Lung Cancer

Liam Murphy

2024-11-14

```r
# View the structure of the dataset
str(lc_data)
```

```
## 'data.frame':    309 obs. of  16 variables:
##  $ GENDER               : chr  "M" "M" "F" "M" ...
##  $ AGE                  : int  69 74 59 63 63 75 52 51 68 53 ...
##  $ SMOKING              : int  1 2 1 2 1 1 2 2 2 2 ...
##  $ YELLOW_FINGERS       : int  2 1 1 2 2 2 1 2 1 2 ...
##  $ ANXIETY              : int  2 1 1 2 1 1 1 2 2 2 ...
##  $ PEER_PRESSURE        : int  1 1 2 1 1 1 1 2 1 2 ...
##  $ CHRONIC.DISEASE      : int  1 2 1 1 1 2 1 1 1 2 ...
##  $ FATIGUE              : int  2 2 2 1 1 2 2 2 2 1 ...
##  $ ALLERGY              : int  1 2 1 1 1 2 1 2 1 2 ...
##  $ WHEEZING             : int  2 1 2 1 2 2 2 1 1 1 ...
##  $ ALCOHOL.CONSUMING    : int  2 1 1 2 1 1 2 1 1 2 ...
##  $ COUGHING             : int  2 1 2 1 2 2 2 1 1 1 ...
##  $ SHORTNESS.OF.BREATH  : int  2 2 2 1 2 2 2 2 1 1 ...
##  $ SWALLOWING.DIFFICULTY: int  2 2 1 2 1 1 1 2 1 2 ...
##  $ CHEST.PAIN           : int  2 2 2 2 1 1 2 1 1 2 ...
##  $ LUNG_CANCER          : chr  "YES" "YES" "NO" "NO" ...
```

```r
# Summary of the dataset
summary(lc_data)
```

```
##     GENDER                AGE           SMOKING      YELLOW_FINGERS
##  Length:309         Min.   :21.00   Min.   :1.000   Min.   :1.00
##  Class :character   1st Qu.:57.00   1st Qu.:1.000   1st Qu.:1.00
##  Mode  :character   Median :62.00   Median :2.000   Median :2.00
##                     Mean   :62.67   Mean   :1.563   Mean   :1.57
##                     3rd Qu.:69.00   3rd Qu.:2.000   3rd Qu.:2.00
##                     Max.   :87.00   Max.   :2.000   Max.   :2.00
##     ANXIETY      PEER_PRESSURE   CHRONIC.DISEASE    FATIGUE
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000
##  Median :1.000   Median :2.000   Median :2.000   Median :2.000
##  Mean   :1.498   Mean   :1.502   Mean   :1.505   Mean   :1.673
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :2.000
##     ALLERGY         WHEEZING     ALCOHOL.CONSUMING    COUGHING
##  Min.   :1.000   Min.   :1.000   Min.   :1.000     Min.   :1.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000     1st Qu.:1.000
```

```
##  Median :2.000   Median :2.000   Median :2.000     Median :2.000
##  Mean   :1.557   Mean   :1.557   Mean   :1.557     Mean   :1.579
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000     3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000   Max.   :2.000     Max.   :2.000
##  SHORTNESS.OF.BREATH SWALLOWING.DIFFICULTY   CHEST.PAIN    LUNG_CANCER
##  Min.   :1.000       Min.   :1.000         Min.   :1.000   Length:309
##  1st Qu.:1.000       1st Qu.:1.000         1st Qu.:1.000   Class :character
##  Median :2.000       Median :1.000         Median :2.000   Mode  :character
##  Mean   :1.641       Mean   :1.469         Mean   :1.557
##  3rd Qu.:2.000       3rd Qu.:2.000         3rd Qu.:2.000
##  Max.   :2.000       Max.   :2.000         Max.   :2.000
```

```r
# Check for NAs
colSums(is.na(lc_data))
```

```
##                GENDER                   AGE               SMOKING
##                     0                     0                     0
##        YELLOW_FINGERS               ANXIETY         PEER_PRESSURE
##                     0                     0                     0
##       CHRONIC.DISEASE               FATIGUE               ALLERGY
##                     0                     0                     0
##              WHEEZING     ALCOHOL.CONSUMING              COUGHING
##                     0                     0                     0
##   SHORTNESS.OF.BREATH SWALLOWING.DIFFICULTY            CHEST.PAIN
##                     0                     0                     0
##           LUNG_CANCER
##                     0
```

```r
# Convert character columns/integer to factors for classification

lc_data <- data.frame(lapply(lc_data, function(x) {
  if (is.integer(x) || is.character(x)) {
    as.factor(x)  # Convert integer and character columns to factors
  } else {
    x  # Keep other column types unchanged
  }
}))

# If Age is a factor and needs to be numeric, convert it by first ensuring the levels are numeric
lc_data$AGE <- as.numeric(as.character(lc_data$AGE))  # Convert factor to numeric properly

# Recode 1 to "No" and 2 to "Yes" for all relevant columns
lc_data <- lc_data %>%
  mutate(across(where(~ all(. %in% c(1, 2))),
                ~ recode(.x, `1` = "No", `2` = "Yes")))

# Check structure after change
str(lc_data)
```

```
## 'data.frame':    309 obs. of  16 variables:
##  $ GENDER               : Factor w/ 2 levels "F","M": 2 2 1 2 1 1 2 1 1 2 ...
##  $ AGE                  : num  69 74 59 63 63 75 52 51 68 53 ...
##  $ SMOKING              : Factor w/ 2 levels "No","Yes": 1 2 1 2 1 1 2 2 2 2 ...
```

```
##  $ YELLOW_FINGERS       : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 1 2 1 2 ...
##  $ ANXIETY              : Factor w/ 2 levels "No","Yes": 2 1 1 2 1 1 1 2 2 2 ...
##  $ PEER_PRESSURE        : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 2 1 2 ...
##  $ CHRONIC.DISEASE      : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
##  $ FATIGUE              : Factor w/ 2 levels "No","Yes": 2 2 2 1 1 2 2 2 2 1 ...
##  $ ALLERGY              : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 2 ...
##  $ WHEEZING             : Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 2 1 1 1 ...
##  $ ALCOHOL.CONSUMING    : Factor w/ 2 levels "No","Yes": 2 1 1 2 1 1 2 1 1 2 ...
##  $ COUGHING             : Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 2 1 1 1 ...
##  $ SHORTNESS.OF.BREATH  : Factor w/ 2 levels "No","Yes": 2 2 2 1 2 2 2 2 1 1 ...
##  $ SWALLOWING.DIFFICULTY: Factor w/ 2 levels "No","Yes": 2 2 1 2 1 1 1 2 1 2 ...
##  $ CHEST.PAIN           : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 1 2 1 1 2 ...
##  $ LUNG_CANCER          : Factor w/ 2 levels "NO","YES": 2 2 1 1 1 2 2 2 1 2 ...
```

```
set.seed(123)
trainIndex <- createDataPartition(lc_data$LUNG_CANCER, p = 0.7, list = FALSE)  # specify the response v
trainData <- lc_data[trainIndex, ]
testData <- lc_data[-trainIndex, ]
```

```
# Ensure that AGE is numeric
lc_data$AGE <- as.numeric(lc_data$AGE)

# Filter the dataset for males who are positive for lung cancer
males_lung_cancer <- subset(lc_data, GENDER == "M" & LUNG_CANCER == "YES")


# Filter the dataset for females who are positive for lung cancer
females_lung_cancer <- subset(lc_data, GENDER == "F" & LUNG_CANCER == "YES")

# Plot the age distribution for these individuals

Age_Females_With_Lung_Cancer  <- ggplot(females_lung_cancer, aes(x = AGE)) +
  geom_histogram(binwidth = 1, fill = "pink", color = "black", alpha = 0.7, stat = "count") +
  labs(title = "Age Distribution for Females Positive for Lung Cancer",
       x = "Age",
       y = "Frequency") +
  theme_minimal()
```

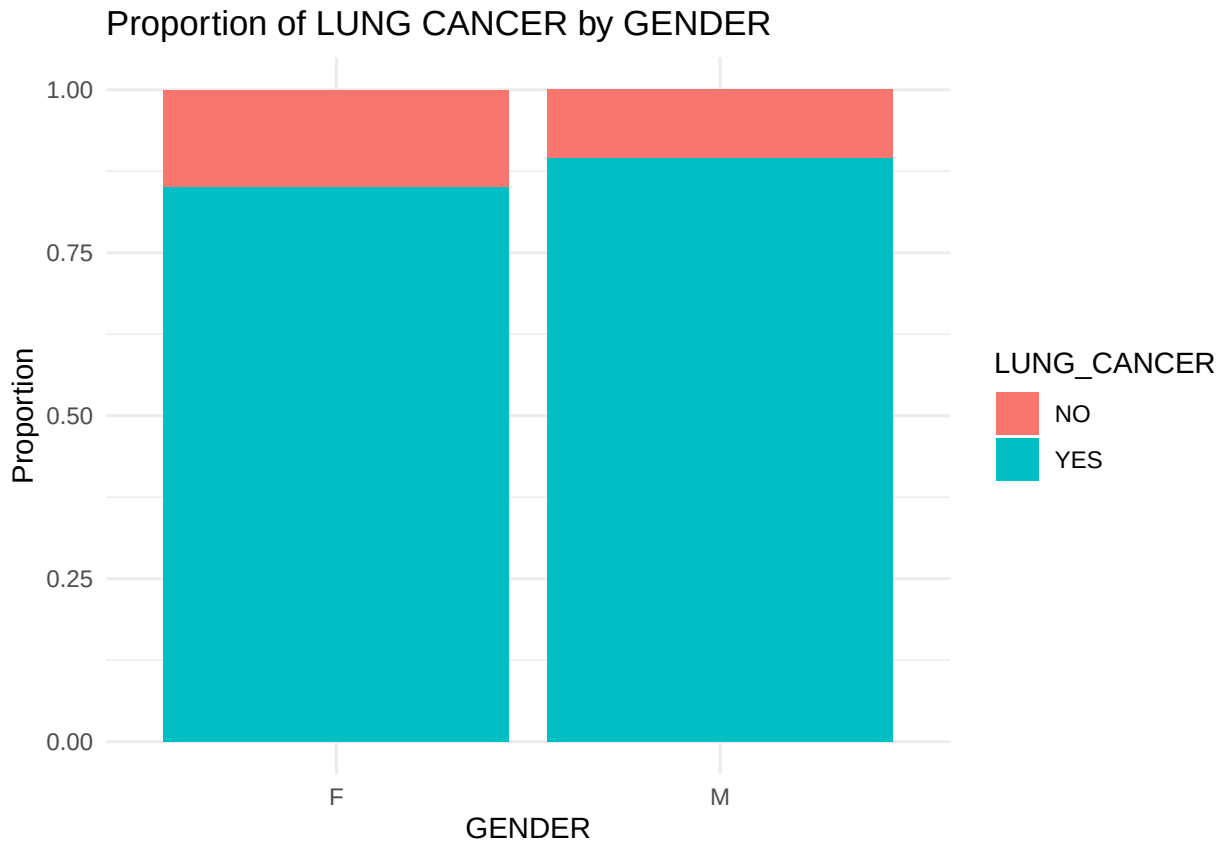
```
## Warning in geom_histogram(binwidth = 1, fill = "pink", color = "black", :
## Ignoring unknown parameters: ‘binwidth‘, ‘bins‘, and ‘pad‘
```

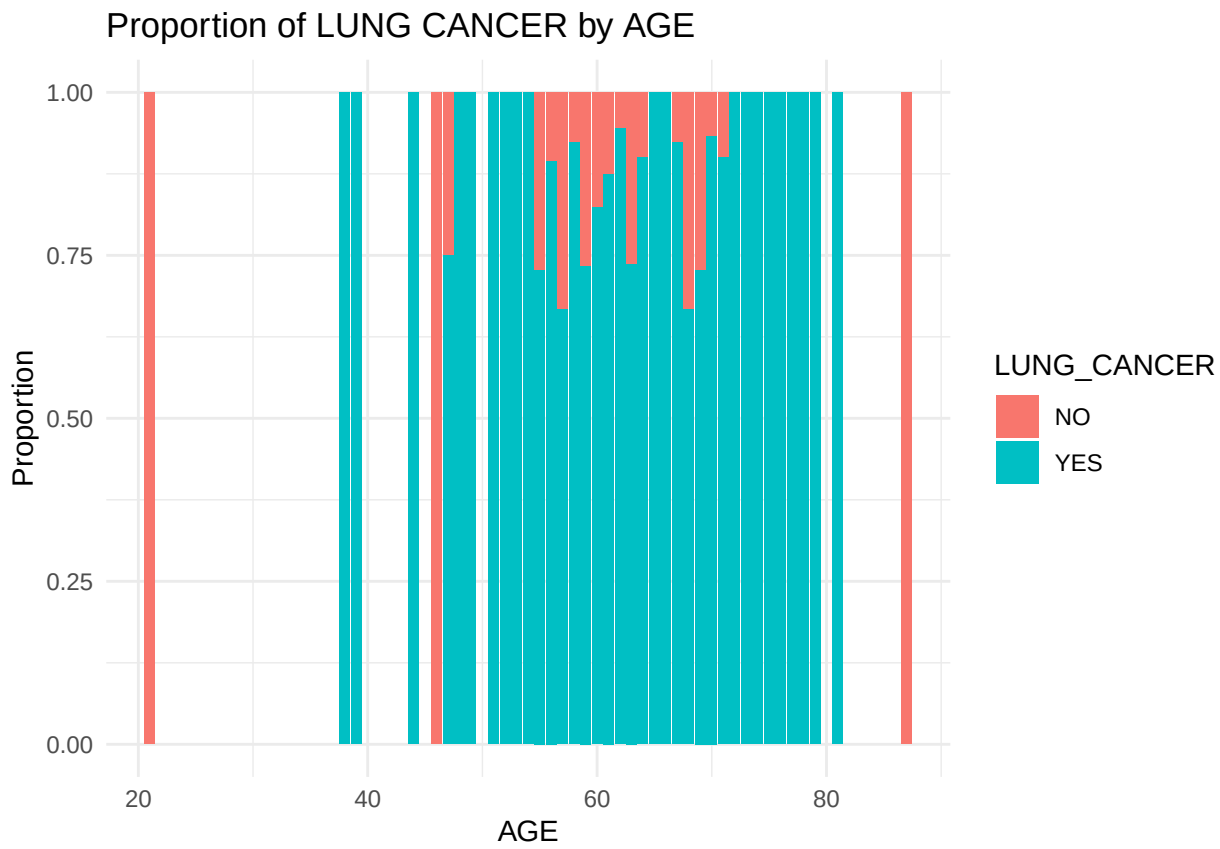
Explore bivariate relationships with lung cancer.

