2019-06

CONDITIONS-BASED MAINTENANCE THROUGH AUTONOMOUS LOGISTICS

Whitaker, Michael

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CONDITIONS-BASED MAINTENANCE THROUGH AUTONOMOUS LOGISTICS

by

Michael Whitaker

June 2019

Thesis Advisor:	Gurminder Singh
Co-Advisor:	Lyn R. Whitaker
Second Reader:	Arijit Das

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE CONDITIONS-BASED MAINTENANCE THROUGH AUTONOMOUS LOGISTICS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Michael Whitaker	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Currently, operators and maintainers cull though numerous electronic reports, display boards, and historical maintenance records to determine and plan for maintenance activities for equipment. However, in recent years, emerging technologies such as the Internet-of-Things, big data analysis, and low-cost sensors and actuators have enabled applications that were not possible previously. From these developments, information that was once unavailable is now accessible through embedded sensors and actuators, providing real-time condition monitoring of complicated machinery. This thesis demonstrates the use of inexpensive COTS hardware devices and open-source software to develop an automated data collection architecture and a data processing framework to implement a preventative maintenance approach for the Marine Corps Medium Tactical Vehicle Replacement (MTVR). Data processing techniques were used to convert raw sensor data collected from on-board MTVR sensors into useable and measurable diagnostic data. Using statistical analysis based on a time series regression model, the diagnostic parameters that closely modeled engine operating conditions were chosen to predict engine usage characteristics of an MTVR engine. The thesis also describes a conditions-based maintenance policy that can be used to enhance preventative maintenance methods and decision support capabilities on Marine Corps ground equipment.

14. SUBJECT TERMS data science, single-board computers, logistics, MTVRS, CBM, conditions-based maintenance, sensors, Raspberry Pi, J1939, SAE	**15. NUMBER OF PAGES** 131
	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	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

CONDITIONS-BASED MAINTENANCE THROUGH AUTONOMOUS LOGISTICS

Michael Whitaker
Captain, United States Marine Corps
BS, Savannah State University, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2019

Approved by: Gurminder Singh
Advisor

Lyn R. Whitaker
Co-Advisor

Arijit Das
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Currently, operators and maintainers cull though numerous electronic reports, display boards, and historical maintenance records to determine and plan for maintenance activities for equipment. However, in recent years, emerging technologies such as the Internet-of-Things, big data analysis, and low-cost sensors and actuators have enabled applications that were not possible previously. From these developments, information that was once unavailable is now accessible through embedded sensors and actuators, providing real-time condition monitoring of complicated machinery. This thesis demonstrates the use of inexpensive COTS hardware devices and open-source software to develop an automated data collection architecture and a data processing framework to implement a preventative maintenance approach for the Marine Corps Medium Tactical Vehicle Replacement (MTVR). Data processing techniques were used to convert raw sensor data collected from on-board MTVR sensors into useable and measurable diagnostic data. Using statistical analysis based on a time series regression model, the diagnostic parameters that closely modeled engine operating conditions were chosen to predict engine usage characteristics of an MTVR engine. The thesis also describes a conditions-based maintenance policy that can be used to enhance preventative maintenance methods and decision support capabilities on Marine Corps ground equipment.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ALIS	Autonomic Logistics Information System
AO	Area of Operations
AOC	Air Operations Center
ARDL	Autoregressive Distributed Lag
CBM	Conditions-based Maintenance
CAN	Controller Area Network
DoD	Department of Defense
ECU	Electronic Control Units
eRM	Enterprise Remote Monitoring
GPS	Global Positioning System
HUMS	Health Usage and Management System
ICAS	Integrated Condition Assessment System
ISO	International Organization for Standardization
JSF	Joint Strike Fighter
MTVR	Medium Tactical Vehicle Replacement
OBD2	On-board Diagnostics
PGN	Parameter Group Number
RCM	Reliability-Centered Maintenance
SAE	Society of Automotive Engineers
SBCT	Stryker Brigade Combat Team
SEDCAS	SBCT Embedded Data Collection and Analysis System
SPN	Suspect Parameter Number
SQL	Structed Query Language
VMEP	Vibration Management Enhancement Program
PMAL	predictive maintenance alert loop

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would like to express my appreciation for all the support and assistance that has been provided to me throughout my time at Naval Postgraduate School. Next, I would like to thank the team and staff at Headquarters, Marine Corps, Department of Installations and Logistics for their funding, support, and guidance. I would like to thank the Marines of Inspector-Instructor Staff, San Jose, California, Combat Logistics Battalion 453 for allowing me access to their vehicles for testing. These Marines included Staff Sergeant Salvador Olveralayna, Staff Sergeant Ivenley Gourdet, and Major Bobby Bradford. Special thanks to Headquarters Battery, 11th Marines, 1st Marine Division, most especially Gunnery Sergeant Marlon Lewis for coordinating truck support, and Corporal Tyler Cosom, for driving the better part of six hours across Camp Pendleton to provide me with the data used for this research. I would like to thank my thesis advisors, Dr. Gurminder Singh and Dr. Lyn Whitaker. Dr. Whitaker, thank you for your valuable comments, suggestions, and statistical eye. Lastly, I would like to thank my amazing wife, Janay, for her unstinting support, patience, and beautiful smile over these past 24 months. Thank you, all.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The purpose of this thesis is to examine a conditions-based maintenance approach to support preventative maintenance policies for Marine Corps ground equipment. Specifically, this research examines implementing conditions-based maintenance from diagnostic data extracted from the Marine Corps Medium Tactical Vehicle Replacement (MTVR). The thesis will begin with a review of the major research questions, describe the importance and benefits of conditions-based maintenance, and present a brief literature review of conditions-based maintenance practices in the Department of Defense (DoD) and certain industry organizations. Using the diagnostic data extracted from the vehicle, this thesis will establish a CBM decision methodology by analyzing diagnostic data using regression analysis. Conclusions from this analysis will be drawn to demonstrate a CBM policy for use across Marine Corps ground equipment.

A. BENEFITS AND IMPORTANCE OF CBM

Marine Corps doctrine recognizes that "logistics is an integral part of war fighting [because it] provides the resources of combat power, brings those resources to the battle, and sustains them throughout the course of operations" [1, p. 1]. Until recently, it has been difficult to achieve and discover optimal maintenance procedures to support military operations because real-time information visibility on equipment conditions was inaccessible [2]. Military organizations relied on paper charts, display boards, and historical maintenance schedules to create maintenance policies and determine optimal maintenance actions. However, with emerging technologies brought forth by IoT devices, radio frequency identification (RFID), equipment sensors, interconnect networks, and computational power, information containing the status of equipment conditions and operations can be monitored and gathered for data analysis and decision support models [2]. To sustain combat power and compete in the war landscape of the 21st century, military organizations have begun to leverage the latest advances in logistics technology systems.

Military organizations have identified logistics data as a critical enabler for their future success. The Army recently published an update to its “Field Manual 3-0

Operations," describing ways to leverage today's logistic data and technology to fight the next adversary. Similarly, the Navy is leveraging data from logistic assessment systems integrated on its ships to find ways to sustain lethality and maintain sea power using data-driven methods aligned along National Defense Strategy pillars [3]. Also, the Air Force is working closely with industry partners, particularly Boeing and Lockheed Martin, to find ways and opportunities to increase fighter plane readiness through investment in logistics systems and software that specialize in data analytics. In the commercial sector, DoD industry partners are actively finding ways to optimize logistical distribution services and supply chains by using computational machines and computer software to prove and create effective logistics strategies. They are leveraging logistics information technology to plan the movement of vast fleets of trucks, aircraft, and ships to transport goods and services from one corner of the globe to the next in the way that reduces cost and increases efficiency in a global economy that is increasingly integrated.

This research explores a way to exploit the real-time logistic data collected from military vehicles and other equipment to enable predictive maintenance. This data provides information revealing insights that drive logistic and operational decisions. Predictive maintenance enables organizations to maximize performance and availability from their physical assets and revolutionizes traditional logistic strategies and supply paradigms [4].

As mentioned in the DoD Guidebook on Conditions-based Maintenance, "the Department of Defense is continually challenged to identify and meet warfighter expectations while making every effort to conduct cost effective sustainment operations" [5, p. 2]. Conditions-based maintenance (CBM) is a preventative maintenance approach that focuses on identifying failures before they occur and monitoring the condition of an item for improved life cycle sustainment. This thesis presents a brief overview of literature on CBM and an insight into a CBM approach as a maintenance strategy [6].

A traditional scheduled maintenance approach does not reveal the health or condition of equipment, whereas a CBM approach captures the experience of the life of a repair part. This benefit allows organizations to make real-time decisions and visualize readiness drivers based on information available. Another major benefit of a CBM approach is reduced downtime and maintenance cost. By identifying failures before they

occur, maintainers and logistics planners can eliminate progressive troubleshooting procedures, which reduces the number of intrusions into equipment, saving time and reducing risk of failure. With reactive or preventive maintenance, a massive inventory of spare parts is necessary because maintainers do not know what is needed until an asset is taken out of operation making repair parts difficult to forecast. Furthermore, predictive maintenance models built on historical data are not exact enough to provide the level of detail needed for accurate forecasting. However, logistic or maintenance-based computer software used to process, collect, and analyze maintenance data, enable CBM revealing relationships and correlations beyond historical maintenance records and day-to-day maintenance charts. A CBM approach enables the estimation of time-to-failure for each part, allowing the forecasting of only necessary parts, optimizing existing supply inventories. CBM also reveals the true cost of life cycle ownership by tracking the remaining life of each equipment component; therefore, increasing availability of parts by creating accurate demand signals, and the development of optimal upgrade and replacement plans based on the estimated life of equipment components. Put briefly, CBM extends the life of equipment while keeping cost low and ensuring high availability of parts. This thesis discusses a low-cost proof-of-concept CBM approach.

B. PROBLEM STATEMENT

This research aims to investigate a preventative maintenance approach to support predictive maintenance of Marine Corps ground equipment. The two key questions are summarized as follows:

- What is a good architecture for automated collection of sensor data that can be used to determine conditions necessary for maintenance actions on Marine Corps equipment?
- How can this collection of sensor data be used to provide in-depth maintenance analysis, improve quality assurance, and generate enhanced maintenance solutions for vehicle or weapon system life cycle support?

C. SCOPE AND LIMITATIONS

This thesis utilizes a low-cost commercial-off-the-shelf (COTS) device to extract real-time diagnostic data from an MTVR. The collected data is processed using an on-computer database file system. While this thesis presents a single use-case for CBM on one piece of equipment, this method can be exported and scaled across a variety of Marine Corps ground equipment. The COTS device simulates an industrial Internet of Things (IoT) device utilized to collect data from on-board equipment sensors. These devices then transfer the data collected into a central server or cloud via an interconnected network. Cloud and computerized maintenance management systems are represented by the on-computer database software. The collected data is then transformed to scaled engineering values where regression analysis is used to create a predictive model. This thesis aims to provide a proof-of-concept CBM methodology that can be transferred to large-scale computing cloud architectures where advanced machine learning algorithms can be applied. MTVRs are equipped with the industry standard bus wiring system to transmit diagnostic data. This plug-and-play capability means vehicles did not have to be modified to support this research.

There are three limitations of this research. First, due to the geographical separation of Naval Postgraduate School from Marine Corps operating forces, testing and validation of hardware used for data extraction was conducted at Combat Logistic Battalion 453, Combat Logistic Regiment 4, 4th Marine Logistics Group in San Jose, CA. Data from two MTVRs was collected for equipment testing and validation. The final diagnostics data was collected from one MTVR at 11th Marines, 1st Marine Division located in Camp Pendleton, CA. Second, data was collected in real-time and only the data available at the time of collection is utilized. Third, diagnostic data of equipment with known failures was not obtained. In practice, CBM is a maintenance policy that recommends performing maintenance only upon need. Typically, this policy is built from probabilistic models derived from the data collected from healthy equipment and equipment with failures. To make up for this limitation, this research modeled the known “normal” operating engine conditions and established warnings and triggers on operating ranges outside the “normal” to trigger maintenance actions.

D. ORGANIZATION OF THESIS

This thesis is organized into four subsequent chapters:

- Chapter II, Logistics in a Distributed Environment contains a brief discussion on logistic concepts and the need for CBM in a distributed environment, the history of conditions-based maintenance in the DoD, and its implementation and integration into current maintenance paradigms.

- Chapter III, Experimentation Methodology and Technical Approach contains a detailed discussion on the experimentation used to develop a prototype and methodology for data collection and retrieval, data analysis, and decision support capabilities for data sets of time series data to enable CBM.

- Chapter IV, Results, Analysis, and Limitations summarizes findings from the experimentation and implementation from Chapter III and discusses the capabilities and limitations of the proposed CBM probabilistic model.

- Chapter V, Conclusions and Recommendations for Future Study concludes the research report, identifying areas where additional research is recommended.

The thesis also includes four appendices: Appendix A contains Python source code on data cleaning and parsing of J1939 log data. Appendix B has R source code for data analysis and plot drawing. Appendix C is electronic control unit wiring diagram on the MTVR 23. The driving course for the MTVR is included in Appendix D, Figure 29.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LOGISTICS IN A DISTRIBUTED ENVIRONMENT

This chapter has four purposes. The first purpose is to briefly explain fundamental methods and practices of military logistics. Emphasis is placed on two Marine Corps doctrinal methods, "push" logistics and "pull" logistics. The second purpose is to identify the significance of computer integration into modern logistic methods to enhance logistic capabilities. This section explains how digitally enabled logistic creates opportunities for new approaches to maintenance. The third purpose is to identify and discuss how DoD organizations and commercial companies leverage digital technologies to increase reliability, sustain lethality, and enable predictive maintenance. The final purpose of this chapter is to discuss a business model to support a digital maintenance strategy.

A. LOGISTICS OVERVIEW

Reinhardt Jünemann, a leading logistics expert and scientist, describes the objective of logistics as "providing the right quantity of the right objects in the right place at the right time for the right price" [7, p. 11]. To achieve these objectives, the Marine Corps, uses a combination of two methods. The first method is an automatic push system wherein a resource is provided to a unit without any action or request from the receiving unit. In this method, resources are distributed and allocated among units based on planned schedules and formulas [8]. An example of this process is the prepositioning of certain repair parts or maintenance teams in front of a unit in the battle space in anticipation of the unit’s future requirements [8]. The second method is the demand-based pull method. This method is used when organizations who require support request replenishment and repair parts from the sourcing unit [8].

Each system provides advantages that the other system lacks. A logistical push system uses known and scheduled logistic requirements to position or deliver resources. This process allows units to have dependable support with little effort in planning. However, the downside of the push method is that a unit can easily become overburdened with excessive supplies and unnecessary parts. A pull logistic system allows for efficiency

and optimal logistic parts flow because units receive only the support that is needed. However, though efficient in terms of reducing the logistic burden, this system is vulnerable to the uncertainties of war as well as unpredictable consumption rates and resource limitations. Resources for these systems are naturally constrained due to fiscal shortfalls, material limitations, the unavailability of skilled laborers or manufacturers, or a combination of all three. These constraints force our logistic systems to rely on an impractical perfect balance of human-projected requirements and logistics planners' estimates as well as the availability of support vehicles and systems to distribute, manage, transport, handle, and deliver supplies [8].

B. THE NEED FOR COMPUTER INTEGRATION INTO LOGISTIC SYSTEMS FOR TODAY'S SUCCESS

Logistics plays in integral role in any military action, whether in war or peace [1]. Moreover, the complexities and uncertainties of war only exacerbate logistical constraints lessening the effectiveness of our logistic methods. In many ways, the characteristics of logistics can be described as a science [1]. The distribution and allocation of materials based on information received, planned schedules, or formulas, form the basis for calculations, deduction, and prediction [1]. Information such as fuel consumption by a convoy of trucks transporting from one location to another or the type of spare parts needed to support a certain vehicle can all be calculated ahead of time. These facts and rules coupled with the exponential growth of computer power and the proliferation of data have created conditions to enhance current logistic methods. The logistical experience of an organization combined with technology enable logistics planners to find opportunity in uncertainty allowing organizations to maintain their lethality and operational effectiveness.

1. Lessening Logistic Constraints through Technology

The state and health of our military equipment is deteriorating and in need of modernization [9]. Current Marine Corps logistic systems fall short to meet demands of a constantly changing Marine Corps [10]. According to a Marine Corps Gazette article titled "One Single Nail," the Marine Corps' "combat experiences in the dusty cities of Iraq and

the inhospitable regions of Afghanistan have not only demonstrated Marine resilience and lethality, but also that our support systems and methodologies are reeling under the pressure of constant demands" [10, p. 5]. The Marine Operating Concept suggests that this situation will continue. The Marine Corps Operating Concept published in 2016 [11] serves as a guide for future force development and the Marine Corps operational approach to overcome future challenges for 2025 and beyond. It defines a set of critical tasks to inform how the Marine Corps will develop its force. It describes the future operating environment as an Anti-access/Area denial (A2AD) threat environment, one that denies our ability to build and sustain large stockpiles of logistics at fixed locations ashore. During the Iraq and Afghanistan wars, the US military had logistical freedom of movement due to the enemy's inability to deny our logistic build-up ashore. The Marine Operating Concept describes the enemy of our next fight as one with an ability to disrupt this build-up as well as our logistic systems [11].

Thus, today, the problem USMC logistics planners face is determining what combination of methods allows them to achieve maximum effectiveness without loss of lethality. It is known that one-hundred percent availability of supplies and resources will never exist inside the Department of Defense (DoD) due to natural constraints; therefore, the flow of available supplies versus the demand for these supplies will always have a natural back and forth sway. To achieve "just in time" logistics, organizations and manufacturers must have the availability of mechanics, skilled laborers, factories, and DoD industry partners, as well as the proper demand signal. As the demand continues to increase for supply and materials and the money available to purchase supplies decreases, our current logistic models will limit our ability to meet forecasted demands.

Improving these logistics methods will require addressing a number of emerging challenges in logistics more broadly. Logistics' primary purpose is to ensure an optimal flow of cargo. Schuldt [7, p. 11] notes that "logistics is generally the planning and controlling" of supply material processes, such as procurement, production, and distribution "rather than the executing of respective operations" that bridge time and space between supply materials, personnel, infrastructure and the final material product.

Logistics planners and manufacturers have the near-impossible task of optimizing a complex system of material flow to match an ever-changing demand. In today's logistical market, such supply chain management is no longer a linear process but a series of complex interconnected supply networks with a large number of participants, services, and requirements [7]. Information is more integral now than before due to rapidly advancing technology and the proliferation of data combined with the increasing growth and demand of supplies. Moreover, the worldwide market of logistics is larger, more competitive, and more connected [4]. The wealth of information available today requires the need for computer integrated systems and programs to effectively synchronize and aggregate the network of logistic processes with operations [7].

2. Marine Operating Concept: A New Way to Fight

Particularly, for the Marine Corps, there is an added layer of complexity in logistics systems due to the inherent characteristics of war. Forces often operate in areas where resources and infrastructure are limited, and the availability of raw materials for manufacturing may not be reliable due to geographic constraints [1]. Consequently, Marine units must master a combination of all logistics methods plus "empathized self-sufficiency" to sustain themselves in austere environments. To minimize the mass of logistics ashore in an A2AD environment, the Marine Corps has shifted its operational approach to one of distributed operations spanning the breadth of the Area of Operations (AO). Distributed logistics in a contested environment aims to generate the benefits of mass without the vulnerabilities of concentration. To achieve this logistic objective, the organization must understand how to ensure the optimal flow of cargo, materials, and supplies by applying the proper logistic function [7].

This achievement is only possible through the use of information technology that supports the management and control of logistic processes that can be shared, standardized and easily accessible throughout the force. The National Defense Strategy explains that to build a more lethal force, organizations must modernize key capabilities by increasing equipment readiness [9]. Modern computing technologies are more adaptable to changing

circumstances due to computational speed. Therefore, it is integral that Marine logisticians and planners challenge themselves to discover other ways to manage logistic process controls through use of information, data, and computational means to gain organizational success and sustain lethality in the future fight. Organizations must be able to leverage logistic technologies to determine and prioritize which cargo needs to be handled or which repair part forecasted and ordered. Schuldt mentions "whenever the term logistics is used, the main focus is on materiels rather than on persons and information" [7, p. 13]. He indicates that in order to achieve the optimal flow of logistics, one must determine which logistical method must be applied. He explains that information plays a vital role in determining this effectively. Logisticians and maintainers are charged to find other ways to enhance our current methodologies through use of information and technology.

C. CONDITIONS-BASED MAINTENANCE DEFINED

The increasing availability of computational power and rapidly advancing analytics opportunities, allow organizations to leverage data from multiple equipment sensors and take advantage of analytical models and visualization tools to optimize their logistic processes, increasing equipment availability while reducing cost. Conditions-based maintenance is a preventative maintenance approach that uses sensor-based monitoring on system components of equipment to predict the actual time to failure of loss of efficiency [12]. A discussion on CBM and its implementation across the DoD follows.

1. Defining CBM

Conditions-based Maintenance is defined as "maintenance performed based on evidence of need, integrating reliability-centered maintenance analysis with those enabling processes, technologies, and capabilities that enhance the readiness and maintenance effectiveness of DoD systems and components" [5, p. 9]. The term “Conditions-based Maintenance” is not new to the DoD. Its use can be traced back to late 2002 with the release of Deputy Under Secretary of Defense for Logistics and Material Readiness memo Conditions-based Maintenance Plus [13]. This memorandum established interim policy and provided definition and guidance to the Military Departments on Conditions-based

Maintenance. It further described the objectives, applicability, and implementation of CBM throughout the DoD acquisition communities and Military Services. Efforts and invigoration for the continued development and implementation of CBM slowed dramatically due to the Invasion of Iraq in 2003 followed by the U.S. War in Afghanistan (Operation Enduring Freedom). It was not until 2012 that strategic alignment supported CBM along with technology advancements in information tools and maintenance equipment. During this time, DoD reissued and updated the Conditions-based Maintenance Policy declaring CBM "an essential readiness enabler" and the "primary strategy for achieving cost-effective equipment system life cycle sustainment" [5, p. 1]. This policy assigned responsibilities and implementation measures for Military Departments and Defense Agencies.

A DoD manual on CBM notes that an effective implementation of CBM requires using a "systems engineering approach to collect data" and information to enable analysis that "supports the decision-making process" for systems and equipment [14, p. 23]. Similarly, the industry defines Conditions-based Maintenance along these same lines. Olivier Sénéchal, an expert in performance measure for maintenance decisions, defines in his paper titled "Performance Indicators Nomenclatures For Decision Making In Sustainable Conditions Based Maintenance" that Conditions-based Maintenance is a "maintenance policy which undertakes maintenance actions before product failures happen, by assessing productivity conditions including operating environments, and predicts the risk of product failures in a real-time way, based on product data gathered" [15, p. 2]. Jardine, et al., [16, p. 1] define CBM as "a maintenance program that recommends maintenance" decision "based on information collected through monitoring" systems. For the logistician, CBM enhances push and pull logistics methods, further closing the gaps presented in the Marine Operating Concept to support distributed operations.

CBM uses onboard diagnostics from sensor data and technology built into the equipment to retrieve information about that system. Vanli discusses that the goal of CBM is to repair or replace parts before they fail, which "is in contrast to traditional maintenance

policies that are based solely on the system age or occurrence of breakdowns" [17, p. 335].

2. Reliability-Centered Maintenance as an Enabler to CBM

One of the key enablers of CBM is Reliability-Centered Maintenance (RCM). RCM uses historical data, equipment reports, and experience from industry experts to build probabilistic models to prevent future failures. DoD 4151.22-M (Reliability Centered Maintenance (RCM) manual defines RCM as "a logical, structured process used to determine the optimal failure management strategies for any system, based on system reliability characteristics and the intended operating context" [14, p. 25]. It establishes the need for reactive and proactive maintenance tasks. RCM is at the heart of CBM, built on years of data. It tracks as-is or current system trends, evaluating measures of effectiveness and their cost drivers, which feeds equipment and cost analysis reports.

The information gained from CBM is used in concert with RCM to improve combat power and lethality in terms of operational and material availability. Traditionally, units operate equipment until it fails and then they begin to progressively troubleshoot to find the issue. They also perform maintenance activities at regular time intervals determined by failure-time data [17]. CBM aims to drive units out of the reactive and preventative phase and into the "react when required" phase and model driven phase (predictive). Figure 1 depicts the evolution of maintenance from this traditional approach to the current predictive and sensor-based approach. Together, CBM combined with RCM form a predictive maintenance loop that allows us to leverage information to measure the variables that fail or degrade over time. This is then used to plan future maintenance activities.

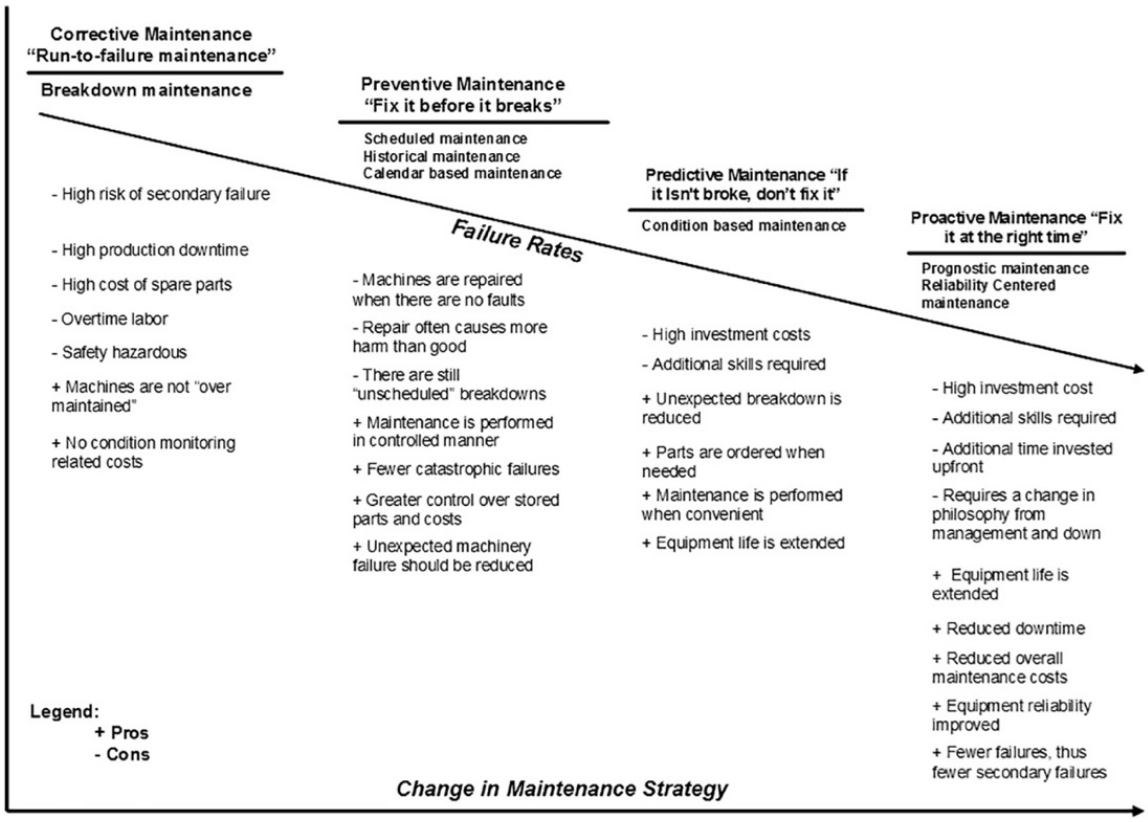


Figure 1. Approach to Maintenance. Source: [18].

D. CBM IMPLEMENTATION AND EXISTING CAPABILITIES ACROSS THE DOD

This section provides a brief discussion on how to enable CBM on ground equipment. Then this section will discuss CBM implementations in the DoD and in certain commercial organizations.

1. The Controller Area Network Data Bus

To conduct CBM, information must be obtained from sensors on equipment. Most modern commercial vehicles contain a high-speed Controller Area Network (CAN) bus, ISO 11898 [19]. This bus provides serial data communications between the electronic control units (ECU) in heavy-duty vehicles. ECUs include engine control units, airbags, temperature systems, anti-brake locking systems, etc. Modern heavy-duty vehicles can have over 70 ECUs [20]. These ECUs exchange thousands of messages per second between

other electrical components in the vehicle carrying data pertaining to wheel speed, torque control, engine temperature, oil diagnostics, and many other parameters [19].

The Society of Automotive Engineers (SAE) defined a communication protocol standard around the mid-90s [19] to allow simple communications across equipment sensors. Modern heavy-duty trucks in the United States, such as tractor trailers, construction equipment, agriculture or forestry equipment, use the Society of Automotive Engineer J1939 Standard (J1939) as a protocol for serial bus communication in internal CAN networks [21]. J1939 provides a higher layer protocol using CAN as the physical layer basis. J1939 offers a standardized method for communication across ECUs from different manufacturers. Logistic commercial companies leverage the J1939 protocol to create a logistic capability to build fleet management solutions [21]. This integration of equipment sensor data to a networked capability provides companies with the infrastructure for implementing conditions-based maintenance, challenging the idea of scheduled maintenance cycles.

