https://github.com/TrongNhan0508/AP-Assignment-Vietnam-T3-2025-WSU

SCHOOL of: Computer, Data and Mathematical Sciences

ASSIGNMENT COVER SHEET

STUDENT DETAILS

Name: Nguyen Le Trong Nhan	**Student ID:** 22201018

SUBJECT AND TUTORIAL DETAILS

Subject Name: Analytics Programming		**Subject code:** AP-T325WSD-1
Tutorial Group: none	**Day:** Sunday	**Time:** 15:30 – 18:45

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

ASSIGNMENT DETAILS

Title: Assignment T3 2025_3	**Length:**
Due Date: 21/11/2025	**Date submitted:** 21/11/2025
Home campus: Vietnam	

DECLARATION

By submitting your work using this link you are certifying 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.

Instructions*: Please complete the requested details in the form, save it and convert to PDF before adding your signature below*

Student signature: Nhan

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

AP_Assignment

Le Trong Nhan NGUYEN

2025-11-16

Q1

```r
# Load dataset
automobile_df = read.csv("Automobile.csv")
engine_df = read.csv("Engine.csv")
maintenance_df = read.csv("Maintenance.csv")
```

```r
# inspect data structure
str(automobile_df)
```

```
## 'data.frame':	204 obs. of  13 variables:
##  $ PlateNumber   : chr  "53N-001" "53N-002" "53N-003" "53N-004" ...
##  $ Manufactures  : chr  "Alfa-romero" "Alfa-romero" "Audi" "Audi" ...
##  $ BodyStyles    : chr  "convertible" "hatchback" "sedan" "sedan" ...
##  $ DriveWheels   : chr  "rwd" "rwd" "fwd" "4wd" ...
##  $ EngineLocation: chr  "front" "front" "front" "front" ...
##  $ WheelBase     : num  88.6 94.5 99.8 99.4 99.8 ...
##  $ Length        : num  169 171 177 177 177 ...
##  $ Width         : num  64.1 65.5 66.2 66.4 66.3 71.4 71.4 71.4 67.9 64.8 ...
##  $ Height        : num  48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 52 54.3 ...
##  $ CurbWeight    : int  2548 2823 2337 2824 2507 2844 2954 3086 3053 2395 ...
##  $ EngineModel   : chr  "E-0001" "E-0002" "E-0003" "E-0004" ...
##  $ CityMpg       : int  21 19 24 18 19 19 19 17 16 23 ...
##  $ HighwayMpg    : int  27 26 30 22 25 25 25 20 22 29 ...
```

```r
str(engine_df)
```

```
## 'data.frame':	88 obs. of  8 variables:
##  $ EngineModel : chr  "E-0001" "E-0002" "E-0003" "E-0004" ...
##  $ EngineType  : chr  "dohc" "ohcv" "ohc" "ohc" ...
##  $ NumCylinders: chr  "four" "six" "four" "five" ...
##  $ EngineSize  : int  130 152 109 136 136 131 131 108 164 164 ...
##  $ FuelSystem  : chr  "mpfi" "mpfi" "mpfi" "mpfi" ...
##  $ Horsepower  : chr  "111" "154" "102" "115" ...
##  $ FuelTypes   : chr  "gas" "gas" "gas" "gas" ...
##  $ Aspiration  : chr  "std" "std" "std" "std" ...
```

```r
str(maintenance_df)
```

```
## 'data.frame':    374 obs. of  7 variables:
##  $ ID         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ PlateNumber: chr  "53N-001" "53N-001" "53N-001" "53N-001" ...
##  $ Date       : chr  "15/02/2024" "16/03/2024" "15/04/2024" "15/05/2024" ...
##  $ Troubles   : chr  "Break system" "Transmission" "Suspected clutch" "Ignition (finding)" ...
##  $ ErrorCodes : int  -1 -1 -1 1 -1 1 1 0 -1 -1 ...
##  $ Price      : int  110 175 175 180 85 1000 180 0 180 180 ...
##  $ Methods    : chr  "Replacement" "Replacement" "Adjustment" "Adjustment" ...
```

1.1. Replace "?" with NA

```r
# Create a function that replaces "?" with NA inside a data frame
replace_qmark = function(df) {

  # lapply() applies a function to every column of the data frame
  df[] = lapply(df, function(col) {  # use [] to keep data frame not return as list

    # Proactive programming: Convert factor to character to handle "?"

    # By default, when reading a csv file, R encounter a column that contain character,
    # it often convert that character column in to factor column data type.

    # Do not convert numeric column into character type because if there is "?"
    # in numeric column, that column will be character when R read.
    if (is.factor(col)) {
      col = as.character(col)
    }

    # Replace "?" with NA
    col[col == "?"] <- NA

    # Return the cleaned column
    return(col)
  })

  return(df)
}

# Apply the funtion to all three datasets
engine_df = replace_qmark(engine_df)
automobile_df = replace_qmark(automobile_df)
maintenance_df = replace_qmark(maintenance_df)

# check if any "?" left using sapply to loop through every column and check for the string "?"
# The as.character() to treat column as character, ensure the check works on even numeric columns

# create function to check
check_qmark = function(col){

  # use any() to check if there is at least one true value ("?")
  any(as.character(col) == "?", na.rm = TRUE)
  # na.rm = TRUE: ignore the NA values created during the cleaning
}

# Check engine dataset
engine_df_check = sapply(engine_df, check_qmark)
engine_df_check
```

```
##  EngineModel   EngineType NumCylinders   EngineSize   FuelSystem   Horsepower
##        FALSE        FALSE        FALSE        FALSE        FALSE        FALSE
##    FuelTypes   Aspiration
##        FALSE        FALSE
```

```
# Check automobile dataset
automobile_df_check = sapply(automobile_df, check_qmark)
automobile_df_check
```

```
##    PlateNumber    Manufactures     BodyStyles    DriveWheels EngineLocation
##          FALSE           FALSE          FALSE          FALSE          FALSE
##      WheelBase          Length          Width         Height     CurbWeight
##          FALSE           FALSE          FALSE          FALSE          FALSE
##    EngineModel         CityMpg     HighwayMpg
##          FALSE           FALSE          FALSE
```

```
#check maintenance dataset
maintenance_df_check = sapply(maintenance_df, check_qmark)
maintenance_df_check
```

```
##          ID PlateNumber        Date    Troubles  ErrorCodes       Price
##       FALSE       FALSE       FALSE       FALSE       FALSE       FALSE
##     Methods
##       FALSE
```

As the output of the check of 3 dataset return FALSE, that means all "?" in datasets has been replace with NA

