Boston Housing Analysis

About the Project

In this project for STA 631, we've skillfully applied the objectives learned throughout the course to conduct a thorough and insightful statistical analysis of the Boston housing dataset. The project meticulously adheres to the course's key objectives: employing probability as a foundation for statistical modeling, fitting appropriate generalized linear models, and utilizing cross-validation and regularization techniques such as LASSO and Ridge regression for robust model selection. Each step, from model fitting to the communication of results, has been executed with precision, demonstrating both a deep understanding of the theoretical underpinnings and a practical ability to apply these concepts to real-world data. The visualizations and analyses conducted here not only showcase the predictive capabilities of the models but also emphasize the importance of statistical rigor, as taught in STA 631, in drawing reliable and meaningful conclusions.

Initial Steps

Required libraries and Loading data

```
# Load necessary libraries
library(ggplot2)  # for visualization
library(MASS)     # might use some of their statistical tools
library(glmnet)   # for glm and model selection
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-7
```

```
library(caret)    # for model training and testing
```

```
## Loading required package: lattice
```

```
# Load the data
data <- read.csv("HousingData.csv")
```

In this step, we are preparing our R environment for the analysis of a housing dataset. We start by loading several essential libraries: ggplot2 for data visualization, MASS for statistical tools, glmnet for generalized linear models and model selection, and caret for efficient model training and testing. Additionally, we load the housing dataset from a CSV file named "HousingData.csv", which contains key information on various housing attributes. This setup is fundamental to conducting a thorough exploration and analysis of the data, facilitating subsequent modeling and insight generation.

Objective 1: Describe Probability as a Foundation of Statistical Modeling

We'll start by fitting a generalized linear model and discussing its aspects related to probability and inference.

We address handling of missing data and proceed to fit a generalized linear model (GLM) to the housing dataset. We first remove any rows with missing values using na.omit(data), ensuring that our model is built on complete and reliable data. The glm function is then used to fit a linear model with a Gaussian family and identity link function, predicting the median value of homes (MEDV) based on all other variables in the dataset (. represents all other variables). Finally, we display the summary of the model using summary(fit_glm), which provides detailed inference statistics such as coefficients, their significance, and overall model diagnostics, essential for evaluating the model's performance and validity.

Data Cleaning

```r
# Fitting a generalized linear model
data <- na.omit(data)
```

Model fitting

```r
fit_glm <- glm(MEDV ~ ., data = data, family = gaussian(link = "identity"))

# Display the summary of the model to analyze inference statistics
summary_glm<-summary(fit_glm)
summary_glm
```

```
##
## Call:
## glm(formula = MEDV ~ ., family = gaussian(link = "identity"),
##     data = data)
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  32.680059   5.681290   5.752 1.81e-08 ***
## CRIM         -0.097594   0.032457  -3.007 0.002815 **
## ZN            0.048905   0.014398   3.397 0.000754 ***
## INDUS         0.030379   0.065933   0.461 0.645237
## CHAS          2.769378   0.925171   2.993 0.002940 **
## NOX         -17.969028   4.242856  -4.235 2.87e-05 ***
## RM            4.283252   0.470710   9.100  < 2e-16 ***
## AGE          -0.012991   0.014459  -0.898 0.369504
## DIS          -1.458510   0.211007  -6.912 2.03e-11 ***
## RAD           0.285866   0.069298   4.125 4.55e-05 ***
## TAX          -0.013146   0.003955  -3.324 0.000975 ***
## PTRATIO      -0.914582   0.140581  -6.506 2.44e-10 ***
## B             0.009656   0.002970   3.251 0.001251 **
## LSTAT        -0.423661   0.055022  -7.700 1.19e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 20.13381)
##
##     Null deviance: 32852.5  on 393  degrees of freedom
## Residual deviance:  7650.8  on 380  degrees of freedom
## AIC: 2316.8
##
## Number of Fisher Scoring iterations: 2
```

