Statistical software for small computers, designed for use by non-professional statisticians

By Cheng Tai Gan

A dissertation presented to the University of Edinburgh for the degree of Master of Philosophy.
1982.

Abstract

A user-friendly computer program for testing the equality of means and elementary statistical calculations is developed on a microcomputer for non-professional statisticians. The program is aimed at reducing the misuse of statistics. Users are asked a number of questions and then are directed to enter their data if answers to the questions are satisfactory. The program examines the data provided by the users and then provides elementary calculations of statistics and selects a test statistic for testing the equality of means. Comments or warnings are issued to users where necessary.

Statistical methods involved in the program are reviewed. Contrary to what many people may believe, the use of non-parametric methods for testing the equality of means in order to avoid the normality assumption (which is required by the parametric tests) does not protect us from a possibility of misuse or misinterpretation of statistics when the distributional properties of data are not known. It is often more dangerous than using parametric tests. There are of course situations where non-parametric methods are appropriate.

New algorithms are developed where no satisfactory algorithms exist. Proofs are given for new algorithms or generalized algorithms.

Acknowledgement

I wish to thank Mr. W. Lutz and Dr. C. G. G. Aitken for the advice and encouragement they have given me over the past two years.

Declaration

I hereby declare that this dissertation has been composed by myself and that the work reported is my own.

Contents

		page
Chapter One	Introduction	1
1.1.	General introduction	1
1.2.	A brief review of the misuse of statistics in medical journals	3
1.3.	Purposes of the project	5
1.4.	The choice of test statistics	6
Chapter Two	Statistical details	10
2.1.	Introduction	10
2.2.	Criteria for selecting statistical methods	15
2.3.	Notation	16
2.4.	Testing of normality	16
2.5.	Tests of a single mean	19
2.6.	Tests of equality of two means	22
2.7.	Tests of equality of several means	24
2.8.	The estimation of the power λ of a transformation	29
2.9.	The detection of 'outliers'	30
2.10.	Fisher's g-statistics	30
2.11.	Summary	31
Chapter Three	Algorithms	32
3.1.	Criteria for selecting algorithms	32
3.2.	Calculations of upper quantiles and percentile points	32
3.3.	Randomization tests	43

	3.4.	Rank tests	50
	3.5.	Calculations of means and sums of deviations about the mean	55
	3.6.	Sorting algorithm	56
	3.7.	Plotting algorithm	57
Chapter Four		Program development	59
	4.1.	Program design	59
	4.2.	Computer languages	62
	4.3.	The use of flow-charts	63
	4.4.	The wording of questions	64
	4.5.	The use of GOTO-statements	65
	4.6.	Program validation	65
	4.7.	The use of range-check option	67
	4.8.	Program optimisation	68
	4.9.	Historical references	69
Chapter Five		Program details	72
	5.1.	Introduction	72
	5.2.	Testing of assumptions	72
	5.3.	Conditions for use of the test statistics	73
	5.4.	Program documentation	75
References			86
Appendix :		Program listing and examples	

Chapter One

Introduction

Section 1.1. General introduction.

A number of statistical techniques are widely used by non-statisticians, for example medical doctors. These users may have little statistical knowledge and may be unaware of the assumptions on which the methods depend. It is not unreasonable to assume that they are primarily concerned with the conclusions they can draw or support after applying statistical methods. Few users are likely to question the validity of the statistical methods, and especially if the results of these statistical methods support their prior beliefs. Sometimes, statistical analyses are only done because many journals require data to be treated statistically. In many situations, these users find it hard to get statistical advice even if they wish it.

With the spread of computing facilities, users may be tempted to use computers for their statistical analyses if programs are available, or by writing their own programs, without paying sufficient attention to statistical aspects or computational accuracy. The spread of microcomputers may make the situation even worse. Users may be tempted to think that they can do data analysis if they know how to use statistical computer programs; they may even feel that it is sufficient for them to feed their data into chosen programs and the computers will do the rest for them. Users may even try analyses by 'trial and error' and then see which analysis supports their prior beliefs.

Dissuading or stopping such users from using statistics beyond their competence is not practical. It is difficult to convince them that certain statistical techniques are beyond their competence, especially if they have done some elementary statistics. It is also not practical to complicate computing facilities (Gentleman (1979, page 94)) as a means to discourage them from using statistics. There is no practical way of stopping anyone with elementary computing skills writing a program to execute, for example, a t statistic.

Reid and Lemon (1980) suggest that the education of users is a possible solution to the misuses of statistics by users. This is ideally true, but is unlikely to be practical for many people. Data seldom satisfy the underlying assumptions on which statistical methods depend. Many statistical properties are not really quantified and the interpretations depend heavily on the actual data in the example being studied. All this makes user education difficult. It is unreasonable to expect most users to be knowledgeable about the robustness of various tests or the choice of a suitable transformation. Knowing a little can be more dangerous than knowing nothing. It has also to be remembered that they have to channel almost all their energy to their own field of study and the study of statistics also requires mathematical skills which they may not have time to master.

These users should not be blamed for misuses of statistical techniques. Statisticians should offer them help where possible. By the careful design of programs, the number of misuses of statistics can be reduced.

Before we start designing programs for these users, it is instructive to review the common types of mistakes.

Section 1.2. A brief review of the misuse of statistics in medical journals.

Badgley (1961) analysed 103 articles published in 1960 in two Canadian medical journals and found 24.3% of the papers contained errors as shown in Table 1. Schor and Karten (1966) published a detailed analysis of the uses of statistical methods in 295 papers published in 1964 from 10 leading medical journals; 53% of these papers were acceptable and 47% were not, (see Table 2). Gore et al (1977) analysed papers in the British Medical Journal during three months of 1976 and they found 42% of the papers had at least one error as shown in Table 3. Glantz (1980) gave an analysis of the use of Student's t-test in one volume each of Circulation Research and Circulation; 46% of the articles in Circulation Research and 27% of the articles in Circulation used the t-test when an analysis of variance or multiple comparisons tests should have been used, (see Table 4).

Some of the errors found in the above review articles which could reasonably be dealt with by computer programs are :

1. Errors with Student's t-test : which are mainly
   (a) Use of the two sample t-test on paired data.
   (b) Assumption of equality of variances.
   (c) The testing of multiple hypotheses.
2. Violation of distributional assumptions.

It is important to note that the actual percentages of misuses could be higher than the findings in the above review articles. The raw data are not usually published and without the raw data a thorough examination of the validity of the statistical methods is not possible.

It is also important to note that any assessment of the validity of the statistical methods involves subjective judgements and the criteria used are also to some extent arbitrary.

Table 1.

Appropriate statistical analysis	42.7%
Inappropriate statistical analysis	24.3%
Additional Analysis required	33.0%

Table 2.

Number of errors per study by type of study				
Type	Number read	Number not acceptable	Number of errors	Average number of errors per unacceptable study
Analytical case	149	108	253	2.34
Description	146	32	39	1.22
Total	295	140	292	2.09

Table 3.

	n	% of total	% of paper that used statistics
No statistical analysis	15	19	--
Acceptable use of statistical method	30	39	48
At least one error	32	42	52

Table 4.

No statistical analysis	20	25	--
Appropriate use of t-test	16	20	27
Inappropriate use of t-test	36	46	61
Analysis of variance	7	9	12

Section 1.3. Purposes of the project.

Glantz (1980) concluded that the system of review of articles submitted to journals had not been able to control the inappropriate use of statistics. Popular computer programs, for example BMDP and SPSS do not help users to avoid misapplying statistical methods. Given that users are allowed to choose their own test statistics, it is almost impossible to prevent them misusing these methods.

The main aims of this project are to develop a computer program which will help non-professional statisticians avoid these pitfalls and to develop this type of program on a microcomputer.

Section 1.4. The choice of test statistics.

There are several ways of designing programs which may decrease the number of misuses of statistical methods. A common approach is to issue warnings to the users. This has not proved to be useful as the warnings are usually ignored. A computer program which does not continue with the analysis when any 'violation' of the assumptions is detected is unlikely to be attractive to users who may feel they have wasted their time entering the data.

If users are allowed to choose the test statistics, then it is difficult to avoid the misuses mentioned above. Users may select the few statistical methods they know and use them inappropriately. Thus it was decided not to permit users to choose their own test statistics. After a series of questions and answers, users may be asked to enter their data; the program will test underlying statistical assumptions and select a test statistic. This approach should reduce the misuse of statistics. If users are prevented from choosing their test statistics, then misuses such as applying the two-sample Student t-test to paired observations and multi-group hypotheses are eliminated provided the users answer the questions correctly and honestly.

In developing the program the following assumptions have been made :

(1) Many users know only a little about statistics.

(2) Prior information about the underlying statistical distributions is very rarely available, and even if it is available, users are not always able to use it.

(3) Users know that they want to test the equality of mean responses between various groups.

Fisher's g-statistics and the coefficient of variation of variances (see 2.7. (A) for its definition) are not frequently used by medical doctors. It is questionable whether programs for such users should provide the g-statistics and the coefficient of variation of variances with which they are not familiar or may have difficulty with the interpretation. The question would be best answered by another more fundamental question, that is whether or not such users should be allowed to handle their own data. The answer to this question is of course "yes" due to practical reasons and no one can stop others from handling their own data.

It is commonly taken for granted that the mean plus or minus twice the standard error provides an estimate of the 95% confidence interval. If data are very skewed and the sample size is small then this interval may be incorrect. Fisher's g_1-statistic is a measure of symmetry, it can provide a rule of thumb as to the validity of this interval. Fisher's g_2-statistic is a measure of 'peakedness' and 'tailedness', not of 'peakedness' only, (see Finucan (1964)). If all these statistics g_1, g_2 and the coefficient of variation of variances are small, then the test statistics are very likely to be valid.

Gore et al (1977) found that an inadequate description of the basic data makes it difficult for the readers to visualise the data. In the discussion of the paper by Stigler (1977), it is suggested that sample skewness and kurtosis should be calculated

routinely as measures of distributional shape. Pearson and Please (1975) demonstrated the close relation between the validity of various tests and skewness and kurtosis. They also pointed out that testing for normality cannot be a substitute for information about distributional properties provided by skewness and kurtosis.

Fisher's g-statistics and the coefficient of variation of variances are good 'rules of thumb' for the validity of various test statistics. The misuse or misinterpretation of various test statistics should not be ascribed to the provisions of Fisher's g-statistics and the coefficient of variation of group variances. The problems start with the data, lack of statistical understanding and the nature of statistical practice. For example, users are told that for the one sample Student's t-test to be valid, data must be of a bell-shaped distribution. However, statisticians themselves may apply the one sample t-test to an apparently U-shaped distribution as they know that the t-statistic converges extremely rapidly to normality for symmetrical distributions (see for examples Geary (1947) and Ractcliffe (1968)).

It is important to note that sample skewness and kurtosis can be seriously affected by a few extreme observations. Mean and variance are also affected as well but to a lesser extent. The sample skewness and kurtosis can be poor measures of distributional shape. On the other hand, because of their sensitivity to extreme observations which usually provide more information about the spread of the data, they also provide more information about distributional shapes. The point here is that it

is difficult to have robust and yet informative measures of distributional shape.

Brief guidelines and interpretation of various statistics can be stored on disk and users should be able to obtain access to them there.

Chapter Two

Statistical details

This chapter describes and reviews all the statistical methods involved in the program. The details of the actual implementation will be given in chapter 5.

Section 2.1. Introduction.

First the validity of various test statistics is discussed in the context of the departure from the 'true' p-value under ideal conditions. It is well-known that this is related to the skewness and kurtosis of the distribution. Lee and Gurland (1977) show that the one-sample t-test can behave very differently in situations with the same population skewness and kurtosis. However, in practice, one cannot distinguish many distributions clearly, thus skewness and kurtosis are still good indicators of the validity of a test statistic. Usually, information about the skewness and kurtosis of a population is not available. These parameters have to be estimated from the data and their estimations require a fairly large sample size, especially for kurtosis. Unless we can estimate them with sufficient accuracy, the theoretical results which depend on them will not be as useful as they may appear to be. Simulation studies are more useful in the sense that more insight into the behaviour of various tests can be achieved even though they do not prove any theory. Real life data are also likely to be very different from simulated data as, for example there may be tied observations or observations from mixed populations which we may not be able to identify.

Another problem is that of measuring robustness. It is very vague to say, for example, that the F-test is robust to the departure of normality provided the error distribution is not too skewed and has well-behaved tails. The theoretical results described below are not very useful for developing computer programs which are proof to misuses unless robustness can be quantified. If sample sizes are small, one may just have to take the validity of the test statistic by faith. Furthermore, the interpretation of "smallness" of sample sizes is also connected with the behaviour of the data, (see for example Ractcliffe (1968)).

Apart from independence, normality is the most important factor regarding the successful development of a 'misuses proof' computer program. If the data are normally distributed, there is always a test of significance which is at least approximately valid. Welch's versions of the t-test and F-test do not assume equality of variances but do assume normality. If one has a very powerful general test for the equality of variances, a valid test statistic and multiple comparisons procedure could then be used. Such a test statistic does not seem to exist. Usually, the fewer assumptions a test statistic makes, the less powerful it is.

Another problem concerns tests for the equality of population variances in cases involving more than one sample. Gans (1981) finds that using the F-test as a preliminary test of the equality of variances of two samples does not give enough

protection against a possible misuse of Student's t-test and he suggests unconditional use of the Welch t-test when sample sizes are not equal. The inability of the F-test to detect inequality of variances is not surprising as it is very sensitive to kurtosis. Lauer and Han (1974) suggest using the F-test with widely varying significance levels depending on the sample sizes. This may seem statistically unsatisfactory as the F-test itself takes account of sample sizes. Bartlett's test for the equality of variances is also well-known for its sensitivity to non-normality. Box's modified Bartlett's test and Dunn's multiple comparisons procedure are asymptotic tests. It is not clear how large the sample sizes must be before these tests are valid in a practical context. Other more robust tests usually suffer from lack of power.

The pitfall of the joint assessment of normality for several groups is that individual non-normality cannot be identified. In addition, an extreme non-normal sample could be masked by other normal samples. However, in practice, we may be more interested in 'combined' normality. It is rare that in a set of samples, one or two samples are very 'non-normal' while others are very 'normal'.

It is tempting to turn to non-parametric methods if the assumption of normality is suspected to be false. Wetherill (1960), Pratt (1964) and Hilgers (1982) have pointed out the danger of turning to Wilcoxon rank sum test when the violation of the assumption of normality is suspected. Teir-Walsh and Toothaker (1974) examined the normal distribution and two

exponential distributions and made a similar point about the use of the Kruskal-Wallis and Normal Scores tests. Strictly speaking, the Wilcoxon rank sum test is a test for the identity of two populations. To use it as a test of shift in location, we must then assume that the two populations have identical shapes. This is a more demanding assumption than the normality assumption required by the t-test knowing that the t-test is not sensitive to non-normality. One has almost no way of making a numerical check on the identity of distributional shapes. Similarly, the Kruskal-Wallis test assumes data from populations with identical shapes which is also more demanding than the F-test. Conover and Iman (1981) show the close relation between parametric statistics and non-parametric rank test statistics. This raises the possibility that, in most cases, the use of a rank transformation is just a waste of information, though the authors have a different motive in their article.

A rank transformation may also change the intended null hypothesis, an outcome of which non-professional statisticians may be unaware. Thus the use of a rank transformation is likely to complicate the interpretation of the statistics. It is, of course, also true that, the use of parametric tests can lead to rejection of the null hypothesis of the equality of means because of other differences, for example the inequality of variances. However, this danger seems to be a lesser one. A parametric transformation changes the intended null hypothesis of the equality of means of original data. Thus complications may also occur in the interpretation of data. However, this complication

seems to be lesser than that of the rank transformation and more definite advice can be given to the users. As the testing of underlying assumptions is to some extent arbitrary, automatic transformation of data to achieve normality or the equality of variances is undesirable. A better approach seems to be to produce parallel analyses.

The assumptions of normality and of the equality of variances are tested. The independence of the data is not tested. In single effect analysis, data do not usually have a natural order and common sense is more important than formal statistical testing. The program questions the users about this. Moreover, there are many types of dependency. The problem is basically a statistical one rather than a computing one. The detection or estimation of a parameter to measure dependence is likely to depend on the order in which the data are presented. More harm is likely to be done by developing programs for non-professional statisticians to work with dependent data. Users should seek the advice of statisticians when working with dependent data.

A statistician may feel uncomfortable about conditioning his statistic on a set of tests of preliminary assumptions which are in fact to some extent also arbitrary. However, these preliminary tests may be better than no check at all, and the testing of these assumptions will also provide some more information about the data.

It should be noted that the above discussion is from a computing point of view rather than a statistician's point of view.

Section 2.2. Criteria for selecting statistical methods.

Many statistical techniques are available. Each one has its own merits and limitations. Thus the following criteria are adopted.

(a) Statistical considerations : A test which is robust within a wider class of distributions is usually preferred.

(b) Implementation considerations : They must be easily implemented on computers. This rules out certain methods, for example, the Studentized Range test for multiple comparisons as this test requires reference to tables at various required significance levels which cannot be easily computed. Most tables are for a very limited number of combinations of values for the parameters, and interpolation may then be necessary.

(c) Definiteness or subjectivity : Test statistics which are more definite and involve less subjective judgements are preferred. For example, the Chi-Square goodness of fit test for normality is ruled out because a change in the number of intervals may lead to a different value.

(d) Popularity : Popular tests are given first priority.

(e) Consistency : For example, the Welch F-test is chosen for testing the equality of means in case of the inequality of variances instead of other alternatives because the Welch t-test is used for the case of two samples. Reasons for the use of the Welch t-test will be given later.

Section 2.3. Notation.

Let x_{ij} $(i=1,2,\ldots,k,\ j=1,2,\ldots,n_i)$ be the j^{th} independent observation on the random variable X_i with mean u_i and variance σ_i^2. Define

$$v_i = n_i - 1, \quad N = \sum_{1}^{k} n_i$$

$$x_{i.} = \sum_{j=1}^{n_i} x_{ij}/n_i$$

$$e_{ij} = x_{ij} - x_{i.}$$

$$x_{..} = \sum_{ij} x_{ij}/N$$

$$s_i^2 = \sum_{j=1}^{n_i} e_{ij}^2/v_i$$

$$s^2 = \sum_{1}^{k} v_i s_i^2/(N - k)$$

and $R(x_{ij})$ be the rank of x_{ij} in the overall sample. Tied observations are treated by averaging their corresponding ranks. Define

$$R_{i.} = \sum_{j=1}^{n_i} R(x_{ij})/n_i$$

Section 2.4. Testing of normality.

(A) Single sample assessment.

Let z_i $(i=1,2,..,n)$ be independent observations on a random variable Z and $z_{(1)} \leq z_{(2)} \leq \ldots \leq z_{(n)}$ be the order statistics. Shapiro and Wilk (1965) present a test of normality,

$$W = \frac{\left(\sum_{i=1}^{n} a_{i,n} z_{(i)}\right)^2}{\sum_{i=1}^{n} (z_i - z_\cdot)^2}$$

where $z_\cdot = \sum_{i=1}^{n} z_i/n$

The coefficients $a_{i,n}$ are given in Shapiro and Wilk (1965) for $n \leq 50$. A comprehensive simulation study by Shapiro et al (1968) indicates that W is a powerful omnibus test for normality.

Small values of W signify abnormality of Z. This test is a one-sided test, it does not distinguish positive and negative skewness, long-tailedness and flat-toppedness. A study by Chen (1971) indicates that the W test is sensitive to non-normality in contaminated normal distributions.

Shapiro et al (1968) also find that the test based on sample skewness and kurtosis can serve as a good test for the departure from normality. Bowman and Shenton (1975) give confidence contours for a test based on sample skewness and kurtosis, but this cannot be implemented on the computers.

For $n>7$, D' Agostino (1970) gives a transformation of sample skewness to normality as follows,

Let $$g = n^{1/2} \sum_{i=1}^{n} (z_i - z_\cdot)^3 / \left(\sum_{i=1}^{n} (z_i - z_\cdot)^2\right)^{3/2}$$

$$\beta = g\left(\frac{(n+1)(n+3)}{6(n-2)}\right)^{1/2}$$

$$b = \frac{3(n^2+27n-70)(n+1)(n+3)}{(n-2)(n+5)(n+7)(n+9)}$$

$$w^2 = -1+(2(b-1))^{1/2}$$

$$\delta = (\ \ln(w)\ )^{-1/2}$$

$$a = \beta[2/(w^2-1)]^{-1/2} \quad \text{then}$$

$$Y = \delta \ln(a+(a^2+1)^{1/2})$$

is approximately distributed as the standard normal distribution. Simulation results in D' Agostino (1970) indicate that this approximation is remarkably accurate. A study by Pearson et al (1977) indicates that tied observations can have serious effects on the Shapiro-Wilk statistic and a hardly significant effect on sample skewness.

(B) Joint assessment for multi-sample problem.

Wilk and Shapiro (1968) give a joint test statistic for several independent samples based on the W-statistic mentioned above. Let W_i be the W-statistic of the i^{th} sample and α_i be the corresponding significance level actually attained for i^{th} sample and F be the cumulative distribution of W_i.

Suppose

$$\alpha_i = F(w_i)$$

$$= \int_{-\infty}^{G_i} (2\pi)^{-1/2} e^{-\frac{t^2}{2}} dt$$

for some G_i, then

$$G = \sum_i G_i / k^{1/2}$$

is distributed as the standard normal distribution. G_i can be obtained by the equation,

$$G_i = \gamma + \delta \ln\left(\frac{w_i - a}{1 - w_i}\right)$$

where $$a = \frac{n_i^2 a_{1,n_i}}{n_i - 1}$$

and γ and δ are given in Shapiro and Wilk (1968). Simulation results in Wilk and Shapiro (1968) indicate that G performs very well.

A joint assessment based on the sample skewness is given in Pearson and Hartley (1972) as follows :

Let $P_i = 1 - Q_i = \Pr(g_i \leq \text{observed value})$

If all the distributions of the samples are positively skewed, then one can use $-2\sum_i \ln(Q_i)$ and $-2\sum_i \ln(P_i)$ if all the distributions of samples are negatively skewed, both statistics are then distributed as chi-square with 2k degrees of freedom.

For a two-sided test, one can define

$$R_i = 2Q_i \quad \text{if } g_i \geq 0$$
$$= 2P_i \quad \text{if } g_i < 0$$

then $-2\sum \ln(R_i)$ is distributed as chi-square with 2k degrees of freedom.

Section 2.5. Tests of a single mean.

For simplicity, we drop the first subscript i as i=1 throughout this section. For testing $u=u_o$, we have the following tests :

(A) Student's t-test.

If X is normally distributed, we have

$$t = n^{1/2}(x_{\cdot} - u_o)/s$$

distributed as t-distribution with v degrees of freedom. Deviations from normality are usually quantified in terms of values of population skewness,

$$\gamma_1 = E(X-u)^3/\sigma^3$$

and kurtosis,

$$\gamma_2 = E(X-u)^4/\sigma^4 - 3.$$

Geary (1947) obtained the following expansions for moments of t,

$$E(t) = -\gamma_1/(2n^{1/2}) - O(n^{-3/2})$$

$$Var(t) = 1 + 2/n + 7\gamma_1^2/(4n) + O(n^{-2}),$$

$$\gamma_1(t) = -2\gamma_1/n^{1/2} - O(n^{-3/2})$$

$$\gamma_2(t) = 2(3 - \gamma_2 + 6\gamma_1^2)/n + O(n^{-2})$$

which are described in Pearson and Please (1975). These results suggest that t is more sensitive to skewness than kurtosis and agree with the simulation studies by Ractcliffe (1968), Pearson and Please (1975), Bowman et al (1977) and Posten (1979) of t-statistics sampling from a wide variety of distributions. A good review article is Cressie (1980).

(B) Randomization test and signed-rank Wilcoxon test.

Let $y_j = x_j - u_o$, (j=1,2..,n) and z_j be the absolute value of y_j. If X is symmetrically distributed about u_o, we have the following tests :

(a) Randomisation test.

Let S_+ be the sum of all positive y_j's and P be the number of subsets of $\{z_j : j=1,2,..n\}$ with sum less than or equal to S_+, then the significance probability = $2^{-n}P$

against the alternative hypothesis $u<u_o$.

In the presence of zeros, they are dropped first and n is reduced accordingly. This test is described in Pratt and Gibbons (1981).

(b) Signed-rank Wilcoxon test.

For large n, the randomisation test computation becomes too heavy. If the z_i are replaced by their ranks, we have the signed-rank Wilcoxon test,

$$W = \text{sum of ranks of positive } y_j\text{'s}$$

$$E(w) = n(n+1)/4 \quad \text{and}$$

$$\text{Var}(w) = n(n+1)(2n+1)/24 - \sum_{t=1}^{e} (d_t^3 - d_t)/48$$

where d_t is the number of z_t's equal to the t^{th} smallest value of the z_t's and e is the number of ties. Zeros are dropped before ranking as suggested by Wilcoxon. Pratt (1959) shows that some difficulties may occur in the Wilcoxon's procedure of zero treatment and he suggests that zeros be dropped only after ranking. Conover (1973) shows that each procedure has its own merits. To be consistent with the treatment of zeros in the randomisation test, Wilcoxon's procedure is adopted.

If the x_j's are differences of paired-observations and

the null hypothesis is that of no treatment effect, then the symmetry assumption is unnecessary.

Section 2.6. Tests of equality of two means.

(A) Two-sample t-test.

If the X_i's are normally distributed and $\sigma_1^2=\sigma_2^2$, then

$$t = (x_{1.}-x_{2.})/[s(n_1^{-1}+n_2^{-1})^{1/2}]$$

is distributed as the t-distribution with n_1+n_2-2 degrees of freedom. It is well-known that t is robust against non-normality and inequality of variances if sample sizes are equal. Studies by Pearson and Please (1975) and Posten (1978) indicate this. A simulation study by Posten (1978) indicates that for population skewness $0\leq\gamma_1\leq 2.0$ and kurtosis $-1.6\leq\gamma_2\leq 4.8$, the t-test performs remarkably well and it is good enough for practical purposes. Graphs in Pearson and Please (1975) indicate that the two-sample t-test is more robust against non-normality than the one-sample t-test.

(B) Welch t-test.

If the X_i's are normally distributed, but the variances are not assumed to be equal, Welch (1947, 1949) gives

$$wt = (x_{1.}-x_{2.})/(s_1^2/n_1+s_2^2/n_2)^{1/2}$$

which is approximately distributed as the t-distribution with d degrees of freedom where

$$d = (s_1^2/n_1+s_2^2/n_2)^2/[s_1^4/(n_1^2v_1)+s_2^4/(n_2^2v_2)]$$

Mickey and Brown (1966) prove that

$$Pr(|t_1|<c) \leq Pr(|wt|<c) \leq Pr(|t_2|<c)$$

where t_1 is distributed as the t-distribution with $\min(v_1,v_2)$ degrees of freedom and t_2 is distributed as the t-distribution with v_1+v_2 degrees of freedom. It can be shown that d falls in the range of $\min(v_1,v_2)$ and v_1+v_2. Thus the Welch t-test should provide a good approximation provided the sample sizes are not too small. Murphy (1967), Lee and Gurland (1975) and Gans (1981) all find that the Welch approximation works remarkably well. Murphy and Gans also find that the Welch t-test is more robust against non-normality than the ordinary t-test.

(C) Two-sample randomisation and Wilcoxon rank sum tests.

If the populations are identically distributed except for possibly a difference in location, we have

(a) Two-sample randomisation test.

Let P be the number of subsets of $\{x_{ij} : i=1,2,\ j=1,2,..,n_i\}$ with n_1 elements and sum less than or equal to $x_{1.}$, then

the significance probability = $P/\binom{N}{n_1}$

against the alternative hypothesis $u_1<u_2$.

This test is described in Pratt and Gibbons (1981).

(b) Wilcoxon rank sum test.

For large sample sizes, the randomisation test computation becomes too heavy. If x_{ij}'s are replaced by their ranks, we have the Wilcoxon rank sum test,

$$W_r = \sum_{j=1}^{n_1} R(x_{1j}), \quad E(W_r) = n_1(2^{-1}(N+1)) \quad \text{and}$$

$$\text{Var}(W_r) = n_1 n_2[(N+1) - \sum_{t=1}^{e} (d_t^3 - d_t)/(N(N-1))]/12$$

where d_t is the number of x_{ij}'s equal to the t^{th} smallest value of the x_{ij}'s and e is the number of ties.

Wetherill (1960) shows that the Wilcoxon rank sum test is a little more robust against the inequality of variances but much more sensitive to skewness and kurtosis than the t-test.

(D) Testing of equality of variances.

Let $F=s_1^2/s_2^2$, then F is F-distributed with v_1 and v_2 degrees of freedom under the assumption of normality.

Various studies (Gayen (1950), Finch (1950) and Pearson and Please (1975)) have shown that F is very sensitive to kurtosis but insensitive to skewness.

Section 2.7. Tests of equality of several means.

(A) Analysis of variance F-test.

Assuming normality and equality of variances, we have

$$F = \sum_{i=1}^{k} n_i(x_{i.} - x_{..})^2/((k-1)s^2)$$

distributed as F-distribution with k-1 and N-k degrees of freedom.

Inequality of the variances has little effect on F if the sample sizes are equal. Non-normality also has little effect if the distribution of the errors is not too skewed and has well behaved tails. However, inequality of the

variances can have a serious effect on the F-statistic if the sample sizes are not equal. Correlation among the data is the most serious departure from the assumptions (see Box (1954), Scheffe (1959, chapter 10), Seber (1980, chapter 5)). Box (1954) shows that if the variances are not equal, then the F-test is dependent on the spread of the distribution of the variances measured by the coefficient of variation of variances,

$$c = [\sum_{i=1}^{k} v_i(\sigma_i^2-\dot{\sigma}^2)^2/(N-k)(\dot{\sigma}^2)^2]^{1/2}$$

where $\dot{\sigma}^2 = \sum_{i=1}^{k} v_i\sigma_i^2/(N-k)$

(B) Welch F-test.

Assuming normality, we have the Welch F-statistic

$$wf = \sum_{i=1}^{k} w_i(x_{i.}-\bar{x})^2/[(k-1)(1+2(k-2)f)]$$

where $w_i = n_i/s_i^2$, $u = \sum_{i=1}^{k} w_i$, $\bar{x} = \sum_{i=1}^{k} w_i x_{i.}/u$

$$f = (k^2-1)^{-1} \sum_{i=1}^{k} (1-w_i/u)^2/(n_i-1)$$

and wf is approximately distributed as F-distribution with k-1 and $(3f)^{-1}$ degrees of freedom.

This test is described in Brown and Forsyth (1974). They find that wf is robust under the inequality of variances, the asymptotic approximation of wf is valid if each sample has at least 10 observations and it is not unreasonable down to 5 observations.

Other possible test statistics are given by James (1951)

and Brown and Forsythe (1974). Dijkstra and Warter (1981) find that none of the tests is uniformly better than the other two.

(C) Kruskal-Wallis test.

If the populations are identically distributed except for possibly a difference in location, then the F-statistic based on ranks (see Conover and Iman (1981)),

$$F_r = (N-k)H/[(k-1)(N-1-H)]$$

is approximately distributed as the F-distribution with k-1 and N-k degrees of freedom where the Kruskal-Wallis test statistic,

$$H = 12 \sum_{i=1}^{k} n_i(R_{i.}-(N+1)/2)^2/[N(N+1)(1- \sum_{t=1}^{e} (d_t^3-d_t)/(N^3-N))]$$

is approximately distributed as chi-square with k-1 degrees of freedom where d_t is the number of the x_{ij}'s equal to the t^{th} smallest value of x_{ij}'s and e is the number of ties. The chi-square approximation of the Kruskal-Wallis test does not take account of the sample sizes.

(D) Multiple comparisons.

If the null hypothesis of the equality of the means is rejected at the α level of significance, then a multiple comparisons procedure may be performed to judge which groups are different from which others.

(a) The null hypothesis rejected by the F-test.

The Bonferroni method produces $100(1 - \alpha)\%$ joint confidence interval of u_i-u_j, $i \neq j$ as

$$x_{i.}-x_{j.} \pm t_{\gamma,p}\, s(n_i^{-1}+n_j^{-1})^{1/2}$$

where $\gamma = \alpha/(k(k-1))$ and $t_{\gamma,p}$ is the upper percentile point of the t-distribution with $p=n_i+n_j-2$ degrees of freedom.

(b) The null hypothesis rejected by the Welch F-test.

A test similar to the Welch t-test called the T_2 procedure in Dunnett (1980b) is used. The $100(1-\alpha)\%$ joint confidence interval for u_i-u_j, $i\neq j$ is

$$x_{i.}-x_{j.} \pm t_{\gamma,u_{ij}} \left(\frac{s_i^2}{n_i}+\frac{s_j^2}{n_j}\right)^{1/2}$$

where $u_{ij} = (s_i^2/n_i+s_j^2/n_j)^2/[s_i^4/(n_i^2v_i)+s_j^4/(n_j^2v_j)]$

$$\gamma = [1-(1-\alpha)^{2/(k(k-1))}]/2$$

and $t_{\gamma,u_{ij}}$ is the γ upper percentile point of the t-distribution with u_{ij} degrees of freedom.

(c) The null hypothesis rejected by the Kruskal-Wallis test.

Dunn's method is used and two samples are judged to be different if

$$|R_{i.}-R_{j.}| > Z_\gamma \left[\left(N(N^2-1)-\sum_{t=1}^{e}(d_t^3-d_t)\right)(n_i^{-1}+n_j^{-1})/(12(N-1))\right]^{1/2}$$

where Z_γ is the upper $\gamma = \alpha/(k(k-1))$ percentile point of the standard normal distribution. This procedure is described in Daniel (1978, page 214).

When variances are equal, Dunnett (1980a) and Stoline (1981)

recommend the Tukey-Kramer method which is based on the Studentized range distribution. On practical consideration, the Bonferroni method is chosen.

When variances are unequal, the T_2 procedure is found to be conservative by Dunnett (1980b). Other possible procedures are also given in Dunnett (1980b). T_2 is chosen partly because of practical considerations. An extensive list of references can be found in Stoline (1981). No study of Dunn's procedure has been found.

(E) Testing equality of variances.

Let $M=(N-k)\ln(s^2)-\sum_{i=1}^{k}(n_i-1)\ln(s_i^2)$. Box (1953) shows that for any parent distributions with the same population kurtosis γ_2, $M/(1+\gamma_2/2)$ is asymptotically distributed as the chi-square distribution with k-1 degrees of freedom. If the parent distribution is normal, $\gamma_2=0$, then for small sample sizes, Bartlett (1937) shows that $M/(1+A)$ is distributed as chi-square with k-1 degrees of freedom, where

$$A = (3(k-1))^{-1}\left(\sum_{i=1}^{k} v_i^{-1}-(N-k)^{-1}\right)$$

This is the traditional test called Bartlett's test and it is well-known that it is very sensitive to non-normality and is the "best" test if the parent distributions are normal (see Gartside (1972), Layard (1973) and Geng et al (1979)). For a reasonable estimate of γ_2, quite large sample sizes are needed. Layard (1973) finds that

$$\dot{\gamma}_2 = N^{-1} \sum_{i=1}^{k} n_i^2 \frac{\sum_{ij} e_{ij}^4}{(\sum_{ij} e_{ij}^2)^2} - 3$$

is badly biased when sampling from non-normal distributions and suggests

$$\ddot{\gamma}_2 = N \frac{\sum_{ij} e_{ij}^4}{(\sum_{ij} e_{ij}^2)^2} - 3$$

as an estimate of γ_2 based on empirical results. To be consistent with other parts of the program, an estimate by Anscombe (1961) is used, that is

$$\overline{\gamma_2} = \frac{N^3}{r(r+2)(1+(N-1)p^4)-3N} \left[\frac{r+2}{r} \frac{\sum_{ij} e_{ij}^4}{(\sum_{ij} e_{ij}^2)^2} - \frac{3}{N} \right]$$

where $r = N-k$ and $1+(N-1)p^2 = \frac{N}{r}$

Section 2.8. The estimation of the power λ of a transformation.

When non-normality of data is detected and no suitable test statistic without assuming normality is available, transformation of the data may be necessary. An estimate of λ is given by Anscombe in the discussion of Box and Cox (1964). The estimate

$$\overline{\lambda} = 1-2\overline{\gamma_1}x_{..}/[3(2+\overline{\gamma_2})s]$$

where $\overline{\gamma_1} = \dfrac{N^2 \sum_{ij} e_{ij}^3}{(1+(N-1)p^3)(r\sum_{ij} e_{ij}^2)^{3/2}}$

$\overline{\gamma_2}$, r and p are defined as in the last section (2.7. (E)).

Section 2.9. The detection of 'outliers'.

Let L and U be the lower and upper 25% quartiles of a given sample respectively and d=1.5(U-L). A data point which is below L or above U with a distance of d or more is declared as an 'outlier'.

Section 2.10. Fisher's g-statistics.

As Fisher's g-statistics can provide good "rules of thumb" for the validity of various test statistics and can also provide information about the distributional properties of data, it is useful to provide users with g_1 and g_2 statistics.
When the sample size is small, g_2 is not very useful and hence it is not provided. Let z_i (i=1,2,..,n) be a random sample,

$$z_\cdot = \sum_{i=1}^{k} z_i/n \qquad k_2 = \sum_{i=1}^{n} (z_i - z_\cdot)^2/(n-1)$$

$$k_3 = n \sum_{i=1}^{n} (z_i - z_\cdot)^3/[(n-1)(n-2)] \qquad \text{and}$$

$$k_4 = \{ \sum_{i=1}^{n} (z_i - z_\cdot)^4 n(n+1)/(n-1) - 3(\sum_{i=1}^{n} (z_i - z_\cdot)^2)^2 \}/[(n-2)(n-3)]$$

then Fisher's g_1 and g_2 statistics are defined as follows :

$$g_1 = \frac{k_3}{k_2^{3/2}} \qquad \text{and}$$

$$g_2 = \frac{k_4}{k_2^2}$$

Section 2.11. Summary.

In this chapter, we describe and review various test statistics involved in the program. The general conclusion is that the use of non-parametric rank tests for testing the equality of means in order to avoid the normality assumption (which is required by the parametric tests) does not protect us from a possibility of misuse or misinterpretation of statistics when the distributional properties of data are not known. It is often more dangerous than using parametic tests. The conclusion drawn from this review coupled with the problem of computational complexity of randomization tests which will be mentioned in the chapter 3 lead to the decisions determining the choice of the test statistics which will be given in the chapter 5.

Chapter Three

Algorithms

This chapter describes all the algorithms necessary to carry out all the computations of statistics described in chapter 2 and the associated functions. Proofs are given for new or generalized algorithms.

Section 3.1. Criteria for selecting algorithms.

There are many algorithms available for a given task. Thus the following criteria are adopted in selecting algorithms.