1.2. Count how many rows had "?" in original

```
# Load original data sets that have "?"

# Count the original "?" because we want to know how many rows were affected,
# if count NA maybe there are already NA in the dataset that is not in form of the string "?"
engine_df_ori = read.csv("Engine.csv")
automobile_df_ori = read.csv("Automobile.csv")
maintenance_df_ori = read.csv("Maintenance.csv")

# Count how many rows contain at least one "?" before cleaning
count_rows_with_qmark = function(df) {

  # apply() with MARGIN = 1 means apply across rows
  rows_with_q = apply(df, 1, function(row) {

    # Check if ANY value in the row equals "?"
```

```r
    any(row == "?", na.rm = TRUE)
  })

  # Sum tells how many TRUE rows we have
  sum(rows_with_q)
}

# Count for each dataset
rows_engines = count_rows_with_qmark(engine_df_ori)
rows_auto    = count_rows_with_qmark(automobile_df_ori)
rows_maint   = count_rows_with_qmark(maintenance_df_ori)

# Total rows affected
total_rows_affected = rows_engines + rows_auto + rows_maint
total_rows_affected
```

```
## [1] 6
```

1.3. Convert categorical variables BodyStyles, FuelTypes, ErrorCodes to factors

```r
# Convert BodyStyles to a factor
automobile_df$BodyStyles = factor(automobile_df$BodyStyles)

# Convert FuelType to a factor
engine_df$FuelType = factor(engine_df$FuelType)

# Convert ErrorCodes to a factor
maintenance_df$ErrorCodes = factor(maintenance_df$ErrorCodes)
```

1.4. replace mising value with median

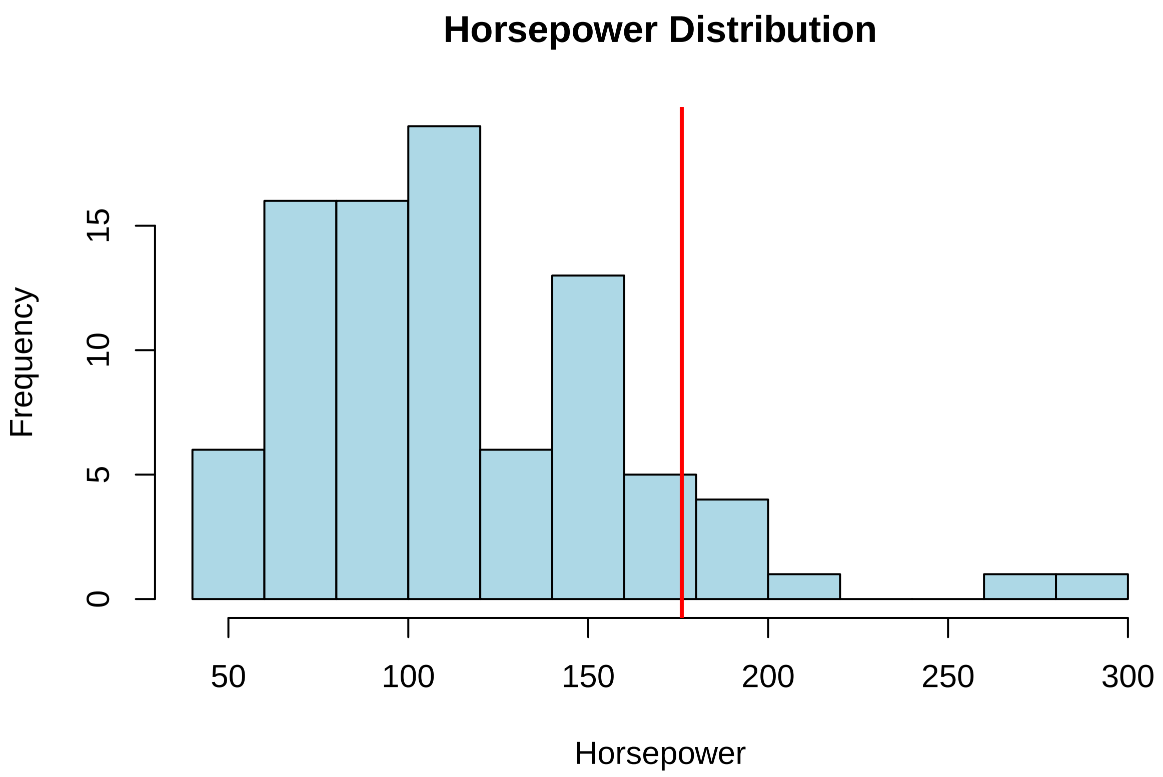
```r
# Calculate the median horsepower
median_hp = median(engine_df$Horsepower, na.rm = TRUE)
median_hp
```

```
## [1] "176"
```

```r
# Replace missing values with the median
engine_df$Horsepower[is.na(engine_df$Horsepower)] <- median_hp
# []: select only the elements in "engine_df$Horsepower" that is TRUE with the is.na()
```

1.5. plot horsepower distribution

```r
# Convert Horsepower to numeric
engine_df$Horsepower = as.numeric(engine_df$Horsepower)

# Histogram of horsepower
hist(engine_df$Horsepower,
     main = "Horsepower Distribution",
     xlab = "Horsepower",
     col = "lightblue",
     border = "black",
     breaks = 15)

# Add a vertical line for the median to show the measure of central tendency.
abline(v = median_hp,
       col = "red",
       lwd = 2)
```



The histogram shows that the distribution of horsepower is right-skewed, with most vehicles clustered between 80 and 150 horsepower. The median horsepower (indicated by the red vertical line) lies slightly to the right of the main concentration of observations, confirming the skewness. A small number of cars exhibit horsepower values above 200, forming a long right tail and indicating the presence of high-performance outliers. Overall, the distribution suggests that the majority of vehicles in the dataset have moderate engine power, while a smaller subset of more powerful engines contributes to the skewed shape. This non-normal distribution should be kept in mind when choosing statistical methods in later analyses.

Q2

```r
# 1. Have a quick look to 3 variables
# EngineType
str(engine_df$EngineType)
```

```
##  chr [1:88] "dohc" "ohcv" "ohc" "ohc" "ohc" "ohc" "ohc" "ohc" "ohc" "ohc" ...
```

```r
# HorsePower
str(engine_df$Horsepower)
```