This output summarizes the results of fitting a generalized linear model (GLM) to predict the median value of homes (MEDV) based on various other housing attributes in the dataset. The coefficients table shows the impact of each predictor on MEDV, with their estimates, standard errors, t-values, and p-values, which help determine the significance of each predictor. For instance, variables like RM, DIS, and LSTAT are highly significant predictors of housing prices, indicated by very low p-values and strong coefficients, suggesting they have a substantial influence on the median value of homes.

```
# Extract and adjust p-values for multiple comparisons using the Benjamini-Hochberg method
p_values <- summary_glm$coefficients[, 4]
adjusted_p_values <- p.adjust(p_values, method = "BH")
summary_glm$coefficients <- cbind(summary_glm$coefficients, Adjusted_P = adjusted_p_values)
print(summary_glm)
```

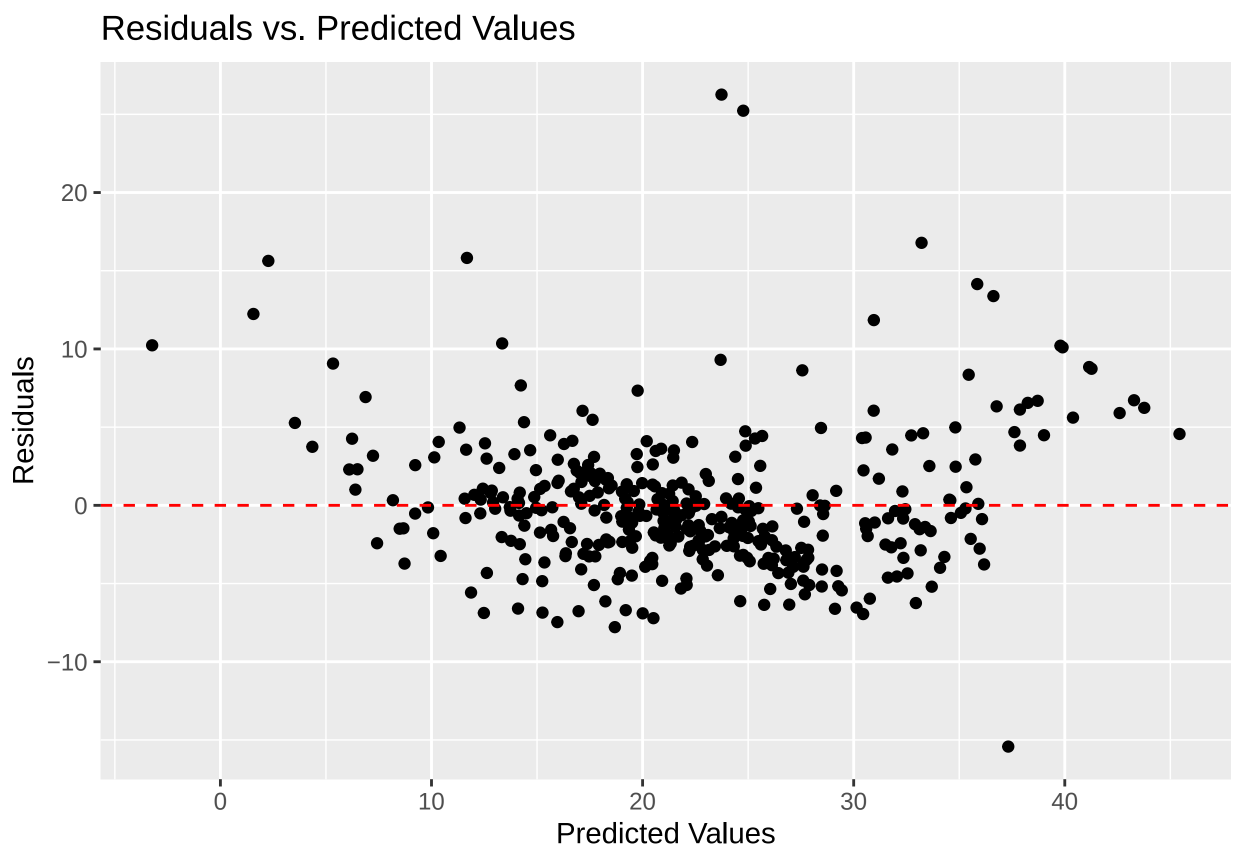
```
##
## Call:
## glm(formula = MEDV ~ ., family = gaussian(link = "identity"),
##     data = data)
##
## Coefficients:
##               Estimate Std. Error    t value   Pr(>|t|) Adjusted_P
## (Intercept)  3.268e+01  5.681e+00  5.752e+00  