WESTERN SYDNEY
UNIVERSITY

SCHOOL OF BUSINESS

ASSIGNMENT COVER SHEET

STUDENT DETAILS

Student name: Nguyễn Thị Ngọc Phương Student ID number: 22118775

UNIT AND TUTORIAL DETAILS

Unit name: Analytics Programming Unit number: AP-T125WSD-1

Tutorial group: _____ Tutorial day and time: Sun 3:30 to 6:45 pm

Lecturer or Tutor name: Assoc. Prof. Nguyen Tan Luy

ASSIGNMENT DETAILS

Title: Individual Assignment

Length: 1923 words (exclude codes and comments within the codes)

Due date: 04/04/2025 Date submitted: 04/04/2025

Home campus (where you are enrolled): Vietnam

DECLARATION

☑ I hold a copy of this assignment if the original is lost or damaged.

☑ I hereby certify that no part of this assignment or product has been copied from any other student's work or from any other source except where due acknowledgement is made in the assignment.

☑ I hereby certify that no part of this assignment or product has been submitted by me in another (previous or current) assessment, except where appropriately referenced, and with prior permission from the Lecturer / Tutor / Unit Coordinator for this unit.

☑ No part of the assignment/product has been written/produced for me by any other person except where collaboration has been authorised by the Lecturer / Tutor /Unit Coordinator concerned.

☑ I am aware that this work will be reproduced and submitted to plagiarism detection software programs for the purpose of detecting possible plagiarism **(which may retain a copy on its database for future plagiarism checking).**

Student's signature:  Nguyễn Thị Ngọc Phương

Note: An examiner or lecturer / tutor has the right to not mark this assignment if the above declaration has not been signed.

Table of Content

1. Introduction

This report presents a data analysis project conducted at the Vietnam Vehicle Maintenance Center. The main objective of the project is to extract information from used vehicle data to support fault diagnosis, which assists in developing more effective maintenance strategies. The project focuses on three interconnected datasets: Engine, Automobile, and Maintenance records. Each dataset provides detailed information on engine specifications, vehicle construction characteristics, and repair history. The analysis in the report is divided into four parts: testing and processing data, analyzing horsepower distribution, identifying common engine faults by fuel type, and investigating factors affecting maintenance methods.

2. Loading Packages

To ensure that the analysis process is efficient and seamless, all tasks are performed in a single R script, as the steps are tightly interconnected. From the start, the tidyverse library package is loaded. This is a collection of tools that are very popular in the data analysis field, including: `dplyr` for data processing, `ggplot2` for visualization, and `readr` for reading CSV files. Tidyverse provides an easy-to-use and consistent syntax, making it easier to write and follow source code, which is especially useful when working with complex structured data (Wickham et al., 2019).

```
# Load all required packages for the entire analysis (Tasks 1–4)
  library(tidyverse)
```

3. Tasks

3.1 Task 1: Data Import and Cleaning, and basic Visualization

The first task was to prepare the data for further analysis by handling missing values, converting variables to the appropriate format, and visualizing a key variable, `Horsepower`. The majority of operations are performed on the `engine` data set, while the remaining two sets (`automobile` and `maintenance`) are also initially processed to ensure data format consistency.

A) Import Data & Replace "?" with NA:

First, the working directory is identified and set up to ensure that the data files can be accessed correctly. Then, three data files, Engine.csv, Automobile.csv, and Maintenance.csv, are imported using the

`read.csv()` function. Then, the `na.strings = "?"` parameter is specified so that R understands that the "?" in the data represents missing values and automatically converts them to `NA`.

After the replacement, the `sum()` function is used to check again if there are any "?" values that have not been converted. The result is 0, indicating that all "?" values have been processed correctly.

```
# Check the current working directory to understand where R is looking for files
getwd()

# Set the working directory to the folder where the data files are stored.
setwd("C:/Users/LENOVO/Downloads/[Bdata] AP/Individual Report")

#Import data and replace "?" with NA
engine <- read.csv("Engine.csv", na.strings = "?")
automobile <- read.csv("Automobile.csv", na.strings = "?")
maintenance <- read.csv("Mantenance.csv", na.strings = "?")

#Checking Output
sum(engine == "?", na.rm = TRUE)
sum(automobile == "?", na.rm = TRUE)
sum(maintenance == "?", na.rm = TRUE)
```

```
> sum(engine == "?", na.rm = TRUE)
[1] 0
> sum(automobile == "?", na.rm = TRUE)
[1] 0
> sum(maintenance == "?", na.rm = TRUE)
[1] 0
```

```
#Result = 0 meaning that there is no "?" remaining in the dataset.
```

B) Convert categorical variables to factors:

Secondly, some categorical variables such as `BodyStyles`, `FuelTypes`, and `ErrorCodes` are converted to factor types using the `as.factor()` function. This ensures that categorical variables can be grouped correctly for later visualization or statistical analysis. The `str()` function is then used to confirm that the variables have been successfully converted to factors.

```
#Convert variable to factors type
automobile$BodyStyles <- as.factor(automobile$BodyStyles)
  #covert BodyStyle to factors type
engine$FuelTypes <- as.factor(engine$FuelTypes)
  #covert FuelTypes to factors type
```

```
maintenance$ErrorCodes <- as.factor(maintenance$ErrorCodes)
  #covert FuelTypes to factors type

#Output checking
str(automobile$BodyStyles)  # str() is used check data type
str(engine$FuelTypes)       # str() is used check data type
str(maintenance$ErrorCodes) # str() is used check data type
```

```
> str(automobile$BodyStyles)
 Factor w/ 5 levels "convertible",..: 1 3 4 4 4 4 5 4 3 4 ...
> str(engine$FuelTypes)
 Factor w/ 2 levels "diesel","gas": 2 2 2 2 2 2 2 2 2 2 ...
> str(maintenance$ErrorCodes)
 Factor w/ 3 levels "-1","0","1": 1 1 1 3 1 3 3 2 1 1 ...
```

```
# Result confirming three variables was converted into factors
```

C) Convert Horsepower to numeric and imputed missing values:

Firstly, the missing values in the `Horsepower` column were handled by calculating the average of the non-missing values (`mean()`) and assigning them to the missing cells (`NA`) using the `mutate()` function combined with `if_else()`. This is a common, simple and effective method when the missing rate is not large and does not cause data bias.

```
# Replace missing value by the mean of Horsepower
mean_horsepower <- mean(engine$Horsepower, na.rm = TRUE)
  #compute the mean of horsepower, remove any missing value for computation
engine <- engine %>%
  mutate(Horsepower = if_else(is.na(Horsepower), mean_horsepower, Horsepower))
  #replace all NA in the Horsepower column by the mean_horsepower.

#Output Checking
sum(is.na(engine$Horsepower))
```

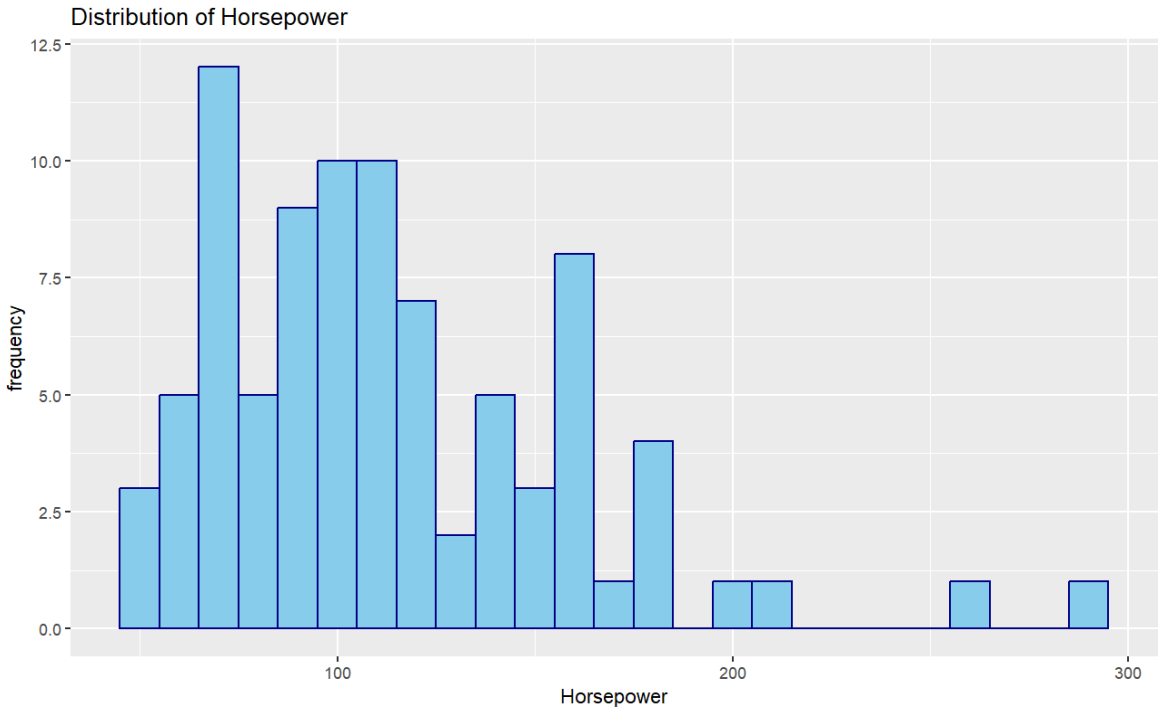
```
> sum(is.na(engine$Horsepower))
[1] 0
```

```
# Result = 0 meaning that there is no missing value within the Horsepower column
```

