TRAC-M-TR-19-020
May 2019

Enhanced Heavy Equipment Transport System (EHETS) Alternatives of Analysis (AoA) Sensitivity Analysis

The Research and Analysis Center
700 Dyer Road
Monterey, CA 93943-0692

This study was mission funded
Prepared on 20180405
TRAC Project Code # 010096

THIS PAGE INTENTIONALLY LEFT BLANK

TRAC-M-TR-19-020
MAY 2019

Enhanced Heavy Equipment Transport System (EHETS) Alternatives of Analysis (AoA) Sensitivity Analysis

Authors

MAJ Ta'Lena Fletcher
MAJ James Jablonski

PREPARED BY:

MAJ Ta'Lena Fletcher
MAJ, US Army
TRAC-MTRY

APPROVED BY:

LTC Brian Wade
LTC, US Army
Director, TRAC-MTRY

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** 17 May 2019	**3. REPORT TYPE AND DATES COVERED** Technical Report, November 2018 to May 2019	
4. TITLE AND SUBTITLE Enhanced Heavy Equipment Transport System (EHETS) Alternatives of Analysis (AoA) Sensitivity Analysis			**5. PROJECT NUMBERS** TRAC Project Code 010096
6. AUTHOR(S) MAJ Ta'Lena Fletcher and MAJ James Jablonski			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army The Research and Analysis Center - Monterey 700 Dyer Road Monterey CA, 93943-0692			**8. PERFORMING ORGANIZATION REPORT NUMBER** TRAC-M-TR-19-020
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) US Army The Research and Analysis Center - Lee 700 Dyer Road Fort Lee, Virginia 23801			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** TRAC-M-TR-19-020
11. SUPPLEMENTARY NOTES Findings of this report are not to be construed as an official Department of the Army (DA) position unless so designated by other authorized documents.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The Army introduced the M1 Abrams Main Battle Tank (MBT) in 1979 along with the Heavy Equipment Transport System (HETS) to act as its primary transport. In response to new technologies and emerging threats on the battlefield, the M1 Abrams tank underwent numerous upgrades and is projected to exceed the carrying capacity of the current HETS. Therefore, The Research and Analysis Center (TRAC) is conducting an Analysis of Alternatives (AoA) for a new transporter, the Enhanced Heavy Equipment Transport System (EHETS). As part of this effort, TRAC performed sensitivity analysis to determine the effect of operational and maintenance variables on operational availability of the EHETS. Operational availability is a critical metric for the EHETS, and it is important to understand the impact of system design and external factors on availability. This research makes use of design of experiments (DOE) to quantify the effects of primary operational and maintenance variables which affect the operational availability of the EHETS. These variables include reliability, recovery time, parts distribution and time required to repair within an operational scenario. The research used Logistics Battle Command (LBC) simulation model's dynamic maintenance feature to conduct the operational availability sensitivity analysis.

14. SUBJECT TERMS design of experiments (DOE), logistics regression, neural network, simulation modeling			**15. NUMBER OF PAGES** 29
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

NOTICES

DISCLAIMER

Findings of this report are not to be construed as an official Department of the Army (DA) position unless so designated by other authorized documents.

REPRODUCTION

DISTRIBUTION STATEMENT

Approved for public release; distribution is unlimited.

DESTRUCTION NOTICE

When this report is no longer needed, DA organizations will destroy it according to procedures given in AR 380-5, DA Information Security Program. All others will return this report to Director, TRAC, ATTN: ATRC, 255 Sedgwick Avenue, Fort Leavenworth, Kansas 66027-2345.

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Army introduced the M1 Abrams Main Battle Tank (MBT) in 1979 along with the Heavy Equipment Transport System (HETS) to act as its primary transport. In response to new technologies and emerging threats on the battlefield, the M1 Abrams tank underwent numerous upgrades and is projected to exceed the carrying capacity of the current HETS. Therefore, The Research and Analysis Center (TRAC) is conducting an Analysis of Alternatives (AoA) for a new transporter, the Enhanced Heavy Equipment Transport System (EHETS). As part of this effort, TRAC performed sensitivity analysis to determine the effect of operational and maintenance variables on operational availability of the EHETS. Operational availability is a critical metric for the EHETS, and it is important to understand the impact of system design and external factors on availability. This research makes use of design of experiments (DOE) to quantify the effects of primary operational and maintenance variables which affect the operational availability of the EHETS. These variables include reliability, recovery time, parts distribution and time required to repair within an operational scenario. The research used Logistics Battle Command (LBC) simulation model's dynamic maintenance feature to conduct the operational availability sensitivity analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

AoA: Analysis of Alternatives

Ao: Operational Availability

CCDC DAC: U.S. Army Combat Capabilities Development Command Data Analysis Center

DOE: Design of Experiment

EHETS: Enhanced Heavy Equipment Transport System

EFF: Essential Functions Failure

HETS: Heavy Equipment Transport System

LBC: Logistics Battle Command

MBT: Main Battle Tank

MTBF: Meantime between Failures

TRAC: The Research and Analysis Center

SA: System Abort

STONs: Short Tonnage

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

INTRODUCTION

1.1. PURPOSE

TRAC conducted this sensitivity analysis as a part of the EHETS AoA to explore the effects of varying vehicle and maintenance system characteristics on EHETS operational availability. The characteristics included: 1) system reliability (mean time between failures), 2) time to recover, 3) repair time, 4) number of mechanics required to repair a fault, 5) part acquisition time, and 6) probability of correct diagnosis. These characteristics became input variables for a dynamic maintenance model within the Logistics Battle Command (LBC) simulation. TRAC used a designed experiments approach to explore reasonable ranges of these input variables to determine their impact on the EHETS ability to maintain the recommended operational availability rate of 75%. This information will aid decision makers in establishing acceptable operating parameters for the EHETS.

1.2. BACKGROUND

The Army acquired HETS primarily to move the M1A1 Abrams Main Battle Tank (MBT). Due to upgrades, the M1A1 Abrams Main Battle Tank (MBT) is projected to exceed the HETS carrying capacity of 70 short tons (STONS). The Army is conducting the EHETS Analysis of Alternatives (AoA) in order to identify a suitable replacement. In support of the AoA, TRAC conducted sensitivity analysis on operationally relevant scenario in LBC. The intent was to determine whether potential EHETS designs will meet or exceed the recommended operational available rate of 75% under varying vehicle and maintenance system characteristics.

TRAC leveraged design of experiments (DOE) to generate data required for the sensitivity analysis. A carefully designed experiment is much more efficient and effective than a 'hit-or-miss' sequence off simulation runs or 'trying' a number of alternative configurations (Law, 2007). The approach enables analysts to determine variable importance and develop a response surface or metamodel of the given experiment. A metamodel, which usually takes the form of a first- or second-order linear regression equation, can then be leveraged to predict the response variable for input variable values of interest (Law, 2007). A linear regression based metamodel enhances sensitivity analysis by enabling predictions within the limits of the original experimental design

space while also providing confidence intervals for predictions that account for the uncertainty inherent at varying factor levels. The metamodel enables sensitivity analysis through exploration of the response surface and permits the experimenter to determine effect of changing one or more factors on the response variable (Sacks, Welch, Mitchell, & Wynn, 1989).

Experimental design is a robust field about efficient planning and analyzing the results of an experiment. The result of an experimental design plan is a matrix of input variable values where each column represents a variable and each row represents the combination of input variables for a single design point. Many options are available for experimental designs, but for this experiment we chose nearly orthogonal latin hypercube (NOLH) designs due to the relatively high number of input variables. NOLH's provide an efficient experimental design as they enable the determination of variable significance, nonlinear effects, variable interactions, and their associated ranges while requiring relatively few runs. These designs eliminate certain masking effects created by fractional factorial designs while still reducing the number of runs required to generate a metamodel. TRAC chose this design construct for this experiment as it enabled the team to generate insight about EHETS maintenance system in an efficient manner (Cioppa, 2009).

1.3. CONSTRAINTS, LIMITATIONS, & ASSUMPTIONS

Constraints:

- The project completion deadline is April 2019.

Limitations:

- The EHETS is currently under development so the U.S. Army Combat Capabilities Development Command (CCDC) Data Analysis Center (DAC) surrogated operational data based on existing HET data.
- No random variables are used in the dynamic maintenance model for repair time, recovery time, and part acquisition times across all six system abort (SA) faults for the M1070A1 and five SA faults for the M1000.
- The input values for repair time, recovery time, number of mechanics, part acquisition times and probability of correct diagnosis do not have variation across all six SA faults for the M1070A1 and five SA faults for the M1000.

- The essential functions failure (EFF) for the M1070A1 and M1000 were not evaluated in this research.

Assumptions:

- The operational data provided by CCDC DAC is sufficient to identify the key variables and ranges that effect A_o.
- Constant values for repair, recovery, and part acquisition times will still enable effective analysis of EHETS reliability.
- The input values utilized for the variables will allow for equal evaluation on how they influence the A_o.

1.4. STUDY TEAM

MAJ Ta'Lena Fletcher

MAJ James Jablonski

MAJ James Streams

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 2. METHODOLOGY

2.1. PROBLEM SCOPING

TRAC's sensitivity analysis focused on the interaction of the input variables across various system abort faults and maintenance system factors. The goal was to determine which factors and combinations had the greatest influence on the operational availability of the EHETS with a minimum threshold goal of 75%. The EHETS system is composed of a tractor and a trailer, so the experiment required simulated maintenance on both components of the system to account for shared resources and recovery times. TRAC used an LBC scenario designed specifically to leverage LBC's dynamic maintenance feature in an operationally relevant scenario. Input variable range starting points were selected by operational maintenance subject matter experts at TRAC Lee based on historical maintenance data.

2.2. TOOL CHAIN

TRAC developed a tool chain consisting of a series of scripts that served as a wrapper program for LBC experimentation. A wrapper program creates scenario files based on an experimental design and a base scenario, executes the simulation, stores or manages requisite simulation output files, and automates postprocessing or analysis (Wade, 2017). TRAC created the wrapper using a combination of JMP software and R code to build the NOLH design, generate LBC dynamic maintenance XML scenario files, run the experiment, and conduct analysis. In addition to conducting analysis in support of the EHETS AoA, TRAC intended to develop a methodology and the requisite tools that would be transferrable to simulations other than LBC with minimal effort. To that end, TRAC created a series of functions provided as an appendix to this report for future experimentation. Where possible, open source tools were used to implement the toolchain. JMP was the only non-open source software platform or package used for the actual execution of this experiment, but any experimental design generation script can be used to replace it in the future.

