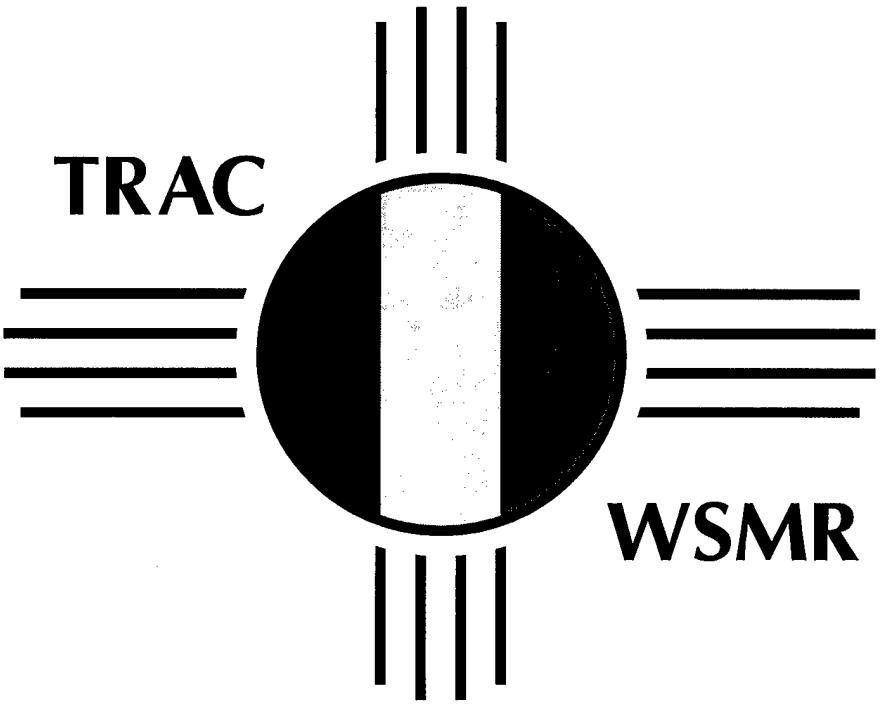
TRAC-WSMR-TM-98-015

Technique for Increasing Efficiency and Accuracy of Data for Mix Analysis



Bruce W. Gafner

July 1998

Technical Memorandum

DESTRUCTION NOTICE

Destroy by any method that will prevent disclosure of contents or reconstruction of the document.

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other official documents.

WARNING

Information and data are based on inputs available at the time of preparation. The results are subject to change.

TRAC-WSMR-TM-98-015

Technique for Increasing Efficiency and Accuracy of Data for Mix Analysis

Bruce W. Gafner

Technical Memorandum

DEPARTMENT OF THE ARMY
TRADOC Analysis Center-White Sands Missile Range (TRAC-WSMR)
White Sands Missile Range, NM 88002-5502

July 1998

Reviewed By

SERVANDO HERNANDEZ
Study Director
TRAC-WSMR

Released

ROY F. REYNOLDS
SES, US Army
Director, TRAC-WSMR

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1998	October 1997 - May 1998

4. TITLE AND SUBTITLE
Technique for Increasing Efficiency and Accuracy of Data for Mix Analysis

5. FUNDING NUMBERS

6. AUTHOR(S)
Bruce W. Gafner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
TRADOC Analysis Center-White Sands Missile Range (TRAC-WSMR)
ATTN: ATRC-WA
White Sands Missile Range, NM 88002-5502

8. PERFORMING ORGANIZATION REPORT NUMBER
TRAC-WSMR-TM-98-015

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
TRADOC Analysis Center-White Sands Missile Range (TRAC-WSMR)
ATTN: ATRC-WA
White Sands Missile Range, NM 88002-5502

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report is an examination of the issues and possible solutions to the problem of statistical overlapping groups. The research which was the basis for the development of improved accuracy for a mix model analysis, was conducted jointly by the Training and Doctrine Command (TRADOC) Analysis Center-White Sands Missile Range (TRAC-WSMR) and New Mexico State University (NMSU) as a dissertation research project by the author. The objective of this research and the conclusion reached in this report is to establish a statistic based method for separating treatment means into distinct groups without group overlap, thereby increasing accuracy of data used for mix analysis. Additionally, using computer processing improves the efficiency of data preparation and adds improved accuracy derived from the developed method. Six different representative data sets were processed using five different multiple comparison procedures and heuristic algorithms. The results were analyzed and the procedures that were most effective at producing the best groupings were identified. The Base Grouping Heuristic provided the best results when used with Tukey's multiple comparison procedure with a significance level of 0.05 or with Fisher's Least Significant Difference procedure with a significance level of 0.01. The Base Grouping Heuristic was then programmed into an automated procedure.

14. SUBJECT TERMS
multiple comparison procedures, mix analysis, mean grouping heuristics

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

Technique for Increasing Efficiency and Accuracy of Data for Mix Analysis

Study Gist

Purpose. This report is an examination of the issues and possible solutions to the problem of overlapping groups of "different" means, using a technique known as the Separation of Treatment Means. The research was the basis for the development of an automated procedure to improve the efficiency and accuracy of data for the Training and Doctrine Command Analysis Center-White Sands Missile Range (TRAC-WSMR) Mix Model.

Objective. The objective of this research is to establish a method for separating treatment means into distinct groups without overlap. Additionally, computer processing improves the efficiency of data preparation and the accuracy derived from the developed method.

Principal Findings. The Base Grouping Heuristic provided the best results when used with Tukey's multiple comparison procedure with a significance level of 0.05 or with Fisher's Least Significant Difference procedure with a significance level of 0.01.

Utility to the Army. This automated procedure will reduce the labor intensive data processing for the TRAC-WSMR Mix Model, which will in turn provide more time for analysis of results. This will result in increased acceptance and validation of the model.

Main Assumptions. Selected data sets are representative of future data sets. Minor data changes will not invalidate the process or results.

Principal Limitation. Algorithm code requires standard Statistical Package for the Social Sciences (SPSS) output and current TRAC-WSMR Mix Model input formats. Code will require modification to work correctly with other data formats.

Scope. This research addresses the Combined Arms and Support Task Force Evaluation Model (CASTFOREM) to TRAC-WSMR Mix Model data analysis process. The CASTFOREM engagement summary file provided the input data. The SPSS software package was used to statistically analyze the data. The algorithm developed to group the data was programmed in C computer code.

Approach. To increase the efficiency of data analysis an algorithm that could be automated was required. Six different representative data sets were processed using five different multiple comparison procedures and heuristic algorithms. The results were analyzed and the procedures that were most effective at producing the best groupings were identified. The best algorithm was then programmed into an automated procedure.

Study Sponsor. TRAC-WSMR

Performing Organization and Principal Author. TRAC-WSMR, Bruce Gafner

Literature Search. Key topics were searched in the Science Citation Index, Defense Technical Information Center (DTIC), and other databases. Subject was addressed to numerous statistical experts.

DTIC Accession Number. TBD

Start and Completion Dates. October 1997 - May 1998

Table of Contents

Acknowledgements

The author greatly appreciates the support that was given for this research and thanks all those involved. Richard Porter, past director of Brigade Combat Directorate, United States Army Training and Doctrine Command (TRADOC) Analysis Center-White Sands Missile Range (TRAC-WSMR), encouraged this research and provided the necessary resources. Mr. Porter was the study director of the Antiarmor Resource Requirements (A2R2) study, which provided the foundation and the catalyst for this research. Richard Laferriere developed the Mix Model in the early 1990s and the methodology has since been used in numerous studies. Mr. Laferriere was the Mix Model team leader for the A2R2 study and provided leadership and instruction to the author in programming and running of the Mix Model. Doug Shoop was very helpful with the programming logic and wrote most of the data reduction programming code. Mary Anne Staffeldt, Ph.D., Industrial Engineering Department, New Mexico State University, encouraged the incorporation of a work related project for Ph.D. dissertation research. Dr. Staffeldt assisted with the statistical analysis of this research.

Technique for Increasing Efficiency and Accuracy of Data for Mix Analysis

Purpose

This report is an examination of issues and possible solutions to the problem of statistical overlapping groups, using a technique known as the Separation of Treatment Means. During the Antiarmor Resource Requirements study conducted in 1996, the issue of overlapping groups, i.e., group means overlapping, was identified as an area that could benefit from additional research. The research, which was the basis for the development of improved accuracy for a mix model (MM) analysis, was conducted jointly by the United States (US) Army Training and Doctrine Command Analysis Center-White Sands Missile Range (TRAC-WSMR) and New Mexico State University as a dissertation research project by the author.

Objective

The objective of this research and the conclusion reached in this report is to establish a statistic based method for separating treatment means into distinct groups without group overlap, thereby increasing accuracy of data used for mix analysis. Additionally, using computer processing improves the efficiency of data preparation and adds improved accuracy derived from the developed method.

Background

TRAC-WSMR currently uses the TRAC-WSMR Mix Model to assist in providing top level US Army decision makers with major weapon system analysis. Figure 1 gives an overview of the TRAC-WSMR Mix Model. The mix model methodology has three major components: Exploratory Data Analysis, Scenario Analysis, and a Decision Support System. Scenario Analysis is the largest and most important component of the Brigade Mix Model (Laferriere, 1991) which was the predecessor of the TRAC-WSMR Mix Model. TRAC-WSMR work addresses the planning of a higher level brigade force structure.