D) Visualize horsepower distribution

Finally, the `Horsepower` data is visualized using a histogram using `ggplot2`. The histogram divides the range of values into equal intervals (`binwidth = 20`) and displays the frequency of each interval. This visualization helps to identify the common horsepower region and assess the dispersion of the data.

```
ggplot(engine, aes(x = Horsepower)) +
  geom_histogram(binwidth = 20, fill = "skyblue", color =  darkblue ) +
    # Creates a histogram with bin width of 20 and styled appearance
  labs(title = "Distribution of Horsepower",
       x = "Horsepower",
       y = "Frequency") # Adds title and labels
```



The results show that the horsepower distribution tends to be skewed to the right, in which the majority of engines have horsepower ranging from 70 to 130. The number of engines with higher horsepower from 150 and above decreases significantly. This shows that the majority of vehicles in the data are of moderate power, suitable for general use.

3.2 Task 2: Horsepower Distribution by Cylinders and Engine Size

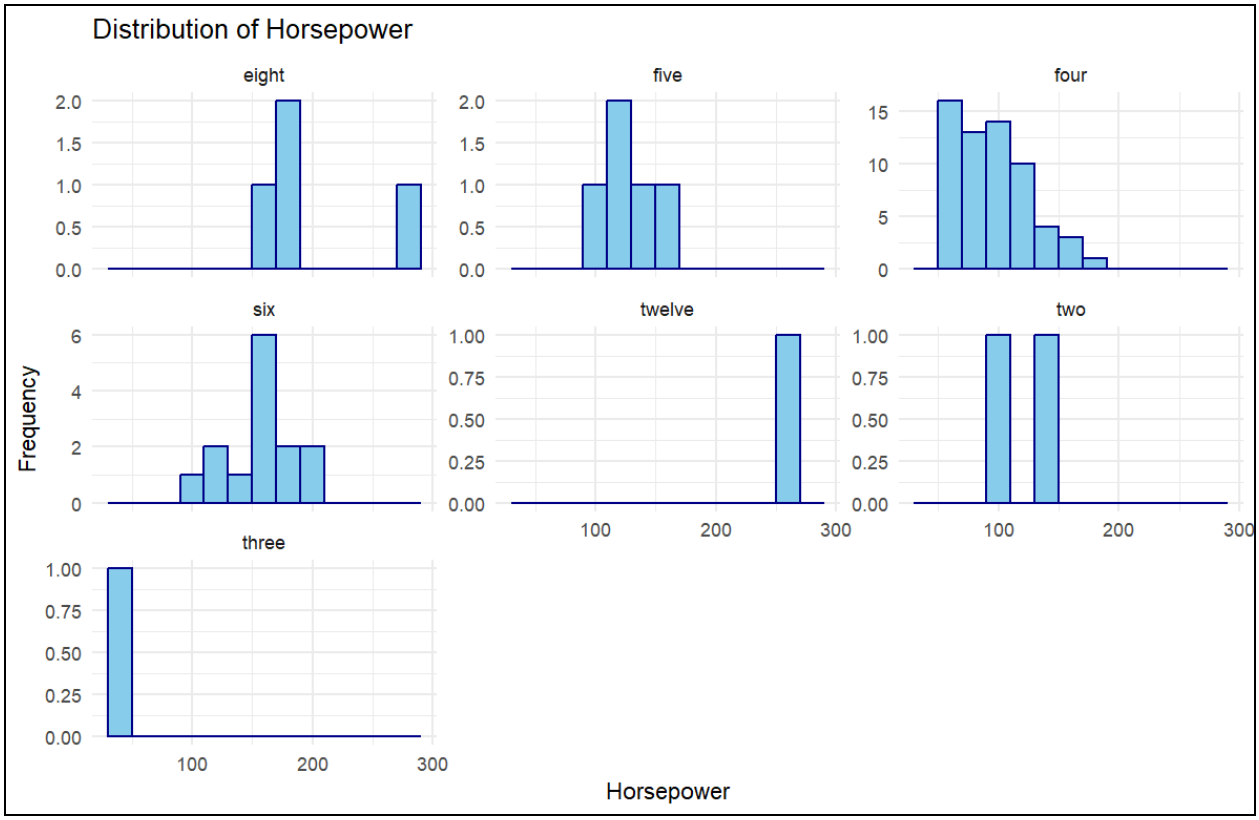
Task 2 explores the relationship between `Horsepower` and two key engineering factors: Number of cylinders and Engine Size. This analysis helps determine how different engine groups affect vehicle performance, through the distribution of horsepower. Histograms are used to visually represent trends within each group.