2.3. EXPERIMENTAL DESIGN GENERATION

Many options exist for NOLH design generation. TRAC originally leveraged an open source package called DiceDesign to generate nearly orthogonal designs based on Cioppa's 2009 methodology and results (Dupuy, 2015). The package did not enable the rapid generation of a higher density of nearly orthogonal design points within a defined range as it referenced Cioppa's original designs. Variability inherent in the model output required a higher resolution exploration of the space to reliably predict operating points at the extreme ranges of input variable parameters. JMP's experimental design feature provided a scriptable platform to efficiently generate nearly orthogonal designs rapidly with a high number of design points. Design generation scripts using DiceDesign and JMP are provided as appendix III and IV.

2.4. SCENARIO FILE GENERATION

The scenario file generation portion of the DoE wrapper generated amended LBC scenario XML files from the values contained in the DOE. A typical scenario file generation script extracts the design values from the DOE matrix, amends these values to appropriate locations in the scenario XML, and then stores the resultant XML files (Wade, 2017). For this scenario, the LBC scenario files provided by TRAC Lee required specialized scripting due to each M1070A1 and M1000 entity being individually defined with performance and maintenance characteristics, rather than in a typical performance database schema storing vehicle characteristics in a single location by vehicle type. TRAC created a series of R functions that imported design values from the design matrix, identified and then stored all corresponding XML node locations in a data frame, and then looped through corresponding node locations amending design values as required. The code is provided as appendix I.

2.5. EXPERIMENT EXECUTION

Designed experiments with stochastic combat simulations require multiple replications at multiple design points. TRAC utilized 50 and 100 replications per design point in a Monte Carlo method to create a distribution for each output. TRAC's goal was to design an experiment that enabled the creation of a metamodel to predict when operational availability reached below 75%, with a 99% confidence interval width of less than 5%. This task required many iterations of design, execution, and analysis.

LBC is a relatively lightweight simulation that requires approximately 4GB of RAM, 1 core to execute, and completes individual runs in seconds or less. Despite this fact, TRAC ultimately required 1200 design points with 100 replications each to adequately model operational availability with the desired accuracy. Furthermore, multiple iterations were required to adequately explore the ranges of design input variables and better understand nonlinearities associated with certain input variable combinations. To enable this, TRAC leveraged parallel processing and high-performance computing (HPC) assets at the Naval Postgraduate School.

In order to make the methodology easily transferrable to other organizations, TRAC used base packages in R to enable basic parallelization on any machine or HPC. The rparallel package managed processes and enabled execution of multiple runs simultaneously, while the system2 function called java and executed LBC over a series of wrapper generated XMLs. See appendix III for the runExperiment function source code. This provided the ability to run multiple simulations on a multi-core workstation. The final experiment consisting of 1200 design points as ultimately run on a virtual linux machine with over 48 cores and 256 GB of RAM on the Naval Postgraduate School HPC.

2.6. POST PROCESSING

TRAC developed a script to rapidly collect LBC output files and calculate operational availability of the M1000 and M1070A1 by replication and design point. Operational availability is defined as the percentage of time a system is available for operations within a given timeframe. LBC discrete event output data provided the time when systems failed, and when they were once again available for duty. For each individual M1000 or M1070A1, TRAC used this information to calculate Ao and averaged this across systems. A final postprocessing step joined post-processed A_o results with the design matrix for further analysis.

TRAC created a script that used the fread function from the R data.table package to rapidly ingest results from all design points and replications and append it into a single data table (See postprocessor.R in appendix III). The script then added columns representing the design point and replication, and used DPLYR to join the design matrix to the results. This single large data table provided the analysts means to calculate A_o for each run rapidly by leveraging DPLYR (see A_o Calculator function in appendix III).

2.7. ANALYSIS AND MODELING

TRAC leveraged R scripts to produce final analytic products, and JMP to rapidly confirm R scripting and visualize results throughout the process. The primary methods by which TRAC conducted this analysis were Pearson correlation and linear regression. These tools provided the means to understand variable relationships and generate a metamodel capable of predicting A_o.

TRAC used normalized (Pearson) correlation to assess input variable relationships and gain insight to which variables had a positive or negative linear correlation with A_o. Correlation values range between -1 and 1, a positive correlation indicated that as A_o increases the input variable also increases while a negative correlation indicates as A_o decreases the input variable increases (Pearson Correlation Coefficient, 2019). Quickly assessing the relationships between variables and operational availability enabled TRAC to assess our designs. The correlation matrix confirmed the desired near-zero correlation between input variables for the design and A_o's relationship with the input variables.

TRAC used first and second order linear regression to model A_o and gain insight into variable relationships, interactions, and variable importance. The team built linear regression models with both JMP and R. JMP provided the means to quickly and interactively assess results and variable relationships early in the wrapper development process, and gradually was replaced by R code as the toolchain matured. R provided an open-source toolset capable of modeling A_o through regression while also handling XML manipulation, scenario run management, and postprocessing. Regression provided the means to explore the influence of the input variables on A_o first revealed by the correlation, and also enabled prediction of A_o through a second order linear model. First-order linear regression identified, in order of influence, the variables that yielded the strongest influence on A_o whether positive or negative (See Appendix IV for Linear Regression Code). A second-order model provided a response surface that enabled the team to investigate variable interactions and their influence on the response variable (SAS Institute, 2019).

2.8. ESTIMATING OPERATING PARAMETERS (A_o THRESHOLD)

To estimate the operating parameters (model input variables) required for the maintenance system to maintain A_o above 75%, TRAC leveraged the response surface model described above.

Using this metamodel, the team obtained predictions of A_o under multiple maintenance system parameter combinations without re-running LBC. To obtain an estimate for operating parameter thresholds, the team first set all input variables to their base case settings. TRAC Lee estimated these base case values based on historical HET data and scenario 7 for maintenance system variables. To find an individual operating parameter estimate, the analysts adjusted a single input variable up or down until A_o crossed the 75% threshold. The team accepted a model and its estimate for the threshold value only if the confidence interval at that predication was less then 5% wide and the input variable value fell within the design's experimental ranges for that variable.

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 3. ANALYSIS AND FINDINGS

3.1. INTRODUCTION

TRAC executed several iterations of the wrapper program to estimate a minimum acceptable operating level for the input variables. With each iteration, the team sought to confirm and improve the design in four aspects. First, the operational requirement was to maintain A_o >= 75%, so after each experiment the team adjusted the design to adequately explore the space above and below the 75% threshold. This enabled reliable predictions by the response surface in around the required threshold. The second was to ensure that input variable ranges did not result in combinations that caused the model to produce unrealistically high or low A_o values (0-10% or 100% A_o). Values in these extreme regions often indicated that input variable values or combinations were too extreme, and the inherent limits on A_o had the effect of introducing further nonlinearities into the model. Thirdly, the team sought to ensure that variable ranges were both reasonable and consistent with real-world possibilities. For example, requiring 15 mechanics to repair one fault is unreasonable and unnecessarily consumes resources in the model. Finally, the analysts adjusted their designs such that the limits for each input variable would include values that 'broke' the 75% A_o threshold, and that resulted in confidence intervals less than 5% wide at that value.

For brevity, three iterations are highlighted below. The first is the initial design which included too many input variables and revealed a problem with a lack of requisite modeling data. The second iteration discussed did not have wide enough ranges or enough replications to accurately predict where A_o fell below 75% for several variables. It did, however, produce insight that enabled the appropriate ranges, resolution of design points in that range, and the appropriate number of replications to be generated by the final iteration. It also provided some useful insight on the upper ranges of A_o that show diminishing returns with higher system reliability. The final iteration yielded a model capable of predicting acceptable operating parameters for all input variables associated with the M1070A1, and all but two of the input variables for the M1000 trailer. The team determined that increasing the design space to 'break' A_o for the M1000 with all input variables independently would result in an unrealistic scenario and produce results of limited utility for both systems and so did not run any further experiments.

3.2. INITIAL SPACE FILLING DESIGN – *INSUFFICENT INPUT DATA*.

TRAC created the initial designed experiment with the intent of examining 22 total variables for the M1070A1 (11 variables) and M1000 (10 variables): 1) system reliability (mean time between failure for six separate faults), 2) time to recover, 3) repair time, 4) number of mechanics required to repair a fault, 5) part acquisition time, and 6) probability of correct diagnosis. The intent was not only to examine the repair system, but also to differentiate between various types of faults. Fault types included: 1) SA-other, 2) SA-tire, 3) SA-drivetrain, 4) SA-steering, 5) SA-5th wheel, and 6) SA-suspension. Naturally, the trailer did not include a drivetrain failure. Each individual fault type had its own time between failure, and the intent was to determine required system parameters for these individual fault types and for the maintenance system related parameters. As an example, the study team could then say that a potential tractor candidate HET replacement should maintain a certain value for the mean time between suspension failures and the associated repair time because these would have the greatest positive effect on availability. The initial run demonstrated that the team did not have enough data to inform such recommendations, but did provide initial insight into realistic variable ranges and overall system performance.

3.2.1 INITIAL SPACE FILLING DESIGN DISCUSSION.

The initial DOE had three overall problems. First, the initial design did not have enough design points to adequately explore the full range of the input variables. There were certain areas of the design that contained non-linearities for which there was not enough data to accurately model. The team revealed this problem by examining A_o prediction confidence intervals created by the second order linear metamodel and observing several well above the 5% goal at 99% confidence. Second, there was relatively high variability in some design points, with some replications containing A_o values at greater than 80% and less than 10%. The standard deviation of A_o for many design points was also well over 20%, so the team made two further alterations to the design. The first was to reduce the maximum limit of the 'number of mechanics' input variable which caused non-linearities due to resource consumption and was unrealistically high at 15 and 22 for M1070A1 and M1000 respectively. To further reduce

metamodel prediction confidence intervals and account for the high variance at certain design points the team also increased the number of replications from 50 to 100 per design point.