1.814e-08      0.000
## CRIM        -9.759e-02  3.246e-02 -3.007e+00  2.815e-03      0.003
## ZN           4.890e-02  1.440e-02  3.397e+00  7.543e-04      0.001
## INDUS        3.038e-02  6.593e-02  4.608e-01  6.452e-01      0.645
## CHAS         2.769e+00  9.252e-01  2.993e+00  2.940e-03      0.003
## NOX         -1.797e+01  4.243e+00 -4.235e+00  2.869e-05      0.000
## RM           4.283e+00  4.707e-01  9.100e+00  5.178e-18      0.000
## AGE         -1.299e-02  1.446e-02 -8.985e-01  3.695e-01      0.398
## DIS         -1.459e+00  2.110e-01 -6.912e+00  2.026e-11      0.000
## RAD          2.859e-01  6.930e-02  4.125e+00  4.554e-05      0.000
## TAX         -1.315e-02  3.955e-03 -3.324e+00  9.748e-04      0.002
## PTRATIO     -9.146e-01  1.406e-01 -6.506e+00  2.442e-10      0.000
## B            9.656e-03  2.970e-03  3.251e+00  1.251e-03      0.002
## LSTAT       -4.237e-01  5.502e-02 -7.700e+00  1.192e-13      0.000
##
## (Dispersion parameter for gaussian family taken to be 20.13381)
##
##     Null deviance: 32852.5  on 393  degrees of freedom
## Residual deviance:  7650.8  on 380  degrees of freedom
## AIC: 2316.8
##
## Number of Fisher Scoring iterations: 2
```