A) Horsepower Distribution by Cylinder Count

Initially, the `NumCylinders` variable is converted to a factor to group the data by number of cylinders. The histogram then uses `facet_wrap()` to break into separate groups for each cylinder type (2, 3, 4, 5, 6, 8, 12). Each group shows the corresponding horsepower distribution, allowing for quick comparisons between engine groups.

```
# Step 1:Convert to factor to ensure correct grouping in plots
engine$NumCylinders <- as.factor(engine$NumCylinders)

# Step 2:Plot the histogram
ggplot(engine, aes(x = Horsepower)) +
  geom_histogram(binwidth = 20, fill = "skyblue", color =  "darkblue") +
   # Creates a histogram with bin width of 20 and styled appearance
  facet_wrap(~ NumCylinders, scales = "free_y") +
   # Splits the histogram into subplots for each cylinder group using facet_wrap
  labs(title = "Distribution of Horsepower",
       x = "Horsepower",
       y = "Frequency") +  # Adds title and labels
  theme_minimal()
   # Apply a clean, minimal theme for better readability
```



The histogram shows that 4-cylinder and 6-cylinder engines dominate the data. The horsepower distribution in the 4-cylinder group is concentrated mainly in the 60–120 horsepower range, while the 6-cylinder group is more spread out and has more cars with higher horsepower. In contrast, the 2, 3, and 12-cylinder groups appear less frequently, indicating that these are less common engine types. In general, the larger the number of cylinders, the higher the horsepower capacity, reflecting the technical nature of the engine.

B) Horsepower Distribution by Engine Size Group

In this section, before grouping `EngineSize`, the `str(engine)` command is used to check the data type of the `EngineSize` and `Horsepower` variables. The result shows that both are numeric or integer, so no data type conversion is required.

Next, the `EngineSize` column is divided into 4 groups according to common displacement levels: 60–100, 101–200, 201–300, and over 300. The grouping is done using the `case_when()` function, and saved to a new column named `EngineSizeGroup`. This variable is then converted to factor type with a

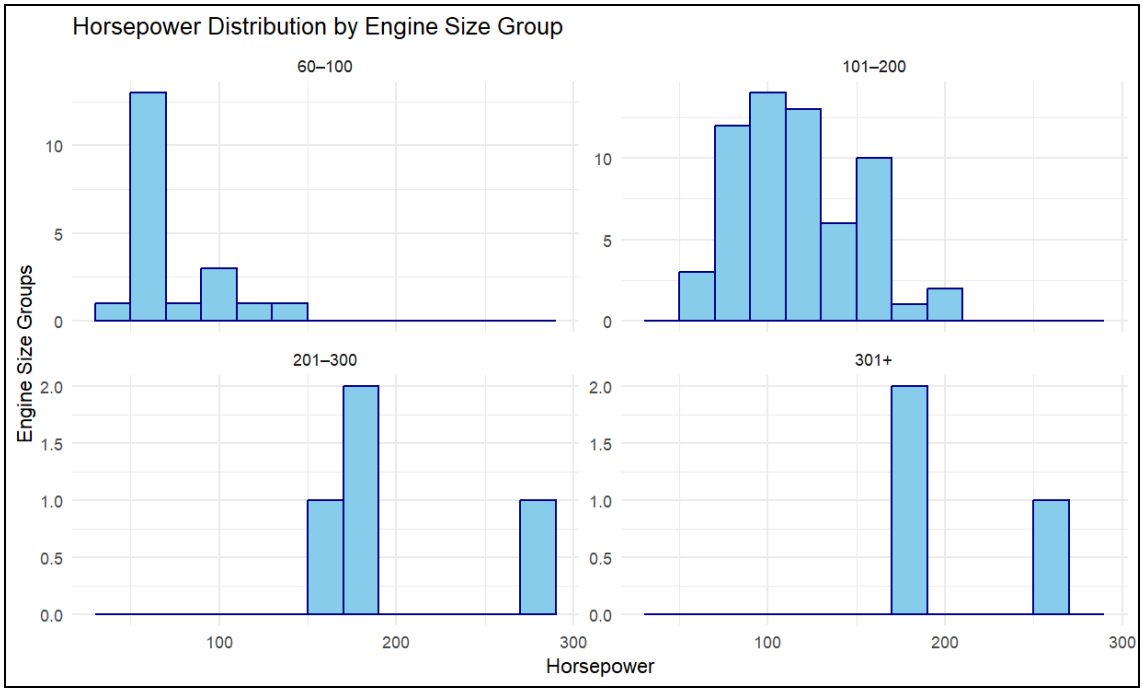
specific order to ensure correct display in the chart. The histogram is then created similar to the previous section, with each engine displacement group displayed separately.

```
#Step 1:Checking data type to ensure later grouping data.
str(engine)

#Step 2:Group EngineSize into 4 categorical size bands
engine <- engine %>%
  mutate(EngineSizeGroup = case_when(
    EngineSize >= 60 & EngineSize <= 100 ~ "60–100",
    EngineSize > 100 & EngineSize <= 200 ~ "101–200",
    EngineSize > 200 & EngineSize <= 300 ~ "201–300",
    EngineSize > 300 ~ "301+"
  ))

# Step 3:Convert EngineSizeGroup to factor for ordering in plots
engine$EngineSizeGroup <- factor(engine$EngineSizeGroup,
                          levels = c("60–100", "101–200", "201–300", "301+"))

# Step 4:Plot histogram by EngineSizeGroup
ggplot(engine, aes(x = Horsepower)) +
  geom_histogram(binwidth = 10, fill = "skyblue", color = "darkblue") +
   # Creates a histogram with bin width of 10 and styled appearance
  facet_wrap(~ EngineSizeGroup, scales = "free_y") +
   # Splits the histogram into subplots for each cylinder group using facet_wrap
  labs(title = "Horsepower Distribution by Engine Size Group",
       x = "Horsepower",
       y = "Engine Size Groups")   # Adds title and labels
  theme_minimal()
   # Apply a clean, minimal theme for better readability
```