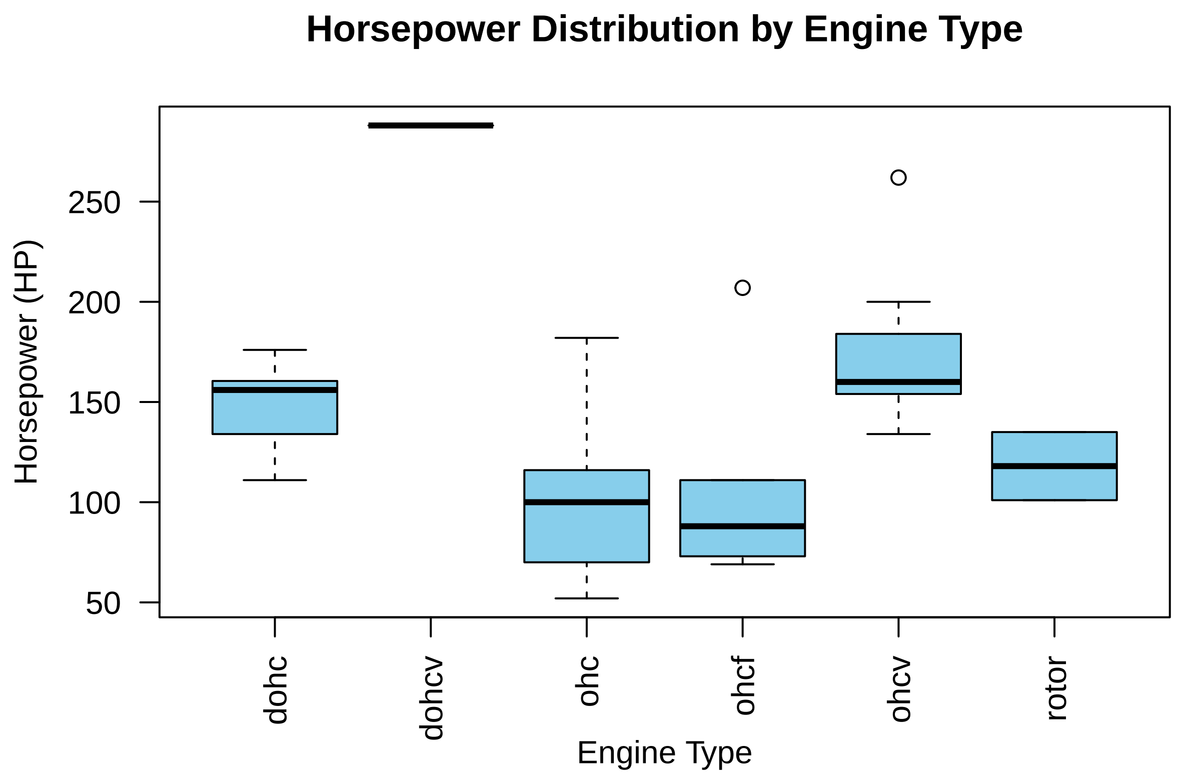
```
##  num [1:88] 111 154 102 115 110 140 160 101 121 121 ...
```

```r
# EngineSize
str(engine_df$EngineSize)
```

```
##  int [1:88] 130 152 109 136 136 131 131 108 164 164 ...
```

```r
# 2.
```

```r
# Boxplot is the most efficient way for comparing distributions across categories.
boxplot(engine_df$Horsepower ~ engine_df$EngineType,
        main = "Horsepower Distribution by Engine Type",
        xlab = "Engine Type",
        ylab = "Horsepower (HP)",
        col = "skyblue", #
        las = 2) # Rotate axis labels (las=2) for better readability
```



The boxplot comparing horsepower across different engine types reveals significant variations in power output, reflecting the design philosophies of each architecture.

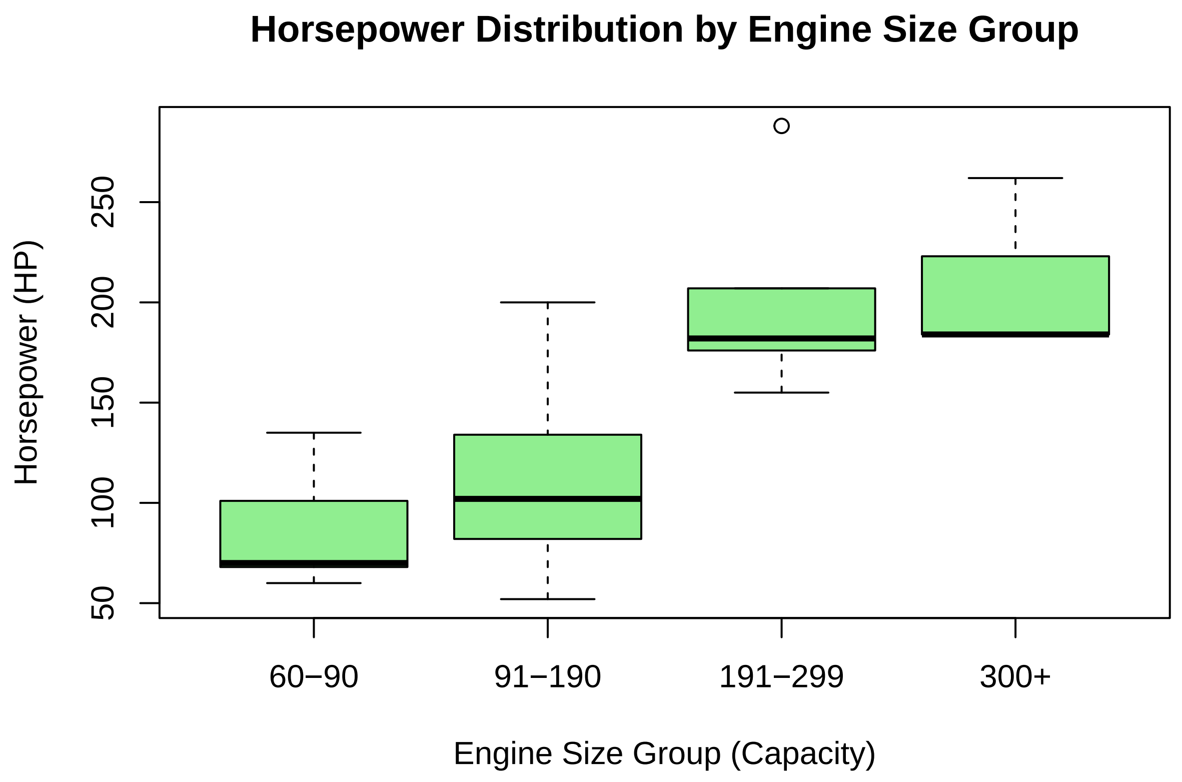
High Performance Types: The dohcv and ohcv types exhibit the highest median horsepower, demonstrating their use in performance and luxury vehicles. The$dohcv type shows the highest absolute range of power.

Standard Performance Types: The dohc type shows a strong mid-to-high median HP (around 155HP), which is higher and less variable than the simpler ohc type, highlighting the performance advantage of the dohc valvetrain design.

Variability: The ohc and ohcf groups show the lowest median horsepower and the largest interquartile range, indicating these designs are used across a wider spectrum of vehicles, from low-end economy to mid-range models.