Military vehicles also have the capability to leverage the SAE J1939 communication standard. The Marine Corps possesses a fleet of logistic vehicles and weapons systems that share the J1939 protocol standard on their CAN serial bus. Information obtained from on-board sensors using this capability can enable CBM. For example, the Medium Tactical Vehicle Replacement (MTVR) fielded in 2001 [22, p. 12] comes equipped with SAE J1708/J1939 Data Bus and Built-in Diagnostics. Several pieces of Marine Corps equipment such as the M142 High Mobility Artillery Rocket System (HIMARS), the Logistic Vehicle System Replacement (LVSR), the Mine-Resistant Ambush Protected (MRAP), the MRAP All-Terrain Vehicle (M-ATV), M1A1 Abrams battle tank, and some engineering equipment have the J1939 data bus capability [23]. These can leverage the same CAN-bus utilized in commercial industries via the J1939 data bus. However, not all Marine Corps equipment is equipped with J1939 sensor diagnostic capabilities. Some use an older serial communication standard not compatible with CAN or the J1939 protocol. These systems rely heavily on scheduled maintenance cycles, which limits their informational logistical reach. This dynamic creates a challenge to the CBM approach. Currently the Marine Corps has no CBM policy and is governed by the 2012

DoD CBM+ Memorandum. This has led to a gap in CBM capabilities in the Marine Corps logistics community, nevertheless, recently the Marine Corps has made several investments to accelerate the execution of CBM [24]. Conditions-based maintenance challenges organizations to develop an environment that can store real-time data as well as use historical maintenance indicators to drive decision and offer a new way of doing business.

2. Health Management Systems as an Enabler for CBM

A discussion on CBM implementations in the DoD and commercial sector follow.

a. CBM in the Air Force

A 1973 study by the Rand Corporation for the Air Force analyzed inefficiencies in maintenance data analysis. Their goal was to understand the relationship between costs and the benefits of scheduled maintenance activities. The Rand study reached three conclusions. First, that time or interval-based scheduled maintenance alone overlooks many factors that lead to increased aircraft sustainment [25]. Second, there is no correlation to the number of aircraft mishaps due to equipment failure and increased frequency of scheduled maintenance [25]. Last, for adequate interpretation of maintenance data, many issues require skilled personnel that understand maintenance policies and procedures and the augmentation from computer-based programs. Since these results, the U.S. Air Force has invested heavily in integrating health-monitoring systems into their platforms.

Currently, the Air Force is implementing CBM on their B-1B and C-5M aircraft [26, p. 62]. Their CBM toolkit consists of real-time data streaming, predictive maintenance algorithms and reporting, active Reliability Centered Maintenance, condition-based inspections, and comprehensive fleet health dashboards [27]. For some time, the Air Force has been developing a CBM ecosystem and "already sees a significant reduction in cost and about a 30 percent reduction in unscheduled maintenance" [26, p. 62]. The Air Force is building a CBM ecosystem that is a combination of predictive analytics and codified processes that captures the maintenance experience through the life of a repair part. The Air Force has adopted a predictive maintenance alert loop (PMAL) process that enables

stakeholders involved in the process to make informed decisions regarding the timing and location of aircraft maintenance actions and performance indicators for failures [28].

The US Air Force's CBM vision is robust. Instead of looking at one aspect of CBM, they aim to propagate a culture in which CBM philosophy can thrive beyond just predicative algorithms and sensors to a new business model (by 2025) [29]. In response to Combatant Commanders' needs for rapid development of new capabilities in the current fight and to outpace our near-peer competitors, the Air Force initiated the Air Operations Center (AOC) Pathfinder effort in August 2017 [30, p. 19]. The USAF Pathfinder program has successfully demonstrated several CBM+ solutions, particularly on the B-1B and C-5M aircraft. On the C-5M, the Air Force was able to detect and replace a faulty primary heat exchanger outlet temperature sensor before it failed. This alleviated an eventual costly air exit door failure and avoided an expense of over $250,000 per occurrence.

b. CBM in the Army

The US Army Material Systems Analysis Activity (AMSAA) is focused on developing onboard systems for ground tactical vehicles to enable CBM. "The onboard system that AMSAA has designed in conjunction with the Aberdeen Test Center collects data from a vehicle's onboard vehicle sensors, data bus, terrain sensors, and global positioning system (GPS) and analyzes the data to determine the vehicle's condition" [31, p. 1]. Since 2008, the US Army has invested heavily and partnered with organizational-level Command and Research and Development units to spearhead strategies and improve ways to reduce maintenance burden while increasing readiness through CBM [32]. The Army used a phased approach to implement CBM into their fleet. They integrated a small health monitoring system "with limited but very specific capabilities that can be deployed across larger fleets" [31, p. 1]. This system allowed them to integrate proven hardware and data analysis methods into other equipment. The approach has allowed the Army a steady way to build a health monitoring system platform into a sustainable, robust, military-grade analysis platform [31]. Today, the US Army has enabled CBM and diagnostic technologies for their Stryker vehicles and Heavy Expanded Mobility Tactical Truck (HEMTT) A4 platforms through their Stryker Brigade Combat Team (SBCT) Embedded Data Collection

and Analysis System (SEDCAS) program [33]. They have also integrated several diagnostic systems and programs such as the Health Usage and Management System (HUMS) and Vibration Management Enhancement Program (VMEP) to their UH-60, CH-47 Chinook, and AH-64 Apache helicopters [34], with plans to expand to other vehicles and platforms [35] in the near future.

Kevin Guite [32], a lead operations research analysis partner with AMSAA, reported that a recent AMSAA study "indicated that approximately 97 percent of the tactical wheeled vehicle fleet and 98 percent of the 1,310 instrumented Strikers were being serviced based solely on time rather than actual use." He also reported that the study shows that maintenance was being conducted to replace fluids or repair parts before conditions were actually necessary due to required time-based maintenance policies. As a result, this has led to further developments and improvements in the Army's maintenance sustainment strategy for CBM. Currently, the Army is using a transportation company in its 25th Infantry Division to conduct experimentation and testing focused on improving maintenance burdens, reducing cost, and evaluating sustainment practices through CBM [32]. It is clear the Army has recognized the benefits of CBM across their fleet of equipment, and studies show the investments are clearly worth it.

c. CBM initiatives at Naval Sea Systems Command

Since 2001, the Navy has trained and certified over 10,000 personnel in RCM and invested in CBM technologies over the last 10 years [3]. Now, with recent improvements in CBM enablers, due to the growth and rapid expansion of Internet of Things (IoT) devices, the Navy is expecting to capitalize on tools and systems to "make vast improvements in [their] Operating Availability, reduce system downtime, and also save money" [3, p. 2]. The Navy has been using the Integrated Condition Assessment System (ICAS) as their common tool to support CBM for the past 20 years [3]. The Navy is already migrating to a new health monitoring system called Enterprise Remote Monitoring (eRM) and Consolidated Machinery Assessment System Ashore (CMAS) that is able to support rapid development for new algorithms and linkage to existing systems and sensors [3]. The Navy is using CBM as an enabler to sustain their fleet and close the digital thread.

d. ALIS enabling CBM for F-35 Lightening

A recent and well-publicized example of CBM within the DoD comes from the F-35 Joint Strike Fighter (JSF) [36, p. 11]. This joint service inter-operable aircraft is equipped with an Autonomic Logistics Information System (ALIS) that automatically responds to failure events and transmits appropriate information across its globally distributed network to technicians. A published report released by the Air Force Logistics Management Agency on CBM initiatives highlighted the health system as an "emerging" example of true prognostic-capabilities embodying the full intent of CBM+ and an unmatched technological capability as compared with other weapon system aircraft [37]. The ALIS creates a logistic environment in which little human intervention is needed to engage the logistics cycle. The ALIS serves as an information conduit, transmitting aircraft heath and maintenance information to appropriate users while in flight through radio frequency links.

The ALIS captures the total flight and maintenance experience from cradle to grave for the JSF, bringing together the training, planning, maintenance, and supply support in one ecosystem. The ability to scale and drive decisions is what makes ALIS powerful and novel as a CBM tool across the DoD. Logistic infrastructure is already transforming to match the flexibility and responsiveness the system provides [38]. ALIS is minimizing cost by optimizing a logistic strategy that operates within the lowest level of maintenance required to successfully meet mission. This keeps the logistical footprint low. Organizations are adjusting their inventory policies to match the flexibility of ALIS, creating a seamless interface into the supply chain [38]. ALIS is paving the way for supply system reform, providing new ways to view part consumptions, and breaking down stovepipes of data transmission, allowing information output of action and recommendation rather than just data [38]. These factors along with a host of other aspects bring us closer to an Autonomous Logistics environment.

e. CBM in Industry

Similar to military, commercial companies leverage CBM through use of health management systems and diagnosed sensors. Southwest Airlines’ aircrafts are equipped

with an onboard network system architecture that "securely connects airline operations and maintenance with key airplane data and software parts making data available for flight crews and airline ground infrastructure" [39]. Southwest Airlines is seeking to move to a predictive maintenance approach by transition to an airplane health management platform managed by Boeing [39]. The small startup FIXD is leveraging data feeds from the On-board Diagnostics (OBD2) on personal vehicles to build a personal diagnostic mobile app. Their app and plug-in sensor keeps track of vehicle diagnostics and notifies drivers when their vehicle is due for maintenance [40]. As described in the article [40], the data from the app can reveal patterns such as parts that appear to fail around the same mileage mark — and dealers can start fixing problems for customers before they even happen. In the same way FIXD leverages the OBD2 controller area network to enable CBM the Marine Corps can leverage the J1939 CAN transmitting off of military vehicles.

E. IMPROVING LIFE CYCLE MAINTENANCE WITH CBM

A discussion on the impact of CBM on current maintenance practices follows.

1. Readiness Mandate

In Fall 2018, the Defense Secretary issued a mandate to achieve 80 percent mission-capable rates for the F-35, F-22, F-16 and F/A-18 aircraft by the end of FY-19 [41]. Though funding is an issue and a separate concern, the Air Force is meeting this demand by working with industry partners, such as Lockheed Martin, to leverage CBM technologies and grow their spare parts pool. Through the use of their health management systems and CBM ecosystem, they are able to conduct parts prioritization and shorten lead times between industry partners, maintainers, and supply, interconnecting all parts of logistics. This increased interconnectivity enabled by CBM builds a community effort aligned from acquisition to operation along a common axiom, increasing the total life cycle of equipment and readiness.

To meet this mandate, organizations must optimize a CBM strategy that increases operational availability by matching maintenance capabilities to dynamic mission needs. This is one of the principal objectives of CBM [18]. To support this strategy, all key players, manufacturers, distributers and providers must define, redefine, and assess ways

for continuous process improvement throughout the life cycle of the system. As stated in the CBM+ Guidebook, "Implementation of CBM+ is not a single event. It is an evolutionary effort that progresses incrementally. DoD managers at all organizational levels, including logistics activities, PMs, depot- and field-level maintainers, and operational commanders face similar management issues during CBM+ implementation. A good manager periodically steps back, reviews the organization's progress, and assesses the initiative results to date" [18, p. 68]. This continuous feedback loop improves the total system life cycle of equipment. Figure 2 shows the relationship between CBM and the total system life cycle.

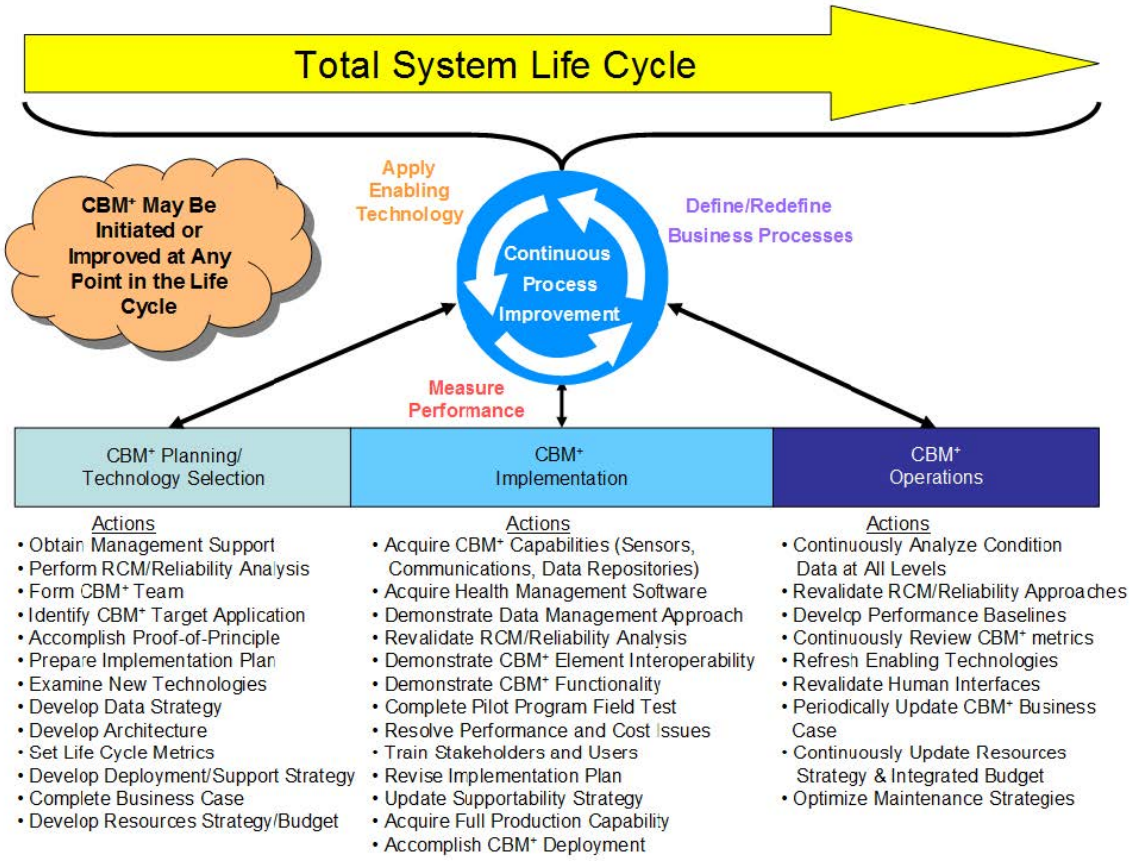


Figure 2. Relationship between CBM and Total System Life Cycle. Source: [18].

2. CBM Business Strategy

Diagnostics or health monitoring systems in many cases have proven to detect future failures and predict time between failures [17]. The statistical models and prediction methods that drive the underlying outputs to these systems enable analysis that captures inter-dependency within relationships [17]. These models have enabled “the transition from a corrective or time-based maintenance approach to a proactive”, and “predictive-based philosophy” as noted in the DoD CBM guidebook [18, p. 24]. CBM drives maintainers and logisticians to design new business paradigms to support an environment that enables constant process improvement.

The Marine Corps must determine its own "specific maintenance business strategies based on operational need, mix of facilities, application of technologies, and availability of skills, organizational structure and resources" [18, p. 24]. This research looks at fundamental ways to apply CBM application to improve maintenance planning and prediction of future equipment failure. The goal of CBM is to reduce maintenance cost manpower constraints while increasing operational availability. In today’s technological age the interconnectivity of sensors and their ability to communicate across devices is rapidly expanding. One method of improving life cycle maintenance through CBM is by leveraging data available from military equipment sensors to forecast maintenance tasks and minimize equipment down time. This eliminates the sheer amount of man hours and scheduled maintenance paper trail created by dated policies and processes.

F. SUMMARY

Traditionally the Marine Corps relies on a "supply-push", "demand-pull" or a combination of both to plan logistic sustainment. The 2016 Marine Operating Concept and the 2018 National Defense Strategy challenge logisticians and planners to modernize these methods and equipment capabilities to build a more lethal force for the future fight. Conditions-based maintenance is used to enhance traditional push/pull logistic methods by leveraging current technologies, historical data, and modern computing power. Information from on-board equipment sensors can be used to develop analytical methods, formulas, and calculations to predict equipment failures and increase equipment sustainment. The

Marine Corps' current maintenance approach is to operate equipment until failure and progressively troubleshoot to diagnose the maintenance actions. CBM is a maintenance model that allows units to transform maintenance approaches to produce timely results through data driven models and conditions monitoring. This allows organizations to match the security challenges demanded by the 21st century. DoD organizations and Military Departments use health monitoring systems to leverage data from on-board sensors to enable CBM. Industry partners and commercial companies, such as airline manufacturers and vehicle companies use similar health monitoring systems and interconnected devices equipped on their equipment to enable CBM across their fleet. This research explores a use case in which the Marine Corps can leverage on-board sensor information collected from ground vehicles to enable a Conditions-based maintenance approach.

THIS PAGE INTENTIONALLY LEFT BLANK

III. EXPERIMENTATION METHODOLOGY AND TECHNICAL APPROACH

CBM stretches across eight broad areas, depicted in Figure 3. This research is focused on five of the eight areas—sensors, condition monitoring, health assessment, data management, and analytics. Sensor refers to the raw electronic communication data transmitted across equipment components [19]. Condition monitoring is a maintenance process of converting raw sensor output to a digital parameter representing a quantifiable physical condition or measurable engineered valued [18]. Health assessment is the capability to derive usable data from sensor measurements or condition monitoring to identify abnormalities or facilitate the creation and maintenance of "normal" operating baseline profiles on equipment [18]. Data management consists of acquiring, manipulating, transmitting, and storing data captured from sensors [18]. Analytics represents the off-system processing of diagnostic data through some form of software to determine the current health state of equipment.

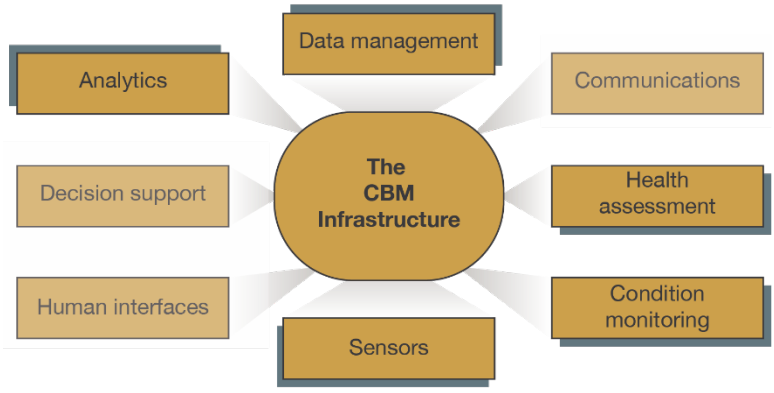


Figure 3. The Infrastructure of CBM. Adapted from [18].

Three things are needed to conduct CBM at the enterprise level. First, a combination of physical logging devices to pull diagnostics data from equipment. Second, a computing architecture network to collect, parse, clean, and process the data. Third, a software tool that reads the maintenance data collected and through some statistical model enables analysis methods that feed data insights to front-end maintenance systems. A CBM

maintenance policy incorporates data insights to trigger maintenance actions. This research explores the most basic form of a CBM ecosystem using the following methodology shown in Figure 4. There are three steps in the process:

1. Raw data collection from on-board sensors using Raspberry Pi, SAE J1939 Interface Board, and standard J1939 CAN cable connector.

2. Data cleaning and decoding using software tools to support condition monitoring and data management. Leveraging software tools, the individual raw log data files collected are parsed and decoded using Python, a programming language. The SAE Digital Annex (document used to convert J1939 log data) is loaded into a database and used to execute queries to convert parsed log data into human-readable messages. This data is loaded into a data frame (which can be scaled to a database or cloud architecture) for further processing.

3. Data analysis is performed using statistical methods to define the data structure and categorize the data according to its properties. During this step we can derive a time-series regression model used for prediction that categorizes “normal” operating conditions.

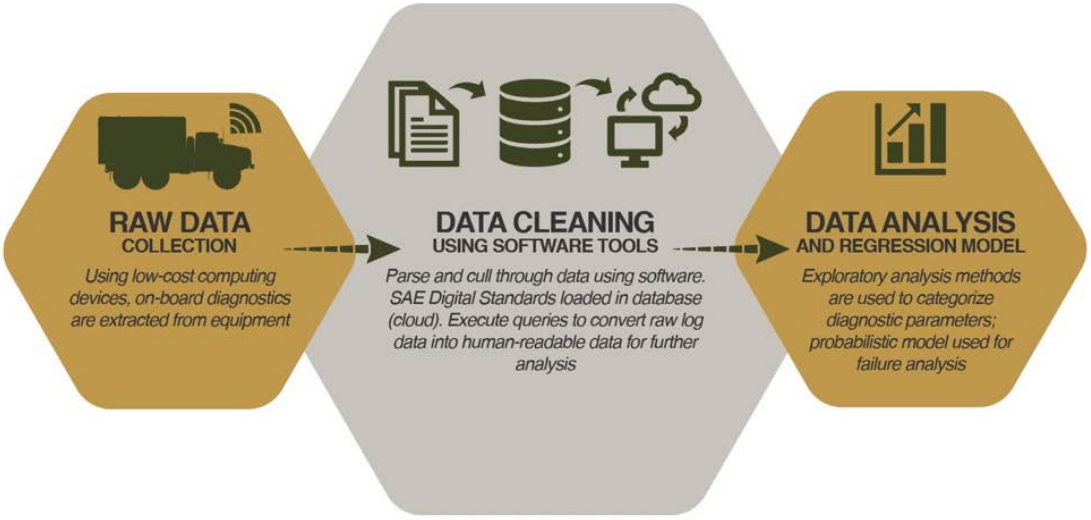


Figure 4. Decoding J1939 Diagnostic Data Methodology

Using a single MK23 Medium Tactical Vehicle Replacement (MTVR) shown in Figure 5, condition monitoring techniques are used to capture data from vehicle components transmitted through the CAN serial bus. This data is then decoded and analyzed to generate information which is used to identify condition thresholds for "normal" vehicle operating conditions, allowing abnormalities to be identified when new data is acquired.

Figure 5. MK23 Medium Tactical Vehicle Replacement. Source: [42].

The MTVR is a widely used ground vehicle across the Marine Corps. Also, the SAE J1939 Port is easily accessible, located behind a removable panel on the passenger-side dashboard [36]. While this research is focused on a single-use CBM model at the tactical level, the data management environment demonstrated in this approach can be scaled and exported into an enterprise-level environment at the strategic level.

A. CBM DECISION MODEL METHODOLOGY

CBM is conducted at time-directed intervals only when it is believed that a failure is likely to occur. Tsang describes “three types of decisions which need to be made in condition-based maintenance” [12, p. 13]:

1. selecting the parameters to be monitored;
2. determining the inspection frequency;
3. establishing the warning limit (the trigger). [12, p. 13]

These steps are used as an over-arching methodology for this research. While the MTVR offered only a select number of diagnostic parameters to monitor, due to dated truck software, parameters with greatest correlation to engine operating conditions were selected from the MTVR. This research will focus on the data available during raw data extracting to present the emerging patterns and characteristics of “normal” engine operating conditions. The correlation coefficient between the engine speed and extracted parameters is used to determine the extent of which parameters are most related. Further discussion on the correlation of data parameters is in Section C, Chapter IV. As a final note, diagnostic parameters vary across equipment sets and depends on the data available, the data extraction hardware used, and the availability of software resources that perform the analysis. Different types of equipment produce different parameters. This research focuses on developing one method for one vehicle that is exportable to different equipment platforms regardless of available parameters.

The MTVR was monitored continuously with data from the truck being transmitted at 250 Kbits/sec for the inspection frequency. This rate equates to one diagnostic message every millisecond. Each message contains a number of measurable vehicle functions. For other cases where the equipment is not monitored continuously, Tsang comments that an inspection interval between when to pull data must be determined "so that action can be taken in time to prevent functional failures or to minimize the consequences of those which cannot be prevented" [12, p. 13].

This research is focused on finding static triggers outside of "normal" operating conditions. CBM programs can use a combination of static or dynamic warning limits as triggers or alarms to diagnose or begin the repair process [12]. Static limits, much like time-directed intervals, use pre-selected thresholds for measured data, for example, changing vehicle oil once it reaches the 10% level because prior RCM/CBM analysis has signaled likely failures after the 10% oil-level threshold. Alternatively, dynamic warning limits monitor the rate of change of a measured parameter. Measuring the rate of change is valuable because a great change in a parameter over a short amount of time could indicate a potential failure even if the actual value of the parameter is in normal operating range. This research uses static warning limits based on the data collected.

B. DECISION MODELS FOR CBM

Many predictive maintenance models exist to prevent failure, detect the likelihood of failure, or discover hidden failures. Additionally, statistical modeling approaches often extend or build from previous methods already discovered. A brief discussion of the models that contributed to the derivation of the final CBM decision model used in this research follows.

1. Linear Regression Analysis

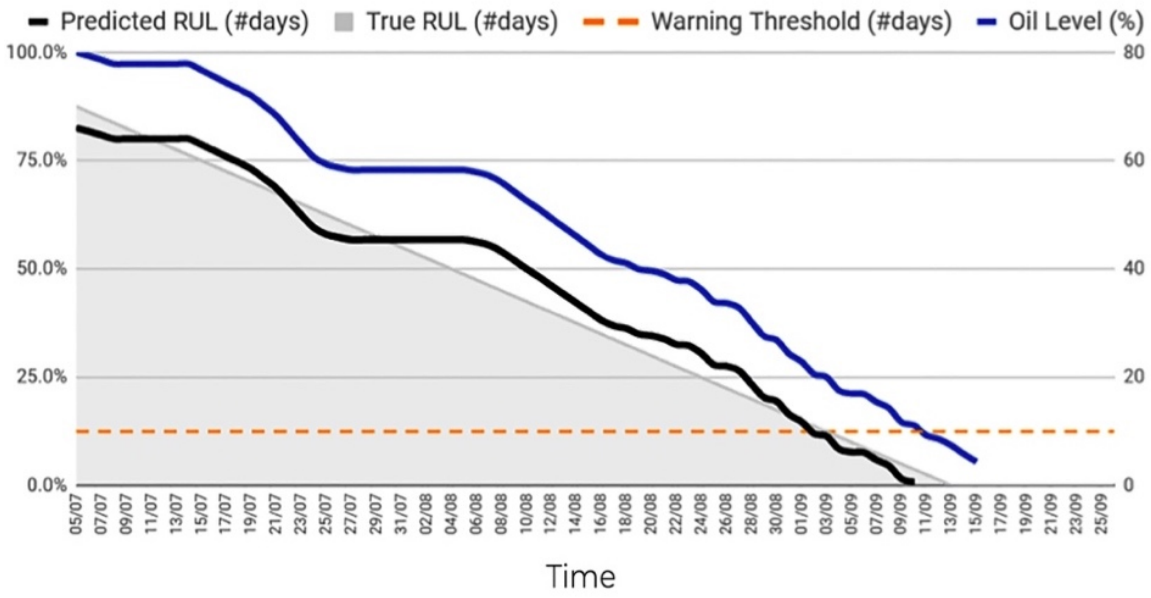
This research uses a time-series regression to describe “normal” conditions of engine parameters in the presence of speed. In simple terms, this model is an extension of linear regression analysis. Using linear regression analysis, we can predict failure or "Remaining Useful Life," the time left before the next failure, based on a predictor (parameter), such as “Oil Level”, and decide a time for maintenance triggers and warnings [20]. Regression analysis investigates the relationship between two or more variables [43]. It is the simplest approach for predicting a quantitative response on the basis of some predictor, x, which is some engine parameter extracted from the MTVR that can be used for predicting failure.

A simple linear regression model is typically expressed as

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{1}$$

where β_0 and β_1 are unknown constants that represent respectively, the intercept and the slope; and y is the random variable representing the response on dependent variables, which in our case is a measure of engine wear; oil level or some other diagnostic parameter is represented by x, and ε is an unobservable random error. The model, applied to a data set, assumes that random error is independent and identically distributed with a mean-zero, Normal distribution. The observations in the extracted data set allow us to estimate β_0 and β_1. When model assumptions are reasonable, the maximum likelihood estimates of β_0 and β_1 are equivalent to the least squares estimates which find the values of β_0 and β_1 to minimize the sum of squared differences between the observed and predicted values of the response variable. When the modeling assumptions are not met, but the relationship between the expected response and predictor variable is linear, least squares estimators of β_0 and β_1 are unbiased and provide sensible estimators of β_0 and β_1.

Consider a vehicle where the only possible failure is from not refilling the oil. Using linear regression, one could observe and record the oil level over a specified time period or until the vehicle fails, and estimate the best possible linear relationship across all recorded samples of the oil level observed. That line is then used to predict the time or oil level percentage of a likely failure, illustrated in Figure 6. This simple regression model allows us to develop procedures for making various predictions and obtain quantitative measures on how closely two variables are related.



As oil level decreases, RUL decreases. Triggers to conduct maintenance are established at the 10% oil level.

Figure 6. Linear Regression Analysis to predict Remaining Useful Life. Source: [20].

