Data Wrangling Assessment Task 3: Dataset challenge

Michael Teixeira S4133975

1. Setup

1.1 Load libraries:

```r
library(tidyverse)
library(knitr)
library(stringr)
library(magrittr)
library(chron)
library(lubridate)
library(skimr)
```

2. Introduction

For this analysis, we are using three datasets from VicRoads Open Data[01]:

- ACCIDENT.csv: Contains detailed information about individual road accidents in Victoria.
  - ACCIDENT_NO: Unique identifier for the accident.
  - ACCIDENTDATE: Date when the accident occurred.
  - ACCIDENTTIME: Time of the accident.
  - ACCIDENT_TYPE: Numeric code for the type of accident (1-9).
  - ACCIDENT_TYPE_DESCRIPTION: Description of the accident type.
  - DAY_OF_WEEK: Numeric code for the day of the week.
  - DAY_OF_WEEK_DESCRIPTION: Description of the day of the week.
  - DCA_CODE: Numeric code for accident classification.
  - DCA_CODE_DESCRIPTION: Description of the accident classification.
  - LIGHT_CONDITION: Numeric code for light condition at the time of the accident (1-9).
  - LIGHT_CONDITION_DESCRIPTION: Description of the light condition.
  - NODE_ID: Unique identifier for the accident location.
  - NO_OF_VEHICLES: Number of vehicles involved in the accident.
  - NO_PERSONS: Total number of people involved in the accident.
  - NO_PERSONS_INJ_2: Number of people with serious injuries.
  - NO_PERSONS_INJ_3: Number of people with other injuries.
  - NO_PERSONS_KILLED: Number of people killed.
  - NO_PERSONS_NOT_INJ: Number of people with no injuries.
  - POLICE_ATTEND: Indicates police attendance (1=Yes, 2=No, 9=Unknown).

    - ROAD_GEOMETRY: Numeric code for road layout where the accident occurred.
    - ROAD_GEOMETRY_DESCRIPTION: Description of the road layout.
    - SEVERITY: Severity of the accident in numeric code.
    - SPEED_ZONE: Speed zone at the accident location.
- ATMOSPHERIC_COND.csv: Includes data about weather conditions during accidents.
    - ACCIDENT_NO: Unique identifier for the accident.
    - ATMOSPH_COND: Code for weather and atmospheric conditions.
    - ATMOSPH_COND_SEQ: Sequence number for multiple atmospheric conditions in the same incident.
    - ATMOSPH_COND_Desc: Description of atmospheric conditions (e.g., Clear, Raining, Snowing).
- ROAD_SURFACE_COND.csv: Contains information about the condition of the road surface during accidents.
    - ACCIDENT_NO: Unique identifier for the accident.
    - SURFACE_COND: Numeric code for road surface conditions.
    - SURFACE_COND_Desc: Description of road surface conditions (e.g., Dry, Wet, Muddy).
    - SURFACE_COND_SEQ: Sequence number for multiple road surface conditions in the same incident.

These datasets will allow us to explore the relationships between road accidents, weather conditions, and road surface conditions in Victoria.

```
# Importing datasets:
accidents <- read.csv("ACCIDENT.csv")
atmospheric <- read.csv("ATMOSPHERIC_COND.csv")
road_surface <- read.csv("ROAD_SURFACE_COND.csv")

# Glimpse of each dataset:
glimpse(accidents)

## Rows: 169,877
## Columns: 23
## $ ACCIDENT_NO        <chr> "T20120000009", "T20120000012", "T20120000013",
"T2…
## $ ACCIDENT_DATE      <chr> "2012-01-01", "2012-01-01", "2012-01-01",
"2012-01-…
## $ ACCIDENT_TIME      <chr> "02:25:00", "02:00:00", "03:35:00", "05:15:00",
"07…
## $ ACCIDENT_TYPE      <int> 4, 1, 1, 4, 4, 4, 2, 1, 1, 2, 4, 1, 1, 4, 8, 4,
6, …
## $ ACCIDENT_TYPE_DESC <chr> "Collision with a fixed object", "Collision
with ve…
## $ DAY_OF_WEEK        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ DAY_WEEK_DESC      <chr> "Sunday", "Sunday", "Sunday", "Sunday",
```

```
"Sunday", "…
## $ DCA_CODE            <int> 171, 110, 160, 173, 171, 183, 108, 116, 120,
102, 1…
## $ DCA_DESC            <chr> "LEFT OFF CARRIAGEWAY INTO OBJECT/PARKED
VEHICLE", …
## $ LIGHT_CONDITION     <int> 5, 3, 3, 5, 1, 5, 3, 5, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ NODE_ID             <int> 249102, 41780, 69811, 22636, 248597, 248598,
53249,…
## $ NO_OF_VEHICLES      <int> 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1,
1, …
## $ NO_PERSONS_KILLED   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, …
## $ NO_PERSONS_INJ_2    <int> 0, 1, 1, 0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 2, 1, 2,
1, …
## $ NO_PERSONS_INJ_3    <int> 2, 0, 0, 1, 2, 0, 1, 0, 0, 1, 1, 2, 0, 0, 0, 0,
0, …
## $ NO_PERSONS_NOT_INJ <int> 0, 2, 0, 0, 1, 0, 1, 1, 1, 1, 0, 4, 1, 0, 0, 0,
0, …
## $ NO_PERSONS          <int> 2, 3, 1, 1, 3, 1, 2, 3, 2, 2, 1, 7, 2, 2, 1, 2,
1, …
## $ POLICE_ATTEND       <int> 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ ROAD_GEOMETRY       <int> 5, 1, 2, 1, 5, 2, 2, 2, 2, 2, 5, 2, 1, 2, 1, 5,
2, …
## $ ROAD_GEOMETRY_DESC <chr> "Not at intersection", "Cross intersection", "T
int…
## $ SEVERITY            <int> 3, 2, 2, 3, 3, 2, 3, 2, 2, 3, 3, 2, 2, 2, 2, 2,
2, …
## $ SPEED_ZONE          <int> 100, 80, 60, 100, 50, 100, 50, 80, 60, 60, 999,
80,…
## $ RMA                 <chr> "Arterial Other", "", "Arterial Other",
"Arterial H…
```

```
glimpse(atmospheric)
```

```
## Rows: 172,120
## Columns: 4
## $ ACCIDENT_NO        <chr> "T20120006834", "T20120006879", "T20120006881",
"T20…
## $ ATMOSPH_COND       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 1, 1, 1, 1, 1,
1, 1…
## $ ATMOSPH_COND_SEQ   <int> 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1…
## $ ATMOSPH_COND_DESC <chr> "Clear", "Clear", "Clear", "Clear", "Clear",
"Clear"…
```

```
glimpse(road_surface)
```

```
## Rows: 170,839
## Columns: 4
```

```
## $ ACCIDENT_NO       <chr> "T20120013874", "T20120013876", "T20120013886",
"T20…
## $ SURFACE_COND      <int> 1, 1, 1, 2, 5, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1,
2, 3…
## $ SURFACE_COND_DESC <chr> "Dry", "Dry", "Dry", "Wet", "Icy", "Dry", "Dry",
"Dr…
## $ SURFACE_COND_SEQ  <int> 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 2…
```

3. Understand our Datasets

We begin by inspecting the data types and structure of the datasets to understand their content and identify any issues. This step helps ensure that data is in the correct format for analysis.

```
# Inspect data types and structure
str(accidents)    # 169877 obs. of  23 variables
str(atmospheric)  # 172120 obs. of  4 variables
str(road_surface) # 170839 obs. of  4 variables

# Summary statistics
summary(accidents)
summary(atmospheric)
summary(road_surface)
```

4. Tyding the Data

We clean and restructure the datasets by removing unnecessary columns, renaming variables for consistency, and merging the datasets.