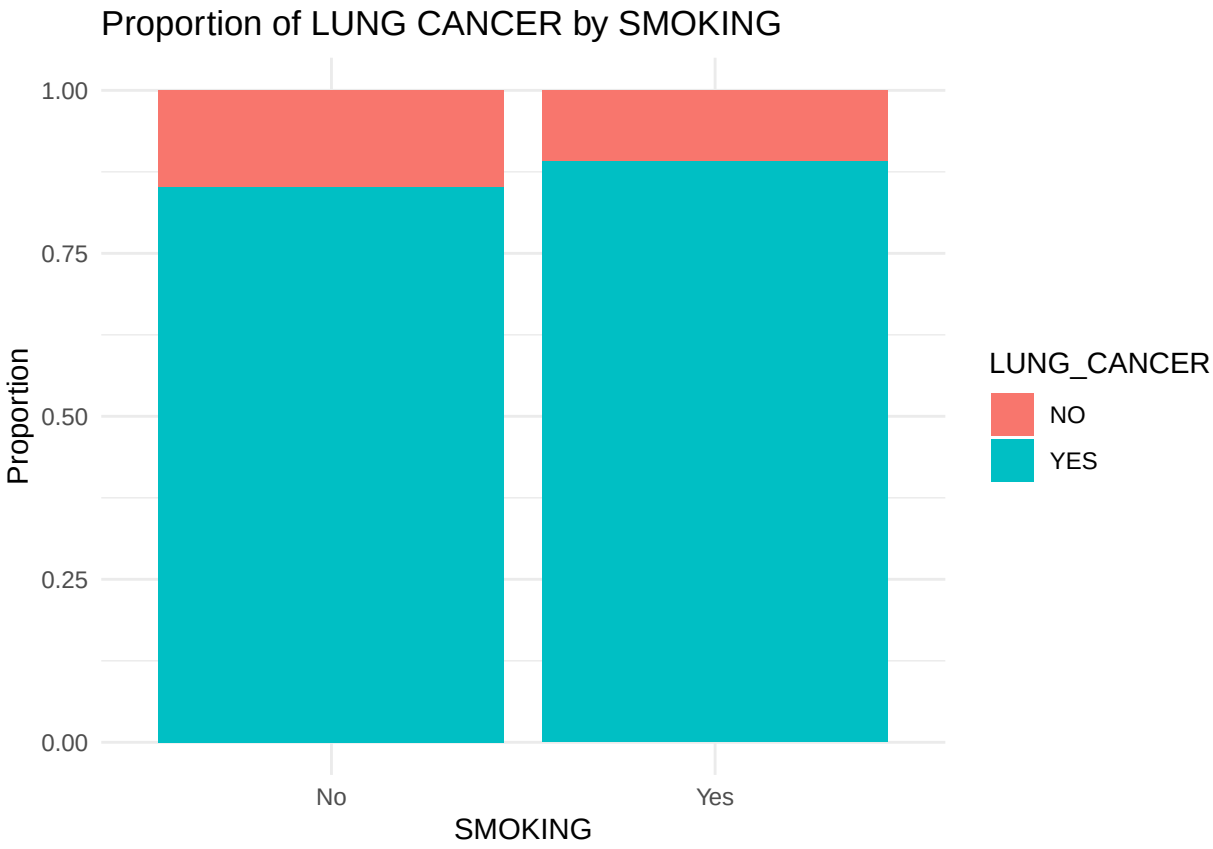
```
# Create a directory to store the plots (if it doesn't exist)
output_dir <- "Figures"
if (!dir.exists("C:/Users/liamm/OneDrive - University of Massachusetts/Documents/R Projects/Lung Cancer
  dir.create("C:/Users/liamm/OneDrive - University of Massachusetts/Documents/R Projects/Lung Cancer Cl
}
# Loop through each variable and save the plot
 for (var in colnames(lc_data)) {
  if (var != "LUNG_CANCER") {
    p <- ggplot(lc_data, aes_string(x = var, fill = "LUNG_CANCER")) +
      geom_bar(position = "fill") +
```

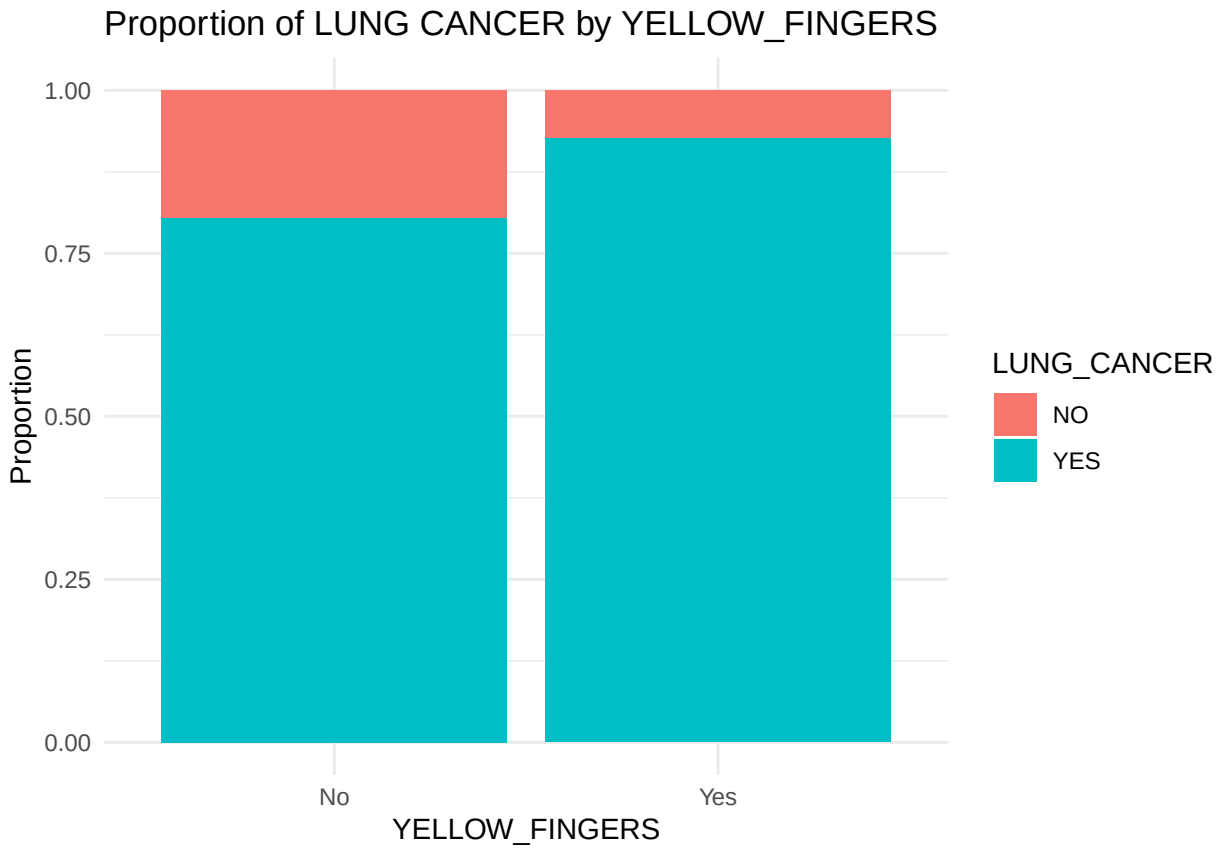
```
      labs(title = paste("Proportion of LUNG CANCER by", var), x = var, y = "Proportion") +
      theme_minimal() +
    theme(
        panel.background = element_rect(fill = "white", color = NA),  # White panel background
        plot.background = element_rect(fill = "white", color = NA)
        )# White plot background
    # Define the filename
    filename <- file.path(output_dir, paste0("plot_", var, ".png"))

    # Save the plot
    ggsave(filename, plot = p, width = 6, height = 4, dpi = 300)

    print(p)
  }
}
```

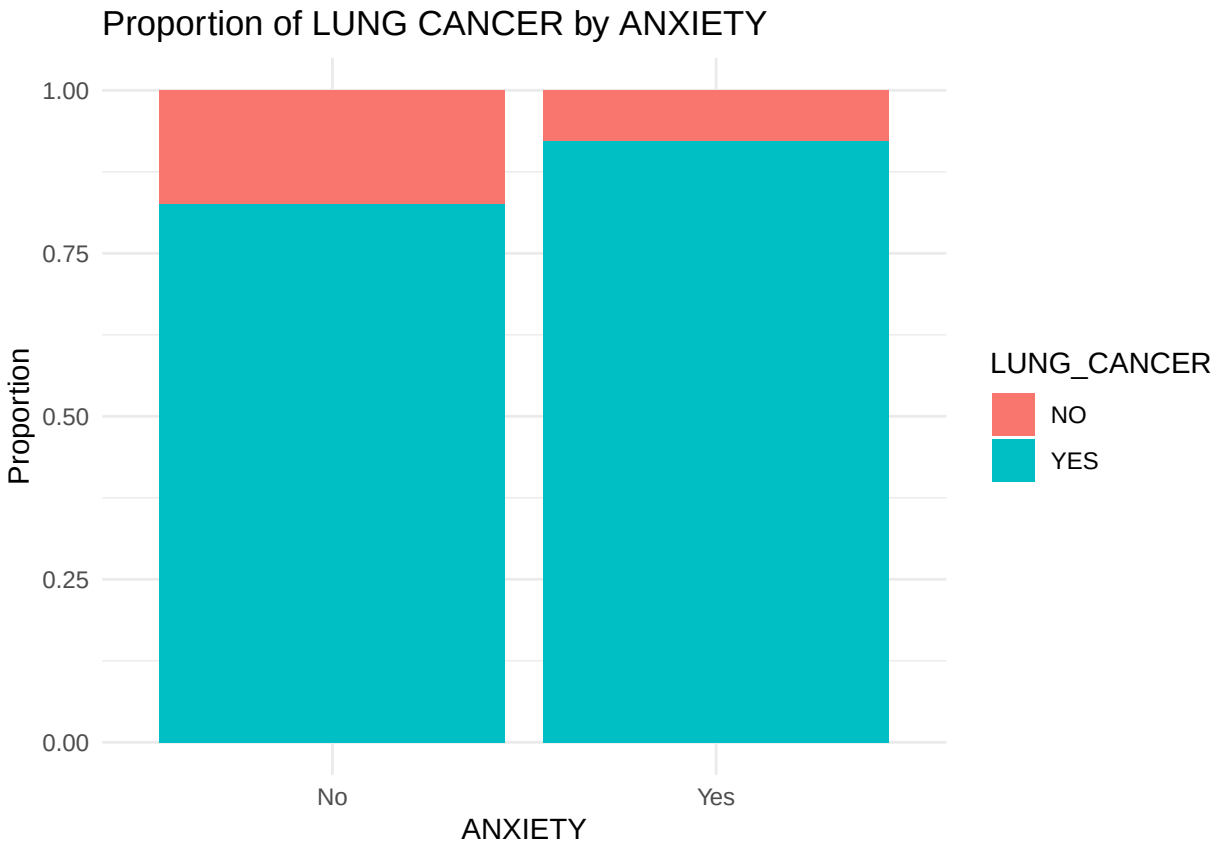
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

