23070164

Question 1

(a) Research Question

This study investigates how socio-economic and demographic factors impact house prices (MedianHousePrice) in London boroughs. By analyzing these relationships, we can provide valuable insights for policymakers, investors, and urban planners to address housing affordability and optimize urban development.

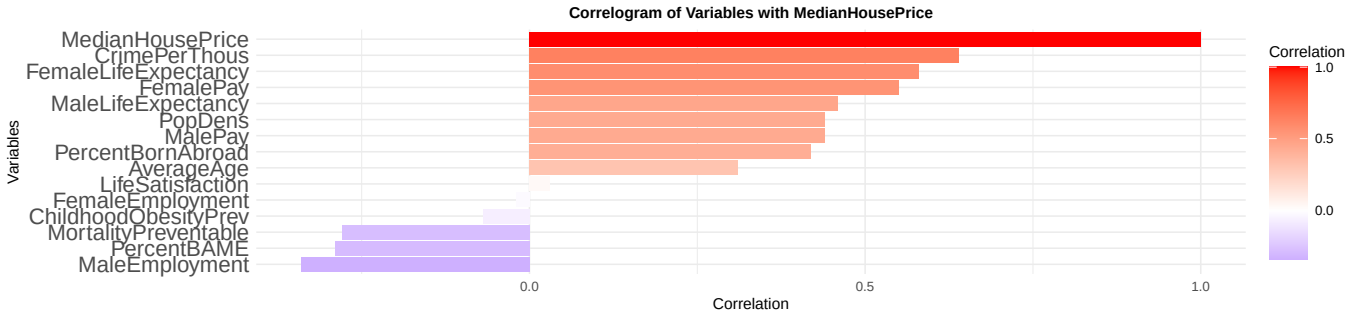
(b) Exploratory Analysis of the Data

Introducing the Dataset (Dataset Overview) and Checking & Handling Missing Data

The dataset contains 33 entries (London boroughs) and 16 variables. The primary variable of interest is MedianHousePrice, which represents the median property price in each borough. Several variables have missing data: MalePay is The variable with the highest missing values (7 out of 33, or 21.2%). Then Variables like FemalePay (9.09%) and others with minimal missing data (<5%, e.g., CrimePerThous, PercentBornAbroad). All missing values across the dataset were addressed using mean imputation. The following are the variable descriptions used in this research: **PopDens** (Population density, 2016), **AverageAge** (Average age, 2016), **PercentBornAbroad** (% born abroad, 2014), **PercentBAME** (% BAME population, 2013), **MaleEmployment** (Male employment rate, 2015), **FemaleEmployment** (Female employment rate, 2015), **MalePay** (Male annual pay, 2015), **FemalePay** (Female annual pay, 2015), **CrimePerThous** (Crime rate per 1,000, 2014/15), **MedianHousePrice** (Median house price, 2014/15), **MaleLifeExpectancy** (Male life expectancy, 2012-14), **FemaleLifeExpectancy** (Female life expectancy, 2012-14), **LifeSatisfaction** (Life satisfaction score, 2011-14), **ChildhoodObesityPrev** (Childhood obesity %, 2013/14), **MortalityPreventable** (Preventable mortality rate, 2012-14).

Correlation Analysis and Summary Statistics

To highlight and identify which variables have the strongest relationship with **MedianHousePrice**, we calculate the correlation and create a correlogram. This visualization makes it easy to identify key predictors at a glance.




```
Summary Statistics for MedianHousePrice:

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 215000  307000  385000  429029  433000 1195000
```

Summary and Interpretation of Results

House prices in London boroughs range from £215,000 to £1,195,000, with a mean of £429,029, showing wide disparities. CrimePerThous ($r = 0.64$), FemalePay, and FemaleLifeExpectancy strongly correlate with prices, reflecting the influence of urbanization, income, and living conditions. Moderate correlations with MalePay ($r = 0.44$), MaleLifeExpectancy ($r = 0.46$), and population density ($r = 0.44$) suggest smaller but still relevant effects. Urban areas with higher density and better living standards tend to have higher prices, making these factors key in regression modeling.

(c) Assumption Checks and Linear Model Fitting

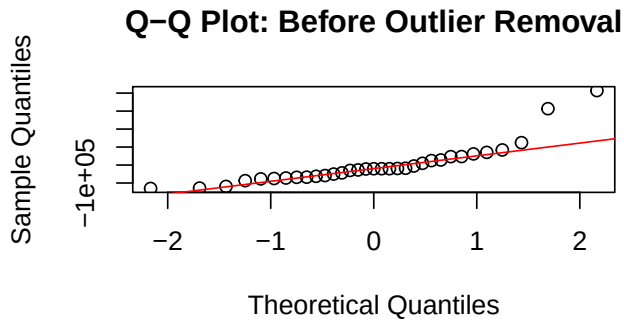
Before scaling and fitting the linear model, it is important to check the assumption of variable correlation using the raw data to ensure the analysis reflects the original dataset. The procedures as follows:

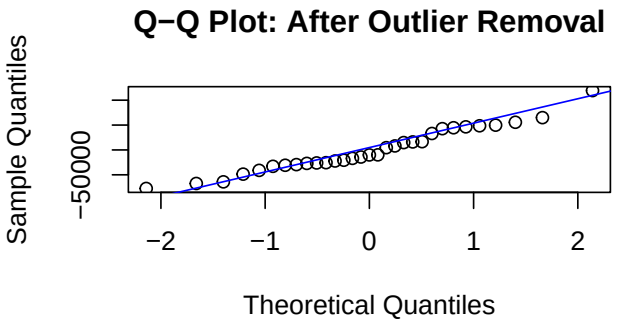
1. Normality of Residuals (Assumption Checks)