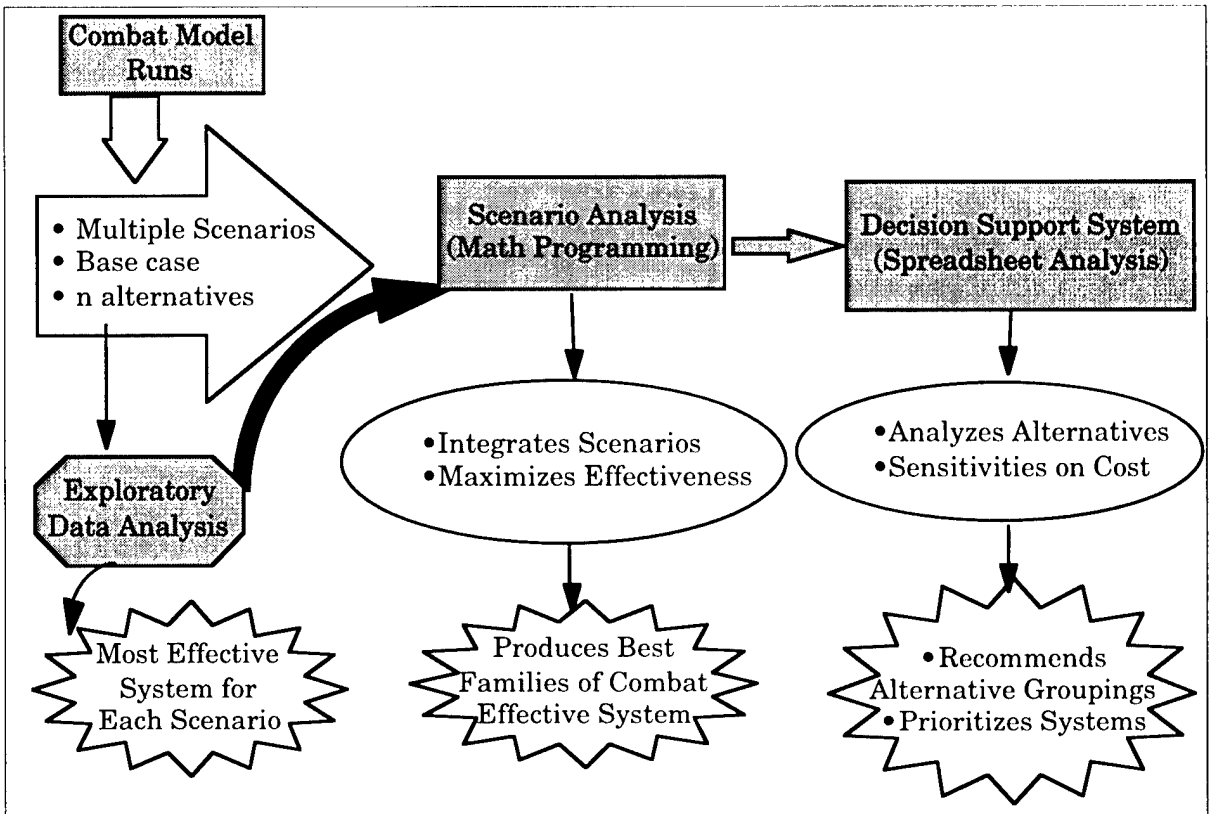


Figure 1. TRAC-WSMR Mix Model

Before running the Mix Model, TRAC-WSMR analysts evaluate the performance of lower level battalion sized task force combat units using the Combined Arms and Support Task Force Evaluation Model (CASTFOREM). CASTFOREM is a stochastic, two-sided, combat battle simulation used to analyze weapon systems, tactics, and organizations. Written in SIMSCRIPT II simulation language, it is the US Army's highest resolution combat simulation model (Mackey, et al., 1995). A team, usually composed of a number of Department of the Army civilians and military operations research analysts, may spend more than 6 months developing a new "base case" scenario for CASTFOREM. After developing an established base case, the team develops alternatives to the base case. Each alternative case usually consists of modifying a selected weapon system or a specific part of a base case scenario. Once the alternatives cases are developed, they are run in the model to determine the contribution of that alternative system, or set of systems, to the outcome of the CASTFOREM battle. There may be many alternative cases. For a recent study, five scenarios were developed and run to compare potential capabilities at two future dates to analyze 26 different alternatives (Porter, 1996). Selected portions of the CASTFOREM output are analyzed and the means of the parameters are statistically grouped into mix model inputs. In the past, analyzing the grouping was labor intensive, inefficient, and required subjective interpretation of some of the data.

Methods for Separation and Grouping of Means

The Grouping Problem

Simultaneous statistical inference and multiple comparison procedures are standard methods for analyzing multiple treatment means. Fisher (1935), Tukey (1952, 1953), Scheffe (1953), and others developed multiple comparison procedures (MCP) which result in grouping similar treatment means together. These procedures have the sometimes discomforting possibility of assigning a treatment to more than one group. This group overlapping and "nonseparation" of means can be troublesome for the interpretation of results to compare the benefits of single systems to the benefits of a mix of systems working together.

The data inputs for the TRAC-WSMR Mix Model are derived from the output of CASTFOREM combat simulations. If four treatments, i.e., CASTFOREM alternatives, (T_1-T_4) are assigned to a specific group (G_1), it is desirable to provide the mix model with one single data input value for all of these four treatments. If there is no significantly discernible difference between the treatments, they should all be given the same value (μ_1). However, with group overlapping, what if one treatment (T_4) is also assigned to group G_2 that has three treatments T_4-T_6 (figure 2)? Again, reason would assign all the treatments of G_2 with value μ_2. Here is where the problem surfaces. Should T_4 that is assigned to both G_1 and G_2, be given the value μ_1, μ_2, or some other value?

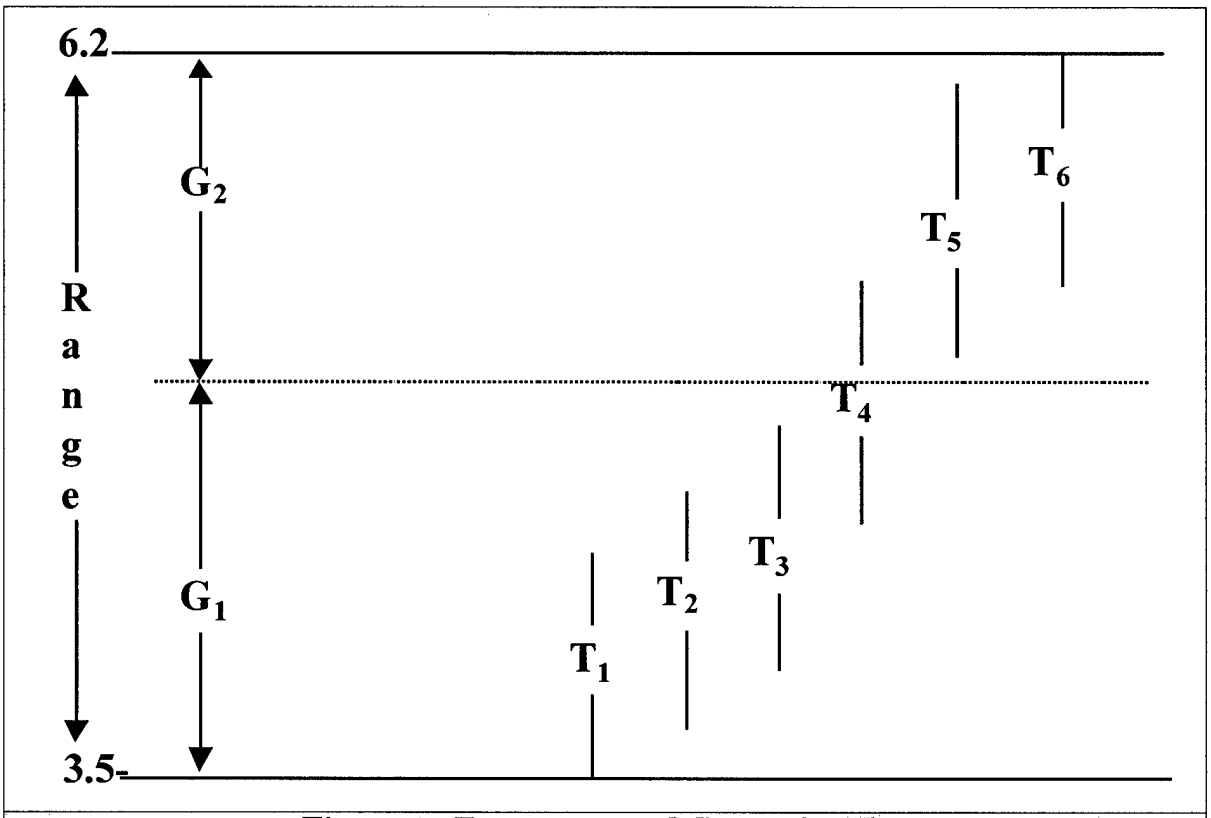


Figure 2. Treatment and Group Overlap

Tukey (1949) attempted to separate treatment means into distinct groups with his Gap Straggler Method, however, this is still an unsolved problem. Miller (1981) believes that separating treatments into families or groups "is the hardest part (of Statistical Inference) ... because it is where statistics takes leave of mathematics and must be guided by subjective judgment" (1981, p. 31). "There are no hard-and-fast rules for where family lines should be drawn, and the statistician must rely on his own judgment for the problem at hand" (Miller, 1981, p 35). For determining a valid data parameter for each mix model input, a specific value must be assigned. There can be no subjective evaluation of the treatment parameter or the consideration of other factors. A basic mix model will have more than 100 specific data parameters for every combat simulation being modeled. To increase the validity of the mix model, a nonsubjective procedure must be established.

Group overlapping is also a concern when specific parameters or factors are presented to management and decision makers. Here, the concern is not what value to assign to each group, but what are the specific and discrete groups. Though distinct groups are favorable, a combination of factors could help describe and analyze possible group overlap. Other factors could be included to help analyze the group separation and breakpoints. Also, a combination of factors could be analyzed to help determine overall treatment grouping. When considering multiple factors and overall treatment grouping, a cluster based procedure may be appropriate. Baustista, Smith, and Steiner (1996) and Conover (1996) have suggested such an approach. In this case, determining the overall treatment groupings may not be as difficult as the problem of determining valid single factor data input values.

Cluster-Nested Procedure

The Baustista, Smith, and Steiner (1996) method is a cluster-nested heuristic, which determines groupings without group overlap. Figure 3 flow charts the steps of the heuristic. Each step is described as follows:

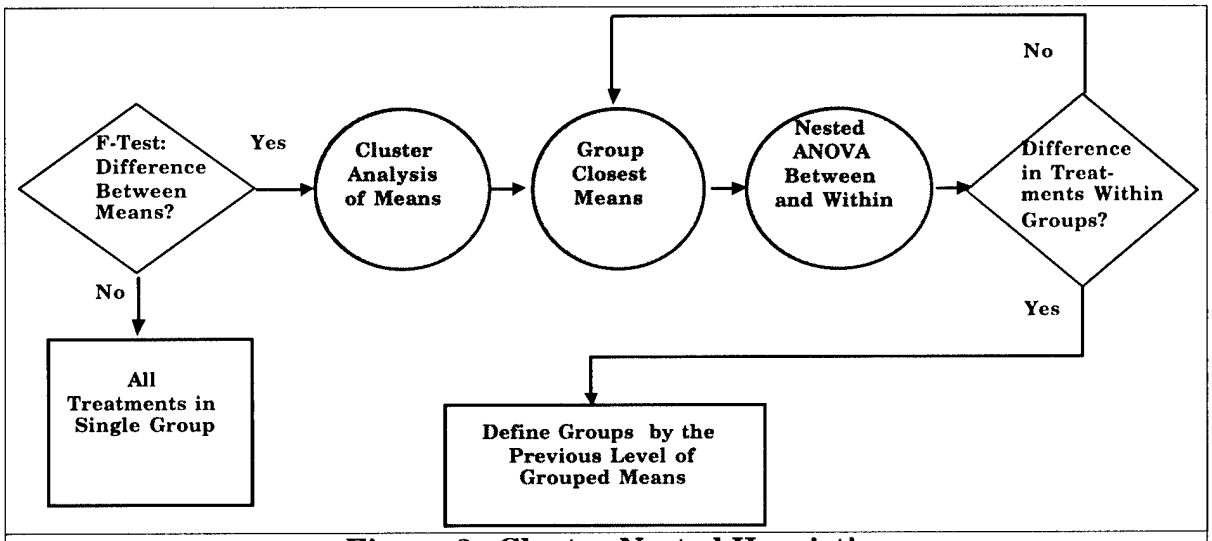


Figure 3. Cluster-Nested Heuristic

Step 1. <Decision Point> F-test to determine if there are differences between means.

Step 2a. Yes - Cluster analysis of the means.

Step 2b. No - All treatments in single group.

Step 3. Group the closest means.

Step 4. Conduct a nested analysis of variance of the between groups and within groups means.

Step 5. <Decision Point> Is there a difference among treatments within groups?

Step 6a. No - repeat steps 3 - 5.

Step 6b. Yes - define groups by the previous level of cluster-nested analysis.

Since it is neither desirable to run the mix model with overlapping groups nor take into account other outside variables, a technique for the development of specific discrete groups and treatment values will improve the process. The questions now become:

- Would a cluster-nested based approach be valid for determining overall treatment means groupings?
- Would a cluster-nested based approach be valid for determining specific single factor means (parameter, input, etc.)?
- Would such a method be more powerful than the already established MCP?
- Can the established MCP be modified and adapted into a heuristic to identify distinct groupings, and provide discrete values?
- Can MCP be modified without introducing unnecessary increased error?

US Army Conference on Applied Statistics

Gafner (1996) presented this problem and the above procedure to the US Army Conference on Applied Statistics in 1996. A panel of statistical experts provided their professional guidance and recommendations for addressing this problem. A summary of the panel discussion during the conference follows:

It was suggested that the cluster-nested based method of Baustista, Smith, and Steiner has no statistical validity and should not be considered. Since group overlapping is the statistical truth and, therefore acceptable, analysts should examine the grouping and explain the overlapping to management and decision makers. This would not be appropriate and sufficient for determining the 100(+) specific data parameter inputs of the mix model.