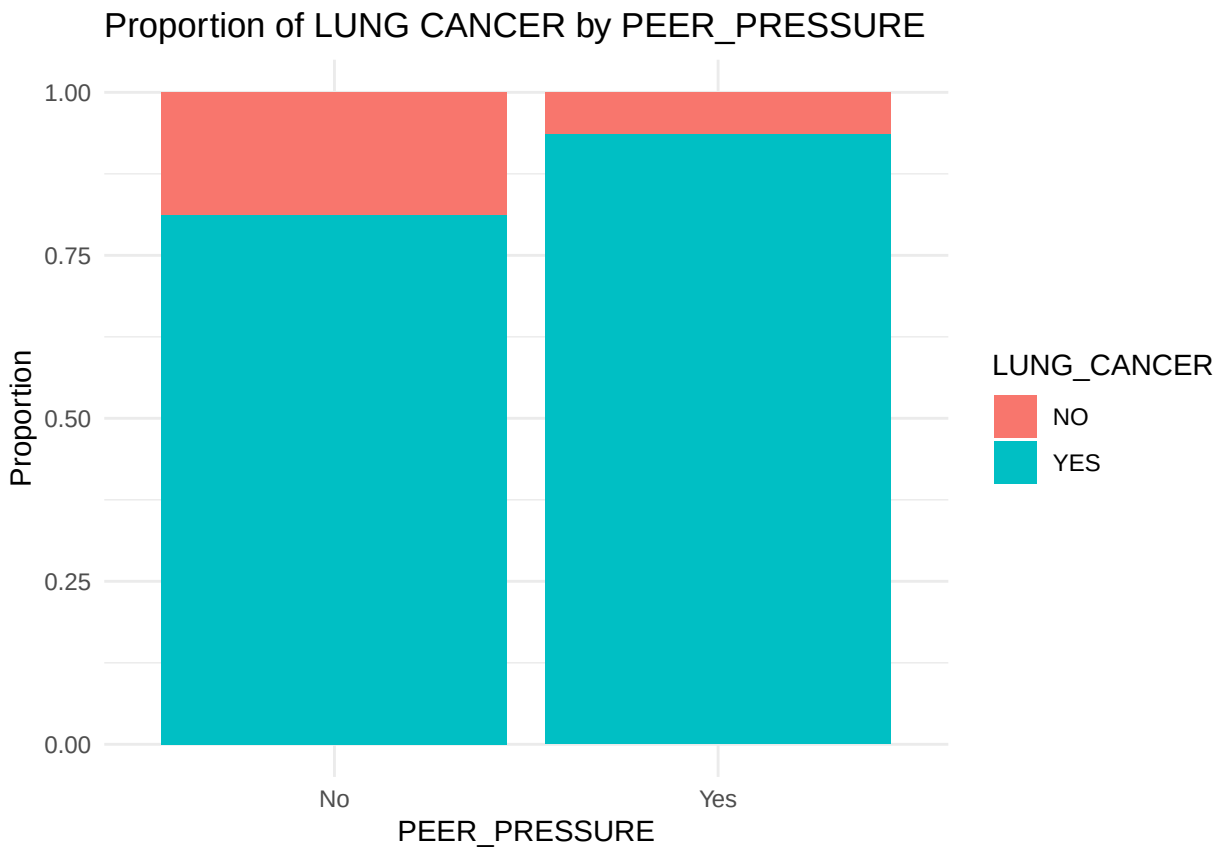


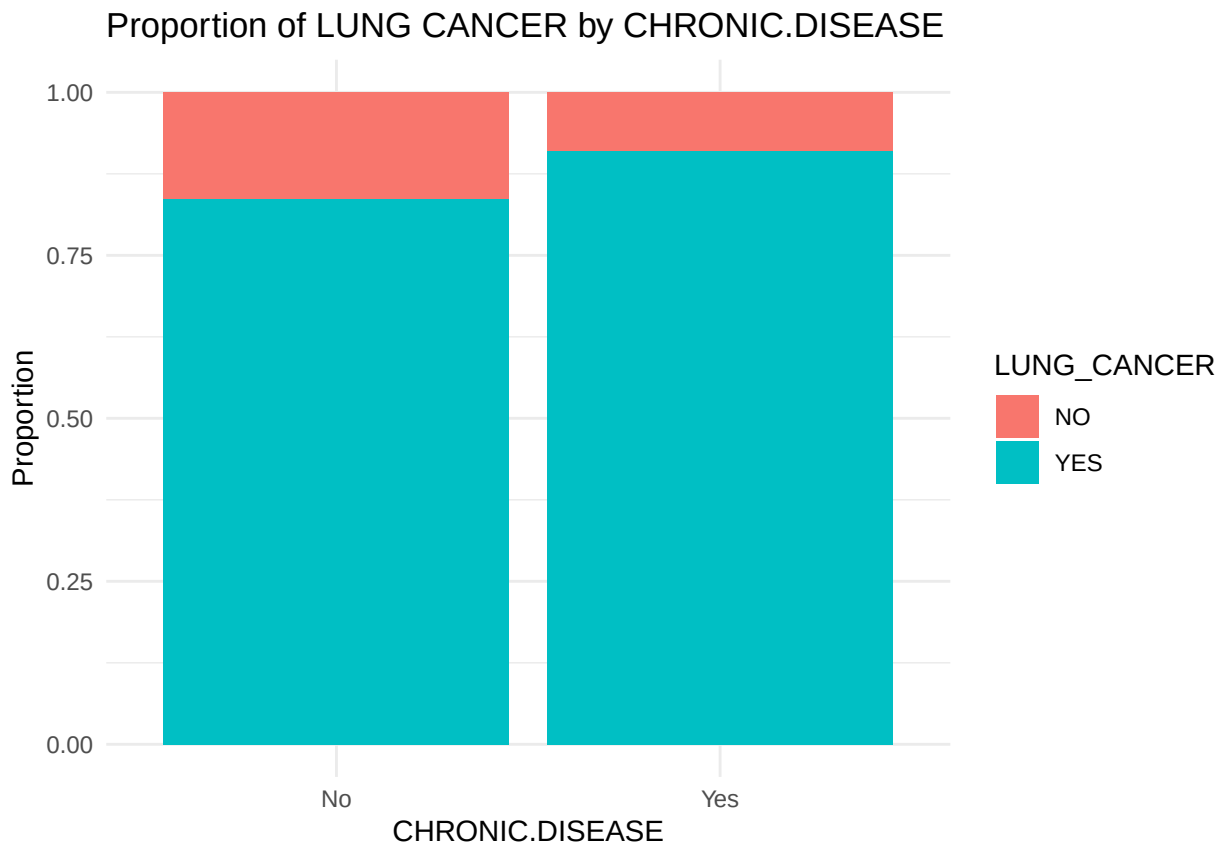

Proportion of LUNG CANCER by AGE
1.00
0.75
0.50
0.25
0.00
Proportion
20
40
60
80
AGE
LUNG_CANCER
NO
YES

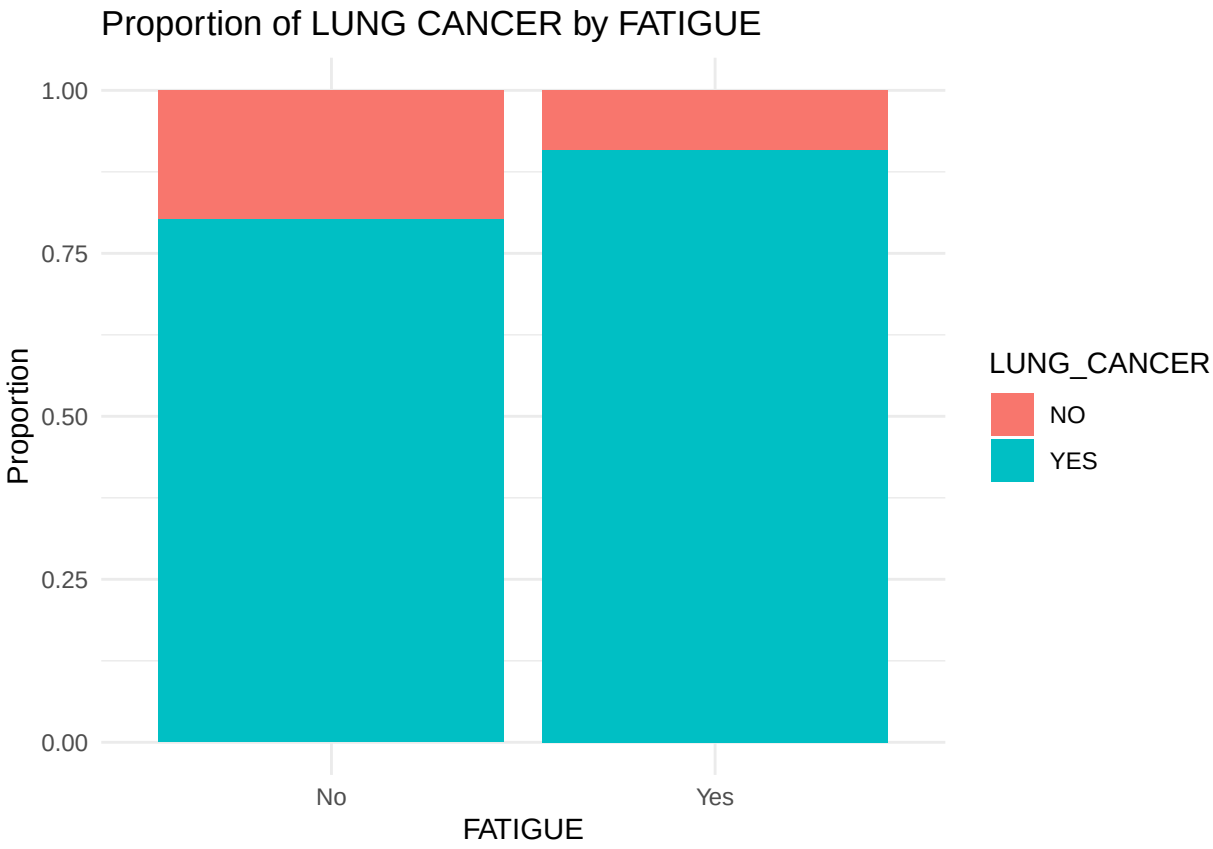

Proportion of LUNG CANCER by SMOKING
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
SMOKING
LUNG_CANCER
NO
YES

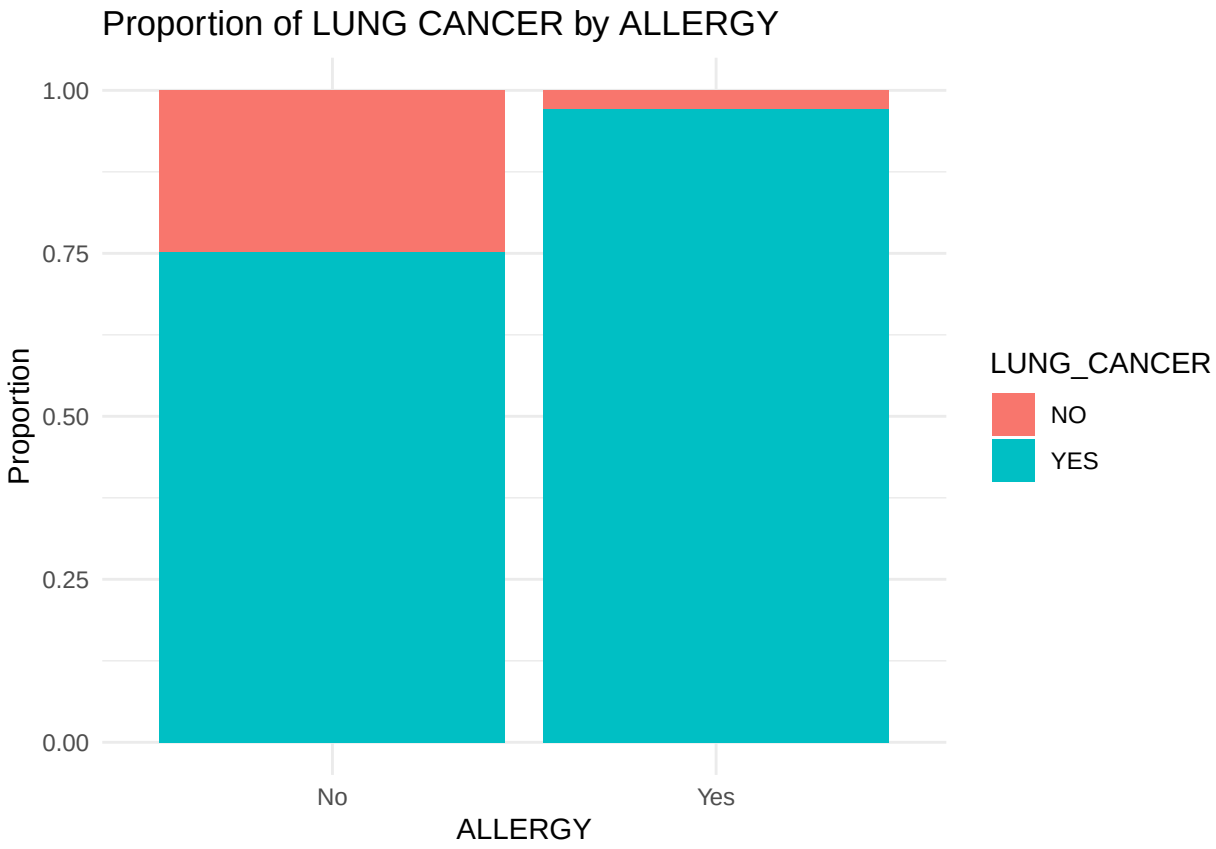

Proportion of LUNG CANCER by YELLOW_FINGERS
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
YELLOW_FINGERS
LUNG_CANCER
NO
YES

