6220_Task

Payel Ghosal

2024-04-16

```
df <- read.csv("D:\\MS-2\\Sem-4\\STAT-6220_Consulting\\acoustics_0214.csv",header=T,sep=",")
names(df)[names(df) == 'ï..1'] <- 'X1'
df$Tone <- as.factor(df$Tone)
```

```
## Recall goal of the analysis is to predict Tone based on other features
#(X1-10, Gender, Duration);
## Not sure about "Group", can be used if this feature greatly improves prediction.
```

```
## Task 0: Read dataset into R or Python. Preliminary
#understanding of the size of the dataset, types and ranges of variables etc.
#Write a short paragraph to communicate the result.
```

```
head(df)
```

```
##      X1    X2    X3    X4    X5    X6    X7    X8    X9   X10 Participant Group
## 1    NA    NA    NA    NA    NA 113.2 119.7 127.3 133.6 145.8         CM1    CM
## 2    NA    NA    NA    NA    NA 106.0 110.5 119.6 126.0 130.9         CM1    CM
## 3    NA    NA  81.0  82.0    NA 105.5 107.9 113.8 121.5 123.3         CM1    CM
## 4 101.0 103.9    NA    NA 144.3 152.6 158.8 164.7 170.4 162.4         CM1    CM
## 5  97.3    NA    NA    NA 139.6 147.9 155.7 159.9 166.5 166.2         CM1    CM
## 6 179.5 177.2 169.7 156.9 139.9    NA    NA    NA    NA    NA         CM1    CM
##   Gender  Duration  Tone
## 1      M 0.5032306 CM_T4
## 2      M 0.5667390 CM_T4
## 3      M 0.4752880 CM_T4
## 4      M 0.3840586 CM_T1
## 5      M 0.3711692 CM_T1
## 6      M 0.3364927 CM_T3
```

```
str(df)
```

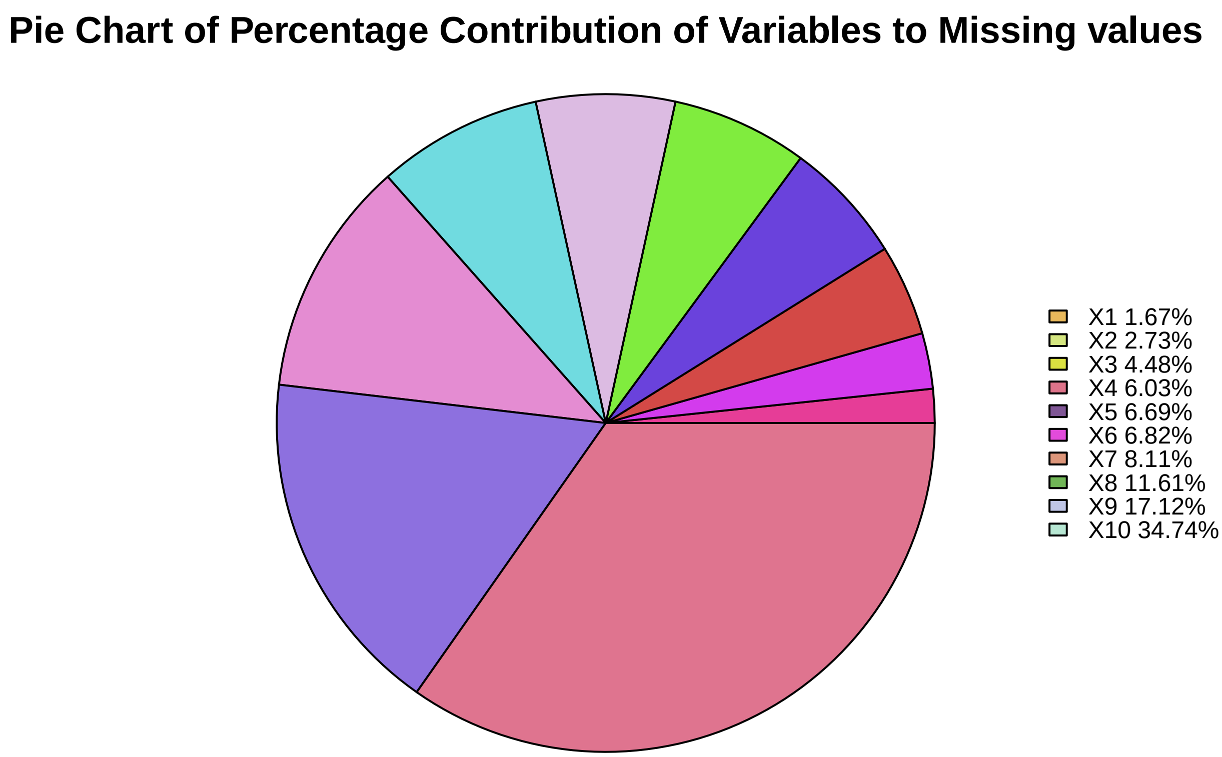
```
## 'data.frame':    9021 obs. of  15 variables:
##  $ X1         : num  NA NA NA 101 97.3 ...
##  $ X2         : num  NA NA NA 104 NA ...
##  $ X3         : num  NA NA 81 NA NA ...
##  $ X4         : num  NA NA 82 NA NA ...
##  $ X5         : num  NA NA NA 144 140 ...
##  $ X6         : num  113 106 106 153 148 ...
##  $ X7         : num  120 110 108 159 156 ...
##  $ X8         : num  127 120 114 165 160 ...
##  $ X9         : num  134 126 122 170 166 ...
##  $ X10        : num  146 131 123 162 166 ...
##  $ Participant: chr  "CM1" "CM1" "CM1" "CM1" ...
##  $ Group      : chr  "CM" "CM" "CM" "CM" ...
##  $ Gender     : chr  "M" "M" "M" "M" ...
```

```
## $ Duration : num 0.503 0.567 0.475 0.384 0.371 ...
## $ Tone : Factor w/ 8 levels "CM_T1","CM_T2",..: 4 4 4 1 1 3 3 1 1 1 ...
```

$X_1, ..., X_{10}$ are the frequencies. Participants, Group, Gender, Tone are categorical variables. Duration is a numeric variable. Missing values might induce problem in the analysis.