(a) Accuracy considerations : An algorithm which is accurate is always preferred. It may be meaningless to achieve, say, 5 decimal accuracy for p-value calculations. However, if it can be achieved at little cost, there seems no reason not to achieve it. Giving an accurate answer is always a good thing. It also gives users confidence in the program.

(b) Practical considerations : If an algorithm requires a lot more codes or computations but has little advantage in accuracy over others, it is avoided.

Section 3.2. Calculations of upper quantiles and percentile points.

Let $z(x)$ be the density function and $Q(x)$ be the upper quantile of the standard normal variable x, that is,

$$z(x) = (2\pi)^{-1/2} e^{-\frac{x^2}{2}} \quad \text{and}$$

$$Q(x) = \int_x^{\infty} z(t)dt$$

(A) Normal distribution.

Moran (1980) gives

$$Q(x) = \frac{1}{2} - \frac{1}{\pi}\left[\frac{x}{3(2)^{1/2}} + \sum_{n=1}^{\infty} n^{-1} e^{-\frac{n^2}{9}} \sin\left(\frac{n\, 2^{1/2}\, x}{3}\right)\right] \text{ ---- (1)}$$

approximately and suggests truncation at n=12 will give 9 decimal accuracy for $|x|<7.0$. This of course cannot be achieved with single precision calculations.

Quantiles for x=0.00(0.02)5.20 are produced and compared with table 2 in Pearson and Hartley (1972), they agree with all six decimals except a few of them differ by 10^{-6}.

There is a loss of accuracy due to the subtraction made in (1), but this is of no practical importance.

There is a computational advantage if the series in (1) is coded in a step-down manner, that is descending in n or adding small terms first.

Other algorithms are given by Cooper (1968a) and Hill (1973) and described by Kennedy and Gentle (1980). They do not appear to be superior to (1). No comparison is made.

Conversely, given a quantile q=Q(x), one wants to evaluate the corresponding percentile point x, and Bailey (1981) gives the approximation,

for $q>1.01\text{x}10^{-6}$,

$$x = t_1[1+0.0078365t_1^2-0.00028810t_1^4+0.0000043728t_1^6]$$

where $t_1 = [-\pi/2\ \ln(4pq)]^{1/2}$, $p\geq 1/2$ and $q=1-p$.

and for $q\leq 1.01\text{x}10^{-6}$,

$$x = t_2 + \frac{0.1633}{t_2^2} + \frac{0.5962}{t_2^3},$$

where $u = -2\ln(q)$ and $t_2 = [u-\ln(2\pi u)]^{1/2}$

Approximations of percentile points are computed at various values of q, they appear to be very accurate.

(B) Student t-distribution.

Let Q(t,v) be the upper quantile of the t-distribution with v degrees of freedom. From Abramowitz and Stegun (1964, Formulae 26.7.3, 26.7.4), one has an exact series expansion,

$$Q(t,v) = (1-A(t,v))/2 \quad \text{where}$$

$$A(t,v) = \frac{2}{\pi}[z+\sin(z)\cos(z)(1+\frac{2}{3}\cos^2(z)+..+\frac{2.4..(v-3)}{1.3..(v-2)}\cos^{v-3}(z))]$$

if v>1 and odd,

$$= \frac{2z}{\pi} \qquad \text{if } v=1,$$

$$= \sin(z)[1+\frac{1}{2}\cos^2(z)+\frac{1.3}{2.4}\cos^4(z)+..+\frac{1.3..(v-3)}{2.4..(v-2)}\cos^{v-2}(z)]$$

if v is even,

where $z = \tan^{-1}(\frac{t}{v^{1/2}})$,

$$\sin^2(z) = \frac{t^2}{t^2+v},$$

$$\cos^2(z) = \frac{v}{t^2+v},$$

Cooper (1968b) who obtained the above formula differently has coded this algorithm using straight-forward term by term evaluation. His coding suffers from :

(a) Exponent underflow and hence a check against underflow is needed when a term is evaluated and added successively. Cooper's routine contains no check against exponent underflow.

(b) Large rounding error.

(c) Slow computing speed.

Using 'nested multiplications' known as Horner's method gives a numerically more stable algorithm as the coefficient of $\cos^2(z)$ in the series decreases as the power of $\cos^2(z)$ increases (see Carnahan et al (1969. page 6)).

A recurrence relation for the 'nested multiplications' is used by Hill (1970) and is described in Kennedy and Gentle (1980). Changing division to multiplication, one obtains a slightly clearer recurrence relation.

$$C_v = 1,$$

$$C_k = C_{k+1}\, b\, (k-1)/k + 1, \quad k=v-2, v-4,.., 3 \text{ or } 2.$$

then

$$A(t,v) = \frac{2}{\pi}[z + (ab)^{1/2} C_3] \qquad \text{if v is odd,}$$

$$= a^{1/2} C_2 \qquad \text{if v is even.}$$

where $a = \sin^2(z)$, $b = \cos^2(z)$ and

$C_3 = 1$ if $v = 1$.

One notices that less operations are required by using Horner's method and exponent underflow is impossible in evaluating C as $C_k \geq 1$ for all k.

The above exact computation works well if the number of degrees of freedom is small. Thus for more than 20 degrees of freedom, the Cornish-Fisher type approximation is used.

$$Q(t,v) = Q(x)$$

where

$$x = z\left[1 + \frac{z^2+3}{d} - \frac{4z^6+33z^4+240z^2+855}{10\ d(d+0.8z^4+100)}\right]$$

$$z = \left[\ a\ \ln\left(1 + \frac{t^2}{v}\right)\ \right]^{1/2}$$

$$a = v-0.5, \qquad d = 48\ a^2$$

which is described in Kennedy and Gentle (1980) and is shown by El Lozy (1982) to be extremely accurate.

Quantiles with the similar t and v values in tables in Hartley and Pearson (1950a) are produced and compared. They agree with all five decimal places except a few of them differ in the fifth decimal place. There is a loss of accuracy when A(t,v) is very small or near 1, but this is of no practical significance.

Conversely, given a quantile q=Q(t,v), one wants to evaluate the corresponding percentile point t, Fisher and Cornish (1960) give the following approximation,

$$t = x\left[\ 1 + \frac{x^2+1}{4n} + \frac{5x^4+16x^2+3}{96n^2} + \frac{3x^6+19x^4+17x^2-15}{384n^3} + \frac{79x^8+776x^6+1482x^4-1920x^2-945}{92160n^4} + \frac{27x^{10}+339x^8+930x^6-1782x^4-765x^2+17955}{368640n^5}\right]$$

where x may be obtained from (A) above.

This approximation has been shown to be very accurate by Sahai and Thompson (1974). The Horner's method is used to evaluate t.

(C) F-distribution.

Let $Q(f,v_1,v_2)$ be the upper quantile of the F-distribution with v_1 and v_2 degrees of freedom.

From Abramowitz and Stegun (1964, formulae 26.6.4,26.6.5., 26.6.6., 26.6.8), one has the following exact expansions.

For v_1 and v_2 both odd,

$$Q(f,v_1,v_2) = 1-A(t,v_2) + B(v_1,v_2)$$

where $A(t,v_2)$ is defined as in the series expansion for the t-distribution but with

$$z = \tan^{-1}\left(\frac{v_1 f}{v_2}\right)^{1/2} \quad \text{and} \quad t = (v_1 f)^{1/2}$$

$$B(v_1,v_2) = 0, \qquad \text{if } v_1=1,$$

$$= \frac{2}{\pi^{1/2}} \frac{\left(\frac{v_2-1}{2}\right)!}{\left(\frac{v_2-2}{2}\right)!} \sin(z) \cos^{v_2}(z) \cdot$$

$$\left[1+\frac{v_2+1}{3}\sin^2(z)+..+\frac{(v_2+1)(v_2+3)..(v_1+v_2-4)}{3 \cdot 5 .....(v_1-2)}\sin^{v_1-3}(z)\right]$$

$$\text{if } v_2 > 1.$$

A recurrence relation for the evaluation of the finite series in $B(v_1,v_2)$ is

$$C_{v_1} = 1,$$

$$v = v_2 - 2,$$

$$C_k = C_{k+2}\, a\, (k+v)/k + 1, \quad k=v_1-2,\ldots,3$$

where $a = \sin^2(z)$ and $C_3 = 0$ if $v_1=1$,

thus

$$B(v_1,v_2) = \frac{2}{\pi^{1/2}} \frac{(\frac{v_2-1}{2})!}{(\frac{v_2-2}{2})!} \sin(z)\cos^{v_2}(z)C_3$$

For v_1 even, one has

$$\cos^2(z) = \frac{v_2}{v_2+v_1 f} = y,$$

$$\sin^2(z) = \frac{v_1 f}{v_2+v_1 f} = 1-y,$$

$$Q(f,v_1,v_2)$$

$$= y^{v_2/2} \cdot$$

$$[1+\frac{v_2}{2}(1-y)+\ldots+\frac{v_2(v_2+2)..(v_2+v_1-4)}{2 \cdot 4 \cdot \cdot (v_1-2)}(1-y)^{(v_1-2)/2}]$$

----- (2)

or

$$Q(f,v_1,v_2)$$

$$= y^{(v_1+v_2-2)/2} \cdot$$

$$[1+\frac{v_1+v_2-2}{2}(\frac{1-y}{y})+\ldots+\frac{(v_1+v_2-2)..(v_2+2)}{2 \cdot 4 \cdot \cdot (v_1-2)}(\frac{1-y}{y})^{(v_1-2)/2}]$$

----- (3)

Ling (1978) finds that formula (3) is more underflow or overflow prone than (2). One notices that y is bounded by 1. When (1-y) is small, $(1-y)^n$ converges to zero faster than $(\frac{1-y}{y})^n$ and y^n as n increases and thus in fact (2) can be more (exponent) underflow prone than (3). However, (3) is more overflow prone than (2). One can protect (2) from exponent underflow in evaluating its series expansion by 'nested multiplications'. Thus (2) is superior to (3). A recurrence relation for the series evaluation in (2) is similar

to that for $B(v_1,v_2)$ above with k terminating at k=2.

For even v_2 and odd v_1, by the reflexive relation,

$$F_{p(v_1,v_2)} = \frac{1}{F_{1-p(v_2,v_1)}}$$

one obtains

$$Q(f,v_1,v_2)$$

$$= 1-(1-y)^{v_1/2}\left[1+\frac{v_1}{2}y+\ldots+\frac{v_1(v_1+2)..(v_2+v_1-4)}{2\ .\ 4\ .\ \ .(v_2-2)}\,y^{(v_2-2)/2}\right]$$

------ (4)

Ling (1978) did not seem to realise that formula (4) is a direct result of (2) by applying the reflexive relation and suggested that when v_1 is odd and v_2 is even, the use of the reflexive relation and (2) is superior to (4).

When $v_1 > 40$ or $v_2 > 40$, an approximation by Paulson (1942),

$$Q(f,v_1,v_2) = Q(x) \quad \text{approximately}$$

where

$$x = \frac{f^{1/3}[1-2/(9v_2)]-[1-2/(9v_1)]}{[2/(9v_1)+f^{2/3}2/(9v_2)]^{1/2}}$$

is used. This approximation is well-known and very accurate. When $v_1 < 15$, the accuracy of this approximation drops and thus the reflexive relation is used, and it appears to be more accurate.

All the recurrence relations for the evaluations of $Q(f,v_1,v_2)$ are numerically more stable than term by term evaluations as the coefficients in the finite series decrease as their power

of y or (1-y) increases, (see Carnahan et al (1969, page 6)). The calculations of the quantiles of the F-distribution are so far the most difficult ones because more parameters are involved. The quantiles at various combinations of v_1, v_2 and f are produced and compared with the tables of F-distribution in Pearson and Hartley (1972). They agree very well.

(D) Chi-square distribution.

Let $Q(c^2,v)$ be the upper quantile of the Chi-square distribution c^2 with v degrees of freedom. From Abramowitz and Stegun (1964, formulae 26.4.4., 26.4.5), one has

$$Q(c^2,v)$$

$$= 2Q(x) + 2z(x) \sum_{r=1}^{\frac{v-1}{2}} \frac{c^{2r-1}}{1 \cdot 3 \cdot \ldots (2r-1)} \qquad \text{if v is odd,}$$

$$= (2\pi)^{1/2}\, z(x)\left[1+ \sum_{r=1}^{\frac{v-2}{2}} \frac{c^{2r}}{2 \cdot 4 \cdot \ldots 2r}\right] \qquad \text{if v is even,}$$

where

$$x = |c|.$$

Similarly, by applying Horner's method, one has the recurrence relation,

$$R_v = 0,$$

$$R_k = (1+R_{k+2})\,\frac{c^2}{k}, \qquad k=v-2,v-4,\ldots,2 \text{ or } 1,$$

$$Q(c^2,v)$$

$$= 2\,Q(x) + 2\,z(x)\,R_1/x, \qquad \text{if } x \neq 0 \text{ and v is odd,}$$

$$= 0 \qquad \text{if } x = 0,$$

$$= (2\pi)^{1/2} z(x)[1+R_2], \qquad \text{if v is even.}$$

This recurrence relation has an exponent underflow protection built into it like others. Since the coefficient of c^2 is getting smaller as r increases, the above relation is numerically more stable than term by term evaluations (see Carnahan et al (1969,page 6)).

For large degrees of freedom (v>40), $Q(c^2,v)$ is approximated as Q(x) where

$$x = w+a/3-wa^2/36-(w^2-13)a^3/1620+7(6w^3+17w)a^4/38880+..$$

$$a = (2/n)^{1/2}$$

$$w = (c^2-v-v\ln(c^2/v))^{1/2}$$

and w has the sign of (c^2-v).

This approximation is described and is shown to be very accurate by El Lozy (1982).

Quantiles with parameters similar to the tables of the Chi-square distribution in Hartley and Pearson (1950b) are produced and compared, they agree very well.

Remark : A statistical analysis of numerical stability of 'nested multiplications' is given by Henrici (1964, page 316-317).

(E) Rank tests.

(a) Signed-rank Wilcoxon test.

The upper quantile is approximated as

$$= Q(x) + \frac{3N^2+3N-1}{10N(N+1)(2N+1)}(x^3-3x)\ z(x).$$

where $x = \dfrac{|W-E(W)| - \frac{1}{2}}{[var(W)]^{1/2}}$

where W, E(W) and Var(W) are given in 2.5. (B) (b).

This approximation has been shown to be very accurate by Claypool and Holbert (1974) provided N is not too small. For small N, a randomisation test on the original data can be used and its algorithm is given in the next section.

(b) Wilcoxon rank sum test.

The upper quantile is approximated as

$$= Q(x) + \frac{n_1^2+n_2^2+n_1n_2+n_1+n_2}{20n_1n_2(n_1+n_2+1)}(x^3-3x)\ z(x)$$

$$\text{where } x = \frac{|W_r-E(W_r)| - \frac{1}{2}}{[\text{var}(W_r)]^{1/2}}$$

where W_r, $E(W_r)$ and $\text{Var}(W_r)$ are given in 2.6.(D).

This approximation has been shown to be very accurate by Verdooren (1963) provided sample sizes are not too small. For small sample sizes, a randomisation test on the original data can be used and its algorithm is given in the next section.

(c) Kruskal-Wallis test.

The upper quantile of the Kruskal-Wallis test is approximated by an approximation given by Wallace (1959), that is the F-statistic on ranks as given in 2.7.(C) and with reference to the F-distribution.

This approximation is not conservative (see Iman and Davenport (1976)), but in general, it is more accurate than the chi-square approximation.

Section 3.3. Randomisation tests.

Green (1977) gives algorithms for one and two sample randomisation tests. He uses heuristic arguments to eliminate unnecessary computations. His algorithms are based on keeping track of partial sums and reversed partial sums. The present algorithms described below are more systematic and proofs of validity are easier and are expected to be faster than Green's algorithms as the combination generator described below is very efficient and heuristic arguments eliminate a large amount of unnecessary computations. However, no comparison is made.

(A) Generations of combinations in lexicographic order.

Mifsud (1963) gives an algorithm for generating m combinations out of n objects in lexicographical order. Gentlemen (1975) gives essentially the same algorithm. Page and Wilson (1979, page 117) again give a similar algorithm which is adopted from Shen (1962). A recent study by Akl (1981) indicates that Mifsud's algorithm is the fastest existing combination generator. In what follows, I shall describe the algorithm given by Page and Wilson and show how it can be modified into a much faster and more suitable algorithm for developing algorithms for randomisation tests. A theoretical analysis of the modified algorithm is also given.

(a) Page-Wilson's algorithm and its improvement.

Denote n objects by {1,2,...n} and let

$A_m = \{a_1, a_2, \ldots, a_m\}$ $(1 \leq m \leq n)$ be an m combination of n objects in lexicographical order. The algorithm generates the next combination as follows :

1. Find the largest i such that $a_i < n-m+i$.
2. Add 1 to a_i.
3. Perform the substitutions, $a_j = a_{j-1}+1$ $j=i+1,\ldots,m$.

One notices that as $1 \leq i \leq m$, it is useful to define $h=m+1-i$ so that $1 \leq h \leq m$ and one then has the equivalent algorithm as follows :

1. Find the smallest h such that $a_{m+1-h} < n+1-h$.
2. Add 1 to a_{m+1-h}.
3. Perform the substitutions, $a_j = a_{j-1}+1$, $j=(m+1-h)+1,\ldots,m$.

One notices that with the above modification, the application of backtrack programming technique is easier. Essentially, a combination C consists of two subsets $C_1 = \{a_1, a_2, .., a_{i-1}\}$ and $C_2 = \{a_i, \ldots, a_m\}$ such that $a_i > a_{i-1}+1$, $a_j = a_{j-1}+1$ for $j>i$ and C_2 has h elements. C_1 is empty when the algorithm is initialized with the first combination {1,2,..,m}. When C_1 is empty and $a_m=n$, all possible combinations are generated. Suppose B is the combination generated next to C and h′ is the smallest integer such that $a_{m+1-h'} < n+1-h'$. Thus B consists of two parts, $B_1=\{b_1, b_2, .., b_{i'-1}\}$ and $B_2=\{b_{i'}, \ldots, b_m\}$ such that $b_j = b_{j-1}+1$, $j>i'$, $b_{i'} > b_{i'-1}+1$ and B_2 has h′ elements where $i'=m+1-h'$. If $a_m < n$, it is trivial that $h'=1$, $b_m = a_m+1$ $i'=i=m$ and $B_1=C_1$. If $a_m=n$, since $a_j = a_{j-1}+1$, for $j>i$ and

$a_i > a_{i-1}+1$, $h'=h+1$, $B_1=\{a_1,a_2,..,a_{i-2}\}$ and $B_2=\{a_{i-1}+1,...,a_{i-1}+h'\}$. Hence, one can conclude that the search of h is unnecessary, h can take only two values, that is 1 when $a_m<n$ and 'old' h +1 when $a_m=n$.

Thus, one has i=m if $a_m<n$ and i='old' i -1 if $a_m=n$.

One thus arrives at a very simple algorithm as follows:

1. If $a_m<n$, then set i=m, add 1 to a_m. Go to 4.
2. Subtract 1 from i and set $p=a_i-i+1$.
3. Perform the substitutions, $a_j=j+p$, $j=i,...,m$.
4. Deliver the combination.

One may initialize the above algorithm as $a_m=n$, i=2 and $a_1=0$. The first combination generated will be $\{1,2,...,m\}$. When $a_1=n+1-m$, all combinations are generated.

(b) A theoretical analysis of the modified combination generator.

Let us consider the following Pascal implementation,

```
a[m]:=n;
i:=2;
a[1]:=0;
last:=n-m+1;
REPEAT
   IF a[m]<n THEN BEGIN
      a[m]:=a[m]+1;
      i:=m;
   END ELSE BEGIN
      i:=i-1;
      p:=a[i]-i+1;
      FOR j:=i TO m DO
         a[j]:=j+p;
   END;
   {* deliver combination here *}
UNTIL a[1]=last;
```

The number of executions of each of a[m]<n and a[1]=last is $\binom{n}{m}$. The number of executions of i:=m is $\binom{n}{m}-\binom{n-1}{m-1}$. The number of executions of each of i:=i-1 and of p:=a[i]-i+1 is $\binom{n-1}{m-1}$. Note that h=m+1-i as given in (a), thus the number of executions (additions) of a[m]:=a[m]+1 plus that of a[j]:=j+p is equal to the sum of all h's immediately after the generations of combinations.

Noting that n and m are arbitrary except $1 \leq m \leq n$, we can calculate the sum of all h's recursively. The sum of h=1 is $\binom{n}{m}-\binom{n-1}{m-1}$. Since h=2, i.e. i=m-1 only if $a_m=n$ and $a_{m-1} \neq n-1$ immediately before the generations of combinations, the sum of all h=2 is 2 times that of h'=h-1=1 in generating m'=m-1 combinations out of n'=n-1 objects. Thus the sum of h=2 is,

$$2\left[\binom{n-1}{m-1}-\binom{n-2}{m-2}\right].$$

Generally, the sum of all h=j is

$$j\left[\binom{n+1-j}{m+1-j}-\binom{n-j}{m-j}\right]$$

Hence the number of additions of a[m]:=a[m]+1 plus that of a[j]:=j+p is equal to

$$\sum_{j=1}^{m} j\left[\binom{n+1-j}{m+1-j}-\binom{n-j}{m-j}\right]$$

$$= \sum_{j=1}^{m} \binom{n+1-j}{m+1-j}-m$$

$$=\binom{n+1}{m}-(m+1).$$

Let us count the number of operations of execution of the s:=e as the number of operations in evaluating e plus the assignment command ":=". The 'cost' of addition,

subtraction, assignment and comparison are considered to be equal. Thus the total number of operations (excluding looping indexing) is about

$$3\binom{n}{m}+2\binom{n+1}{m}+4\binom{n-1}{m-1}$$

$$=5\binom{n}{m}+6\binom{n-1}{m-1}+\binom{n-1}{m-2}$$

The method of analysis above is different from that of Mifsud's algorithm described in Reingold et al. (1977, page 181). Experimental operation countings of Mifsud's algorithm are given by Akl (1981).

(B) Calculation of the one-tailed probability of a two sample randomisation test.

Suppose $x_1, x_2, \ldots, x_n$ are n numbers in ascending order and that one wants to calculate the number of m ($m \leq n$) combinations of x_i's whose sum is less than or equal to a given number S. Without loss of generality, assume that $S_o + x_n \leq S$ where $S_o = \sum_{i=1}^{m-1} x_i$, otherwise one can eliminate x_n and reduce n accordingly.

Suppose C and B are combinations of {1,2,...,n} described in (A) (a), that is,

$$C = \{a_1, a_2, .., a_{i-1}, a_i, .., a_m\}$$

$$B = \{a_1, a_2, \ldots, a_{i-1}, \ldots., a_{m-1}, a_m+1\} \quad \text{if } a_m < n.$$

$$B = \{a_1, a_2, \ldots, a_{i-2}, a_{i-1}+1, \ldots, a_{i-1}+h+1\} \quad \text{if } a_m = n.$$

$$B_2 = \{a_{i-1}+1, \ldots\ldots, a_{i-1}+h+1\}$$

$$S_1 = \sum_{j=1}^{i-1} x_{a_j},$$

$$S_2 = \sum_{j=i}^{m} x_{a_j},$$

$$S_c = S_1 + S_2$$

One has sum of x_j's, $j \in B$, $S_b = S'_1 + S'_2$ where

$S'_1 = S_1$ if $a_m < n$.

$= S_1 - x_{a_{i-1}}$ if $a_m = n$.

$S'_2 = x_{a_m+1}$ if $a_m < n$.

$= \sum_{j \in B_2} x_j$ if $a_m = n$.

Two heuristic arguments can be used to eliminate unnecessary computations when the sum of a combination is greater than S,

(1) If C_1 (given in (A) (a) above) is empty, i.e. i=1, we exhaust all possible combinations.

(2) We can start to backtrack as $x_j \geq x_{a_m}$, $j=a_m+1,..n$.

The above algorithm is simple and mathematically correct, but this does not imply its implementation on computers will always give the correct answer. The problem is that computers can only represent a discrete and finite set of numbers. If data are small in values and the checking of equality of two (real) numbers are not avoided by applying the trichotomy law of real numbers, the implementation of the above algorithm can give us wrong answers. For example,

(i) Sample 1 data 1.0, 2.0, 3.0, 4.0, 5.0

Sample 2 data 3.0, 4.0, 4.0, 4.0, 5.0

gives the correct p-value = 0.174603

(ii) Sample 1 data 0.1, 0.2, 0.3, 0.4, 0.5

Sample 2 data 0.3, 0.4, 0.4, 0.4, 0.5

gives a wrong p-value = 0.071429

Green's Fortran routine suffers from this problem. An obvious solution is to convert all data to integers first and then do the computations. From a computing point of view, this is not very good as there is an overflow problem. By scaling up data before doing any computations and applying the trichotomy law of real numbers, one can avoid the problem of comparing two real numbers on computers in implementing the above algorithm. The scaling factor should make the data more "discrete" and thus the factor should effect the representations of numbers on the computers. A suitable factor is 2.0×10^4.

(C) Calculation of the one-tailed probability of one sample randomisation test.

Using the identity,

$$\sum_{m=0}^{n} \binom{n}{m} = 2^n$$

the above algorithm for the two sample randomisation test can be used for the one sample randomisation test. However, one more heuristic argument can be introduced. That is, if there exists $k<n$ such that there is no k combination whose sum is less than or equal to observed sum, then no j combination ($j=k+1,..,n$) need be considered.

One may be interested in knowing how fast an algorithm for the one sample randomisation test can be. Shamos (1976) has proved that the randomisation test for matched-pairs is NP-hard. For an introduction to computational complexity theory and NP-problems, see for example Reingold et al (1977, chapter 9). This result tells us that one should not waste

time looking for a fast algorithm for a matched-paired randomisation test because it cannot possibly exist.

The present algorithm has exponential-time complexity (if the heuristic arguments are not used). From the identity of the binomial expansion given above, we see that this remark also applies to the calculation of a two-sample randomisation test.

Section 3.4. Rank tests.

Berchtold (1979) gives an algorithm for the signed rank Wilcoxon test which in fact is a special case of an algorithm given by Lehmann (1975, page 131). Pittner (1981) gives an algorithm for the Mann-Whitney test. Kummer (1981) gives an algorithm for the two sample Wilcoxon test and he uses Berchtold's algorithm to prove his algorithm and points out that his algorithm can be used to calculate any rank statistics. However, none of the algorithms calculate the 'tied correction factor', namely k^3-k for a tied group of k data. In what follows, I shall prove a very simple result from which various algorithms can be derived. Let R(z) be the rank of number z in the usual sense and ties are treated by averaging their corresponding ranks.

(A) Basic result, algorithms and proofs.

Result (1):

Let $z_1,z_2,\ldots,z_n$ be a sequence of real numbers. Define

$$d_{ij} = z_i - z_j \qquad i\neq j, 1\leq i,j\leq n$$

$$r_i = \{ \#d_{ij}>0 : 1\leq j\leq n\}$$

$$= \{ \#z_j : z_j<z_i,\ 1\leq j\leq n,\ j\neq i\}$$

$$\lambda_i = \{ \#d_{ij}=0 : 1\le j\le n, i\ne j\}$$

$$= \{ \#z_j : z_j=z_i,\ 1\le j\le n,\ j\ne i\}$$

then

$$R(z_i) = 1+r_i+\lambda_i/2 \qquad \text{and for } 1\le k\le n,$$

$$\sum_{i=1}^{k} R(z_i) = k+\{ \#d_{ij}>0 : 1\le i\le k, 1\le j\le n\}+\{ \#d_{ij}=0 : 1\le i\le k, 1\le j\le n\}/2.$$

Proof :

Without loss of generality, assume that

$z_i=z_{i+1}=\ldots=z_{i+\lambda_i}$, $\lambda_i\ge 0$ and $z_i\ne z_j$ for $j\ne i,\ldots,i+\lambda_i$

then

$$R(z_i) = \frac{1}{\lambda_i+1}\{\frac{\lambda_i+1}{2}(1+r_i+(1+r_i)+\lambda_i)\}$$

$$= 1 + r_i+\lambda_i/2.$$

$$\sum_{i=1}^{k} R(z_i) = k+ \sum_{i=1}^{k} r_i+ \sum_{i=1}^{k} \lambda_i/2$$

$$= k+\{ \#d_{ij}>0 : 1\le i\le k, 1\le j\le n\}+\{ \#d_{ij}=0 : 1\le i\le k, 1\le j\le n\}/2$$

(a) Signed-rank Wilcoxon test algorithm.

Let x_i , i=1,2,..,n be a random sample and assume

$x_i\ne 0$ for all i.

Define $d_{ij} = x_i-|x_j|, \quad i\ne j, 1\le i,j\le n$

$$p = \{ \#x_i>0 : 1\le i\le n\}$$

and W be the sum of ranks of all positive x_i's, then

the signed-rank Wilcoxon test

$$W = p+\{ \#d_{ij}>0 : 1\le i,j\le n\}+\{\#d_{ij}=0 : 1\le i,j\le n\}/2.$$

Proof :

Let $z_i=|x_i|$ for all i and $e_{ij}=z_i-z_j$, $i\neq j$, $1\leq i,j\leq n$

then by the Result (1),

$W = p+\{ \#e_{ij}>0 : x_i>0,\ 1\leq i,j\leq n\}+\{ \#e_{ij}=0 : x_i>0,\ i=1,2,\ldots n\}/2.$

But $d_{ij} = x_i-|x_j|>0$ iff $x_i>0$ and $e_{ij}>0$

$d_{ij} = x_i-|x_j| = 0$ iff $x_i>0$ and $e_{ij} = 0$

thus the result follows.

Remark : This algorithm is different from Berchtold's algorithm.

(b) Wilcoxon rank sum test algorithm.

Let x_i, i=1,2,..,n and y_j, j=1,2,..m be two random samples.

Define

$$d_{ij} = x_i-x_j \quad i\neq j, 1\leq i,j\leq n.$$

$$d_{i,n+j} = x_i-y_j,\ 1\leq j\leq m$$

and W_r be the sum of ranks of x_i's, then

$$W_r = n+\{ \# d_{ij}>0 : 1\leq i\leq n, 1\leq j\leq n+m\}+\{ \# d_{ij}=0 : 1\leq i\leq n, 1\leq j\leq n+m\}/2$$

Proof :

Define

$$z_i = x_i,\ i=1,2,..n.$$

$$z_{n+j} = y_j,\quad j=1,2,\ldots m.$$

then the result follows from Result (1).

(c) Kruskal-Wallis test algorithm.

Let x_{ij}, i=1,2,..k, j=1,2,..,n_i be k independent random samples of size n_i, i=1,2,..,k.

Define

$$p_{ij} = x_{ij}-x_{sk}\ s=1,2..p,\ k=1,2,..,n_s,\ s\neq i \text{ or } k\neq j$$

then the rank sum of sample i,

$$R_i = n_i + \{ \#p_{ij}>0\}+\{ \#p_{ij}=0\}/2$$

and hence the Kruskal-Wallis statistic,

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} n_i^{-1}(R_i - \frac{n_i(N+1)}{2})^2$$

where $N = \sum_{i=1}^{k} n_i$ can be calculated accordingly.

Proof :

Since in ranking the i^{th} sample against other samples, the division of other samples is immaterial, one can consider all the other samples as 'one' sample and rank i^{th} sample against it. The result follow from the algorithm for the Wilcoxon rank sum test.

Remark : One computer procedure is needed for calculating rank sums for the signed-rank Wilcoxon, the Wilcoxon rank sum and the Kruskal-Wallis tests. The algorithm for the signed-rank Wilcoxon test is redundant.

Algorithms for other rank statistics can be derived in a similar way.

(B) Improvement of algorithms.

All the algorithms stated above can be further improved to take account of the 'tie correction factor'.

Without loss of generality, consider a sequence of numbers z_i, i=1,2,...n and define

$$d_{ij} = z_i - z_j, \quad 1 \le i < j \le n$$

$$\lambda_n = 0$$

$$\lambda_i = \{ \#z_j : z_j = z_i, i < j < n \}, \quad 1 \le i < n.$$

If $d_{ij} \neq 0$, then $d_{ij} > 0$ if and only if $d_{ji} < 0$, so when comparing z_i and z_j, one adds 1 to either $R(z_i)$ or $R(z_j)$, but not both. Similarly, $d_{ij}=0$ if and only if $d_{ji}=0$, so when comparing z_i and z_j, if $d_{ij}=0$, one adds 0.5 to both $R(z_i)$ and $R(z_j)$. Now suppose that $z_i = z_{i+1} = \ldots = z_{i+k-1}$ $(k \ge 1)$ is a k-fold tie and $d_{ij} \neq 0$ for $j \neq i+1, \ldots, i+k-1$. By the definition of d_{ij}, one gets $\lambda_{i+j-1} = k-j$, $j=1,2,\ldots k$. Thus for any k-fold tie $k \ge 2$, there exists one and only one $\lambda = 1$. This implies that the number of ties is identified by the number of $\lambda = 1$. Thus the number of ties is equal to $\{ \#\lambda_i = 1 : i = 1,2,..,n \}$. One also knows that the last value of λ in a group of k-fold ties $(k \ge 2)$ is zero and the number of λ equal to zero is equal to that of $\lambda = 1$. Thus the number of tied observations in z_i, $i=1,2,\ldots,n$ is $\{ \#\lambda_i > 0 \} + \{ \#\lambda_i = 1 \}$. However, one may be more interested in the largest tie group and it can be easily calculated. For the 'tied correction factor' of statistics, a common factor of k-fold ties is $k^3 - k$. Note that

$$k = \lambda_i + 1 \text{ and } \lambda_{i+j-1} = k-j, \; j=1,2,..,k.$$

$$k^3 - k = \sum_{j=1}^{k} [(\lambda_{i+j-1}+1)^3 - (\lambda_{i+j-1}+1) - (\lambda^3_{i+j-1} - \lambda_{i+j-1})]$$

$$= \sum_{j=1}^{k} 3\lambda_{i+j-1}(\lambda_{i+j-1}+1)$$

In programming, d_{ij} must not be evaluated and it is used for mathematical convenience. The comparison of two numbers should be direct. However, if the actual implementation of comparison on computers is not by direct comparison, for example $z_i > z_j$ is evaluated as $z_i - z_j > 0$, then it makes not much difference.

One may also have a problem in comparing two real numbers on computers. In practice, data are fairly discrete and the checking of equality of two real numbers can be avoided by applying the trichotomy law of real numbers, thus it should not cause much problem.

Exact analyses of the above algorithms for rank tests are impossible. It is obvious that the time complexity of all algorithms are polynomial. The storage complexity is almost optimal i.e. minimal storage as almost no extra space is needed to carry out all the computations. No analysis is attempted here.

Section 3.5. Calculations of means and sums of deviations about the mean.

Formulae for calculations of means and sums of deviation about the mean are required for computing basic statistics and test statistics. Many articles have been published about algorithms for mean and sums of squares, (Welford (1962), Neely (1966), Young and Cramer (1971) and Ling (1974)). Ling (1974) finds that formulae are generally data dependent and no one is consistently better than any others, but generally, two-pass formulae are better.

Formulae used for computing mean and power sums of deviations about the mean are as follows :

$$M_1 = \sum_1^n x_i/n$$

$$P^r = \sum_1^n (x-M_1)^r$$

$$M_2 = P^1/n$$

Mean, $M = M_1+M_2$ which is known as Neely's algorithm.

$$\sum_1^n (x_i-M)^2 = P^2-nM_2^2$$

$$\sum_1^n (x_i-M)^3 = P^3-3M_2P^2+2nM_2^3$$

$$\sum_1^n (x_i-M)^4 = P^4-4M_2P^3+6P^2M_2^2-3nM_2^4$$

Section 3.6. Sorting algorithm.

A sorting procedure is required to sort the data in the ascending order to carry out for example the Shapiro-Wilk test for normality. The sorting algorithm is translated from the Algol's version of the sorting procedure due to Singleton (1969). All the five GOTO statements are eliminated. A remark on the algorithm is given by Griffin and Redish (1970).

Section 3.7. Plotting algorithm.

For plotting histograms, a plotting algorithm is required. A number of algorithms for plotting graphs have been published, (Thayer and Storer (1969), Nelder (1976) and Stirling (1981)). All these algorithms start with trying to get a "neat" step-size. For a computer program running without the users' intervention, they do not seem to be very satisfactory. A better algorithm should also take the number of data points into consideration. For example, it is generally undesirable to have the number of intervals more than the number of data points. Thus the number of intervals required should be "estimated". An algorithm which takes account of the consideration mentioned above should use the range of data and the number of data points to get a "neat" step-size and an "estimated" number of intervals. A limit to the number of intervals is necessary in a computer program.

Denote the rounded up number of z by round(z). Let R be the range and S be a number such that $1.0 \leq RS \leq 10.0$ where $S=10^p$ for some integer p and let k=round(RS). Suppose the limit of the number of intervals is about 20 and the number of data points is n. The algorithm is as follows :

(1) If $5<k<11$ then step=S and N=k. Go to (3).

(2) Set Q=20R, N=round(Q/round(Q/k)).

If N is odd and N>5 then increase N by 1.

If N=14 or N=18 then increase N by 2.

step = NS/20.

If step > 3.0 then round up step to an integer.

(3) If N<=10 and N<n then half the step.

If N>20 or N>n then double the step.

Number of intervals=round(range/step).

The "estimation" of the first plotting position is dependent on the output device and a suitable value for outputs on the screen or line printers is round(max/step+1) multiplied by step which is larger than the maximum value of the data where max is the maximum value of the data. The number of intervals may have to be increased by 1 or 2 in order to cover the minimum value of the data.

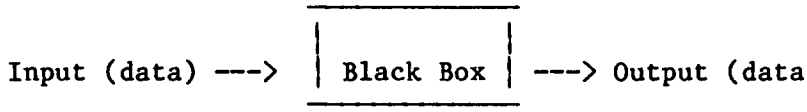
Chapter Four

Program Development

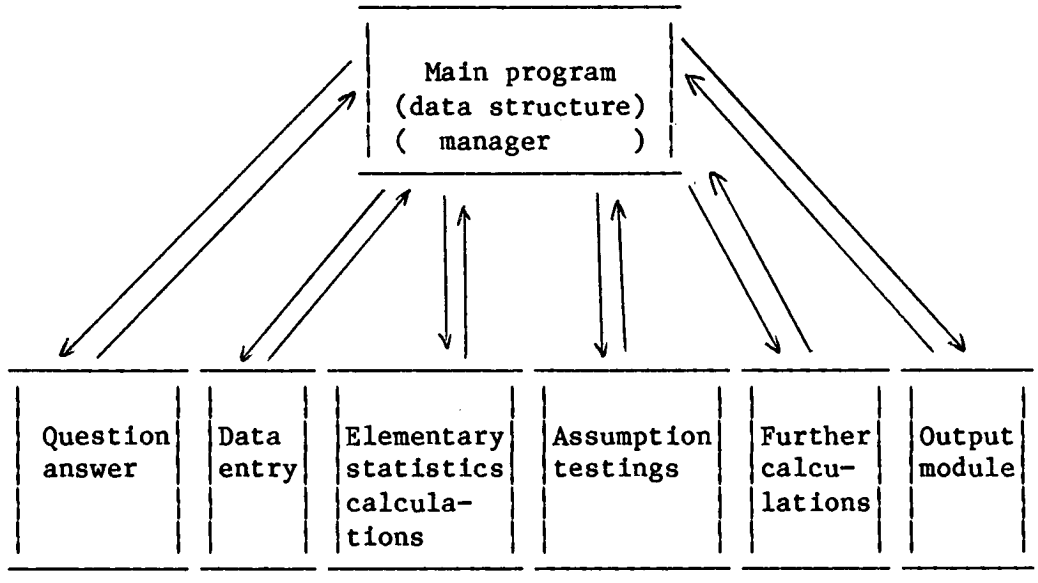
Since the recognition of software problems, an extensive study of programming methodology has been done by computer professionals. Experience has shown the application of methods contributed by computer professionals has improved program quality. This chapter describes the ideas and the steps in the development of our program.