2. Failure Rate Model

An alternative to a linear regression model is a failure rate model. This model is used to estimate the likelihood of failure of an item over time and helps describe the level of deterioration at a designated time interval, which is commonly used in CBM decision modeling and worth noting. The survival or reliability function is $R(t)$, which describes the probability that an object will survive beyond a specified time [12]. For our purposes, $R(t)$ describes the reliability of a piece of equipment at some usage time or age limit. One method to conduct CBM is through a time-directed interval (similar to static triggers) based on a failure model defined by the failure rate, expressed as

$$h(t) = \frac{f(t)}{R(t)} \tag{2}$$

where the probability density function of failure is $f(t)$. Tsang defines the failure rate of equipment as a hazard or failure function that measures the risk of failure at time t among equipment that have survived to time t [12]. Georgia-Ann Klutke defines, "A fundamental tenant of reliability theory is that the [failure] function displays a 'bathtub shape'" [44, p. 1]. This statement means that when the equipment is new or the usage is low, the failure rate, $h(t)$, decreases. However, as time of use and age increase, the equipment falls into a "useful life" period, when the failure rate is constant. This leads to a “wear-out” period, when failure rate increases over time [12, p. 3], [44]. The static triggers (time intervals to begin the repair process or diagnosis) are placed right before the vehicle enters the “wear-out” period. Extensions of the failure rate model allow the inclusion of diagnostic parameters. By leveraging a failure rate model, future failures are more likely to be captured and prevented before they begin.

3. Time Series Analysis

The first two decision models, linear regression and failure rate modeling, are two classical approaches popularly used to predict equipment failure rates and prescribe CBM policies. Each approach requires a dataset of “good operating conditions” and “failed operating conditions” to test for valid predictions. For the purpose of this research, a time series model best fits the data extracted from the MTVR. Due to scope and limitations, only one dataset of parameters was extracted from one MTVR. A dataset of “failed” truck data was not captured; however, the following method used can be scalable across a pool of trucks or equipment set.

In this research, we use a regression model that closely reflects the behavior of the data extracted from the MTVR. To categorize “normal” operating conditions, we measured temperature and pressure parameters of components because they can be modeled as an increasing function of age (usage). For example, when engine speed increases, operating temperatures react in response to the engine speed, or when pressure levels of engine fluids rise or fall, usage indicators of engine components correspond. Time series analysis is the analysis of a series of data points over time. A time series, Y_t, is a discrete time, continuous

state process where time $t = 1,2,\ldots,= T$ are discrete time points spaced at uniform time intervals, particularly at a single occurrence of a random event [45], [46].

$$Y_t = \mu + \beta_0 X_t + u_t \tag{3}$$

With regard to the data collected, the operating temperature of an engine parameter is dependent on the temperature prior to the current temperature; a time series model with lag values captures this changing value characteristic over time. Furthermore, the current engine temperature also depends on current and past (lagged) engine speeds. These engine speeds X_t measured at $t = 1,2,\cdots,T$ are the independent or predictor variables analogous to those used in linear regression.

An autoregressive distributed lag (ARDL) time series can be used to express the behavior of the dependent variable Y_t as follows [46, p. 682]:

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \sum_{j=0}^{r} \beta_j x_{t-j} + \varepsilon_t \tag{4}$$

where, as in the linear regression model, ε_t are independent and identically mean-zero, Normally distributed; p and r are the lag length for the dependent and predictor variables respectively, and μ, $\gamma_t,\ldots,\gamma_p$, $\beta_0,\ldots,\beta_r$, are unknown constants to be estimated. Since at any time t, y_t, x_t and their lagged values are observable, and the ARDL in Equation 4 has the form of a linear model, ordinary least squares gives efficient estimates of the unknown constants [46]. Using this model, we are able to predict values of Y_t using x_t and values of both the dependent and predictor variables that have occurred before.

C. DATA ORGANIZATION

1. Data Properties

The data collected revealed four measures of readable engine temperature parameters, two measures of vehicle speed, one measure of engine fuel consumption, and one measure of engine oil pressure, depicted in Table 1.

Table 1. List of Human-readable Diagnostic Parameters

PGN	PGN LABEL	SPN	SPN NAME
61444	Electronic Engine Controller 1	190	Engine Speed (RPM)
65262	Engine Temperature 1	110	Engine Coolant Temperature
		174	Engine Fuel 1 Temperature 1
65263	Engine Fluid Level/Pressure 1	100	Engine Oil Pressure 1
65265	Cruise Control/Vehicle Speed 1	84	Wheel-based Vehicle Speed
65266	Fuel Economy	183	Engine Fuel Rate
65270	Intake/Exhaust Conditions 1	105	Engine Intake Manifold 1 Temperature
65272	Transmission Fluids	177	Transmission Oil Temperature 1

Table list of PGNs and SPNs relating to temperature, speed, oil pressure, and fuel rate from MTVR

Out of the 9,000 diagnostic messages available in the J1939 protocol (not including proprietary messages), the MTVR used for experimentation produced 23 unique message IDs. For this research, 17 diagnostic messages contained information that decoded to measurable data. The other messages contained "on/off" message indicators of truck functions, such as lights on/off indicators, air condition in use, passenger door open, etc. Subsequently, the MTVR produced 82 data parameters, though only 32 were human-readable. Much of the data extracted from the truck logged a "Not Applicable" message indicator. This is due to truck settings designated by the manufacturer and dated truck software.

2. Data Categorization to Detect Normalization

The data is categorized by the subset of parameters that contributes to "normal" operating conditions. Temperature values of engine components revealed measurable observations under a variety of operating conditions. Lag values of temperature were measured to monitor residuals. The residuals in the data reveal the variability of the dependent variables (the temperature and pressure parameters) under conditions of an independent regressor (engine speed or RPMs). Outliers in data are discarded. Using

residual analysis through normalizing quantile-quantile (QQ) plots and robust autocorrelation function estimation allowed the verification of prediction results. The model built conceptualizes normal behavior.

In this method, we develop a method for one MTVR that is exportable to another MTVR. Limits are set around the “normal” conditions, and CBM policies are established for alarms/triggers based on “out of normal” temperature readings.

D. SUMMARY

The proposed methodology for this research incorporates five of the eight areas of a CBM infrastructure. The methodology is based on a three-step approach cataloging the process to collect raw data from on-board equipment sensors, then the process to derive usable data from sensor measurements, and the manipulation, storing, and transmitting of diagnostic data for off-system processing to determine data insights for front-end maintenance systems. The statistical methods presented, linear regression and failure rate modeling, are common approaches used to reflect the behavior of equipment operating conditions. These methods are the underlying models that form the basis of any CBM maintenance policy. This thesis uses an extension of the regression model, an ARDL time-series, to characterize the data collected from an MTVR to detect normalization, which is used to recommend a CBM maintenance policy.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS, ANALYSIS, AND LIMITATIONS

This chapter focuses on answering the problem statement, presented in Chapter I, of determining a good architecture for the automated collection of sensor data that can be used for a CBM approach on Marine Corps equipment and how this collection of sensor data can be used to provide in-depth maintenance analysis and generate enhanced maintenance solutions for vehicle life-cycle support. This chapter expands upon the methodology presented in Chapter III: We use a standard data engineering approach to log, parse, decode, and analyze raw sensor data collected from the MTVR. In Section A of this chapter, we present a collection method to extract raw sensor data from ground equipment. It begins with an overview of the hardware tools required and then demonstrates how to implement tools to extract vehicle data. Then, in Section B, this chapter derives a technique using the J1939 protocol to process the collected data to produce useful data features for further analysis. Finally, in Section C, it examines and analyzes the collected data to build a predictive model of "normal" operating conditions. Figure 7 maps our experimentation workflow to our overall methodology.

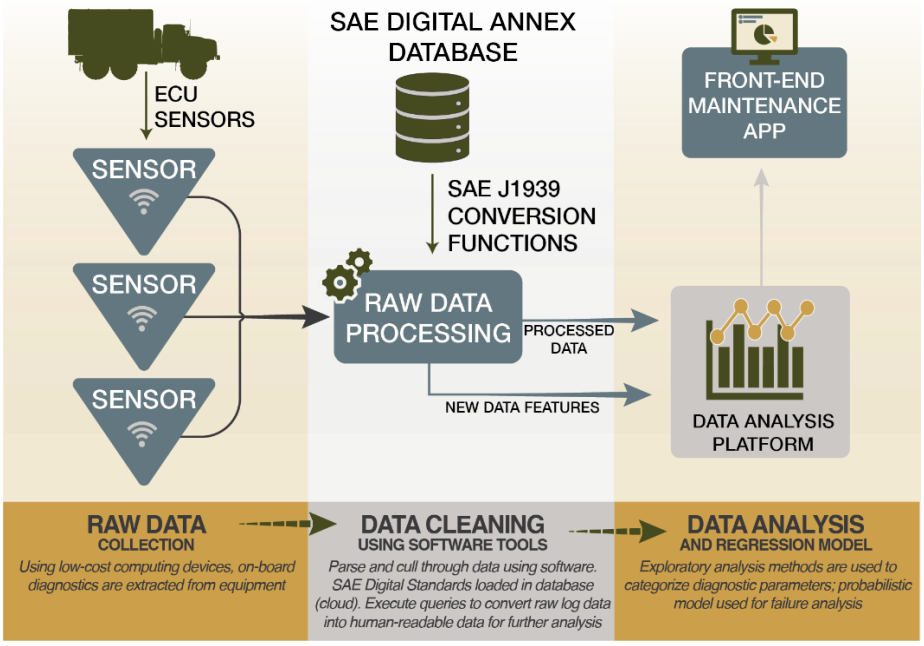


Figure 7. Data Engineering Workflow

A. EXTRACTING VEHICLE CAN-BUS DATA

The first step in extracting vehicle CAN-bus data is configuring hardware tools and devices to read and log raw diagnostic data from equipment sensors to support condition monitoring. The MTVR uses CAN as a diagnostic data-bus technology which, is the industry standard for diagnostic data transmission. The advantages of CAN field-bus technology are reduced wiring, reliable communication, reduced production cost, and easy implementation and service capabilities [19].

1. Hardware

This section covers the three main components required to extract on-board diagnostics from military equipment: a single-board computer, the SAE interface board, and the SAE J1939 connector cable (discussed in Subsections a. and b., respectively).

a. Single-Board Computer and SAE Interface Board: Using Low-cost Microcontroller Devices to Collect Data from Equipment

The Raspberry Pi with an attached SAE interface board is programmed to extract data from the CAN-bus serial port. The Raspberry Pi is a handheld computer device that is used to solve problems, run applications, and prototype technology solutions. It is suitable for our research purposes due to its low cost (approximately $35) and sufficient computing power [47]. This device can be coupled with an SAE interface board which provides two independent CAN-bus interfaces that are used to extract data from a CAN-bus serial port. The Raspberry Pi used in this research (Figure 8) is the Raspberry Pi 3 Model B+, which supports 1GB SDRAM, 2.4GHz and 5GHz wireless LAN, Bluetooth, HDMI, USB ports, camera port, and display ports [47]. A MicroSD is used as the hard drive and stores data. It is powered by a 5-volt micro USB supply. As shown in Figure 8, the SAE interface board is the PiCAN2 Duo CAN-Bus board, offered for less than $100. It uses the Microchip MCP2515 CAN controller with MCP2551 CAN transceiver [48]. Connections to the Raspberry Pi are made via a 4-way screw terminal. This board also comes with a 5-volt 1-amp power supply which powers the Raspberry Pi as well via the screw terminal [48].

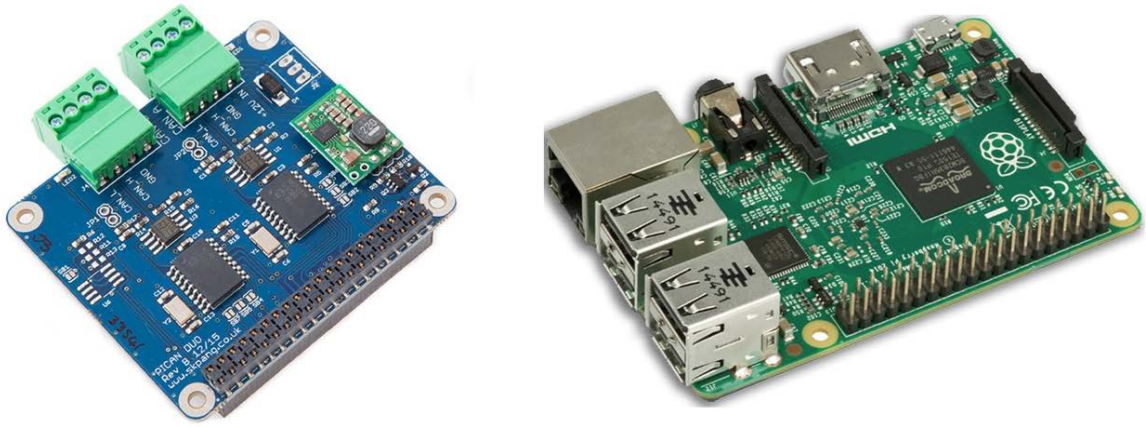


Figure 8. PiCan2 Duo Board (left). Raspberry Pi 3 Model B+ (right). Source: [47], [48].

b. 9-pin Deutsch Connector to CAN Serial Bus

A connector cable is required to connect the Raspberry Pi to the MTVR diagnostic port (J1939 CAN-bus). The SAE J1939 Connecter, shown in Figure 9, is a Deutsch HD10 9-pin round connector cable that connects to the CAN serial port on heavy-duty vehicles. It is a standard connector for diagnostic purposes and costs approximately $25. The connecter can be wired into the SAE interface board connected to the Raspberry Pi. By simply connecting the SAE Interface board to the Raspberry Pi via the screw terminals and configuring the wiring of the SAE J1939 connecter into the SAE Interface board in accordance with the CAN Standard [49], on-board diagnostics can be logged from any equipment equipped with the CAN-bus serial port. Figure 9 shows the complete data-extracting hardware configuration, including the wiring configuration for the SAE J1939 connector into the SAE interface board.

Top-left: 9-pin Deutsch J1939 connector cable. Top-right: wiring configuration of SAE connector wires to interface board CAN inputs. Bottom-left: physical connection of 9-Pin SAE connector to MTVR J1939 port. Bottom-right: exposed 9-pin Deutsch J1939 female connector on MTVR.

Figure 9. Connection of 9-pin Deutsch J1939 Connector to MTVR and SAE Interface Board

2. J1939 Protocol Architecture

Once the hardware has been assembled, the log data collected can be decoded and converted into measurable values using the SAE J1939 standard. The SAE J1939 is a

communication standard used by many vehicle and heavy-duty truck manufacturers for communications across engine ECUs. A discussion on how to interpret and use this communication protocol follows.

a. The SAE J1939 Standard

The Society of Automotive Engineers (SAE) has defined a high-speed CAN-bus communication standard that supports "real-time, closed-loop control functions, simple information exchanges, and diagnostic data exchanges between electronic control units throughout the vehicle" [50] for heavy-duty vehicles and equipment. The SAE J1939 provides a higher-layer communication protocol using CAN-bus as the physical basis. It is the key protocol among devices that transmit electronic signals via ECUs across vehicle components [19], allowing for one language across commercial manufacturers.

This standard specifies how to handle diagnostic messages transmitted via the ECUs from vehicle sensors. Each message of J1939 is 29-bits in size. Wilfried Voss explains that “the main characteristics of the J1939” 29-bit message are the embedded “Suspect Parameter Numbers (SPN) and Parameter Group Numbers (PGN), which point to a large set of predefined vehicle data and control functions” [19, p. 19]. Figure 10 shows the J1939 message format frame architecture, which includes the message identifier that contains a PGN and the associated data fields that contain the SPNs. SPNs are the data parameters of a J1939 message, which contain information such as engine speed or RPMs. A PGN is a unique ID inside the 29-bit J1939 message identifier that defines the diagnostic function of a J1939 message. PGNs are used to group together associated data parameters (SPNs) within the data field of a J1939 message [19], [20]. For example, we can have a raw 29-bit message identifier such as 0x18feee00. Raw message identifiers are logged as hexadecimal numbers by default and divide into three sections: a priority field, which is three bits in length; an 18-bit PGN field that identifies the message data parameters; and an 8-bit source address containing the node address of the ECU that transmitted the data. In the case of message ID 0x18feee00, the PGN starts at bit 6, with length 18 (indexed from 1). The outcome is PGN 0xfeee, or, in decimal, PGN 65262.

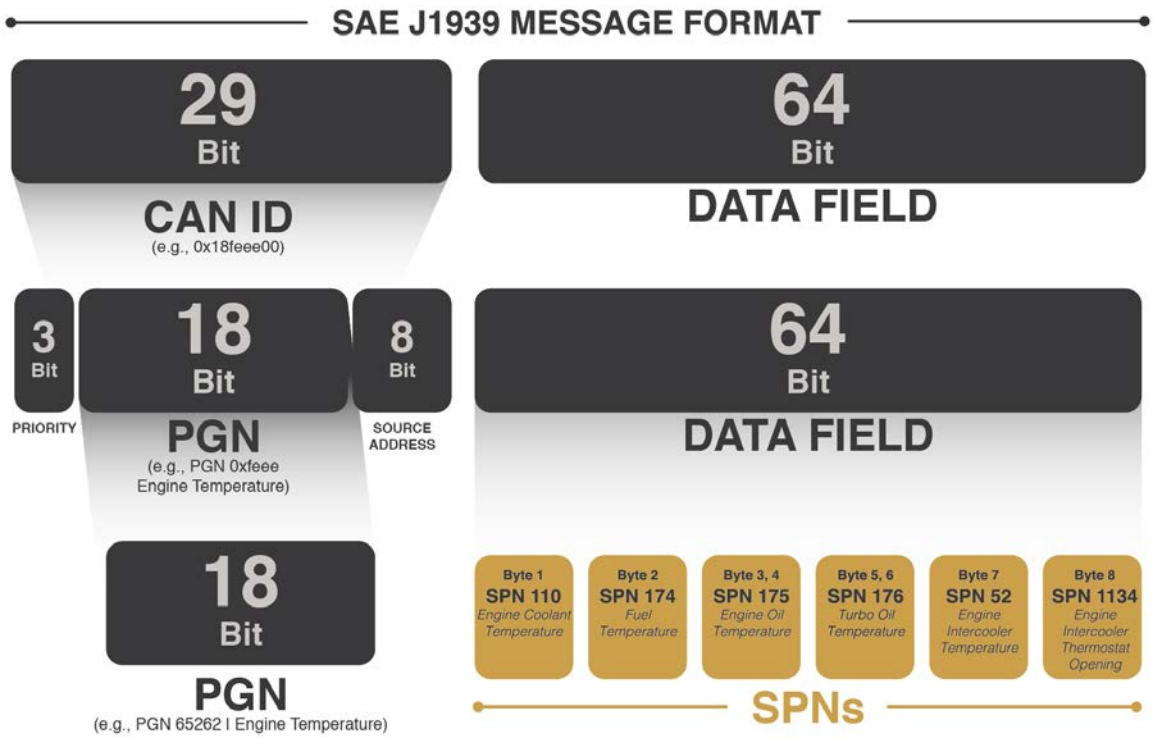
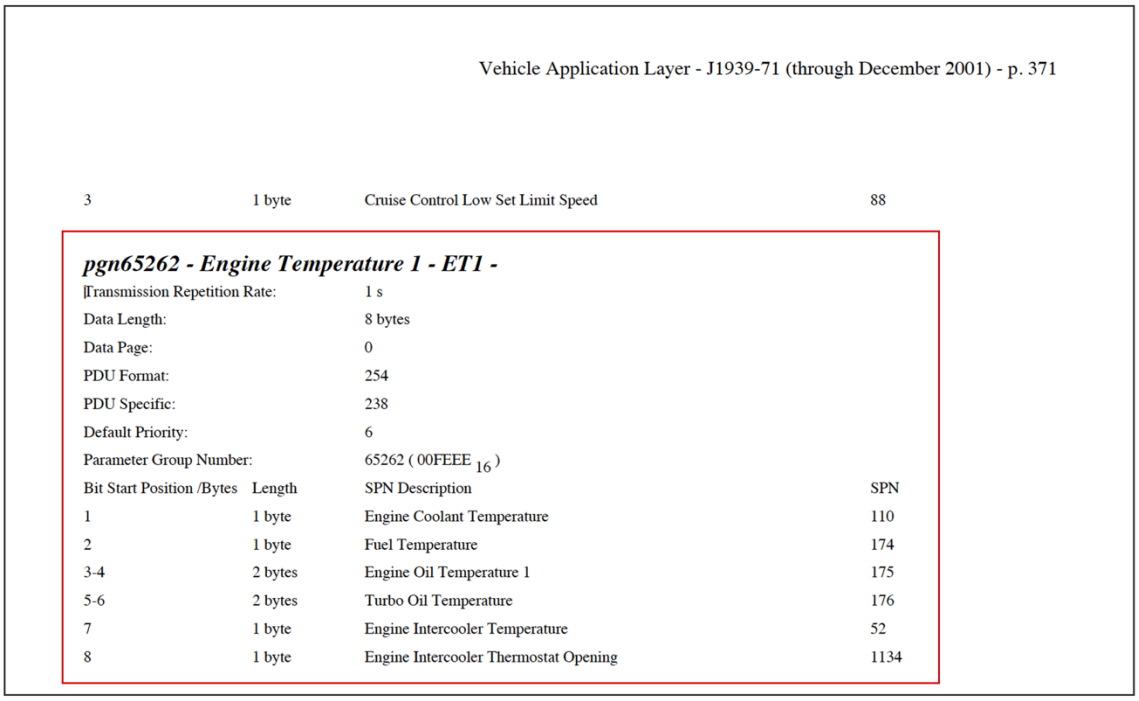


Figure 10. SAE J1939 Message Frame Architecture.
Adapted from [19], [20].

To interpret this information, the “SAE J1931/71 Vehicle Application” reference document is used to look up the specific PGN ID. This document is one of several reference documents in the J1939 Standards Collection that defines message functions, conversion rules, and the bit-level logic required to convert cross-manufacturer J1939 messages into human-readable data [20], shown in Figure 11. PGNs along with their grouped SPNs are listed and defined in the J1939 Standards Collection. Using the lookup list in this document, the example PGN 65262 referenced earlier decodes to "Engine Temperature 1 - ET1" [51, p. 371].



Vehicle Application Layer - J1939-71 (through December 2001) - p. 371

3	1 byte	Cruise Control Low Set Limit Speed	88

pgn65262 - Engine Temperature 1 - ET1 -

Transmission Repetition Rate:	1 s
Data Length:	8 bytes
Data Page:	0
PDU Format:	254
PDU Specific:	238
Default Priority:	6
Parameter Group Number:	65262 ($00FEEE_{16}$)

Bit Start Position /Bytes	Length	SPN Description	SPN
1	1 byte	Engine Coolant Temperature	110
2	1 byte	Fuel Temperature	174
3-4	2 bytes	Engine Oil Temperature 1	175
5-6	2 bytes	Turbo Oil Temperature	176
7	1 byte	Engine Intercooler Temperature	52
8	1 byte	Engine Intercooler Thermostat Opening	1134

Figure 11. J1939/71 Parameter Group Definition of PGN65262. Source: [51].

Further, this document defines details associated with the PGN, including priority, transmission rate, and a list of associated SPNs (the data parameters). For this PGN, there are six associated SPNs: engine coolant temperature, fuel temperature, engine oil temperature, turbo oil temperature, engine intercooler temperature, and engine intercooler thermostat opening; each SPN can be looked up in the J1939/71 for further details. The SPNs in the example point to vehicle data associated with Engine Temperature [19]. For every message ID there is an embedded 8-byte CAN data field carrying the actual data for a specific parameter reflected via an SPN. Further details on decoding individual SPNs will be covered in Section B, Chapter IV.

b. Software Tools and the SAE Digital Annex

During live diagnostic extracting, it is impractical to look up specific PGNs and SPNs using the physical J1939/71 reference document. So, to make the information accessible in a more useful way, it was necessary to build a database using the

programming languages Python, SQLite and R, and an electronic version of the SAE J1939/71 reference document.

In this research, Python was used to extract, import, parse, and decode raw J1939 CAN-bus diagnostic data. Python is an interpreted, object-oriented, high-level programming language similar to Java, Perl, or PHP [52]. It is a scripting language that is simple and known for its easily understood syntax relative to other programming languages. These features make Python ideal for text parsing and rapid prototyping. Python is also highly compatible with other software applications and supports add-on modules and packages to enhance its capabilities. Additionally, Python is the default programming language installed on the Raspberry Pi.

SQLite, a highly scalable query language, is used to create a database structured around the data available in the J1939 message ID (PGN) and associated SPNs. A companion spreadsheet for SAE J1939 reference document is available in an Excel format called the SAE Digital Annex, which contains the latest PGNs and SPNs published in the J1939/71. This electronic document is used to build the SAE Digital Annex Database used in this research. SQLite is native to the Python library and can be imported as a module. It offers useful database functionality without the overhead of an entire data management server environment such as MYSQL or Amazon Web Server, making it ideal for small to medium applications for prototyping or testing. Data can be scaled and exported to a larger database or cloud architecture if necessary.

Data analysis was conducted using R, an open-source statistical programming language commonly used in statistical research [53]. R is highly extensible; compatible with Python and SQL; and ideal for exploratory data analysis and linear and nonlinear modeling.

B. VEHICLE DATA FILTERING AND PROCESSING SPNS

This research explores 8 of the 32 parameters extracted:

- SPN 190 Engine Speed (RPM)
- SPN 100 Engine Oil Pressure

- SPN 110 Engine Coolant Temperature
- SPN 174 Engine Fuel Temp
- SPN 84 Wheel-based Vehicle Speed
- SPN 105 Engine Intake Temp
- SPN 177 Transmission Oil Temp
- SPN 183 Engine Fuel Rate

The CAN-bus transmits log data at 250 Kbit/sec which equates to a logged time stamp every millisecond [19]. Data was collected over an 8-hour period. Due to the immense amount of timestamp observations, we down-sampled the data from microseconds to seconds while averaging the data in-between. Down sampling is a common technique used in statistics to remove some data points by averaging out (or through some other mathematical function) data points, creating new data points representing the groups of points removed.

1. Data Collection Criteria

To ensure the data represented realistic operating conditions, the data was collected under the following conditions:

- The MTVR was on and sufficiently warm.
- Engine speed was above idling speed.
- The driving course allowed for a variety of terrain to reveal dynamic vehicle conditions. Figure 29, Appendix D depicts driving course.
- Operating time for data collection was conducted in 45-minute intervals.

Data collected while the MTVR was in idle before the clutch was engaged was not used (for example, starting the truck). Data was collected over a period of different days;

however, for this research, only one day of data will be used. The MTVR carried no load during data collection.

2. Interpreting and Parsing MTVR Raw Sensor Data

The placement of the sensors by the diesel engine manufacturer is shown in Figure 28 in Appendix C. These sensors provide diagnostic measures on engine component temperatures, various speed measures, fluid levels, and electrical on-off-switch indicators based on data transmitted through the CAN-bus network. The log data from these ECUs is extracted via the J1939 9-pin connector and a Raspberry Pi with SAE Interface board attached, as described in Section A (Figure 8, 9). Raw CAN-bus data logs a timestamp, CAN-bus logger port location, a J1939 message ID in hexadecimal, and the associated SPN control functions. Figure 12 shows the raw sensor data output from the MTVR.

