
C SUBROUTINE EXPF(LIST,N,RP,F,NOF)
C THIS SUBROUTINE FINDS THE THEORETICAL EXPONENTIAL
C DISTRIBUTION F, WHERE L=RP(1)=SCALE PARAMETER LAMBDA
      REAL A,L,X,LIST(400),RP(50),F(400)
      INTEGER I
      L=RP(1)
      DO 10 I=1,N
        X=LIST(I)
        IF(X.LE.0) F(I)=0
        IF(X.GT.0) F(I)=1.0-EXP(-(L*X))
      10 CONTINUE
      WRITE(NOF,101) L
      101 FORMAT(1X,'WITH HYPOTHESISED DISTRIBUTUION ',
      * 'EXPONENTIAL'/1X,'WITH PARAMETER LAMEDA=',F(10.4))
      RETURN

```

```

      END
C-----
C SUBROUTINE FINDS(LIST,LPTR,S,K)
C THIS SUBROUTINE FINDS THE DIFFERENCES BETWEEN TWO
C EMPIRICAL CDFS FOR THE TWO VECTORS CONTAINED IN LIST
C 1ST VAR X HAS # ELMS M AND COUNTER I; 2D VAR Y HAS # ELMS
C N AND COUNTER J; E CDF ARE (I-1)/M AND (J-1)/N; STEP
C COUNTER OF SMALLER VAR UNTIL N+1 OR M+1, THEN INCR OTHER
      REAL EPSILN, FN, FM, X(100), Y(100), DIF, ADIF, S(200),
      * ATMAX, ATMIN, T SUP, PDIF, LIST(400), RP(50), TMAX, TMIN
      INTEGER I, J, K, L, M, N, KEYI(100), KEYJ(100), JJ, IP(50),
      * LPTR(17), NCF, TREAT(17), II
      EPSILN=0.0001
      M=LPTR(2)-1
      N=LPTR(3)-LPTR(2)
      FN=FLOAT(N)
      FM=FLCAT(M)
      DO 10 I=1, M
          KEYI(I)=I
          X(I)=LIST(I)
10      CONTINUE
          CALL SHSCRT(X, KEYI, M)
          DO 12 J=1, N
              KEYJ(J)=J
              Y(J)=LIST(M+J)
12      CONTINUE
          CALL SHSCRT(Y, KEYJ, N)
          K=0
          L=0
          I=1
          J=1
20      CONTINUE
          IF ((I.GT.M).AND.(J.GT.N)) L=1
101      FORMAT(1X, 'ITERATION #, I, J, STOP?', 4I5, ' DIF:', F10.4)
111      FORMAT(1X, 'S(K):', F10.4)
          II=I
          IF(I.GT.M) II=M
          JJ=J
          IF(J.GT.N) JJ=N
          DIF=X(II)-Y(JJ)
          ADIF=ABS(DIF)
          IF(ADIF.GT.EPSILN) GC TO 25
          IF(I.LE.M) I=I+1
          PDIF=FLCAT(I-1)/FM-FLCAT(J-1)/FN
          K=K+1
          S(K)=PCIF
          WRITE (9,1C1) K, I, J, L, DIF
          IF (K.GT.0) WRITE (9,111) S(K)
          IF(J.LE.N) J=J+1
          K=K+1
          S(K)=PCIF
          WRITE (9,1C1) K, I, J, L, DIF
          IF (K.GT.0) WRITE (9,111) S(K)
          IF(L.EQ.1) GO TO 30
          GC TO 20
25      CONTINUE
          IF((DIF.LT.0).AND.(I.LE.M)) I=I+1
          IF((DIF.GT.0).AND.(J.LE.N)) J=J+1
          K=K+1
          S(K)=FLCAT(I-1)/FM-FLOAT(J-1)/FN
          WRITE (9,1C1) K, I, J, L, DIF
          IF (K.GT.0) WRITE (9,111) S(K)
          IF((DIF.LT.0).AND.(I.GT.M).AND.(J.LE.N)) J=J+1
          IF((DIF.GT.0).AND.(J.GT.N).AND.(I.LE.M)) I=I+1
30      RETURN
      END

```

```

C-----
C      SUBROUTINE FINDT(S,F,N,TSUP,TMIN,TMAX,NOF)
C      THIS FINDS THE DIFFERENCE IN CDF'S OF THEORETICAL
C      DISTRIBUTION F AND EMPIRICAL DIST S AT THE X'S DETERMINED
C      BY THE VALUES OF THE SAMPLE DATA VARIABLE ELEMENTS. THE
C      MAX ABSOLUTE DIFFERENCE IS TSUP. T IS VECTOR OF DIFFERENCE:
      REAL S(400),F(400),TSUP,ATMIN,ATMAX,TMIN,TMAX,T(800)
      INTEGER N,I,II,NOF
      NP1=N+1
      T(1)=F(1)
      T(2)=F(1)-S(1)
      DO 10 II=2,N
         I=II-1
         IT2=(II*2)
         IT1=IT2-1
         T(IT1)=F(II)-S(I)
         T(IT2)=F(II)-S(II)
10      CONTINUE
      TMIN=2.0
      TMAX=-2.0
      DO 12 J=1,IT2
         IF(T(J).GT.TMAX) TMAX=T(J)
         IF(T(J).LT.TMIN) TMIN=T(J)
12      CONTINUE
      ATMAX=ABS(TMAX)
      ATMIN=ABS(TMIN)
      TSUP=ATMIN
      IF(ATMAX.GT.TSUP) TSUP=ATMAX
101     WRITE(NO,101) TMAX,TMIN,TSUP
      FORMAT(1X,'MAXIMUM DIFFERENCE, F(X)-S(X),',F10.4/1X,
      *' MINIMUM DIFFERENCE:',F10.4/1X,
      *' LARGEST ABSOLUTE DIFFERENCE:',F10.4)
      RETURN
      END

```

```

C-----
C      SUBROUTINE MUVAR(LIST,N,MU,VAR)
C      THIS SUBROUTINE FINDS THE SAMPLE MEAN AND VARIANCE
      REAL LIST(400),MU,VAR,XBAR,SUMSQ
      INTEGER N
      XBAR=0.0
      SUMSQ=0.0
      DO 10 I=1,N
         XBAR=XBAR+LIST(I)
10      CONTINUE
      MU=XBAR/FLCAT(N)
      DO 20 I=1,N
         SUMSQ=SUMSQ+(LIST(I)-MU)**2
20      CONTINUE
      VAR=SUMSQ/FLCAT(N-1)
      WRITE(NO,101) MU,VAR
101     WRITE(NO,101) MU,VAR
      FORMAT(1X,' ESTIMATED MU',F10.3,' VARIANCE',F10.3)
      RETURN
      END

```

```

C-----
C      SUBROUTINE NORMF(LIST,N,RP,F,NOF)
C      THIS SUBROUTINE FINDS THE THEORETICAL NORMAL
C      DISTRIBUTION F, WHERE MU=RP(1), SIGMA SQUARED=RP(2)
      REAL PX,MU,SIGSQ,X,LIST(400),RP(50),F(400)
      INTEGER I
      MU=RP(1)
      SIGSQ=RP(2)
      DO 10 I=1,N
         X=LIST(I)
         Z=(X-MU)/SIGSQ
         CALL NCRMV(1,Z,PX)
         F(I)=PX

```



```

10 CONTINUE
WRITE(NOF,101) MU,SIGSQ
101 FORMAT(1X,'WITH HYPOTHESISED DISTRIBUTUION NORMAL'/1X,
* WITH PARAMS MU=',F10.4,', AND SIGMA SQUARED=',F10.4')
RETURN
END
-----
C SUBROUTINE NOFMV(OPTION,Z,P)
C IF OPTION=1, THIS TAKES A Z (STANCRD NORMAL) AND RETURNS
C A P, IF OPTION=2, TAKES A P AND RETURNS A Z
C NUMERICAL INTEGRATION, TRAPAZCIDAL, STEP=0.1 STD DEV
C INITIALIZE BCTH Z AND P IN CALLING SUBROUTINE
REAL Z,P,PI,CCNST,OLD,NEW,ZZ,PP,CUMP,X,DELTP,EXCESP,
*DELX
INTEGER OPTION,I,J,UL,ONE
PI=3.1415927
CUMP=0
CONST=1.0/(2*PI)**0.5
CLD=CCNST
IF (OPTION.EQ.2) GO TO 50
ZZ=ABS(Z)
UL=IFIX(ZZ*10.0)
DO 10 I=1,UL
X=FLOAT(I)*0.1
NEW=CCNST*EXP(-0.5*X*X)
CUMP=CUMP+0.05*(OLD+NEW)
CLD=NEW
IF (CUMP.GT.0.48) GO TO 12
WRITE(9,104) I,NEW,CUMP,X
FCRMT(1X,'I',15,'NEW,CUMP,X',3F7.3)
104 CONTINUE
10 DELX=ZZ-X
NEW=CCNST*EXP(-0.5*ZZ*ZZ)
CUMP=CUMP+C.5*DELX*(OLD+NEW)
WRITE(9,104) I,NEW,CUMP,X
12 P=CUMP+C.5
IF (Z.LT.0.0) P=0.5-CUMP
GO TO 50
50 CONTINUE
PP=P-0.5
X=0.0
IF (P.LT.0.5) PP=0.5-P
60 X=X+0.1
NEW=CCNST*EXP(-0.5*X*X)
DELTP=0.05*(OLD+NEW)
CUMP=CUMP+DELTP
CLD=NEW
WRITE(9,109) X,NEW,CUMP
109 FCRMT(1X,'X,NEW,CUMP',3F7.3)
IF (CUMP.LT.PP) GO TO 60
EXCESP=CUMP-PP
ZZ=X-0.1*EXCESP/DELTP
Z=ZZ
IF (P.LT.0.5) Z=-ZZ
95 WRITE(9,101) OPTION,P,Z
101 FORMAT(1X,'OPTION, PROB, VAL OF Z:',15,2F10.3)
RETURN
END
-----
C SUBROUTINE PAFCOM(V,T,S,NCF)
C THIS FIND THE DIFFERENCES BETWEEN THE ITH AND JTH ELEMENT
C OF VECTOR V, FOR ALL I<J, 1,...,T; PRINTS THE DIFFERENCES
C AND THE SUPPLIED STANCRD DEVIATION S
REAL V(17),S
INTEGER NCF,T,I,J,TM1,IP1
WRITE(NOF,101) S
101 FORMAT(9X,'//FAIRWISE COMPARISONS//'/1X,'IF AND ONLY',

```

```

** IF THE NULL HYPOTHESIS IS REJECTED //1X, THESE,
** COMPARISONS HAVE MEANING. COMPARE THE //1X,
** DESIRED QUANTILE OF THE T DISTRIBUTION //1X,
** TIMES THE STANDARD DEVIATION //1X,
** WITH THE FOLLOWING ABSOLUTE DIFFERENCES IN THE //1X,
** SUMS OF ROW RANKS: //1X,
** I J |SR(I)-SR(J)| //1X

```

```

TM1=T-1
CO 10 I=1, TM1
IP1=I+1
DO 20 J=IP1, T
X=V(I)-V(J)
AX=ABS(X)
WRITE(102, 102) I, J, AX
FCRMT(1X, 215, 3X, F10.4)
102 CONTINUE
20 CONTINUE
10 RETURN
END

```

```

C-----
SUBROUTINE RANQUE(NVALS, VALS, SVALS, RANK, Q, KEY, NTREAT,
* NBLCK, CPTICN)
* DIMENSION VALS(NVALS), Q(200), RANK(NVALS), RANQ(400),
* SVALS(NVALS), KEY(NVALS), JJV(400), RANGE(200), QKEY(200)
* INTEGER KEY, NEWTIE, OLDTIE, LL, NTREAT, NBLCK, CPTICN,
* JJV, NL, JJ, QKEY
* REAL DIF, VALS, SVALS, RANK1, RANK, AVE, EPSILN, RANG, Q, RANGE

```

```

C-----
C THESE TWO SUBROUTINES RANK VALUES
C Q THE RANK OF RANGES WITHIN BLOCKS
C DIF DIFFERENCE BETWEEN KTH AND K+1TH SORTED VALUE
C EPSILN TOLERANCE OF ROUNDING ERROR, I.E., CLOSE ENOUGH TO
C SAY TWO VALS ARE TIED
C NEWTIE KTH AND K+1TH SORTED VALUES ARE TIED (LOGICAL INT)
C OLDTIE K-1TH AND KTH SORTED VALUES ARE TIED (LOGICAL INT)
C KEY ORIGINAL INDEX OF VALS, E.G., KEY(1) HAS ORIG INDEX
C OF SMALLEST VAL, KEY(NVALS) HAS INDEX OF LARGEST VA
C RANK RANK OF THE UNSORTED (ORIGINAL) VALUES
C LL LOWEST INDEX OF TIED SORTED VALUES
C AVE AVERAGE OF LL & HIGHEST INDEX OF TIED SORTED VALUE
C RANQ IF OPTION=1, TOTAL SET OF RANKS, =2 RANKS W/I BLOCK
C =3 RANKS W/I BLOCK PLUS RANKS OF RANGES OF BLOCKS

```

```

C-----
IF (CPTICN.GE.2) GO TO 51
NL=NVALS
CO 10 I=1, NVALS
SVALS(I)=VALS(I)
KEY(I)=I
10 CONTINUE
CALL RANQ (SVALS, KEY, NL, RANQ)
CO 15 I=1, NVALS
RANK(I)=RANQ(I)
15 CONTINUE
GO TO 95
51 NL=NTREAT
DO 20 I=1, NBLCK
QKEY(I)=I
DO 25 J=1, NL
JJ=I+(J-1)*NBLCK
SVALS(J)=VALS(JJ)
JJV(J)=~J
KEY(J)=J
25 CONTINUE
CALL RANQ (SVALS, KEY, NL, RANQ)
RANGE(I)=SVALS(NL)-SVALS(1)
WRITE (9, 1C7) (RANG(K), K=1, NL)

```

```

107   FCRMAT (1X, 7F10.3)
      DC 30 J=1,NL
      RANK(JJV(J))=RANQ(J)
30    CONTINUE
20    CONTINUE
      IF (OPTICN.EQ.3) CALL RNQ(RANGE,QKEY,NBLOCK,Q)
95    RETURN
      END

```

```

-----
C     SUBROUTINE RHCER(X,NPAIR,RHO,B1,B0,OPTION)
C     THIS SUBROUTINE TAKES X, A VECTOR COMPOSED OF TWO EQUAL
C     LENGTHED VARIABLES, AND FINDS THE CORRELATION COEFFICIENT
C     RHO (PEARSON'S: CCVARIANCE/STDEVI*STDEV2)
      REAL X(400),RFO,SSA,SSB,FN,AB,DA,DB,AMEAN,BMEAN,
      *B1,B0,RHC,CENCM
      INTEGER NPAIR,OPTION
      FN=FLOAT(NPAIR)
      AMEAN=0.0
      BMEAN=0.0
      SSA=0.0
      SSB=0.0
      AB=0.0
      B1=0.0
      B0=0.0
      DO 10 I=1,NPAIR
         AMEAN=AMEAN+X(I)
         BMEAN=BMEAN+X((I+NPAIR))
10    CONTINUE
      AMEAN=AMEAN/FN
      BMEAN=BMEAN/FN
      DO 20 I=1,NPAIR
         DA=X(I)-AMEAN
         DB=X((I+NPAIR))-BMEAN
         AB=AB+(DA*DB)
         SSA=SSA+(DA*DA)
         SSB=SSB+(DB*DB)
20    CONTINUE
      DENOM=(SSA**0.5)*(SSB**0.5)
      RHO=AB/DENCM
      IF (OPTICN.EQ.1) B1=AB/SSB
      IF (OPTICN.EQ.1) B0=AMEAN-B1*BMEAN
      IF (OPTICN.EQ.2) B1=AB/SSA
      IF (OPTICN.EQ.2) B0=BMEAN-B1*AMEAN
      WRITE(9,122) RHO,B0,B1
122   FORMAT(1X,' RFO,B0,B1:',3F10.4)
      RETURN
      END

```

```

-----
C     SUBROUTINE RNC(SVALS,KEY,NL,RANQ)
C     THIS SUBROUTINE SORTS AND RANKS A VECTOR FROM RANQUE
C     OR ANY OTHER SUBROUTINE; SVALS ARE UNSORTED VALUES WHICH
C     WHICH ARE RETURNED SORTED AND DETERMINE RANKS; KEY IS THE
C     VECTOR 1,...,NL WHERE NL IS # ELMS IN SVAL; RANQ IS RANKS
C     OLDTIE,NEWTIE ARE LOGICALS TO START AND END TIED VALUES
      REAL SVALS(NL),RANQ(NL),EPSILN,
      *AVE,DIF
      INTEGER NL,I,LL,IP1,NEWTIE,OLDTIE,KEY(NL),K
      EPSILN=0.0001
      WRITE(9,222)
222   FORMAT(10X,'**** START RANKING IN RNC ****')
      CALL SHSORT(SVALS,KEY,NL)
      OLDTIE=0
      DO 10 I=1,NL
         RANQ(I)=FLCAT(I)
10    CONTINUE
      DO 30 I=1,NL
         NEWTIE=0
30    CONTINUE

```

```

      K=KEY(I)
      RANQ(K)=FLCAT(I)
      IF (I.GT.1) WRITE (9,104) I,K,RANQ(K),RANQ(I)
C104  * * RANQ(I),FIC.3)
      IP1=I+1
      DIF=SVALS(IP1)-SVALS(I)
      IF (I.EC.NL) DIF=1.0
      IF (DIF.LT.EPSILN) NEWTIE=1
      IF (NEWTIE.EQ.CLDTIE) GO TO 29
      IF (NEWTIE.LT.CLDTIE) GO TO 15
      LL=I
      GC TO 29
15     AVE=0.5*(FLOAT(I+LL))
      DO 20 J=LL,I
      RANQ(KEY(J))=AVE
20     CONTINUE
29     CLDTIE=NEWTIE
      WRITE (9,1C7) (RANQ(K),K=1,NL)
C30    CONTINUE
      WRITE (9,121)
121    FORMAT(1X,'SORTEC VALUES')
      WRITE (9,1C7) (SVALS(K),K=1,NL)
      WRITE (9,122)
122    FORMAT(1X,'RANKS CF VALUES')
      WRITE (9,1C7) (RANQ(K),K=1,NL)
      WRITE (9,1C8) (KEY(K),K=1,NL)
C107   FORMAT (1X,7F10.3)
C108   FORMAT (1X,7I10)
      RETURN
      END

```

```

C-----
C SUBROUTINE RPARAM(NIP,NRP,IP,RP)
C THIS READS THE INTEGER AND REAL PARAMETERS OF THE OPTIONS
C FILE FOR EITHER MSTRDR OR CELLER
      INTEGER NIP,NRP,IP(50)
      REAL RP(1C)
101    FORMAT(16I5)
102    FORMAT(8F10.3)
      READ(10,1C1) NIP
      IF (NIP.GE.1) READ(10,101) (IP(I),I=1,NIP)
      READ(10,101) NRP
      IF (NRP.GE.1) REAC(10,102) (RP(I),I=1,NRP)
      RETURN
      END

```

```

C-----
C SUBROUTINE THYP(A,X,NCF)
C THIS WRITES THE RESULT FOR A KOLMOGOROV, LILLIEFORS
C TEST TO THE CLPUT FILE WHEN THE HYP=1
      WRITE(NOF,101) X
101    FORMAT(1X,'THE NULL HYPOTHESIS: F*(X)=S(X)'/1X,'THE',
      * ' TEST STATISTIC IS LARGER OF THE ABSCLUTES VALUES' /
      * 2X,' OF THE TWC DIFFERENCES: '/1X,F10.4)
      RETURN
      END

```

```

C-----
C SUBROUTINE THYPB(X,NCF)
C THIS WRITES THE RESULT FOR A KOLMOGOROV, LILLIEFORS
C TEST TO THE CLPUT FILE WHEN THE HYP=2
      WRITE(NOF,101) X
101    FORMAT(1X,'THE NULL HYPOTHESIS: F*(X)<=S(X)'/1X,'THE',
      * ' TEST STATISTIC IS LARGEST DIFFERENCE' /
      * 2X,' OF F*(XI)-S(XI): '/1X,F10.4)
      RETURN
      END

```

```

C-----
      SUBROUTINE THYPC(X,NOF)

```

```

C THIS WRITES THE RESULT FOR A KOLMOGOROV, LILLIEFORS
C TEST TO THE OUTPUT FILE WHEN THE HYP=3
  WRITE(NOF,101) X
101  FORMAT(1X,'THE NULL HYPOTHESIS: F*(X)=>S(X)*/1X,'THE ',
* TEST STATISTIC IS SMALLEST (OR MOST NEGATIVE)'/
* 2X,'OF F*(XI)-S(XI): '/1X,F10.4)
  RETURN
  END
C*****
C THIS SECTION OF THE PROGRAM CONTAINS THE TEST SUBROUTINES
C IN THE ORDER OF THEIR TEST NUMBER; TESTS ARE SEPARATED BY
C -/-/
  SUBROUTINE BINST(NOF,RCCT,RP,IP,ALPHAT,CILL,CIUL)
C THIS MASTER SUBROUTINE FINDS ALPHA HAT OF A BINOMIAL TEST
C BASED ON THE TOTAL NUMBER N, THE NUMBER OF SUCCESSES T,
C THE HYPOTHESIZED PROBABILITY PSTAR, AND FORM OF HYPOTHESIS.
C THE NULL HYPOTHESIS IS EVALUATED. IT THEN FINDS LOWER AND
C UPPER CONFIDENCE INTERVAL BOUNDS (CILL,CIUL) FOR ACTUAL
C PROBABILITY BASED ON THE NR OF SUCCESSES AND FAILURES.
C IF N<20, EXACT BINOMIAL COMPUTATIONS WILL BE PERFORMED,
C OTHERWISE THE NORMAL APPROXIMATION WILL BE USED.
  INTEGER NOF,IP(50),RCCT(50),N,T,HYP
  REAL ALPHA,ALPHAT,CIUL,CILL,RP(50),PSTAR,ALPHYP,
  * NKV(21)
  T=RCCT(1)
  N=RCCT(2)+T
  PSTAR=RP(1)
  ALPHYP=1.0-RP(2)
  HYP=IP(1)
  WRITE(NOF,201) N,T,PSTAR,ALPHYP
201  FORMAT(10X,'BINOMIAL TEST: '/1X,'OF ',15,' ELEMENTS ',
* 'OF DATA SET ',15,' WERE SUCCESSES '/1X,
* ' THE HYPOTHESIZED P STAR IS: ',F10.5/
* 1X,' THE LEVEL OF SIGNIFICANCE IS: ',F10.5)
  IF (HYP.EC.1) WRITE(NOF,301)
301  FORMAT(1X,
* ' THE NULL HYPOTHESIS IS P=P STAR, ALT IS P<>P STAR')
  IF (HYP.EC.2) WRITE(NOF,302)
302  FORMAT(1X,
* ' THE NULL HYPOTHESIS IS P<=P STAR, ALT IS P>P STAR')
  IF (HYP.EC.3) WRITE(NOF,303)
303  FORMAT(1X,
* ' THE NULL HYPOTHESIS IS P>=P STAR, ALT IS P<P STAR')
  IF (N.GT.20) GO TO 30
  CALL BINH2C(N,T,PSTAR,HYP,ALPHAT,NKV)
  CALL CIBINC(N,T,ALPHYP,NKV,CILL,CIUL)
  GO TO 99
30  CONTINUE
  CALL BINORM(N,T,PSTAR,HYP,ALPHAT,ALPHYP,CILL,CIUL)
99  CONTINUE
  WRITE(9,101) ALPHAT,CILL,CIUL
  WRITE(NOF,101) ALPHAT,CILL,CIUL
101  FORMAT(1X,' ALPHA HAT, CI LOWER, UPPER',3F10.3)
  RETURN
  END
C-/-/
  SUBROUTINE BINORM
  * (N,T,PSTAR,HYP,ALPHAT,ALPHYP,CILL,CIUL)
C THIS SUBROUTINE FINDS THE ALPHA HAT OF A BINOMIAL DIST
C AND THE CONFIDENCE LIMITS OF THE SAMPLES' PROBABILITY
C CALLING THE SUBROUTINE NORMV
C ALPHYP THE ALPHA SPECIFIED IN TEST 1.0-LEVEL OF SIGNIF
C ALPHAT THE ALPHA RETURNED BY THE NORMV SUBROUTINE BASED
C ON THE NUMBER OF SUCCESSES AND TOTAL SAMPLE
C ZHAT THE EMPIRICAL Z = STANDARD NORMAL N SUCC MINUS
C E(N SUCCESSES)=(N*PSTAR), DIVICED BY STD DEV (NPQ)
C ZHYP Z RETURNED FROM NORMV BASED ON ALPHA* OF HYP

```

```

INTEGER N,T,HYP
REAL ZHAT,Z,ALPH,ALPHAT,CILL,CIUL,NPST,STDEV,ZHYP
NPST=FLOAT(N)*PSTAR
ZHAT=(FLCAT(T)-NPST)/((NPST*(1.0-PSTAR))**.5)
CALL NORMV(1,ZHAT,ALPHAT)
IF (HYP.GT.1) GO TO 20
  IF (ZHAT.GT.0) ALPHAT=(1.0-ALPHAT)*2.0
  IF (ZHAT.LE.0) ALPHAT=ALPHAT*2.0
  GO TO 3C
20 IF (HYP.EQ.2) ALPHAT=1.0-ALPHAT
30 STDEV=(FLCAT(T*(N-T))/((FLOAT(N))**.3))**.5
CALL NORMV(2,ZHYP,ALPHAT)
TDIVN=FLCAT(T)/FLCAT(N)
CILL=TDIVN+ZHYP*STDEV
CIUL=TDIVN-ZHYP*STDEV
RETURN
END

```

```

C-----
C SUBROUTINE NCHUZZK(N,NKV)
C THIS SUBROUTINE PRODUCES THE NKV VECTOR OF N CHOOSE K
C FOR K=0,... THE INDICES OF NKV ARE ONE
C HIGHER THAN THE CORRESPONDING K, E.G., I=1,...,N+1
REAL NUM,DENOM,NKV(21)
INTEGER UL,MEC,NP1,IP1,NP1MI,N,I
NP1=N+1
UL=IFIX (N*.5)
MED=IFIX (NP1*.5)
NKV(1)=1
NKV(NP1)=1
NUM=1.0
DENOM=1.0
DO 10 I=1,UL
  IP1=I+1
  NP1MI=NP1-I
  NUM=NUM*FLCAT(NP1MI)
  DENOM=DENOM*FLCAT(I)
  NKV(IP1)=NUM/DENOM
  IF(I.LT.MEC) NKV(NP1MI)=NKV(IP1)
10 CONTINUE
WRITE (9,101)
101 FORMAT(1X,' N CHOOSE K VECTOR')
WRITE (9,102) (NKV(I),I=1,NP1)
102 FORMAT(8F10.3)
RETURN
END

```

```

C-----
C SUBROUTINE BINPMF(P,N,NKV,VECLIM,PQV,PMF,CUMPR)
C THIS SUBROUTINE RETURNS THE VECTOR PQV CORRESPONDING TO
C P**K TIMES Q**(N-K) FOR K=0,...,VECLIM, AND PART OR ALL
C OF PMF FOR BINOMIAL. LET VECLIM = N TO PRODUCE ENTIRE PMF.
C FOR LINE SEARCH, IT WILL BE LESS. VECTOR NKV FROM NCHUZZK
C CUMPR IS CUMULATIVE PROBABILITY
INTEGER I,VECLIM,N,K,UL
REAL P,Q,PQV(21),PV(21),QV(21),PHOLD,QHOLD,NKV(21),
*CUMPR,DIVER,PG,PMF(21)
CUMPR=0.0
C=1.0-P
UL=VECLIM+1
NP1=N+1
DIVER=P/C
WRITE(9,103) P,Q,PQ,DIVER,VECLIM
103 FORMAT(1X,' P, C, PQ, DIVER, ',4F10.7,' VECLIM',I5)
PQ=Q**N
DO 20 I=1,UL
  PV(I)=PQ
  PQ=PQ*DIVER
  PMF(I)=PQV(I)*NKV(I)
20

```

```

      CUMPR=CUMPR+PMF(I)
20  CONTINUE
    WRITE (9,101) VECLIM, P, CUMPR
101  FORMAT (1X, 'VECLIM', I5,
    *      ' P', F8.3, ' CUM PROB:', F8.3, ' P Q VECTOR:')
    WRITE (9,102) (PQV(I), I=1, NP1)
102  FORMAT(8F10.3)
    RETURN
    END

```

```

-----
C  SUBROUTINE BINH20(N, T, PSTAR, HYP, ALPHAT, NKV)
C  THIS FIGURES THE ALPHA HAT VALUE, AND RETURNS THAT
C  VALUE AND THE N CHOSE K VECTOR TO THE MAIN SUBROUTINE
C  BINT ST.
    REAL PSTAR, ALPHAT, NKV(21), PQV(21), PMF(21), CUMPR
    INTEGER N, T, HYP, I, TP1, NP1
    CALL NCHLZK(N, NKV)
    NP1=N+1
    CALL BINPMF(PSTAR, N, NKV, N, PQV, PMF, CUMPR)
    TP1=T+1
    ALPH1=0
    ALPH2=0
    IF (HYP.EQ.3) GO TO 20
    DO 15 I=TP1, NP1
      ALPH1=ALPH1+PMF(I)
15  CONTINUE
    ALPH=ALPH1
    IF (HYP.EQ.2) GO TO 30
    DO 25 I=1, TP1
      ALPH2=ALPH2+PMF(I)
25  CONTINUE
    ALPH=ALPH2
30  IF (HYP.NE.1) GO TO 99
    IF (ALPH.GT.ALPH1) ALPH=ALPH1
    ALPH=ALPH*2.0
99  WRITE(9,101) ALPH
101  FORMAT(1X, ' VALUE OF ALPHA HAT:', F10.3)
    ALPHAT=ALPH
    RETURN
    END

```