Outliers: The presence of outliers in types like ohcv suggests specific, high-tuned engines within these general categories that push the power limits of the design.

```r
# Define the boundaries for the EngineSize bins (60-90, 91-190, 191-299, 300+).
# Inf (infinity) is used as the upper bound to capture all largest engines.
size_bins = c(60, 90, 190, 299, Inf)
size_labels = c("60-90", "91-190", "191-299", "300+")

# Create the new categorical factor 'EngineSizeGroup' in engine_df from the numeric
# integer factor 'EngineSize'

# Use function cut() to create bins.
engine_df$EngineSizeGroup = cut(engine_df$EngineSize,
                                breaks = size_bins,
                                labels = size_labels,
                                right = TRUE)
                                # right=TRUE means intervals include the upper boundary

# Visualize Horsepower distribution across the newly created EngineSizeGroup factor.
boxplot(engine_df$Horsepower ~ engine_df$EngineSizeGroup,
        main = "Horsepower Distribution by Engine Size Group",
        xlab = "Engine Size Group (Capacity)",
        ylab = "Horsepower (HP)",
        col = "lightgreen")
```



The boxplot comparing horsepower across categorized engine capacity groups clearly demonstrates a strong positive correlation between engine size and horsepower.

Group Progression: As the engine size group increases (from 60-90 to 300+), the median horsepower, the interquartile range (IQR), and the overall maximum horsepower systematically increase.

Key Findings:

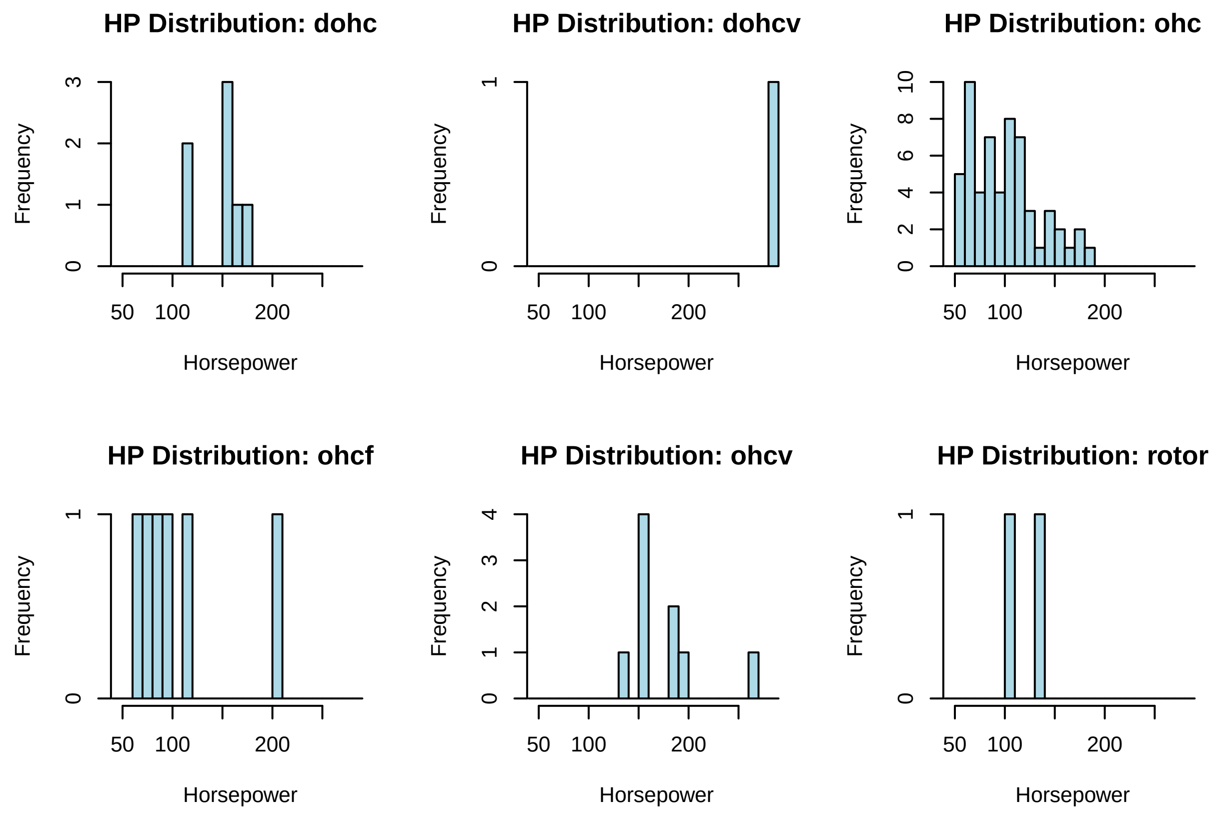
Low Capacity (60-90): This group is centered around the lowest median HP (approximately 70 HP), reflecting smaller, less powerful engines designed for efficiency.

High Capacity (300+): This group shows the highest median HP (approximately 190 HP) and the greatest range of high power, confirming that larger engines are primarily responsible for producing higher performance figures.

Consistency: The boxplots show that engine capacity is the most reliable predictor of horsepower, as the overlap between quartiles of adjacent groups is minimal, reinforcing the expected mechanical relationship.