Initial tests (Q-Q Plot, Shapiro-Wilk p-value: 1.28e-05) revealed non-normal residuals caused by outliers in rows 1 and 20. Removing these reduced the dataset to 31 observations. The Shapiro-Wilk p-value improved to 0.3455, indicating normal residuals and meeting the regression model's normality assumption.

```
Shapiro-Wilk Test p-value (Before Outlier Removal): 1.279607e-05
```

```
Shapiro-Wilk Test p-value (After Outlier Removal): 0.3454737
```





2. Linearity, Homoscedasticity and Multicollinearity (Assumption Checks)

```
                    Test                     Result Satisfied
1              Linearity              Correlation = 0       Yes
2 Homoscedasticity Breusch-Pagan p-value = 0.5425       Yes
3 Multicollinearity (VIF)               Max VIF = 9.67       Yes
```

Interpretation of Linearity, Homoscedasticity and Multicollinearity

- Linearity: Relationship between predictors and response is approximately linear.
- Homoscedasticity: Breusch-Pagan test confirms constant variance ($p = 0.41$).
- Multicollinearity:
  - High VIF for `MaleLifeExpectancy` (9.67) and `FemaleLifeExpectancy` (6.38).
  - These variables may require removal or regularization (e.g., Lasso).

Conclusion: All assumptions are satisfied. No transformations are required.

3. Scaling Data and Fit Linear Model

```r
# Fit the full linear model using all predictors
linear_model_scaled <- lm(MedianHousePrice ~ CrimePerThous + FemalePay + FemaleLifeExpectancy +
                            MalePay + MaleLifeExpectancy + PopDens, data = scaled_housing_data)
# Display the summary of the full model
summary(linear_model_scaled)
```

```
Call:
lm(formula = MedianHousePrice ~ CrimePerThous + FemalePay + FemaleLifeExpectancy +
    MalePay + MaleLifeExpectancy + PopDens, data = scaled_housing_data)

Residuals:
   Min     1Q Median     3Q    Max
-77445 -28288 -10049  37882 119282

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           393482       8948  43.972  < 2e-16 ***
CrimePerThous          52459      13456   3.899  0.00068 ***
FemalePay              59970      19563   3.065  0.00531 **
FemaleLifeExpectancy   31676      22971   1.379  0.18063
MalePay                -5510      19549  -0.282  0.78048
MaleLifeExpectancy     26385      28281   0.933  0.36013
PopDens                36976      17324   2.134  0.04322 *
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 49820 on 24 degrees of freedom
Multiple R-squared:  0.904, Adjusted R-squared:   0.88
F-statistic: 37.67 on 6 and 24 DF,  p-value: 4.633e-11
```

Interpretation of Linear Model

The model explains 90.4% of the variability in MedianHousePrice, with an adjusted R^2 of 88%, indicating a strong fit. Significant predictors—CrimePerThous, FemalePay, and PopDens—suggest urban demand drives higher prices. Weaker or non-significant effects are seen for MalePay and life expectancy. Residual diagnostics confirm normality and homoscedasticity, highlighting socioeconomic and urban factors as key drivers of house prices across London boroughs.

(d) Perform Stepwise Regression

```
--- Top Two Stepwise Models ---

   Method  AdjR2    AIC    BIC                                                Predictors
 Both_Dir 0.8852 763.75 772.36 CrimePerThous FemalePay FemaleLifeExpectancy PopDens
 Backward 0.8852 763.75 772.36 CrimePerThous FemalePay FemaleLifeExpectancy PopDens
```

Interpretation

Stepwise regression found that both directions and backward elimination selected CrimePerThous, FemalePay, FemaleLifeExpectancy, and PopDens, while forward selection included weaker predictors like MalePay and MaleLifeExpectancy. Both directions and backward elimination achieved an adjusted R^2 of 0.8852 with lower AIC (763.75) and BIC (772.36) than forward selection. These models are preferred for their simplicity and superior performance.

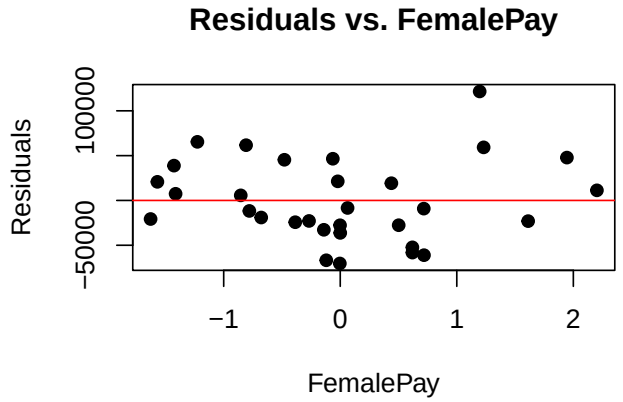
(e) Stepwise Model Suggestion and Model Refinement