Another suggestion by the panel was that the grouping problem should be analyzed beyond basic hypothesis testing, i.e., MCP. A Bayesian approach or some other methods might be appropriate. Bayesian statistics have been studied in other military modeling and simulation applications (Dewitz, 1996). Its usefulness within the mix model methodology will need to be researched. Duncan (1955, 1961, 1965) and Lehmann (1957) used Bayesian approaches for MCP. However, Miller "does not consider Bayes strategies to be a practical answer, since ... almost universally impossible to specify a priori probabilities" (Miller, 1981, p 33).

Increasing the number of replications per treatment was also suggested by the panel. This may be a valid means for establishing group separation and a test could be easily conducted. However, one of the potential benefits of mix model analysis is to reduce the amount of computing time used for CASTFOREM and other combat model simulations. Given enough computing capacity and time, all possible force mixes could be run in CASTFOREM, which would eliminate the need and benefits of a mix model. Increased replications for the stochastic model would also greatly increase the overall computing time. Currently, CASTFOREM alternatives are run for 21 replications, and with more than a few alternatives, increased replications would be unwieldy.

Another concern addressed by the panel was the determination of the appropriate value for the amount of error allowed in a statistical test, α. An $\alpha = 0.05$ gives a confidence level $(1-\alpha)$ of 0.95. Historically, $\alpha = 0.05$ has been used as an initial value. Changes of the value of α may result in group shifting but may not result in defining distinct groups. Values of $\alpha = 0.10$ and $\alpha = 0.01$ can be used to determine the resulting groups. Past experience of varying α has not resulted in obtaining distinct groups.

To address the overall treatment grouping, multiple factors could be considered. During the panel discussion, Garver (1996) suggested that groupings based specifically on one factor, such as Blue Losses, are not all that meaningful by themselves. Therefore, groupings should consider a combination of other factors such as cost or Threat kills. Multiple factor comparisons appear to be an area for additional research. A cluster analysis based approach that considers multiple factors may turn out to be a helpful method for analyzing overall treatment grouping. However, for determining single factor input parameters, multiple factors would be inappropriate. The mix model itself takes all of the different input parameters and models the interaction effects of multiple factors.

Conover (1996) provided a nonstatistical insight. He suggested that if management and decision makers are interested in distinct groupings, then the exactness of the statistics is not (all) that important. If a specific number of groups has been or could be specified, then one could just conduct a cluster procedure that separates the treatments into distinct groups. A combination of Conover's and Garver's suggestion may be a valid approach for the overall treatment group separations. A multiple factor cluster analysis for a specified number of groups would be rather straight forward. If the number of

groups is not specified, the cluster-nested procedure is a possible solution for determining the number of groups.

With this research into existing techniques in mind, it was possible to move into the development of a technique for separation and grouping of means. This technique would fit the needs of processing multiple combat model output data into meaningful and statistically correct groupings for use in the TRAC-WSMR Mix Model.

Base Grouping Heuristic (BGH)

Now let us return to the mix model problem of specifying one factor input data parameters. The exact statistics may not be that important here. Due to the closeness of many means, assigning to a treatment the value of any group mean (to which it belongs) should not introduce much error into the model. There would be no additional type I error, and in agreement with Baustista, Smith, and Steiner's cluster-nested procedure, there would be slightly more type II error.

To investigate the above, the following "base grouping" heuristic procedure is proposed for determining mix model input parameter values:

Step 1. Conduct multiple comparison tests (least significant difference (LSD), Scheffe, Tukey).

Step 2. Check for group overlap. If yes, continue.

Step 3. Combine all treatments of all groups which contain the base case treatment into a single "base" group (remaining treatments are significantly different from the base case treatment) and record mean of group.

Step 4. Proceed to the first treatment that is statistically different from base case (or break point treatment) and designate that treatment as the current break point treatment.

Step 5. Repeat as per step 3, combing all treatments that are different from the base case (or previous break points) but not different from the current break point treatment.

Step 6. Repeat steps 4 and 5 until all treatments are grouped.

The BGH may introduce additional type I and II error. Members at opposite ends of the base group may be significantly different and should not have been grouped together (a type I error). Nonbase treatments may belong to overlapping groups but are not assigned to both groups (a type II error). To test this method, the possible errors introduced by using the steps above was analyzed. Questions to be answered by this analysis were: Given a specified α, how much more error is introduced and what effect could the different groupings and increased error have on the mix model final results? To address the questions, the mix model input parameters of specific Blue system kills

on Threat systems were examined. This allowed for the determination of which method would introduce less error. Data came from three CASTFOREM scenarios that had been used for many combat system analyses.

Analysis of Test Data Sets

The following is an analysis of six sets of results from CASTFOREM data. This data was used in a recent US Army study. Two data sets were selected from each scenario. Data sets 1, 2, 5, and 6 had 9 treatments, each with 21 replications (samples). Sets 3 and 4 had 10 treatments with 11 replications (samples). Complete data sets are located in appendix A. A treatment is a specific CASTFOREM alternative in which changes have been made to the base case treatment. The base case alternative should be viewed as a placebo case, i.e., the treatment that is expected without additional changes.

All of the data sets are results from significant firing system/target pairings. This is to insure that the developed grouping procedure applies to the important shooter and target pairings relating to critical study issues. Other shooter and target pairings, although important, are not considered to minimize the testing of the procedure. To help insure the validity of the recommended grouping procedure across different situations, two attributes of the data are varied. The first attribute is the resulting location of treatment 1, the base case or placebo case, mean. There are two considerations for this attribute, the value can appear at either extreme, or somewhere between the extremes. The second attribute is the inclusion of any treatments with mean value of 0.0 (lowest value). This situation can occur when the shooter of concern is not modeled in one or more of the treatments. Preceding the analysis of each of the following six data sets, these two attributes will be identified.

Each of the data sets are grouped according to the cluster-nested approach and three different multiple comparisons using the previously suggested BGH. Fisher's LSD, Scheffe's and Tukey's honest significant difference (HSD) are the three MCP used. Fisher's LSD is the most powerful, Scheffe's is the most conservative, and Tukey's HSD is considered a moderate procedure (Dowdy and Warden, 1991). Dowdy and Warden also noted that Fisher's LSD is likely to have type I errors and Scheffe's is likely to have type II errors. Two different α values (0.05 and 0.01) are used for Scheffe's and Fisher's LSD to see what effect changes in α will have on the errors and resulting groupings. $\alpha = 0.05$ is used for Tukey's procedure. Statistical Package for the Social Sciences (SPSS) is a statistical package resident on the TRAC network that provides a full range of statistics based on input values from the population (SPSS, 1983). SPSS was used to run these statistics and the cluster-nested procedure.

Data Set 1

The columns of the following tables list the treatment mean values arranged in increasing mean value order to simplify the groupings. The rows of the table list the procedures used to determine the groupings.

Data set 1 (table 1) has the attributes that treatment 1 mean (base case/placebo case) is not located at an extreme and the lowest values for this set are 0.0.

Table 1. Data Set 1

Treatment		**4**	**5**	**9**	**3**	**8**	**1**	**6**	**2**	**7**
Treatment Mean		**0.0**	**0.0**	**7.5**	**9.3**	**11.2**	**11.7**	**11.8**	**12.2**	**16.0**
Procedure	α	Groupings								
Cluster/Nested	NA	1	1	2	2	2	2	2	2	3
LSD	0.05	1	1	2	2	3	3	3	3	4
LSD*	0.01	1	1	2	3	3	3	3	3	4
Scheffe*	0.05	1	1	2	3	3	3	3	3	4
Scheffe	0.01	1	1	2	2	2	2	2	2	2
Tukey (HSD)*	0.05	1	1	2	3	3	3	3	3	4

**Grouping procedures give same results*
Overall - Range: 0.0-24.0, Mean: 8.85, Standard Deviation: 6.05

Analysis of Results for Data Set 1: Scheffe (0.05), LSD (0.01) and Tukey give the same resulting groupings. This grouping can not be reached by any step of the cluster-nested procedure. One additional step in the cluster-nested procedure would give the same result as Scheffe (0.01). The previous step in the cluster-nested procedure gave the same result as LSD (0.05). All solutions result in treatments 4 and 5 (value 0.0) being in one group that is different from all other group. Scheffe (0.01) results in an extremely conservative grouping with treatment 4 and 5 grouped together and every other treatment in a second group. This grouping is likely to have a type II error. Excluding this grouping, all the other groupings place treatment 7 in a group by itself.

The "final grouping" is the grouping breakdown that should be used for the mix model analysis. After analysis of all of the suggested groupings, the final grouping is determined to be the best breakdown.

Table 2. Data Set 1 Final Grouping: (Group 3 is Base Group)

Treatment	4	5	9	3	8	1	6	2	7
Group	1	1	2	3	3	3	3	3	4
Group Mean	0.0	0.0	7.5	11.2	11.2	11.2	11.2	11.2	16.0

This grouping is the same as Scheffe (0.05), LSD (0.01), and Tukey. There is possible error of placement for treatment 3 in group 2 or in group 3. If treatment 3 is placed in group 2 instead of 3, the mean of group 2 would be 8.4 and of group 3 will be 11.7. Although the differences between groups change slightly, the only change that might cause a difference is the value assigned to treatment 3. Treatment 3 would be different from the base case, treatment 1, and its value would decrease from 11.2 to 8.4.

Data Set 2

Data set 2 has the attributes that: the lowest value(s) for this set are 0.0 and treatment 1 mean (base case/placebo case) is located at an extreme.

Table 3. Data Set 2										
Treatment		**5**	**3**	**2**	**8**	**9**	**7**	**4**	**6**	**1**
Treatment Mean		**0.0**	**0.0**	**0.0**	**57.0**	**62.2**	**74.1**	**75.3**	**83.1**	**91.0**
Procedure	α	Groupings								
Cluster/Nested	NA	1	1	1	2	2	3	3	4	5
LSD	0.05	1	1	1	2	2	3	3	4	5
LSD*	0.01	1	1	1	2	2	3	3	4	4
Scheffe	0.05	1	1	1	2	3	3	3	4	4
Scheffe	0.01	1	1	1	2	2	2	3	3	3
Tukey (HSD)*	0.05	1	1	1	2	2	3	3	4	4

**Grouping procedures give same results*
Overall - Range: 0.126, Mean: 49.2, Standard Deviation: 38.0

Analysis of Results for Data Set 2: LSD (0.05) and the cluster-nested procedure have the same resulting grouping but may be too powerful, resulting in a possible type II error. LSD (0.01), Tukey, and the cluster-nested with one additional step give the same result. All solutions result in treatments 5, 3, and 2 (value 0.0) being in one group and treatment 8 being in another group. Scheffe (0.01) results in an extremely conservative grouping and is likely to have a type I error. Except for Scheffe (0.05), all the other groupings result with treatment 9 being in a group with treatment 8. The final decision is where to place treatment 6. The 95 percent confidence range for treatment 6 is 74.8-91.4, for treatment 1 is 84.7-97.4 and for treatment 4 is 68.3-79.9. Comparing these ranges, 86 percent of the range of treatment 6 overlaps the range of treatment 1 (11.8/16.6). Only 30 percent of the range of treatment 6 overlaps the range of treatment 4 (5.1/16.6). Therefore, as in three of the procedures, treatment 6 should be grouped with treatment 1.

Table 4. Final Grouping: (Group 4 is Base Group)									
Treatment	5	3	2	8	9	7	4	6	1
Group	1	1	1	2	2	3	3	4	4
Group Mean	0.0	0.0	0.0	59.6	59.6	74.7	74.7	87.1	87.1

If treatment 6 is placed in group 3, the mean of group 3 changes to 77.5 and of group 4 changes to 91. The deltas between groups would not change significantly; however, the value assigned to treatment 6 would decrease by 9.5

Data Set 3

Data set 3 has the attributes that the lowest value(s) for this set are not 0.0 and the treatment 1 mean (base case/placebo case) is not located at an extreme. Also, this data set has a tight range of values. With the exception of treatment 6, all the other values are within 1.7 of each other.