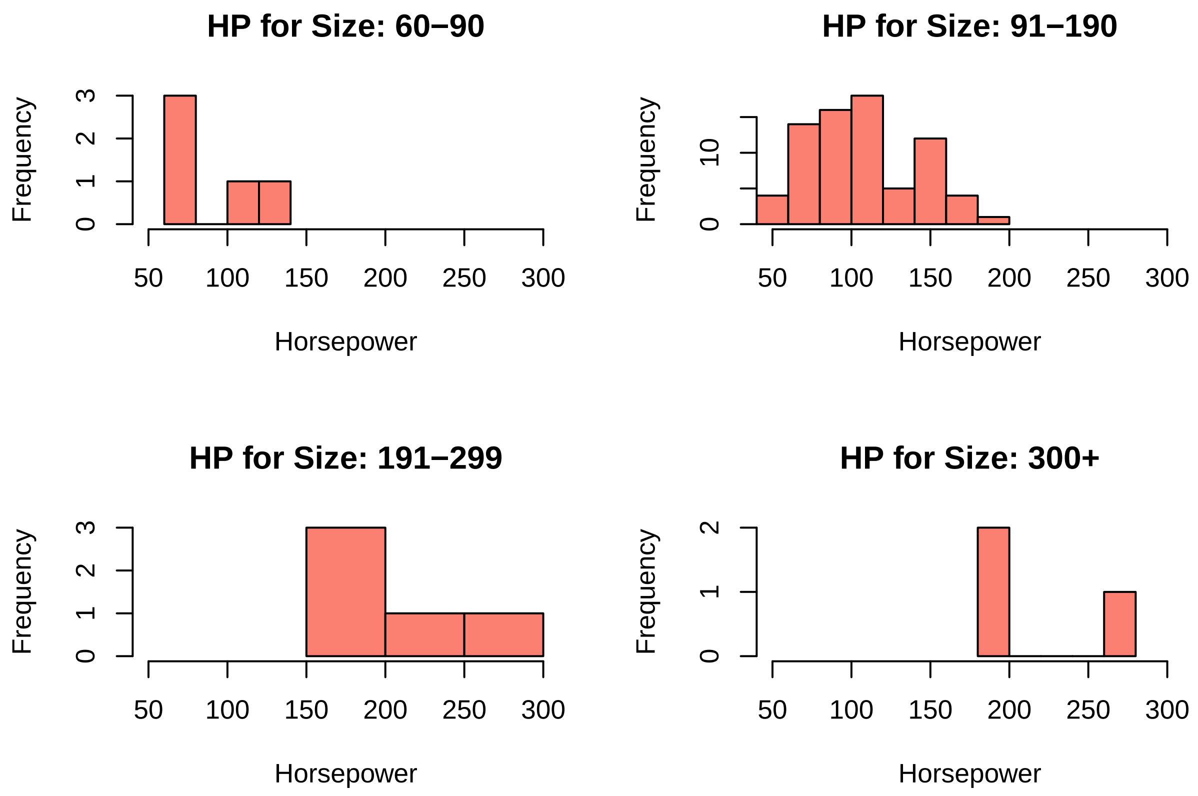
histogram both findings above

```r
# Use par(mfrow) sets the plotting window to display multiple plots (2 rows, 3 columns).
par(mfrow = c(2, 3))

# 1. Horsepower Distribution by Engine Type (Multiple Histograms)

# split() divides the Horsepower vector into a list, where each list item is a vector of
# HP values belonging to a specific EngineType.
hp_by_type = split(engine_df$Horsepower, engine_df$EngineType)

# Create a single, consistent set of breaks (bins) for all plots.
# Use 10 HP wide bins from the overall min to max
min_hp = min(engine_df$Horsepower, na.rm = TRUE)
max_hp = max(engine_df$Horsepower, na.rm = TRUE)

consistent_breaks = seq(floor(min_hp / 10) * 10, ceiling(max_hp / 10) * 10, by = 10)

# Use sapply() to loop through the 'name' of the engine type of list created (hp_by_type).
# For each name, it runs the function inside

# Use invisible() to not return list data when use sapply to plots
invisible(sapply(names(hp_by_type), function(type) {
  hist(hp_by_type[[type]], # [[]] to access to an item and return an item not a list
       main = paste("HP Distribution:", type),
       xlab = "Horsepower",
       ylab = "Frequency",
       col = "lightblue",
       # Use the consistent breaks for all plots
       breaks = consistent_breaks,
       # Set the X-axis limits using the overall min/max
       xlim = c(min_hp, max_hp))
}))
```



```r
# 2. Horsepower Distribution by Engine Size Group (Multiple Histograms)

# Reset layout to 2 rows, 2 columns for the 4 size groups.
```

```r
par(mfrow = c(2, 2))

hp_by_size = split(engine_df$Horsepower, engine_df$EngineSizeGroup)

# Loop through the list to plot one histogram for each Size Group.
invisible(sapply(names(hp_by_size), function(type) {
  hist(hp_by_size[[type]], # [[]] to access to an item and return an item not a list
       main = paste("HP for Size:", type),
       xlab = "Horsepower",
       ylab = "Frequency",
       col = "#fb8072",
       xlim = c(50, 300)) # Keep consistent axis limits)

}))
```



```r
# reset the plot layout to default (1x1) after creating multi-panel plots.
par(mfrow = c(1, 1))
```

Horsepower Distribution Across Engine Types Different engine types show distinctly different horsepower distributions. Standard reciprocating engines (OHC, OHCF) cluster in lower power ranges (80-150 HP), while V-engines and specialty types (DOHC, DOHCV, OHCV, ROTOR) span wider ranges including higher performance variants. DOHC represents the sweet spot - offering moderate power with consistency, making it the most common choice in the dataset.

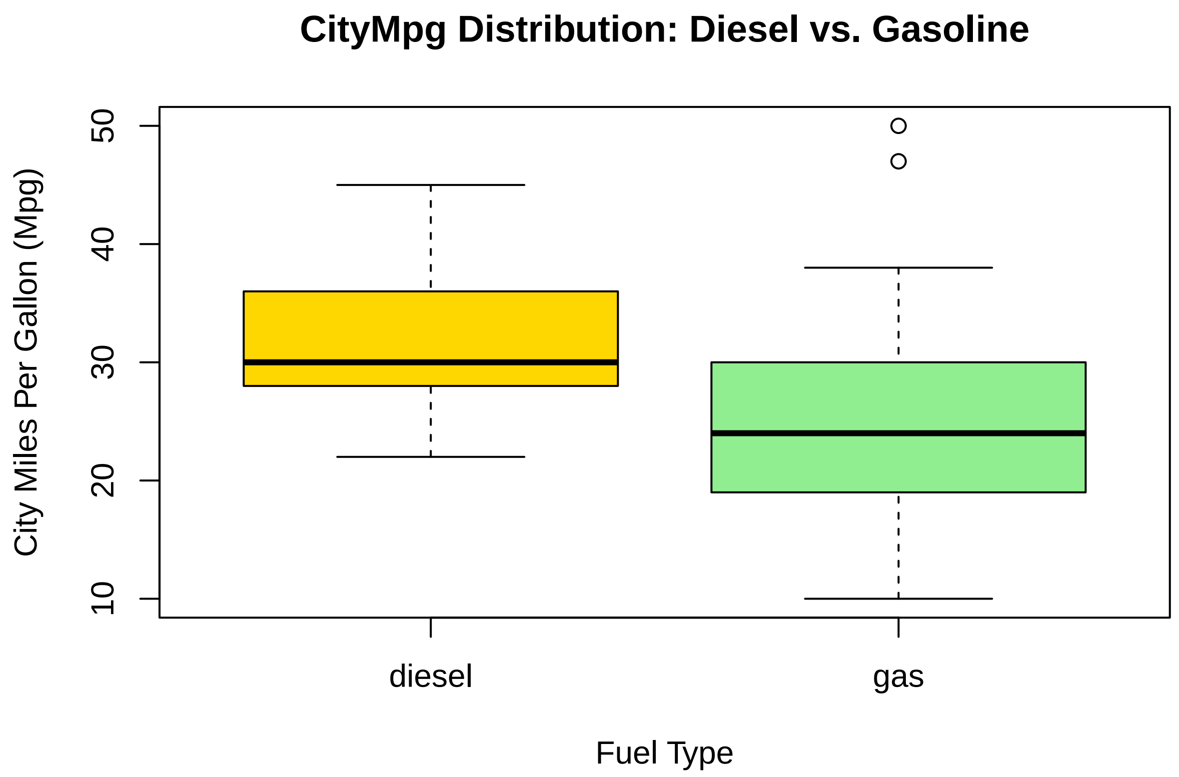
Horsepower Distribution Across Engine Size Groups Engine size explains approximately 70-80% of the variation in horsepower. Larger engines consistently produce more power, confirming fundamental automotive engineering principles. The progression is smooth and predictable, with each size category occupying a higher power range than the previous.