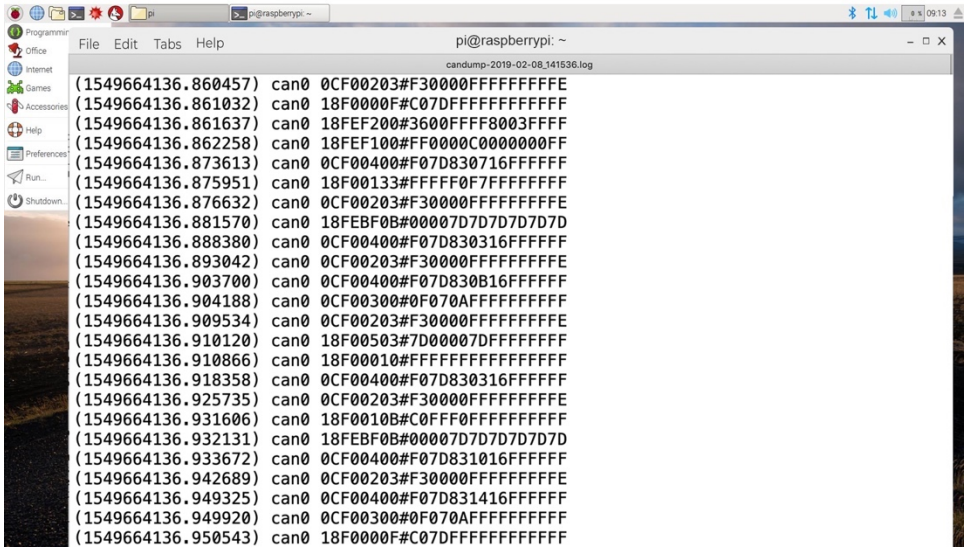
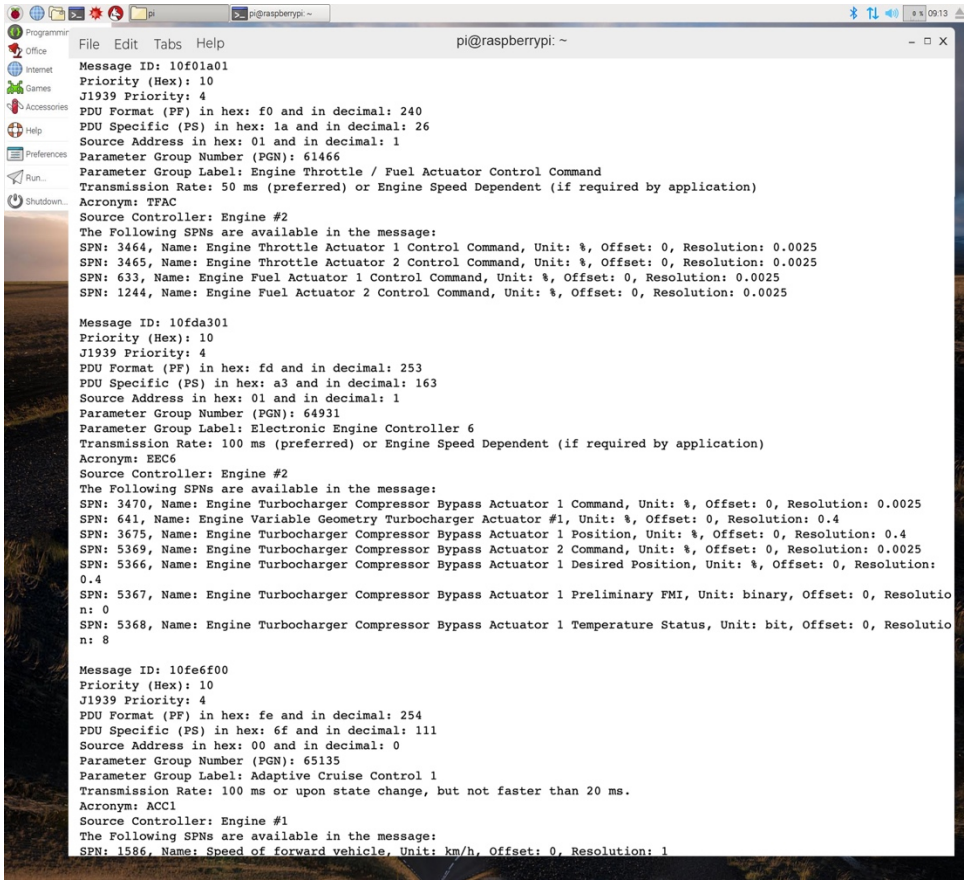


Figure 12. CAN-bus Logging Output Screenshot

Utilizing Python programming software and a relational database, we can read, parse, and decode each line of log data outputted from the MTVR in real-time. Annex A shows the Python script created to decode, parse, and store J1939 CAN-bus log files, adapted from [54]. Hexadecimal values of PGN message IDs are converted to decimal values. The decoded PGN and its associated SPNs are queried against the SAE J1939

database (which is created and loaded from the SAE Digital Annex) to convert J1939 diagnostic message into readable diagnostic parameters. For every time stamp observation, the output provides a description of each SPN (data parameters), which includes the data length, resolution, operating value range, and data type of the measured MTVR data parameters, shown in Figure 13. This information can be manually looked up in the "SAE J1939-71 Vehicle Application Layer" reference, but as mentioned in Subsection 2, Section A, it is impractical to look up this information during real-time data logging due to the amount and speed of data transmission. Over four million rows of data was decoded and parsed, totaling over 6 hours of driving time, to create the diagnostic dataset for analysis.



PGN message IDs are decoded to decimal values and associated SPN descriptions of conversion functions.

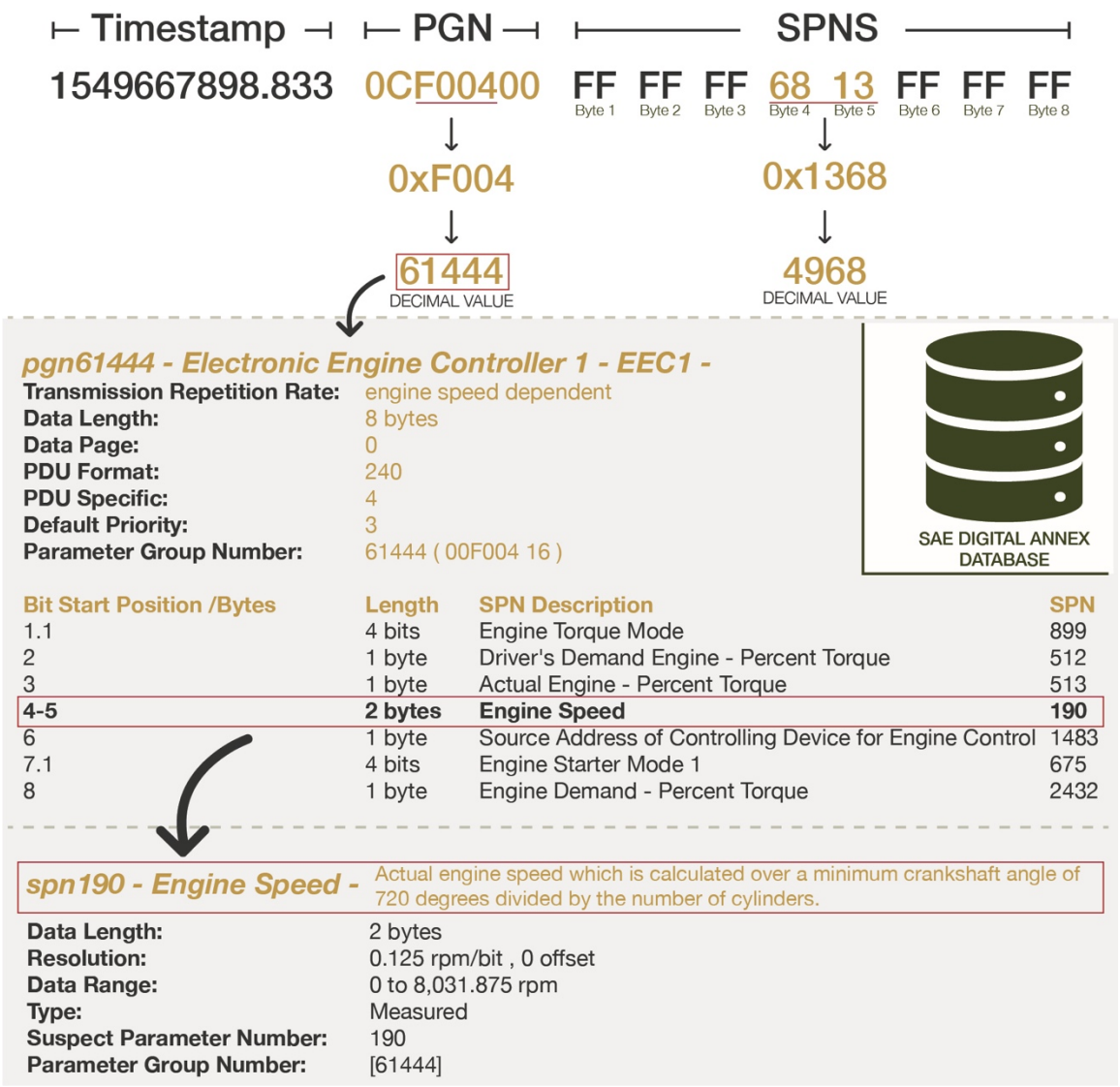
Figure 13. Decoded J1939 Message of PGNs and SPN Conversion Rules from Raw CAN-bus Data

3. Decoding SPNs

The decoded J1939 messages are used to convert SPN parameters to scaled engineering values that measure parameter changes over time. To convert SPN observations, we extract the proper number of data bytes from the 64-bit data message according to the conversion rules for this PGN message ID. These conversion rules are defined in the SAE J1939/71 reference document. Figure 14 shows a single observation from our log data that decodes to PGN 61444, Electronic Engine Controller 1 with SPN 190, engine speed (RPM). The raw message ID is 0x0CF00400. We parse the PGN value from this message ID starting at bit 9. The outcome is PGN 0xF004, which is 61444 in decimal.

Figure 14. One observation Sample from CAN-Bus Data

That value is then referenced using the J1939 protocol. For this research it was queried against the created SAE Digital Annex database. From the J1939 Standard, we identify that there are hexadecimal values in the byte position 4 and 5, 0x68 and 0x13 respectively, of the SPN data frame. Note how some SPN data bytes in Figure 14 are FF or 255 decimal, which means “not available” or “not applicable.” The SAE Digital Annex database is then queried again to retrieve the correct conversion rules for each message ID based on the J1939 protocol. Figure 15 shows the conversion rules related to the SPN in byte position 4-5 of PGN ID 61444, which is SPN190, Engine Speed. In the case of the SPN relevant to data bytes 4 and 5 of Figure 15, Intel Byte Order is used to swap bits (due to how large bits are stored in serial bus memory) taking the decimal form of 0x1368. The values are converted to decimal 4968. We then use the J1939 conversion rules, specifically the resolution and the offset, to scale this value to actual engine speed. Scaled values are computed as the raw decimal value of the SPN multiplied by the scale factor (the resolution) plus the offset, as expressed in Equation 5.



PGN IDs are queried from the database to retrieved SPN functions which then are referenced for their conversion rules to scale bit values to measured values.

Figure 15. J1939 SPN Conversion Rules for Engine Speed. Adapted from [51].

Each SPN has its own conversion rules, but the same scaling formula is utilized. To arrive at the actual RPM, we take the raw decimal value 4968 multiplied by 0.125 RPM/bit plus an offset of 0.

$$ScaledValue = Offset + Scale * RawDecimalValue \quad (5)$$

The result is 621 RPM.

Figure 16 shows the converted output of the MTVR diagnostics for Engine Speed (RPM). We repeat this method for all other SPNs in our PGN message ID. This shows us the change in our engine parameters over time. With this method all PGN and SPNs available are parsed, decoded, and converted to measured values to be used for further data analysis.

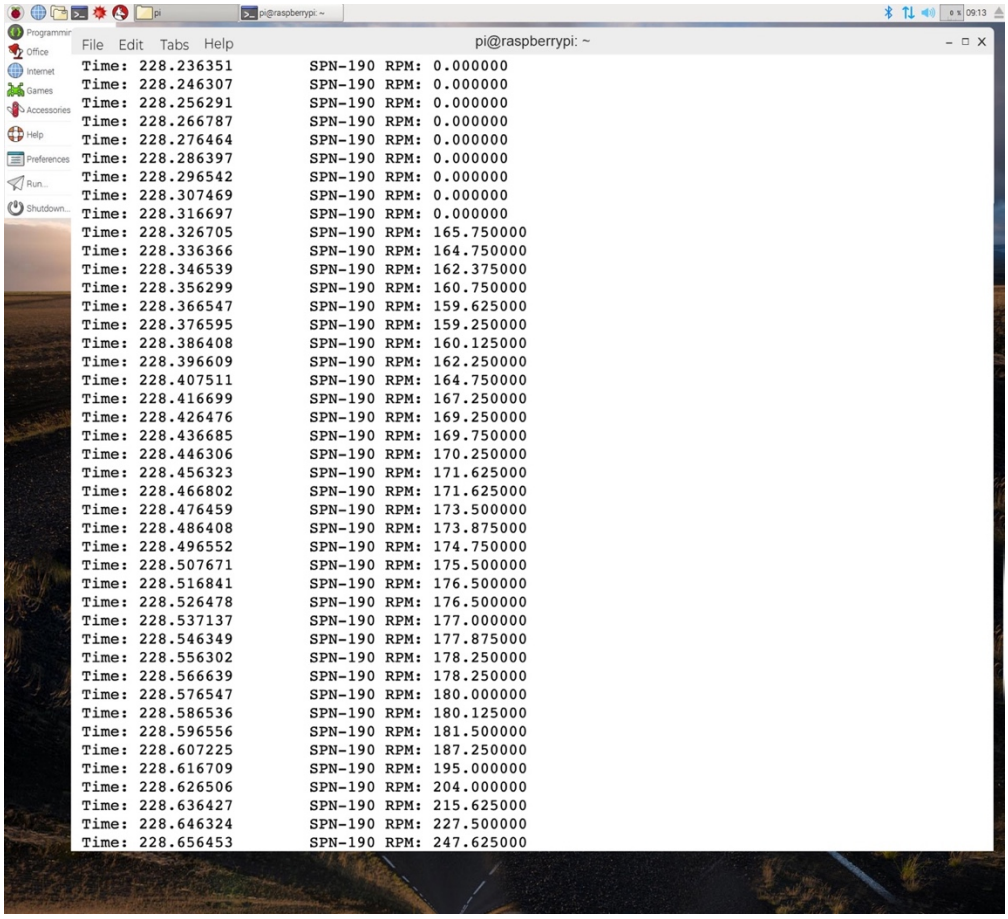


Figure 16. Converted J1939 Message Data for SPN190 Engine Speed

C. DATA PROCESSING AND ANALYSIS

After the CAN-bus log data was cleaned, parsed, and decoded into measurable values, the data was loaded into a database for further analysis. Utilizing R statistical software [53], "not applicable" SPNs were discarded, leaving 19,949,624 observations of diagnostics totaling over five hours of operating time and eight measurable parameters

relating to engine components for analysis. In this section, first, we describe the characteristics of the data collected from the MTVR, followed by the analysis performed. Next is an assessment of the usable diagnostic parameters to be used for further analysis. Finally, we describe the statistical model used to support a CBM approach.

1. Datasets

In order to reduce the number of observations and see noticeable change in measured diagnostic parameters, the data was down-sampled from microseconds to seconds. To handle missing values from down-sampling, we used linear interpolation between missing points to approximate the mean. This yielded a smoother time series with low bias. Specifically, the *imputeTS* package [55] available in R was used for "NA" interpolation. The final dataset resulted in a data-frame of eight diagnostic parameters and 551 observations. This dataset was comprised of data collected from one MTVR, as a result, we split the dataset into 30-minute increments, creating a total of 12 datasets of diagnostics for modeling. Two out of the 12 datasets contained time intervals where no data was actively collected. These datasets were discarded and explain the gap in our time series from 18:39:58 to 20:01:28. Table 2 shows the data and duration of the 10 datasets used for analysis.

Table 2. Data and Duration of Diagnostic Data

Dataset	Date	Duration (30 min)	Observations
A	20 Mar 2019	15:50:00	59
B	20 Mar 2019	16:20:00	60
C	20 Mar 2019	16:50:00	60
D	20 Mar 2019	17:20:00	58
E	20 Mar 2019	17:50:00	60
F	20 Mar 2019	18:20:00	39
I	20 Mar 2019	19:50:00	38
J	20 Mar 2019	20:20:00	60
K	20 Mar 2019	20:50:00	60
L	20 Mar 2019	21:20:00	57

Dataset G and Dataset H removed due to being empty datasets

The data was collected from an MTVR driving through a variety of road conditions. Figure 29, in Appendix D displays a map of the driving course. Data was not collected until the clutch was engaged and the engine sufficiently warm. This was to allow variability in our datasets. Idle engines do not have changing values. Maximum slope of driving course was no more than 10 degrees or 17%. Maximum speed was no more than 55 miles per hour with a minimum speed of 0 due to traffic conditions. Of note, the time stamps do not equate to the actual time the data was collected due to being set to the system time of the Raspberry Pi, which was not accurate. This has no effect on the outcome of our model.

2. Summary of Data Parameters

The parameters and their range of values are summarized in Table 3. Engine Speed is measured and recorded from a range of 0 to 8,000 RPM. The MTVR high-idle speed is 1,500 RPM. The average RPM speed from the data collected is 1,231 RPM. Coolant, fuel, engine intake manifold, and transmission oil temperature values are measured in degrees Celsius ranging from -40 to 210 degrees. The average operating temperature of engine coolant and fuel temperature are 85.5 degrees Celsius and 44.6 degrees Celsius, respectively. Engine intake manifold temperature and transmission oil temperature averaged operating values of 23.9 degrees Celsius and 80 degrees Celsius. Engine fuel rate measures the amount of fuel consumed by engine per unit of time. It is measured in liters per hour ranging from 0 to 3,212.75 liters per hour. The average fuel rate from our observations is 21 liters per hour which is approximately 5.56 gallons per hour. Engine oil pressure measures the gage pressure of oil in the engine provided by the oil pump. It is measured in kilopascal (kPa) with an operating range of 0 kPa to 1000 kPa. The average oil pressure from our observations is 234 kPa. Our last parameter used for analysis is wheel-based vehicle speed. This measures the speed for the vehicle as calculated from the wheel. The operating range is 0 to 250 kilometers per hour (km/h) with an average of 46.5 km/h (28 mph) from our observations. Measured values that operated outside the data range for each specific parameter based on the conversion rules were converted to "NA" and discarded.

Table 3. MTVR Diagnostic Parameters

Parameters	Units	Minimum Value	Maximum Value
Engine Speed	RPM	0	8,031.875
Engine Coolant Temperature	Degree Celsius	-40	210
Engine Fuel Temperature	Degree Celsius	-40	210
Engine Oil Pressure	kPa	0	1000
Wheel-based Vehicle Speed	km/h	0	250.996
Engine Fuel Rate	L/h	0	3,212.75
Engine Intake Manifold Temperature	Degree Celsius	-40	210
Transmission Oil Temperature	Degree Celsius	-273	1735

Information based on "J1939-71 Vehicle Application Layer" Reference

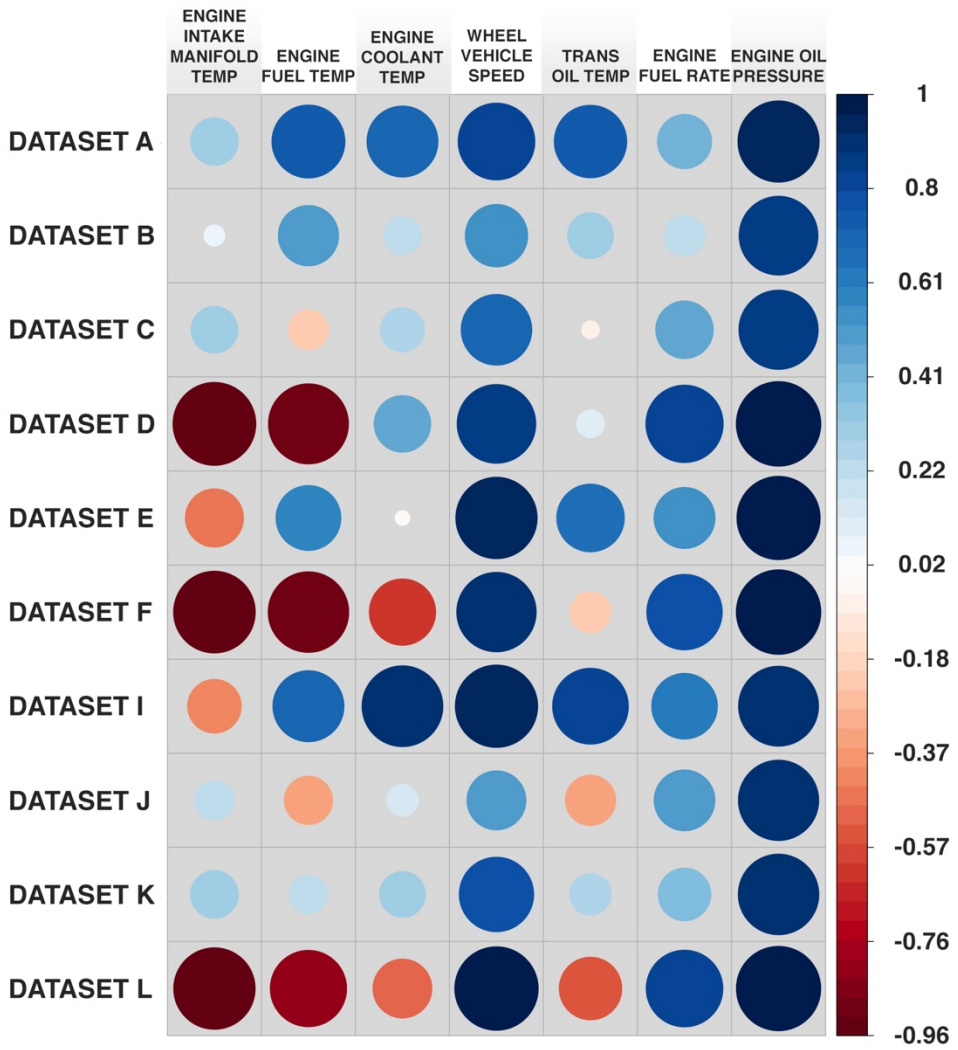
3. Parameter and Dataset Selection

A correlation matrix was created for each dataset to measure the strength of the linear relationship between pairs of parameters. Table 4 shows the correlation between engine speed and the other parameters. Highlighted in red are values with strong negative correlation. Highlighted in blue are values with strong positive correlation according to our dataset. The strongest correlation coefficient for each dataset is highlighted in bold. For this research, values greater than 0.7 and less than -0.7 are considered a strong correlation. The correlation coefficient of engine speed to engine oil pressure suggest there is a strong correlation across our datasets. Engine speed is an independent variable and the correlation matrix suggest there is a strong relationship between engine speed and engine temperatures.

Table 4. Correlation Table between Engine Speed and Remaining Parameters

Dataset	Engine Intake Temp	Engine Fuel Temp	Engine Coolant Temp	Wheel Vehicle Speed	Trans Oil Temp	Engine Fuel Rate	Engine Oil Pressure	
A	0.3165	0.7431	0.7027	0.8221	0.7331	0.4105	**0.9198**	**RPM**
B	0.0620	0.5072	0.2029	0.5442	0.2943	0.2330	**0.8659**	**RPM**
C	0.3042	-0.2185	0.2727	0.6960	-0.0446	0.4626	**0.8753**	**RPM**
D	-0.9567	-0.8926	0.4479	0.8633	0.1074	0.8343	**0.9967**	**RPM**
E	-0.4723	0.5878	-0.0307	0.9235	0.6385	0.5193	**0.9665**	**RPM**
F	-0.9309	-0.9044	-0.6134	0.8802	-0.2362	0.7945	**0.9955**	**RPM**
I	-0.4021	0.7034	0.9104	**0.9462**	0.8017	0.6021	0.8906	**RPM**
J	0.2168	-0.3263	0.1441	0.4862	-0.3547	0.5159	**0.8996**	**RPM**
K	0.3185	0.2080	0.2929	0.7728	0.2411	0.3867	**0.9042**	**RPM**
L	-0.9247	-0.8120	-0.4836	0.9708	-0.5464	0.8197	**0.9916**	**RPM**

We observe weak relationships across datasets J, K, C, B, and E. Therefore, these datasets are not used for further analysis. Datasets A, D, F, I, and L show the strongest relationships across the correlation coefficients which suggest these datasets better reveal the behavior of engine operating conditions. Figure 17 shows the correlation density plot of Table 4. This figure shows that engine oil pressure has the highest correlation coefficient across datasets followed by wheel-based vehicle speed, engine fuel temperature, and then engine intake manifold temperature.



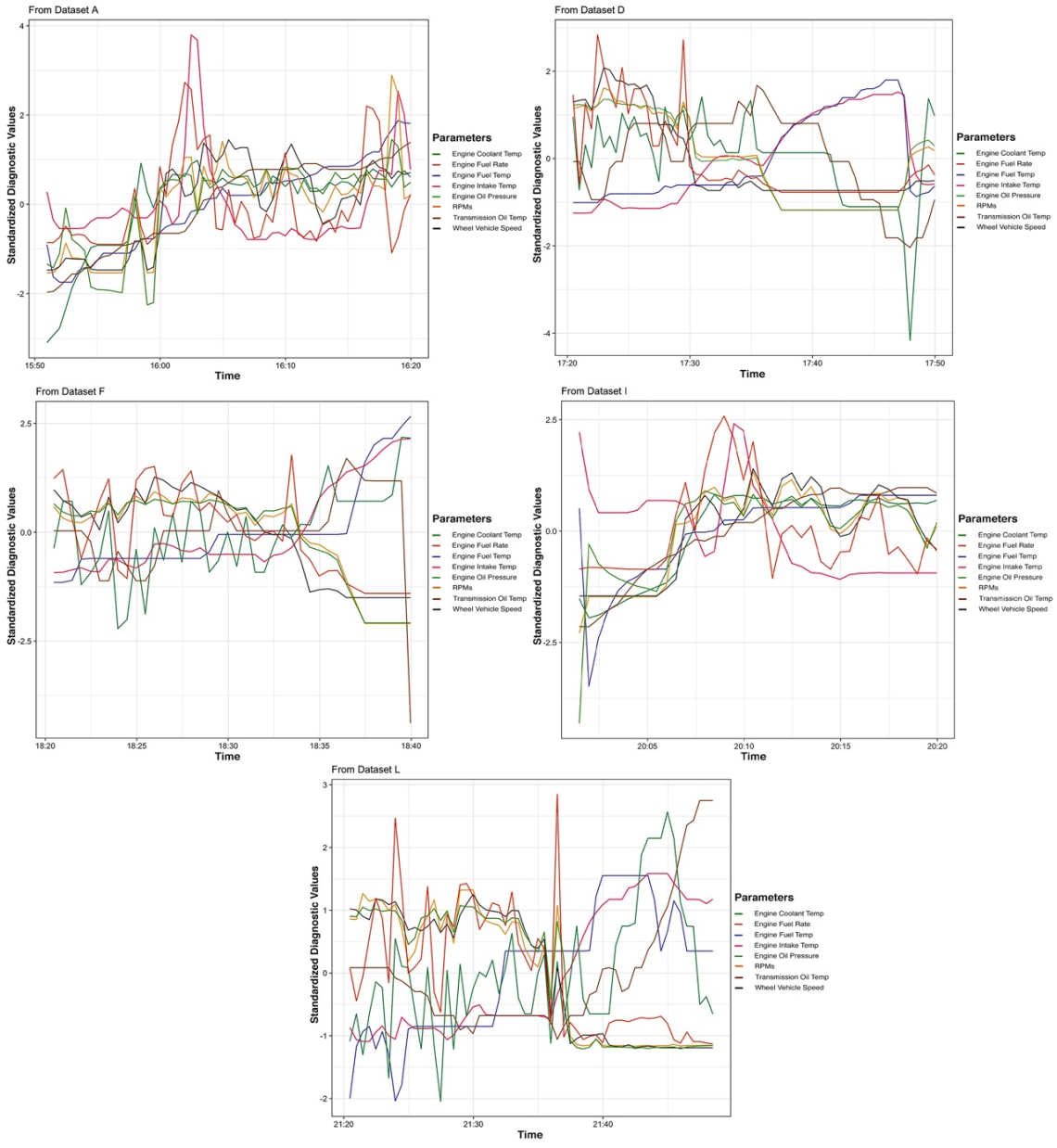
Circles sizes and colors change and shift relative to their correlation value displayed in the number line on the right. Large blue circles suggest strong positive correlation. Large red circles suggest strong negative correlation.

Figure 17. Correlation Density Plot of Engine Speed by Datasets

4. Data Plots

We plot each parameter against time to observe the behavior of the engine across our datasets. Plots are shown in Figure 18. We standardize our diagnostic values to plot them on the same scale so we can observe how each parameter interacts across the dataset with respect to other parameters. Specifically, the data plots show how each parameter responds to engine functions over time. The largest trends across are datasets are the observation of engine speed (RPMs) as a function of fluid temperature parameters. As

engine speed increases, temperature values respond. Engine fluid temperatures exhibit similar trends across datasets A, F, and I. In dataset D we identify points where coolant temperature drops substantially with large positive spikes.



Top-left: Dataset A. Top-right: Dataset D. Bottom-left: Dataset F. Bottom-right: Dataset I. Bottom-center: Dataset L.

Figure 18. Standardized Diagnostic Plots of Datasets

The drop in engine oil temperature in dataset F is explained by discarded datasets G and H. We conclude that engine oil pressure closely follows engine speed, which was indicated by the correlation matrix in Table 4 and Figure 17. Further investigation is needed to examine how each variable is related to engine speed.

5. Subset Selection

Datasets F and L are chosen as our final modeling datasets. dataset F will be used to fit our model and dataset L will be used to test our model. These datasets show the strongest correlation of engine speed across parameters. Specifically, datasets F and L show how engine oil pressure closely models engine speed. From our chosen datasets we display a matrix of scatter plots of all pairs of parameters (a pairs plot) to search for interaction and relationships among predictors against the response. In Figure 19 we observe that across the time stamp row our temperature parameters show a mostly linear relationship as time of engine operation increases. RPMs and wheel-based vehicle speed share a similar trend across temperature parameters which suggest they may be correlated. Engine oil pressure and RPMs show the strongest linear relationship across our datasets. No distinct patterns can be determined for engine coolant temperature, engine fuel temperature, and engine intake manifold temperature in our pairs plots.