Section 4.1. Program design.

A computer program consists of two main components, data structure and algorithms. One has thus two possible ways of starting a program. One is to specify or develop the algorithms first. Another is to specify the data structures and leave the algorithms until later. The latter approach, that is the data structure oriented approach is adopted in designing the program. In general, a program may be seen as a black box as shown below.

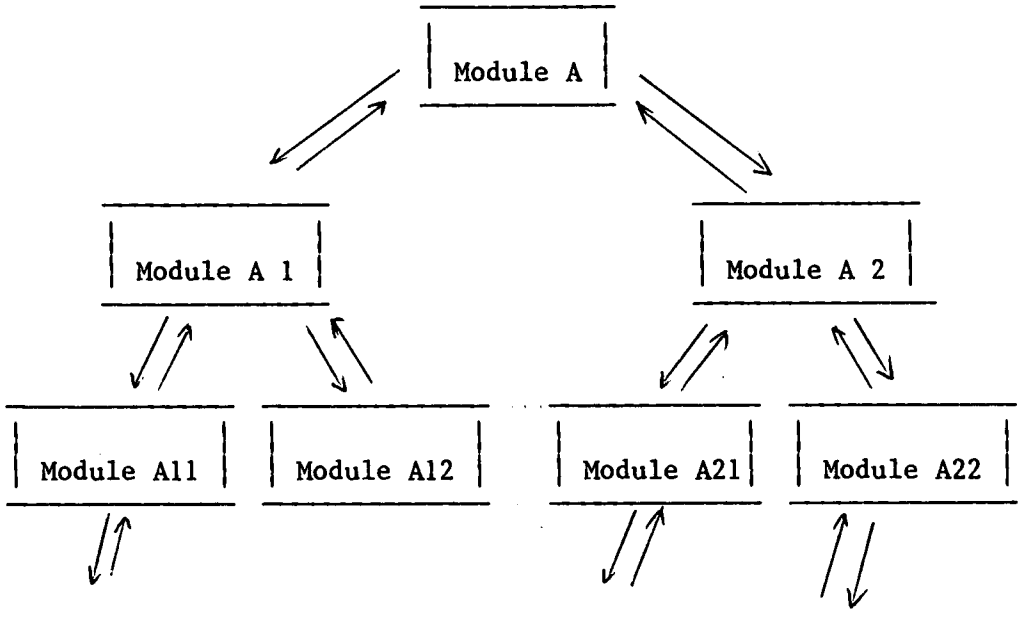


Obviously, one needs at least three modules, one for input, one for output and the other for carrying out the necessary operations (black box). After a few refinements, the following data flow diagram may be constructed.



One has thus specified the basic module interface and data flow of the program. Arrows show the directions of flow of the data. Conceptually, it is useful to imagine a data structure manager who passes data to and receives data from the various modules. No communication is allowed between modules.

Each module can now be treated independently and may be further subdivided into various sub-modules. Such divisions are continued until each sub-module or sub-sub-module does an identifiable task which is small enough to be solved without much effort. It is useful to imagine local data structure managers who pass data to and receive data from their sub-ordinates. A hierarchical diagram may look as shown below.



The above diagram is conceived as a data flow diagram, not only as a diagram for indicating the division of tasks. Lower levels receive data from higher levels and can do only tasks as directed by the higher levels. So far, algorithmic aspects are ignored and it is assumed that all the necessary algorithms are available. At this stage, one is concerned with what is to be done with a given set of data, not how it is to be done. The problem of computer languages is not considered. In practice, one has to identify some of the difficult or time-consuming modules during the design process. For example, time has to be allowed for the development of new algorithms where none exists. If algorithms cannot be developed in a reasonable time or no polynomial-time algorithms can possibly exist, changes may have to be made. Algorithms also affect data structures. One may also have to consider the programming language to be used, for example 'clean' data communications between modules are impossible in Basic. The above approach is still

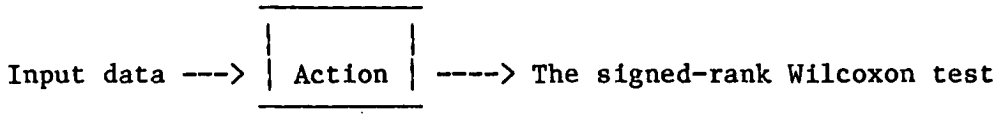
applicable, though, even if implementation is to be in an unstructured language. It is always possible to translate a structured solution to an unstructured language.

In the above approach, algorithms are operations or actions on data structures. The following example illustrates the idea.

Problem : Calculation of signed-rank Wilcoxon test.

Input : A set of data.

Output : The signed-rank Wilcoxon test (data).



How the Wilcoxon test is to be calculated is the job of "Action". If data are to be sorted, it is the job of "Action" to call a sorting routine to act on the data. The algorithms of "Action" have no direct connection with the input and output. As long as the specified output is met, the problem is solved.

Section 4.2. Computer Languages.

It is perhaps more useful to compare various languages rather than to look for a perfect language. For a microcomputer, the two widely available languages are Basic and Pascal. Some comparisons between these two languages follow :

Basic	Pascal
(a) Poor degree of standardisation.	Good degree of standardisation.
(b) Poor data structuring.	Powerful data structuring.
(c) Variable names are usually not meaningful	Variable names can be meaningful and more self-explanatory.
(d) Codes are generally difficult to read.	Codes are more readable.
(e) All variables are global and none can be passed to routines as parameters.	Variables can be localised or passed to procedures (routines) as parameters.

Whilst some other arguments favour Basic, the above arguments are sufficient for me to favour Pascal. These arguments are in fact central to programming. There are also additional powerful facilities in the UCSD (University of California at San Diego) Pascal system.

Section 4.3. The use of flow-charts.

The use of flow-charts as a development aid has been criticized by many computer scientists. One of the reasons is that flow-charts do not depict data structures which are central to programming. However, flow-charts can be very useful in documentation. Users or maintenance programmers can grasp the basic logic of a program without much effort by studying the associated flow-charts. Flow-charts are used as an aid in the documentation after the program has been completed.

Section 4.4. The wording of questions.

It is very difficult to have questions which are useful, concise and informative. For example, the use of the term 'statistical independence' is not comprehensible to many but a long explanation is undesirable and may not necessarily convey the precise meaning. Questions are also required to be useful. If randomization is regarded as a standard practice, then a question about randomization is necessary. There are however situations where randomization is impossible. Experiments may be done without control groups because it is impossible to have control groups, for example patients may be very ill and they have to be treated immediately. It is not clear whether or not these questions should be asked. If one sticks closely to the theoretical requirements, too many data sets may be rejected and users are likely to become frustrated. Users may even try to by-pass questions (see Sales (1980)) as they are primarily interested in the results produced by applying statistical methods.

The questions in the program were revised a number of times and raised the following topics :

(1) What kind of data are being presented ?

(2) The number of data collected per case and the number of groups.

(3) Whether or not the data are in the form of paired-observations (if only two groups).

(4) Whether or not there is any connection between cases.

(5) Whether or not randomization has been done.

(6) The possibility of differences in the populations because of factors other than that which the users intend to compare.
(7) The importance of the order in which the data are collected.

Users are required to answer all these questions as necessary. This may possibly make it more difficult to by-pass questions as users may not know which questions they have answered 'wrongly'.

Section 4.5. The use of GOTO-statements.

The UCSD Pascal system has a compiled-time option for the use of GOTO-statements. It is required to instruct the compiler if GOTO-statements are to be used. GOTO-statements usually make a program difficult to read. Theoretically, it is possible, with structured languages to develop programs without GOTO-statements. One may argue that it is unwise to evaluate Boolean expressions or make use of extra codes for the GOTO-free programs. The elimination of GOTO-statements does not automatically lead to better programs. However, no GOTO-statement is used in the present program.

Section 4.6. Program validation.

Program validation consists of testing and verification. Howden (1980) used several techniques to uncover 92 errors in IMSL (International Mathematical and Statistical Library) programs. The main difficulty is that programs are dynamic objects. This is even more difficult on microcomputers as many debugging and testing aids are not readily available.

(A) Program testing.

It is important to distinguish between experimenting and testing. Testing is an organised process to uncover errors and unexpected performance in programs, it is not to show programs are working on a few selected samples of inputs. Test data will necessarily be a small sample of all possible inputs. Testing is thus inadequate for achieving a complete understanding of logical or performance features. However, testing is a necessary and fundamental step to reveal certain obvious and unexpected performances. Special attention should be paid to the performance under 'boundary conditions'. It is important to ensure that a program or a procedure should not be fatal in 'boundary conditions' even if it has to perform in a degraded way, for example by loss of accuracy. One example is a routine by Cooper (1968) which cannot handle 'boundary conditions' (small t-values with large degrees of freedom). This routine can be said to have been subject to experiment but not tested. The claim of 11 decimal places accuracy is doubtful.

Procedures are developed and tested independently wherever possible . However, not all procedures can be tested independently because they rely on or require information from other procedures. The majority of procedures, usually also the more difficult procedures, can be tested independently. After each procedure has been tested, the program is integrated by including these modules. At the beginning, some modules may be empty or have only a few statements which may be deleted at a later stage. For example, one needs only simple input and

output for testing some parts of the program. The intended input and output modules can be integrated at a later stage when necessary. It is obvious that the question-answer module can be the last one to be integrated. Each procedure is integrated into its 'residence' module only when it is necessary. This can also save us hours, or even days, of unnecessary compilations and re-compilations for debugging and testing. Any new errors are almost certainly due to the inclusion of new procedures.

(B) Program verifications.

Basically, there are two approaches to verification, the static approach and the constructive approach. The static approach regards a program as a mathematical object and uses assertions and mathematical proofs. For a 'large' program, the static approach is not practical and thus the second approach is adopted. Verification is done through careful construction.

Section 4.7. The use of the range-check option.

The UCSD Pascal system has a compiled-time option which allowed us to turn off the range check. If a (small) procedure has been analysed and validated it may be sensible to turn off the range-check if the procedure is time-critical. For example, it is sensible to turn off the range-check in the randomization test procedure because of its amount of computations. However, it is not sensible to turn off the range-check for a 'large' program to minimise the computing time as program testing can never show the absence of bugs.

Section 4.8. Program optimisation.

The most important property of any computer program is its correctness. If a program is not correct, optimisation will be meaningless. Program structure has a tremendous effect on program correctness (which is the main objective of structured programming). Optimisation must therefore take program structure into consideration. It is insufficient to optimise a program for computing time and storage. If a program or procedure has been validated, one can then transform it into a more efficient program or procedure. However, in the process of transformation (optimisation), it is important that its correctness must be maintained.

Optimization should also take maintenance into consideration. Thus clarity of a program should not be traded off against speed and storage. If a time-critical procedure cannot be reconciled with clarity, it is important to make such a procedure 'disposable'. If a maintenance programmer has difficulty in understanding it, he can then dispose of the procedure and replace it by a new procedure or a better algorithm. Disposability is a desirable property.

The switching off of a range-check is of course a potentially dangerous way of optimisation against time. One should not do this unnecessarily. Unless it is certain that no polynomial-time algorithm can possibly exist, it is better to look for a new algorithm if the current algorithm is far too slow. An exponential-time algorithm behaves quite independently of computer power.

Section 4.9. Historical references.

This section concerns various points which have been learnt through producing the program to which reference can be made in the future and mistakes may possibly be avoided.

(A) Computing experiences.

(a) The use of a microcomputer as a machine : It was a mistake to use a microcomputer as a machine to develop the program. It may be reasonable to use microcomputers to develop 'small' programs. They are not suitable for the development of 'large' programs. Microcomputers may be cheap, but they are very expensive in terms of man power. For example, it takes more than 15 minutes to compile a program of three thousand lines. If a compiled listing is required, then it would take more than half an hour. Many may consider this as reasonable, but it is unwise to use a microcomputer as there are more powerful machines equipped with powerful software, for example editor and file management. This does not mean one should not use microcomputers at all. A better approach may be to develop the programs on larger machines and then transfer them to a microcomputer if they are to be run on it. However, one may argue that one has to develop programs on the target machines because all machines have their own peculiarities. This argument is not necessarily true as it is possible to develop parts which are different from 'standard' languages on the target machines. In the case of Pascal, input and output are the least well-defined, and one may develop input/output module on the target machine and develop the other modules on a larger machine.

(b) Coding was done too early : Coding should have been delayed as long as possible as it is the simplest part of program development. More time should have been spent on the design (including the design of algorithms) so that changes after coding are minimised. Program testing is one of the most time-consuming activities in program production. It is unwise to spend time on debugging as bugs should have been avoided in the first place. Careful thought and design are even more important if the development is on a microcomputer as testing is a lot harder in terms of error messages, time taken and system software facilities.

Some may argue that program development is an evolutionary process and that a complete design is not possible. One can always find something which should have been added or coded in a more understandable way. This can be very true when there is no historical reference to which one can refer but it should not preclude the need of design.

(B) Changes.

(a) The estimation of the power of a transformation :

The Box-Cox estimation of the power of a transformation was first programmed; it was found that a considerable amount of time was needed to do all the computations.

This was later changed to Anscombe's estimation which requires much less computations and is more suitable for an interactive program.

(b) A test of accuracy of the data : A chi-square test of accuracy on the distribution of the last digits of the data points was originally programmed. No definite advice can be given to the users as the accuracy largely depends on the kind of data. The test can be very crude as data are entered in the free format. Equally important, users are likely to ignore this kind of advice; for this reason, it was later deleted.

Chapter Five

Program details

This chapter describes the details of the program. Examples of outputs and the listing of the program are in the Appendix.

Section 5.1. Introduction.

It is generally known that the most authoritative documentation of a program is the program text itself, not any comments inserted in the text to explain computational processes. If the program text itself is not readable, comments serve little purpose.

Theoretically, one should document a program during the coding process. In practice, one may choose to code first and document later.

Section 5.2. Testing of assumptions.

(a) Testing of normality.

Data are declared "normal" if both the Shapiro-Wilk test and the test based on the g_1-statistic do not detect any departure from normality. The significance level for both tests is 5%.

(b) Testing of the equality of variances.

Variances are declared "unequal" if the coefficient of variation of variances is greater than or equal to 1 or variances are significantly different at 5 % level of significance detected by test statistics.

Section 5.3. Conditions for use of the test statistics.

The conditions for the use of each test are very arbitrary and some may seem unreasonable. For example, one may argue that it is impossible to assess the distributional properties or the equality of variances if sample sizes are small. However, conditions have to be set for each test for practical purposes. The details of each individual test are given in chapter 2
In all cases, non-parametric tests may be used only on original (untransformed) data.

(A) Testing of single mean.

(a) The Student's t-test.

This test is used if one or more of the following conditions are met.

(1) Data are normally distributed.

(2) Sample size is at least 80.

(3) Sample size is at least 15 and data are symmetrically distributed.

(4) When non-parametric tests are not used.

(b) The signed-rank Wilcoxon test.

This test is used if all the following conditions are met.

(1) Data are paired-observations.

(2) Data are not normally distributed.

(3) Sample size is less than 80, but greater than 15 (excluding zeros).

(c) The one-sample randomisation test.

This test is used if the conditions (1) and (2) for the signed-rank Wilcoxon test are met and the sample size (excluding zeros) is not more than 15.

(B) Testing of two means.

(a) The Wilcoxon rank sum test.

This test is used if all the following conditions are met.

(1) Data are scores.

(2) Data are symmetrically, but not normally distributed and variances are equal.

(3) At least one of the group has a sample size of at least 10.

(b) The two-sample randomisation test.

This test is used if both the following conditions are met.

(1) Data are symmetrically but not normally distributed.

(2) Both sample sizes are less than 10.

(c) The two-sample t-test.

This test is used if nonparametric tests are not used and one or more of the following conditions are met.

(1) Variances are equal.

(2) Sample sizes are equal.

(3) At least one of the groups has a sample size of less than 10.

(d) The Welch t-test.

This test is used when none of the other three tests is suitable.

(C) Testing of several means.

(a) The Kruskal-Wallis test.

This test is used if all the following conditions are met.

(1) Data are scores.

(2) Data are symmetrically but not normally distributed and variances are equal.

(3) The average sample size is not less than 4, that is sample sizes are not too "small".

(b) The F-test.

This test is used if the Kruskal-Wallis test is not used and one or more of the following conditions are met.

(1) Data are normally distributed and variances are equal.

(2) Sample sizes are equal and the coefficient of variation of variances is less than 1.

(3) At least one of the samples has a sample size of less than 10.

(c) The Welch F-test.

This test is used if the other two tests are not used.

Section 5.4. Program documentation

Title : Mean

Date : June 1982.

Machine : Apple II plus microcomputer with 64 K of memory.

Medium : Both source codes and object codes on disk.

Language : Apple UCSD Pascal (Version II.1)

Synopsis : A program for

(1) Calculation of mean, median, standard deviation, standard error of mean, maximum and minimum, the range and Fisher's g-statistics.

(2) Testing the equality of means for one, two or more samples.

It consists of the following test statistics :

(1) One and two sample Student t-test.

(2) One and two sample randomisation tests.

(3) Two sample Welch t-test.

(4) Signed-rank Wilcoxon and Wilcoxon rank sum tests.

(5) Kruskal-Wallis test.

(6) Analysis of variance F-test and Welch F-test.

(7) Multiple comparisons.

The program also gives confidence intervals for one and two sample problems where parametric tests are used.

P-values are also given.

Description :

The program examines data provided by the users and selects a test statistic for testing the equality of means or provides elementary calculations of statistics. For elementary statistical calculations (without testing a hypothesis), a comment is issued to users if data are skewed. For testing hypotheses, a warning is issued to users if any departure from the underlying statistical assumptions is detected. For testing the equality of several means, multiple comparisons are performed at one of the levels 0.01, 0.05, 0.10, 0.15, or 0.20 if the p-value is less than 0.20. The level is

chosen such that it is the smallest of the values given above which is greater than the p-value.

The program also suggests a transformation for data if a suitable transformation is found in order to achieve normality or the equality of variances. Alternatively, users may choose to transform their data themselves. Transformations provided for are square root, logarithmic, reciprocal and arcsine and users are free to choose their own transformation from these four.

An analysis on the original data is always given. A note is given to users if a transformation of the data has been made.

A comment is also given in each of the following situations :

(1) One or more samples have data with values at least half of which are equal.

(2) One or more samples are not symmetrically distributed.

(3) Outlying observations are present where "outlying" is as defined in 2.10 of chapter 2.

Inputs :

(A) Inputs from the keyboard.

Inputs are interactive and all inputs from the key-board are validated by the program.

The three commands which can be used at any point

of the program are :

(1) HELP or help : This gives users suitable help at any point of the program. No help will be given if the problem is trivial.

(2) QUIT or quit : This stops the program and returns to the operating system level. It is not an interrupt command.

(3) REJ or rej : This is a backward eliminator. It rejects inputs backwardly one by one. Users are asked to re-type their input if an input is rejected.

Users need not have to count the number of observations for each group, the input terminator for data for a group is END or end. The program counts the number of data points for a group and it responds interactively to the users.

Data can be validated sample by sample and the following features are provided :

(1) Display of data on the screen.

(2) Making corrections.

(3) Making deletions.

(4) Making additions.

The program interprets all strings starting with the above three commands and the input terminator as commands and the input terminator respectively.

(B) Inputs from disk files.

The validation of inputs from disk file is done by the system.

Help files which may be accessed by the procedure readfile are :

(1) "datakind.text" : for explaining kinds of data.

(2) "inform.text" : for information about the program.

(3) "paired.text" : for explaining paired-observations.

(4) "random.text" : for explaining whether or not randomisation has been done.

(5) "biased.text" : for explaining whether or not the data reflect differences of means of the intended factor.

(6) "connect.text" : for explaining whether or not cases are related.

(7) "order.text" : for explaining whether or not there is an order effect on the data.

(8) "explain.text" : for explaining the meanings of statistics.

(9) "addconst.text" : for explaining that a constant must be added before a transformation is made.

The data files which are accessed by the procedure shapirowilktest are "shapwilk.3t030" and "shapwilk.31t050". For a given sample size (n>2), the components in the files are in

following order :

(1) Coefficients $a_{i,n}$ as described in 2.9.

(2) Significance level at 5% level.

(3) Normalization factors given by Shapiro and Wilk (1968).

"shapwilk.3to30" is for sample sizes from 3 to 30 and the other is from 31 to 50.

Outputs :

The following outputs are given :

(1) The data are given

(a) in their original form for each case, or

(b) as the differences of the two members of each pair for paired data, or

(c) as the differences for each observation from a given theoretical mean, or

(d) as the differences of two observations from each case, or

(e) as transformed data of (a) or (d).

The data are given in the order in which they are entered. Sample sizes are also given.

(2) Histograms.

(3) Summary of statistics : mean, median, standard deviation, standard error, maximum and minimum values, the range and Fisher's g_1 and g_2 statistics.

(4) A test statistic and a table for the analysis of variance (if applicable).

(5) P-value.

(6) Confidence interval (if applicable).

(7) Pairs of groups with sample means which have been found to be significantly different (if applicable).

(8) Sums of ranks and means of sums of ranks for each group (if applicable).

(9) Comments or warnings.

Validation of the outputs is done by the system. Outputs can be on the screen or on the printer. No limit on the number of copies of outputs may be made. A notice is issued to users if they have not had a hard-copy output.

Restrictions :

(A) Data entry

(1) The maximum number of groups is 20.

(2) The maximum number of data for a group is the integral part of 400 divided by the number of groups.

These two restrictions can be easily changed. Data entry is restricted to the keyboard, but it can be changed by modifying the input module (with no change in other modules).

Data can only be validated sample by sample and once data for a sample are accepted, they cannot be changed. A warning is issued to users if there is no chance of further changes of data for a sample.

Range-check :

No run-time range-check is performed in the procedure randomisationtest.

Possible changes for other environments :

The following UCSD Pascal features which are either different from or not provided in the Standard Pascal are used :

(1) Function POS in the global procedure matchstr. This can be removed by writing an equivalent function.

(2) Function LENGTH in the global procedure getdata. This can be removed by writing an equivalent function.

(3) EXIT (from a procedure) in procedures verifydata, validatedata, keyindata, readdata and quit. These may be removed by the use of GOTO statements or boolean variables.

(4) Declaration of STRING as a type of packed array of characters.

(5) Provision of a procedure for STRING output.

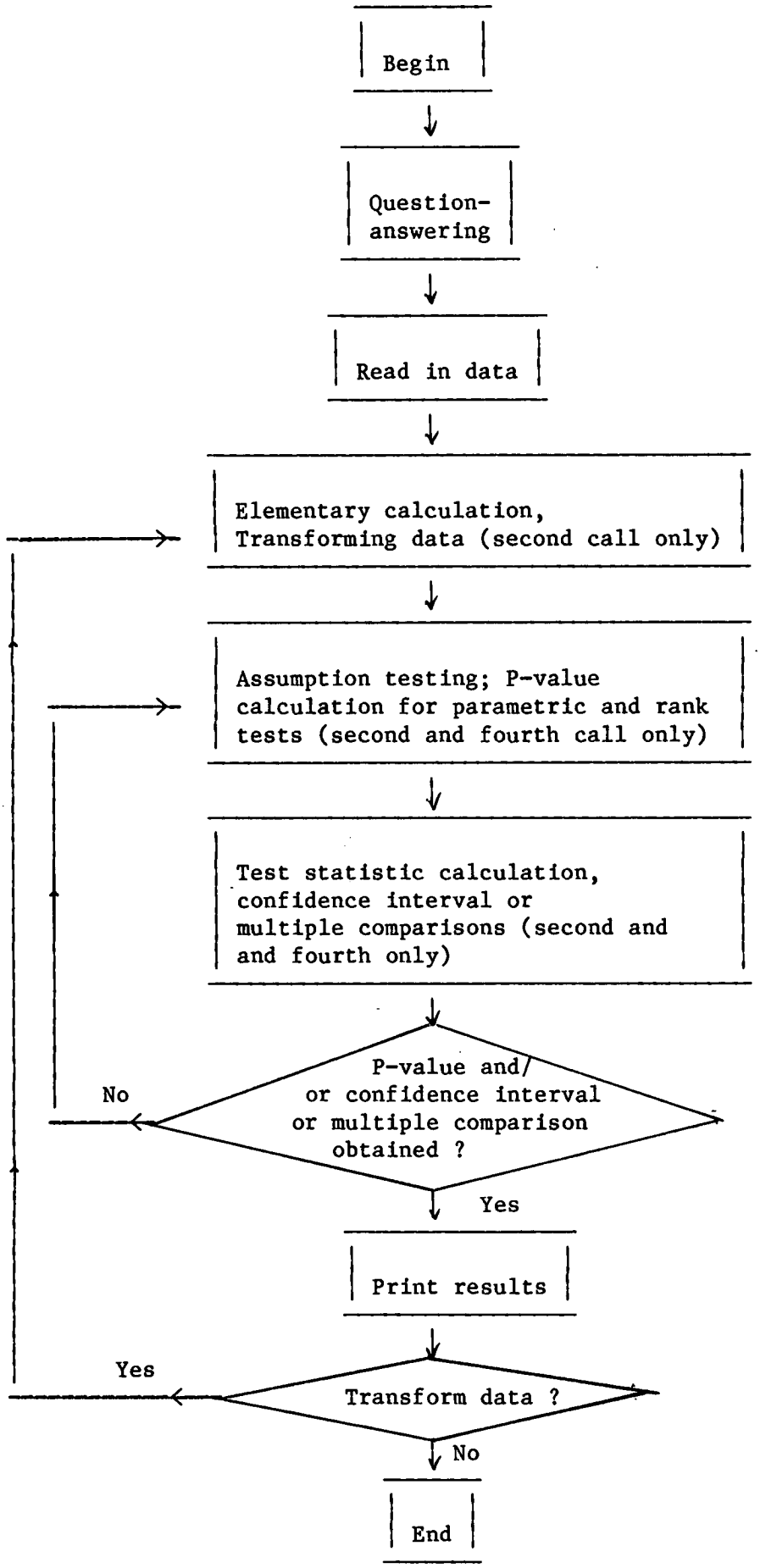
(6) Removal or replacement of the USES TRANSCEND.

(7) The overlay feature SEGMENT.

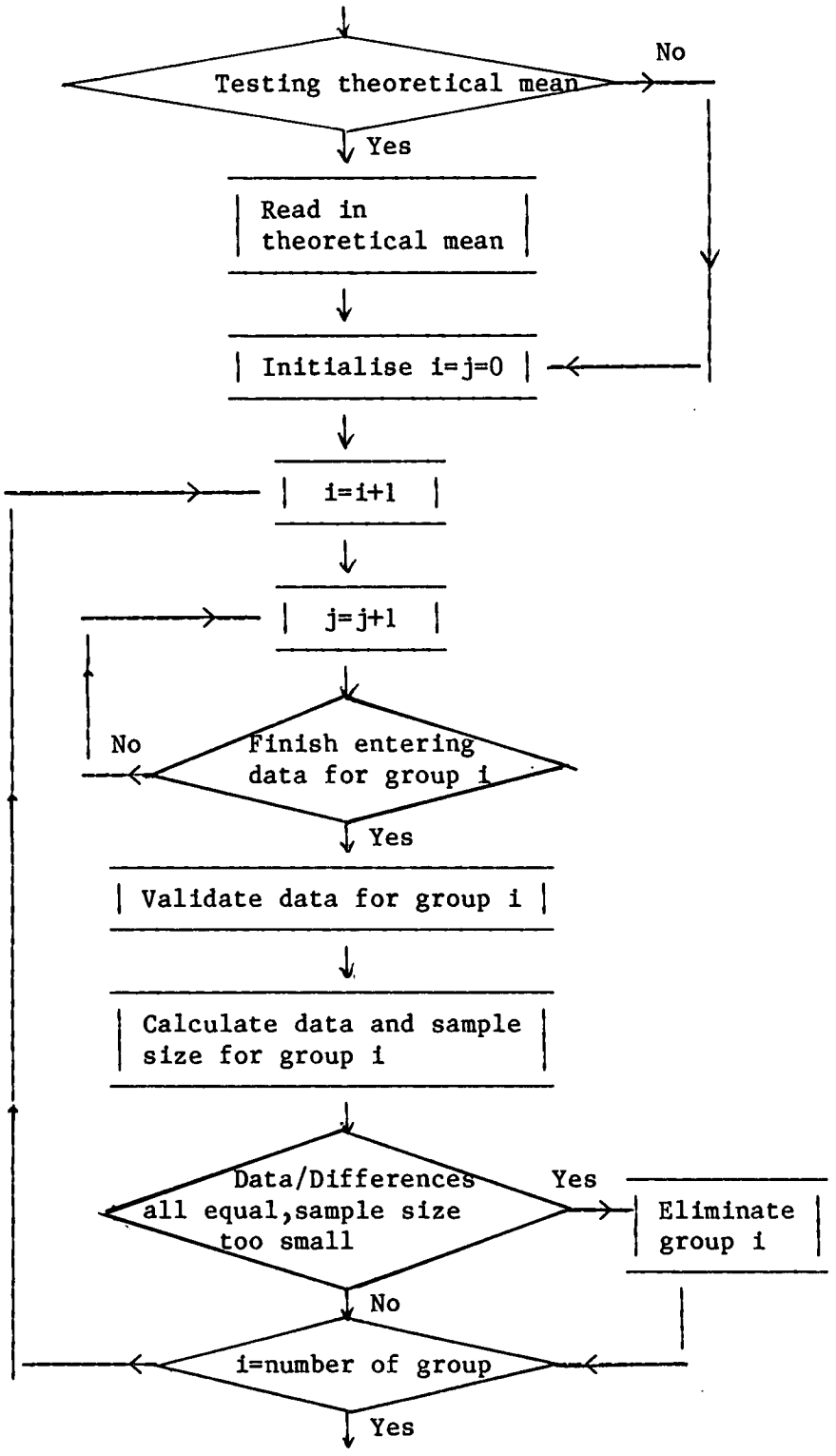
(8) Change ATAN to ARCTAN in procedure dotransformation.

Flow-charts :

(A) Calling sequence of modules for testing hypothesis.


Begin
Question-answering
Read in data
Elementary calculation, Transforming data (second call only)
Assumption testing; P-value calculation for parametric and rank tests (second and fourth call only)
Test statistic calculation, confidence interval or multiple comparisons (second and and fourth only)
P-value and/ or confidence interval or multiple comparison obtained ?
No
Yes
Print results
Transform data ?
Yes
No
End

(B) Flow-chart for "read in data" module.



For other modules, the programs texts are self-explanatory.

Bug in the Apple UCSD Pascal system :

The system procedure READLN and READ may not read data properly at the end of a data disk file. A redundant number is added at the end of each (numerical) data file. READLN does not allow back-spacing for entering real numbers from the key-board.

References:

ABRAMOWITZ M. and STEGUN I. A. (Eds.) (1964)
Handbook of Mathematical Functions
U. S. Department of Commerce
National Bureau of Standards. Applied Mathematics Series
No. 55, Washington.

AKL S. G. (1981) A comparison of combination generation mehtods. ACM Trans. Math. Softw. 7, 1, 42-45.

ANSCOMBE F. J. (1961) Examination of residuals.
Fourth Berkeley Symposium on Math. and Prob.
Edited by Neyman J., 1-36.

BADGLEY R. F. (1961) An assessment of research method reported in 102 scientific articles from two Canadian medical journals. Can. Med. Assoc. J., 85, 246-250.

BAILEY B. J. R. (1981) Alternatives to Hasting's approximation to the inverse of the normal cumulative distribution function. App. Stat. 30, 3, 275-276.

BARTLETT M. S. (1937) Properties of sufficiency and statistical tests. Proc. Roy. Soc. 160, 268-282.

BERCHTOLD (1979) A program for the Wilcoxon Signed Rank test. Biometrical J. 21, 2, 167-169.

BOWMAN K. O. and SHENTON L. R. (1975) Omnibus test contour for departure from normality based on $\sqrt{b_1}$ and b_2. Biometrika, 62, 243-250.

- - - - - - , BEAUCHAMP J. and SHENTON L. (1977) The distribution of t-Statistic under non-normality. Inter. Stat. Rev. 45, 233-242.

BOX G. E. P. (1953) Non-normality and test of variances. Biometrika, 40, 318-335.

- - - - - - (1954) Some theorems on quadratic forms applied in the study of analysis of variance problem I. Effect of inequality of variance in one-way classification. Ann. Math. Stat. 25, 290-302.

- - - - - - and COX D. R. (1964) An analysis of transformations (with discussion). J. Roy. Stat. Soc. B 26, 211-252.

BROWN M. B. and FORSYTHE A. B. (1974) The small sample behaviour of some statistics which test the equality of several means. Technometrics, 16, 1, 129-132.

CARNAHAN B., LUTHER H. A. and WILKS J. O. (1969)
Applied numerical methods.
John Wiley & Sons Inc.

CHEN E. H. (1971) The power of the Shapiro-Wilk W test for normality from the contaminated normal distribution. J. Amer. Stat. Assoc. 66, 760-762.

CLAYPOOL P. L. and HOLBERT D. (1974) Accuracy of Normal and Edgeworth approximations to the distribution of the Wilcoxon signed rank statistic. J. Amer. Stat. Assoc. 69, 255-258.

CONOVER W. J. (1973) On methods of handling ties in the Wilcoxon Signed-Rank procedure. J. Amer. Stat. Assoc. 68, 985-988.

- - - - - - and IMAN R. L. (1981) Rank transformation as a bridge between parametric and non-parametric statistics. The Amer. Stat. 35, 3, 124-129.

COOPER B. E. (1968a) Algorithm AS 2 : The normal integral. App. Stat. 17, 186-188.

- - - - - - (1968b) Algorithm AS 3 : The integral of Student t-distribution. App. Stat. 17, 189-190.

CRESSIE N. (1980) Relaxing assumptions in one sample t-test. The Aust. J. Stat. 22, 2, 143-153

DANIEL W. W. (1978)
Applied nonparametric Statistics.
Houghton Mifflin Company.

D' AGOSTINO R. B. (1970) Transformation to normality of null distribution of g_1. Biometrika, 57, 679-681.

DIJKSTRA J. B. and WERTER P. S. P. J. (1981) Testing the equality of several means when the population variances are unequal. Comm. Stat. Simul. Comp. B10 (6), 557-569.

DUNNETT C. W. (1980a) Pairwise multiple comparisons in the homogeneous variances, unequal sample size case. J. Amer. Stat. Assoc. 75, 789-795.

- - - - - - (1980b) Pairwise multiple comparisons in the unequal variances case. J. Amer. Stat. Assoc. 75, 796-800.

EL LOZY M. (1982) Efficient computation of the distribution functions of Student's, Chi-squared and F to moderate accuracy. J. Stat. Comp. Simul. 14, 179-189.

FEIR-WALSH B. J. and TOOTHAKER L. E. (1974) An empirical comparison of the ANOVA F-test, normal scores test and Kruskal-Wallis test under violation of assumptions. Educational and Psychological measurement, 34, 789-799.

FINCH D. J. (1950) The effect of non-normality on the z-test when used to compare the variances in two populations. Biometrika, 37, 186-189.

FINUCAN H. M. (1964) A note on kurtosis. J. Roy. Stat. Soc. B26, 111-112.

FISHER R. A. and CORNISH E. A. (1960) The percentile points of distributions having known cumulants. Technometrics, 2, 209-226.

GANS D. J. (1981) Use of a preliminary test in comparing two sample means. Comm. Stat. Simul. Comp. B10, 2, 163-173.

GARTSIDE P. S. (1972) A study of methods for comparing several variances. J. Amer. Stat. Assoc. 67, 342-346.

GAYEN A. K. (1950) The distribution of the variance ratio in random samples of any size drawn from non-normal universes. Biometrika, 37, 236-255.

GEARY R. C. (1947) Testing for normality. Biometrika, 34, 209-242.

GENG S., WANG W. J. and MILLER C. (1979) Small sample size comparisons of tests for homogeneity of variances by Monte-Carlo. Comm. Stat. Simul. Comp. B8, 4, 379-389.

GENTLEMAN J. F. (1975) Algorithm AS 88 : Generation of nCr combinations by simulating nested FORTRAN DO loop. App. Stat. 24, 374-376.

- - - - - - - (Ed.) (1979)
Proceed. of the Comp. Sci. and Stat.: 12th Annual Symposium on the Interface.
University of Waterloo, Waterloo, Ontario, Canada.

GLANTZ S. A. (1980) Biostatistics : How to detect, correct and prevent errors in the medical literature. Circulation, 61, 1, 1-6.

GORE S., JONE I. G. and RYTTER E. C. (1977) Misuses of statistical methods: Critical assessment of articles in B. M. J. from January to March 1976. Br. Med. J. 1, 85-87.

GREEN B. F. (1979) A practical interactive program for randomization tests of of location. The Amer. Stat. 31, 37-39.

- - - - - (1980) FORTRAN routine for randomization tests (private communication). Department of Psychology, The Johns Hopkins University, Baltimore, Maryland 21218, U. S. A.

GRIFFIN R. and REDISH K. A. (1970) Remark on Algorithm 347 [M1]: An efficient algorithm for sorting with minimal storage. Comm. Assoc. Comp. Mach. 13, 54.

HARTLEY H. O. and PEARSON E. S. (1950a) Tables of probability of integral of the t-distribution. Biometrika, 37, 168-172.

- - - - - - and - - - - - - - (1950b) Tables of the x^2-integral and of the cumulative Poisson distribution. Biometrika, 37, 313-325.

HENRICI P. (1964)
Elements of numerical analysis.
Wiley International Edition.
John Wiley & Sons.

HILGERS R. (1982) On the Wilcoxon-Mann-Whitney-test as a nonparametric analogue and extension of t-test. Biometrical J. 1, 24, 3-15.

HILL G. W. (1970) Algorithm 395 : Student's t-distribution [S14]. Comm. Assoc. Comp. Mach. 3, 10, 617-619.

HILL I. D. (1973) Algorithm AS 66 : The normal integral. App. Stat. 22, 424-427.

HOWDEN W. E. (1980) Applicability of software validation techniques to scientific programs. ACM Trans. Prog. Lag. and Sys. 2, 3, 307-320.