Table 5. Data Set 3											
Treatment		**6**	**8**	**3**	**10**	**1**	**4**	**9**	**5**	**7**	**2**
Treatment Mean		**5.2**	**8.5**	**8.8**	**9.0**	**9.1**	**9.2**	**9.4**	**9.8**	**10. 0**	**10.2**
Procedures	α	Groupings									
Cluster/Nested	NA	1	2	2	2	2	2	2	3	3	3
LSD*	0.05	1	2	2	2	2	2	2	2	2	2
LSD*	0.01	1	2	2	2	2	2	2	2	2	2
Scheffe*	0.05	1	2	2	2	2	2	2	2	2	2
Scheffe	0.01	1	2	2	2	2	2	2	2	2	2
Tukey (HSD)*	0.05	1	2	2	2	2	2	2	2	2	2

**Grouping procedures give same results*
Overall - Range: 2-14, Mean: 8.92, Standard Deviation: 2.14

Analysis of the Results from Data Set 3: With one additional step, the cluster-nested procedure gives the same result as all of the other procedures. With treatment 1 located in the middle, this makes sense. We are trying to determine what groups are different from the base (treatment 1). At the extremes, treatment 2 with mean of 10.2 could be different from treatment 8 with mean of 8.5. However, neither are much different from treatment 1 mean of 9.1. Clearly, treatment 6 mean of 5.2 is different and should be in a group by itself. This was the only case where all of the MCP groupings had the same result. Only the cluster-nested procedure was different and it reached this result with one additional step.

Table 6. Final Grouping: (Group 2 is Base Group)										
Treatment	7	8	3	10	1	4	9	5	7	2
Group	1	2	2	2	2	2	2	2	2	2
Group Mean	5.2	9.3	9.3	9.3	9.3	9.3	9.3	9.3	9.3	9.3

Data Set 4

Data set 4 has the attributes that the lowest value(s) for this set are 0.0, and treatment 1 mean (base case/placebo case) is located at an extreme.

Table 7. Data Set 4											
Treatment		**5**	**3**	**2**	**8**	**4**	**9**	**7**	**10**	**6**	**1**
Treatment Mean		**0.0**	**0.0**	**0.0**	**6.3**	**8.6**	**8.9**	**10.3**	**10.4**	**11.0**	**11.4**
Procedures	α	Groupings									
Cluster/Nested	NA	1	1	1	2	3	3	4	4	5	5
LSD*	0.05	1	1	1	2	3	3	4	4	4	4
LSD*	0.01	1	1	1	2	3	3	4	4	4	4
Scheffe	0.05	1	1	1	2	3	3	3	3	3	3
Scheffe	0.01	1	1	1	2	3	3	3	3	3	3
Tukey (HSD)*	0.05	1	1	1	2	3	3	4	4	4	4

**Grouping procedures give same results*
Overall - Range: 0-13, Mean: 6.68, Standard Deviation: 4.86

Analysis of Results from Data Set 4: Both LSDs, Tukey, and the cluster-nested procedure with one additional step give the same result. Both Scheffes give the same result but may be too conservative. The 95 percent confidence range of treatment 1 is

10.4-12.3, of 9 is 7.6-10.2, and of 8 is 4.8-7.8. Treatment 9 has no overlap with treatment 1 and an overlap of only 0.2 with treatment 8. Therefore, as suggested by three of the groupings, treatment 9 should be grouped separate from treatments 1 and 8.

Table 8. Data Set 4 Final Grouping: (Group 4 is Base Group)

Treatment	5	3	2	8	4	9	7	10	6	1
Group	1	1	1	2	3	3	4	4	4	4
Group Mean	0.0	0.0	0.0	6.3	8.8	8.8	10.8	10.8	10.8	10.8

Data Set 5

Data set 5 has the attributes that the lowest value(s) for this set are not 0.0 and treatment 1 mean (base case/placebo case) is not located at an extreme.

Table 9. Data Set 5

Treatment		**4**	**5**	**9**	**8**	**6**	**7**	**2**	**1**	**3**
Treatment Mean		**1.8**	**2.0**	**2.6**	**3.0**	**3.5**	**3.6**	**3.8**	**4.1**	**4.9**
Procedures	α	Groupings								
Cluster/Nested	NA	1	1	2	3	3	3	3	4	5
LSD	0.05	1	2	2	3	3	4	4	4	5
LSD*	0.01	1	1	2	3	3	3	3	3	3
Scheffe	0.05	1	1	1	2	2	2	2	2	2
Scheffe	0.01	1	1	1	2	2	2	2	2	2
Tukey (HSD)*	0.05	1	1	2	3	3	3	3	3	3

**Grouping procedures give same results*
Overall - Range: 0-7, Mean: 3.7, Standard Deviation: 1.37

Analysis of Results from Data Set 5: LSD (0.01), Tukey, and two additional steps of cluster-nested procedure result in the same solution. The cluster-nested and LSD (0.05) procedures seem too powerful with five resulting groups. Both Scheffes may be too conservative with only two groupings. Comparing the 95 percent confidence ranges, we see the range of treatment 1 is 3.68-4.60 and of treatment 8 is 2.50-3.6. With no overlap between the treatments, these treatments should be in separate groups.

Table 10. Final Grouping: (Group 3 is Base Group)

Treatment	4	5	9	8	6	7	2	1	3
Group	1	1	2	2	3	3	3	3	3
Group Mean	1.9	1.9	2.8	2.8	4.0	4.0	4.0	4.0	4.0

Data Set 6

Data set 6 has the attributes that the lowest value(s) for this set are not 0.0 and treatment 1 mean (base case/placebo case) is not located at an extreme.

Table 11. Data Set 6										
Treatment		**5**	**2**	**3**	**4**	**6**	**7**	**8**	**1**	**9**
Treatment Mean		**17.9**	**22.4**	**22.6**	**23.2**	**24.7**	**25.4**	**26.2**	**26.9**	**28.1**
Procedures	α	Groupings								
Cluster/Nested	NA	1	2	2	2	2	2	2	2	2
LSD*	0.05	1	2	2	2	3	3	3	3	3
LSD*	0.01	1	2	3	3	3	3	3	3	3
Scheffe*	0.05	1	2	2	2	2	2	2	2	2
Scheffe	0.01	1	2	2	2	2	2	2	2	2
Tukey (HSD)*	0.05	1	2	2	2	2	2	2	2	2

**Grouping procedures give same results*
Overall - Range: 3-40, Mean: 24.1, Standard Deviation: 6.08

Analysis of the Results from Data Set 6: Both Scheffes, Tukey, and the cluster-nested procedures give the same result with only two groups. Both LSDs have three groups, but break the groups at different locations. Considering the 95 percent confidence range of treatment 1 is 24.7-29.0 along with the range of treatment 4 of 20.5-25.8, we can see that there is very little overlap. Therefore, there is little chance of the treatments being the same. LSD (0.05) would match the previous step of the cluster-nested procedure.

Table 12. Data Set 6 Final Groups: (Group 3 is Base Group)									
Treatment	5	2	3	4	6	7	8	1	9
Group	1	2	2	2	3	3	3	3	3
Group Mean	17.9	22.7	22.7	22.7	26.3	26.3	26.3	26.3	26.3

Summary of Analysis of Test Data Sets

Reviewing the six sets of data, Tukey and Fisher LSD (0.01) procedures were the best of those used in the BHG. They identified the recommended grouping five of the six times. Data set 6 was the only situation for which these procedures did not identify the final grouping. At the extremes, neither a very powerful procedure (LSD 0.05) nor a conservative procedure (Scheffe) performed nearly as well. These procedures identified the final grouping three or less times.

The nested-cluster procedure seems to produce fairly reliable results and is usually within one step or two of the recommended groupings. The only situation in which the cluster-nested did not come close was in data set 1. Within this data set, treatment 1 was in the center and there were groups at both ends that did not include treatment 1. Further analysis of additional data sets would help to see if this trend holds.

With the Tukey and LSD (0.01) procedures giving the best results, it is necessary to examine the one case in which these procedures did not result in the final grouping. Data set 6 had a relatively small standard deviation (6.08) compared to the range (3-40) and mean (24.1). Although the overall range (3-40) was fairly wide, the 95 percent confidence range for most of the treatments was much narrower. A review of the raw data provides the information that treatment 5 has by far the lowest value (3) of the

values in the data set. This one extremely low value might be suspect. However, the next closest value is still small (10) and is in the same treatment. Again, further analysis of additional data sets may help.

In conclusion, to establish group separation use a moderate multiple comparison such as Tukey's HSD, or use the powerful LSD procedure with a smaller alpha (0.01) with the BGH procedure. Since the cluster-nested procedure did not perform as well as the BGH with the above MCP, the cluster-nested procedure was discarded as an appropriate method for analyzing mix analysis data.

Data Reduction and Manipulation

For each specific use, the TRAC-WSMR Mix Model requires some modifications. For example, the model constraints and the objective function may change for each application. Such changes require detailed review to incorporate the goals of the model user. Upon completion of the CASTFOREM simulations, a quick turn around time from the mix model is desired to meet study objectives. Before the mix model runs may begin, the required input data must first be preprocessed from the results of CASTFOREM battle simulations. Data preprocessing consumes a large percentage of the total time required to obtain mix model results. This section presents the research and development of procedures to expedite the data preprocessing.

With the establishment of the BGH as the recommended treatment/group separation procedure, the development of a practical detailed algorithm could proceed. When completed, the algorithm was programmed into a computer program that provides quick execution for efficiency in processing model results. The data preprocessing routine transforms CASTFOREM output data into the required model input formats.

Preprocessing the data from the combat simulations consists of five major data manipulation steps:

Step 1. Extraction of CASTFOREM output data.

Step 2. Statistical analysis of the data.

Step 3. Grouping of data.

Step 4. Generation of base input data tables.

- A complete base interaction m x n matrix for all resources (m) and tasks (n) (friendly *m* system kills of enemy *n* systems, a *friendly killer victim scoreboard*).
- A base resource expenditure table (friendly *m* systems lost).

Step 5. Generation of exceptions to base data table/matrix and concatenating exceptions from each scenario.

The development of simple UNIX based programming scripts helped improve step one. These scripts may have the possibility of additional development and improvement. Further development would continue past step one and execute the SPSS statistical program to start step two. Before proceeding to step three, the only additional step two requirement is a manual analysis of the SPSS output. This analysis is a quick review and check of the data for any noticeable errors and to get an overall understanding of significant trends and insights. Insights and any problems are recorded and forwarded to the appropriate combat battle analysts. The output of the SPSS program provides formatted data for step four.

The development of a standard grouping algorithm was a preliminary step for streamlining the data preprocessing. As an algorithm, the BGH procedure described in the previous section expedites the development of a high-level language program to perform this task. As a heuristic and as did hand generation of data groupings, the algorithm may produce some errors. Besides the potential time savings, especially with large amounts of data or under the pressure of time, the algorithm is more reliable than manual processing. The algorithm also provides for standardized manipulation of the data sets. The grouping algorithm picks up at step three and continues through step five.

As output from step two, each data parameter in the SPSS output is listed and followed with a lower left triangle (LLT) format table. The LLT shows which treatments produce statistically significant differences for the variable of concern. As an example:

Table 13. Example SPSS Output (Data Set 1 With LSD .01 or Tukey's)

Mean	Treatment	Treatment 4	Treatment 5	Treatment 9	Treatment 3	Treatment 8	Treatment 1	Treatment 6	Treatment 2	Treatment 7
0.00	4									
0.00	5									
7.48	9	*	*							
9.33	3	*	*							
11.24	8	*	*	*						
11.67	1	*	*	*						
11.76	6	*	*	*						
12.19	2	*	*	*	*					
16.00	7	*	*	*	*	*	*	*	*	

**Indicates significant difference*

Reading down the LLT, treatments 4 and 5 are not different from any prior treatment. Treatments 9 and 3 are different from the first two (4 and 5). Treatments 8, 1, and 6 are different from the first three (4, 5, 9), and treatment 2 is additionally different from the fourth treatment (3). Finally, treatment 7 is different from all of the other eight treatments.

