Project_1

2024-11-13

```r
# Load necessary libraries
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.4.2
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2

## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(fastDummies)
```

```
## Warning: package 'fastDummies' was built under R version 4.4.2
```

```r
library(ggplot2)
library(VIM)
```

```
## Warning: package 'VIM' was built under R version 4.4.2

## Loading required package: colorspace

## Loading required package: grid
```

```
## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
library(e1071)  # For SVM and Naive Bayes models
```

```
## Warning: package 'e1071' was built under R version 4.4.2
```

```r
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.4.2

## corrplot 0.95 loaded
```

```r
# Load data from "Normalized_Data" sheet
file_path <- "Mine_Dataset.xls"
data <- read_excel(file_path, sheet = "Normalized_Data")

# Step 1: Map values in columns 'M' and 'S'

# Map values in column 'S' to categorical descriptions
data <- data %>%
  mutate(
    S = case_when(
      S == 0   ~ "dry and sandy",
      S == 0.2 ~ "dry and humus",
      S == 0.4 ~ "dry and limy",
      S == 0.6 ~ "humid and sandy",
      S == 0.8 ~ "humid and humus",
      S == 1   ~ "humid and limy",
      TRUE     ~ "undefined"
    ),
    S = factor(S)
  )

# Map values in column 'M' to target categories
data <- data %>%
```

```r
  mutate(
    M_category = case_when(
      M == 1 ~ "Null",
      M == 2 ~ "Anti-Tank",
      M == 3 ~ "Anti-personnel",
      M == 4 ~ "Booby Trapped Anti-personnel",
      M == 5 ~ "M14 Anti-personnel",
      TRUE   ~ "undefined"
    ),
    M_category = factor(M_category)
  )
head(data)
```

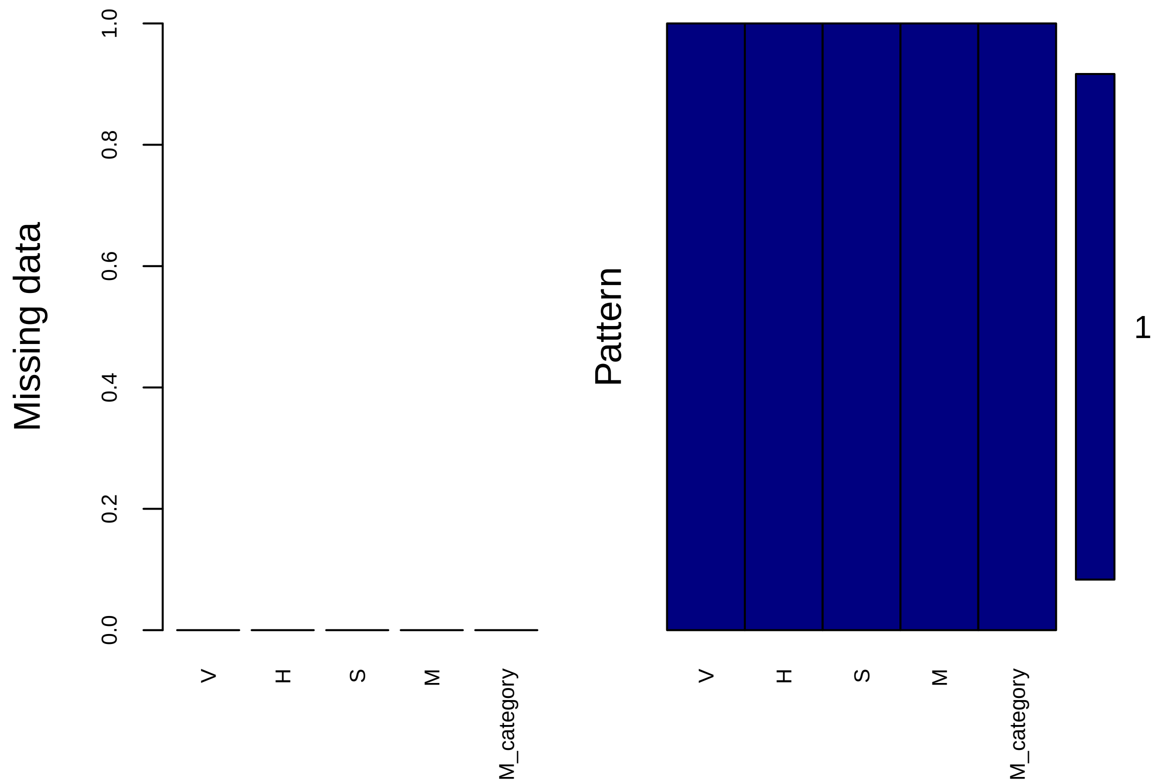
```
## # A tibble: 6 x 5
##       V     H S                  M M_category
##   <dbl> <dbl> <fct>          <dbl> <fct>
## 1 0.338 0     dry and sandy      1 Null
## 2 0.320 0.182 dry and sandy      1 Null
## 3 0.287 0.273 dry and sandy      1 Null
## 4 0.256 0.455 dry and sandy      1 Null
## 5 0.263 0.545 dry and sandy      1 Null
## 6 0.241 0.727 dry and sandy      1 Null
```

Step 2: Data Cleaning

```r
## 1. Remove duplicates
data <- data %>% distinct()

## 2. Check for missing values
missing_values <- sapply(data, function(x) sum(is.na(x)))
print(missing_values)
```

```
##          V          H          S          M M_category
##          0          0          0          0          0
```

```r
# Visualize missing data
aggr(data, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE, labels=names(data), cex.axis=.7, gap=3
```

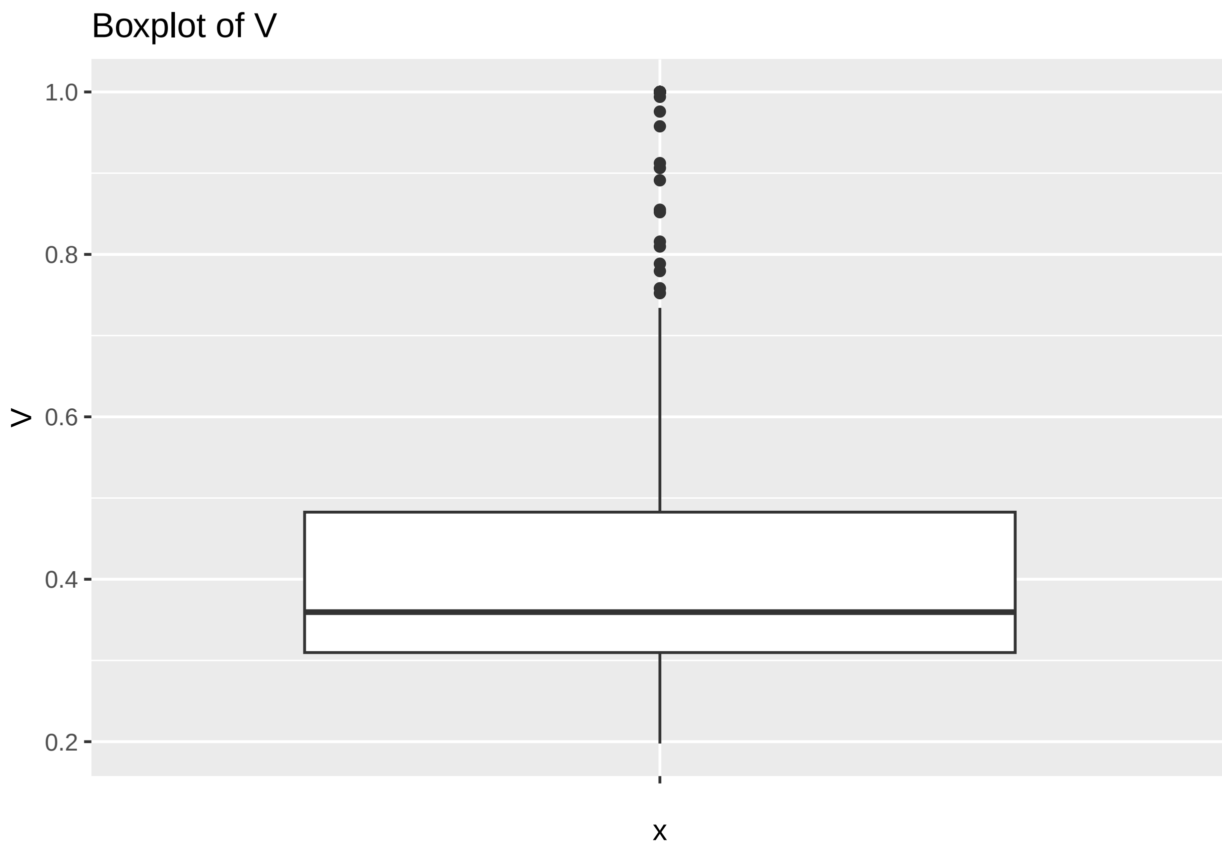



```
##
## Variables sorted by number of missings:
##    Variable Count
##           V     0
##           H     0
##           S     0
##           M     0
##  M_category     0
```

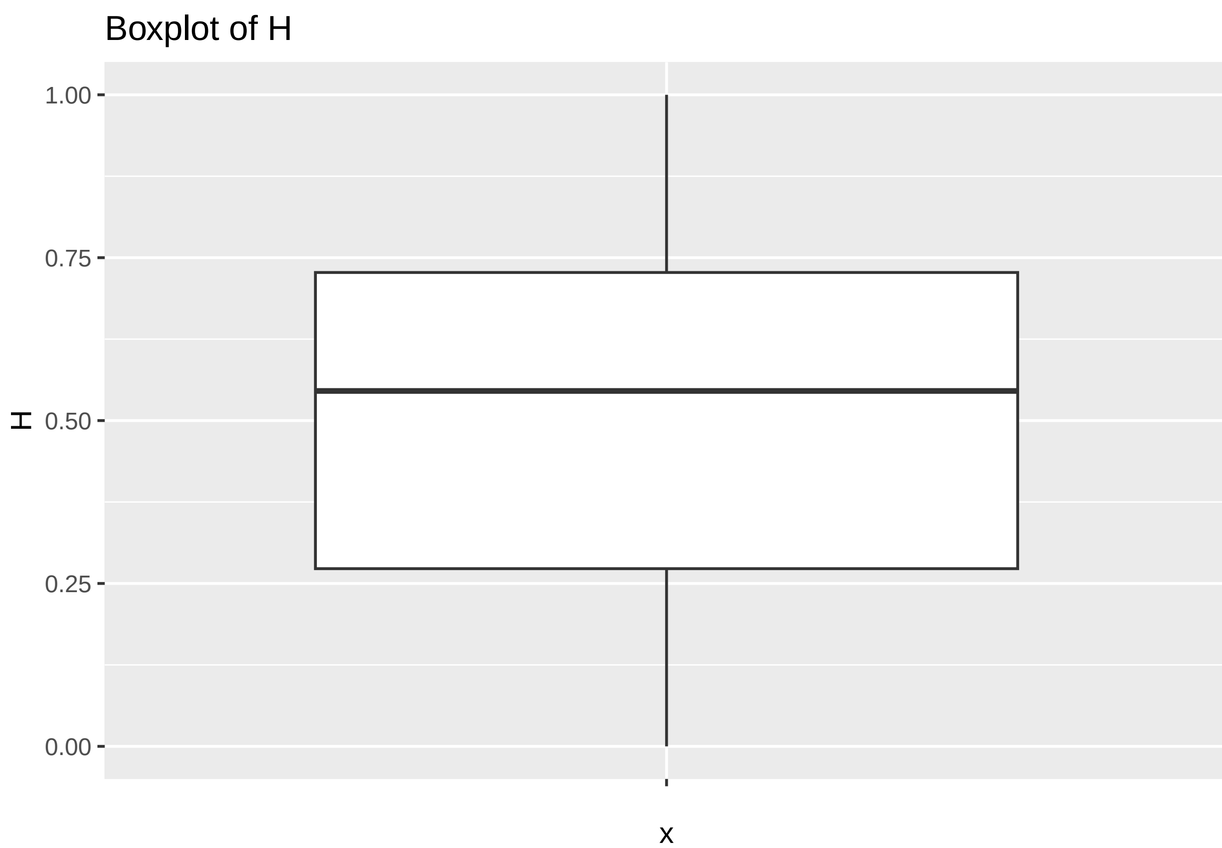
```r
## 3. Handle categorical variables (Create dummy variables for 'S' and remove the original 'S' and 'M')
data <- dummy_cols(data, select_columns = "S", remove_first_dummy = TRUE)
data <- data %>% select(-S, -M)

## 4. Check data structure
str(data)
```

```
## tibble [338 x 8] (S3: tbl_df/tbl/data.frame)
##  $ V                 : num [1:338] 0.338 0.32 0.287 0.256 0.263 ...
##  $ H                 : num [1:338] 0 0.182 0.273 0.455 0.545 ...
##  $ M_category        : Factor w/ 5 levels "Anti-personnel",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ S_dry and limy    : int [1:338] 0 0 0 0 0 0 0 0 0 0 ...
##  $ S_dry and sandy   : int [1:338] 1 1 1 1 1 1 1 1 0 0 ...
##  $ S_humid and humus: int [1:338] 0 0 0 0 0 0 0 0 0 0 ...
##  $ S_humid and limy : int [1:338] 0 0 0 0 0 0 0 0 0 0 ...
##  $ S_humid and sandy: int [1:338] 0 0 0 0 0 0 0 0 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
## 5. Scale numeric columns (e.g., V and H) for models that require scaling
data_scaled <- data %>%
  mutate(across(c(V, H), scale))

## 6. Visualize outliers using boxplots for numeric features
ggplot(data, aes(x = "", y = V)) +
  geom_boxplot() +
  labs(title = "Boxplot of V")
```




```r
ggplot(data, aes(x = "", y = H)) +
  geom_boxplot() +
  labs(title = "Boxplot of H")
```




```
## Alternatively, calculate Z-scores for outlier detection
data <- data %>%
  mutate(
    V_z = (V - mean(V)) / sd(V),
    H_z = (H - mean(H)) / sd(H)
  )

## 7. Create interaction term between V and H (optional feature engineering)
data$V_H_interaction <- data$V * data$H


## 10. Final Check for data structure and summary
str(data)
```

```
## tibble [338 x 11] (S3: tbl_df/tbl/data.frame)
##  $ V                 : num [1:338] 0.338 0.32 0.287 0.256 0.263 ...
##  $ H                 : num [1:338] 0 0.182 0.273 0.455 0.545 ...
##  $ M_category        : Factor w/ 5 levels "Anti-personnel",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ S_dry and limy    : int [1:338] 0 0 0 0 0 0 0 0 0 0 ...
##  $ S_dry and sandy   : int [1:338] 1 1 1 1 1 1 1 1 0 0 ...
##  $ S_humid and humus: int [1:338] 0 0 0 0 0 0 0 0 0 0 ...
##  $ S_humid and limy : int [1:338] 0 0 0 0 0 0 0 0 0 0 ...
##  $ S_humid and sandy: int [1:338] 0 0 0 0 0 0 0 0 1 1 ...
##  $ V_z               : num [1:338] -0.472 -0.564 -0.733 -0.89 -0.857 ...
##  $ H_z               : num [1:338] -1.663 -1.069 -0.772 -0.178 0.12 ...
##  $ V_H_interaction   : num [1:338] 0 0.0582 0.0783 0.1165 0.1434 ...
```

```
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
summary(data)
```

```
##        V                H                                       M_category
##  Min.   :0.1977   Min.   :0.0000   Anti-personnel                :66
##  1st Qu.:0.3097   1st Qu.:0.2727   Anti-Tank                     :70
##  Median :0.3595   Median :0.5455   Booby Trapped Anti-personnel:66
##  Mean   :0.4306   Mean   :0.5089   M14 Anti-personnel            :65
##  3rd Qu.:0.4826   3rd Qu.:0.7273   Null                          :71
##  Max.   :1.0000   Max.   :1.0000
##  S_dry and limy   S_dry and sandy  S_humid and humus S_humid and limy
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000    Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000    1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000    Median :0.0000
##  Mean   :0.1657   Mean   :0.1746   Mean   :0.1716    Mean   :0.1686
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000    3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000    Max.   :1.0000
##  S_humid and sandy      V_z               H_z          V_H_interaction
##  Min.   :0.0000    Min.   :-1.1894   Min.   :-1.6628   Min.   :0.0000
##  1st Qu.:0.0000    1st Qu.:-0.6174   1st Qu.:-0.7716   1st Qu.:0.1009
##  Median :0.0000    Median :-0.3632   Median : 0.1195   Median :0.1880
##  Mean   :0.1686    Mean   : 0.0000   Mean   : 0.0000   Mean   :0.1966
##  3rd Qu.:0.0000    3rd Qu.: 0.2655   3rd Qu.: 0.7136   3rd Qu.:0.2743
##  Max.   :1.0000    Max.   : 2.9076   Max.   : 1.6048   Max.   :0.6218
```