```

-----
C  SUBROUTINE CIBINO(N, T, ALPHYP, NKV, CILL, CIUL)
C  THIS USES A BISECTION SEARCH TO FIND THE UPPER AND
C  LOWER LIMITS (CILL, CIUL) TO THE PROBABILITY OF SUCCESS
C  GIVEN THE NUMBER OF SUCCESSES T AND TOTAL SAMPLE SIZE N.
C  NKV FROM NCHLZK.
C  THIS USES SYMMETRY OF BINOMIAL, E.G., A DIST WITH
C  8 SUCCESSES OF 10 WILL HAVE THE LIMITS OF PROBABILITIES
C  1.0 - LIMITS OF A DIST WITH 2 SUCC OF 10 (UPPER=>LOWER)
    INTEGER N, T, TT, I, J, TT1, LCOUNT
    REAL ALPHYP, NKV(21), CILL, CIUL, MLEP, LALPH, UALPH
    UALPH=ALPHYP*.5
    LALPH=1.0-UALPH
    NDIV2=IFIX(FLCAT(N+1)*.5)
    TT=T
    IF (T.GT.NDIV2) TT=N-T
    TT1=TT-1
    MLEP=FLOAT(TT)/FLOAT(N)
    PLL=0.0
    PUL=0.0
    IF (TT1.GE.0) PLL=BISEX(N, TT1, NKV, LALPH)
    IF (TT.LE.N) PUL=BISEX(N, TT, NKV, UALPH)
    IF (T.LE.NDIV2) GO TO 10
    CILL=1.0-PLL
    CIUL=1.0-PUL
    GO TO 20
10  CILL=PLL

```

```

      CIUL=PUL
      IF (CILL.LT.0.0) CILL=0.0
      IF (CIUL.GT.1.0) CIUL=1.0
20    WRITE (9,101) CILL,CIUL
101  FORMAT(1X,' PERCENT LOWER LIMIT',F10.3,
      * ' UPPER LIMIT',F10.3)
      RETURN
      END

```

```

C-----
C      FUNCTION BISEX(N,T,NKV,ALPH)
C THIS DOES A BISECTION SEARCH TO FIND THE EXACT P FOR WHICH
C THERE IS A 1.-ALPHA/2 OR ALPHA/2 PROB OF T SUCCESSSES IN
C N TRIALS. NKV IS VECTOR OF N CHOICE K,K=0,...,N
      REAL NKV(21),ALPH,LO,MED,HI,EPSILN,CUMPR,PQV(21),
      *GAP,P,EPSM1,PMF(21)
      INTEGER N,T,LCCOUNT,LLIM
      EPSILN=0.002
      EPSM1=1.0-EPSILN
      LCCOUNT=0
      LLIM=64
      IF (ALPH.LT.0.5) GC TC 10
      LO=0.0
      HI=FLCAT(T)/FLOAT(N)
      GC TO 20
10    LO=FLOAT(T)/FLOAT(N)
      HI=1.0
20    MED=0.5*(LO+HI)
      WRITE(9,201) LO,MED,HI,LCCOUNT
      IF ((MED.LE.EPSILN).OR.(MED.GE.EPSM1)) GC TO 95
201  FORMAT(1X,' LC,MED,HI, LOOP COUNT',3F10.5,I5)
      LCCOUNT=LCCOUNT+1
      CALL BINPMF(MED,N,NKV,T,PQV,PMF,CUMPR)
      GAP=CUMPR-ALPH
      IF ((ABS(GAP).LT.EPSILN).OR.(LCCOUNT.GT.LLIM)) GO TO 95
      IF (GAP.LT.0.0) GO TO 30
      LO=MED
      GC TO 20
30    HI=MED
      GO TC 20
95    WRITE (9,101) ALPH,MED
101  FORMAT(1X,' ALPHA:',F10.3,' CI LIMIT:',F10.3)
      BISEX=MED
      RETURN
      END

```

```

C-----
C      SUBROUTINE QUANTL(NOF,RCCT,RP,IP,ALPHAT,CILL,CIUL,
      * LIST,LPTR,TIEV,TYPCEL)
C THIS FINDS THE PROBABILITY THAT A SPECIFIED X,XSTAR
C IS THE SPECIFIED QUANTILE POINT QSTAR IN A DISTRIBUTION
C REPRESENTED BY THE VALS OF SAMPLE. THE CONFIDENCE
C INTERVALS (WHICH ENCLUSE POPULATION QSTAR) ARE GIVEN
      INTEGER NCF,RCCT(50),IP(50),LPTR(17),TIEV(2),T1,T2,
      * T2P2,T1P1,N,NP1,RHAT,SHAT,KEY(400),TYPCEL,HYP
      REAL RP(50),ALPHAT,CILL,CIUL,LIST(400),NKV(21),PSTAR,
      * PMF(21),ALPH1,ALPH2,ALFMIN,ZHAT1,ZHAT2,PQV(21),
      * CUMPR,ZSTAR,RP,CCNFCC,LTAIL,UTAIL,ZRHAT,ZSHAT,ALPHYP
      WRITE(NOF,523)
523  FORMAT(1CX,'**QUANTILE TEST**')
      T1=RCCT(1)-TIEV(1)
      T2=RCCT(1)+TIEV(2)
      HYP=IP(1)
      PSTAR=RP(1)
      ALPHYP=1.0-RP(2)
      HALFA=0.5*ALPHYP
      N=RCCT(2)+RCCT(1)
      WRITE(NOF,334) T1,T2,N
334  FORMAT(1X,' NMBER OF DATA ELEMENTS LESS THAN X*',I5/

```



```

*IX,' NUMBER OF ELEMENTS LESS THAN OR EQUAL TO X*',
*IS/IX,' TCTAL NUMBER OF ELEMENTS',I5)
NP1=N+1
IF (N.GT.20) GO TO 30
CALL NCHUZK(N,NKV)
CALL BINPMF(PSTAR,N,NKV,N,PQV,PMF,CUMPR)
CALL CI20Q (ALHPY,N,PMF,RHAT,SHAT,CONFCC)
WRITE (9,102) T1,T2,RHAT,SHAT
102 FORMAT(IX,' T1,T2,RHAT,SHAT',4I5,' PMF:')
WRITE(S,103) (PMF(I),I=1,NP1)
103 FCRMAT(IX,10F6.3)
T2P2=T2+2
TIPI=TI+1
ALPHA1=0.0
ALPHA2=0.0
DO 10 I=1, TIPI
ALPHA1=ALPHA1+PMF(I)
10 CONTINUE
DO 15 I=T2F2,NP1
ALPHA2=ALPHA2+PMF(I)
15 CCNTINUE
WRITE(S,104) ALPHA1,ALPHA2
104 FORMAT(IX,' ALPHA1, ALPHA2:',2F7.3)
GO TO 40
30 STDEV=(FLCAT(N)*PSTAR*(1.0-PSTAR))**.5
NP=FLCAT(N)*PSTAR
ZHAT1=(T1-NP)/STDEV
CALL NCRMV(1,ZHAT1,ALPH1)
ZHAT2=(T2-NP)/STDEV
CALL NCRMV(1,ZHAT2,ALPH2)
ALPH2=1.0-ALPH2
CALL NCRMV(2,ZSTAR,HALFA)
RHAT=1+(IFIX(NP+ZSTAR*STDEV-EPSILN))
SHAT=1+(IFIX(NP-ZSTAR*STDEV-EPSILN))
ZRHAT=(RHAT-NP)/STDEV
CALL NCRMV(1,ZRHAT,LTAIL)
ZSHAT=(SHAT-NP)/STDEV
CALL NCRMV(1,ZSHAT,UTAIL)
UTAIL=1.0-LTAIL
CONFCC=1.0-LTAIL-UTAIL
WRITE(S,108) CCNFCC,LTAIL,UTAIL
108 FORMAT(IX,' CONFCC,LTAIL,UTAIL',3F7.3)
40 ALFMIN=ALPHA1
IF (ALPHA1.GT.ALPHA2) ALFMIN=ALPHA2
WRITE(S,174) ALPHA1,ALPHA2,ALFMIN
174 FORMAT(IX,' ALPHA1, ALPHA2, ALFMIN:',3F7.3)
IF (HYP.EC.1) ALPHAT=2.0*ALFMIN
IF (HYP.EC.2) ALPHAT=ALPH2
IF (HYP.EC.3) ALPHAT=ALPH1
DO 5 I=1,N
KEY(I)=I
5 CONTINUE
IF (TYPCEL.EQ.5) GO TO 95
CALL SHSCFT (LIST,KEY,N)
WRITE (9,103) (LIST(I),I=1,N)
CILL=LIST(RHAT)
IF(SHAT.GT.N) SHAT=N
CIUL=LIST(SHAT)
WRITE (9,101) ALPHAT,CILL,CIUL
WRITE (NCF,101)ALPHAT,LIST(RHAT),LIST(SHAT),RHAT,SHAT
101 FORMAT(IX,'ALPHA HAT, LOWER AND UPPER BOUNDS',3F9.3/IX,
* 'INDEX OF LOWER LIMIT',I5,' , INDEX OF UPPER LIMIT',I5)
GO TO 99
95 WRITE (9,107) ALPHAT,RHAT,SHAT
WRITE (NCF,107) ALPHAT,RHAT,SHAT
107 FORMAT(IX,'ALPHA HAT',F7.3,
* ' INDICES OF LOWER AND UPPER BOUNDS',2I5)

```

```

99 WRITE (9,109) CONF CO
WRITE (NCF,109) CCFNCC
109 FORMAT(1X,'INTERVAL CONFIDENCE COEFFICIENT:',F10.3)
RETURN
END

C-----
C SUBROUTINE CI20Q (ALPHYP,N,PMF,RHAT,SHAT,CONF CO)
C THIS SUBROUTINE FINDS INDICES RHAT AND SHAT THAT WILL
C INCLUDE WILL GIVE A CONFIDENCE COEFFICIENT OF AT LEAST
C 1.0-ALPHA, USED FOR QUANTILE TEST.
C RHAT IS INDEX OF LOWER LIMIT, SHAT IS INDEX OF UPPER
C BOUNC. SINCE INDICES OF PMF ARE 1
C TO N+1, S=SHAT, R=RHAT
C INTEGER RHAT,SHAT,RMAX,SMIN,N,NP1,I,IM1,J,JP1
C REAL PMF(21),CDF(23),RCDF(23),ALPHYP,HALFA,BEST,CRIT,
* CRIT1,CRIT2,CONF CO
NP1=N+1
HALFA=0.5*ALPHYP
I=1
CCF(1)=PMF(1)
10 CONTINUE
IM1=I
IF (CCF(I).GE.HALFA) GO TO 15
I=I+1
CDF(I)=CDF(IM1)+PMF(I)
WRITE(9,191) I,PMF(I),CDF(I)
191 FCRMAT(1X,'I,PMF(IM1),CDF(I)',I5,2F10.4)
IF (CDF(I).LE.HALFA) GO TO 10
15 RHAT=IM1
J=NP1+1
RCDF(J)=0
20 CONTINUE
JP1=J
J=J-1
RCDF(J)=RCDF(JP1)+PMF(J)
WRITE(9,192) J,RCDF(J),PMF(J)
192 FCRMAT(1X,'J,RCDF(J),PMF(J)',I5,2F10.4)
IF (RCDF(J).LE.HALFA) GO TO 20
SHAT=J
CONF CO=1.0-CDF(RHAT)-RCDF(SHAT)
WRITE (9,101) CDF(RHAT),RCDF(SHAT),CCFNCC,RHAT,SHAT
101 FORMAT(1X,'LO,UP,CONF CO,INDICES, CI:',/1X,
* 3F10.4,2I5)
RETURN
END

C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C SUBROUTINE COXSTU (LIST,LPTR,IP,RP,NCF,NTEST)
C THIS SUBROUTINE DIVIDES THE ONE VARIABLE IN TWO, THEN
C PERFORMS THE SIGN TEST TO TEST THE TREND AS A MATTER OF
C PAIRING THE FIRST HALF DATA ELEMENTS WITH THE SECOND HALF
C THERE ARE TWO OPTIONS: 1=ONE VARIABLE, IN PRESENT ORDER,
C 2=CORRELATION, RANKS OF THE INDEPENDENT VARIABLE
C ORDERING THE DEPENDENT VARIABLE
C INTEGER LPTR(17),IP(50),NCF,TLPTR(3),NHALF,MEC,NHT2,
* HYP,INDEPC,DEPC,KEY(200),NACTUL
C REAL LIST(400),RP(50),TLIST(400),SLIST(400),DEP(200),
* INDEP(200)
WRITE(NCF,523)
WRITE(9,523)
523 FORMAT(1X,'**COX AND STUART TEST FOR TREND**')
IF (IP(2).EQ.1) GO TO 7
N=LPTR(2)-1
INDEPC=0
DEPC=0
IF(IP(3).LT.IP(4)) INDEPC=N
IF(IP(4).LT.IP(3)) DEPC=N
CO 5 I=1,N

```

```

      KEY(I)=I
      INDEP(I)=LIST(I+INDEPC)
      DEP(I)=LIST(I+DEPC)
5     CONTINUE
      CALL SHSORT(INDEP,KEY,N)
      CO 6 I=1,N
          SLIST(I)=DEP(KEY(I))
          WRITE(9,918) I,KEY(I),SLIST(I)
918   FORMAT(IX,'IN COXSTU: I,KEY,SLIST: ',2I5,F10.3)
6     CONTINUE
      GO TO 9
7     CO 8 I=1,N
          SLIST(I)=LIST(I)
8     CONTINUE
9     NHALF=IFIX(0.5*(LPTR(2)-1))
      NHT2=2*NHALF
      TLPTR(1)=1
      TLPTR(2)=1+NHALF
      TLPTR(3)=1+NHT2
      NACTUL=LPTR(2)-1
      HYP=IP(1)
      CO 10 I=1,NHALF
          TLIST(I)=SLIST(I)
10    CONTINUE
      IF (NACTUL.GT.NHT2) GO TO 30
      DO 20 I=NHALF,NACTUL
          TLIST(I)=SLIST(I)
20    CONTINUE
      GO TO 50
30    LL=2+NHALF
      DO 40 I=LL,NACTUL
          IM1=I-1
          TLIST(IM1)=SLIST(I)
40    CONTINUE
      IF (HYP.EQ.1) WRITE (NOF,111)
111   FORMAT(IX,' NULL HYPOTHESIS: THERE IS NO TRENC ')
      IF (HYP.EQ.2) WRITE (NOF,112)
112   FORMAT(IX,' NULL HYPOTHESIS: THERE IS NO UPWARD TREND ')
      IF (HYP.EQ.3) WRITE (NOF,113)
113   FORMAT(IX,
50     * ' NULL HYPOTHESIS: THERE IS NO DOWNWARD TREND ')
      CALL SGNT(TLIST,TLPTR,NOF,NHT2,RP,IP,NTEST)
      RETURN
      END
C- - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - / - - - - - /
SUBROUTINE SGNT(LIST,LPTR,NOF,TOTELM,RP,IF,NTEST)
      REAL A(200),B(200),RP(50),LIST(400),P,EPSSILN,ADIF,DIF,
* NKV(21),ALPHAT,ALPHYP,CIU,CILL,ALPHA1,ALPHA2
      INTEGER LPTR(17),I,J,NELMV,IE,K,M,NPLUS,NMINUS,NTOT,
* HYP,IP(50),NTEST,NOF
C     NELMV IS THE NUMBER OF ELEMENTS IN VECTORS A AND B
      NPLUS=0
      NMINUS=0
523   IF (NTEST.EQ.5) WRITE(NOF,523)
      FORMAT(10X,'**SIGN TEST**')
      NELMV=LPTR(2)-1
      EPSSILN=0.0001
      HYP=IP(1)
      IF (HYP.EQ.1) WRITE (NOF,111)
      IF (HYP.EQ.1) WRITE (9,111)
111   FORMAT(IX,' HC: P(+) = P(-)' / 1X, ' H1: P(+) <> P(-)')
      IF (HYP.EQ.2) WRITE (NOF,112)
112   FORMAT(IX,' HC: P(+) <= P(-)' / 1X, ' H1: P(+) > P(-)')
      IF (HYP.EQ.3) WRITE (NOF,113)
113   FORMAT(IX,' HC: P(+) >= P(-)' / 1X, ' H1: P(+) < P(-)')
      ALPHYP=1.-C-RP(1)
      DO 14 I=1,NELMV

```

```

      J=LPTR(2)-1+I
      DIF=LIST(I)-LIST(J)
      WRITE(9,917) I,J,LIST(I),LIST(J),DIF
917  FORMAT(1X,' IN SIGN:I,J,L(I-J),DIF:',2I5,3F10.3)
      ADIF=ABS(DIF)
      IF(ADIF.LE.EPSILN) GO TO 14
      IF (DIF.LT.0) NPLUS=NPLUS+1
      IF (DIF.GT.0) NMINUS=NMINUS+1
14   CONTINUE
      WRITE(9,987) NPLUS,NMINUS
      WRITE(NOF,986) NPLUS,NMINUS
987  FORMAT(1X,' SIGN TEST +,-',2I5)
986  FORMAT(1X,' NUMBER OF (+):',I5,' NUMBER OF (-):',I5)
      NTOT=NPLUS+NMINUS
      IF (NTOT.LE.20) CALL NCHUZE(NTOT,NKV)
      IF (NTOT.LE.20) CALL BINF20
      * (NTCT,NPLUS,0.5,HYP,ALPHAT,NKV)
      IF (NTOT.GT.20) CALL
      * BINORM(NTCT,NPLUS,0.5,HYP,ALPHAT,ALPHYP,CILL,CIUL)
      ALPHA2=1.C-ALPHA1
      ALFMIN=ALPHA1
      IF (ALPHA1.GT.ALPHA2) ALFMIN=ALPHA2
      IF (HYP.EC.1) ALPHAT=2.0*ALFMIN
      IF (HYP.EC.2) ALPHAT=ALPHA2
      IF (HYP.EC.3) ALPHAT=ALPHA1
      WRITE(NOF,102) NELMV,NTOT,NPLUS,ALPHAT,ALPHYP
102  FORMAT(1X,' NLMBR OF PAIRS:',I5,' N CF '
      * ' PAIRED VALUES NOT TIED',I5,' NUMBER OF FLUSSES',I5,
      * '/' ALPHA HAT:',F10.4/' HYPOTHESISED ALPHA ',F10.4)
      IF (ALPHAT.LT.ALPHYP) WRITE(NOF,104)
      IF (ALPHAT.GE.ALPHYP) WRITE(NOF,105)
104  FORMAT(1X,' REJECT THE NULL HYPOTHESIS ')
105  FORMAT(1X,' DO NOT REJECT THE NULL HYPOTHESIS ')
99   RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
SUBROUTINE MANWHT(LPTR,LIST,NOF,RP)
  INTEGER N1,N2,NT,NHOLD,I,J,LPTR(17),IER
  REAL A(400),LIST(400),RP(1),U,Z,R(400),ALPH,ALPHAT
  WRITE(NOF,523)
523  FORMAT(10X,'**MANN WHITNEY TEST**')
      N1=LPTR(2)-LPTR(1)
      N2=LPTR(3)-LPTR(2)
      NT=N1+N2
      IF(N1.LE.N2) GO TO 30
      DO 10 I=1,NT
          J=LPTR(3)-I
          A(I)=LIST(J)
10   CONTINUE
      NHOLD=N2
      N2=N1
      N1=NHOLD
      GO TO 35
30   DO 32 I=1,NT
          A(I)=LIST(I)
32  CONTINUE
35  CALL UTEST(A,R,N1,N2,U,Z,IER)
      IF (IER.LT.1) GO TO 40
      WRITE(NOF,103)
103  FORMAT(1X,' ALL VALUES OF ONE SUBGROUP ARE TIED ')
40   IF (N2.LT.20) GO TO 95
      CALL NORMV(1,Z,ALPHAT)
      WRITE(NOF,101) U,Z,ALPHAT
101  FORMAT(1X,' TEST STATISTIC U:',F10.4,
      * ' STANDARDIZED Z :',F10.4/1X,
      * ' ALPHA HAT:',F10.4)
      ALPH=1.0-RP(1)

```

```

IF(ALPHAT.LT.ALPH) WRITE(NOF,104)
IF(ALPHAT.GE.ALPH) WRITE(NOF,105)
104 FORMAT(1X,' REJECT THE NULL HYPOTHESIS ')
105 FORMAT(1X,' DC NOT REJECT THE NULL HYPOTHESIS ')
GC TC 99
95 WRITE(401) U,N1,N2
401 FORMAT(1X,' TEST STATISTIC U:',F10.3/1X,
* ' WITH N1=',I4,' , AND N2=',I4)
99 RETURN
END
C-----
SUBROUTINE MCNEMR(RCCT,RP,IP,NOF)
C THIS SUBROUTINE FIGURES THE VALUES OF THE MCNEMAR TEST:
C IF N<21, IT USES THE EXACT VALUES OF THE BINOMIAL,ELSE
C USES THE LINEAR INTERPOLATION OF CHI-SQUARE DF=1
INTEGER RCCT(4),IP(50),B,C,N,T2,NKV(21)
REAL RP(5C),ALPHAT,ALPHYP,T1
WRITE (NOF,523)
523 FORMAT(10X,'**MCNEMAR TEST**')
E=RCCT(2)
C=RCCT(3)
ALPHYP=1.C-RP(1)
T2=B
N=B+C
IF (N.GT.20) GO TO 30
CALL BINH20(N,B,0.5,1,ALPHAT,NKV)
ALPHAT=C.5*ALPHAT
WRITE (NOF,103) B,ALPHYP,ALPHAT
103 * FORMAT(1X,' T2 ',I5,
* ' HYPCTHESISED ALPHA,ALPHA HAT ',2F7.3)
GO TO 40
30 T1=(FLOAT((B-C)**2))/FLCAT(B+C)
CALL CHIGNE (T1,ALPHAT)
WRITE (NOF,104) T1,ALPHYP,ALPHAT
104 * FORMAT(1X,' T1 ',F9.3,
* ' HYPCTHESISED ALPHA,ALPHA HAT ',2F7.3)
40 IF (ALPHAT.LE.ALPHYP) WRITE (NOF,101)
101 FORMAT(1X,' REJECT THE NULL HYPOTHESIS ')
IF (ALPHAT.GT.ALPHYP) WRITE (NOF,102)
102 FORMAT(1X,' DC NOT REJECT THE NULL HYPOTHESIS ')
WRITE (NOF,111) (RCCT(I),I=1,4)
111 FORMAT(1X,' MCNEMAR TABLE: '/25X,' AFTER: '/1X,' BEFORE: ',
* 12X,' SUCCESS FAILURE' /1X,' SUCCESS ',3X,2I10/1X,
* ' FAILURE ',3X,2I10)
RETURN
END
C-----
SUBROUTINE CHIGNE (X,ALPHAT)
C THIS SUBROUTINE FINDS AN APPRGXIMATION FOR
C CHI-SQUARE DF=1 BY LINEAR INTERPOLATION
REAL CHI(8),ALPHAT,P(8)
INTEGER N
CHI(1)=0
CHI(2)=1.323
CHI(3)=2.706
CHI(4)=3.841
CHI(5)=5.024
CHI(6)=6.635
CHI(7)=7.879
CHI(8)=10.828
P(1)=0
P(2)=0.75
P(3)=0.9
P(4)=0.95
P(5)=0.975
P(6)=0.995
P(7)=0.999

```

```

10 F(8)=0.999
CONTINUE
I=I+1
IF ((X.GT.CHI(I)).AND.(I.LT.8)) GO TO 10
IF (I.LT.8) GC TO 20
PHAT=C.999
GC TO 95
20 IM1=I-1
RATIO=(X-CHI(IM1))/(CHI(I)-CHI(IM1))
PHAT=P(IM1)+RATIO*(P(I)-P(IM1))
95 ALPHAT=1.-C-PHAT
RETURN
END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C SUBROUTINE FRIEDM (NTREAT,LPTR,LIST,IP,RP,NOF,TOTELM)
C THIS DOES EITHER THE FRIEDMAN AND DURBIN TESTS;
C IP(1)=0, FRIEDMAN, =1, DURBIN; IF DURBIN, RP(2)=FILLER,
C IP(2)=K=# TREATS PER BLOCK; IP(3)=R=# BLCCKS PER TREAT
C FOR BOTH, RP(1)=CCNF COEF. RANQUE RANKS THE VALUES
C BLOCK BY BLOCK; PARCOM C CMPARES THE SUMS OF RANKS OF
C DIFFERENT TREATMENTS TO TEST FOR PAIRWISE DIFFERENCE
INTEGER NCF,NTREAT,LPTR(17),K,B,I,J,RI,TOTELM,
*DF1,DF2,KEY(40),NBLOCK,INTEST(400),T,R,IP(50)
REAL LIST(400),T2,SVALS(400),RK(400),B2,A2,RJ,
*EPSILN,BK25,BM1,DENOM,NUM,MP1,SRJS,NUMVAR,DENVAR,
*STDEV,SRJ(17),TERM1,TERM2,RP(50),Q(17)
CO 3 I=1,TOTELM
INTEST(I)=1
3 CONTINUE
MP1=-999999.9
IF (IP(1).EQ.1) GC TC 5
WRITE(NOF,=24)
524 FORMAT(10X, '**FRIEDMAN TEST**')
GO TC 8
5 WRITE(NOF,=25)
625 FORMAT(10X, '**DURBIN TEST**')
DO 6 I=1,TOTELM
IF (LIST(I).GT.MP1) MP1=LIST(I)
6 CONTINUE
MP1=MP1+1.C
R=IP(3)
K=IP(2)
DO 7 I=1,TOTELM
IF (LIST(I).NE.RP(2)) GO TO 7
LIST(I)=MP1
INTEST(I)=0
7 CONTINUE
WRITE(9,919) RP(2),MP1
919 FORMAT(1X, 'INTEST VALUES, DURBIN; RP(2),MP1',2F10.3)
WRITE(9,818) (INTEST(I),I=1,TOTELM)
818 FORMAT(1X,12I5)
8 EPSILN=0.C01
NBLOCK=LPTR(2)-1
CALL RANQUE
* (TOTELM,LIST,SVALS,RK,Q,KEY,NTREAT,NBLOCK,2)
WRITE(9,315) (RK(I),I=1,TOTELM)
315 FORMAT(1X,6F10.3)
SRJS=0.0
A2=0.0
E=LPTR(2)-1
T=NTREAT
DO 10 J=1,T
RJ=0
DC 15 I=1,E
RI=I+LPTR(J)-1
IF (INTEST(RI).EQ.0) GO TO 15
RJ=RJ+RK(RI)

```

```

                A2=A2+(RK(RI)**2)
                WRITE (9,312) I,J,RI,RJ,A2
312          FCFORMAT(1X,'I J,RI',3I5,'RJ,A2',2F10.3)
15          CONTINUE
                SRJ(J)=RJ
                SRJS=SRJS+(RJ*RJ)
10          CONTINUE
                IF (IP(1).EQ.1) GO TO 50
                K=T
                B2=SRJS/(FLOAT(B))
                DENOM=A2-B2
                WRITE(9,316) A2,B2,DENOM
316          FORMAT(1X,'A2,B2,DENOM',3F10.3)
                IF (DENOM.LE.EPSILN) GO TO 99
                BM1=FLOAT(B-1)
                BK25=0.25*(FLCAT(B*K*((K+1)**2)))
                NUM=BM1*(B2-BK25)
                T2=NUM/DENCM
                DF1=K-1
                CF2=(B-1)*(K-1)
                WRITE (NCF,112) T2,DF1,DF2
                WRITE (9,112) T2,CF1,DF2
112          FORMAT(1X,'VALUE OF T2:',F10.3,' DF1:',I5,' DF2:',I5)
                NUMVAR=2.C*B*(A1-B1)
                DENVAR=(K-1)*(B-1)
                STDEV=(NUMVAR/DENVAR)**0.5
                CALL PARCCM(SRJ,T,STDEV,NOF)
                GO TO 99
50          CONTINUE
                DF1=12*(T-1)
                CF2=(R*T*(K-1)*(K+1))
                TERM1=SRJS*FLCAT(DF1)/FLCAT(DF2)
                OF1=T-1
                CF2=3*R*CF1*(K+1)/(K-1)
                TERM2=FLCAT(DF2)
                T3=TERM1-TERM2
                WRITE(9,101) TERM1,TERM2,SRJS,T3
101          FORMAT(1X,'TERM1-2,SRJS,T3',4F10.4)
                WRITE(NOF,102) T3,DF1
102          FORMAT(1X,'TEST STATISTIC T3',F10.4,/1X,
* ' TO BE COMPARED WITH CHISQUARED DISTRIBUTION'/1X,
* ' WITH DEGREES OF FREEDOM=',I5)
                DF2=B*K*CF1
                NUMVAR=FLCAT((R*(K+1)*(K-1)))
                NUMVAR=NUMVAR*(FLCAT(CF2)-T3*FLOAT(T))
                CF2=6*DF1*(6*K-T-B+1)
                DENVAR=FLCAT(CF2)
                STDEV=(NUMVAR/DENVAR)**0.5
                CALL PARCCM(SRJ,T,STDEV,NOF)
99          CONTINUE
                RETURN
                END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C     SUBROUTINE QUADE (NTREAT,LPTR,LIST,NCF,TCTELM)
C     THIS PERFORMS THE QUADE TEST.  RANQUE RANKS THE DATA
C     ELEMENTS BLOCK BY BLOCK AND RANKS THE RANGES OF THE
C     BLOCK ELEMENTS (INTO Q).  PARCOM HAS PAIRWISE COMPARISON
C     OF SUMS OF TREATMENT RANKS, TO TEST FOR EQUAL EFFECT
                INTEGER NCF,NTREAT,LPTR(17),K,B,I,J,RI,K,B,TCTELM,
*     OF1,DF2,KEY(400),NBLOCK
                REAL LIST(TOTELM),T2,SVALS(400),R(400),B2,A2,RJ,NUM,
*     EPSILN,BK25,EM1,Q(200),A1,B1,S(400),SIJ,SJ2,SJ,MEAN,
*     DENCM,DENVAR,NUMVAR,STDEV,SRJ(17)
                WRITE(NOF,524)
524          FORMAT(10X,'**QUADE TEST**')
                K=NTREAT

```