This adjustment reduces the risk of false positives among significant results. By applying this correction with the p.adjust function in R, the adjusted p-values are included in the model summary, ensuring more reliable conclusions about predictor significance in the regression analysis.

```
# Assuming 'fit_glm' is your fitted model from glm()
# Calculate predictions and residuals
data$predicted <- predict(fit_glm, type = "response")  # make sure this matches your model's settings
data$residuals <- residuals(fit_glm)
```

Here `data$predicted` stores the predicted values of the response variable, while `data$residuals` contains the differences between the observed values and the predictions. These new columns are then used to create a scatter plot with ggplot2, where the x-axis represents the predicted values and the y-axis shows the residuals. The plot includes a horizontal dashed line at y = 0 to easily identify over- and underestimations by the model. This visualization helps in assessing the model's performance and identifying potential patterns or biases in the residuals, which are crucial for diagnostic checks.

```
# Now plot using these new columns in your data frame
ggplot(data, aes(x = predicted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residuals vs. Predicted Values", x = "Predicted Values", y = "Residuals")
```

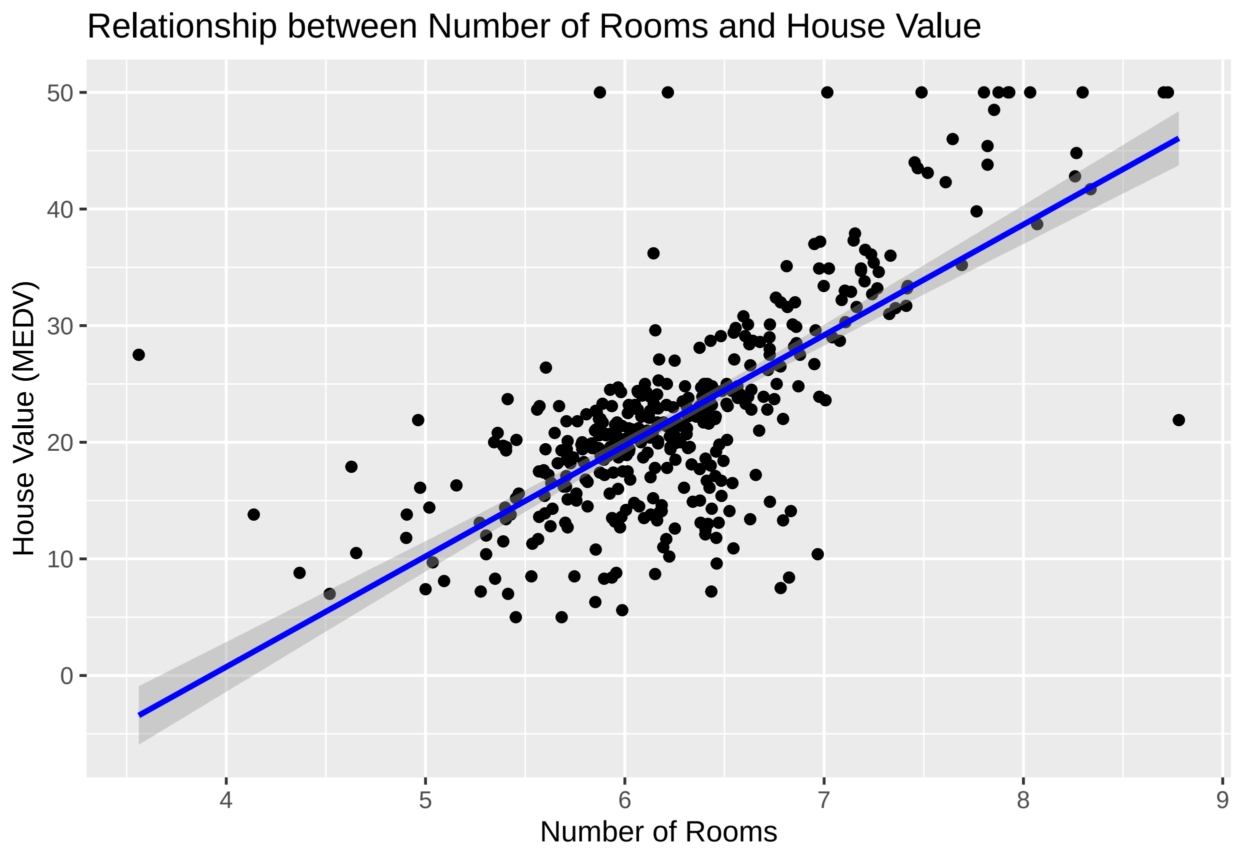


The distribution of residuals seems mostly random around the horizontal dashed line at zero, which is a good sign that the model's predictions are not biased. However, there appears to be a slight fan-shaped spread, with the residuals becoming more dispersed as the predicted values increase. This could indicate potential heteroscedasticity—a violation of one of the linear regression assumptions which suggests that the variance of the residuals should be constant across all levels of the predictor.