Now let us step through the BGH algorithm described in the previous section. Starting with treatment 1, the base (placebo), is different from the first three

treatments. Therefore, treatments 4, 5, and 9 are excluded from the base group. Below treatment 1, treatment 7 is the first and only treatment different from treatment 1 (as the number of stars are greater than or equal to the position of treatment 1). So treatment 7 is excluded from the base group. Treatment 7 can also be placed in its own separate group. The base group now contains five treatments (3, 8, 1, 6, 2). Although the LLT shows that treatment 2 is different from treatment 3, they are grouped together since neither are different from the base case treatment 1. Going back up the LLT, since treatment 1 was different from the first three, we move to treatment 9. Treatment 9 is different from the first two treatments and is in a group by itself. Moving to treatment 5, at the position of the number that is different from treatment 9, there are no differences shown. With no additional differences, all the treatments from treatment 5 to the first treatment are in the one final group. Here, just treatment 5 and treatment 4 are grouped together.

Having grouped all the treatments, we record the mean of the variable for all the treatments within each group. The base group is labeled as "base" and the different treatments within the base group are no longer a concern. For this example, the average of the five "base" alternative means:

$$(9.33 + 11.24 + 11.67 + 11.76 + 12.19)/5 = 11.24$$

is assigned to base variable V_{mn} in the base matrix. For many variables in the complete M x N matrix, no treatments are different from the base. These variables usually correspond to minor players within the different battles. After completing this algorithm for all the base variable values, the base m x n matrix is written out to a file. The same basic algorithm is also applied to resource expenditures, i.e., friendly systems lost.

The treatments/groups that are different from the base group are addressed next. The treatments that compose groups that are different from the base are recorded in a separate file and the mean value of the group is assigned. For this example, treatment 7 has a value of 16.0, treatment 9 has a value of 7.48 and treatments 4 and 5 are assigned the mean value of 0.0.

Four parameters are required for each adjusted variable. These are recorded as: For Battle *B,* if treatment *t*, system *m*, by system/task *n*, updated V_{mn}. The above data set example would result in four sets or lines of data in the "adjustments" file (table 14).

Table 14. Adjustments File

B	t	m	n	V_{mn}
B1	4	13	25	0.0
B1	5	13	25	0.0
B1	7	13	25	16.0
B1	9	13	25	7.48

For Battle = B1, m = 13 n = 25

This algorithm is executed for each m x n variable (V_{mn}) in the battle. The algorithm is then repeated for each CASTFOREM battle being considered within the overall scenario used in the study.

Once the algorithm was completely developed, an efficient program to execute this algorithm was developed. This SPSS/MM program is written in the C language. The SPSS/MM program has 500(+) lines of codes and required an expenditure of approximately 100 staff hours of work (appendix B). The program provided the overall structure and incorporates probably 90 percent of the SPSS/MM program. Compiled and run on a Hewlett Packard HP-700 UNIX workstation, the generation of more correct and properly formatted mix model input data is now almost instantaneous.

For future mix model analyses, this program alone will be a significant contribution. Besides the elimination of possible human induced errors, this program will save at least 10 staff hours of data intensive work per combat simulation battle. With only two mix model analyses consisting of five battles each, the break-even point for the research and development of the program will be reached quickly.

Conclusions

For improved efficiency, the TRAC-WSMR Mix Model requires automated data processing and specific nonsubjective input data. These data often come from statistical overlapping groups. This research recommends using Tukey's honest significant difference multiple comparison procedure or Fisher's least significant difference with $\alpha = 0.01$ for the base grouping heuristic for the analysis of Mix Model input data. Based on the data sets analyzed in this research, automated processing of input data combined with these procedures will not decrease the accuracy of the input data. This in turn will increase the accuracy, validity, and efficiency of TRAC-WSMR Mix Model.

Appendix A. CASTFOREM Data Sets

This appendix lists the complete data sets described in the main body of this report.

Data Set 1									
Alts: Rep/Trt	**0 Trt1**	**1 Trt2**	**2 Trt3**	**3 Trt4**	**4 Trt5**	**8 Trt6**	**9 Trt7**	**10 Trt8**	**11 Trt9**
1	14	14	7	0	0	10	11	11	9
2	14	14	9	0	0	6	18	16	9
3	13	9	12	0	0	14	18	11	4
4	15	19	9	0	0	11	18	11	7
5	8	14	18	0	0	21	24	13	8
6	10	9	14	0	0	11	21	14	8
7	13	6	4	0	0	11	8	13	4
8	12	10	15	0	0	13	19	10	10
9	17	19	13	0	0	11	24	7	5
10	12	8	11	0	0	14	16	10	4
11	9	21	9	0	0	9	11	16	6
12	8	19	9	0	0	12	14	10	6
13	12	7	11	0	0	16	14	13	8
14	9	9	6	0	0	9	12	14	8
15	17	9	7	0	0	18	18	12	5
16	9	12	6	0	0	8	18	10	9
17	7	6	6	0	0	7	13	9	7
18	11	10	4	0	0	13	18	11	9
19	11	14	9	0	0	14	16	8	9
20	14	15	11	0	0	13	14	8	6
21	10	12	6	0	0	6	11	9	16
Mean	11.67	12.19	9.33	0.00	0.00	11.76	16.00	11.24	7.48

Alts - alternatives *Rep - repetition* *Trt - treatment*

Data Set 2									
Alts: Rep/Trt	**0 Trt1**	**1 Trt2**	**2 Trt3**	**3 Trt4**	**4 Trt5**	**8 Trt6**	**9 Trt7**	**10 Trt8**	**14 Trt9**
1	105	0	0	65	0	54	59	36	27
2	88	0	0	65	0	98	84	51	39
3	81	0	0	65	0	68	99	86	65
4	69	0	0	100	0	92	70	71	75
5	90	0	0	63	0	93	78	62	58
6	89	0	0	95	0	77	93	70	51
7	93	0	0	74	0	74	73	64	70
8	86	0	0	79	0	118	63	49	79
9	95	0	0	73	0	90	62	71	59
10	95	0	0	74	0	75	53	78	49
11	126	0	0	66	0	96	66	54	78
12	105	0	0	59	0	57	63	65	31
13	82	0	0	79	0	48	82	66	40
14	68	0	0	78	0	80	46	75	43
15	101	0	0	81	0	87	96	55	71
16	89	0	0	73	0	91	108	79	50
17	93	0	0	73	0	109	83	41	67
18	81	0	0	96	0	81	79	47	35
19	71	0	0	45	0	98	69	79	62
20	95	0	0	79	0	98	75	68	73
21	110	0	0	74	0	62	80	39	76
Mean	91.05	0.00	0.00	74.10	0.00	83.14	75.29	62.19	57.05

Data Set 3										
Alts: Rep/Trt	**0 Trt1**	**1 Trt2**	**2 Trt3**	**3 Trt4**	**4 Trt5**	**8 Trt6**	**9 Trt7**	**10 Trt8**	**11 Trt9**	**14 Trt10**
1	9	9	8	8	10	2	9	9	9	9
2	8	9	10	8	10	3	9	12	9	10
3	8	11	9	11	9	6	10	10	8	8
4	14	11	10	12	11	6	12	6	9	7
5	9	9	6	9	10	8	9	6	10	8
6	10	11	10	11	10	5	11	10	11	10
7	8	8	9	9	11	3	10	10	10	12
8	6	9	9	6	8	7	9	9	8	11
9	8	10	8	6	8	4	9	7	8	9
10	8	11	9	11	8	8	9	9	9	8
11	12	14	9	10	13	5	13	6	12	7
Mean	9.09	10.18	8.82	9.18	9.82	5.18	10.00	8.55	9.36	9.00

Data Set 4										
Alts: Rep/Trt	**0 Trt1**	**1 Trt2**	**2 Trt3**	**3 Trt4**	**4 Trt5**	**8 Trt6**	**9 Trt7**	**10 Trt8**	**11 Trt9**	**14 Trt10**
1	12	0	0	8	0	11	13	5	12	8
2	10	0	0	11	0	8	12	7	11	9
3	13	0	0	7	0	13	12	11	10	12
4	12	0	0	8	0	10	12	7	10	12
5	10	0	0	6	0	12	10	5	10	10
6	9	0	0	6	0	13	8	6	9	6
7	12	0	0	10	0	11	11	2	9	10
8	12	0	0	9	0	11	9	6	8	13
9	10	0	0	11	0	9	5	8	7	12
10	12	0	0	11	0	11	11	7	6	11
11	13	0	0	8	0	12	10	5	6	11
Mean	11.36	0.00	0.00	8.64	0.00	11.00	10.27	6.27	8.91	10.36

Data Set 5									
Alts: Rep/Trt	**0 Trt1**	**1 Trt2**	**2 Trt3**	**3 Trt4**	**4 Trt5**	**5 Trt6**	**6 Trt7**	**7 Trt8**	**8 Trt9**
1	3	4	5	4	2	4	4	3	4
2	3	5	3	2	1	5	4	3	4
3	4	4	5	4	2	3	2	2	3
4	5	5	5	2	3	5	2	4	1
5	6	3	5	2	2	2	3	2	1
6	3	3	7	1	1	3	3	3	2
7	4	3	4	2	3	5	3	2	2
8	4	4	5	3	2	3	4	3	2
9	4	3	4	2	2	3	4	3	3
10	5	4	5	2	2	1	3	0	3
11	3	4	5	2	3	4	3	4	2
12	4	4	5	2	3	3	3	3	3
13	4	4	5	2	2	3	4	2	3
14	2	5	4	4	2	4	3	4	3
15	6	4	6	1	2	3	2	5	4
16	5	3	5	2	1	4	5	2	2
17	4	2	4	1	1	3	4	2	2
18	4	4	4	1	1	6	5	5	3
19	5	4	6	2	1	4	5	4	3
20	5	5	5	2	0	3	3	4	3
21	4	3	6	0	1	2	7	4	2
Mean	4.14	3.81	4.90	2.05	1.76	3.48	3.62	3.05	2.62

Data Set 6									
Alts: Rep/Trt	**0 Trt1**	**1 Trt2**	**2 Trt3**	**3 Trt4**	**4 Trt5**	**8 Trt6**	**9 Trt7**	**10 Trt8**	**11 Trt9**
1	29	22	29	15	13	26	28	29	27
2	34	26	21	26	23	26	25	26	28
3	22	31	22	30	17	16	21	27	23
4	23	18	22	22	26	28	29	30	35
5	26	24	23	21	15	27	19	30	28
6	25	26	27	28	3	23	27	30	23
7	29	21	25	27	15	25	17	26	28
8	39	12	18	20	22	33	21	20	29
9	27	25	20	25	13	24	21	25	31
10	31	16	28	28	14	26	24	34	21
11	30	14	32	37	21	22	40	27	33
12	29	20	11	30	21	28	29	27	33
13	25	15	27	20	17	23	29	25	24
14	33	19	15	25	19	24	28	23	29
15	24	33	14	24	11	24	20	17	28
16	20	20	15	19	22	32	28	24	32
17	23	15	30	18	29	11	22	27	30
18	23	28	32	24	17	30	29	35	25
19	26	23	29	19	26	20	21	24	27
20	26	27	12	16	21	26	24	23	36
21	20	36	22	13	10	24	32	21	21
Mean	26.86	22.43	22.57	23.19	17.86	24.67	25.43	26.19	28.14