IMAN R. L. and DAVENPORT J. M. (1976) New approximation to the exact distribution of the Kruskal-Wallis test statistic. Comm. Stat. Theory and Method, A5, 14, 1335-1348.

JAMES G. S. (1951) The comparison of several groups of observations when the ratio of population variances are unknown. Biometrika, 38, 324-329.

KENNEDY W. J. JR. and GENTLE J. E. (1980)
Statistical Computing.
Marcel Dekker Inc. New York and Basel.

KUMMER G. (1981) Formula for the computation of the Wilcoxon test and other rank test. Biometrical J. 23, 3, 237-243.

LAYARD M. W. J. (1973) Robust large sample tests for homogeneity of variances. J. Amer. Stat. Assoc. 68, 195-198.

LAUER G. N. and HAN C. P. (1974) Power of Cochran's test in Behrens-Fisher problem. Technometrics, 16, 545-549.

LEE A. F. S. and GURLAND J. (1975) Size and power of test of equality of means of two normal populations with unequal variances. J. Amer. Stat. Assoc. 70, 933-941.

- - - - - - and - - - - - - (1977) One sample t-test when sampling from a mixture of normal distribution. Ann. Stat. 5, 4, 1, 802-807.

LEHMANN E. L. (1975)
Nonparametrics : statistical methods based on ranks.
Holden Day Inc.
Mcgraw Hill International Book Company.

LING R. F. (1974) Comparison of several algorithms for computation means and variances. J. Amer. Stat. Assoc. 69, 859-866.

- - - - - (1978) A study of the accuracy of some approximations for t, x^2, and F tail probabilities. J. Amer. Stat. Assoc. 73, 274-283.

MICKEY M. R. and BROWN M. B. (1966) Bounds on the distribution function of Behren-Fisher statistics. Ann. Math Stat. 37, 639-642

MIFSUD C. J. (1963) Algorithm 154 : Combination in lexicographical order. Comm. Assoc. Comp. Mach. 6, 3, 103.

MURPHY B. P. (1967) Some two sample tests when the variances are unequal : A simulation study. Biometrika, 54, 679-683.

MORAN P. A. P. (1980) Calculation of the normal distribution function. Biometrika, 67, 675-676.

NEELY P. M. (1966) Comparison of several algorithms of computation of means, standard deviations and correlation coefficients. Comm. Assoc. Comp. Mach. 7, 496-499.

NELDER J. A. (1976) Algorithm AS 96 : A simple algorithm for scaling graphs. App. Stat. 25, 1, 94-96.

PAULSON E. (1942) An approximation normalization of the analysis of variance distribution. Ann. Math. Stat. 13, 233-235.

PAGE E. S. and WILSON L. B. (1979)
An introduction to computational combinatorics.
Cambridge Computer Science Text 9.

PEARSON E. S. and HARTLEY H. O. (Eds.) (1972)
Biometrika tables for statisticians, Volume 2.
Cambridge University Press.

- - - - - - , D' AGOSTINO R. B. and BOWMAN K. O. (1977) Tests for departure from normality : Comparison of power. Biometrika, 64, 231-246.

- - - - - - and PLEASE N. (1975) Relation between the shape of population distribution and the robustness of four simple statistics. Biometrika, 62, 223-241.

PITTNER P. M. (1981) An algorithm for the Mann-Whitney U-test. Biometrical J. 23, 1, 105-107.

POSTEN H. A. (1978) The robustness of the two-sample t-test over the Pearson-system. J. Stat. Comp. Simul. 6, 295-311.

- - - - - - -(1979) The robustness of the one sample t-test over the Pearson-system. J. Stat. Comp. Simul. 9, 133-149.

PRATT J. W. (1959) Remarks on zeros and ties in the Wilcoxon Signed-Rank procedures. J. Amer. Stat. Assoc. 54, 655-667.

- - - - - (1964) Robustness of some procedures for the two sample location problem. J. Amer. Stat. Assoc. 59, 665-680.

- - - - - and GIBBONS J. D. (1981)
Concepts of nonparametric theory.
Springer series in statistics, Springer-Verlag.

RACTCLIFFE J. F. (1968) The effect on the t-distribution of non-normality in the sampled population. App. Stat. 17, 42-48.

REID A. J. and LEMON J. S. (1980) A review of interactive statistical packages in British universities and polytechnics. COMPSTAT 80, Proced. in Comp. Stat. Physica-Verlag.Wien.

REINGOLD E. M., NIEVERGELT J. and DEO N. (1977)
Combinatorial algorithms.
theory and practice.
Prentice-Hall, Inc. Englewood, Cliff,
New Jersey 07632.

SAHAI H. and THOMPSON W. O. (1974) Comparisons of approximations to percentiles of t, x^2 and F-distributions.
J. Stat. Comp. Simul. 3, 81-93.

SALES D. J. (1980) Testing for inconsistent uses of variates in regression models. COMSTAT 80, Proceed. in Comp. Stat. Physica-Verlag Wein.

SCHOR S. and KARTEN I. (1966) Statistical evaluation of medical journal manuscripts. J. Amer. Med. Assoc. 195, 1123-1127.

SCHEFFE H. (1959)
The analysis of variance.
John Wiley & Sons In. New York.

SEBER G. A. F. (1980)
The linear hypothesis : a general theory.
Griffin's Statistical Monographs and Courses No. 19.
(second edition).

SHAMOS M. I. (1976) Geometry and Statistics : Problems at the interface. in Algorithm and Complexity new directions and recent results. Edited by Traub J. F., 251-280.

SHAPIRO S. S. and WILK M. B. (1965) An analysis of variance test of normality (complete sample). Biometrika, 52, 591-611.

- - - - - - and - - - - - - (1968) Approximation for the null distribution of the W-statistic. Technometrics, 10, 861-866.

- - - - - - and - - - - - and CHEN M. J. (1968) A comprehensive study of various tests for normality. J. Amer. Stat. Assoc. 63, 1343-1372.

SHEN M. K. (1962) On the generation of permutations and combinations. BIT, 2, 228-231.

SINGLETON R. C. (1969) An efficient algorithm for sorting with minimal storage [M1]. Comm. Assoc. Comp. Mach. 12, 185-187.

STIGLER S. M. (1977) Do robust estimators work with real data ? (with discussion). Ann. Stat. 5, 1055-1098.

STIRLING W. D. (1981) Algorithm AS 168 : Scale selection and formatting. App. Stat. 30, 339-344.

STOLINE M. R. (1981) The status of multiple comparisons in one-way ANOVA designs. The Amer. Stat. 35, 134-141.

THAYER R. P. and STORER R. F. (1969) Algorithm AS 21 : Scale selection and formatting. App. Stat. 30, 3, 339-344.

VERDOOREN L. R. (1963) Extended tables of critical values for Wilcoxon's test statistic. Biometrika, 50, 177-186.

WALLACE D. L. (1959) Simplified Beta-approximations to the Kruskal-Wallis H-test. J. Amer. Stat. Assoc. 54, 225-230.

WELCH B. L. (1947) The generalization of "student" problem when several different population variances are involved. Biometrika, 34, 28-35.

- - - - - - (1949) Further note on Aspin's tables and on certain approximation to the tabled function. Biometrika, 36, 293-296.

WELFORD B. P. (1962) Note on a method for calculating corrected sums of squares and products. Technometrics, 4, 419-420.

WETHERILL G. B. (1960) The Wilcoxon test and non-null hypothses. J. Roy. Stat. B, 27, 402-418.

WILK M. B. and SHAPIRO S. S. (1968) The joint assessment of normality of several independent samples. Technometrics, 10, 825-839.

YOUNGS E. A. and CRAMER E. M. (1971) Choice of sums and sums of product algorithms. Technometrics, 13, 657-665.