```

E=LPT R(2)-1
A1=0
B1=0
SJ2=0
T1=0
CF1=K-1
CF2=DF1*(B-1)
EPSILN=0.CC1
MEAN=0.5*(FLOAT((K+1)))
CALL RANGLC(TCTELP,LIST,SVALS,R,Q,KEY,K,B,3)
DO 10 J=1,K
  SJ=0
  DO 15 I=1,B
    RI=I+LPT R(J)-1
    SIJ=Q(I)*(R(RI)-MEAN)
    A1=A1+SIJ*SIJ
    SJ=SJ+SIJ
    WRITE(9,101) I,J,RI,R(RI),Q(I),SIJ,A1
101  FORMAT(1X,' I,J,RI',3I3,' R,Q(I),SIJ,A1',4F9.3)
15  CONTINUE
  SRJ(J)=SJ
  SJ2=SJ2+SJ*SJ
10  CONTINUE
  B1=SJ2/(FLCAT(B))
  CIF=A1-B1
  IF (DIF.LT.EPSILN) GO TO 95
  T1=(FLOAT(B-1))*B1/DIF
95  WRITE(9,102) A1,B1,T1,DF1,DF2
  WRITE(NCF,102) A1,B1,T1,DF1,DF2
102  FORMAT(1X,'A1',F10.3,' B1',F10.3/
* 1X,' TEST STATISTIC T1:',F10.3,' DF1',I5,' DF2',I5)
  NUMVAR=2.C*(A1-B1)*FLOAT(B)
  DENVAR=FLCAT(CF2)
  STDEV=(NUMVAR/DENVAR)**0.5
  CALL PARCCM(SFJ,K,STDEV,NCF)
  RETURN
END
C- - - - -
C  SUBROUTINE COCHR(NROW,NCOL,RCCT,IP,RP,NOF)
C  THIS SUBROUTINE TAKES ROW AND COLUMN COUNTS RCCT
C  FOR SUCCESSES, DERIVED IN CELLER, AND PERFORMS THE
C  COMPUTATION OF THE CHI SQUARED STATISTIC
  INTEGER SCJSQ,SRISQ,RCCT(50),NSUCC,NROW,NCOL,NUM,
*NOF,IP(50),DENOM,DF
  REAL CHI,FP(5C)
  NSUCC=0
  SCJSQ=0
  SRISQ=0
  DO 10 J=1,NCOL
    IC=J+NROW
    SCJSQ=SCJSQ+RCCT(IC)**2
10  CONTINUE
  DO 20 I=1,NROW
    NSUCC=NSUCC+RCCT(I)
    SRISQ=SRISQ+RCCT(I)**2
20  CONTINUE
  WRITE(NCF,101) SRISQ,SCJSQ,NSUCC,NCOL
101  FORMAT(10X,'*COCHARAN TEST**',/1X,
* 'SUM OF THE SQUARES: R(I)',I5,' C(J)',I5/1X,'FOR',I5,
* ' SUCCESSES IN',I5,' TREATMENTS')
  NUM=(NCOL*(NCOL-1)*SCJSQ)-((NCOL-1)*NSUCC*NSUCC)
  DENOM=(NCOL*NSUCC)-SRISQ
  CHI=FLOAT(NUM)/FLCAT(DENOM)
  CF=NCOL-1
102  WRITE(NCF,102) NUM,DENOM,CHI,DF
  FORMAT(1X,' NUMERATOR,DENOMINATOR:',2I5/1X,' CHI ',
* 'SQUARED VALUE:',F10.3/1X'DEGREES OF FREEDOM:',I5)

```



```
RETURN
END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
SUBROUTINE RCCNT(RCCT,NRCW,NCOL,NGF)
C THIS SUBROUTINE PERFORMS THE CHI-SQUATED TEST ON CELL
C COUNTS GIVEN BY THE VECTOR RCCT (ROW & COLUMN CELL COUNT)
INTEGER RCCT(50),NROW,NCCL,NGF,MAX,RC(50),CC(50),DF,
*UL,LL
REAL EIJ,CHI
MAX=NROW
IF(NCOL.GT.NRCW) MAX=NCOL
DO 10 I=1,MAX
  RC(I)=0
  CC(I)=0
10 CONTINUE
WRITE(9,102) NROW,NCOL
WRITE(NOF,102) NRCW,NCOL
102 FORMAT(1X,' IN RCCNT: NROW,NCOL: ',2I5,' RCCT:')
DO 15 I=1,NROW
  LL=1+NCCL*(I-1)
  UL=NCCL*I
  WRITE(9,103) (RCCT(K),K=LL,UL)
  WRITE(NCF,103) (RCCT(K),K=LL,UL)
103 FORMAT(1X,16I5)
15 CONTINUE
N=0
DO 20 I=1,NROW
  DO 22 J=1,NCOL
    K=J+(I-1)*NCCL
    CC(J)=CC(J)+RCCT(K)
    RC(I)=RC(I)+RCCT(K)
    N=N+RCCT(K)
22 CONTINUE
20 CONTINUE
CHI=0.0
DO 30 I=1,NROW
  DO 32 J=1,NCOL
    EIJ=(FLOAT(CC(J)*RC(I)))/FLOAT(N)
    K=J+(I-1)*NCCL
    CHI=CHI+((FLCAT(RCCT(K))-EIJ)**2)/EIJ
678 WRITE(9,678) I,J,K,EIJ,RCCT(K)
32 FORMAT(1X,' I,J,K,EIJ,CIJ: ',3I5,2F9.3)
30 CONTINUE
DF=(NROW-1)*(NCOL-1)
WRITE(9,101) CHI,DF
WRITE(NOF,101) CHI,DF
101 FORMAT(1X,' CHI-SQUARED VALUE',F10.3,
* ' DEGREES OF FREEDOM',I5)
RETURN
END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
SUBROUTINE GDFIT(LIST,LPTR,IP,RP,NOF)
C THIS GROUP OF SUBS FINDS PARAMETERS IF NECESSARY AND
C COMPARES AN ACTUAL DISTRIBUTION TO A THEORETICAL DIST.
C AVAILABLE THEORETICAL DISTRIBUTIONS ARE NCRML,UNIFORM,
C EXPONENTIAL. SAMPLE SIZE 15 OR MORE DESIRED.
C IP(1)=# DIST; IP(2)=# ESTIMATED PARAMS; IP(3)=N OF ERLANG
C RP(1)=MU,A,LAMBDA; RP(2)=VAR,B,ALPHA; RP(3)=CCNF COEF
REAL RP(50),LIST(400),CUMP(80),PARTV(80),CHI,MU,VAR
INTEGER LPTR(17),N,EJ(80),ESTPAR,NDIST,NCELL,NOF,DF,
*IP(50),RCCT(8C)
WRITE(NOF,523)
523 FORMAT(1X,'** GOODNESS OF FIT TEST (CHI SQUARE)**')
N=LPTR(2)-1
CHI=0.0
CALL CPVGEN(N,CUMP,EJ,NCELL)
```

```

CALL MUVAR(LIST,N,MU,VAR)
RP(4)=MU
RP(5)=VAR
NDIST=IP(1)
IF (NDIST.EC.1)
* CALL NCRMD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NCF)
IF (NDIST.EQ.2)
* CALL UNID(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
IF (NDIST.EQ.3)
* CALL EXP(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
IF (NDIST.EQ.4)
* CALL ERLD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NCF)
IF (NDIST.EQ.5)
* CALL WEID(CUMP,IP,RP,PARTV,LIST,N,NCELL,NCF)
CALL CELCNT(RCCT,LIST,N,NCELL,PARTV)
WRITE(NOF,121)
121 FORMAT(1X,' CELL NR, EJ(J) , O(J) , PARTITION VALUE
* 27('-'')
CO 25 I=1,NCELL
WRITE(NCF,122) I,EJ(I),RCCT(I)
122 FORMAT(1X,3(I5,' '))
IF(I.LT.NCELL) WRITE(NOF,123) PARTV(I)
123 FORMAT(1X,3('-----|'),F10.3)
25 CONTINUE
WRITE(NCF,124) N,N
124 FORMAT(27('-'')/9X,2I8)
CO 10 I=1,NCELL
CHI=CHI+FLCAT((EJ(I)-RCCT(I))*2)/FLOAT(EJ(I))
10 CONTINUE
DF=NCELL-IF(2)-1
WRITE(9,101) N,NCELL,CHI,DF
WRITE(NOF,101) N,NCELL,CHI,DF
101 FORMAT(1X,'WITH SAMPLE LENGTH',I4,' DIVIDED INTO',I3,
* ' CELLS',/1X,'CHI-SQUARED VALUE OF',F10.3/1X,'DF=',I5)
IF(IP(2).EC.0) WRITE(NOF,112) RP(4),RP(5)
112 FORMAT(1X/1X,' SAMPLE MEAN AND VARIANCE:',2F10.4)
RETURN
END

```

```

C-----
C THIS SUBROUTINE CPVGEN(N,CUMP,EJ,NCELL)
C AND THE THE CUMULATIVE PROBABILITY VECTOR CUMP
C AND THE THE CUMULATIVE PROBABILITY VECTOR CUMP
REAL LIST(400),CUMP(80),PARTV(80),ND5,FN
INTEGER LPTR(17),N,EJ(80),ESTPAR,NDIST,NCELL,NOF,IND5,
* CUM(80),INCEL,MER,REM
FN=FLOAT(N)
IF (N.GT.52) GO TO 3
ND5=FN/5.0
IND5=FIX(ND5)
REM=N-(5*IND5)
NCELL=IND5
IF (REM.GT.2) NCELL=IND5+1
CO 12 I=1,NCELL
EJ(I)=5
12 CONTINUE
IF (REM.EC.0) GO TO 20
IF (REM.LE.2) EJ(1)=6
IF (REM.GE.3) EJ(1)=4
IF (REM.EQ.2) EJ(NCELL)=6
IF (REM.EQ.3) EJ(NCELL)=4
20 CONTINUE
GO TO 29
23 NCELL=10
INCEL=FIX(FN/10.0)
REM=N-10*INCEL
MER=10-REM
CO 24 I=1,10

```

```

      IF (I.LE.MER) EJ(I)=INCEL
      IF (I.GT.MER) EJ(I)=INCEL+1
24  CONTINUE
29  CUM(1)=EJ(1)
      CUMP(I)=CUM(1)/FN
      DO 30 I=2,NCELL
          IM1=I-1
          CUM(I)=EJ(I)+CUM(IM1)
          CUMP(I)=FLCAT(CUM(I))/FN
30  CONTINUE
101 WRITE(9,101) N,NCELL
      FORMAT(1X,' CUMP GENERATOR FOR N=',I5,
* '# CELLS',I5,' CUMP VECTOR:')
102 WRITE(9,102) (CUMP(I),I=1,NCELL)
      FORMAT(1X,8F7.3)
      RETURN
      END
C-----
C THIS SUBROUTINE NOFMD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NGF)
C THE SUBROUTINE FINDS THE PARTITIONS VALUES FOR DIVIDING
C THE SAMPLE INTO CELLS, BASED ON THE CUMULATIVE PRCB VECTOR
C CUMP THE PARTITION VECTOR PARTV IS NCELL-1 LONG
      REAL LIST(400),CUMP(80),PARTV(80),RP(50),STDEV,MU,VAR
      INTEGER N,NCELL,IP(50)
      WRITE(NOF,524) IP(2)
524  FORMAT(1X,'NCFMAL DIST',I3,' PARAMETERS ESTIMATED')
      IF (IP(2).EQ.C) GO TO 5
          MU=RP(4)
          VAR=RP(5)
          GO TO 10
5    MU=RP(1)
      VAR=RP(2)
10   STDEV=VAR**0.5
      NCM1=NCELL-1
      WRITE(9,111) MU,VAR,STDEV,NCM1
111  FORMAT(1X,'MU,VAR,STDEV,NCM1',3F8.3,I5)
      DO 20 I=1,NCM1
          CALL NCRMV(2,Z,CUMP(I))
          PARTV(I)=STDEV*Z+MU
          WRITE(9,121) I,CUMP(I),Z,PARTV(I)
121  FORMAT(1X,'I,CUMP(I),Z,PARTV(I)',I5,3F8.3)
120  CONTINUE
20   RETURN
      END
C-----
C THIS ROUTINE CELCNT(RCCT,LIST,N,NCELL,PARTV)
C OF THE SAMPLE CORRESPONDING TO THE EJ'S OF THE THEORETICAL
C DISTRIBUTION
      REAL LIST(400),PARTV(80)
      INTEGER RCCT(80),N,NCELL,KEY(400)
      J=1
      DO 5 I=1,N
          KEY(I)=I
5    CONTINUE
      CALL SHSCRT(LIST,KEY,N)
      DO 6 I=1,NCELL
          RCCT(I)=0
6    CONTINUE
      NCM1=NCELL-1
      DO 20 I=1,N
          IF(LIST(I).LE.PARTV(NCM1)) GO TO 10
          RCCT(NCELL)=RCCT(NCELL)+1
          GO TO 20
10   CONTINUE
          IF(LIST(I).LE.PARTV(J)) GO TO 15
          J=J+1

```

```

                IF (J.LT.NCM1) GO TO 10
                IF (J.EQ.NCM1) GO TO 8
15      RCCT(J)=RCCT(J)+1
20      CONTINUE
        RETURN
        END
C-----
C      SUBROUTINE UNID(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
C      THIS SUBROUTINE FINDS THE PARTITIONS VALUES FOR DIVIDING
C      THE SAMPLE INTO CELLS, BASED ON THE CUMULATIVE PRCB VECTOR
C      CUMP THE PARTITION VECTOR PARTV IS NCELL-1 LONG
        REAL LIST(400),CUMP(80),PARTV(80),RP(50),STDEV,A,B,C
        INTEGER N,NCELL,IP(50)
        WRITE(NO,524) IP(2)
524     FORMAT(1X,'UNIFORM DIST,',I3,' PARAMETERS ESTIMATED')
        A=RP(1)
        B=RP(2)
        C=B-A
        NCM1=NCELL-1
        DO 20 I=1,NCM1
            PARTV(I)=CLMP(I)*C+A
20      CONTINUE
        RETURN
        END
C-----
C      SUBROUTINE EXFD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
C      THIS SUBROUTINE FINDS THE PARTITIONS VALUES FOR DIVIDING
C      THE SAMPLE INTO CELLS, BASED ON THE CUMULATIVE PRCB
C      VECTOR CUMP THE PARTITION VECTOR PARTV IS NCELL-1 LONG
        REAL LIST(400),CUMP(80),PARTV(80),RP(50),LAMBDA
        INTEGER N,NCELL,IP(50)
        WRITE(9,524) IP(2)
524     FORMAT(1X,'EXPONENTIAL DISTRIBUTION,',I3,
*      ' PARAMETERS ESTIMATED')
        LAMBDA=RP(1)
        IF (IP(2).EQ.C) LAMBDA=1.0/RP(4)
        NCM1=NCELL-1
        DO 20 I=1,NCM1
            PARTV(I)=(ALOG(1-CUMP(I)))/(-LAMBDA)
20      CONTINUE
        RETURN
        END
C-----
C      SUBROUTINE ERLD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
C      THIS FINDS THE X TO CORRESPOND TO THE F OF CUMP
C      BY NEWTON-RAPHSON SEARCH IN ERLNUT, TO YIELD PARTV
        REAL RP(50),LIST(400),CUMP(80),PARTV(80),CHI,DN,F,X,
*      LAMBDA, FN
        INTEGER LFTR(17),N,EJ(80),ESTPAR,NDIST,NCELL,NOF,DF,
*      IP(50),RCCT(80),FACT,NCM1
        NCM1=NCELL-1
        FN=FLOAT(IP(3))
        LAMBDA=RP(1)
        IF (IP(1).EQ.1) LAMBDA=FN/RP(4)
        NN=IP(3)-1
        FACT=1
        DO 5 I=1,NN
            FACT=FACT*I
5      CONTINUE
        FN=FLOAT(FACT)
        DO 10 I=1,NCM1
            F=CUMP(I)
            CALL ERLNUT(F,X,NN,FN,LAMBDA)
            PARTV(I)=X
10     CONTINUE
        RETURN
        END

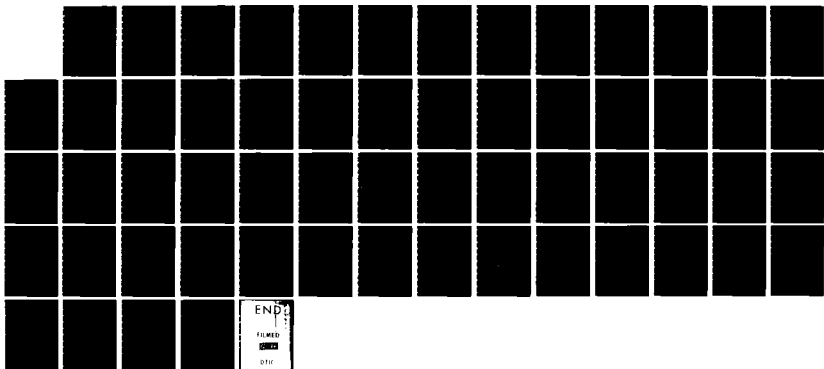
```

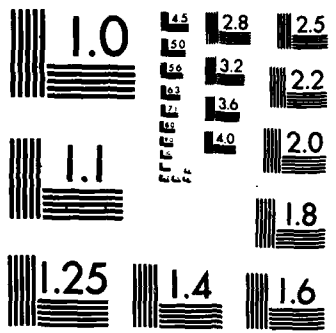
AD-A137 170 NONPARAMETRIC STATISTICS TEST SOFTWARE PACKAGE(U) NAVAL 3/3
POSTGRADUATE SCHOOL MONTEREY CA P J O'BRIEN SEP 83

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

C-----
C SUBROUTINE ERLNUT(F,X,NN, FN, LAMBDA)
C THIS IS THE NEWTON-RAPHSON TECH WHICH TAKES AN F AND
C RETURNS AN X; NN=N-1; FN=(N-1)
REAL F,X, FN, LAMBDA, XLO, XHI, XMED, NF, DIF, ADIF, SLOPE,
* L, LX, LXN, FACT
INTEGER NN, N
NF=FLOAT(NN)
EPSILN=0.0001
L=LAMBDA
X=(NF+1.0)/L
IF(F.GT.0.7) X=X*2*F
IF(F.LT.0.3) X=X*3*F
XLO=0
XHI=100.0*C*X
10 CONTINUE
LX=L*X
LXN=0
FACT=1
DO 15 J=1, NN
FACT=FACT*FLGAT(J)
LXN=LXN+(LX**J)/FACT
15 CONTINUE
FX=1.0-((LXN+1.0)*EXP(-LX))
DIF=F-FX
ADIF=ABS(DIF)
SLOPE=(L*(LX**NN)*EXP(-LX))/FN
IF(DIF.GT.C) XLO=X
IF(DIF.LE.C) XHI=X
X=X+DIF/SLOPE
IF((X.LT.XLO).OR.(X.GT.XHI)) X=0.5*(XLO+XHI)
101 WRITE(9,10) F,FX,DIF,SLOPE,XLO,X,XHI
FORMAT(' F,FX,DIF,SLOPE,XLO,X,XHI:',3F11.7/20X,4F11.7)
IF(ADIF.GT.EPSILN) GO TO 10
RETURN
END

```

```

C-----
C SUBROUTINE WEID(CUMP, IP, RP, PARTV, LIST, N, NCELL, NOF)
C THIS SUBROUTINE FINDS THE PARTITIONS VALUES FOR DIVIDING
C THE SAMPLE INTO CELLS, BASED ON THE CUMULATIVE PRCB
C VECTOR CUMP THE PARTITION VECTOR PARTV IS NCELL-1 LONG
REAL LIST(400), CUMP(80), PARTV(80), RP(50), LAMBDA,
* APOW, ALPH, F, X
INTEGER N, NCELL, IF(50)
WRITE(9,524) IP(2)
524 FORMAT(1X, 'WEIBULL DISTRIBUTION, ', I3,
* ' PARAMETERS ESTIMATED')
LAMBDA=RP(2)
APOW=1.0/FP(1)
NCM1=NCELL-1
DO 20 I=1, NCM1
F=CUMP(I)
X=((ALOG(1.0-C-F))**APOW)/(-LAMBDA)
PARTV(I)=X
20 CONTINUE
RETURN
END

```

```

C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C SUBROUTINE WILCOX(LIST, LPTR, IP, RP, NOF)
C THIS USES RANQUE TO RANK THE ABSOLUTE VALUES OF
C THE 1ST TREATMENT MINUS THE 2D TREATMENT, THEN SUMS THE
C RANKS OF THE PCS DIFFERENCES AND THE NEG DIFFERENCES
C TO GET A T VALUE. THE VAR SHIFT ALLOWS A CONSTANT TO BE
C ADDED TO THE 1ST TREATMENT, E.G., TO SEE IF THE 1ST TREAT
C PLUS A CONSTANT IS SIGNIFICANTLY LARGER THAN THE 2D TREAT
REAL LIST(400), RP(50), ADIF(200), SVALS(400), RANK(400),
* SRPOS, SRNEG, SRANK, SRSQ, P, PR, Z, ALPH1, ALPH2, ALFMIN,

```

```

* CIF, ALPHAT, SHIFT
INTEGER LPTR(17), KEY(400), IP(50), NOF, Q(200),
* NNEG, NPCS, NOCIF, POSDIF(200), IR
SRANK=0.0
SRSQ=0.0
EPSILN=0.0001
IR=0
NNEG=0
SRPOS=0.0
SRNEG=0.0
NPOS=0
NVALS=LPTF(3)-1
NBLOCK=LPTR(2)-1
SHIFT=RP(1)
DO 10 I=1, NBLOCK
  K=I+NBLOCK
  DIF=LIST(K)+SHIFT-LIST(I)
  ABDIF=ABS(CIF)
  IF(ABDIF.LT.EPSILN) GC TO 10
  IR=IR+1
  IF(DIF.LE.C) GC TC 13
  NPCS=NPOS+1
  PCSCIF(IR)=1
13  IF(DIF.GE.C) GC TO 15
  NNEG=NNEG+1
  PCSDIF(IR)=-1
15  ADIF(IR)=AES(DIF)
10  CONTINUE
  NODIF=NBLOCK-NPOS-NNEG
  CALL RANQUE(IF, ADIF, SVALS, RANK, Q, KEY, 1, NBLOCK, 1)
  WRITE(9, 211) (ADIF(I), I=1, IR)
  WRITE(9, 211) (RANK(I), I=1, IR)
211  FORMAT(1X, 'DIF/RNK:', 7F8.2)
  DO 20 I=1, NBLOCK
    P=FLOAT(POSDIF(I))
    IF(P.GT.0.1) SRPOS=SRPOS+RANK(I)
    IF(P.LT.-0.1) SRNEG=SRNEG+RANK(I)
    PR=P*RANK(I)
    SRANK=SRANK+PR
    SRSQ=SRSQ+PR*PR
    ADIF(I)=ADIF(I)*P
20  CONTINUE
  WRITE(NOF, 101) NBLOCK, SHIFT, NPCS, NNEG, NOCIF, SRPOS,
  SRNEG
101  * FORMAT(10X, '**WILCOXCN SIGNED RANK TEST**'/1X,
  * 'IN', I4, ' PAIRS, '/1X, ' WITH 1ST VARIABLE SUBTRACTED',
  * ' FROM THE 2C PLUS A SHIFT OF', F9.3/1X, ' THERE ARE',
  * ' 14, ' POSITIVE DIFFERENCES, ', I4, ' NEGATIVES, AND'/
  * ' 5X, I4, ' TIE(S) '/1X, ' SUM OF POSITIVE RANKS:', F10.2/
  * ' 1X, ' SUM OF NEGATIVE RANKS:', F10.2)
  IF (NBLOCK.LT.20) GO TO 50
  Z=SRANK/(SRSQ**0.5)
  CALL NORMV(1, 2, ALPH1)
  ALPH2=1.0-ALPH1
  ALFMIN=ALPH1
  IF(ALPH1.GT.ALPH2) ALFMIN=ALPH2
  IF(HYP.EQ.1) ALPHAT=2.0*ALFMIN
  IF(HYP.EQ.2) ALPHAT=ALPH1
  IF(HYP.EQ.3) ALPHAT=ALPH2
  ALPHYP=1.0-RP(2)
  WRITE(NCF, 102) SRANK, SRSQ, Z, ALPHAT, ALPHYP
102  FORMAT(1X, 'SUM OF SIGNED RANKS:', F10.2/
  * ' SUM OF SQUARED SIGNED RANKS:', F10.2/ ' T:', F10.3/
  * ' ALPHA HAT:', F10.3, 3X, 'HYPOTHESISED ALPHA:', F10.3)
  IF(HYP.EQ.1) WRITE(NOF, 131)
  IF(HYP.EQ.2) WRITE(NCF, 132)
  IF(HYP.EQ.3) WRITE(NOF, 133)

```



```

131  FORMAT(1X,'H0: E(X)=E(Y), H1: E(X)<>E(Y)')
132  FORMAT(1X,'H0: E(X)>=E(Y), H1: E(X)<E(Y)')
133  FORMAT(1X,'H0: E(X)<=E(Y), H1: E(X)>E(Y)')
      IF(ALPHYP.GT.ALPHAT) WRITE (NOF,121)
      IF(ALPHYP.LE.ALPHAT) WRITE (NOF,122)
121  FORMAT(1X,' REJECT THE NULL HYPOTHESIS')
122  FORMAT(1X,' DC NOT REJECT THE NULL HYPOTHESIS')
      GO TC 95
50   WRITE(NO,103)
      CALL RANQUE(NELOCK,ADIF,SVALS,RANK,Q,KEY,1,NBLOCK,1)
      WRITE(NO,104) (SVALS(I),I=1,NBLCK)
103  FORMAT(1X,'BECAUSE OF SMALL SAMPLE SIZE, TABLES //
* 1X,' SHOULD BE USED. THE DIFFERENCES (NOT THE //
* 1X,' ABSOLUTE DIFFERENCES) FOLLOW IN RANK ORDER:')
104  FORMAT(1X,5F10.3)
95   RETURN
      END

```

```

C -/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C   SUBROUTINE MFCIA (NTREAT,LIST,LPTR,NIP,IP,NOF)
C   THIS SUBROUTINE FINDS THE MEDIAN VALUE OF A NUMBER OF
C   VARIABLES/TREATMENTS, FINDS CELL COUNT OF #ABOVE/BELOW
C   GRAND MEDIAN IN EACH TREAT, AND PERFORMS A R*C CONTINGENCY
C   TEST ON THAT CELL COUNT MATRIX, USING FORM P.172 CONNVER
C   IN ORDER TO ALLOW GROUPING OF VARIABLES INTO AGGREGATE
C   VARIABLES; THE AGVARI VECTOR CONTAINS THE INDEX OF AGVAR
C   FOR EACH VAR (E.G., VAR 1&3 INTO AGVAR 1, VAR 2,4,5 INTO
C   AGVAR 2; THE AGVARI IS 1,2,1,2,2). THIS INFORMATION IS IN
C   CONTAINED IN THE IP VECTOR. NAGVAR IS THE NUMBER OF
C   THE NUMBER OF AGGREGATE VARIABLES(E.G., HERE, 2)
C   IF AGGREGATE VARIABLES ARE NOT USED, NIP=NUMBER OF INTEGER
C   VARIABLES=0, NAGVAR=NTREAT= NUMBER
C   OF VARIABLES IN DATA SET; AGVARI IS 1,2,...,NTREAT
C   NAV(I) IS THE NUMBER OF ELEMENTS IN AGVAR(I)
C   NA(I) IS # ELMS ABOVE MEDIAN IN AGVAR(I);NB(I)=#BELOW
C   Q50 IS GRAND MED;
      INTEGER MEDI,MED2,NB(15),NA(15),LL,UL,DF,NAV(17),
* LPTR(17),KEY(400),AGVARI(17),NAGVAR,IP(17),NIP
* A,B,N,NTREAT,NN,AN,AB
      REAL LIST(400),V,NUM,T,CONST,SVAL(400),Q50
      A=0
      B=0
      T=0
      IF (NIP.GT.0) GO TO 3
      NAGVAR=NTREAT
      DO 2 I=1,NTREAT
      AGVARI(I)=I
      NAV(I)=0
2     CONTINUE
      GO TO 1C
3     CONTINUE
      NAGVAR=0
      DO 4 I=1,NTREAT
      AGVARI(I)=IP(I)
      IF (IF(I).GT.NAGVAR) NAGVAR=IP(I)
      NAV(I)=0
4     CONTINUE
10    DO 15 I=1,NAGVAR
      NB(I)=0
      NA(I)=0
15    CONTINUE
      N=LPTR((NTREAT+1))-1
      MEDI=IFIX(.5*FLOAT(N+1))
      MED2=IFIX(.5*FLOAT(N+2))
      DO 18 I=1,N
      KEY(I)=I
      SVAL(I)=LIST(I)
18    CONTINUE

```

```

CALL SHSCRT(SVAL,KEY,N)
Q50=0.5*(SVAL(MED1)+SVAL(MED2))
C111 WRITE(NOF,111) MED1,MED2,SVAL(MED1),SVAL(MED2),Q50
      FORMAT(1X,'MED1,MED2',2I5,' SVAL(1,2),MEDIAN',2F10.3)
      DO 30 I=1,NTREAT
         LL=LPTR(I)
         UL=LPTR((I+1))-1
         K=AGVARI(I)
         NAV(K)=NAV(K)+UL-LL+1
         DO 20 J=LL,UL
            V=LIST(J)
            IF(V.GT.Q50) NA(K)=NA(K)+1
            IF(V.LE.Q50) NB(K)=NB(K)+1
      20 CONTINUE
      30 CONTINUE
      DO 35 I=1,NAGVAR
         A=A+NA(I)
         B=B+NB(I)
      35 CONTINUE
      NN=N*N
      AB=A*B
      CONST=FLCAT(NN)/FLOAT(AB)
      DO 50 I=1,NAGVAR
         IF (NAV(I).EQ.0) GO TO 50
         AN=A*NAV(I)
         NUM=(FLCAT(NA(I)))-FLOAT(AN)/FLOAT(N)**2
         T=T+CONST*NUM/FLOAT(NAV(I))
      50 CONTINUE
      DF=NAGVAR-1
      101 WRITE(NOF,101) Q50,NAGVAR
         FORMAT(1X,'THE MEDIAN VALUE OF THE TEST IS',F10.3/1X,
          * 'THE CELL COUNTS OF THE',I3,
          * 'TREATMENTS OR AGGREGATES ARE:')
      102 WRITE(NCF,102) (I,I=1,NAGVAR)
         FORMAT(13X,16I4)
      104 WRITE(NOF,104) (NA(I),I=1,NAGVAR),A
         FORMAT(1X,'> MEDIAN',16I4)
      105 WRITE(NOF,105) (NB(I),I=1,NAGVAR),B
         WRITE(NOF,105) (NAV(I),I=1,NAGVAR),N
      103 FORMAT(1X,'=< MEDIAN',16I4)
      105 FORMAT(1X,' TCTALS',16I4)
      106 WRITE(NCF,106) T,DF
         FORMAT(1X/1X,'CHI-SQUARED VALUE',F10.3/
          * 1X,' DEGREES OF FREEDOM',I5)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
SUBROUTINE KRLWAL(LIST,LPTR,IP,RP,NOF,NTREAT,TREAT)
C THIS SUMS THE RANKS OF EACH TREATMENT AND
C THE SQUARE OF ALL RANKS TO FIND THE OVERALL CHI SQUARE
C STATISTIC FOR THE NULL HYPOTHESIS THAT ALL TREATMENTS
C HAVE AN EQUAL MEAN. IF REJECT THAT NULL, THEN THE
C ADDITIONAL NUMBERS COMPUTED HAVE SOME MEANING
REAL TERM2,R,SSR,SSQ,T,LIST(400),RP(50),RANK(400),
* SRIDNI,6(20C),FN(17),SVALS(400),RNI(15)
* INTEGER LPTR(17),IP(50),NOF,TREAT(15),NTREAT,NJ(17),
* N,UL,LL,KEY(400),DF1,DF2
N=LPTR((NTREAT+1))-1
DF1=N-1
CF2=N-NTREAT
CALL RANGUE(N,LIST,SVALS,RANK,Q,KEY,NTREAT,0,1)
DO 10 I=1,NTREAT
   RNI(I)=0.0
10 CONTINUE
SSR=0.0
SRIDNI=0.0
DO 20 I=1,NTREAT