4.1 Drop unnecessary variables and rename remaining variables

```
# Function to convert to lowercase:
lower_case <- function(x) {
  x %>% tolower()}

# Drop unnecessary variables and rename remaining variables in accidents
dataset:
accidents %<>%
  select(-DAY_OF_WEEK, -ACCIDENT_TYPE, -RMA, -ROAD_GEOMETRY, -DCA_CODE, -
NODE_ID) %>%
  rename(
    accident_id = ACCIDENT_NO,
    date = ACCIDENT_DATE,
    time = ACCIDENT_TIME,
    type_desc = ACCIDENT_TYPE_DESC,
    day_of_week = DAY_WEEK_DESC,
    desc_CA = DCA_DESC,
    total_people_involved = NO_PERSONS,
    serious_injuries = NO_PERSONS_INJ_2,
    minor_injuries = NO_PERSONS_INJ_3,
```

```
    fatalities = NO_PERSONS_KILLED,
    uninjured_people = NO_PERSONS_NOT_INJ
  )%>%
rename_all(lower_case)


# Drop unnecessary variables and rename remaining variables in atmospheric
dataset:
atmospheric %<>%
  select(-ATMOSPH_COND, -ATMOSPH_COND_SEQ) %>%
  rename(
    accident_id = ACCIDENT_NO,
    weather_condition = ATMOSPH_COND_DESC
  )

# Drop unnecessary variables and rename remaining variables in road_surface
dataset:
road_surface %<>%
  select(-SURFACE_COND, -SURFACE_COND_SEQ) %>%
  rename(
    accident_id = ACCIDENT_NO,
    surface_condition = SURFACE_COND_DESC
  )
```

4.2 Merge datasets

```
# Merge datasets
merged_data <- accidents %>%
  left_join(atmospheric, by = "accident_id") %>%
  left_join(road_surface, by = "accident_id")

## Warning in left_join(., road_surface, by = "accident_id"): Detected an
unexpected many-to-many relationship between `x` and `y`.
## ℹ Row 1429 of `x` matches multiple rows in `y`.
## ℹ Row 12445 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

4.3 Convert variables to appropriate types

```
# Convert merged dataset variables to appropriate types
merged_data %<>%
  mutate(
    accident_id = as.factor(accident_id),
    date = as.Date(date, format = "%Y-%m-%d"),
    time = times(time),
    type_desc = as.factor(type_desc),
    day_of_week = factor(day_of_week,
                         levels = c("Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday", "Sunday"),
                         labels = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat",
"Sun"),
```

```
                          ordered = TRUE),
    desc_ca = as.factor(desc_ca),
    light_condition = factor(light_condition,
                                levels = c(1, 2, 3, 4, 5, 6, 9),
                                labels = c("Day",
                                           "Dusk/Dawn",
                                           "Dark Street lights on",
                                           "Dark Street lights off",
                                           "Dark No street lights",
                                           "Dark Street lights unknown",
                                           "Unknown")),
    severity = factor(severity,
                         levels = c(1, 2, 3, 4),
                         labels = c("Fatal accident",
                                    "Serious injury accident",
                                    "Other injury accident",
                                    "Non injury accident")),
    police_attend = factor(police_attend,
                              levels = c(1, 2, 3),
                              labels = c("Yes",
                                         "No",
                                         "Unknown")),
    road_geometry_desc = as.factor(road_geometry_desc),
    weather_condition = as.factor(weather_condition),
    surface_condition = as.factor(surface_condition)
  )
```

4.4 Check for duplicates

```
# Check for duplicate accident_id
duplicate_ids <- merged_data %>%
  group_by(accident_id) %>%
  filter(n() > 1)

nrow(duplicate_ids)

## [1] 6374

head(duplicate_ids)

## # A tibble: 6 × 19
## # Groups:   accident_id [3]
##   accident_id  date       time     type_desc day_of_week desc_ca
light_condition
##   <fct>        <date>     <times>  <fct>     <ord>       <fct>   <fct>
## 1 T20120000206 2012-01-04 10:44:00 Vehicle … Wed         OFF CA… Day
## 2 T20120000206 2012-01-04 10:44:00 Vehicle … Wed         OFF CA… Day
## 3 T20120000557 2012-01-09 09:30:00 collisio… Mon         STRUCK… Day
## 4 T20120000557 2012-01-09 09:30:00 collisio… Mon         STRUCK… Day
## 5 T20120000751 2012-01-11 12:40:00 Collisio… Wed         RIGHT … Day
## 6 T20120000751 2012-01-11 12:40:00 Collisio… Wed         RIGHT … Day
```

```
## # ℹ 12 more variables: no_of_vehicles <int>, fatalities <int>,
## #   serious_injuries <int>, minor_injuries <int>, uninjured_people <int>,
## #   total_people_involved <int>, police_attend <fct>, road_geometry_desc
<fct>,
## #   severity <fct>, speed_zone <int>, weather_condition <fct>,
## #   surface_condition <fct>
```

```
# Remove all duplicate rows
merged_data %<>%
  distinct(accident_id, .keep_all = TRUE)  # Keep all columns, but only
unique accident_id values

# compact view of our tidy data
glimpse(merged_data)
```

```
## Rows: 169,877
## Columns: 19
## $ accident_id           <fct> T20120000009, T20120000012, T20120000013,
T20120…
## $ date                  <date> 2012-01-01, 2012-01-01, 2012-01-01, 2012-
01-01,…
## $ time                  <times> 02:25:00, 02:00:00, 03:35:00, 05:15:00,
07:30:…
## $ type_desc             <fct> Collision with a fixed object, Collision
with ve…
## $ day_of_week           <ord> Sun, Sun, Sun, Sun, Sun, Sun, Sun, Sun, Sun,
Sun…
## $ desc_ca               <fct> LEFT OFF CARRIAGEWAY INTO OBJECT/PARKED
VEHICLE,…
## $ light_condition       <fct> Dark No street lights, Dark Street lights
on, Da…
## $ no_of_vehicles        <int> 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1,
1, …
## $ fatalities            <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, …
## $ serious_injuries      <int> 0, 1, 1, 0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 2, 1,
2, …
## $ minor_injuries        <int> 2, 0, 0, 1, 2, 0, 1, 0, 0, 1, 1, 2, 0, 0, 0,
0, …
## $ uninjured_people      <int> 0, 2, 0, 0, 1, 0, 1, 1, 1, 1, 0, 4, 1, 0, 0,
0, …
## $ total_people_involved <int> 2, 3, 1, 1, 3, 1, 2, 3, 2, 2, 1, 7, 2, 2, 1,
2, …
## $ police_attend         <fct> Yes, Yes, Yes, Yes, Yes, Yes, No, Yes, Yes,
Yes,…
## $ road_geometry_desc    <fct> Not at intersection, Cross intersection, T
inter…
## $ severity              <fct> Other injury accident, Serious injury
accident, …
## $ speed_zone            <int> 100, 80, 60, 100, 50, 100, 50, 80, 60, 60,
```

```
999, …
## $ weather_condition     <fct> Clear, Clear, Clear, Clear, Clear, Clear, Clear,…
## $ surface_condition     <fct> Dry, Dry, Dry, Dry, Dry, Dry, Dry, Dry, Dry, Dry…
```

We removed DAY_OF_WEEK, ACCIDENT_TYPE, RMA, DCA_CODE, NODE_ID and ROAD_GEOMETRY from the accidents dataset as this information was either redundant or not relevant to our analysis. We renamed variables for clarity and consistency across datasets. For example, ACCIDENT_NO was renamed to accident_id to serve as a clear identifier across all datasets. We removed a total of 6374 duplicates from the database.

5. Creating new variables

We created a new variable total_casualties by summing fatalities, serious_injuries, and minor_injuries. We also extracted month and year from date.