Missing Data Visualization:

```r
set.seed(18)
miss <- data.frame(is.na(df))
pmiss <- apply(miss,2,mean)
slice <-pmiss*100

slice<- slice[slice>0]

labls <-names(slice)

pct <- round(slice/sum(slice)*100 , 2)

labls <- paste(labls, pct)

labls <- paste(labls,"%",sep="")

#library(RColorBrewer)
library(randomcoloR)

#par->opar()
par(mar=c(2,0,2,0))

pie(slice,radius = 1,labels=NA,col=distinctColorPalette(30),
main="Pie Chart of Percentage Contribution of Variables to Missing values")

#legend_order <- matrix(1:30,ncol=2,byrow=T)
par(mar=c(0,0,0,0))
legend("right",labls,fill=distinctColorPalette(30),cex=0.75,bty="n")
```



Missing Data handling:

```
set.seed(18)
imp_df <- mice(df, m=5, maxit=0)
```

```
## Warning: Number of logged events: 3
```

```
set.seed(18)
summary(imp_df)
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##          X1          X2          X3          X4          X5          X6
##       "pmm"       "pmm"       "pmm"       "pmm"       "pmm"       "pmm"
##          X7          X8          X9         X10 Participant       Group
##       "pmm"       "pmm"       "pmm"       "pmm"          ""          ""
##      Gender    Duration        Tone
##          ""          ""          ""
## PredictorMatrix:
##    X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 Participant Group Gender Duration Tone
## X1  0  1  1  1  1  1  1  1  1   1           0     0      0        1    1
## X2  1  0  1  1  1  1  1  1  1   1           0     0      0        1    1
## X3  1  1  0  1  1  1  1  1  1   1           0     0      0        1    1
## X4  1  1  1  0  1  1  1  1  1   1           0     0      0        1    1
```

```
## X5  1  1  1  1  0  1  1  1  1   1           0     0      0        1    1
## X6  1  1  1  1  1  0  1  1  1   1           0     0      0        1    1
## Number of logged events:  3
##   it im dep     meth         out
## 1  0  0     constant Participant
## 2  0  0     constant       Group
## 3  0  0     constant      Gender
```

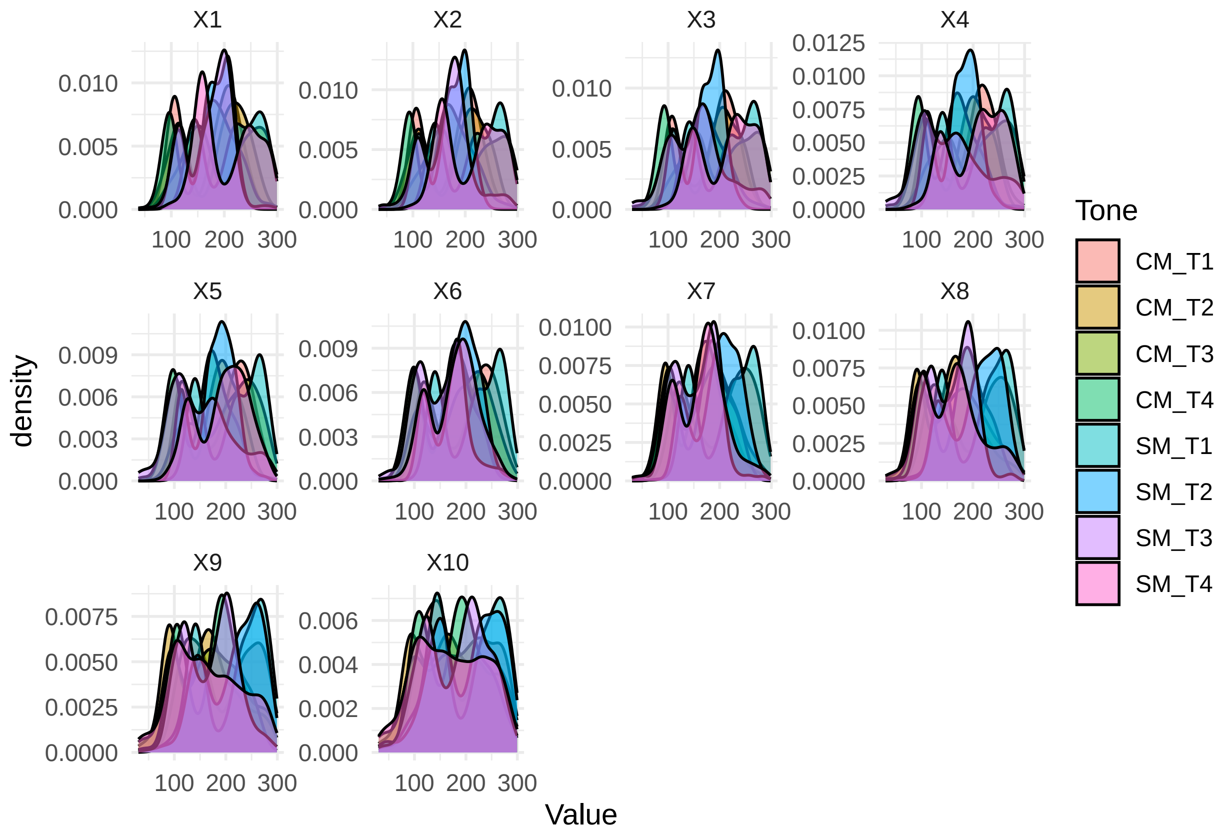
```r
set.seed(18)
final_df <- complete(imp_df,1)
head(final_df)
```

```
##      X1    X2    X3    X4    X5    X6    X7    X8    X9   X10 Participant Group
## 1 182.3 169.3 222.4 162.5 271.1 113.2 119.7 127.3 133.6 145.8         CM1    CM
## 2 123.4 249.4 114.0 155.1 173.4 106.0 110.5 119.6 126.0 130.9         CM1    CM
## 3  96.7 104.1  81.0  82.0 128.3 105.5 107.9 113.8 121.5 123.3         CM1    CM
## 4 101.0 103.9 256.2 288.8 144.3 152.6 158.8 164.7 170.4 162.4         CM1    CM
## 5  97.3 106.8 181.7 105.7 139.6 147.9 155.7 159.9 166.5 166.2         CM1    CM
## 6 179.5 177.2 169.7 156.9 139.9 181.5 220.6 241.3 189.9 231.0         CM1    CM
##   Gender  Duration  Tone
## 1      M 0.5032306 CM_T4
## 2      M 0.5667390 CM_T4
## 3      M 0.4752880 CM_T4
## 4      M 0.3840586 CM_T1
## 5      M 0.3711692 CM_T1
## 6      M 0.3364927 CM_T3
```