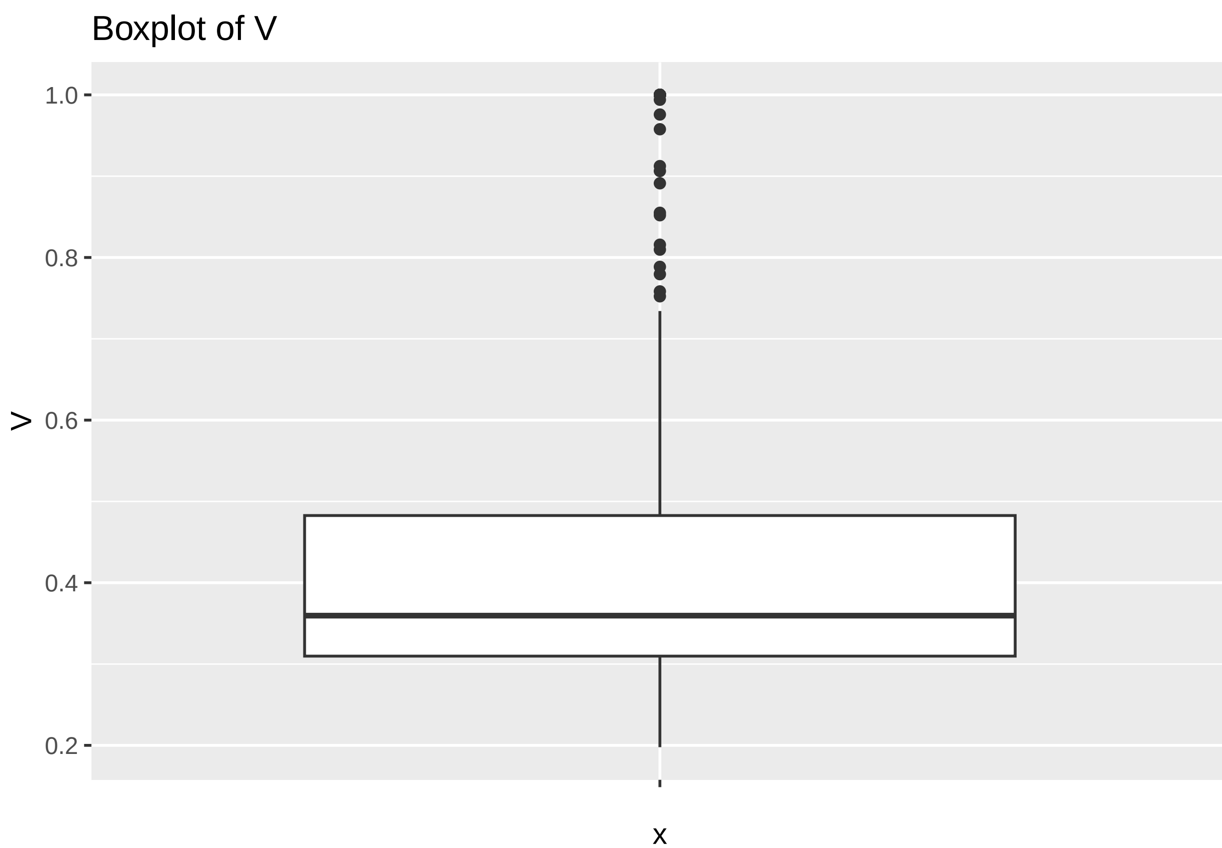
Step 3: Exploratory Data Analysis (EDA)

```r
# Summary statistics
summary(data)
```

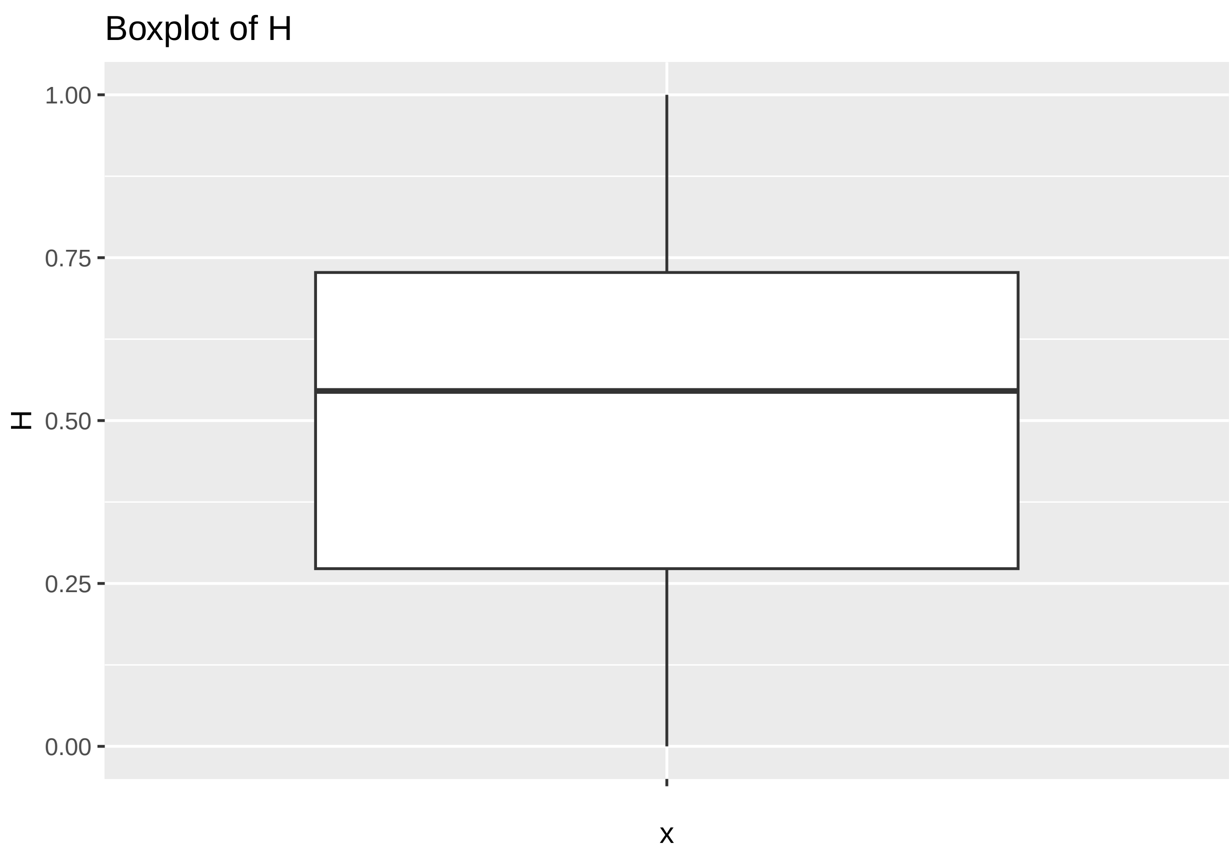
```
##        V                H                                       M_category
##  Min.   :0.1977   Min.   :0.0000   Anti-personnel                :66
##  1st Qu.:0.3097   1st Qu.:0.2727   Anti-Tank                     :70
##  Median :0.3595   Median :0.5455   Booby Trapped Anti-personnel:66
##  Mean   :0.4306   Mean   :0.5089   M14 Anti-personnel            :65
##  3rd Qu.:0.4826   3rd Qu.:0.7273   Null                          :71
##  Max.   :1.0000   Max.   :1.0000
##  S_dry and limy   S_dry and sandy  S_humid and humus S_humid and limy
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000    Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000    1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000    Median :0.0000
##  Mean   :0.1657   Mean   :0.1746   Mean   :0.1716    Mean   :0.1686
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000    3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000    Max.   :1.0000
##  S_humid and sandy      V_z               H_z          V_H_interaction
##  Min.   :0.0000    Min.   :-1.1894   Min.   :-1.6628   Min.   :0.0000
##  1st Qu.:0.0000    1st Qu.:-0.6174   1st Qu.:-0.7716   1st Qu.:0.1009
##  Median :0.0000    Median :-0.3632   Median : 0.1195   Median :0.1880
##  Mean   :0.1686    Mean   : 0.0000   Mean   : 0.0000   Mean   :0.1966
```

```
##  3rd Qu.:0.0000    3rd Qu.: 0.2655   3rd Qu.: 0.7136   3rd Qu.:0.2743
##  Max.   :1.0000    Max.   : 2.9076   Max.   : 1.6048   Max.   :0.6218
```

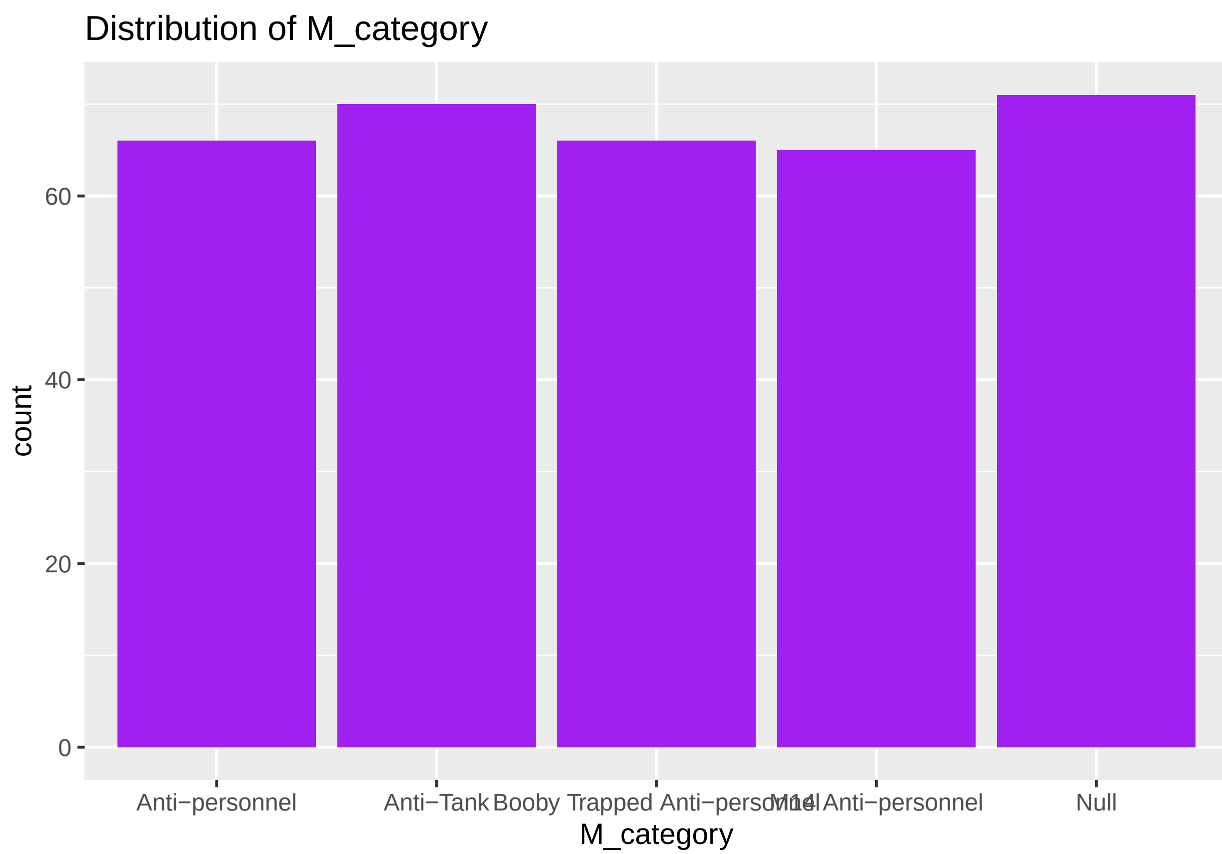
```r
# Boxplots to detect outliers for 'V' and 'H'
ggplot(data, aes(x = "", y = V)) + geom_boxplot() + labs(title = "Boxplot of V")
```