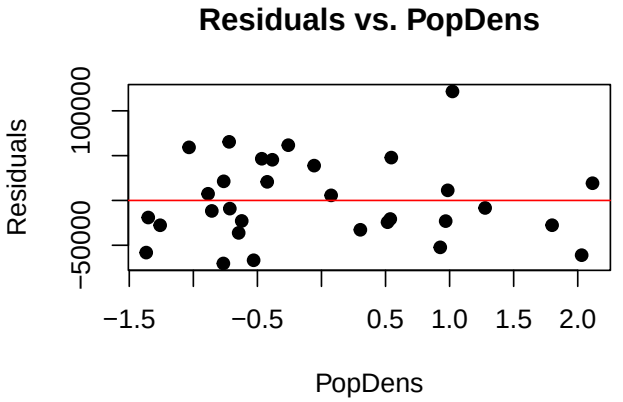
```
Call:
lm(formula = MedianHousePrice ~ CrimePerThous + FemalePay + FemaleLifeExpectancy +
    PopDens, data = scaled_housing_data)

Residuals:
   Min     1Q Median     3Q    Max
-70306 -27653  -9113  30084 121596

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           393482       8752  44.959  < 2e-16 ***
CrimePerThous          51608      13123   3.932 0.000557 ***
FemalePay              59433      11785   5.043    3e-05 ***
FemaleLifeExpectancy   50222      11112   4.520 0.000119 ***
PopDens                27769      13358   2.079 0.047631 *
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 48730 on 26 degrees of freedom
Multiple R-squared:  0.9005,    Adjusted R-squared:  0.8852
F-statistic: 58.83 on 4 and 26 DF,  p-value: 1.189e-12
```






```r
# --- Step 3: Define and Fit Model Refinement --- #
# Apply polynomial transformation to variables that show non-linear patterns
stepwise_refined <- lm(MedianHousePrice ~ CrimePerThous + poly(FemalePay, 2) +
                       FemaleLifeExpectancy +PopDens,data = scaled_housing_data)
```

```
Call:
lm(formula = MedianHousePrice ~ CrimePerThous + poly(FemalePay,
    2) + FemaleLifeExpectancy + PopDens, data = scaled_housing_data)

Residuals:
   Min     1Q Median     3Q    Max
-58026 -20652  -8576  25797 121250

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           393482       8162  48.212  < 2e-16 ***
CrimePerThous          37819      13732   2.754   0.0108 *
poly(FemalePay, 2)1   356928      61844   5.771 5.15e-06 ***
poly(FemalePay, 2)2   114718      51832   2.213   0.0362 *
FemaleLifeExpectancy   48895      10380   4.711 7.89e-05 ***
PopDens                32649      12650   2.581   0.0161 *
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 45440 on 25 degrees of freedom
Multiple R-squared:  0.9168,    Adjusted R-squared:  0.9002
F-statistic:  55.1 on 5 and 25 DF,  p-value: 1.077e-12
```

Interpretation of Stepwise Model Suggestion

The base linear model explains 90.05% of the variability in MedianHousePrice, achieving an adjusted R^2 of 0.8852. Key predictors such as CrimePerThous, FemalePay, and FemaleLifeExpectancy are highly significant, while PopDens has a weaker but still significant effect. Residuals deviate by approximately $\pm 48,730$, indicating good predictive capability. The refined model improves performance, explaining 91.68% of the variability with an adjusted } R^2 of 0.9002. The standard error of residuals decreases to $45,440$, with a non-linear effect for FemalePay. As FemalePay increases, its positive impact on housing prices diminishes, reflecting affordability constraints at higher income levels. This suggests that small increases in pay can significantly impact housing prices in lower-income regions, while in wealthier areas, further income increases yield reduced effects.