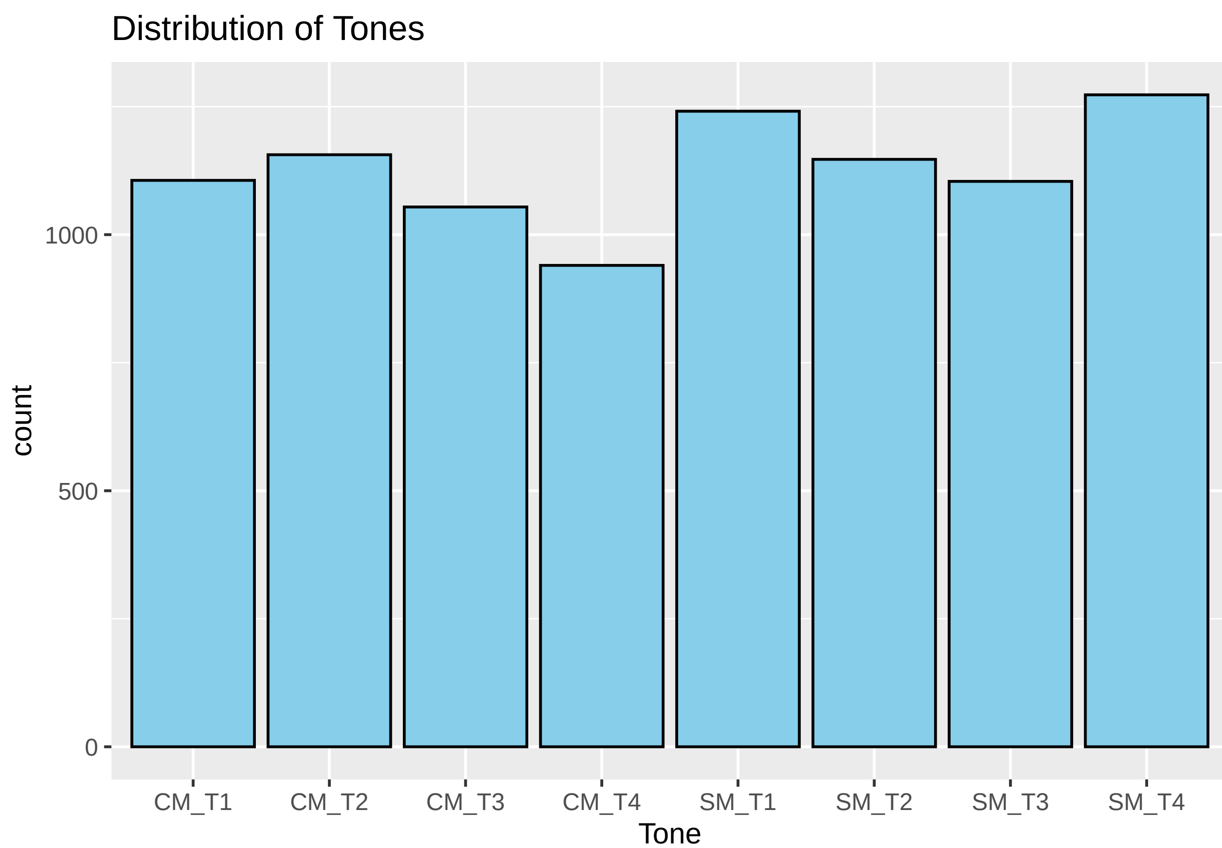
TASK-1:

Compare the features that correspond to the 8 Tones. (E.g. Produce representative shapes X1-10 for each Tone. Can control for Gender if needed. Likely need to deal with "NA" at this stage.)