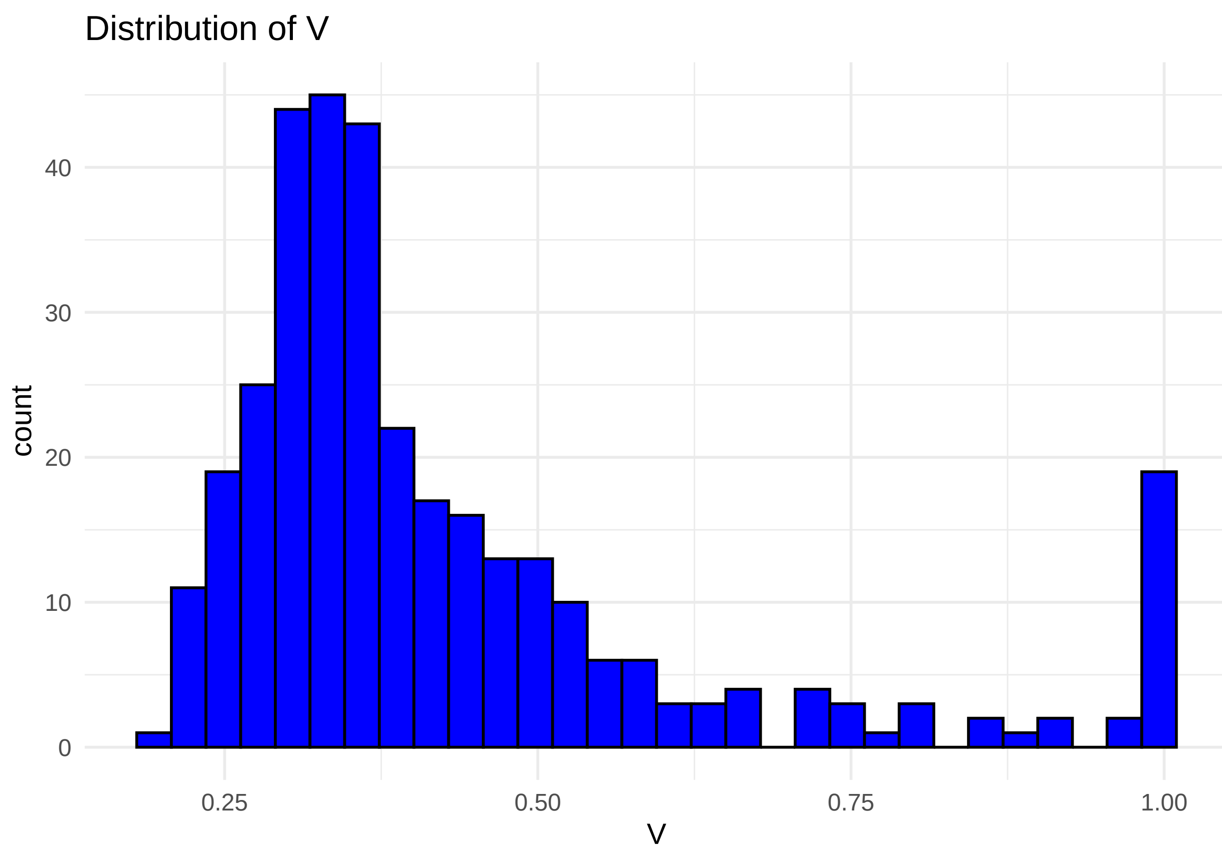

```r
ggplot(data, aes(x = "", y = H)) + geom_boxplot() + labs(title = "Boxplot of H")
```



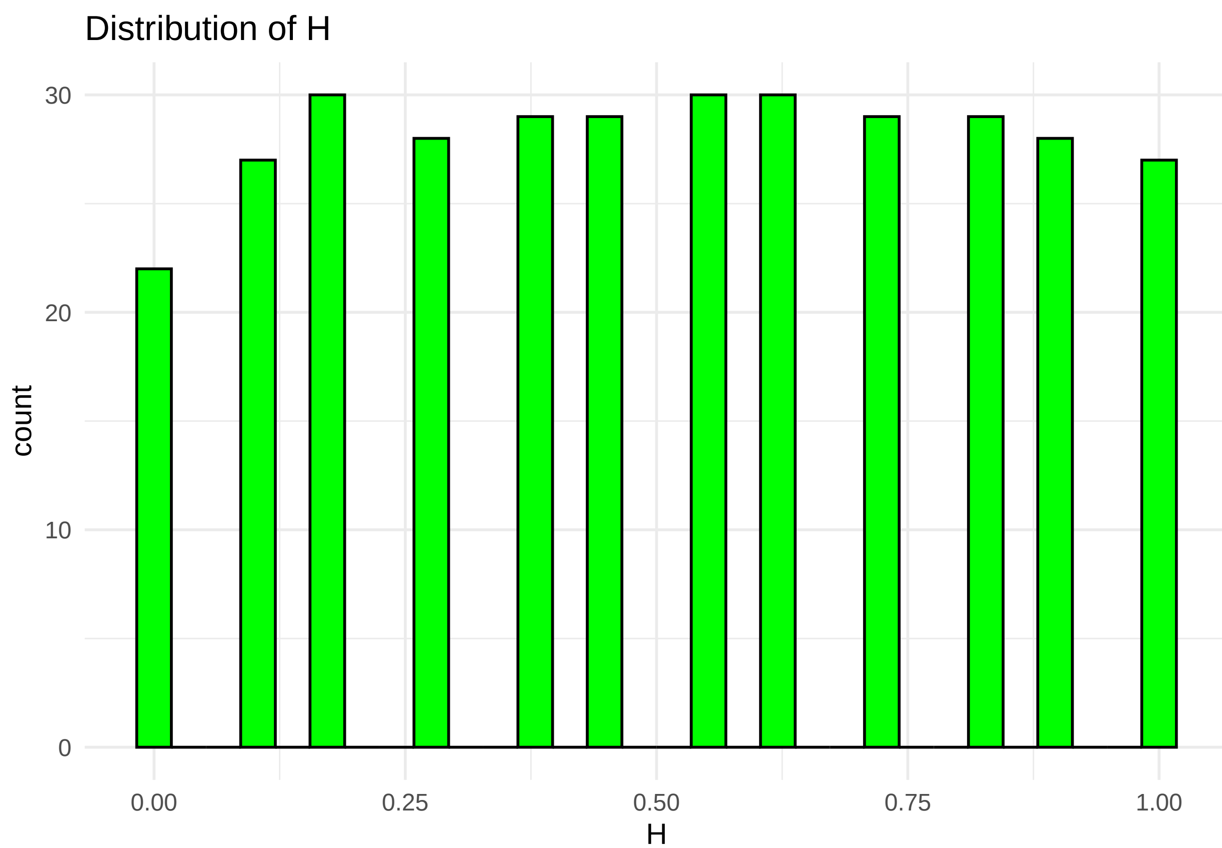

```
# Bar plot for 'M_category'
ggplot(data, aes(x = M_category)) + geom_bar(fill = "purple") + labs(title = "Distribution of M_categor
```



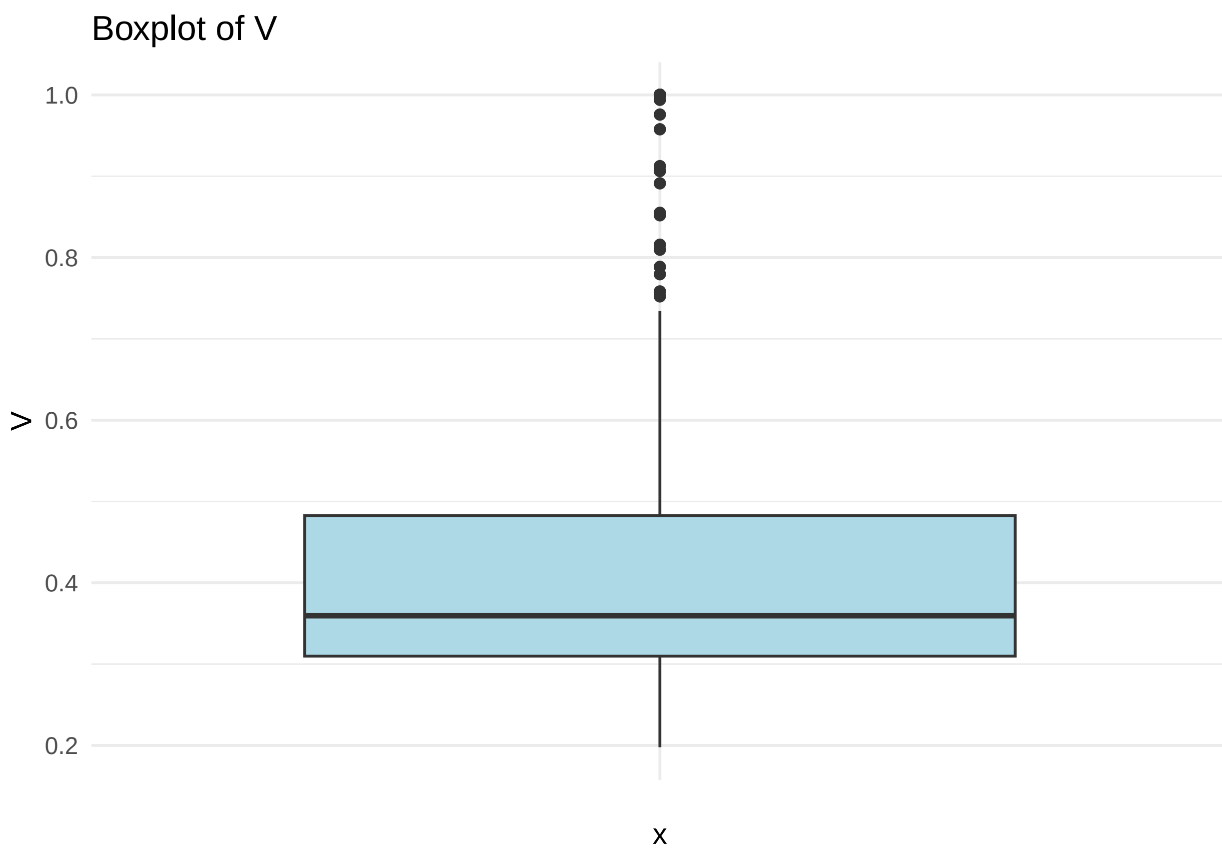

```r
# Distribution plots for numeric columns 'V' and 'H'
ggplot(data, aes(x = V)) +
  geom_histogram(bins = 30, fill = "blue", color = "black") +
  labs(title = "Distribution of V") +
  theme_minimal()
```



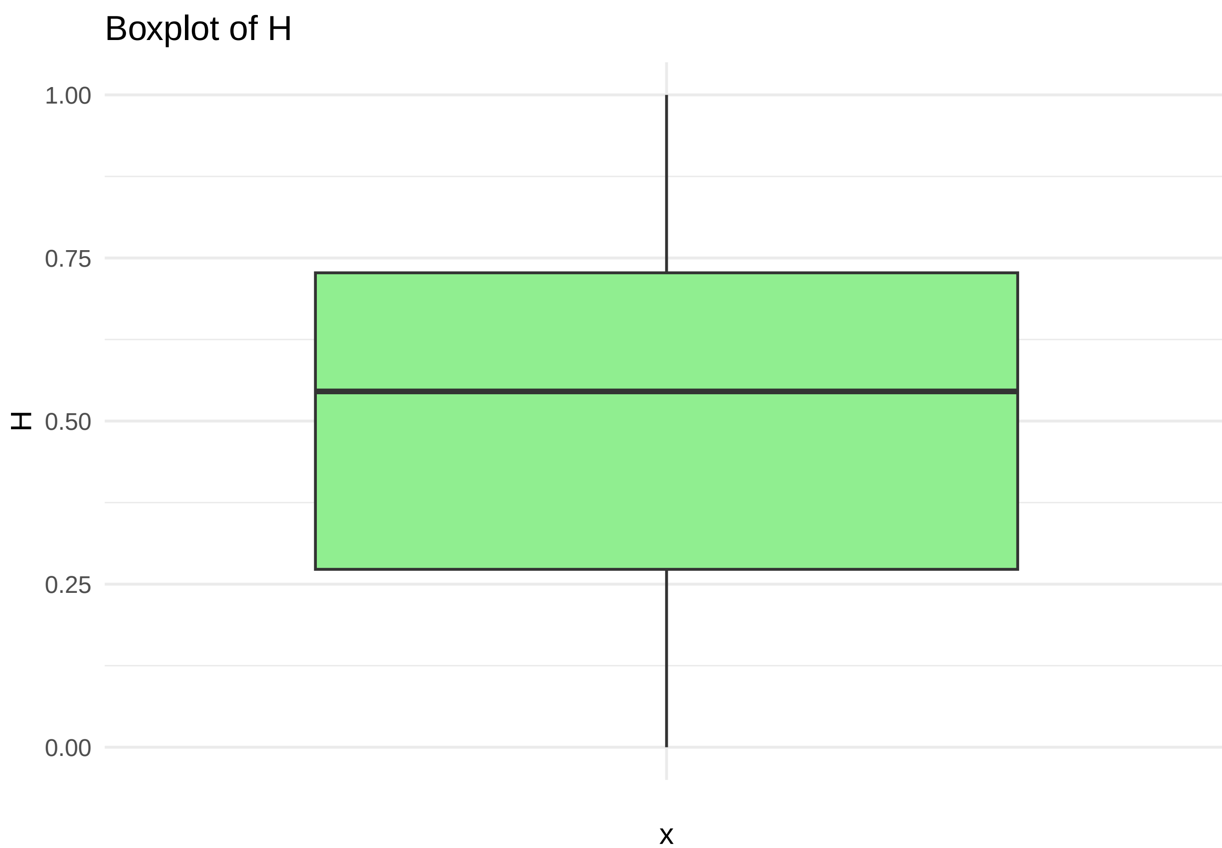

```r
ggplot(data, aes(x = H)) +
  geom_histogram(bins = 30, fill = "green", color = "black") +
  labs(title = "Distribution of H") +
  theme_minimal()
```