(f) Lasso Regression Function

```r
# --- Lasso Regression Algorithm ---
# Step 1: Initialize the function to take y (response), X (predictors),
# and lambda_values (penalty grid).
# Step 2: Create an empty matrix to store the coefficients for each lambda.
# Step 3: For each lambda value:
#         a. Define the lasso loss function (RSS + L1 penalty).
#         b. Calculate RSS: Sum of squared residuals between actual y
```

```r
#            and predicted values.
#         c. Add penalty: L1 regularization to encourage sparsity.
#         d. Minimize the loss using the `nlm` optimization function to
#            find the optimal coefficients.
# Step 4: Return a matrix of coefficients for each lambda, labeled by variable
#            names and lambda values.

# --- Define the LassoRegression Function ---
LassoRegression <- function(y, X, lambda_values) {

  # --- Step 1: Initialize variables ---
  n <- nrow(X)               # Number of observations (rows in X)
  p <- ncol(X)               # Number of predictors (columns in X)

  # Matrix to store the coefficients (including intercept) for each lambda value
  coefficients <- matrix(0,
                         nrow = length(lambda_values),
                         ncol = p + 1)  # +1 for the intercept column

  # --- Step 2: Loop through each lambda value ---
  for (i in seq_along(lambda_values)) {
    lambda <- lambda_values[i]  # Select the current lambda

    # --- Step 3: Define the loss function ---
    # This function computes the lasso objective: RSS + L1 penalty
    loss_function <- function(beta) {
      beta_0 <- beta[1]          # Intercept term (beta_0)
      beta_rest <- beta[-1]       # Remaining coefficients

      # --- Step 4: Calculate RSS ---
      rss <- sum((y - (beta_0 + X %*% beta_rest))^2)  # Residual Sum of Squares

      # --- Step 5: Apply L1 penalty ---
      penalty <- lambda * sum(abs(beta_rest))  # Note: Intercept is NOT penalized

      # --- Step 6: Return total loss ---
      return(rss + penalty)
    }

    # --- Step 7: Optimize the loss function ---
    # Use nlm to minimize the loss function, starting from zero-initialized coefficients
    optimized_result <- nlm(f = loss_function,
                            p = rep(0, p + 1),  # Initial guess
                                                #(all coefficients set to zero)
                            iterlim = 500)      # Set a maximum iteration limit

    # --- Step 8: Store optimized coefficients ---
    coefficients[i, ] <- optimized_result$estimate
  }

  # --- Step 9: Label the coefficient matrix ---
  colnames(coefficients) <- c("Intercept", colnames(X))  # Add column names for
                                                         # variables
  rownames(coefficients) <- paste0("Lambda_", lambda_values)  # Label rows with
                                                              #lambda values

  # --- Step 10: Return the final matrix of coefficients ---
  return(coefficients)
}

# --- End of LassoRegression Function ---

#Lets test the function by simulating the data

# --- Step 1: Simulate Data ---
# Generate a matrix X (50 observations, 5 predictors) and a response vector y
set.seed(42)  # Ensure reproducibility
n_obs <- 50    # Number of observations
n_pred <- 5    # Number of predictors

# Generate predictors and response variable
X_sim <- matrix(rnorm(n_obs * n_pred), nrow = n_obs, ncol = n_pred)
colnames(X_sim) <- paste0("X", 1:n_pred)  # Assign predictor names

# Define true coefficients, including some zeros to simulate sparsity
true_beta <- c(3, -2, 0, 1.5, 0)  # Sparse true coefficients
y_sim <- X_sim %*% true_beta + rnorm(n_obs, mean = 0, sd = 0.5)  # Add noise to response

# --- Step 2: Define lambda values ---
# Test different levels of penalization
lambda_test_values <- c(0.1, 1, 10, 100, 1000)

# --- Step 3: Apply Lasso Regression Function ---
```

```r
lasso_results_sim <- LassoRegression(y = y_sim, X = X_sim, lambda_values = lambda_test_values)

# --- Step 4: Display and Verify Results ---
cat("\nLasso Regression Coefficients for Simulated Data:\n")
```

```
Lasso Regression Coefficients for Simulated Data:
```

```r
print(round(lasso_results_sim, 4))
```

```
           Intercept     X1      X2     X3     X4    X5
Lambda_0.1    -0.0044 3.0497 -2.0790 0.0146 1.4487 0.093
Lambda_1      -0.0069 3.0452 -2.0701 0.0058 1.4389 0.085
Lambda_10     -0.0145 2.9911 -1.9834 0.0001 1.3514 0.000
Lambda_100    -0.1477 2.5610 -1.1106 0.0000 0.3071 0.000
Lambda_1000    0.0000 0.0000  0.0000 0.0000 0.0000 0.000
```

- The Lasso regression function successfully shrinks all coefficients to zero for sufficiently large λ values, confirming correct implementation of the penalty mechanism. This demonstrates that the function effectively applies L1 regularization and performs variable selection as expected.

(g) Applying Lasso Regression

```
--- Optimal Lambda from Cross-Validation ---

[1] 464.1589

--- Final Summary of Coefficients ---

             Predictor Lambda_46.4159 Lambda_232.0794 Lambda_464.1589 Lambda_928.3178
1          (Intercept)      4789.9003       4790.2184       4790.6370       4789.7857
2        CrimePerThous       713.4797        712.9455        712.3152        713.6796
3 FemaleLifeExpectancy       492.4674        491.8168        490.9198        492.8667
4            FemalePay       647.9688        648.4523        649.0185        647.7053
5   MaleLifeExpectancy       228.6435        225.9946        222.7323        229.7808
6              MalePay       330.5210        330.6708        330.8731        330.4133
7              PopDens       534.2808        531.6934        528.3652        535.4122
  Lambda_4641.5888   Original  Stepwise
1       4789.6832 393482.258 393482.26
2        713.8519  52459.351  51607.49
3        493.2351  31676.092  50221.86
4        647.1089  59969.620  59432.69
5        231.8989  26384.965        NA
6        330.0414  -5509.902        NA
7        537.4126  36975.779  27769.34
```

Rationale for Regularization and Lambda Tuning

- Prior tests using $\lambda = 10^7$ led to all coefficients except the intercept shrinking to zero, confirming the need for a balanced approach to penalty strength.
- The wide tuning range (10^{-5} to 10^6) identified an optimal lambda through cross-validation, ensuring effective regularization while maintaining predictive accuracy.
- The Lasso regression model, optimized at $\lambda = 464.16$ via cross-validation, reveals key findings compared to both the original and stepwise models. Even under strong regularization (up to $\lambda = 4641.59$), no predictors were fully shrunk to zero, indicating robust associations with the response variable.
- In model comparisons, the original linear model showed large, potentially unstable coefficients (e.g., CrimePerThous: 52459.35), suggesting overfitting. The stepwise model, while simpler, retained multicollinearity issues despite smaller coefficients (e.g., CrimePerThous: 51607.49). In contrast, Lasso balanced accuracy and simplicity with regularized, stable coefficients (e.g., CrimePerThous: 712).
- Cross-validation confirmed that $\lambda = 464.16$ minimizes MSE, improving generalization without sacrificing interpretability. Lasso regression effectively handles complex data with multicollinearity, offering both stability and predictive performance.