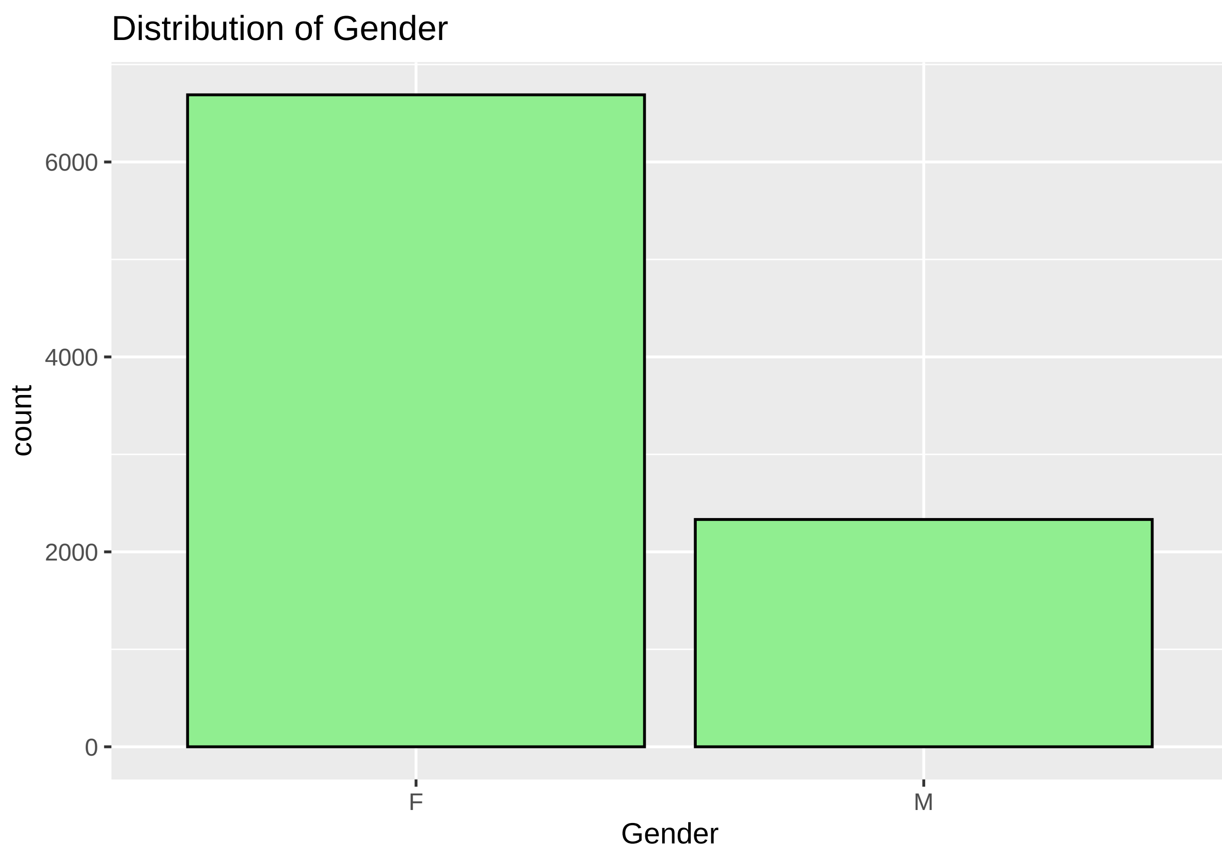
```r
df_freq <- final_df[,c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","Tone")]
library(reshape2)
df_melt <- melt(df_freq,id.vars = "Tone",variable.name = "Frequency",value.name = "Value")
ggplot(df_melt,aes(x=Value,fill=Tone))+
  geom_density(alpha=0.5)+
  facet_wrap(~Frequency,scales="free")+
  theme_minimal()
```



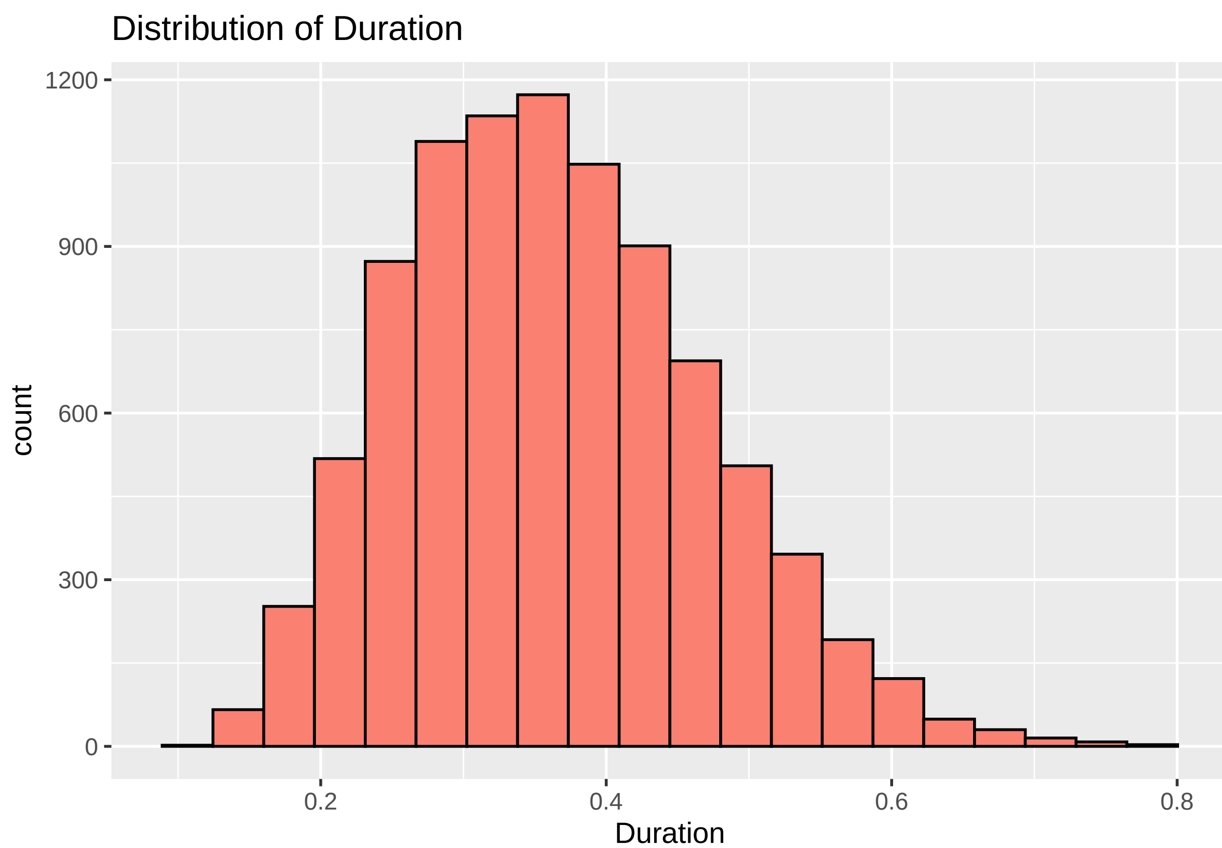

```r
# Distribution of Tone
ggplot(final_df, aes(x = Tone)) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Tones")
```