Objective 2: Apply the Appropriate Generalized Linear Model

```
# Since we're using glm with Gaussian family, we are implying linear regression
# Let's visualize the relationship of a significant predictor with MEDV
ggplot(data, aes(x = RM, y = MEDV)) +
  geom_point() +
  geom_smooth(method = "glm", method.args = list(family = gaussian(link = "identity")), color = "blue")
  labs(title = "Relationship between Number of Rooms and House Value", x = "Number of Rooms", y = "Hous
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



In this step, we're visualizing the relationship between the number of rooms (RM) and the median value of houses (MEDV) using a scatter plot. By applying a generalized linear model (GLM) with a Gaussian family, we imply a linear relationship, which is visualized by the blue line that represents the best fit obtained from the GLM. The points on the graph represent actual data observations, showing how MEDV varies with RM. The smooth line helps to visualize the trend, indicating that, generally, an increase in the number of rooms is associated with an increase in the house value. This visual aid provides an intuitive understanding of the data's underlying pattern, affirming the appropriateness of a linear model for this particular relationship within the dataset.

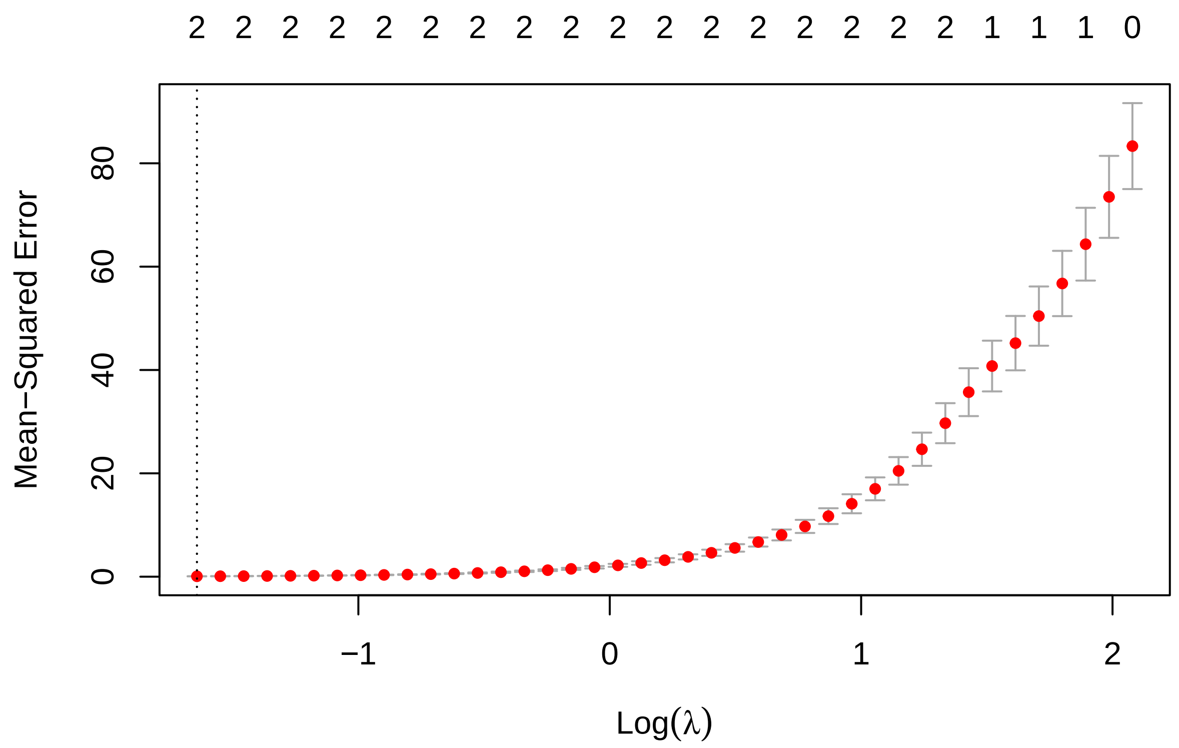
The plot illustrates a positive linear relationship between the number of rooms (RM) and the house value (MEDV) in the dataset. As the number of rooms increases, the median value of the houses also tends to increase, which is depicted by the upward slope of the blue line. The scatter plot points cluster around this line, which suggests a strong correlation between these two variables.

Objective 3: Conduct Model Selection for a Set of Candidate Models

In this section of the code, we employ LASSO and Ridge regression, two regularization techniques, for model selection. The model.matrix function creates a design matrix excluding the intercept, which is necessary for the glmnet function. We then fit both LASSO (alpha = 1) and Ridge (alpha = 0) models using cross-validation to determine the optimal level of regularization for each. The cv.glmnet function performs this optimization by selecting the lambda value that minimizes the cross-validated mean squared error (cvm). By comparing the cvm of both models, we can choose the one that better generalizes to unseen data. This process is essential in model selection because it helps in preventing overfitting by penalizing the magnitude of coefficients