```r
# Create total_casualties and extract month and year
merged_data %<>%
  mutate(
    total_casualties = fatalities + serious_injuries + minor_injuries,
    month = month(date, label = TRUE, abbr = TRUE),  # Extract month
    year = year(date)  # Extract year
  )

# Display the first few rows of only the new variables with accident_id.
head(merged_data %>% select(accident_id, total_casualties, month, year))

##    accident_id total_casualties month year
## 1 T20120000009                2   Jan 2012
## 2 T20120000012                1   Jan 2012
## 3 T20120000013                1   Jan 2012
## 4 T20120000018                1   Jan 2012
## 5 T20120000021                2   Jan 2012
## 6 T20120000028                1   Jan 2012
```

The month variable helps analyze trends and patterns across different months, while the year variable provides temporal context for the data. The new total_casualties variable provides a single measure of accident severity in terms of human impact.

6. Handling missing values

In this step we will be handling missing values, checking for inconsistencies and standardizing missing and unclear values:

6.1 Check for missing values

```r
# Check for missing values:
sum(is.na(merged_data))

## [1] 599
```

```r
colSums(is.na(merged_data))
```

```
##           accident_id                  date                  time 
##                     0                     0                     0 
##             type_desc           day_of_week               desc_ca 
##                     0                     0                     0 
##       light_condition         no_of_vehicles            fatalities 
##                     0                     0                     0 
##      serious_injuries        minor_injuries      uninjured_people 
##                     0                     0                     0 
## total_people_involved         police_attend    road_geometry_desc 
##                     0                   599                     0 
##              severity            speed_zone     weather_condition 
##                     0                     0                     0 
##     surface_condition      total_casualties                 month 
##                     0                     0                     0 
##                  year 
##                     0
```

6.2 Check for inconsistencies or unexpected values

```r
# List of categorical variables to check for inconsistencies
categorical_vars <- c("type_desc", "day_of_week", "desc_ca",
                       "light_condition",
"road_geometry_desc","police_attend",
                       "severity", "weather_condition", "surface_condition")

# Function to display unique values for categorical variables
check_categorical_vars <- function(data, variables) {
  for (var in variables) {
    cat("Unique values for", var, ":\n")
    print(table(data[[var]]))
    cat("\n")
  }
}

check_categorical_vars(merged_data, categorical_vars)
```

We identified 599 missing entries in the police_attend variable.

```r
# Check for missing values in 'police_attend'
missing_count <- sum(is.na(merged_data$police_attend))
cat("Number of missing values in 'police_attend':", missing_count, "\n")
```

```
## Number of missing values in 'police_attend': 599
```

```r
# Replace NA values with 'Unknown'
merged_data$police_attend[is.na(merged_data$police_attend)] <- "Unknown"

# Display summary of 'police_attend'
summary(merged_data$police_attend)
```

```
##     Yes      No Unknown
##  125985   43293     599

# Replace "Not known" with "Unknown" in weather_condition
merged_data %<>%
  mutate(weather_condition = recode(weather_condition, "Not known" = 
"Unknown"))

# Replace "Unk." with "Unknown" in surface_condition
merged_data %<>%
  mutate(surface_condition = recode(surface_condition, "Unk." = "Unknown"))
```

- We identified and counted missing values in the dataset.
- Checked categorical variables for any inconsistencies or unexpected values.
- Replaced missing values in the severity column with “Unknown”.
- Standardized categorical values by replacing specific codes (“Not known”, “Unk.”) with “Unknown” in the weather_condition and surface_condition columns.

7.Detect and Handle Outliers

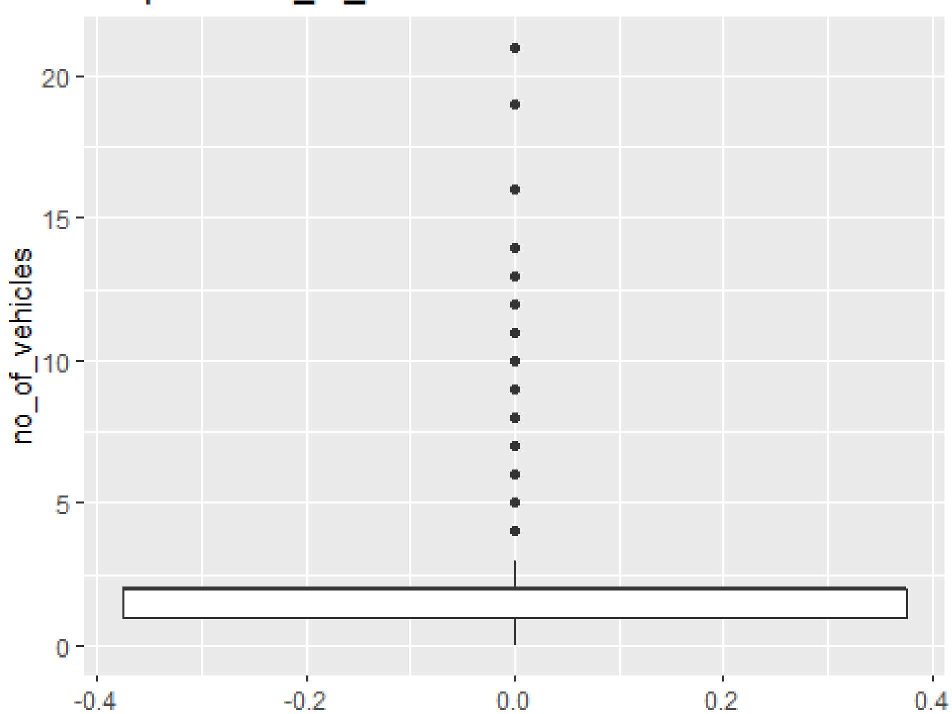
In this section, we’ll perform outlier detection for numeric columns to identify and address any extreme values that could affect the analysis. We will also visualize the data before and after removing outliers.

For each numeric variable, we’ll calculate the Interquartile Range (IQR)[02] and define outliers as values outside 1.5 times the IQR from the quartiles.

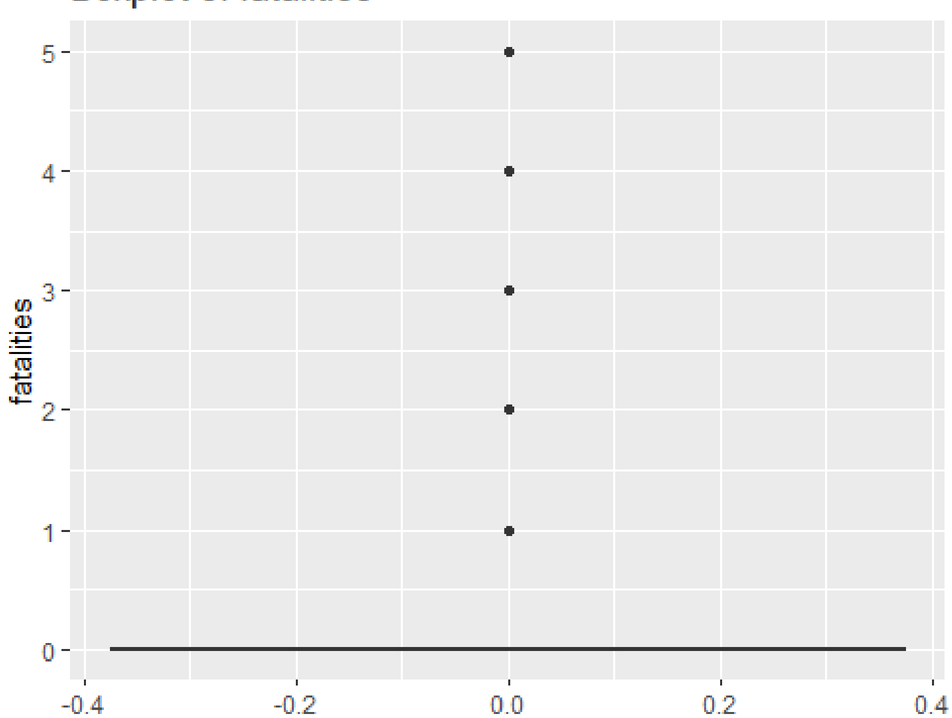
7.1 We will creat a list of all variables and a function to generate boxplots:

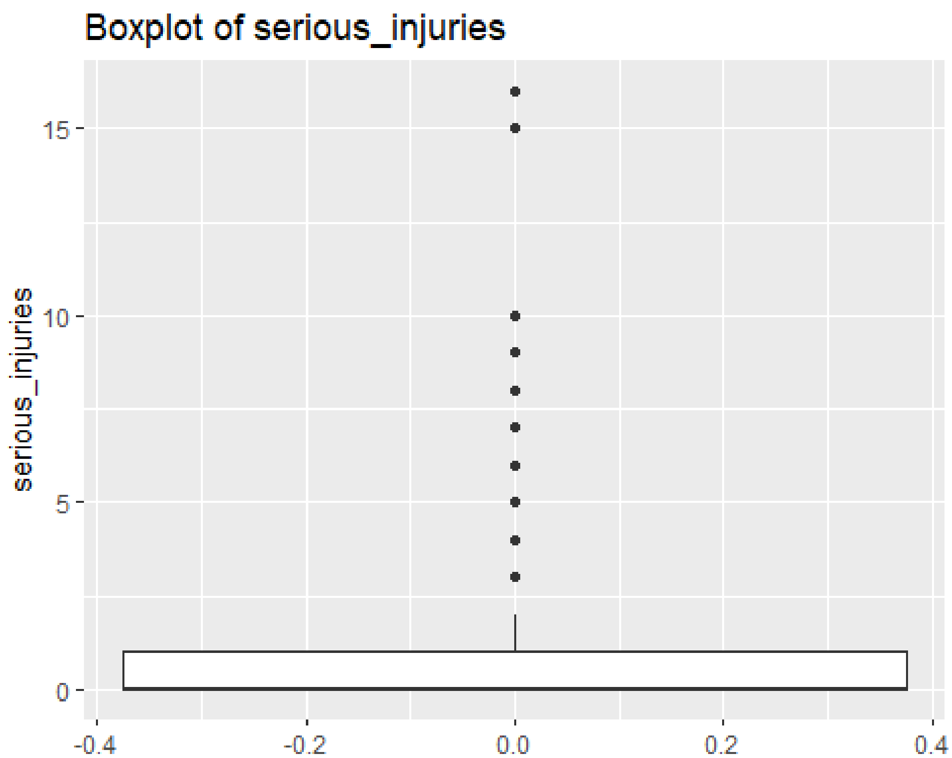
```
# List of numeric variables:
numeric_vars <- c("no_of_vehicles", "fatalities", "serious_injuries",
                  "minor_injuries", "uninjured_people", 
"total_people_involved",
                  "speed_zone")