```

```

LL=LPTR(I)
UL=LPTR((I+1))-1
NJ(I)=UL+1-LL
FN(I)=FLOAT(NJ(I))
DG 30 J=LL,UL
      R=RANK(J)
      RNI(I)=RNI(I)+R
      SSR=SSR+R*R
30  CONTINUE
      SRIDNI=SRIDNI+(RNI(I)*RNI(I))/FLOAT(NJ(I))
20  CONTINUE
      TERM2=FLCAT(N*(N+1)*(N+1))/4.0
      SSQ=(SSR-TERM2)/FLOAT(N-1)
      T=(SRIDNI-TERM2)/SSQ
101  WRITE(NCF,101) N,NTREAT,SSR
      FORMAT(10X,'**KRUSKAL-WALLIS TEST**',1X,' IN SET OF',
* 15,' DATA ELEMENTS, DIVIDED INTO',15,' SAMPLES',1X,
* 'THE SUM OF SQUARED RANKS WAS',F10.3,
* 'THE SUMS OF RANKS/SAMPLE:' /
*/1X,' VAR NR',5X,' NR DATA ELEM',5X,' SUM OF RANKS')
      CO 40 I=1,NTREAT
      WRITE(NCF,102) TREAT(I),NJ(I),RNI(I)
102  FORMAT(1X,15,EX,15,12X,F8.2)
40  CONTINUE
      WRITE(NOF,103) T,DF1,DF2
103  FORMAT(1X/1X,'YIELD A CHI SQUARE VALUE OF:',F10.3/1X,
* 'WITH',15,' DEGREES OF FREEDOM. IF AND ONLY IF',1X,
* 'NULL HYPOTHESIS REJECTED, FOL NUMBERS CAN',1X,
* 'TEST WHETHER SPECIFIC PAIRS OF VARIABLES HAVE',1X,
* 'THE SAME MEAN. IF ABS DIFFERENCE IN MEAN RANKS',1X,
* '|R(I)/N(I)-R(J)/N(J)| > POOLED STANDARD DEVIATION',
* 1X,' TIMES THE T QUANTILE 1-ALPHA/2 THEN REJECT THE',
* 1X,' HYPOTHESIS THAT THE PAIR OF MEANS ARE EQUAL',1X,
* ' (DEGREES OF FREEDOM HERE:',14,')',1X,1X,
* ' J ABS CIF MEAN RANK POOLED STANDARD DEVIATION')
      CONST=(SSQ*(FLOAT(N)-1.0-T)/FLOAT(DF2))*0.5
      NTM1=NTREAT-1
      CO 60 I=1,NTM1
          IP1=I+1
          DO 70 J=IP1,NTREAT
              ACMR=ABS((RNI(I)/FN(I))- (RNI(J)/FN(J)))
              SDPOOL=CONST*((1.0/FN(I))+(1.0/FN(J)))
              WRITE(NOF,104) I,J,ACMR,SDPOOL
104  FORMAT(2X,2I3,6X,F8.3,14X,F10.3)
70  CONTINUE
60  CONTINUE
      RETURN
      END
C-----
SUBROUTINE SQLARK (LIST,LPTR,NTREAT,TREAT,IP,RP,NOF)
C THIS TESTS WHETHER A DATA SET HAS EQUAL VARIANCE AMONG
C ALL VARIABLES BY (1) FINDING THE ABSOLUTE DEVIATIONS OF
C EACH VARIABLE'S ELEMENTS, IN SUBROUTINE ADEV, (2) RANKING
C THOSE DEVIATIONS (RANCUE), AND (3) SUMMING, SQUARING, ETC.,
C THOSE RANKS AS REQUIRED TO FIND THE TEST STATISTIC
REAL LIST(400),DEV(400),RP(50),FN,FM,FNT,R(400),
* SUMR J(17),FNJ(17),R2MU,RMU,T1,T2,NUM,RSQ,SSR2J,
* SUMR2J(17)
INTEGER LPTR(17),NTREAT,TREAT(17),IP(50),NOF,KEY(400)
NT=L PTR((NTREAT+1))-1
FNT=FLOAT(NT)
CALL ADEV(LIST,LPTR,NTREAT,DEV)
CO 3 I=1,NT
      KEY(I)=I
3  CONTINUE
CALL RNQ(DEV,KEY,NT,R)
CO 5 J=1,NTREAT

```

```

      SUMR2J(J)=C
      SUMRJ(J)=0
5  CONTINUE
      SUMR=0.0
      SUMR2=0.0
      SUMR4=0.0
      J=1
      CO 10 I=1,NT
          RSQ=R(I)*R(I)
          SUMR=SUMR+R(I)
          SUMR2=SUMR2+RSQ
          SUMR4=SUMR4+RSC*R SQ
          INCRJ=L PTR((J+1))
          IF(I.GE.INCRJ) J=J+1
          SUMRJ(J)=SUMRJ(J)+R(I)
          SUMR2J(J)=SUMR2J(J)+RSQ
10  CONTINUE
      R2MU=SUMR2/FNT
      RMU=SUMR/FNT
      IF (NTREAT.GT.2) GO TO 40
      FN=FLOAT(L PTR(2)-1)
      FM=FLCAT(L PTR(3)-L PTR(2))
      DTERM2=FN*FM/(FNT-1.0)
      DTERM1=CTE FM2/FNT
      DENOM=(DTERM1*SUMR4-DTERM2*R2MU*R2MU)*#.5
      NUM=SUMR2J(1)-FN*R2MU
      T1=NUM/DENOM
101  WRITE(9,101) DTERM1,DTERM2,SUMR4,R2MU,SUMR2J(1)
      FORMAT(1X,'DTERM1-2,SUMR4,R2MU,SUMJ2.1',5F5.4)
102  WRITE(NCF,102) T1
      FCRMAT(10X,'**** SQUARED RANK TEST FOR ;
* ' EQUAL VARIANCE ****'/1X,'TEST STATISTIC ;
* ' FOR THIS TWO SAMPLE DATA SET IS',F10.4)
40  GO TO 95
      CONTINUE
      DSQ=(SUMR4-FNT*R2MU*R2MU)/(FNT-1.0)
      SSJMU=C.0
      DC 50 J=1,NTREAT
          SSSR2J=SUMR2J(J)*SUMR2J(J)
          SSJMU=SSJMU+SSSR2J/FLOAT(L PTR((J+1))-L PTR(J))
50  CONTINUE
      T2=(SSJMU-FNT*R2MU*R2MU)/DSQ
103  WRITE(9,103) SSJMU,R2MU,SUMR4,DSQ
      FORMAT(1X,'SSJMU,R2MU,SUMR4,DSQ',2F10.4,F20.4,F10.4)
104  WRITE(NOF,104) NTREAT, T2
      FORMAT(1CX,
* '**** SQUARED RANK TEST FOR EQUAL VARIANCE ****'/
* ' IN A DATA SET OF ',I3,' VARIABLES'/1X,
* ' THE TEST STATISTIC:',F10.4)
95  RETURN
      END
-----
SUBROUTINE ADEV(LIST,LPTR,NTREAT,DEV)
REAL LIST(400),RP(50),DEV(400),VARX,MU,X(200)
INTEGER LPTR(17),K,NTREAT,IP(50),NOF,LL,IP1,NI
CO 10 I=1,NTREAT
    IP1=I+1
    LL=L PTR(I)
    NI=L PTR(IP1)-LL
    DO 12 K=1,NI
12  CONTINUE
        X(K)=LIST((LL+K-1))
        CALL MLVAR(X,NI,MU,VARX)
    DO 30 K=1,NI
30  CONTINUE
        J=LL+K-1
        DEV(J)=ABS(LIST(J)-MU)

```

```

      LI=LPTR(I)
      WRITE(9,105) I,LI,LIST(LI),MU,DEV(LI)
105  FORMAT(1X,'VAR#',LPTR;LIST,MU,DEV OF LPTR:',2I3,3F10.4)
10   CONTINUE
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C   SUBROUTINE HARTLE(LIST,LPTR,NTREAT,TREAT,IP,RP,NOF)
C   THIS SUBROUTINE FINDS THE STATISTIC FOR THE HARTLEY'S
C   TEST FOR EQUAL VARIANCE, AN F RATIO BETWEEN THE SMALLEST
C   AND LARGEST VARIANCE IN A SET OF DATA.
      REAL LIST(400),RP(50),X(200),VMIN,VMAX,VARX,FRATIO,MU
      INTEGER LPTR(17),NTREAT,TREAT(16),IP(50),NOF,LL,UL,
      *IP1,NI
      VMIN=10)CC00.C
      VMAX=-10.C
      DO 10 I=1,NTREAT
      IP1=I+1
      LL=LPTR(I)
      NI=LPTR(IP1)-LL
      DO 12 K=1,NI
      X(K)=LIST((LL+K-1))
12   CONTINUE
      CALL MLVAR(X,NI,MU,VARX)
      IF(VARX.GE.VMIN) GO TO 20
      VMIN=VARX
      IVMIN=TREAT(I)
20   IF(VARX.LE.VMAX) GO TO 10
      VMAX=VARX
      IVMAX=TREAT(I)
10   CONTINUE
      FRATIO=VMAX/VMIN
      WRITE(NOF,101) NTREAT,IVMIN,VMIN,IVMAX,VMAX,FRATIO
101  FORMAT(10X,
      *'***** HARTLEY TEST FOR EQUAL VARIANCE****'/2X,' IN',
      *' A DATA SET WHICH COMPARED ',I3,' VARIABLES, '/1X,
      *' VARIABLE',I3,' HAD THE SMALLEST VARIANCE:',F10.4,
      *', '/1X,' WHILE VARIABLE',I3,' HAD THE LARGEST:',F10.4,
      *'/1X,' WHICH GIVES AN F RATIO OF:',F10.4)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C   SUBROUTINE CORREL(LIST,LPTR,NTREAT,TREAT,NOF,IP,RP)
C   THIS SUBROUTINE TAKES LIST, COMPOSED OF TWO EQUAL LENGTH
C   VARIABLES, PERFORMS PEARSON'S RHO ON THEM, THEN RANKS
C   EACH VARIABLE, PERFORMS PEARSON'S RHO ON EACH TO YIELD
C   SPEARMAN'S RHO, AND IF REQUESTED, FIGURES KENDALL'S TAU
C   IP(1) IS OPTION: -1=SPEARMAN ONLY, 0=PEARSON&SPEARMAN,
C   1=PEARSON, SPEARMAN AND KENDALL
      REAL LIST(400),RP(50),SVALS(200),RANK(400),B1,B0,TAU,
      *TRANKS(200),TVALS(200),SRANK(200),TRANG(200)
      INTEGER LPTR(17),KEY(400),NTREAT,TREAT(17),NOF,
      *TKEY(200),SKEY(200),OPT,IP(50)
      NPAIR=LPTR(2)-1
      IF (IP(1).LT.C) GO TO 5
      WRITE(NOF,201)
201  FORMAT(10X,'**CORRELATION**')
      CALL RHOER(LIST,NPAIR,RHO,B1,B0,0)
      WRITE(NOF,101) RHO
101  FORMAT('PEARSON'S CORRELATION ON THE ',
      *' ACTUAL VALUES OF TWO VARS:'/10X,F10.3)
5   DO 10 I=1,NPAIR
      K=I+NPAIR
      SKEY(I)=I
      TKEY(I)=I
      SVALS(I)=LIST(I)
      TVALS(I)=LIST(K)

```



```

201  FORMAT(10X, 'NON-PARAMETRIC REGRESSION')
      EPSILN=3.0001
      OPTION=IF(1)
      DIP(1)=-1
      XBAR=0
      YBAR=0
      IX=3-OPTION
      IY=OPTION
      NPAIR=LPTR(2)-1
      CALL RHOER(LIST, NPAIR, RHO, B1, B0, OPTION)
      WRITE (NOF, 101) TREAT(IY), TREAT(IX), B1, B0
101  * WITH DEPENDENT VARIABLE AS VARIABLE #', I5/1X,
      * AND EXPLANATORY/INDEPENDENT VARIABLE AS #', I5/1X,
      * THE LEAST SQUARES FIT OF LINEAR REGRESSION IS: '/1X,
      * DEP VAR = ', F9.3, ' * INDEP VAR + ', F10.3)
      B1R=RP(1)
      IF (IP(2).LT.1) B1R=B1
      IXL=LPTR(IX)-1
      IYL=LPTR(IY)-1
      WRITE(9, 673) IXL, IYL
678  FORMAT(1X, ' X & Y INDICES:', 2I5)
      DO 20 I=1, NPAIR
          J=NPAIR+I
          X=LIST(IXL+I)
          UX(J)=X
          Y=LIST(IYL+I)-B1R*X
          UX(I)=Y
      WRITE(9, 987) LX(I), UX(J)
987  FORMAT(1X, 'X, RESID:', 2F10.4)
20  CONTINUE
      WRITE(NOF, 102) B1R
102  * TO TEST CORRELATION BETWEEN THE INDEPENDENT VARIABLE',
      * ' AND THE RESIDUALS OF THE FIT, '/
      * 1X, ' WHERE B1=', F10.4)
      CALL CORREL(UX, LPTR, NTREAT, TREAT, NOF, DIP, RP)
      NESTY=IP(3)
      IF(NESTY.EQ.0) GO TO 24
      DO 21 I=1, NPAIR
          XSUM=XSUM+LIST(IXL+I)
          YSUM=YSUM+LIST(IYL+I)
21  CONTINUE
      IF(IP(2).EQ.1) B0=(YSUM-B1*XSUM)/FLOAT(NPAIR)
      WRITE(NOF, 121) NESTY, B1
121  * ESTIMATE OF', I3, ' DEPENDENT VARIABLES BY ',
      * 'REGRESSION, '/1X, ' WITH B1=', F10.4)
      DO 22 I=1, NESTY
          EY=B1*RP((2+I))+B0
          WRITE(NCF, 122) RP((2+I)), EY
122  * INCEP VAR VALUE:', F10.4,
      * ' DEP VAR ESTIMATE:', F10.4)
22  CONTINUE
24  IF(IP(4).EQ.0) GO TO 95
      IS=0
      QALFD2=1.0-((1.0-RP(2))*0.5)
      N=NPAIR
      IF(N.GT.60)
          * W=QALFD2*((FLOAT(N*(N-1)*(2*N+5))/18.0)**0.5)
      IF(N.LE.60) W=RP(2)
      NPM1=NPAIR-1
      DO 30 I=1, NPM1
          LL=I+1
          CO 35 J=LL, NPAIR
              DENOM=LIST(IXL+I)-LIST(IXL+J)
              ADEM=ABS(DENOM)
              IF(ADEM.LT.EPSILN) GO TO 35
              IS=IS+1

```

S(IS)=(LIST(IYL+I)-LIST(IYL+J))/DENOM
KEY(IS)=IS

```
35          CCNTINUE
30          CONTINUE
          CALL SHSORT(S,KEY,IS)
          I17=17
          I47=47
          WRITE(9,876) S(I17),S(I47),IS
876          FCRMAT(IX,'SLOPES, #SLOPES',2F9.4,IS)
          NR=IFIX(0.5*(IS-W))
          NS=IFIX(0.5*(IS+W)+1.96)
          WRITE(NCF,103) W,NR,S(NR),NS,S(NS)
103          FCRMAT(IX,'BASED ON A VALUE OF W=',F10.4/IX,
          * 'INDICES OF CONFIDENCE INTERVAL, & VALUES OF SLOPE:' /
          * IX,'LOWER BOUND INDEX:',I3,' WITH VALUE:',F10.4/IX,
          * 'UPPER BOUND INDEX:',I3,' WITH VALLE:',F10.4)
95          RETURN
          END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C          SUBROUTINE MONORE(LIST,LPTR,IP,RP,TREAT,NCF)
C          THIS SUBROUTINE PERFORMS MONOTONE REGRESSION BASED ON THE
C          LEAST SQUARE FIT OF THE RANKS OF 2 VARIABLES. IF THE
C          USER DESIRES, ESTIMATES OF SPECIFIC VALUES OF DEPENDENT
C          VARIABLE Y BASED ON VALUES OF X CONTAINED IN RP VECTOR.
C          ALSO, IF THE USER DESIRES, A TRACE OF THE REGRESSION WILL
C          BE LISTED.
C          IP(1): # OF DEPENDENT VARIABLE (1 OR 2)
C          IP(2): # OF SPECIFIED X VALUES FOR ESTIMATION OF Y VALUES
C          IP(3): DO YOU WANT A TRACE? 1=YES 0=NO
          REAL LIST(400),X(200),Y(200),RX(200),RY(200),TERM2,
          * B2,A2, FN,RP(50),SXX,SXY
          INTEGER KEYX(200),KEYY(200),NPAIR,IX,IY,IP(50),
          * LPTR(17),TREAT(17)
          NPAIR=LPTR(2)-1
          FN=FLOAT(NPAIR)
          IX=(2-IP(1))*NPAIR
          IY=(IP(1)-1)*NPAIR
          WRITE(9,210) IX,IY
210          FORMAT(IX,' INTO MONCRE; INDEX OF FIRST X,Y:',2I5)
          DO 10 I=1,NPAIR
              X(I)=LIST(I+IX)
              Y(I)=LIST(I+IY)
              KEYX(I)=I
              KEYY(I)=I
10          CONTINUE
          CALL RNQ(X,KEYX,NPAIR,RX)
          CALL RNQ(Y,KEYY,NPAIR,RY)
          SXY=0.0
          SXX=0.0
          DO 15 I=1,NPAIR
              X(I)=LIST(I+IX)
              Y(I)=LIST(I+IY)
15          CONTINUE
          DO 20 I=1,NPAIR
              SXY=SXY+RX(I)*RY(I)
              SXX=SXX+RX(I)*RX(I)
20          CONTINUE
          TERM2=FN*((FN+1.0)**2)/4.0
          B2=(SXY-TERM2)/(SXX-TERM2)
          A2=(1.0-B2)*(FN+1.0)/2.0
          WRITE(NCF,101) TREAT(IP(1)),TREAT((3-IP(1))),B2,A2
101          FCRMAT(IX,'**MONOTONE REGRESSION**'/IX,' USING ',
          * 'LEAST SQUARES REGRESSION ON THE RANKS OF VARIABLES' /
          * IX,' WITH VARIABLE #',I3,' AS DEPENDENT VARIABLE Y' /
          * IX,' AND VARIABLE #',I3,' AS INDEPENDENT VARIABLE X' /
          * IX,' R(Y) = ',F9.3,' * R(X) + ',F10.3)
          NEST=IP(2)
```



```

N=NP AIR
IF (NEST.GT.0)
* CALL ESTY(A2,B2,X,Y,KEYX,KEYY,RX,RY,N,NEST,RP,NDF)
IF (IP(3).EQ.1)
* CALL TRACE(A2,B2,X,Y,RX,RY,KEYX,KEYY,N,NDF)
RETURN
END

```

```

C-----
SUBROUTINE ESTY
* (A2,B2,X,Y,KEYX,KEYY,RX,RY,N,NEST,RP,NDF)
C THIS FINDS THE ESTIMATES OF THE DEPENDENT VAR Y
C FOR VALUES OF THE INDEPENDENT VAR X
REAL A2,B2,X(200),Y(200),RX(200),RY(200),RP(50)
INTEGER KEYX(200),KEYY(200),KEY(200),NEST,N,XSTART,
* YSTART
IF (NEST.EQ.0) GO TO 95
XSTART=0
DO 5 I=1,NEST
KEY(I)=I
5 CONTINUE
CALL SHSCR1(RP,KEY,NEST)
WRITE(9,211) RP(1),KEY(1),NEST
211 FORMAT(1X,'INTO ESTY: RP(1),KEY(1),NEST: .. 10.3,215)
DO 10 I=1,NEST
VALX=RP(I)
CALL VTCR(X,RX,KEYX,VALX,RNKX,N,XSTART)
IF (RNKX.GE.1) GO TO 15
WRITE(NDF,101) I,VALX,X(KEYX(1)),X(KEYX(N))
101 FORMAT(1X,'EST #',I3,' NOT POSSIBLE; X=',F10.4/
* 9X,' FALLS OUTSIDE RANGE OF',F10.4,' TO',F10.4)
GO TO 10
15 RNKY=A2+B2*RNKX
YSTART=0
CALL RTCV(Y,RY,KEYY,VALY,RNKY,N,YSTART)
WRITE(NDF,102) I,VALX,VALY
102 FORMAT(1X,'EST #',I3,
* ': X=',F10.4,' => E(Y|X) OF',F10.4)
10 CONTINUE
95 RETURN
END

```

```

C-----
SUBROUTINE VTCR(Z,RZ,KEYZ,VALZ,RNKZ,N,ZSTART)
C HAS A VALUE, FINDS ITS RANK
REAL Z(200),RZ(200),RNKZ,VALZ
INTEGER KEYZ(200),N,ZSTART
RNKZ=-1
IF ((VALZ.LT.Z(KEYZ(1))).OR.(VALZ.GT.Z(KEYZ(N))))
* GO TO 95
CALL INTEFF(Z,RZ,KEYZ,VALZ,RNKZ,N,ZSTART)
GO TO 97
95 WRITE(9,101) VALZ,Z(KEYZ(1)),Z(KEYZ(N))
101 FORMAT(1X,'OUT OF RANGE:VAL,LL,UL:',3F10.4)
97 RETURN
END

```

```

C-----
SUBROUTINE RTCV(Z,RZ,KEYZ,VALZ,RNKZ,N,ZSTART)
C HAS A RANK, FINDS ITS VALUE
REAL Z(200),RZ(200),RNKZ,VALZ
INTEGER KEYZ(200),N,ZSTART
IF ((RNKZ.GT.RZ(KEYZ(1))).AND.(RNKZ.LT.RZ(KEYZ(N))))
* GO TO 50
IF (RNKZ.LE.RZ(KEYZ(1))) VALZ=Z(KEYZ(1))
IF (RNKZ.GE.RZ(KEYZ(N))) VALZ=Z(KEYZ(N))
GO TO 55
50 CALL INTERP(RZ,Z,KEYZ,RNKZ,VALZ,N,ZSTART)
95 RETURN
END

```

```

C-----
C SUBROUTINE INTERP(A,B,KEY,AVAL,BVAL,N,ISTART)
C THIS SUBROUTINE DOES LINEAR INTERPOLATION: KEY IS THE
C ORDER OF VALS A AND B. AVAL IS KNOWN, FIND BVAL. ISTART
C IS THE INDEX OF KEY AT WHICH TO START LOOKING: WITH PRIOR
C KNOWLEDGE, SUCH AS IF AVAL'S WERE SORTED, ISTART MAY BE
C GREATER THAN C
REAL A(200),B(200),AVAL,BVAL,PROP,BPRCP
INTEGER I,ISTART,N,KL,KU,KEY(200)
I=ISTART
10 CONTINUE
I=I+1
KU=KEY(I)
IF((AVAL.GT.A(KU)).AND.(I.LT.N)) GO TO 10
KL=KEY(I-1)
PROP=(AVAL-A(KL))/(A(KU)-A(KL))
BPROP=PRCF*(B(KU)-B(KL))
BVAL=B(KL)+BPROP
ISTART=I-1
RETURN
END

```

```

C-----
C SUBROUTINE TRACE(A2,B2,X,Y,RX,RY,KEYX,KEYY,N,NOF)
C THIS SUBROUTINE FINDS THE TRACE OF THE REGRESSION
C IN GENERAL BY TAKING THE ACTUAL RANK OF Y, FINDING
C THE EXPECTED RANK OF X AND THE CORRESPONDING VALUE OF X,
C AND THEN PLOTTING THAT X WITH THE ACTUAL VALUE OF Y.
REAL EX(200),EY(200),X(200),Y(200),RX(200),RY(200),
*RYB,A2,B2,RYS
INTEGER KEYX(200),KEYY(200),N,NOF
K=0
WRITE(NOF,101)
101 FORMAT(10X,' TRACE OF THE MONOTONE REGRESSION */1X,
* X Y
RYS=A2+B2*(RX(KEYX(1)))
IF(RYX.LT.RY(KEYY(1))) GO TO 5
K=K+1
EX(K)=X(KEYX(1))
YSTART=0
CALL RTCV (Y,RY,KEYY,YY,RYS,N,YSTART)
EY(K)=YY
WRITE(NCF,102) EX(K),EY(K)
5 CO 10 I=1,N
J=KEYY(I)
ERX=(RY(J)-A2)/B2
XSTART=0
CALL RTCV (X,RX,KEYX,XX,ERX,N,XSTART)
K=K+1
EX(K)=XX
EY(K)=Y(J)
WRITE(NCF,102) EX(K),EY(K)
10 CONTINUE
RYB=A2+B2*(RX(KEYX(N)))
IF(RYB.GT.RY(KEYY(N))) GO TO 20
K=K+1
EX(K)=X(KEYX(N))
YSTART=0
CALL RTCV (Y,RY,KEYY,YY,RYB,N,YSTART)
EY(K)=YY
WRITE(NCF,102) EX(K),EY(K)
102 FORMAT(1X,2F10.4)
20 CALL PLOT1(EX,EY,N,1)
RETURN
END

```

```

C-----
C SUBROUTINE KCLMOG (LIST,LPTR,TREAT,IP,RP,NOF)
C THIS SUBROUTINE FINDS THE TEST STATISTIC FOR THE

```

```

C KOLMCGORCV STATISTIC FOR SAMPLES OF UNDER 31 ELEMENTS.
C DEPENDING ON NULL HYP, THE TEST STATISTIC IS THE LARGEST,
C SMALLEST OR GREATEST ABSOLUTE DIFFERENCE F(X)-S(X),
C WHERE F IS THE HYPOTHESISED THEORETICAL DIST AND
C S IS THE EMPIRICAL CDF. IP(1) = CHOICE OF DISTR;
C IP(2) IS THE CHOICE OF HYPOTHESIS
C SEE THE DISTRIBUTION SUBROUTINES FOR MEANINGS OF THE
C OTHER PARAMETERS RP AND IP.