APPENDIX : PROGRAM LISTING AND EXAMPLES

PROGRAM LISTING

```
PROGRAM MEANPROGRAM(DATAKIND.TEXT,INFORM.TEXT,PAIRED.TEXT,RANDOM.TEXT,
            BIASED.TEXT,CONNECT.TEXT,ORDER.TEXT,EXPLAIN.TEXT,
            SHAPWILK.3TO30,SHAPWILK31TO50,ADDCONST.TEXT);

(*$S+*) (*SWAPPING MODE FOR MORE SPACE*)
(************************************)
(*                                  *)
(* A PROGRAM FOR ONE EFFECT ANALYSIS *)
(*  BY CHENG-TAI GAN  JUNE, 1982.    *)
(************************************)

USES TRANSCEND (*LIBRARY ROUTINES FOR TRANSCENTAL FUNCTIONS*);

CONST MAXGROUP= 20;     (*MAXIMUM NUMBER OF GROUP   *)
      LIMIT   = 400;    (*MAXIMUM NUMBER OF ALL DATA*)

TYPE GROUPINDEX      = 1..MAXGROUP;
     DATAINDEX       = 1..LIMIT;
     DATASET         = ARRAY [DATAINDEX] OF REAL;
     GROUPSIZE       = ARRAY [GROUPINDEX] OF 0..LIMIT;
     GROUPSTAT       = ARRAY [GROUPINDEX] OF REAL;
     DATATYPE        = (SCORE,CONTINUOUS,COUNT,BINOMIAL);
     TYPEOFTRANSFORMATION  = (IDENTITY,SQUAREROOT,LOGARITHMIC,RECIPROCAL,
                              ARCSINE);

     STATISTIC  = RECORD
                    MEAN     : GROUPSTAT; (*MEAN[I]   =MEAN OF GROUP I     *)
                    MEDIAN   : GROUPSTAT; (*MEDIAN[I] =MEDIAN OF GROUP I   *)
                    VARIANCE : GROUPSTAT; (*VARIANCE[I]=VARIANCE OF GROUP I*)
                    CV       : REAL;      (*COEFFICIENT OF VARIANTION      *)
                                          (*OF ALL GROUP VARIANCES         *)
                    G1       : GROUPSTAT; (*G1[I]=FISHER'S G1-STATISTIC OF *)
                                          (*GROUP I                        *)
                    G2       : GROUPSTAT; (*G2[I]=FISHER'S G2-STATISTIC OF *)
                                          (*GROUP I                        *)
                    MINIMUM  : GROUPSTAT; (*MINIMUM[I]=MINIMUM OF DATA FOR *)
                                          (*GROUP I                        *)
                    MAXIMUM  : GROUPSTAT; (*MAXIMUM[I]=MAXIMUM OF DATA FOR *)
                                          (*GROUP I                        *)
                  END;
```

```
TEST    = RECORD
            NAME         : STRING;      (*NAME OF TEST STATISTIC          *)
                                        (*FOR ELEMENTARY CALCULATIONS,    *)
                                        (*NAME='T-STATISTIC' FOR CONS-    *)
                                        (*TRUCTING CONFIDENCE INTERVAL    *)
            VALUE        : REAL;        (*VALUE OF TEST STATISTIC         *)
            RANKSUM      : GROUPSTAT;   (*RANKSUM[I]=RANKSUM OF GROUP I  *)
                                        (*FOR ONE-SAMPLE PROBLEMS,        *)
                                        (*RANKSUM[2] IS THE SUM OF RANKS *)
                                        (*OF NEGATIVE NUMBERS             *)
            TIECORR      : REAL;        (*TIE CORRECTION FACTOR FOR RANK *)
                                        (*STATISTIC                       *)
            RANKSUMTEST  : BOOLEAN;     (*TRUE ONLY IF A RANK STATISTIC  *)
                                        (*IS USED                         *)
            RANDOMTEST   : BOOLEAN;     (*TRUE ONLY IF RANDOMIZATION      *)
                                        (*TEST IS USED                    *)
            DISTRIBUTION: (TDISTRIBUTION,FDISTRIBUTION,KRUSKALWALLIS,
                           SIGNEDWILCOXON,TWOWILCOXON,RANDOM);

            DFN          : INTEGER;     (*DEGREES OF FREEDOM OF NUMERATOR*)
                                        (*FOR PARAMETRIC TESTS            *)
            DFD          : INTEGER;     (*DEGREES OF FREEDOM OF DENOMINA-*)
                                        (*TOR FOR PARAMETRIC TESTS        *)
            NOOFNONZERO  : INTEGER;     (*NO. OF DATA NOT EQUAL TO ZERO  *)
                                        (*FOR PURPOSE OF APPROX. P-VALUE *)
                                        (*OF SIGNED RANK WILCOXON TEST    *)
            PVALUE       : REAL;        (*ONE-SIDED P-VALUE OF TEST       *)
                                        (*STATISTIC                       *)
            SIGLEV       : REAL;        (*SIGNIFICANCE LEVEL,  TWO-SIDED *)
                                        (*FOR ONE OR TWO SAMPLE PROBLEMS *)
            VALID        : BOOLEAN;     (*TRUE IF TEST IS VALID, FALSE    *)
                                        (*OTHERWISE                       *)
          END;

INTERVAL  = RECORD
              UPPERLIMIT  : REAL;       (*UPPER LIMIT OF CONFIDENCE LEVEL*)
              LOWERLIMIT  : REAL;       (*LOWER LIMIT OF CONFIDENCE LEVEL*)
            END;

GRAPH     = RECORD
              FREQUENCY     : ARRAY [GROUPINDEX,1..25] OF INTEGER;
                                        (*FREQUENCY[I,K]=FREQUENCY OF    *)
                                        (*GROUP I IN INTERVAL K          *)

              NOOFINTERVAL : 1..25;     (*NO OF INTERVAL OF HISTOGRAMS   *)
              HEIGHT       : 1..60;     (*HEIGHT OF HISTOGRAM            *)
              STEP         : REAL;      (*STEP SIZE OF INTERVALS         *)
              MAXMIDPOINT  : REAL;      (*MAXIMUM OF MIDPOINTS ON SCALE *)
              DECPL        : INTEGER;   (*NO. OF DECIMAL PLACES ON SCALE*)
              REPRESENTCASE: INTEGER;   (*NO. OF CASES AN * REPRESENTS  *)
            END;
```

```
VAR
    DESCRIPTIVESTATISTIC    : STATISTIC;
    TESTSTATISTIC           : TEST;
    CONFIDENCEINTERVAL      : INTERVAL;
    HISTOGRAM               : GRAPH;
    DATA : DATASET;  (*DATA OF ALL GROUPS WITH ORIGINAL ORDER          *)
    X    : DATASET;  (*DUPLICATE OF DATA, BUT IN ASCENDING ORDER BY    *)
                     (*GROUP, OR USE TO CARRY INFORMATION              *)

    GPSIZE  : GROUPSIZE; (*GPSIZE[I]=GROUP SIZE OF GROUP I             *)
    DATAKIND : DATATYPE; (*KIND OF DATA IS BEING PRESENTED             *)

    GROUP,       (*NUMBER OF GROUP, SET TO 1 IF PAIRED-GROUPS          *)
    TOTAL,       (*TOTAL NUMBER OF DATA CASES                          *)
    MINGPSIZE,   (*MINIMUM OF GROUP SIZES OF ALL GROUPS                *)
    MAXGPSIZE,   (*MAXIMUM OF GROUP SIZES OF ALL GROUPS                *)
    GPSIZEALLOW,(*MAXIMUM OF GROUP SIZE ALLOWED FOR EACH GROUP         *)
    OUTLIER,     (*TOTAL NUMBER OF OUTLIERS OF GROUPS                  *)
    PROBLEM,     (*PROBLEM NUMBER WHICH USERS CHOOSE                   *)
    DIFFPAIR     (*TOTAL NUMBER OF PAIRS OF GROUPS WITH                *)
                 (*DIFFERENCES IN MEANS                                *)
             : INTEGER;

    BSS,         (*BETWEEN GROUPS SUM OF SQUARES                       *)
    WSS,         (*WITHIN GROUPS SUM OF SQUARES                        *)
    MSE,         (*MEAN SQUARE ERROR                                   *)
    ADDCONST,    (*NUMBER ADDED TO EACH DATA POINT BEFORE MAKING       *)
                 (*TRANSFORMATION                                      *)
    MINDATA,     (*MINIMUM OF ALL DATA                                 *)
    THEOMEAN,    (*THEORETICAL MEAN TO BE TESTED                       *)
    KURTOSIS     (*KURTOSIS OF RESIDUALS OF ALL GROUPS                 *)
             : REAL;



    NORMAL,           (*TRUE ONLY IF ALL DATA FOR GROUPS ARE NORMAL    *)
    SYMMETRY,         (*TRUE ONLY IF ALL DATA ARE NORMAL OR DATA FOR   *)
                      (*EVERY GROUP IS SYMMETRICAL                     *)
    EQVARIANCE,       (*TRUE ONLY IF VARIANCES ARE 'EQUAL'             *)
    PAIRED,           (*TRUE ONLY IF DATA ARE PAIRED OBSERVATIONS      *)
    EXAMINEDATA,      (*CONTROL TO ASK FOR EXAMINING DATA AND/OR       *)
                      (*TESTING ASSUMPTIONS IF NEEDED                  *)
    GETTEST,          (*CONTROL TO ASK FOR COMPUTING TEST STATISTIC    *)
                      (*TRUE IF NEEDED                                 *)
    GETPVALUE,        (*CONTROL TO ASK FOR COMPUTING P-VALUE           *)
                      (*TRUE IF NEEDED                                 *)
    WANTTRANSFORM,    (*CONTROL TO ASK FOR TRANSFORMING DATA           *)
                      (*TRUE IF WANTED                                 *)
    TESTTHEOMEAN,     (*TRUE ONLY IF TESTING THEORETICAL MEAN          *)
    TAKEDIFFERENCE,   (*TRUE IF DATA ARE PAIRED OR TWO DATA FROM A     *)
                      (*CASE ARE COLLECTED, FALSE OTHERWISE            *)
    TOOMANYEQ,        (*TRUE IF AT LEAST ONE GROUP HAS HALF OR MORE OF *)
                      (*ITS DATA EQUAL, FALSE OTHERWISE                *)
    RESUME,           (*TRUE IF PROGRAM OR ANALYSIS IS CONTINUED,      *)
                      (*FALSE OTHERWISE                                *)
    NONSTOP           (*CONTINUE INFINITELY UNTIL OUT OF THIS PROGRAM  *)
                      (*ALWAYS TRUE                                    *)
             : BOOLEAN;

    TRANSFORM : TYPEOFTRANSFORMATION;
```

```
FUNCTION T(I,J : INTEGER) : INTEGER;

   (*MAPPING MULTIPLE ARRAY INTO ONE DIMENSIONAL ARRAY DATA, OR X*)

   FORWARD;

PROCEDURE QUIT(S : STRING);

   (*COMMAND: 'QUIT'*)

   FORWARD;

FUNCTION HELP(VAR S : STRING) : BOOLEAN;

   (*COMMAND: 'HELP'*)

   FORWARD;

FUNCTION REJECT(VAR S : STRING) : BOOLEAN;

   (*COMMAND: 'REJECT'*)

   FORWARD;

FUNCTION ENDING(VAR S : STRING) : BOOLEAN;

   (*INPUT TERMINATOR: 'END'*)

   FORWARD;

FUNCTION MATCHSTR(    STR    : STRING;
                  VAR SOURCE : STRING) : BOOLEAN;

   (*MATCHING STR WITH SOURCE*)

   FORWARD;

PROCEDURE READSTR(HELPREJVALID : BOOLEAN;
                  VAR ANSWER      : STRING);

   (*PROCEDURE FOR READING STRING*)

   FORWARD;
```

```
PROCEDURE READINTEGER(MIN,MAX       : INTEGER;
                      HELPREJVALID  : BOOLEAN;
                      VAR S         : STRING;
                      VAR DATUM     : INTEGER);

   (*PROCEDURE FOR READING INTERGER *)

   FORWARD;



PROCEDURE GETDATA(PROMPT,FORM : STRING;
                  LOWERBOUND,UPPERBOUND : REAL;
                  VAR S       : STRING;
                  VAR DATUM   : REAL);

  (*PROCEDURE FOR READING DATA, REAL AND INTEGER*)

  FORWARD;

PROCEDURE READFILE(FILENAME:STRING);

 (*READING FILENAME FROM DISK AND OUTPUT IT ON SCREEN*)

 FORWARD;


(*$IQUESTION.TEXT*)

(*$IREADDATA.TEXT*)

(*$ISTAT.TEXT*)

(*$IASSUMEDIST.TEXT*)

(*$I:CALCULATE.TEXT*)

(*$I#5:RESULT.TEXT*)



FUNCTION T;

(*DECLARED FORWARD..PARA:( I,J : INTEGER*)

BEGIN
   T:=(I-1)*GPSIZEALLOW+J;
END (*T*);



PROCEDURE QUIT;

(*DECLARED FORWARD..PARA:(S : STRING*)


BEGIN
   IF MATCHSTR('QUIT',S) OR MATCHSTR('quit',S) THEN
      EXIT(MEANPROGRAM);
END (*QUIT*);
```

```
FUNCTION HELP;

(*DECLARED FORWARD..PARA:(VAR S : STRING*)

BEGIN
   HELP:=MATCHSTR('HELP',S) OR MATCHSTR('help',S);
END (*HELP*);


FUNCTION REJECT;

(*DECLARED FORWARD..PARA:(VAR S : STRING*)

BEGIN
   REJECT:=MATCHSTR('REJ',S) OR MATCHSTR('rej',S);
END (*REJECT*);


FUNCTION ENDING;

(*DECLARED FORWARD..PARA:(VAR S : STRING*)


BEGIN
   ENDING:=MATCHSTR('END',S) OR MATCHSTR('end',S);
END (*ENDING*);


FUNCTION MATCHSTR;

(*DECLARED FORWARD..PARA:(STR :STRING; VAR SOURCE : STRING*)


VAR MATCH:BOOLEAN;

BEGIN
   MATCH:=POS(STR,SOURCE)=1;
   IF MATCH THEN
      SOURCE:=STR; (*STRIP SOURCE TO STR*)
   MATCHSTR:=MATCH;
END (*MATCHSTR*);
```

```
PROCEDURE READSTR;

(*DECLARED FORWARD..PARA:(HELPREJVALID : BOOLEAN; VAR ANSWER : STRING*)


BEGIN
   (*$I-*)   (*TURN I/O CHECK OFF*)
   REPEAT
      WRITE(' (Y OR N) ');
      READLN(ANSWER);
      QUIT(ANSWER);
      IF HELPREJVALID AND (REJECT(ANSWER) OR HELP(ANSWER)) THEN
         EXIT(READSTR)
      ELSE IF ENDING(ANSWER) OR REJECT(ANSWER) THEN
         WRITELN('YOU CANNOT USE  ',ANSWER,' HERE.');
      IF ANSWER='y' THEN     (*ALWAYS RETURN ANSWER IN CAPITAL LETTERS*)
         ANSWER:='Y'
      ELSE IF ANSWER='n' THEN
         ANSWER:='N';
   UNTIL (ANSWER='Y') OR (ANSWER='N');
   WRITELN;
   (*$I+*)  (*I/O CHECK BACK ON*)
END (*READSTR*);


PROCEDURE READINTEGER;

(*DECLARED FORWARD..PARA:( MIN,MAX : INTEGER; HELPREJVALID : BOOLEAN;*)
(*                         VAR S : STRING; VAR DATUM : INTEGER       *)

(*MIN & MAX ARE POSITIVE INTEGERS AND MIN<MAX                        *)

VAR P,Q,TEMPDATUM : REAL;

BEGIN
   P:=MIN-0.5;
   Q:=MAX+0.5;
   REPEAT
      (*SET BOUNDS TO LARGE VALUE TO FREE THE BOUND CHECK IN PROCEDURE*)
      (*GETDATA.                                                       *)
      GETDATA('TYPE IN A NUMBER. ','INTEGER',-1.0E37,1.0E37,S,TEMPDATUM);
      IF HELPREJVALID AND (REJECT(S) OR HELP(S)) THEN
         EXIT(READINTEGER)
      ELSE IF ENDING(S) OR REJECT(S) THEN
         WRITELN('YOU CANNOT USE ',S,' HERE.')
      ELSE IF (P<TEMPDATUM)<>(TEMPDATUM<Q) THEN
         WRITELN('MUST BE BETWEEN ',MIN,' AND ',MAX,' INCLUSIVE.');
   UNTIL (P<TEMPDATUM)=(TEMPDATUM<Q);
   DATUM:=ROUND(TEMPDATUM);
   WRITELN;
END (*READINTEGER*);
```

```
PROCEDURE GETDATA;

(*DECLARED FORWARD..PARA:( PROMPT,FORM : STRING            *)
(*                         LOWERBOUND,UPPERBOUND : REAL    *)
(*                         VAR S : STRING; VAR DATUM : REAL*)


VAR SIGN,LEN,I,SPACE : INTEGER;
    BLANK,SUCCESS : BOOLEAN;
    P : REAL;


    PROCEDURE NEXTCHAR;

    BEGIN
       SUCCESS:=LEN=I;
       IF NOT SUCCESS THEN
          I:=I+1;
    END (*NEXTCHAR*);

BEGIN
   (*$I-*)  (*TURN I/O CHECK OFF*)
   REPEAT
      REPEAT
         WRITE(PROMPT);
         READLN(S);
         LEN:=LENGTH(S);
         BLANK:=TRUE;
         IF LEN>30 THEN
            WRITELN('>>ENTRY TOO LONG.')
         ELSE
            WHILE (LEN>0) AND BLANK DO
               IF S[LEN]=' ' THEN   (*ELIMINATE SPACES AT THE BACK*)
                  LEN:=LEN-1
               ELSE
                  BLANK:=FALSE;
      UNTIL NOT BLANK;

      QUIT(S);
      IF HELP(S) OR ENDING(S) OR REJECT(S) THEN
         EXIT(GETDATA);

      (*START CONVERT S TO A NUMBER, DATUM*)

      SUCCESS:=FALSE;
      DATUM:=0;
      SIGN:=1;                      (*SIGN=1 MEANS POSITIVE*)
      I:=1;
      WHILE S[I]=' ' DO             (*ELIMINATE SPACES IN THE FRONT*)
         I:=I+1;
      SPACE:=I-1;                   (*NUMBER OF SPACES*)
      IF I<LEN THEN                 (*CHECK SIGN*)
         IF (S[I]='-') THEN BEGIN
            SIGN:=-1;
            I:=I+1;
         END ELSE IF S[I]='+' THEN
            I:=I+1;
```

```
                                    (*INTEGRAL PART*)
      WHILE (NOT SUCCESS) AND  (S[I] IN ['0'..'9']) DO BEGIN
         DATUM:=10*DATUM+ORD(S[I])-ORD('0');
         NEXTCHAR;
      END;

                                    (*FRACTIONAL PART*)
      IF FORM='NUMERIC' THEN BEGIN
         IF (S[I]='.') AND (LEN-SPACE>1) THEN
            NEXTCHAR;
         P:=1.0;
         WHILE (NOT SUCCESS) AND (S[I] IN ['0'..'9']) DO BEGIN
            P:=P*1.0E-1;
            DATUM:=DATUM+(ORD(S[I])-ORD('0'))*P;
            NEXTCHAR;
         END;
      END;

      IF SUCCESS THEN BEGIN
         DATUM:=SIGN*DATUM;
         SUCCESS:=((LOWERBOUND<=DATUM)=(DATUM<=UPPERBOUND));
         IF NOT SUCCESS THEN
            WRITELN('MUST BE BETWEEN ',LOWERBOUND,' AND',UPPERBOUND:10:2,
                    ' INCLUSIVE.');
      END ELSE BEGIN                (*REPORT ERROR*)
         WRITELN('^':(LENGTH(PROMPT)+I),' ERROR ! ');
         WRITELN(FORM,' OR COMMAND OR INPUT TERMINATOR EXPECTED.');
      END;
   UNTIL SUCCESS;
   (*$I+*)  (*I/O CHECK BACK ON*)
END (*GETDATA*);


PROCEDURE READFILE;

 (*DECARED FORWARD..PARA:( FILENAME : STRING*)

VAR F : TEXT;
    S : STRING;

BEGIN
   (*$I+*)  (*I/O CHECK ON*)
   RESET(F,FILENAME);
   WHILE NOT EOF(F) DO BEGIN
      READLN(F,S);
      WRITELN(S);
   END;
   CLOSE(F);
END (*READFILE*);
```

```
BEGIN                (*        MAIN PROGRAM        *)
   NONSTOP:=TRUE;
   WHILE NONSTOP DO BEGIN
      QUESTION(TESTSTATISTIC,
               PROBLEM,GROUP,
               THEOMEAN,ADDCONST,
               DATAKIND,
               TRANSFORM,
               WANTTRANSFORM,PAIRED,TAKEDIFFERENCE,
               TESTTHEOMEAN,GETTEST,GETPVALUE,RESUME);

      IF RESUME THEN
         READDATA(DATA,
                  GPSIZE,
                  GROUP,TOTAL,GPSIZEALLOW,
                  DATAKIND,
                  THEOMEAN,
                  PAIRED,TAKEDIFFERENCE,
                  TESTTHEOMEAN,RESUME);

      WHILE RESUME DO BEGIN
         EXAMINEDATA:=TRUE;
         RESUME:=FALSE;

         BASICSTAT (DATA,X,
                    DESCRIPTIVESTATISTIC,
                    TESTSTATISTICS,
                    HISTOGRAM,
                    GPSIZE,
                    GROUP,TOTAL,
                    OUTLIER,MINGPSIZE,MAXGPSIZE,
                    BSS,WSS,MSE,KURTOSIS,MINDATA,ADDCONST,
                    TRANSFORM,
                    WANTTRANSFORM,PAIRED,TOOMANYEQ);
         REPEAT
            ASSUMPTIONDISTRIBUTION(EXAMINEDATA,GETPVALUE,
                                   X,
                                   DESCRIPTIVESTATISTIC,
                                   TESTSTATISTIC,
                                   GPSIZE,
                                   PROBLEM,GROUP,TOTAL,
                                   BSS,WSS,MSE,KURTOSIS,
                                   NORMAL,EQVARIANCE,SYMMETRY);
            CALCULATION(GETTEST,
                        DATA,X,
                        DESCRIPTIVESTATISTIC,
                        TESTSTATISTIC,
                        CONFIDENCEINTERVAL,
                        TRANSFORM,
                        GPSIZE,
                        DIFFPAIR,
                        BSS,MSE,
                        GROUP,TOTAL,MINGPSIZE,
                        MAXGPSIZE,GPSIZEALLOW,
                        NORMAL,EQVARIANCE,
                        SYMMETRY,
                        TESTTHEOMEAN,
                        DATAKIND);


         UNTIL NOT GETPVALUE;
```

```
        PRINTRESULTS(DATA,X,
                     DESCRIPTIVESTATISTIC,
                     TESTSTATISTIC,
                     HISTOGRAM,
                     CONFIDENCEINTERVAL,
                     GPSIZE,
                     DATAKIND,
                     PROBLEM,GROUP,TOTAL,OUTLIER,DIFFPAIR,
                     THEOMEAN,BSS,WSS,MSE,MINDATA,ADDCONST,
                     TRANSFORM,
                     PAIRED,TAKEDIFFERENCE,NORMAL,SYMMETRY,
                     TESTTHEOMEAN,TOOMANYEQ,
                     WANTTRANSFORM,RESUME);
        IF RESUME THEN BEGIN (*POSSIBLE ONLY FOR PROBLEM 2*)
           GETTEST:=TRUE;
           GETPVALUE:=TRUE;
        END;
      END;
   END;
END (*MAIN PROGRAM*).
```

```
SEGMENT PROCEDURE QUESTION(VAR TESTSTATISTIC : TEST;
                           VAR PROBLEM,GROUP : INTEGER;
                           VAR THEOMEAN,ADDCONST : REAL;
                           VAR DATAKIND      : DATATYPE;
                           VAR TRANSFORM     : TYPEOFTRANSFORMATION;
                           VAR WANTTRANSFORM,PAIRED,TAKEDIFFERENCE : BOOLEAN;
                           VAR TESTTHEOMEAN,GETTEST,GETPAVLUE    : BOOLEAN;
                           VAR RESUME                            : BOOLEAN);



VAR POINT : INTEGER; (*DATA POINT PER CASE *)



PROCEDURE CONSULTSTATISTICIAN;

   FORWARD;

PROCEDURE DEFINITION;

   FORWARD;


PROCEDURE PAUSE;

 (*PAUSE A WHILE TO INSTRUCT USERS*)

   FORWARD;


PROCEDURE CLEAR;


BEGIN
   PAIRED:=FALSE;
   RESUME:=FALSE;
   THEOMEAN:=0.0;
   ADDCONST:=0.0;
   TRANSFORM:=IDENTITY;
   WANTTRANSFORM:=FALSE;
END;



PROCEDURE CHOOSEPROBLEM(VAR PROBLEM : INTEGER);

VAR DUMMYSTR : STRING;

BEGIN
   PAGE(OUTPUT);
   WRITELN;
   WRITELN('THIS PROGRAM CAN DEAL WITH THE FOLLOWING PROBLEMS.');
   WRITELN;
   WRITELN('WHICH ONE OF THESE IS YOUR PROBLEM ?');
   WRITELN;
   WRITELN('1. CALCULATIONS OF MEAN, STANDARD DEVIATION, MEDIAN, RANGE,');
   WRITELN('   NO TESTING HYPOTHESIS.');
   WRITELN('2. COMPARISON OF MEANS OR ONE WAY ANALYSIS OF VARIANCE.');
   WRITELN('3. INFORMATION ABOUT THIS PROGRAM.');
   WRITELN('4. BRIEF EXPLANATIONS OF STATISTICS.');
   WRITELN('5. STOP.');
   READINTEGER(1,5,FALSE,DUMMYSTR,PROBLEM);
END;
```

```
PROCEDURE GETINFORMATION(FIRSTQUES : INTEGER;
                         VAR GROUP,POINT  : INTEGER;
                         VAR DATAKIND : DATATYPE;
                         VAR PAIRED,RESUME : BOOLEAN);



VAR NOQUESASK : 0..10;
    QUESNO    : 1..11;
    DUMMYINT  : INTEGER;
    QUESASK   : ARRAY [1..10] OF 1..10;
    DUMMYSTR,R,S,U,V,W : STRING;
    KINDOFDATA : 1..4;


    PROCEDURE ANSWER(L,U              : INTEGER;
                     HELPFILENAME     : STRING;
                     VAR QUESNO,DATUM : INTEGER;
                     VAR INFORM       : STRING);

       BEGIN
          IF QUESNO<=3 THEN
             READINTEGER(L,U,TRUE,INFORM,DATUM)
          ELSE
             READSTR(TRUE,INFORM);
          IF HELP(INFORM) THEN BEGIN
             QUESNO:=QUESNO-1;
             DEFINITION;
             IF HELPFILENAME<>'NOHELP' THEN
                READFILE(HELPFILENAME);
          END ELSE IF REJECT(INFORM) THEN BEGIN
             IF QUESNO=FIRSTQUES THEN
                EXIT(GETINFORMATION);
             QUESNO:=QUESASK[NOQUESASK]-1;
             NOQUESASK:=NOQUESASK-1;
          END ELSE BEGIN
             NOQUESASK:=NOQUESASK+1;
             QUESASK[NOQUESASK]:=QUESNO;
          END;
          WRITELN;
       END (*ANSWER*);
```

```
PROCEDURE GETDATAKIND(VAR DATAKIND : DATATYPE);

   VAR KINDOFDATA : 1..4;

   BEGIN
      WRITELN('WHAT KIND OF DATA DO YOU HAVE ?');
      WRITELN('1. SCORES ASSIGNED TO CASES, BUT NOT BINARY DATA.');
      WRITELN('2. MEASUREMENT OR CONTINUOUS SCALE DATA.');
      WRITELN('3. COUNTS.');
      WRITELN('4. BINOMIAL PROPORTIONS.');
      ANSWER(1,4,'DATAKIND.TEXT',QUESNO,KINDOFDATA,DUMMYSTR);
      CASE KINDOFDATA OF
           1: DATAKIND:=SCORE;
           2: DATAKIND:=CONTINUOUS;
           3: DATAKIND:=COUNT;
           4: DATAKIND:=BINOMIAL;
      END;
   END (*GETDATAKIND*);



PROCEDURE CHECKRANDOMIZATION(VAR QUESNO : INTEGER; VAR S : STRING);


   BEGIN
      CASE QUESNO OF
           5: WRITELN('DO YOU ALLOCATE TREATMENTS TO CASES AT RANDOM');
           6: WRITELN('DO YOU ASSIGN CASES TO TREATMENTS AT RANDOM');
           7: BEGIN
                 WRITELN('DO YOU DIVIDE CASES RANDOMLY INTO GROUPS AND');
                 WRITE('THEN ALLOCATE TREATMENTS TO GROUPS');
              END;
      END;
      IF QUESNO<7 THEN
         WRITE('AND THEN FORM GROUPS FROM CASES WITH THE SAME TREATMENT');
      WRITE(' ?');
      ANSWER(0,0,'RANDOM.TEXT',QUESNO,DUMMYINT,S);
   END (*CHECKRANDOMIZATION*);



BEGIN
   S:='Y';
   R:='N';
   NOQUESASK:=0;
   QUESNO:=FIRSTQUES;
   WHILE QUESNO<=10 DO BEGIN
      CASE QUESNO OF
           1:     GETDATAKIND(DATAKIND);
           2:     BEGIN
                     WRITELN('HOW MANY GROUPS DO YOU HAVE ? ');
                     ANSWER(1,MAXGROUP,'NOHELP',QUESNO,GROUP,DUMMYSTR);
                  END;
           3:     BEGIN
                     WRITELN('HOW MANY DATA DO YOU COLLECT FROM EACH CASE ? ');
                     WRITELN('1. ONE.');
                     WRITELN('2. TWO.');
                     WRITELN('3. MORE THAN TWO OR UNEQUAL.');
                     ANSWER(1,3,'NOHELP',QUESNO,POINT,DUMMYSTR);
                  END;
```

```
        4:      IF (GROUP=2) AND (POINT=1) THEN BEGIN
                   WRITELN('ARE YOUR OBSERVATIONS PAIRED ?');
                   ANSWER(0,0,'PAIRED.TEXT',QUESNO,DUMMYINT,W);
                   PAIRED:=W='Y';
                END;
        5,6,7:  IF (NOT PAIRED) AND (GROUP>1) THEN BEGIN
                   S:='N';
                   CHECKRANDOMIZATION(QUESNO,S);
                   IF S='Y' THEN
                      QUESNO:=7;
                END ELSE
                   S:='Y';
        8:      BEGIN
                   WRITELN('IS IT POSSIBLE THAT THERE ARE FACTORS OTHER',
                           ' THAN');
                   WRITELN('THE ONE WHICH YOU WISH TO INVESTIGATE WHICH MAY'
                           );
                   WRITELN('LEAD TO A DIFFERENCE IN YOUR DATA ?');
                   ANSWER(0,0,'BIASED.TEXT',QUESNO,DUMMYINT,R);
                END;
        9:      BEGIN
                   IF (GROUP>1) AND (NOT PAIRED) THEN
                      WRITE('BETWEEN GROUPS OR WITHIN A GROUP, ');
                   WRITELN('IS THERE ANY CONNECTION BETWEEN');
                   IF PAIRED THEN
                      WRITE('PAIRS ?')
                   ELSE
                      WRITE('CASES ?');
                   ANSWER(0,0,'CONNECT.TEXT',QUESNO,DUMMYINT,U);
                END;
        10:     BEGIN
                   WRITELN('IS THE ORDER IN WHICH YOU COLLECT');
                   WRITE('YOUR DATA IMPORTANT ?');
                   ANSWER(0,0,'ORDER.TEXT',QUESNO,DUMMYINT,V);
                END;
     END;
     QUESNO:=QUESNO+1;
  END;
  RESUME:=(POINT<=2)
          AND (S='Y')
          AND (R='N')
          AND (U='N')
          AND (V='N');
  IF NOT RESUME THEN
     CONSULTSTATISTICIAN;
END (*GETINFORMATION*);
```

```
PROCEDURE PROCESSINFORMATION(VAR TESTSTATISTIC : TEST;
                             VAR DATAKIND      : DATATYPE;
                             VAR TRANSFORM     : TYPEOFTRANSFORMATION;
                             VAR GROUP,POINT,PROBLEM           : INTEGER;
                             VAR TESTTHEOMEAN,GETTEST,GETPVALUE : BOOLEAN;
                             VAR PAIRED,TAKEDIFFERENCE         : BOOLEAN);

BEGIN
   WITH TESTSTATISTIC DO BEGIN
      RANDOMTEST:=FALSE;
      RANKSUMTEST:=FALSE;
      IF PROBLEM=1 THEN BEGIN
         TESTTHEOMEAN:=FALSE;
         TAKEDIFFERENCE:=FALSE;
         SIGLEV:=0.05;
         NAME:='T-STATISTIC';
         GETTEST:=FALSE;
         GETPVALUE:=FALSE;
      END ELSE IF PROBLEM=2 THEN BEGIN
         TESTTHEOMEAN:=(GROUP=1) AND (POINT=1);
         GETTEST:=TRUE;
         GETPVALUE:=TRUE;
         IF PAIRED THEN
            GROUP:=1;          (*RESET GROUP TO 1*)
         TAKEDIFFERENCE:=PAIRED OR (POINT=2);
         IF DATAKIND=BINOMIAL THEN
            TRANSFORM:=ARCSINE;
      END;
   END;
END;



PROCEDURE CONSULTSTATISTICIAN;

(*DECLARED FORWARD*)


BEGIN
   WRITELN;
   WRITELN('CARE ABOUT YOUR DATA IS NECESSARY.');
   WRITELN('PLEASE CONSULT YOUR STATISTICIAN.');
   WRITELN('TYPE RETURN KEY TO CONTINUE.');
   READLN;
END (*CONSULTSTATISTICIAN*);


PROCEDURE PAUSE;

  (*DECLARED FORWARD*)

VAR I : INTEGER;

BEGIN
   WRITELN;
   WRITELN('TYPE CTRL AND S SIMULTANEOUSLY TO STOP OUTPUT ON THE SCREEN.');
   WRITELN('HIT ANY KEY TO CONTINUE.');
   FOR I:=1 TO 12000 DO (*NOTHING, JUST PAUSE A WHILE*)
END (*PAUSE*);
```

```
PROCEDURE DEFINITION;

BEGIN
   WRITELN('DEFINITIONS :-');
   WRITELN('1: A ''CASE'' IS ONE SINGLE EXPERIMENTAL SUBJECT, E.G. A PATIENT.'
           );
   WRITELN('2: A ''GROUP'' IS A COLLECTION OF CASES, E.G. 10 PATIENTS.');
   WRITELN;
   WRITELN('COMMANDS : TYPE');
   WRITELN(' ''HELP''  FOR HELP.');
   WRITELN(' ''QUIT''  TO STOP.');
   WRITELN(' ''REJ''   FOR IMMEDIATE REJECTION OR BACKWARD ELIMINATION.');
   WRITELN;
   WRITELN('NOTATIONS :');
   WRITELN(' ''Y'' STANDS FOR YES.');
   WRITELN(' ''N'' STANDS FOR NO.');
   WRITELN;
END (*DEFINITION*);



BEGIN                    (*PROCEDURE, QUESTION*)
   CLEAR;
   REPEAT
      CHOOSEPROBLEM(PROBLEM);
      PAGE(OUTPUT);
      IF PROBLEM<3 THEN BEGIN
         WRITELN('PLEASE ANSWER CAREFULLY.');
         WRITELN;
         DEFINITION;
         WRITELN('QUESTIONS BEGIN:-');
         WRITELN;
      END;
      CASE PROBLEM OF
           1: BEGIN
                 GROUP:=1;
                 POINT:=1;
                 GETINFORMATION(9,GROUP,POINT,DATAKIND,PAIRED,RESUME);
              END;
           2: GETINFORMATION(1,GROUP,POINT,DATAKIND,PAIRED,RESUME);
           3: BEGIN
                 PAUSE;
                 READFILE('INFORM.TEXT');
                 PAUSE;
              END;
           4: BEGIN
                 PAUSE;
                 READFILE('EXPLAIN.TEXT');
                 PAUSE;
              END;
           5: QUIT('QUIT');
      END;
   UNTIL RESUME;
   PROCESSINFORMATION(TESTSTATISTIC,
                      DATAKIND,
                      TRANSFORM,
                      GROUP,POINT,PROBLEM,
                      TESTTHEOMEAN,GETTEST,GETPVALUE,
                      PAIRED,TAKEDIFFERENCE);
END                      (*QUESTION*);
```

```
SEGMENT PROCEDURE READDATA(VAR DATA : DATASET;
                           VAR GPSIZE : GROUPSIZE;
                           VAR GROUP,TOTAL,GPSIZEALLOW : INTEGER;
                           VAR DATAKIND : DATATYPE;
                           VAR THEOMEAN : REAL;
                           VAR PAIRED,TAKEDIFFERENCE : BOOLEAN;
                           VAR TESTTHEOMEAN,RESUME   : BOOLEAN);


TYPE TEMPDATA=ARRAY [1..2,1..LIMIT] OF REAL;
     PROMPTDATA=ARRAY [1..2] OF STRING;


PROCEDURE VALIDATEDATA(VAR Y : TEMPDATA;
                       VAR PROMPT : PROMPTDATA;
                           DATAFORMAT,SUBJECT : STRING;
                           LOWERBOUND,UPPERBOUND : REAL;
                           NOOFSUBJECT,I,L    : INTEGER;
                       VAR ADDITION : BOOLEAN);

   (*FOR VALIDATING DATA*)

   FORWARD;

PROCEDURE KEYINDATA(VAR Y : TEMPDATA;
                    VAR PROMPT : PROMPTDATA;
                        DATAFORMAT : STRING;
                        LOWERBOUND,UPPERBOUND : REAL;
                    VAR I,J,L : INTEGER;
                    VAR SUBJECT,MESSAGE : STRING;
                        VERIFY    : BOOLEAN);

   (*FOR INPUTTING DATA*)

   FORWARD;

PROCEDURE CALCULDATAGPSIZE(VAR Y : TEMPDATA;
                               NOOFSUBJECT,I : INTEGER;
                               TAKEDIFFERENCE : BOOLEAN;
                           VAR DATA   : DATASET;
                           VAR GPSIZE : GROUPSIZE;
                               THEOMEAN : REAL);

   (*FOR TAKING DIFFERENCES OF OBSERVATIONS IF TWO OBSERVATIONS FROM   *)
   (*EACH CASE OR ASSIGNING OBSERVATIONS TO ARRAY DATA AND SAMPLE SIZES*)

   FORWARD;


PROCEDURE INSTRUCTION;

   (*HELP INSTRUCTION*)

   FORWARD;
```

```
PROCEDURE DATAPLEASE;

   (*INPUT INSTRUCTION*)

   FORWARD;

PROCEDURE NOTICE;

   (*GIVE NOTICE TO USERS*)

   FORWARD;

PROCEDURE EXPLAINMEAN(DATAKIND : DATATYPE);

   (*EXPLANATION FOR USERS *)

   FORWARD;

PROCEDURE CHECKGROUPLEFT(    NOOFGROUPLEFT : INTEGER;
                         VAR RESUME  : BOOLEAN);

   (*CHECK THE NUMBER OF GROUPS LEFT*)

   FORWARD;



PROCEDURE READINDATA(    DATAKIND : DATATYPE;
                         PAIRED,TAKEDIFFERENCE,TESTTHEOMEAN : BOOLEAN;
                     VAR GROUP,TOTAL,GPSIZEALLOW : INTEGER;
                     VAR DATA    : DATASET;
                     VAR GPSIZE : GROUPSIZE;
                     VAR THEOMEAN : REAL;
                     VAR RESUME : BOOLEAN);

   (*MAIN PROCEDURE FOR INPUTTING DATA*)


VAR PROMPT : PROMPTDATA;
    Y : TEMPDATA;
    LOWERBOUND,UPPERBOUND : REAL;
    GPNO,GPACCEPT,GPREJECT,J,K,L,NOOFSUBJECT : INTEGER;
    MESSAGE,SUBJECT,DATAFORMAT : STRING;
    ADDITION,NOTALLEQUAL : BOOLEAN;
```

```
    PROCEDURE SETUPPROMPT;

    (*SETTING UP PROMPT FOR DATA*)

    BEGIN
       SUBJECT:='CASE ';
       IF TAKEDIFFERENCE THEN BEGIN
          L:=2;
          IF PAIRED THEN BEGIN
             SUBJECT:='PAIR ';
             PROMPT[1]:='FIRST  GROUP ';
             PROMPT[2]:='SECOND GROUP ';
          END ELSE BEGIN
             PROMPT[1]:='FIRST  OBSERVATION ';
             PROMPT[2]:='SECOND OBSERVATION ';
          END;
       END ELSE BEGIN
          L:=1;
          PROMPT[1]:='OBSERVATION '
       END;
       CASE DATAKIND OF
            SCORE,CONTINUOUS : LOWERBOUND:=-1.0E30;
            COUNT,BINOMIAL   : LOWERBOUND:=0;
       END;
       IF DATAKIND=BINOMIAL THEN BEGIN
          UPPERBOUND:=100;
          NOTICE;
       END ELSE
          UPPERBOUND:=1.0E30;
       IF DATAKIND=COUNT THEN
          DATAFORMAT:='INTEGER'
       ELSE
          DATAFORMAT:='NUMERIC';
    END (*SETUPPROMPT*);



BEGIN
   SETUPPROMPT;
   GPSIZEALLOW:=LIMIT DIV GROUP;
   WRITELN('NOTE: 1. NUMBER OF GROUPS = ',GROUP);
   WRITELN('      2. MAXIMUM NUMBER OF DATA PER GROUP ALLOWED = ',
            GPSIZEALLOW,'.');

   DATAPLEASE;
   IF TESTTHEOMEAN THEN
      REPEAT
         GETDATA('ENTER YOUR THEORETICAL MEAN ','NUMERIC',LOWERBOUND,
                 UPPERBOUND,MESSAGE,THEOMEAN);
         IF HELP(MESSAGE) THEN BEGIN
            INSTRUCTION;
            EXPLAINMEAN(DATAKIND)
         END ELSE IF REJECT(MESSAGE) OR ENDING(MESSAGE) THEN
            WRITELN('YOU CANNOT USE  ',MESSAGE,' HERE.');
      UNTIL NOT HELP(MESSAGE);
```

```
   GPNO:=0;
   GPACCEPT:=1;
   GPREJECT:=0;
   TOTAL:=0;
   REPEAT
      WRITELN;
      GPNO:=GPNO+1;
      NOOFSUBJECT:=0;
      REPEAT
         REPEAT
            NOOFSUBJECT:=NOOFSUBJECT+1;
            KEYINDATA(Y,PROMPT,DATAFORMAT,LOWERBOUND,UPPERBOUND,
                      GPNO,NOOFSUBJECT,L,SUBJECT,MESSAGE,FALSE);
         UNTIL ENDING(MESSAGE);
         NOOFSUBJECT:=NOOFSUBJECT-1;
         ADDITION:=FALSE;
         VALIDATEDATA(Y,PROMPT,DATAFORMAT,SUBJECT,LOWERBOUND,UPPERBOUND,
                      NOOFSUBJECT,GPNO,L,ADDITION);
      UNTIL NOT ADDITION;
      CALCULDATAGPSIZE(Y,NOOFSUBJECT,GPACCEPT,TAKEDIFFERENCE,
                       DATA,GPSIZE,THEOMEAN);
      IF GPSIZE[GPACCEPT]<3 THEN BEGIN
         GPREJECT:=GPREJECT+1;
         WRITELN('GROUP SIZE OF THIS GROUP IS TOO SMALL.');
         CHECKGROUPLEFT(GROUP-GPREJECT,RESUME);
      END ELSE BEGIN
         K:=T(GPACCEPT,1);
         J:=K;
         REPEAT
            J:=J+1;
            NOTALLEQUAL:=DATA[J]<>DATA[K];
         UNTIL (J=GPSIZE[GPACCEPT]) OR (NOTALLEQUAL);
         IF NOTALLEQUAL THEN BEGIN
            TOTAL:=TOTAL+GPSIZE[GPACCEPT];
            GPACCEPT:=GPACCEPT+1;
         END ELSE BEGIN
            GPREJECT:=GPREJECT+1;
            WRITELN;
            IF TAKEDIFFERENCE THEN
               WRITE('DIFFERENCES OF ',PROMPT[1],' AND ',PROMPT[2])
            ELSE
               WRITE(PROMPT[1],'S');
            WRITELN('ARE ALL EQUAL.');
            IF GROUP=2 THEN
               WRITELN('THERE IS NO POINT IN DOING THE COMPARISON.');
            CHECKGROUPLEFT(GROUP-GPREJECT,RESUME);
         END;
      END;
   UNTIL GPNO=GROUP;
   GROUP:=GPACCEPT-1;
   IF (GROUP=1) AND (GPSIZE[1]<80) THEN
      (*RE-DEFINE GPSIZEALLOW IN CASE SIGNED-RANK WILCOXON TEST*)
      (*IS USED, THIS IS FOR THE USE OF PASSING DATA TO THE    *)
      (*PROCEDURE FOR CALCULATING RANK SUMS, PRODUCES NO       *)
      (*SIDE EFFECT.                                          *)
      GPSIZEALLOW:=LIMIT DIV 2;
   IF GPREJECT>0 THEN
      WRITELN('NUMBER OF GROUPS ELIMINATED FROM THE ANALYSIS IS ',
              GPREJECT,'.');
END (*READINDATA*);
```

```
PROCEDURE VALIDATEDATA;


(*DECLARED FORWARD*)
(*PARA:( VAR Y : TEMPDATA; VAR PROMPT : PROMPTDATA*)
(*       DATAFORMAT,SUBJECT : STRING                *)
(*       LOWERBOUND,UPPERBOUND : REAL               *)
(*       NOOFSUBJECT,I,L : INTEGER                  *)
(*       VAR ADDITION : BOOLEAN                     *)

VAR DUMMY,CHANGE : STRING;
    NOCHANGE : BOOLEAN;
    Q        : INTEGER;



    PROCEDURE VERIFYDATA(VAR Y      : TEMPDATA;
                         VAR PROMPT : PROMPTDATA;
                             DATAFORMAT,SUBJECT : STRING;
                             LOWERBOUND,UPPERBOUND : REAL;
                             Q,NOOFSUBJECT,I,L  : INTEGER;
                         VAR ADDITION : BOOLEAN);


       VAR J,K,R : INTEGER;
           MESSAGE     : STRING;


       PROCEDURE CHECKCOMMAND(MESSAGE : STRING);

          BEGIN
             IF REJECT(MESSAGE) THEN
                EXIT(VERIFYDATA)
             ELSE IF HELP(MESSAGE) THEN
                INSTRUCTION;
          END;



       PROCEDURE MAKECHANGE(VAR Q,J,NOOFSUBJECT : INTEGER;
                                SUBJECT : STRING;
                                PROMPT  : PROMPTDATA);

          VAR MESSAGE : STRING;

          BEGIN
             REPEAT
                WRITE('WHICH ',SUBJECT);
                CASE Q OF
                     2: WRITELN('TO BE CORRECTED ? ');
                     3: WRITELN('TO BE DELETED ? ');
                END;
                READINTEGER(1,NOOFSUBJECT,TRUE,MESSAGE,J);
                CHECKCOMMAND(MESSAGE);
             UNTIL NOT HELP(MESSAGE);
             WRITELN;
```

```
         CASE Q OF
              2: WRITE('CORRECTION ');
              3: WRITE('DELETION ');
         END;
         WRITELN(SUBJECT,J);
         IF Y[1,J]>1.0E36 THEN BEGIN
            WRITELN('DELETED ALREADY !');
         END ELSE BEGIN
            WRITELN('DATA ENTERED.');
            FOR R:=1 TO L DO
               WRITELN(PROMPT[R],Y[R,J]:14:3);
         END;
      END (*MAKECHANGE*);



BEGIN
   CASE Q OF
        1:  BEGIN  (*DISPLAY DATA ON THE SCREEN*)
               WRITE(PROMPT[1]:29);
               IF L=2 THEN
                  WRITE(PROMPT[2]:18);
               WRITELN;
               FOR J:=1 TO NOOFSUBJECT DO BEGIN
                  WRITE(SUBJECT,J:3);
                  FOR R:=1 TO L DO
                     IF Y[R,J]>1.0E36 THEN
                        WRITE('DELETED':16)
                     ELSE
                        WRITE(Y[R,J]:15:3);
                  WRITELN;
               END;
            END;
        2:  REPEAT  (*MAKING CORRECTION*)
               MAKECHANGE(Q,J,NOOFSUBJECT,SUBJECT,PROMPT);
               DATAPLEASE;
               KEYINDATA(Y,PROMPT,DATAFORMAT,LOWERBOUND,UPPERBOUND,
                         I,J,L,SUBJECT,MESSAGE,TRUE);
               CHECKCOMMAND(MESSAGE);
               WRITE('ANY MORE CORRECTIONS ?');
               READSTR(TRUE,MESSAGE);
               CHECKCOMMAND(MESSAGE);
            UNTIL MESSAGE='N';
        3:  REPEAT  (*MAKING DELETION*)
               MAKECHANGE(Q,J,NOOFSUBJECT,SUBJECT,PROMPT);
               WRITE('ANY MORE DELETIONS ?');
               READSTR(TRUE,MESSAGE);
               CHECKCOMMAND(MESSAGE);
               IF NOT REJECT(MESSAGE) THEN (*DO DELETION*)
                  FOR R:=1 TO L DO
                     Y[R,J]:=1.0E37;       (*SET TO ILLEGAL INPUT*)
            UNTIL MESSAGE='N';
        4:  BEGIN  (*MAKING ADDITION*)
               ADDITION:=TRUE;
               WRITELN;
               WRITELN('DATA ENTRY CONTINUES.');
               EXIT(VALIDATEDATA);
            END;
   END;
END (*VERIFYDATA*);
```

```
BEGIN
   WRITELN;
   IF GROUP>1 THEN
      WRITELN('THIS IS GROUP ',I);
   WRITELN;
   NOCHANGE:=FALSE;
   WRITELN('DO YOU WANT TO DISPLAY, CORRECT, DELETE OR ADD ANY DATA ?');
   READSTR(FALSE,CHANGE);
   REPEAT
      WHILE CHANGE='Y' DO BEGIN
         WRITELN;
         WRITELN('NOTE: YOU CAN DO ANY ONE OF THE FOLLOWING FIRST.');
         WRITELN;
         WRITELN('WHAT DO YOU WANT ?');
         WRITELN('1. DISPLAY DATA.');
         WRITELN('2. ERROR CORRECTION.');
         WRITELN('3. DELETION.');
         WRITELN('4. ADDITION.');
         READINTEGER(1,4,FALSE,DUMMY,Q);
         REPEAT
            VERIFYDATA(Y,PROMPT,DATAFORMAT,SUBJECT,LOWERBOUND,UPPERBOUND,
                       Q,NOOFSUBJECT,I,L,ADDITION);
            WRITELN;
            WRITELN('ANY MORE DISPLAY, CORRECTION, DELETION OR ADDITION ?');
            READSTR(FALSE,CHANGE);
         UNTIL (CHANGE='N') OR (Q=1);
      END;
      IF CHANGE='N' THEN BEGIN
         WRITELN('WARNING : LAST CHANCE FOR YOU TO MAKE CHANGES FOR THIS',
                 ' GROUP.');
         WRITELN;
         WRITE('DO YOU WANT TO MAKE ANY MORE CHANGES ?');
         READSTR(FALSE,CHANGE);
         NOCHANGE:=CHANGE='N';
      END;
   UNTIL NOCHANGE;
END (*VALIDATEDATA*);
```

```
PROCEDURE KEYINDATA;

(*DECLARED FORWARD*)
(*PARA: ( VAR Y : TEMPDATA; VAR PROMPT : PROMPTDATA*)
(*         DATAFORMAT : STRING                     *)
(*         LOWERBOUND,UPPERBOUND : REAL            *)
(*         VAR I,J,L : INTEGER                     *)
(*         VAR SUBJECT,MESSAGE : STRING            *)
(*             VERIFY : BOOLEAN                    *)

VAR K : INTEGER;
    DATUM : REAL;

BEGIN
   WRITELN;
   IF GROUP>1 THEN
      WRITE('GROUP ',I:2,'    ');
   WRITELN('THIS IS ',SUBJECT,J);
   K:=1;
   WHILE K<=L DO BEGIN
      GETDATA(PROMPT[K],DATAFORMAT,LOWERBOUND,UPPERBOUND,MESSAGE,DATUM);
      IF HELP(MESSAGE) THEN
         INSTRUCTION
      ELSE IF ENDING(MESSAGE) THEN BEGIN
         WRITELN;
         IF VERIFY THEN
            WRITELN('YOU CANNOT USE END HERE.')
         ELSE IF J=1 THEN
            WRITELN('AT LEAST ONE MORE GROUP EXPECTED.')
         ELSE IF K=2 THEN
            WRITELN('ONE MORE DATA POINT EXPECTED !')
         ELSE
            EXIT(KEYINDATA);
         DATAPLEASE;
      END ELSE IF REJECT(MESSAGE) THEN BEGIN
         WRITELN;
         IF VERIFY THEN
            WRITELN('YOU CANNOT USE REJ HERE.')
         ELSE IF (K=1) AND (J=1) THEN
            WRITELN('NO DATA TO BE REJECTED.')