Appendix B. SPSS - Mix Model Algorithm Code

```
/*    SPSS/MM program     SPSS to Mix Model Program
*
*  kvsb.c: Killer Victim Score Board generation program
*          This program reads a SPSS generated ONEWAY output file
*          and summarizes the significance information into a base KV
*          Scoreboard for use in the TRAC-WSMR Mix Model.
*
*  Written by
*  Bruce Gafner & Doug Shoop
*  U.S. Army TRADOC Analysis Command, WSMR, NM
*
*
*/

#include <stdio.h>
#include <string.h>
#include "kvsb.h"

char buf[BUFSIZE];              /* General purpose buffer */
char line[MAXLENGTH_IN];         /* Line buffer */

/****************/
/***** main *****/
/****************/

main(argc, argv)
int argc;
char *argv[];
{

  FILE          *infile;                  /* Input file */
  FILE          *outfile;                 /* Output file for base KVSB */
  FILE          *outfile2;                /* Output file for differences*/
  char          *ptr;                     /* Gen purpose char pointer */
  int           abort = FALSE;            /* Abort flag */
  float         basegroupmean;            /* Mean for group containing the
                                             BASE alt */
  int           basegroupnum;             /* Group number of the group
                                             containing the BASE alt */
  char          firer[MAXNAME_LENGTH];    /* Firer name */
  int           found = FALSE;            /* Found beginning of data flag */
  char          infilename[255];          /* Input file name */
  char          killnum[MAXNAME_LENGTH]; /* Kill number */
  LLTLINE       lltarray[MAXALTS];        /* Lower Left Triangle array */
  int           numalts;                  /* Number of alts */
  char          outfilename[255];         /* Output file name */
  char          outfilename2[255];        /* Output file name */
  char          progname[31];             /* Program name string */
  int           status = TRUE;            /* Status of a function call */
```

```
char          target[MAXNAME_LENGTH];  /* Target name */
float         totalmean;               /* Total mean for all alts */
KVSB          kvsb[MAXPAIRS];          /* Base KV score board */
int           kvsb_itemcnt;            /* Count of Firer to target
                                          pairs */
ALT_KVSB      alt_kv[MAXPAIRS];    /* Alternative different than the
                                      base kvsb */
int           alt_kv_itemcnt;      /* Count to case different from base
                                      kvsb*/
extern char   buf[BUFSIZE];        /* General purpose buffer */
extern char   line[MAXLENGTH_IN]; /* Line buffer */

void         echo_data();          /*Function to echo data read from
                                     file */
int          get_llt_data();       /*Function to retrieve data from file*/
void         proc_diff();          /* Process data that are different */
void         proc_nodiff();        /* Process data that are not
                                      different*/
void         write_kvsb();         /* Procss to write to base kvsb */
void         write_alt_kv();       /* Procss to write to alt kvsb */

/* Save the program name */
if( ptr = strrchr(argv[0],'/') ) strcpy(progname,ptr+1);
else strcpy(progname,argv[0]);

/* Check the command line args */
if(argc != 3) {
  fprintf(stderr,"\nUsage:  %s  input_data_file
    output_file\n\n",progname);
  exit(1);
}
else {
  /* Save the arguments */
  strcpy(infilename,   argv[1]);
  strcpy(outfilename,  argv[2]);
  strcpy(outfilename2, argv[2]);
  strcat(outfilename2, "_diff");

} /* End else */

/* Check if input file exists and is readable */
infile = fopen(infilename, "r");
if(infile == NULL) {
  sprintf(buf," Unable to open %s",infilename);
  perror(buf);
  exit(1);
}

/* Check if output file is writeable */
outfile = fopen(outfilename, "w");
if(outfile == NULL) {
  sprintf(buf," Unable to open %s",outfilename);
  perror(buf);
  exit(1);
}
```

```
/* Check if output file is writeable */
outfile2 = fopen(outfilename, "w");
if(outfile2 == NULL) {
  sprintf(buf," Unable to open %s",outfilename2);
  perror(buf);
  exit(1);
}

kvsb_itemcnt = 0;       /* Init the to target pair index */
alt_kv_itemcnt  = 0;    /* Init the index for alternatives different
                           than base case */
/* Position the input file to the beginning of data section */
found = FALSE;                      /* while records in file */
while(fgets(line, MAXLENGTH_IN, infile) != NULL) {
  if(!strstr(line,"- - O N E W A Y - -")) continue;
  found = TRUE;
  break;
}
/* If beginning of data section was not found,
    let user know and exit */
if(!found) { fprintf(stderr,"\nERROR:  ONEWAY data section not
                found.  Check that your\n");
  fprintf(stderr,"        input file is the correct one.\n\n");
  exit(1);
}

/* Process the input file */
while(status = get_llt_data(infile,killnum,firer,target,lltarray,
    &totalmean,&numalts)) {
  /* Init the base group mean and number */
  basegroupmean = -9999.0;
  basegroupnum  = -1;
  /* Perform operations on the data according to status */
  switch(status) {
      case DIFFERENT:
                    /* Put your difference code here */

                      proc_diff(killnum,firer,target,lltarray,
                      totalmean,numalts,&basegroupmean,
                      &basegroupnum,kvsb,kvsb_itemcnt,alt_kv,
                      &alt_kv_itemcnt);
                      kvsb_itemcnt++;
                           break;
      case NO_DIFFERENCE:

                      proc_nodiff(firer,target,totalmean,
                      kvsb,kvsb_itemcnt);
                      kvsb_itemcnt++;
                         break;
      case NO_DATA:
                           break;
      case READ_ERROR:
                           fprintf(stderr,"Corrupt file.
```

```
                                Repair and rerun program.\n");
                               fprintf(stderr,"A good place to start
                                looking is at or after:\n");
                               fprintf(stderr,"Variable
                                %-10.10s",killnum);
                               if(strlen(firer)) fprintf(stderr,"
                                %-8.8s versus %s\n",firer,target);
                               fprintf(stderr,"\n\n");
                               abort = TRUE;
                               break;
        case FINISHED:
                               break;
    } /* End switch */
    /* If we need to abort, break out of processing loop */
    if(abort) break;

  } /* End while status */

  /* write kvsb to a file or to stdout */
  write_kvsb(kvsb,kvsb_itemcnt,argc,outfilename);

  /* write the alternatives taht are  different than base */
  write_alt_kv(alt_kv,alt_kv_itemcnt,argc,fopen(outfilename2,"w"));

  /* Close the files */
  fclose(infile);
  fclose(outfile);
  fclose(outfile2);

} /* End main */



/************************/
/***** get_llt_data *****/
/************************/

int
get_llt_data(infile,killnum,firer,target,lltarray,totalmean,numalts)
FILE    *infile;
char    killnum[];
char    firer[];
char    target[];
LLTLINE lltarray[];
float   *totalmean;
int     *numalts;
{

  char  *ptr;                       /* Gen purpose char pointer */
  char  *stat;                      /* Status of an fgets */
  int   alt;                        /* Alternative index */
  float fprob;                      /* F Probability */
  int   found;                      /* Found flag */
```

```
  float fratio;                          /* F Ratio */
  int   i,j;                             /* Gen purpose indexes */

  extern char buf[BUFSIZE];              /* General purpose buffer */
  extern char line[MAXLENGTH_IN];        /* Line buffer */

  /* Init the data */
  strcpy(killnum,"");
  strcpy(firer,"");
  strcpy(target,"");
  for(i = 0; i < MAXALTS; i++) {
    lltarray[i].mean = 0.0;
    strcpy(lltarray[i].alt,"");
    lltarray[i].num_diff = 0;
    lltarray[i].group = 0;
    for(j =0; j < MAXALTS; j++) lltarray[i].difflag[j] = FALSE;
  }
  *totalmean = 0.0;
  *numalts = 0;

  /* Get the KILLNUM, FIRER, and TARGET */
  found = FALSE;
  while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) { /* while
records in file */
    if(!strstr(line," Variable  ")) continue;
    sscanf(line," Variable %s %s versus %s",killnum,firer,target);
    found = TRUE;
    /* break apart killnum into filler and target */
    if (!strlen(firer)) {
      /* firer = killnum until reach an "_" */
      /* target will be _+1 */
      i = 0;
      while (killnum[i] != '_') {
         firer[i] = killnum[i];
         i++;
      }
firer[i] = '_';
firer[i+1] = '\0';
        i++;
      strcpy(target,&killnum[i]);
    }
    break;
  }
  /* If EOF or error encountered return finished status */
  if(!stat) return(FINISHED);
  /* If not found, return a read error status */
  if(!found) {
    fprintf(stderr,"\nERROR:  Unable to locate KILLNUM, FIRER, and
            TARGET.\n");
    return(READ_ERROR);
  }

  /* Get the F Ratio and F Prob */
  found = FALSE;                    /* while records in file */
```

```
while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
  if(!strstr(line,"Between Groups")) continue;
  /* Skip the D.F, Sum of Squares, Mean Squares,
    but read F Ratio and F Prob */
  /* If nothing was scanned in, there was not data.
    Position the file to the */
  /* next page and return the no data status. */
  if(!sscanf(line,"Between Groups %*i %*f %*f %f
          %f",&fratio,&fprob)) { /* while records in file */
    while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
      if(!strstr(line,"- - O N E W A Y - -")) continue;
      break;
    }
    return(NO_DATA);
  } /* End if !sscanf of the fratio and fprob */
  found = TRUE;
  break;
}
/* If not found, return a read error status */
if(!found) {
  fprintf(stderr,"\nERROR:Unable to locate F Ratio and F Prob.\n");
  return(READ_ERROR);
}

/* Get the total mean */
found = FALSE;
/* Init a counter.
    We are interested in the second line that begins with Total */
i = 0;                  /* while records in file */
while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
  if(strncmp(line,"Total  ", 7)) continue;
  /* Check if this is the second one */
  if(++i != 2) continue;
  /* Skip the Count field and get the Mean */
  sscanf(line,"Total %*i %f",totalmean);
  /* Position the file to the next page */
  while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
                                  /* while records in file */
    if(!strstr(line,"- - O N E W A Y - -")) continue;
    found = TRUE;
    break;
  }
  break;
}
/* If not found, return a read error status */
if(!found) {
  fprintf(stderr,"\nERROR:  Unable to locate total mean.\n");
  return(READ_ERROR);
}

/* Look for the line that says whether or not there is a difference
     between groups */
found = FALSE;                          /* while records in file */
while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
```

```
    if(strstr(line,"(*) Indicates significant differences")) {
      /* You found a difference so break out */
      found = TRUE;
      break;
    }
    else if(strstr(line,"- No two groups are significant different")) {
      /*There was not difference,so position the file to the next
            page and return the no difference status. */
      while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
                                          /* while records in file */
        if(!strstr(line,"- - O N E W A Y - -")) continue;
        break;
      }
      return(NO_DIFFERENCE);
    }
    continue;
  }
  /* If the difference/no difference line was not found,
      return a read error status */
  if(!found) {
    fprintf(stderr,"\nERROR:  Unable to find statement of difference/no
significant difference.\n");
    return(READ_ERROR);
  }

  /* Get to the beginning of the lower left triangle data lines */
  found = FALSE;                      /* while records in file */
  while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
    if(!strstr(line,"Mean  ")) continue;
    /*You are one blank line away. Read the blank line to get past it.*/
    fgets(line, MAXLENGTH_IN, infile);
    /* You are at the first line of the data.
      Now read in the array until */
    /* you get to a blank line.  A blank line terminates the data. */
    found = TRUE;
    alt = 0;                          /* while records in file */
    while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
      /* Get the mean and alternative name.   If you see a form feed, then
      you have reached the end of the data, so break out. If you don't get
      a scan for the mean and alt, then you had a blank line and are
      finished. If it is possible that the array may run over on to the
      next page, we need to revise this code to pick up reading those that
      fall on that page. */
      if(strchr(line,'\014')) break; /* Found a form feed,
                                        so end of data */
      if(ptr = strchr(line,'\n')) *ptr = '\0';
                        /* If present, Remove trailing newline */
      if(!strlen(line)) break;
      sscanf(line,"%f %s",&lltarray[alt].mean, lltarray[alt].alt);
      /* Set difference flags ... */
      /* Use i as the index into the difflag array */
      i = 0;
      /* For each * set a corresponding difflag element to true */
      ptr = line;
```

```
      while((ptr = strchr(ptr, '*')) != NULL) {
        lltarray[alt].difflag[i++] = TRUE;
        ptr++;
      }
      /* Save the number of differences for this alt */
      lltarray[alt].num_diff = i;
      /* Increment the alt number */
      alt++;
    } /* End while to read in the array */
    /* Set the number of alts to the last alt value */
    *numalts = alt;
    /* Position the file to the next page and return the
       different status */                /* while records in file */
    while((stat = fgets(line, MAXLENGTH_IN, infile)) != NULL) {
      if(!strstr(line,"- - O N E W A Y - -")) continue;
      break;
    }
    return(DIFFERENT);
  } /* End while to get to the beginning of the lower left triangle data
       lines */
  /* If not found, return a read error status */
  if(!found) {
    fprintf(stderr,"\nERROR:  Unable to locate lower left triangle
                          data.\n");
    return(READ_ERROR);
  }

} /* End get_llt_data */




/*********************/
/***** proc_diff *****/
/*** BASE case fix ***/
/*********************/

void
proc_diff(killnum,firer,target,lltarray,totalmean,numalts,
            basegroupmean,basegroupnum,kvsb,kvsb_itemcnt,
            alt_kv,alt_kv_itemcnt)
char        killnum[];
char        firer[];
char        target[];
LLTLINE     lltarray[];
float       totalmean;
int         numalts;
float       *basegroupmean;
int         *basegroupnum;
KVSB        kvsb[];
int         kvsb_itemcnt;
ALT_KVSB    alt_kv[];
int         *alt_kv_itemcnt;
```

```
{

  int   col;              /* Column index */
  int   end_group;        /* End of group index */
  float group_mean;       /* Avg of means inside a group */
  int   group_num;        /* Group number */
  int   num_diff;         /* Number differences */
  int   num_in_group;     /* Number of alts in a group */
  int   start_group;      /* Start of group index */
  float sum_of_means;     /* Sum of means in the group */
  int   this_alt;         /* Current alt index */
  int   base_1 = FALSE;   /* If base group # is 1, check for mean */
  int   base_cnt;         /* Counter to get to BASE case */
  int   first_base;       /* First alt in Base Group */
  int   last_base;        /* Last alt in Base Group */
  int   i;                /* Alt index for check */

  /* Init some of the data */
  base_cnt    = 0;
  this_alt    = 0;
  last_base   = 0;
  first_base  = 0;

  /* count until reach the base case */
  while (!
          (!strcmp(lltarray[this_alt].alt,"BASE")  ||