The results from the chart show that the 60–100 displacement group has horsepower concentrated mainly below 100, reflecting the power limit of small engines. Meanwhile, the 101–200 and 201–300 groups show a clear increase in horsepower and have a wider distribution. This shows that these are popular engine groups in mid-range and high-end vehicles. In particular, the group above 300, although less common, has the highest horsepower. This is often found in sports cars or heavy trucks.

Overall, this analysis clearly shows a familiar technical relationship: larger displacement engines are often accompanied by higher power output, and also reflects the diversity in design and application of engine types in practice.

2.3 Task 3: Analyze engine troubles and compare by fuel type

Task 3 focuses mainly on identifying common engine troubles in the data and comparing these faults between two types of vehicles: gasoline and diesel. The goal is to find the differences so that third parties can have an overview and easily provide appropriate maintenance directions for each type of fuel.

A) Filter only trouble engines

In the first step, the data is filtered using the `filter()` function to retain only vehicles having a problem or is suspected of having a problem. An error code of 1 indicates an engine-related error, while -1

indicates a problem with another component. Removing vehicles without errors helps to focus the analysis on the group that needs maintenance. The `table()` command is used to double-check the results, ensuring that only two types of error codes remain, 1 and -1.

```
# Filter to only trouble cases (engine or other component failure)
trouble_cases <- maintenance %>%
  filter(ErrorCodes != "0")
# ErrorCode = 1 (engine fails) or -1 (component fails)

#Output checking
table(trouble_cases$ErrorCodes)
# Confirm all ErrorCodes in filtered data are non-zero
# Should only show 1 and -1
```

```
> table(trouble_cases$ErrorCodes)

 -1   0   1
171   0 191
```

B) Top 5 Most Common Troubles (All Engines)

After filtering the data, the next step is to combine the maintenance table with the vehicle table and the engine table using the `inner_join()` function to get more information about the fuel type and engine code. Next, `group_by()` and `summarise()` were used to count the frequency of each error in the Troubles column. After that, the data were sorted by frequency descending using `desc()` and using `head(5)` to get the 5 most common errors.

```
# Step 1: Merge trouble_cases with automobile and engine data to get FuelTypes and
EngineModel information
trouble_cases <- trouble_cases %>%
  inner_join(automobile, by = "PlateNumber") %>%
   # Combine the automobile and troble_cases data using PlateNumber as a key.
  inner_join(engine, by = "EngineModel")
   # Combine that with engine data using EngineModel as a key.

# Step 2: View top 5 most common engine troubles
top_troubles <- trouble_cases %>%
# create new variables
  group_by(Troubles) %>%
   # grouping data into same trouble type
  summarise(Frequency = n()) %>%
```

```
   # determine the times each type of trouble appear and store in Frequency variable
  arrange(desc(Frequency)) %>%
   # sort the Frequency data from most common to least common.
  head(5)
   # Pick out the first 5 row

#Output checking
print(top_troubles)

#Result
```

	Troubles	Frequency
	<chr>	<int>
1	Cylinders	39
2	Chassis	25
3	Ignition (finding)	23
4	Noise (finding)	20
5	Loss of driving ability	16

The results showed that the most common troubles were related to Cylinders, Chassis, and Ignition. These faults can all have a major impact on operating performance. Therefore, early identification helps prioritize inspection of vulnerable components.