         ELSE BEGIN
            K:=L+1-K;
            IF K=L THEN
               J:=J-1;
            WRITELN(PROMPT[K],'OF ',SUBJECT,J,' REJECTED.');
            WRITE('RE-ENTER ');
         END;
      END ELSE BEGIN
         Y[K,J]:=DATUM;
         K:=K+1;
      END;
   END;
END (*KEYINDATA*);
```

```
PROCEDURE CALCULDATAGPSIZE;



(*DECLARED FORWARD*)
(*PARA: ( VAR Y : TEMPDATA; NOOFSUBJECT,I :INTEGER        *)
(*            TAKEDIFFERENCE : BOOLEAN; VAR DATA : DATASET*)
(*            VAR GPSIZE : GROUPSIZE; THEOMEAN : REAL     *)

VAR J,K:INTEGER;

BEGIN
   K:=T(I,0);
   IF TAKEDIFFERENCE THEN BEGIN
      FOR J:=1 TO NOOFSUBJECT DO IF Y[1,J]<1.0E37 THEN BEGIN
          K:=K+1;
          DATA[K]:=Y[1,J]-Y[2,J];
      END;
   END ELSE
      FOR J:=1 TO NOOFSUBJECT DO IF Y[1,J]<1.0E37 THEN BEGIN
          K:=K+1;
          DATA[K]:=Y[1,J]-THEOMEAN;(*THEOMEAN IS INITIALIZED TO 0*)
      END;
   GPSIZE[I]:=K-T(I,0);
END (*CALCULDATA*);




PROCEDURE NOTICE;

(*DECLARED FORWARD*)

BEGIN
   WRITELN;
   WRITELN('IMPORTANT :');
   WRITELN('PLEASE ENTER DATA AS PERCENTAGES BUT LEAVE % OUT.');
   WRITELN('EXAMPLE: ENTER 0.70 OR 70 % AS 70 .');
END (*NOTICE*);


PROCEDURE EXPLAINMEAN;

(*DECLARED FORWARD..PARA : ( DATAKIND : DATATYPE*)


BEGIN
   WRITELN('SINCE YOU HAVE ONE GROUP AND ONE DATA ITEM FROM EACH CASE, YOU ');
   WRITELN('ARE COMPARING YOUR EXPERIMENTAL MEAN WITH A THEORETICAL MEAN.');
   IF DATAKIND=SCORE THEN BEGIN
      WRITELN('WARNING: IT MAY BE MEANINGLESS TO COMPARE AN EXPERIMENTAL');
      WRITELN('MEAN OF SCORES ASSIGNED TO CASES WITH A THEORETICAL MEAN.');
   END;
   INSTRUCTION;
END (*EXPLAINMEAN*);
```

```
PROCEDURE CHECKGROUPLEFT;

(*DECLARED FORWARD..PARA: (NOOFGROUPLEFT : INTEGER; VAR RESUME : BOOLEAN*)


BEGIN
   IF NOOFGROUPLEFT>1 THEN BEGIN
      WRITELN;
      WRITELN('THIS GROUP WILL BE ELIMINATED FROM THE ANALYSIS.');
   END ELSE BEGIN
      WRITELN('TYPE RETURN KEY TO CONTINUE !');
      READLN;
      RESUME:=FALSE;
      EXIT(READDATA);
   END;
END;




PROCEDURE INSTRUCTION;

(*DECLARED FORWARD*)

BEGIN
   WRITELN;
   WRITELN('COMMANDS :   TYPE');
   WRITELN(' ''HELP''  FOR HELP.');
   WRITELN(' ''QUIT''  TO STOP.');
   WRITELN(' ''REJ'' FOR IMMEDIATE REJECTION OR BACKWARD ELIMINATION.');
   WRITELN('INPUT TERMINATOR : ''END'' FOR DATA ENTRY OF A GROUP.');
   WRITELN;
   WRITELN('NOTATIONS :');
   WRITELN(' ''Y'' STANDS FOR YES.');
   WRITELN(' ''N'' STANDS FOR NO.');
   WRITELN;
END (*INSTRUCTION*);



PROCEDURE DATAPLEASE;

(*DECLARED FORWARD*)


BEGIN
   WRITELN;
   WRITELN('PLEASE ENTER YOUR DATA.');
   WRITELN;
END;


BEGIN                    (*PROCEDURE, READDATA*)
   WRITELN;
   INSTRUCTION;
   READINDATA(DATAKIND,
              PAIRED,TAKEDIFFERENCE,TESTTHEOMEAN,
              GROUP,TOTAL,GPSIZEALLOW,
              DATA,GPSIZE,THEOMEAN,
              RESUME);
   WRITELN;
   WRITELN('PLEASE WAIT !');
   WRITELN;
   WRITELN('ANALYSIS IN PROGRESS.');
END                      (*READDATA*);
```

```
SEGMENT PROCEDURE BASICSTAT(VAR DATA,X : DATASET;
                           VAR DESCRIPTIVESTATISTIC : STATISTIC;
                           VAR TESTSTATISTIC : TEST;
                           VAR HISTOGRAM : GRAPH;
                           VAR GPSIZE : GROUPSIZE;
                               GROUP,TOTAL : INTEGER;
                           VAR OUTLIER,MINGPSIZE,MAXGPSIZE : INTEGER;
                           VAR BSS,WSS,MSE,KURTOSIS,MINDATA,ADDCONST : REAL;
                           VAR TRANSFORM : TYPEOFTRANSFORMATION;
                           VAR WANTTRANSFORM,PAIRED,TOOMANYEQ : BOOLEAN);

(*X IS DUPLICATE OF DATA BUT IN ASCENDING ORDER*)

VAR MAXDATA : REAL; (*MAXIMUM OF ALL DATA*)

PROCEDURE  DOTRANSFORMATION(    TRANSFORM : TYPEOFTRANSFORMATION;
                           VAR DATA  : DATASET;
                           VAR GPSIZE: GROUPSIZE;
                               GROUP : INTEGER;
                               ADDCONST : REAL);

(*   TRANSFORMING DATA    *)

CONST ONERADIAN=57.2957795 (*DEGREES*);

VAR I,J : INTEGER;
      P : REAL;
BEGIN
   CASE TRANSFORM OF
        SQUAREROOT  : FOR I:=1 TO GROUP DO
                         FOR J:=T(I,1) TO T(I,GPSIZE[I]) DO
                            DATA[J]:=SQRT(DATA[J]+ADDCONST);
        LOGARITHMIC : FOR I:=1 TO GROUP DO
                         FOR J:=T(I,1) TO T(I,GPSIZE[I]) DO
                            DATA[J]:=LOG(DATA[J]+ADDCONST);
        RECIPROCAL  : FOR I:=1 TO GROUP DO
                         FOR J:=T(I,1) TO T(I,GPSIZE[I]) DO
                            DATA[J]:=1/(DATA[J]+ADDCONST);
        ARCSINE     : FOR I:=1 TO GROUP DO
                         FOR J:=T(I,1) TO T(I,GPSIZE[I]) DO BEGIN
                            P:=DATA[J];
                            IF P<=0 THEN
                               P:=0.25/GPSIZE[I]
                            ELSE IF P>=99.9995 THEN
                               P:=100-0.25/GPSIZE[I];
                            (*ATAN(.)=ARCSINE(P) IN RADIAN*)
                            DATA[J]:=ONERADIAN*ATAN(SQRT(P/(100-P)));
                         END;
   END;
END (*DOTRANSFORMATION*);
```

```
PROCEDURE CALSTATISTIC(VAR DATA    : DATASET;
                       VAR GPSIZE : GROUPSIZE;
                       VAR GROUP,TOTAL          : INTEGER;
                       VAR MEAN,VARIANCE,G1,G2 : GROUPSTAT;
                       VAR BSS,WSS,MSE,CV,KURTOSIS : REAL;
                       VAR TRANSFORM : TYPEOFTRANSFORMATION);


VAR I,J,L,U,DF : INTEGER;

    SS,DEVPWRTHREE,DEVPWRFOUR,GM,TEMPMEAN,N : REAL;
    P1,P2,P3,P4,M2,SQM2,NSQM2,P,SQP,SUMP3,SUMP4 : REAL;


    FUNCTION G1STAT(N : INTEGER;
                    DEVPWRTHREE,SUMOFSQ : REAL) : REAL;

       (*CALCULATE FISHER'S G1-STATISTIC*)

       BEGIN
          G1STAT:=N*SQRT(N-1)*DEVPWRTHREE/((N-2)*SUMOFSQ*SQRT(SUMOFSQ));
       END (*G1STAT*);


    FUNCTION G2STAT(N : INTEGER;
                    DEVPWRFOUR,SUMOFSQ : REAL) : REAL;

       (*CALCULATE FISHER'S G2-STATISTIC*)

       BEGIN
          G2STAT:=(N-1.0)*(N*(N+1.0)*DEVPWRFOUR-3*(N-1.0)*SQR(SUMOFSQ))/
                  ((N-2.0)*(N-3.0)*SQR(SUMOFSQ));
       END (*G2STAT*);



    PROCEDURE GETTRANSFORMKURTOSIS(GM,SS3,SS4,WSS,MSE : REAL;
                                   GROUP,TOTAL,DF : INTEGER;
                                   VAR TRANSFORM      : TYPEOFTRANSFORMATION;
                                   VAR KURTOSIS : REAL);

       VAR D,P,R,PWR,SKEWNESS : REAL;

       BEGIN
            (*ESTIMATE POWER OF TRANSFORMATION, PWR     *)
            (*CALCULATE KURTOSIS & SKEWNESS OF RESIDUALS*)

          P:=TOTAL;
          R:=SQR(GROUP/(DF*(P-1.0)));
                   (*KURTOSIS OF ALL RESIDUALS*)
          KURTOSIS:=SQR(P)*P*((DF+2.0)*SS4/(DF*SQR(WSS))-3.0/P)/
                    (DF*(DF+2.0)*(1.0+(P-1.0)*SQR(R))-3.0*P);

                   (*SKEWNESS OF ALL RESIDUALS*)
          D:=2+KURTOSIS;
          IF (TRANSFORM=IDENTITY) AND (D<>0) THEN BEGIN
             SKEWNESS:=SS3*SQR(TOTAL)/
                       (DF*WSS*SQRT(DF*WSS)*(1+(P-1.0)*R*SQRT(R)));
             PWR:=1.0-2.0*SKEWNESS*GM/(3.0*D*SQRT(MSE));
             IF PWR<-0.60 THEN
                TRANSFORM:=RECIPROCAL
             ELSE IF PWR<0.25 THEN
                TRANSFORM:=LOGARITHMIC
             ELSE IF PWR<0.8 THEN
                TRANSFORM:=SQUAREROOT;
          END;
       END (*GETTRANSFORMKURTOSIS*);
```

```
BEGIN
   WSS:=0.0;
   SUMP3:=0.0;
   SUMP4:=0.0;
   FOR I:=1 TO GROUP DO BEGIN
      L:=T(I,1);
      U:=T(I,GPSIZE[I]);
      TEMPMEAN:=0.0;
      FOR J:=L TO U DO
         TEMPMEAN:=TEMPMEAN+DATA[J];
      TEMPMEAN:=TEMPMEAN/GPSIZE[I];  (*FIRST ESTIMATE OF MEAN*)

      (*CALCULATE MEAN, VARIANCE, WITHIN GROUPS SUM OF SQUARES,WSS*)
      (*G1 & G2 STATISTICS.                                      *)

      P1:=0.0;
      P2:=0.0;
      P3:=0.0;
      P4:=0.0;
      FOR J:=L TO U DO BEGIN
         P:=DATA[J]-TEMPMEAN;
         SQP:=SQR(P);
         P1:=P1+P;
         P2:=P2+SQP;
         P3:=P3+SQP*P;
         P4:=P4+SQR(SQP);
      END;
      N:=GPSIZE[I];
      M2:=P1/N;
      SQM2:=SQR(M2);

      MEAN[I]:=TEMPMEAN+M2;
      NSQM2:=N*SQM2;
      SS:=P2-NSQM2;
      WSS:=WSS+SS;
      VARIANCE[I]:=SS/(N-1);

      DEVPWRTHREE:=P3-M2*(3*P2-2*NSQM2);
      DEVPWRFOUR:=P4-4*M2*P3+SQM2*(6*P2-3*NSQM2);

      (*CALCULATE SUMS OF DEVIATIONS FROM MEAN TO THE POWER OF THREE,*)
      (*SUMP3 & FOUR SUMP4 OF ALL RESIDUALS.                         *)

      IF DEVPWRFOUR>(1.0E37-SUMP4) THEN (*GIVE USERS MESSAGE*)
         WRITELN('MESSAGE : OVERFLOW MAY OCCUR. DATA VARY GREATLY.');

      SUMP3:=SUMP3+DEVPWRTHREE;
      SUMP4:=SUMP4+DEVPWRFOUR;

      G1[I]:=G1STAT(GPSIZE[I],DEVPWRTHREE,SS);
      IF GPSIZE[I]>49 THEN
         G2[I]:=G2STAT(GPSIZE[I],DEVPWRFOUR,SS);
   END;
   DF:=TOTAL-GROUP;
   MSE:=WSS/DF;              (*MEAN SQUARE ERROR*)
```

```
   IF GROUP>1 THEN BEGIN
      GM:=0.0;
      FOR I:=1 TO GROUP DO
         GM:=GM+GPSIZE[I]*MEAN[I];
      GM:=GM/TOTAL;            (*FIRST ESTIMATE OF GRAND MEAN*)

      (*CALCULATE GRAND MEAN, BETWEEN GROUPS SUM OF SQUARES, BSS*)
      (*COEFFICIENT OF VARIATION OF VARIANCES, CV.              *)

      P1:=0.0;
      P2:=0.0;
      P3:=0.0;
      FOR I:=1 TO GROUP DO BEGIN
         P:=MEAN[I]-GM;
         P1:=P1+GPSIZE[I]*P;
         P2:=P2+GPSIZE[I]*SQR(P);
         P3:=P3+(GPSIZE[I]-1.0)*SQR(VARIANCE[I]-MSE);
      END;
      GM:=GM+P1/TOTAL;
      BSS:=P2-SQR(P1)/TOTAL;
      CV:=SQRT(P3/DF)/MSE;

      GETTRANSFORMKURTOSIS(GM,SUMP3,SUMP4,WSS,MSE,GROUP,TOTAL,DF,
                           TRANSFORM,KURTOSIS);
   END;
END (*CALSTATISTIC*);



PROCEDURE SORT(VAR Z      : DATASET;
               VAR GPSIZE : GROUPSIZE;
                   GROUP  : INTEGER);

(*SORT Z IN ASCENDING ORDER BY GROUPS*)
(*MAXIMUM NUMBER OF ITEMS OF Z IS (2 TO POWER BOUND+1)-1*)


CONST BOUND=8;

VAR ORDER : BOOLEAN;
    A,AA : REAL;
    I,P,Q,J,K,PP,PQ,L,S : INTEGER;
    IU,IL : ARRAY [0..BOUND] OF INTEGER;

BEGIN
   FOR I:=1 TO GROUP DO BEGIN
      P:=T(I,1);
      Q:=T(I,GPSIZE[I]);

      (*SORT FROM Z[P] TO Z[Q] IN ASCENDING ORDER *)

      S:=0;
      PP:=P;
      ORDER:=(Q-P)>10;
      REPEAT
         IF ORDER THEN
            REPEAT
               PQ:=ROUND((P+Q)/2);
               A:=Z[PQ];
               K:=P;
               L:=Q;
               IF Z[P]>A THEN BEGIN
                  Z[PQ]:=Z[P];
```

```
                Z[P]:=A;
                A:=Z[PQ];
             END;
             IF Z[Q]<A THEN BEGIN
                Z[PQ]:=Z[Q];
                Z[Q]:=A;
                A:=Z[PQ];
                IF Z[P]>A THEN BEGIN
                   Z[PQ]:=Z[P];
                   Z[P]:=A;
                   A:=Z[PQ];
                END;
             END;
             REPEAT
                REPEAT
                   L:=L-1;
                UNTIL Z[L]<=A;
                AA:=Z[L];
                REPEAT
                   K:=K+1;
                UNTIL Z[K]>=A;
                IF K<=L THEN BEGIN
                   Z[L]:=Z[K];
                   Z[K]:=AA;
                END;
             UNTIL K>L;
             IF L-P>Q-K THEN BEGIN
                IL[S]:=P;
                IU[S]:=L;
                P:=K;
             END ELSE BEGIN
                IL[S]:=K;
                IU[S]:=Q;
                Q:=L;
             END;
             S:=S+1;
          UNTIL Q-P<11;
       ORDER:=(P=PP) AND (P<Q);
       IF NOT ORDER THEN BEGIN
          FOR P:=P+1 TO Q DO BEGIN
             A:=Z[P];
             K:=P-1;
             IF Z[K]>A THEN BEGIN
                REPEAT
                   Z[K+1]:=Z[K];
                   K:=K-1;
                UNTIL Z[K]<=A;
                Z[K+1]:=A;
             END;
          END;
          S:=S-1;
          IF S>=0 THEN BEGIN
             P:=IL[S];
             Q:=IU[S];
             ORDER:=(Q-P)>10;
          END;
       END;
    UNTIL S<0;
 END;
END (*SORTING*);
```

```
PROCEDURE MEDIANEQDATA(VAR X       : DATASET;
                       VAR GPSIZE : GROUPSIZE;
                       VAR GROUP   : INTEGER;
                       VAR MEDIAN : GROUPSTAT;
                       VAR TOOMANYEQ : BOOLEAN);

(*CALCULATE MEDIANS AND CHECK WHETHER TOO MANY*)
(*DATA POINTS IN A GROUP ARE EQUAL            *)

VAR I,L : INTEGER;
    W : REAL;

BEGIN
   TOOMANYEQ:=FALSE;
   FOR I:=1 TO GROUP DO BEGIN
      L:=T(I,(GPSIZE[I] DIV 2));
      W:=X[L];
      TOOMANYEQ:=(GPSIZE[I]>4) AND ((X[T(I,1)]=W) OR (X[T(I,GPSIZE[I])]=W));
      IF ODD(GPSIZE[I]) THEN
         MEDIAN[I]:=X[L+1]
      ELSE
         MEDIAN[I]:=0.5*(W+X[L+1]);
   END;
END (*CALMEDIAN*);


PROCEDURE CALMINMAX(VAR X : DATASET;
                    VAR GPSIZE : GROUPSIZE;
                    VAR GROUP,MINGPSIZE,MAXGPSIZE : INTEGER;
                    VAR MINIMUM,MAXIMUM : GROUPSTAT;
                    VAR MINDATA,MAXDATA : REAL);

(*CALCULATE MINIMUM & MAXIMUM OF DATA FOR EACH GROUP AND MINIMUM *)
(*OF ALL DATA MINDATA, AND MINIMUM AND MAXIMUM OF GPSIZES        *)

VAR I : INTEGER;

BEGIN
   FOR I:=1 TO GROUP DO BEGIN
      MINIMUM[I]:=X[T(I,1)];
      MAXIMUM[I]:=X[T(I,GPSIZE[I])];
   END;
   MINDATA:=MINIMUM[1];
   MAXDATA:=MAXIMUM[1];
   MINGPSIZE:=GPSIZE[1];
   MAXGPSIZE:=GPSIZE[1];
   FOR I:=2 TO GROUP DO BEGIN
      IF GPSIZE[I]<MINGPSIZE THEN
         MINGPSIZE:=GPSIZE[I]
      ELSE IF GPSIZE[I]>MAXGPSIZE THEN
         MAXGPSIZE:=GPSIZE[I];
      IF MINIMUM[I]<MINDATA THEN
         MINDATA:=MINIMUM[I];
      IF MAXIMUM[I]>MAXDATA THEN
         MAXDATA:=MAXIMUM[I];
   END;
END (*CALMINMAX*);
```

```
PROCEDURE CHECKOUTLIEREQDATA(VAR X       : DATASET;
                             VAR GPSIZE : GROUPSIZE;
                             VAR GROUP,OUTLER : INTEGER;
                             VAR TOOMANYEQ    : BOOLEAN);

(*CHECK OUTLIERS AND WHETHER OR NOT TOO MANY*)
(*EQUAL VALUED DATA POINTS IN A GROUP       *)

VAR I,J,K,L,U:INTEGER;
    W:REAL;

BEGIN
   OUTLIER:=0;
   FOR I:=1 TO GROUP DO
      IF GPSIZE[I]>4 THEN BEGIN

         (*GET INDICES OF LOWER AND UPPER 25% QUARTILES OF CUT OFF POINTS*)

         J:=GPSIZE[I] DIV 4;
         L:=T(I,J);
         U:=T(I,(GPSIZE[I]-J));
         W:=1.5*(X[U]-X[L]);
         TOOMANYEQ:=TOOMANYEQ OR (W=0);
         FOR K:=T(I,1) TO L-1 DO
            IF W<(X[L]-X[K]) THEN
               OUTLIER:=OUTLIER+1;
         FOR K:=U+1 TO T(I,GPSIZE[I]) DO
            IF W<(X[K]-X[U]) THEN
                OUTLIER:=OUTLIER+1;
      END;
END (*CHECKOUTLIEREQDATA*);



PROCEDURE CONSTRUCTHISTOGRAM(VAR HISTOGRAM : GRAPH;
                             VAR X : DATASET;
                             VAR GPSIZE : GROUPSIZE;
                                 GROUP,MAXGPSIZE : INTEGER;
                                 MINDATA,MAXDATA : REAL);

VAR MARK : ARRAY [1..25] OF REAL;
    TIMES,Q,RANGE : REAL;
    I,J,K,MAXFREQUENCY : INTEGER;


    FUNCTION ROUNDREAL(Q : REAL) : REAL;

    (*ROUND Q TO AN 'INTEGER' IN REAL*)

       VAR R : REAL;

       BEGIN
          R:=0.0;
          WHILE Q>32766.0 DO BEGIN (*TO AVOID OVERFLOW*)
             R:=R+32766.0;
             Q:=Q-32766.0;
          END;
          ROUNDREAL:=R+ROUND(Q);
       END (*ROUNDREAL*);
```

```
PROCEDURE GETSTEPNOOFINTERVALDECPL(     RANGE : REAL;
                                    MAXGPSIZE : INTEGER;
                                    VAR STEP  : REAL;
                                    VAR NOOFINTERVAL,DECPL : INTEGER);

   VAR R,SCALEFACTOR,Q : REAL;
       N : INTEGER;

   BEGIN
      R:=RANGE;
      SCALEFACTOR:=1.0;
      WHILE R<1.0 DO BEGIN
         SCALEFACTOR:=SCALEFACTOR/10.0;
         R:=R*10.0;
      END;
      WHILE R>10.0 DO BEGIN
         SCALEFACTOR:=SCALEFACTOR*10.0;
         R:=R/10.0;
      END;

      N:=ROUND(R);
      IF (5<N)=(N<11) THEN
         STEP:=SCALEFACTOR
      ELSE BEGIN
         Q:=20.0*R;
         N:=ROUND(Q/ROUND(Q/N));
         IF ODD(N) AND (N>5) THEN
            N:=N+1;
         IF (N=14) OR (N=18) THEN
            N:=N+2;
         STEP:=N*SCALEFACTOR/20.0;

         IF STEP>3.0 THEN
            STEP:=ROUNDREAL(STEP);

      END;

      N:=ROUND(RANGE/STEP); (*FIRST ESTIMATE OF NO. OF INTERVAL*)

      IF (N<=10) AND (N<MAXGPSIZE) THEN    (*NO. OF INTERVALS TOO FEW*)
         STEP:=0.5*STEP
      ELSE IF (N>20) OR (N>MAXGPSIZE) THEN (*NO. OF INTERVALS TOO MANY*)
         STEP:=2.0*STEP;

      NOOFINTERVAL:=ROUND(RANGE/STEP);

      IF (STEP<1.0E-3) OR (STEP>1.0E3) THEN
         DECPL:=0
      ELSE IF STEP<=0.05 THEN
         DECPL:=5
      ELSE
         DECPL:=3;

   END (*GETSTEPNOOFINTERVALDECPL*);
```

```
BEGIN
   WITH HISTOGRAM DO BEGIN
      RANGE:=MAXDATA-MINDATA;
      GETSTEPNOOFINTERVALDECPL(RANGE,MAXGPSIZE,STEP,NOOFINTERVAL,DECPL);

      MAXMIDPOINT:=ROUNDREAL((MAXDATA+STEP)/STEP)*STEP;

      (*ADJUST NO. OF INTERVALS TO COVER MINDATA*)

      Q:=MAXMIDPOINT-(NOOFINTERVAL-1)*STEP;
      WHILE Q>=MINDATA DO BEGIN
         Q:=Q-STEP;
         NOOFINTERVAL:=NOOFINTERVAL+1;
      END;
      Q:=MAXMIDPOINT+0.5*STEP;
      FOR I:=1 TO NOOFINTERVAL DO
         MARK[I]:=Q-I*STEP;
      MAXFREQUENCY:=0;
      FOR I:=1 TO GROUP DO BEGIN
         FOR K:=1 TO NOOFINTERVAL DO
            FREQUENCY[I,K]:=0;
         K:=1;
         FOR J:=T(I,GPSIZE[I]) DOWNTO T(I,1) DO BEGIN
            WHILE X[J]<MARK[K] DO
               K:=K+1;
            FREQUENCY[I,K]:=FREQUENCY[I,K]+1;
         END;
         FOR J:=1 TO K DO
            IF FREQUENCY[I,J]>MAXFREQUENCY THEN
               MAXFREQUENCY:=FREQUENCY[I,J];
      END;

      IF (GROUP MOD 4 =0) THEN
         HEIGHT:=15
      ELSE IF GROUP<4 THEN
         HEIGHT:=60 DIV GROUP
      ELSE IF (GROUP MOD 3 =0) OR (GROUP=5) THEN
         HEIGHT:=20
      ELSE
         HEIGHT:=15;

      REPRESENTCASE:=(MAXFREQUNCY DIV HEIGHT)+1;
   END;
END (*CONSTRUCTHISTOGRAM*);



BEGIN              (*PROCEDURE, BASICSTAT*)
   WITH DESCRIPTIVESTATISTIC DO BEGIN
      IF WANTTRANSFORM THEN
         DOTRANSFORMATION(TRANSFORM,DATA,GPSIZE,GROUP,ADDCONST);
      CALSTATISTIC(DATA,GPSIZE,GROUP,TOTAL,
                   MEAN,VARIANCE,G1,G2,BSS,WSS,MSE,CV,KURTOSIS,TRANSFORM);
      X:=DATA;             (*DUPLICATE DATA IN X*)
      SORT(X,GPSIZE,GROUP);
      MEDIANEQDATA(X,GPSIZE,GROUP,MEDIAN,TOOMANYEQ);
      CALMINMAX(X,GPSIZE,GROUP,MINGPSIZE,MAXGPSIZE,MINIMUM,MAXIMUM,
                MINDATA,MAXDATA);
      CHECKOUTLIEREQDATA(X,GPSIZE,GROUP,OUTLIER,TOOMANYEQ);
      CONSTRUCTHISTOGRAM(HISTOGRAM,X,GPSIZE,GROUP,MAXGPSIZE,MINDATA,MAXDATA);
   END;
END                (*BASICSTAT*);
```

```
SEGMENT PROCEDURE ASSUMPTIONDISTRIBUTION
                            (VAR EXAMINEDATA,GETPVALUE : BOOLEAN;
                             VAR X : DATASET;
                             VAR DESCRIPTIVESTATISTIC : STATISTIC;
                             VAR TESTSTATISTIC        : TEST;
                             VAR GPSIZE : GROUPSIZE;
                                 PROBLEM,GROUP,TOTAL : INTEGER;
                                 BSS,WSS,MSE,KURTOSIS : REAL;
                             VAR NORMAL,EQVARIANCE,SYMMETRY : BOOLEAN);

(* X IS DUPLICATE OF DATA *)



PROCEDURE SHAPIROWILKTEST(VAR X : DATASET;
                          VAR VARIANCE : GROUPSTAT;
                          VAR GPSIZE : GROUPSIZE;
                              GROUP : INTEGER;
                          VAR NORMAL,SYMMETRY : BOOLEAN);

   (*DOING SHAPIRO-WILK TEST FOR NORMALITY*)

   FORWARD;

PROCEDURE SKEWNESSTEST(VAR G1      : GROUPSTAT;
                       VAR GPSIZE : GROUPSIZE;
                           GROUP  : INTEGER;
                       VAR NORMAL,SYMMETRY : BOOLEAN);

   (*TESTING NORMALITY USING SAMPLE SKEWNESS*)

   FORWARD;

PROCEDURE TESTEQUALVARIANCE(VAR VARIANCE : GROUPSTAT;
                            VAR GPSIZE   : GROUPSIZE;
                                TOTAL,GROUP : INTEGER;
                                CV,KURTOSIS : REAL;
                            VAR EQVARIANCE  : BOOLEAN);

   (*TESTING EQUALITY OF VARIANCES*)

   FORWARD;


FUNCTION TDIST(T : REAL; DF :INTEGER) : REAL;

   (*CALCULATE ONE TAILED QUANTILE OF T-DISTRIBUTION*)
   (*WITH VALUE T AND DEGREES OF FREEDOM DF         *)

   FORWARD;

FUNCTION FDIST(F : REAL; V1,V2 : INTEGER) : REAL;

   (*CALCULATE UPPER TAILED QUANTILE OF F-DISTRIUTION*)
   (*WITH VALUE F AND DEGREES OF FREEDOM V1 AND V2   *)

   FORWARD;
```

```
FUNCTION CHISQ(XSQ : REAL; DF : INTEGER) : REAL;

   (*CALCULATE UPPER-TAILED QUANTILE OF CHI-SQUARE DISTRIBUTION*)
   (*WITH VALUE XSQ AND DEGREES OF FREEDOM DF                  *)

   FORWARD;

FUNCTION NORMALD(Y : REAL) : REAL;

   (*CALCULATE ONE-TAILED QUANTILE OF NORMAL DISTRIBUTION*)

   FORWARD;




PROCEDURE TESTASSUMPTION(VAR X      : DATASET;
                         VAR GPSIZE : GROUPSIZE;
                         VAR DESCRIPTIVESTATISTIC : STATISTIC;
                         VAR NORMAL,SYMMETRY,EQVARIANCE : BOOLEAN;
                         VAR PROBLEM,GROUP,TOTAL : INTEGER;
                         VAR KURTOSIS : REAL);

BEGIN
   WITH DESCRIPTIVESTATISTIC DO BEGIN
      NORMAL:=TRUE;
      SYMMETRY:=TRUE;
      IF PROBLEM=1 THEN
         SKEWNESSTEST(G1,GPSIZE,GROUP,NORMAL,SYMMETRY)
      ELSE BEGIN

         (*TESTING NORMALITY*)
         (*DECLARE NORMAL IF BOTH SHAPIROWILKTEST*)
         (*AND TESTSKEWNESS DECALRE NORMAL*)

         SHAPIROWILKTEST(X,VARIANCE,GPSIZE,GROUP,NORMAL,SYMMETRY);
         SKEWNESSTEST(G1,GPSIZE,GROUP,NORMAL,SYMMETRY);
      END;
      IF GROUP>1 THEN
         TESTEQUALVARIANCE(VARIANCE,GPSIZE,TOTAL,GROUP,CV,KURTOSIS,
                           EQVARIANCE)
      ELSE
         EQVARIANCE:=TRUE;
   END;
END (*TESTASSUMPTION*);




PROCEDURE PVALUEANDSIGLEV(VAR TESTSTATISTIC : TEST;
                              TOTAL         : INTEGER);


   (*GETTING P-VALUE AND SIGNIFICANCE LEVEL, SIGLEV*)
```

```
VAR P,Q,D,SQD : REAL;

    FUNCTION NORMALDENSITY(XSQ : REAL) : REAL;

       (*NORMAL DISTRIBUTION DENSITY*)

       CONST ONEDIVROOT2PI=0.39894228   (*=1/ROOT(2.PI)*);

       BEGIN
          NORMALDENSITY:=ONEDIVROOT2PI*EXP(-XSQ/2);
       END (*NORMALDENSITY*);



    PROCEDURE CHOOSELEVEL(PVAL,A,B,C,D,E : REAL;
                          VAR SIGLEV     : REAL);

       (*CHOOSING THE LEVEL AT WHICH MEAN DIFFERENCES ARE SIGNIFICANT*)

       BEGIN
          IF PVAL<A THEN
             SIGLEV:=0.01
          ELSE IF PVAL<B THEN
             SIGLEV:=0.05
          ELSE IF PVAL<C THEN
             SIGLEV:=0.1
          ELSE IF PVAL<D THEN
             SIGLEV:=0.15
          ELSE IF PVAL<E THEN
             SIGLEV:=0.2;
          IF (GROUP<3) AND ((SIGLEV>0.1) OR (SIGLEV=0)) THEN
             SIGLEV:=0.05;          (*FOR CONSTRUCTING 95% CON. INTERVAL*)
       END (*CHOOSELEVEL*);

BEGIN
   WITH TESTSTATISTIC DO BEGIN
      SIGLEV:=0;
      D:=VALUE;
      CASE DISTRIBUTION OF
           FDISTRIBUTION: BEGIN (*F-DISTRIBUTION*)
                             PVALUE:=FDIST(D,DFN,DFD);
                             CHOOSELEVEL(PVALUE,0.01,0.05,0.1,0.15,0.2,
                                         SIGLEV);
                          END;
           TDISTRIBUTION: BEGIN (*T-DISTRIBUTION*)
                             PVALUE:=TDIST(D,DFD);
                             CHOOSELEVEL(PVALUE,0.005,0.025,0.05,0.075,0.10,
                                         SIGLEV);
                          END;
           KRUSKALWALLIS: BEGIN (*USE F-DISTRIBUTION APPROXIMATION*)
                             PVALUE:=FDIST(DFD*D/(DFN*(TOTAL-1-D)),DFN,DFD);
                             CHOOSELEVEL(PVALUE,0.01,0.05,0.1,0.15,0.2,
                                         SIGLEV);
                          END;
```

```
            SIGNEDWILCOXON  :
                              IF D>13 THEN
                                 PVALUE:=0
                              ELSE BEGIN
                                 P:=NOOFNONZERO;
                                 SQD:=SQR(D);
                                 PVALUE:=NORMALD(D)+NORMALDENSITY(SQD)*
                                         (3*P*(P+1.0)-1.0)*D*(SQD-3)
                                         /(10*P*(P+1.0)*(2*P+1.0));
                              END;
            TWOWILCOXON      : (*APPROX. WILCOXON TEST*)
                              IF D>13 THEN (*PVALUE VERY SMALL*)
                                 PVALUE:=0
                              ELSE BEGIN
                                 P:=GPSIZE[1];
                                 Q:=GPSIZE[2];
                                 SQD:=SQR(D);
                                 PVALUE:=NORMALD(D)+NORMALDENSITY(SQD)*
                                         (SQR(P)+SQR(Q)+P*Q+TOTAL)*D*(SQD-3)/
                                         (20.0*P*Q*(TOTAL+1.0));
                              END;
            RANDOM         : PVALUE:=VALUE;
      END;
   END;
END (*PVALUEANDSIGLEV*);



PROCEDURE  SHAPIROWILKTEST;

(*DECLARED FORWARD*)
(*PARA: (VAR  X : DATASET; VAR VARIANCE : GROUPSTAT*)
(*       VAR GPSIZE : GROUPSIZE; GROUP : INTEGER   *)
(*       VAR NORMAL,SYMMETRY : BOOLEAN             *)


VAR I,J,K,L,P,Q,R,FIRST : INTEGER;
    WILK : TEXT;
    COEF : ARRAY [1..30] OF REAL;
    D,W,H : REAL;

BEGIN

   (*$I+*) (*TURN I/O CHECK ON*)

   P:=0;
   H:=0;
   I:=0;
   REPEAT
      I:=I+1;
      IF (GPSIZE[I]>2) AND (GPSIZE[I]<51) THEN BEGIN
         P:=P+1;
         IF GPSIZE[I]<31 THEN BEGIN
            FIRST:=3;
            RESET(WILK,'SHAPWILK.3TO30')
         END ELSE BEGIN
            FIRST:=31;
            RESET(WILK,'SHAPWILK.31TO50');
         END;
         FOR J:=FIRST TO GPSIZE[I] DO BEGIN
            K:=J DIV 2;
            FOR L:=1 TO K+3 DO
               READ(WILK,COEF[L]);
         END;
```

```
         Q:=T(I,GPSIZE[I])+1;
         R:=T(I,0);
         FOR J:=1 TO K DO
            W:=W+COEF[J]*(X[Q-J]-X[R+J]);
         W:=SQR(W)/(GPSIZE[I]*VARIANCE[I]);
         D:=W-GPSIZE[I]*SQR(COEF[1])/(GPSIZE[I]-1);
         IF D<=0 THEN
            NORMAL:=FALSE
         ELSE
            H:=H+COEF[K+2]+COEF[K+3]*LN(D/(1-W));(*STANDARDISED NORMAL*)
         CLOSE(WILK);
      END;
   UNTIL (I=GROUP) OR (NOT NORMAL);
   IF (P>1) AND NORMAL THEN BEGIN
      H:=H/SQRT(P);(*STANDARDISED NORMAL*)
      NORMAL:=(H>-1.645);
   END ELSE IF P=1 THEN
      NORMAL:=W>COEF[K+1];
END (*SHAPIROWILKTEST*);



PROCEDURE SKEWNESSTEST;

(*DECLARED FORWARD*)
(*PARA:( VAR  G1 : GROUPSTAT; VAR GPSIZE : GROUPSIZE          *)
(*           GROUP : INTEGER; VAR NORMAL, SYMMETRY : BOOLEAN *)

CONST ROOT1DIV6=0.40824829;

VAR I,P,Q : INTEGER;
    ONESIDE,TWOSIDE,N,Z,W,C,D : REAL;

BEGIN
   ONESIDE:=0;
   TWOSIDE:=0;
   P:=0;
   Q:=0;
   FOR I:=1 TO GROUP DO
      IF GPSIZE[I]>7 THEN BEGIN
        IF G1[I]>0 THEN
           Q:=Q+1;
        P:=P+1;
      END;
   I:=0;
   REPEAT
      I:=I+1;
      N:=GPSIZE[I];
      C:=ROOT1DIV6*G1[I]*SQRT((N-2)*(N+1)*(N+3)/(N*(N-1)));
      IF N>7 THEN BEGIN
         W:=SQRT(2*(3*(SQR(N)+27*N-70)*(N+1)*(N+3)/
                    ((N-2)*(N+5)*(N+7)*(N+9))-1))-1;
         C:=C/SQRT(2/(W-1));
         Z:=LN(C+SQRT(SQR(C)+1))/SQRT(0.5*LN(W));(*STANDARDIZED NORMAL*);
         SYMMETRY:=SYMMETRY AND (ABS(Z)<1.96);
```

```
         IF P>1 THEN BEGIN
            IF ABS(Z)>3 THEN
               NORMAL:=FALSE
            ELSE IF (Q=P) OR (Q=0) THEN
               ONESIDE:=ONESIDE-2*LN(NORMALD(Z))
            ELSE
               TWOSIDE:=TWOSIDE-2*LN(2*NORMALD(Z));
         END ELSE IF P=1 THEN
            NORMAL:=ABS(Z)<1.96;
      END ELSE
         SYMMETRY:=SYMMETRY AND (ABS(C)<1.96);
   UNTIL (I=GROUP) OR (NOT SYMMETRY);
   P:=2*P;
   IF NORMAL THEN
      IF ONESIDE>0THEN
         NORMAL:=CHISQ(ONESIDE,P)>=0.05
      ELSE
         NORMAL:=(CHISQ(TWOSIDE,P)>=0.025) AND (CHISQ(TWOSIDE,P)<=0.975);
END (*TESTSKEWNESS*);



PROCEDURE TESTEQUALVARIANCE;

(*DECLARED FORWARD*)
(*PARA: (VAR VARIANCE : GROUPSTAT; VAR GPSIZE : GROUPSIZE*)
(*           TOTAL,GROUP : INTEGER; CV,KURTOSIS : REAL   *)
(*           EQVARIANCE : BOOLEAN                        *)


VAR BARTEST,D : REAL;
    I,J,K : INTEGER;

BEGIN
   EQVARIANCE:=CV<1;(*IF CV>=1  THEN VERY UNEQUAL*)
   IF EQVARIANCE THEN
      IF GROUP=2 THEN BEGIN (*F-TEST*)
         IF VARIANCE[1]<VARIANCE[2] THEN
            J:=2
         ELSE
            J:=1;
         K:=3-J;
         EQVARIANCE:=(FDIST(VARIANCE[J]/VARIANCE[K],GPSIZE[J]-1,GPSIZE[K]-1)
                     >0.05)
      END ELSE BEGIN (*BARLETT'S TEST*)
         D:=0;
         J:=GROUP-1;
         K:=TOTAL-GROUP;
         FOR I:=1 TO GROUP DO
            D:=D+(GPSIZE[I]-1)*LN(VARIANCE[I]);
         BARTEST:=K*LN(MSE)-D;
         IF (NOT NORMAL) AND (KURTOSIS>-2) THEN
            D:=1+0.5*KURTOSIS          (*BOX'S ESTIMATE*)
         ELSE BEGIN
            D:=0;                      (*BARTLETT'S ESTIMATE*)
            FOR I:=1 TO GROUP DO
               D:=D+1/(GPSIZE[I]-1.0);
            D:=1+(D-1/K)/(3*J);
         END;
         EQVARIANCE:=(CHISQ(BARTEST/D,J)>0.05);
      END;
END (*TESTEQUALVARIANCE*);
```

```
FUNCTION TDIST;

(*DECLARED FORWARD..PARA:( T : REAL; DF : INTEGER*)


CONST ONEDIVPI=0.3183098862 (*=1/PI*);
      UPPERBOUND=200;
      LOWERBOUND=1.0E-15;

VAR SINESQ,COSINESQ,Z,X,A,D : REAL;
    K : INTEGER;

BEGIN
   T:=ABS(T);
   IF T>UPPERBOUND THEN
      TDIST:=0.0
   ELSE IF T<LOWERBOUND THEN
      TDIST:=0.5
   ELSE IF DF<21 THEN BEGIN
      X:=SQR(T);
      D:=X+DF;
      COSINESQ:=DF/D;
      SINESQ:=X/D;
      IF DF=1 THEN
         A:=0
      ELSE BEGIN

         (*EXACT SERIES SUMMATION USING RECURRENCE RELATION*)

         A:=1;
         K:=DF-2;
         WHILE K>1 DO BEGIN
            A:=COSINESQ*A*(K-1)/K+1;
            K:=K-2;
         END;
      END;
      IF ODD(DF) THEN
         TDIST:=0.5-ONEDIVPI*(ATAN(T/SQRT(DF))+SQRT(COSINESQ*SINESQ)*A)
      ELSE
         TDIST:=0.5-0.5*SQRT(SINESQ)*A;
   END ELSE BEGIN

      (*CORNISH-FISHER TYPE APPROXIMATION*)

      A:=DF-0.5;
      D:=48*SQR(A);
      Z:=A*LN(1+SQR(T)/DF);
      X:=SQRT(Z)*(1+(Z+3)/D+(Z*(Z*(4*Z+33)+240)+855)/
         (10*D*(D+0.08*SQR(Z)+100)));
      TDIST:=NORMALD(X);
   END;
END (*TDIST*);
```

```
FUNCTION FDIST;

(*DECLARED FORWARD..PARA: (F : REAL; V1,V2 :INTEGER*)


CONST TWODIVPI=0.6366197723675813430755351 (*=2/PI*);
      ONETHIRD=0.333333333333 (*=1/3*);
      TWOTHIRD=0.666666666667 (*=2/3*);
      TWODIVNINE=0.2222222222 (*=2/9*);
      UPPERBOUND=1.0E6;
      LOWERBOUND=1.0E-15;

VAR TEMPFDIST,COSINESQ,SINESQ,P,Q,A,B : REAL;
    I : INTEGER;
    USEREFLEXIVE : BOOLEAN;


    FUNCTION POWER(X,V : REAL) : REAL;

       (*COMPUTE X TO THE POWER OF V*)

       CONST LNBOUND=1.0E-35;
             EXPBOUND=-86;


       VAR Q : REAL;

       BEGIN
          (*EQUATE POWER(X,V) TO ZERO IF TOO SMALL, GUARD AGAINST UNDERFLOW*)

          IF X>LNBOUND THEN BEGIN
             Q:=V*LN(X);
             IF Q>EXPBOUND THEN
                POWER:=EXP(Q)
             ELSE
                POWER:=0;
          END ELSE
             POWER:=0.0;
       END (*POWER*);


    FUNCTION FINITESERIES(Y : REAL; I,J : INTEGER) : REAL;

       (*COMPUTE EXACT SERIES EXPANSION OF F-DISTRIBUTION*)

       VAR C : REAL;

       BEGIN
          I:=I-2;
          J:=J-2;
          C:=1;
          IF (I+J)>0 THEN

             (*RECURRENCE RELATION*)
             WHILE I>1 DO BEGIN
                C:=C*Y*(I+J)/I+1.0;
                I:=I-2;
             END;
          FINITESERIES:=C;
       END (*FINITESERIES*);
```

```
BEGIN
   IF (F>UPPERBOUND) AND (V2>1) THEN
      FDIST:=0
   ELSE IF F<LOWERBOUND THEN
      FDIST:=1
   ELSE BEGIN
      USEREFLEXIVE:=((V1>1) AND (V2=1))
                 OR ((ODD(V1) AND (V1<41)) AND ((NOT ODD(V2)) AND (V2<41)))
                 OR ((V1<15) AND (V2>40));
      IF USEREFLEXIVE THEN BEGIN
         I:=V1;
         V1:=V2;
         V2:=I;
         F:=1/F;
      END;
      IF (V1<41) AND (V2<41) THEN BEGIN
         P:=V1*F;
         Q:=P+V2;
         SINESQ:=P/Q;   (*SQUARE OF SINE(ATAN(SQRT(P/V2)))*)
         COSINESQ:=V2/Q;
         IF ODD(V1) AND ODD(V2) THEN BEGIN
            Q:=ATAN(SQRT(P/V2));
            IF V2=1 THEN
               A:=TWODIVPI*Q
            ELSE BEGIN
               A:=TWODIVPI*(Q+SQRT(COSINESQ*SINESQ)*
                  FINITESERIES(COSINESQ,V2,1));
               IF V1>1 THEN BEGIN

                  (*CALCULATE Q=((V2-1)/2)!*)
                  Q:=1;
                  FOR I:=((V2-1) DIV 2) DOWNTO 2 DO
                     Q:=Q*I;

                  (*CALCULATE P=2/(SQRT(PI)*((V2-2)/2)!)*)
                  (*NOTE: (-1/2)!=ROOT(PI)              *)
                  P:=1;
                  FOR I:=((V2-2) DIV 2) DOWNTO 0 DO
                     P:=P*(I+0.5);
                  P:=TWODIVPI/P;
                  B:=P*Q*SQRT(SINESQ)*POWER(COSINESQ,0.5*V2)*
                     FINITESERIES(SINESQ,V1,V2);
               END ELSE
                  B:=0;
            END;
            TEMPFDIST:=1-(A-B);
         END ELSE
            TEMPFDIST:=POWER(COSINESQ,0.5*V2)*FINITESERIES(SINESQ,V1,V2);
      END ELSE BEGIN
         P:=TWODIVNINE/V1;
         Q:=TWODIVNINE/V2;
         B:=POWER(F,ONETHIRD)*(1-Q)+P-1;
         B:=B/SQRT(P+Q*POWER(F,TWOTHIRD));
         IF B>0 THEN
            TEMPFDIST:=NORMALD(B)
         ELSE
            TEMPFDIST:=1-NORMALD(B);
      END;
      IF USEREFLEXIVE THEN
         FDIST:=1-TEMPFDIST
      ELSE
         FDIST:=TEMPFDIST;
   END;
END (*FDIST*);
```

```
FUNCTION CHISQ;

(*DECLARED FORWARD. PARA:( XSQ : REAL; DF : INTEGER*)


CONST ONETHIRD=0.333333333333 (*=1/3*);
      ROOTTWODIVPI=0.79788456 (*=SQUARE ROOT OF 2/PI*);
      LOWERBOUND=1.0E-16;
      UPPERBOUND=1.0E6;
      SERIESUPPERBOUND=105;


VAR I : INTEGER;
    R,RSQ,X,A,ASQ : REAL;

BEGIN
   IF XSQ<LOWERBOUND THEN
      CHISQ:=1.0
   ELSE IF XSQ>UPPERBOUND THEN
      CHISQ:=0
   ELSE IF DF<41 THEN BEGIN
      IF XSQ>SERIESUPPERBOUND THEN
         CHISQ:=0.0
      ELSE BEGIN

         (*EXACT SERIES SUMMATION USING RECURRENCE RELATION*)

         I:=DF-2;
         R:=0;
         WHILE I>0 DO BEGIN
            R:=(1+R)*XSQ/I;
            I:=I-2;
         END;
         IF ODD(DF) THEN
            CHISQ:=2*NORMALD(SQRT(XSQ))+ROOTTWODIVPI*EXP(-0.5*XSQ)*R/SQRT(XSQ)
         ELSE
            CHISQ:=EXP(-0.5*XSQ)*(1+R);
      END;
   END ELSE BEGIN

      RSQ:=XSQ-DF-DF*LN(XSQ/DF);
      R:=SQRT(RSQ);
      ASQ:=2.0/DF;
      A:=SQRT(ASQ);
      IF XSQ<DF THEN
         R:=-R;
      X:=R+A*(ONETHIRD-(RSQ-13.0)*ASQ/1620.0)
          -ASQ*R*(1/36-7*ASQ*(6.0*RSQ+17.0)/38880.0);
      IF X>0 THEN
         CHISQ:=NORMALD(X)
      ELSE
         CHISQ:=1-NORMALD(X);
   END;
END (*CHISQ*);
```

```
FUNCTION NORMALD;

(*DECLARED FORWARD..PARA:(  Y :REAL*)


CONST ONEDIVPI=0.3183098862;
      ONEDIV3ROOT2=0.23570226;
      ROOT2DIV3=0.47140452;
      UPPERBOUND=5.25;
      LOWERBOUND=1.0E-16;

VAR I : INTEGER;
    X,P : REAL;

BEGIN
   Y:=ABS(Y);
   IF Y<LOWERBOUND THEN
      NORMALD:=0.5
   ELSE IF Y<UPPERBOUND THEN BEGIN
      P:=ONEDIV3ROOT2*Y;
      X:=ROOT2DIV3*Y;
      FOR I:=12 DOWNTO 1 DO
         P:=P+EXP(-SQR(I)/9)*SIN(I*X)/I;
      NORMALD:=0.5-ONEDIVPI*P;
   END ELSE
      NORMALD:=0.0;
END (*NORMALD*);




BEGIN                       (* PROCEDURE, ASSUMPTIONDISTRIBUTION *)
   IF EXAMINEDATA THEN BEGIN
      TESTASSUMPTION(X,GPSIZE,DESCRIPTIVESTATISTIC,
                     NORMAL,SYMMETRY,EQVARIANCE,PROBLEM,GROUP,TOTAL,
                     KURTOSIS);
      EXAMINEDATA:=FALSE;
   END ELSE IF GETPVALUE THEN BEGIN
      PVALUEANDSIGLEV(TESTSTATISTIC,TOTAL);
      GETPVALUE:=FALSE;
   END;
END                         (*ASSUMPTIONDISTRIBUTION*);
```

```
SEGMENT PROCEDURE CALCULATION(VAR GETTEST : BOOLEAN;
                              VAR DATA,X  : DATASET;
                              VAR DESCRIPTIVESTATISTIC : STATISTIC;
                              VAR TESTSTATISTIC        : TEST;
                              VAR CONFIDENCEINTERVAL   : INTERVAL;
                                  TRANSFORM    : TYPEOFTRANSFORMATION;
                              VAR GPSIZE   : GROUPSIZE;
                              VAR DIFFPAIR : INTEGER;
                                  BSS,MSE  : REAL;
                                  GROUP,TOTAL,MINGPSIZE : INTEGER;
                                  MAXGPSIZE,GPSIZEALLOW : INTEGER;
                                  NORMAL,EQVARIANCE : BOOLEAN;
                              VAR SYMMETRY : BOOLEAN;
                                  TESTTHEOMEAN : BOOLEAN;
                                  DATAKIND : DATATYPE);


(* X IS USED FOR PASSING DATA TO FUNCTION RANDOMIZATIONTEST OR RANKTEST   *)
(* PROCEDURE, NOT NECESSARILY DUPLICATE OF THE DATA AND MAY CARRY RESULTS *)
(* ABOUT PAIRWISE COMPARISONS FOR MULTIPLE GROUPS AFTER SECOND CALL.      *)



FUNCTION RANDOMIZATIONTEST(VAR MEAN : GROUPSTAT;
                           VAR Z    : DATASET;
                               N,M  : INTEGER) : REAL;

   (*FOR CALCULATING ONE-SIDED P-VALUE OF RANDOMIZATION TEST*)

   FORWARD;


FUNCTION ONESAMPLETTEST(SAMPLEMEAN,SAMPLEVARIANCE : REAL;
                        SAMPLESIZE  : INTEGER) : REAL;

   (*FOR CALCULATING ONE SAMPLE T-TEST*)

   FORWARD;


FUNCTION TWOTTEST(VAR MEAN   : GROUPSTAT;
                  VAR GPSIZE : GROUPSIZE;
                      MSE    : REAL) : REAL;

   (*FOR CALCULATING TWO SAMPLE T-TEST*)

   FORWARD;


PROCEDURE WELCHTTEST(VAR MEAN,VARIANCE : GROUPSTAT;
                     VAR GPSIZE        : GROUPSIZE;
                     VAR DFD : INTEGER;
                     VAR WELCHT : REAL);

   (*FOR CALCULATING WELCH T-TEST*)

   FORWARD;
```

```
PROCEDURE WELCHFTEST(VAR MEAN,VARIANCE : GROUPSTAT;
                         GROUP,DFN : INTEGER;
                     VAR DFD : INTEGER;
                     VAR GPSIZE : GROUPSIZE;
                     VAR WELCHF : REAL);

   (*FOR CALCULATING WELCH F-TEST*)

   FORWARD;


PROCEDURE RANKTEST(VAR Z      : DATASET;
                   VAR GPSIZE : GROUPSIZE;
                   VAR RANKSUM : GROUPSTAT;
                       GROUP,TOTAL  : INTEGER;
                       NAME : STRING;
                   VAR VALUE,TIECORR : REAL);

   (*FOR CALCULATING RANK STATISTICS AND TIE CORRECTION FACTOR *)

   (*FOR SIGNED-RANK WILCOXON TEST, GROUP=2 AS POSITIVE NUMBERS*)
   (*ARE PASSED AS FIRST GROUP AND ABSOLUTE VALUES OF NEGATIVE *)
   (*NUMBERS AS THE SECOND GROUP. TOTAL IS THE NUMBER OF       *)
   (*NON-ZERO DATA                                            *)

   FORWARD;


FUNCTION INVERTDIST(P : REAL; N : INTEGER) : REAL;

   (*FOR CALCULATING THE UPPER PERCENTAGE POINT OF T-DISTRIBUTION*)

   FORWARD;

FUNCTION INVERNORMAL(P : REAL) : REAL;

   (*FOR CALCULATING THE UPPER PERCENTAGE POINT OF STD. NORMAL DISTRIBUTION*)

   FORWARD;



PROCEDURE TESTOFSINGLEMEAN(VAR DATA,X : DATASET;
                           VAR GPSIZE : GROUPSIZE;
                           VAR DESCRIPTIVESTATISTIC : STATISTIC;
                           VAR TESTSTATISTIC  : TEST;
                               GPSIZEALLOW    : INTEGER;
                           VAR SYMMETRY : BOOLEAN;
                               NORMAL,TESTTHEOMEAN : BOOLEAN;
                               TRANSFORM : TYPEOFTRANSFORMATION);

(*TEST OF SINGLE MEAN*)

VAR I : INTEGER;
    GP  : GROUPSIZE;
    TTEST,WILCOXTEST : BOOLEAN;
```

```
BEGIN
   TTEST:=(GPSIZE[1]>=80)
          OR NORMAL OR TESTTHEOMEAN
          OR ((GPSIZE[1]>15) AND SYMMETRY);
   WILCOXTEST:=(NOT TTEST)
               AND (TRANSFORM=IDENTITY);


   WITH DESCRIPTIVESTATISTIC, TESTSTATISTIC DO BEGIN
      (*IF T-TEST NOT APPLY, TRY RANDOMIZATION TEST*)
      RANDOMTEST:=(NOT TTEST)
                  AND (GPSIZE[1]<20)
                  AND (TRANSFORM=IDENTITY);
      IF RANDOMTEST THEN BEGIN
         NOOFNONZERO:=0;
         FOR I:=1 TO GPSIZE[1] DO
            IF DATA[I]<>0 THEN BEGIN
               NOOFNONZERO:=NOOFNONZERO+1;
               X[NOOFNONZERO]:=DATA[I];     (*X NOT DUPLICATE OF DATA*)
            END;
         RANDOMTEST:=NOOFNONZERO<16;
      END;
      IF RANDOMTEST THEN BEGIN
         DFN:=0;
         DFD:=0;
         RANDOMTEST:=TRUE;
         NAME:='PAIRED RANDOMIZATION';
         VALUE:=RANDOMIZATIONTEST(MEAN,X,NOOFNONZERO,NOOFNONZERO);
                                              (*THIRD PARAMETER IS REDUNDANT*)
         VALID:=TRUE;
         SYMMETRY:=TRUE; (*TRUE UNDER NULL HYPOTHESIS*)
         DISTRIBUTION:=RANDOM;
      END ELSE IF WILCOXTEST THEN BEGIN
         NAME:='PAIRED SIGNED-RANK WILCOXON';
         DFN:=0;
         DFD:=0;
         RANKSUMTEST:=TRUE;
         GP[1]:=0;
         GP[2]:=0;
         FOR I:=1 TO GPSIZE[1] DO   (* X BEING USED TO PASS DATA, PASSING *)
            IF DATA[I]>0 THEN BEGIN (* POSITIVE DATA AS FIRST GROUP       *)
               GP[1]:=GP[1]+1;      (* AND ABSOLUTE VALUES OF NEGATIVE    *)
               X[GP[1]]:=DATA[I];   (* DATA AS SECOND GROUP               *)
            END ELSE IF DATA[I]<0 THEN BEGIN