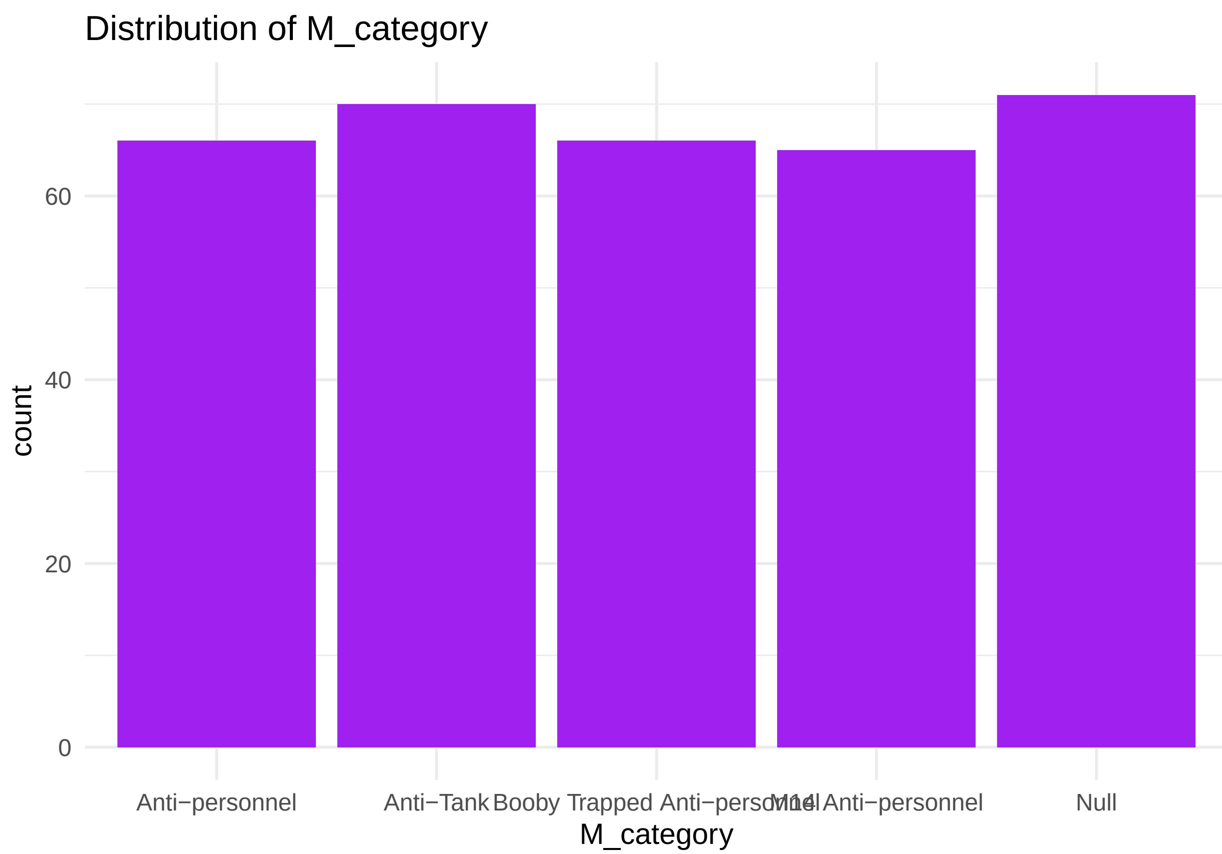

```r
# Boxplots to detect outliers for 'V' and 'H'
ggplot(data, aes(x = "", y = V)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "Boxplot of V") +
  theme_minimal()
```



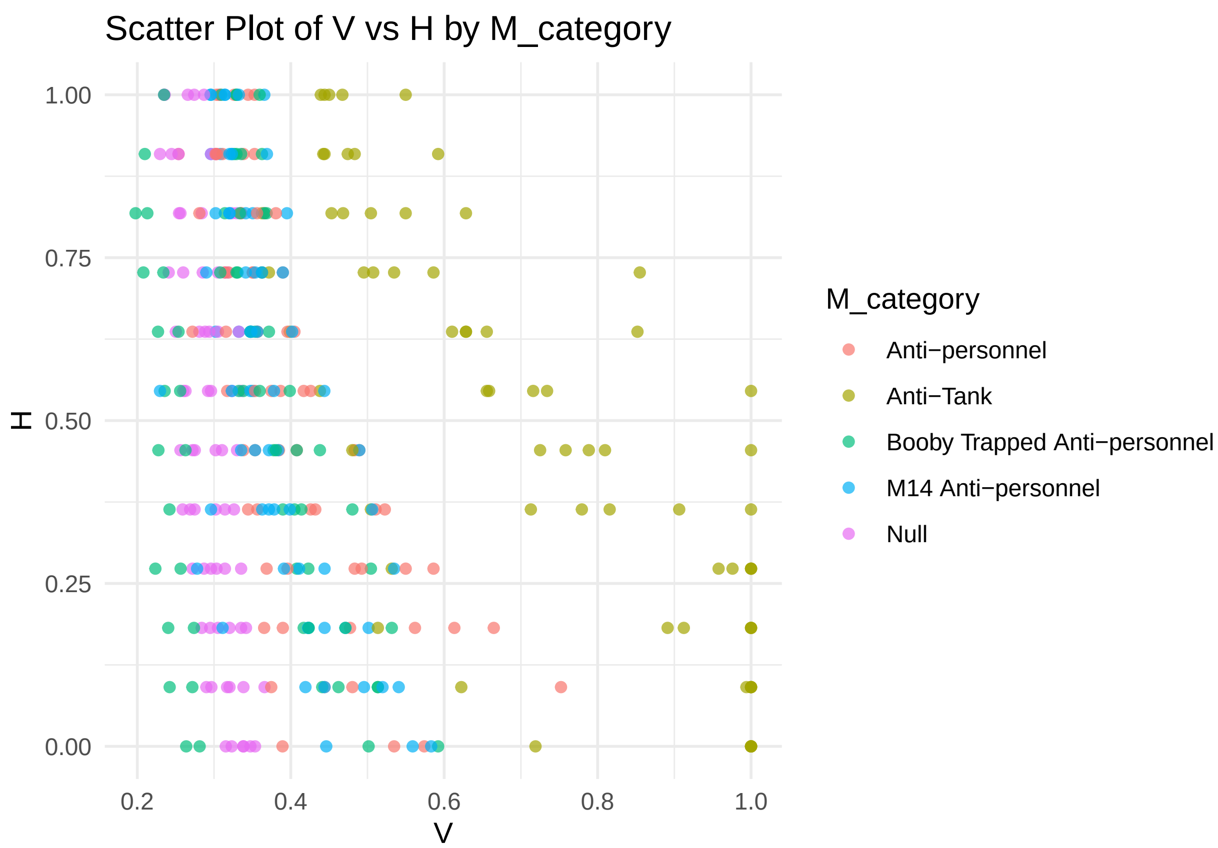

```
ggplot(data, aes(x = "", y = H)) +
  geom_boxplot(fill = "lightgreen") +
  labs(title = "Boxplot of H") +
  theme_minimal()
```



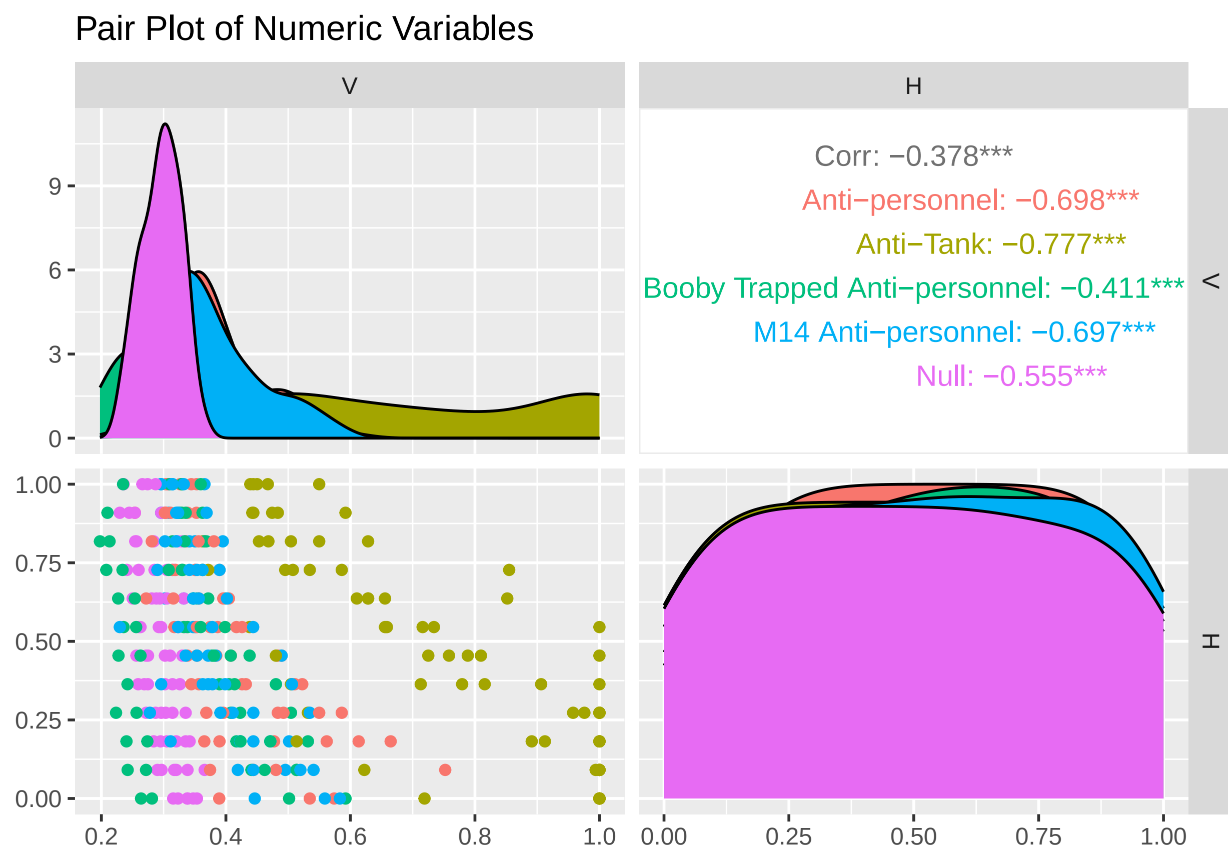

```r
# Bar plot for 'M_category' to show class distribution
ggplot(data, aes(x = M_category)) +
  geom_bar(fill = "purple") +
  labs(title = "Distribution of M_category") +
  theme_minimal()
```




```r
# Scatter plot for V vs H colored by M_category
ggplot(data, aes(x = V, y = H, color = M_category)) +
  geom_point(alpha = 0.7) +
  labs(title = "Scatter Plot of V vs H by M_category") +
  theme_minimal()
```




```r
# Pair plot for V and H, colored by M_category
ggpairs(data, columns = c("V", "H"), aes(color = M_category)) +
  labs(title = "Pair Plot of Numeric Variables")
```



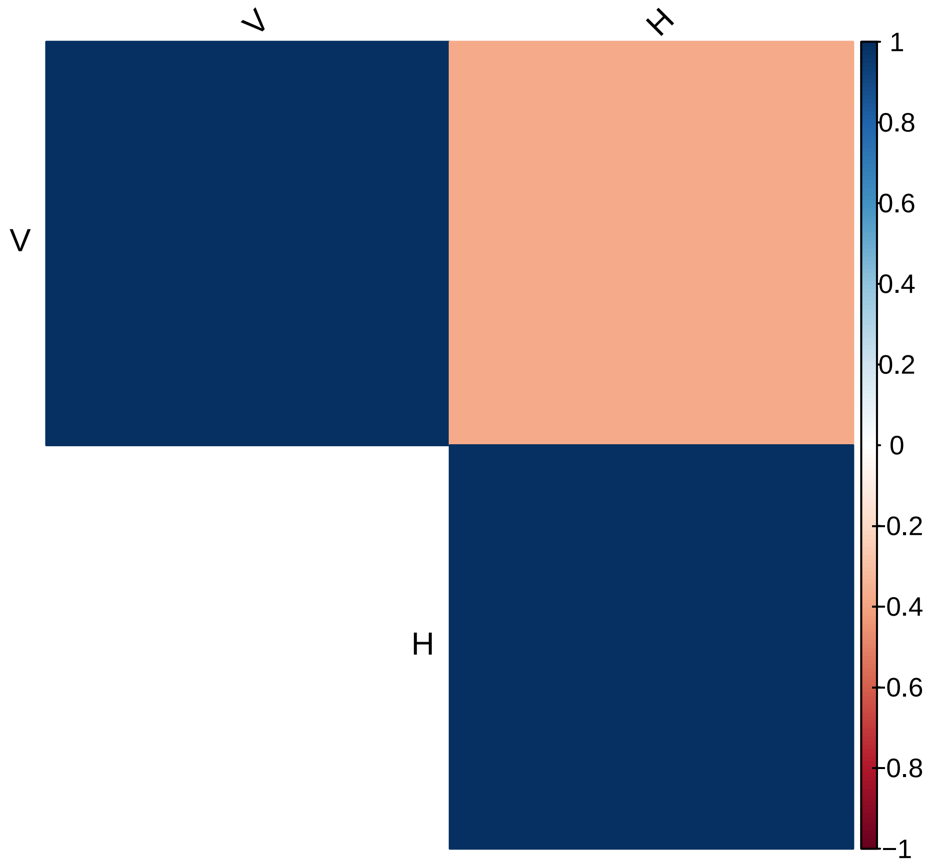

```
# Outlier detection using Z-scores for V and H
data <- data %>%
  mutate(
    V_z = (V - mean(V, na.rm = TRUE)) / sd(V, na.rm = TRUE),
    H_z = (H - mean(H, na.rm = TRUE)) / sd(H, na.rm = TRUE)
  )

# Display potential outliers (absolute Z-score > 3)
outliers <- data %>% filter(abs(V_z) > 3 | abs(H_z) > 3)
print("Potential outliers based on Z-scores:")
```

```
## [1] "Potential outliers based on Z-scores:"
```