(h) Conclusion

- The choice of the best model depends on both interpretability and predictive performance, particularly considering multicollinearity. The **Lasso regression** emerges as the best model due to its balanced handling of complexity and predictive accuracy.
- The **original model** retains all variables, including MaleLifeExpectancy and FemaleLifeExpectancy, which have high VIF values (9.67 and 6.38, respectively). This indicates significant multicollinearity, causing instability in large coefficients like 26384.97 and 31676.09. Despite achieving a high R^2 of 0.90, the model risks overfitting and poor generalization.
- **Stepwise regression**, by contrast, removes high-VIF predictors (MalePay and MaleLifeExpectancy), making non-linear effects clearer. FemaleLifeExpectancy's coefficient increases to 50221.86, reflecting reduced multicollinearity's impact. However, adjusted R^2 falls slightly to 0.89, highlighting a trade-off between simplicity and explanatory power.
- In **Lasso regression**, predictors are retained but their coefficients are regularized. Under $\lambda \approx 464$, MaleLifeExpectancy and FemaleLifeExpectancy have reduced coefficients (222.73 and 490.92), indicating weaker influence. This regularization enhances model stability and resilience to unseen data. Lasso achieves strong predictive performance with improved generalization, making it the optimal model for managing complex, multicollinear datasets.

Question 2

(a) Exploratory Analysis

This analysis explores the factors affecting diabetes among Pima Indian women using logistic regression (dataset from Kaggle). The dataset consists of 768 observations and 9 variables. The initial phase involves summarizing the dataset and visualizing key features to detect patterns and potential data issues.

Handling Missing Values Using Mean Imputation

Certain variables contain zero values that should be treated as missing because it is not logically approved in real life such as `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, `Age`, and `BMI`. To ensure accurate analysis, we replace these values with the mean of their respective columns **before** proceeding with summary statistics.

Dataset Summary, Checking Zero Values, and Correlation with Outcome

Variable	Mean	Median	Q1	Q3	Zero_Values	Correlation
Pregnancies	3.8450521	3.00000	1.00	6.0000	111	0.2219
Glucose	121.6867628	117.00000	99.75	140.2500	0	0.4929
BloodPressure	72.4051842	72.20259	64.00	80.0000	0	0.1661
SkinThickness	29.1534196	29.15342	25.00	32.0000	0	0.2153
Insulin	155.5482234	155.54822	121.50	155.5482	0	0.2144
BMI	32.4574637	32.40000	27.50	36.6000	0	0.3119
Age	33.2408854	29.00000	24.00	41.0000	0	0.2384
Outcome	0.3489583	0.00000	0.00	1.0000	500	1.0000

```
Outcome Counts:  0=500, 1=268

Proportions:  0=0.651, 1=0.349
```

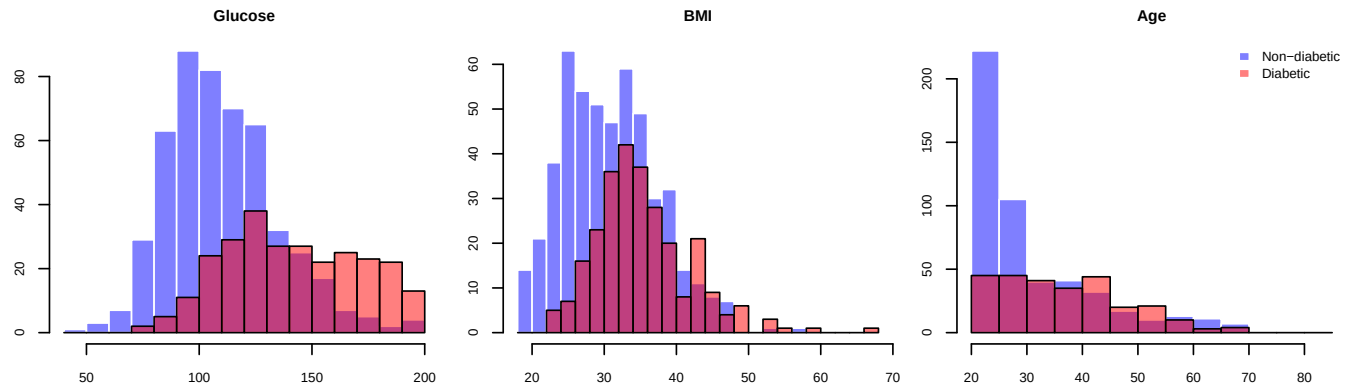
Observations

Class Distribution: The dataset is not perfectly balanced, with 65.1% of cases being non-diabetic (Outcome = 0) and 34.9% being diabetic (Outcome = 1). While this distribution is not extremely imbalanced,the dataset is usable for logistic regression despite the class imbalance. However, accuracy alone may not be a reliable performance metric, as the model might be biased toward predicting non-diabetic cases. To mitigate this issue, additional metrics such as **ROC/AUC, Precision, Recall, and F1-Score** should be considered to better evaluate the model's predictive capability.

Interpretation of Summary, Correlation, and Histogram

Summary statistics reveal that diabetic individuals generally have higher glucose levels (median: 117), BMI (median: 32.4), and are older (mean: 33.24 years). In contrast, variables like BloodPressure, SkinThickness, and Insulin show weaker discrimination between the two groups. Histogram visualizations confirm that diabetics tend to have elevated glucose and BMI values. There is strong separation in glucose distribution, while BMI also highlights the association between obesity and diabetes. Age shows moderate overlap between groups, indicating some variability. Correlation analysis supports these findings. Glucose has the strongest correlation with the outcome (0.4929), followed by BMI (0.3119), emphasizing their importance as predictors. Age (0.2384) and Pregnancies (0.2219) show moderate correlations, while Blood Pressure and Skin Thickness have weaker relationships. These results highlight the need to prioritize glucose control and healthy body weight for diabetes risk management.