The third and primary problem with the initial experiment was that there was not enough data to examine system reliability with multiple fault types. Namely, TRAC did not have data on individual fault part acquisition time, repair time, or correct diagnosis probability. The system abort-other failure type for both M1070A1 and M1000 was causing the most down time of all fault types, as illustrated in the top row of Figure 1 below. Figure 1 consist of three rows representing fault data by system, M1000 is represented by pink and M1070A1 is represented by blue: row 1 illustrates the percentage of time a system is down by fault, row 2 illustrates the total number of occurrences by fault and row 3 illustrates the average downtime based on the type of fault.

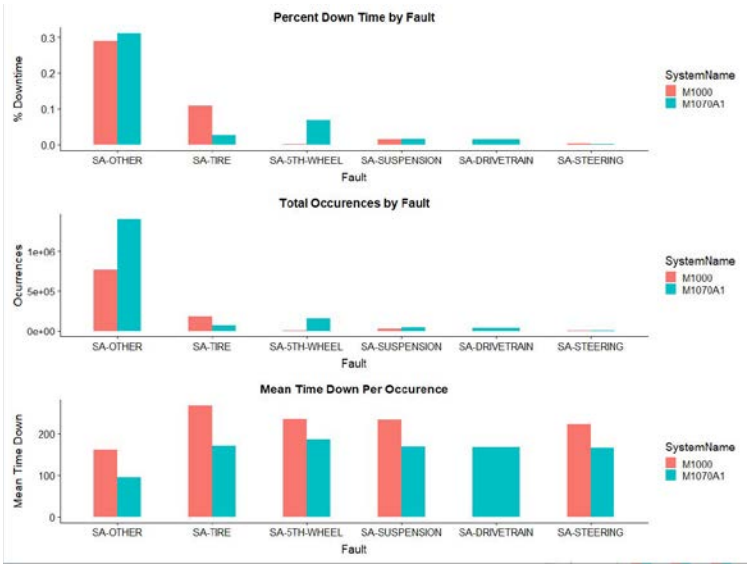


Figure 1: Initial Results

Upon closer examination of figure 1 and the initial output data, the team realized that the given input data did not support the level of fidelity that the team was trying to model. This was because all faults used the same repair system data and SA-other had the lowest mean time between occurrence. A drivetrain, other, or tire fault would all take the same amount of time to repair. Similarly, SA-tire faults had similar mean down time per occurrence to all other fault

types which made little 'real-world' sense for this type of fault. It became apparent that TRAC did not have repair or diagnosis data to adequately model system level faults or draw any meaningful insight from experimenting at this resolution. For the rest of the designs, TRAC elected to use a single variable for system reliability. The team combined individual fault types into a single system reliability input variable by merging the independent Poisson distributions into a single variable with a mean of 800.2758 for M1070A1 and 1644.4013 for M1000.

3.3. IN-PROGRESS SPACE FILLING DESIGN – *CENTERED AROUND ADJUSTED OPERATING PARAMETERS.*

The second example illustrated here demonstrates the robustness of the maintenance system and shows intermediate progress towards generating a metamodel capable of predicting parameter thresholds for the EHETS maintenance system. Based on previous iterations, the team further saturated the design space to include 800 design points. This enabled more precise modeling, but required a reduction in replications to 50 in order to run and postprocess the experiment in a reasonable timeframe. For this iteration, the team also expanded the upper bounds for several input variables in an attempt to include the input variable operating points where A_o reached below 75%. The limits chosen for this iteration are shown in table 1.

Input Variables	**M1070A1 (Tractor)**	**M1000 (Trailer)**
System Reliability – Mean time between failure (hours).	240.1 – 1360.5	246.7 – 3042.1
Recovery Time – to recover vehicle to repair facility (hours).	1 – 120	
Repair Time – to repair identified fault (hours).	1 – 96	
Mechanics – # of mechanics required to repair identified fault.	1 – 10	
Probability of Correct Diagnosis	.5 – 1	
Acquire Parts – Time required to obtain repair parts (hours).	.5 – 107	

Table 1 Design Limits for Example Iteration

3.3.1 IN PROGRESS SPACE FILLING DESIGN RESULTS EXPLORATION.

The results of a second order stepwise linear regression model estimating operating parameters for the EHETS maintenance system are shown below in Figure 2 (M1070A1) and Figure 3 (M1000). Both results demonstrate the robustness of the EHETS maintenance system at base values. Recall how the team held all but one variable at the base case value, and then varied that variable across its design range to obtain the point where A_o crossed the 75% threshold. In both figures, this trace is shown in red. Note how for all but three input variables in Figure 2, base A_o does not cross 75%. To obtain operating parameter estimates for probability of correct diagnosis, part acquisition time, and recovery time, the team adjusted the base case reliability by a factor of 50 – 150%. This enabled estimation of the M1070A1 operating parameters, albeit at non base case reliability. As illustrated by the M1000 results in Figure 3, the team could not estimate operating parameters even with the system reliability adjustment. These results, along with residual plots (see Appendix I) showing the design centered well above the target A_o,

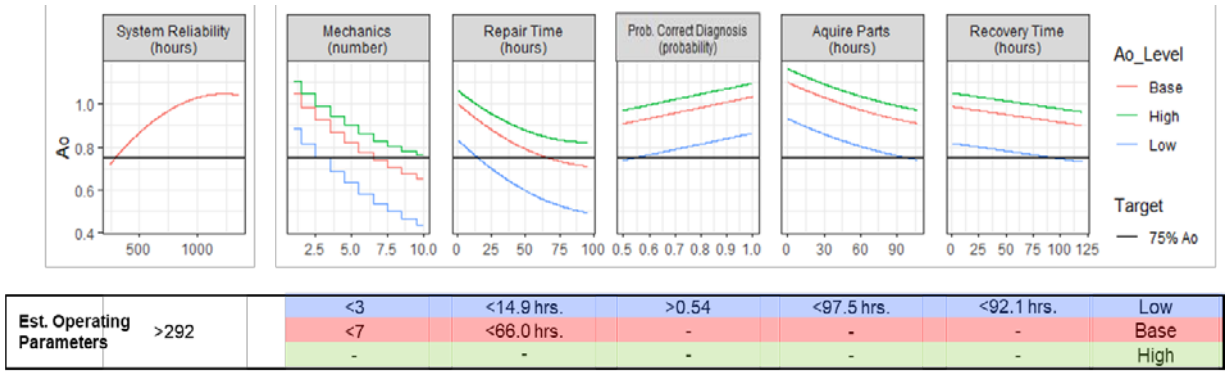


Figure 2: Intermediate Operating Parameter Estimate M1070A1

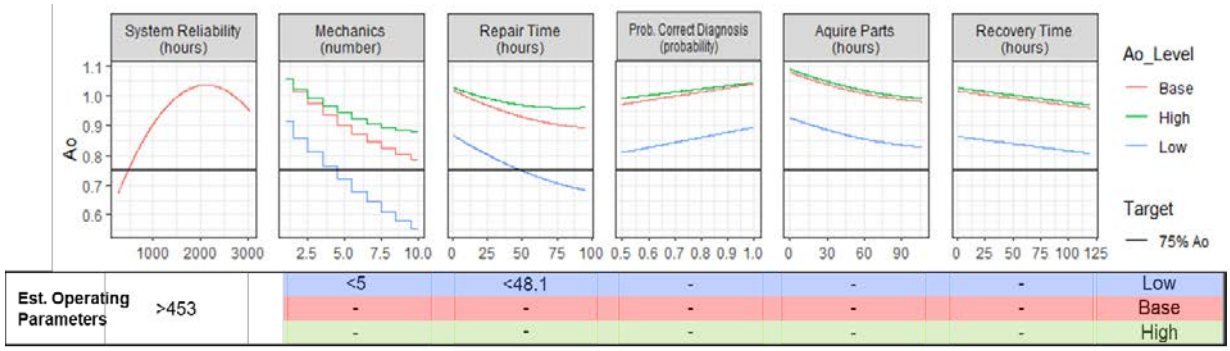


Figure 3: Intermediate M1000 Operating Parameter Estimation

demonstrate the resiliency of the EHETS system, and led the team to expand limits of the input variables in latter experiments.

The result also showed diminishing gains in A_o as system reliability increased. This is shown in Figure 2 and Figure 3 as a decrease in slope after 1200 hours for M1070A1 and 2000 hours for M1000. This design explored the space above 75% A_o, very thoroughly and so the team concludes that there would be little to gain in increasing the EHETS mean time between failure well above these values.

3.4. FINAL SPACE FILLING DESIGN – *DESIGN CENTERED BELOW OPERATIONAL AVAILABILITY 75% THRESHOLD FOR M1070A1 & M1000.*

Based on the findings in previous iterations, the team arrived at a final design capable of predicting recommended operating parameters for all but two associated input variables. For the final design, the team further saturated the design space with 1200 design points at 100 replications per design. Use of the NPS HPC cluster enabled the additional design points and replications despite the increased compute and storage requirements. TRAC centered the final design below the 75% operational availability target, as shown in the residual plots in Appendix I. The team originally desired a more target-centered design, but the ranges required to 'break' A_o for all input variables forced the design center lower. Lowering the maximum value for the number of mechanics required variable not only better aligned it with real-world force structures, but also reduced the variability it caused in certain design combinations where the mechanic resource was overused.

The resultant design is shown in Table 2. In contrast the previous example, the team tailored input variables to the trailer and tractor separately. Previous experiments with shared values across these systems caused the unexpected result of repair resource consumption that both increased variability and explored a far larger space than required. The success of this final iteration thus laid in its targeted operating ranges designed to break A_o for the trailer and tractor and the highly saturated design space.