            !strcmp(lltarray[this_alt].alt,"base") ||
            !strcmp(lltarray[this_alt].alt,"Base")) ) {
    base_cnt++;
    this_alt++;
  }
  /* number that base is different from */
  first_base =  lltarray[base_cnt].num_diff;

  /* find last alt not different from base */
  while (lltarray[this_alt].num_diff <= base_cnt &&
      this_alt < numalts) {
    this_alt++;
  }
  last_base = (this_alt - 1);
  /* assign values to base group */
  sum_of_means = 0.0;
  num_in_group =0;
  for (this_alt = first_base; this_alt <= last_base; this_alt++) {
    sum_of_means += lltarray[this_alt].mean;
    num_in_group++;
  }
  group_mean = sum_of_means/num_in_group;
  /* always assign group num 0 to BASE  */
  group_num   = 0;
  *basegroupmean = group_mean;
  *basegroupnum  = group_num;
```

```
  /* work forward through the LLT array grouping the alternatives that are
not different */
  this_alt = last_base +1;
  while (this_alt < numalts) {
    group_num++;
    start_group = this_alt;
    end_group = this_alt;
    /* Init the sum of means and number of alts in the group */
    sum_of_means = 0.0;
    num_in_group = 0;
    /* Group the alts */
    while (lltarray[this_alt].num_diff <= start_group &&
            this_alt < numalts) {
      sum_of_means += lltarray[this_alt].mean;
      num_in_group++;
      this_alt++;
      end_group++;
    }
    /* Get the average mean for the group */
    group_mean = sum_of_means/num_in_group;
    /*Assign the group number and avg mean
            to each member of the group*/
    for(this_alt = start_group; this_alt <= (end_group-1); this_alt++) {
      lltarray[this_alt].group = group_num;
      lltarray[this_alt].group_mean = group_mean;
    }
  } /* End working forward through the LLT array */

  /* Work backward from base through the LLT array */
  start_group = (lltarray[base_cnt].num_diff -1);
  end_group = (lltarray[base_cnt].num_diff -1);
  /* grouping the alternatives that are not different */
  while(end_group > 0) {
    /* Increment the group number */
    group_num++;
    /* Init the sum of means and number of alts in the group */
    sum_of_means = 0.0;
    num_in_group = 0;
    /* Group the alts */
    this_alt = start_group;
    end_group = end_group -(end_group - ltarray[start_group].num_diff );
    while(this_alt >= end_group) {
      sum_of_means += lltarray[this_alt].mean;
      num_in_group++;
      this_alt--;
    }
    /* Get the average mean for the group */
    group_mean = sum_of_means/num_in_group;
    /*Assign the group number and avg mean
      to each member of the group*/
    for(this_alt = start_group; this_alt >= end_group; this_alt--) {
      lltarray[this_alt].group = group_num;
      lltarray[this_alt].group_mean = group_mean;
```

```
    }
    /* Set the start group index */
    start_group = end_group - 1;
  } /* End working backward through the LLT array */

    /* Check for base value == to 0.0 and only one alt != 0.0 */
    /* This other alternative would be a alternative system   */
    /* Include this alternative in base kvsb data structure   */

  if (*basegroupnum == 0 && *basegroupmean <0.00000001) {
     base_1 = FALSE;
     for ( i = 0; i < numalts; i++ ) {
      if (lltarray[i].group != 0) {
         base_1 = TRUE;
           break;
      }
     }
  }  /* if base == 0  and only two groups  */
  if (base_1 == TRUE && lltarray[i].group == 1) {
     kvsb[kvsb_itemcnt].num_killed = lltarray[i].group_mean;
     strcpy(kvsb[kvsb_itemcnt].target,target);
     strcpy(kvsb[kvsb_itemcnt].firer,firer);
  }
      /* if above is not true then continue below */
  else {
      /* put base data in scoreboard */
     strcpy(kvsb[kvsb_itemcnt].firer,firer);
     strcpy(kvsb[kvsb_itemcnt].target,target);
     kvsb[kvsb_itemcnt].num_killed = *basegroupmean;

      /* put differing alts to another data structure */
     for(this_alt = numalts-1;this_alt >= 0; this_alt--) {
      if (lltarray[this_alt].group != *basegroupnum &&
            lltarray[this_alt].group_mean >= 0.00001) {
         strcpy(alt_kv[*alt_kv_itemcnt].alt,lltarray[this_alt].alt);
         strcpy(alt_kv[*alt_kv_itemcnt].firer,firer);
         strcpy(alt_kv[*alt_kv_itemcnt].target,target);
         alt_kv[*alt_kv_itemcnt].num_killed =
            lltarray[this_alt].group_mean;
         *alt_kv_itemcnt += 1;
      }
 /*Check to make sure the alt_kv_itemcnt has not exceeded MAXPAIRS */
      if(*alt_kv_itemcnt >= MAXPAIRS) {
         fprintf(stderr,"ERROR:  Maximum number of pairs has exceeded
                  %d.\n",MAXPAIRS);
           fprintf(stderr,"        Ask programmer that MAXPAIRS be
                        adjusted.\n\n",MAXPAIRS);
           exit(1);
      }
    }
  }

} /* End proc_diff */
```

```
/********************/
/**** proc_nodiff****/
/********************/

void
proc_nodiff(firer,target,totalmean,kvsb,kvsb_itemcnt)

char  firer[];
char  target[];
float totalmean;
KVSB  kvsb[];
int   kvsb_itemcnt;
{

   strcpy(kvsb[kvsb_itemcnt].firer,firer);
   strcpy(kvsb[kvsb_itemcnt].target,target);
   kvsb[kvsb_itemcnt].num_killed = totalmean;
      /* basegroupmean = totalmean when no diff */

} /* end proc_nodiff   */



/************************************/
/* Loads & print kvsb matrix format */
/************************************/

void
write_kvsb(kvsb,kvsb_itemcnt,argc,outfilename)

KVSB  kvsb[];
int   kvsb_itemcnt;
int   argc;
char  outfilename[];
{
  int  i;
  char tgt[MAXPAIRS][LIST_ITEM_LEN];
  int  tgt_itemcnt = 0;

  int  add_item();
  void print_kvsb();

  /* Load the tgt list from the kvsb */
  for(i=0;i<kvsb_itemcnt;i++) {
     if (strcmp(kvsb[i].target,"RED") &&
             strcmp(kvsb[i].target,"BLUE") )
      add_item(tgt, kvsb[i].target, &tgt_itemcnt);
  }
  /* Print the kvsb matrix */
  if(argc == 3) {
    print_kvsb(kvsb, kvsb_itemcnt, tgt, tgt_itemcnt,
      fopen(outfilename,"w"));
```

```
  }
  else {
    print_kvsb(kvsb, kvsb_itemcnt, tgt, tgt_itemcnt, stdout);
  }

}



/*********************************************/
/* Loads & print alts differing from base kvsb */
/*********************************************/

void
write_alt_kv(alt_kv,alt_kv_itemcnt,argc,fp)

ALT_KVSB  alt_kv[];
int     alt_kv_itemcnt;
int     argc;
FILE    *fp;
{
  int  i;


  if(argc == 3) {
     fprintf(fp, "** ----------- BLUE Kills ----------------------\n");
     fprintf(fp, "**     If     in     Then  Kills of Changes  \n");
     fprintf(fp, "**    Below Battle   below below     to      \n");
     fprintf(fp, " \n");

     for(i=0;i<alt_kv_itemcnt;i++) {
      if ( strcmp(alt_kv[i].firer,"RED") &&
            strcmp(alt_kv[i].target,"RED")
          && strcmp(alt_kv[i].firer,"BLUE") &&
            strcmp(alt_kv[i].target,"BLUE" ))
            fprintf(fp,"LTJ1('%-8s','HR??','%-6s','%-6s') =
            %5.2f;\n",alt_kv[i].alt,
                    alt_kv[i].firer,alt_kv[i].target,alt_kv[i].num_killed);
     }
  }
  else fprintf(stderr, "Unable to open differences file");
}



/*********************************/
/********  add_item *************/
/*********************************/

/* Add item to the list in alphabetical order. */


int
```

```
add_item(list, newitem, itemcnt)
char list[MAXPAIRS][LIST_ITEM_LEN];
char *newitem;
int  *itemcnt;
{

  /* If newitem is null, just return */
  if(!newitem || !*newitem) {

    fprintf(stderr,"WARNING:  Attempted to added NULL item
      to list.\n");
    return(1);
  }

  /* If list is full, just return */
  if(*itemcnt == MAXPAIRS) {
    fprintf(stderr,"ERROR:  Item list is full.
      Item %s not added to list.\n",newitem);
    return(0);
  }

  /* If item doesn't exist, add it to the list,increment item count,
     and resort */

  if(!bsearch(newitem,list,*itemcnt,LIST_ITEM_LEN,strcmp)) {
    strcpy(list[*itemcnt],newitem);
    *itemcnt += 1;
    /* Resort the list */
    qsort(list,*itemcnt,LIST_ITEM_LEN,strcmp);
  }

  /* Return success code */
  return(1);

} /* End add_item */



/*************************************/
/********  get_list_index ************/
/*************************************/

/* Find the list index of an item from a sorted list. */
/* Item index is returned if found.  If not found, a */
/* negative value is returned. */


int
get_list_index(list, itemcnt, item)
char list[][LIST_ITEM_LEN];
int  itemcnt;
char *item;
{
```

```
  char *ptr;
  int  idx;

  /* If newitem is null, return */
  if(!item || !*item) return(-1);

  /* Compute the index */
  ptr = (char *) bsearch(item,list,itemcnt,LIST_ITEM_LEN,strcmp);
  if(ptr) {
    idx = ((int)ptr-(int)list)/LIST_ITEM_LEN;
  }
  else {
    idx = -1;
  }

  /* Return index */
  return(idx);

} /* End get_list_index */



/********************/
/***** kvsbcmp *****/
/********************/

/* Compares two KVSB values with respect to the firer+tgt values */

int
kvsbcmp(val1,val2)
KVSB *val1;
KVSB *val2;
{

  char buf1[31];
  char buf2[31];

  /* Combine the firer and target for val1 */
  strcpy(buf1,val1->firer);
  strcat(buf1,"+");
  strcat(buf1,val1->target);

  /* Combine the firer and target for val2 */
  strcpy(buf2,val2->firer);
  strcat(buf2,"+");
  strcat(buf2,val2->target);

  /* Return the result of the comparison of the two buffers */
  return(strcmp(buf1,buf2));

} /* End kvsbcmp */
```

```
/*********************************/
/********  print_kvsb *************/
/*********************************/

/* Print a KVSB matrix. */


void
print_kvsb(kvsb, kvsb_itemcnt, tgt, tgt_itemcnt, fp)
KVSB kvsb[];
int  kvsb_itemcnt;
char tgt[][LIST_ITEM_LEN];
int  tgt_itemcnt;
FILE *fp;
{

  char *tgt_null_format = "%-6.6s  ";
      /* Format for a null value under target */
  char *tgt_real_format = "%5.2f   ";
      /* Format for a real value under target */
  char buf[81];                 /* Gen purpose buffer */
  int  firsttime;               /* First time through flag */
  int  i;                       /* Gen purpose index */
  int  j;                       /* Gen purpose index */
  int  k;                       /* Gen purpose index */
  int  numblk;                  /* Number of block sections */
  int  block;                   /* Block section for printing */
  int  lastcol;                 /* Last column printed */
  int  nextcol;                 /* Next column to print */


  int get_list_index();         /* Function to find tgt column number */
  int kvsbcmp();                /* KVSB order comparison function */

  /* Determine number of blocks to print */
  numblk = tgt_itemcnt/MAXCOL;
  if ((tgt_itemcnt%MAXCOL) != 0) {
    numblk++;
  }
  /* Sort the kvsb array */
  qsort(kvsb,kvsb_itemcnt,sizeof(KVSB),kvsbcmp);

  /* break into  sections or blocks */
  for (block = 0; block < numblk; block++) {
    if (block != 0)
      fprintf(fp,"\n\n+");

    /* Print the target column headers */
    fprintf(fp,"\n%14s"," ");
    /* print column headers up to MAXCOL */
    for(j = (block*MAXCOL); j < ((block+1)*MAXCOL); j++) {
```

```
      fprintf(fp,tgt_null_format,tgt[j]);
    }
    fprintf(fp,"\n");

    /* Null the buffer */
    strcpy(buf,"");
    /* Set firsttime flag */
    firsttime = 1;

    /* For each firer in the kvsb... */
    for(i = 0; i < kvsb_itemcnt; i++) {

      /* If this is a new firer ... */
      if(strcmp(buf,kvsb[i].firer)) {

 /* If not first time through, complete line with null values */
        if(!firsttime) {
          for(k = lastcol; k < MAXCOL; k++) {
            fprintf(fp,tgt_null_format,"    ");
fflush(fp);
          }
   /* Reset the last column index */
          lastcol = block*MAXCOL;
        } /* End if not first time through */
 /* Start a new line */
        fprintf(fp,"\nHR??.%-8s",kvsb[i].firer);
fflush(fp);
 /* Save the name in the buffer for comparing to next firer name */
        strcpy(buf,kvsb[i].firer);
 /* Init the last printed column position */
        lastcol  = block*MAXCOL;
 /* Reset the first time through flag */
        firsttime = 0;
      } /* end if new firer */

      /* Get the column number to print the mean under */
      nextcol = get_list_index(tgt, tgt_itemcnt, kvsb[i].target);

      if (nextcol >= (block*MAXCOL) && nextcol < ((block+1)*MAXCOL)) {

 /*Move to next print column filling nulls with appropriate
                                                        characters*/
        for(k = lastcol; k < nextcol; k++) {
          fprintf(fp,tgt_null_format,"    ");
fflush(fp);
        }
 /* Print the mean in its column */
        fprintf(fp,tgt_real_format,kvsb[i].num_killed);

fflush(fp);
      }

      /* Else we already processed this target in the previous block
           so bypass it now */
```

```
      else continue;

      /* Update the last column printed index */
      lastcol = nextcol + 1;

    } /* End for each firer in the kvsb */
  } /* End block cycle */
} /* End print_kvsb */



/*********************/
/***** echo_data *****/
/*********************/

void
echo_data(killnum,firer,target,lltarray,totalmean,numalts,
           basegroupmean,basegroupnum)
char    killnum[];
char    firer[];
char    target[];
LLTLINE lltarray[];
float   totalmean;
int     numalts;
float   basegroupmean;
int     basegroupnum;
{

  int i,j;

  fprintf(stdout,"killnum   = %s\n",killnum);
  fprintf(stdout,"firer     = %s\n",firer);
  fprintf(stdout,"target    = %s\n",target);
  fprintf(stdout,"totalmean = %.4f\n",totalmean);
  fprintf(stdout,"num alts  = %d\n",numalts);
  fprintf(stdout,"\n");
  if(numalts) {
    fprintf(stdout,"Mean    ALT        Num Diff   Group Num   Group
      Mean\n\n");
    for(i =0; i < numalts; i++) {
      fprintf(stdout,"%9.4f   %-10.10s %5d      %5d        %9.4f   ",
            lltarray[i].mean,lltarray[i].alt,
            lltarray[i].num_diff,lltarray[i].group,
            lltarray[i].group_mean);
      for(j = 0; j < numalts; j++) {
        if(lltarray[i].difflag[j]) fprintf(stdout,"* ");
        else fprintf(stdout,"  ");
      }
      fprintf(stdout,"\n");
    }
    if(basegroupmean < -9999.1 || basegroupmean > -9998.9) {
      fprintf(stdout,"BASE group mean = %f\n", basegroupmean);
    }
```

```
    if(basegroupnum >= 0) {
      fprintf(stdout,"BASE group number = %d\n", basegroupnum);
    }
  } /* End if numalts */
  fprintf(stdout,"\n\n");

} /* End echo_data */
/* lltsum.h */

#define BUFSIZE          96   /* Size of buffer */
#define DIFFERENT        1    /* Differences detected among groups */
#define TRUE              1       /* True */
#define FALSE             0       /* False */
#define FINISHED          0       /* Finished reading file */
#define MAXLENGTH_IN      1024    /* Max length of a read line */
#define MAXLENGTH_OUT     132     /* Max length of an output line */
#define MAXNAME_LENGTH    31      /* Max length of a name */
#define MAXALTS           50      /* Max number of alternatives */
#define MAXPAIRS  1000   /* Max number of firer to target pairings */
#define NO_DATA           2       /* No data for killnum */
#define NO_DIFFERENCE     3 /* No difference detected among groups */
#define READ_ERROR        4       /* Error in reading data file */
#define LIST_ITEM_LEN     11      /* Length of list items */
#define MAXCOL            12   /* Max number of columns per block */

typedef struct {
                float   mean;
                char    alt[MAXNAME_LENGTH];
                int     group;
                float   group_mean;
                int     num_diff;
                char    difflag[MAXALTS];
} LLTLINE;

typedef struct {
                char    firer[MAXNAME_LENGTH];
            char  target[MAXNAME_LENGTH];
            float num_killed;
}KVSB;

typedef struct {
                char    alt[MAXNAME_LENGTH];
                char    firer[MAXNAME_LENGTH];
            char  target[MAXNAME_LENGTH];
            float num_killed;
} ALT_KVSB;

typedef struct {
            char  b_system[MAXNAME_LENGTH];
            float num_lost;
} BSL;

typedef struct {
                char    alt[MAXNAME_LENGTH];
```

```
            char  b_system[MAXNAME_LENGTH];
            float num_lost;
} ALT_BSL;
```