Combined Histogram for Key Predictors



(b) Assumption Check and Fit Logistic Regression

	Assumption	Value	Interpretation
1	Multicollinearity (VIF)	1.4065057	No serious multicollinearity
2	Linearity of Logit (Correlation)	0.1293484	Logit linearity acceptable (within tolerance)
3	Independence (Durbin-Watson)	1.9521800	Residuals are independent

Fit full/initial model using following code :

```r
# Fit an initial logistic regression model
initial_model <- glm(Outcome ~ ., data = diabetes_data, family = "binomial")
summary(initial_model)
```

```
Call:
glm(formula = Outcome ~ ., family = "binomial", data = diabetes_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   -8.7888259  0.7968125 -11.030  < 2e-16 ***
Pregnancies    0.1189487  0.0320005   3.717 0.000202 ***
Glucose        0.0375339  0.0038437   9.765  < 2e-16 ***
BloodPressure -0.0097940  0.0084819  -1.155 0.248214
SkinThickness  0.0042600  0.0131400   0.324 0.745787
Insulin       -0.0006754  0.0011635  -0.580 0.561603
BMI            0.0962908  0.0176646   5.451 5.01e-08 ***
Age            0.0140395  0.0094277   1.489 0.136442
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 993.48  on 767  degrees of freedom
Residual deviance: 722.05  on 760  degrees of freedom
AIC: 738.05

Number of Fisher Scoring iterations: 5
```

Assumption Check

All assumptions have been satisfied: there is no serious multicollinearity (VIF = 1.41), logit linearity is acceptable (correlation = 0.13), and residuals are independent (Durbin-Watson = 1.95).

Model Interpretation

Higher **glucose levels**, **BMI**, and **number of pregnancies** are the strongest predictors of diabetes. **Blood pressure**, **skin thickness**, and **insulin levels** do not show significant effects. **Age** has some impact but is less important than glucose or BMI. The model's residual deviance (722.05) indicates improved fit compared to the null model.

(c) Model Refinement

Model Selection Process

Step 1: Initial Deviance Reduction Analysis

The process began by assessing each predictor's contribution through deviance reduction using a Chi-square test. Out of the 7 initial predictors (Pregnancies, Glucose, BMI, BloodPressure, SkinThickness, Insulin, and Age), only three predictors—Pregnancies, Glucose, and BMI—were found to be significant and included in the base model.
Next, we tested two-way and three-way interaction terms among these selected predictors. The deviance reductions and p-values are summarized below:

Term	Degree	Deviance Reduction	P-Value
Pregnancies * Glucose * BMI	3	2.9652	0.5637
Pregnancies * Glucose	2	1.7368	0.1875
Glucose * BMI	2	0.2124	0.6449
Pregnancies * BMI	2	0.0078	0.9297
Pregnancies	1	0.0000	NA
Glucose	1	0.0000	NA
BMI	1	0.0000	NA

Although some interaction terms showed minor deviance reductions, none were statistically significant (p-values > 0.05).

Step 2: Model Generation

Using a base model with the main effects (Pregnancies, Glucose, and BMI), generated and evaluated all candidate models automatically with two-way and three-way interactions.

Step 3: Cross-Validation and Performance Evaluation

Each model was evaluated using 10-fold cross-validation, with the following performance metrics:
- **Accuracy**: The proportion of correctly classified outcomes.
- **F1 Score**: The harmonic mean of precision and recall.
- **AIC/BIC**: Metrics that assess model fit, penalizing for complexity.
- **Deviance**: A measure of goodness-of-fit, where lower values indicate better fit.

Despite reasonable performance metrics, including an accuracy of 76.17%, interaction terms were not statistically significant. Therefore, the initial model containing only the main effects was retained.

Step 4: Final Model Selection

The final model includes only the main effects of Pregnancies, Glucose, and BMI. It demonstrated:
- An accuracy of 76.17% and reasonable F1-score performance.
- Significant main effects for each predictor.
- No significant improvement from adding interaction terms, supporting the decision to retain the simpler model.

```r
# Step 12: Use final model obtained from cross validation
final_model <- glm(Outcome ~ Pregnancies + Glucose + BMI ,
                   data = diabetes_data, family = "binomial")

# Print the summary
summary(final_model)
```

```
Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BMI, family = "binomial",
    data = diabetes_data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.909720   0.687087 -12.967  < 2e-16 ***
Pregnancies  0.138102   0.027195   5.078 3.81e-07 ***
Glucose      0.037242   0.003456  10.776  < 2e-16 ***
BMI          0.092289   0.014645   6.302 2.94e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 993.48  on 767  degrees of freedom
Residual deviance: 725.35  on 764  degrees of freedom
AIC: 733.35

Number of Fisher Scoring iterations: 5
```