```

```

REAL LIST(400),S(400),F(400),RP(50),FN,TSTAT
INTEGER LPTR(17),IP(50),NOF,IDIIST,HYP,KEY(400)
WRITE(NOF,101)
101 FORMAT(1CX,'**** KOLMCGORCV TEST ****')
FYP=IP(2)
N=LPTR(2)-1
FN=FLOAT(N)
IDIIST=IP(1)
DO 10 I=1,N
  S(I)=FLCAT(I)/FN
  KEY(I)=I
10 CONTINUE
CALL SHSORT(LIST,KEY,N)
IF(IDIIST.EQ.1) CALL NCRMF(LIST,N,RP,F,NOF)
IF(IDIIST.EQ.2) CALL EXPF(LIST,N,RP,F,NOF)
IF(IDIIST.EQ.3) CALL UNIF(LIST,N,RP,F,NOF)
IF(IDIIST.EQ.4) CALL ERLF(LIST,N,RP,IP,F,NOF)
IF(IDIIST.EQ.5) CALL WEIBF(LIST,N,RP,F,NOF)
CALL FINCT(S,F,N,TSUP,TMIN,TMAX,NOF)
IF(HYP.EQ.1) CALL THYPA(TSUP,NOF)
IF(HYP.EQ.2) CALL THYPB(TMAX,NOF)
IF(HYP.EQ.3) CALL THYPC(TMIN,NOF)
RETURN
END

```

```

-----
C SUBROUTINE ERLF(LIST,N,RP,IP,F,NOF)
C THIS FINDS THE THEORETICAL ERLANG DISTRIBUTION F, WHERE
C NN=IP(1)=SHAPE PARAMETER, AND L=RP(1)=SCALE PARAMETER
REAL LX,LXN,L,X,LIST(400),RP(50),F(400),DN
INTEGER IP(50),I,N,FACT,NNM1
NN=IP(1)
NNM1=NN-1
DO 30 I=1,N
  FACT=1
  X=LIST(I)
  LX=L*X
  LXN=0
  DO 23 J=1,NNM1
    FACT=FACT*J
    LXN=LXN+(LX**J)/FACT
23 CONTINUE
  IF (X.LE.0) F(I)=0
  IF (X.GT.0) F(I)=1.0-(1.0+LXN)*EXP(-LX)
30 CONTINUE
WRITE(NOF,101) NN,L
101 FORMAT(1X,'WITH HYPOTHESISED DISTRIBUTUION ERLANG'/1X,
* 'WITH PARAMETERS N=',I5,' AND LAMBDA=',F10.4)
RETURN
END

```

```

-----
C SUBROUTINE WEIBF(LIST,N,RP,F,NOF)
C THIS FINDS THE THEORETICAL WEIBULL DISTRIBUTION F, WHERE
C A=RP(1)=SHAPE PARAMETER, AND L=RP(2)=SCALE PARAMETER
C LAMBDA
REAL A,L,X,LIST(400),RP(50),F(400)
INTEGER I
A=RP(1)
L=RP(2)
DO 10 I=1,N

```

```

      X=LIST(I)
      IF(X.LE.0) F(I)=0
      IF(X.GT.0) F(I)=1.0-EXP(-((L*X)**A))
10   CONTINUE
      WRITE(NOF,101) A,L
101  FORMAT(1X,'WITH HYPOTHESED DISTRIBUTION WEIEULL'/1X,
* 'WITH PARAMETERS ALPHA=',F10.4,' AND LAMBOA=',F10.4)
      RETURN
      END
C-----
      SUBROUTINE UNIF(LIST,N,RP,F,NOF)
C THIS SUBROUTINE FINDS THE THEORETICAL UNIFORM DISTRIBUTION
C F, WHERE A=RP(1)=LOWER BOUND, B=RP(2)=UPPER BOUND
      REAL C,A,B,X,LIST(400),RP(50),F(400)
      INTEGER I
      A=RP(1)
      B=RP(2)
      C=B-A
      DO 10 I=1,N
      X=LIST(I)
      IF(X.LE.A) F(I)=0
      IF(X.GE.B) F(I)=1
      IF((X.GT.A).AND.(X.LT.B)) F(I)=(X-A)/C
10   CONTINUE
      WRITE(NOF,101) A,B
101  FORMAT(1X,'WITH HYPOTHESED DISTRIBUTION UNIFORM',
* '/1X,'WITH PARAMETERS LOWER LIMIT=',F10.4,
* ' AND UPPER LIMIT=',F10.4)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
C      SUBROUTINE LILIEF(LIST,LPTR,TREAT,IP,RP,NOF)
C LILIEFORS TEST FOR SAMPLES OF UNDER 31 ELEMENTS.
C DEPENDING ON HYPOTHESIS; TEST STATISTIC IS THE LARGEST
C SMALLEST OR GREATEST ABSOLUTE DIFFERENCE F(X)-S(X),
C WHERE F IS THE HYPOTHESED THEORETICAL DISTRIBUTION AND
C S IS THE EMPIRICAL CDF. IP(1) IS THE CHOICE OF
C DISTRIBUTIONS; IP(2) IS CHOICE OF HYPOTHESIS. PARAMETERS
C ARE ESTIMATED: RP(1)=SAMPLE MEAN,RP(2)=VARIANCE
C SEE THE DISTRIBUTION SUBROUTINES FOR THE MEANINGS OF THE
C OTHER PARAMETERS RP AND IP.
      REAL LIST(400),S(400),F(400),RP(50),FN,TSTAT,MU,
* TMIN,TMAX,VAR,TSUP
      INTEGER LPTR(17),IP(50),NOF,IDIST,HYP,KEY(400)
      HYP=IP(2)
      N=LPTR(2)-1
      FN=FLCAT(N)
      CALL MUVAF(LIST,N,MU,VAR)
      WRITE(NOF,101) MU,VAR
101  FORMAT(10X,'*** LILLIEFORS TEST ***'/1X,
* 'WITH COMPUTED SAMPLE MEAN=',F10.4,
* ', AND VAR=',F10.4)
      RP(1)=MU
      RP(2)=VAR
      IDIST=IP(1)
      IF(IDIST.EQ.2) RP(1)=1.0/MU
      DO 10 I=1,N
      S(I)=FLCAT(I)/FN
      KEY(I)=I
10   CONTINUE
      CALL SHSCRT(LIST,KEY,N)
      IF(IDIST.EQ.1) CALL NCRMF(LIST,N,RP,F,NOF)
      IF(IDIST.EQ.2) CALL EXPF(LIST,N,RP,F,NOF)
      CALL FINCT(S,F,N,TSUP,TMIN,TMAX,NOF)
      IF(HYP.EQ.1) CALL THYPA(TSUP,NOF)
      IF(HYP.EQ.2) CALL THYPB(TMAX,NOF)

```

```

IF (HYP.EC.3) CALL THYPC(TMIN,NOF)
RETURN
END

```

```

C-----
C SUBROUTINE WILKSH (LIST,LPTR,TREAT,IP,RP,NOF)
C THIS SUBROUTINE FINDS THE TEST STATISTIC FOR THE
C SHAPIRO-WILK TEST FOR NORMALITY, N < 31 ELEMENTS.
C IP(1) = NUMBER OF COMPARISONS = TRUNC(# DATA ELEMS/2)
C RP(1),...RP(K) = WEIGHTS TO SCALE COMPARISONS
C THE DENOMINATOR IS THE SUM OF SQUARED DEVIATIONS OF X; AND
C HERE IT IS FOUND BY SXX = (N-1)*VAR(X)
C THE NUMERATOR IS SQUARE OF SUMS OF COMPARISONS OF VALUES
C OF EQUAL DEPTH FROM END POINTS, E.G., WITH N DATA ELMS,
C X(N)-X(1), X(N-1)-X(2); A(I) IS MULTIPLIED TIMES THE
C DIFFERENCE
REAL LIST(400),S(400),F(400),RP(50),FN,TSTAT,MU,VAR,
*NUM,DENOM,MU,VAR,TSUP,SUMWT
INTEGER LPTR(17),IP(50),NOF,IDIST,HYP,K,KEY(400)
N=LPTR(2)-1
CALL MUVAF(LIST,N,MU,VAR)
DENOM=FLCAT(N-1)*VAR
DO 5 I=1,N
  KEY(I)=I
5 CONTINUE
CALL SHSORT(LIST,KEY,N)
K=IP(1)
SUMWT=0.0
DO 10 I=1,K
  TEMP=RP(I)*(LIST(N-I+1)-LIST(I))
  WRITE(6,210) RP(I),LIST(N-I+1),LIST(I),TEMP
210 FORMAT(1X,'RP',LISTHI,LISTLO,TEMP',4F10.4)
  SUMWT=SUMWT+TEMP
10 CONTINUE
NUM=SUMWT*SUMWT
T3=NUM/DENOM
WRITE(NOF,101) N,MU,VAR,T3,NUM,DENOM
101 FORMAT(1X,'**** SHAPIRO-WILK TEST FOR NORMALITY ****'
*/1X,'WITH A SAMPLE SIZE OF',I5/1X,'WITH COMPUTED ',
* 'SAMPLE MEAN=',F10.4,' AND VAR=',F10.4/1X,
* 'THE TEST STATISTIC=',F10.4/1X,
* 'FROM NUMERATOR',F10.4,' AND DENOMINATOR=',F10.3)
RETURN
END

```

```

C-----
C SUBROUTINE SMIRN2 (LIST,LPTR,TREAT,IP,RP,NOF)
C THE SUBROUTINE TAKES ELEMENTS OF TWO VARIABLES, SORTS
C THEM, AND COMPARES THEIR EMPIRICAL CDF'S (FOUND BY
C DIVIDING THE RANK MINUS BY THE TOTAL NUMBER OF ELEMENTS)
C I,M REFER TO FIRST VAR X; J,N REFER TO SECOND VAR Y
C K IS COUNTER FOR DIFFERENCE VECTOR S, L=1 MEANS LAST
C PASS IP(1) IS THE INDEX OF HYPOTHESIS TYPE: 1=H0: NO DIF
C 2=H0: F(X) <= G(Y), 3=H0: F(X) > G(Y)
REAL EPSILN, FN, FM, X(100), Y(100), DIF, ADIF, S(200), TMIN,
* TMAX, ATMAX, ATMIN, TSUP, PDIF, LIST(400), RP(50), TSTAT
INTEGER I,J,K,L,M,N,KEYI(100),KEYJ(100),JJ,IF(50),
* LPTR(17),NOF,HYP,II,TREAT(17)
CALL FINCS(LIST,LPTR,S,K)
TMIN=2.0
TMAX=-2.0
HYP=IP(1)
DO 35 I=1,K
  IF (S(I).GT.TMAX) TMAX=S(I)
  IF (S(I).LT.TMIN) TMIN=S(I)
35 CONTINUE
ATMIN=ABS(TMIN)
ATMAX=ABS(TMAX)
TSUP=ATMIN

```

```

IF (ATMAX.GT.TSUP) T SUP=ATMAX
WRITE (NCF,102) TREAT(1),TREAT(2),TMIN,TMAX,T SUP
102 FORMAT(10X,'***** KOLMOGOROV TEST *****'/1X,
* WITH DATA SET VAR #',I3,' AS VARIABLE X '/6X,
* AND VAR #',I3,' AS VARIABLE Y'/1X,
* THE MAX DIFFERENCE F(X)<G(Y):',F10.4/1X,
* THE MAX DIFFERENCE F(X)>G(Y):',F10.4/1X,
* THE OVERALL GREATEST DIFFERENCE:',F10.4)
IF (HYP.EQ.1) TSTAT=T SUP
IF (HYP.EQ.2) TSTAT=T MAX
IF (HYP.EQ.3) TSTAT=T MIN
WRITE (NOF,103) TSTAT
103 FORMAT(1X/1X,'TEST STATISTIC:',F10.4)
RETURN
END

```

```

C-----
C SUBROUTINE CRVONM(LIST,LPTR,TREAT,IP,RP,NCF)
C THIS SUBROUTINE FINDS THE TEST STATISTIC T2 OF THE
C CRAMER-VON MISES TEST, AND DOES LINEAR INTERPOLATION
C TO ARRIVE AT THE ALPHA HAT, ALPHAT.
REAL EPSILN, FN, FM, S(400), SSUM, L SIG, ALPHAT, ALPHYP,
* T2, ATMAX, ATMIN, T SUP, PDIF, LIST(400), RP(50), CCNST
INTEGER J, K, L, M, N, KEYI(100), KEYJ(100), II, JJ, IF(50),
* LPTR(17), NOF, I, TREAT(17)
CALL FINCS(LIST,LPTR,S,K)
M=LPTR(2)-1
N=LPTR(3)-LPTR(2)
CONST=FLCAT(M*N)/FLOAT((M+N)*(M+N))
ALPHYP=1.-C-RP(1)
SSUM=0.0
DO 35 I=1,K
SSUM=SSUM+(S(I)*S(I))
35 CONTINUE
T2=CONST*SSUM
WRITE(9,121) M,N,K,CONST,SSUM,T2
121 FORMAT(1X,M,N,K,CONST,SSUM,T2:',3I4,3F10.4)
CALL CRVALF(T2,ALPHAT)
WRITE (NCF,102) T2,ALPHAT,ALPHYP
102 FORMAT(10X,'***** CRAMER-VON MISES TEST *****'/1X,
* THE TEST STATISTIC T2:',F10.4/1X,
* GIVES THE ALPHA HAT OF:',F10.4/1X,
* COMPARED TO THE HYPOTHESISED ALPHA:',F10.4/1X,
* NULL HYP H0: F(X)=G(X) FOR ALL X',/1X,
* ALT HYP H1: F(X)>G(X) FOR AT LEAST ONE X')
IF (ALPHYP.LE.ALPHAT) WRITE (NOF,111)
IF (ALPHYP.GT.ALPHAT) WRITE (NOF,112)
111 FORMAT(1X,' DO NOT REJECT THE NULL HYPOTHESIS')
112 FORMAT(1X,' REJECT THE NULL HYPOTHESIS')
RETURN
END

```

```

C-----
C SUBROUTINE CRVALF(T2,ALPHAT)
REAL W(14),ALF(14),ALPHAT,PART,T2,DE,NU,L SIG
W(1)=0.0
W(2)=0.04
W(3)=0.062
W(4)=0.079
W(5)=0.097
W(6)=0.115
W(7)=0.147
W(8)=0.184
W(9)=0.241
W(10)=0.347
W(11)=0.461
W(12)=0.743
W(13)=1.168
W(14)=99

```

```

ALF(1)=0.0
ALF(2)=0.1
ALF(3)=0.2
ALF(4)=0.3
ALF(5)=0.4
ALF(6)=0.5
ALF(7)=0.6
ALF(8)=0.7
ALF(9)=0.8
ALF(10)=0.9
ALF(11)=C.95
ALF(12)=C.99
ALF(13)=C.9999
ALF(14)=1.0

```

```

I=0
10 I=I+1
IF (T2.GT.W(I)).AND.(I.LE.13) GO TO 10
IM1=I-1
NU=T2-W(IM1)
DE=W(I)-W(IM1)
PART=NU/DE
LSIG=ALF(IM1)+PART*(ALF(I)-ALF(IM1))
ALPHAT=1.0-LSIG
WRITE(9,131) I, PART, T2, ALF(IM1), ALF(I), ALPHAT,
* W(IM1), W(I), NU, DE
131 FORMAT(IX,'I',FART,T2,ALF-ALF,ALPHAT',I3,5F10.4/1X,
* ' W(IM1),W(I),NU,DE',4F10.4)
RETURN
END

```

```

SUBROUTINE UTEST
-----
PURPOSE
TEST WHETHER TWO INDEPENDENT GROUPS ARE FROM SAME
POPULATION BY MEANS OF MANN-WHITNEY U-TEST

USAGE
CALL UTEST(A,R,N1,N2,U,Z,IER)

DESCRIPTION OF PARAMETERS
A - INPUT VECTOR OF CASES CONSISTING OF TWO INDEPENDENT
    GROUPS. SMALLER GROUP PRECEDES LARGER GRGUP. LENGTH
    IS N1+N2.
R - OUTPUT VECTOR OF RANKS. SMALLEST VALUE IS RANKED 1.
    LARGEST IS RANKED N. TIES ARE ASSIGNED AVERAGE OF TIE
    RANKS. LENGTH IS N1+N2.
N1 - NUMBER OF CASES IN SMALLER GROUP.
N2 - NUMBER OF CASES IN LARGER GROUP.
L - STATISTIC USED TO TEST HOMOGENEITY OF THE TWO
    GROUPS (OUTPUT).
Z - MEASURE OF SIGNIFIGANCE OF U IN TERMS OF NORMAL
    DISTRIBUTION (OUTPUT).
IER- 0, IF NO ERROR.
    - 1, IF ALL VALUES OF ONE GROUP ARE TIED.

REMARKS
Z IS SET TO ZERO IF N2 IS LESS THAN 20

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
RANK
TIE

METHOD
DESCRIBED IN S. SIEGEL, 'NONPARAMETRIC STATISTICS FOR THE
BEHAVIORAL SCIENCES', MCGRAW-HILL, NEW YORK, 1956,
CHAPTER 6

```

.....

```

C
SUBROUTINE UTEST(A,R,N1,N2,U,Z,IER)
C DIMENSION A(1),R(1)
C RANK SCORES FROM BOTH GROUP TOGETHER IN ASCENDING ORDER, A
C ASSIGN TIED OBSERVATIONS AVERAGE OF TIED RANKS
C
N=N1+N2
CALL RANK(A,R,N)
Z=0.0
C
SUM RANKS IN LARGER GROUP
C
R2=0.0
NP=N1+1
DO 10 I=NP,N
10 R2=R2+R(I)
C
CALCULATE L
C
FNX=N1*N2
FN=N
FN2=N2
UP=FNX+FN2*((FN2+1.0)/2.0)-R2
U=FNX-UP
20 IF(UP-U) 2C,3C,30
U=UP
C
TEST FOR N2 LESS THAN 20
30 IF(N2-20) 80,40,40
C
COMPUTE STANDARD DEVIATION
40 KT=1
CALL TIE(R,N,KT,TS)
IF(TS) 50,6C,50
50 IF(TS-(FN*FN*FN-FN)/12) 52,51,52
51 IER=1
GO TO 80
52 S=SQRT((FNX/(FN*(FN-1.0)))*(((FN*FN*FN-FN)/12.0)-TS))
GO TO 70
60 S=SQRT(FNX*(FN+1.0)/12.0)
C
COMPUTE Z
70 Z=(U-FNX*C.5)/S
80 RETURN
END
C
.....
C
SUBROUTINE RANK(A,R,N)
C DIMENSION A(1),R(1)
C INITIALIZATION
DO 10 I=1,N
10 R(I)=0.0
C FIND RANK OF DATA
DO 100 I=1,N
C TEST WHETHER DATA POINT IS ALREADY RANKED
IF(R(I)) 2C,20,100
C DATA POINT TO BE RANKED
20 SMALL=0.0
EQUAL=0.0
X=A(I)
DO 50 J=1,N
IF(A(J)-X) 30,40,50
C COUNT NUMBER OF DATA POINTS WHICH ARE SMALLER

```



```

30 SMALL=SMALL+1.0
GO TC 50
C   COUNT NUMBER OF DATA POINTS WHICH ARE EQUAL
40 EQUAL=EQUAL+1.0
R(J)=-1.0
50 CONTINUE
C   TEST FOR TIE
C   IF(EQUAL-1.0) 60, 60, 70
C   STORE RANK OF DATA POINT WHERE NO TIE
60 R(I)=SMALL+1.0
GO TC 100
C   CALCULATE RANK OF TIED DATA POINTS
70 P=SMALL + (EQUAL + 1.0)*0.5
DO 90 J=I,N
IF(R(J)+1.0) 90, 80, 90
80 R(J)=P
90 CONTINUE
100 RETURN
END

C   SUBROUTINE TIE(R,N,KT,T).....
C   DIMENSION R(1)
C   INITIALIZATION
T=0.0
Y=0.0
5 X=1.0E38
IND=0

C   FIND NEXT LARGEST RANK
DO 30 I=1,N
IF(R(I)-Y) 30,30,10
10 IF(R(I)-X) 20,30,30
20 X=R(I)
IND=IND+1
30 CONTINUE
C   IF ALL RANKS HAVE BEEN TESTED, RETURN
IF(IND) 90,90,40
40 Y=X
CT=0.0
C   COUNT TIES
DO 60 I=1,N
IF(R(I)-X) 60,50,60
50 CT=CT+1.0
60 CONTINUE
C   CALCULATE CORRECTION FACTOR
IF(CT) 70,5,70
70 IF(KT-1) 75,80,75
75 T=T+CT*(CT-1.0)/2.0
GO TC 5
80 T=T+(CT*CT*CT-CT)/12.0
GO TC 5
90 RETURN
END

```

APPENDIX G

PASCAL SOURCE CODE: LOCHINVAR

```

PROGRAM LOCHI (INPUT,OUTPUT,QIN,COU,A,B,C,D,E,F,CP);
CONST
  TOT_DATA=400;
  TOT_VARS=16;
  L_NAME=50;
  L_I_PAR=50;
  L_E_B=50;

TYPE
  INT=INTEGER;
  OUTFILE=1..16; CUTFSET=SET OF OUTFILE;
  T_W_C=1..50; T_W_CEL=SET OF T_W_C;
  T_W_CELSET=SET OF T_W_CEL;
  T_W_CELSET=SET OF T_W_CELSET;
  T_W_CELSET=SET OF T_W_CELSET;
  VECTOR=1..2; T_ARRAY=1..TOT_VARS;
  VECTORI=ARRAY[1..TOT_VARS] OF INTEGER;
  STRING=PACKED ARRAY[1..L_NAME] OF CHAR;
  STR=ARRAY[1..L_I_PAR] OF STRING;
  MA_PAR=ARRAY[1..L_I_PAR] OF REAL;
  I_PAR=ARRAY[1..L_I_PAR] OF INTEGER;
  VEC_E_B=ARRAY[1..L_E_B] OF INTEGER;

VAR
  CELL:TESTIM-CEL;
  STATE:MAP; I_PAR: I_PAR;
  V_NAME:MAP; I_PAR: I_PAR;
  PART_VAL: I_PAR; I_PAR;
  E_B:VECTOR;
  F-SPEC:ARRAY[1..TOT_VARS] OF STRING;
  NCT:ARRAY[1..TOT_VARS] OF STRING;
  OP:ARRAY[1..TOT_VARS] OF STRING;
  LIST:VECTOR;
  X,Y:ARRAY[1..TOT_VARS] OF STRING;
  N_CELL:ARRAY[1..TOT_VARS] OF STRING;
  NIF:ARRAY[1..TOT_VARS] OF STRING;
  IN:ARRAY[1..TOT_VARS] OF STRING;
  OUTSET:CLTFSET;

  (* OUTPUT FOR CPTIGNS FILE IS WRITTEN IN SIX PLACES: IN AND
  OUT FILE # IN FILE IO, THE TEST NUMBER IN WNTST,
  IF CELL COUNTS ARE USED IN W_CELLS, VAR
  VARIABLES INCLUDED IN TEST IN WHICH, VAR
  EXCLUDED BLOCKS IN EX_BLOC, AND OTHER PARAMETER IN W_PAKAM*)
  
```

```

(* VARIABLES:
CELL_CT
CELL_CTR
CELLIST
EQUAL_S
H_A_H_C
I_P_V_T
IN_NAME
FILE_TYP
LIST_IN_F
N_OUT_F
N_VARS
N_C_T
N_I_P
N_ROW
N_COL
N_TEST
N_ELMS
N_EQ_S
PART_VAL
PROCEDURE RST:(*ALLOWS NEW ENTRIES FROM THE TERMINAL*)
BEGIN
  TERMIN(CIN);
END;
(*FUNCTION YES: BOOLEAN;
BEGIN
  REPEAT RST;
  WRITELN(QOUT, ' ENTER Y OR N ');
  READLN(QIN, Y, OR_N); I:=ORD(Y, OR_N);
  IF (I > 200) THEN WRITELN(QOUT,
    ' YOU USED ALTERNATE <APL> CHARACTERS. TRY AGAIN');
  UNTIL (I IN (.149, 168.));
  YES:=(I=168);
END;
(*PROCEDURE W_NTEST(IN_TEST:INT);
BEGIN
  CELLS:=0;
  (****ADD IN THE TOTAL NUMBERS OF NTEST WITH CELLS****)
  IF (IN_TEST IN CELL_T) THEN CELLS:=1;

```

```

WRITELN(CP, A_TEST:5, CELLS:5);
(*W_TEST*)
PROCEDURE W_CELLS(TYP_CEL I, P1, N_COL, N_TO_RD: INT; PART_VAL: R_PAR;
  CELTST: I_PAR);
VAR I, J: INT;
(*TYP_CEL: 1-PARTITION BINOM 2-1ST BINOM 3-PARTITION, TOTALS, DIRECT ENTRY OF CELL TOTALS
4-FIXED ROW/COL TOTALS, 5-DIRECT ENTRY OF CP FILE, TOTALS
60-S-MCNAME: 61-CELL COUNT & NAME IN TEST CP FILE, FILE
62-CELL COUNT IN TEST OP FILE, NAME IN DATA FILE
63-CELL COUNT PARTITIONED, NAME IN DATA FILE
70-S-COEF: 71-CELL COUNT & NAME IN TEST OP FILE
72-CELL COUNT IN TEST OP FILE, NAME IN DATA FILE
73-CATA PARTITIONED NAME IN DATA FILE *)
(*I, P1: FOR 1, 2: SUCCESS HIGH/ON LIST: =1//FCR 3-5: N_ROW*)
BEGIN
  WRITELN(CP, TYP_CEL:5, I, P1:5, N_COL:5, A_TC_RD:5);
  IF (TYP_CEL IN (1, 3, 63, 73, 7)) THEN BEGIN
    IF (N_TO_RD > 0) THEN
      FOR I:=1 TO N_TO_RD DO
        WRITE(OP, PART_VAL(I, 1):10:4);
    IF ((I=N_TO_RD) OR (I MOD 8)) THEN WRITELN(OP); END;
  END;
  IF (TYP_CEL IN (4, 15, 61, 62, 71, 72, 7)) THEN BEGIN
    FOR I:=1 TO N_TO_RD DO
      WRITE(OP, CELTST(I, 1):5);
    IF ((I=N_TO_RD) OR (I MOD 16)) THEN WRITELN(OP); END;
  END;
  (*W_CELLS*)
  (*PROCEDURE SETT_CELL(N_EQ_S, MAX_R, MAX_C: INT; VAR N_ROW, N_COL: INT);
  VAR JJ, P_F: TO FIRST VAR OF PAIR, COL REFERS TO SECOND OF PAIR*)
  BEGIN
    CELTST(I, 1):=0; PART_VAL(I, 1):=0;
    IF (MAX_R=2) THEN N_ROW:=2; IF (MAX_C=3) THEN N_COL:=2;
    IF REPEAT RST;
      WRITELN(QOUT, ENTER # ROWS = # DIVISIONS IN 1ST VARIABLE);
      READLN(CIN, N_ROW);
      WRITELN(QOUT, ENTER # COLUMNS = # DIVISIONS IN 2D VAR.); RST;
      READLN(CIN, N_COL); WRITELN(QOUT, N_ROW:5, N_COL:5, CORRECT?);
      UNTIL YES; YOU MAY ENTER PARTITION VALUES TO DIVIDE THE VALUES
      OF THE FIRST VARIABLE INTO ROWS, THE SECOND VAR INTO COLUMNS;
      WRITELN(QCLT, OR YOU MAY SPECIFY FIXED ROW AND COLUMN TOTALS.);
      REPEAT RST; WRITELN(QOUT, ENTER P FOR PARTITION OR F FOR FIXED);
      READLN(CIN, P); P_F:=ORD(H)

```

```

IF (P F=151) THEN BEGIN(//PARTITION//*)
REPEAT (*S START ROW DO PARTITION JM1:=J-1;
FOR J:=2 TO NOUT; ENTER PARTITION VALUE THAT SEPARATES*);
WRITELN(QOUT; ROW; JM1:5; AND ROW; J:5);
RST:READLN(QIN; PART VAL(.JM1.)); END; (*FOR J*)

FOR J:=1 TO (N ROW-1) DO
WRITELN(QOUT; J:5; PART VAL(.J.):10:3; 'CORRECT?')
UNTIL (YES); (*END ROW PARTITIONS *)
REPEAT (*S START CCL DO PARTITION JM1:=I-1;
FOR I:=2 TO NOUT; ENTER PARTITION VALUE THAT SEPARATES*);
WRITELN(QOUT; CCL; JM1:5; AND CCL; I:5);
RST:READLN(QIN; PART VAL(.JM1+N_RCM-1.)); END; (*FOR I*)

FOR I:=1 TO (N CCL-1) DO
WRITELN(QOUT; PART VAL(.I+N_ROW-1.):10:3; 'CORRECT?')
UNTIL (YES); (*END CCL PARTITIONS*)
WRITELN(QOUT; (N_RCM+COL-2); PART_VAL, CELIST);
END; (*IF P F=131*) (*// PARTITION //*)
IF (P F=134) THEN BEGIN (*// FIXED ROW/CCL TOTALS //*)
SIZE_RCM:=N; CCL:=N EQ_S; I:=1;
WRITELN(QOUT; THERE ARE A TOTAL OF; N EQ_S; PAIRS*);
REPEAT (*S FIXED ROW TOTALS *)
REPEAT (*S PAIRS LEFT FOR ROW ASSIGNMENT*);
RST:WRITELN(QOUT; ENTER FIXED TOTAL FOR ROW; I:5);
READLN(QIN; HOLD); IF (HOLD<=SIZE_ROW) THEN BEGIN
CELLIST(.I.);:=HOLD; SIZE_ROW:=SIZE_ROW-HOLD; I:=I+1; END;
UNTIL (SIZE_ROW<=0); K:=1; I:=I-1;
IF (I<N_ROW) THEN BEGIN REPEAT TOTALS; I:=5; NOT EQUAL TO*);
WRITELN(QOUT; # VALS FOR ROW; N_ROW; J; ENTER N TO CHANGE*);
; # ROWS OR V ENTERED VALUES*);
RST:READLN(QIN; H); K:=ORD(H) UNTIL (K IN (.149,165.));
IF (K=149) THEN BEGIN SIZE_ROW:=N EQ_S; I:=1; END; (*IF K=165*)
IF (*IF I<N_ROW*)
END; (*S FIXED CCL TOTALS*)
REPEAT (*S PAIRS LEFT FOR COL ASSIGNMENT*);
RST:WRITELN(QOUT; ENTER FIXED TOTAL FOR COL; J:5);
READLN(QIN; HOLD); IF (HOLD<=SIZE_COL) THEN BEGIN
CELLIST(.J.);:=HOLD; SIZE_COL:=SIZE_COL-HOLD;
J:=J+1; END; (*IF HOLD<=SIZE_COL*)
UNTIL (SIZE_COL<=0); J:=J-1;
IF (J<N_COL) THEN REPEAT

```

```

WRITELN(QOUT, '# VALS FOR COL TOTALS: ', J:5, ' NOT EQUAL TO ',
; PREVIOUSLY ENTERED #COLS, #N_CGL:5, ' ENTER N TC CHANGE ',
; #COLS OR V TO CHANGE VALUES ');
RSI:READLN(QIN, HI); K:=ORD(H) UNTIL (K IN (.149,165.));
IF (K=149) THEN N_CGL:=J;
IF (K=165) THEN BEGIN SIZE_COL:=N_EQ_S; J:=1; END; (*IF K=165*)
END; (*J<>N_CGL*)
UNTIL (J=N_CGL) = (#COL TOTALS*)
WRITELN(COLS(4, N_ROW, N_T), PART_VAL, CELIST);
ENC: (*IF P F=14*)
END; (*SETT_CELL*)
-----*)
(*PROCEDURE DICCT: (* GETS RULE TO DICOTOMIZE DATA SET *)
VAR I:INTEGER;
BEGIN
WRITELN(QOUT, 'TC PERFORM THIS TEST. THE DATA MUST BE ',
; EITHER PARTITION (I.E. DIVIDE DATA INTO TC
; SETS OR ABOVE SET);
; VALUES CF ONE SET);
REPEAT READLN(QIN, TYPD); UNTIL (TYPD=ORD(TYPDIC));
UNTIL (DICTY=151) THEN BEGIN
IF (DICTY=151) WRITELN(QOUT, ' ENTER P FOR PARTITION, L FOR LIST');
REPEAT RST:=1; WRITELN(QOUT, ' ENTER PARTITION VALUE');
WRITELN(QIN, PART_VAL(1));
UNTIL (YES); S_HI:=0; (*IMPLIES FAILURES HIGH > PARTITION*)
WRITELN(QOUT, ' ARE SUCCESSSES THOSE ABOVE THE PARTITION?');
IF (YES) THEN S_HI:=1; W_CELLS(1, S_HI_LST, 2, 1, PART_VAL, CELIST);
END;
IF (DICTY=147) THEN BEGIN
REPEAT WRITELN(QOUT, ' ENTER THE NUMBER CF DIFFERENT VALUES IN ',
; LIST (SMALLER OF TWO POSSIBLE)); RST:=READLN(QIN, DICOP);
WRITELN(QOUT, DICOP, ' CORRECT?'); UNTIL (YES);
REPEAT FOR I:=1 TO DICOP DO BEGIN WRITELN(QOUT, ' ENTER VALUE ',
; I); RST:=READLN(QIN, PART_VAL(I)); END;
WRITELN(QOUT, ' THESE ARE THE VALUES YOU ENTERED: ');
FOR I:=1 TO DICOP DO WRITE(QOUT, PART_VAL(I), ' ');
WRITELN(QOUT); WRITELN(QOUT, ' CORRECT?');
UNTIL (YES); S_HI:=0; (*IMPLIES FAILURES WERE LISTED*)
WRITELN(QOUT, ' WERE THE LISTED VALUES SUCCESSSES?');
IF (YES) THEN S_HI_LST:=1;
W_CELLS(2, DICOP, PART_VAL, CELIST);
END; (*DICOT*)
-----*)
(*PROCEDURE DICOT*)