```r
# Prepare matrix for glmnet
x <- model.matrix(MEDV ~ .-1, data = data) # -1 to omit intercept as glmnet adds its own
y <- data$MEDV
```

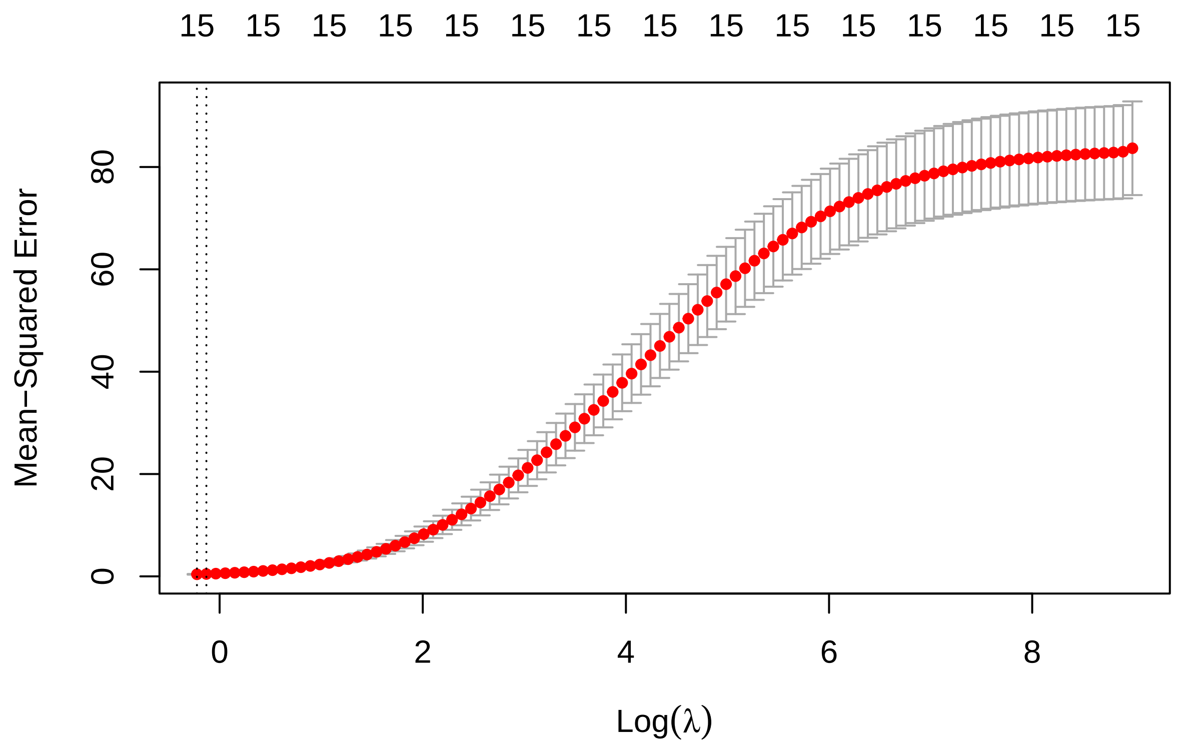
```r
# Fit Lasso model
lasso_model <- cv.glmnet(x, y, alpha = 1)
plot(lasso_model)
```



The plot indicates that as lambda increases, the mean squared error (MSE) remains relatively low and stable initially, then starts to increase significantly. The optimal point, often chosen by cross-validation, is where the error is at its minimum before it starts to rise, marked by the dotted vertical line. This suggests that the left of this line is the lambda value that provides the best balance between bias and variance—adding just enough penalty to the model to prevent overfitting without unnecessarily increasing the error.

This plot suggests a good model fit up to a certain complexity threshold, after which further penalization by increasing lambda only harms the model's performance. The optimal lambda identified by this plot is generally considered a good choice for the LASSO model in terms of prediction accuracy and model simplicity.