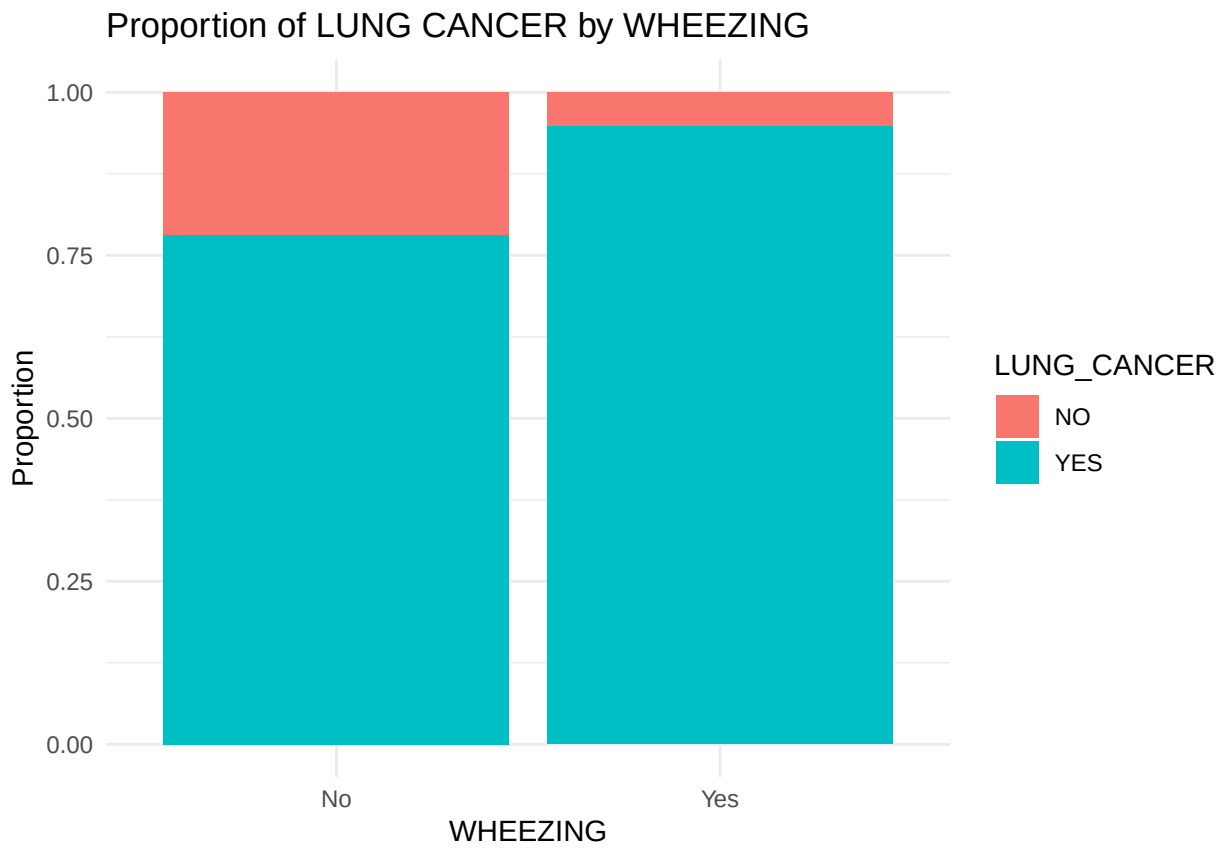

Proportion of LUNG CANCER by ANXIETY
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
ANXIETY
LUNG_CANCER
NO
YES

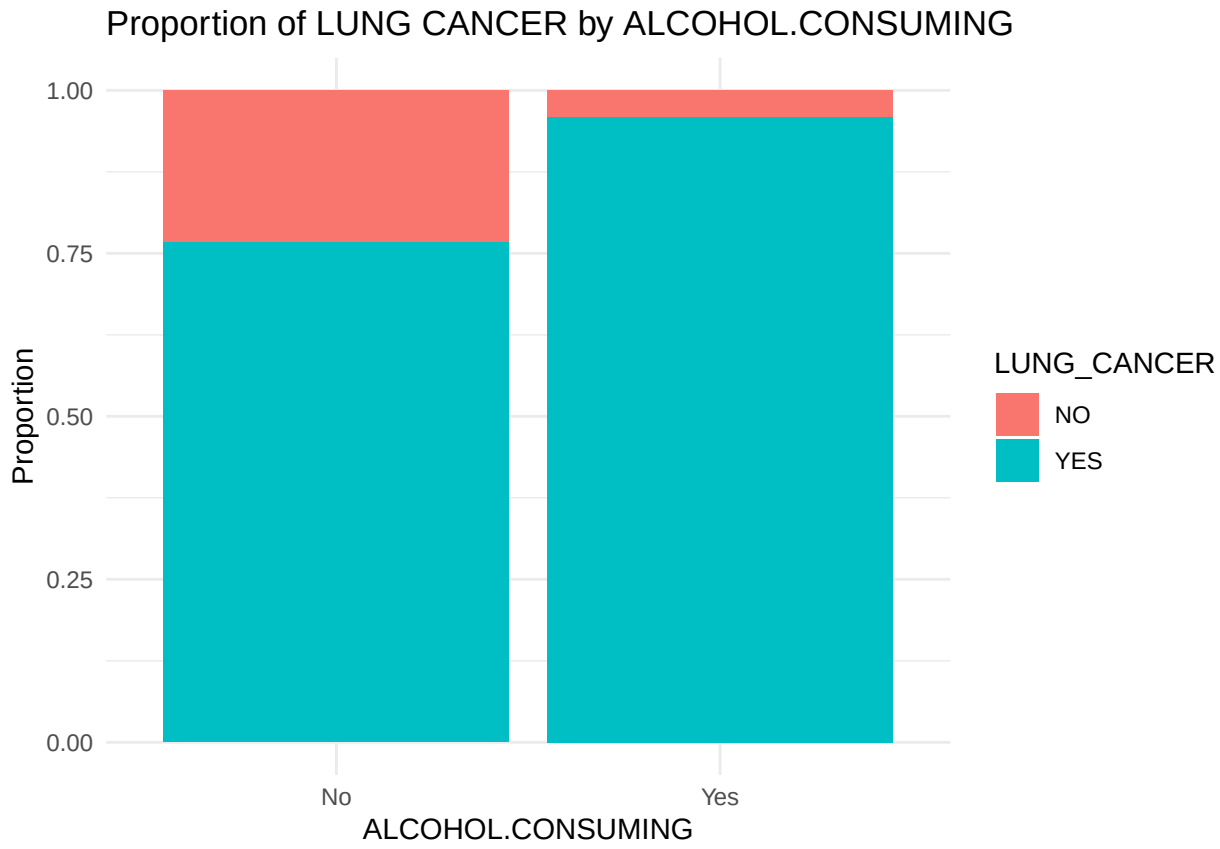

Proportion of LUNG CANCER by PEER_PRESSURE
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
PEER_PRESSURE
LUNG_CANCER
NO
YES

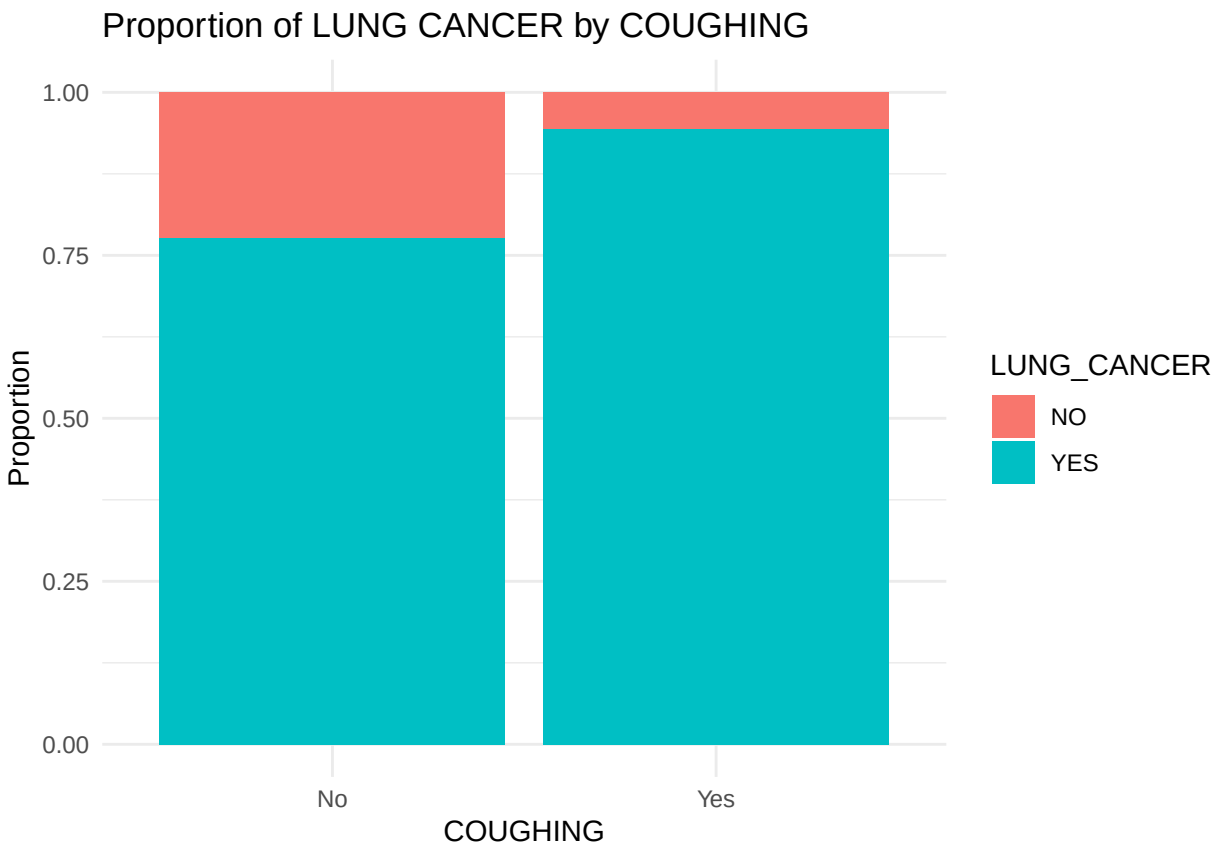

Proportion of LUNG CANCER by CHRONIC.DISEASE
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
CHRONIC.DISEASE
LUNG_CANCER
NO
YES

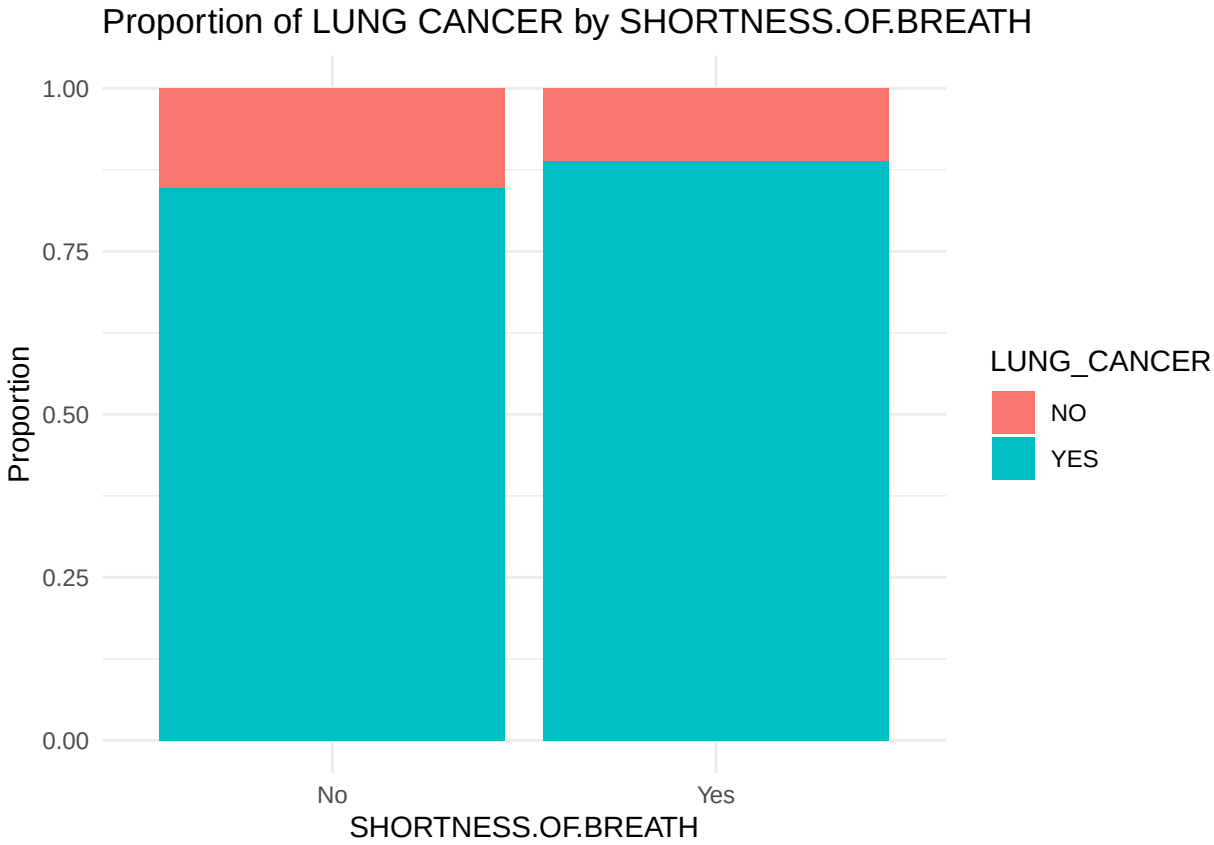

Proportion of LUNG CANCER by FATIGUE
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
FATIGUE
LUNG_CANCER
NO
YES

