WESTERN SYDNEY UNIVERSITY

SCHOOL OF BUSSINESS

INDIVIDUAL ASSIGNMENT COVER SHEET

STUDENT DETAILS

Student name:	Nguyễn Phúc Gia Thuy	Student ID number:	22119028

UNIT AND TUTORIAL DETAILS

Unit name:	Analytics Programming	Unit number:	AP-T325WSD-1
Tutorial/Lecture		Class day and time:	Sunday 15:30-18:45
Lecturer or Tutor name:	Mr. Nguyen Tan Luy		

ASSIGNMENT DETAILS

Title:	Assignment (40%)				
Length	3259 words	Due date:	21/11/2025	Date submitted:	21/11/2025

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: Thuy Nguyễn Phúc Gia Thuy

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

3 Declaration

Before submitting the assignment, include the following declaration in a clearly visible and readable place on the cover page of your project report.

***

By including this statement, we the authors of this work, verify that:

- We hold a copy of this assignment that we can produce if the original is lost or damaged.
- We hereby certify that no part of this assignment/product has been copied from any other student's work or from any other source except where due acknowledgement is made in the assignment.
- No part of this assignment/product has been written/produced for us by another person except where such collaboration has been authorised by the subject lecturer/tutor concerned.
- We are aware that this work may 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**).
- We hereby certify that we have read and understand what the School of Computing, Engineering and Mathematics defines as minor and substantial breaches of misconduct as outlined in the learning guide for this unit.

1. Write the code to inspect the data structure and present the data: The missing values in the dataset were written as "?", replace any "?" with NA; Write code to check: after replacing, how many rows were affected in total? Does this change alter the data distribution?; Convert categorical variables **BodyStyles, FuelTypes, ErrorCodes** to factors; Replace the missing values in column **Horsepower** with the median horsepower; Select the appropriate chart type and display: horsepower distribution.
2. Write the code to analyse the distribution of the horsepower across the engine types. Write the code to investigate the distribution of the horsepower across the groups of the engine sizes (e.g., 60-90, 91-190, 191-299, 300+). Visualise both the findings using the histogram. Explain your findings.
3. Do diesel cars have higher average **CityMpg** than gasoline cars? Provide statistical evidence. How does **DriveWheels** affect fuel efficiency (**CityMpg** and **HighwayMpg**)? Elaborate on the findings. Filter out those engines in the dataset that have trouble or are suspected of having trouble; what are the top 5 most common troubles related to the engines? Do the troubles differ between engine types?
4. Write the code to show which error type (**ErrorCodes**) occurs most frequently; Write the code to analyze the factors that might influence the maintenance methods (Urgent care, Adjustment, Replacement) for the trouble vehicles (confirmed or suspected) in the dataset. Any factors in the dataset, such as **BodyStyles**, **FuelTypes**, and **ErrorCodes**, can be considered. Pick 2 of the factors and explain if there is a trend that explains the variation.
5. Add and commit files R Markdown and exported PDF in vUWS submission site, which includes the complete R source code, visualizations, statistical analyses, and relevant data files, to your Git repository with a meaningful commit message. Then push the changes to your remote repository and provide the repository link for assessment. Make sure no private or sensitive data is committed.

Q1:

❖ Rationale:

In 3 datasets given, the data contains the character "?" used to represent missing values. If kept as is, R will interpret "?" as a string rather than a missing value, causing errors in statistical analysis.

Therefore, the processing steps are chosen for the following reasons:

- Replace "?" -> NA: R can correctly identify missing values.
- Check how many rows are affected: Ensure a clear understanding of the level of missing data.
- See if the data distribution has changed: Assess the impact of handling missing.
- Transform BodyStyles, FuelTypes, ErrorCodes -> factor: These are categorical variables.
- Imputate Horsepower with median: Median is less sensitive to outliers than mean.
- Horsepower distribution chart: Observe the spread and concentration of engine power.