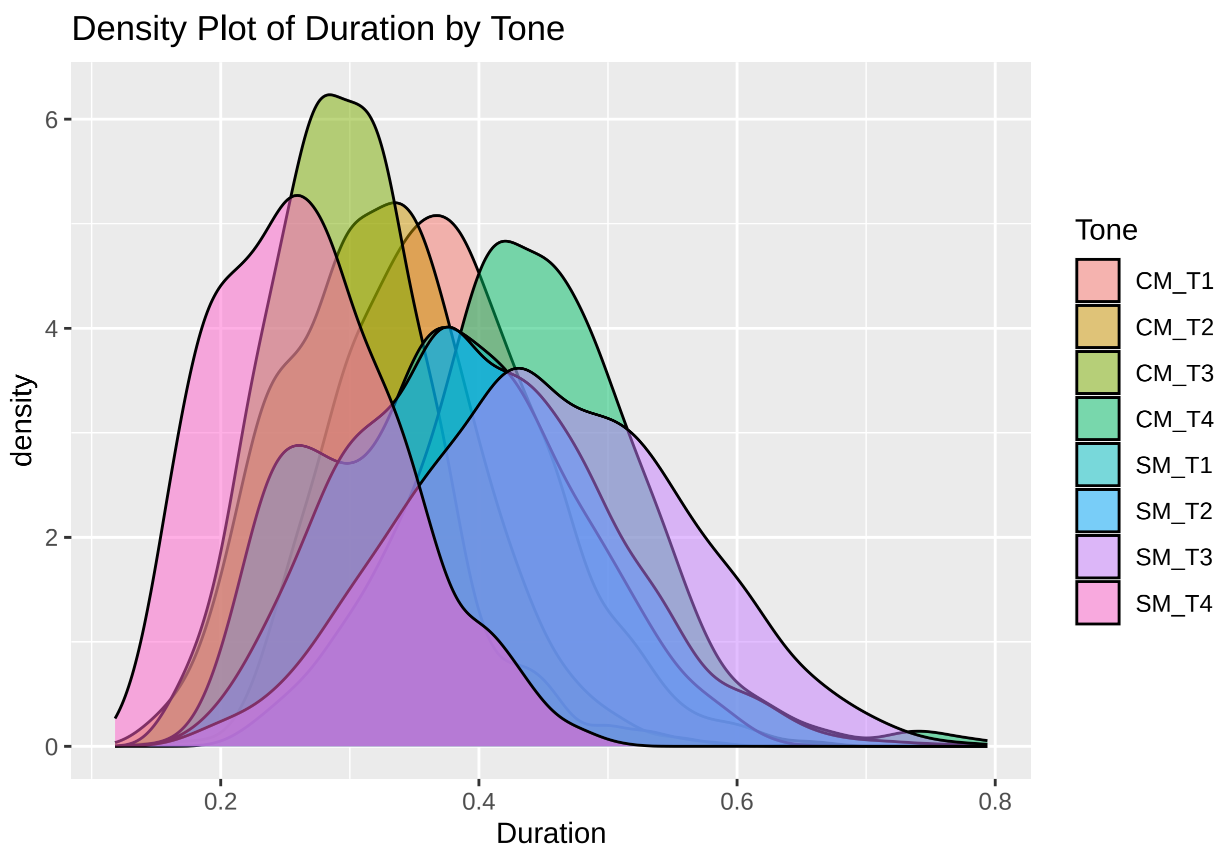

```
# Distribution of Gender
ggplot(final_df, aes(x = Gender)) +
  geom_bar(fill = "lightgreen", color = "black") +
  labs(title = "Distribution of Gender")
```




```r
# Distribution of Duration
ggplot(final_df, aes(x = Duration)) +
  geom_histogram(fill = "salmon", color = "black", bins = 20) +
  labs(title = "Distribution of Duration")
```




```
# Relationship between Duration and Tone
ggplot(final_df, aes(x = Duration, fill = Tone)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Duration by Tone")
```



Task 2:

Create a new dataset that only retains 2 or 3 of the 8 Tones. Briefly summarize the size, types and range of variables for the new dataset. (The goal is to form a simplified version of the problem, before eventually studying all 8 tones.)

```
# Choose 2  tones to retain
tones_to_retain <- c("CM_T1", "CM_T2")  # Adjust as needed

# Filter the original dataset to retain only the selected tones
df_subset <- subset(final_df, Tone %in% tones_to_retain)

# Summary of the new dataset
summary(df_subset)
```

```
##        X1              X2              X3              X4
##  Min.   : 39.0   Min.   : 35.0   Min.   : 37.1   Min.   : 44.5
##  1st Qu.:116.2   1st Qu.:116.3   1st Qu.:117.3   1st Qu.:117.7
##  Median :195.7   Median :194.4   Median :194.7   Median :193.8
##  Mean   :177.8   Mean   :176.9   Mean   :177.7   Mean   :178.4
##  3rd Qu.:222.0   3rd Qu.:220.0   3rd Qu.:221.3   3rd Qu.:222.5
##  Max.   :298.6   Max.   :295.6   Max.   :293.0   Max.   :294.0
##
##        X5              X6              X7              X8
##  Min.   : 37.7   Min.   : 34.2   Min.   : 36.4   Min.   : 30.5
##  1st Qu.:119.5   1st Qu.:120.3   1st Qu.:124.0   1st Qu.:124.9
```

```
##  Median :189.8   Median :182.6   Median :174.7   Median :167.8
##  Mean   :178.2   Mean   :177.4   Mean   :177.2   Mean   :175.9
##  3rd Qu.:224.3   3rd Qu.:225.5   3rd Qu.:227.2   3rd Qu.:230.5
##  Max.   :291.7   Max.   :295.6   Max.   :298.5   Max.   :298.1
##
##       X9              X10         Participant           Group
##  Min.   : 30.2   Min.   : 30.2   Length:2262        Length:2262
##  1st Qu.:123.3   1st Qu.:125.0   Class :character   Class :character
##  Median :167.4   Median :169.8   Mode  :character   Mode  :character
##  Mean   :176.2   Mean   :178.6
##  3rd Qu.:232.8   3rd Qu.:233.5
##  Max.   :297.9   Max.   :299.5
##
##     Gender             Duration          Tone
##  Length:2262        Min.   :0.1402   CM_T2  :1156
##  Class :character   1st Qu.:0.2928   CM_T1  :1106
##  Mode  :character   Median :0.3450   CM_T3  :   0
##                     Mean   :0.3477   CM_T4  :   0
##                     3rd Qu.:0.3980   SM_T1  :   0
##                     Max.   :0.6631   SM_T2  :   0
##                                      (Other):   0
```

```r
str(df_subset)
```

```
## 'data.frame':    2262 obs. of  15 variables:
##  $ X1         : num  101 97.3 110.6 122 110.9 ...
##  $ X2         : num  104 107 119 136 123 ...
##  $ X3         : num  256 182 130 152 134 ...
##  $ X4         : num  289 106 143 168 148 ...
##  $ X5         : num  144 140 159 174 162 ...
##  $ X6         : num  153 148 171 120 177 ...
##  $ X7         : num  159 156 134 96 179 ...
##  $ X8         : num  164.7 159.9 98.6 101.2 100.5 ...
##  $ X9         : num  170.4 166.5 100.5 101.7 95.8 ...
##  $ X10        : num  162.4 166.2 98.4 100.1 92.3 ...
##  $ Participant: chr  "CM1" "CM1" "CM1" "CM1" ...
##  $ Group      : chr  "CM" "CM" "CM" "CM" ...
##  $ Gender     : chr  "M" "M" "M" "M" ...
##  $ Duration   : num  0.384 0.371 0.363 0.33 0.354 ...
##  $ Tone       : Factor w/ 8 levels "CM_T1","CM_T2",..: 1 1 1 1 1 1 2 2 1 1 ...
```

Task 3:

Create an "enriched dataset" that includes (at least one, but can be as many as you want) new features that you think has potential in predicting Tone. Results from Task 2 may be inspiring.

```r
# Example: Adding a new feature by calculating the mean of frequencies
df_enriched <- final_df
df_enriched$Mean_Frequency <- rowMeans(final_df[,
                                                c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9",
                                       na.rm = TRUE)

# Summary of the enriched dataset
summary(df_enriched)
```

```
##        X1              X2              X3              X4       
##  Min.   : 37.9   Min.   : 34.4   Min.   : 30.2   Min.   : 30.1  
##  1st Qu.:146.6   1st Qu.:144.2   1st Qu.:140.0   1st Qu.:135.7  
##  Median :190.7   Median :186.6   Median :187.0   Median :189.5  
##  Mean   :186.9   Mean   :185.6   Mean   :184.9   Mean   :184.4  
##  3rd Qu.:227.0   3rd Qu.:227.1   3rd Qu.:229.1   3rd Qu.:228.6  
##  Max.   :299.3   Max.   :299.1   Max.   :298.7   Max.   :299.1  
##                                                                 
##        X5              X6              X7              X8       
##  Min.   : 30.1   Min.   : 30.7   Min.   : 30.4   Min.   : 30.1  
##  1st Qu.:132.7   1st Qu.:131.5   1st Qu.:130.8   1st Qu.:129.2  
##  Median :187.9   Median :183.6   Median :180.0   Median :177.4  
##  Mean   :182.3   Mean   :179.3   Mean   :177.5   Mean   :178.3  
##  3rd Qu.:226.3   3rd Qu.:220.8   3rd Qu.:219.2   3rd Qu.:226.6  
##  Max.   :299.6   Max.   :299.0   Max.   :298.7   Max.   :299.8  
##                                                                 
##        X9             X10         Participant           Group          
##  Min.   : 30.2   Min.   : 30.2   Length:9021        Length:9021       
##  1st Qu.:130.1   1st Qu.:130.9   Class :character   Class :character  
##  Median :181.0   Median :183.3   Mode  :character   Mode  :character  
##  Mean   :181.9   Mean   :184.1                                        
##  3rd Qu.:234.2   3rd Qu.:238.8                                        
##  Max.   :299.2   Max.   :299.5                                        
##                                                                       
##     Gender             Duration           Tone      Mean_Frequency  
##  Length:9021        Min.   :0.1181   SM_T4  :1273   Min.   : 60.34  
##  Class :character   1st Qu.:0.2855   SM_T1  :1241   1st Qu.:145.10  
##  Mode  :character   Median :0.3554   CM_T2  :1156   Median :188.54  
##                     Mean   :0.3635   SM_T2  :1147   Mean   :182.52  
##                     3rd Qu.:0.4318   CM_T1  :1106   3rd Qu.:217.55  
##                     Max.   :0.7940   SM_T3  :1104   Max.   :293.94  
##                                      (Other):1994                   
```

```r
str(df_enriched)
```

```
## 'data.frame':    9021 obs. of  16 variables:
##  $ X1            : num  182.3 123.4 96.7 101 97.3 ...
##  $ X2            : num  169 249 104 104 107 ...
##  $ X3            : num  222 114 81 256 182 ...
##  $ X4            : num  162 155 82 289 106 ...
##  $ X5            : num  271 173 128 144 140 ...
##  $ X6            : num  113 106 106 153 148 ...
##  $ X7            : num  120 110 108 159 156 ...
##  $ X8            : num  127 120 114 165 160 ...
##  $ X9            : num  134 126 122 170 166 ...
##  $ X10           : num  146 131 123 162 166 ...
##  $ Participant   : chr  "CM1" "CM1" "CM1" "CM1" ...
##  $ Group         : chr  "CM" "CM" "CM" "CM" ...
##  $ Gender        : chr  "M" "M" "M" "M" ...
##  $ Duration      : num  0.503 0.567 0.475 0.384 0.371 ...
##  $ Tone          : Factor w/ 8 levels "CM_T1","CM_T2",..: 4 4 4 1 1 3 3 1 1 1 ...
##  $ Mean_Frequency: num  165 141 106 170 143 ...
```