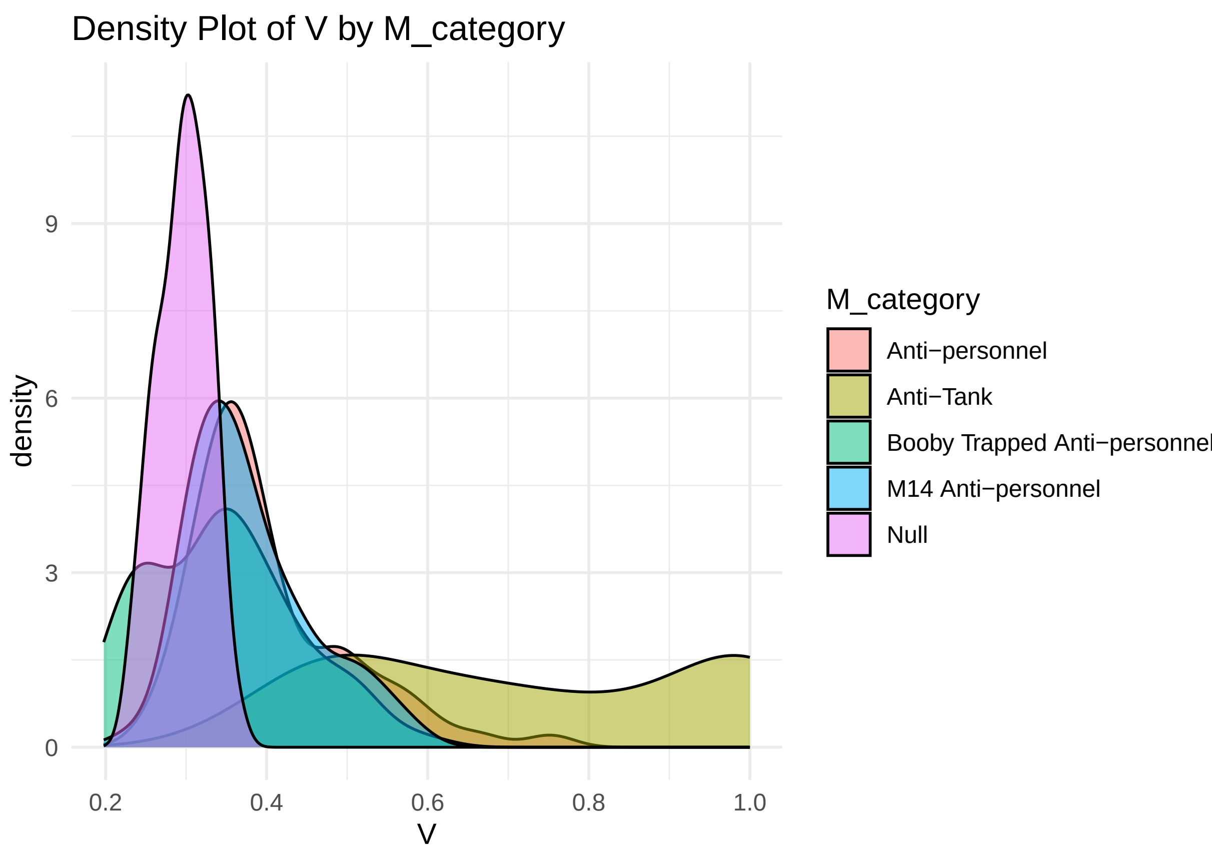
```
print(outliers)
```

```
## # A tibble: 0 x 11
## # i 11 variables: V <dbl>, H <dbl>, M_category <fct>, S_dry and limy <int>,
## #   S_dry and sandy <int>, S_humid and humus <int>, S_humid and limy <int>,
## #   S_humid and sandy <int>, V_z <dbl>, H_z <dbl>, V_H_interaction <dbl>
```

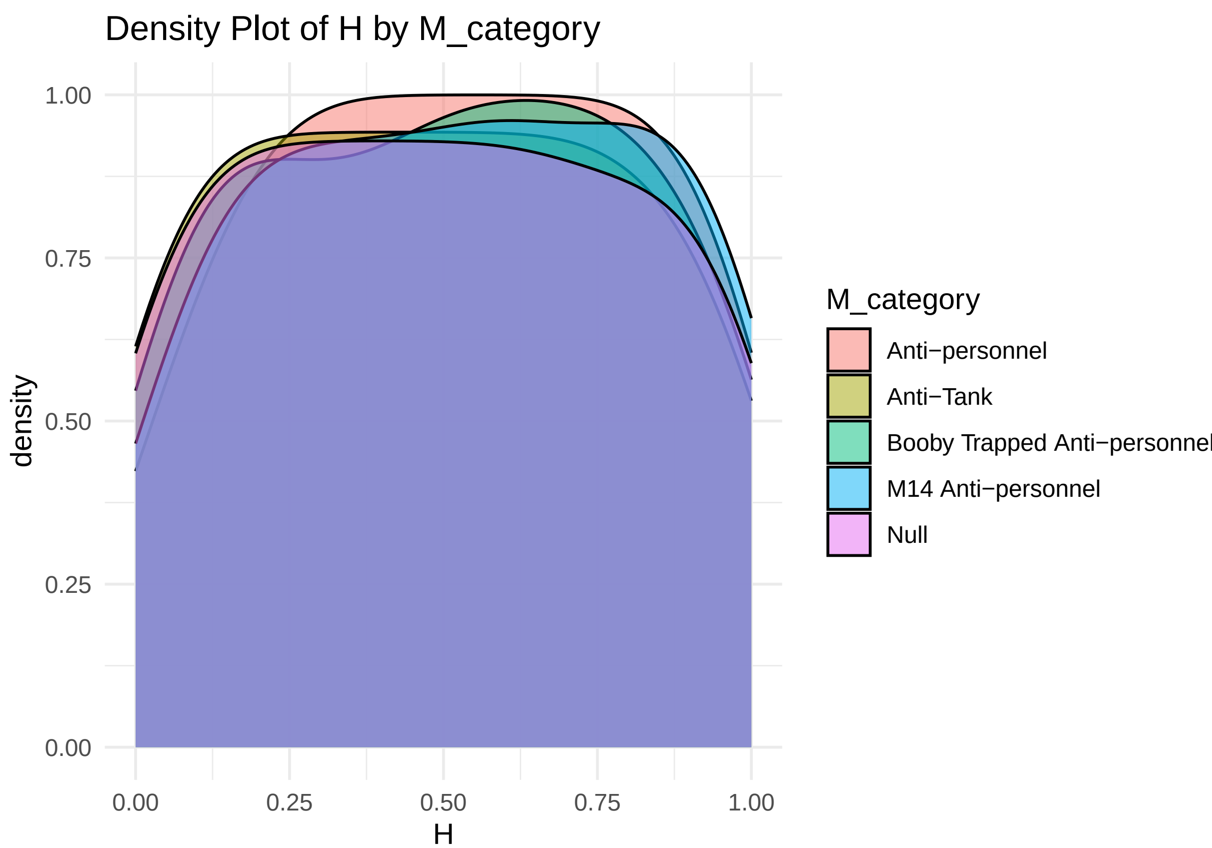
```
# Correlation matrix and heatmap for numeric variables
numeric_data <- data %>% select(V, H)
cor_matrix <- cor(numeric_data)
corrplot(cor_matrix, method = "color", type = "upper", tl.col = "black", tl.srt = 45)
```



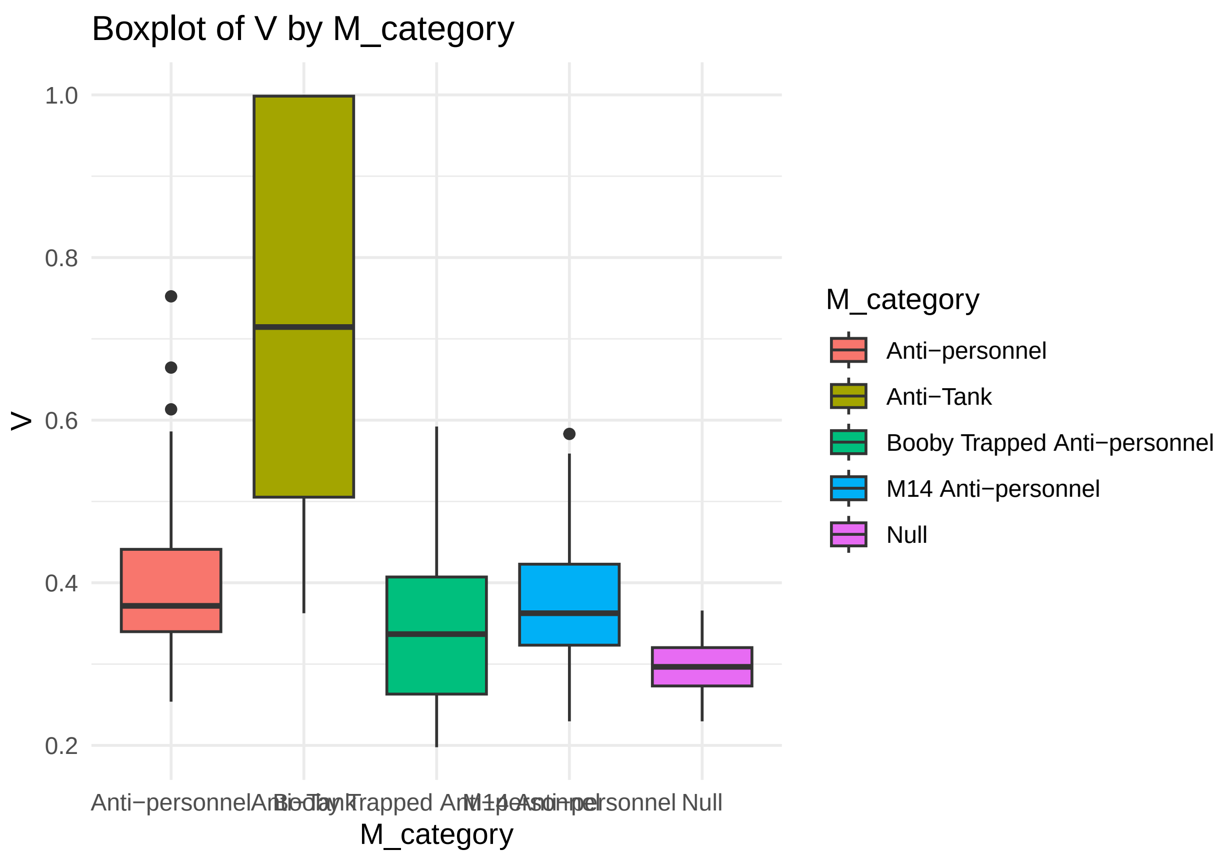

```r
# Density plots for V and H, colored by M_category
ggplot(data, aes(x = V, fill = M_category)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of V by M_category") +
  theme_minimal()
```