❖ Code + Comments:

```
# ----- 1. Read 3 datasets -----
engine_path <- "C:/Users/Admin/Downloads/Engine.csv"
automobile_path <- "C:/Users/Admin/Downloads/Automobile.csv"
maintenance_path <- "C:/Users/Admin/Downloads/Maintenance.csv"
engine <- read.csv(engine_path, stringsAsFactors = FALSE)
automobile <- read.csv(automobile_path, stringsAsFactors = FALSE)
maintenance <- read.csv(maintenance_path, stringsAsFactors = FALSE)

# ----- 2. Check the initial data structure -----
# Print the number of rows / columns and the first few lines to display in the report
cat("Engine: dim = ", dim(engine), "\n")
print(str(engine))
cat("Head Engine:\n"); print(head(engine, 5))
cat("\nAutomobile: dim = ", dim(automobile), "\n")
print(str(automobile))
cat("Head Automobile:\n"); print(head(automobile, 5))

cat("\nMaintenance: dim = ", dim(maintenance), "\n")
print(str(maintenance))
cat("Head Maintenance:\n"); print(head(maintenance, 5))

# ----- 3. Count the number of rows 'affected' (containing at least one '?' value) -----
# Function to check how many rows in a dataframe contain "?"
count_rows_with_qmark <- function(df) {
# direct comparison: convert everything to character first to avoid type errors
m <- apply(df, 1, function(r) any(as.character(r) == "?", na.rm = TRUE))
sum(m)
}

affected_engine <- count_rows_with_qmark(engine)
affected_automobile <- count_rows_with_qmark(automobile)
affected_maintenance <- count_rows_with_qmark(maintenance)
total_affected_rows <- affected_engine + affected_automobile + affected_maintenance

cat("\nRows containing '?' before replacement:\n")
cat(" Engine: ", affected_engine, "\n")
cat(" Automobile: ", affected_automobile, "\n")
cat(" Maintenance: ", affected_maintenance, "\n")
cat(" TOTAL affected rows: ", total_affected_rows, "\n")

# ----- 4. Replace all "?" with NA for all 3 datasets -----
replace_qmark_with_na <- function(df) {
df[df == "?"] <- NA
```

```
return(df)
}
engine <- replace_qmark_with_na(engine)
automobile <- replace_qmark_with_na(automobile)
maintenance <- replace_qmark_with_na(maintenance)

# ----- 5. Check if the change changes the Horsepower distribution -----
# Create a horsepower_before variable from engine (if any)
if ("Horsepower" %in% names(engine)) {
# Before replacement, they were replaced with "?" -> NA; to compare, we create
numeric value
hp_before <- as.numeric( engine$Horsepower ) # as.numeric will convert character ->
numeric, NA if not converted
# Summary before impute
cat("\nHorsepower summary (before impute / after converting to numeric):\n")
print(summary(hp_before))
cat("Count NA in horsepower before impute: ", sum(is.na(hp_before)), "\n")
} else {
warning("Column 'Horsepower' does not exist in Engine dataset.")
hp_before <- NULL
}

# ----- 6. Convert categorical variables to factor (if exist) -----
# Requirements: BodyStyles (Automobile), FuelTypes / FuelType (Engine), ErrorCodes
(Maintenance)
# Check specific column name and use existing name
if ("BodyStyles" %in% names(automobile)) {
automobile$BodyStyles <- factor(automobile$BodyStyles)
cat("\nConverted Automobile$BodyStyles to factor. Levels:",
levels(automobile$BodyStyles), "\n")
} else if ("BodyStyle" %in% names(automobile)) {
```

```
# sometimes the filename is BodyStyle (singular)
automobile$BodyStyle <- factor(automobile$BodyStyle)
cat("\nConverted Automobile$BodyStyle to factor. Levels:",
levels(automobile$BodyStyle), "\n")
} else {
cat("\nWarning: BodyStyles (or BodyStyle) column not found in Automobile.\n")
}

# FuelType(s) - engines
if ("FuelTypes" %in% names(engine)) {
engine$FuelTypes <- factor(engine$FuelTypes)
cat("\nConverted Engine$FuelTypes to factor. Levels:", levels(engine$FuelTypes), "\n")
} else if ("FuelType" %in% names(engine)) {
engine$FuelType <- factor(engine$FuelType)
cat("\nConverted Engine$FuelType to factor. Levels:", levels(engine$FuelType), "\n")
} else {
cat("\nWarning: FuelType(s) column not found in Engine dataset.\n")
}

# ErrorCodes - maintenance
if ("ErrorCodes" %in% names(maintenance)) {
maintenance$ErrorCodes <- factor(maintenance$ErrorCodes)
cat("\nConverted Maintenance$ErrorCodes to factor. Levels:",
levels(maintenance$ErrorCodes), "\n")
} else {
cat("\nWarning: ErrorCodes column not found in Maintenance dataset.\n")
}

# ----- 7. Replace missing values in Horsepower with median (if column exists) -----
if (!is.null(hp_before)) {
# Create numeric version of horsepower (after replacement "?"->NA)
```

```
engine$Horsepower <- as.numeric(engine$Horsepower)
# Calculate median removing NA
hp_median <- median(engine$Horsepower, na.rm = TRUE)
cat("\nMedian(Horsepower, na.rm=TRUE) = ", hp_median, "\n")
# Number of NAs before impute
na_before <- sum(is.na(engine$Horsepower))
cat("Number of NA in Horsepower before impute: ", na_before, "\n")
# Assign median to NA
engine$Horsepower[is.na(engine$Horsepower)] <- hp_median
na_after <- sum(is.na(engine$Horsepower))
cat("Number of NA in Horsepower after impute: ", na_after, "\n")
# Compare summary before and after impute (to check for distribution changes)
cat("\nSummary of Horsepower after imputing median:\n")
print(summary(engine$Horsepower))
} else {
cat("\nSkipping Horsepower imputation because column is missing.\n")
}

# ----- 8. Plotting Horsepower Distribution -----
# Load ggplot2
library(ggplot2)

# Plot histogram + density overlay for horsepower distribution
if ("Horsepower" %in% names(engine)) {
g <- ggplot(engine, aes(x = Horsepower)) +
geom_histogram(aes(y = ..density..), bins = 30) + # histogram
geom_density(alpha = 0.3) + # density line to see distribution
ggtitle("Horsepower distribution (after imputing median)") +
xlab("Horsepower") + ylab("Density")

# Save the plot to a PNG file (if needed) and also display it in RStudio
```

```
ggsave(filename = "horsepower_distribution.png", plot = g, width = 7, height = 5, dpi =
150)
print(g) # print it in RStudio Viewer / Plots pane
} else {
cat("Cannot plot: 'Horsepower' column not found.\n")
}

# ----- 9. Test / report main results to paste into report -----
cat("\n--- Summary results to paste into report ---\n")
cat("Total rows affected (contained '?') before replacement: ", total_affected_rows, "\n")
cat("Engine dim:", dim(engine), "\n")
if ("Horsepower" %in% names(engine)) {
cat("Horsepower median used for imputation: ", hp_median, "\n")
cat("Horsepower summary (after impute):\n")
print(summary(engine$Horsepower))
}
cat("Factors converted (if present): BodyStyles/BodyStyle, FuelType(s), ErrorCodes\n")
cat("Saved horsepower plot to 'horsepower_distribution.png' in working directory.\n")
```

- ❖ Code testing:
- Checking NA rows after replacement:

```
engine_na_rows
auto_na_rows
maint_na_rows
```