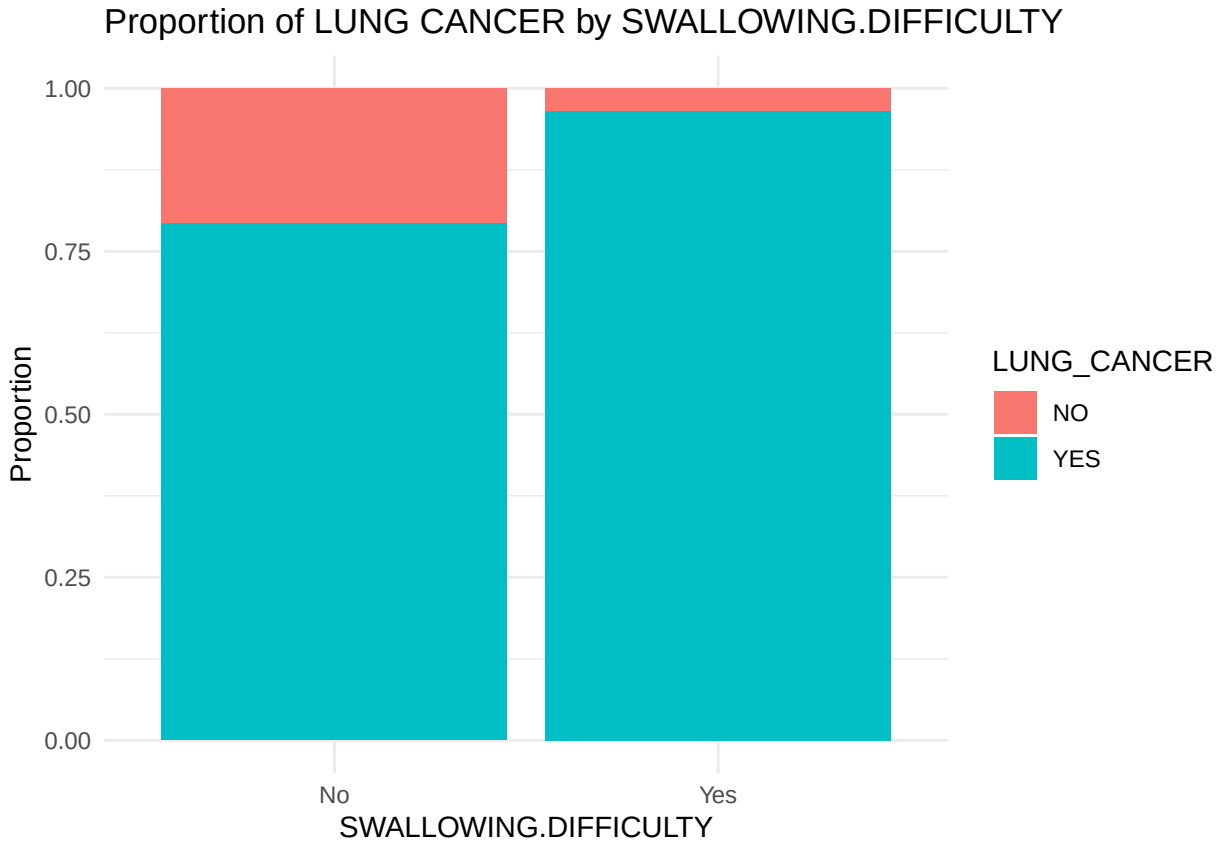

Proportion of LUNG CANCER by ALLERGY
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
ALLERGY
LUNG_CANCER
NO
YES

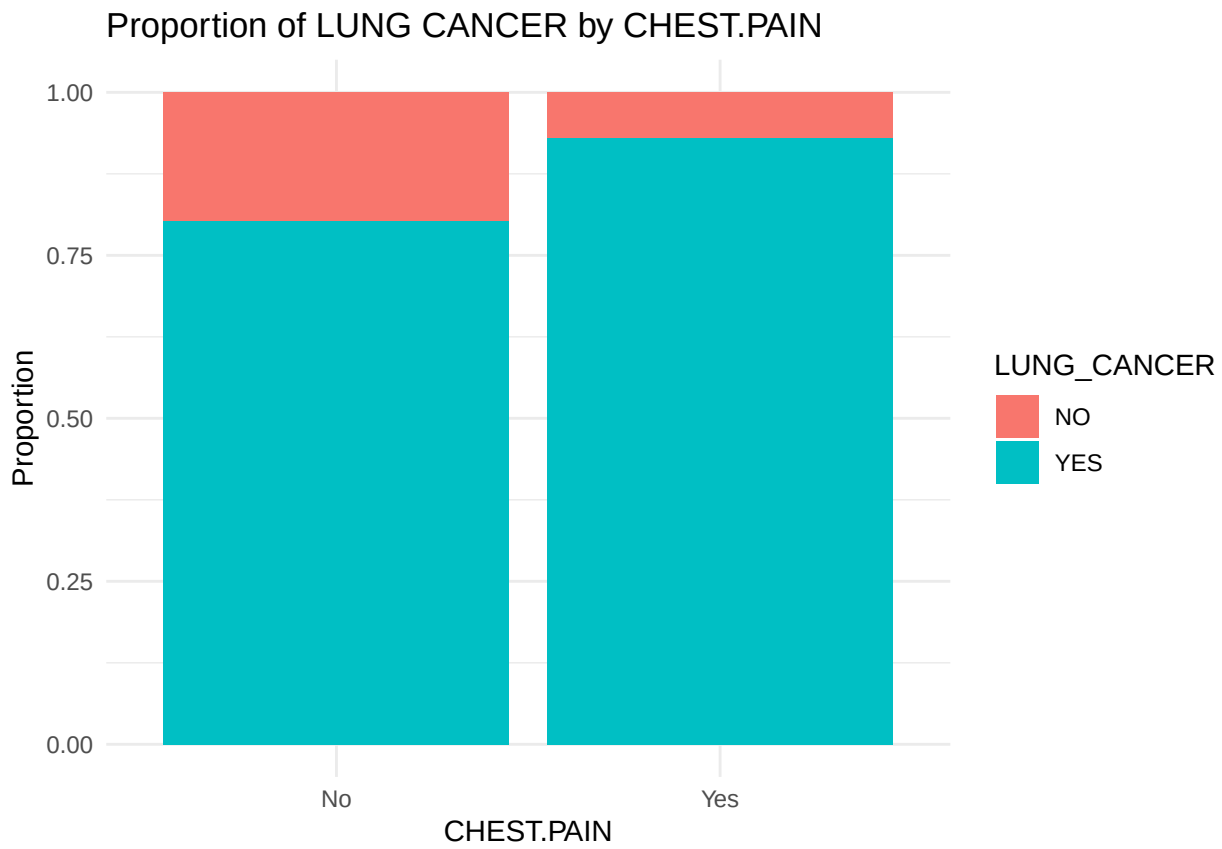

Proportion of LUNG CANCER by WHEEZING
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
WHEEZING
LUNG_CANCER
NO
YES


Proportion of LUNG CANCER by ALCOHOL.CONSUMING
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
ALCOHOL.CONSUMING
LUNG_CANCER
NO
YES


Proportion of LUNG CANCER by COUGHING
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
COUGHING
LUNG_CANCER
NO
YES


Proportion of LUNG CANCER by SHORTNESS.OF.BREATH
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
SHORTNESS.OF.BREATH
LUNG_CANCER
NO
YES


Proportion of LUNG CANCER by SWALLOWING.DIFFICULTY
1.00
0.75
0.50
0.25
0.00
Proportion
No
Yes
SWALLOWING.DIFFICULTY
LUNG_CANCER
NO
YES




```
# Train Logistic Regression Model

logit_model <- train(
  LUNG_CANCER ~ ., data = trainData,
  method = "glm",  # 'glm' method for Generalized Linear Model (logistic regression)
  family = binomial(),  # Ensure logistic regression
  trControl = trainControl(method = "cv", number = 5)
)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Print Logistic Regression Model Summary
print(logit_model)
```

```
## Generalized Linear Model
##
## 217 samples
##  15 predictor
##   2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 174, 173, 174, 174, 173
## Resampling results:
##
```

```
##   Accuracy   Kappa
##   0.9031712  0.5341278
```

```r
# Make predictions
predictions_logistic <- predict(logit_model, testData)

# Confusion Matrix
confusionMatrix(predictions_logistic, testData$LUNG_CANCER)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO YES
##        NO   4   2
##        YES  7  79
##
##                Accuracy : 0.9022
##                  95% CI : (0.8224, 0.9543)
##     No Information Rate : 0.8804
##     P-Value [Acc > NIR] : 0.3266
##
##                   Kappa : 0.4218
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.36364
##             Specificity : 0.97531
##          Pos Pred Value : 0.66667
##          Neg Pred Value : 0.91860
##              Prevalence : 0.11957
##          Detection Rate : 0.04348
##    Detection Prevalence : 0.06522
##       Balanced Accuracy : 0.66947
##
##        'Positive' Class : NO
##
```

ROC Curve:

True Positive Rate (TPR) or Sensitivity:

TPR

True Positives/True Positives + False Positive

Measures the proportion of actual positives correctly identified.

False Positive Rate (FPR):

FPR

False Positives /False Positives + True Negatives

Measures the proportion of actual negatives incorrectly identified as positive.

Thresholds:

The model assigns probabilities to each instance, and the ROC curve is generated by varying the decision threshold for classifying an observation as "positive" or "negative."

```
# Get predicted probabilities
probabilities <- predict(logit_model, testData, type = "prob")[,2]

# Compute ROC curve
roc_curve <- roc(testData$LUNG_CANCER, probabilities)
```

```
## Setting levels: control = NO, case = YES
```

```
## Setting direction: controls < cases
```

```
# Calculate and print AUC in the console
auc_value <- auc(roc_curve)
cat("AUC:", auc_value, "\n")
```

```
## AUC: 0.8832772
```

The X-axis represents FDR or how many negative samples were incorrectly classified.

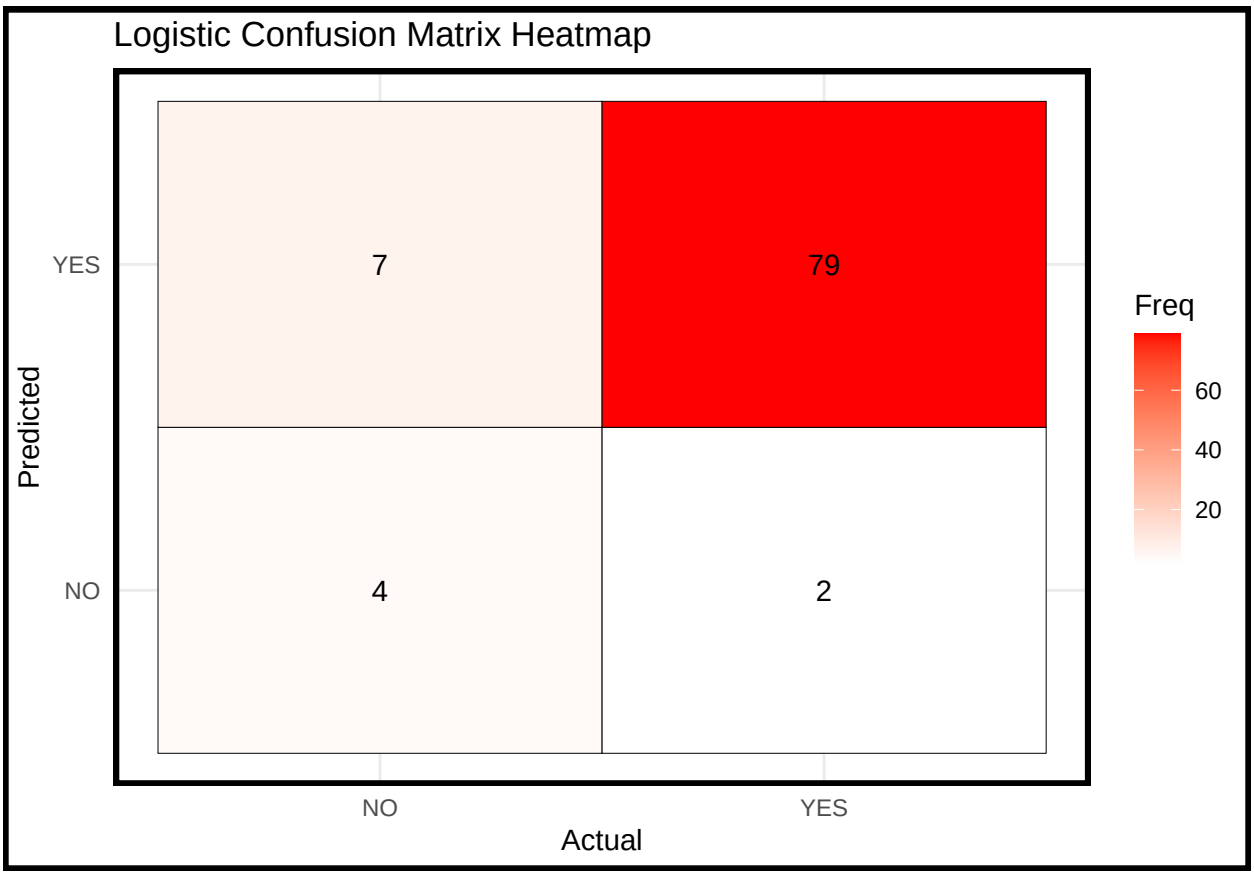
The y-axis represents TPR or how many positive samples were correctly classified.