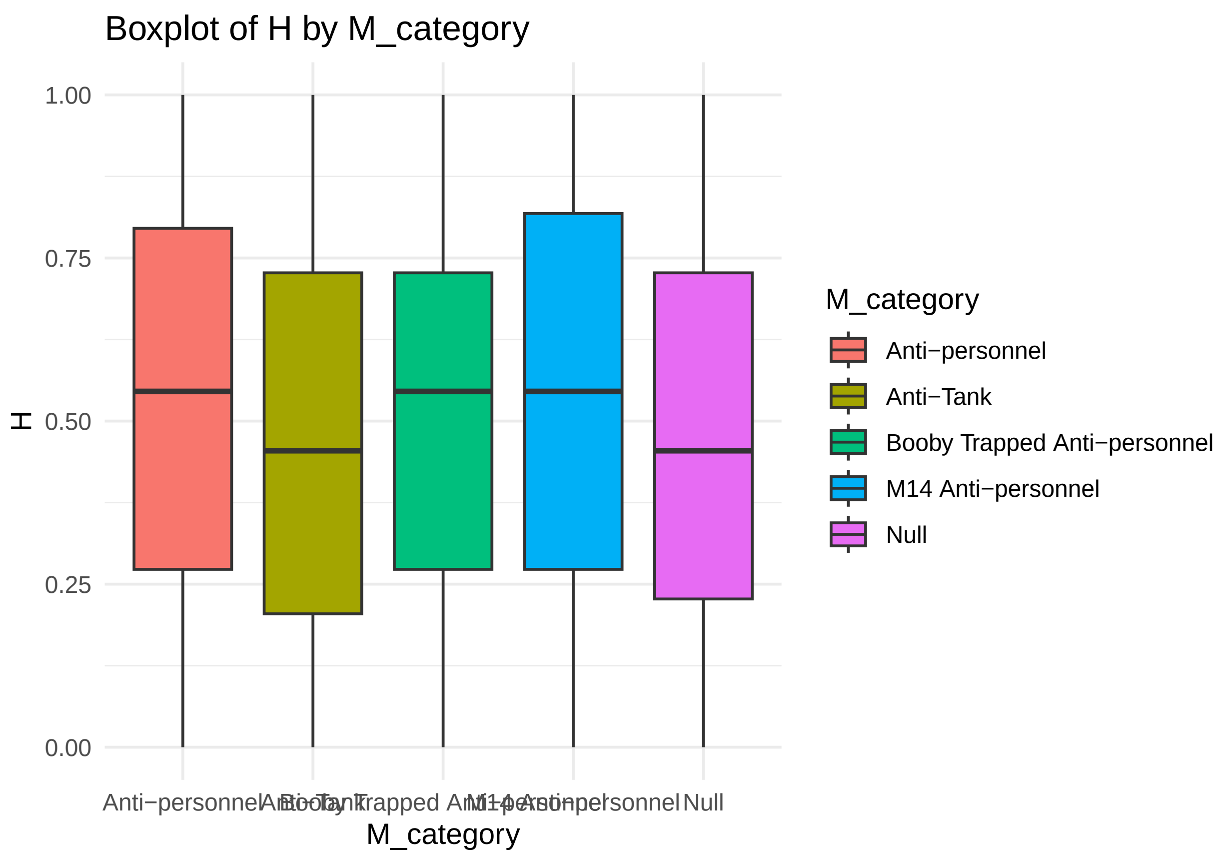

```r
ggplot(data, aes(x = H, fill = M_category)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of H by M_category") +
  theme_minimal()
```




```r
# Boxplots of V and H grouped by M_category
ggplot(data, aes(x = M_category, y = V, fill = M_category)) +
  geom_boxplot() +
  labs(title = "Boxplot of V by M_category") +
  theme_minimal()
```




```r
ggplot(data, aes(x = M_category, y = H, fill = M_category)) +
  geom_boxplot() +
  labs(title = "Boxplot of H by M_category") +
  theme_minimal()
```




```
# Summary statistics grouped by M_category for V and H
grouped_summary <- data %>%
  group_by(M_category) %>%
  summarise(
    V_mean = mean(V, na.rm = TRUE),
    V_sd = sd(V, na.rm = TRUE),
    H_mean = mean(H, na.rm = TRUE),
    H_sd = sd(H, na.rm = TRUE)
  )
print("Summary statistics for V and H by M_category:")
```

```
## [1] "Summary statistics for V and H by M_category:"
```

```
print(grouped_summary)
```

```
## # A tibble: 5 x 5
##   M_category                   V_mean   V_sd H_mean  H_sd
##   <fct>                         <dbl>  <dbl>  <dbl> <dbl>
## 1 Anti-personnel                0.402 0.0982  0.528 0.296
## 2 Anti-Tank                     0.721 0.222   0.487 0.311
## 3 Booby Trapped Anti-personnel  0.345 0.0926  0.507 0.306
## 4 M14 Anti-personnel            0.380 0.0763  0.530 0.306
## 5 Null                          0.296 0.0317  0.496 0.316
```

Step 4: Data Modeling

```r
# Split data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(data$M_category, p = 0.7, list = FALSE)
train_data <- data[trainIndex, ]
test_data <- data[-trainIndex, ]
# Replace spaces with underscores in column names
names(train_data) <- gsub(" ", "_", names(train_data))
names(test_data) <- gsub(" ", "_", names(test_data))

# Re-usable formula
train_formula <- M_category ~ V + H + S_dry_and_limy + S_dry_and_sandy + S_humid_and_humus + S_humid_anc

print(head(train_data))
```

```
## # A tibble: 6 x 11
##        V     H M_category S_dry_and_limy S_dry_and_sandy S_humid_and_humus
##    <dbl> <dbl> <fct>               <int>           <int>             <int>
## 1 0.338 0     Null                    0               1                 0
## 2 0.320 0.182 Null                    0               1                 0
## 3 0.287 0.273 Null                    0               1                 0
## 4 0.256 0.455 Null                    0               1                 0
## 5 0.263 0.545 Null                    0               1                 0
## 6 0.241 0.727 Null                    0               1                 0
## # i 5 more variables: S_humid_and_limy <int>, S_humid_and_sandy <int>,
## #   V_z <dbl>, H_z <dbl>, V_H_interaction <dbl>
```

```r
print(head(test_data))
```

```
## # A tibble: 6 x 11
##       V     H M_category S_dry_and_limy S_dry_and_sandy S_humid_and_humus
##   <dbl> <dbl> <fct>               <int>           <int>             <int>
## 1 0.235 1     Null                    0               1                 0
## 2 0.330 0.455 Null                    0               0                 0
## 3 0.335 0.545 Null                    0               0                 0
## 4 0.256 0.818 Null                    0               0                 0
## 5 0.236 1     Null                    0               0                 0
## 6 0.284 0.182 Null                    0               0                 0
## # i 5 more variables: S_humid_and_limy <int>, S_humid_and_sandy <int>,
## #   V_z <dbl>, H_z <dbl>, V_H_interaction <dbl>
```

```r
# Define training control for cross-validation
train_control <- trainControl(method = "cv", number = 10)

# Train models

# Logistic Regression
log_model <- train(train_formula, data = train_data, method = "multinom", trControl = train_control)
```

```
## # weights:  45 (32 variable)
```

```
## initial  value 344.419713
## iter  10 value 237.112852
## iter  20 value 203.330668
## iter  30 value 197.349830
## iter  40 value 196.974478
## iter  50 value 196.436250
## iter  60 value 196.416757
## final  value 196.416139
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 277.119749
## iter  20 value 271.844706
## final  value 271.838281
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 237.202033
## iter  20 value 203.779297
## iter  30 value 198.107570
## iter  40 value 197.821251
## iter  50 value 197.513325
## iter  60 value 197.505531
## final  value 197.505528
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 231.604124
## iter  20 value 199.469242
## iter  30 value 196.344337
## iter  40 value 196.073425
## iter  50 value 195.774008
## iter  60 value 195.770271
## final  value 195.770260
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 272.385020
## iter  20 value 270.434855
## iter  30 value 270.430281
## iter  30 value 270.430279
## iter  30 value 270.430278
## final  value 270.430278
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 231.681555
## iter  20 value 199.907816
## iter  30 value 197.098579
## iter  40 value 196.915550
## iter  50 value 196.750005
## final  value 196.747394
## converged
## # weights:  45 (32 variable)
```

```
## initial  value 347.638589
## iter  10 value 240.050427
## iter  20 value 206.071892
## iter  30 value 201.660402
## iter  40 value 201.201155
## iter  50 value 200.800082
## iter  60 value 200.794682
## iter  70 value 200.789707
## iter  80 value 200.788774
## iter  90 value 200.788459
## iter 100 value 200.787625
## final  value 200.787625
## stopped after 100 iterations
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 274.019716
## iter  20 value 272.116524
## iter  30 value 272.111753
## iter  30 value 272.111751
## iter  30 value 272.111751
## final  value 272.111751
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 240.114789
## iter  20 value 206.493285
## iter  30 value 202.327383
## iter  40 value 201.982506
## iter  50 value 201.753944
## iter  60 value 201.750601
## iter  70 value 201.747786
## final  value 201.747737
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 231.548040
## iter  20 value 200.660644
## iter  30 value 197.892111
## iter  40 value 197.492379
## iter  50 value 197.160435
## iter  60 value 197.157312
## final  value 197.157295
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 274.761138
## iter  20 value 272.874052
## final  value 272.867943
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 231.639444
## iter  20 value 201.236743
## iter  30 value 198.661225
```

```
## iter  40 value 198.382792
## iter  50 value 198.194643
## final  value 198.192789
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 229.709685
## iter  20 value 195.013087
## iter  30 value 192.148035
## iter  40 value 191.594041
## iter  50 value 191.180264
## iter  60 value 191.155518
## final  value 191.154724
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 271.986183
## iter  20 value 269.609687
## iter  30 value 269.605001
## final  value 269.604991
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 229.796319
## iter  20 value 195.644482
## iter  30 value 192.976880
## iter  40 value 192.594261
## iter  50 value 192.393007
## iter  60 value 192.385270
## final  value 192.385259
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 229.496486
## iter  20 value 196.863929
## iter  30 value 192.309922
## iter  40 value 191.915151
## iter  50 value 191.385629
## iter  60 value 191.375152
## final  value 191.375057
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 273.928114
## iter  20 value 271.736179
## iter  30 value 271.727092
## final  value 271.727066
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 229.593301
## iter  20 value 197.383671
## iter  30 value 193.162096
## iter  40 value 192.882221
```

```
## iter  50 value 192.590237
## iter  60 value 192.587824
## final  value 192.587691
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 247.268021
## iter  20 value 207.671129
## iter  30 value 203.261851
## iter  40 value 202.799141
## iter  50 value 202.392205
## iter  60 value 202.382867
## final  value 202.382822
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 278.160890
## iter  20 value 275.870951
## iter  30 value 275.862205
## iter  30 value 275.862203
## iter  30 value 275.862202
## final  value 275.862202
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 247.427793
## iter  20 value 208.124488
## iter  30 value 203.956302
## iter  40 value 203.608820
## iter  50 value 203.377531
## iter  60 value 203.372129
## iter  60 value 203.372127
## iter  60 value 203.372127
## final  value 203.372127
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 233.488404
## iter  20 value 203.112631
## iter  30 value 199.742460
## iter  40 value 199.356777
## iter  50 value 198.996834
## iter  60 value 198.986268
## final  value 198.986181
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 277.369102
## iter  20 value 275.669665
## final  value 275.664579
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 233.589233
```

```
## iter  20 value 203.672085
## iter  30 value 200.470216
## iter  40 value 200.187111
## iter  50 value 199.979581
## iter  60 value 199.974317
## iter  60 value 199.974316
## iter  60 value 199.974316
## final  value 199.974316
## converged
## # weights:  45 (32 variable)
## initial  value 349.248027
## iter  10 value 236.365031
## iter  20 value 202.445460
## iter  30 value 198.661072
## iter  40 value 198.197179
## iter  50 value 197.763096
## iter  60 value 197.751750
## final  value 197.751351
## converged
## # weights:  45 (32 variable)
## initial  value 349.248027
## iter  10 value 278.811295
## iter  20 value 274.769488
## iter  30 value 274.759169
## iter  30 value 274.759168
## iter  30 value 274.759168
## final  value 274.759168
## converged
## # weights:  45 (32 variable)
## initial  value 349.248027
## iter  10 value 236.453825
## iter  20 value 202.995948
## iter  30 value 199.481817
## iter  40 value 199.169221
## iter  50 value 198.960444
## iter  60 value 198.956550
## final  value 198.956547
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 230.809915
## iter  20 value 199.393691
## iter  30 value 195.878723
## iter  40 value 195.206862
## iter  50 value 194.651098
## iter  60 value 194.648277
## final  value 194.648184
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 274.068315
## iter  20 value 271.930338
## iter  30 value 271.925132
## final  value 271.925116
```

```
## converged
## # weights:  45 (32 variable)
## initial  value 344.419713
## iter  10 value 230.900703
## iter  20 value 199.920353
## iter  30 value 196.650257
## iter  40 value 196.146113
## iter  50 value 195.837623
## iter  60 value 195.834957
## final  value 195.834925
## converged
## # weights:  45 (32 variable)
## initial  value 384.655661
## iter  10 value 256.185819
## iter  20 value 226.286432
## iter  30 value 221.953767
## iter  40 value 221.517390
## iter  50 value 221.311064
## final  value 221.308366
## converged
```

```r
# Decision Tree
tree_model <- train(train_formula, data = train_data, method = "rpart", trControl = train_control)

# Random Forest
rf_model <- train(train_formula, data = train_data, method = "rf", trControl = train_control)

# K-Nearest Neighbors (KNN)
knn_model <- train(train_formula, data = train_data, method = "knn", trControl = train_control)

# Naive Bayes
nb_model <- train(train_formula, data = train_data, method = "naive_bayes", trControl = train_control)

# Support Vector Machine (SVM)
svm_model <- train(train_formula, data = train_data, method = "svmLinear", trControl = train_control)
```