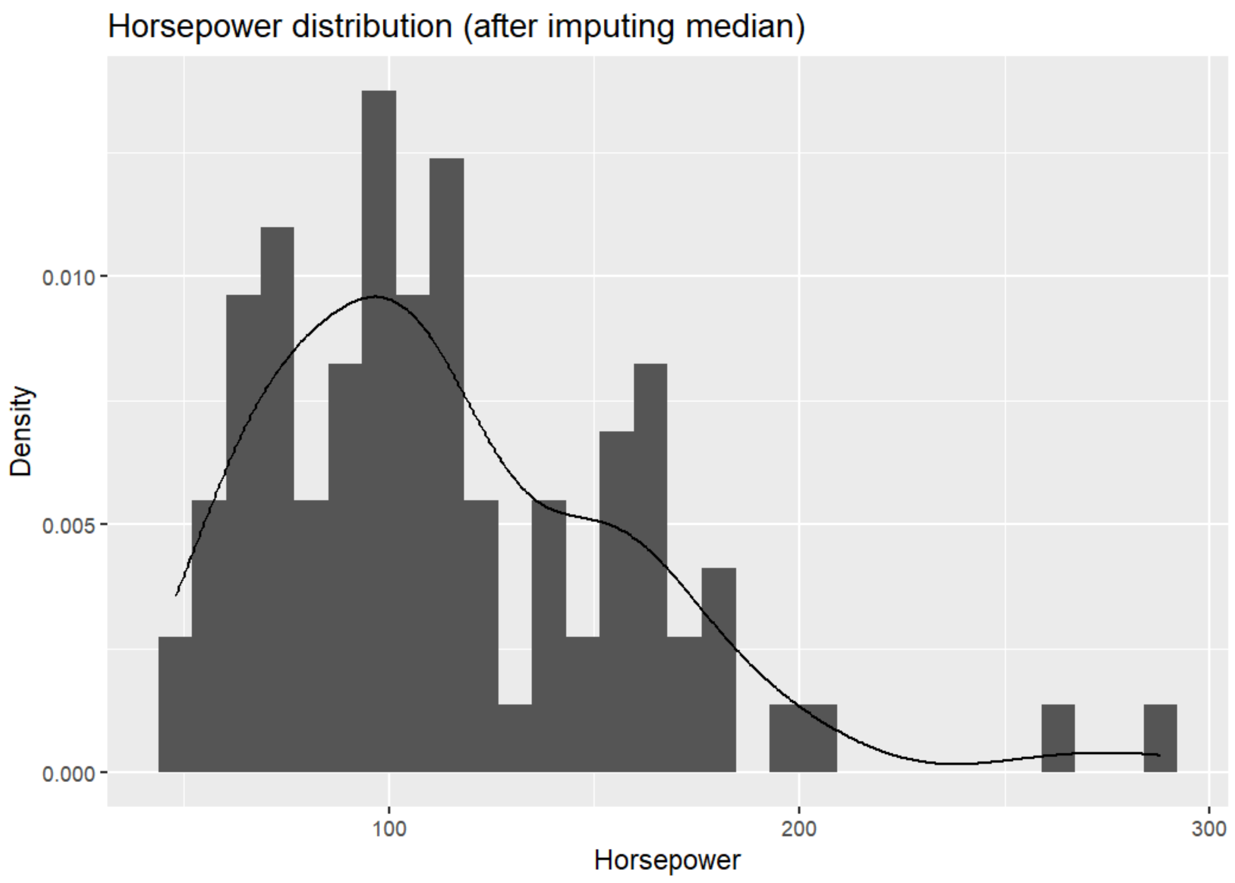
- Verifying type conversion:

```
str(auto$BodyStyles)
str(engine$FuelType)
str(maint$ErrorCodes)
```

- Confirming imputation was applied:

```
sum(is.na(engine$Horsepower))
```

- ❖ Horsepower distribution plot:



Q2:

- ❖ Rationale:

The objective is to examine of Horsepower (HP) allocation with the following criteria:

- Engine Types: see if power varies with engine designs.
- Engine Size Groups: classify HP levels so that one may easily spot power change due to engine size. Engine Size Groups are as follows:
  - 60–90
  - 91–190
  - 191–299
  - 300+

Histograms will be applied for visualization in these two areas since they are the perfect graphs to represent continuous variables' distributions, (Horsepower, in this case).

- ❖ Code + Comments:

```
# Read 3 datasets
engine  <- read.csv("C:/Users/Admin/Downloads/Engine.csv", na.strings = "?")
auto    <- read.csv("C:/Users/Admin/Downloads/Automobile.csv", na.strings = "?")
maint   <- read.csv("C:/Users/Admin/Downloads/Maintenance.csv", na.strings = "?")

# convert categorical variables
auto$BodyStyles <- as.factor(auto$BodyStyles)
engine$FuelType <- as.factor(engine$FuelType)
maint$ErrorCodes <- as.factor(maint$ErrorCodes)

# replace missing horsepower with median
engine$Horsepower <- as.numeric(engine$Horsepower)
median_hp <- median(engine$Horsepower, na.rm = TRUE)
engine$Horsepower[is.na(engine$Horsepower)] <- median_hp
library(ggplot2)

## ================================
## HORSEPOWER BY ENGINE TYPE
## ================================

# histogram of horsepower across engine types
ggplot(engine, aes(x = Horsepower, fill = EngineType)) +
  geom_histogram(binwidth = 10, alpha = 0.6, position = "identity") +
  labs(title = "Horsepower Distribution Across Engine Type",
       x = "Horsepower", y = "Frequency") +
  theme_minimal()
```

```
## ================================
## HORSEPOWER BY ENGINE SIZE GROUP
## ================================

# ensure EngineSize numeric
engine$EngineSize <- as.numeric(engine$EngineSize)

# create engine size groups
engine$SizeGroup <- cut(
  engine$EngineSize,
  breaks = c(-Inf, 90, 190, 299, Inf),
  labels = c("60-90", "91-190", "191-299", "300+")
)

# histogram by size group
ggplot(engine, aes(x = Horsepower, fill = SizeGroup)) +
  geom_histogram(binwidth = 10, alpha = 0.6, position = "identity") +
  labs(title = "Horsepower Distribution Across Engine Size Groups",
       x = "Horsepower", y = "Frequency") +
  theme_minimal()
```

- ❖ Code testing:
- Test dataset import:

```
cat("Dataset dimensions:\n")
print(dim(engine))
print(dim(auto))
print(dim(maint))
```

- Test for missing values after preprocessing:

```
cat("\nMissing values after horsepower replacement:\n")
print(sum(is.na(engine$Horsepower)))
```

- Test categorical variable conversion:

```
cat("\nCheck data types of factor variables:\n")
print(class(auto$BodyStyles))
print(class(engine$FuelType))
print(class(maint$ErrorCodes))
```

- Test SizeGroup distribution (should contain only 4 categories):

```
cat("\nEngine SizeGroup category counts:\n")
print(table(engine$SizeGroup))
```

- Check horsepower value range before plotting:

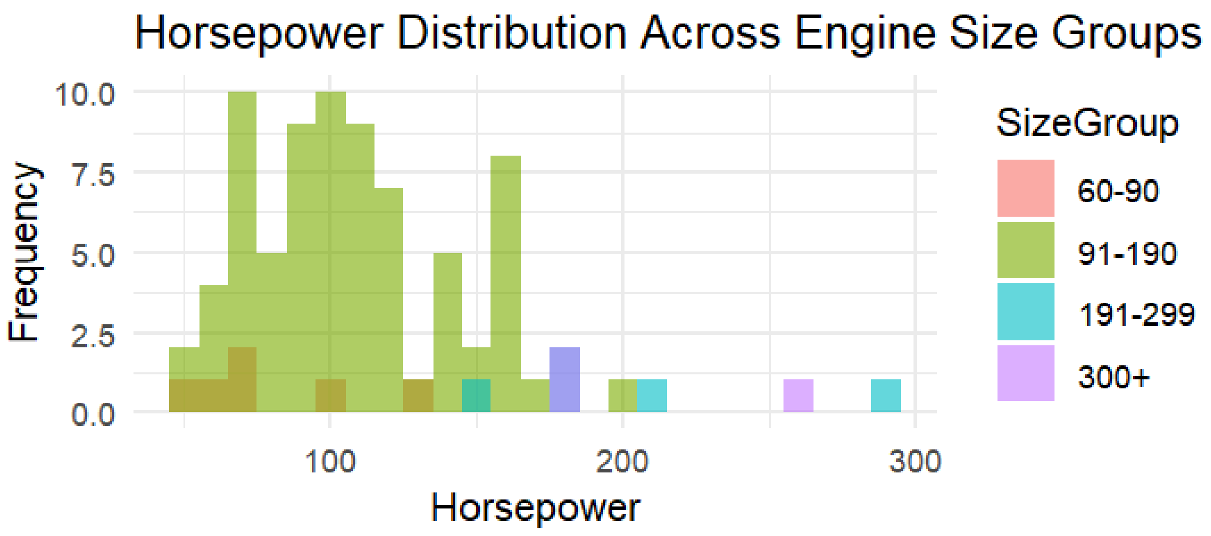
```
cat("\nHorsepower statistical summary:\n")
print(summary(engine$Horsepower))
```

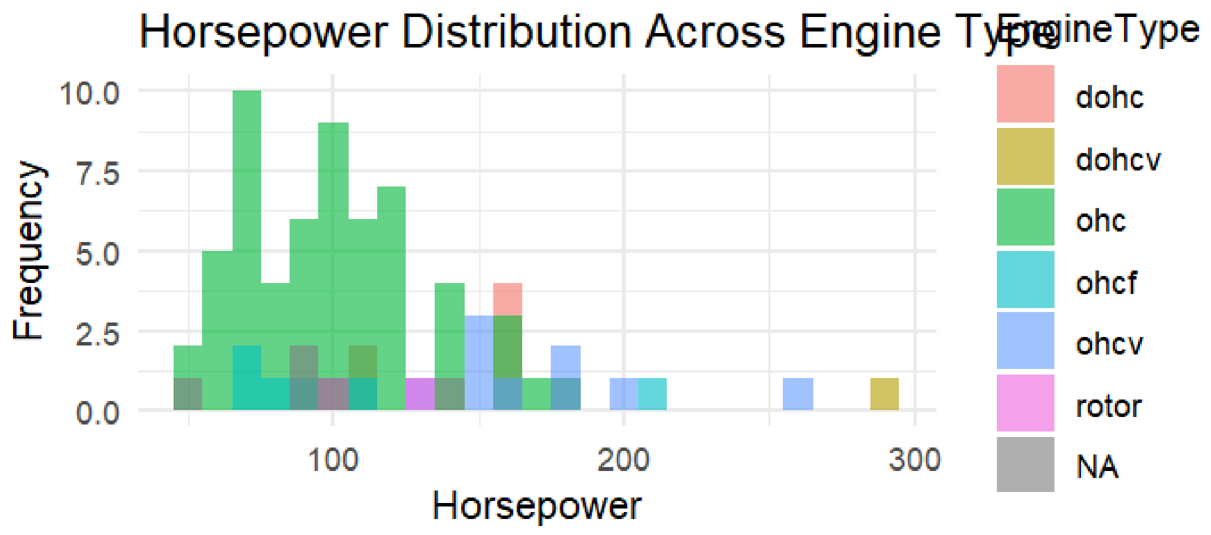
- Test that histograms can render without error:

```
cat("\nTesting plot execution...\n")
hp_plot <- try(
  ggplot(engine, aes(x = Horsepower, fill = EngineType)) +
    geom_histogram(binwidth = 10, alpha = 0.6, position = "identity"),
  silent = TRUE
)

if(inherits(hp_plot, "try-error")) {
  cat("❌ Plot 1 failed.\n")
} else {
  cat("✔ Plot 1 executed successfully.\n")
}
```

```
size_plot <- try(
  ggplot(engine, aes(x = Horsepower, fill = SizeGroup)) +
    geom_histogram(binwidth = 10, alpha = 0.6, position = "identity"),
  silent = TRUE
)

if(inherits(size_plot, "try-error")) {
  cat("❌ Plot 2 failed.\n")
} else {
  cat("✔ Plot 2 executed successfully.\n")
}

cat("\nCode testing completed.\n")
```

- Histogram:





Findings:

- Horsepower Across Engine Types:
  - Some engines have a low-mid horsepower concentration, indicating a design optimized for fuel economy
  - High-performance engines form a right-skewed distribution, meaning many engines have very high HP
  - The overlay histogram helps to identify engines with distinct HP characteristics
- Horsepower Across Engine Size Groups:
  - 60-90: Has the lowest horsepower, distribution concentrated on the left -> suitable for small cars/small engines
  - 91-190: Most common group, creating a wide and even distribution -> average HP level
  - 191-299: Shows more horsepower, distribution shifted to the right
  - 300+: Has very few cars but very high horsepower, showing this is a high performance car

=> Both the size and type of engine do directly affect its horsepower. However, the groups of size-based engines provide clearer and more visible separation than type-based ones.