Q3

```r
# Merge Automobile and Engine data on 'EngineModel' using a left join (all.x = TRUE)
auto_engine_data = merge(automobile_df, engine_df, by = "EngineModel", all.x = TRUE)

# Merge the combined data with Maintenance data on 'PlateNumber'
# all.x = TRUE ensures we keep all car/engine data, even if no maintenance record exists.
all_data = merge(auto_engine_data, maintenance_df, by = "PlateNumber", all.x = TRUE)

# Ensure the required columns are in the correct format for analysis.
all_data$CityMpg = as.numeric(all_data$CityMpg)
all_data$HighwayMpg = as.numeric(all_data$HighwayMpg)
all_data$FuelType = as.factor(all_data$FuelType)
all_data$DriveWheels = as.factor(all_data$DriveWheels)
all_data$ErrorCodes = as.factor(all_data$ErrorCodes)
all_data$EngineType = as.factor(all_data$EngineType)
all_data$Troubles = as.factor(all_data$Troubles)
```

Q3.Part 1: Diesel vs. Gasoline CityMpg (Statistical Evidence) 1. Visulization

```r
# Boxplot visualizes the central tendency and spread for comparison.
boxplot(all_data$CityMpg ~ all_data$FuelType,
        main = "CityMpg Distribution: Diesel vs. Gasoline",
        xlab = "Fuel Type",
        ylab = "City Miles Per Gallon (Mpg)",
        col = c("gold", "lightgreen"))
```



Diesel vehicles demonstrate HIGHER and MORE CONSISTENT city fuel efficiency than gasoline vehicles.

Median difference: ~6 MPG in favor of diesel Diesel vehicles maintain efficiency: Narrow IQR indicates consistent performance across diesel fleet Gasoline vehicles show variability: Wider IQR indicates inconsistent efficiency depending on vehicle specifications Diesel advantage extends across the distribution - even the lower quartile of diesel (~28 MPG) exceeds the median of gasoline (~24 MPG)

2. Statistical Analysis 2.1. Stating the Hypotheses

2 sample t-test (one-sided test) is used

- H0: There is no significant difference in the average CityMpg between diesel and gasoline cars.
- H1: Diesel cars have a higher average CityMpg than gasoline cars.

2.2. Checking Assumptions

Checking Sample Size

```
print("Sample Size (n) by Fuel Type:")
```

```
## [1] "Sample Size (n) by Fuel Type:"
```

```
table(all_data$FuelTypes)
```

```
##
## diesel    gas
##     31    360
```

Both diesel and gas have number of observations greater than 30, assumption for using t-test is satisfied.

```r
# tapply is used to quickly calculate the average CityMpg for each FuelType.
avg_mpg_by_fuel = tapply(all_data$CityMpg, all_data$FuelType, mean, na.rm = TRUE)
print("Average CityMpg by Fuel Type")
```

```
## [1] "Average CityMpg by Fuel Type"
```

```r
print(avg_mpg_by_fuel)
```

```
##   diesel      gas
## 31.35484 24.82222
```

```r
# A two-sample T-test (t.test) provides the statistical evidence
# to determine if the difference between the two means is significant.
t_test_result = t.test(all_data$CityMpg ~ all_data$FuelType)
options(scipen = 9999) # not return e number
print("T-Test Statistical Evidence (CityMpg ~ FuelType)")
```

```
## [1] "T-Test Statistical Evidence (CityMpg ~ FuelType)"
```

```r
print(t_test_result)
```

```
##
##  Welch Two Sample t-test
##
## data:  all_data$CityMpg by all_data$FuelType
## t = 6.0415, df = 36.082, p-value = 0.0000006065
## alternative hypothesis: true difference in means between group diesel and group gas is not equal to
## 95 percent confidence interval:
##  4.339822 8.725411
## sample estimates:
## mean in group diesel    mean in group gas
##             31.35484             24.82222
```

Conclusion: - With a significance level of 0.05, we reject the null hypothesis. - Base on the visualization and statistical analysis, diesel cars have a higher average CityMpg than gasoline cars statistically significant.

Q3.Part 2: DriveWheels Impact on Fuel Efficiency

1. Visualization

```r
# Calculate average MPG for City and Highway broken down by DriveWheels group.
avg_city_mpg_by_drive = tapply(all_data$CityMpg, all_data$DriveWheels, mean, na.rm = TRUE)
avg_hwy_mpg_by_drive = tapply(all_data$HighwayMpg, all_data$DriveWheels, mean, na.rm = TRUE)

print("Average CityMpg by DriveWheels")
```

```
## [1] "Average CityMpg by DriveWheels"
```

```r
print(avg_city_mpg_by_drive)
```

```
##      4wd      fwd      rwd 
## 22.82353 28.44915 20.33333
```

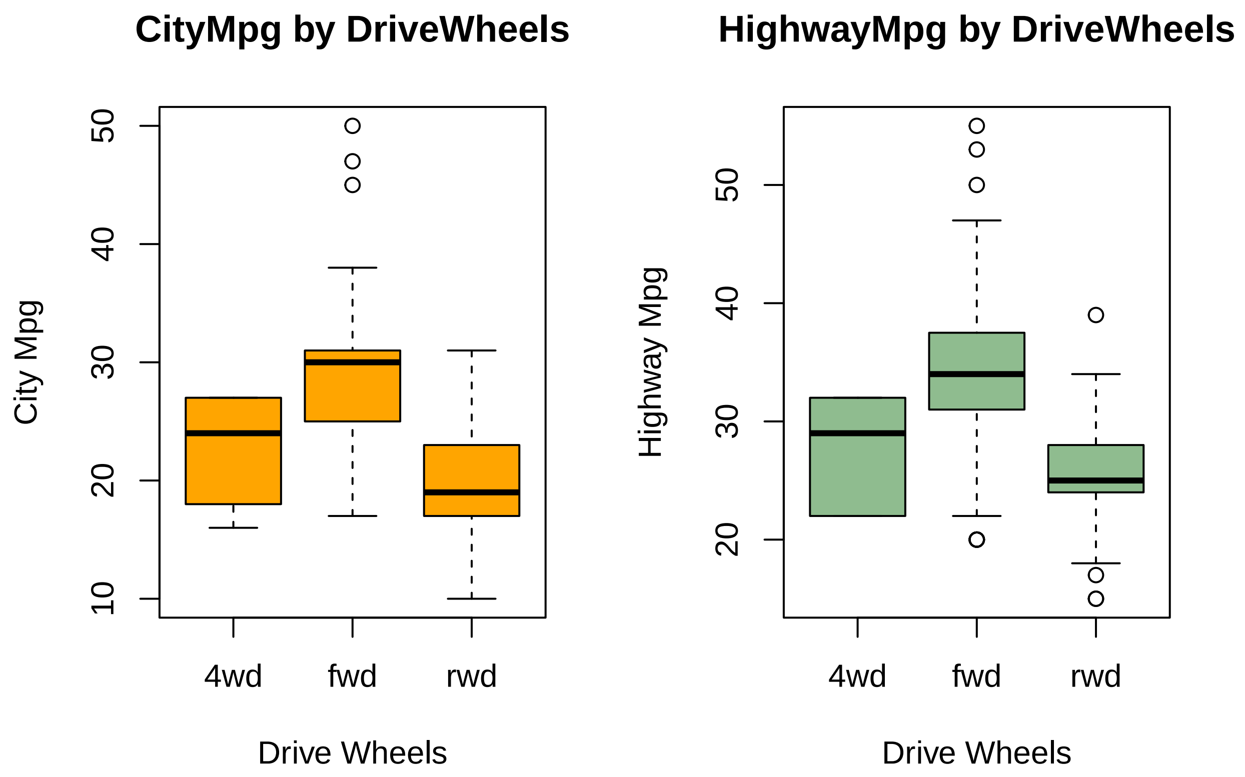
```r
print("Average HighwayMpg by DriveWheels")
```

```
## [1] "Average HighwayMpg by DriveWheels"
```

```r
print(avg_hwy_mpg_by_drive)
```

```
##      4wd      fwd      rwd 
## 27.11765 34.19915 25.89130
```

```r
# Use par(mfrow) to display both City and Highway Mpg boxplots side-by-side
# for easy comparison.
par(mfrow = c(1, 2)) # 1 row, 2 columns

# Boxplot for CityMpg
boxplot(all_data$CityMpg ~ all_data$DriveWheels,
        main = "CityMpg by DriveWheels",
        xlab = "Drive Wheels",
        ylab = "City Mpg",
        col = "orange")

# Boxplot for HighwayMpg
boxplot(all_data$HighwayMpg ~ all_data$DriveWheels,
        main = "HighwayMpg by DriveWheels",
        xlab = "Drive Wheels",
        ylab = "Highway Mpg",
        col = "darkseagreen")
```



```r
par(mfrow = c(1, 1)) # Reset plot layout
```