Step 5: Model Evaluation

```r
# Predictions for each model
log_pred <- predict(log_model, test_data)
tree_pred <- predict(tree_model, test_data)
rf_pred <- predict(rf_model, test_data)
knn_pred <- predict(knn_model, test_data)
nb_pred <- predict(nb_model, test_data)
svm_pred <- predict(svm_model, test_data)

# Calculate accuracy for each model
log_accuracy <- mean(log_pred == test_data$M_category)
tree_accuracy <- mean(tree_pred == test_data$M_category)
rf_accuracy <- mean(rf_pred == test_data$M_category)
knn_accuracy <- mean(knn_pred == test_data$M_category)
```

```r
nb_accuracy <- mean(nb_pred == test_data$M_category)
svm_accuracy <- mean(svm_pred == test_data$M_category)

# Print model accuracies
cat("Logistic Regression Accuracy:", log_accuracy, "\n")
```

```
## Logistic Regression Accuracy: 0.4545455
```

```r
cat("Decision Tree Accuracy:", tree_accuracy, "\n")
```

```
## Decision Tree Accuracy: 0.4646465
```

```r
cat("Random Forest Accuracy:", rf_accuracy, "\n")
```

```
## Random Forest Accuracy: 0.5959596
```

```r
cat("KNN Accuracy:", knn_accuracy, "\n")
```

```
## KNN Accuracy: 0.3434343
```

```r
cat("Naive Bayes Accuracy:", nb_accuracy, "\n")
```

```
## Naive Bayes Accuracy: 0.4343434
```

```r
cat("SVM Accuracy:", svm_accuracy, "\n")
```

```
## SVM Accuracy: 0.5353535
```

```r
# Confusion matrices for detailed evaluation
confusionMatrix(log_pred, test_data$M_category)
```

```
## Confusion Matrix and Statistics
##
##                                 Reference
## Prediction                        Anti-personnel Anti-Tank
##   Anti-personnel                               4         0
##   Anti-Tank                                    1        21
##   Booby Trapped Anti-personnel                 6         0
##   M14 Anti-personnel                           5         0
##   Null                                         3         0
##                                 Reference
## Prediction                        Booby Trapped Anti-personnel M14 Anti-personnel
##   Anti-personnel                                             3                 11
##   Anti-Tank                                                  1                  0
##   Booby Trapped Anti-personnel                               5                  3
##   M14 Anti-personnel                                         5                  3
##   Null                                                       5                  2
##                                 Reference
```

```
## Prediction                     Null
##   Anti-personnel                  3
##   Anti-Tank                       0
##   Booby Trapped Anti-personnel    4
##   M14 Anti-personnel              2
##   Null                           12
##
## Overall Statistics
##
##                Accuracy : 0.4545
##                  95% CI : (0.3541, 0.5577)
##     No Information Rate : 0.2121
##     P-Value [Acc > NIR] : 6.245e-08
##
##                   Kappa : 0.3172
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Anti-personnel Class: Anti-Tank
## Sensitivity                         0.2105           1.0000
## Specificity                         0.7875           0.9744
## Pos Pred Value                      0.1905           0.9130
## Neg Pred Value                      0.8077           1.0000
## Prevalence                          0.1919           0.2121
## Detection Rate                      0.0404           0.2121
## Detection Prevalence                0.2121           0.2323
## Balanced Accuracy                   0.4990           0.9872
##                      Class: Booby Trapped Anti-personnel
## Sensitivity                                      0.26316
## Specificity                                      0.83750
## Pos Pred Value                                   0.27778
## Neg Pred Value                                   0.82716
## Prevalence                                       0.19192
## Detection Rate                                   0.05051
## Detection Prevalence                             0.18182
## Balanced Accuracy                                0.55033
##                      Class: M14 Anti-personnel Class: Null
## Sensitivity                             0.1579      0.5714
## Specificity                             0.8500      0.8718
## Pos Pred Value                          0.2000      0.5455
## Neg Pred Value                          0.8095      0.8831
## Prevalence                              0.1919      0.2121
## Detection Rate                          0.0303      0.1212
## Detection Prevalence                    0.1515      0.2222
## Balanced Accuracy                       0.5039      0.7216
```

```
confusionMatrix(tree_pred, test_data$M_category)
```

```
## Confusion Matrix and Statistics
##
##                              Reference
## Prediction                     Anti-personnel Anti-Tank
```

```
##   Anti-personnel                        6        0
##   Anti-Tank                             5       21
##   Booby Trapped Anti-personnel          0        0
##   M14 Anti-personnel                    0        0
##   Null                                  8        0
##                               Reference
## Prediction                    Booby Trapped Anti-personnel M14 Anti-personnel
##   Anti-personnel                                         4                  7
##   Anti-Tank                                              5                  2
##   Booby Trapped Anti-personnel                           0                  0
##   M14 Anti-personnel                                     0                  0
##   Null                                                  10                 10
##                               Reference
## Prediction                     Null
##   Anti-personnel                  2
##   Anti-Tank                       0
##   Booby Trapped Anti-personnel    0
##   M14 Anti-personnel              0
##   Null                           19
##
## Overall Statistics
##
##                Accuracy : 0.4646
##                  95% CI : (0.3638, 0.5677)
##     No Information Rate : 0.2121
##     P-Value [Acc > NIR] : 1.94e-08
##
##                   Kappa : 0.3238
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Anti-personnel Class: Anti-Tank
## Sensitivity                        0.31579           1.0000
## Specificity                        0.83750           0.8462
## Pos Pred Value                     0.31579           0.6364
## Neg Pred Value                     0.83750           1.0000
## Prevalence                         0.19192           0.2121
## Detection Rate                     0.06061           0.2121
## Detection Prevalence               0.19192           0.3333
## Balanced Accuracy                  0.57664           0.9231
##                      Class: Booby Trapped Anti-personnel
## Sensitivity                                       0.0000
## Specificity                                       1.0000
## Pos Pred Value                                       NaN
## Neg Pred Value                                    0.8081
## Prevalence                                        0.1919
## Detection Rate                                    0.0000
## Detection Prevalence                              0.0000
## Balanced Accuracy                                 0.5000
##                      Class: M14 Anti-personnel Class: Null
## Sensitivity                             0.0000      0.9048
## Specificity                             1.0000      0.6410
```

```
## Pos Pred Value                           NaN      0.4043
## Neg Pred Value                        0.8081      0.9615
## Prevalence                            0.1919      0.2121
## Detection Rate                        0.0000      0.1919
## Detection Prevalence                  0.0000      0.4747
## Balanced Accuracy                     0.5000      0.7729
```

```
confusionMatrix(rf_pred, test_data$M_category)
```

```
## Confusion Matrix and Statistics
##
##                              Reference
## Prediction                     Anti-personnel Anti-Tank
##   Anti-personnel                            9         2
##   Anti-Tank                                 1        19
##   Booby Trapped Anti-personnel              3         0
##   M14 Anti-personnel                        3         0
##   Null                                      3         0
##                              Reference
## Prediction                     Booby Trapped Anti-personnel M14 Anti-personnel
##   Anti-personnel                                          3                  7
##   Anti-Tank                                               1                  0
##   Booby Trapped Anti-personnel                           12                  6
##   M14 Anti-personnel                                      3                  3
##   Null                                                    0                  3
##                              Reference
## Prediction                     Null
##   Anti-personnel                  1
##   Anti-Tank                       0
##   Booby Trapped Anti-personnel    1
##   M14 Anti-personnel              3
##   Null                           16
##
## Overall Statistics
##
##                Accuracy : 0.596
##                  95% CI : (0.4926, 0.6934)
##     No Information Rate : 0.2121
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4945
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                         Class: Anti-personnel Class: Anti-Tank
## Sensitivity                           0.47368           0.9048
## Specificity                           0.83750           0.9744
## Pos Pred Value                        0.40909           0.9048
## Neg Pred Value                        0.87013           0.9744
## Prevalence                            0.19192           0.2121
## Detection Rate                        0.09091           0.1919
## Detection Prevalence                  0.22222           0.2121
```

```
## Balanced Accuracy                  0.65559          0.9396
##                      Class: Booby Trapped Anti-personnel
## Sensitivity                                       0.6316
## Specificity                                       0.8750
## Pos Pred Value                                    0.5455
## Neg Pred Value                                    0.9091
## Prevalence                                        0.1919
## Detection Rate                                    0.1212
## Detection Prevalence                              0.2222
## Balanced Accuracy                                 0.7533
##                      Class: M14 Anti-personnel Class: Null
## Sensitivity                             0.1579      0.7619
## Specificity                             0.8875      0.9231
## Pos Pred Value                          0.2500      0.7273
## Neg Pred Value                          0.8161      0.9351
## Prevalence                              0.1919      0.2121
## Detection Rate                          0.0303      0.1616
## Detection Prevalence                    0.1212      0.2222
## Balanced Accuracy                       0.5227      0.8425
```

```r
confusionMatrix(knn_pred, test_data$M_category)
```

```
## Confusion Matrix and Statistics
##
##                                Reference
## Prediction                       Anti-personnel Anti-Tank
##   Anti-personnel                              4         2
##   Anti-Tank                                   1        18
##   Booby Trapped Anti-personnel                7         0
##   M14 Anti-personnel                          2         1
##   Null                                        5         0
##                                Reference
## Prediction                       Booby Trapped Anti-personnel M14 Anti-personnel
##   Anti-personnel                                            5                 10
##   Anti-Tank                                                 0                  0
##   Booby Trapped Anti-personnel                              6                  4
##   M14 Anti-personnel                                        5                  1
##   Null                                                      3                  4
##                                Reference
## Prediction                       Null
##   Anti-personnel                    9
##   Anti-Tank                         0
##   Booby Trapped Anti-personnel      5
##   M14 Anti-personnel                2
##   Null                              5
##
## Overall Statistics
##
##                Accuracy : 0.3434
##                  95% CI : (0.2509, 0.4456)
##     No Information Rate : 0.2121
##     P-Value [Acc > NIR] : 0.00175
##
##                   Kappa : 0.18
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Anti-personnel Class: Anti-Tank
## Sensitivity                         0.2105           0.8571
## Specificity                         0.6750           0.9872
## Pos Pred Value                      0.1333           0.9474
## Neg Pred Value                      0.7826           0.9625
## Prevalence                          0.1919           0.2121
## Detection Rate                      0.0404           0.1818
## Detection Prevalence                0.3030           0.1919
## Balanced Accuracy                   0.4428           0.9222
##                      Class: Booby Trapped Anti-personnel
## Sensitivity                                      0.31579
## Specificity                                      0.80000
## Pos Pred Value                                   0.27273
## Neg Pred Value                                   0.83117
## Prevalence                                       0.19192
## Detection Rate                                   0.06061
## Detection Prevalence                             0.22222
## Balanced Accuracy                                0.55789
##                      Class: M14 Anti-personnel Class: Null
## Sensitivity                            0.05263     0.23810
## Specificity                            0.87500     0.84615
## Pos Pred Value                         0.09091     0.29412
## Neg Pred Value                         0.79545     0.80488
## Prevalence                             0.19192     0.21212
## Detection Rate                         0.01010     0.05051
## Detection Prevalence                   0.11111     0.17172
## Balanced Accuracy                      0.46382     0.54212
```

```r
confusionMatrix(nb_pred, test_data$M_category)
```

```
## Confusion Matrix and Statistics
##
##                                Reference
## Prediction                       Anti-personnel Anti-Tank
##   Anti-personnel                              5         0
##   Anti-Tank                                   3        21
##   Booby Trapped Anti-personnel                0         0
##   M14 Anti-personnel                          2         0
##   Null                                        9         0
##                                Reference
## Prediction                       Booby Trapped Anti-personnel M14 Anti-personnel
##   Anti-personnel                                            4                 10
##   Anti-Tank                                                 2                  2
##   Booby Trapped Anti-personnel                              3                  0
##   M14 Anti-personnel                                        3                  0
##   Null                                                      7                  7
##                                Reference
## Prediction                       Null
##   Anti-personnel                    6
```

```
##   Anti-Tank                       0
##   Booby Trapped Anti-personnel    1
##   M14 Anti-personnel              0
##   Null                           14
##
## Overall Statistics
##
##                Accuracy : 0.4343
##                  95% CI : (0.335, 0.5377)
##     No Information Rate : 0.2121
##     P-Value [Acc > NIR] : 5.751e-07
##
##                   Kappa : 0.2883
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Anti-personnel Class: Anti-Tank
## Sensitivity                        0.26316           1.0000
## Specificity                        0.75000           0.9103
## Pos Pred Value                     0.20000           0.7500
## Neg Pred Value                     0.81081           1.0000
## Prevalence                         0.19192           0.2121
## Detection Rate                     0.05051           0.2121
## Detection Prevalence               0.25253           0.2828
## Balanced Accuracy                  0.50658           0.9551
##                      Class: Booby Trapped Anti-personnel
## Sensitivity                                       0.1579
## Specificity                                       0.9875
## Pos Pred Value                                    0.7500
## Neg Pred Value                                    0.8316
## Prevalence                                        0.1919
## Detection Rate                                    0.0303
## Detection Prevalence                              0.0404
## Balanced Accuracy                                 0.5727
##                      Class: M14 Anti-personnel Class: Null
## Sensitivity                            0.00000      0.6667
## Specificity                            0.93750      0.7051
## Pos Pred Value                         0.00000      0.3784
## Neg Pred Value                         0.79787      0.8871
## Prevalence                             0.19192      0.2121
## Detection Rate                         0.00000      0.1414
## Detection Prevalence                   0.05051      0.3737
## Balanced Accuracy                      0.46875      0.6859
```

```
confusionMatrix(svm_pred, test_data$M_category)
```

```
## Confusion Matrix and Statistics
##
##                               Reference
## Prediction                     Anti-personnel Anti-Tank
##   Anti-personnel                            8         2
##   Anti-Tank                                 1        19
```

```
##   Booby Trapped Anti-personnel              7         0
##   M14 Anti-personnel                        0         0
##   Null                                      3         0
##                               Reference
## Prediction                     Booby Trapped Anti-personnel M14 Anti-personnel
##   Anti-personnel                                          4                  9
##   Anti-Tank                                               0                  0
##   Booby Trapped Anti-personnel                           11                  4
##   M14 Anti-personnel                                      1                  4
##   Null                                                    3                  2
##                               Reference
## Prediction                     Null
##   Anti-personnel                  3
##   Anti-Tank                       0
##   Booby Trapped Anti-personnel    5
##   M14 Anti-personnel              2
##   Null                           11
##
## Overall Statistics
##
##                Accuracy : 0.5354
##                  95% CI : (0.4323, 0.6362)
##     No Information Rate : 0.2121
##     P-Value [Acc > NIR] : 1.804e-12
##
##                   Kappa : 0.4193
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Anti-personnel Class: Anti-Tank
## Sensitivity                        0.42105           0.9048
## Specificity                        0.77500           0.9872
## Pos Pred Value                     0.30769           0.9500
## Neg Pred Value                     0.84932           0.9747
## Prevalence                         0.19192           0.2121
## Detection Rate                     0.08081           0.1919
## Detection Prevalence               0.26263           0.2020
## Balanced Accuracy                  0.59803           0.9460
##                      Class: Booby Trapped Anti-personnel
## Sensitivity                                       0.5789
## Specificity                                       0.8000
## Pos Pred Value                                    0.4074
## Neg Pred Value                                    0.8889
## Prevalence                                        0.1919
## Detection Rate                                    0.1111
## Detection Prevalence                              0.2727
## Balanced Accuracy                                 0.6895
##                      Class: M14 Anti-personnel Class: Null
## Sensitivity                            0.21053      0.5238
## Specificity                            0.96250      0.8974
## Pos Pred Value                         0.57143      0.5789
## Neg Pred Value                         0.83696      0.8750
```

```
## Prevalence                          0.19192      0.2121
## Detection Rate                      0.04040      0.1111
## Detection Prevalence                0.07071      0.1919
## Balanced Accuracy                   0.58651      0.7106
```