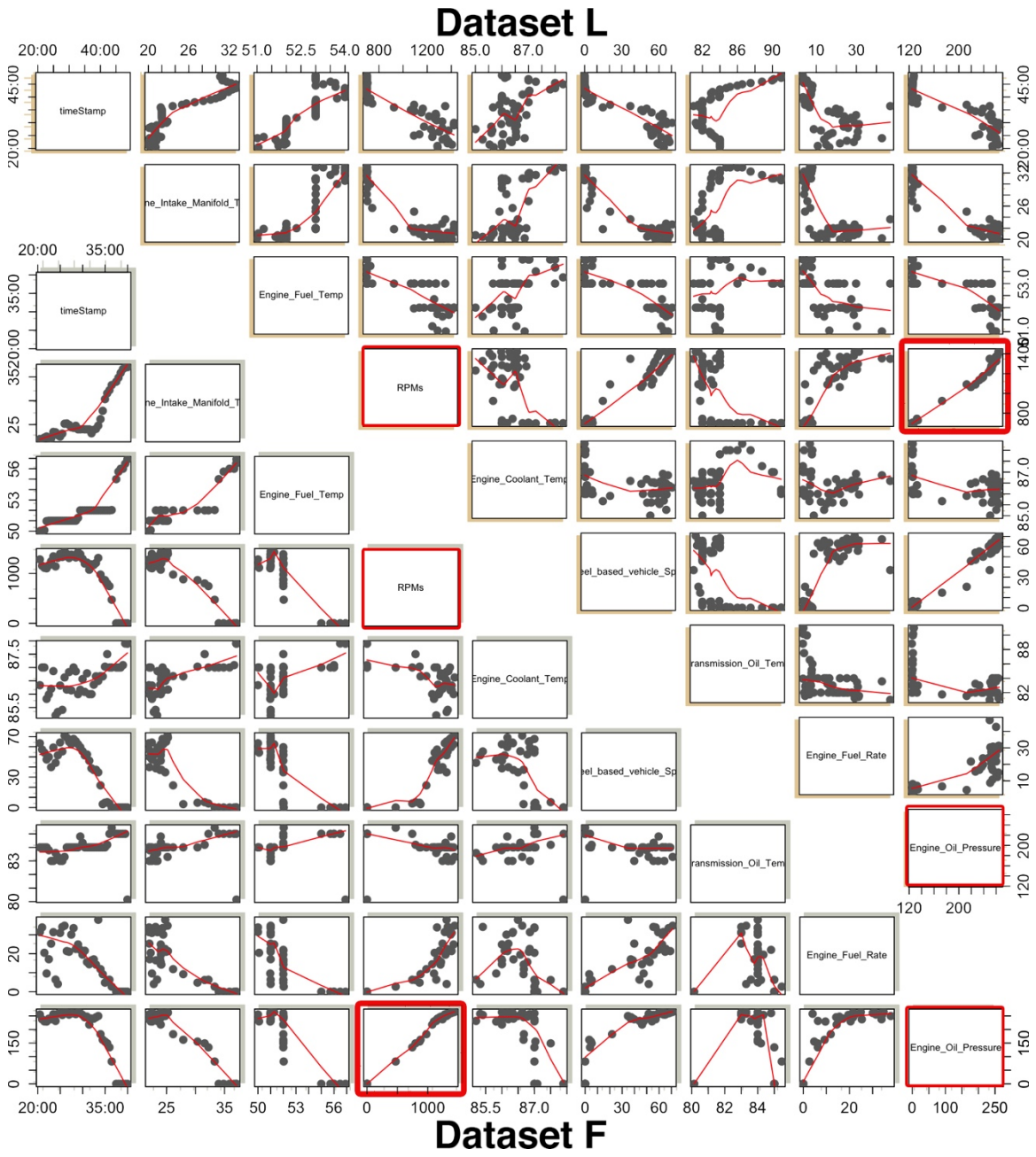


Figure 19. Pairs Plot Dataset F and L

6. Using Engine Oil Pressure as a Predictor of Engine Usage Conditions

From our exploratory analysis on the data extracted from the MTVR and our correlation matrix charts, we determine that engine oil pressure closely characterized the condition of the engine in relation to engine speed. Monitoring engine oil for maintenance

indicators is one of the earliest methods of preventative maintenance. Historically, engine oil conditions describe how an engine is operating over time. Oil pressure measures the amount of oil and lubrication moving through the engine components, as engines wear down or oil gets low or dirty, engine oil pressure responds as an early indicator to an issue. An engine's efficiency and "remaining useful life" can be characterized by the frictional loss of power related to its cylinders and pistons [56]. Engine oil pressure measures this frictional viscosity revealing early indicators of performance or degradation issues relating to engine components [56]. Our data shows that engine speed may closely predict engine oil pressure which may function as a good indicator of engine usage conditions. We can observe from our data plots that after the engine is sufficiently warm, oil pressures closely models the operation of the engine. Our statistical model will characterize the behavior of engine oil pressure to determine "normal" operating conditions of the engine.

7. Building an Autoregressive Distributed Lag Time-series Model

We use an autoregressive distributed lag model to characterize engine usage. This model, expressed in Equation 4, shown in Chapter III, Section B, Subsection 3, uses r and p to specify respective lag lengths of our dependent variable, y_t, which is engine oil pressure and our independent variable, x_t, engine speed. The coefficients, $i = 1,2,\ldots,r$ and $j = 1,2,\ldots,p$ are the estimated coefficients for our engine oil pressure and engine speed, respectively. From our datasets, the number of lags are represented by one 30-second time length from the previous observation (this is based on how the datasets were split earlier). Autocorrelations determine how each observation relates to its past observations. In Figure 20, we show the autocorrelations associated with dataset F. The time lag is indicated on the horizontal axis depicted as "Lag Length." The height of each vertical line estimates the value of the estimated autocorrelation at that lag. The plot shows that autocorrelations are largest at the low lags on the left and decrease as the lag increases to the right. This implies that each observation is positively associated with its recent past, but the association becomes weaker as the lag increases. Engine oil pressure series shows strong persistence, meaning that the current value is closely related to those that proceed it. While we are interested in determining how a series of observations relates to its most

recent past, we must test to ensure that the error values of our final model do not follow this pattern. This will prevent our model predictions from being incorrectly specified or invalid.

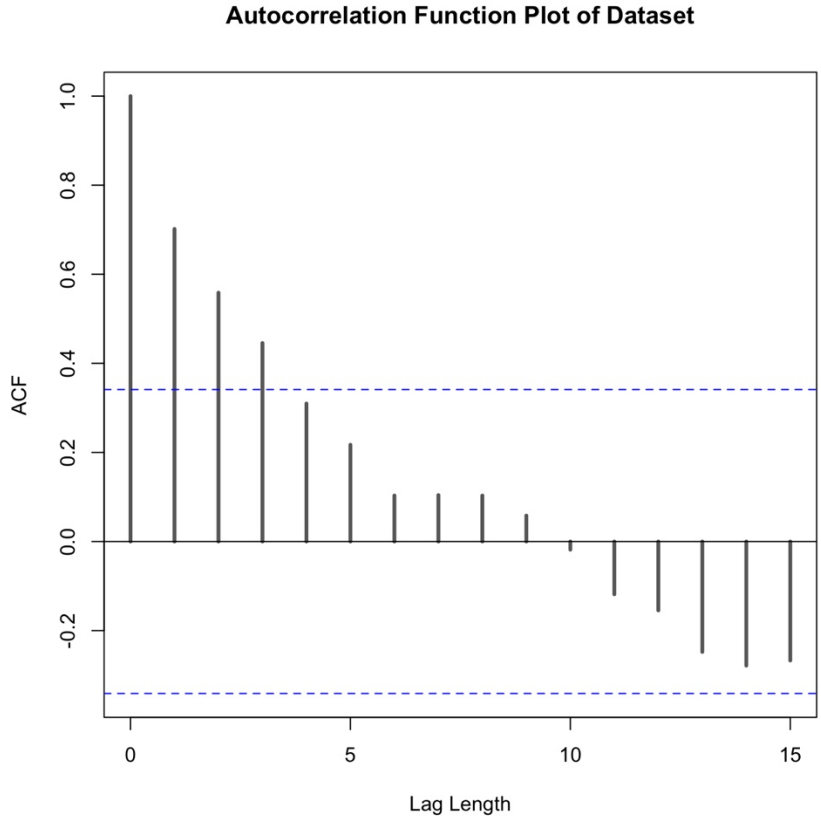
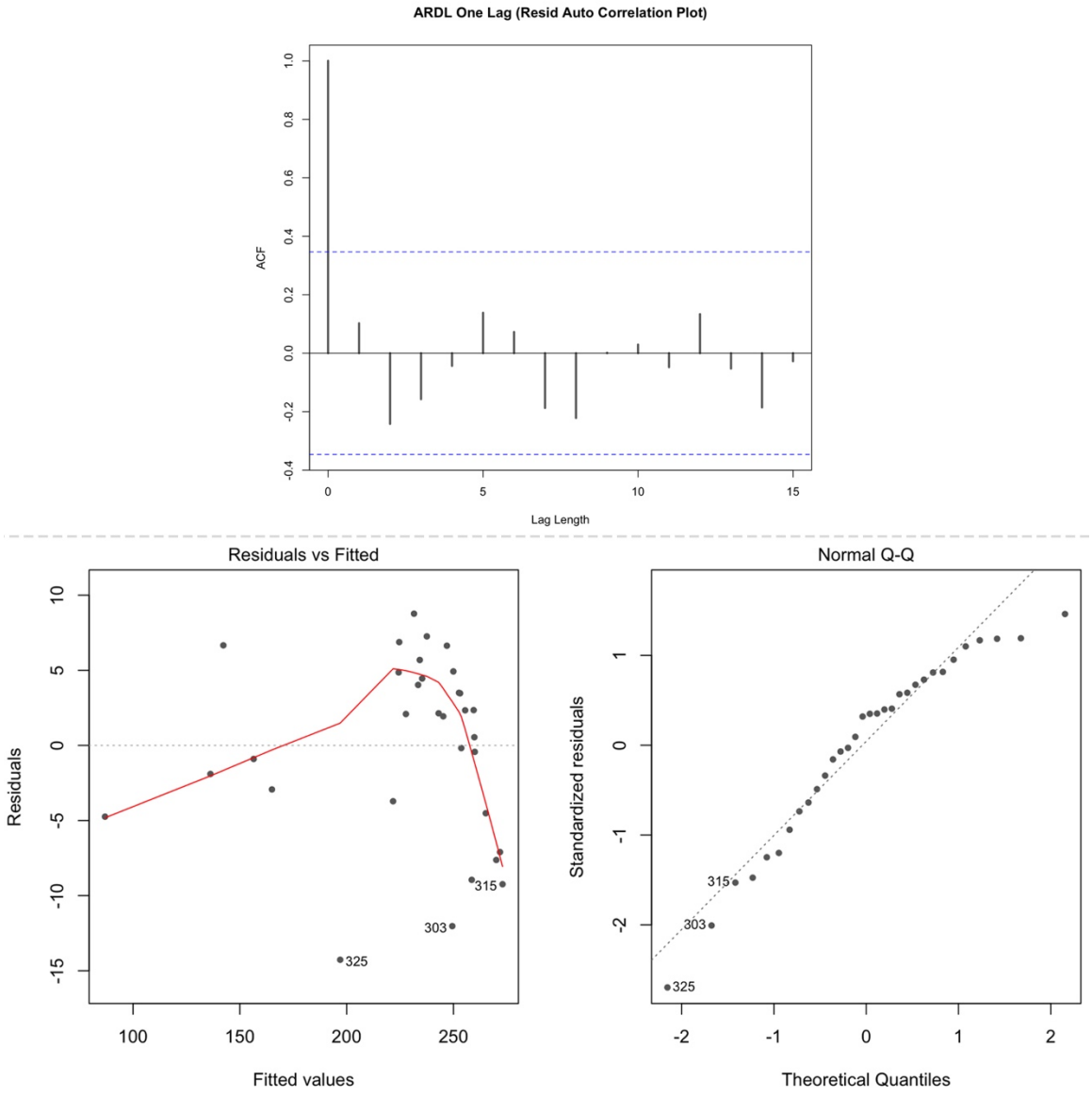


Figure 20. Autocorrelation Function Plot of Dataset

For our dataset we need to ensure the residuals of our fitted model shows no indication of autocorrelation. Our goal is to find optimal lag lengths for r and p that best characterizes engine conditions. Using the R language function "lm" [53], we build a linear model against our predictor with engine oil pressure as the response. To determine the optimal lengths for r and p we fit ARDL models with lags 1, 2, 3, 4, and 5. The results of all five of these models were similar and we present only the results of ARDL time series with one lag ($r = p = 1$) and one with five lags ($r = p = 5$).

a. ARDL With One Lag

We introduce one lag into our time series model. In Figure 21, we display the autocorrelation of the residuals, the residuals vs fitted, and the quantile-quantile (QQ) plot of the residuals for normality.



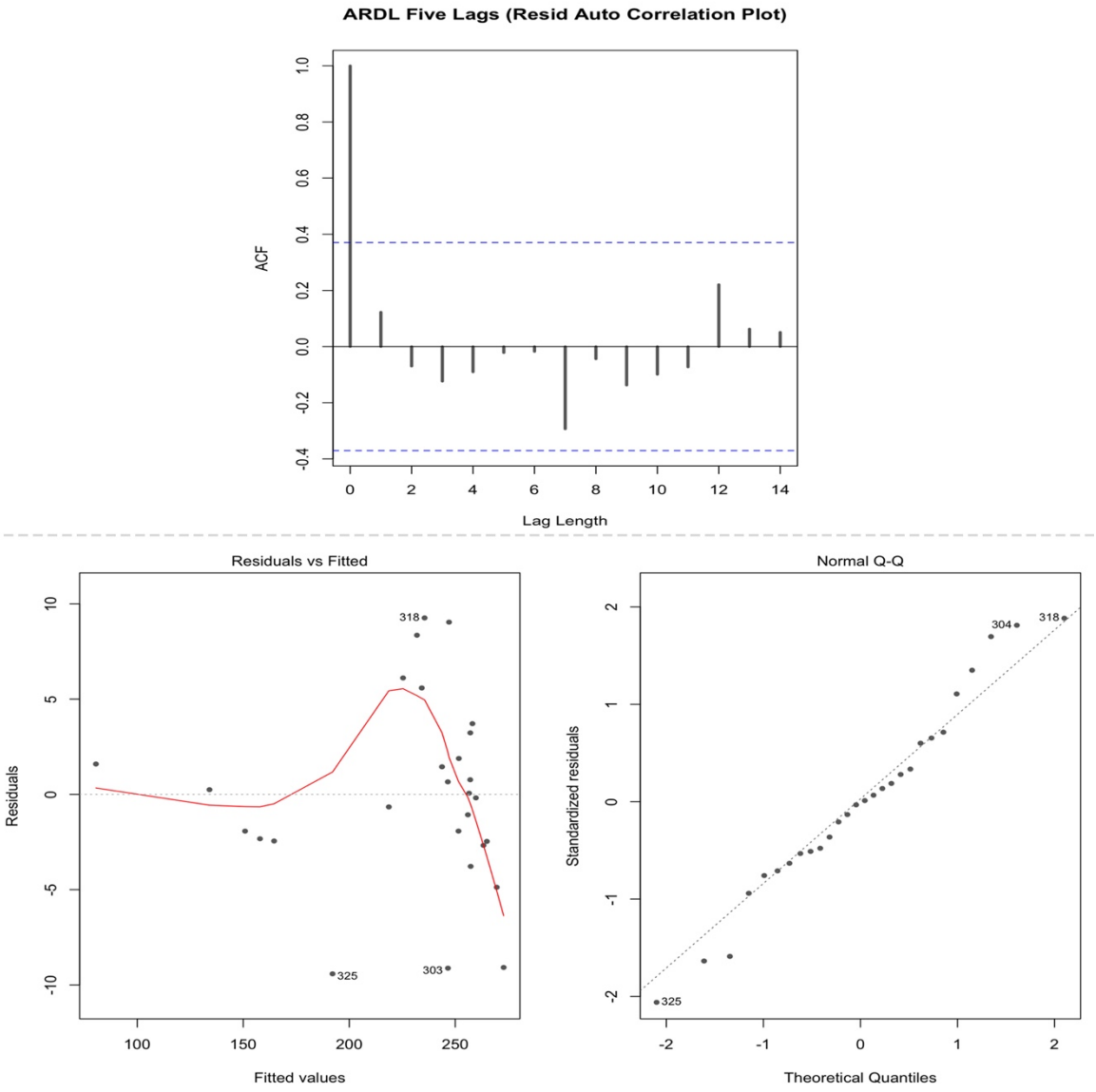
Top-center: auto correlation function for time series with one lag, bottom-left: plot of fitted values against residuals, bottom-right: QQ normal plot of standardized residuals

Figure 21. Auto Correlation and Residual Plots with One Lag

We run a Durbin Watson (DW) statistics test for autocorrelation in the residuals from our ARDL model. The results show no autocorrelation detected in our sample with a DW value of 1.7611 and a p-value of 0.1601. We do not reject the null hypothesis that the error terms are not auto-correlated in accordance with the Durbin Watson significance tables. The residuals vs fitted plot in Figure 21 suggest unequal variance in our model, but this is accounted for by the small number of observations with smaller fitted values in our data set from down sampling. Of note, we used Cook's distances to determine if our data had any influential observations, and standardized residuals to identify any outliers. Neither set of diagnostic statistics revealed influential observations or outliers. The plot revealed no outliers or leverage points in our data. Observations with measured values of 0 in our dataset were removed and identified as an administrative error due to data being collected while the MTVR was stopped and reset between runs.

b. ARDL With Five Lags

We introduce five lags into our time series model. In Figure 22, we display the autocorrelation of the residuals as well as the residual versus fitted plot along with the QQ plot of the residuals for normality. The QQ plot suggest slight skewness and heavy tails in our five-lagged model. Residuals vs fitted plot reveals similar variance issues observed with one-lag model. Autocorrelations reveal no statistically significant time periods and suggest residuals are not autocorrelated. Durbin Watson test confirms assumptions from autocorrelation function plot.



Top-center: auto correlation function for time series with five lag, bottom-left: plot of fitted values against residuals, bottom-right: QQ normal plot of standardized residuals

Figure 22. Auto Correlation and Residual Plots with Five Lags

c. Lag Length Results and Summary

To ensure we do not overfit we keep the model with one lag. Table 5 shows our model summary. The model describes the *intercept* and *coefficients* of the dependent and independent variables with one lag.

Table 5. Linear Model Regression for Time Series One Lag

Terms	Coefficients	Terms	Coefficients
μ	-12.04	x_t	0.16
y_{t-1}	0.87	x_{t-1}	-0.13

D. SUMMARY STATISTICS (PREDICTION RESULTS)

Simple models are good for a variety of reasons: they are practical, easy to interpret, and repeatable across data domains. Using an ARDL times series model with one lag we predict engine oil pressure. Verification and summary of prediction results follows.

1. Prediction

The prediction results of our model were fitted and validated against actual values of engine oil pressure from dataset F. Figure 23 shows the overlay of predicted results versus actual results in our dataset.

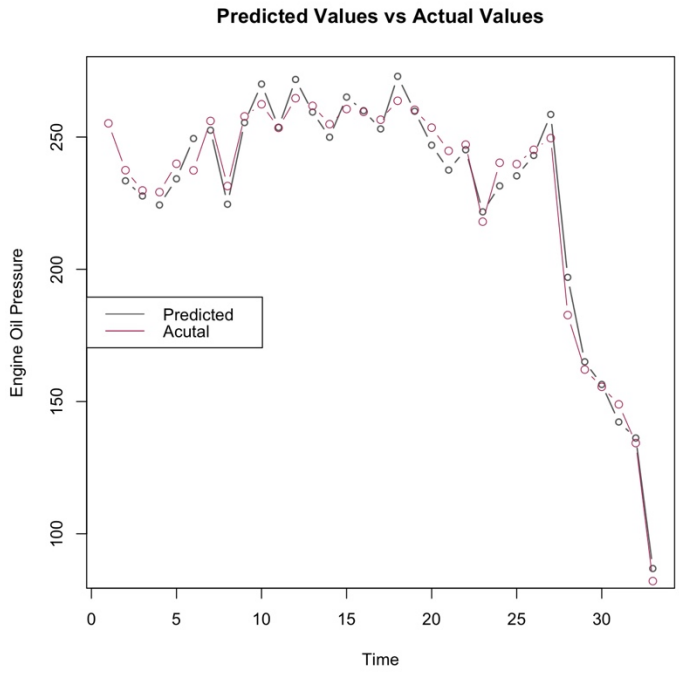


Figure 23. Plot of Predicted Engine Oil Pressure Versus Actual

From our validated results we observe a very high prediction trend with little deviation from actual values. The R-squared value is used to measure goodness of fit for a model using values 0 to 1. For our model, the R-squared is 0.98. We used dataset L as our test dataset, which was set aside before model fitting, to verify the predicted results of our model on new data. In Figure 24 we see the plot of predicted values from our model against new data from dataset L.

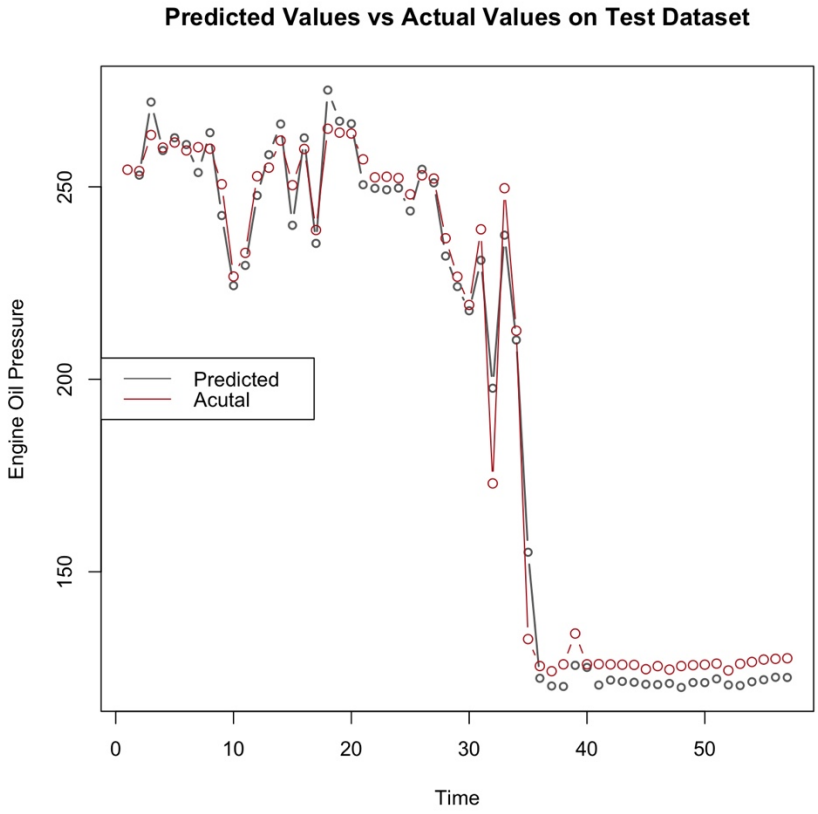


Figure 24. Plot of Predicted Engine Oil Pressure versus Actual on New Data

2. Residuals

The residuals reveal the variability of the predictor variable. Our plots show little deviance from actual values of engine oil pressure. We analyze the difference between

our observed values and predicted values given by our model on dataset L, shown in Figure 25.

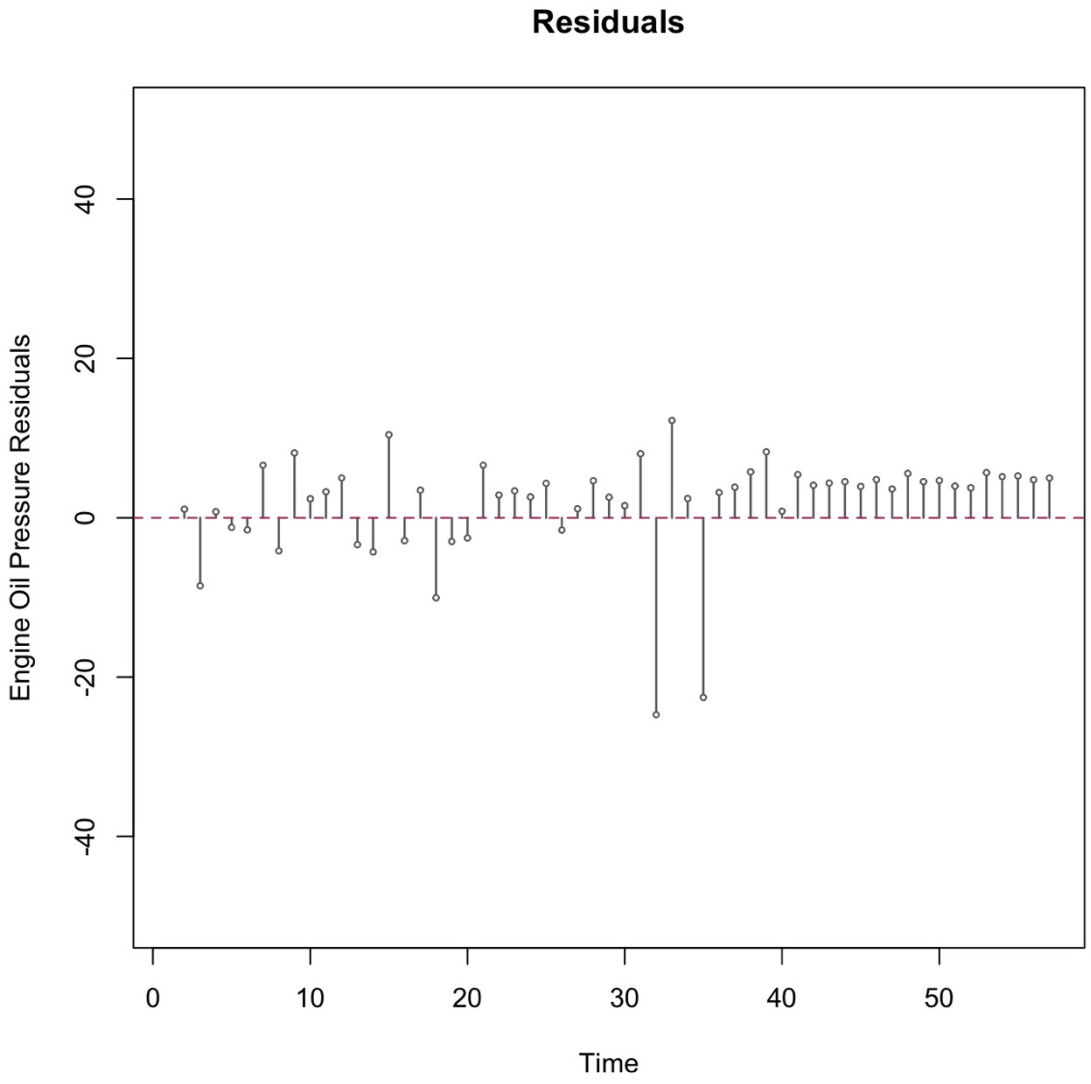


Figure 25. Residual Plot of Predicted Values

The max deviation from actual values is 25 and the mean deviation from actual values is less than 6 with a mean absolute deviation percentage of 0.5% across the range of possible values, 0-1000. Our prediction results indicate that variance is not an issue and our model does well at predicting engine oil pressure as an indicator of engine usage conditions.

3. Autocorrelation Function

We plot the autocorrelation function to observe any indication of autocorrelations in our residuals, shown in Figure 26. Only one time period is statistically significant, however, it is less than 0.3. Overall, our plot suggest acutely low measures of autocorrelation across our residuals.