References

Baustista, M.G., Smith, D.W., and Steiner, R.L. (1996): A Cluster-Based Approach to Means Separation, Journal of Agricultural, Biological, and Environmental Statistics, 2,2:179-197.

Brooke, A., Kendrick D., and Meeraus, A. (1988): GAMS - A User's Guide, The Scientific Press, CA.

Conover, W.J. (1996): Texas Tech University, Member of Audience - see Gafner, B.

Dewitz, (1996): Bayesian Analysis Methodology, US Army Material Systems Analysis Activity (AMSAA), MD.

Dowdy, S. and Warden, S.(1991): Statistics for Research, 2d ed., John Wiley & Sons, Inc., New York.

Duncan, D.B. (1955): Multiple Range and Multiple F Tests, Biometrics, 11:1-42.

_____ (1961): Bayes Rules for a Common Multiple Comparison Problem and Related Student-t Problems, Ann. Math. Statist., 32:1013-1033.

_____ (1965): A Bayesian Approach to Multiple Comparisons, Technometrics, **7**:171-222.

Fisher, R.A. (1935): The Design of Experiments, Oliver & Boyd, Ltd., London.

Gafner, B.W. (1996): Grouping and Separation of Means Problem, presented as a clinical problem, U.S. Army Conference on Applied Statistics, 23 Oct., Monterey, CA.

Garver, D.P. (1996): US Naval Postgraduate School, Panelist - see Gafner, B.

Laferriere, R.R.(1991): Brigade Mix Model, TRAC-WSMR, published in the proceedings of the 6th ROK-US Defense Analysis Seminar.

Lehmann, E.L. (1957): A Theory of Some Multiple Decision Problems, I, Ann. Math. Statist., 28:1-25.

Mackey, D.C. et al.(1995): CASTFOREM Users Guide, TRAC-WSMR-TD-95-024, TRADOC Analysis Center-White Sands Missile Range (TRAC-WSMR), NM.

Miller, R. G. Jr.(1981): Simultaneous Statistical Inference, 2d ed., Springer-Verlag, New York.

Porter, R.W. (1996) Anti Armor Resource Requirements Study (Classified), TRAC-WSMR, NM.

Scheffe (1953): A Method for Judging all Constraints in the Analysis of Variance, Biometrika, 40:87-104.

SPSS, (1983): SPSS User's Guide, SPSS Inc., Chicago.

Tukey, J.W. (1949): Comparing Individual Means in the Analysis of Variance, Biometrics, 5:99-114.

_____(1952): Allowances for Various Types of Error Rates. Unpublished IMS address, Virginia Polytechnic Institute, Blacksburg, VA.

_____ (1953): The Problem of Multiple Comparisons, Unpublished manuscript.

Acronyms

	A
Alts	alternatives
	B
BHG	base grouping heuristic
	C
CASTFOREM	Combined Arms and Support Task Force Evaluation Model
	H
HSD	honest significant difference
	L
LLT	lower left triangle
LSD	least significant difference
	M
MCP	multiple comparison procedures
MM	Mix Model
	R
Rep	repetition
	S
SPSS	Statistical Package for the Social Sciences
	T
TRAC-WSMR	TRADOC Analysis Center-White Sands Missile Range
TRADOC	Training and Doctrine Command
Trt	treatment
	U
US	United States