Step 6: Hyperparameter Tuning

```r
# Hyperparameter grid for Logistic Regression (multinom)
log_grid <- expand.grid(.decay = c(0.1, 0.01, 0.001))

# Hyperparameter grid for Decision Tree (rpart)
tree_grid <- expand.grid(.cp = seq(0.01, 0.1, by = 0.01))

# Hyperparameter grid for Random Forest (rf)
rf_grid <- expand.grid(.mtry = c(2, 3, 4, 5, 6))

# Hyperparameter grid for K-Nearest Neighbors (knn)
knn_grid <- expand.grid(.k = c(3, 5, 7, 9))  # Number of neighbors

# Support Vector Machine (svmLinear)
svm_grid <- expand.grid(.C = 30, .sigma = 0.7)  # Cost parameter

# Hyperparameter grid for Naive Bayes
nb_grid <- NULL  # Naive Bayes typically doesn't require hyperparameter tuning

# Logistic Regression
log_model <- train(train_formula, data = train_data, method = "multinom",
                   trControl = train_control, tuneGrid = log_grid)
```

```
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 275.157901
## iter  20 value 273.530476
## final  value 273.526413
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 241.380929
## iter  20 value 229.094798
## iter  30 value 228.546165
## iter  40 value 228.536732
## iter  40 value 228.536731
## iter  40 value 228.536731
## final  value 228.536731
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 234.526598
## iter  20 value 209.187096
## iter  30 value 206.261501
## iter  40 value 206.226230
```

```
## iter  50 value 206.221803
## final  value 206.221769
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 273.565420
## iter  20 value 270.966215
## final  value 270.960280
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 244.049330
## iter  20 value 225.924678
## iter  30 value 225.121321
## final  value 225.112299
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 238.493693
## iter  20 value 205.960972
## iter  30 value 201.758982
## iter  40 value 201.728720
## iter  50 value 201.727406
## final  value 201.727349
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 275.166661
## iter  20 value 273.539035
## iter  30 value 273.533372
## final  value 273.533364
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 249.181144
## iter  20 value 229.266169
## iter  30 value 228.727596
## final  value 228.722191
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 241.257458
## iter  20 value 207.645943
## iter  30 value 205.609489
## iter  40 value 205.585854
## final  value 205.585815
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 275.768640
## iter  20 value 274.407836
## iter  30 value 274.402786
## final  value 274.402780
## converged
```

```
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 241.193321
## iter  20 value 228.475529
## iter  30 value 227.919139
## final  value 227.912484
## converged
## # weights:  45 (32 variable)
## initial  value 347.638589
## iter  10 value 233.813219
## iter  20 value 206.816027
## iter  30 value 204.827308
## iter  40 value 204.804626
## iter  50 value 204.802877
## final  value 204.802872
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 272.437821
## iter  20 value 270.960569
## final  value 270.956508
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 239.341844
## iter  20 value 227.638184
## iter  30 value 227.058280
## final  value 227.043756
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 232.391667
## iter  20 value 208.066377
## iter  30 value 205.332586
## iter  40 value 205.270454
## iter  50 value 205.257983
## final  value 205.257956
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 273.787773
## iter  20 value 272.212603
## final  value 272.209068
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 238.171920
## iter  20 value 226.394736
## iter  30 value 225.870993
## iter  40 value 225.862248
## iter  40 value 225.862247
## iter  40 value 225.862247
## final  value 225.862247
## converged
```

```
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 230.657206
## iter  20 value 204.808220
## iter  30 value 202.485011
## iter  40 value 202.454590
## iter  50 value 202.452991
## final  value 202.452934
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 276.204512
## iter  20 value 274.479244
## final  value 274.474286
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 242.307181
## iter  20 value 229.867531
## iter  30 value 229.172865
## iter  40 value 229.167834
## iter  40 value 229.167834
## iter  40 value 229.167834
## final  value 229.167834
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 235.409111
## iter  20 value 207.703164
## iter  30 value 206.257763
## iter  40 value 206.230775
## iter  50 value 206.228740
## final  value 206.228691
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 275.287653
## iter  20 value 272.061175
## final  value 272.053384
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 238.654915
## iter  20 value 225.304976
## iter  30 value 224.722472
## final  value 224.717768
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 231.149764
## iter  20 value 201.942260
## iter  30 value 200.203665
## iter  40 value 200.179709
## final  value 200.179695
```

```
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 276.800475
## iter  20 value 274.877265
## iter  30 value 274.872752
## iter  30 value 274.872750
## iter  30 value 274.872750
## final  value 274.872750
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 241.972015
## iter  20 value 229.610435
## iter  30 value 229.067759
## final  value 229.065281
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 239.544028
## iter  20 value 207.963681
## iter  30 value 205.866312
## iter  40 value 205.836630
## final  value 205.836559
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 276.196715
## iter  20 value 272.474056
## final  value 272.464543
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 245.480041
## iter  20 value 226.908814
## iter  30 value 226.220050
## iter  40 value 226.213322
## iter  40 value 226.213321
## iter  40 value 226.213321
## final  value 226.213321
## converged
## # weights:  45 (32 variable)
## initial  value 346.029151
## iter  10 value 237.836164
## iter  20 value 205.602684
## iter  30 value 202.616832
## iter  40 value 202.579732
## iter  50 value 202.577554
## final  value 202.577510
## converged
## # weights:  45 (32 variable)
## initial  value 384.655661
## iter  10 value 257.083285
## iter  20 value 230.715357
```

```
## iter  30 value 227.460558
## iter  40 value 227.415977
## iter  50 value 227.410380
## final  value 227.410344
## converged
```

```r
# Decision Tree
tree_model <- train(train_formula, data = train_data, method = "rpart",
                    trControl = train_control, tuneGrid = tree_grid)

# Random Forest
rf_model <- train(train_formula, data = train_data, method = "rf",
                  trControl = train_control, tuneGrid = rf_grid)

# K-Nearest Neighbors (KNN)
knn_model <- train(train_formula, data = train_data, method = "knn",
                   trControl = train_control, tuneGrid = knn_grid)

# Support Vector Machine (SVM)
svm_model <- train(train_formula, data = train_data, method = "svmRadial",
                   trControl = train_control, tuneGrid = svm_grid)

# Naive Bayes
nb_model <- train(train_formula, data = train_data, method = "naive_bayes",
                  trControl = train_control, tuneGrid = nb_grid)
```

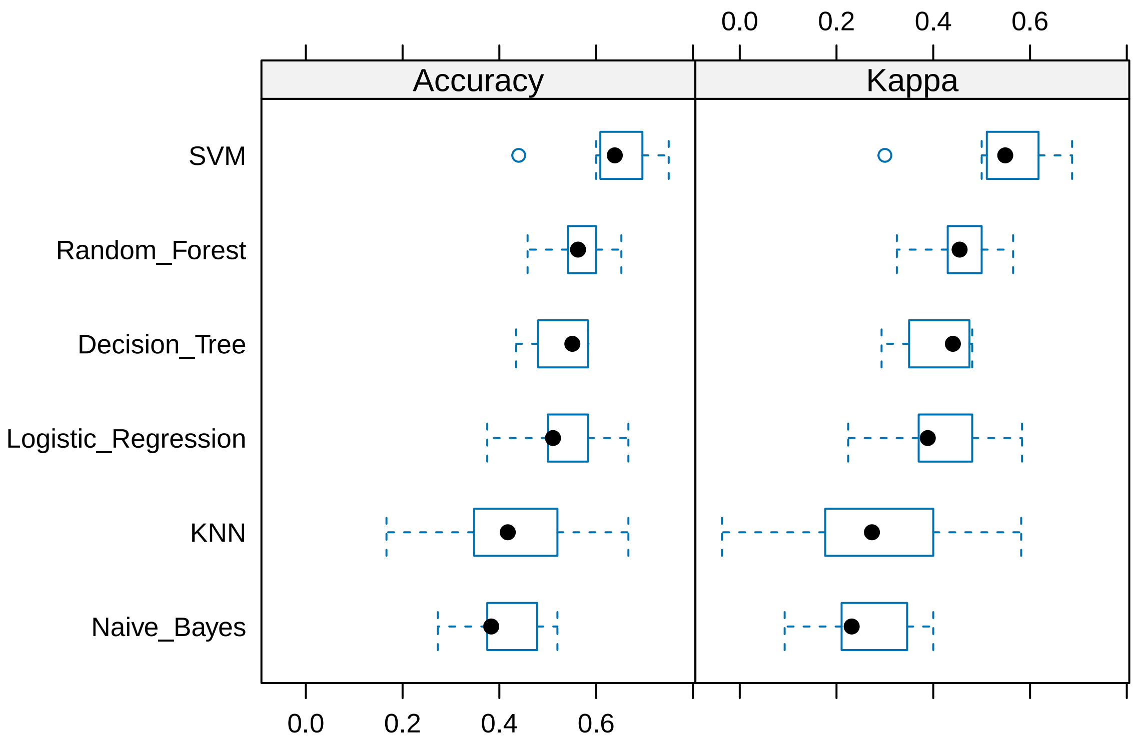
Step 7: Model Comparison

```r
# Compare models
model_comparison <- resamples(list(Logistic_Regression = log_model,
                                   Decision_Tree = tree_model,
                                   Random_Forest = rf_model,
                                   KNN = knn_model,
                                   SVM = svm_model,
                                   Naive_Bayes = nb_model))

# Print model comparison results
summary(model_comparison)
```

```
##
## Call:
## summary.resamples(object = model_comparison)
##
## Models: Logistic_Regression, Decision_Tree, Random_Forest, KNN, SVM, Naive_Bayes
## Number of resamples: 10
##
## Accuracy
##                          Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## Logistic_Regression 0.3750000 0.5000000 0.5108696 0.5313406 0.5729167 0.6666667
## Decision_Tree       0.4347826 0.4954167 0.5508333 0.5319545 0.5803571 0.5833333
## Random_Forest       0.4583333 0.5462500 0.5626087 0.5651159 0.5958333 0.6521739
```

```
## KNN                 0.1666667 0.3586957 0.4173913 0.4187319 0.5045833 0.6666667
## SVM                 0.4400000 0.6127717 0.6385870 0.6372174 0.6917391 0.7500000
## Naive_Bayes         0.2727273 0.3750000 0.3831522 0.4043959 0.4628623 0.5200000
##                     NA's
## Logistic_Regression    0
## Decision_Tree          0
## Random_Forest          0
## KNN                    0
## SVM                    0
## Naive_Bayes            0
##
## Kappa
##                            Min.   1st Qu.    Median      Mean   3rd Qu.
## Logistic_Regression  0.22413793 0.3704896 0.3886075 0.4137828 0.4665988
## Decision_Tree        0.29314421 0.3699514 0.4405172 0.4145322 0.4711269
## Random_Forest        0.32467532 0.4348542 0.4544118 0.4559061 0.4939956
## KNN                 -0.03671706 0.1925997 0.2732353 0.2729365 0.3800654
## SVM                  0.30000000 0.5155874 0.5487397 0.5462471 0.6131829
## Naive_Bayes          0.09278351 0.2113882 0.2313866 0.2544002 0.3264721
##                          Max. NA's
## Logistic_Regression 0.5835141    0
## Decision_Tree       0.4805195    0
## Random_Forest       0.5650118    0
## KNN                 0.5816993    0
## SVM                 0.6869565    0
## Naive_Bayes         0.4000000    0
```

```r
# To plot the comparison results
bwplot(model_comparison)
```



STep 8: Evaluating Best Model

```r
# Choose the best model based on accuracy (SVM in this case)
best_model <- svm_model  # SVM as the best model

# Evaluate on test data
test_pred <- predict(best_model, test_data)
test_accuracy <- mean(test_pred == test_data$M_category)
cat("Test Accuracy of Best Model (SVM):", test_accuracy, "\n")
```

```
## Test Accuracy of Best Model (SVM): 0.6161616
```

```r
# Confusion matrix
confusionMatrix(test_pred, test_data$M_category)
```

```
## Confusion Matrix and Statistics
##
##                                  Reference
## Prediction                         Anti-personnel Anti-Tank
##    Anti-personnel                               9         0
##    Anti-Tank                                    1        21
##    Booby Trapped Anti-personnel                 3         0
##    M14 Anti-personnel                           3         0
##    Null                                         3         0
##                                  Reference
## Prediction                         Booby Trapped Anti-personnel M14 Anti-personnel
```

```
##   Anti-personnel                                          3                   8
##   Anti-Tank                                               0                   0
##   Booby Trapped Anti-personnel                           14                   3
##   M14 Anti-personnel                                      2                   4
##   Null                                                    0                   4
##                             Reference
## Prediction                   Null
##   Anti-personnel                2
##   Anti-Tank                     0
##   Booby Trapped Anti-personnel  2
##   M14 Anti-personnel            4
##   Null                         13
##
## Overall Statistics
##
##                Accuracy : 0.6162
##                  95% CI : (0.513, 0.7122)
##     No Information Rate : 0.2121
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5199
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Anti-personnel Class: Anti-Tank
## Sensitivity                        0.47368           1.0000
## Specificity                        0.83750           0.9872
## Pos Pred Value                     0.40909           0.9545
## Neg Pred Value                     0.87013           1.0000
## Prevalence                         0.19192           0.2121
## Detection Rate                     0.09091           0.2121
## Detection Prevalence               0.22222           0.2222
## Balanced Accuracy                  0.65559           0.9936
##                      Class: Booby Trapped Anti-personnel
## Sensitivity                                       0.7368
## Specificity                                       0.9000
## Pos Pred Value                                    0.6364
## Neg Pred Value                                    0.9351
## Prevalence                                        0.1919
## Detection Rate                                    0.1414
## Detection Prevalence                              0.2222
## Balanced Accuracy                                 0.8184
##                      Class: M14 Anti-personnel Class: Null
## Sensitivity                             0.2105      0.6190
## Specificity                             0.8875      0.9103
## Pos Pred Value                          0.3077      0.6500
## Neg Pred Value                          0.8256      0.8987
## Prevalence                              0.1919      0.2121
## Detection Rate                          0.0404      0.1313
## Detection Prevalence                    0.1313      0.2020
## Balanced Accuracy                       0.5490      0.7647
```