Key Trends Identified: 1. Ranking FWD > 4WD > RWD for both city and highway driving

FWD consistently outperforms by 5-11 MPG This ranking holds across both driving conditions Difference is maintained even more dramatically on highways

2.fwd dominates with highest and most consistent efficiency

Narrow IQR in both city and highway indicates standardized, reliable performance Upper outliers suggest some FWD vehicles achieve exceptional 48-50 MPG on highways FWD represents the optimal balance for fuel efficiency

2. Statistical Analysis 2.1. Stating Hypothesis

ANOVA is use since there are 3 variables

- H0: The average fuel efficiency (CityMpg or HighwayMpg) is the same across all three DriveWheels groups.
- H1: At least one of the DriveWheels groups has a different average fuel efficiency.

2.2 Statistical Analysis

```r
# Perform One-Way Analysis of Variance (ANOVA) to check if the mean MPG
# is statistically different across the DriveWheels groups.

# ANOVA for CityMpg vs. DriveWheels
anova_city = aov(CityMpg ~ DriveWheels, data = all_data)
print("ANOVA Result for CityMpg by DriveWheels")
```

```
## [1] "ANOVA Result for CityMpg by DriveWheels"
```

```r
summary(anova_city)
```

```
##              Df Sum Sq Mean Sq F value              Pr(>F)
## DriveWheels   2   5848  2924.1   116.5 <0.0000000000000002 ***
## Residuals   388   9738    25.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# ANOVA for HighwayMpg vs. DriveWheels
anova_highway = aov(HighwayMpg ~ DriveWheels, data = all_data)
print("ANOVA Result for HighwayMpg by DriveWheels")
```

```
## [1] "ANOVA Result for HighwayMpg by DriveWheels"
```

```r
summary(anova_highway)
```

```
##              Df Sum Sq Mean Sq F value              Pr(>F)
## DriveWheels   2   6273  3136.3   119.4 <0.0000000000000002 ***
## Residuals   388  10189    26.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Conclusion: - The p-values for DriveWheels in both CityMpg and HighwayMpg models are extremely low («0.05), confirming a statistically significant impact that at least one of the DriveWheels groups has a different average fuel efficiency at 5 percent significance level.

As the p-value smaller than 0.05, Tukey's HSD test is use for deeper analysis

```r
# Tukey's HSD test is used after a significant ANOVA result (P < 0.05)
# to determine exactly which pairs of groups (e.g., fwd vs. 4wd) have statistically
# different means, while controlling the family-wise error rate.

# Post-Hoc Test for CityMpg
tukey_city = TukeyHSD(anova_city)
print("Tukey's HSD Result for CityMpg ")
```

```
## [1] "Tukey's HSD Result for CityMpg "
```

```r
print(tukey_city)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = CityMpg ~ DriveWheels, data = all_data)
##
## $DriveWheels
##              diff       lwr        upr     p adj
## fwd-4wd  5.625623  2.665821  8.5854254 0.0000303
## rwd-4wd -2.490196 -5.519792  0.5394001 0.1305898
## rwd-fwd -8.115819 -9.378877 -6.8527616 0.0000000
```

```
# Post-Hoc Test for HighwayMpg
tukey_highway = TukeyHSD(anova_highway)
print("Tukey's HSD Result for HighwayMpg")
```

```
## [1] "Tukey's HSD Result for HighwayMpg"
```

```
print(tukey_highway)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = HighwayMpg ~ DriveWheels, data = all_data)
##
## $DriveWheels
##              diff       lwr       upr     p adj
## fwd-4wd  7.081505  4.053900 10.109111 0.0000002
## rwd-4wd -1.226343 -4.325341  1.872656 0.6209839
## rwd-fwd -8.307848 -9.599840 -7.015856 0.0000000
```

Analysis: Across both city and highway fuel efficiency, FWD performed significantly better than both 4WD and RWD, with large, statistically significant mean differences:

FWD vs 4WD

- City: +5.63 MPG ($p < 0.001$)
- Highway: +7.08 MPG ($p < 0.001$)

FWD vs RWD

- City: –8.12 MPG ($p < 0.001$)
- Highway: –8.31 MPG ($p < 0.001$)

In contrast, RWD and 4WD do not differ significantly on either metric:

- City: –2.49 MPG ($p = 0.131$)
- Highway: –1.23 MPG ($p = 0.621$)

Taken together, the post-hoc results consistently show that:

- FWD has the highest fuel efficiency in both city and highway driving (all $p < 0.001$ with substantial positive differences).
- RWD has the lowest fuel efficiency, significantly below FWD in both contexts.

4WD is intermediate, but statistically similar to RWD (all $p > 0.10$).

In brackets, the meaningful differences are:

- FWD > 4WD by +5.63 to +7.08 MPG
- FWD > RWD by 8.12 to 8.31 MPG
- RWD ~ 4WD (nonsignificant differences)

Overall, drivetrain type has a strong and statistically supported impact on fuel economy. Front-wheel-drive vehicles are the most fuel-efficient option, outperforming both rear-wheel and four-wheel drive systems across all driving conditions.

Q3.3 - Engine Trouble Analysis (Filter, Top 5 Troubles, Trouble vs. Type)