# Create a function to generate boxplots
create_boxplots <- function(data, variables) {
  for (var in variables) {
    p <- ggplot(data, aes_string(y = var)) +
      geom_boxplot() +
      labs(title = paste("Boxplot of", var), y = var)
    print(p)
  }
}

create_boxplots(merged_data, numeric_vars)
```

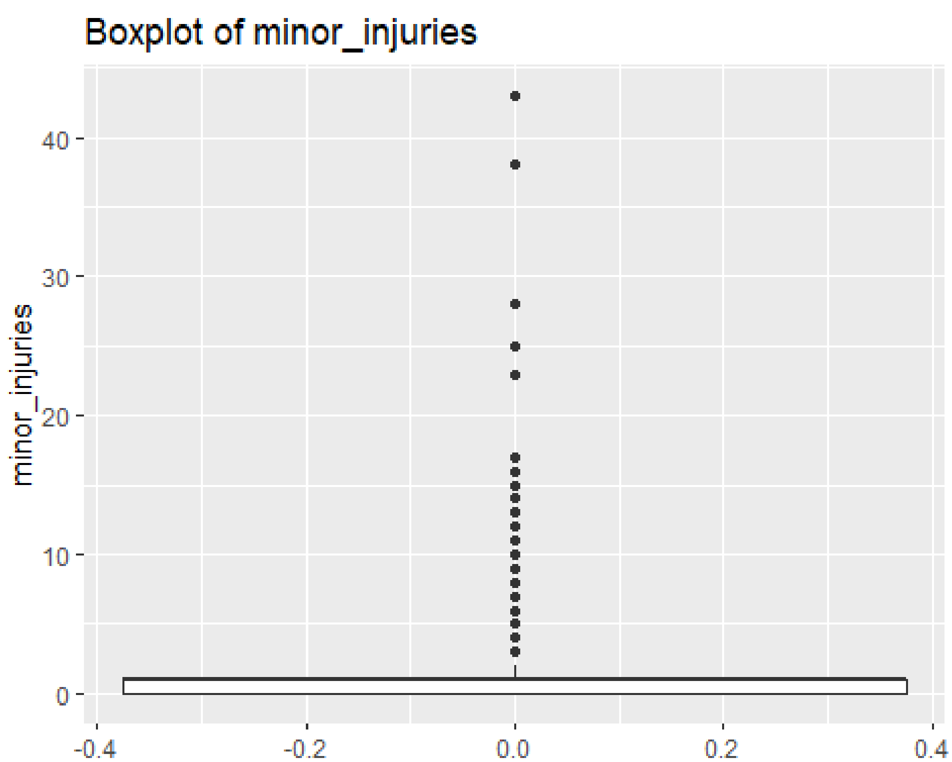
Boxplot of no_of_vehicles

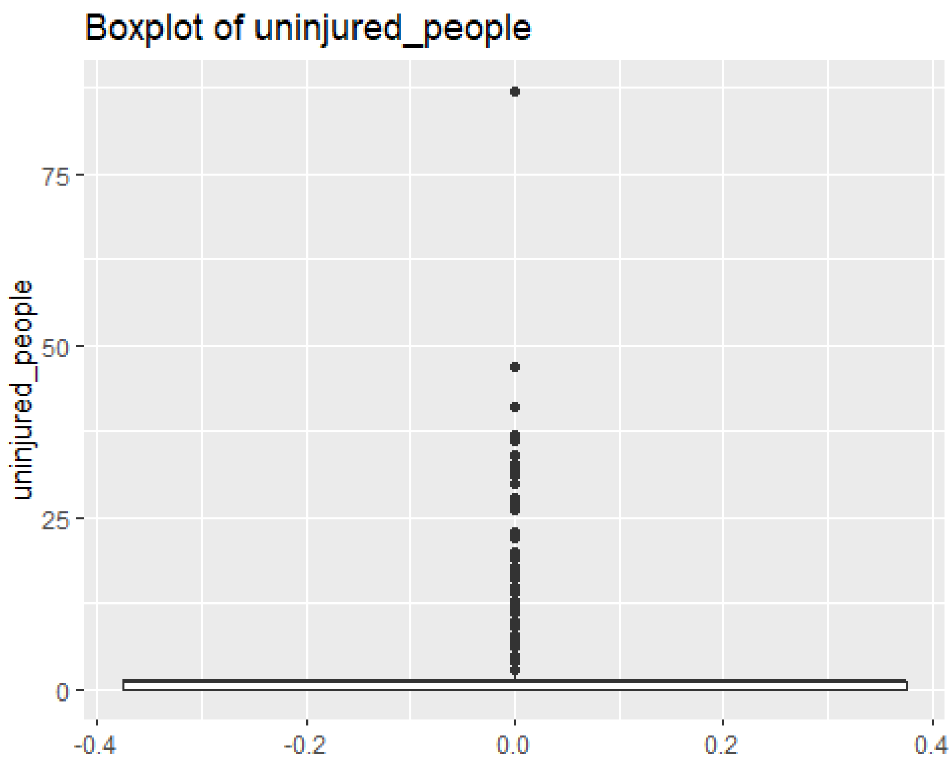


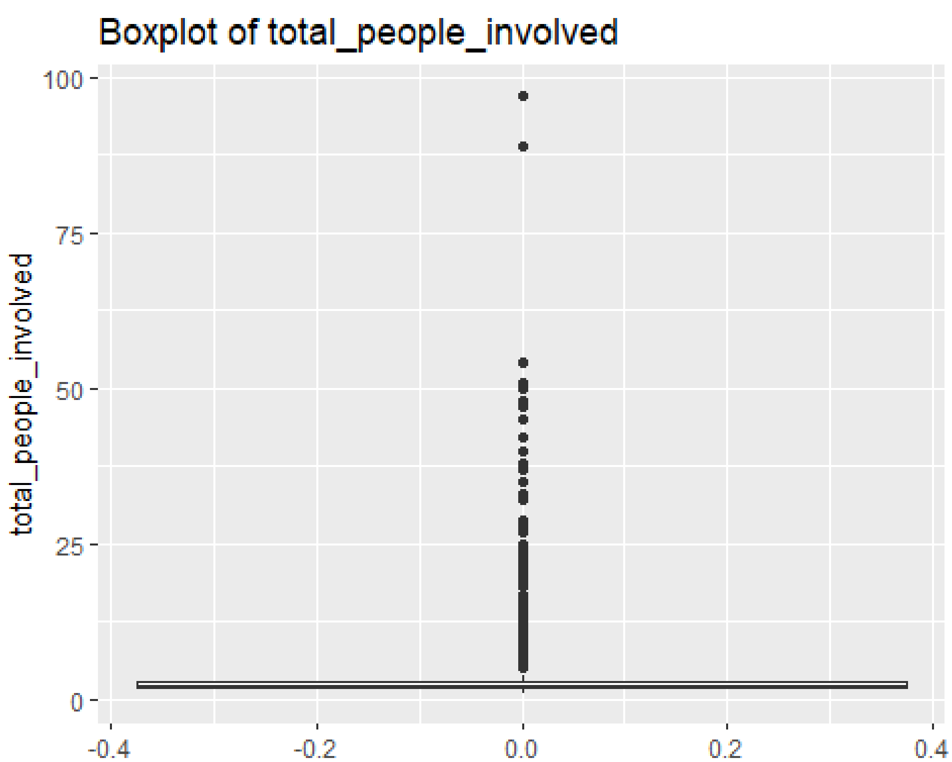
Boxplot of fatalities

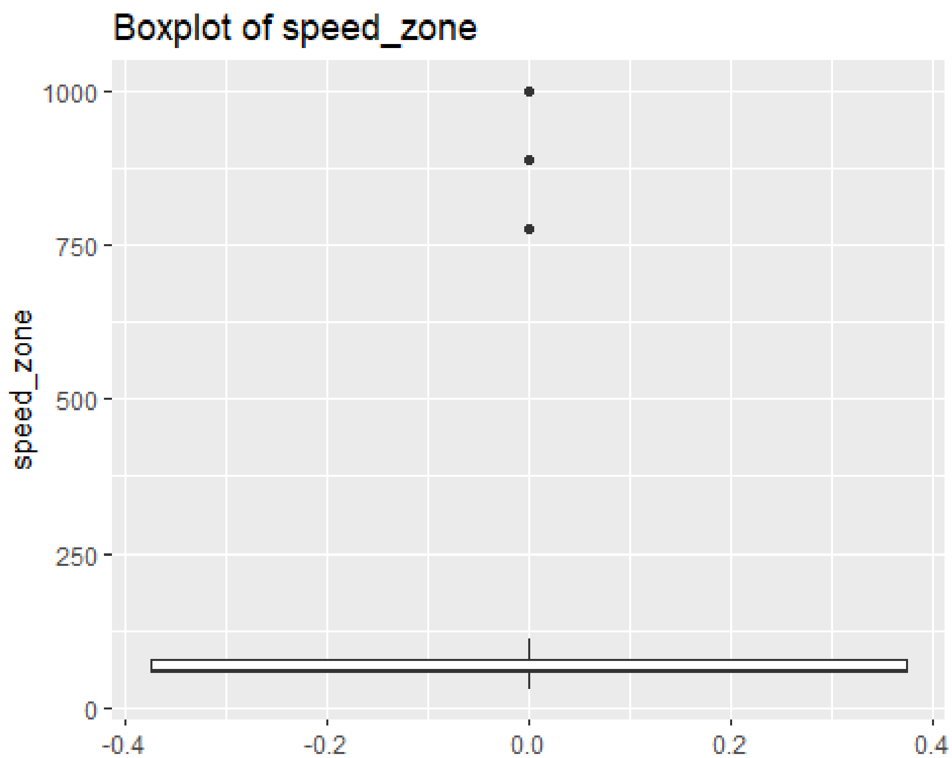



Boxplot of serious_injuries
serious_injuries
0
5
10
15
-0.4
-0.2
0.0
0.2
0.4


Boxplot of minor_injuries
minor_injuries
0
10
20
30
40
-0.4
-0.2
0.0
0.2
0.4


Boxplot of uninjured_people
uninjured_people
75
50
25
0
-0.4
-0.2
0.0
0.2
0.4


Boxplot of total_people_involved
total_people_involved
100
75
50
25
0
-0.4
-0.2
0.0
0.2
0.4



*7.1.1 Outlier Detection **‘no_of_vehicles’**:*

```
# Calculate Q1 (25th percentile) and Q3 (75th percentile):
Q1 <- quantile(merged_data$no_of_vehicles, 0.25, na.rm = TRUE)
Q3 <- quantile(merged_data$no_of_vehicles, 0.75, na.rm = TRUE)

# Calculate IQR (Interquartile Range)
IQR <- Q3 - Q1

# Define outlier fence:
lower_fence <- Q1 - 1.5 * IQR
upper_fence <- Q3 + 1.5 * IQR

# Print number of outliers:
cat("Lower Fence for Outliers:", lower_fence, "\n")

## Lower Fence for Outliers: -0.5

cat("Upper Fence for Outliers:", upper_fence, "\n")

## Upper Fence for Outliers: 3.5

# Find outliers:
outliers <- merged_data %>%
  filter(no_of_vehicles < lower_fence | no_of_vehicles > upper_fence)

# Count the number of outliers:
```

```r
num_outliers <- nrow(outliers)
cat("Number of Outliers:", num_outliers, "\n")