Top left corner is the ideal performance, where TPR=1 and FPR=0

A model whose ROC curve is close to the diagonal line has little discriminatory power.

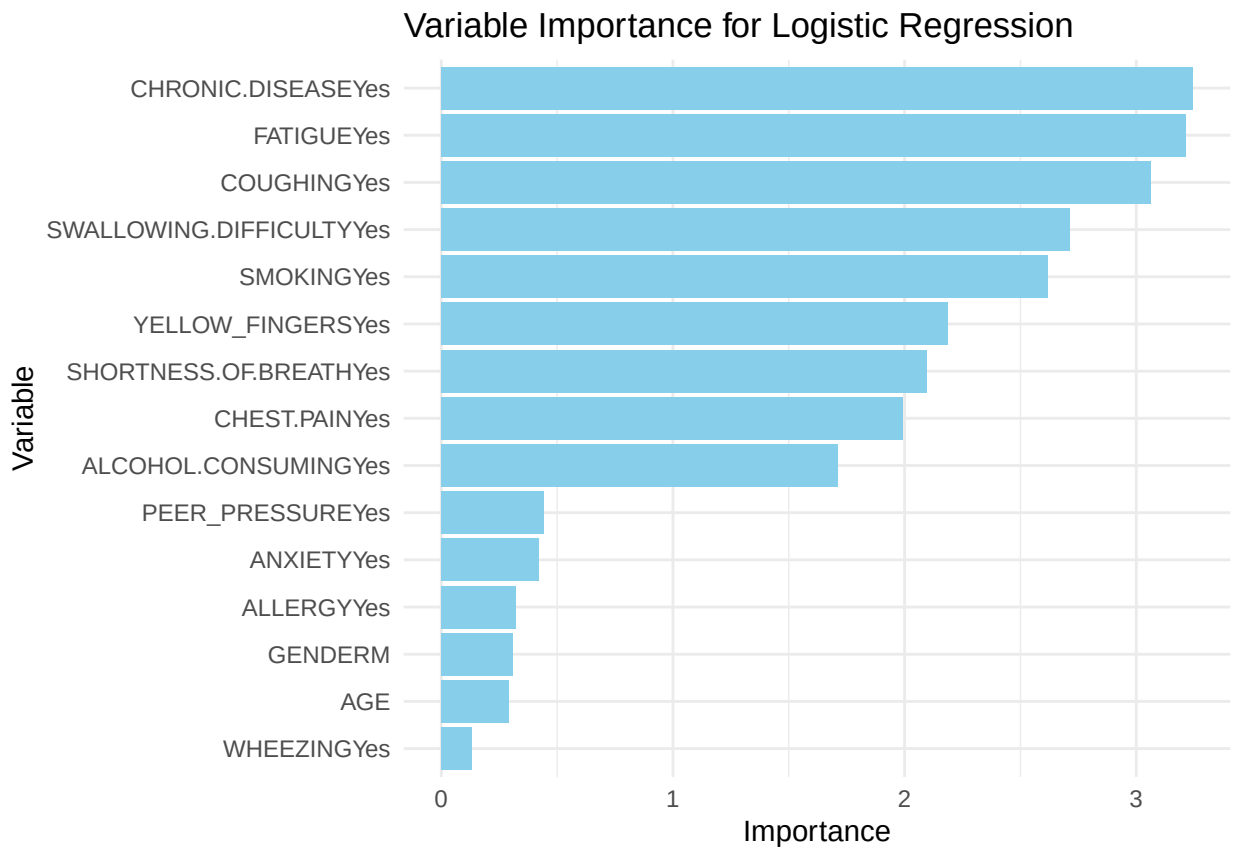
```
# Create confusion matrix
conf_matrix <- confusionMatrix(predictions_logistic, testData$LUNG_CANCER)

# Convert to table
conf_mat_table <- as.data.frame(conf_matrix$table)


 ggplot(conf_mat_table, aes(Reference, Prediction, fill = Freq)) +
  geom_tile(color = "black") +  # Add black border to each tile
  scale_fill_gradient(low = "white", high = "red") +
  geom_text(aes(label = Freq), color = "black") +
  labs(title = "Logistic Confusion Matrix Heatmap", x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(
    plot.background = element_rect(fill = "white", color = "black", size = 2),  # Black outline around
    panel.border = element_rect(color = "black", fill = NA, size = 2)  # Black outline around the panel
  )
```

```
## Warning: The ‘size‘ argument of ‘element_rect()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```




```
var_imp <- varImp(logit_model, scale = FALSE)

# Create the plot with a blue line
 ggplot(var_imp, aes(x = reorder(Variable, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "skyblue") +  # Blue bars
  coord_flip() +  # Flip coordinates for better readability
  labs(title = "Variable Importance for Logistic Regression", x = "Variable", y = "Importance") +
  theme_minimal()
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```




```
# Define the custom grid with only the 'mtry' parameter
# Generates a sequence of numbers starting at 2  going up to the "# of columns" in the data set increme

tune_grid <- expand.grid(mtry = seq(2, ncol(trainData) - 1, by = 1))

set.seed(123)

# Train the Random Forest model
rf_model <- train(
  LUNG_CANCER ~ ., data = trainData,
  method = "rf",
  trControl = trainControl(method = "cv", number = 5),
  tuneGrid = tune_grid
)

# Model summary
print(rf_model)
```

```
## Random Forest
##
## 217 samples
##  15 predictor
##   2 classes: 'NO', 'YES'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 173, 174, 174, 174, 173
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9124736  0.5156750
##    3    0.9124736  0.5365124
##    4    0.9078224  0.5353562
##    5    0.9078224  0.5335466
##    6    0.9170190  0.5935555
##    7    0.9123679  0.5659729
##    8    0.9124736  0.5611292
##    9    0.9124736  0.5611292
##   10    0.9079281  0.5486576
##   11    0.9079281  0.5486576
##   12    0.9032770  0.5342924
##   13    0.8987315  0.4978752
##   14    0.8986258  0.5008994
##   15    0.9032770  0.5152646
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

```r
# Make predictions
predictions_rf <- predict(rf_model, testData)

# Confusion Matrix
confusionMatrix(predictions_rf, testData$LUNG_CANCER)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO YES
##        NO   1   1
##        YES 10  80
##
##                Accuracy : 0.8804
##                  95% CI : (0.7961, 0.9388)
##     No Information Rate : 0.8804
##     P-Value [Acc > NIR] : 0.57943
##
##                   Kappa : 0.1215
##
##  Mcnemar's Test P-Value : 0.01586
##
##             Sensitivity : 0.09091
##             Specificity : 0.98765
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.88889
##              Prevalence : 0.11957
##          Detection Rate : 0.01087
##    Detection Prevalence : 0.02174
##       Balanced Accuracy : 0.53928
##
```

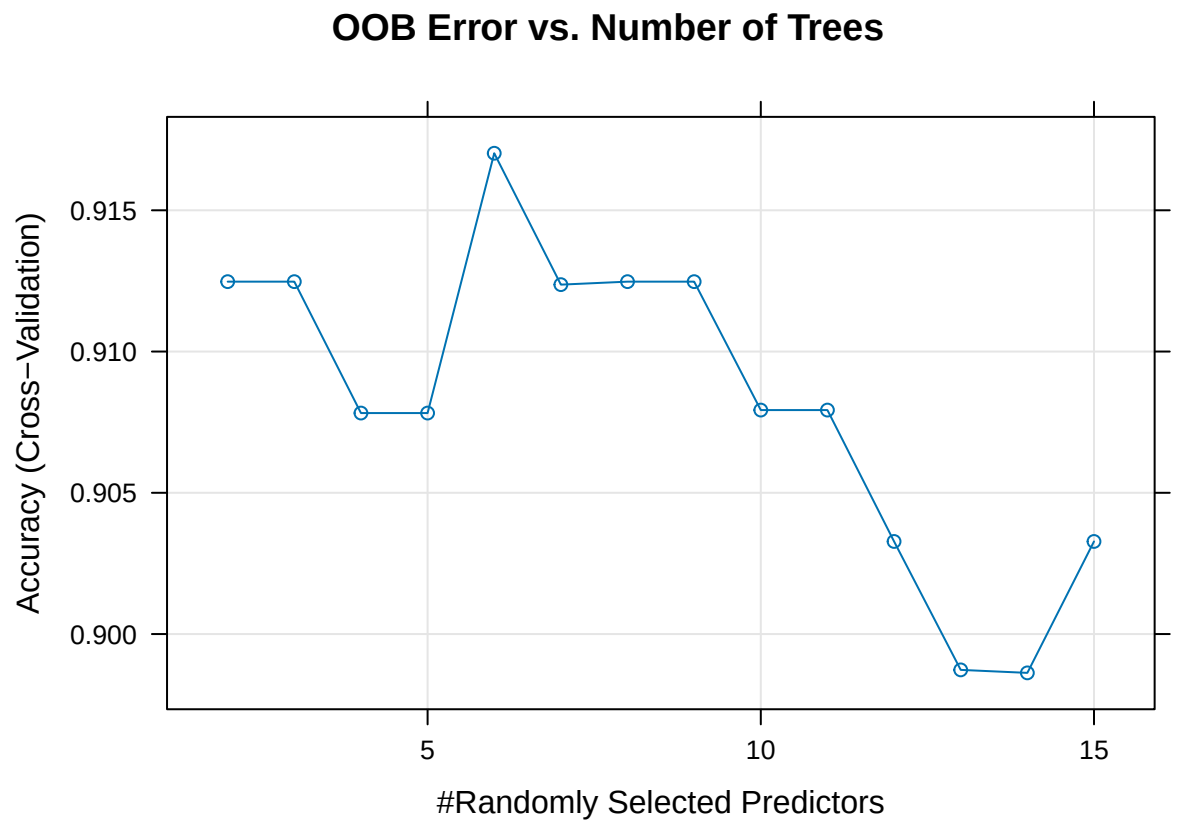
```
##         'Positive' Class : NO
##
```

The Out-of-Bag (OOB) error is a performance metric used with Random Forest models. It provides an estimate of the model's prediction error without needing a separate test data set. When building the decision tree the training data is sampled with replacement which creates a "bootstrapped sample". The sample that is not included in the bootstrapped sample is considered the "out-of-bag" sample. The OOB error is the proportion of misclassified observations in the OOB data, averaged across all trees in the Random Forest. OOB error approximates how well the model will perform on unseen data.