               GP[2]:=GP[2]+1;
               X[GP[2]+GPSIZEALLOW]:=-DATA[I];
            END;
         NOOFNONZERO:=GP[1]+GP[2];
         RANKTEST(X,GP,RANKSUM,2,NOOFNONZERO,NAME,VALUE,TIECORR);
         DISTRIBUTION:=SIGNEDWILCOXON;
         SYMMETRY:=TRUE; (*TRUE UNDER NULL HYPOTHESIS*)
         VALID:=TRUE;
      END ELSE BEGIN
         IF TESTTHEOMEAN THEN
            NAME:='ONE-SAMPLE T-TEST'
         ELSE
            NAME:='PAIRED T-TEST';
         DFN:=0;
         DFD:=GPSIZE[1]-1;
         VALUE:=ONESAMPLETTEST(MEAN[1],VARIANCE[1],GPSIZE[1]);
         DISTRIBUTION:=TDISTRIBUTION;
         VALID:=TTEST;
      END;
   END;
END (*TESTOFSINGLEMEAN*);
```

```
PROCEDURE TESTOFTWOMEANS(VAR X      : DATASET;
                         VAR GPSIZE : GROUPSIZE;
                         VAR DESCRIPTIVESTATISTIC : STATISTIC;
                         VAR TESTSTATISTIC : TEST;
                             GROUP,TOTAL,MINGPSIZE : INTEGER;
                             MSE : REAL;
                             NORMAL,SYMMETRY,EQVARIANCE : BOOLEAN;
                             TRANSFORM : TYPEOFTRANSFORMATION;
                             DATAKIND : DATATYPE);

(*TEST OF EQUALITY OF TWO MEANS*)


BEGIN
   WITH DESCRIPTIVESTATISTIC, TESTSTATISTIC DO BEGIN
      RANDOMTEST:=(NOT NORMAL)
                  AND SYMMETRY AND (TRANSFORM=IDENTITY)
                  AND ((GPSIZE[1]<10) AND (GPSIZE[2]<10));
      RANKSUMTEST:=(DATAKIND=SCORE)
                   AND (NOT RANDOMTEST)
                   AND (NOT NORMAL)
                   AND (SYMMETRY AND EQVARIANCE)
                   AND (TRANSFORM=IDENTITY);
      IF RANDOMTEST THEN BEGIN
         NAME:='TWO-SAMPLE RANDOMIZATION';
         DFN:=0;
         DFD:=0;
         VALUE:=RANDOMIZATIONTEST(MEAN,X,TOTAL,GPSIZE[1]);
         DISTRIBUTION:=RANDOM;
         VALID:=TRUE;
      END ELSE IF RANKSUMTEST THEN BEGIN
         NAME:='WILCOXON RANK SUM';
         DFN:=0;
         DFD:=0;
         RANKTEST(X,GPSIZE,RANKSUM,GROUP,TOTAL,NAME,VALUE,TIECORR);
         DISTRIBUTION:=TWOWILCOXON;
         VALID:=TRUE;
      END ELSE IF (GPSIZE[1]=GPSIZE[2])
                  OR EQVARIANCE
                  OR (MINGPSIZE<10) THEN BEGIN
         NAME:='TWO-SAMPLE T-TEST';
         DFN:=1;
         DFD:=TOTAL-2;
         DISTRIBUTION:=TDISTRIBUTION;
         VALUE:=TWOTTEST(MEAN,GPSIZE,MSE);
         VALID:=NORMAL OR SYMMETRY;
      END ELSE BEGIN
         NAME:='WELCH T-TEST';
         DFN:=1;
         (*DFD TO BE ESTIMATED*)

         WELCHTTEST(MEAN,VARIANCE,GPSIZE,DFD,VALUE);
         DISTRIBUTION:=TDISTRIBUTION;
         VALID:=NORMAL OR SYMMETRY;
      END;
   END;
END (*TESTOFTWOMEANS*);
```

```
PROCEDURE TESTOFSEVERALMEANS(VAR X      : DATASET;
                             VAR GPSIZE : GROUPSIZE;
                             VAR DESCRIPTIVESTATISTIC : STATISTIC;
                             VAR TESTSTATISTIC : TEST;
                                 GROUP,TOTAL,MINGPSIZE,MAXGPSIZE : INTEGER;
                                 MSE : REAL;
                                 NORMAL,SYMMETRY,EQVARIANCE : BOOLEAN;
                                 TRANSFORM : TYPEOFTRANSFORMATION;
                                 DATAKIND : DATATYPE);

(*TEST OF EQUALITY OF SEVERAL MEANS*)

VAR SIZENOTTOOSMALL : BOOLEAN;

BEGIN
   SIZENOTTOOSMALL:=(TOTAL DIV GROUP)>3;
   WITH DESCRIPTIVESTATISTIC, TESTSTATISTIC DO BEGIN
      RANKSUMTEST:=SIZENOTTOOSMALL
                   AND (DATAKIND=SCORE)
                   AND (NOT NORMAL)
                   AND (TRANSFORM=IDENTITY)
                   AND SYMMETRY AND EQVARIANCE;
      DFN:=GROUP-1;
      DFD:=TOTAL-GROUP;
      IF RANKSUMTEST THEN BEGIN
         NAME:='KRUSKAL-WALLIS';
         RANKTEST(X,GPSIZE,RANKSUM,GROUP,TOTAL,NAME,VALUE,TIECORR);
         VALID:=TRUE;
         DISTRIBUTION:=KRUSKALWALLIS;
      END ELSE IF EQVARIANCE
                  OR (MINGPSIZE<10)
                  OR ((MINGPSIZE=MAXGPSIZE) AND (CV<1)) THEN BEGIN
         NAME:='F-TEST';
         DISTRIBUTION:=FDISTRIBUTION;
         VALUE:=BSS/(MSE*DFN);
         VALID:=NORMAL OR SYMMETRY;
      END ELSE BEGIN
         NAME:='WELCH F-TEST';
         DISTRIBUTION:=FDISTRIBUTION;
         WELCHFTEST(MEAN,VARIANCE,GROUP,DFN,DFD,GPSIZE,VALUE);
         VALID:=NORMAL OR SYMMETRY;
      END;
   END;
END (*TESTOFSEVERALMEANS*);



PROCEDURE PAIRWISECOMPARISONORCONFIDENCEINTERVAL
                               (VAR DESCRIPTIVESTATISTIC : STATISTIC;
                                VAR TESTSTATISTIC : TEST;
                                VAR GPSIZE : GROUPSIZE;
                                VAR CONFIDENCEINTERVAL : INTERVAL;
                                VAR X : DATASET;
                                    MSE : REAL;
                                VAR GROUP,DIFFPAIR : INTEGER;
                                    EQVARIANCE : BOOLEAN);

    (*FOR CARRYING OUT PAIR-WISE COMPARISONS*)
    (*OR CONSTRUCTING CONFIDENCE INTERVAL   *)
```

```
VAR W,RECIPGPSIZE : GROUPSTAT;
    I,J,DF : INTEGER;
    Q,D,E : REAL;


    PROCEDURE CONSTRUCT(VAR CONFIDENCEINTERVAL : INTERVAL;
                        VAR MEAN,VARIANCE : GROUPSTAT;
                        VAR GPSIZE : GROUPSIZE;
                            GROUP,DFD     : INTEGER;
                            MSE,SIGLEV  : REAL);


       (*CONSTRUCT CONFIDENCE INTERVAL*)

       VAR U,D : REAL;

       BEGIN
          IF GROUP=1 THEN BEGIN
             U:=MEAN[1];
             D:=INVERTDIST(SIGLEV/2,(GPSIZE[1]-1))*SQRT(VARIANCE[1]/GPSIZE[1])
          END ELSE BEGIN
             U:=MEAN[1]-MEAN[2];
             D:=INVERTDIST(SIGLEV/2,DFD);
             IF EQVARIANCE THEN
                D:=D*SQRT(MSE*(1/GPSIZE[1]+1/GPSIZE[2]))
             ELSE
                D:=D*SQRT(VARIANCE[1]/GPSIZE[1]+VARIANCE[2]/GPSIZE[2]);
          END;
          WITH CONFIDENCEINTERVAL DO BEGIN
             UPPERLIMIT:=U+D;
             LOWERLIMIT:=U-D;
          END;
       END (*CONSTRUCT*);


    PROCEDURE COMPARE(V,E : REAL; VAR DIFFPAIR : INTEGER);

       (*DOING COMPARISON*)

       VAR K : INTEGER;

       BEGIN
          IF ABS(V)>E THEN BEGIN
             DIFFPAIR:=DIFFPAIR+1;
             K:=2*DIFFPAIR-1;
             X[K]:=I;    (*HERE X IS USED TO CARRY INFORMATION CONCERNING*)
             X[K+1]:=J; (*PAIRS WITH SIG. DIFFERENT MEANS.              *)
          END;
       END (*COMPARE*);


BEGIN
   DIFFPAIR:=0;
   WITH DESCRIPTIVESTATISTIC, TESTSTATISTIC DO
      IF SIGLEV>0 THEN BEGIN
         IF (GROUP<=2) AND (NOT RANDOMTEST) AND (NOT RANKSUMTEST) THEN
            CONSTRUCT(CONFIDENCEINTERVAL,MEAN,VARIANCE,GPSIZE,
                      GROUP,DFD,
                      MSE,SIGLEV)
         ELSE IF GROUP>2 THEN BEGIN
            IF EQVARIANCE THEN BEGIN
               Q:=SIGLEV/(GROUP*(GROUP-1.0));
```

```
            IF NAME='KRUSKAL-WALLIS' THEN BEGIN
               D:=INVERNORMAL(Q)*SQRT((TOTAL*(TOTAL+1.0)-TIECORR/
                  (TOTAL-1.0))/12.0);
               FOR I:=1 TO GROUP DO
                  W[I]:=RANKSUM[I]/GPSIZE[I]
            END ELSE (*NAME='F-TEST'*) BEGIN
               D:=INVERTDIST(Q,DFD)*SQRT(MSE);
               W:=MEAN;
            END;
            FOR I:=1 TO GROUP DO
               RECIPGPSIZE[I]:=1/GPSIZE[I];
            FOR I:=1 TO GROUP-1 DO
               FOR J:=I+1 TO GROUP DO
                  COMPARE((W[I]-W[J]),
                          (D*SQRT(RECIPGPSIZE[I]+RECIPGPSIZE[J])),DIFFPAIR);
         END ELSE (*NAME='WELCH F-TEST'*) BEGIN
            Q:=0.5*(1-EXP(2/(GROUP*(GROUP-1))*LN(1-SIGLEV)));
            FOR I:=1 TO GROUP DO
               W[I]:=VARIANCE[I]/GPSIZE[I];
            FOR I:=1 TO GROUP-1 DO
               FOR J:=I+1 TO GROUP DO BEGIN
                  D:=W[I]+W[J];
                  DF:=ROUND(SQR(D)/
                     (SQR(W[I])/(GPSIZE[I]-1.0)+SQR(W[J])/(GPSIZE[J]-1.0)));
                  E:=INVERTDIST(Q,DF)*SQRT(D);
                  COMPARE((MEAN[I]-MEAN[J]),E,DIFFPAIR);
               END;
         END;
      END;
   END;
END (*PAIRWISECOMPARISONORCONFIDENCEINTERVAL*);



FUNCTION RANDOMIZATIONTEST;


(*DECLARED FORWARD..PARA:(VAR  MEAN : GROUPSTAT; VAR Z : DATASET*)
(*                           N,M : INTEGER                      *)
(*NO RANGE CHECK IN THIS PROCEDURE*)

VAR K : INTEGER;
    OBSERSUM,COMB,C : REAL;


    PROCEDURE ARRANGEANDSCALEUPZ;

       (*ARRANGE AND SCALE-UP Z TO MAKE ITS REPRESENTATION MORE DISTINCT*)

       VAR I,K : INTEGER;

       BEGIN
         (*$R-*)    (*TURN OFF THE RANGE CHECK*)

         IF GROUP=2 THEN BEGIN
            (*REARRANGE Z TO EASE CALCULATION*)
            (*Z[1] TO Z[M] IS THE FIRST SAMPLE*)
            (*Z[M+1] TO Z[N] IS THE SECOND SAMPLE*)

            K:=T(2,0)-M;
            FOR I:=M+1 TO N DO
               Z[I]:=Z[K+I];
         END;
         FOR I:=1 TO N DO
            Z[I]:=2.0E4*Z[I];
       END (*ARRANGEANDSCALEUPZ*);
```

```
PROCEDURE GETOBSERSUM(VAR Z : DATASET;
                      VAR OBSERSUM : REAL;
                      VAR N,M : INTEGER);

   (*OBTAIN THE OBSERVED SUM*)

   VAR I,J,K : INTEGER;

   BEGIN
      (*$R-*)        (*TURN OFF THE RANGE CHECK*)

      OBSERSUM:=0;
      IF GROUP=1 THEN BEGIN

         (* OBSERSUM=MINIMUM(ABSOLUTE(SUM OF NEGATIVE Z), *)
         (*                  ABSOLUTE(SUM OF POSITIVE Z)) *)

         IF MEAN[1]>0 THEN BEGIN
            (*SUM OF POSITIVE Z > SUM OF NEGATIVE Z*)
            FOR I:=1 TO N DO IF Z[I]<0 THEN
               OBSERSUM:=OBSERSUM-Z[I];
         END ELSE
            FOR I:=1 TO N DO IF Z[I]>0 THEN
               OBSERSUM:=OBSERSUM+Z[I];
      END ELSE BEGIN

         (*OBSERSUM IS THE SUM OF THE SAMPLE WITH SMALLER MEAN*)
         IF MEAN[1]>MEAN[2] THEN BEGIN
            J:=M+1;
            K:=N;
            M:=N-M; (*SET M TO SAMPLE SIZE OF SECOND SAMPLE*)
         END ELSE BEGIN
            J:=1;
            K:=M;
         END;
         FOR I:=J TO K DO
            OBSERSUM:=OBSERSUM+Z[I];
      END;
   END (*GETOBSERSUM*);



PROCEDURE TAKEABSOLUTEZ;

   VAR I : INTEGER;

   BEGIN
      (*$R-*)      (*TURN OFF THE RANGE CHECK*)

      FOR I:=1 TO N DO
         Z[I]:=ABS(Z[I]);
   END (*TAKEABSOLUTEZ*);
```

```
PROCEDURE SORTZ;

   (*SORT Z IN ASCENDING ORDER*)

   VAR I,J : INTEGER;
       C : REAL;

   BEGIN
      (*$R-*)    (*TURN OFF THE RANGE CHECK*)
      FOR I:=1 TO N DO
         FOR J:=I+1 TO N DO
            IF Z[I]>Z[J] THEN BEGIN
               C:=Z[I];
               Z[I]:=Z[J];
               Z[J]:=C;
            END;
   END (*SORTZ*);



FUNCTION COUNT(VAR Z : DATASET;
                   N,R : INTEGER;
                   OBSERSUM : REAL) : REAL;

   (*COUNT NUMBER OF COMBINATIONS R OUT OF N NUMBERS WITH SUM<=OBSERSUM*)


   VAR A : ARRAY [1..20] OF 1..20;
       LAST,I,J,K : INTEGER;
       POSSIBLE,SUM,PARSUM,S : REAL;


   BEGIN
      (*$R-*)    (*TURN OFF THE RANGE-CHECK*)

      SUM:=0;
      FOR I:=1 TO R DO BEGIN
         A[I]:=I;
         SUM:=SUM+Z[I];
      END;
      IF SUM>OBSERSUM THEN
         COUNT:=0
      ELSE BEGIN
         POSSIBLE:=1;
         PARSUM:=SUM-Z[R];
         S:=OBSERSUM-PARSUM;
         WHILE (Z[N]>S) AND (N>R) DO
            N:=N-1 (*ELIMINATE Z[N]*);
         IF R=1 THEN
            POSSIBLE:=N
         ELSE IF R<N THEN BEGIN
            K:=R;
            LAST:=N+1-R;
            REPEAT
               IF K<N THEN BEGIN
                  PARSUM:=SUM-Z[K];
                  S:=OBSERSUM-PARSUM;
                  REPEAT
                     K:=K+1;
                     IF Z[K]>S THEN
                        K:=N   (*SET CONTROL TO BACK-TRACK*)
                     ELSE
                        POSSIBLE:=POSSIBLE+1
                  UNTIL K=N;
                  J:=R;
               END ELSE (* K=N *) BEGIN
```

```
                    J:=J-1;                          (*BACK-TRACK*)
                    K:=A[J];
                    PARSUM:=PARSUM-Z[K];
                    SUM:=PARSUM;
                    FOR I:=J TO R DO BEGIN
                       K:=K+1;
                       A[I]:=K;
                       SUM:=SUM+Z[K];
                    END (*K=A[R] ON EXITING THIS LOOP*);
                    IF SUM>OBSERSUM THEN BEGIN
                       IF J=1 THEN
                          A[1]:=LAST
                       ELSE
                          K:=N;    (*SET CONTROL TO BACK-TRACK*)
                    END ELSE (*SUM<=OBSERSUM*)
                       POSSIBLE:=POSSIBLE+1;
                 END;
              UNTIL A[1]=LAST;
           END;
           COUNT:=POSSIBLE;
        END;
     END (*COUNT*);



  FUNCTION COMBINATION(P,Q : INTEGER) : REAL;

     (*COMPUTE COMBINATION Q OUT OF P, THUS P>=Q*)

     FUNCTION FACTORIAL(W:INTEGER):REAL;

        (*FACTORIAL OF W*)

        VAR F : REAL;
            J : INTEGER;

        BEGIN
           F:=1;
           FOR J:=W DOWNTO 2 DO
              F:=F*J;
           FACTORIAL:=F;
        END (*FACTORIAL*);


  BEGIN
     COMBINATION:=FACTORIAL(P)/(FACTORIAL(Q)*FACTORIAL(P-Q));
  END (*COMBINATION*);


BEGIN
   ARRANGEANDSCALEUPZ;
   GETOBSERSUM(Z,OBSERSUM,N,M);
   IF GROUP=1 THEN
      TAKEABSOLUTEZ;
   SORTZ;
   IF GROUP=1 THEN BEGIN
         COMB:=1.0;              (*THIS IS EQUAL TO COUNT(N,0)*)
         IF OBSERSUM>0 THEN BEGIN
            K:=0;
            REPEAT
               K:=K+1;
               C:=COUNT(Z,N,K,OBSERSUM);
               COMB:=COMB+C;
            UNTIL (K=N) OR (C=0);
         END;
```

```
         (* IF OBSERSUM=0 THEN ANY SUM OF A NON-EMPTY *)
         (* SUBSET OF Z[1] TO Z[N] IS > OBSERSUM      *)

         RANDOMIZATIONTEST:=COMB/EXP(N*LN(2));
   END ELSE
      RANDOMIZATIONTEST:=COUNT(Z,N,M,OBSERSUM)/COMBINATION(N,M);
   (*$R+*)   (*RANGE-CHECK BACK ON*)
END (*RANDOMIZATIONTEST*);



FUNCTION ONESAMPLETTEST;

(*DECLARED FORWARD..PARA:( SAMPLEMEAN,SAMPLEVARIANCE : REAL*)
(*                        SAMPLESIZE :INTEGER            *)


BEGIN
   ONESAMPLETTEST:=SAMPLEMEAN/SQRT(SAMPLEVARIANCE/SAMPLESIZE);
END (*ONETTEST*);



FUNCTION TWOTTEST;

(*DECLARED FORWARD..PARA:(VAR MEAN : GROUPSTAT             *)
(*                       VAR GPSIZE : GROUPSIZE; MSE :REAL*)


BEGIN
   TWOTTEST:=(MEAN[1]-MEAN[2])/SQRT(MSE*(1/GPSIZE[1]+1/GPSIZE[2]));
END (*TWOTTEST*);



PROCEDURE WELCHTTEST;

(*DECLARED FORWARD*)
(*PARA:(VAR MEAN,VARIANCE : GROUPSTAT       *)
(*      VAR GPSIZE : GROUPSIZE              *)
(*      VAR DFD : INTEGER; VAR WELCHT : REAL*)


VAR D,E,F : REAL;

BEGIN
   D:=VARIANCE[1]/GPSIZE[1];
   E:=VARIANCE[2]/GPSIZE[2];
   F:=D+E;
   DFD:=ROUND(SQR(F)/(SQR(D)/(GPSIZE[1]-1.0)+SQR(E)/(GPSIZE[2]-1.0)));
   WELCHT:=(MEAN[1]-MEAN[2])/SQRT(F);
END (*WELCHTTEST*);
```

```
PROCEDURE WELCHFTEST;

(*DECLARED FORWARD*)
(*PARA: ( VAR MEAN,VARIANCE : GROUPSTAT                *)
(*            GROUP,DFN : INTEGER; VAR DFD : INTEGER  *)
(*        VAR GPSIZE : GROUPSIZE; VAR WELCHF : REAL   *)


VAR W : GROUPSTAT;
    I : INTEGER;
    R,S,U,WM,NN,DN : REAL;


BEGIN
   U:=0;
   S:=0;
   FOR I:=1 TO GROUP DO BEGIN
      W[I]:=GPSIZE[I]/VARIANCE[I];
      U:=U+W[I];
      S:=S+W[I]*MEAN[I];
   END;
   WM:=S/U;
   DN:=0;
   R:=0;
   NN:=0;
   FOR I:=1 TO GROUP DO BEGIN
      S:=MEAN[I]-WM;
      R:=R+W[I]*S;
      NN:=NN+W[I]*SQR(S);
      DN:=DN+SQR((U-W[I])/U)/(GPSIZE[I]-1.0);
   END;
   NN:=(NN-SQR(R)/U)/DFN;
   S:=DN/(SQR(GROUP)-1.0);
   DFD:=ROUND(1/(3.0*S));
   WELCHF:=NN/(1.0+2*(GROUP-2.0)*S);
END (*WELCHFTEST*);
```

```
PROCEDURE RANKTEST;

(*DECLARED FORWARD..PARA:( Z : DATASET; GPSIZE : GROUPSIZE        *)
(*                         RANKSUM : GROUPSTAT;                   *)
(*                         GROUP,TOTAL : INTEGER;                 *)
(*                         NAME : STRING; VAR VALUE,TIECORR : REAL*)

CONST ROOT24=4.8989795    (*SQUARE ROOT OF 24*);
      ROOT12=3.4641016    (*SQUARE ROOT OF 12*);

VAR I,J,K,L,M,FOLD : INTEGER;
    P,Q,S : REAL;

BEGIN
   FOLD:=0;
   TIECORR:=0;
   FOR I:=1 TO GROUP DO
      RANKSUM[I]:=GPSIZE[I];
   FOR I:=1 TO GROUP DO
      FOR J:=T(I,1) TO T(I,GPSIZE[I]) DO BEGIN
         FOR K:=I TO GROUP DO BEGIN
            IF K>I THEN
               M:=T(K,1)
            ELSE
               M:=J+1;
            FOR L:=M TO T(K,GPSIZE[K]) DO
               IF Z[J]<Z[L] THEN
                  RANKSUM[K]:=RANKSUM[K]+1
               ELSE IF Z[J]>Z[L] THEN
                  RANKSUM[I]:=RANKSUM[I]+1
               ELSE BEGIN
                  RANKSUM[I]:=RANKSUM[I]+0.5;
                  RANKSUM[K]:=RANKSUM[K]+0.5;
                  FOLD:=FOLD+1;
               END;
         END;
         IF FOLD>0 THEN BEGIN
            TIECORR:=TIECORR+3*FOLD*(FOLD+1.0);
            FOLD:=0;
         END;
      END;
   IF NAME='PAIRED SIGNED-RANK WILCOXON' THEN

      (*NORMALIZATION FOR APPROXIMATION OF P-VALUE*)
      VALUE:=ROOT24*(ABS(RANKSUM[1]-TOTAL*(TOTAL+1.0)/4)-0.5)
             /SQRT(TOTAL*(TOTAL+1.0)*(2.0*TOTAL+1.0)-0.5*TIECORR)
   ELSE IF NAME='WILCOXON RANK SUM' THEN BEGIN

      (*NORMALIZATION FOR APPROXIMATION OF P-VALUE*)
      P:=GPSIZE[1];
      Q:=GPSIZE[2];
      VALUE:=ROOT12*(ABS(RANKSUM[1]-0.5*P*(TOTAL+1.0))-0.5)/
             SQRT(P*Q*(TOTAL+1.0-TIECORR/(TOTAL*(TOTAL-1.0))));
   END ELSE (*NAME='KRUSKAL-WALLIS'*) BEGIN
      P:=TOTAL;
      Q:=0.5*(P+1);
      S:=0;
      FOR I:=1 TO GROUP DO
         S:=S+SQR(RANKSUM[I]-GPSIZE[I]*Q)/GPSIZE[I];
      VALUE:=12.0*S/(P*(P+1)*(1-TIECORR/(P*(SQR(P)-1))));
   END;
END (*RANKTEST*);
```

```
FUNCTION INVERTDIST;

(*DECLARED FORWARD..PARA:( P : REAL; N : INTEGER*)


(* P: P-VALUE, N: DEGREES OF FREEDOM        *)

VAR X,XSQ,A,B,C,D,E,Q : REAL;

BEGIN
   X:=INVERNORMAL(P);
   XSQ:=SQR(X);
   A:=(XSQ+1)/4.0;
   B:=((5.0*XSQ+16.0)*XSQ+3.0)/96.0;
   C:=(((3.0*XSQ+19.0)*XSQ+17)*XSQ-15)/384;
   D:=((((79.0*XSQ+776.0)*XSQ+1482)*XSQ-1920)*XSQ-945)/92160.0;
   E:=(((((27.0*XSQ+339.0)*XSQ+930.0)*XSQ-1782.0)*XSQ-765.0)*XSQ+17955.0)
      /368640.0;
   Q:=1/N;
   INVERTDIST:=X*(1+Q*(A+Q*(B+Q*(C+Q*(D+Q*E)))));
END (*INVERTDIST*);




FUNCTION INVERNORMAL;

(*DECLARED FORWARD..PARA:(VAR  P : REAL *)


CONST PIDIVTWO=1.570796327 (*=PI/2*);
      TWOPI=6.283185308    (*=2.PI*);

VAR T,TSQ : REAL;

BEGIN
   IF P>1.01E-6 THEN BEGIN
      TSQ:=-PIDIVTWO*LN(4*P*(1-P));
      T:=SQRT(TSQ);
      INVERNORMAL:=T*(TSQ*(TSQ*(0.0000043728*TSQ-0.00028810)+0.0078365)+1);
   END ELSE BEGIN
      T:=-2*LN(P);
      TSQ:=T-LN(TWOPI*T);
      T:=SQRT(TSQ);
      INVERNORMAL:=T+(0.1633+0.5962/T)/TSQ;
   END;
END (*INVERNORMAL*);




BEGIN               (*PROCEDURE, CALCULATION*)
   IF GETTEST THEN BEGIN
      GETTEST:=FALSE;
      IF GROUP=1 THEN
```

```
BEGIN                (*PROCEDURE, CALCULATION*)
   IF GETTEST THEN BEGIN
      GETTEST:=FALSE;
      IF GROUP=1 THEN
         TESTOFSINGLEMEAN(DATA,X,GPSIZE,DESCRIPTIVESTATISTIC,TESTSTATISTIC,
                          GPSIZEALLOW,SYMMETRY,NORMAL,TESTTHEOMEAN,
                          TRANSFORM)
      ELSE IF GROUP=2 THEN
         TESTOFTWOMEANS(X,GPSIZE,DESCRIPTIVESTATISTIC,TESTSTATISTIC,
                        GROUP,TOTAL,MINGPSIZE,MSE,
                        NORMAL,SYMMETRY,EQVARIANCE,TRANSFORM,DATAKIND)
      ELSE
         TESTOFSEVERALMEAN(X,GPSIZE,DESCRIPTIVESTATISTIC,TESTSTATISTIC,
                           GROUP,TOTAL,MINGPSIZE,MAXGPSIZE,
                           MSE,
                           NORMAL,SYMMETRY,EQVARIANCE,TRANSFORM,DATAKIND);
   END ELSE
      PAIRWISECOMPARISONORCONFIDENCEINTERVAL(DESCRIPTIVESTATISTIC,
                                             TESTSTATISTIC,
                                             GPSIZE,
                                             CONFIDENCEINTERVAL,
                                             X,
                                             MSE,
                                             GROUP,DIFFPAIR,
                                             EQVARIANCE);
END                  (*CALCULATION*);
```

```
SEGMENT PROCEDURE PRINTRESULTS(VAR DATA,X : DATASET;
                               VAR DESCRIPTIVESTATISTIC : STATISTIC;
                               VAR TESTSTATISTIC        : TEST;
                               VAR HISTOGRAM            : GRAPH;
                               VAR CONFIDENCEINTERVAL   : INTERVAL;
                               VAR GPSIZE : GROUPSIZE;
                                   DATAKIND : DATATYPE;
                                   PROBLEM,GROUP,TOTAL,OUTLIER : INTEGER;
                                   DIFFPAIR : INTEGER;
                                   THEOMEAN,BSS,WSS,MSE,MINDATA : REAL;
                               VAR ADDCONST     : REAL;
                               VAR TRANSFORM : TYPEOFTRANSFORMATION;
                                   PAIRED,TAKEDIFFERENCE : BOOLEAN;
                                   NORMAL,SYMMETRY       : BOOLEAN;
                                   TESTTHEOMEAN,TOOMANYEQ: BOOLEAN;
                               VAR WANTTRANSFORM,RESUME  : BOOLEAN);



 VAR Z : TEXT;
     S,CHANNEL : STRING;
     HARDCOPY  : BOOLEAN; (*TRUE ONLY IF A HARD COPY HAS BEEN PRODUCED*)



PROCEDURE DRAWLINE(VAR Z : TEXT;
                       D : INTEGER;
                       TRAIL : CHAR);

   (*PROCEDURE FOR DRAWING LINE OF D LENGTH WITH TRAIL*)

   FORWARD;



PROCEDURE CHOOSECHANNEL(VAR Z : TEXT;
                        VAR CHANNEL : STRING;
                        VAR HARDCOPY : BOOLEAN);
VAR DUMMYSTR : STRING;
    J : INTEGER;

BEGIN
   WRITELN;
   WRITELN('WHAT KIND OF OUTPUT DO YOU WANT ?');
   WRITELN('NOTE: YOU MAY HAVE MORE THAN ONE OUTPUT,');
   WRITELN('      THESE WILL BE PRODUCED SEQUENTIALLY.');
   WRITELN('1. CONSOLE:');
   WRITELN('2. PRINTER:');
   READINTEGER(1,2,FALSE,DUMMYSTR,J);
   CASE J OF
        1: CHANNEL:='CONSOLE:';
        2: CHANNEL:='PRINTER:';
   END;
   IF NOT HARDCOPY THEN
      HARDCOPY:=(CHANNEL<>'CONSOLE:');
   REWRITE(Z,CHANNEL);
   WRITELN(Z,'RESULTS');
END (*CHOOSECHANNEL*);
```

```
PROCEDURE PRINTDATA(VAR Z : TEXT;
                    VAR DESCRIPTIVESTATISTIC : STATISTIC;
                    VAR DATA : DATASET;
                    VAR GPSIZE : GROUPSIZE;
                        GROUP,TOTAL : INTEGER;
                        ADDCONST    : REAL;
                        TESTTHEOMEAN,PAIRED,TAKEDIFFERENCE : BOOLEAN);

VAR I,J : INTEGER;

BEGIN
   WRITELN(Z,'DATA : ');
   IF TESTTHEOMEAN THEN
      WRITELN(Z,'OBSERVATION - THEORETICAL MEAN')
   ELSE IF PAIRED THEN
      WRITELN(Z,'FIRST GROUP - SECOND GROUP')
   ELSE IF TAKEDIFFERENCE THEN
      WRITELN(Z,'FIRST OBSERVATION - SECOND OBSERVATION');
   IF GROUP>1 THEN
      WRITELN(Z,'TOTAL NUMBER =',TOTAL);
   IF WANTTRANSFORM THEN BEGIN
      WRITE(Z,'TAKING ');
      CASE TRANSFORM OF
           SQUAREROOT   : WRITE(Z,'SQUARE ROOT');
           LOGARITHMIC  : WRITE(Z,'LOGARITHM');
           RECIPROCAL   : WRITE(Z,'RECIPROCAL');
           ARCSINE      : WRITE(Z,'ARCSINE');
      END;
      WRITE(Z,' (DATA');
      IF ADDCONST<>0 THEN
         WRITE(Z,' +',ADDCONST:8:3);
      WRITELN(Z,')');
      IF TRANSFORM=ARCSINE THEN BEGIN
         WRITELN(Z,'NOTE: 0 IS COUNTED AS 0.25/GROUP SIZE AND 100');
         WRITELN(Z,'      AS 100-(0.25/GROUP SIZE) BEFORE TRANSFORMATION.');
      END ELSE IF TRANSFORM=LOGARITHMIC THEN
         WRITELN(Z,'NOTE: THE BASE OF THE LOGARITHM IS 10.');
   END;
   WITH DESCRIPTIVESTATISTIC DO
       FOR I:=1 TO GROUP DO BEGIN
          WRITELN(Z);
          IF GROUP>1 THEN
             WRITE(Z,'GROUP',I:3);
          WRITELN(Z,'NUMBER =':12,GPSIZE[I]:4,'MINIMUM =':13,
                    MINIMUM[I]:10:3,
                    'MAXIMUM =':11,MAXIMUM[I]:10:3);
          FOR J:=1 TO GPSIZE[I] DO BEGIN
             WRITE(Z,DATA[T(I,J)]:13:3);
             IF ((J MOD 6)=0) OR (J=GPSIZE[I]) THEN
                WRITELN(Z);
          END;
       END;
   WRITELN(Z);
END (*PRINTDATA*);
```

```
PROCEDURE PRINTHISTOGRAM(VAR Z : TEXT;
                         VAR HISTOGRAM : GRAPH;
                             GROUP     : INTEGER);

VAR I,J,K,P,Q,L,M : INTEGER;

BEGIN
   WITH HISTOGRAM DO BEGIN
      WRITELN(Z,'HISTOGRAM');
      DRAWLINE(Z,9,'=');
      I:=0;
      M:=60 DIV HEIGHT;
      REPEAT
         P:=I+1;
         L:=1;
         WRITELN(Z);
         WRITE(Z,'MIDPOINTS':12);
         IF GROUP>1 THEN
            WRITE(Z,'GROUP':7,P:3);
         WHILE (L<M) AND (P<GROUP) DO BEGIN
            P:=P+1;
            L:=L+1;
            WRITE(Z,'GROUP':(HEIGHT-3),P:3);
         END;
         WRITELN(Z);
         DRAWLINE(Z,78,'-');
         FOR J:=1 TO NOOFINTERVAL DO BEGIN
            WRITE(Z,(MAXMIDPOINT-(J-1)*STEP):12:DECPL,' )','  ');
            P:=I;
            L:=0;
            REPEAT
               P:=P+1;
               L:=L+1;
               Q:=HEIGHT;
               K:=FREQUENCY[P,J];
               WHILE K>=REPRESENTCASE DO BEGIN
                  WRITE(Z,'*');
                  Q:=Q-1;
                  K:=K-REPRESENTCASE;
               END;
               IF K>0 THEN BEGIN
                  WRITE(Z,'X');
                  Q:=Q-1;
               END;
               WRITE(Z,' ':Q);
            UNTIL (P=GROUP) OR (L=M);
            WRITELN(Z);
         END;
         I:=I+L;
         DRAWLINE(Z,78,'-');
      UNTIL I=GROUP;
      WRITE(Z,'NOTE : AN * REPRESENTS ',REPRESENTCASE,' CASE(S).');
      IF REPRESENTCASE>1 THEN
         WRITELN(Z,' AN X REPRESENTS LESS THAN ',REPRESENTCASE,
                   ' CASES.');
      WRITELN(Z);
   END;
END (*PRINTHISTOGRAM*);
```

```
PROCEDURE PRINTDESCRIPTIVESTATISTIC(VAR Z : TEXT;
                                    VAR DESCRIPTIVESTATISTIC : STATISTIC;
                                        GROUP : INTEGER);

VAR I,J : INTEGER;
    A : REAL;
    PRINTALLG2 : BOOLEAN;
BEGIN
   WRITELN(Z);
   WRITELN(Z,'SUMMARY':42);
   DRAWLINE(Z,80,'=');
   IF GROUP>1 THEN
      WRITE(Z,'GROUP')
   ELSE
      WRITE(Z,'  ');
   WRITELN(Z,'   MEAN    MEDIAN   STD. DEV.  STD. ERROR OF MEAN  RANGE',
             'G1':7,'G2':9);
   DRAWLINE(Z,80,'-');
   PRINTALLG2:=TRUE;
   WITH DESCRIPTIVESTATISTIC DO BEGIN
      FOR I:=1 TO GROUP DO BEGIN
         IF GROUP>1 THEN
            WRITE(Z,I:3);
         A:=SQRT(VARIANCE[I]);
         WRITE(Z,MEAN[I]:10:3,MEDIAN[I]:10:3,A:11:4,(A/SQRT(GPSIZE[I])):13:4,
                 (MAXIMUM[I]-MINIMUM[I]):14:3,G1[I]:8:3);
         IF GPSIZE[I]>49 THEN
            WRITELN(Z,G2[I]:10:3)
         ELSE BEGIN
            WRITELN(Z,'*':7);
            PRINTALLG2:=FALSE;
         END;
      END;
      DRAWLINE(Z,80,'=');
      IF GROUP>1 THEN
         WRITELN(Z,'COEFFICIENT OF VARIATION OF GROUP VARIANCES =',CV:6:3);
   END;
   WRITELN(Z,'NOTE: G1 AND G2 ARE FISHER''S G-STATISTICS.');
   IF NOT PRINTALLG2 THEN
      WRITELN(Z,'      * : TOO FEW DATA FOR USEFUL ESTIMATE.');
   IF OUTLIER>0 THEN BEGIN
      WRITELN(Z,'      INTERPRET TABLE ABOVE WITH CARE, BECAUSE');
      WRITELN(Z,OUTLIER:9,' DATA POINT(S) MAY BE TOO EXTREME.');
   END;
END (*PRINTDESCRIPTIVESTATISTIC*);
```

```
PROCEDURE PRINTTESTSTATISTIC(VAR Z : TEXT;
                             VAR TESTSTATISTIC : TEST;
                             VAR GPSIZE        : GROUPSIZE;
                                 BSS,WSS,MSE : REAL);

VAR I : INTEGER;

    PROCEDURE PRINTRANKSUM(VAR Z : TEXT;
                           VAR RANKSUM : GROUPSTAT;
                           VAR GPSIZE  : GROUPSIZE;
                               NOOFNONZERO : INTEGER);

       VAR I : INTEGER;

       BEGIN
          WRITELN(Z);
          IF GROUP=1 THEN BEGIN
             DRAWLINE(Z,30,'=');
             WRITELN(Z,'RANK SUM':30);
             DRAWLINE(Z,30,'-');
             WRITELN(Z,'POSITIVE DATA',RANKSUM[1]:15:1);
             WRITELN(Z,'NEGATIVE DATA',RANKSUM[2]:15:1);
             DRAWLINE(Z,30,'=');
             IF NOOFNONZERO<GPSIZE[1] THEN
                WRITELN(Z,'NOTE: ZEROS ARE EXCLUDED FROM CALCULATIONS.');
          END ELSE BEGIN
             DRAWLINE(Z,48,'=');
             WRITELN(Z,'GROUP    RANK SUM   GROUP SIZE  MEAN OF RANK SUM');
             DRAWLINE(Z,48,'-');
             FOR I:=1 TO GROUP DO
                WRITELN(Z,I:3,RANKSUM[I]:10:1,GPSIZE[I]:10,(RANKSUM[I]/
                          GPSIZE[I]):16:3);
             DRAWLINE(Z,48,'=');
          END;
       END (*PRINTRANKSUM*);


    PROCEDURE PRINTANOVATABLE(VAR Z : TEXT;
                                  DFN,DFD : INTEGER;
                                  BSS,WSS,MSE : REAL);


       BEGIN
          WRITELN(Z,'ANALYSIS OF VARIANCE':40);
          DRAWLINE(Z,60,'=');
          WRITELN(Z,'SOURCE','D. F.':15,'      SUM OF SQUARES     ',
                    'MEAN SQUARES');
          DRAWLINE(Z,60,'-');
          WRITELN(Z,'BETWEEN GROUPS',DFN:6,BSS:18:4,(BSS/DFN):18:4);
          WRITELN(Z,'WITHIN GROUPS',DFD:7,WSS:18:4,MSE:18:4);
          DRAWLINE(Z,60,'-');
          WRITELN(Z,'TOTAL',(DFN+DFD):15,(BSS+WSS):18:4);
          DRAWLINE(Z,60,'=');
       END (*PRINTANOVATABLE*);
```

```
BEGIN
   WRITELN(Z);
   WITH TESTSTATISTIC DO BEGIN
      IF TESTTHEOMEAN THEN
         WRITELN(Z,'THEORETICAL MEAN TESTED =',THEOMEAN:8:3);
      IF NAME='F-TEST' THEN
         PRINTANOVATABLE(Z,DFN,DFD,BSS,WSS,MSE)
      ELSE IF RANKSUMTEST THEN
         PRINTRANKSUM(Z,RANKSUM,GPSIZE,NOOFNONZERO);
      WRITE(Z,'TEST STATISTIC IS ',NAME);
      IF ((NOT RANDOMTEST) AND (NOT RANKSUMTEST)) OR (GROUP>2) THEN BEGIN
         WRITELN(Z,'=',VALUE:10:4);
         IF NOT RANKSUMTEST THEN BEGIN
            WRITE(Z,'WITH DEGREES OF FREEDOM =');
            IF DFN>1 THEN
               WRITE(Z,DFN,' AND ');
            WRITELN(Z,DFD);
         END;
      END ELSE IF RANKSUMTEST THEN
         (*SIGNED-RANK WILCOXON TEST OR WILCOXON RANK SUM TEST*)
         WRITELN(Z,'=',RANKSUM[1]:12:1);
      WRITELN(Z);
      WRITE(Z,'P-VALUE =',PVALUE:8:6);
      IF GROUP<3 THEN
         WRITE(Z,'   (ONE-SIDED)');
      WRITELN(Z);
      IF (GROUP<3) AND (PVALUE<0.5) AND (NOT RANDOMTEST) THEN
         WRITELN(Z,'=':9,(2*PVALUE):8:6,'   (TWO-SIDED)');
      WRITELN(Z);
   END;
END (*PRINTTESTSTATISTIC*);


PROCEDURE PRINTCONFIDENCEINTERVAL(VAR Z : TEXT;
                                  VAR CONFIDENCEINTERVAL : INTERVAL;
                                  VAR TESTSTATISTIC  : TEST;
                                      TESTTHEOMEAN,TAKEDIFFERENCE : BOOLEAN;
                                      GROUP : INTEGER);

(*CONFIDENCE INTERVAL FOR MEAN*)


BEGIN
   WITH TESTSTATISTIC, CONFIDENCEINTERVAL DO
      IF (NOT RANDOMTEST) AND (NOT RANKSUMTEST) THEN BEGIN
         WRITE(Z,ROUND(100*(1-SIGLEV)):3,'% CONFIDENCE INTERVAL OF ');
         IF TAKEDIFFERENCE OR (GROUP=2)THEN
            WRITE(Z,'DIFFERENCE OF MEANS (FIRST-SECOND)')
         ELSE
            WRITE(Z,'EXPERIMENTAL MEAN');
         IF TESTTHEOMEAN THEN
            WRITELN(Z,' - THEORETICAL MEAN')
         ELSE
            WRITELN(Z);
         WRITELN(Z,' BASED ON ',NAME,' IS');
         WRITELN(Z,'(',LOWERLIMIT:10:4,',':3,UPPERLIMIT:15:4,')');
      END;
END (*PRINTCONFIDENCEINTERVAL*);
```

```
PROCEDURE PRINTPAIRWISEDIFFERENCE(VAR Z : TEXT;
                                  VAR X : DATASET;
                                  VAR TESTSTATISTIC : TEST;
                                      DIFFPAIR      : INTEGER);


VAR I,J : INTEGER;

BEGIN
   IF DIFFPAIR>0 THEN BEGIN
      DRAWLINE(Z,78,'-');
      WRITELN(Z,'MEANS ARE DIFFERENT AT',TESTSTATISTIC.SIGLEV:6:3,
               ' SIG. LEVEL.');
      FOR I:=1 TO DIFFPAIR DO BEGIN
         J:=2*I;
         WRITELN(Z,'DUE TO DIFFERENCE OF MEANS OF GROUPS',ROUND(X[J-1]):4,
                  ' AND',ROUND(X[J]):4);
      END;
      DRAWLINE(Z,78,'-');
   END;
END (*PRINTPAIRWISEDIFFERENCE*);




PROCEDURE COMMENTFIRSTPROBLEM(VAR Z : TEXT;
                                  SYMMETRY : BOOLEAN);

BEGIN
   IF NOT SYMMETRY THEN BEGIN
      WRITELN(Z,'COMMENT: DATA ARE NOT SYMMETRICAL, THE MEAN MAY BE A POOR');
      WRITELN(Z,'MEASURE OF CENTRAL TENDENCY, AND THE MEDIAN IS PREFERABLE.');
   END;
END (*COMMENTFIRSTPROBLEM*);




PROCEDURE COMMENTSECONDPROBLEM(VAR Z : TEXT;
                                   TOOMANYEQ : BOOLEAN);

BEGIN
   IF TOOMANYEQ THEN BEGIN
      WRITE(Z,'COMMENT: DATA ');
      IF GROUP>1 THEN
         WRITE(Z,'FOR AT LEAST ONE OF THE GROUPS ');
      WRITELN(Z,'HAVE TOO MANY EQUAL VALUES.');
   END;
END (*COMMENTSECONDPROBLEM*);
```

```
PROCEDURE GIVEWARNING(VAR Z : TEXT;
                      VAR TESTSTATISTIC : TEST;
                          GROUP : INTEGER;
                          NORMAL,SYMMETRY : BOOLEAN);


BEGIN
    WRITELN(Z);
    WITH TESTSTATISTIC DO BEGIN
       IF (NOT VALID) OR (NOT SYMMETRY) THEN BEGIN
          WRITELN(Z,'WARNING:');
          DRAWLINE(Z,8,'=');
          IF NOT SYMMETRY THEN BEGIN
             WRITE(Z,'DATA ');
             IF GROUP>1 THEN
                WRITE(Z,'FOR AT LEAST ONE OF THE GROUPS ');
             WRITELN(Z,'ARE NOT SYMMETRICALLY DISTRIBUTED.');
          END;
          IF (NOT VALID) AND (NOT NORMAL) THEN BEGIN
             WRITELN(Z,'NORMALITY ASSUMPTION DOES NOT HOLD.');
             WRITELN(Z,'EFFECTS OF DEPARTURE FROM THE ASSUMPTION CAN');
             WRITELN(Z,'INVALIDATE THE TEST STATISTIC.');
             WRITELN(Z);
             WRITELN(Z,'YOU SHOULD CONSULT A STATISTICIAN.');
             DRAWLINE(Z,33,'=');
          END;
       END;
    END;
END (*GIVEWARNING*);




PROCEDURE CONSULT(VAR Z : TEXT);

BEGIN
   WRITELN(Z,'NOTE:');
   WRITELN(Z,'1. THE NULL HYPOTHESIS IS NOW THE EQUALITY OF MEANS ');
   WRITELN(Z,'   ON THE TRANSFORMED DATA.');
   WRITELN(Z,'2. IF CONCLUSIONS DRAWN FROM ANALYSES ON UNTRANSFORMED DATA');
   WRITELN(Z,'   AND TRANSFORMED DATA ARE DIFFERENT THEN CHOOSE THE ONE IN');
   WRITELN(Z,'   WHICH BOTH G1 AND G2 (IF GIVEN) STATISTICS ARE SMALLER OR');
   WRITELN(Z,'   MORE EQUAL OTHERWISE CHOOSE THE ONE WITH THE MORE MEANINGFUL'
           );
   WRITELN(Z,'   INTERPRETATION.');
END (*CONSULT*);
```

```
PROCEDURE CHOOSETRANSFORMATION(VAR DATA     : DATASET;
                               VAR GPSIZE   : GROUPSIZE;
                                   DATAKIND : DATATYPE;
                               VAR TRANSFORM    : TYPEOFTRANSFORMATION;
                               VAR MINDATA,ADDCONST : REAL;
                               VAR RESUME : BOOLEAN);

VAR I,L : INTEGER;
    S   : STRING;

    FUNCTION SOMEDATAAREZERO(VAR DATA   : DATASET;
                             VAR GPSIZE : GROUPSIZE) : BOOLEAN;

       (*CHECKING FOR DATA EQUAL ZERO*)

       VAR I,J,K : INTEGER;

       BEGIN
          IF MINDATA<0 THEN BEGIN
             I:=0;
             REPEAT
                I:=I+1;
                J:=T(I,0);
                K:=T(I,GPSIZE[I]);
                REPEAT
                   J:=J+1;
                UNTIL (DATA[J]=0) OR (J=K);
             UNTIL (I=GROUP) OR (DATA[J]=0);
             SOMEDATAAREZERO:=DATA[J]=0;
          END ELSE IF MINDATA>0 THEN
             SOMEDATAAREZERO:=FALSE
          ELSE
             SOMEDATAAREZERO:=TRUE;
       END (*SOMEDATASREZERO*);


    PROCEDURE FIXADDCONST(DATAKIND : DATATYPE;
                          VAR MINDATA,ADDCONST : REAL);

       (*FIX MINIMUM VALUE TO BE ADDED BEFORE TRANSFORMATION*)

       VAR DATUM,LOWERBOUND,UPPERBOUND : REAL;

       BEGIN
          READFILE('ADDCONST.TEXT'); (*READ EXPLANATION FROM DISK FILE*)
          WRITE('NUMBER TO BE ADDED MUST BE ');
          IF (TRANSFORM=LOGARITHMIC) OR (TRANSFORM=RECIPROCAL) THEN
             WRITE('LARGER THAN')
          ELSE
             WRITE('AT LEAST');
          WRITELN(ABS(MINDATA):10:3);
          CASE DATAKIND OF
               SCORE,CONTINUOUS : LOWERBOUND:=-1.0E30;
               COUNT,BINOMIAL : LOWERBOUND:=0;
          END;
          IF DATAKIND=BINOMIAL THEN
             UPPERBOUND:=100
          ELSE
             UPPERBOUND:=1.0E30;
          REPEAT
             GETDATA('PLEASE ENTER A VALID NUMBER. ','NUMERIC',
                     LOWERBOUND,UPPERBOUND,S,DATUM);
          UNTIL ((DATUM>ABS(MINDATA))
                  AND ((TRANSFORM=RECIPROCAL) OR (TRANSFORM=LOGARITHMIC)))
                OR ((DATUM>=ABS(MINDATA)) AND (TRANSFORM=SQUAREROOT));
          ADDCONST:=DATUM;
       END (*FIXADDCONST*);
```

```
BEGIN
   WRITELN;
   WRITELN('WHICH TRANSFORMATION DO YOU WANT TO TRY ?');
   WRITELN('1. SQUARE ROOT.');
   WRITELN('2. LOGARITHMIC.');
   WRITELN('3. RECIPROCAL.');
   IF TRANSFORM=ARCSINE THEN BEGIN
      L:=5;
      WRITELN('4. ARCSINE.');
   END ELSE
      L:=4;
   WRITELN(L,'. NOT TO PROCEED.');
   WRITELN;
   IF TRANSFORM=IDENTITY  THEN
      WRITELN('NO HELP GIVEN, BECAUSE NO SUITABLE TRANSFORMATION FOUND.')
   ELSE BEGIN
      WRITE('TRANSFORMATION SUGGESTED IS ');
      CASE TRANSFORM OF
           SQUAREROOT  : WRITELN('SQUARE ROOT.');
           LOGARITHMIC : WRITELN('LOGARITHMIC.');
           RECIPROCAL  : WRITELN('RECIPROCAL.');
           ARCSINE     : WRITELN('ARCSINE.');
      END;
   END;
   WRITELN;
   READINTEGER(1,L,FALSE,S,I);
   RESUME:=I<L;
   IF RESUME THEN BEGIN
      CASE I OF
           1: BEGIN
                 TRANSFORM:=SQUAREROOT;
                 IF MINDATA<0 THEN
                    FIXADDCONST(DATAKIND,MINDATA,ADDCONST)
                 ELSE
                    ADDCONST:=0;
              END;
           2: BEGIN
                 TRANSFORM:=LOGARITHMIC;
                 IF MINDATA<=0 THEN
                    FIXADDCONST(DATAKIND,MINDATA,ADDCONST)
                 ELSE
                    ADDCONST:=0;
              END;
           3: BEGIN
                 TRANSFORM:=RECIPROCAL;
                 IF SOMEDATAAREZERO(DATA,GPSIZE) THEN
                    FIXADDCONST(DATAKIND,MINDATA,ADDCONST)
                 ELSE
                    ADDCONST:=0;
              END;
           4: BEGIN
                 TRANSFORM:=ARCSINE;
                 ADDCONST:=0;
              END;
      END;
      WRITELN;
      WRITELN('PLEASE WAIT ! ANALYSIS CONTINUES.');
   END;
END (*CHOOSETRANSFORMATION*);
```

```
PROCEDURE DRAWLINE;

(*DECLARED FORWARD..PARA:(VAR  Z : TEXT; D : INTEGER; TRAIL : CHAR*)


VAR V : INTEGER;

BEGIN
   FOR V:=1 TO D DO
      WRITE(Z,TRAIL);
   WRITELN(Z);
END (*DRAWLINE*);




BEGIN                          (*SUB-PROBRAM, PRINTRESULTS*)
   WRITELN;
   WRITELN('HERE IS YOUR RESULT.');
   WRITELN;
   HARDCOPY:=FALSE;
   REPEAT
      CHOOSECHANNEL(Z,CHANNEL,HARDCOPY);
      PRINTDATA(Z,DESCRIPTIVESTATISTIC,DATA,GPSIZE,GROUP,TOTAL,ADDCONST,
                TESTTHEOMEAN,PAIRED,TAKEDIFFERENCE);
      PRINTHISTOGRAM(Z,HISTOGRAM,GROUP);
      PRINTDESCRIPTIVESTATISTIC(Z,DESCRIPTIVESTATISTIC,GROUP);
      CASE PROBLEM OF
            1: BEGIN
                  COMMENTFIRSTPROBLEM(Z,SYMMETRY);
                  PRINTCONFIDENCEINTERVAL(Z,CONFIDENCEINTERVAL,
                                          TESTSTATISTIC,
                                          TESTTHEOMEAN,TAKEDIFFERENCE,
                                          GROUP);
               END;

            2: BEGIN
                  PRINTTESTSTATISTIC(Z,TESTSTATISTIC,GPSIZE,BSS,WSS,MSE);
                  IF GROUP<3 THEN
                     PRINTCONFIDENCEINTERVAL(Z,CONFIDENCEINTERVAL,
                                             TESTSTATISTIC,
                                             TESTTHEOMEAN,TAKEDIFFERENCE,
                                             GROUP)
                  ELSE
                     PRINTPAIRWISEDIFFERENCE(Z,X,TESTSTATISTIC,DIFFPAIR);
                  COMMENTSECONDPROBLEM(Z,TOOMANYEQ);
                  GIVEWARNING(Z,TESTSTATISTIC,GROUP,NORMAL,SYMMETRY);
               END;
      END;
      IF WANTTRANSFORM THEN
         CONSULT(Z);
      CLOSE(Z);
```

```
      WRITELN;
      IF NOT HARDCOPY THEN
         WRITELN('WARNING : YOU HAVE NOT GOT ANY OUTPUT ON THE PRINTER.');
      WRITE('DO YOU WANT FURTHER OUTPUT ?');
      READSTR(FALSE,S);
      IF (PROBLEM=2) AND (S='N') AND (GROUP>1) THEN
         IF (NOT WANTTRANFORM) AND (DATAKIND<>SCORE) THEN BEGIN
            IF NOT TESTSTATISTIC.VALID THEN BEGIN
               WRITELN('NOTE: THE STATISTICAL ASSUMPTIONS ON WHICH THIS TEST');
               WRITELN('      DEPENDS ARE NOT MET.');
               WRITE('DO YOU WISH TO TRANSFORM YOUR DATA ?');
            END ELSE
               WRITELN('DO YOU WISH TO ANALYSE YOUR DATA ON SOME OTHER',
               ' SCALE ?');
            READSTR(FALSE,S);
            RESUME:=S='Y';
            IF RESUME THEN BEGIN
               WRITELN;
               WRITELN('WARNING: THE RESULTS OF THE ANALYSIS JUST',
                       ' NOW WILL BE LOST.');
               WRITE('DO YOU WANT FURTHER OUTPUT ?');
               READSTR(FALSE,S);
            END;
         END;
   UNTIL S='N';
   IF RESUME THEN BEGIN
      CHOOSETRANSFORMATION(DATA,GPSIZE,DATAKIND,TRANSFORM,
                           MINDATA,ADDCONST,RESUME);
      WANTTRANSFORM:=RESUME;
   END;
END                                (*PRINTRESULT*);
```