## Number of Outliers: 4649

# Analyse outliers:
summary(outliers$no_of_vehicles)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.000   4.000   4.000   4.408   5.000  21.000

outliers %<>%
  arrange(desc(no_of_vehicles))
head(outliers$no_of_vehicles)

## [1] 21 19 16 14 13 13
```

The values appear realistic, and no additional action is needed for the outliers in `merged_data$no_of_vehicles`.

We repeat similar analysis for other numeric variables:

*7.1.2 Outlier Detection **fatalities**:*

```r
# Calculate Q1 (25th percentile) and Q3 (75th percentile):
Q1 <- quantile(merged_data$fatalities, 0.25, na.rm = TRUE)
Q3 <- quantile(merged_data$fatalities, 0.75, na.rm = TRUE)

# Calculate IQR (Interquartile Range)
IQR <- Q3 - Q1

# Define outlier fence:
lower_fence <- Q1 - 1.5 * IQR
upper_fence <- Q3 + 1.5 * IQR

# Print number of outliers:
cat("Lower Fence for Outliers:", lower_fence, "\n")

## Lower Fence for Outliers: 0

cat("Upper Fence for Outliers:", upper_fence, "\n")

## Upper Fence for Outliers: 0

# Find outliers:
outliers <- merged_data %>%
  filter(fatalities < lower_fence | fatalities > upper_fence)

# Count the number of outliers:
num_outliers <- nrow(outliers)
cat("Number of Outliers:", num_outliers, "\n")

## Number of Outliers: 2789
```

```r
# Analyse outliers:
summary(outliers$fatalities)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.078   1.000   5.000

outliers %<>%
  arrange(desc(fatalities))
head(outliers$fatalities)

## [1] 5 5 5 4 4 4
```

The values appear realistic, and no additional action is needed for the outliers in `merged_data$fatalities`.

7.1.3 Outlier Detection ***serious_injuries***

```r
# Calculate Q1 (25th percentile) and Q3 (75th percentile):
Q1 <- quantile(merged_data$serious_injuries, 0.25, na.rm = TRUE)
Q3 <- quantile(merged_data$serious_injuries, 0.75, na.rm = TRUE)

# Calculate IQR (Interquartile Range)
IQR <- Q3 - Q1

# Define outlier fence:
lower_fence <- Q1 - 1.5 * IQR
upper_fence <- Q3 + 1.5 * IQR

# Print number of outliers:
cat("Lower Fence for Outliers:", lower_fence, "\n")

## Lower Fence for Outliers: -1.5

cat("Upper Fence for Outliers:", upper_fence, "\n")

## Upper Fence for Outliers: 2.5

# Find outliers:
outliers <- merged_data %>%
  filter(serious_injuries < lower_fence | serious_injuries > upper_fence)

# Count the number of outliers:
num_outliers <- nrow(outliers)
cat("Number of Outliers:", num_outliers, "\n")

## Number of Outliers: 1467

# Analyse outliers:
summary(outliers$serious_injuries)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   3.000   3.000   3.462   4.000  16.000
```

```
outliers %<>%
  arrange(desc(serious_injuries))
head(outliers$serious_injuries)

## [1] 16 15 10  9  8  8
```

The values appear realistic, and no additional action is needed for the outliers in `merged_data$serious_injuries`.

*7.1.4 Outlier Detection **minor_injuries***

```
# Calculate Q1 (25th percentile) and Q3 (75th percentile):
Q1 <- quantile(merged_data$minor_injuries, 0.25, na.rm = TRUE)
Q3 <- quantile(merged_data$minor_injuries, 0.75, na.rm = TRUE)

# Calculate IQR (Interquartile Range)
IQR <- Q3 - Q1

# Define outlier fence:
lower_fence <- Q1 - 1.5 * IQR
upper_fence <- Q3 + 1.5 * IQR

# Print number of outliers:
cat("Lower Fence for Outliers:", lower_fence, "\n")

## Lower Fence for Outliers: -1.5

cat("Upper Fence for Outliers:", upper_fence, "\n")

## Upper Fence for Outliers: 2.5

# Find outliers:
outliers <- merged_data %>%
  filter(minor_injuries < lower_fence | minor_injuries > upper_fence)

# Count the number of outliers:
num_outliers <- nrow(outliers)
cat("Number of Outliers:", num_outliers, "\n")

## Number of Outliers: 5150

# Analyse outliers:
summary(outliers$minor_injuries)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   3.000   3.000   3.000   3.541   4.000  43.000

outliers %<>%
  arrange(desc(minor_injuries))
head(outliers$minor_injuries)

## [1] 43 38 28 25 23 23
```

The values appear realistic, and no additional action is needed for the outliers in merged_data$minor_injuries.

*7.1.5 Outlier Detection **uninjured_people***

```
# Calculate Q1 (25th percentile) and Q3 (75th percentile):
Q1 <- quantile(merged_data$uninjured_people, 0.25, na.rm = TRUE)
Q3 <- quantile(merged_data$uninjured_people, 0.75, na.rm = TRUE)

# Calculate IQR (Interquartile Range)
IQR <- Q3 - Q1

# Define outlier fence:
lower_fence <- Q1 - 1.5 * IQR
upper_fence <- Q3 + 1.5 * IQR

# Print number of outliers:
cat("Lower Fence for Outliers:", lower_fence, "\n")

## Lower Fence for Outliers: -1.5

cat("Upper Fence for Outliers:", upper_fence, "\n")

## Upper Fence for Outliers: 2.5

# Find outliers:
outliers <- merged_data %>%
  filter(uninjured_people < lower_fence | uninjured_people > upper_fence)

# Count the number of outliers:
num_outliers <- nrow(outliers)
cat("Number of Outliers:", num_outliers, "\n")

## Number of Outliers: 13850

# Analyse outliers:
summary(outliers$uninjured_people)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00    3.00    3.00    3.89    4.00   87.00

outliers %<>%
  arrange(desc(uninjured_people))
head(outliers$uninjured_people)

## [1] 87 87 47 41 41 41
```

The values appear realistic, and no additional action is needed for the outliers in merged_data$uninjured_people.

*7.1.6 Outlier Detection **total_people_involved***

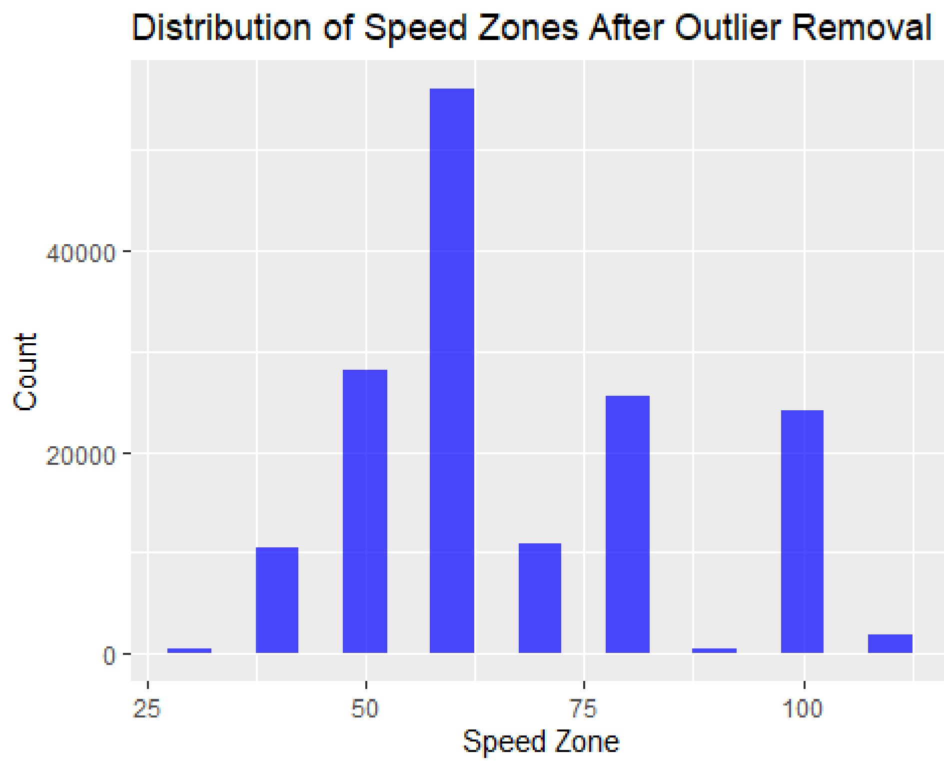
```
# Calculate Q1 (25th percentile) and Q3 (75th percentile):
Q1 <- quantile(merged_data$total_people_involved, 0.25, na.rm = TRUE)
Q3 <- quantile(merged_data$total_people_involved, 0.75, na.rm = TRUE)