C) Compare Troubles by Fuel Type

The final part of Task 3 is to compare troubles by fuel type. The data is divided into two groups: gasoline and diesel vehicles. For each group, the top 5 most common troubles are listed as in the previous step. Then, these two dataframe are combined to compare each type of trouble between the two groups of vehicles.

```
# Step 1: Get top 5 troubles for Gas engines
top_gas_troubles <- trouble_cases %>%
# create new variables
  filter(FuelTypes == "gas") %>%
   # pick out only gas engines
  group_by(Troubles) %>%
   # grouping data into same trouble type
  summarise(Frequency = n()) %>%
   # determine the times each type of trouble appear and store in Frequency variable
  arrange(desc(Frequency)) %>%
  # sort the Frequency data from most common to least common.
  head(5)
  # Pick out the first 5 row
```

```r
#Output checking
print(top_gas_troubles)
```

	Troubles	Frequency
	<chr>	<int>
1	Cylinders	36
2	Ignition (finding)	22
3	Chassis	21
4	Noise (finding)	19
5	Loss of driving ability	16

```r
# Step 2: Get top 5 troubles for Diesel engines
top_diesel_troubles <- trouble_cases %>%
# create new variables
  filter(FuelTypes == "diesel") %>%
   # pick out only gas engines
  group_by(Troubles) %>%
   # grouping data into same trouble type
  summarise(Frequency = n()) %>%
   # determine the times each type of trouble appear and store in Frequency variable
  arrange(desc(Frequency)) %>%
  # sort the Frequency data from most common to least common.
  head(5)
  # Pick out the first 5 row

#Output checking
print(top_diesel_troubles)
```

	Troubles	Frequency
	<chr>	<int>
1	Chassis	4
2	Cam shaft	3
3	Cylinders	3
4	Crank shaft	2
5	Steering wheel	2

```r
#Step 3: Combine into one table for comparison
top_troubles_combined <- full_join(
  top_diesel_troubles %>%
    rename(Diesel_Frequency = Frequency),
    # rename Frequency column to Diesel_Frequency for clarity
  top_gas_troubles %>%
    rename(Gas_Frequency = Frequency),
    # rename Frequency column to Gas_Frequency for clarity
  by = "Troubles"
  # Join both tables by the common 'Troubles' column
)

print(top_troubles_combined) # Display results
```

Table of top 5 troubles by fuel types

	Troubles	Diesel_Frequency	Gas_Frequency
	<chr>	<int>	<int>
1	Chassis	4	21
2	Cam shaft	3	NA
3	Cylinders	3	36
4	Crank shaft	2	NA
5	Steering wheel	2	NA
6	Ignition (finding)	NA	22
7	Noise (finding)	NA	19
8	Loss of driving ability	NA	16

The results show that gasoline vehicles are more likely to have a wider range of problems, especially those related to ignition, noise, and drivability. This may reflect the greater complexity and sensitivity of components in modern gasoline engines, especially electronics and sensors. Diesel vehicles, on the other hand, are more likely to have mechanical or structural problems, such as chassis, camshafts, crankshafts, and steering wheels. These problems may stem from the general usage characteristics of diesel vehicles. Typically, they are used to carry heavy loads. This causes the engine to endure large torque and operate continuously, thus easily leading to damage compared to gasoline cars (Barsh Auto Service, 2024).

Notably, cylinder-related problems were the most common type of problem in the entire data set, especially in gasoline vehicles, with up to 36 cases. This suggests that cylinders are important and susceptible to wear and tear, regardless of engine type.

In short, each type of fuel tends to have different types of failures, so maintenance strategies also need to be adjusted accordingly to the characteristics of each vehicle group.

2.4 Task 4: Analyzing Maintenance Methods by Engine Error and Fuel Type

Task 4 aims to investigate what factors in the data might influence the maintenance method chosen (e.g. Adjustment, Replacement, or Urgent Care). Among the many factors, this section focuses on two key variables, `FuelType` and `ErrorCodes`, to see if these groups tend to be maintained differently.