```
# Extract and print OOB error
oob_error <- rf_model$err.rate[500, "OOB"]
cat("OOB Error Rate:", oob_error, "\n")
```

```
## OOB Error Rate:
```

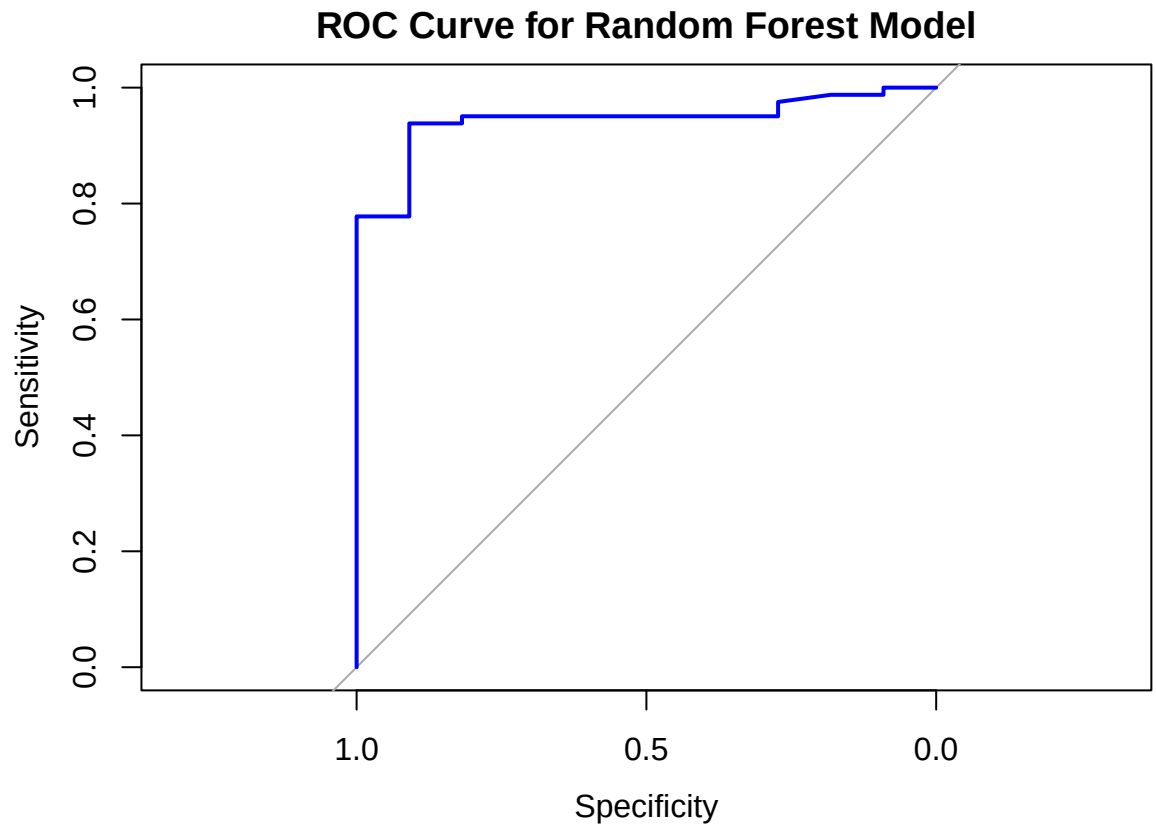
```
# Visualize OOB error over trees
plot(rf_model, main = "OOB Error vs. Number of Trees")
```




```
# Get predicted probabilities
probabilities_rf <- predict(rf_model, testData, type = "prob")[,2]

# Compute ROC curve
roc_rf <- roc(testData$LUNG_CANCER, probabilities_rf)
```

```
## Setting levels: control = NO, case = YES

## Setting direction: controls < cases
```

```
# Plot the ROC curve
 plot(roc_rf, main = "ROC Curve for Random Forest Model", col = "blue", lwd = 2)
```

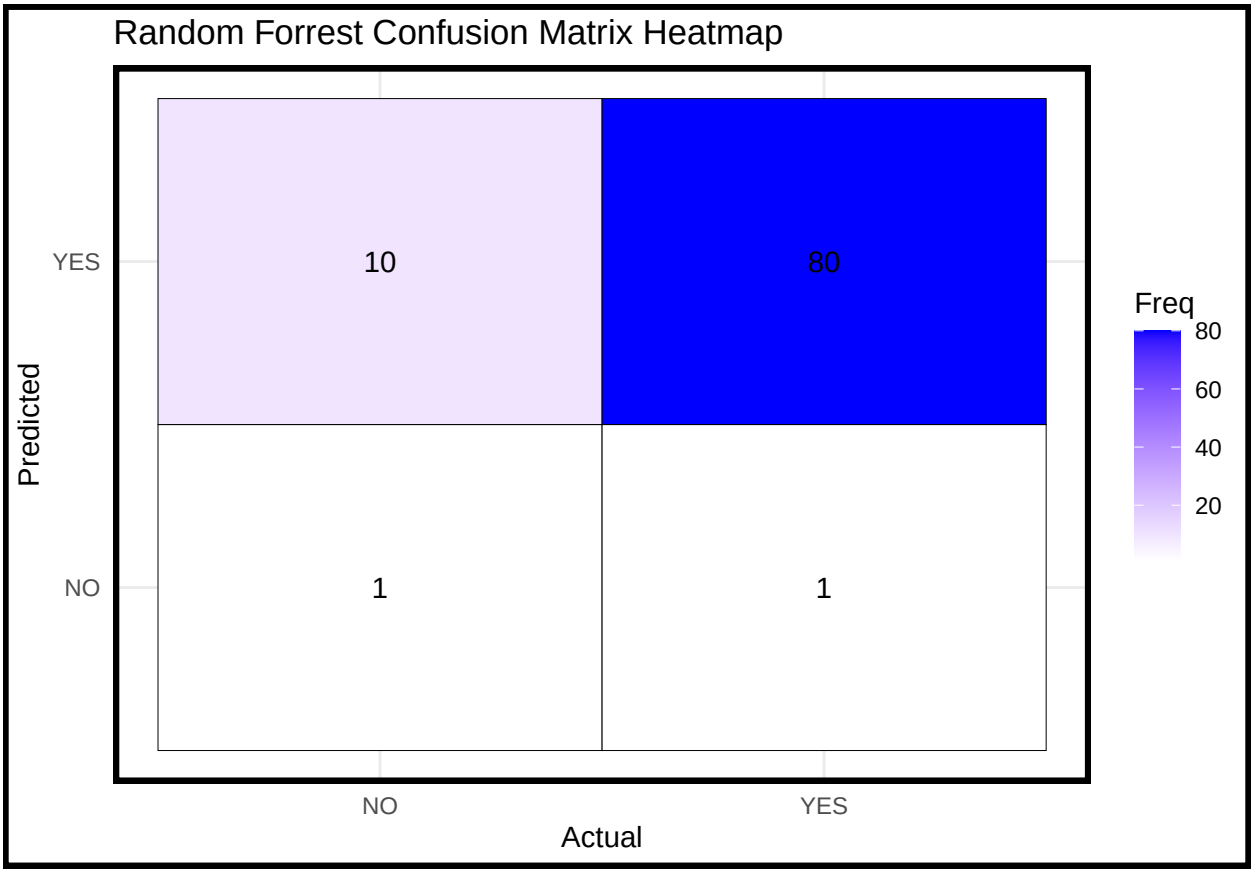



```
auc_rf <- auc(roc_rf)
cat("AUC:", auc_rf, "\n")
```

```
## AUC: 0.9444444
```

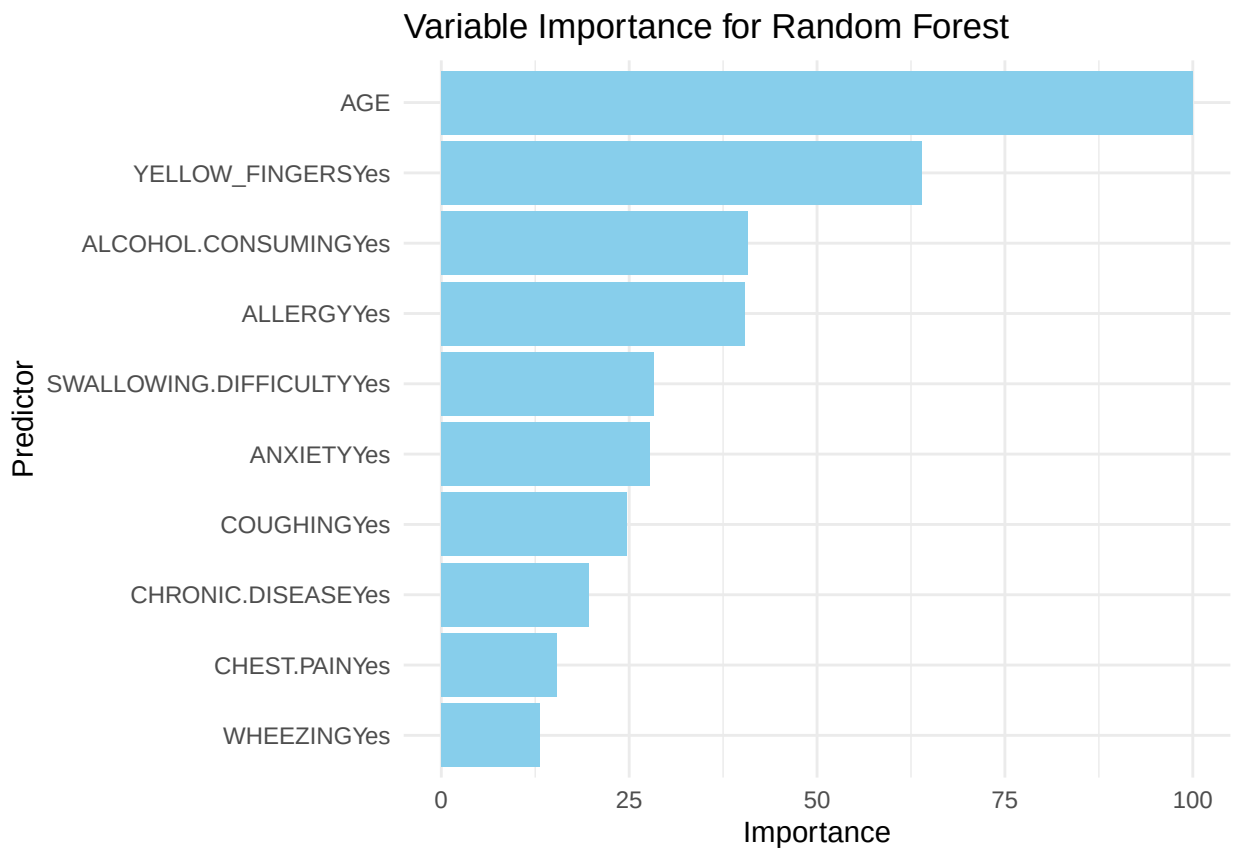
```
# Make predictions
predictions_rf <- predict(rf_model, testData)

# Confusion Matrix
conf_matrix_rf <- confusionMatrix(predictions_rf, testData$LUNG_CANCER)

# Convert to a table
conf_mat_table_rf <- as.data.frame(conf_matrix_rf$table)

# Plot heatmap
 ggplot(conf_mat_table_rf, aes(Reference, Prediction, fill = Freq)) +
  geom_tile(color = "black") +  # Add black border to each tile
```

```
  scale_fill_gradient(low = "white", high = "blue") +
  geom_text(aes(label = Freq), color = "black") +
  labs(title = "Random Forrest Confusion Matrix Heatmap", x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(
    plot.background = element_rect(fill = "white", color = "black", size = 2),  # Black outline around
    panel.border = element_rect(color = "black", fill = NA, size = 2)  # Black outline around the panel
  )
```




```
# Extract variable importance
var_imp_rf <- varImp(rf_model, scale = TRUE)


# Convert to data frame for ggplot
var_imp_df <- var_imp_rf$importance
var_imp_df$Variable <- rownames(var_imp_df)
var_imp_df <- var_imp_df[order(var_imp_df$Overall, decreasing = TRUE), ]

# Limit to top N variables (e.g., top 10)
top_n <- 10
var_imp_df_top <- head(var_imp_df, top_n)

# Plot with ggplot2
 ggplot(var_imp_df_top, aes(x = reorder(Variable, Overall), y = Overall)) +
  geom_bar(stat = "identity", fill = "skyblue") +
```

```
  coord_flip() +
  labs(title = "Variable Importance for Random Forest",
       x = "Predictor",
       y = "Importance") +
  theme_minimal() +
  theme(axis.text.y = element_text(angle = 0, hjust = 1))  # Make text horizontal for readability
```




```
# Train a KNN model
set.seed(123)  # Set seed for reproducibility
knn_model <- train(
  LUNG_CANCER ~ ., data = trainData,
  method = "knn",  # Specify KNN method
  trControl = trainControl(method = "cv", number = 5),  # 5-fold cross-validation
  tuneGrid = expand.grid(k = 3:10)  # Tune the number of neighbors (k)
)