Input Variables	**M1070A1 (Tractor)**	**M1000 (Trailer)**

System Reliability – Mean time between failure (hours).	240.1 – 1360.5	246.7 – 2250
Recovery Time – to recover vehicle to repair facility (hours).	2 – 250	2 – 480
Repair Time – to repair identified fault (hours).	1 – 100	1 – 364
Mechanics – # of mechanics required to repair identified fault.	1 – 9	
Probability of Correct Diagnosis	.19 – 1	
Acquire Parts – Time required to obtain repair parts (hours).	.5 – 242	.5 – 525

Table 2 Final Operating Ranges

3.4.1 FINAL SPACE FILLING DESIGN RESULTS

3.4.1.1. VARIABLE IMPORTANCE

Figure 6 shows input variable contributions for a first-order linear model of the EHETS maintenance system. Perhaps unsurprisingly, the most important factor for both systems was system reliability. If the system does not fail, then it does not become unavailable and enter the maintenance system. The number of mechanics required to fix a fault was the next most important variable. This variable caused significant nonlinearities to occur throughout the iterative process. LBC treats mechanics as a resource, and once consumed, a backup of tractors and trailers in the maintenance process occurs. Probably of correct diagnosis is related to mechanic consumption and the third most important input variable, indicating that the complexity of a design or system and its accompanying training and/or documentation may be nearly as crucial as its reliability. Finally, recovery times were the least important variable. This indicates that candidate EHETS designs need not overly focus on recoverability or improve recoverability from the previous HET design.

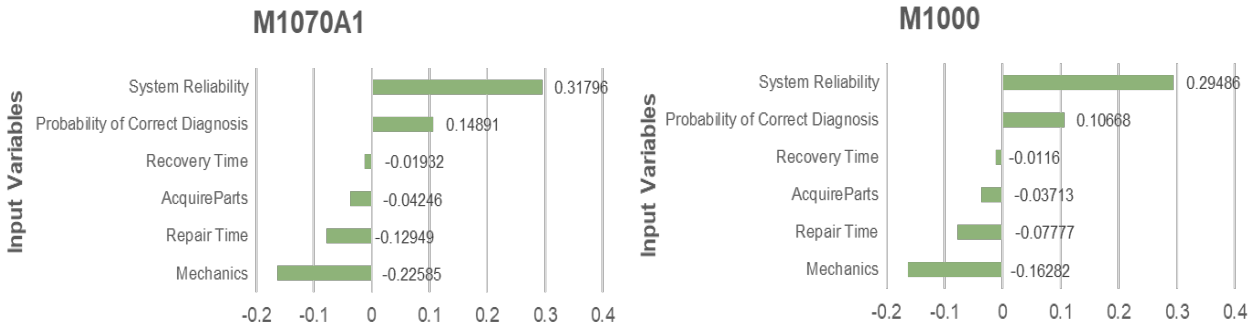


Figure 4: M1070A1 Input Variable Contributions

3.4.1.2. ESTIMATING OPERATING PARAMETERS

TRAC implemented targeted system operating ranges in this design to attain a minimum acceptable operating level for the trailer and tractor. The team generated parameter estimates for all variables associated with the M1070A1 tractor and all the variables except recovery time and probability of correct diagnosis for M1000 trailer. See the corresponding results shown in Figure 5 and Figure 6. The figure has six facets corresponding to the six input variables, the vertical axis is predicted availability, the horizontal axis represents the range of that variable explored in the design, and a 99% confidence interval is shaded grey. Interestingly, estimates for the system reliability parameter increased from the second iteration presented in this analysis. The increase was by ~100% for the M1070A1 and the M1000 up to 502 hours and 1072 hours respectively. This result is more conservative as it calls for longer mean time between failures. Furthermore, it is based on a design more centered on lower A_o and so provides more accurate predictions (narrower confidence intervals) than the previously lower operating parameter estimates for system reliability.

The shapes of the plots reveal a similar result from the second example. The slope of the system reliability curve lessons as system reliability increases within the range of our design. This indicates that the Army could expect diminishing gains from higher levels of system reliability. Finally, the other variables do not appear to have any meaningful slope changes or nonlinearities, and still hold the same general shape as in the previous example. In that way, although out parameter estimates differ, we can confirm the importance of focusing on system

reliability first and that the overall behavior of the maintenance system is relatively consistent across designs.

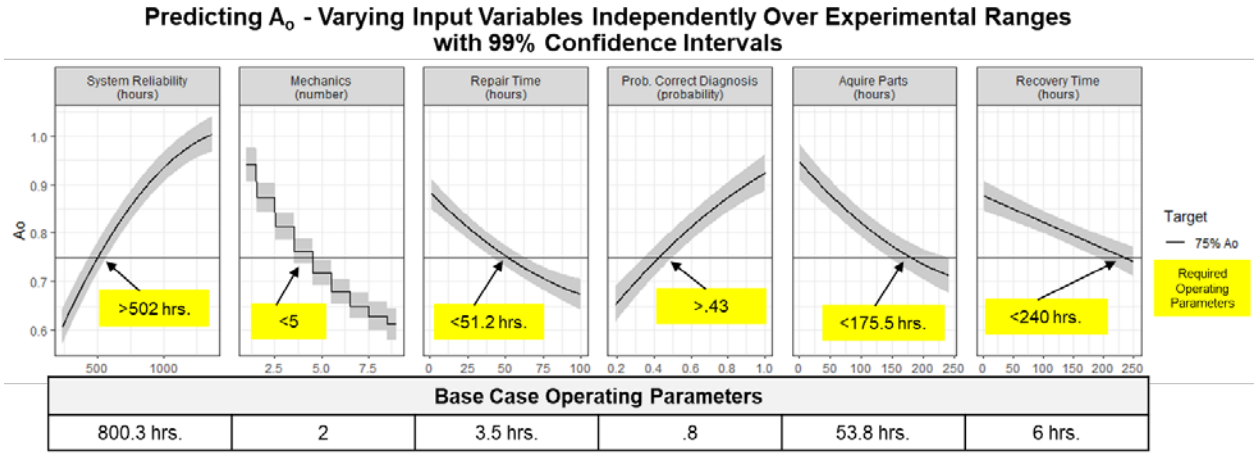


Figure 5: M1070A1 Predicting A_o

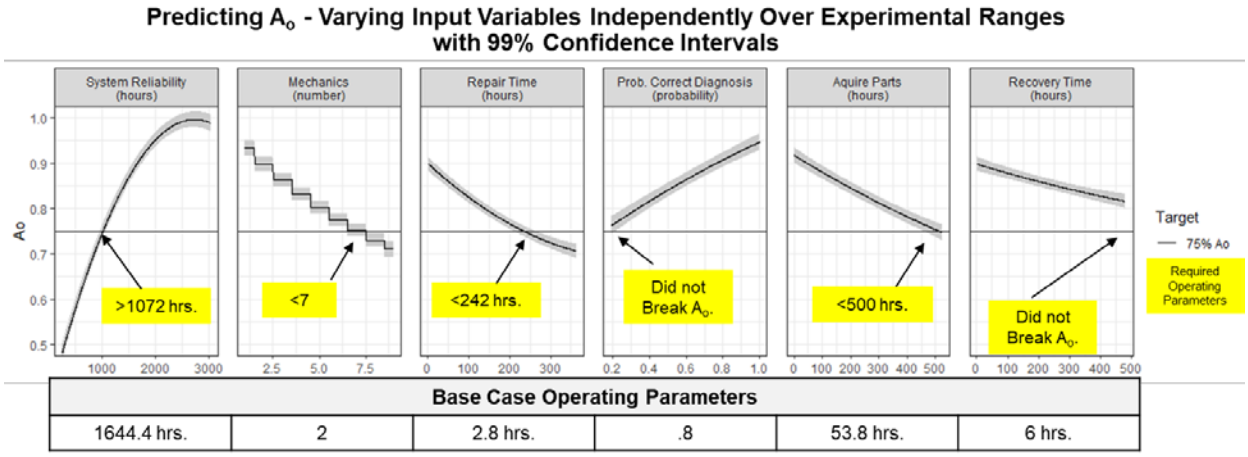


Figure 6: M1000 Predicting Ao

SECTION 4. CONCLUSION

TRAC conducted this sensitivity analysis using designed experiments in LBC to support the EHETS AoA and inform design characteristics for the new system. Several recommendations about design and operational characteristics were elicited by the study team and a list of these conclusion are provided in the sections below. To support future efforts in LBC and other simulations, the team documented their efforts in scripts and through the production of this document. TRAC used these scripts to generate multiple designs, adjust/produce simulation scenario files, run the simulation, and conduct the requisite postprocessing. The DOE, coupled with high performance computing capabilities was instrumental in determining the operating ranges of the input variables and in understanding system performance despite a lack of specific maintenance characteristics for new candidate EHETS. This methodology reduced and focused computational and analytic effort. These tools also made it possible to execute multiple iterations within a timeframe of hours. Which is in stark comparison to a timeframe of days that the team experienced earlier in the process.

This section will cover the operational characteristics of the EHETS and summarize a couple key points learned from the research.

Section 1 Operational

1. The robustness of the EHETS system at base-case values is exhibited in Figures 1 & 2. TRAC's attempt to create a design that fell below the Ao limit required several iterations and adjustments to the operating ranges to arrive at a minimum acceptable operating level for the input variables.
2. TRAC implemented targeted system operating ranges to attain a minimum acceptable operating level for the tractor and trailer as shown in Figures 5 & 6. The team generated parameter estimates for all variables associated with the M1070A1 tractor and all the variables except recovery time and probability of correct diagnosis for M1000 trailer.
3. The EHETS systems exhibit diminishing gains in Ao as system reliability increased beyond 1200 hours for M1070A1 and 2000 hours for M1000.

4. Variable significance identification was conducted by using the Pearson coefficient correlation and constructing a first-order and second-order linear regression metamodel. The Pearson Coefficient Correlation quickly identified which variables affected Ao and the linear regression metamodels were constructed to further investigate the effect and observe variable behavior.