Task 4:

Pick one classfication method (e.g. Multinomial logistic, KNN, LDA) using old features, and then using enriched features. Compare results.

```r
# Set seed for reproducibility
set.seed(123)

# Generate random indices for splitting
indices <- sample(1:nrow(final_df), size = nrow(final_df), replace = FALSE)

# Calculate the number of rows for the training set (e.g., 80% of the data)
train_size <- round(0.8 * nrow(final_df))

# Split indices into training and testing indices
train_indices <- indices[1:train_size]
test_indices <- indices[(train_size + 1):nrow(final_df)]

# Split the data using the indices
train_data <- final_df[train_indices, ]
test_data <- final_df[test_indices, ]
```

Task 5:

Perform some kind of feature selection for the above method

```r
# Example: Using random forest classifier with old features
rf_model_old <- randomForest(Tone ~ ., data = train_data)

predictions <- predict(rf_model_old, test_data)
(Confusion_matrix_old <- table(Observed = test_data$Tone, Predicted = predictions))
```

```
##          Predicted
## Observed CM_T1 CM_T2 CM_T3 CM_T4 SM_T1 SM_T2 SM_T3 SM_T4
##    CM_T1   192     2     5     5     0     0     0     0
##    CM_T2     0   199    15     4     0     0     0     0
##    CM_T3     2    31   178     3     0     0     0     0
##    CM_T4    14     7     7   147     0     0     0     0
##    SM_T1     0     0     0     0   248     1     2     3
##    SM_T2     0     0     0     0     2   226     9     0
##    SM_T3     0     0     0     0     1     8   213     7
##    SM_T4     0     0     0     0     2     0     3   268
```

```r
# Example: Using random forest classifier with enriched features
rf_model_enriched <- randomForest(Tone ~ ., data = train_data)

predictions <- predict(rf_model_enriched, test_data)
(Confusion_matrix_enriched <- table(Observed = test_data$Tone, Predicted = predictions))
```

```
##          Predicted
## Observed CM_T1 CM_T2 CM_T3 CM_T4 SM_T1 SM_T2 SM_T3 SM_T4
##    CM_T1   193     3     4     4     0     0     0     0
##    CM_T2     0   201    13     4     0     0     0     0
##    CM_T3     3    28   180     3     0     0     0     0
##    CM_T4    15     6     7   147     0     0     0     0
```

```
##    SM_T1     0     0     0     0   248     1     2     3
##    SM_T2     0     0     0     0     3   226     8     0
##    SM_T3     0     0     0     0     0     8   214     7
##    SM_T4     0     0     0     0     2     0     2   269
```

```r
# Example: Using variable importance from random forest model
importance <- importance(rf_model_enriched)
varImportance <- data.frame(Variables = row.names(importance),
                            Importance = round(importance[, "MeanDecreaseGini"], 2))
varImportance <- varImportance[order(varImportance$Importance, decreasing = TRUE), ]

# Select top features
top_features <- varImportance$Variables[1:5]  # Select top 5 features, adjust as needed

# Subset dataset with top features
df_selected <- df_enriched[, c("Tone", top_features)]

# Train classifier with selected features
rf_model_selected <- randomForest(Tone ~ ., data = train_data[, c("Tone", top_features)])

predictions <- predict(rf_model_selected, test_data)
(Confusion_matrix_selected <- table(Observed = test_data$Tone, Predicted = predictions))
```

```
##          Predicted
## Observed CM_T1 CM_T2 CM_T3 CM_T4 SM_T1 SM_T2 SM_T3 SM_T4
##    CM_T1   175     8    10    11     0     0     0     0
##    CM_T2     4   194    13     7     0     0     0     0
##    CM_T3     9    31   166     8     0     0     0     0
##    CM_T4     9     8    10   148     0     0     0     0
##    SM_T1     0     0     0     0   243     2     3     6
##    SM_T2     0     0     0     0     2   214    18     3
##    SM_T3     0     0     0     0     2    14   206     7
##    SM_T4     0     0     0     0     6     7     5   255
```

In the context of classification models, a confusion matrix provides a summary of the performance of the model on a test dataset. It presents the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by the model:

True Positive (TP): The model correctly predicts instances of the positive class (e.g., correctly predicts a tone as Tone1 when it actually is Tone1).

True Negative (TN): The model correctly predicts instances of the negative class (e.g., correctly predicts a tone as not being Tone1 when it is actually not Tone1).

False Positive (FP): The model incorrectly predicts instances of the negative class as the positive class (e.g., predicts a tone as Tone1 when it is actually not Tone1).

False Negative (FN): The model incorrectly predicts instances of the positive class as the negative class (e.g., predicts a tone as not being Tone1 when it is actually Tone1).

Metric Interpretation:

- Accuracy: Overall accuracy of the model in correctly predicting tones. It is calculated as (TP + TN) / (TP + TN + FP + FN). That is the proportion of correctly classified instances out of all instances. Higher accuracy indicates better overall performance.