# Calculate IQR (Interquartile Range)
IQR <- Q3 - Q1

# Define outlier fence:
lower_fence <- Q1 - 1.5 * IQR
upper_fence <- Q3 + 1.5 * IQR

# Print number of outliers:
cat("Lower Fence for Outliers:", lower_fence, "\n")

## Lower Fence for Outliers: 0.5

cat("Upper Fence for Outliers:", upper_fence, "\n")

## Upper Fence for Outliers: 4.5

# Find outliers:
outliers <- merged_data %>%
  filter(total_people_involved < lower_fence | total_people_involved > 
upper_fence)

# Count the number of outliers:
num_outliers <- nrow(outliers)
cat("Number of Outliers:", num_outliers, "\n")

## Number of Outliers: 11026

# Analyse outliers:
summary(outliers$total_people_involved)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   5.000   5.000   5.000   6.106   6.000  97.000

outliers %<>%
  arrange(desc(total_people_involved))
head(outliers$total_people_involved)

## [1] 97 89 54 51 50 48
```

The values appear realistic, and no additional action is needed for the outliers in `merged_data$total_people_involved`.

*7.1.7 Outlier Detection **speed_zone**:*

In this section, we will compare the number of rows in the dataset before and after the removal of outliers in the speed_zone variable. We will also summarize the changes.

```r
# Calculate Q1 (25th percentile) and Q3 (75th percentile):
Q1 <- quantile(merged_data$speed_zone, 0.25, na.rm = TRUE)
Q3 <- quantile(merged_data$speed_zone, 0.75, na.rm = TRUE)

# Calculate IQR (Interquartile Range)
IQR <- Q3 - Q1

# Define outlier fence:
lower_fence <- Q1 - 1.5 * IQR
upper_fence <- Q3 + 1.5 * IQR

# Print number of outliers:
cat("Lower Fence for Outliers:", lower_fence, "\n")

## Lower Fence for Outliers: 30

cat("Upper Fence for Outliers:", upper_fence, "\n")

## Upper Fence for Outliers: 110

# Find outliers:
outliers <- merged_data %>%
  filter(speed_zone < lower_fence | speed_zone > upper_fence)

# Count the number of outliers:
num_outliers <- nrow(outliers)
cat("Number of Outliers:", num_outliers, "\n")

## Number of Outliers: 12140

# Examine outliers
print(table(outliers$speed_zone))

##
##   777   888   999
##   308  1055 10777

# Remove outliers
merged_data_clean <- merged_data %>%
  filter(speed_zone >= lower_fence & speed_zone <= upper_fence)
# 7. Visualize the speed_zone distribution after removing outliers
ggplot(merged_data_clean, aes(x = speed_zone)) +
  geom_histogram(binwidth = 5, fill = "blue", alpha = 0.7) +
  labs(title = "Distribution of Speed Zones After Outlier Removal",
       x = "Speed Zone", y = "Count")
```




```
# 8. Compare before and after
cat("Rows before outlier removal:", nrow(merged_data), "\n")

## Rows before outlier removal: 169877

cat("Rows after outlier removal:", nrow(merged_data_clean), "\n")

## Rows after outlier removal: 157737

cat("Rows removed:", nrow(merged_data) - nrow(merged_data_clean), "\n")

## Rows removed: 12140

# 9. Summary of clean data
summary(merged_data_clean$speed_zone)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   30.00   60.00   60.00   67.54   80.00  110.00

merged_data <- merged_data_clean
```

For the speed_zone variable, we identified outliers above 110 km/h. These values are inconsistent with Victoria's standard speed limits. We removed a total of 12140 outliers as they likely represent data entry errors.

For all other outliers, we chose to retain them. Despite their appearance as outliers on the boxplot, these values remain realistic. This is because, in most accidents, the number of vehicles or persons involved is typically low, which aligns with the observed data distribution.

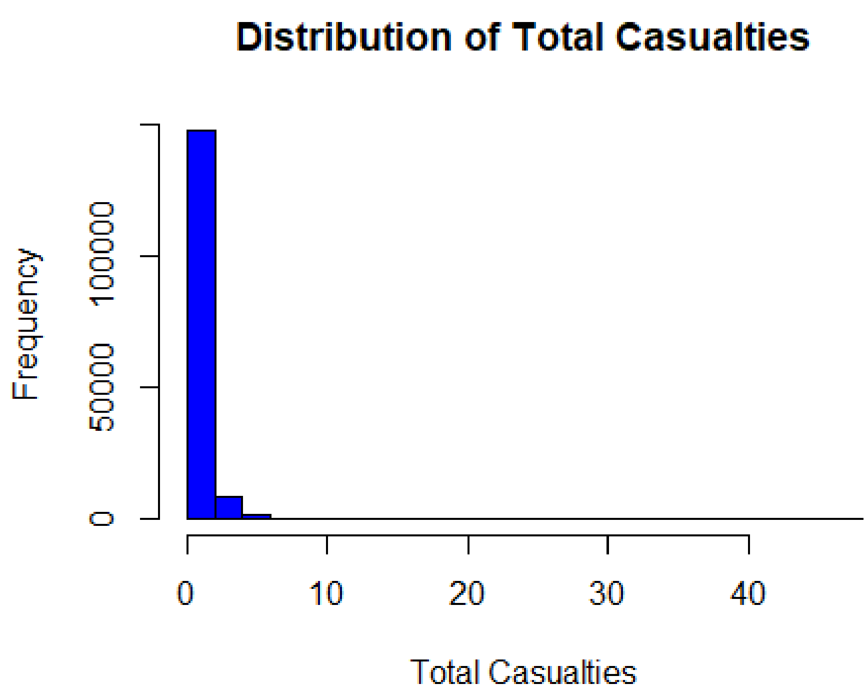
8. Transform the Data

In this section, we will explore the distribution of numeric variables and apply transformations to normalize the data if necessary.

8.1 Histogram Visualization

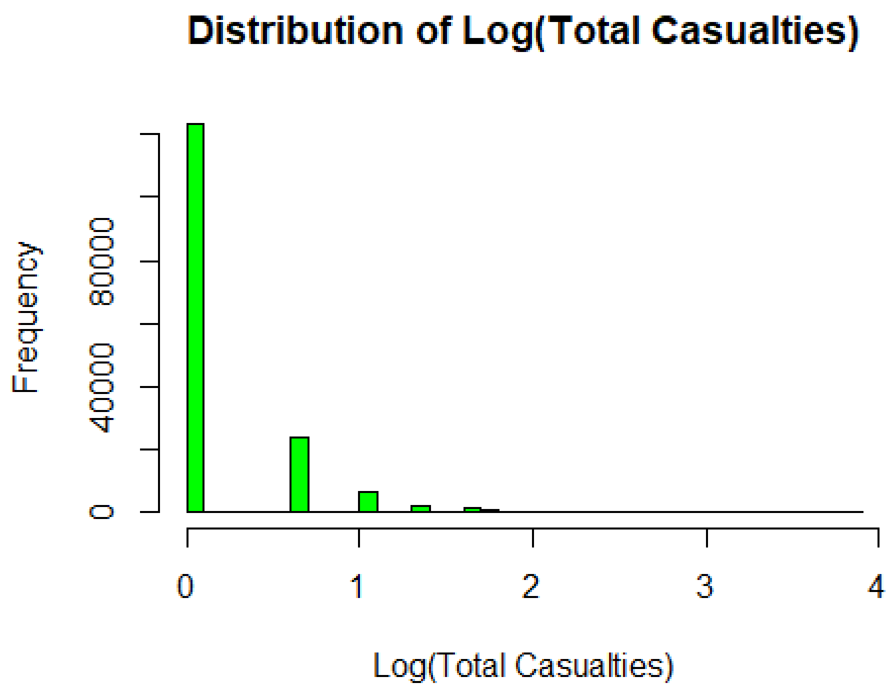
We will plot histograms for the total_casualties and no_of_vehicles columns before and after applying a log transformation.