```

```

FUNCTION HYP(H_A+B_h_c:STRING;K:INT):INT;
VAR HYPER:CHAR;
BEGIN
  WRITELN(QCUT, 'A); WRITELN(QOUT,H,B); WRITELN(QOUT,H,C);
  REPEAT RST; WRITELN(QOUT, ' ENTER A,B,OR C');
  READ(QIN, HYPER); I:=ORD(HYPER)-128
  UNTIL I IN (1,2,3,1);
  HYP:=I;
END;
(*-----*)
PROCEDURE W_PARAM(N_I_P,N_R_P:INT;I_P:I_P;I_PAR;R_P:R_PAR);
VAR I:INT;
BEGIN
  WRITELN(CP,N_I_P,5);
  IF (N_I_P>0) THEN FOR I:=1 TO N_I_P DO BEGIN
    WRITELN(OP,I_P(I),5);
    IF ((I=N_I_P) OR (0=(I MOD 16))) THEN WRITELN(OP); END;
  WRITELN(CP,N_R_P,5);
  IF (N_R_P>0) THEN FOR I:=1 TO N_R_P DO BEGIN
    WRITELN(OP,R_P(I),10,3);
    IF ((I=N_R_P) OR (0=(I MOD 8))) THEN WRITELN(CP); END;
  IF (*W_PARAF*)
END; (*W_PARAF*)
(*-----*)
PROCEDURE NONE_EX;
VAR I:INT;
BEGIN
  I:=0;
  WRITELN(OP, I,5);
  WRITELN(OP, I,5);
END; (*NONE_EX*)
(*-----*)
FUNCTION P_C:REAL;
VAR PERCENT:REAL;
BEGIN REPEAT PRST;
  WRITELN(QCUT, ' ENTER NUMBER>0, IF <1, START WITH 0.1);
  READ(QIN, PERCENT);
  PERCENT:=PERCENT;
END; (*P_C*)
(*-----*)
FUNCTION L_SIG:REAL; (*RETURNS A LEVEL OF SIGNIFICANCE*)
BEGIN
  WRITELN(QCUT, ' ENTER LEVEL OF SIGNIFICANCE, ALPHA');
  L_SIG:=1.0-(P_C);
END;
(*-----*)
PROCEDURE FILE_ID(VAR NIF,N_INF:INTEGER);
VAR J,K,N_OUT,F,L:INTEGER;
BEGIN REWRITE(OP, 'NAME = LOCHI.OP.A, RECFM=F, LRECL=80'); L:=0;

```

```

REPEAT RST;WRITELN(QOUT, WHICH FILE NAME FOR OUTPUT? U-2.);
READLN(CIN, CUTF);K:=ORD(COUT)-153 UNTIL (K IN (.11, .16, .1));
NIF:=N IN F-128;(*NIF IS NUMBER READ BY FORTRAN PROGRAM*)
IF (NIF=5) THEN NIF:=7;(*TC PROTECT TEST F FROM INTERFERENCE*)
WRITELN(OP, NIF, K, 5);

END;
(*PROCEDURE WHICH VEC(N VARS, N C-T; INT);
-----*)
VAR V1, V2, I: INT;
HELD:=FELCEST;
HOLDING:=BCCLEAN; (*TRUE IF HELD IS NOT EMPTY*)
BEGIN
HOLDING:=FALSE;
IF (N VARS>1) AND (N C T=1) THEN REPEAT RST;N_V_T:=1;
WRITELN(QOUT, OF, N VARS, 3, VARIABLES IN DATA SET,.);
WRITELN(COUT, WHICH VARIABLE FOR THIS TEST?);
READ(QIN, V_T(1..1)) UNTIL (V_T(1..1)<=N VARS);
IF (N VARS=1) AND (N C T=1) THEN BEGIN N_V_T:=1;
WRITELN(QOUT, CF, N VARS, 3, VARIABLES IN DATA SET, T);
READ(QIN, V1, V2); UNTIL (V1<=N VARS) AND (V2<=N VARS);
HELD:=(V1=2) AND (N C T=2); HOLDING:=TRUE; END;
IF (N_V_T(1..1)=1) AND (N C T=2) THEN BEGIN N_V_T:=2;
IF (N VARS>2) AND (N C T>2) THEN
BEGIN
WRITELN(QOUT, DO YOU WANT ALL, N VARS, 3, INCLUDED IN TEST.);
IF (YES) THEN BEGIN FOR I:=1 TO N_VARS DO V_T(I..I):=I;
ELSE
BEGIN
HOLDING:=TRUE; HELD:=(.255.); RST; REPEAT
WRITELN(QOUT, HOW MANY VARIABLES ARE INCLUDED IN TEST?);
READ(QIN, V_T) UNTIL (N_V_T<=N VARS); RST;
WRITELN(QOUT, ENTER INDICES OF VARIABLES TO BE IN TEST.);
FOR I:=1 TO N_V_T DO
REPEAT OKAY:=FALSE; RST;
WRITELN(QOUT, ENTER INDEX OF INCLUDED VARIABLE, I, 5);
READ(QIN, VI); IF (VI<=N VARS) AND NOT (VI IN HELD) THEN
BEGIN HELD:=HELD+(.V1.); OKAY:=TRUE; END
UNTIL OKAY;
END;
END;
J:=0;
IF HOLDING THEN
FOR I:=1 TO N_VARS DO

```



```

IF (I IN HELD) THEN BEGIN
  J:=J+1; V_T(.J.):=I;END;
WRITELN(OP,N_V_T(.J.));
IF (N_V_T>1) THEN FOR I:=1 TO (N_V_T-1) DO WRITE(OP,V_T(.I.):5);
IF (N_V_T>C) THEN WRITELN(OP,V_T,N_V_T(.J.):5);
END; (*WRICH_VEC*)
-----*
PROCEDURE EXCLUD_E(N_EQ_S:INT);
VAR I,J,K,L,FCLD:INT;
XCLUD:=BOCLEAN; (*TRUE IF WANT TO EXCLUDE SOME TREATMENTS*)
GOOD:=BAD_CUPE:=VEC_E_B;
HELD:=HELDEST;
BEGIN
WRITELN(QOUT, ' DO YOU WISH TO EXCLUDE ANY BLOCKS FROM TEST');
XCLUD:=YES; IF NOT XCLUD THEN NONE_EX;
IF XCLUD THEN
  BEGIN RST;
  I:=0; J:=0; K:=0; (*COUNTERS FOR HOLD,GOCC, BAD VECTORS*)
  HELD:=(N_EQ_S+1) * (N_EQ_S+1) * (N_EQ_S+1);
  WRITELN(QOUT, 'HOW MANY BLOCKS DO YOU WISH TO EXCLUDE?');
  REPEAT WRITELN(QOUT, ' UP TO ',N_EQ_S:3, ' BLOCKS'); RST;
  REACLN(QIN,N_E_B) UNTIL (N_E_B<N_EQ_S);
  IF (N_E_B>0) THEN FOR I:= 1 TO N_E_B DO
    BEGIN
      WRITELN(QOUT, ' IN ASCENDING ORDER, '
        , ENTER INDICES, I -,N_EQ_S:4);
      REPEAT
        WRITELN(QOUT, ' ENTER INDEX FOR EXCLUDED BLOCK NR',I:3);RST;
        READ (QIN,HOLD); CKAY:=FALSE;
        IF (HOLD IN (.1..N_EQ_S)) AND (NOT(HOLD IN HELD)) THEN
          BEGIN OKAY:=TRUE;HELD:=HELD+(.HOLD.);END
        UNTIL OKAY;
      END; J:=0;
      FOR I:=1 TO N_EQ_S DO
        FOR IF (I IN HELD) THEN BEGIN
          J:=J+1; E_B(.J.):=I;END;
        WRITELN(OP,N_E_B:5);
        IF (N_E_B>0) THEN FOR L:=1 TO N_E_B DO BEGIN
          WRITE(OP,E_B(L.:5);
          IF ((L=N_E_B) OR (O=(L MOD 16))) THEN WRITELN(OP);END;
        END;
      END;
    END;
  (*PROCEDURE CEL_CTR(MAX_R,MAX_C:INT;VAR N_ROW,N_COL:INT);
  VAR J,I:INT;
  BEGIN
  IF (MAX_R=1) AND (MAX_C=2) THEN REPEAT RST; N_ROW:=1;N_COL:=2;
  WRITELN(QOUT, ' ENTER NUMBER OF SUCCESS, THEN NUMBER OF FAILURES');

```

```

READLN(CIN,CELIST(1,1),CELIST(2,1));
WRITELN(COUT,CELIST(1,1),CELIST(2,1));
IF (MAX_R>2) OR (MAX_C>2) THEN REPEAT RST;
WRITELN(QOUT,ENTER NUMBER OF ROWS AND COLUMNS);
READLN(CIN,N_ROW,N_COL);
WRITELN(COUT,ROWS,N_ROW, # COLUMNS,N_COL,5, CORRECT?);
UNTIL YES;
IF (MAX_R=2) THEN N_ROW:=2; IF (MAX_C=2) THEN N_COL:=2;
IF (MAX_F>1) THEN BEGIN
WRITELN(QOUT,ENTER CELL COUNTS BY ROW);
REPEAT FOR I:=1 TO N_ROW DO
FOR J:=1 TO N_COL DO BEGIN RST;
WRITELN(QIN,CELIST(I,J)); END;
READLN(QIN,ROW DO BEGIN
FOR I:=1 TO N_ROW DO BEGIN
FOR J:=1 TO (N_COL-1) DO
WRITE(CELIST(I,J)); END;
WRITELN(QOUT,CELIST(I,N_COL)); END;
WRITELN(QOUT,CORRECT?);
UNTIL YES;
END; (* IF MAX_R > 1 *)
W CELLS(5,N_ROW,N_COL,(N_ROW*N_COL),PART_VAL,CELIST);
END; (*CELIST*)
-----*
PROCEDURE MONCREG(N_VARS,N_EQ_S:INT);
VAR I,K:INT;
BEGIN
WHICH_VEC(N_VARS,2);
EXCLUDE(N_EQ_S);
WRITELN(COUT,IN Y IS THE DEPENDENT VARIABLE, AND );
WRITELN(COUT,E(Y,X)=A+B*(X-E(X)); B<>0);
WRITELN(COUT,IS THE LOWER INDEXED VARIABLE OF THE TWO THE);
IF YES THEN I:=1 ELSE I:=2;
WRITELN(COUT,DO YOU WANT TO PREDICT VALUES OF X?);
IF NOT(YES) THEN I_P(2.0):=0 ELSE
BEGIN REPEAT
RST; WRITELN(QOUT,ENTER # VALUES TO BE PREDICTED.);
READLN(QIN,I); WRITELN(QOUT,I,5, CORRECT?);
UNTIL (YES AND (I > 0));
I_P(2.0):=I;
K:=I; I:=1 TO K DO
FOR BEGIN
REPEAT

```

```

RST: WRITELN(QCUT,'ENTER X VALUE #',I,5);
READLN(QIN,X);WRITELN(COUT,X:10:3,' CORRECT?');
UNTIL YES:
  R_P(I,1):=X;
  END; (*FOR K DO *)
WRITELN(COUT,'DO YOU WANT TO HAVE A TRACE OF THE REGRESSION?');
IF YES THEN I_P(3,1):=1 ELSE I_P(3,1):=0;
K:=I_P(2,1);
WPARAM(3,K,I_P,R_P);
END; (*MONOREG*)
-----*)
(*PROCEDURE GDFIT_U(VAR R_P:R_PAR);
VAR ALPHA,BETA:REAL;
BEGIN
  WRITELN(QOUT,'THE UNIFORM DISTRIBUTION HAS AS PARAMETERS');
  WRITELN(COUT,'THE ENDPOINTS OF THE DISTRIBUTION');
  WRITELN(COUT,'ENTER ALPHA (LOWER ENDPOINT) AND BETA');
  REPEAT RST: WRITELN(QOUT,'ALPHA < BETA');
  READLN(QIN,ALPHA,BETA); UNTIL (ALPHA<BETA);
  I_P(2,1):=0;R_P(1,1):=ALPHA;R_P(2,1):=BETA;
  I_P(2,1):=0;
  END; (*GDFIT_U*)
-----*)
(*PROCEDURE GDFIT_EX(VAR I_P:1_PAR;VAR R_P:R_PAR);
VAR LAMBDA:REAL;EST:BCOLEAN;
BEGIN
  WRITELN(QOUT,'THE EXPONENTIAL DISTRIBUTION HAS PARAMETERS');
  WRITELN(COUT,'LAMBDA, THE RATE OR SCALE PARAMETER');
  WRITELN(COUT,'YOU MAY ENTER LAMBDA OR HAVE THE PROGRAM');
  WRITELN(COUT,'ESTIMATE BY METHOD OF MOMENTS. THE LATTER');
  WRITELN(COUT,'METHOD WILL USE THE SAMPLE MEAN AS THE');
  WRITELN(COUT,'RECIPROCAL OF THE RATE. DO YOU WISH TO');
  WRITELN(COUT,'ENTER A VALUE OF LAMBDA?');
  IF YES THEN BEGIN REPEAT RST:WRITELN(COUT,'LAMBDA=');
  READLN(QIN,LAMBDA); I_P(2,1):=0;
  WRITELN(COUT,'LAMBDA=, LAMBDA:10:3 ' CORRECT?');
  UNTIL YES; R_P(1,1):=LAMBDA;
  ELSE I_P(2,1):=1;
  END;
  END; (*GDFIT_EX*)
-----*)
(*PROCEDURE GDFIT_N(VAR I_P:1_PAR;VAR R_P:R_PAR);
BEGIN
  WRITELN(QOUT,'YOU MAY SPECIFY THE THEORETICAL DISTRIBUTIONS');
  WRITELN(COUT,'TWO PARAMETERS--MEAN AND VARIANCE--OR HAVE THE');
  WRITELN(COUT,'PROGRAM ESTIMATE FROM THE SAMPLE BY THE METHOD');
  WRITELN(COUT,'OF MOMENTS. DO YOU WANT TO SPECIFY PARAMETERS?');
  SPEC:=YES; IF SPEC THEN BEGIN REPEAT
  WRITELN(QOUT,'ENTER MEAN AND VARIANCE FOR THEORETICAL DIST');

```

```

RST: REACLN(QIN,MU,VARIANCE);WRITELN(QCUT,MU,MU:10:3,
VAR,VARIANCE:10:3,CORRECT?) UNTIL YES;
R_P(1)=MU;R_P(2)=VARIANCE;I_P(2)=0;
END;(*IF SPEC*)
IF NOT SPEC THEN I_P(2)=2;
END;(*GOOD FIT*)
-----*)
PROCEDURE RC_CCNT(N_VARS,N_EQ_S:INT;CELL_CT:BOOLEAN);
VAR I,J:INT;
BEGIN
IF CELL_CT THEN CEL_CTR(25,25,N_ROW,N_COL);
IF (NOT CELL_CT) THEN BEGIN
SETT_CELL(N_EQ_S,25,25,N_ROW,N_COL);
WHICH_VEC(N_VARS,2);
EXCLUD_B(N_EQ_S);END;(*IF NOT CELL_CT*)
I_P(1)=0;R_P(1)=L_P(1);
W_PARAM(0,1,I_P,R_P);
END;(*RC_CNT*)
-----*)
PROCEDURE BINCH(N_VARS:INT;CELL_CT:BOOLEAN);
(*THIS IS TEST PROCEDURE FOR BINOMIAL TEST*)
VAR THIS_VAR,I,J,K:INT;(*THIS_VAR IS INDEX OF A
BEGIN
IF CELL_CT THEN CEL_CTR(1,2,N_ROW,N_COL);
IF NOT CELL_CT THEN BEGIN
DICTO:WHICH_VEC(N_VARS,1);NONE_EX;END;(*IF NOT CELL_CT*)
N_ROW:=2;N_I_P:=1;
WRITELN(QCUT);ENTER HYPOTHIZED P*,E.G.,0.8*);
R_P(1)=P;C:HYPOTHESIS CF FORM: P=P*
H_A:=A:NULL F: P<=P*
H_B:=B:NULL F: P>=P*
H_C:=C:NULL F: P>=P*
I_P(1)=FYP(F_A,F_B,H_C,96);
R_P(2)=L_SIG;
W_PARAM(1,2,I_P,R_P);
END;(*BINOM*)
-----*)
PROCEDURE COXSTU(N_VARS,N_EQ_S:INT;EQUAL_S:BOOLEAN);
(*THIS IS TEST PROCEDURE FOR COX-STUART TEST FCR TREND*)
VAR THIS_VAR,TWO,I,J,K,INDEP,DEP:INT;
BEGIN
I_P(2)=1;N_I_P:=2;
IF (N_VARS>1) AND EQUAL_S THEN BEGIN
WRITELN(QCUT);YOU HAVE TWO CPTIONS FCR THE COX-STUART TEST:);
WRITELN(QCUT);1. CHOOSE ONE VARIABLE; TEST FOR TREND IN THE ;
WRITELN(QCUT);2. CHOOSE TWO VARIABLES; TEST FCR POSITIVE CR ;
WRITELN(QCUT);3. CHOOSE TWO VARIABLES; TEST FCR NEGATIVE CR ;
WRITELN(QCUT);4. CHOOSE TWO VARIABLES; TEST FCR CORRELATION, HAVING THE RANKS OF ONE VARIABLE ;

```

```

* (INDEPENDENT) ORDER THE VALUES OF THE OTHER (DEPENDENT) *
* VARIABLE FOR THE TEST FOR TRENC. *
REPEAT RST: WRITELN(QUIT, ' ENTER CPTICN (1 OR 2) ');
REALLN(QIN, 1) UNTIL (I IN (.1..2.)); I_P(.2.):=1;
IF (I REPEAT RST: WRITELN(QOUT, ' CF, N_VARS:5, VARIABLES WHICH IS ',
DEPENDENT (WILL HAVE ITS ORDER REARRANGED)');
REALLN(CIN, DEP) UNTIL (DEP<=N_VARS);
REPEAT RST: WRITELN(QOUT, ' WHICH VARIABLE IS INDEPENDENT? ');
READLN(QIN, INDEP) UNTIL (INDEP<=N_VARS) AND (INDEP<>DEP);
RST: WRITELN(QOUT, ' INDEX OF DEPENDENT: ', DEP:5,
UNTIL (YES) I_P(.3.):=-DEP; I_P(.4.):=INCEP; N_I_P:=4;
(* THIS IN LIEU OF WHICH_VAR *)
I:=DEP; J:=INDEP; TWC:=2;
IF (DEP>INDEP) THEN BEGIN I:=J; J:=DEP; END; (*IF DEP>INDEP*)
WRITELN(OP, I:5, J:5);
END; (*IF I_P(.2.)=2*) AND E_SIZE*)
IF (I_P(.2.)=1) THEN WHICH_VEC(N_VARS, 1);
IF (I_P(.2.)=1) THEN EXCLUD_B(N_EQ_S) ELSE NONE_EX;
N_R_P:=1;
H_A:=A; F0: NO TREND H1: TREND OF INCREASE OR DECREASE *;
H_B:=B; H0: DECREASE OR NO TREND H1: TREND OF INCREASE *;
H_C:=C; F0: INCREASE OR NO TREND H1: TREND OF DECREASE *;
I_P(.1.):=FVP(F_A, F_B, H_C, 134);
R_P(.1.):=L_SIG;
WPARAM(N_I_P, 1, I_P, R_P);
END; (*COX STAR*)
-----*)
PROCEDURE QUANTILE(N_VARS:INT; CELL_CT:BOOLEAN);
(* THIS VAR, I, J, K:INT: (*THIS_VAR IS INDEX OF A
BEGIN
R_P:=2; N_I_P:=1;
WRITELN(QOUT, ' ENTER QUANTILE (I.E., P OF G(P)), E.G., 0.8 *);
R_P(.1.):=P; C;
RST: WRITELN(QOUT, ' ENTER HYPOTHESIZED X* FOR G(P) ');
READLN(QIN, PART_VAL(.1.)); CELIST(.1.):=1;
WHICH_VEC(N_VARS, 1);
NCNE_EX: (*NC BLOCKS EXCLUDEU*)
H_A:=A; A:NULL F: P(X<X*)=P *;
H_B:=B; B:NULL F: P(X<X*)<P *;
H_C:=C; C:NULL F: P(X<X*)>P *;

```

```

I_P(1) := TYP(A, T_B, M_C, 106);
R_P(2) := L_SIG;
W_PARAM(1, 2, I_P, R_P);
END; (* QLAN ILE *)
-----*)
PROCEDURE MANHPT(NVARS:INT);
VAR I, J, K, N_E_B:INT;
BEGIN
  WHICH_VEC(N_VARS, 2);
  R_P(I) := L_SIG; I_P(1) := 0; N_E_B := 0;
  WRITELN(OP, N_E_B, 5);
  W_PARAM(0, 1, I_P, R_P);
END;
-----*)
PROCEDURE SIGTEST(NVARS, N_EQ_S:INT);
VAR I, J, K:INT;
BEGIN
  WHICH_VEC(N_VARS, 2);
  EXCLUD(B(N_EQ_S));
  H_A := A: KULL FYP HO: P(+)=P(-); ALT HYP H1: P(+)<>P(-) ; ;
  H_B := B: NULL FYP HO: P(+)<=P(-); ALT HYP H1: P(+)>P(-) ; ;
  H_C := C: NULL FYP HO: P(+)>=P(-); ALT HYP H1: P(+)<P(-) ; ;
  I_P(1) := FYP(F_A, F_B, H_C, 123);
  R_P(I) := L_SIG;
  W_PARAM(1, 1, I_P, R_P);
END; (* SIGTEST *)
-----*)
PROCEDURE FRIECPMAN(N_VARS, N_EQ_S:INT);
VAR I, J, K:INT; XX:FEAL;
BEGIN
  WHICH_VEC(N_VARS, N_VARS);
  EXCLUD(B(N_VARS));
  WRITELN(QOUT, ' THIS WILL ALLOW THE FRIEDMAN OR DURBAN TEST'); ;
  WRITELN(QOUT, ' <CURBAN IS INCOMPLETE BLOCK CESIGN: NEEDS FILLER>'); ;
  WRITELN(QOUT, ' DO YOU WANT TO PERFORM THE FRIEDMAN TEST?'); ;
  IF YES THEN BEGIN
    R_P(1) := L_SIG; I_P(1) := 0;
    W_PARAM(1, 1, I_P, R_P); END (* IF *)
  ELSE PEAT
    REACTL(QIN, XX); WRITELN(QOUT, XX, 10, 3, ' CORRECT?'); ;
  UNTIL YES;
  REPEAT
    WRITELN(QOUT, ' ENTER K=# TREATS/BLOCK'); ;
    WRITELN(QOUT, ' AND R=# BLOCKS/TREAT'); ;
    REACTL(QIN, I, J); WRITELN(QOUT, I, 5, J, 5, ' CORRECT?'); ;
  UNTIL YES;

```

```

I_P(2,1)=I;I_P(3,1)=J
R_P(2,1)=X;I_P(1,1)=I;R_P(1,1)=L_SIG;
W_PARAM(3,2,I_P,R_P);
END; (*ELSE*)
END; (*FRIEDMAN*)
-----*)
PROCEDURE MEC_AG(N_VARS:INT;VAR I_P:I_PAR;VAR N_I_P:INT);
VAR KK,I:INT;
BEGIN
  N_I_P:=0;
  FOR I:=1 TO N_VARS DO BEGIN
    REPEAT
      WRITELN(COUT,'ENTER AGGREGATE VAR # FOR DATA SET VAR, I:5);
      RST:READLN(QIN, KK);WRITELN(QOUT, KK:5, ' CORRECT?');
    UNTIL ((YES) AND (KK IN (.0..N_VARS)));
    IF (KK > 0) THEN BEGIN
      N_I_P:=N_I_P+1;I_P(N_I_P,1)=KK;
      V_T(N_I_P,1)=I;END; (*IF KK>0*)
    END;
    WRITELN(CP,N_I_P:5);
    FOR I:=1 TO N_I_P DO WRITE(OP,V_T(I,1):5);
    WRITELN(CF);
  END; (*MED_AG*)
-----*)
PROCEDURE MEDIAN(N_VARS:INT);
BEGIN
  WRITELN(COUT,'DO YOU WANT TO COMBINE TWO CR MCRE VARIABLES?');
  WRITELN(QOUT,' INTO ONE OR MORE AGGREGATE VARIABLES?');
  IF NCT(YES) THEN BEGIN
    WHICH_VEC(N_VARS,N_VARS);
    N_I_P:=0;
  ELSE
    MED_AG(N_VARS,I_P,N_I_P);
  NONE_EX;
  R_P(1,1)=L_SIG;
  W_PARAM(N_I_P,1,I_P,R_P);
  END; (*MEDIAN*)
-----*)
PROCEDURE KRUMAL(N_VARS:INT);
BEGIN
  WHICH_VEC(N_VARS,N_VARS);
  NONE_EX;
  R_P(1,1)=I;I_P(1,1)=0;
  W_PARAM(0,1,I_P,R_P);
  END; (*KRUMAL*)
-----*)

```



```

REPEAT RST; WRITELN(QOUT,'HOW MANY VALUES OF X?');
READLN(CIN,I_P(.3));WRITELN(QOUT,I_P(.3));4; CORRECT?);
UNTIL YES;
K:=1; DO K BEGIN RST;
FOR WRITELN(CIN,R_P(.2+I)); END; (*FOR I*)
ELSE
END; (* PROGRAM CAN COMPUTE CONFIDENCE INTERVALS FOR I;
WRITELN(QOUT,' THE SLOPE PARAMETER B1: FOR LARGE DATA SETS IT:);
WRITELN(QOUT,' IS EXPENSIVE--0.5*N**2; N=#PAIRS);R_P(.2.):=0;
IF (N EQ S<61) THEN TO DO SO YOU NEED W VALUE FOR N=, N EQ S);
WRITELN(QOUT,' THEN REPEAT RST; BEGIN I_P(.4.):=1;
IF NOT (YES) THEN ENTER W FOR N=, N EQ S);
WRITELN(QOUT,' ENTER W FOR N=, N EQ S);
READLN(CIN,R_P(.2));WRITELN(QOUT,R_P(.2.):10:4,'CORRECT?');
UNTIL YES;
K:=I_P(.3.)+2;
W_PARAM(4,K,I_P,R_P);
END; (* REGRET *)
) (* PROCEDURE SMIRNOV(N_VARS:INT);
BEGIN
WHICH VEC(N_VARS:2);
NONE EX;R_P(1.):=0;
H_A:=; A:NULL;HYPOTHESIS CF FORM: F(X)=G(X);
H_B:=; B:NULL; F(X)=S(X), F(X) = 1ST POPULATION DIST.;
H_C:=; C:NULL; F(X)<S(X) AND G(X) = 2D DIST.;
I_P(1.):=HYP(A,H_B,H_C,369);
W_PARAM(1.0,I_P,R_P);
END;
) (* PROCEDURE SQUARANK(N_VARS:INT);
BEGIN
WHICH VEC(N_VARS,N_VARS);
NONE EX;I_P(1.):=0;R_P(.1.):=0;
W_PARAM(C.0,I_P,R_P);
END;
) (* PROCEDURE HARTLEY(N_VARS:INT);
BEGIN
WHICH VEC(N_VARS,N_VARS);
NONE EX;I_P(1.):=0;R_P(.1.):=0;
W_PARAM(0.0,I_P,R_P);
END;
) (*

```

```

PROCEDURE CRAPVONP(N_VARS:INT);
BEGIN
  WHICH_VEC(N_VARS,2);
  NONE_EX(I_PT,1):=0;R_P(1.1.):=0;
  W_PARAM(0.0.1_P,R_P);
END;
-----*)
PROCEDURE COCFRAN(N_VARS,N_EQ_S:INT;CELL_CT:BOOLEAN);
VAR IT,N_EX,NB,NT,I,J:INT;ILIST:ARRAY(1..400) OF INT;
HELD:=HELDST;
BEGIN
  CH_MIND:=FALSE;PAFT_VAL(1.1.):=0;CELIST(1.1.):=0;HELD:=(.255.);
  IF NOT CELL_CT THEN BEGIN
    TEST USES DICOTOMIZED DATA I.E.!!
    EACH ELEMENT CAN BE CLASSIFIED AS A SUCCESS THE!!
    OR A FAILURE FOR THE PROGRAM TO COUNT THEM!!
    SUCCESSES OR FAILURES! EACH TREATMENT OR VARS IABLE!!
    MUST BE DIVIDABLE INTO SUCCESSES TREATMENT!!
    ALL VALUES BELOW 19 ARE SUCCESSSES IN TREATMENT!!
    TWO ALL VALUES ABOVE 27 ARE SUCCESSSES ET CANNES OR!!
    YOUR DATA SET IS NOT DIVISIBLE IN THIS MANNER OR!!
    YOU DON'T WANT TO ENTER THIS CAN ENTER!!
    WHETHER A BLOCK/TREATMENT IS A SUCCESS OR FAILURE!!
    ONE CELL AT A TIME. THAT IS, YOU WANT TO CHANGE TO THIS!!
    LATTER METHOD, THAT IS, ENTERING SUCCESS OR FAILURE!!
    ONE CELL AT A TIME?);CH_MIND:=YES;
  IF NOT CH_MIND THEN BEGIN
    FOR J:=1 TO N_VARS DO BEGIN
      FOR I:=1 TO N_VARS DO BEGIN
        IF YES THEN BEGIN
          V_T(1..N_V_T.1.):=J; IT:=IT+1;
          REPEAT
            WRITELN(QOUT,'ENTER NUMBER WHICH DIVIDES TREATMENT',J:5);
            WRITELN(QOUT,'INTO SUCCESSES CR FAILURES:');
            RST:READLN(QIN,XX);WRITELN(QOUT,XX:10:3,' CORRECT?');
          UNTIL YES;
          REPEAT
            WRITELN(QOUT,'ARE VALUES BELOW',X:10:3,' SUCCESSES?');
            IF YES THEN Y:=1.0 ELSE Y:=-1.0;PART_VAL(IT#2.):=Y;
            IF (Y<C) THEN WRITELN(QOUT,'FOR TREATMENT',J:4);
            IF (Y>C) THEN WRITELN(QOUT,'(IT#2)-1.:1C:3,' ARE SUCCESSES');
            IF (Y=C) THEN WRITELN(QOUT,'(IT#2)-1.:1C:3,' ARE SUCCESSES');
            WRITELN(QOUT,'PART_VAL(.(IT#2)-1.):1C:3,' ARE SUCCESSES');
            WRITELN(QOUT,'CORRECT?');
          UNTIL YES;
        END;
      END;
    END;
  END;
  IF I IS IN TEST *