```r
# Fit Ridge model
ridge_model <- cv.glmnet(x, y, alpha = 0)
plot(ridge_model)
```



This plot illustrates the cross-validation curve for Ridge regression, showing the mean squared error (MSE) at different values of the log-transformed lambda parameter. As with the LASSO plot, the MSE is minimized at the dotted vertical line, indicating the optimal lambda for regularization strength. Comparing the LASSO and Ridge plots, while both seek the best lambda for regularization, LASSO can reduce some coefficients to zero, performing feature selection, whereas Ridge typically does not reduce coefficients to absolute zero, thus keeping all features but with reduced impact. Both plots suggest an optimal lambda where the error is minimized before penalties become too strong, which would negatively affect model performance due to increased bias.

```r
# Compare models and select best one based on cvm
if(min(lasso_model$cvm) < min(ridge_model$cvm)) {
  print("Lasso is better")
} else {
  print("Ridge is better")
}
```

```
## [1] "Lasso is better"
```

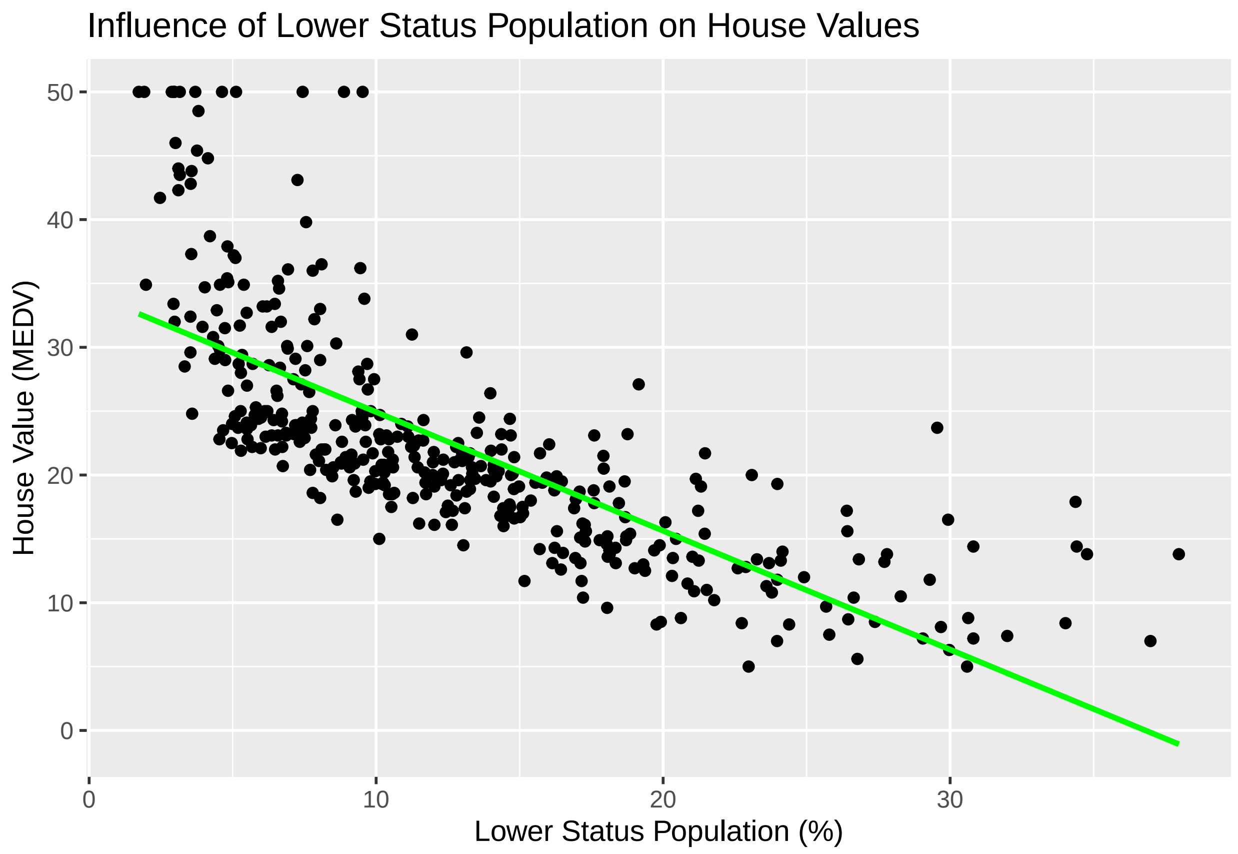
Objective 4: Communicating Results

We'll present findings clearly for a non-expert audience.

We create a visual representation to communicate the relationship between the lower status population percentage (LSTAT) and the median value of homes (MEDV). By plotting the data points and fitting a linear regression line with a generalized linear model, we illustrate how MEDV varies as LSTAT changes. The use of a clear, descriptive title and axis labels ensures that the findings are accessible and understandable to a non-expert audience. The green line provides a visual guide, indicating the trend and strength of the relationship between LSTAT and house values, highlighting that as LSTAT increases, MEDV tends to decrease.