# Model summary
print(knn_model)
```

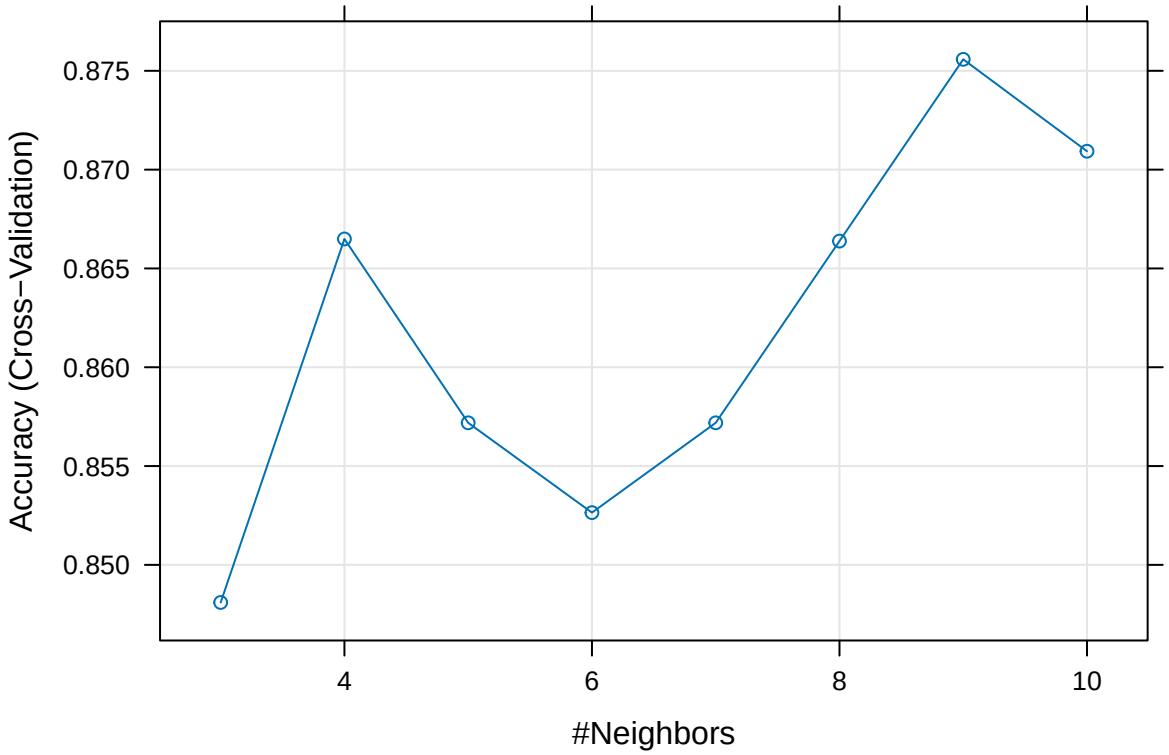
```
## k-Nearest Neighbors
##
## 217 samples
##  15 predictor
##   2 classes: 'NO', 'YES'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 173, 174, 174, 174, 173
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    3  0.8480973  0.13015830
##    4  0.8664905  0.24112517
##    5  0.8571882  0.10666423
##    6  0.8526427  0.02359166
##    7  0.8571882  0.06966723
##    8  0.8663848  0.03095987
##    9  0.8755814  0.05135135
##   10  0.8709302  0.04328684
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```r
# Make predictions on the test data
predictions_knn <- predict(knn_model, testData)

# Confusion Matrix
confusionMatrix(predictions_knn, testData$LUNG_CANCER)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO YES
##        NO   0   0
##        YES 11  81
##
##                Accuracy : 0.8804
##                  95% CI : (0.7961, 0.9388)
##     No Information Rate : 0.8804
##     P-Value [Acc > NIR] : 0.579428
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.002569
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.8804
##              Prevalence : 0.1196
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : NO
##
```

```r
# Plot KNN performance for different k values
plot(knn_model)
```

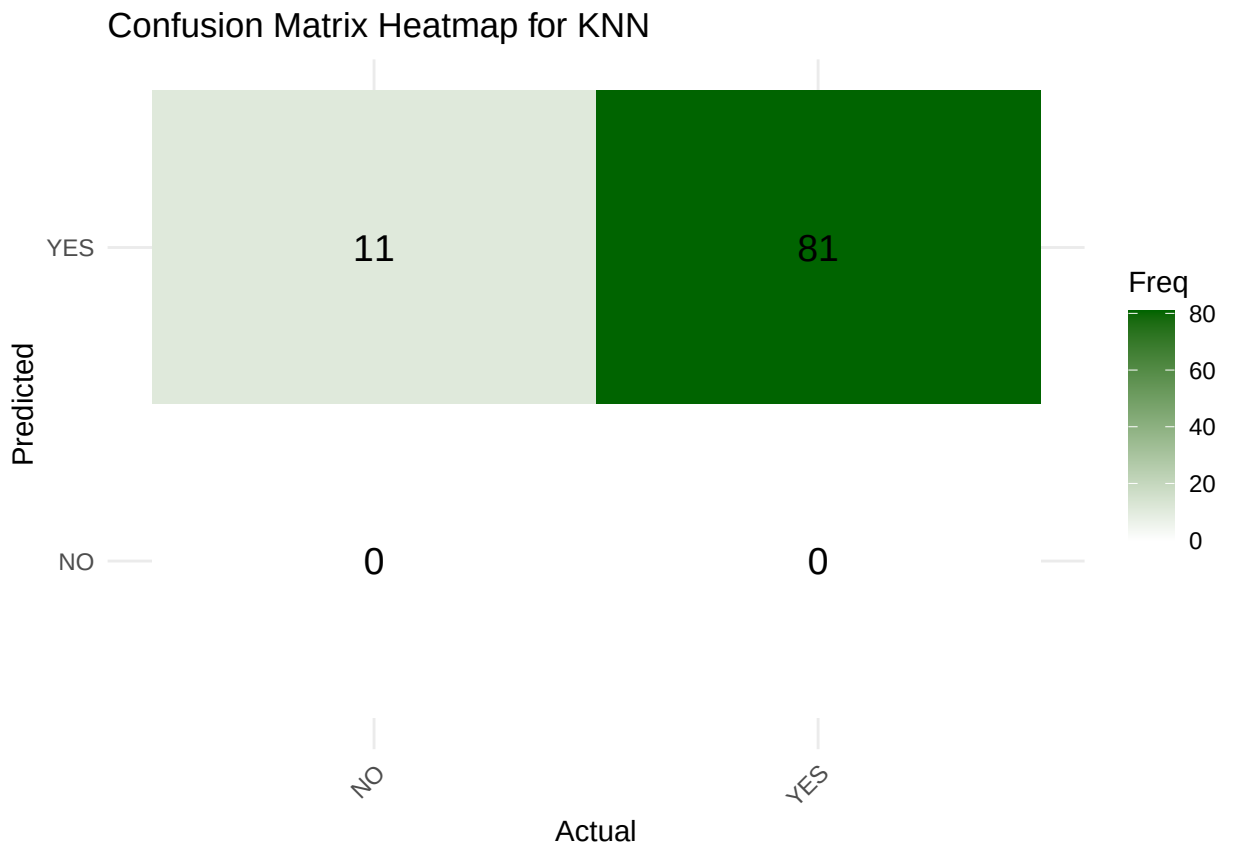


According to the accuracy plot k = 3 and k = 5 have the best accuracy. Therefore, k = 5 was used for the model.

```r
# Make predictions on the test data
predictions_knn <- predict(knn_model, testData)

# Create confusion matrix
conf_matrix <- confusionMatrix(predictions_knn, testData$LUNG_CANCER)

# Convert to table
conf_mat_table_KNN <- as.data.frame(conf_matrix$table)


KNN_confusion_matrix <- ggplot(conf_mat_table_KNN , aes(Reference, Prediction, fill = Freq)) +
  geom_tile(color = "black") +  # Add black border to each tile
  scale_fill_gradient(low = "white", high = "darkgreen") +
  geom_text(aes(label = Freq), color = "black") +
  labs(title = "KNN Confusion Matrix Heatmap", x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(
    plot.background = element_rect(fill = "white", color = "black", size = 2),  # Black outline around
    panel.border = element_rect(color = "black", fill = NA, size = 2)  # Black outline around the panel
  )
```

The results from the ROC curve suggest a strong model performance. Normally models closer to .5 are considered to have no discriminatory ability and models closer to 1 are considered to have high discriminatory ability.

```
# Generate predictions from KNN model (assuming knn_model and testData are defined)
predictions_knn <- predict(knn_model, testData)

# Generate confusion matrix
cm <- confusionMatrix(predictions_knn, testData$LUNG_CANCER)

# Convert confusion matrix into a data frame for plotting
cm_data <- as.data.frame(cm$table)
colnames(cm_data) <- c("Predicted", "Actual", "Freq")

# Create a heatmap of the confusion matrix
ggplot(cm_data, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "black", size = 5) +
  scale_fill_gradient(low = "white", high = "darkgreen") +
  labs(title = "Confusion Matrix Heatmap for KNN", x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Train a Linear Discriminant Analysis (LDA) model
lda_model <- train(
```

```
  LUNG_CANCER ~ ., data = trainData,
  method = "lda",
  trControl = trainControl(method = "cv", number = 5)
)

# Print the LDA model summary
print(lda_model)
```

```
## Linear Discriminant Analysis
##
## 217 samples
##  15 predictor
##   2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 175, 173, 173, 173, 174
## Resampling results:
##
##   Accuracy   Kappa
##   0.8983842  0.5546323
```

```
# Get predicted probabilities
probabilities_lda <- predict(lda_model, testData, type = "prob")[,2]

# Compute ROC curve
roc_lda <- roc(testData$LUNG_CANCER, probabilities_lda)
```

```
## Setting levels: control = NO, case = YES
```

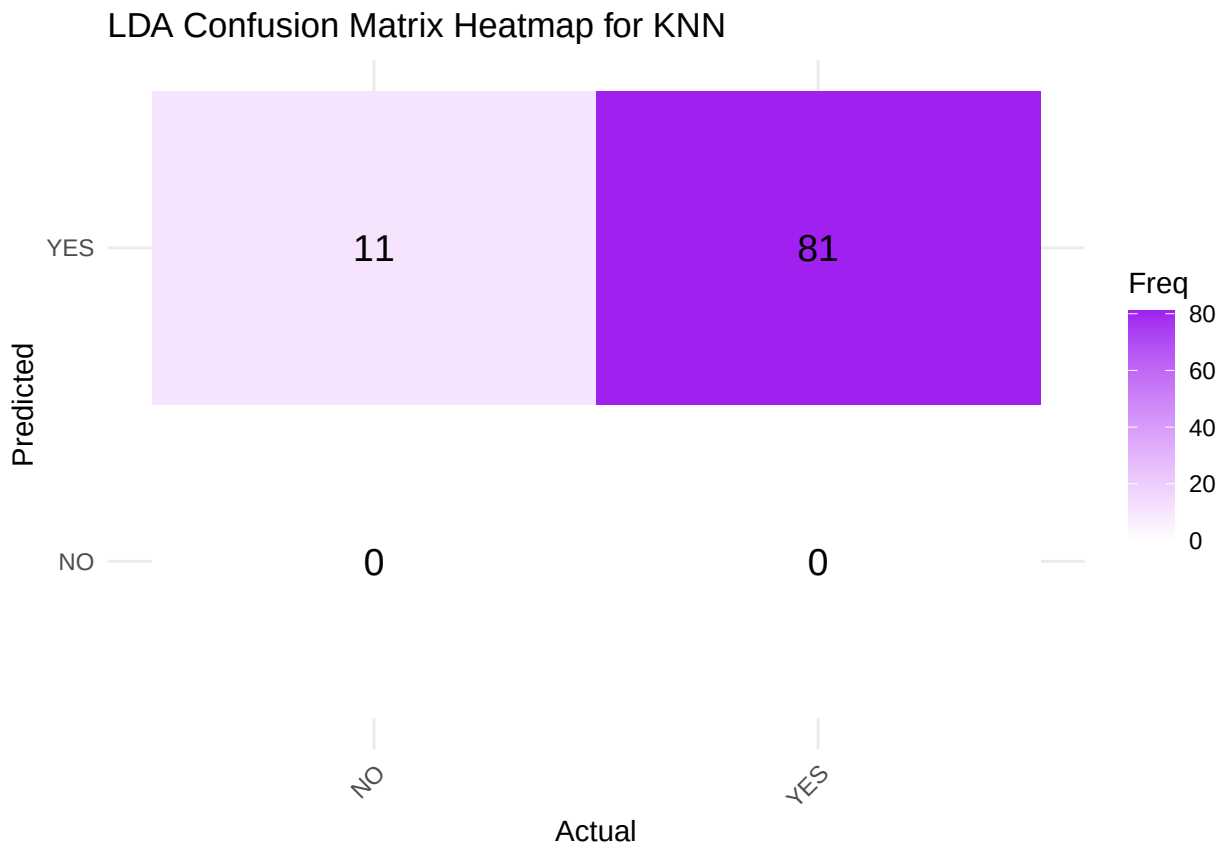
```
## Setting direction: controls < cases
```

```
# Make predictions using the LDA model
predictions_lda <- predict(lda_model, testData)

# Confusion Matrix for LDA
cm_lda <- confusionMatrix(predictions_lda, testData$LUNG_CANCER)
print(cm_lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO YES
##        NO   7   4
##        YES  4  77
##
##                Accuracy : 0.913
##                  95% CI : (0.8358, 0.9617)
##     No Information Rate : 0.8804
##     P-Value [Acc > NIR] : 0.2151
##
```

```
##                  Kappa : 0.587
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.63636
##             Specificity : 0.95062
##          Pos Pred Value : 0.63636
##          Neg Pred Value : 0.95062
##              Prevalence : 0.11957
##          Detection Rate : 0.07609
##    Detection Prevalence : 0.11957
##       Balanced Accuracy : 0.79349
##
##        'Positive' Class : NO
##
```

```r
# Convert confusion matrix into a data frame for plotting
cm_lda <- as.data.frame(cm$table)
colnames(cm_lda) <- c("Predicted", "Actual", "Freq")

# Create a heatmap of the confusion matrix
 ggplot(cm_lda, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "black", size = 5) +
  scale_fill_gradient(low = "white", high = "purple") +
  labs(title = "LDA Confusion Matrix Heatmap for KNN", x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


LDA Confusion Matrix Heatmap for KNN
YES
NO
11
81
0
0
Predicted
Actual
NO
YES
Freq
80
60
40
20
0