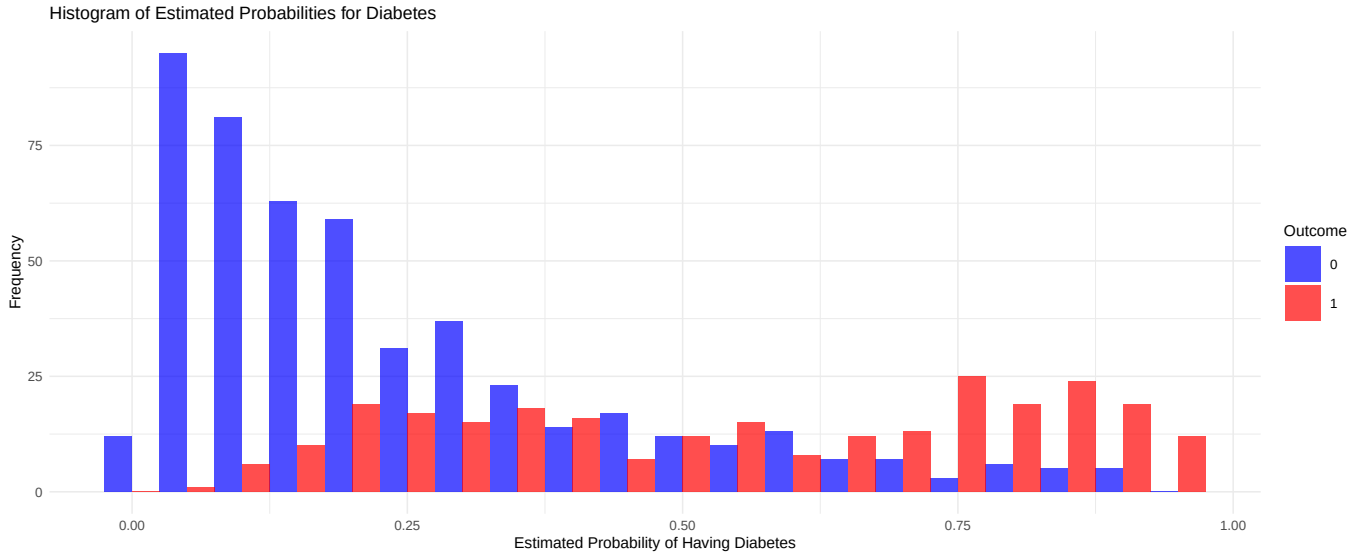
Interpretation of best model

Using $\alpha = 5\%$, the model aims to predict the likelihood of a positive outcome (e.g., diabetes) based on the combination of Pregnancies, Glucose, and BMI. From the results, we can draw the following conclusions:

- The model suggests that individuals with higher values of Glucose, BMI, or Pregnancies are more likely to have a positive outcome.
- Glucose levels appear to be the most influential factor. This indicates that higher blood glucose is strongly associated with a higher risk of a positive outcome.
- The model accounts for 27% of the variability in the data, meaning that while these three predictors help explain part of the risk, there are still other factors that could affect the outcome that are not included in this model.
- All predictors have statistically significant effects, meaning their contributions are unlikely to be due to random chance.

Overall, the model highlights the importance of monitoring glucose levels and BMI when assessing the likelihood of a positive outcome in this dataset.

(d) Estimated Probabilities (of having diabetes) using Histogram



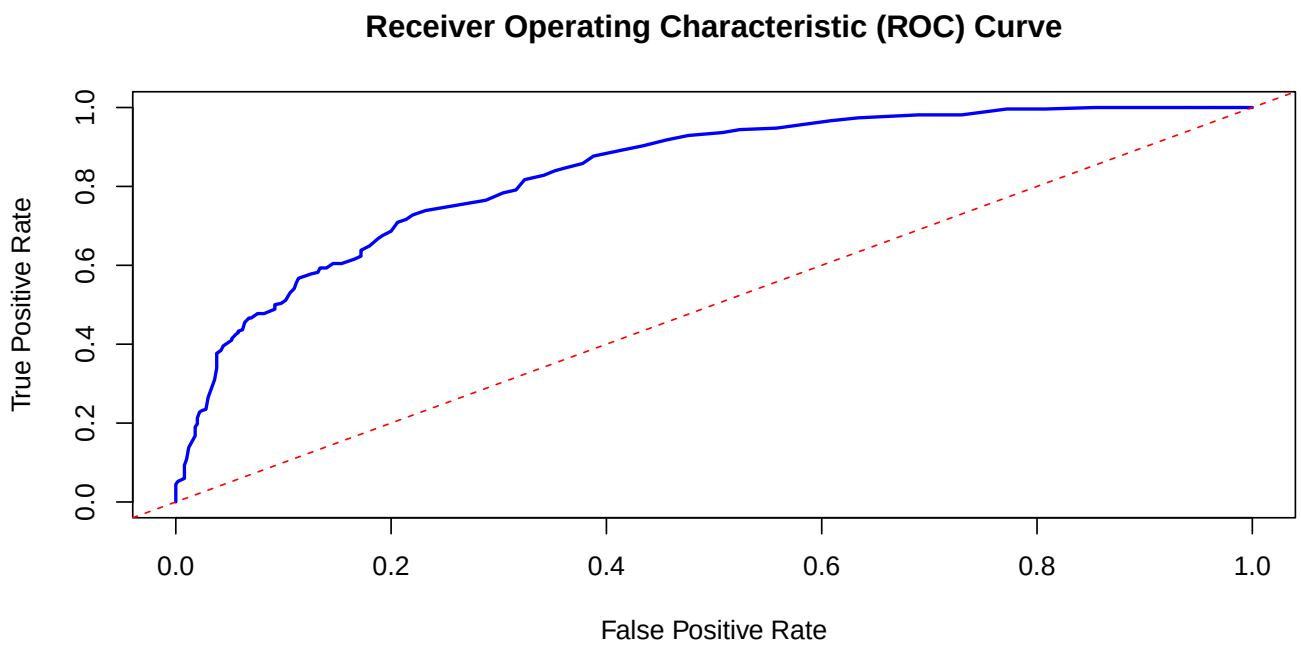
Interpretation of the Estimated Probability Histogram:

The histogram shows how the predicted probabilities from the logistic regression model are distributed across the two outcome groups.