```

```

END; (*FOR I=1,N VARS *)
W CELLS(73,N-EQ,IT,(2*IT),PART_VAL, CELIST);
WRI TELN(CP,N-V,T,5);
FOR I:=1 TO C K V T I DO BEGIN
  WRI TE(OP,V-I,(.I.),5);
  IF ((I=IT) OR (O=(I MOD 16))) THEN WRI TELN(OP); END;

EXCLUD E IN CHARGE);
END; (*IF NOT CELL CHG*)
IF (CELL WRI TELN(COUT,IN,NT); BEGIN REPEAT
  RST: REACLN(CIN,NB,NT); WRI TELN(QOUT,'NUMBER OF TREATMENTS',NB:5);
  WRI TELN(COUT,'NUMBER OF TREATMENTS',NT:5,' CORRECT?');
  UNTIL YES;
  FOR I:=1 TO NB DO CELIST(.I.):=0;
  FOR I:=1 TO NB DO BEGIN K:=(I-1)*NT+J;
    FOR J:=1 TO NT DO BEGIN K:=1:3,' TREATMENT',J:3,' SUCCESS?';J;}
    WRI TELN(QOUT,'IS BLOCK',I:3,' ELSE ILIST(.K.):=0; END;(*FOR J*)
    IF YES THEN ILIST(.K.):=1 ELSE ILIST(.K.):=0;
    (*UNTIL ACCEPT THE 1/0 MATRIX*)
  REPEAT WRI TELN(COUT,'BLOCK*TREATMENT MATRIX,I=SUCCESS:');
    FOR I:=1 TO NB DO BEGIN WRI TE(QOUT,I:4,' ');
      FOR J:=1 TO C (NT-1) DO
        WRI TE(QOUT,ILIST(.I+1-1)*NT),J:3);
        WRI TELN(QOUT,ILIST(.I)*NT),J:3); END;(*FOR I*)
        WRI TELN(COUT,'CORRECT?'); OKAY:=YES;
        IF NOT OKAY THEN BEGIN
          WRI TELN(QOUT,ENTER INDEX OF BLOCK TO BE CORRECTED:');
          RST: READLN(QIN,I); WRI TELN(COUT,'CURRENT BLOCK',I:5);
          WRI TE(QOUT,I:4,' ');
          FOR J:=1 TO C (NT-1) DO
            WRI TELN(QOUT,ILIST(.I+1-1)*NT),J:3);
            FOR J:=1 TO C NT DO BEGIN K:=(I-1)*NT+J;
              WRI TELN(QOUT,' TREATMENT',J:3,' SUCCESS?');
              IF YES THEN ILIST(.K.):=1 ELSE ILIST(.K.):=0;
              WRI TELN(QOUT,ILIST(.K.):5);
              WRI TELN(QOUT,' IF NOT OKAY *')
            END;(*FOR J*)ENC;(* IF NOT OKAY *)
          UNTIL OKAY;
          FOR I:=1 TO NB DO BEGIN K:=(I-1)*NT+J;
            CELIST(.I.):=CELIST(.I.)+ILIST(.K.);
            CELIST(.J+NB.):=CELIST(.J+NB.)+ILIST(.K.);
            ENC;(*FOR J DO*)
            IF NOT CH_MIND THEN TYP_CEL:=71;
            IF CH_MIND THEN TYP_CEL:=72;
            IF CELLS(TYP_CEL,NB,NT,(NB+NT),PART_VAL, CELIST);
            W-CELLS(CT*)
          END;(*IF CELL_CT*)

```

```

R_P(1,1)=L_SIG;L_P(1,1)=0;
W_PARAM(0,1)=R_P;
END: (*COCHRAN*)
(*PROCEDURE WILCOXON(N_VARS,N_EQ_S:INT);
VAR SHIFT:REAL;
BEGIN
WRITELN(QCUT); THE WILCOXON SIGNED RANK TEST WILL EVALUATE THE;
WRITELN(QCUT); DIFFERENCES BETWEEN TWO EQUAL LENGTH VARIABLES;
WRITELN(QCUT); THIS PROGRAM SUBTRACTS THE VALUES OF THE VARIABLE;
WRITELN(QCUT); WITH THE SMALLER INDEX FROM THE VALUES OF THE;
WRITELN(QCUT); VARIABLE WITH THE LARGER INDEX TO THE PROGRAM;
WRITELN(QCUT); ALSO ALLOWS ADDING A CONSTANT TO THE HIGHER INDEX;
WRITELN(QCUT); VARIABLES, WHICH ALL GWS FLEXIBILITY IN DATA;
WRITELN(QCUT); ANALYSIS. E.G. IF YOU CAN REJECT THE NULL HYP;
WRITELN(QCUT); E(X)>=E(Y), IMPLYING THAT E(X)<E(Y), WILL YOU;
WRITELN(QCUT); STILL REJECT IF YOU ADD 200 TO EACH VALUE OF X?;
WRITELN(QCUT); THE DEFAULT VALUE OF THE CONSTANT=0;
WRITELN(QCUT); DO YOU WANT TO ADD A CONSTANT TO THE HIGHER INDEX;
WRITELN(QCUT); VARIABLE;
WRITELN(QCUT); REPEAT;
IF YES THEN WRITELN(QCUT); REPEAT CONSTANT; READLN(QIAN,R_P(1,1));
IF YES THEN WRITELN(QCUT); REPEAT CONSTANT; CORRECT?); UNTIL YES;
ELSE WRITELN(QCUT); IS THE LOWER INDEX VARIABLE;
EXCLUDE(N_VARS);
WRITELN(QCUT); H0: E(X)=E(Y), ALT HYP H1: E(X)<E(Y);
WRITELN(QCUT); H0: E(X)>=E(Y), ALT HYP H1: E(X)<E(Y);
WRITELN(QCUT); H0: E(X)<=E(Y), ALT HYP H1: E(X)>E(Y);
WRITELN(QCUT); I_P(1,1)=L_SIG;
R_PARAM(1,2)=I_P,R_P);
WILCOXON(N_VARS,N_EQ_S);
END: (*WILCOXON*)
(*PROCEDURE MCNEMAR(N_EQ_S,N_VARS:INT;EQUAL_S,CELL_CTR:BOOLEAN);
VAR TWO,BEFORE,AFTER,I,J:INT;
S_B,S_A:REAL;
A_SU_B,MID:=FALSE;CH_MIND:BOOLEAN;
PART_VAL(1,1)=0;
WRITELN(QCUT); THIS TEST REQUIRES COUNTS IN FOUR CELLS;
WRITELN(QCUT); TWO DIMENSIONS WITH TWO LEVELS FOR EACH DIMENSION;
WRITELN(QCUT); FOR EXAMPLE, IF BEFORE/AFTER ARE THE DIMENSIONS, AND;
WRITELN(QCUT); SUCCESS/FAILURE THE LEVELS, # MARKS THE CELLS;
WRITELN(QCUT); SUCCESS FAILURE;

```



```

WRITELN(QGLT, ' CORRECT?')
UNTIL YES: (*REPEAT UNTIL*)
I:=0: IF (BEFORE>AFTER) THEN I:=1: PART_VAL(.1, I):=S: B: PART_VAL(.2-1, I):=S: A: VAL(.3+1, I):=XX:
IF A-SUB THEN XX:=-1.0: PART_VAL(.4-1, I):=XX:
W CELLS(63, I):=1.0: PART_VAL(CELIST, I):
PART_VAL(.5, I):=1.0: WRITELN(OP, TWO:5)>(*IN LIEU OF WHICH_VEC*)
IF ELSE WRITELN(OP, AFTER:5, BEFORE:5);
EXCLUD_B(N-EQ-S);
END: (* IF A-VARS>I AND EQUAL_S *)
END: (* IF NOT A-CELL_CTR *)
IF CELL_CTR CR CH_MIND THEN BEGIN
WRITELN(COUT, ' ENTER THE COUNT OF SUCCESS-SUCCESS CELL:');
WRIRST: REACLN(CIN, CELIST(.1, I)); SUCCESS-FAILURE CELL:');
WRITELN(COUT, ' ENTER COUNT OF SUCCESS-FAILURE CELL:');
WRIRST: REACLN(CIN, CELIST(.2, I)); FAILURE-SUCCESS CELL:');
WRITELN(COUT, ' ENTER COUNT OF FAILURE-SUCCESS CELL:');
WRIRST: REACLN(CIN, CELIST(.3, I)); FAILURE-FAILURES CELL:');
WRITELN(COUT, ' ENTER COUNT OF FAILURE-FAILURES CELL:');
WRIRST: REACLN(CIN, CELIST(.4, I)); AFTER:');
WRITELN(COUT, ' BEFORE: SUCCESS FAILURE:');
WRITELN(COUT, ' SUCCESS , CELIST(.1, I):10, CELIST(.2, I):10);
WRITELN(COUT, ' FAILURE , CELIST(.3, I):10, CELIST(.4, I):10);
WRITELN(COUT, ' WRITELN(QOUT, 'CORRECT?');
UNTIL YES:
IF NOT CH_MIND THEN TYP_CEL:=61:
IF CH_MIND THEN TYP_CEL:=62:
W CELLS(TYP_CEL, 2, 2, 4, PART_VAL, CELIST);
END: (* IF CELL_CTR OR CH_MIND *)
N_PARAM:=I: N_I:=0: R_P:=0:
W_PARAM(0, I, I, R_PT);
END: (* MCNEMAR *)
PROCEDURE MCNEMARI;
BEGIN MCNEMAR(0, 2, FALSE, TRUE);
END;
(*-----*)
PROCEDURE E_SIZE(N VARS: INTEGER; VAR SIZE: VECTOR I;
VAR EQUAL_S: BOOLEAN; VAR N_EQ_S: INT);
(*CHECKS TO SEE IF ALL VARIABLES OR TREATMENT HAVE EQUAL NUMBER
OF ELEMENTS: IF SO, REPORTS THE NUMBER OF ELEMENTS*)
VAR I, J, K: INTEGER;

```

```

BEGIN
  EQUAL_S:=TRUE;N_EQ_S:=0;
  IF (N_VARS>1) THEN BEGIN
    FCR I:=2 TC N_VARS DO
      IF (VAR SIZE (I-1))<>VAR SIZE (I) THEN EQUAL_S:=FALSE;
    IF EQUAL_S THEN WRITE (QOUT,THE VARIABLES/TREATMENTS);
    IF NOT EQUAL_S THEN WRITE (QOUT,THE VARIABLES/TREATMENTS);
    IF ARE NOT OF EQUAL SIZE: PAIRED COMPARISON TESTS ARE NOT POSSIBLE;
    IF EQUAL_S THEN N_EQ_S:=VAR_SIZE(I);
  END;
END;
(*PROCEDURE MILKES(IN_V_T:INT);
  VAR Z:R_PAR; I:INT;
  BEGIN
    K:=IN_V_T DIV 2; I_P(.1.):=K;N_I_P:=1;
    CASE IN_V_T OF
      3: BEGIN I:=15;Z(.1.):=0.4255;Z(.2.):=0.2944;Z(.3.):=0.2487;
        Z(.4.):=0.1870;Z(.5.):=0.1630;Z(.6.):=0.1415;
        Z(.7.):=0.1115;Z(.8.):=0.0862;Z(.9.):=0.0697;
        Z(10.):=0.0381;Z(11.):=0.0227;Z(12.):=0.151;Z(13.):=0.0076;END;
      29: BEGIN I:=15;Z(.1.):=0.4291;Z(.2.):=0.2968;Z(.3.):=0.2499;
        Z(.4.):=0.1864;Z(.5.):=0.1616;Z(.6.):=0.1395;
        Z(.7.):=0.1022;Z(.8.):=0.0822;Z(.9.):=0.0650;
        Z(10.):=0.0320;Z(11.):=0.0084;Z(12.):=0.151;Z(13.):=0.0;END;
      28: BEGIN I:=14;Z(.1.):=0.4328;Z(.2.):=0.2952;Z(.3.):=0.2510;
        Z(.4.):=0.1857;Z(.5.):=0.1601;Z(.6.):=0.1372;
        Z(.7.):=0.0965;Z(.8.):=0.0778;Z(.9.):=0.0598;
        Z(10.):=0.0253;Z(11.):=0.0084;Z(12.):=0.151;Z(13.):=0.252;
        Z(14.):=0.4366;Z(15.):=0.3018;Z(16.):=0.1346;
        Z(17.):=0.1848;Z(18.):=0.1584;Z(19.):=0.1346;
        Z(20.):=0.0923;Z(21.):=0.0728;Z(22.):=0.0540;
        Z(23.):=0.4407;Z(24.):=0.3043;Z(25.):=0.2533;
        Z(26.):=0.1836;Z(27.):=0.1563;Z(28.):=0.1316;
        Z(29.):=0.0876;Z(30.):=0.0672;Z(31.):=0.0476;
        Z(32.):=0.0094;Z(33.):=0.0094;END;
      25: BEGIN I:=13;Z(.1.):=0.4450;Z(.2.):=0.3069;Z(.3.):=0.2543;
        Z(.4.):=0.1822;Z(.5.):=0.1539;Z(.6.):=0.1283;
        Z(.7.):=0.0823;Z(.8.):=0.0610;Z(.9.):=0.0403;
        Z(10.):=0.0;Z(11.):=0.0;END;
      24: BEGIN I:=12;Z(.1.):=0.4493;Z(.2.):=0.3098;Z(.3.):=0.2554;
        Z(.4.):=0.1807;Z(.5.):=0.1512;Z(.6.):=0.1245;
        Z(.7.):=0.0957;Z(.8.):=0.0535;Z(.9.):=0.0321;
        Z(10.):=0.0107;Z(11.):=0.0;Z(12.):=0.4542;Z(13.):=0.3126;Z(14.):=0.2563;
        Z(15.):=0.1787;Z(16.):=0.1480;Z(17.):=0.1201;
        Z(18.):=0.2139;Z(19.):=0.122;Z(20.):=0.081;Z(21.):=0.054;
        Z(22.):=0.032;Z(23.):=0.010;Z(24.):=0.000;Z(25.):=0.000;Z(26.):=0.000;
        Z(27.):=0.000;Z(28.):=0.000;Z(29.):=0.000;Z(30.):=0.000;Z(31.):=0.000;
        Z(32.):=0.000;Z(33.):=0.000;Z(34.):=0.000;Z(35.):=0.000;Z(36.):=0.000;
        Z(37.):=0.000;Z(38.):=0.000;Z(39.):=0.000;Z(40.):=0.000;Z(41.):=0.000;
        Z(42.):=0.000;Z(43.):=0.000;Z(44.):=0.000;Z(45.):=0.000;Z(46.):=0.000;
        Z(47.):=0.000;Z(48.):=0.000;Z(49.):=0.000;Z(50.):=0.000;Z(51.):=0.000;
        Z(52.):=0.000;Z(53.):=0.000;Z(54.):=0.000;Z(55.):=0.000;Z(56.):=0.000;
        Z(57.):=0.000;Z(58.):=0.000;Z(59.):=0.000;Z(60.):=0.000;Z(61.):=0.000;
        Z(62.):=0.000;Z(63.):=0.000;Z(64.):=0.000;Z(65.):=0.000;Z(66.):=0.000;
        Z(67.):=0.000;Z(68.):=0.000;Z(69.):=0.000;Z(70.):=0.000;Z(71.):=0.000;
        Z(72.):=0.000;Z(73.):=0.000;Z(74.):=0.000;Z(75.):=0.000;Z(76.):=0.000;
        Z(77.):=0.000;Z(78.):=0.000;Z(79.):=0.000;Z(80.):=0.000;Z(81.):=0.000;
        Z(82.):=0.000;Z(83.):=0.000;Z(84.):=0.000;Z(85.):=0.000;Z(86.):=0.000;
        Z(87.):=0.000;Z(88.):=0.000;Z(89.):=0.000;Z(90.):=0.000;Z(91.):=0.000;
        Z(92.):=0.000;Z(93.):=0.000;Z(94.):=0.000;Z(95.):=0.000;Z(96.):=0.000;
        Z(97.):=0.000;Z(98.):=0.000;Z(99.):=0.000;Z(100.):=0.000;
    END;
  END;
*)

```

```

Z( 8. ):=0.0541;Z( 9. ):=0.0696;Z( 10. ):=0.0459;Z( 11. ):=0.0228;
Z( 12. ):=0.0000;END;
Z( 1. ):=0.4590;Z( 2. ):=0.3156;Z( 3. ):=0.2573;
Z( 4. ):=0.1764;Z( 5. ):=0.1443;Z( 6. ):=0.1150;
Z( 7. ):=0.0918;Z( 8. ):=0.0768;Z( 9. ):=0.0622;END;
Z( 1. ):=0.0464;Z( 2. ):=0.0385;Z( 3. ):=0.0257;
Z( 4. ):=0.0173;Z( 5. ):=0.0139;Z( 6. ):=0.0092;
Z( 7. ):=0.0053;Z( 8. ):=0.0026;Z( 9. ):=0.0000;END;
Z( 1. ):=0.4734;Z( 2. ):=0.3211;Z( 3. ):=0.2565;
Z( 4. ):=0.1686;Z( 5. ):=0.1334;Z( 6. ):=0.1013;
Z( 7. ):=0.0740;Z( 8. ):=0.0514;Z( 9. ):=0.0327;
Z( 10. ):=0.0232;Z( 11. ):=0.0140;Z( 12. ):=0.0093;
Z( 1. ):=0.1941;Z( 2. ):=0.1524;Z( 3. ):=0.1271;
Z( 4. ):=0.1030;Z( 5. ):=0.0886;Z( 6. ):=0.0700;
Z( 7. ):=0.0587;Z( 8. ):=0.0488;Z( 9. ):=0.0400;
Z( 10. ):=0.0333;Z( 11. ):=0.0272;Z( 12. ):=0.0221;
Z( 1. ):=0.1587;Z( 2. ):=0.1163;Z( 3. ):=0.0886;
Z( 4. ):=0.0663;Z( 5. ):=0.0496;Z( 6. ):=0.0373;
Z( 7. ):=0.0290;Z( 8. ):=0.0254;
Z( 9. ):=0.0221;Z( 10. ):=0.0196;Z( 11. ):=0.0175;
Z( 12. ):=0.0152;Z( 1. ):=0.0000;END;
Z( 1. ):=0.5056;Z( 2. ):=0.3505;Z( 3. ):=0.2521;
Z( 4. ):=0.1447;Z( 5. ):=0.1005;Z( 6. ):=0.0593;
Z( 7. ):=0.0330;Z( 8. ):=0.0249;
Z( 9. ):=0.0188;Z( 10. ):=0.0135;Z( 11. ):=0.0095;
Z( 12. ):=0.0078;Z( 1. ):=0.0000;END;
Z( 1. ):=0.5251;Z( 2. ):=0.3318;Z( 3. ):=0.2460;
Z( 4. ):=0.1240;Z( 5. ):=0.0727;Z( 6. ):=0.0240;
Z( 7. ):=0.0325;Z( 8. ):=0.0242;
Z( 9. ):=0.0199;Z( 10. ):=0.0153;Z( 11. ):=0.0112;
Z( 12. ):=0.0092;Z( 1. ):=0.0922;Z( 2. ):=0.0303;
Z( 3. ):=0.0226;Z( 4. ):=0.0169;Z( 5. ):=0.0122;
Z( 6. ):=0.0095;Z( 7. ):=0.0073;Z( 8. ):=0.0055;
Z( 9. ):=0.0041;Z( 10. ):=0.0033;Z( 11. ):=0.0026;
Z( 12. ):=0.0021;Z( 1. ):=0.6052;Z( 2. ):=0.3244;
Z( 3. ):=0.1976;Z( 4. ):=0.1547;Z( 5. ):=0.1174;
Z( 6. ):=0.0888;Z( 7. ):=0.0652;Z( 8. ):=0.0473;
Z( 9. ):=0.0355;Z( 10. ):=0.0255;Z( 11. ):=0.0174;
Z( 12. ):=0.0100;END;
Z( 1. ):=0.6233;Z( 2. ):=0.3031;Z( 3. ):=0.1401;
Z( 4. ):=0.0643;Z( 5. ):=0.0280;Z( 6. ):=0.0075;
Z( 7. ):=0.0646;Z( 8. ):=0.2413;Z( 9. ):=0.0875;
Z( 10. ):=0.6872;Z( 11. ):=0.1667;END;
Z( 1. ):=0.7071;Z( 2. ):=0.0000;END;
Z( 1. ):=0.7071;END;
END; (*CASE*)

```



```

N_R_P:=I;
N_PARAM(N_I,F^A_R_P,I_P,Z);
END: (*WILKES*)
-----*)
PROCEDURE LILLIE;
VAR I:INT;
BEGIN
WRITELN(QCUT,'CHOOSE BETWEEN THE FOLLOWING DISTRIBUTIONS:');
REPEAT
  RST:=0;
  WRITELN(QCUT,' 1: NORMAL');
  WRITELN(QCUT,' 2: EXPONENTIAL');
  READLN(QIN,I);
  UNTIL (I IN (1,2));
  H_A:=A:NULL; H_B:=B:NULL; H_C:=C:NULL; H_P:=P(1):=0;
  H_C:=C:NULL; H_P:=P(1):=0;
  I_T_P(2):=FYP(A,H_B,H_C,347);
  W_PARAM(2,C,I_P,R_P);
END: (*LILLIE*)
-----*)
PROCEDURE UPAR(VAR R_P:R_PAR);
VAR X,Y:REAL;
BEGIN
  REPEAT
    WRITELN(QCUT,'ENTER LOWER LIMIT A AND UPPER LIMIT B');
    RST:=READ(QIN,X,Y);
    WRITELN(QCUT,'A=',X:10:3,' B=',Y:10:3,' CCRRECT?');
  UNTIL Y=0;
  R_P(1):=X; R_P(2):=Y; N_R_P:=2;
END: (*UPAR*)
-----*)
PROCEDURE WPAR(VAR R_P:R_PAR);
VAR X,Y:REAL;
BEGIN
  REPEAT
    WRITELN(QCUT,'WITH CDF OF FORM:');
    WRITELN(QCUT,'F(X)=1 - EXP(-(LAMBDA*X)**ALPHA)');
    WRITELN(QCUT,'ENTER SHAPE PARAMETER ALPHA; AND');
    WRITELN(QCUT,'RATE PARAMETER LAMBDA:');
    RST:=READ(QIN,X,Y);
    WRITELN(QCUT,'ALPHA=',X:10:3,' LAMBDA=',Y:10:3,' CORRECT?');
  UNTIL Y=0;
  R_P(1):=X; R_P(2):=Y; N_R_P:=2;
END: (*WPAR*)
-----*)
PROCEDURE ERPAR(VAR R_P:R_PAR; VAR I_P:I_PAR);
VAR Y:REAL; I:INT;

```

```

BEGIN REPEAT
WRITELN(QCUT, 'ENTER INTEGER SHAPE PARAMETER N, AND');
WRITELN(QCUT, 'RATE PARAMETER LAMBDA:');
RST: READ(QIN, I, Y);
WRITELN(QCUT, I, Y);
UNTIL YES;
R_P(1.0):=Y; I_P(3.1):=I; N_R_P:=1; N_I_P:=2;
END;
(*-----*)
PROCEDURE NPAR(VAR R_F:R_PAR);
VAR X,Y:REAL;
BEGIN REPEAT
WRITELN(QCUT, 'ENTER PARAMETER MU--E(X)--AND');
WRITELN(QCUT, 'SIGMA--VAR(X):');
RST: READ(QIN, X, Y);
WRITELN(QCUT, MU:=1, X:10:3, SIGMA:=, Y:10:3, 'CORRECT?');
UNTIL YES;
R_P(1.0):=X; R_P(2.0):=Y; N_R_P:=2;
END;
(*-----*)
PROCEDURE EXPAR(VAR R_P:R_PAR);
VAR X,Y:REAL;
BEGIN REPEAT
WRITELN(QCUT, 'ENTER RECIPROCAL OF E(X); I.E.,');
WRITELN(QCUT, 'RATE PARAMETER LAMBDA:');
RST: READ(QIN, X);
WRITELN(QCUT, LAMBDA:=, X:10:3, 'CORRECT?');
UNTIL YES;
R_P(1.0):=X; N_R_P:=1;
END;
(*-----*)
PROCEDURE KOLMCG;
VAR I:INT;
BEGIN WRITELN(QCUT, 'CHOOSE AMONG THE FOLLOWING DISTRIBUTIONS:');
REPEAT
WRITELN(QCUT, '1. NORMAL');
WRITELN(QCUT, '2. EXPONENTIAL');
WRITELN(QCUT, '3. UNIFORM <CONTINUOUS>');
WRITELN(QCUT, '4. ERLANG <GAMMA WITH INTEGER SHAPE PARAM>');
WRITELN(QCUT, '5. WEIBULL');
RST: WRITELN(QCUT, I, 'ENTER CHOICE 1-5');
READLN(QIN, I);
UNTIL ((YES) AND (I IN (1..5)));
CASE I OF

```

```

1: NPAR(R_P);
2: EXPAR(R_P);
3: UPPAR(R_P);
4: EPPAR(R_P);
5: WPAR(R_P);
END; (*CASE*)
I_P(1):=1;
H_A:=: A: NULL HYPOTHESIS CF FORM: F*(X)=S(X) ;
H_B:=: B: NULL F: F*(X)>S(X), F*(X) IS SPEC THEOR DIST ;
H_C:=: C: NULL F: F*(X)<S(X) AND S(X) IS EMP CDF ;
H_P(2):=FYP(F_A,F_B,H_C(347));
IF (I=4) THEN N_I_P:=2;
IF (I=2) CR (I=4) THEN N_R_P:=1 ELSE N_R_P:=2;
N_PARAM(N_I_P,N_R_P,I_P,R_P);
END; (*KCLMOG*)
-----*
PROCEDURE GDFIT_W(VAR R_P:R_PAR;VAR I_P:I_PAR);
VAR X,Y:REAL;
BEGIN
N_R_P:=0;R_P(1):=0;
WRITELN(QOUT,'WITH CDF OF FORM:');
WRITELN(QOUT,'F(X)=1 - EXP(-(LAMBDA*X)**ALPHA)');
I_P(2):=0;
REPEAT
WRITELN(QOUT,'ENTER SHAPE PARAMETER ALPHA, AND');
WRITELN(QOUT,'RATE PARAMETER LAMBDA:');
RST: READ(QIN,X,Y); WRITELN(QOUT,
'ALPHA=',X,'LAMBDA=',Y,'CORRECT?');
UNTIL YES;
R_P(1):=X;R_P(2):=Y;N_R_P:=2;
END; (*GDFIT_W*)
-----*
PROCEDURE GDFIT_FF(VAR I_P:I_PAR;VAR R_P:R_PAR);
VAR BETA,N,I,J:INT;SPEC:BOOLEAN;ALPHA:REAL;X:REAL;
BEGIN
WRITELN(QOUT,'N IS THE INTEGER SHAPE PARAMETER, AND LAMBDA');
WRITELN(QOUT,'IS THE RATE OR SCALE PARAMETER, YOU MUST SPECIFY');
WRITELN(QOUT,'N; YOU CAN CHOOSE TO ENTER A LAMBDA PARAMETER');
WRITELN(QOUT,'OR ALLOW THE PROGRAM TO ESTIMATE IT');
WRITELN(QOUT,'DC YOU WISH TO ENTER A LAMBDA PARAMETER?');
IF YES THEN
RST: I_P(2):=0;
REPEAT
WRITELN(QOUT,'ENTER LAMBDA');
WRITELN(QOUT,'LAMBDA=',X,'CORRECT?');
R_P(1):=X;
UNTIL YES;
R_P(2):=1;
ELSE
REPEAT
WRITELN(QOUT,'ENTER SHAPE PARAMETER N, INTEGER > 1');

```

```

READLN(CIN,N);WRITELN(QOUT,'N=',N,5,' CORRECT?');
UNTIL ((YES) AND (N > 1));
I_P(3):=N;
END; (*GDFIT_ER*)
-----*
PROCEDURE GD_FIT(N_VARS: INT);
VAR
  I,J: INTEGER;
BEGIN
  FOR I:=1 TO 3 DO BEGIN I_P(I):=0; R_P(I):=0.0; END; (*FOR I*)
  WRITELN(QOUT); CHOOSE AMONG FIVE THEORETICAL DISTRIBUTIONS:
  WRITELN(QOUT); 1: NORMAL
  WRITELN(QOUT); 2: UNIFORM
  WRITELN(QOUT); 3: EXPONENTIAL
  WRITELN(QOUT); 4: ERLANG (GAMMA WITH INTEGER BETA PARAMETER);
  WRITELN(QOUT); 5: WEIBULL;
  REPEAT
    WRITELN(QOUT);
    WRITELN(QIN,I_P(1..5)); WHICH DISTRIBUTION (1-5);
    READLN(QIN,I_P(1..5)); UNTIL (I_P(1..5) IN (.1..5.));
  CASE (I_P(1..5)) OF
    1: GDFIT_U(N,I_P,R_P);
    2: GDFIT_U(R_P);
    3: GDFIT_EX(I_P,R_P);
    4: GDFIT_ER(I_P,R_P);
    5: GDFIT_W(R_P,I_P);
  END; (*CASE*)
  R_P(3):=L_SIG;
  W_PARAM(3,3,I_P,R_P);
END; (*GD_FIT*)
-----*
PROCEDURE CHECK_N(N_VARS: INT; VAR N_TEST IN V T: INT; N_ELM S: VE C TOR I);
(* THIS PROCEDURE SCREENS TO MAKE SURE THAT THE GOODNESS OF FIT
TEST GETS AT LEAST 15 DATA ELEMENTS, AND THAT KOLMOGOROV, LILLIEFORS,
AND SHAPIRO-WILKES TEST GET NO MORE THAN 30 *)
VAR ONE: INT; G_V: ECOLAN;
BEGIN
  ONE:=1;
  IF (N_VARS=1) THEN N_V_T:=1 ELSE
  BEGIN
    WRITELN(QOUT); OF THE N_VARS:3, VARS IN THE DATA SET:);
    WRITELN(QOUT); WHICH ONE IS USED IN THIS TEST?);
    REPEAT
      WRITELN(QOUT); ENTER INDEX 1 - , N_VARS:3);
      RST: READLN(QIN,N_V_T); G_V:=FALSE;
      IF (N_V_T IN (1..N_VARS.T)) THEN G_V:=TRUE;
      IF G_V THEN BEGIN
        WRITELN(QOUT); INDEX=:N_V_T:3, CORRECT?);
        G_V:=YES; END; (*IF G_V*)
      UNTIL G_V;
    END;
  END;
END;