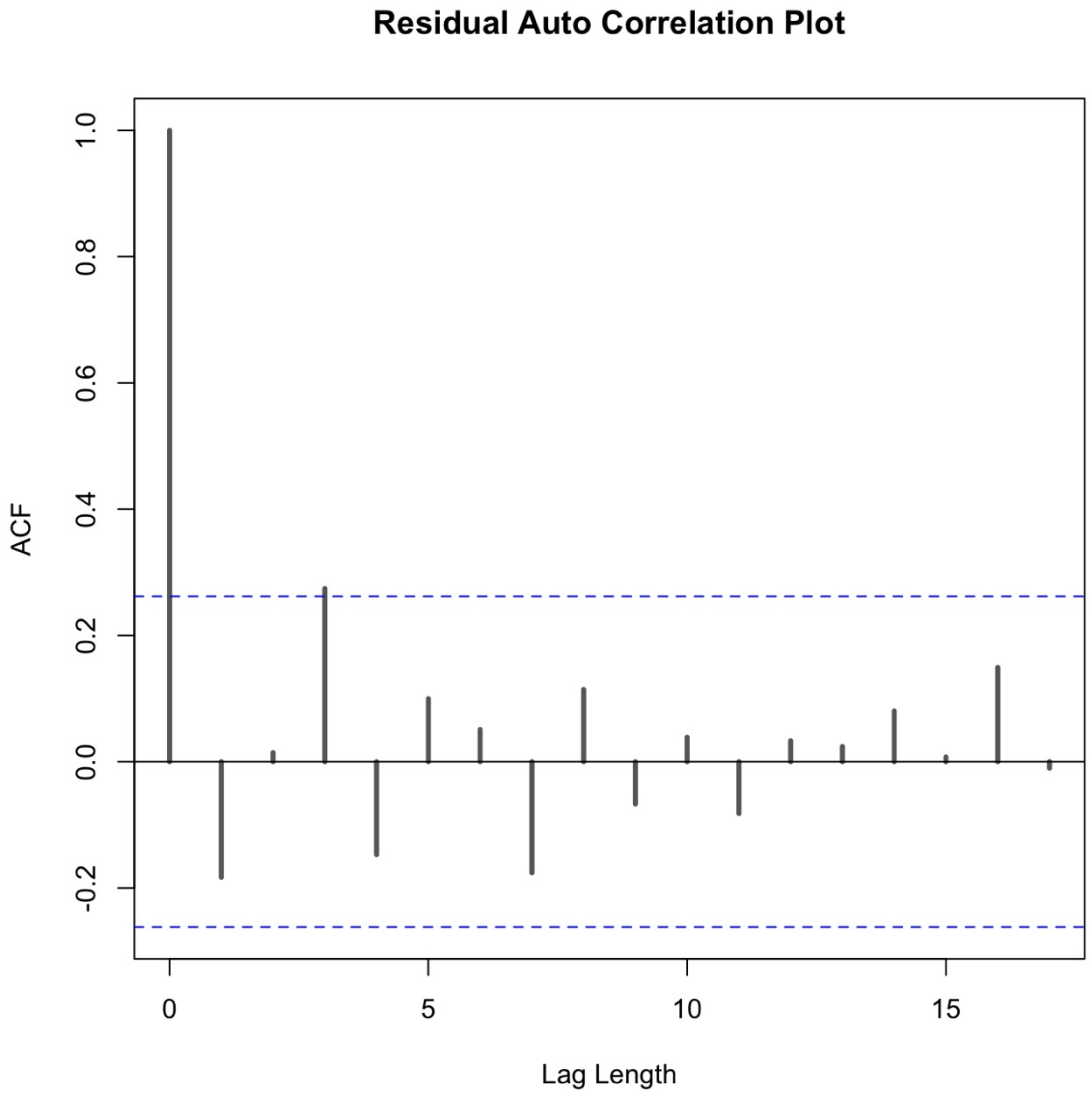


Figure 26. Residual Autocorrelation Function Plot

E. SUMMARY AND LIMITATIONS

Although our fitted model does well at predicting engine oil pressure, one limitation of our model is that only one predictor, engine speed, was used for prediction. Many parameters exists in the MTVR that combine to identify more features and characteristics that describe engine health and conditions.

The R-squared of our model suggest a goodness of fit. We validated and verified prediction results on actual data. Our plots compared predicted values versus actual values in dataset F and L. The autocorrelation plots show no autocorrelation in our residuals. The residual standard deviation was notably low with minimum prediction error. Model diagnostic plots revealed variance may not be equal across fitted values against residuals, however, QQ plot of normality suggest a common distribution with little to no skewness in tails. We assume that residuals are randomly distributed around our regression line. Our model assumptions are confirmed when viewing residual plots of validated predicted values.

Engine oil pressure responds to engine temperature and engine speed. Specifically, our model predicts the usage and health of an MTVR engine based on engine oil pressure. Engine speed was used as a predictor because it best demonstrates the use of an engine under certain driving conditions. When the engine is hot and working hard, engine speed responds. Usage parameters such as fluid temperature, fuel consumption, and oil pressure respond to engine speed The fitted model is an ARDL time series with one lag that characterizes engine oil pressure as a function of engine usage condition. From this model we characterize the “normal” operating behavior of an MTVR. We observe that when parameters increase or decrease they carry their values on to the next factor (autocorrelation). Our lag value of predicted engine oil pressure allows us to monitor the residuals. The variability of residuals describe the range of “normal” operating conditions. Based on these predicted values, conditions that push residuals outside of their “normal” variability range indicate and trigger maintenance actions.

V. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE STUDY

This thesis demonstrated the use of inexpensive COTS hardware devices and open-source software to develop an automated data collection architecture and a data processing framework to determine a preventative maintenance approach to support predictive maintenance of Marine Corps ground equipment. Data processing techniques were used to convert raw sensor data collected from on-board MTVR sensors into useable and measurable diagnostic data. Using a statistical analysis technique and a time series regression model, the diagnostic parameters that closely modeled engine operating conditions were chosen to predict engine usage characteristics of an MTVR engine. This predictive model was then used to describe a conditions-based maintenance approach.

Using the data collected, we are able to provide in-depth maintenance analysis, improve quality assurance, and generate enhanced maintenance solutions for vehicle life cycle support by designing a CBM policy designed around sensor data received from certain vehicle components. The data collected was categorized into twelve 30-minute datasets. The correlation coefficient was used to determine the relationship between parameters and engine usage conditions. We used an auto regressive distributed lag time series model with engine speed as the independent variable and engine oil pressure as the dependent variable. We monitored engine oil pressure to determine "normal" engine usage conditions. A change in engine speed that produced engine oil pressure values outside of its predicted range caused area for concern and form the basis of our CBM policy.

A. CONDITION MONITORING AS A CBM POLICY

For each component being monitored, rules are established to determine the "zone of opportunistic replacement" [57, p. 11]. For this thesis, our monitored engine components were engine fluid temperatures, engine speed, and engine oil pressure. We fit an auto regressive distributed lag time series model to predict at time, t, and engine speed, x_t, the future observed engine oil pressure, Y_t. The residuals of our model are used to determine the expected observed deterioration level of our monitored components. In our case, engine

oil pressure helps determine the current operating health of the engine or "remaining useful life." The decision to trigger maintenance is based on reading values of residuals outside of the "normal" operating range. For this thesis, sustained values greater than 30% of our maximum residual deviance exceed "normal" thresholds and trigger a maintenance decision. Below is our decision model for a condition-monitoring CBM maintenance policy from the data collected from the MTVR, adapted from Albert Tsang's[12] CBM decision making methodology.

The proposed method for known failures

1. Catalog known failures based on past failures and the data collected over time in a "known-failure" database or central server. Apply an "alert" condition code to each failure to track failure types based on symptoms, operating environment, mission profile, and driving conditions.

2. Associate known failures (by querying the database) with applicable maintenance tasks (preventive or corrective). One example of this is conducting an "opportunistic replacement" based on the amount of failure data collected to indicate a part is likely to fail.

The proposed method for new failures:

1. For new failures not cataloged in "known-failure" database or central server, begin fault-finding task.

2. Categorize failure based on symptoms, operating environment, mission profile, and driving conditions in "known-failure" database.

3. Update "alert" condition codes in "known-failure" database

4. Conduct corrective or preventative maintenance action or apply modification.

5. Update stakeholders, such as maintenance key players, manufacturer, and supply distributers and providers, in CBM network.

Figure 27 presents a CBM "trigger/alert" classification diagram to expand upon the CBM policy stated previously.

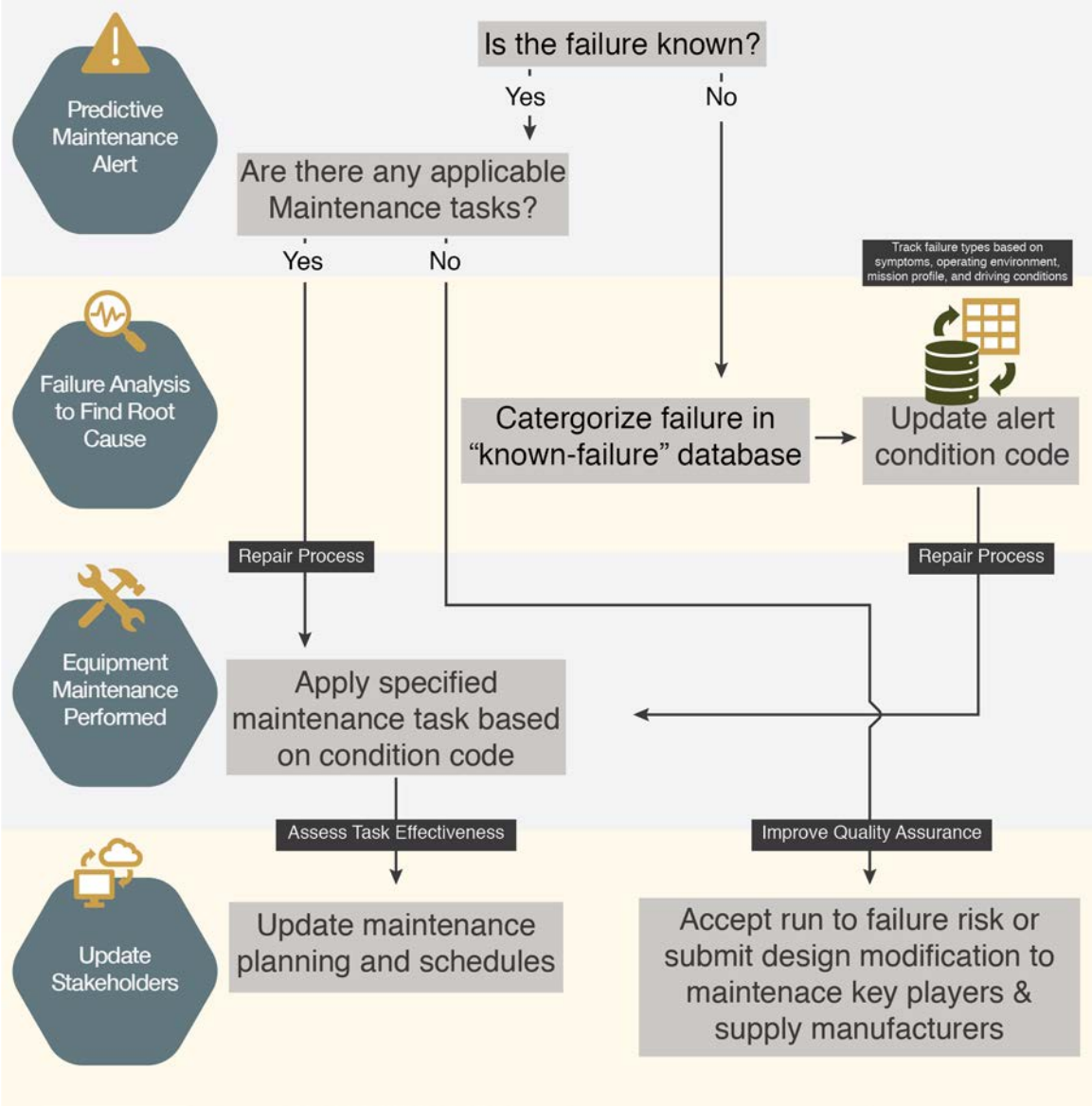


Figure 27. CBM Failure Alert Decision Policy. Adapted from [12], [29].

B. GENERALIZABILITY

While this thesis explored one-method to conduct CBM on a single piece of equipment, the method can be generalized and used to categorize the behavior for a fleet

of MTVRs and other Marine Corps ground equipment or weapon systems Using mission profiles for truck operating conditions software functions can be defined to compute and return the highest correlating features relating to engine conditions. This collection of data features can then be stored in a central server or cloud network where the data can be processed to output measured values. After the data is processed, predictive maintenance analytics using machine learning models can be used to reveal data insights, trends, or operational limits that trigger or alert user maintenance actions at the component and system level. This data can feed into an interconnected cross-organizational data store of maintenance data revealing relationships between pieces of information beyond single-equipment discovery. The value of CBM is realized at the intersection of data aggregation and interchange where a variety of possible CBM solutions can combine to solve complex maintenance issues.

C. DATA LIMITATIONS OF MTVR 23

The software and capability exist to extract on-board diagnostic data from ground equipment; however, this research showed that the J1939 data-bus technology installed in the MTVR 23 was underdeveloped. Out of 9,000 diagnostic messages available in the "J1939 Standards Collection," the MTVR only produced 82 data parameters from 23 unique parameter group numbers. Further, much of the data logged a 0xFF suspect parameter number, meaning "not applicable." This constraint limited our research focus to only the data available during real-time diagnostic extraction from the MTVR. Other parameters such as exhaust gas temperature, battery voltage, total engine running hours, and engine oil temperature, would have been useful to determine mechanical wear and tear across all vehicle components, but were unavailable from the MTVR 23. It would be beneficial to conduct further data extraction experimentation on other Marine Corps ground equipment, such as HMMWVs, engineering equipment and generators, M777 Howitzers, and M1A1 tanks capable of utilizing the J1939 data-bus technology.

D. RECOMMENDATIONS FOR FUTURE STUDY

This research did not address three critical areas that are integral to creating a CBM environment; namely, data security and authentication from ground equipment to server to

front-end application, integrating the CBM data management architecture into the current Marine Corps command and control infrastructure, and secondary and tertiary analytic insights from using additional analysis methods. A brief discussion on areas recommended for future study follows.

1. Security of Data Transmission

By default, J1939 data-bus is not encrypted. This diagnostic data applies no layers of security protocol. During this research, we were able to directly log raw CAN-bus data with only the 9-Pin J1939 connecter and the Raspberry Pi. This plug-and-play convenience presents notable security risk for diagnostic data being read from equipment. If this data were to be pushed to a cloud or central service network via WIFI or Bluetooth, this data can easily be altered, read, or copied during transmission. This leaves way for malware injection or manipulated data transmission to CBM maintenance servers. Further research is needed to examine encryption protocols and security practices to ensure secure transmission of diagnostic data and data protection for the data sitting in maintenance servers.

2. Integration into Marine Network

This research did not address the data strategy necessary to build the data environment to manage, store, and process the data collected from equipment using CBM. The data environment includes integrating existing network architectures to combine with CBM networked data; building deployment and support strategies for sensitive equipment and weapon systems capable of producing on-board diagnostics; establishing a network policy for cross-organization data sharing (internal and external) revealing equipment conditions, performance, and history; and implementing data management access protocols to ensure maintenance users and administrative personnel are qualified to operate, maintain, and view data collected in maintenance servers. Future research in the CBM data management and communications domains will help document the next steps needed in integrating CBM into the Marine Corps network environment.

3. Additional Model-Based Reasoning Methods

This research presents one statistical model that derives a "normal" engine operating profile from the MTVR, however, many other statistical and machine learning methods can be applied to derive further insights and trend analysis. This thesis used a model that best matched the capabilities of the data collected, however, over time, as larger datasets filled with dense diagnostics parameters are collected, the use of more complex statistical analysis methods such as decision trees, classification models, logistic regression, and machine learning modeling, can be used to solve problems, create new opportunities, and drive maintenance decisions. Future research in this area can take a deeper look at combining different statistical models to best fit the demands and constraints of the Marine Corps operating environment.

APPENDIX A. PYTHON SOURCE CODE FOR DECODING J1939 DATA

```
# --------------------------------
# DECODING J1939
# Thesis: Decoding J1939 to Enable Conditions-based Maintenance
# Whitaker, Michael Capt, USMC
# Adapted from [54]
# Python 3.6
#
# --------------------------------
import sqlite3  # import the sqlite 3 extensions for Python
import csv
import glob
import os
import datetime
file_list = glob.glob(os.path.join(os.getcwd() + '/LOG FILES/', "20 marc mtvr 592420", "*.TXT"))
datalines = []
for file_path in file_list:
    with open(file_path) as f_input:
        datalines.extend(f_input.readlines())
print(datalines[0:10])  # shows the first few entries in the list. These entries should match the log file

# remove header information, start from end of list as not to skip index from values shifting
for line in reversed(datalines):
    if line.startswith("#"):
        datalines.remove(line)
    if line.startswith('T'):
        datalines.remove(line)
# print(datalines[0:10])  # shows the first few entries in the list after cleaning
#
################################################################################################
#PARSING THE LOG FILE AND CREATING DATA STRUCTURES
#TO HOLD DATA FOR FUR HER PROCESSING
#
################################################################################################
DLCs = {}  # Data length Code Dictionary that will be used to find all IDs.
timeStamp_list = []  # store the timestamp from the log file
IDList = []  # store the IDs in sequence
rawData = []  # Store the data fields in sequence
for line in datalines:  # iterate through the entire file starting at line 0
    elements = line.split(';')  # Separate the entries by commas
    # convert Time to iso and parse
    t = elements[0] # 20T155058861
    dt_obj = datetime.datetime.strptime('201903' + t, '%Y%m%dT%H%M%S%f')
    t = int(dt_obj.strftime('%Y%m%d%H%M%S%f'))
    timeStamp_list.append(t)  # add timestamp to list
    # ## this took almost 3 minutes
```

```
    #  '20170828T160536247'
    ID = elements[2]  # extract the PGN associated with the CAN identifier
    # cf00400 or 18febf0b
    IDList.append(ID)  # add the ID to a list
    # DLCText = 8  # The DLC text field has a colon, so separate the label from the text
    # Dlc:8
    DLC = 8  # Get the data length code as an integer
    #  8
    # Store the DLC in a dictionary with the ID as a key. This will ensure only unique IDs are used as keys.
    DLCs[ID] = DLC
    # {'14fef031': 8}
    # this splits up the HEX value into SPNs
    templist = []  # hold the 32 bit hex value as I build it
    for i in range(len(elements[3][:-1])):
        if i % 2 == 0:
            templist.append(elements[3][:-1][i] + elements[3][:-1][i + 1])  # store the data field as a list of text strings
    rawData.append(templist)
    # ['ff', 'ff', 'ff', 'ff', 'ff', 'fc', '00', 'ff']
IDs = DLCs.keys()  # the keys in the DLCs dictionary are all the unique IDs that have not been duplicated.
# print('ID\tDLC')  # Print the heading of a table
# for ID in sorted(DLCs.keys()):  # iterate through all the sorted dictionary keys
#     print('%s\t%i' % (ID, DLCs[ID]))  # display the dictionary contents
#
# print('Number of Unique IDs: %i' % len(IDs))  # display the total number of messages
#
################################################################################################
##############
# C R E A T I N G    A   D A T A B A S E
"'Create a database to decode the raw data in accordance with SAE Technical Standards.  The data inputted
into
the database id copyrighted by SAE"'

#
################################################################################################
##############
# ### Digital Annex ###
pathtoDA        = '/Users/whitakermichael/NPS/_THESIS/DigitialAnnex/SAEDigitalSAEcsvs'
SPNsandPGNs     = os.path.join(pathtoDA, "SPNsandPGNs.csv")
Slots           = os.path.join(pathtoDA, "slots.csv")
src_address_hwy = os.path.join(pathtoDA, "SourceAddressesOnHighway.csv")
man             = os.path.join(pathtoDA, "Manfuacturers.csv")
glb_addr        = os.path.join(pathtoDA, "SourceAddresses.csv")
# #SQL Statements to create tables
nameofdatabse = "SAEJ1939_db"
filename = '/Users/whitakermichael/NPS/_THESIS/Decoding J1939/' + nameofdatabse
if os.path.isfile(filename):
    os.remove(filename)  # delete the database if it already exists. This prevents errors for duplicate tables
conn = sqlite3.connect(filename)
conn.text_factory = str  # This command enables the strings to be decoded by the sqlite commands
cursor = conn.cursor()
cursor.execute('CREATE TABLE SourceAddressesOnHighWay (ID INT, Name STRING, Notes STRING,
DateModified STRING)')
```

```
cursor.execute('CREATE TABLE Manufacturers (ID INT,MANUFACTURER,LOCATION,DateLastModified)')
cursor.execute('CREATE TABLE SLOTS (SLOTIdentifier INT,SLOTName STRING,SLOTType STRING,Scaling STRING,'
          'Range STRING,Offset STRING, Length STRING ,DateModified STRING)')
cursor.execute('CREATE TABLE SPNandPGN (PGN INT, ParameterGroupLabel, PGNLength, TransmissionRate, Acronym, pos,'
          'SPNlength INT, SPN INT, Name, Description, DataRange, OperationalRange, Resolution, Offset, Units,'
          'DateSPNAddedToPGN, StatusOfSPNAdditionToPGN, DateSPNModified, SPNDoc, PGNDoc, BitField)')
cursor.execute('CREATE TABLE SourceAddresses (ID INT, Name STRING, Notes STRING, DateModified STRING)')
with open(src_address_hwy, 'r') as f:
   reader = csv.reader(f)
   for row in reader:
      cursor.execute('INSERT INTO SourceAddressesOnHighWay VALUES (?,?,?,?)', row)
with open(man, 'r') as f:
   reader = csv.reader(f)
   for row in reader:
      cursor.execute('INSERT INTO Manufacturers VALUES (?,?,?,?)', row)
with open(SPNsandPGNs, 'r') as f:
   reader = csv.reader(f)
   for row in reader:
      cursor.execute('INSERT INTO SPNandPGN VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)', row)
with open(Slots, 'r') as f:
   reader = csv.reader(f)
   for row in reader:
      cursor.execute('INSERT INTO SLOTS VALUES (?,?,?,?,?,?,?,?)', row)
with open(glb_addr, 'r') as f:
   reader = csv.reader(f)
   for row in reader:
      cursor.execute('INSERT INTO SourceAddresses VALUES (?,?,?,?)', row)
conn.commit()  # save changes
# Test to make sure tables are uploaded in database
cursor.execute('SELECT * from SourceAddresses').fetchone()
# goto terminal log into database and fix unicode error
# ('\ufeff0', <- THIS IS THE UNICODE ERROR
#  'Engine #1',
#  'The #1 on the Engine CA is to identify that this is the first PA being used for the particular function, Engine.
#  It may only be used for the NAME Function of 0, Function Instance 0, and an ecu instance of 0, which is commonly
#  know as the “first engine”.',
#  '')
# ########################################
# RUN THIS IF DATABASE IS ALREADY CREATED
# ########################################
nameofdatabse = "SAEJ1939_db"
filename = '/Users/whitakermichael/NPS/_THESIS/Decoding J1939/' + nameofdatabse
conn = sqlite3.connect(filename)
conn.text_factory = str  # This command enables the strings to be decoded by the sqlite commands
cursor = conn.cursor()
# TEST
```

```
cursor.execute('SELECT * from SourceAddresses').fetchone()
#
################################################################################################
##############
# D E C O D I N G   J 1 9 3 9
'''Use database and SAE technical specificaitions to parse 64 bit message and 18 bit PGN header.
This allows us to pull out functions from ECU communication in vehicles.'''
#
################################################################################################
##############
SA_Name = {}  # Source Address
SA_Note = {}  # Source notes
SPNs = {}
decodingDictionary = {}
for IDText in sorted(IDs):
    SAText = IDText[-2:]
    try:
        SA = int(SAText, 16)  # Convert the hex string into an integer
    except ValueError:
        print("SAText:", SAText)
        print("IDtext:", IDText)
        continue
    PFText = IDText[-6:-4]
    PF = int(PFText, 16)
    if PF < 240:  # PDU1 format
        DA = int(IDText[-4:-2], 16)
        PSText = '00'
        PS = DA
    else:
        PSText = IDText[-4:-2]
        PS = int(PSText, 16)
        DA = 255  # Broadcast
    PriorityText = IDText[-8:-6]
    Priority = int(PriorityText, 16) >> 2  # bit shift the priority integer by 2 to account for hte Dp and EDP fields.
    print('\nMessage ID: %s' % IDText)
    decodingDictionary.setdefault(IDText, [])
    print('Priority (Hex): %s' % PriorityText)
    print('J1939 Priority: %i' % Priority)
    decodingDictionary[IDText].append(Priority)
    print('PDU Format (PF) in hex: %s and in decimal: %i' % (PFText, PF))
    decodingDictionary[IDText].append(PF)
    print('PDU Specific (PS) in hex: %s and in decimal: %i' % (PSText, PS))
    decodingDictionary[IDText].append(PS)
    print('Source Address in hex: %s and in decimal: %i' % (SAText, SA))
    decodingDictionary[IDText].append(SA)
    PGNText = PFText + PSText
    PGN = int(PGNText, 16)
    print('Parameter Group Number (PGN): %i' % PGN)
    decodingDictionary[IDText].append(PGN)
    PGNData = cursor.execute('SELECT ParameterGroupLabel,TransmissionRate,Acronym FROM
SPNandPGN WHERE PGN=?',
                             [PGN]).fetchone()
```

```
    try:
        print('Parameter Group Label: %s' % PGNData[0])
        decodingDictionary[IDText].append(PGNData[0])
    except TypeError:
        break
    print('Transmission Rate: %s' % PGNData[1])
    decodingDictionary[IDText].append(PGNData[1])
    acronym = str(PGNData[2])
    print('Acronym: %s' % acronym)
    decodingDictionary[IDText].append(acronym)
    # Source Addresses
    if SA < 94 or SA > 247:
        sourceAddressData = cursor.execute('SELECT Name,Notes FROM SourceAddresses WHERE ID=?',
[SA]).fetchone()
        SA_Name[SA] = sourceAddressData[0]
        SA_Note[SA] = sourceAddressData[1]
    elif SA > 159 and SA < 248:
        sourceAddressData = cursor.execute('SELECT Name,Notes FROM SourceAddressesOnHighWay
WHERE ID=?', [SA]).fetchone()
        SA_Name[SA] = sourceAddressData[0]
        SA_Note[SA] = sourceAddressData[1]
    else:
        SA_Name[SA] = 'SAE Future Use'
        SA_Note[SA] = 'Used SAE future Use or for dynamic address assignment'
    print('Source Controller: %s' % SA_Name[SA])
    decodingDictionary[IDText].append(SA_Name[SA])
    print('The Following SPNs are available in the message:')
    TempDict = {}
    for SPNData in cursor.execute('SELECT SPN,Name,Units,Offset,Resolution,pos, SPNlength, DataRange
FROM SPNandPGN WHERE PGN=?', [PGN]):
        try:
            SPN = int(SPNData[0])
        except ValueError:
            SPN = -1
        Name  = str(SPNData[1])
        Units = str(SPNData[2])
        # PARSE OFFSET VALUE
        offset_val = SPNData[3].split()  # ['-40', '°C']
        if len(offset_val) > 0:
            try:
                Offset = float(offset_val[0].replace(',', ''))
            except ValueError:
                Offset = 0
        else:
            Offset = 0

        # PARSE RESOLUTION
        res = SPNData[4].split(' ')
        res = res[0].split('/bit')
        res = res[0].split('/')
        try:
            if len(res) == 2:
```

```
            Resolution = float(res[0]) / float(res[1])
        else:
            Resolution = float(res[0])
    except ValueError:
        Resolution = 0
    #  PARSE BIT START POSITION FILL WITH -GRAND VAL
    try:
        pos = int(SPNData[5][0])
    except IndexError:
        pos = int(-88888)
    length_measure = SPNData[6]
    try:
        if length_measure.split()[1].startswith('bit'):
            continue
    except IndexError:
        continue
    #  PARSE DATA RANGE
    temp_datarange_list = []
    data_range_str = SPNData[7]
    for val in data_range_str.split():
        try:
            min_or_max_val = float(val.replace(',', ''))
        except ValueError:
            min_or_max_val = ()
            continue
        temp_datarange_list.append(min_or_max_val)
    date_Range = tuple(temp_datarange_list)
    if len(date_Range) < 1:
        # date_Range = (-8888, -88888)
        continue
    SPNs[SPN] = [Name, Units, Offset, Resolution, (IDText, PGN), acronym, pos, length_measure,
date_Range]
    TempDict[SPN] = [Name, Units, ('Offset', Offset), Resolution, (IDText, PGN), acronym, ('pos', pos,
length_measure), date_Range]
    # print(SPNData[4])
    print('SPN: %i, Name: %s, Unit: %s, Offset: %g, Resolution: %g, BitStartPosition: %i, length: %s '
          'DataRange: min= %d, max= %d' % (SPN, Name, Units, Offset, Resolution, pos, length_measure,
date_Range[0], date_Range[1]))
  decodingDictionary[IDText].append(TempDict)
# close connection to Database
# conn.close()

#
################################################################################################
##############
#         C R E A T E   T A B L E   T O   H O L D   O U T P U T   O F
#             D E C O D E D   J 1 9 3 9   M E S S A G E S
#
################################################################################################
##############
conn = sqlite3.connect(filename)
conn.text_factory = str  # This command enables the strings to be decoded by the sqlite commands
```

```
cursor = conn.cursor()
cursor.execute('DROP TABLE decodedValues')  # drop table if it exist
cursor.execute('CREATE TABLE decodedValues (time FLOAT,ID STRING, PF INT, PS INT, DA INT, PGN
INT, SA INT, Priority INT,'
          'Acronym STRING, Meaning STRING, Unitss STRING) ')
# ADD SPN COLUMN TO TABLE FOR EVERY UNIQUE SPN VALUE
for SPN in sorted(SPNs.keys()):
   print('{}\t{}\t{}' .format(SPNs[SPN][4], SPN, SPNs[SPN][0]))
   cursor.execute('ALTER TABLE decodedValues ADD COLUMN SPN%i' % SPN)  # This adds only the SPNs
that are in the data stream.
#
##############################################################################################
###############
#          T H I S   A D D S   D E C O D E D   V A L U E S   T O   D A T A B A S E
#
##############################################################################################
###############
spn_list = [k for k, v in SPNs.items()] # added to test specific spns
for timestamp, PGNID, payload in zip(timeStamp_list, IDList, rawData):
   # "20190320173110134000", "18f00010", ['c0', '7d', 'ff', 'ff', 'ff', 'ff', 'ff', 'ff']
   for key, value in decodingDictionary.items():
      if PGNID[-6:] == key[-6:]:
         for spn, spnDataValue in value[9].items():  # this is the spn dictionary
            # if spn in spn_list: #this was added to test specific SPNs
            position = spnDataValue[6][1]
            SP_length = spnDataValue[6][2].split()[0]
            SPN = spn
            Unit = spnDataValue[1]
            Resolution = spnDataValue[3]
            Offset = spnDataValue[2][1]
            Max_dataRange = spnDataValue[7][1]
            if position < 8:
               if int(SP_length) > 1:
                  # intel byte order (the right byte switched with left byte)
                  # this is why 0-based index is working out
                  binary = int(payload[position] + payload[position - 1], 16)
                  Value = Resolution * binary + Offset
                  if Value > Max_dataRange:
                     if Unit == 'count':
                        Value = 0
                     else:
                        Value = 9999  # THIS MEANS NO DATA (NA)
                  # print(f'{timestamp} -  {PGNID} - {SPN} - {Value}')
                  cursor.execute("'INSERT INTO decodedValues(time, ID, PF, PS, PGN, SA, Priority, Unitss,
SPN%i)
                                 VALUES(%i, \"%s\", %i, %i, %i, %i, %i, \"%s\", %f)"'
                                 % (spn, timestamp, PGNID, decodingDictionary[key][1],
                                    decodingDictionary[key][2], decodingDictionary[key][4],
                                    decodingDictionary[key][3], decodingDictionary[key][0], Unit, Value))
               else:
                  # do this
                  binary = int(payload[position-1], 16)
```

```
                Value = Resolution * binary + Offset
                if Value > Max_dataRange:
                    if Unit == 'count':
                        Value = 0
                    else:
                        Value = 9999  # THIS MEANS NO DATA (NA) OR FALSE VALUES
                # print(f'{timestamp} -  {PGNID} - {SPN} - {Value}')
                cursor.execute("'INSERT INTO decodedValues(time, ID, PF, PS, PGN, SA, Priority, Unitss,
SPN%i)
                               VALUES(%i, \"%s\", %i, %i, %i, %i, %i, \"%s\", %f)"'
                               % (spn, timestamp, PGNID, decodingDictionary[key][1],
                                  decodingDictionary[key][2], decodingDictionary[key][4],
                                  decodingDictionary[key][3], decodingDictionary[key][0], Unit, Value))
            else:
                length, signal_range = spnDataValue[6][2].split()
                if signal_range.startswith('bit'):
                    continue  # there should be no SPNs with bit specified for length
                else:
                    binary = int(payload[position - int(length)], 16)  # makes it index 7 vs 8
                    Value = Resolution * binary + Offset
                    if Value > Max_dataRange:
                        if Unit == 'count':
                            Value = 0
                        else:
                            Value = 9999  # THIS MEANS NO DATA (NA)
                    cursor.execute("'INSERT INTO decodedValues(time, ID, PF, PS, PGN, SA, Priority, Unitss,
SPN%i)
                                   VALUES(%i, \"%s\", %i, %i, %i, %i, %i, \"%s\", %f)"'
                                   % (spn, timestamp, PGNID, decodingDictionary[key][1],
                                      decodingDictionary[key][2], decodingDictionary[key][4],
                                      decodingDictionary[key][3], decodingDictionary[key][0], Unit, Value))
            # print(f'{timestamp} - {PGNID} - {SPN} - {Value}')
conn.commit()
conn.close()
```