- Individuals without diabetes (Outcome = 0, blue) mostly have low predicted probabilities. This indicates that the model classifies them as having a low risk for diabetes.
- Individuals with diabetes (Outcome = 1, red) tend to have higher predicted probabilities, suggesting that the model identifies them as higher risk.
- Ideally, a well-performing model should display clear separation between these groups, with non-diabetic cases near zero and diabetic cases near one.

- If there is substantial overlap between the two distributions, the model may have difficulty distinguishing between the two groups, leading to potential misclassifications.
- Reduced overlap implies greater model confidence, while significant overlap suggests that further refinement or additional features may improve the model's classification ability.

(e) ROC Curve




```
The calculated AUC is: 0.8357
```

Interpretation of the ROC Curve and Analysis:

- This ROC curve plots the **True Positive Rate (Sensitivity)** against the **False Positive Rate** at various threshold levels.
- Ideally, a well-performing model will have a curve that moves closer to the **top-left corner**, indicating high TPR with low FPR.
- **The diagonal red line** represents a random classifier (i.e., a model that predicts no better than chance). A good model should have its curve well above this line.
- The **shape of the ROC curve** helps assess the trade-off between sensitivity and specificity for different threshold values.
- The calculated AUC value is **0.8357**, indicating that the model has a good balance between sensitivity and specificity.

(f) Optimal Threshold Function

```r
# --- Step 1: Define the Function ---
find_optimal_threshold <- function(probabilities, actual_outcomes) {
  # Define thresholds between 0 and 1, stepping by 0.01
  thresholds <- seq(0, 1, by = 0.01)

  # Initialize a vector to store the combined metric values
  optimal_metric <- numeric(length(thresholds))

  # --- Step 2: Loop through each threshold ---
  for (i in seq_along(thresholds)) {
    threshold <- thresholds[i]

    # Classify observations based on the current threshold
    predicted_class <- ifelse(probabilities > threshold, 1, 0)

    # Confusion matrix components
    tp <- sum(predicted_class == 1 & actual_outcomes == 1)  # True positives
    fp <- sum(predicted_class == 1 & actual_outcomes == 0)  # False positives
    tn <- sum(predicted_class == 0 & actual_outcomes == 0)  # True negatives
    fn <- sum(predicted_class == 0 & actual_outcomes == 1)  # False negatives

    # Avoid division by zero errors and calculate TPR and FPR
    tpr <- ifelse((tp + fn) > 0, tp / (tp + fn), 0)  # True Positive Rate
    fpr <- ifelse((fp + tn) > 0, fp / (fp + tn), 0)  # False Positive Rate
```

```
    # Calculate the optimization metric (TPR + (1 - FPR))
    optimal_metric[i] <- tpr + (1 - fpr)
  }

  # --- Step 3: Find the Optimal Threshold ---
  best_index <- which.max(optimal_metric)  # Index of maximum metric value
  best_threshold <- thresholds[best_index]  # Optimal threshold

  # Display the optimal threshold and its metric value
  cat("Optimal Threshold based on (TPR + (1 - FPR)): ", best_threshold,
      " with a metric value of: ", round(optimal_metric[best_index], 6), "\n")

  return(best_threshold)  # Return the optimal threshold
}

# --- Step 4: Use the Function ---
predicted_probabilities <- fitted(final_model)  # Get predicted probabilities
optimal_threshold <- find_optimal_threshold(predicted_probabilities, diabetes_data$Outcome)
```

```
Optimal Threshold based on (TPR + (1 - FPR)):  0.34  with a metric value of:  1.507612
```

```
# --- Step 5: Calculate Accuracy using optimal threshold ---
final_predictions <- ifelse(predicted_probabilities > optimal_threshold, 1, 0)
accuracy <- mean(final_predictions == diabetes_data$Outcome)
cat("Accuracy at Optimal Threshold: ", round(accuracy * 100, 2), "%\n")
```

```
Accuracy at Optimal Threshold:  76.17 %
```

Explanation of the Code for Part (f):

- Step 1: Define the Function
  The function `find_optimal_threshold` calculates the optimal threshold by maximizing the metric TPR + (1 − FPR). It iterates through thresholds from 0 to 1, classifying predictions based on each threshold.
- Step 2: Loop through Each Threshold
  For each threshold:
- Classify predictions as 1 (positive) or 0 (negative).
- Compute confusion matrix components: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).
- Calculate TPR and FPR: Step 3: Find the Optimal Threshold
  The threshold with the highest metric value is selected as optimal. The function returns and prints this threshold.
- Step 4: Use the Function
  The model's predicted probabilities are passed to the function, and the optimal threshold is found.

Part D: Histogram Interpretation

The histogram shows that individuals without diabetes (blue) have low predicted probabilities, while diabetic cases (red) generally have higher predicted values. However, overlap exists between the probabilities in the range of 0.2 to 0.7, which may lead to misclassification. This overlap suggests that a lower threshold may improve sensitivity.

Optimal Threshold

The optimal threshold based on the metric TPR + (1 - FPR) is 0.34, with a metric value of 1.507612 . A lower threshold helps identify more diabetic cases but increases false positives. This trade-off is appropriate due to the overlap seen in the histogram.

ROC Curve and AUC

The ROC curve measures how well the model balances sensitivity and the false positive rate across different thresholds. The calculated Area Under the Curve (AUC) is 0.8357, which indicates good model performance. AUC values near 1 imply that the model can effectively separate diabetic and non-diabetic cases.

Accuracy at Optimal Threshold

The model achieves 76.17% accuracy at the optimal threshold of 0.34. While this indicates a generally good fit, accuracy should not be the sole evaluation metric due to the class imbalance. The ROC curve and AUC provide a more reliable assessment of model performance.