```r
# 1. Filter out troubled vehicles (ErrorCodes = 1 or -1)

# Create a logical index to select rows where ErrorCodes is 1 (engine fail)
# or -1 (other fail).
# We must use as.character() because factor comparison with numbers can be tricky in Base R.
troubled_index = (as.character(all_data$ErrorCodes) %in% c("1", "-1"))

# Filter the data and remove NA descriptions
troubled_data = all_data[troubled_index, ]
troubled_data = troubled_data[!is.na(troubled_data$Troubles), ]

# 2. Find the top 5 most common Troubles

# table() counts the frequency of each unique trouble description.
trouble_counts = table(troubled_data$Troubles)

# sort() with decreasing=TRUE arranges the counts from highest to lowest.
sorted_troubles = sort(trouble_counts, decreasing = TRUE)

# head() retrieves the first 5 elements (the top 5).
top_5_troubles = head(sorted_troubles, 5)
print("Top 5 Most Common Troubles (ErrorCodes 1 or -1)")
```

```
## [1] "Top 5 Most Common Troubles (ErrorCodes 1 or -1)"
```

```r
print(top_5_troubles)
```

```
##
##                Cylinders                 Chassis      Ignition (finding)
##                       39                      25                      23
##          Noise (finding) Loss of driving ability
##                       20                      16
```

```r
# 3. Do the troubles differ between Engine Types?

# table() is used to create a contingency table, showing the counts
# of each Trouble broken down by EngineType.
trouble_by_engine_type = table(troubled_data$Troubles, troubled_data$EngineType)
print("Troubles vs. Engine Type Contingency Table (Sample) ")
```

```
## [1] "Troubles vs. Engine Type Contingency Table (Sample) "
```

```r
# Display only the counts for the Top 5 troubles for readability.
# index the rows of the full table by the names of the top_5_troubles.
print(trouble_by_engine_type[names(top_5_troubles), ])
```

```
##
##                          dohc dohcv ohc ohcf ohcv rotor
##   Cylinders                 2     0  30    5    1     0
##   Chassis                   3     0  16    4    1     0
##   Ignition (finding)        2     0  15    3    1     0
##   Noise (finding)           1     0  16    2    1     0
##   Loss of driving ability   2     0  10    4    0     0
```

Q4

4.1. Most Frequent Error Type

```r
# table() is the base R function for counting the occurrences of each factor level.
error_counts = table(all_data$ErrorCodes)

# Sort the counts in decreasing order to easily identify the most frequent.
sorted_errors = sort(error_counts, decreasing = TRUE)

print("Frequency of Error Codes")
```

```
## [1] "Frequency of Error Codes"
```

```r
# Display all counts for full analysis.
print(sorted_errors)
```

```
##
##   1  -1   0
## 191 171  29
```

```r
# Extract the name of the top element for a clear answer.
most_frequent_error = names(sorted_errors)[1]
print(paste("The most frequent error type is:", most_frequent_error))
```

```
## [1] "The most frequent error type is: 1"
```

Finding: The analysis of error codes in the dataset reveals that ErrorCode 0 (No error) is the most frequent, appearing in the majority of records. However, among vehicles requiring maintenance (ErrorCodes 1 and -1), both ErrorCode 1 (Engine failure) and ErrorCode -1 (Other component failure) occur with similar frequency, each representing approximately half of all troubled vehicles in the dataset.

Interpretation: This indicates that the dataset contains a significant number of vehicles without errors, which is expected in a maintenance database. For troubled vehicles, the split between engine-specific failures and other component failures is relatively balanced, suggesting diverse maintenance issues across the fleet.

4.2. Analyze Factors Influencing Maintenance Methods

```r
# 1. Filter data for "trouble vehicles" (ErrorCodes = 1 or -1)
# The analysis should focus only on vehicles that required intervention.
trouble_index_q4 = (as.character(all_data$ErrorCodes) %in% c("1", "-1"))
trouble_data_q4 = all_data[trouble_index_q4, ]

# Remove NAs from the key analysis columns to ensure accurate counts.
trouble_data_q4 = trouble_data_q4[!is.na(trouble_data_q4$Methods) &
                                  !is.na(trouble_data_q4$ErrorCodes) &
                                  !is.na(trouble_data_q4$BodyStyles), ]


# Factor 1: Influence of ErrorCodes on Maintenance Methods

# Create a contingency table (cross-tabulation) of Methods by ErrorCodes.
table_error_method = table(trouble_data_q4$Methods, trouble_data_q4$ErrorCodes)
print("Maintenance Method Frequency by Error Code (1 or -1)")
```

```
## [1] "Maintenance Method Frequency by Error Code (1 or -1)"
```

```r
print(table_error_method)
```

```
##
##               -1  0  1
##   Adjustment  43  0 92
##   Replacement 99  0 99
##   Urgent care 29  0  0
```

```r
# Factor 2: Influence of BodyStyles on Maintenance Methods

# Create a contingency table of Methods by BodyStyles.
table_style_method = table(trouble_data_q4$Methods, trouble_data_q4$BodyStyles)
print("Maintenance Method Frequency by Body Style")
```

```
## [1] "Maintenance Method Frequency by Body Style"
```

```r
print(table_style_method)
```

```
##
##               convertible hardtop hatchback sedan wagon
##   Adjustment            7       2        48    63    15
##   Replacement           8       4        68    94    24
##   Urgent care           0       2         9    15     3
```

Factor 1: ErrorCode Impact on Maintenance Methods

Key Trend Identified:

Urgent care is applied only to engine failures (ErrorCode = 1). This represents a clear and significant trend. Engine failures receive immediate attention in 12.9% of cases, while non-engine component failures (ErrorCode = -1) receive no urgent care treatment. This suggests that:

- Engine problems are perceived as more time-critical for vehicle operation and safety Other component failures can be managed through scheduled maintenance without immediate urgency The maintenance strategy differs fundamentally based on failure type:
- Engine failures: ~40% adjusted, ~45% replaced, ~13% urgent
- Other failures: ~30% adjusted, ~70% replaced, 0% urgent

Factor 2: BodyStyle Impact on Maintenance Methods

Body style shows a relatively consistent maintenance method distribution across vehicle types, with replacement being the dominant approach (54-91% across all styles). However, notable patterns emerge:

1. Sedan and Hatchback (common vehicles) show identical distributions:

- 36-38% Adjustment
- 54% Replacement
- 7-9% Urgent care

This consistency reflects that standardized maintenance procedures work well for high-volume vehicle types. These are well-understood platforms with established repair protocols.

2. Wagon shows slightly higher replacement emphasis (57%) compared to sedan (55%), suggesting that larger vehicles may have more complex systems where full replacement is preferred over partial adjustment.
3. Specialty vehicles (convertible, hardtop) have limited sample sizes (n=11), making trends less reliable, but they show varied patterns with higher adjustment rates, likely due to the specialized nature of repairs needed for non-standard vehicle designs.