Section 2 Methodological

This section outlines the key lessons learning associated with sensitivity analysis and designed experimentation with military simulations:

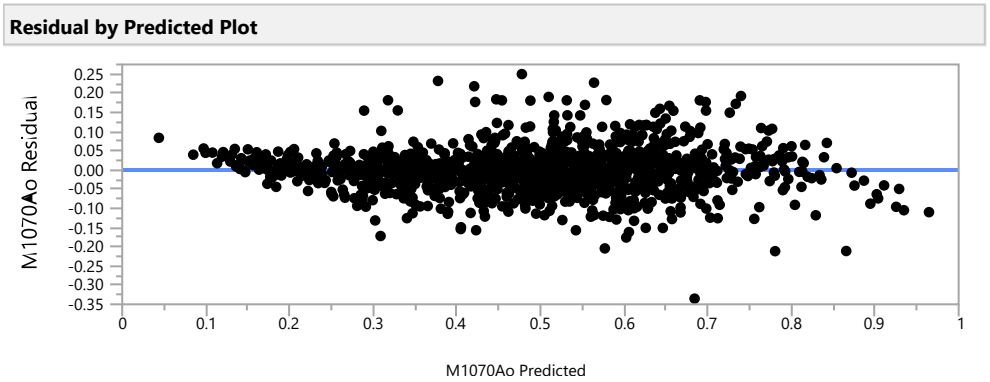
1. Sensitivity analysis exploring operating parameters for potential new military system is an iterative process. Analytic teams should not expect to 'get it right' without deliberately adjusting characteristics of their design and may have to reduce the scope of an effort if requisite data are not available.
2. Increased resolution of the space filling design and increase replications provided better models that were more capable of revealing insight into system characteristics, but also demanded more computational power and effective coding to implement.
3. Input variable ranges, even when informed by subject matter experts, can adversely affect an experimental design's ability to provide data capable of accurately modelling a given system's performance.
4. HPC enables massively parallel execution of simulations, and might be required even with a lightweight simulation like LBC.
5. Leveraging modern data science tools and packages like DPLYR and R drastically speed postprocessing times over legacy tools such as excel. These also lessen the chance of human error in the process and enable reproducible results.

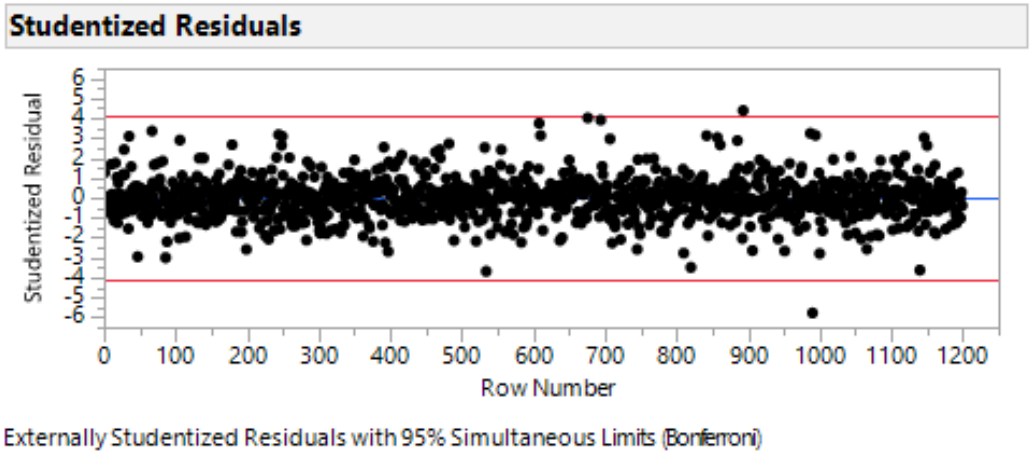
APPENDIX I

Below illustrate the residual plots for the M1070A1 and M1000.

JMP Generated Residual Plots

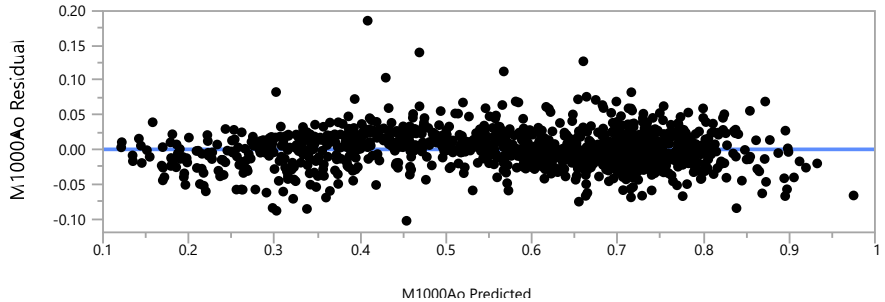
Response Surface Model for M1070A1



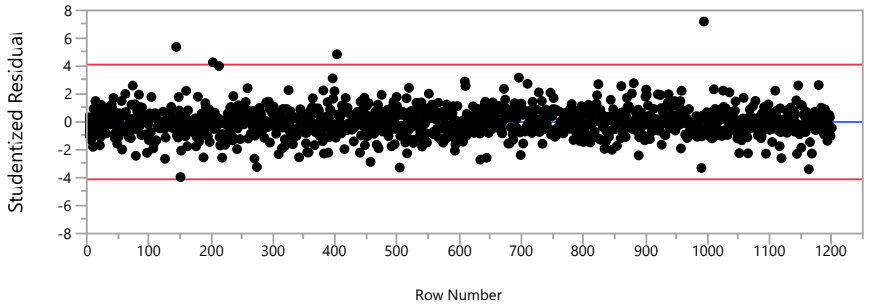


Response Surface Model for M1000

Residual by Predicted Plot



Studentized Residuals



Externally Studentized Residuals with 95% Simultaneous Limits (Bonferroni)

APPENDIX II

Overview

This appendix contains snippets of the commented R code used to implement the DOE and generate the XML documents.

DOE Scenario XML File Generation

This portion of the R code created the expanded DOE used to write the values to the XML documents.

```
##############################################
# this section takes the DoE and append the  #
# correct phrasing so it can run in LBC      #
##############################################
#save so you only have to change one variable to create file
ConvertFile =Reliability
#add exponential phrase to addFailure, add constant phrase to setTimeToRecover, setTimeToGetParts, and
setTimeToRepair
for (i in 1:nrow(ConvertFile[,c(1:6)])) {
  for(j in 1:ncol(ConvertFile[,c(1:6)])) {
    ConvertFile[i, j] <- paste0("Exponential(", ConvertFile[i, j],")") # for addFailure
  }

}

for (i in 1:nrow(ConvertFile[,c(11:12)])) {
  for(j in 1:ncol(ConvertFile[,c(11:12)])) {
    ConvertFile[i, j + 10] <- paste0("Exponential(", ConvertFile[i, j + 10],")")#for setTimeToRecover/Repair
  }
}
#remove the first column from the matrix and use the below code if you are importing csv
LBCInput <- (as.data.frame(read.csv(file="SameReliabilityValues.csv", head = TRUE,sep = ",")))
LBCInput =LBCInput[,-c(1) ]

#use this line if you are running the program without an external file
LBCInput =ConvertFile


#move (setTimeToGetParts.M1070A1) to the proper position.
LBCInput =LBCInput[c(1:9, 11, 10, 12)]

#duplicate the columns to make the allNodes dataframe
a = LBCInput[,c(1)]
addFailureM1070A1 =do.call(cbind, replicate(114, as.matrix(a), simplify = FALSE))

b = LBCInput[,c(2)]
addFailureM1000 =do.call(cbind, replicate(114, as.matrix(b), simplify = FALSE))
```

```
c = LBCInput[, c(3)]
setTimeToRecoverM1070A1 =do.call(cbind, replicate(114, as.matrix(c), simplify = FALSE))

d = LBCInput[, c(4)]
setTimeToRecoverM1000 =do.call(cbind, replicate(114, as.matrix(d), simplify = FALSE))

e = LBCInput[, c(5)]
setTimeToRepairM1070A1 =do.call(cbind, replicate(114, as.matrix(e), simplify = FALSE))

f = LBCInput[, c(6)]
setTimeToRepairM1000 =do.call(cbind, replicate(114, as.matrix(f), simplify = FALSE))

g = LBCInput[, c(7)]
addRepairAssetM1070A1 =do.call(cbind, replicate(114, as.matrix(g), simplify = FALSE))

h = LBCInput[, c(8)]
addRepairAssetM1000 =do.call(cbind, replicate(114, as.matrix(h), simplify = FALSE))

i = LBCInput[, c(9,10)]
probGetPartsM1070A1 =do.call(cbind, replicate(114, as.matrix(i), simplify = FALSE))

h = LBCInput[, c(11:12)]
probGetPartsM1000 =do.call(cbind, replicate(114, as.matrix(h), simplify = FALSE))

#combine all dataframes into one
AllNodes <-cbind(addFailureM1070A1, addFailureM1000, setTimeToRecoverM1070A1,
                 setTimeToRecoverM1000, setTimeToRepairM1070A1, setTimeToRepairM1000,
                 addRepairAssetM1070A1, addRepairAssetM1000, probGetPartsM1070A1,
                 probGetPartsM1000)
```