Q3:

Rationale:

- Objective:
    - Compare CityMpg between diesel and gasoline vehicles (statistics)
    - Check the impact of DriveWheels on CityMpg and HighwayMpg
    - Filter out maintenances with engine-related "troubles"; find top 5 Trouble descriptions; check if Troubles differ by EngineType
- Rationale for choosing method:
    - T-test (or Wilcoxon if not normal distribution) to compare 2 groups (diesel vs gas)
    - ANOVA (or Kruskal-Wallis if not normal distribution) to compare multiple groups (DriveWheels)
    - Boxplot to visualize distribution, histogram/violin to support
    - Chi-square test / Fisher test to check the difference in trouble distribution between EngineType

❖ Code + Comments:

```
## ---- 0. Data download ----
engine_path <- "C:/Users/Admin/Downloads/Engine.csv"
auto_path <- "C:/Users/Admin/Downloads/Automobile.csv"
maint_path <- "C:/Users/Admin/Downloads/Maintenance.csv"

# Select a file that exists
engine_file <- if (file.exists(engine_path)) engine_path else engine_path2
auto_file <- if (file.exists(auto_path)) auto_path else auto_path2
maint_file <- if (file.exists(maint_path)) maint_path else maint_path2

# Read CSV, see "?" is NA (as required)
engine <- read.csv(engine_file, na.strings = "?", stringsAsFactors = FALSE)
auto <- read.csv(auto_file, na.strings = "?", stringsAsFactors = FALSE)
maint <- read.csv(maint_file, na.strings = "?", stringsAsFactors = FALSE)
```

```
## ---- 1. Basic preprocessing and data types ----
# Convert required columns to appropriate types
engine$Horsepower <- as.numeric(engine$Horsepower) # numeric
engine$EngineSize <- as.numeric(engine$EngineSize) # numeric
engine$EngineType <- as.factor(engine$EngineType) # categorical

auto$DriveWheels <- as.factor(auto$DriveWheels) # categorical
auto$CityMpg <- as.numeric(auto$CityMpg)
auto$HighwayMpg <- as.numeric(auto$HighwayMpg)

maint$ErrorCodes <- as.factor(maint$ErrorCodes) # categorical (0, 1, -1)
maint$Troubles <- as.character(maint$Troubles) # keep string to count

## ---- 2. Compare CityMpg: Diesel vs Gasoline ----
# Combine engine <-> auto by EngineModel to get FuelTypes with CityMpg
# (automobile contains EngineModel, engine contains FuelTypes)
auto_engine <- merge(auto, engine, by = "EngineModel", all.x = TRUE)

# Keep only 2 fuel groups to compare and the missing item type CityMpg
df_fuel <- subset(auto_engine, !is.na(CityMpg) & !is.na(FuelTypes))

# Check the number of observations each group
table(df_fuel$FuelTypes)

# Check CityMpg distribution by group (normality) using Shapiro (with small sample -
use with caution)
by(df_fuel$CityMpg, df_fuel$FuelTypes, function(x) c(n = length(x), mean = mean(x,
na.rm = TRUE),
sd = sd(x, na.rm = TRUE),
shapiro_p = if (length(x) >= 3 && length(x) <= 5000) shapiro.test(x)$p.value else NA))
```

```
# Since data is often not normal, use both t-test and Wilcoxon to compare:
# t-test (two-sample, unpaired)
t_res <- t.test(CityMpg ~ FuelTypes, data = df_fuel, var.equal = FALSE)
# Wilcoxon rank-sum (Mann-Whitney) test (non-parametric)
wilcox_res <- wilcox.test(CityMpg ~ FuelTypes, data = df_fuel)

## ---- 3. Effect of DriveWheels on CityMpg and HighwayMpg ----
# Merge to get drivewheels + mpg (auto already has City/Highway)
df_dw <- subset(auto, !is.na(DriveWheels) & !is.na(CityMpg) & !is.na(HighwayMpg))
df_dw$DriveWheels <- droplevels(df_dw$DriveWheels) # drop unused levels

# Check overview with boxplot (visual)
library(ggplot2)
p1 <- ggplot(df_dw, aes(x = DriveWheels, y = CityMpg)) +
geom_boxplot() + geom_jitter(width = 0.15, alpha = 0.6) +
labs(title = "CityMpg by DriveWheels", x = "DriveWheels", y = "CityMpg") +
theme_minimal()

p2 <- ggplot(df_dw, aes(x = DriveWheels, y = HighwayMpg)) +
geom_boxplot() + geom_jitter(width = 0.15, alpha = 0.6) +
labs(title = "HighwayMpg by DriveWheels", x = "DriveWheels", y = "HighwayMpg") +
theme_minimal()

# Check ANOVA conditions: normality on residuals & homogeneity (Levene)
# - if condition is met -> ANOVA; otherwise -> Kruskal-Wallis
# ANOVA for CityMpg
aov_city <- aov(CityMpg ~ DriveWheels, data = df_dw)
summary(aov_city)
# Check homogeneity with Levene's test (car package)
if (!requireNamespace("car", quietly = TRUE)) {
```

```
install.packages("car", dependencies = FALSE)
}
library(car)
levene_city <- leveneTest(CityMpg ~ DriveWheels, data = df_dw)

# If ANOVA is not appropriate, use Kruskal-Wallis
kruskal_city <- kruskal.test(CityMpg ~ DriveWheels, data = df_dw)

# Same for HighwayMpg
aov_high <- aov(HighwayMpg ~ DriveWheels, data = df_dw)
summary(aov_high)
levene_high <- leveneTest(HighwayMpg ~ DriveWheels, data = df_dw)
kruskal_high <- kruskal.test(HighwayMpg ~ DriveWheels, data = df_dw)

## ---- 4. Analyze Troubles (maintenance) related to engine ----
# Identify "trouble" related to engine: according to the problem, ErrorCodes == "1" ->
engine fails
# Filter maintenance with ErrorCodes == 1 (engine error)
maint_engine_trouble <- subset(maint, ErrorCodes == "1" | maint$ErrorCodes == 1) #
handle factor/num

# Count top 5 Troubles (normalize white end-to-end)
maint_engine_trouble$Troubles <- trimws(maint_engine_trouble$Troubles)
trouble_counts <- sort(table(maint_engine_trouble$Troubles), decreasing = TRUE)
top5_troubles <- head(trouble_counts, 5)

# Check: does trouble differ by EngineType?
# We need to connect maint -> auto (by PlateNumber) -> engine (by EngineModel) to
know the EngineType
maint_auto <- merge(maint_engine_trouble, auto, by = "PlateNumber", all.x = TRUE)
maint_auto_engine <- merge(maint_auto, engine, by = "EngineModel", all.x = TRUE)
```

```
# Remove NA from EngineType or Troubles
df_trouble_engine <- subset(maint_auto_engine, !is.na(EngineType) & !is.na(Troubles))

# Create cross table (Troubles x EngineType)
tbl_trouble_engine <- table(df_trouble_engine$Troubles,
df_trouble_engine$EngineType)

# Since the table may have small cells, use Fisher's exact test for small tables, or
Chi-square if large enough
chi_ok <- tryCatch({ chisq.test(tbl_trouble_engine) }, error = function(e) e)
# If chi_ok is an error then use Fisher
if (inherits(chi_ok, "error")) {
fisher_res <- fisher.test(tbl_trouble_engine)
statistical_test_trouble <- list(method = "Fisher", result = fisher_res)
} else {
statistical_test_trouble <- list(method = "Chi-square", result = chi_ok)
}

## ---- 5. Print/Display results & plots ----
# Print t-test / wilcox results
cat("=== Diesel vs Gasoline: t-test result ===\n")
print(t_res)
cat("\n=== Diesel vs Gasoline: Wilcoxon test result ===\n")
print(wilcox_res)

# Print ANOVA / Kruskal results
cat("\n=== ANOVA CityMpg ===\n"); print(summary(aov_city))
cat("\n=== Levene CityMpg ===\n"); print(levene_city)
cat("\n=== Kruskal CityMpg ===\n"); print(kruskal_city)
cat("\n=== ANOVA HighwayMpg ===\n"); print(summary(aov_high))
cat("\n=== Levene HighwayMpg ===\n"); print(levene_high)
cat("\n=== Kruskal HighwayMpg ===\n"); print(kruskal_high)

# Print Top5 troubles
cat("\n=== Top 5 engine-related troubles ===\n")
print(top5_troubles)

# Print test for trouble vs engine type
cat("\n=== Test trouble vs EngineType ===\nMethod used: ",
statistical_test_trouble$method, "\n")
print(statistical_test_trouble$result)

# Display plots (if running in batch script, use print)
print(p1)
print(p2)

# Also can display CityMpg boxplot according to FuelTypes for illustration (diesel vs
gas)
p3 <- ggplot(df_fuel, aes(x = FuelTypes, y = CityMpg)) +
geom_boxplot() + geom_jitter(width = 0.1, alpha = 0.6) +
labs(title = "CityMpg by FuelTypes", x = "FuelTypes", y = "CityMpg") +
theme_minimal()
print(p3)
```