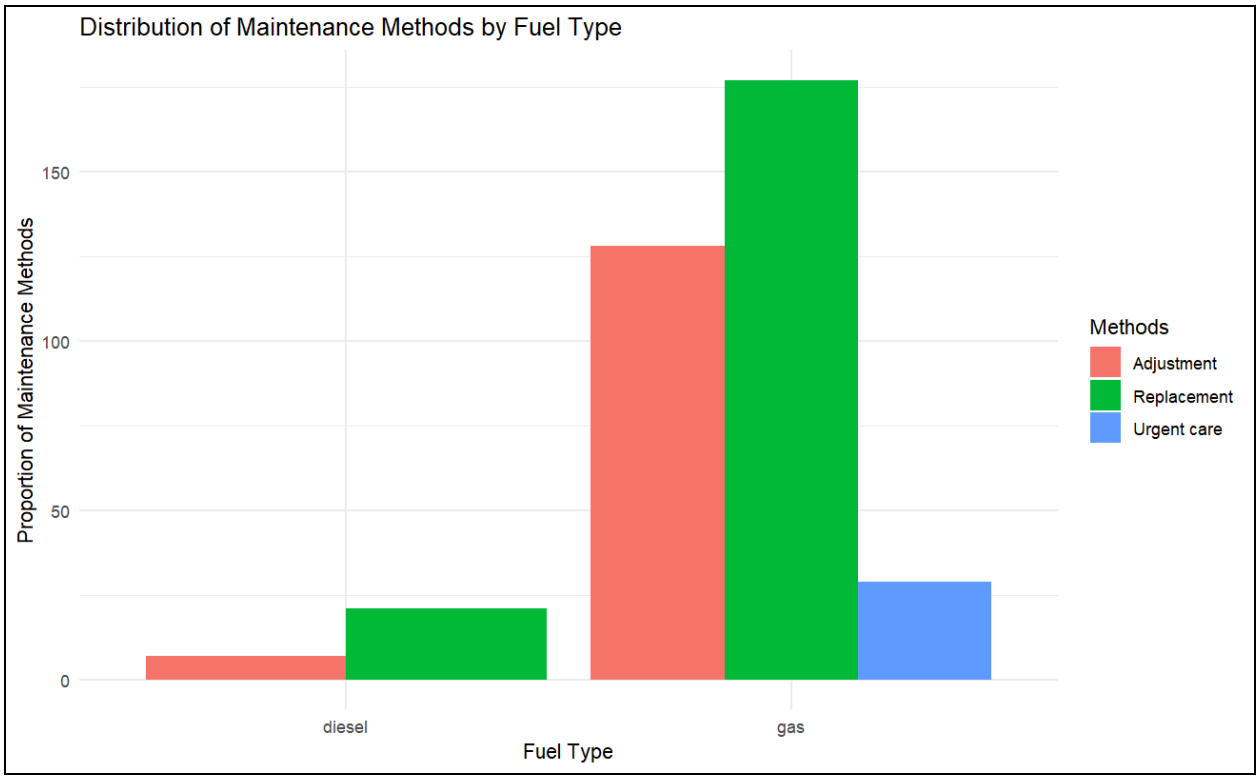
A) Maintenance Method by Fuel Type Analysis

At first, the data from the `Maintenance` dataframe is filtered to only retain vehicles with errors, that is rows with a non-zero `ErrorCode` value. This table is then joined to the `automobile` and `engine` dataframe to obtain additional information about the fuel type and engine model of each vehicle. The data is then grouped by two variables: ErrorCodes and Methods, which counts the number of times each maintenance method was applied to each type of error code. Then, the results are presented in a bar chart, allowing for a clear comparison between the two error groups.

```
# Step 1: Filter only vehicles with confirmed or suspected troubles
trouble_vehicles <- maintenance %>%
  filter(ErrorCodes != 0)

# Step 2: Join trouble_vehicles with maintenance and engine data to include
FuelTypes
trouble_vehicles <- trouble_vehicles %>%
  left_join(automobile, by = "PlateNumber")  %>%
  left_join(engine, by = "EngineModel")

# Step 3: Visualize Maintenance Methods by Fuel Type
ggplot(trouble_cases_extended, aes(x = FuelTypes, fill = Methods)) +
  geom_bar(position = "dodge") +
     # Create a stacked bar chart where each bar represents a fuel type
     # 'fill' scales the height of each segment to show proportion (not count)
  labs(
    title = "Distribution of Maintenance Methods by Fuel Type",
    x = "Fuel Type",
    y = "Proportion of Maintenance Methods"
  ) + # Add a clear title, label the x-axis & y-axis to explain what the chart shows
  theme_minimal()
  # Apply a clean, minimal theme for better readability
```