APPENDIX B. R SOURCE CODE FOR DATA ANALYSIS AND PLOT DRAWING OF MTVR DIAGNOSTIC DATA

```
# ----------------------------------
# DATA ANALYIS OF MTVR DIAGNOSTIC DATA
# Thesis: Decoding J1939 to Enable Conditions-based Maintenance
# Whitaker, Michael Capt, USMC
#
# R version 3.6.0
#
# ----------------------------------
# load Data Set that that has NA removed and interpolated
DiagnosticData <- readRDS("~/NPS/_THESIS/Decoding J1939/Rworkindir/J1939DecodedValues__NoNA.rds")
str(DiagnosticData)
# 'data.frame': 1237373 obs. of  9 variables:
#  $ time  : num  20190320155058860032 20190320155058876416 20190320155058888704 20190320155058905088 20190320155058909184 ...
# $ SPN174: num  29 29 29 29 29 29 29 29 29 29 ...
# $ SPN190: num  714 716 714 716 715 ...
# $ SPN110: num  74 74 74 74 74 74 74 74 74 74 ...
# $ SPN84 : num  0 0 0 0 0 0 0 0 0 0 ...
# $ SPN105: num  27 27 27 27 27 27 27 27 27 27 ...
# $ SPN177: num  51 51 51 51 51 51 51 51 51 51 ...
# $ SPN183: num  5.05 5.05 5.05 5.05 5.05 5.05 5.05 5.05 5.05 5.05 ...
# $ SPN100: num  176 176 176 176 176 176 176 176 176 176 ...

######## D A T A S E T   C O L U M N S #############:
### TIME
### SPN174 (Engine Fuel Temp),
### SPN190 (RPMs),
### SPN110 (Engine Coolant Temperature ),
### SPN84  (Wheel based Vehicle Speed),
### SPN105 (Engine Intake Temp),
### SPN177 (Transmission Oil Temp),
### SPN183 (Engine Fuel Rate),
### SPN100 (Engine Oil Pressure),

options( scipen = 9999)
# Separate dianosticData$time to down sample data by 30 second intervals
# TimeStamp: 20190320155058860032
# CONVERT TIME STAMP TO BELOW FOR EASY AGGREGATION
# YEAR-MM-DD HH:MM:SS:sss:sss
# 2019-03-20 15:50:58:860:032
df<-DiagnosticData  # make copy to df
```

```
str(df)
df$time
tt<- as.character(df$time) # convert time to character
tt<-transform(tt, date=substr(tt, 1, 8), Time = substr(tt, 9, 14), Seconds = substr(tt, 15, 20))
str(tt)

df$date <- tt$date
df$Time <- tt$Time
df$Seconds<- tt$Seconds
head(df, n=20)
tt$date

dfnew1 <- df[,c("time", "date", "Time", "Seconds","SPN174", "SPN190", "SPN110", "SPN84", "SPN105",
"SPN177", "SPN183", "SPN100")]
head(dfnew1, n=20)
dfnew1$date <- as.Date(dfnew1$date , "%Y%m%d")
head(dfnew1,20)
dfnew1$Time <- format(strptime(dfnew1$Time, "%H%M%S", tz="America/Los_angeles"), "%H:%M:%S")
dfnew1$timeStamp <- as.POSIXct(paste(dfnew1$date, dfnew1$Time), format="%Y-%m-%d %H:%M:%S")
head(dfnew1, 30)
# Building final data frame
dfnew2 <- dfnew1[,c("timeStamp", "date", "Time", "Seconds","SPN174", "SPN190", "SPN110", "SPN84",
"SPN105", "SPN177", "SPN183", "SPN100")]
head(dfnew2)
# make a data.table frame for data to do time series
diagnosticDT <- dfnew2
# aggreagate mean of SPNS by 30sec intervals
require(dplyr)
require(lubridate)
diagnosticDT <- diagnosticDT %>%
   group_by(timeStamp = cut(timeStamp, breaks="30 secs")) %>%
   summarize(Engine_Intake_Manifold_Temp = mean(SPN105),
          Engine_Fuel_Temp=mean(SPN174),
          RPMs = mean(SPN190),
          Engine_Coolant_Temp = mean(SPN110),
          Wheel_based_vehicle_Speed = mean(SPN84),
          Transmission_Oil_Temp = mean(SPN177),
          Engine_Fuel_Rate = mean(SPN183),
          Engine_Oil_Pressure = mean(SPN100),
          seconds = Seconds[Engine_Fuel_Temp])
table(complete.cases(diagnosticDT))  # only one FALSE
which(!complete.cases(diagnosticDT)) # last one

# There is one NA value need to remove that gets dropped
diagnosticDT<- diagnosticDT[complete.cases(diagnosticDT), ]
```

```
head(diagnosticDT, 20)
# This makes diagnosticDT$timeStamp a FACTOR
# Convert this back to date object
diagnosticDT$timeStamp<-as.POSIXct(diagnosticDT$timeStamp, format="%Y-%m-%d %H:%M:%S")
# split data into 30 minutes groups and make a new data coloumn
diagnosticDT$by30min <- cut(diagnosticDT$timeStamp, breaks="30 min")
# move the 30min grouping to the front of data frame
x <- "by30min"
diagnosticDT<- diagnosticDT[c(x, setdiff(names(diagnosticDT), x))]
head(diagnosticDT)
# Move Seconds column to the beginning of dataframe
diagnosticDT<- subset(diagnosticDT, select=c(1:2,11,3:10))
head(diagnosticDT)
str(diagnosticDT)
# Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 551 obs. of  11 variables:
# $ by30min                  : Factor w/ 12 levels "2019-03-20 15:50:00",..: 1 1 1 1 1 1 1 1 1 1 ...
# $ timeStamp                : POSIXct, format: "2019-03-20 15:50:58" "2019-03-20 15:51:28" "2019-03-20
15:51:58" "2019-03-20 15:52:28" ...
# $ seconds                  : Factor w/ 15625 levels "000000","000064",..: 2486 4297 4475 4421 4599 4545 4723
4669 4911 4793 ...
# $ Engine_Intake_Manifold_Temp: num  23.4 20.8 20 20 20 ...
# $ Engine_Fuel_Temp         : num  24.6 22.4 22 22 22 ...
# $ RPMs                     : num  702 700 745 908 805 ...
# $ Engine_Coolant_Temp      : num  73.2 73.8 74.3 75.9 77.6 ...
# $ Wheel_based_vehicle_Speed  : num  0 0 1.84 6.44 6.15 ...
# $ Transmission_Oil_Temp    : num  51.9 52 52.8 53.9 55 ...
# $ Engine_Fuel_Rate         : num  3.81 3.74 5.69 9.87 6.44 ...
# $ Engine_Oil_Pressure      : num  171 168 180 219 192 ...

### SPN174 (Engine Fuel Temp),
### SPN190 (RPMs),
### SPN110 (Engine Coolant Temperature ),
### SPN84  (Wheel based Vehicle Speed),
### SPN105 (Engine Intake Temp), <-- RESPONSE
### SPN177 (Transmission Oil Temp),
### SPN183 (Engine Fuel Rate),
### SPN100 (Engine Oil Pressure),
diagnosticDT
# #
#    by30min timeStamp           seconds Engine_Intake_M… Engine_Fuel_Temp  RPMs Engine_Coolant_…
Wheel_based_veh… Transmission_Oi…
#    <fct>   <dttm>              <fct>           <dbl>          <dbl> <dbl>          <dbl>          <dbl>          <dbl>
# 1 2019-0… 2019-03-20 15:50:58 159040           23.4           24.6  702.           73.2           0              51.9
# 2 2019-0… 2019-03-20 15:51:28 274944           20.8           22.4  700.           73.8           0              52
# 3 2019-0… 2019-03-20 15:51:58 286336           20             22    745.           74.3           1.84           52.8
# 4 2019-0… 2019-03-20 15:52:28 282880           20             22    908.           75.9           6.44           53.9
```

```
# 5 2019-0… 2019-03-20 15:52:58 294272          20          22    805.          77.6          6.15          55
# 6 2019-0… 2019-03-20 15:53:28 290816          20          22.8  804.          78.7          5.96          55
# 7 2019-0… 2019-03-20 15:53:58 302208          20.1        23    793.          79.7          5.92          55.9
# 8 2019-0… 2019-03-20 15:54:28 298752          20.7        23    703.          80.8          0.532         56.2
# 9 2019-0… 2019-03-20 15:54:58 314240          21          23.5  700.          81            0             57.4
# 10 2019-0… 2019-03-20 15:55:28 306688         21          24    700.          81            0             58
# … with 541 more rows, and 2 more variables: Engine_Fuel_Rate <dbl>, Engine_Oil_Pressure <dbl>

##### all my Parameters are numeric, which is a good thing
# data split into 30 minute levels and saved into separate dataframes
# 12 total based on timestamps
DiagnosticData <- readRDS("~/NPS/_THESIS/Decoding J1939/Rworkindir/diagnosticDT.rds") # <- put the downloaded RDS file here
timegaps<-levels(diagnosticDT$by30min)
diagnosticInterval <- split(diagnosticDT, diagnosticDT$by30min)
diagnosticInterval

df_A <- diagnosticInterval[[1]]
df_B <- diagnosticInterval[[2]]
df_C <- diagnosticInterval[[3]]
df_D <- diagnosticInterval[[4]]
df_E <- diagnosticInterval[[5]]
df_F <- diagnosticInterval[[6]]
df_G <- diagnosticInterval[[7]]
df_H <- diagnosticInterval[[8]]
df_I <- diagnosticInterval[[9]]
df_J <- diagnosticInterval[[10]]
df_K <- diagnosticInterval[[11]]
df_L <- diagnosticInterval[[12]]

### C O R R E L A T I O N ####

# Correlation between engine_temp and rps
colname<-colnames(diagnosticDT)
( colname<-as.vector(colname[4:11]) )
diagnosticCor<-subset(diagnosticDT[,c(colname)])
diagnostic.cor <- round(cor(diagnosticCor), 4)
diagnostic.cor
#                              Engine_Intake_Manifold_Temp Engine_Fuel_Temp   RPMs Engine_Oil_Pressure
Wheel_based_vehicle_Speed Transmission_Oil_Temp Engine_Fuel_Rate
# Engine_Intake_Manifold_Temp                1.00            0.35 -0.72              0.03                   -0.56
0.21                   -0.23
# Engine_Fuel_Temp                           0.35            1.00 -0.21              0.31                    0.00
0.74                   -0.06
# RPMs                                      -0.72           -0.21  1.00              0.23                    0.84
0.05                    0.57
```

```
# Engine_Oil_Pressure                    0.03          0.31  0.23          1.00              0.33              0.61
0.25
# Wheel_based_vehicle_Speed                -0.56          0.00  0.84              0.33              1.00
0.22          0.57
# Transmission_Oil_Temp                    0.21          0.74  0.05              0.61              0.22
1.00          0.03
# Engine_Fuel_Rate                      -0.23         -0.06  0.57          0.25              0.57              0.03
1.00

### G E T   C O R R E L A T I O N   F O R THE ENTIRE DATA FRAME NO WITHOUT 30MIN INTERVALS
##############
Install.packages("corrplot")
library("corrplot")
colname<-colnames(diagnosticDT)
( colname<-as.vector(colname[4:11]) )
length(colname)  # should be 8
par(mfrow=c(1,1))
require(corrplot)
col2 <- colorRampPalette(c("#67001F", "#B2182B", "#D6604D", "#F4A582",
                           "#FDDBC7", "#FFFFFF", "#D1E5F0", "#92C5DE",
                           "#4393C3", "#2166AC", "#053061"))

corrplot(diagnostic.cor,
         type="upper",
         col = col2(50),
         bg = "gray87",
         cl.ratio = 0.2,
         cl.align = "r",
         tl.col = "gray8",
         cex.lab=1.5,
         main="Dataset A Correlation Plot "
)
### G E T   C O R R E L A T I O N   F O R   A L L   T H E   D A T A S E T S O N E   #############
############################ O N E  A T  A  T I M E ( A - L ) ###################################
### 1
diagnosticCor.A<-subset(df_A[,c(colname)])
diagnostic.cor.A <- round(cor(diagnosticCor.A), 4)
# only one at a TIME
corrplot(diagnostic.cor.A,
         type="upper",
         col = col2(50),
         bg = "gray87",
         cl.ratio = 0.2,
         cl.align = "r",
         tl.col = "gray8",
         cex.lab=1.5,
```

```
        main="Dataset A Correlation Plot "
        )
### 2
diagnosticCor.B<-subset(df_B[,c(colname)])
diagnostic.cor.B <- round(cor(diagnosticCor.B), 4)
corrplot(diagnostic.cor.B,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset B Correlation Plot"
)
### 3
diagnosticCor.C<-subset(df_C[,c(colname)])
diagnostic.cor.C <- round(cor(diagnosticCor.C), 4)
corrplot(diagnostic.cor.C,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset C Correlation Plot"
)
### 4
diagnosticCor.D<-subset(df_D[,c(colname)])
diagnostic.cor.D <- round(cor(diagnosticCor.D), 4)
corrplot(diagnostic.cor.D,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset D Correlation Plot"
)
### 5
diagnosticCor.E<-subset(df_E[,c(colname)])
diagnostic.cor.E <- round(cor(diagnosticCor.E), 4)
corrplot(diagnostic.cor.E,
        type="upper",
```

```
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset E Correlation Plot"
)
### 6
diagnosticCor.F<-subset(df_F[,c(colname)])
diagnostic.cor.F <- round(cor(diagnosticCor.F), 4)
corrplot(diagnostic.cor.F,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset F Correlation Plot"
)
### 7
diagnosticCor.I<-subset(df_I[,c(colname)])
diagnostic.cor.I <- round(cor(diagnosticCor.I), 4)
corrplot(diagnostic.cor.I,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset I Correlation Plot"
)
### 8
diagnosticCor.J<-subset(df_J[,c(colname)])
diagnostic.cor.J <- round(cor(diagnosticCor.J), 4)
corrplot(diagnostic.cor.J,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset J Correlation Plot"
```

```
)
### 9
diagnosticCor.K<-subset(df_K[,c(colname)])
diagnostic.cor.K <- round(cor(diagnosticCor.K), 4)
corrplot(diagnostic.cor.K,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset K Correlation Plot"
)
### 10
diagnosticCor.L<-subset(df_L[,c(colname)])
diagnostic.cor.L <- round(cor(diagnosticCor.L), 4)
corrplot(diagnostic.cor.L,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset L Correlation Plot"
        )

# DATASETS A , D, F, L, I LOOK THE BEST
corrplot(data,
        is.corr=FALSE,
        type="upper",
        col = col2(50),
        bg = "gray87",
        cl.ratio = 0.2,
        cl.align = "r",
        tl.col = "gray8",
        cex.lab=1.5,
        main="Dataset Correlation Plot")

#### E N D ######
# load  Dataset correlation matrix that I made in excel as a csv
data    <-    read.csv("/Users/whitakermichael/NPS/_THESIS/Decoding    J1939/Rworkindir/coormatrix.csv",
stringsAsFactors=FALSE, row.names = 1)
head(data)
str(data)
```

```
data.m <- as.matrix(data)
corrplot(data.m,
         is.corr=FALSE,
         col = col2(50),
         bg = "gray87",
         cl.ratio = 0.2,
         cl.align = "r",
         tl.col = "gray8",
         tl.cex=0.5,
         tl.srt=90,
         main="Dataset Correlation Plot")
############## P L O T S ###################################
#plot all predictions across time for the Five Data Frames
####
###### plot  DF_A
######
install.packages("RColorBrewer")
library(RColorBrewer)
library(ggplot2)
library(reshape2)
library(col)
# Standardized data columns
Scaled.df_A <- as.data.frame(scale(df_A[,4:11]))  # <- all diagnostic parmeters
Scaled.df_A$RPMs
#check to make sure its 1 sd
sd(Scaled.df_A$RPMs)
sd(Scaled.df_A$Engine_Intake_Manifold_Temp)
## add scaled data to dataframe for plotting
df_A.with.scaled <- cbind(df_A[,1:3], Scaled.df_A)
head(df_A.with.scaled, 20)
ggplot(data=df_A.with.scaled, aes(x=timeStamp, y=values, color=Parameters)) +
  geom_line(aes(y=Engine_Intake_Manifold_Temp, col="Engine_Intake_Manifold_Temp"),size=.7)+
  geom_line(aes(y=Engine_Fuel_Temp, col="Engine_Fuel_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Coolant_Temp, col="Engine_Coolant_Temp"), size=0.7)+
  geom_line(aes(y=RPMs, col="RPMs"), size=0.7)+
  geom_line(aes(y=Wheel_based_vehicle_Speed, col="Wheel_based_vehicle_Speed"), size=0.7)+
  geom_line(aes(y=Transmission_Oil_Temp, col="Transmission_Oil_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Rate, col="Engine_Fuel_Rate"), size=0.7)+
  geom_line(aes(y=Engine_Oil_Pressure, col="Engine_Oil_Pressure"), size=0.7)+
  scale_colour_brewer(palette = "Dark2")+
  labs(title="Standard Plot of Diagnostic Values",
       subtitle="From Dataset A",
       y="Standardized Diagnostic Values",
       x="Time")+
  theme_bw(base_size = 16,
           base_line_size = .5)
```

```
# pairs plot
pairs(timeStamp ~ Engine_Intake_Manifold_Temp + Engine_Fuel_Temp + RPMs + Engine_Coolant_Temp +
       Wheel_based_vehicle_Speed + Transmission_Oil_Temp + Engine_Fuel_Rate + Engine_Oil_Pressure,
      lower.panel=NULL,
      panel=panel.smooth,
      pch=20, cex=1,
      lwd=0.3,
      col="dimgray",
      main=paste("Pairs Plot of Diagnostics", "\nDataset A"),
      cex.axis = 1.3,
      data=df_A)
####
###### plot  DF_D
######
# install.packages("RColorBrewer")
# library(RColorBrewer)
# library(ggplot2)
# library(reshape2)
# library(col)
# Standardized data columns
Scaled.df_D <- as.data.frame(scale(df_D[,4:11]))  # <- all diagnostic parmeters
Scaled.df_D$RPMs
#check to make sure its 1 sd
sd(Scaled.df_D$RPMs)
sd(Scaled.df_D$Engine_Intake_Manifold_Temp)
## add scaled data to dataframe for plotting
df_D.with.scaled <- cbind(df_D[,1:3], Scaled.df_D)
head(df_D.with.scaled, 20)
ggplot(data=df_D.with.scaled, aes(x=timeStamp, y=values, color=Parameters)) +
  geom_line(aes(y=Engine_Intake_Manifold_Temp, col="Engine_Intake_Manifold_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Temp, col="Engine_Fuel_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Coolant_Temp, col="Engine_Coolant_Temp"), size=0.7)+
  geom_line(aes(y=RPMs, col="RPMs"), size=0.7)+
  geom_line(aes(y=Wheel_based_vehicle_Speed, col="Wheel_based_vehicle_Speed"), size=0.7)+
  geom_line(aes(y=Transmission_Oil_Temp, col="Transmission_Oil_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Rate, col="Engine_Fuel_Rate"), size=0.7)+
  geom_line(aes(y=Engine_Oil_Pressure, col="Engine_Oil_Pressure"))+
  scale_colour_brewer(palette = "Dark2")+
  labs(title="Standard Plot of Diagnostic Values",
       subtitle="From Dataset D",
       y="Standardized Diagnostic Values",
       x="Time" )+
  theme_bw(base_size = 16,
           base_line_size = .5)
################################
###### p a i r s   p l o t #####
```

```
######### DATASET D AND I ######
pairs(timeStamp ~ Engine_Intake_Manifold_Temp + Engine_Fuel_Temp + RPMs + Engine_Coolant_Temp +
       Wheel_based_vehicle_Speed + Transmission_Oil_Temp + Engine_Fuel_Rate + Engine_Oil_Pressure,
      panel=panel.smooth,
      lower.panel=NULL,
      pch=20, cex=2,
      lwd=1,
      col="dimgray",
      main=paste("Pairs Plot of Diagnostics", "\nDataset L"),
      cex.axis = 1.3,
      data=df_L)
pairs(timeStamp ~ Engine_Intake_Manifold_Temp + Engine_Fuel_Temp + RPMs + Engine_Coolant_Temp +
       Wheel_based_vehicle_Speed + Transmission_Oil_Temp + Engine_Fuel_Rate + Engine_Oil_Pressure,
      upper.panel=NULL,
      panel=panel.smooth,
      pch=20, cex=2,
      lwd=1,
      col="dimgray",
      main=paste("Pairs Plot of Diagnostics", "\nDataset F"),
      cex.axis = 1.3,
      data=df_F)
####
###### plot  DF_F
######
Scaled.df_F <- as.data.frame(scale(df_F[,4:11]))  # <- all diagnostic parmeters
Scaled.df_F$RPMs
#check to make sure its 1 sd
sd(Scaled.df_F$RPMs)
sd(Scaled.df_F$Engine_Intake_Manifold_Temp)
## add scaled data to dataframe for plotting
df_F.with.scaled <- cbind(df_F[,1:3], Scaled.df_F)
head(df_F.with.scaled, 20)
ggplot(data=df_F.with.scaled, aes(x=timeStamp, y=values, color=Parameters)) +
  geom_line(aes(y=Engine_Intake_Manifold_Temp, col="Engine_Intake_Manifold_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Temp, col="Engine_Fuel_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Coolant_Temp, col="Engine_Coolant_Temp"), size=0.7)+
  geom_line(aes(y=RPMs, col="RPMs"), size=0.7)+
  geom_line(aes(y=Wheel_based_vehicle_Speed, col="Wheel_based_vehicle_Speed"), size=0.7)+
  geom_line(aes(y=Transmission_Oil_Temp, col="Transmission_Oil_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Rate, col="Engine_Fuel_Rate"), size=0.7)+
  geom_line(aes(y=Engine_Oil_Pressure, col="Engine_Oil_Pressure"), size=0.7)+
  scale_colour_brewer(palette = "Dark2")+
  labs(title="Standard Plot of Diagnostic Values",
       subtitle="From Dataset F",
       y="Standardized Diagnostic Values",
       x="Time" )+
```

```
  theme_bw(base_size = 16,
           base_line_size = .5)
# pairs plot
pairs(timeStamp ~ Engine_Intake_Manifold_Temp + Engine_Fuel_Temp + RPMs + Engine_Coolant_Temp +
        Wheel_based_vehicle_Speed + Transmission_Oil_Temp + Engine_Fuel_Rate + Engine_Oil_Pressure,
      lower.panel=NULL,
      panel=panel.smooth,
      pch=20, cex=1,
      lwd=0.3,
      col="dimgray",
      main=paste("Pairs Plot of Diagnostics", "\nDataset F"),
      cex.axis = 1.3,
      data=df_F)
####
###### plot  DF_L
######
# install.packages("RColorBrewer")
# library(RColorBrewer)
# library(ggplot2)
# library(reshape2)
# library(col)
# Standardized data columns
Scaled.df_L <- as.data.frame(scale(df_L[,4:11]))  # <- all diagnostic parmeters
Scaled.df_L$RPMs
#check to make sure its 1 sd
sd(Scaled.df_L$RPMs)
sd(Scaled.df_L$Engine_Intake_Manifold_Temp)
## add scaled data to dataframe for plotting
df_L.with.scaled <- cbind(df_L[,1:3], Scaled.df_L)
head(df_L.with.scaled, 20)
ggplot(data=df_L.with.scaled, aes(x=timeStamp, y=values, color=Parameters)) +
  geom_line(aes(y=Engine_Intake_Manifold_Temp, col="Engine_Intake_Manifold_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Temp, col="Engine_Fuel_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Coolant_Temp, col="Engine_Coolant_Temp"), size=0.7)+
  geom_line(aes(y=RPMs, col="RPMs"), size=0.7)+
  geom_line(aes(y=Wheel_based_vehicle_Speed, col="Wheel_based_vehicle_Speed"), size=0.7)+
  geom_line(aes(y=Transmission_Oil_Temp, col="Transmission_Oil_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Rate, col="Engine_Fuel_Rate"), size=0.7)+
  geom_line(aes(y=Engine_Oil_Pressure, col="Engine_Oil_Pressure"), size=0.7)+
  scale_colour_brewer(palette = "Dark2")+
  labs(title="Standard Plot of Diagnostic Values",
       subtitle="From Dataset L",
       y="Standardized Diagnostic Values",
       x="Time" )+
  theme_bw(base_size = 16,
           base_line_size = .5)
```

```
# pairs plot
pairs(timeStamp ~ Engine_Intake_Manifold_Temp + Engine_Fuel_Temp + RPMs + Engine_Coolant_Temp +
       Wheel_based_vehicle_Speed + Transmission_Oil_Temp + Engine_Fuel_Rate + Engine_Oil_Pressure,
      upper.panel=NULL,
      panel=panel.smooth,
      pch=20, cex=1,
      lwd=1,
      col="dimgray",
      main=paste("Pairs Plot of Diagnostics", "\nDataset L"),
      cex.axis = 1.3,
      data=df_L)
Scaled.df_I <- as.data.frame(scale(df_I[,4:11]))  # <- all diagnostic parmeters
Scaled.df_I$RPMs
#check to make sure its 1 sd
sd(Scaled.df_I$RPMs)
sd(Scaled.df_I$Engine_Intake_Manifold_Temp)
## add scaled data to dataframe for plotting
df_I.with.scaled <- cbind(df_I[,1:3], Scaled.df_I)
head(df_I.with.scaled, 20)
ggplot(data=df_I.with.scaled, aes(x=timeStamp, y=values, color=Parameters)) +
  geom_line(aes(y=Engine_Intake_Manifold_Temp, col="Engine_Intake_Manifold_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Temp, col="Engine_Fuel_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Coolant_Temp, col="Engine_Coolant_Temp"), size=0.7)+
  geom_line(aes(y=RPMs, col="RPMs"), size=0.7)+
  geom_line(aes(y=Wheel_based_vehicle_Speed, col="Wheel_based_vehicle_Speed"), size=0.7)+
  geom_line(aes(y=Transmission_Oil_Temp, col="Transmission_Oil_Temp"), size=0.7)+
  geom_line(aes(y=Engine_Fuel_Rate, col="Engine_Fuel_Rate"), size=0.7)+
  geom_line(aes(y=Engine_Oil_Pressure, col="Engine_Oil_Pressure"), size=0.7)+
  scale_colour_brewer(palette = "Dark2")+
  labs(title="Standard Plot of Diagnostic Values",
       subtitle="From Dataset I",
       y="Standardized Diagnostic Values",
       x="Time" )+
  theme_bw(base_size = 16,
           base_line_size = .5)
# pairs plot
pairs(timeStamp ~ Engine_Intake_Manifold_Temp + Engine_Fuel_Temp + RPMs + Engine_Coolant_Temp +
       Wheel_based_vehicle_Speed + Transmission_Oil_Temp + Engine_Fuel_Rate + Engine_Oil_Pressure,
      lower.panel=NULL,
      panel=panel.smooth,
      pch=20, cex=1,
      lwd=0.5,
      col="dimgray",
      main=paste("Pairs Plot of Diagnostics", "\nDataset I"),
      cex.axis = 1.3,
      data=df_I)
```

```
# Dimensions of the dataset
numseconnds <- nrow(diagnosticDT)
numDiagnosticParam <- ncol(diagnosticDT) - 1
library(ggplot2)  # for creating graphs
par(mfrow = c(3, 4))
for(spns in 3:numDiagnosticParam + 1) {
  with(diagnosticDT,
       plot(timeStamp, unlist(diagnosticDT[,spns]),
            type="l",
            main=paste(colnames(diagnosticDT)[spns],"vs Time"),
            ylab=colnames(diagnosticDT)[spns], xlab="Time",
            pch=20, cex=0.7, lwd=2, col="dimgray")
  )
}
############## M O D E L I N G  #################

# What is your research question? Does X affect Y? What are the predictors of Y?
# only using
# DATASETS A , D, F, L, I
str(df_A)
# Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 59 obs. of  11 variables:
#  $ by30min                  : Factor w/ 12 levels "2019-03-20 15:50:00",..: 1 1 1 1 1 1 1 1 1 1 ...
#  $ timeStamp                : POSIXct, format: "2019-03-20 15:50:58" "2019-03-20 15:51:28" "2019-03-20
15:51:58" "2019-03-20 15:52:28" ...
#  $ seconds                  : Factor w/ 15625 levels "000000","000064",..: 2486 4297 4475 4421 4599 4545
4723 4669 4911 4793 ...
#  $ Engine_Intake_Manifold_Temp: num  23.4 20.8 20 20 20 ...
#  $ Engine_Fuel_Temp         : num  24.6 22.4 22 22 22 ...
#  $ RPMs                     : num  702 700 745 908 805 ...
#  $ Engine_Coolant_Temp      : num  73.2 73.8 74.3 75.9 77.6 ...
#  $ Wheel_based_vehicle_Speed  : num  0 0 1.84 6.44 6.15 ...
#  $ Transmission_Oil_Temp    : num  51.9 52 52.8 53.9 55 ...
#  $ Engine_Fuel_Rate         : num  3.81 3.74 5.69 9.87 6.44 ...
#  $ Engine_Oil_Pressure      : num  171 168 180 219 192 ...
## ##### ###
install.packages("leaps")
install.packages("faraway")
#  Load packages we'll need
#
library(faraway)
library(MASS)
library(leaps)
##########################################################################################
#########################

################## A R D L ##################
```

