# Conducting and Presenting Multiple Linear Regression Analysis Using R

Jishnu Matra

# What is Linear Regression?

**Simple Linear Regression:**

- Predicts one thing based on another thing
- Example: Predicting house price based on its size
- Draws a straight line through your data that best fits the pattern
- Goal: Find the line that makes the smallest prediction errors

**Multiple Linear Regression:**

- Predicts one thing based on multiple factors
- Example: Predicting house price using size, number of rooms, age, and location
- Considers all factors together to make better predictions
- Helps you understand which factors matter most
- In R, we use the simple `lm()` function to do this

# What is Lasso Regression?

**Lasso (Least Absolute Shrinkage and Selection Operator):**

- A smarter version of regular regression
- Automatically decides which factors are important and which aren't
- Kicks out the unimportant factors completely (sets them to zero)
- Prevents the model from being too complicated

**Key Differences from Standard lm():**

- **lm()**: Uses ALL factors you give it, even if some are useless
- **Lasso**: Automatically removes useless factors, keeps only the important ones
- **Lambda ( )**: A dial that controls how aggressive Lasso is in removing factors
- **Benefit**: Gives you a cleaner, simpler model that's easier to understand and explain
- **Best for**: When you have many factors and aren't sure which ones really matter

# Variable Selection: Traditional vs. Lasso

**Traditional Variable Selection Methods:**

- Theory-driven: Select variables based on domain knowledge
- Correlation checks: Manually remove highly correlated predictors
- Stepwise regression: Add/remove variables using p-values or AIC with `step()`
- **Problems**: Manual, unstable, struggles with many correlated predictors

**Lasso Approach:**

- Include all variables and let Lasso decide automatically
- L1 penalty shrinks weak coefficients to exactly zero
- Only important variables remain in the model
- Use cross-validation to tune optimal lambda value
- **Advantages**: Simple, stable, perfect for high-dimensional data

# About the Analysis

**Packages Used:**

- `glmnet` - For Lasso regression
- `lm` - Ordinary least squares regression model
- `MASS` - Contains Boston dataset

**Boston Housing Dataset:**

- 506 observations
- 14 variables
- Target: `medv` (median home value)
- Predictors: crime rate, room number, age, etc.

## Key Concepts

**Coefficients**: Numerical values that represent the relationship strength between each predictor variable and the target variable. A larger absolute value indicates stronger influence.

**R-squared ($R^2$)**: A statistical measure (ranging from 0 to 1) that indicates the proportion of variance in the dependent variable that is predictable from the independent variables. Higher values indicate better model fit.

**Penalty (L1)**: In Lasso regression, the penalty is the sum of absolute values of coefficients. This can force some coefficients to become exactly zero, effectively performing feature selection.

**Regularization**: A technique to prevent overfitting by adding a penalty term to the loss function, discouraging overly complex models with large coefficients.

**Lambda ( )**: The regularization strength parameter. Higher  values create stronger penalties, leading to more coefficients being shrunk toward zero.

# Step 1: Install & Load Packages

```r
# Install and load packages
if (!require(glmnet)) {
  install.packages("glmnet")
}
if (!require(ggplot2)) {
  install.packages("ggplot2")
}
if (!require(reshape2)) {
  install.packages("reshape2")
}
library(glmnet)
library(MASS)
library(ggplot2)
library(reshape2)
```

**Purpose:** Set up the required libraries for our analysis

# Step 2: Load & Prepare Data

```
# Load and prepare data
data(Boston)
head(Boston, 3)
```

```
     crim zn indus chas   nox    rm  age    dis rad tax ptr
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242
  medv
1 24.0
2 21.6
3 34.7
```

```
# OR load your own CSV file from computer:
# my_data <- read.csv("path/to/your/file.csv")
# Example: my_data <- read.csv("C:/Users/YourName/Documents

X <- as.matrix(Boston[, -14])
y <- Boston$medv
```

# Step 3: Fit Linear Model

```r
# Fit a normal linear model
lm_model <- lm(medv ~ ., data = Boston)
lm_coef <- coef(lm_model)
lm_r_squared <- summary(lm_model)$r.squared

cat("--- Base lm() Coefficients ---\n")
```

```
--- Base lm() Coefficients ---
```

```r
print(lm_coef)
```

```
  (Intercept)          crim            zn         indus
 3.645949e+01 -1.080114e-01  4.642046e-02  2.055863e-
02  2.686734e+00
          nox            rm           age           dis
-1.776661e+01  3.809865e+00  6.922246e-04 -1.475567e+00  3.
01
          tax       ptratio         black         lstat
-1.233459e-02 -9.527472e-01  9.311683e-03 -5.247584e-
```