```
# Create histograms for 'total_casualties' before and after log
transformation

# Original distribution of 'total_casualties'
p1<- hist(merged_data$total_casualties,
     breaks = 30,
     main = "Distribution of Total Casualties",
     xlab = "Total Casualties",
     col = "blue",
     border = "black")
```

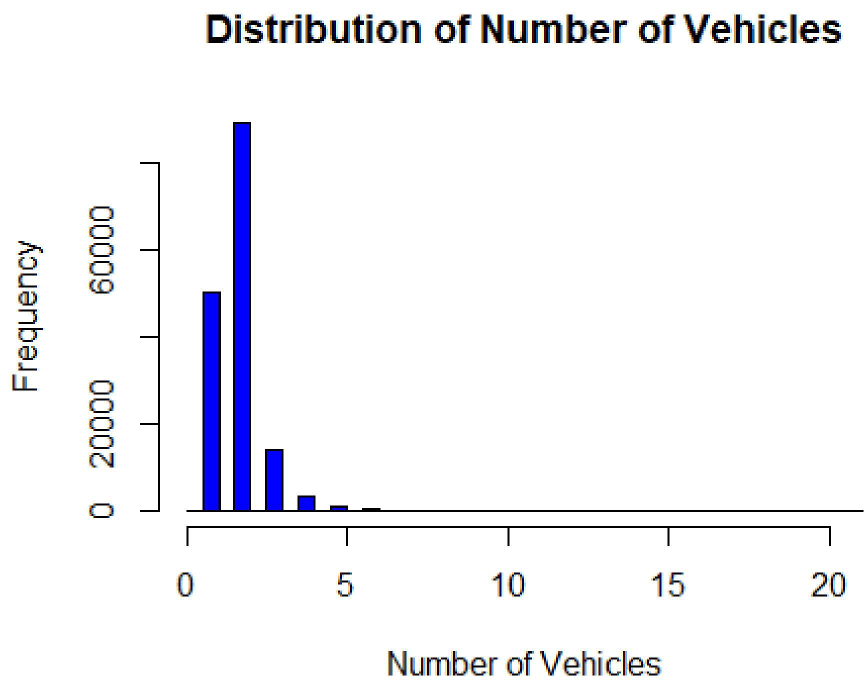



```
# Distribution of log-transformed 'total_casualties'
p2<- hist(log(merged_data$total_casualties),
     breaks = 30,
```

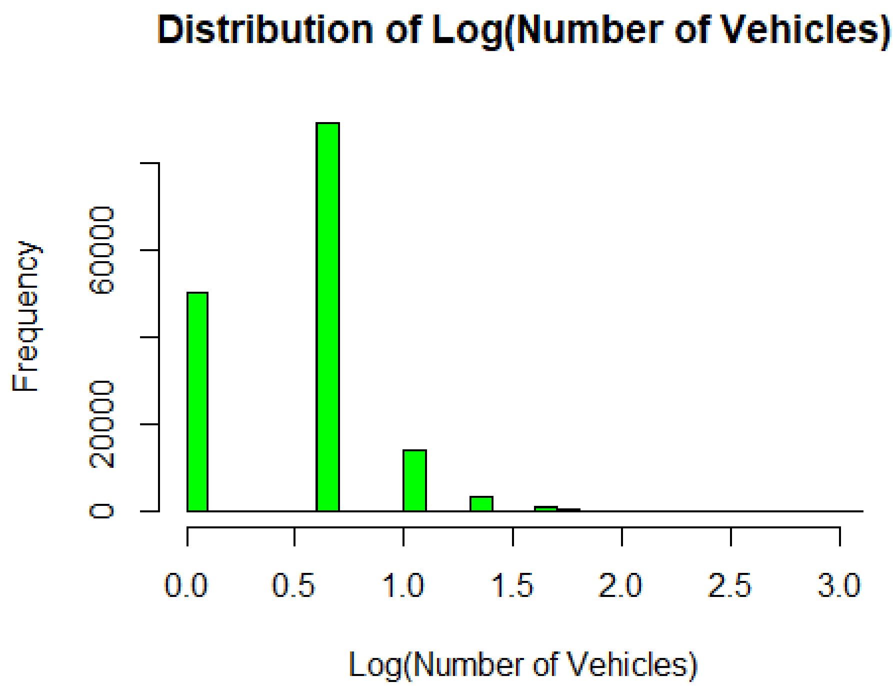
```
     main = "Distribution of Log(Total Casualties)",
     xlab = "Log(Total Casualties)",
     col = "green",
     border = "black")
```




```
# Create histograms for 'no_of_vehicles' before and after log transformation

# Original distribution of 'no_of_vehicles'
p1<- hist(merged_data$no_of_vehicles,
     breaks = 30,
     main = "Distribution of Number of Vehicles",
     xlab = "Number of Vehicles",
     col = "blue",
     border = "black")
```




```
# Distribution of log-transformed 'no_of_vehicles'
p2<- hist(log(merged_data$no_of_vehicles),
     breaks = 30,
     main = "Distribution of Log(Number of Vehicles)",
     xlab = "Log(Number of Vehicles)",
     col = "green",
     border = "black")
```



The log transformation of `no_of_vehicles` and `total_casualties` helps normalize the data distribution. This transformation improves the interpretation of relationships between variables by stabilizing variance and making the distribution more symmetrical.

9. Presentation link

https://www.loom.com/share/09e8b3ec0782438ebd5ba8afc8db0248?sid=764b1c4c-f4a2-473d-9a54-faf9d25f6109

10. References

[01]Department of Transport (2024) *Victoria Road Crash Data*, Discover Victoria's Open Data website, accessed 8 August 2024. https://discover.data.vic.gov.au/dataset/victoria-road-crash-data

[02]Iglewicz B, Hoaglin DC (1993) How to detect and handle outliers, Wiley, New York.