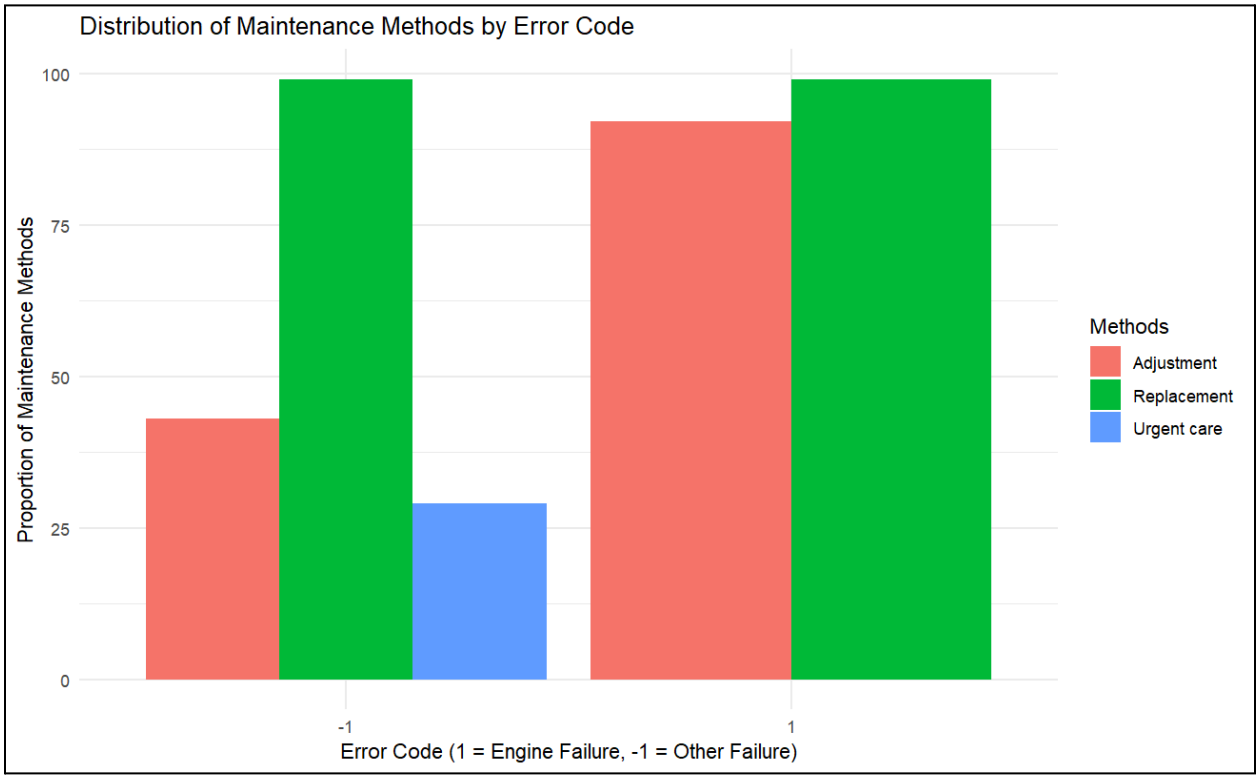
Results from the chart show that gasoline cars often have methods such as "Adjustment" and "Replacement" applied, while diesel cars have a significantly higher rate of using the "Replacement" method. This may reflect the fact that diesel vehicles are often used for heavier transport purposes, which causes more serious damage and requires component replacement rather than simple adjustment.

B) Maintenance Method by ErrorCode Analysis

This section analyzes whether the type of error, engine-related error (ErrorCode = 1) or other component error (ErrorCode = -1), has any impact on the maintenance method. The pre-filtered dataset is grouped by ErrorCodes and Methods, and a bar chart is then created for visual comparison.

```
# Visualize Maintenance Methods by ErrorCode
ggplot(trouble_cases_extended, aes(x = ErrorCode, fill = Methods)) +
  geom_bar(position = "dodge") +
     # Create a stacked bar chart where each bar represents a fuel type
     # 'fill' scales the height of each segment to show proportion (not count)
  labs(
    title = "Distribution of Maintenance Methods by Error Code",
    x = "Error Code (1 = Engine Failure, -1 = Other Failure)",
```

```
  y = "Proportion of Maintenance Methods"
) + # Add a clear title, label the x-axis & y-axis to explain what the chart shows
theme_minimal()
# Apply a clean, minimal theme for better readability
```



The results show that for vehicles with engine troubles, the most commonly used method is “Urgent care” or “Replacement”. This is reasonable because engine errors are often serious and directly affect vehicle performance. On the contrary, vehicles with failures in other parts such as the steering system and chassis often only need “Adjustment” maintenance, which means slight adjustments or tweaks to fix the problem.

From the above two analyses, it can be concluded that both fuel type and trouble type have a clear relationship with the maintenance method applied. This is very useful for building separate maintenance procedures suitable for each vehicle type and fault condition, contributing to optimizing repair time and cost.

Reference

Barsh Auto Service. (2024, April 26). *What Are The Differences Between Diesel And Gas Engines?* Barshauto. https://www.barshauto.com/what-are-the-differences-between-diesel-and-gas-engines

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R.,. . . Yutani, H. (2019). Welcome to the Tidyverse. *The Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686