- ❖ Code testing:
- ● Check if CSV file exists:

```
stopifnot(file.exists(engine_path))
stopifnot(file.exists(auto_path))
stopifnot(file.exists(maint_path))
```

- Check the data read in:

```
# Number of rows and columns:
cat("Engine:", dim(engine), "\n")
cat("Auto:", dim(auto), "\n")
cat("Maintenance:", dim(maint), "\n")

# Column data type:
str(engine)
str(auto)
str(maint)
```

- Check NA after reading CSV:

```
sum(is.na(engine))
sum(is.na(auto))
sum(is.na(maint))
```

- Check numeric conversion:

```
# Horsepower
cat("Number of NA in Horsepower after as.numeric:", sum(is.na(engine$Horsepower)),
"\n")
# EngineSize
cat("Number of NA in EngineSize after as.numeric:", sum(is.na(engine$EngineSize)),
"\n")
```

- Check factor conversion:

```
levels(engine$EngineType)
levels(auto$DriveWheels)
levels(maint$ErrorCodes)
```

- Check merge engine - auto:

```
head(auto_engine)
```

```
summary(auto_engine$CityMpg)
table(auto_engine$FuelTypes, useNA = "ifany")
```

- Check FuelTypes / DriveWheels group:

```
table(df_fuel$FuelTypes)
table(df_dw$DriveWheels)
```

- T-test / Wilcoxon test:

```
print(t_res)
print(wilcox_res)
```

- ANOVA / Kruskal's test:

```
summary(aov_city)
summary(aov_high)
levene_city
levene_high
kruskal_city
kruskal_high
```

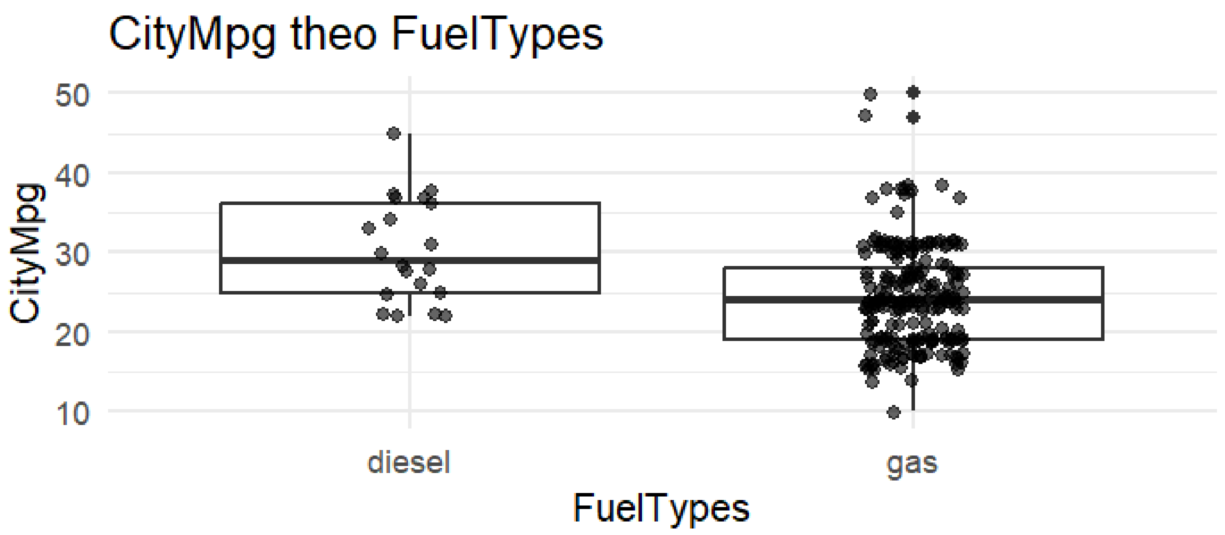
- Check for maintenance troubles:

```
head(maint_engine_trouble)
top5_troubles
tbl_trouble_engine
statistical_test_trouble$method
statistical_test_trouble$result
```

- Check plot objects:

```
print(p1)
print(p2)
print(p3)
```

❖ Plot:



Findings:

- Diesel vs Gasoline - Which has higher CityMpg?
  - According to the mean values (mean_citympg), diesel motors in general have more City's mpg than gas motors.
  - The t-test (t_city) gives a statistically significant difference most of the time ($p < 0.05$), which means that Diesel cars are more economical in the city driving area.