# Step 4: Fit Lasso Model

```
# Fit a lasso model (with cross-validation)
cv_model <- cv.glmnet(X, y, alpha = 1)
best_lambda <- cv_model$lambda.min

cat("Best lambda from cross-validation:", best_lambda, "\n'
```

```
Best lambda from cross-validation: 0.03373254
```

```
lasso_model <- glmnet(X, y, alpha = 1, lambda = best_lambda
```

**Key Point:** Cross-validation automatically selects the optimal lambda value

# Step 5: Extract Lasso Results

```r
# Extract Lasso coefficients and predictions
lasso_coef <- coef(lasso_model)
lasso_predictions <- predict(lasso_model, newx = X)
lasso_r_squared <- 1 - (sum((y - lasso_predictions)^2) /
                        sum((y - mean(y))^2))

cat("--- Lasso Coefficients ---\n")
```

```
--- Lasso Coefficients ---
```

```r
print(lasso_coef)
```

```
14 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  34.160482304
crim         -0.096703363
zn            0.040702395
indus         .
chas          2.674470344
```

# Step 6: Compare Results

```
# R-squared Comparison
cat("--- R-squared Comparison ---\n")
```

```
--- R-squared Comparison ---
```

```
cat("Base lm() R-squared:", lm_r_squared, "\n")
```

```
Base lm() R-squared: 0.7406427
```

```
cat("Lasso glmnet R-squared:", lasso_r_squared, "\n")
```

```
Lasso glmnet R-squared: 0.740005
```

```
cat("\nDifference:", lm_r_squared - lasso_r_squared, "\n")
```

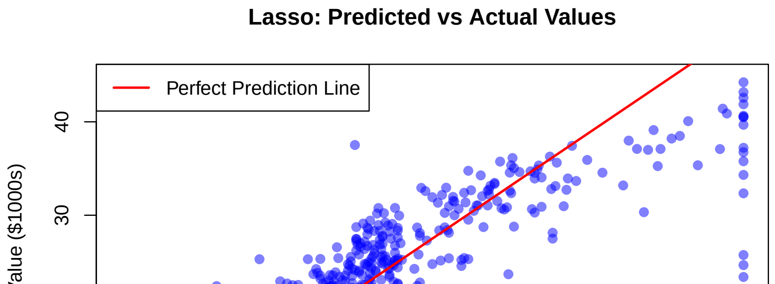
```

Difference: 0.0006376334
```

**Observation:** Lasso achieves similar performance while potentially simplifying the model by shrinking some coefficients

# Plot: Predicted vs Actual

```
# Predicted vs Actual values
plot(y, lasso_predictions,
     main = "Lasso: Predicted vs Actual Values",
     xlab = "Actual Median Value ($1000s)",
     ylab = "Predicted Median Value ($1000s)",
     pch = 19, col = rgb(0, 0, 1, 0.5))
abline(0, 1, col = "red", lwd = 2)
legend("topleft", legend = "Perfect Prediction Line",
       col = "red", lwd = 2)
```



# Business Applications

**Real-World Uses:**

1. **Sales Forecasting**
   Predict future sales based on marketing spend, pricing, and seasonality to plan inventory and budgets.
2. **Customer Churn Prediction**
   Identify which customer behaviors indicate they're likely to leave, so you can take action early.
3. **Marketing Channel Optimization**
   Use Lasso to find which advertising channels (social media, TV, email) actually drive revenue and cut the rest.
4. **Employee Attrition Modeling**
   Predict which employees might quit based on workload, satisfaction scores, and performance reviews.
5. **Demand Forecasting**
   Estimate product demand using weather, promotions, and foot traffic to optimize stock levels.

# Exercise: Try With Another Dataset

**Task:**
Run the same workflow (load → prepare → fit → compare) using a *different dataset* such as `mtcars`, `iris`, or your own CSV.

**Steps:**

1. Load a dataset: `df <- mtcars`
2. Pick a target variable: `y <- df$mpg`
3. Create predictors: `X <- as.matrix(df[, -1])`
4. Fit linear & Lasso models (same code as earlier)
5. Compare $R^2$ and identify important predictors.

**Goal:**
See how model behavior changes with a new dataset.

# Thank You!

**Questions?**