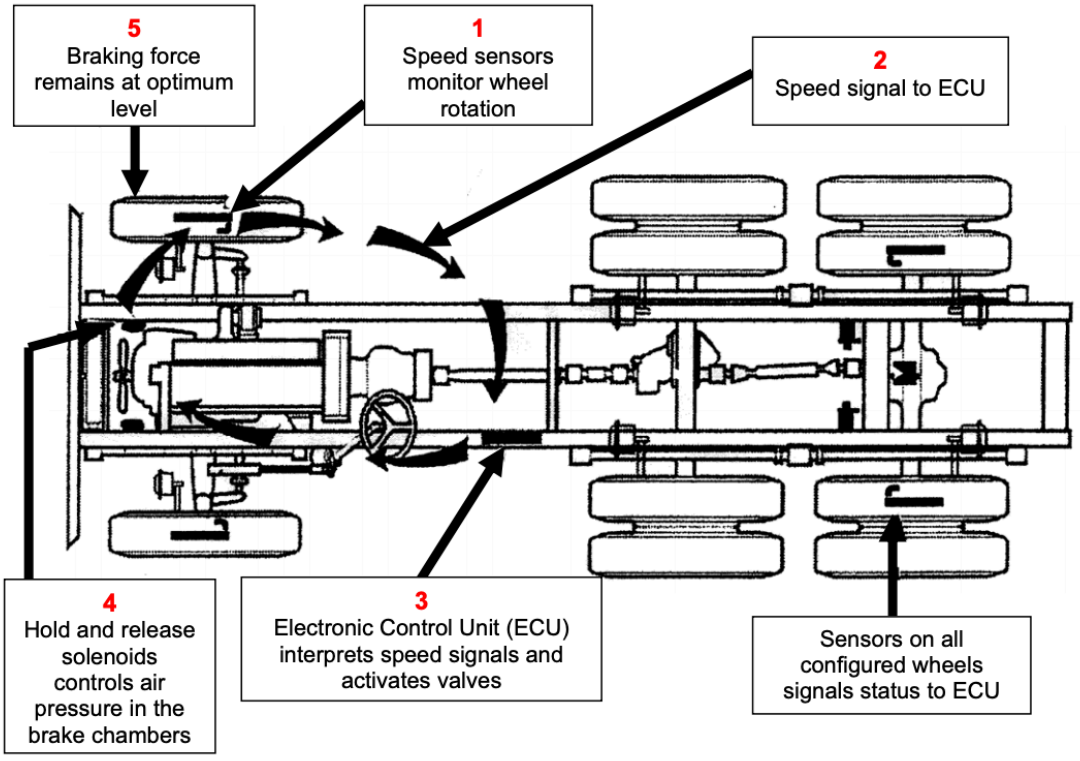
```
########## T I M E   S E R I E S ############

#################################################################################################
########################

#### Engine speed is the independent variable, X
#### we need to find which parameter (temp, pressure) interacts most with speed.
### For this dataset it was Engine Oil Pressure
### Datasets F and L have the closet  corr coefficients and their plots have similar patterns. B/c we did three loops, the first loops and
####the second loop are represented
### plot each one against speed
# lm with Engine temp as Y
#### we need to build a lm with RPM as Y and Engine Oil pressure as X, b/c it has the highest correlation coefficient in our datasets
### make a new data frame, oil pressure, RPM, and Time 1...N
# engine oil pressure (yt) rpm(xt)
t
# r = (rpm(x)) and p= (oil pressure(y)) tell you what the time lag is for y [], and x respectively.
# make a new data frame
# t = length of time stamps, y(t) is oil pressure, x(t) is RPM, y(t-1), x(t-1), ...
# new data
dim(df_F) #[1] 39 11
dim(df_L) #[1] 57 11
new.data.F <- df_F[df_F$RPMs !=0,] # takes out row where RPMs are zero, we know this is from Dataset G and H that had no observations
(t <- rownames(new.data.F))
new.data.F <- cbind(t=t, new.data.F) # this is not needed at this point
str(new.data.F)
acf(new.data.F[ ,c("RPMs") ], plot=TRUE, main="Autocorrelation Function Plot of Dataset",
    lwd=3, col="dimgray", xlab="Lag Length", cex=2)
par(mfrow=c(1,1))
new.data.F<-new.data.F[, !(colnames(new.data.F) %in% c('by30min', 'timeStamp', 'seconds'))]
str(new.data.F)
# make t column a numeric so we can make t-1 columns
# new.data.F$t <- as.numeric(levels(new.data.F$t)[new.data.F$t])# do not need this
str(new.data.F)
#make final data frame
# the time of parameters before matter at tminust
df <- new.data.F[ ,c("t","Engine_Oil_Pressure", "RPMs")]
acf(as.numeric(as.matrix(df)), plot=TRUE)
nrow <- dim(df)[1]
df$y.tminus1 <- c(NA, df$Engine_Oil_Pressure[1:nrow-1])
df$y.tminus2 <- c(NA, df$y.tminus1[1:nrow-1])
df$y.tminus3 <- c(NA, df$y.tminus2[1:nrow-1])
df$x.tminus1 <- c(NA, df$RPMs[1:nrow-1])
```

```
df$x.tminus2 <- c(NA, df$x.tminus1[1:nrow-1])
df$x.tminus3 <- c(NA, df$x.tminus2[1:nrow-1])
head(df)
acf(as.numeric(as.matrix(df)), plot=TRUE)
# model 0
# 0 lags
# r = p = 0
new.data<-df
lm.nolag<- lm(Engine_Oil_Pressure~RPMs, data=new.data)
summary(lm.nolag)
par(mfrow=c(1,1))
par(mfrow=c(2,2))
plot(lm.nolag)
acf(lm.nolag, plot=TRUE)
require(lmtest)
dwtest(lm.nolag)
dwtest(lm.onelag) # the error times are not auto-correlated, alt hypothesis the error terms negatively auto correlated
#there is no positive or neg autocorrelation in our error times
dwtest(lm.twolag)
# model 1
# 1 Lag
# r = p = 1
# throw out the zeros in RPMs and Engine Pressure we are seeing
# zeros come from when truck was reset after conducting driving lap
new.data
lm.onelag<- lm(Engine_Oil_Pressure~y.tminus1 + RPMs + x.tminus1, data = new.data)
summary(lm.onelag)
acf(resid(lm.onelag), plot=TRUE, lwd=3, col="dimgray", main="ARDL One Lag (Resid Auto Correlation Plot)", xlab="Lag Length", cex=2)
acf(resid(lm.twolag))
acf(resid(lm.fivelag))
par(mfrow=c(1,2))
plot(lm.onelag, col="dimgray", pch=20, cex=1, lwd=1)
dwtest(lm.onelag)
###############
# PREDICTION
###############
y.hat<- predict(lm.onelag, new.data)
# acf(y.hat[-1], plot=TRUE)
y.hat
length(y.hat)
par(mfrow=c(1,1))
plot(1:33, y.hat, type = "b", col="dimgray", lwd=1.3, cex=.8,
    xlab="Time", ylab="Engine Oil Pressure",
    main="Predicted Values vs Actual Values")
```

```
points((1:33), new.data$Engine_Oil_Pressure, type="b", col="maroon", lwd=.8)
legend("left", legend=c("Predicted", "Acutal"),
       col=c("dimgray", "maroon"), lty=1, cex=1)
dim(new.data)
# now try on dataset
# lets make the test dataset have the same number predictor columns
nrow <- dim(df_L)[1]
df_L$y.tminus1 <- c(NA, df_L$Engine_Oil_Pressure[1:(nrow-1)])
df_L$x.tminus1 <- c(NA, df_L$RPMs[1:(nrow-1)])
L.y.hat<- predict(lm.onelag, df_L)
df_L
par(mfrow=c(1,1))
plot(1:length(L.y.hat), L.y.hat, type = "b", col="dimgray", lwd=1.5,cex=.8,
     xlab="Time", ylab="Engine Oil Pressure",
     main="Predicted Values vs Actual Values on Test Dataset")
points((1:length(L.y.hat)), df_L$Engine_Oil_Pressure, type="b", col="firebrick", lwd=1)
legend("left", legend=c("Predicted", "Acutal"),
       col=c("dimgray", "firebrick"), lty=1, cex=1)
#########################################################
# FINAL PLOTS
# oil pressure residuals
plot(1:length(L.y.hat), (df_L$Engine_Oil_Pressure-L.y.hat), col="dimgray", lwd=1.3, type="p", cex=.5,
     xlab="Time", ylab="Engine Oil Pressure Residuals",
     main="Residuals")
abline(0,0, lty=2, col="maroon")
#these are the difference of predicted and actual
a<-(df_L$Engine_Oil_Pressure-L.y.hat) # residuals
acf(as.matrix(a[-1]), plot = TRUE, lwd=3, col="dimgray",
    main="Residual Auto Correlation Plot", xlab="Lag Length", cex=2)
mean(abs(a[-1]))  # drop the first index b/c it is "NA"
#mean absolute deviation 72. is only
(5.08/1000)* 100
# mean std is 7% of range, model does well job at predicting Engine Oil pressure, which determine engine
operating conditions

# model 2
# 2 Lag
# r = p = 2
lm.twolag<-  lm(Engine_Oil_Pressure~y.tminus1 + y.tminus2 + RPMs + x.tminus1 + x.tminus2, data =
new.data)
summary(lm.twolag)
par(mfrow=c(2,2))
plot(lm.twolag)

# model 3
# 5 lag
```

```
# r = 5 = p
# add to more columns for x and y
df_L$y.tminus1 <- c(NA, df$Engine_Oil_Pressure[1:nrow-1])
df$y.tminus2 <- c(NA, df$y.tminus1[1:nrow-1])
df$y.tminus3 <- c(NA, df$y.tminus2[1:nrow-1])
df$y.tminus4 <- c(NA, df$y.tminus3[1:nrow-1])
df$y.tminus5 <- c(NA, df$y.tminus4[1:nrow-1])
df$x.tminus1 <- c(NA, df$RPMs[1:nrow-1])
df$x.tminus2 <- c(NA, df$x.tminus1[1:nrow-1])
df$x.tminus3 <- c(NA, df$x.tminus2[1:nrow-1])
df$x.tminus4 <- c(NA, df$x.tminus3[1:nrow-1])
df$x.tminus5 <- c(NA, df$x.tminus4[1:nrow-1])
new.data<-df
lm.fivelag<- lm(Engine_Oil_Pressure~y.tminus1 + y.tminus2 +df$y.tminus3 +df$y.tminus4 + df$y.tminus5 +
                RPMs + x.tminus1 + x.tminus2 + x.tminus3 + x.tminus4 + x.tminus5, data = new.data)
summary(lm.fivelag)
par(mfrow=c(1,1))
acf(resid(lm.fivelag), plot=TRUE, lwd=3, col="dimgray",
   main="ARDL Five Lags (Resid Auto Correlation Plot)", xlab="Lag Length", cex=2)

par(mfrow=c(1,2))
plot(lm.fivelag, col="dimgray", pch=20, cex=1, lwd=1)
dwtest(lm.fivelag)
```

APPENDIX C. R ECU DIAGRAM FOR MTVR



ECUs on the MTVR transmit diagnostic data through the CAN serial bus using the J1939 Protocol

Figure 28. ECU diagram for MTVR. Source: [58].

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. PENDLETON DRIVING COURSE

Figure 29. Camp Pendleton Driving Course Loop

- Start/End: 43 Area Las Pulgas Rd,
- Head northwest on Basilone Rd, turn north on San Mateo Rd.
- Turn around point: 63 Area Fire station,
- South on Basilone Rd, west on Vandegrift Blvd, Head northwest on Stuart Mesa Rd, turn east on Las Pulgas Canyon Road.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] *Marine Corps Doctrinal Publication 4, Logistics*, MCDP 4. Washington, DC, USA: Headquarters United States Marine Corps, 1997 [Online]. Available: https://www.marines.mil/Portals/59/Publications/MCDP%204%20Logistics.pdf

[2] J. H. Shin and H. B. Jun, "On condition based maintenance policy," *J. Comput. Des. Eng.*, vol. 2, no. 2, pp. 119–127, Apr. 2015 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2288430014000141

[3] J. Kenny, "NAVSEA CBM+," presented at DoD Maintenance Symposium., Tampa Downtown Conv. Center, Tampa Bay, FL, Dec. 18, 2018. [Online]. Available: https://www.sae.org/events/dod/attend/program/presentations/Kenny.pdf

[4] A. Menard, "How can we recognize the real power of the internet of things," *McKinsey Co.*, p. 8, Nov. 2017 [Online]. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/how-can-we-recognize-the-real-power-of-the-internet-of-things

[5] *Condition Based Maintenance Plus (CBM+) for Materiel Maintenance*, DoDI 4151.22. Department of Defense, Washington DC, USA, 2012 [Online]. Available: https://www.acq.osd.mil/log/MPP/.policy.html/7_DODI415122p1.pdf

[6] S. P. Yue, "Modeling of engine parameters for condition-based maintenance of the MTU series 2000 diesel engine," M.S. thesis, Dept. of Oper. Res., NPS, Monterey, CA, USA, 2016. [Online]. Available: http://hdl.handle.net/10945/50512

[7] A. Schuldt, *Multiagent Coordination Enabling Autonomous Logistics*. Berlin, Heidelberg: Springer, 2011.

[8] *Marine Corps Warfighting Publication 4-1, Logistics Operations*. Washington, DC, USA: Headquarters United States Marine Corps, 1999 [Online]. Available: https://www.marines.mil/Portals/59/Publications/MCWP%203-40.pdf?ver=2017-03-15-124213-007

[9] J. Mattis, "Summary of the 2018 national defense strategy," Department of Defense, Washington, DC, USA, 2018 [Online]. Available: https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf

[10] J. McLean, "One single nail: No excuses for failing to deliver," *Mar. Corps Gaz. Prof. J. US Mar.*, vol. 92, no. 2, p. 6, Feb. 2008.

[11] "The Marine corps operating concept: How an expeditionary force operates in the 21st century," Headquarters United States Marine Corps, Washington, DC, USA, Sep. 2016 [Online]. Available: https://www.mccdc.marines.mil/Portals/172/Docs/MCCDC/young/MCCDC-YH/document/final/Marine%20Corps%20Operating%20Concept%20Sept%202016.pdf?ver=2016-09-28-083439-483

[12] A. H. C. Tsang, "Condition-based maintenance: Tools and decision making," *J. Qual. Maint. Eng.*, vol. 1, no. 3, pp. 3–17, Sep. 1995 [Online]. Available: https://www-emeraldinsight-com.libproxy.nps.edu/doi/full/10.1108/13552519510096350

[13] Diane K. Morales, "Conditions based maintenance plus memorandum," Department of Defense, Washington, DC, USA, Nov. 2002.

[14] *Reliability Centered Maintenance (RCM)*, DoD 4151.22-M. Department of Defense, 2011 [Online]. Available: https://www.wbdg.org/FFC/DOD/DODMAN/415122-M.pdf

[15] O. Sénéchal, "Performance indicators nomenclatures for decision making in sustainable conditions based maintenance," *IFAC-Pap.*, vol. 51, no. 11, pp. 1137–1142, Jan. 2018 [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896318315659

[16] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mech. Syst. Signal Process.*, vol. 20, no. 7, pp. 1483–1510, Oct. 2006 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0888327005001512

[17] O. A. Vanli, "A failure time prediction method for condition-based maintenance," *Qual. Eng.*, vol. 26, no. 3, pp. 335–349, Jul. 2014 [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/08982112.2013.830740

[18] *Condition Based Maintenance Plus DoD Guidebook*, Washington, DC, USA: Department of Defense, 2008 [Online]. Available: https://www.dau.mil/guidebooks/Shared%20Documents%20HTML/Condition%20Based%20Maintenance%20Plus%20(CBM+)%20Guidebook.aspx#toc81

[19] W. Voss, *A Comprehensible Guide to J1939*. Copperhill Technologies Corporation, 2008.

[20] C. S. S. Electronics, "Can bus explained - a simple intro (2019)," *CSS Electronics*. Accessed Nov 13, 2018. [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus

[21] Y. Burakova, B. Hass, L. Millar, and A. Weimerskirch, “Truck hacking: An experimental analysis of the SAE J1939 standard,” in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, Austin, TX, 2016, p. 10. [Online]. Available: https://www.usenix.org/conference/woot16/workshop-program/presentation/burakova

[22] L. F. Lara, “Analysis of the medium tactical vehicle replacement (MTVR) contractor logistics support (cls) contract,” M.S. thesis, Dept. of Bus. and Pub. Pol., NPS, Monterey, CA, USA, 2001. [Online]. Available: http://hdl.handle.net/10945/9705

[23] J. Guerrero, “Diagnostic analyzer test set (DATS-I).” Dr Diesel Technologies, 2012 [Online]. Available: http://www.drdieseltech.com/uploads/DATS_II_FZ-G1_US_Army.pdf

[24] *Ground Equipment Maintenance Program*, MCO 4790.25. Washington, DC, USA: Headquarters United States Marine Corps, 2014 [Online]. Available: https://www.marines.mil/Portals/59/MCO%204790.25.pdf

[25] M. Dade, “Examples of Aircraft Scheduled Maintenance Analysis Problems,” RAND Corp., Santa Monica, CA, USA, Rep. R-1299-R, 1973. [Online]. Available: https://www.rand.org/content/dam/rand/pubs/reports/2006/R1299.pdf

[26] *Air Force Readiness: Hearing before the Subcommittee on Readiness and Management Support Committee on Armed Services*. 115th Cong. (2018) [Online]. Available: https://www.armed-services.senate.gov/hearings/18-04-18-air-force-modernization

[27] “Conditioned based maintenance plus: Is your program ready?,” *Dayton Aerospace*. Aug. 6, 2018 [Online]. Available: https://daytonaero.com/cbm-plus-implementation-is-your-program-ready/

[28] B. Biagini, “B-1 CBM+ Goes Live!,” *Air Force Materiel Command News*, Nov. 15, 2018 [Online]. Available: https://www.afmc.af.mil/News/Article-Display/Article/1692433/b-1-cbm-goes-live/

[29] S. Bleymaier, “Condition based maintenance plus,” presented at DoD Maintenance Symposium., Tampa Downtown Conv. Center, Tampa Bay, FL, Dec. 18, 2018 [Online]. Available: https://www.sae.org/events/dod/attend/program/presentations/Bleymaier.pdf

[30] *Air Force, Force Structure and Modernization Programs: Presentation to the Senate Armed Services Committee Subcommittee on Airland Forces United States Senate*. 115th Cong (2018) (statement of Brian Robinson, USAF Assistant Deputy Chief of Staff, Operations; statement of Jerry Harris, USAF Deputy Chief of Staff for Strategic Plans and Requirements; statement of Arnold Bunch, Military Deputy, Office of the Assistant Secretary of the Air Force for Acquisition) [Online]. Available: https://www.armed-services.senate.gov/imo/media/doc/Bunch-Harris-Robinson_04-18-18.pdf

[31] M. Bounds, M. Calomeris, M. Pohland, and M. Shepler, "Army logistician: On the road to condition-based maintenance for army vehicles," *Army Logist. Prof. Bull. U. S. Army Logist.*, vol. 40, no. 4, Aug. 2008 [Online]. Available: https://alu.army.mil/alog/issues/julaug08/condbased_maintennance.html

[32] K. Guite, "If it ain't broke…," *Army AL&T Magazine*, Oct.-Dec. Sep. 11, 2018. [Online]. Available: https://www.dvidshub.net/news/printable/292156

[33] R. Scheltema, G. Smith, D. Jeska, and J. Banks, "SBCT embedded data collection and analysis system (SEDCAS)," p. 31.

[34] J. Banks, T. Bair, K. Reichard, D. Blackstock, D. McCall, and J. Berry, "A demonstration of a helicopter health management information portal for U.S. Army aviation," in *2005 IEEE Aerospace Conference*, Big Sky, MT, USA, 2005, pp. 3748–3755 [Online]. Available: http://ieeexplore.ieee.org/document/1559681/

[35] D. M. Cutter and O. R. Thompson, "Condition-Based Maintenance Plus Select Program Survey:," Defense Technical Information Center, Fort Belvoir, VA, Jan. 2005 [Online]. Available: http://www.dtic.mil/docs/citations/ADA430668

[36] D. M. Bartos, "Enabling automated equipment condition data updates through use of vehicle integrated sensor networks and single board computers," M.S. thesis, Dept. of Comp. Sci., NPS, Monterey, CA, USA, 2018.

[37] T. Smith, "USAF Condition-Based Maintenance Plus (CBM+) Initiative," Air Force Logistics Management Agency, Maxwell AFB, Gunter Annex, AL, Final Report No. LM200301800, Sep. 2003.

[38] A. Hess, G. Calvello, and T. Dabney, "PHM a key enabler for the JSF autonomic logistics support concept," in *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*, 2004, vol. 6, pp. 3543-3550 Vol.6.

[39] W. Bellamy, "Southwest COO Talks Predictive Maintenance in Wake of 1380 at AMC/AEEC 2018," *Access Intelligence: Avionics International*, Apr. 24, 2018 [Online]. Available: https://www.aviationtoday.com/2018/04/24/southwest-coo-talks-predictive-maintenance-amcaeec-2018/

[40] N. Naughton, “Apps beyond appts,” *Automot. News*, vol. 90, no. 6728, pp. 0022–0022, Jun. 2016 [Online]. Available: http://libproxy.nps.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=116055083&site=ehost-live&scope=site

[41] C. Albon, “Air Force eyes FY-19 reprogramming to meet Mattis’ 80 percent fighter readiness mandate,” *Insid. Aircr. Alert*, Nov. 2018 [Online]. Available: https://search.proquest.com/docview/2136224224/citation/924A30AF73934B3FPQ/1

[42] “Bidding now open for USMC ITV and MTVR vehicles (non-DLA) on govplanet.com,” *PRNewswire.com*, Mar. 13, 2019. [Online]. Available: https://www.prnewswire.com/news-releases/bidding-now-open-for-usmc-itv-and-mtvr-vehicles-non-dla-on-govplanetcom-300811390.html

[43] J. Devore, *Probability and Statistics for Engineering and the Sciences*, 9th Edition. Boston, MA: Cengage Learning, 2016.

[44] G. A. Klutke, P. C. Kiessler, and M. A. Wortman, “A critical look at the bathtub curve,” *IEEE Trans. Reliab.*, vol. 52, no. 1, pp. 125–129, Mar. 2003.

[45] M. I. Ullah, “Time series analysis and forecasting,” *Basic Statistics and Data Analysis*, Dec. 27, 2013. [Online]. Available: http://itfeature.com/time-series-analysis-and-forecasting/time-series-analysis-forecasting

[46] W. H. Greene, *Econometric Analysis*, 6. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2008.

[47] “Raspberry Pi 3 Model B+.” Raspberry Pi Foundation [Online]. Available: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[48] SK Pang Electronics, “PiCAN 2 DUO with SMPS Datasheet.” SK Pang Electronics, 2016 [Online]. Available: http://skpang.co.uk/

[49] DG Technologies, “DG technologies product pinouts and industry connectors reference guide,” Famington Hills, MI, Apr. 23, 2014 [Online]. Available: https://www.dgtech.com/wp-content/uploads/2016/04/Pinouts_ICR.pdf

[50] “SAE J1939 standards collection,” *https://www.sae.org*. [Online]. Available: https://www.sae.org/publications/collections/content/j1939_dl/

[51] “Vehicle application layer J1939/71.” SAE International, 2003 [Online]. Available: https://www.sae.org/standards/content/j1939/71_201309/

[52] Python Software Foundation, “What is Python? Executive summary,” *www.python.org*. [Online]. Available: https://www.python.org/doc/essays/blurb/

[53] R Core Team, “R: A Language and Environment for Statistical Computing,” Vienna, Austria, 2019 [Online]. Available: https://www.R-project.org/

[54] Jeremy Daily, “Using a database to decode J1939 messages,” *The University of Tulsa Crash Reconstruction Research Consortium*, 2013. [Online]. Available: http://tucrrc.utulsa.edu/Databases/ParseLogFile.py

[55] Steffen Moritz and Thomas Bartz-Beielstein, “imputeTS: Time series missing value imputation in R,” *R J.*, vol. 9, no. 1, pp. 207–218, 2017 [Online]. Available: https://journal.r-project.org/archive/2017/RJ-2017-009/index.html

[56] S. Sharma, M. Upreti, B. Sharma, and K. Poddar, “Analysis of variation in oil pressure in lubricating system,” presented at the 2nd International Conference on Condensed Matter And Applied Physics (ICC 2017), Bikaner, India, 2018, p. 4 [Online]. Available: http://aip.scitation.org/doi/abs/10.1063/1.5033171

[57] B. Castanier, A. Grall, and C. Bérenguer, “A condition-based maintenance policy with non-periodic inspections for a two-unit series system,” *Reliab. Eng. Syst. Saf.*, vol. 87, no. 1, pp. 109–120, Jan. 2005 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0951832004001097

[58] “Trainee guide for Medium Tactical Vehicle Replacement (MTVR) operators course, A-730-0045A.” Naval Education and Training Command, Mar-2007 [Online]. Available: https://navytribe.files.wordpress.com/2015/11/a-730-0045.pdf

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California