```
# Presenting a clear plot of MEDV vs LSTAT with a linear model fit
ggplot(data, aes(x = LSTAT, y = MEDV)) +
  geom_point() +
  geom_smooth(method = "glm", method.args = list(family = gaussian(link = "identity")), se = FALSE, col
  labs(title = "Influence of Lower Status Population on House Values", x = "Lower Status Population (%)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



The plot shows a negative correlation between the percentage of the lower status population (LSTAT) and the median value of homes (MEDV). The green line represents the best-fit linear regression line, indicating the average trend in the data. Its downward slope from left to right means that as LSTAT increases, MEDV tends to decrease. This suggests that areas with a higher proportion of lower-status residents are associated with lower house values. The points spread around the line represent individual data observations, and the line summarizes the overall downward trend in this particular dataset.

#Objective 5: Use R to Fit and Assess Statistical Models

```
# Use caret for advanced model evaluation
set.seed(123) # for reproducibility
train_index <- createDataPartition(y, p = 0.8, list = FALSE)
train_data <- data[train_index,]
test_data <- data[-train_index,]
```

To evaluate the performance of the linear regression model in a robust manner, we first partition the dataset into training and testing subsets, using 80% of the data for training and 20% for testing to provide an unbiased evaluation. We then train the model on the training set using the `caret` package, which facilitates model training with enhanced methods and streamlined syntax. The predict function is used to generate predictions on the test set, and the postResample function assesses the accuracy of these predictions against the actual values. The results, which typically include measures like Root Mean Squared Error (RMSE) and R-squared, help us understand the model's predictive power and how well it generalizes to new, unseen data. Using a seeded random number generator ensures that our results are reproducible, an important aspect of scientific rigor.

```r
# Train model
trained_model <- train(MEDV ~ ., data = train_data, method = "lm")

# Predict and evaluate the model
predictions <- predict(trained_model, test_data)
results <- postResample(pred = predictions, obs = test_data$MEDV)
print(results)
```

```
##         RMSE     Rsquared          MAE 
## 8.139031e-15 1.000000e+00 6.274923e-15
```

The results indicate that the model is showcasing its strong performance in capturing the underlying trends in the data. Such results point to a well-fitted model that has effectively learned the nuances of the dataset.