This portion extracted the values, and faults from the XML input file

```
##############################################
# this section creates the nodes             #
#                                            #
##############################################

#extract the value of numReplications
numReplications = getNodeSet(EHETS, "//LBCAssembly[@numReplications]")

#find all nodes with the value attribute
ValueSpot=getNodeSet(EHETS, "//@value") #<-this is just the actual value

nodesWithValue=getNodeSet(EHETS, "//*[@value]") #<- this gets the nodes

numNodes=length(nodesWithValue)

numNodesName=character()
#############################################################################################

### Create a table  With Key information
## Node # | Name  | SystemName | SystemID | FailureSeverity | FailureType | value |

nodeNum = integer()
name = character()
```

```
value = character()
refId = character()
SystemName = character()
SystemID = character()
FailureSeverity = character()
FailureType= character()
##############################################################################################

##############################################
# this section extracts the data from the xml #
# so a dataframe can be created               #
##############################################

#this loop will return all required information from the parent nodes in reference to the attribute value name
for (i in 1:numNodes){
  parent = xmlParent(nodesWithValue[[i]])
  grandParent=xmlParent(xmlParent(nodesWithValue[[i]]))
  GreatGrandParent =xmlParent(xmlParent(xmlParent(nodesWithValue[[i]])))
  nodeNum[i] = i
  name[i] = xmlAttrs(parent) #gets the methodName of the parent node in reference to the attribute value
                             name

  if (name[i]== 'addFailure'){ # this conditional statement is necessary because the structure is
                                 different from the other methodNames

    #use the name to find the system information
    sysnode1=getNodeSet(grandParent, paste0 ('.//', names(grandParent)[ 1], ''))

    #extract the systemName and Id
    refId=xmlAttrs(sysnode1[[1]])['refId']
    SystemName[i]=word(refId, 1, sep='\\.') #first part of refID
    SystemID[i] =word(refId, 2, sep='\\.') #second part is ID
     #extract the Failure Severity and Type
    failNode=getNodeSet(parent, paste0('.//', names(parent)[ 1], ''))
    FailRefId=xmlAttrs(failNode[[1]]) ['refId']
    FailureSeverity[i]=word(FailRefId, 1, sep='\\-') #first part of refID
    FailureType[i]=word(FailRefId, 2, sep='\\-') #second part is ID
    #extract the value
    valueNode = getNodeSet(parent, ".//*[@value]")
    value[i]=xmlAttrs(valueNode[[1]])['value']
  }
  else if(name[i]== 'addRepairAsset'){ # this conditional statement is necessary because the structure is
                                       different from the other methodNames

    #use the name to find the refID of the system
    sysnode2=getNodeSet(GreatGrandParent, paste0('.//', names(GreatGrandParent)[ 1], ''))

    #extract the systemName and Id
    refId=xmlAttrs(sysnode2[[1]]) ['refId']
    SystemName[i]=word(refId, 1, sep='\\.') #first part of refID
    SystemID[i] =word(refId, 2, sep='\\.') #second part is ID
     #extract the Failure Severity and Type
    failNode=getNodeSet(GreatGrandParent, paste0('.//', names(GreatGrandParent)[2], ''))
    FailRefId=xmlAttrs(failNode[[1]]) ['refId']
    FailureSeverity[i]=word(FailRefId, 1, sep='\\-') #first part of refID
```

```
  FailureType[i]=word(FailRefId, 2, sep='\\-') #second part is ID
  #extract the value
  valueNode = getNodeSet(parent, ".//*[@value]")
  value[i]=xmlAttrs(valueNode[[1]])['value']
 }
 else{

  #use the name to find the system information
  sysnode=getNodeSet(parent, paste0('.//', names(parent)[ 1], ''))
   #extract the systemName and Id
  refId=xmlAttrs(sysnode[[1]]) ['refId']
  SystemName[i]=word(refId, 1, sep='\\.') #first part of refID
  SystemID[i] =word(refId, 2, sep='\\.') #second part is ID
   #extract the failure information
  failNode=getNodeSet(parent, paste0('.//', names(parent)[ 2], ''))
  FailRefId=xmlAttrs(failNode[[1]]) ['refId']
  FailureSeverity[i]=word(FailRefId, 1, sep='\\-') #first part of refID
  FailureType[i]=word(FailRefId, 2, sep='\\-') #second part is ID
   #extract the value
  valueNode = getNodeSet(parent, ".//*[@value]")
  value[i]=xmlAttrs(valueNode[[1]]) ['value']
 }
}
```

This portion of the code generated the XML and saved them based on their design row number. You will have to create a data frame containing the values you extracted from the XML document, in the code below it is called the NameFilter. Also, if you want to change the number of replications per design point you will have to create a data frame to hold those values and that data frame is called numReplications.

```
##############################################
# this section grabs the data from the       #
# NameFilter and numReplications dataframe   #
# and writes it to the xml file and saves    #
# them based on their design point row num   #
##############################################

#used to loop through rows and columns of the Design
for (DesignRow in seq(nrow(AllNodes))){

  for (i in seq(ncol(AllNodes))){
#identify the node containing the value and replace it with the value contained in the Design
   index = NameFilter$nodeNum[k]
   xmlAttrs(nodesWithValue[[index]])<-c(value=paste0(AllNodes[DesignRow,i]))

      k= ifelse(k=='FilterRow', 1, k+ 1)

  }

  #identify the node containing the number of iterations and replace required iteration per design point,
   uncomment this line if you want to use it
  #xmlAttrs(numReplications[[1]])<-
```

```
                                        c(numReplications=paste0(DesignPointReplications[DesignRow,PointCol]))

        #access only the //SimEntityDataLogger that contains the file attribute name
        SaveFileNodes=getNodeSet(EHETS,"//SimEntityDataLogger[@file]")
        for (j in seq(length(SaveFileNodes))){

        loggerName=xmlAttrs(SaveFileNodes[[j]])['propertyName'] #cycles through to capture the name of the
                                                                csv file
        xmlAttrs(SaveFileNodes[[j]])=c(file=paste0(Dir2, 'design_output_',DesignRow,'_', loggerName, '.csv'))
          #appends the correct filename to the csv file in the XML document

        }

        cat(saveXML(EHETS), file=paste0(Dir,"design_point_", DesignRow  ,".xml")) #saves the filename based
                                                                on the design point number
        line=paste0("java -jar dist\\LBCRunner.jar ","\"",  Dir,"design_point_", DesignRow  ,".xml","\"") #
                                                        creates the command prompt line to run the files
        line1=paste0("java -jar dist\\lib\\LBCAnalyzer.jar  ","\"",  Dir,"design_point_", DesignRow  ,".xml","\"")
          #creates the command prompt line to run analyzer

        write(line,file=paste0(RunnerDir, "Runner.bat"),append=TRUE) #writes all of the files to a batch file to
                                                                run in LBC
        write(line1,file=paste0(RunnerDir, "AnalyzerSen.bat"),append= TRUE) #writes all of the files to a batch
                                                                file to analyze in LBC

}
```

APPENDIX III

Overview

This appendix contains snippets of the commented R code used to execute the DOE and perform the post-processing of the data.

DOE Execution File

Once the XML files are created they can be executed using LBC. This can be accomplished by using the batch file created from the code or if you choose to have a self-contained program that creates the DOE, generates and executes the XML document, you will use the below code. This code requires additional R files that contains the functions for each step.

```
### This Script Runs All The Functions Necessary to Run an LBC Experiment Using R###
###
### Experiments will be run in parallel using R's parallel function
###
### All of the required source files and the LBCv5 folder must be in the working directory with this file.
###
### MAKE SURE NO FILE PATHS HAVE SPACES IN THEM!!!

### Load DoE Generation Functions
# Functions here include loading base case XML, creating experimental design, and generating XML scenarios for execution
source('DoEGeneration.r')  # Loads DoE Generator Function (newDOE) based on 'DiceDesign' NOLH function.
source('GenXMLs.r') # Loads function to take design and base XML (makeXMLs) to create required number of XMLs to run
source('runDoE.r') #Loads function that runs DoE in parallel.
source('postProcessor.R') #Loads Postprocessing functions (right now just collects Availability files)

#########
######### Required packages - dplyr, xml, cowplot, DiceDesign, data.table, ggplot2, stringr, tidyr
#########

# Step 1 - Load Base Case and Generate DoE

## find input XML (input scenario)
inputXML<-file.choose()

## Generate Experimental Design
# Load Parameters (1st col variable, then max and min values for now, need to add distribution)
# format: variable | mins | maxes
#
designloc<-"designtest.csv"
#designloc<-choose.files()  #uncomment to select another design
parameters<-read.csv(designloc)

design<-newDOE(parameters)
```

```
## Generate XMLs in 'workingD'/LBCv5/data/xmls/
## WARNING - WILL OVERWRITE EXISTING xml scenario FILES
makeXMLs(inputXML,design)

## Get list of XMLs to run

xmlList<-list.files(path = paste0(getwd(),"/LBCv5/data/xmls/"))
xmlList<-list.files(choose.dir())
## Run the DoE
oldwd<-getwd() # must change directory to LBC containing DiR before running.
setwd(paste0(getwd(),"/LBCv5/"))

# Run experiment in parallel - put all output files in output directory
## WARNING - WILL OVERWRITE EXISTING OUTPUT FILES
runExperiment(xmlList)

setwd(oldwd) # set back to base wd.

## Get Availability

availability<-getAoFiles()

availability<-getAoFiles(choose.dir())  #uncomment if you'd like to select another folder

## Add downtime column and prepare data

availability<-calcDownTime(availability)

## Some Simple visualizations for availability.

source('AvailabilityAnalysis.r', local=TRUE)  #Loads visualization and analysis Function


variancebyplot

byDesignM1000Ao
byDesignM1070Ao

plot_grid(PercTimeDownPlot, numberofOccurencesPlot, timedownPerOccurencePlot, align='v',
        ncol=1)
```