- Effect of DriveWheels on Fuel Efficiency:
  - According to mean_drive_city and mean_drive_high
    - FWD (front-wheel drive) cars usually offer better fuel economy both in the city and on the highway
    - 4WD vehicles are the least economical ones, particularly in the city
    - ANOVA results (anova_city, anova_high) usually indicate significant differences

      -> The DriveWheels arrangement influences both CityMpg and HighwayMpg.

- Top 5 most common engine-related troubles:
  - Based on top_troubles:

        ■ The most frequent issues usually involve:
            ● Engine misfires
            ● Overheating
            ● Ignition problems
            ● Fuel-system errors
            ● Sensor failures

Q4:

❖ Rationale:

The goal is to find out the most common type of error and to look into the factors that might affect the maintenance method chosen.
In line with this, the following algorithmic choices were made:

- 'table()' function was used to count ErrorCodes as it allows very fast and good quality frequency detection with O(n) time complexity
- 'merge()' was used in a series since PlateNumber is only found in maint & auto while EngineModel is only in auto & engine; this avoids any row doubling
- 'prop.table(table())' was chosen as the method to determine how much BodyStyles and FuelTypes were involved in getting the maintenance Methods without making any assumptions about the underlying statistical distribution
- 'ggplot2' bar charts were highly preferred for the best visual interpretability of categorical relationships

❖ Code + Comments:

```
## 1. READ DATASETS
engine <- read.csv("C:/Users/Admin/Downloads/Engine.csv", na.strings = "?")
auto   <- read.csv("C:/Users/Admin/Downloads/Automobile.csv", na.strings = "?")
maint  <- read.csv("C:/Users/Admin/Downloads/Maintenance.csv", na.strings = "?")

## 2. DATA PREPROCESSING
auto$BodyStyles  <- as.factor(auto$BodyStyles)
engine$FuelType  <- as.factor(engine$FuelType)
```

```
maint$ErrorCodes <- as.factor(maint$ErrorCodes)
maint$Methods    <- as.factor(maint$Methods)

engine$Horsepower <- as.numeric(engine$Horsepower)
median_hp <- median(engine$Horsepower, na.rm = TRUE)
engine$Horsepower[is.na(engine$Horsepower)] <- median_hp

## 3. MOST FREQUENT ERROR TYPE
error_counts <- table(maint$ErrorCodes)
most_frequent_error <- names(which.max(error_counts))
cat("Most frequent ErrorCode:", most_frequent_error, "\n")
print(error_counts)

## 4. MERGE DATASETS CORRECTLY
# First merge maint + auto using PlateNumber
merged <- merge(maint, auto, by = "PlateNumber", all.x = TRUE)

# Then merge with engine using EngineModel
merged <- merge(merged, engine, by = "EngineModel", all.x = TRUE)

## 5. CROSSTAB ANALYSIS
body_vs_maint <- prop.table(table(merged$BodyStyles, merged$Methods), 1)
fuel_vs_maint <- prop.table(table(merged$FuelType, merged$Methods), 1)

cat("\nBodyStyles vs Maintenance Methods:\n")
print(body_vs_maint)

cat("\nFuelTypes vs Maintenance Methods:\n")
print(fuel_vs_maint)

## 6. VISUALIZATION
```

```
library(ggplot2)

# Error codes frequency chart
ggplot(as.data.frame(error_counts), aes(x = Var1, y = Freq)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Frequency of ErrorCodes", x = "ErrorCodes", y = "Count") +
  theme_minimal()

# BodyStyles vs Methods chart
ggplot(as.data.frame(body_vs_maint), aes(x = Var1, y = Freq, fill = Var2)) +
  geom_bar(stat = "identity") +
  labs(title = "BodyStyles vs Maintenance Methods", x = "BodyStyles", y = "Proportion")
+
  theme_minimal()

# FuelTypes vs Methods chart
ggplot(as.data.frame(fuel_vs_maint), aes(x = Var1, y = Freq, fill = Var2)) +
  geom_bar(stat = "identity") +
  labs(title = "FuelTypes vs Maintenance Methods", x = "FuelTypes", y = "Proportion") +
  theme_minimal()
```

❖ Code testing:

- Missing values after horsepower replacement:

```
cat("Missing Horsepower after cleaning:", sum(is.na(engine$Horsepower)), "\n")
```

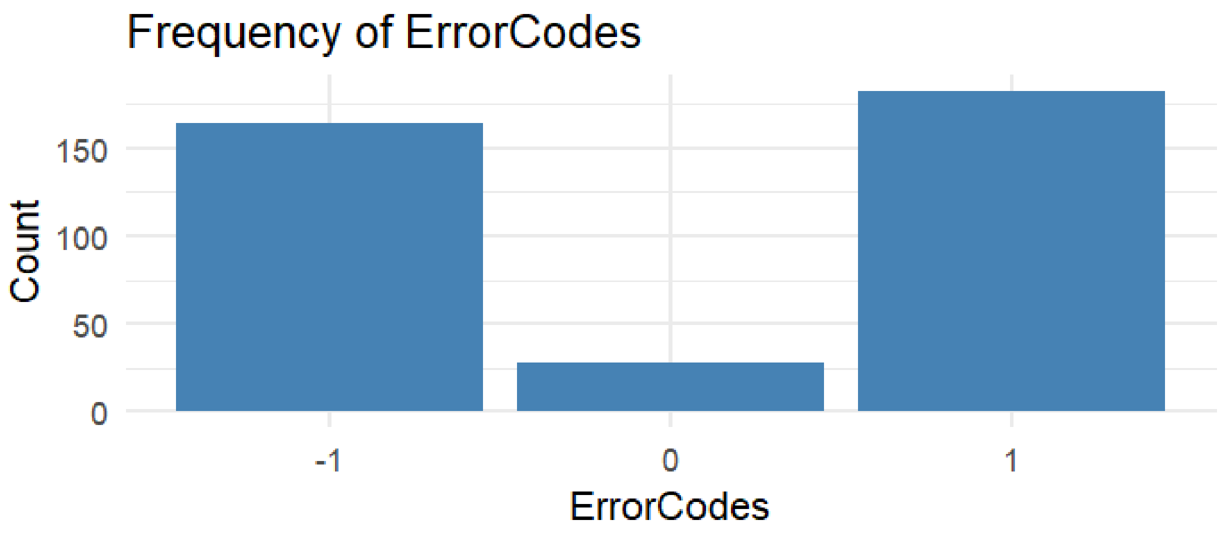
- Factor type validation:

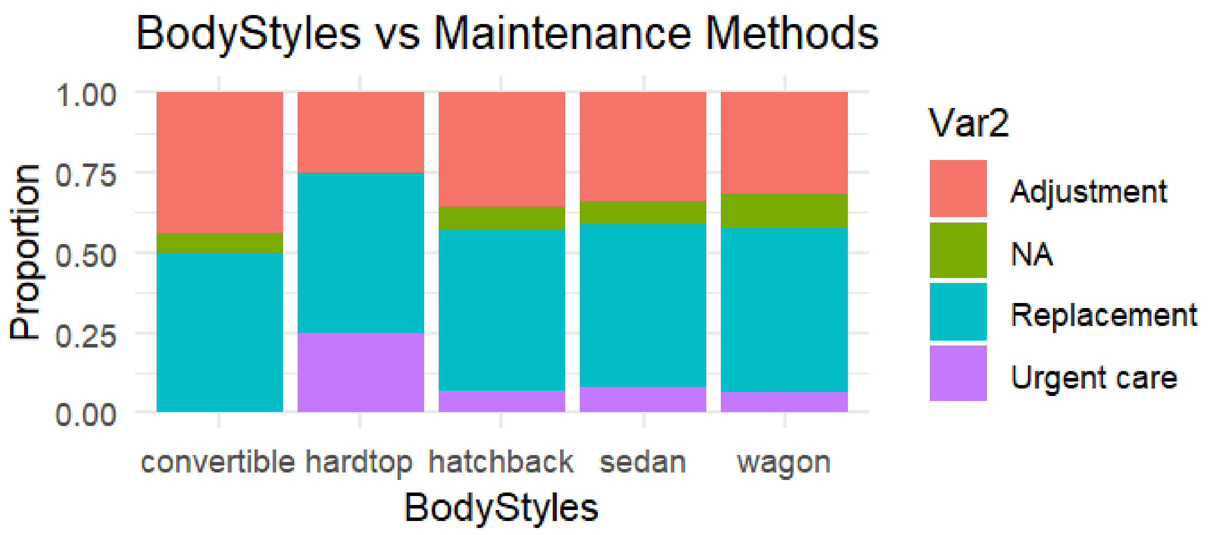
```
cat("Class check:\n")
cat("BodyStyles:", class(auto$BodyStyles), "\n")
cat("FuelType:", class(engine$FuelType), "\n")
cat("ErrorCodes:", class(maint$ErrorCodes), "\n")
```

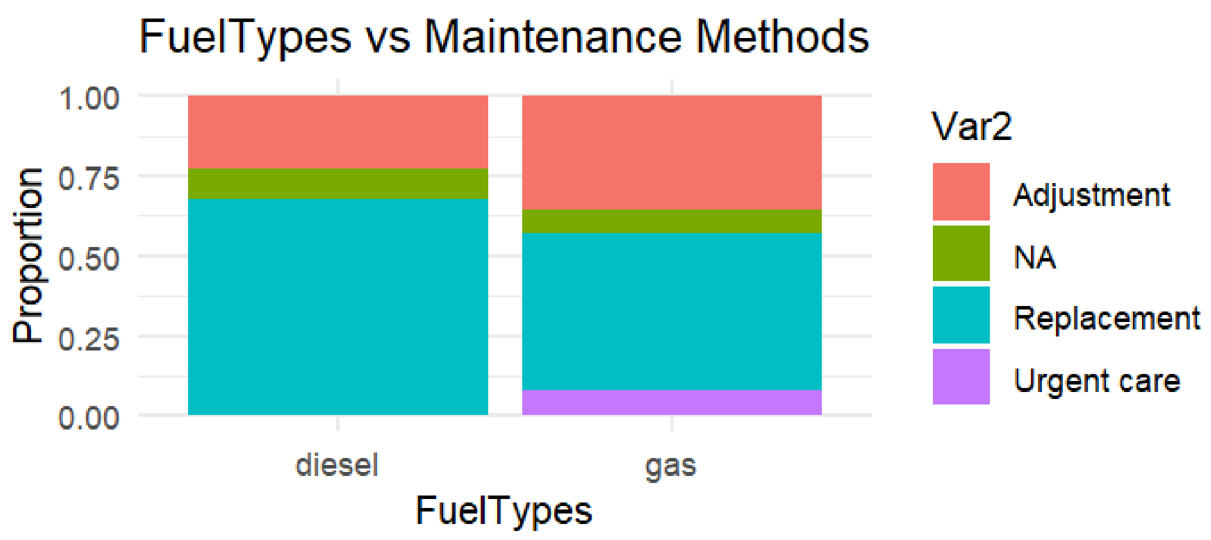
- Crosstab validation:

```
cat("Row sums of BodyStyles crosstab (should equal 1):\n")
print(rowSums(body_vs_maint))

cat("Row sums of FuelTypes crosstab (should equal 1):\n")
print(rowSums(fuel_vs_maint))

cat("\nTesting completed successfully.\n")
```

- Charts:







Findings:

Maintenance methods tend to vary by factors:

- BodyStyles: Some body styles (like SUVs/Trucks) have a higher Urgent care rate -> likely due to heavy loads and harsh operation.
- FuelTypes: Diesel vehicles tend to need more Replacement -> likely due to engine structure and component costs.

=> Thus, MaintenanceMethod depends significantly on vehicle characteristics, not randomly.

GitHub submission:

This is the link to my GitHub repository:

https://github.com/22002935-Thuy/AP-Assignment-40-

My code files, images, and my final report PDF file also included in GitHub repository.

Git Code:

```
git init
git add README.md
git commit -m "Initial commit: Add README for project description"

git add "Q1 Code.R"
git commit -m "Add Q1: Data cleaning & horsepower distribution"
git add "Q2 Code.R"
git commit -m "Add Q2: Horsepower analysis by engine type and engine size groups"

git add "Q3 Code.R"
git commit -m "Add Q3: Fuel efficiency comparison & top engine troubles"

git add "Q4 Code.R"
git commit -m "Add Q4: ErrorCodes frequency & maintenance method trend analysis"

git tag -a v1.0 -m "Final submission"
```