EXAMPLES

EXAMPLE 1.

RESULTS
DATA :
FIRST GROUP - SECOND GROUP

```
    NUMBER =    9     MINIMUM =     -5.000  MAXIMUM =      9.000
       -1.000          1.000        -2.000       -5.000         2.000        -1.000
       -1.000          9.000        -2.000
```

HISTOGRAM
=========

```
   MIDPOINTS
-------------------------------------------------------------------------------
      10.000 )
       9.000 )  *
       8.000 )
       7.000 )
       6.000 )
       5.000 )
       4.000 )
       3.000 )
       2.000 )  *
       1.000 )  *
       0.000 )
      -1.000 )  ***
      -2.000 )  **
      -3.000 )
      -4.000 )
      -5.000 )  *
      -6.000 )
-------------------------------------------------------------------------------
```

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
0.000	-1.000	3.9051	1.3017	14.000	1.603	*

NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.
INTERPRET TABLE ABOVE WITH CARE, BECAUSE
1 DATA POINT(S) MAY BE TOO EXTREME.

TEST STATISTIC IS PAIRED RANDOMIZATION
P-VALUE = 0.531250 (ONE-SIDED)

EXAMPLE 2.

RESULTS
DATA :
TOTAL NUMBER =21

GROUP 1 NUMBER = 11 MINIMUM = 0.000 MAXIMUM = 4.000

1.000	1.000	1.000	1.000	1.000	1.000
2.000	0.000	0.000	4.000	3.000	

GROUP 2 NUMBER = 10 MINIMUM = 0.000 MAXIMUM = 5.000

0.000	0.000	0.000	0.000	0.000	0.000
0.000	1.000	1.000	5.000		

HISTOGRAM
=========

MIDPOINTS	GROUP 1	GROUP 2
5.500)		
5.000)		*
4.500)		
4.000)	*	
3.500)		
3.000)	*	
2.500)		
2.000)	*	
1.500)		
1.000)	******	**
0.500)		
0.000)	**	*******
-0.500)		

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

GROUP	MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
1	1.364	1.000	1.2060	0.3636	4.000	1.226	*
2	0.700	0.000	1.5670	0.4955	5.000	2.785	*

COEFFICIENT OF VARIATION OF GROUP VARIANCES = 0.259
NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.
INTERPRET TABLE ABOVE WITH CARE, BECAUSE
1 DATA POINT(S) MAY BE TOO EXTREME.

TEST STATISTIC IS TWO-SAMPLE T-TEST= 1.0937
WITH DEGREES OF FREEDOM =19

P-VALUE = 0.143888 (ONE-SIDED)
= 0.287775 (TWO-SIDED)

95% CONFIDENCE INTERVAL OF DIFFERENCE OF MEANS (FIRST-SECOND)
BASED ON TWO-SAMPLE T-TEST IS
(-0.6065 , 1.9338)
COMMENT: DATA FOR AT LEAST ONE OF THE GROUPS HAVE TOO MANY EQUAL VALUES.

WARNING:
========
DATA FOR AT LEAST ONE OF THE GROUPS ARE NOT SYMMETRICALLY DISTRIBUTED.
NORMALITY ASSUMPTION DOES NOT HOLD.
EFFECTS OF DEPARTURE FROM THE ASSUMPTION CAN
INVALIDATE THE TEST STATISTIC.

YOU SHOULD CONSULT A STATISTICIAN.
==========

EXAMPLE 3.

RESULTS

DATA :

TOTAL NUMBER =60

GROUP 1 NUMBER = 17 MINIMUM = 0.600 MAXIMUM = 2.300

1.230	2.300	1.250	1.000	0.900	0.800
0.700	0.600	0.950	0.980	1.100	1.200
1.100	1.100	1.200	1.300	1.200	

GROUP 2 NUMBER = 31 MINIMUM = 0.500 MAXIMUM = 8.700

0.500	1.500	1.500	1.400	1.350	1.440
0.660	0.650	0.670	0.680	0.900	1.000
1.000	1.200	1.400	1.300	1.500	2.300
1.800	1.700	1.660	1.500	7.800	7.700
7.900	8.000	6.000	6.500	6.300	6.200
8.700					

GROUP 3 NUMBER = 12 MINIMUM = 0.560 MAXIMUM = 0.980

0.700	0.800	0.900	0.870	0.980	0.780
0.800	0.900	0.560	0.660	0.780	0.800

HISTOGRAM

=========

MIDPOINTS	GROUP 1	GROUP 2	GROUP 3
9.000)			
8.500)		*	
8.000)		***	
7.500)		*	
7.000)			
6.500)		**	
6.000)		**	
5.500)			
5.000)			
4.500)			
4.000)			
3.500)			
3.000)			
2.500)	*	*	
2.000)		*	
1.500)	**	**********	
1.000)	**********	****	*********
0.500)	**	*****	***
0.000)			

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

GROUP	MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
1	1.112	1.100	0.3652	0.0886	1.700	2.088	*
2	2.991	1.500	2.8302	0.5083	8.200	1.019	*
3	0.794	0.800	0.1148	0.0331	0.420	-0.496	*

COEFFICIENT OF VARIATION OF GROUP VARIANCES = 0.930
NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.
INTERPRET TABLE ABOVE WITH CARE, BECAUSE
1 DATA POINT(S) MAY BE TOO EXTREME.

TEST STATISTIC IS WELCH F-TEST= 14.3380
WITH DEGREES OF FREEDOM =2 AND 32

P-VALUE = 0.000036

MEANS ARE DIFFERENT AT 0.010 SIG. LEVEL.
DUE TO DIFFERENCE OF MEANS OF GROUPS 1 AND 2
DUE TO DIFFERENCE OF MEANS OF GROUPS 1 AND 3
DUE TO DIFFERENCE OF MEANS OF GROUPS 2 AND 3

WARNING:
========
DATA FOR AT LEAST ONE OF THE GROUPS ARE NOT SYMMETRICALLY DISTRIBUTED.
NORMALITY ASSUMPTION DOES NOT HOLD.
EFFECTS OF DEPARTURE FROM THE ASSUMPTION CAN
INVALIDATE THE TEST STATISTIC.

YOU SHOULD CONSULT A STATISTICIAN.
==========

RESULTS
DATA :
TOTAL NUMBER =60
TAKING SQUARE ROOT (DATA)

GROUP 1 NUMBER = 17 MINIMUM = 0.775 MAXIMUM = 1.517

1.109	1.517	1.118	1.000	0.949	0.894
0.837	0.775	0.975	0.990	1.049	1.095
1.049	1.049	1.095	1.140	1.095	

GROUP 2 NUMBER = 31 MINIMUM = 0.707 MAXIMUM = 2.950

0.707	1.225	1.225	1.183	1.162	1.200
0.812	0.806	0.819	0.825	0.949	1.000
1.000	1.095	1.183	1.140	1.225	1.517
1.342	1.304	1.288	1.225	2.793	2.775
2.811	2.828	2.449	2.550	2.510	2.490
2.950					

GROUP 3 NUMBER = 12 MINIMUM = 0.748 MAXIMUM = 0.990

0.837	0.894	0.949	0.933	0.990	0.883
0.894	0.949	0.748	0.812	0.883	0.894

HISTOGRAM
=========

MIDPOINTS	GROUP 1	GROUP 2	GROUP 3
3.200)			
3.000)		*	
2.800)		****	
2.600)		**	
2.400)		**	
2.200)			
2.000)			
1.800)			
1.600)	*	*	
1.400)		**	
1.200)	***	**********	
1.000)	**********	****	****
0.800)	***	*****	********
0.600)			

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

GROUP	MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
1	1.043	1.049	0.1595	0.0387	0.742	1.312	*
2	1.561	1.225	0.7569	0.1359	2.242	0.838	*
3	0.889	0.894	0.0659	0.0190	0.242	-0.696	*

COEFFICIENT OF VARIATION OF GROUP VARIANCES = 0.897
NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.
INTERPRET TABLE ABOVE WITH CARE, BECAUSE
1 DATA POINT(S) MAY BE TOO EXTREME.

TEST STATISTIC IS WELCH F-TEST= 17.0557
WITH DEGREES OF FREEDOM =2 AND 35

P-VALUE = 0.000007

MEANS ARE DIFFERENT AT 0.010 SIG. LEVEL.
DUE TO DIFFERENCE OF MEANS OF GROUPS 1 AND 2
DUE TO DIFFERENCE OF MEANS OF GROUPS 1 AND 3
DUE TO DIFFERENCE OF MEANS OF GROUPS 2 AND 3

WARNING:
========
DATA FOR AT LEAST ONE OF THE GROUPS ARE NOT SYMMETRICALLY DISTRIBUTED.
NORMALITY ASSUMPTION DOES NOT HOLD.
EFFECTS OF DEPARTURE FROM THE ASSUMPTION CAN
INVALIDATE THE TEST STATISTIC.

YOU SHOULD CONSULT A STATISTICIAN.
==========

NOTE:
1. THE NULL HYPOTHESIS IS NOW THE EQUALITY OF MEANS
ON THE TRANSFORMED DATA.
2. IF CONCLUSIONS DRAWN FROM ANALYSES ON UNTRANSFORMED DATA
AND TRANSFORMED DATA ARE DIFFERENT THEN CHOOSE THE ONE IN
WHICH BOTH G1 AND G2 (IF GIVEN) STATISTICS ARE SMALLER OR
MORE EQUAL OTHERWISE CHOOSE THE ONE WITH THE MORE MEANINGFUL
INTERPRETATION.

EXAMPLE 4.

RESULTS
DATA :
TOTAL NUMBER =22

GROUP 1 NUMBER = 11 MINIMUM = 53.000 MAXIMUM = 137.000

57.000	120.000	101.000	137.000	119.000	117.000
104.000	73.000	53.000	68.000	118.000	

GROUP 2 NUMBER = 11 MINIMUM = 22.000 MAXIMUM = 96.000

89.000	30.000	82.000	50.000	39.000	22.000
57.000	32.000	96.000	31.000	88.000	

HISTOGRAM
=========

MIDPOINTS	GROUP 1	GROUP 2
150.000)		
140.000)	*	
130.000)		
120.000)	****	
110.000)		
100.000)	**	*
90.000)		**
80.000)		*
70.000)	**	
60.000)	*	*
50.000)	*	*
40.000)		*
30.000)		***
20.000)		*

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

GROUP	MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
1	97.000	104.000	29.1067	8.7760	84.000	-0.411	*
2	56.000	50.000	27.8352	8.3926	74.000	0.333	*

COEFFICIENT OF VARIATION OF GROUP VARIANCES = 0.045
NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.

TEST STATISTIC IS TWO-SAMPLE T-TEST= 3.3764
WITH DEGREES OF FREEDOM =20

P-VALUE = 0.001500 (ONE-SIDED)
= 0.003000 (TWO-SIDED)

99% CONFIDENCE INTERVAL OF DIFFERENCE OF MEANS (FIRST-SECOND)
BASED ON TWO-SAMPLE T-TEST IS
(6.4529 , 75.5471)

EXAMPLE 5.

RESULTS
DATA :
TOTAL NUMBER =24

GROUP 1 NUMBER = 6 MINIMUM = 56.000 MAXIMUM = 95.000
64.000 72.000 68.000 77.000 56.000 95.000

GROUP 2 NUMBER = 6 MINIMUM = 77.000 MAXIMUM = 97.000
78.000 91.000 97.000 82.000 85.000 77.000

GROUP 3 NUMBER = 6 MINIMUM = .63.000 MAXIMUM = 93.000
75.000 93.000 78.000 71.000 63.000 76.000

GROUP 4 NUMBER = 6 MINIMUM = 49.000 MAXIMUM = 70.000
55.000 66.000 49.000 64.000 70.000 68.000

HISTOGRAM
=========

MIDPOINTS	GROUP 1	GROUP 2	GROUP 3	GROUP 4
100.000)				
95.000)	*	*	*	
90.000)		*		
85.000)		*		
80.000)		**	*	
75.000)	*	*	**	
70.000)	**		*	**
65.000)	*		*	**
60.000)				
55.000)	*			*
50.000)				*
45.000)				

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

GROUP	MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
1	72.000	70.000	13.3417	5.4467	39.000	0.963	*
2	85.000	83.500	7.7717	3.1728	20.000	0.679	*
3	76.000	75.500	9.8793	4.0332	30.000	0.808	*
4	62.000	65.000	8.2219	3.3566	21.000	-0.939	*

COEFFICIENT OF VARIATION OF GROUP VARIANCES = 0.462
NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.

ANALYSIS OF VARIANCE

SOURCE	D. F.	SUM OF SQUARES	MEAN SQUARES
BETWEEN GROUPS	3	1636.50	545.500
WITHIN GROUPS	20	2018.00	100.900
TOTAL	23	3654.50	

TEST STATISTIC IS F-TEST= 5.4063
WITH DEGREES OF FREEDOM =3 AND 20

P-VALUE = 0.006876

MEANS ARE DIFFERENT AT 0.010 SIG. LEVEL.
DUE TO DIFFERENCE OF MEANS OF GROUPS 2 AND 4

EXAMPLE 6.

RESULTS
DATA :
TOTAL NUMBER =39

GROUP 1 NUMBER = 16 MINIMUM = 6.000 MAXIMUM = 13.000

12.000	13.000	12.000	8.000	8.000	9.000
10.000	10.000	10.000	7.000	10.000	6.000
7.000	7.000	7.000	7.000		

GROUP 2 NUMBER = 23 MINIMUM = 6.000 MAXIMUM = 17.000

6.000	8.000	10.000	8.000	10.000	10.000
11.000	11.000	17.000	16.000	15.000	14.000
15.000	15.000	14.000	14.000	12.000	13.000
13.000	12.000	12.000	13.000	13.000	

HISTOGRAM
=========

MIDPOINTS	GROUP 1	GROUP 2
18.000)		
17.000)		*
16.000)		*
15.000)		***
14.000)		***
13.000)	*	****
12.000)	**	***
11.000)		**
10.000)	****	***
9.000)	*	
8.000)	**	**
7.000)	*****	
6.000)	*	*
5.000)		

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

GROUP	MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
1	8.937	8.500	2.1438	0.5359	7.000	0.510	*
2	12.261	13.000	2.7339	0.5701	11.000	-0.490	*

COEFFICIENT OF VARIATION OF GROUP VARIANCES = 0.224
NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.

GROUP	RANK SUM	GROUP SIZE	MEAN OF RANK SUM
1	198.5	16	12.406
2	581.5	23	25.283

TEST STATISTIC IS WILCOXON RANK SUM= 198.5

P-VALUE = 0.000372 (ONE-SIDED)
= 0.000744 (TWO-SIDED)

EXAMPLE 7.

RESULTS
DATA :
FIRST GROUP - SECOND GROUP

NUMBER = 30	MINIMUM = -3.000	MAXIMUM = 8.000			
-2.000	0.000	0.000	1.000	0.000	0.000
4.000	4.000	1.000	1.000	5.000	3.000
5.000	3.000	-1.000	1.000	-1.000	5.000
8.000	2.000	2.000	2.000	-3.000	-2.000
1.000	4.000	8.000	2.000	3.000	-1.000

HISTOGRAM
=========

MIDPOINTS	
9.000)	
8.000)	**
7.000)	
6.000)	
5.000)	***
4.000)	***
3.000)	***
2.000)	****
1.000)	*****
0.000)	****
-1.000)	***
-2.000)	**
-3.000)	*
-4.000)	

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
1.833	1.500	2.7428	0.5008	11.000	0.498	*

NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.

TEST STATISTIC IS PAIRED T-TEST= 3.6611
WITH DEGREES OF FREEDOM =29

P-VALUE = 0.000498 (ONE-SIDED)
= 0.000996 (TWO-SIDED)

99% CONFIDENCE INTERVAL OF DIFFERENCE OF MEANS (FIRST-SECOND)
BASED ON PAIRED T-TEST IS
(0.4532 , 3.2135)

EXAMPLE 8.

RESULTS
DATA :
TOTAL NUMBER =14

GROUP 1 NUMBER = 5 MINIMUM = 1.000 MAXIMUM = 9.000
4.500 9.000 3.500 1.000 5.500

GROUP 2 NUMBER = 5 MINIMUM = 2.000 MAXIMUM = 11.000
2.000 8.500 10.000 11.000 6.500

GROUP 3 NUMBER = 4 MINIMUM = 7.000 MAXIMUM = 14.000
7.000 12.500 14.000 12.500

HISTOGRAM
=========

MIDPOINTS	GROUP 1	GROUP 2	GROUP 3
15.000)			
14.000)			*
13.000)			**
12.000)			
11.000)		*	
10.000)		*	
9.000)	*	*	
8.000)			
7.000)		*	*
6.000)	*		
5.000)	*		
4.000)	*		
3.000)			
2.000)		*	
1.000)	*		
0.000)			

NOTE : AN * REPRESENTS 1 CASE(S).

SUMMARY

GROUP	MEAN	MEDIAN	STD. DEV.	STD. ERROR OF MEAN	RANGE	G1	G2
1	4.700	4.500	2.9283	1.3096	8.000	0.458	*
2	7.600	8.500	3.5602	1.5922	9.000	-1.137	*
3	11.500	12.500	3.0822	1.5411	7.000	-1.673	*

COEFFICIENT OF VARIATION OF GROUP VARIANCES = 0.176
NOTE: G1 AND G2 ARE FISHER'S G-STATISTICS.
* : TOO FEW DATA FOR USEFUL ESTIMATE.

GROUP	RANK SUM	GROUP SIZE	MEAN OF RANK SUM
1	22.0	5	4.400
2	37.0	5	7.400
3	46.0	4	11.500

TEST STATISTIC IS KRUSKAL-WALLIS= 6.4198

P-VALUE = 0.023638

MEANS ARE DIFFERENT AT 0.050 SIG. LEVEL.
DUE TO DIFFERENCE OF MEANS OF GROUPS 1 AND 3