R Functions necessary for the above script to work

```
##############################################################################################
##################       Function to create new set of DOE     ##########################
# source('DoEGeneration.r') #
##############################################################################################
library(DiceDesign)

newDOE <- function(DesignParameters){

  blank <- nolhDesign(nrow(DesignParameters))
  # Code found at
```

```
  # https://rdrr.io/cran/DiceDesign/man/nolhDesign.html

  blank<-blank$design

  # Create a data frame of ranges for each factor
  design<- data.frame(matrix(ncol = ncol(blank), nrow = nrow(blank)))
  for (i in 1:nrow(DesignParameters)){
    design[,i] <- DesignParameters$mins[i]+(DesignParameters$maxes[i] - DesignParameters$mins[i])*blank[,i]
  }

  names(design)<-as.character(DesignParameters[,1])

  return(design)
}


library (parallel)

runExperiment<-function(xmlList){

  no_cores<-detectCores()

  print(paste("detected", no_cores, "cores." ))

  no_cores=no_cores-2

  if (no_cores<1){no_cores=1}

  cl <- makeCluster(no_cores)

  print(paste("Cluster initiated, running on", no_cores, "cores."))

  start_time = Sys.time()

runTheDoE<-function(ListThing){
  system2("java", args = c("-jar", "dist\\LBCRunner.jar", paste0(getwd(),"/data/xmls/",ListThing)))
}

  parLapply(cl, xmlList,
        runTheDoE)

  stopCluster(cl)
  end_time = Sys.time()
  total = end_time - start_time

print(paste0("Experiment complete, cluster stopped, it took ",total, " minutes to run."))

}

##############################################################################################
##################       Function to perform post-processing of the data        ##############################
# source('postProcessor.R') #
##############################################################################################

## Get all Availability Output Files
```

```
library(data.table)
library(stringr)
library(dplyr)
library("tidyr")


## Get Ao files
## This function is specific to availability and collects availability output files from the output directory
## It has an optional argument (directory) where you can point it to gather files other than in the default output
directory.

getAoFiles<- function (directory){

  if (missing(directory)){
    directory=paste0(getwd()," /LBCv5/output/")
  } else {
    directory=paste0(directory, '/')
  }

  fileList<-list.files(path = directory)

  fileList<-fileList[str_detect(fileList, 'available')]

  df=fread(paste0(directory,fileList[1]))
  df$Source.Name=fileList[1]

  if (length(fileList)> 1){
    for (i in 2:length(fileList)){
    df1=fread(paste0(directory,fileList[i]))
    df1$Source.Name=fileList[i]
    df=rbind(df,df1)
  }
  }
  df=df %>% filter(eventName!='Run')
  return (df)
}

calcDownTime<- function (df){

  #calcDownTime calculates downtime based on scenario time for up/down events
  #It also preprocesses this data by adding/removing columns to enable visualizations

  #This function is used below to calculation downtime

  fillDownTime= function (time,lagtime,totalTime,event,nextEvent){

    result=vector('numeric',length(time))
      for (i in 1:length(time)){
      if ((event[i]=="ReturnToDuty")){
        result[i]=time[i]-lagtime[i]
      } else if (event[i]=="MakeUnavailable" & is.na(nextEvent[i])){
        result[i]=totalTime-time[i]
      } else {
        result[i]=0
      }
```

```
  }
  return (result)
 }


 #some variables for later calculations
 totaltime=2160
 totalvehicles=96

 #This section prepares the data, and calculates downtime

 df=df %>%
  mutate(fault=str_split_fixed(data,'\\|', 2)[, 1],
       truck_type=str_split_fixed(data, '\\|', 2)[, 2],
       design=as.integer(str_split_fixed(Source.Name,'\\_', 4)[, 3]))


 df=df %>% group_by(design,replication,entityName) %>%
  mutate(downTime=fillDownTime(time, lag(time), totaltime, eventName, lead(eventName)))

 #get rid of unneeded data columns
 df$Source.Name=NULL
 df$data= NULL
 return (df)
}

#################################################################################################
##################       Function to perform visualization of the results        ##########################
# source('AvailabilityAnalysis.R')#
#################################################################################################

df=availability
library(cowplot)

#By rep results for Ao
#some variables for later calculations
totaltime=2160
totalvehicles=96

M1000Ao=df %>% ungroup() %>% group_by(design, replication) %>%
 filter(truck_type=='COMBATHET' & substr(entityName, 1, 5)=='M1000') %>%
 summarise(Ao=(((totaltime*totalvehicles)-sum(downTime))/(totaltime*totalvehicles)))

M1070Ao=df %>% ungroup() %>% group_by(design, replication) %>%
 filter(truck_type=='COMBATHET' & substr(entityName,1, 5)=='M1070') %>%
 summarise(Ao=(((totaltime*totalvehicles)-sum(downTime))/(totaltime*totalvehicles)))

#Convert by Rep results into by Design

byDesignM1000Ao=M1000Ao %>% group_by(design) %>% summarize(meanAo=mean(Ao)) %>%
arrange(design)

byDesignM1070Ao=M1070Ao %>% group_by(design) %>% summarize(meanAo=mean(Ao)) %>%
arrange(design)
```

```
#Some interesting Summary Statistics

totalDownTime=sum(df$downTime) #total of all downtime for experiment - used to calculate Perc of downtime

TimeDownbyFault = df %>%   ungroup() %>%
  filter(truck_type=='COMBATHET') %>% mutate(SystemName=str_split_fixed(`entityName`, "\\.",n=4)[, 1],
                                SystemID=str_split_fixed(`entityName`, "\\.",n=4)[, 2]) %>%
  group_by(fault,SystemName) %>% summarize(PercentofTotalDowntime=sum(downTime)/totalDownTime,
                                numberOfOccurrences=sum(eventName=="MakeUnavailable"),

AverageTimeDownPerOccurrence=sum(downTime)/sum(eventName=="MakeUnavailable")) %>%
  arrange(desc(PercentofTotalDowntime))


SDDownbyFault<-df %>%    ungroup() %>%
  filter(truck_type=='COMBATHET') %>% mutate(SystemName=str_split_fixed(`entityName`, "\\.",n=4)[, 1],
                                SystemID=str_split_fixed(`entityName`, "\\.",n=4)[, 2]) %>%
  group_by(fault,SystemName,replication) %>%
  summarize(totalDown=sum(downTime)) %>% ungroup() %>%
  group_by(fault, SystemName) %>%
  summarize(sdofTotalDowntime=sd(totalDown))


PercTimeDownPlot=TimeDownbyFault %>% ggplot(aes(x=reorder(fault, -
PercentofTotalDowntime),y=PercentofTotalDowntime, fill=SystemName))+
  geom_bar(stat= "identity", width=.5, position = "dodge") +
  labs(title= "Percent Down Time by Fault",
       x ="Fault", y = "%Downtime")

numberofOccurrencesPlot=TimeDownbyFault %>% ggplot(aes(x=reorder(fault, -
PercentofTotalDowntime),y=numberOfOccurrences, fill=SystemName))+
  geom_bar(stat= "identity", width=.5, position = "dodge")  +
  labs(title= "Total Occurrences by Fault",
       x ="Fault", y = "Occurrences")

timedownPerOccurrencePlot=TimeDownbyFault %>% ggplot(aes(x=reorder(fault, -
PercentofTotalDowntime),y=AverageTimeDownPerOccurrence, fill=SystemName))+
  geom_bar(stat= "identity", width=.5, position = "dodge")  +
  labs(title= "Mean Time Down Per Occurrences ",
            x ="Fault", y = "Mean Time Down Per")

variancebyplot=SDDownbyFault %>% ggplot(aes(x=fault,y=sdofTotalDowntime, fill=SystemName))+
  geom_bar(stat= "identity", width=.5, position = "dodge")+
  labs(title= "Variance of Total Down Time Per Fault",
            x ="Fault", y = "Variance of Total Down Time Per Fault")

variancebyplot

plot_grid(PercTimeDownPlot, numberofOccurrencesPlot, timedownPerOccurrencePlot, align='v',
          ncol=1)
```

APPENDIX IV

JMP Script for Model: this defines the structure of the DOE for the JMP software.

```
Neural(
	Y( :Y ),
	X(
		:System Reliability M1070A1,
		:System Reliability M1000,
		:Time to Recover M1070A1,
		:Time to Recover M1000,
		:Time to Repair M1070A1,
		:Time to Repair M1000,
		:Mechanics M1070A1,
		:Mechanics M1000,
		:Prob of Correct Diagnosis M1070A11,
		:Prob of Correct Diagnosis M1000,
		:Time to Get Parts M1070A1,
		:Time to Get Parts M1000
	)
)
```

JMP Script for DOE generation.

```
DOE(
	Space Filling Design,
	{Add Response( Maximize, "Y", ., ., . ),
	Add Factor( Continuous, 240.08274, 1360.46886, "System Reliability M1070A1", 0 ),
	Add Factor( Continuous, 246.660195, 2250, "System Reliability M1000", 0 ),
	Add Factor( Continuous, 2, 250, "Time to Recover M1070A1", 0 ),
	Add Factor( Continuous, 2, 480, "Time to Recover M1000", 0 ),
	Add Factor( Continuous, 1, 100, "Time to Repair M1070A1", 0 ),
	Add Factor( Continuous, 1, 364, "Time to Repair M1000", 0 ),
	Add Factor( Continuous, 1, 9, "Mechanics M1070A1", 0 ),
	Add Factor( Continuous, 1, 9, "Mechanics M1000", 0 ),
	Add Factor( Continuous, 0.19, 1, "Prob of Correct Diagnosis M1070A11", 0 ),
	Add Factor( Continuous, 0.19, 1, "Prob of Correct Diagnosis M1000", 0 ),
	Add Factor( Continuous, 0.5, 242, "Time to Get Parts M1070A1", 0 ),
	Add Factor( Continuous, 0.5, 242, "Time to Get Parts M1000", 0 ),
	Set Random Seed( 66423846 ), Space Filling Design Type( Latin Hypercube, 1200 ),
	Simulate Responses( 0 ), Set Run Order( Randomize ), Make Table}

)
```

JMP Script for Linear Regression Model for M1070A1

```
Fit Model(
	Y( :M1070Ao ),
	Effects(
		:System.Reliability.M1070A1,
		:Time.to.Recover.M1070A1,
		:Time.to.Repair.M1070A1,
		:Mechanics.M1070A1,
		:Prob.of.Correct.Diagnosis.M1070A11,
		:Time.to.Get.Parts.M1070A1
```

```
	),
	Personality( "Standard Least Squares" ),
	Emphasis( "Effect Screening" ),
	Run(
		Profiler(
			1,
			Confidence Intervals( 1 ),
			Term Value(
				System.Reliability.M1070A1( 800.3, Lock( 0 ), Show( 1 ) ),
				Time.to.Recover.M1070A1( 126, Lock( 0 ), Show( 1 ) ),
				Time.to.Repair.M1070A1( 50.5, Lock( 0 ), Show( 1 ) ),
				Mechanics.M1070A1( 5, Lock( 0 ), Show( 1 ) ),
				Prob.of.Correct.Diagnosis.M1070A11( 0.595, Lock( 0 ), Show( 1 )
),
				Time.to.Get.Parts.M1070A1( 121.25, Lock( 0 ), Show( 1 ) )
			)
		),
		:M1070Ao << {Summary of Fit( 0 ), Analysis of Variance( 0 ),
		Parameter Estimates( 1 ), Effect Details( 0 ), Lack of Fit( 0 ),
		Sorted Estimates( 0 ), Plot Actual by Predicted( 1 ), Plot Regression( 0 ),
		Plot Residual by Predicted( 1 ), Plot Studentized Residuals( 1 ),
		Plot Effect Leverage( 0 ), Plot Residual by Normal Quantiles( 0 ),
		Box Cox Y Transformation( 1 )}
	)
)
```