- Precision: Proportion of correctly predicted positive cases out of all predicted positive cases. It is calculated as TP / (TP + FP). That is the proportion of true positive predictions out of all positive predictions made by the model. Higher precision indicates fewer false positives, meaning the model is making fewer incorrect positive predictions.
- Sensitivity: Proportion of correctly predicted positive cases out of all actual positive cases. It is calculated as TP / (TP + FN).That is the proportion of true positive predictions out of all actual positive instances. Higher sensitivity indicates that the model is better at capturing all instances of the positive class, reducing false negatives.
- Specificity: Proportion of correctly predicted negative cases out of all actual negative cases. It is calculated as TN / (TN + FP). That is the proportion of true negative predictions out of all actual negative instances. Higher specificity indicates that the model is better at identifying instances that are not of the positive class, reducing false positives.
- F1 Score: Harmonic mean of precision and recall. It provides a balance between precision and recall. It is calculated as 2 * (Precision * Recall) / (Precision + Recall). A higher F1 score indicates better balance between precision and recall.

```r
# Define function to calculate metrics
calculate_metrics <- function(confusion_matrix) {
  # Total counts for each class
  total_actual <- colSums(confusion_matrix)
  total_predicted <- rowSums(confusion_matrix)

  # True positives for each class
  true_positives <- diag(confusion_matrix)

  # False positives for each class
  false_positives <- total_predicted - true_positives

  # False negatives for each class
  false_negatives <- total_actual - true_positives

  # True negatives for each class
  true_negatives <- sum(confusion_matrix) - true_positives -
    false_positives - false_negatives

  # Accuracy for each class
  accuracy <- (true_positives + true_negatives) /
    (true_positives + true_negatives + false_positives + false_negatives)

  # Precision for each class
  precision <- true_positives / (true_positives + false_positives)

  # Sensitivity (recall) for each class
  sensitivity <- true_positives / total_actual

  # Specificity for each class
  specificity <- true_negatives / (true_negatives + false_positives)

  # F1 score for each class
  f1_score <- 2 * (precision * sensitivity) / (precision + sensitivity)

  # Combine metrics into a data frame
  metrics <- data.frame(Tone = rownames(confusion_matrix),
```

```r
                          Accuracy = accuracy,
                          Precision = precision,
                          Sensitivity = sensitivity,
                          Specificity = specificity,
                          F1_Score = f1_score)

  return(metrics)
}

# Calculate metrics
calculate_metrics(Confusion_matrix_old)
```

```
##         Tone  Accuracy Precision Sensitivity Specificity  F1_Score
## CM_T1 CM_T1 0.9844789 0.9411765   0.9230769   0.9924812 0.9320388
## CM_T2 CM_T2 0.9672949 0.9128440   0.8326360   0.9878594 0.8708972
## CM_T3 CM_T3 0.9650776 0.8317757   0.8682927   0.9774859 0.8496420
## CM_T4 CM_T4 0.9778271 0.8400000   0.9245283   0.9829787 0.8802395
## SM_T1 SM_T1 0.9939024 0.9763780   0.9802372   0.9961315 0.9783037
## SM_T2 SM_T2 0.9889135 0.9535865   0.9617021   0.9929892 0.9576271
## SM_T3 SM_T3 0.9833703 0.9301310   0.9383260   0.9898542 0.9342105
## SM_T4 SM_T4 0.9916851 0.9816850   0.9640288   0.9967235 0.9727768
```

```r
calculate_metrics(Confusion_matrix_enriched)
```

```
##         Tone  Accuracy Precision Sensitivity Specificity  F1_Score
## CM_T1 CM_T1 0.9839246 0.9460784   0.9146919   0.9930948 0.9301205
## CM_T2 CM_T2 0.9700665 0.9220183   0.8445378   0.9891443 0.8815789
## CM_T3 CM_T3 0.9678492 0.8411215   0.8823529   0.9787500 0.8612440
## CM_T4 CM_T4 0.9783814 0.8400000   0.9303797   0.9829891 0.8828829
## SM_T1 SM_T1 0.9939024 0.9763780   0.9802372   0.9961315 0.9783037
## SM_T2 SM_T2 0.9889135 0.9535865   0.9617021   0.9929892 0.9576271
## SM_T3 SM_T3 0.9850333 0.9344978   0.9469027   0.9904943 0.9406593
## SM_T4 SM_T4 0.9922395 0.9853480   0.9641577   0.9973770 0.9746377
```

```r
calculate_metrics(Confusion_matrix_selected)
```

```
##         Tone  Accuracy Precision Sensitivity Specificity  F1_Score
## CM_T1 CM_T1 0.9717295 0.8578431   0.8883249   0.9819540 0.8728180
## CM_T2 CM_T2 0.9606430 0.8899083   0.8049793   0.9846449 0.8453159
## CM_T3 CM_T3 0.9550998 0.7757009   0.8341709   0.9700935 0.8038741
## CM_T4 CM_T4 0.9706208 0.8457143   0.8505747   0.9834356 0.8481375
## SM_T1 SM_T1 0.9883592 0.9566929   0.9604743   0.9929078 0.9585799
## SM_T2 SM_T2 0.9745011 0.9029536   0.9029536   0.9853223 0.9029536
## SM_T3 SM_T3 0.9728381 0.8995633   0.8879310   0.9853690 0.8937093
## SM_T4 SM_T4 0.9811530 0.9340659   0.9409594   0.9882583 0.9375000
```

Conclusion:

Therefore, given the high values observed for all metrics across all cases, we can assert that the classification demonstrates robust performance. However, the scenario involving the selected features shows comparatively lower performance in contrast to the cases involving old and enriched features. This suggests the necessity of being more cautious during the feature selection process for this dataset. Henceforth, future efforts may involve exploring alternative methodologies for feature selection to enhance the efficacy of the model.