```

```

END: (*ELSE N_VARS>1*)
IN_V_T:=N_ELPS(.N_V_T.);
IF (IN_TEST=8) AND (IN_V_T<15) THEN
BEGIN
WRITELN(COUT,'BECAUSE THE NUMBER OF DATA ELEMENTS IN THE ');
WRITELN(COUT,'CHOSEN VARIABLE, IN_V_T, IS < 15, THE ');
WRITELN(COUT,'THE PROGRAM CANNOT PERFORM THE CHISQUARED ');
WRITELN(COUT,'GOODNESS OF FIT TEST. CHOOSE AMONG: ');
REPEAT
WRITELN(COUT,'20 KOLMOGOROV: USER SPECIFIED PARAMETERS, ');
WRITELN(COUT,'21 NORMAL EXPONENTIAL, UNIFORM, ERLANG, WEIBULL, ');
WRITELN(COUT,'22 LILLIEFORS: PROGRAM COMPLETED PARAMETERS, ');
WRITELN(COUT,'23 NORMAL EXPONENTIAL, ');
WRITELN(COUT,'24 SHAPIRO-WILKES: PROGRAM COMPUTED ');
WRITELN(COUT,'25 PARAMETERS, NORMAL ONLY, ');
WRITELN(COUT,'26 ENTER CHOICE, 20-22, ');
READLN(CIN,N_TEST);
IF (N_TEST IN (.20..22.)) THEN G_V:=TRUE;
IF G_V THEN BEGIN
WRITELN(COUT,N_TEST:5,' CORRECT? ');
G_V:=YES; END; (* IF G_V *)
UNTIL G_V;
END; (* IF N_TEST=8 *)
IF (IN_TEST IN (.20,21,22.)) AND (IN_V_T > 30) THEN
BEGIN
WRITELN(COUT,'BECAUSE THE NUMBER OF DATA ELEMENTS, IN_V_T:4);
WRITELN(COUT,'IS GREATER THE 30, THE PRGGRAM WILL ');
WRITELN(COUT,'AUTOMATICALLY OFFER YOU THE CHISQUARE ');
WRITELN(COUT,'GOODNESS OF FIT TEST, ');
N_TEST:=8;
END; (* IN_V_T > 30 *)
WRITELN(COUT,N_TEST);
WRITELN(CP,ONE:5);WRITELN(OP,N_V_T:5); NONE_EX;
END; (* CHECK_N *)
-----*)
PROCEDURE CHECK_FILE (VAR N_EQ,S,N IN F,N_OUT,F,N_VARS:INTEGER;
VAR RIGHT_FLAG,QUAL,S:BOOLEAN;VAR N_ELMS:VECTOR);
VAR DATA_NAME TO DETERMINE IF IT IS THE CORRECT FILE.
(* WRITES DATA NAME TO DETERMINE IF IT IS THE CORRECT FILE.
IF SC, CHECKS IF VARIABLES OR TREATMENTS ARE OF EQUAL LENGTH*)
VAR I,J,K:INTEGER;
BEGIN
OUTSET:=(.11..16.);
I:=C;
CASE (N IN F) OF
SET(A,NAME = LOCHI,A,A');
REPEAT I:=I+1; READ(A,NAME(.));
UNTIL (I=50) OR (F_NAME(I)=.); OR EOLN(A);
REAC (A,N_VARS); FOR I:=1 TO N_VARS
DO READ(A,N_ELMS(.I.));END;

```

```

130: BEGIN REPEAT (I:=1) ; NAME = LOC(FI, B, 'A') ;
      UNTIL (N_VARS) ; OR (F_NAME(I, I)) ; OR EOLN(B) ;
      READ (B_VARS) ; FOR I:=1 TO N_VARS DO READ(B, N_ELMS(.I.)) ; END ;

131: BEGIN REPEAT (I:=1) ; NAME = LOC(CI, 'A') ;
      UNTIL (N_VARS) ; OR (F_NAME(I, I)) ; OR EOLN(C) ;
      READ (C_VARS) ; FOR I:=1 TO N_VARS DO READ(C, N_ELMS(.I.)) ; END ;

132: BEGIN REPEAT (I:=1) ; NAME = LOC(DI, 'A') ;
      UNTIL (N_VARS) ; OR (F_NAME(I, I)) ; OR EOLN(D) ;
      READ (D_VARS) ; FOR I:=1 TO N_VARS DO READ(D, N_ELMS(.I.)) ; END ;

133: BEGIN REPEAT (I:=1) ; NAME = LOC(EI, 'A') ;
      UNTIL (N_VARS) ; OR (F_NAME(I, I)) ; OR EOLN(E) ;
      READ (E_VARS) ; FOR I:=1 TO N_VARS DO READ(E, N_ELMS(.I.)) ; END ;

134: BEGIN REPEAT (I:=1) ; NAME = LOC(FI, 'A') ;
      UNTIL (N_VARS) ; OR (F_NAME(I, I)) ; OR EOLN(F) ;
      READ (F_VARS) ; FOR I:=1 TO N_VARS DO READ(F, N_ELMS(.I.)) ; END ;

END ; (*CASE*)
WRITELN(COUT) ; FOR I:=1 TO 49 DO WRITE(QOUT, F_NAME(.I.)) ;
WRITELN(COUT) ; IS THIS THE DATA SET YOU WANTED? ;
IF (YES) THEN RIGHT_F:=TRUE ELSE RIGHT_F:=FALSE ;
IF (RIGHT_F) THEN
  E_SIZE:=TA_VARS, N_ELMS, EQUAL_S, N_EQ_S ; (*J=FILETYPE OF DATA*)
  IF (RIGHT_F) THEN FILE_ID(INF, N_INF) ;
END ; (*CHECK_FILE*)

(*PROCEDURE MENU(VAR STATE:TEST_N;N_VARS:INT;EQUAL_S:BOOLEAN);
BEGIN
STATE:=1 ;
WRITELN(QOUT) ; PICK AMONG THE FOLLOWING NON-PARAMETRIC TESTS : ;
WRITELN(QOUT) ; 1. B. NOMIAL ;
WRITELN(QOUT) ; 2. B. QUANTILE ;
WRITELN(QOUT) ; 3. COX-STUART ; TEST FOR TREND <IF 2 VARS, PAIR> ;
IF (N_VARS>1) THEN BEGIN STATE:=STATE+(.4, 13, 14, 15, 16, 23, 24.)) ;
  WRITELN(QOUT) ; 4. MANN-WHITNEY ; END ;
  IF ((N_VARS>1) AND EQUAL_S) THEN
  BEGIN
  STATE:=STATE+(5, 7.))+(.9..12.)) ;
  WRITELN(QOUT) ; 5. SIGN <PAIR> ;
  WRITELN(QOUT) ; 6. MC NEMAR <PAIR> ;
  WRITELN(QOUT) ; 7. R*C CONTINGENCY TABLE <PAIR> ;
  WRITELN(QOUT) ; 8. (CH-SQUARED TEST FOR INDEPENDENCE) ;
  END ;
WRITELN(QOUT) ; 9. GOODNESS OF FIT, N>14 (CHI-SQUARE) ;
IF ((N_VARS>1) AND EQUAL_S) THEN BEGIN

```

```

WRITELN(SCUT, 9, WILCOXGN SIGNED RANK <PAIR>);
WRITELN(SCUT, 10, RANK CORRELATION (KENDALL & SPEARMAN) <PAIR>);
WRITELN(SCUT, 11, NON-PARAMETRIC LINEAR REGRESSION <PAIR>);
WRITELN(SCUT, 12, NON-PARAMETRIC MONOTONE REGRESSION <PAIR>);
END;
IF (N_VARS > 1) THEN BEGIN
  WRITELN(SCUT, 13, MEDIAN);
  WRITELN(SCUT, 14, KRUSKAL/WALLIS);
  WRITELN(SCUT, 15, SQUARED RANK TEST FOR EQUAL VARIANCE);
  WRITELN(SCUT, 16, MARTLEY TEST FOR EQUAL VARIANCE);
  WRITELN(SCUT, 17, FREIDMAN OR DURBIN STATE=(1, 2, 24));
  WRITELN(SCUT, 18, QUADE <BLOCKED>);
  WRITELN(SCUT, 19, COCHRAN <BLOCKED>);
  WRITELN(SCUT, 20, KOLMOGOROV USER SPECIFIED PARAMETERS N < 31);
  WRITELN(SCUT, 21, NORMAL EXPONENTIAL, UNIFORM, ERLANG, WEIBULL);
  WRITELN(SCUT, 22, SHAPIRO-WILKES PROGRAM COMPUTED PARAMETERS);
  WRITELN(SCUT, 23, SMIRNOV (2-SAMPLE));
  WRITELN(SCUT, 24, CRAMER-VON MISES (2-SAMPLE));
END;
IF (N_VARS > 1) THEN BEGIN
  WRITELN(SCUT, 25, CRAMER-VON MISES);
  WRITELN(SCUT, 26, CRAMER-VON MISES (2-SAMPLE));
END;
END;
(*MENU*)
PROCEDURE WHICH TEST(N_VARS:INT; EQUAL_S:BOOLEAN; N_EQ_S:INT;
  N_ELMS:VECTO; I,J,K:INTEGER;
  VAR N_PARAMS:N_TEST; STATE:TEST_N);
BEGIN
  REPEAT (STATE, N_VARS, EQUAL_S);
  WRITELN(SCUT, ENTER # OF CHOSEN TEST, OR 0 TO SEE LIST AGAIN);
  READ(QIN, N_TEST);
  UNTIL N_TEST IN STATE;
  IF (N_TEST IN (8, 20, 21, 22)) THEN
    CASE N_TEST OF
      1: BINGM(N_VARS, FALSE);
      2: QUANTILE(N_VARS, FALSE);
      3: COXSU(N_VARS, N_EQ_S, EQUAL_S);
      4: MANNWIT(N_VARS);
      5: SIGNTEST(N_VARS, N_EQ_S);
      6: MCNEMAR(N_VARS, N_EQ_S, EQUAL_S, FALSE);
      7: RC_FIT(N_VARS, N_EQ_S, FALSE);
      8: GD_FIT(N_VARS, N_EQ_S);
      9: WILCOX(N_VARS, N_EQ_S);
      10: CORREL8(N_VARS, N_EQ_S);
    END;
  END;
END;

```

```

11: REGRET(N_VARS,N_EQ,S);
12: MONOPEG(N_VARS,N_EQ,S);
13: MEDIAN(N_VARS);
14: KRUSKAL(N_VARS);
15: SQUARANK(N_VARS);
16: HARTLEY(N_VARS,N_EQ,S);
17: FRIECCMAN(N_VARS,N_EQ,S);
18: QUALE(N_VARS,N_EQ,S);
19: COCHRAN(N_VARS,N_EQ,S,FALSE);
20: KOLMCG;
21: LILLIE;
22: WILKECV(N_VARS);
23: SMIRACV(N_VARS);
24: CRAPVCNM(N_VARS);

```

```

END; (*WHICH_TEST*)
-----*)

```

```

PROCEDURE NAME(VAR F_NAME:STRING;
VAR V_NAME:PAIR; VAR N_VARS:INTEGER);

```

```

VAR I,J,NV,ULL:INTEGER;
BLANK:CHAR;

```

```

BEGIN
FOR I:=1 TO 45 DO F_NAME(I,1):=' ' ; IF NAME(,50):=' ' ;
WRITELN(QOUT,'ENTER NAME OF DATA SET'); I:=I+1; RST;

```

```

REPEAT
I:=I+1; REAC(QIN,F_NAME(I,1));
UNTIL (EOF(QIN) OR EOLN(QIN) OR (I=49));

```

```

REPEAT
WRITELN(QOUT,' ENTER NUMBER--IN DIGITS,F,G;3---');
WRITELN(QOUT,' OF VARIABLES OR TREATMENTS');
RST; REACLN(QIN,N_VARS);
IF (N_VARS>15) THEN WRITELN(QOUT,
' MAXIMUM 15 VARIABLES OR TREATMENTS');

```

```

UNTIL (N_VARS<=15);
FOR I:=1 TO (N_VARS+2) DO
FOR J:=1 TO 50 DO V_NAME(I,J):=' ' ;
FOR I:=1 TO N_VARS DO
BEGIN

```

```

J:=1; WRITELN(QOUT,'ENTER NAME OF VARIABLE NR ',I);
REPEAT

```

```

J:=J+1; READ(QIN,V_NAME(I,J));
UNTIL (EOF(QIN) OR EOLN(QIN) OR (J=50));

```

```

END; (* NAME *)

```

```

-----*)
PROCEDURE E_BY_VAR(VAR LIST:VECTOR; VAR N_ELMS:VECTOR; VAR N_VARS,
N_EQ,S:INT; VAR EQUAL_S:BOCLEAN);

```

```

VAR NV,ULL,LL,T_VARS,FILE_TYP,CH_CNE,J,I:INT;

```



```

BEGIN
(* THIS PROCEDURE READS AND WRITES DATA BY VARIABLES *)
WRITELN(QOUT, ' WHEN ENTERING DATA ELEMENTS <1, START WITH 0. ');
LIST_C:=0; VAR_C:=0; LIST_PTR(0):=1;
REPEAT(*UNTIL NOT RE-ENTER ALL DATA*)
FOR NV:=1 TO N_VARS DO
BEGIN
REPEAT(*UNTIL THE VARIABLE IS GOOD*)
VAR_C:=0; LIST_C:=LIST_PTR((NV-1))-1; GOOD_VAR:=TRUE;
REPEAT(*UNTIL ALL OF VAR IS ENTERED*)
WRITELN(QOUT, ' ENTER VALUES FOR VARIABLE ', NV);
LIST_C:=LIST_C+1;
VAR_C:=VAR_C+1;
READ(QIN, LIST_C);
WHILE (NOT EDLN(QIN) DC);
BEGIN
LIST_C:=LIST_C+1;
VAR_C:=VAR_C+1;
READ(QIN, LIST_C);
END;
WRITELN(QOUT, ' ADD MORE ELEM TO SAME VARIABLE? ');
WRITELN(QOUT, ' OR CHECK THIS VAR? OR GO TO NEXT? ');
REPEAT
WRITELN(QOUT, ' ');
RS( ' (CH_ONE=129) OR (CH_ONE=131) OR (CH_ONE=135) );
UNTIL (CH_ONE<129);
IF (CH_ONE=131) THEN
BEGIN
WRITELN(QOUT, ' NUMBER OF ELEM ENTERED ', LIST_C);
WRITELN(QOUT, ' NUMBER OF ELEM IN THIS VAR ', VAR_C);
WRITELN(QOUT, ' ');
FOR I:=LIST_PTR((NV-1)) TO LIST_C DO
WRITE(QOUT, LIST(I)):10:3; WRITELN(QOUT, ' ');
WRITELN(QOUT, ' WANT TO RE-ENTER THIS VARIABLE? ');
IF (YES) THEN GOOD_VAR:=FALSE;
END;
N_ELEM((NV)):=VAR_C;
LIST_PTR((NV)):=LIST_C+1;
UNTIL GOOD_VAR;
END;
SUMMARY CF DATA INPUT STEP *)
WRITELN(QOUT, ' YOU HAVE ENTERED ', N_VARS:3, ' VARIABLES, WITH THE ',
FOLLOWING NUMBER OF ELEMENTS IN EACH');
WRITELN(QOUT, ' VAR NR
FOR I:=1 TO N_VARS DO

```

```

WRITELN(COUT, I, N_ELMS(.I.));
WRITELN(COUT, ' DO YOU WISH TO RE-ENTER THE ENTIRE DATA?');
UNTIL NOT(VES); (* RE-ENTER ALL DATA *)
PREPARE TC SEND DATA TO FILE *)
N_ELMS(.O.) := N_VARS;
FOR I := (1+N_VARS) TO 15 DO N_ELMS(.I.) := 0;
END; (* E_BY_VARS *)
(*-----*)
PROCEDURE E_PAIRS (VAR LIST:VECTOR; VAR N_ELMS:VECTCRI;
VAR N_VARS, N_EQ_S:INT; VAR EQUAL_S:BOOLEAN);
BEGIN
REPEAT RST:WRITELN(COUT, ' HOW MANY PAIRS WILL YOU ENTER?');
READLN(QIN, N_PAIRS); WRITELN(COUT, N_PAIRS, ' CORRECT?');
UNTIL YES; N_PAIRS := N_PAIRS;
FOR I := 1 TO N_PAIRS DO BEGIN
RST:WRITELN(COUT, ' ENTER VALUES FOR PAIR', I, ':');
REALN(QIN, LIST(.I.)); LIST(.I+N_PAIRS)); END; (* FOR I := *)
WRITELN(COUT, ' DO YOU WANT TO CHECK FOR POSSIBLE ERRORS?');
IF YES THEN FOR I := 1 TO N_PAIRS DO
WRITELN(COUT, LIST(.I.), '10:3, LIST(.I+N_PAIRS), '10:3, I:5);
IF YES THEN FOR I := 1 TO N_PAIRS DO
WRITELN(COUT, ' DO YOU WANT TO MAKE A CORRECTION?');
IF YES THEN FOR I := 1 TO N_PAIRS DO
WRITELN(COUT, ' ENTER INDEX OF PAIR TO BE CORRECTED');
RST:READLN(QIN, J); RST:
WRITELN(COUT, ' CURRENT PAIR', J, ':5, 'VALUES:', LIST(.J.), '10:3,
LIST(.I+N_PAIRS), '10:3);
WRITELN(COUT, ' ENTER CORRECT VALUES');
REALN(QIN, LIST(.J.)); LIST(.J+N_PAIRS));
WRITELN(COUT, ' DO YOU WANT TO MAKE ANOTHER CORRECTION?');
UNTIL NOT(YES);
FOR I := 2 TO 16 DO N_ELMS(.I.) := 0;
N_VARS := 2; N_ELMS(.O.) := 2;
EQUAL_S := TRUE;
E_PAIRS;
END; (* E_PAIRS *)
(*-----*)
PROCEDURE NEW_DATA (VAR N_EQ_S, N_VARS, FILE_TYP:INT;
VAR EQUAL_S:BOOLEAN; VAR N_ELMS:VECTCRI);
VAR NV, UL, LL, TT, VARS, CH_CNE, I, J:INTEGER;
BLANK:CHAR;
BEGIN
BLANK := ' '; LIST C := 0;
FOR I := 1 TO 30 DO LIST(.I.) := 0.0;
RST:
NAME R (F_NAME, V_NAME, N_VARS);
REPEAT

```

```

WRITELN(QOUT, ' ENTER FILETYPE (A-F) TO WHICH DATA GOES');
RS := READLN(QIN, CH1); FILE TYP := ORD(CH1);
IF (FILE TYP IN (129, 134, 135)) THEN K := FILE TYP - 128;
IF (N_VARS < 2) THEN BEGIN
  BY_VAR(LIST, N_ELMS, N_VARS, A_EQ_S, EQUAL_S);
  UNTIL (N_VARS < 2) THEN BEGIN
    VARIABLES HAVE AN EQUAL NUMBER OF ' ';
    WRITELN(QOUT, ' IF BOTH VARIABLES THERE ARE TWO WAYS TO ENTER DATA: ');
    WRITELN(QOUT, ' ELEMENTS, THEN ENTER ALL DATA OF VARIABLE 1, THEN ALL OF 2, ');
    WRITELN(QOUT, ' (V) ENTER PAIRED DATA VALUES TOGETHER;');
    CR (P) ENTER PAIRED DATA VALUES TOGETHER;');
  END;
  IF THE VARIABLES ARE NOT OF EQUAL SIZE:');
  REPEAT UNTIL (YCU MUST CHOOSE THE FORMER (V));');
  WRITELN(QOUT, ' ENTER P FOR BY PAIR GR V FOR BY VARIABLE/TREATMENT');
  READLN(QIN, H); J := ORD(H); UNTIL (J IN (151, 165, J));
  IF (J = 151) THEN E_PAIRS(LIST, N_ELMS, N_VARS, N_REC_S, EQUAL_S);
  IF (J = 165) THEN E_BY_VAR(LIST, N_ELMS, N_VARS, N_REC_S, EQUAL_S);
  N_VARS := N_VARS (< 2*);
  END; (* IF N_VARS < 2*);
  LIST_C := 0;
  IF (N_VARS = 1) THEN LIST_C := N_ELMS(1, 1);
  IF (N_VARS > 1) THEN
    FOR J := 1 TO (N_VARS) DO LIST_C := LIST_C + N_ELMS(J, J);
  CASE (FILE TYP) OF
    129: BEGIN
      WRITE(A, NAME = LOCHI, A, RECFM=F, LRECL=80);
      FOR I := 1 TO 50 DO WRITE(A, F_NAME(1, 1)); WRITELN(A);
      FOR I := 0 TO 15 DO WRITE(A, N_ELMS(1, 1):5); WRITELN(A);
      FOR J := 1 TO LIST_C DO
        BEGIN
          WRITE(A, LIST(J, J):10:3);
          IF ((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(A);
        END;
      FOR I := 1 TO (N_VARS+1) DO
        BEGIN
          FOR J := 1 TO 50 DO WRITE(A, V_NAME(1, J, J));
          WRITELN(A); END;
        END;
      BEGIN
        FEWRITE(B, NAME = LOCHI, B, A, RECFM=F, LRECL=80);
        FOR I := 1 TO 50 DO WRITE(B, F_NAME(1, 1)); WRITELN(B);
        FOR J := 0 TO 15 DO WRITE(B, N_ELMS(1, 1):5); WRITELN(B);
        FOR J := 1 TO LIST_C DO
          BEGIN
            WRITE(B, LIST(J, J):10:3);
            IF ((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(B);
          END;
        FOR I := 1 TO (N_VARS+1) DO
          BEGIN
            FOR J := 1 TO 50 DO WRITE(B, V_NAME(1, J, J));
            WRITELN(B); END;
          END;

```

```

131: BEGIN REWRITE(C,'NAME = LOCHI.C.A,RECFM=F,LRECL=80');
FOR I:=1 TO 50 DO WRITE(C,F_NAME(.I.)); WRITELN(C);
FOR J:=0 TO 15 DO WRITE(C,N_ELMS(.I.):5); WRITELN(C);
BEGIN
WRITE(C,LIST(J):10:3);
IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(C);
END;
FOR I:=1 TO (N_VARS+1) DO BEGIN
FOR J:=1 TO 50 DO WRITE(C,V_NAME(.I.,J.));
WRITELN(C);END;
END;
132: BEGIN REWRITE(D,'NAME = LOCHI.D.A,RECFM=F,LRECL=80');
FOR I:=1 TO 50 DO WRITE(D,F_NAME(.I.)); WRITELN(D);
FOR J:=0 TO 15 DO WRITE(D,N_ELMS(.I.):5); WRITELN(D);
BEGIN
WRITE(D,LIST(J):10:3);
IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(D);
END;
FOR I:=1 TO (N_VARS+1) DO BEGIN
FOR J:=1 TO 50 DO WRITE(D,V_NAME(.I.,J.));
WRITELN(D);END;
END;
133: BEGIN REWRITE(E,'NAME = LOCHI.E.A,RECFM=F,LRECL=80');
FOR I:=1 TO 50 DO WRITE(E,F_NAME(.I.)); WRITELN(E);
FOR J:=0 TO 15 DO WRITE(E,N_ELMS(.I.):5); WRITELN(E);
BEGIN
WRITE(E,LIST(J):10:3);
IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(E);
END;
FOR I:=1 TO (N_VARS+1) DO BEGIN
FOR J:=1 TO 50 DO WRITE(E,V_NAME(.I.,J.));
WRITELN(E);END;
END;
134: BEGIN REWRITE(F,'NAME = LOCHI.F.A,RECFM=F,LRECL=80');
FOR I:=1 TO 50 DO WRITE(F,F_NAME(.I.)); WRITELN(F);
FOR J:=0 TO 15 DO WRITE(F,N_ELMS(.I.):5); WRITELN(F);
BEGIN
WRITE(F,LIST(J):10:3);
IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(F);
END;
FOR I:=1 TO (N_VARS+1) DO BEGIN
FOR J:=1 TO 50 DO WRITE(F,V_NAME(.I.,J.));
WRITELN(F);END;
END;

```

```

END;
WRITELN(QOUT, 'NOW, DO A TEST WITH THIS DATA SET?');
IF FILE IC(N,F) IN F;
  WHICH_TEST(N_VARS, EQUAL_S, N_EQ_S, N_ELMS);
END; (*NEW_DATA*)
-----*)
(*PROCEDURE CELL TEST;
VAR I,J,K,N_TEST,N_OUT_F:INTEGER;
BEGIN
  IN_F:=138;
  _ID(N,F);
  FILE NAME, V_NAME, N_VARS);
  NAME:=('6,7,19');
  STATE=(1,6,7,19);
  WRITELN(QOUT, 'CHOOSE AMONG THE FOLLOWING TESTS:');
  WRITELN(QOUT, '1. BINOMIAL; 6. MCNEMAR; 7. R*C CONTINGENCY');
  WRITELN(QOUT, '15. CCCHRAN');
  REPEAT UNTIL (WRITELN(QOUT, 'ENTER TEST NUMBER');
  REPEAT UNTIL (WRITELN(QOUT, 'UNTIL (N_TEST IN STATE);
  W_TEST IN TEST);
CASE 1: BINOM(N_VARS, TRUE);
6: MCNEMAR(N_VARS, 99, TRUE);
7: R*CCCONT(N_VARS, 0, TRUE);
19: CCCHRAN(N_VARS, 0, TRUE);
END;
FOR I:=1 TO 49 DO WRITE(OP, F_NAME(I));
WRITELN(OP);
FOR J:=1 TO N_VARS DO BEGIN
  FOR I:=1 TO 49 DO WRITE(OP, V_NAME(J,I));
  WRITELN(OP);
END; (*CELL TEST*)
-----*)
(*PROCEDURE CF_NEWCA;
BEGIN
  WRITELN(QOUT, 'YOU HAVE A CHOICE OF TWO METHODS TO ENTER DATA:');
  WRITELN(QOUT, '1. ENTER EACH VALUE OF DATA SET');
  WRITELN(QOUT, 'VALUES MUST BE NUMERICAL, AND MEASUREMENT AT');
  WRITELN(QOUT, 'LEAST ORDINAL. ALL TESTS MAY BE PERFORMED THIS WAY');
  WRITELN(QOUT, '2. ENTER ONLY THE COUNT OF SUCCESSSES/FAILURES');
  WRITELN(QOUT, 'GR ROW/COLUMN CELL COUNT');
  WRITELN(QOUT, 'DATA VALUES AND LEVEL OF MEASUREMENT MAY BE');
  WRITELN(QOUT, 'NCMINAL, BUT COUNTS MUST BE INTEGERS >= 0');
  WRITELN(QOUT, 'THE FOLLOWING TESTS MAY BE PERFORMED THIS WAY:');
  WRITELN(QOUT, 'BINOMIAL, MCNEMAR, R*C CONTINGENCY (CHI-SQUARE)');
  WRITELN(QOUT, '');
END;

```

```

WRITELN(QOUT);
END: (*CH_NEWDA*)
-----*)
PROCEDURE OLD_DATA;
BEGIN
  REPEAT (*UNTIL RIGHT FILE*)
  REPEAT (*UNTIL CHCHOICE IS A TO F*)
  WRITELN(QOUT);
  * WHICH CATA SET IS TO BE USED? 'A'-'F'...);
  RST; READLN(QIN,CH_1);N_IN_F:=CRD(CH_1);
  UNTIL (N_IN_F IN (128,135));
  CHECK_FILE(N_IN_F,S,N_IN_F,N_OUT_F,N_VARS,RIGHT_F,EQUAL_S,N_ELMS);
  UNTIL (RIGHT_F;N_VARS,EQUAL_S,N_EQ_S,N_ELMS);
  WHICH_TEST(*OLD_DATA*)
END: (*OLD_DATA*)
-----*)
BEGIN
  CUTSET:=(1,1,2,6,7,9.); (*INDICES OF TESTS WITH CELLS*)
  CELL_TERMOLT:=(1,2,6);
  RST; WRITELN(QOUT);
  WRITELN(QOUT);DC YOU WANT TO DO A TEST CN A PREVIOUSLY ,
  IF ENTERED CATA SET?);
  IF YES THEN CLD_DATA
  ELSE THEN BEGIN
    REPEAT NEWCA;
    REPEAT RST;
    WRITELN(QOUT, 'ENTER V FOR VALUE OR C FOR COUNT');
    READLN(QIN,CH_1); V OR C:=ORD(CH_1);
    UNTIL (V OR C IN (1,131,135));
    CELL_CT:=(V OR C=131);
    IF NOT(CELL_CT) THEN
      NEW_DATA(N_EQ_S,N_VARS,N_IN_F,EQUAL_S,N_ELMS)
    ELSE CELL_TERMOLT;
  END: (*IF NOT OLD DATA*)
  WRITELN(QOUT); TO PERFORM TEST, TYPE "CRUNCH" IN CMS .);
  WRITELN(QOUT);
END.

```

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22134	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3. LTC Stephen J. Paek, Code 55Ph Department of Operations Research Naval Postgraduate School Monterey, California 93943	3
4. Deputy Under Secretary of the Army (Operations Research) Room 2E621, The Pentagon Washington, D.C. 20310	1
5. Deputy Chief of Staff for Operations and Plans Attn: DAMO-ZD (Mr. E.B. Sandiver III) Room 3A538, The Pentagon Washington, D.C. 20310	1
6. Headquarters U.S. Army Training and Doctrine Command Attn: ATTG Ft. Monroe, Virginia 23651	1
7. Combined Arms Operations Research Activity Attn: ATZL-CAC-A (COL A. West) Ft. Leavenworth, Kansas 66027	1
8. Director U.S. Army Concepts Analysis Agency 8120 Woodmont Avenue Bethesda, Maryland 20814	1
9. Director U.S. Army TRADOC Systems Analysis Activity Attn: Dr. W. Payne White Sands Missile Range, New Mexico 88002	1

10. CDR Charles Hutchins, Code 55 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93943
11. LTCOL Joseph Mullane, Code 54Mn 1
Department of Administrative Sciences
Naval Postgraduate School
Monterey, California 93943
12. LTCOL Michael Hester, Code MPI-20 1
Headquarters Marine Corps
Washington, D.C. 20310
13. MAJ Philip J. O'Brien 4
541 - 44th Street
Des Moines, Iowa 50312
14. Commanding General 2
(Attn: Dr. Drysbick)
Development Center, MCDEC
Quantico, Virginia 22342

END

FILMED

02 - 84

DTIC