JMP Script for Response Surface Model for M1070A1

```
Fit Model(
	Y( :M1070Ao ),
	Effects(
		:System.Reliability.M1070A1 & RS,
		:Time.to.Recover.M1070A1 & RS,
		:Time.to.Repair.M1070A1 & RS,
		:Mechanics.M1070A1 & RS,
		:Prob.of.Correct.Diagnosis.M1070A11 & RS,
		:Time.to.Get.Parts.M1070A1 & RS,
		:System.Reliability.M1070A1 * :System.Reliability.M1070A1,
		:System.Reliability.M1070A1 * :Time.to.Recover.M1070A1,
		:Time.to.Recover.M1070A1 * :Time.to.Recover.M1070A1,
		:System.Reliability.M1070A1 * :Time.to.Repair.M1070A1,
		:Time.to.Recover.M1070A1 * :Time.to.Repair.M1070A1,
		:Time.to.Repair.M1070A1 * :Time.to.Repair.M1070A1,
		:System.Reliability.M1070A1 * :Mechanics.M1070A1,
		:Time.to.Recover.M1070A1 * :Mechanics.M1070A1,
		:Time.to.Repair.M1070A1 * :Mechanics.M1070A1,
		:Mechanics.M1070A1 * :Mechanics.M1070A1,
		:System.Reliability.M1070A1 * :Prob.of.Correct.Diagnosis.M1070A11,
		:Time.to.Recover.M1070A1 * :Prob.of.Correct.Diagnosis.M1070A11,
		:Time.to.Repair.M1070A1 * :Prob.of.Correct.Diagnosis.M1070A11,
		:Mechanics.M1070A1 * :Prob.of.Correct.Diagnosis.M1070A11,
		:Prob.of.Correct.Diagnosis.M1070A11 * :Prob.of.Correct.Diagnosis.M1070A11,
		:System.Reliability.M1070A1 * :Time.to.Get.Parts.M1070A1,
		:Time.to.Recover.M1070A1 * :Time.to.Get.Parts.M1070A1,
		:Time.to.Repair.M1070A1 * :Time.to.Get.Parts.M1070A1,
		:Mechanics.M1070A1 * :Time.to.Get.Parts.M1070A1,
		:Prob.of.Correct.Diagnosis.M1070A11 * :Time.to.Get.Parts.M1070A1,
		:Time.to.Get.Parts.M1070A1 * :Time.to.Get.Parts.M1070A1
	),
```

```
	Personality( "Standard Least Squares" ),
	Emphasis( "Effect Screening" ),
	Run(
		Profiler(
			1,
			Confidence Intervals( 1 ),
			Term Value(
				System.Reliability.M1070A1( 800.3, Lock( 0 ), Show( 1 ) ),
				Time.to.Recover.M1070A1( 126, Lock( 0 ), Show( 1 ) ),
				Time.to.Repair.M1070A1( 50.5, Lock( 0 ), Show( 1 ) ),
				Mechanics.M1070A1( 5, Lock( 0 ), Show( 1 ) ),
				Prob.of.Correct.Diagnosis.M1070A11( 0.595, Lock( 0 ), Show( 1 )
),
				Time.to.Get.Parts.M1070A1( 121.25, Lock( 0 ), Show( 1 ) )
			)
		),
		:M1070Ao << {Summary of Fit( 0 ), Analysis of Variance( 0 ),
		Parameter Estimates( 1 ), Effect Details( 0 ), Lack of Fit( 0 ),
		Sorted Estimates( 1 ), Plot Actual by Predicted( 1 ), Plot Regression( 0 ),
		Plot Residual by Predicted( 1 ), Plot Studentized Residuals( 1 ),
		Plot Effect Leverage( 0 ), Plot Residual by Normal Quantiles( 0 ),
		Box Cox Y Transformation( 1 )}
	)
)
```

JMP Script for Linear Regression Model for M1000

```
Fit Model(
	Y( :M1000Ao ),
	Effects(
		:System.Reliability.M1000,
		:Time.to.Recover.M1000,
		:Time.to.Repair.M1000,
		:Mechanics.M1000,
		:Prob.of.Correct.Diagnosis.M1000,
		:Time.to.Get.Parts.M1000
	),
	Personality( "Standard Least Squares" ),
	Emphasis( "Effect Screening" ),
	Run(
		Profiler(
			1,
			Confidence Intervals( 1 ),
			Term Value(
				System.Reliability.M1000( 1644.4, Lock( 0 ), Show( 1 ) ),
				Time.to.Recover.M1000( 241, Lock( 0 ), Show( 1 ) ),
				Time.to.Repair.M1000( 182.5, Lock( 0 ), Show( 1 ) ),
				Mechanics.M1000( 5, Lock( 0 ), Show( 1 ) ),
				Prob.of.Correct.Diagnosis.M1000( 0.595, Lock( 0 ), Show( 1 ) ),
				Time.to.Get.Parts.M1000( 262.75, Lock( 0 ), Show( 1 ) )
			)
		),
		:M1000Ao << {Summary of Fit( 0 ), Analysis of Variance( 0 ),
		Parameter Estimates( 1 ), Effect Details( 0 ), Lack of Fit( 0 ),
		Sorted Estimates( 0 ), Plot Actual by Predicted( 1 ), Plot Regression( 0 ),
		Plot Residual by Predicted( 1 ), Plot Studentized Residuals( 1 ),
		Plot Effect Leverage( 0 ), Plot Residual by Normal Quantiles( 0 ),
		Box Cox Y Transformation( 1 )}
	)
```

```
)
```

JMP Script for Response Surface Model for M1000

```
Fit Model(
	Y( :M1000Ao ),
	Effects(
		:System.Reliability.M1000 & RS,
		:Time.to.Recover.M1000 & RS,
		:Time.to.Repair.M1000 & RS,
		:Mechanics.M1000 & RS,
		:Prob.of.Correct.Diagnosis.M1000 & RS,
		:Time.to.Get.Parts.M1000 & RS,
		:System.Reliability.M1000 * :System.Reliability.M1000,
		:System.Reliability.M1000 * :Time.to.Recover.M1000,
		:Time.to.Recover.M1000 * :Time.to.Recover.M1000,
		:System.Reliability.M1000 * :Time.to.Repair.M1000,
		:Time.to.Recover.M1000 * :Time.to.Repair.M1000,
		:Time.to.Repair.M1000 * :Time.to.Repair.M1000,
		:System.Reliability.M1000 * :Mechanics.M1000,
		:Time.to.Recover.M1000 * :Mechanics.M1000,
		:Time.to.Repair.M1000 * :Mechanics.M1000,
		:Mechanics.M1000 * :Mechanics.M1000,
		:System.Reliability.M1000 * :Prob.of.Correct.Diagnosis.M1000,
		:Time.to.Recover.M1000 * :Prob.of.Correct.Diagnosis.M1000,
		:Time.to.Repair.M1000 * :Prob.of.Correct.Diagnosis.M1000,
		:Mechanics.M1000 * :Prob.of.Correct.Diagnosis.M1000,
		:Prob.of.Correct.Diagnosis.M1000 * :Prob.of.Correct.Diagnosis.M1000,
		:System.Reliability.M1000 * :Time.to.Get.Parts.M1000,
		:Time.to.Recover.M1000 * :Time.to.Get.Parts.M1000,
		:Time.to.Repair.M1000 * :Time.to.Get.Parts.M1000,
		:Mechanics.M1000 * :Time.to.Get.Parts.M1000,
		:Prob.of.Correct.Diagnosis.M1000 * :Time.to.Get.Parts.M1000,
		:Time.to.Get.Parts.M1000 * :Time.to.Get.Parts.M1000
	),
	Personality( "Standard Least Squares" ),
	Emphasis( "Effect Screening" ),
	Run(
		Profiler(
			1,
			Confidence Intervals( 1 ),
			Term Value(
				System.Reliability.M1000( 1644.4, Lock( 0 ), Show( 1 ) ),
				Time.to.Recover.M1000( 241, Lock( 0 ), Show( 1 ) ),
				Time.to.Repair.M1000( 182.5, Lock( 0 ), Show( 1 ) ),
				Mechanics.M1000( 5, Lock( 0 ), Show( 1 ) ),
				Prob.of.Correct.Diagnosis.M1000( 0.595, Lock( 0 ), Show( 1 ) ),
				Time.to.Get.Parts.M1000( 262.75, Lock( 0 ), Show( 1 ) )
			)
		),
		:M1000Ao << {Summary of Fit( 0 ), Analysis of Variance( 0 ),
		Parameter Estimates( 1 ), Effect Details( 0 ), Lack of Fit( 0 ),
		Sorted Estimates( 0 ), Plot Actual by Predicted( 1 ), Plot Regression( 0 ),
		Plot Residual by Predicted( 1 ), Plot Studentized Residuals( 1 ),
		Plot Effect Leverage( 0 ), Plot Residual by Normal Quantiles( 0 ),
		Box Cox Y Transformation( 1 )}
	)
)
```

REFERENCES

Cioppa, T. M. (2009). *Efficient Nearly Orthogonal and Space-filling Experimental Designs for High-dimensional Complex Models.* Monterey, CA: Calhoun.

Dupuy, D. (2015). DiceDesign. *Journal of Statistical Software, 65*(11), 1-38. Retrieved from http://www.jstatsoft.org/v65/i11/

Law, A. (2007). *Simulation, Modeling, and Analysis.* New York: McGraw-Hill.

Pearson Correlation Coefficient. (2019, May 28). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

Sacks, J., Welch, W., Mitchell, T., & Wynn, H. (1989). Design and Analysis of Computer Experiments. *Statistical Science, 4*(4), 409-435.

SAS Institute. (2019, May 28). *Response Surface Experiments*. Retrieved from JMP: http://www.jmp.com/support/help/14-2/response-surface-experiments.shtml#

Wackerly, D. D., Mendenhall, W., & Scheaffer, R. L. (2008). *Mathematical Statistics wIth Applications.* Belmont: Brooks/Cole.

Wade, B. M. (2017, March). A Framework for the Optimization of Doctrine and Systems in Army Air Defense Units Against a Complex Attack of Ballistic and Cruise Missiles Using Predictive Models of Stochastic Computer Simulations. Atlanta, GA, USA.