Project 2: Predicting Life Expectancy

Marissa Alfieri, Jenny Zheng, Letitia Caspersen

2025-12-10

Introduction

Life expectancy says a lot about how people live and the conditions they experience around the world. In our first project, we focused on how life expectancy differed across continents, how it changed from 2000 to 2015, and how it related to CO_2 emissions. This project builds on that work by shifting from describing differences in life expectancy to predicting life expectancy using multiple linear regression. Instead of only comparing groups, the goal here is to understand how several environmental, health, and structural factors work together to shape global life expectancy.

The data come from Kaggle's Life Expectancy 2000–2015 dataset (https://www.kaggle.com/datasets/vrec99/life-expectancy-2000-2015), which combines information from the World Health Organization, World Bank, and the United Nations.

Multiple Linear Regression Model for Life Expectancy

In this section, we model life expectancy using multiple predictors in order to better understand how different factors relate to global life expectancy. Rather than analyzing each variable separately, multiple linear regression allows us to examine how these predictors work together to explain variation in life expectancy.

The Hypotheses

This test evaluates whether the regression model provides any predictive value at all:

$$H_0 : \beta_1 = \beta_2 = \beta_3 = \cdots = 0$$
$$H_a : \text{At least one } \beta_j \neq 0$$

The response variable is life expectancy, and the predictors include CO_2 emissions, health expenditure, adult obesity, continent, and least-developed status.

The full regression model can be written in matrix form as:

$$Y = X\beta + \varepsilon$$

where Y is the vector of life expectancy values, X is the design matrix containing the predictors, β is the vector of regression coefficients, and ε represents the random error.

Model Construction

```r
data= read.csv("Life_Expectancy_00_15.csv", sep = ";")
Y = data$Life.Expectancy
X1 = data$CO2.emissions
X2 = data$Health.expenditure
```

```r
X3 = data$Obesity.among.adults
C = model.matrix(~ Continent, data)[ , -1]
D  = model.matrix(~ Least.Developed, data)[ , -1]
X = cbind(1, X1, X2, X3, C, D)
```

Here, an intercept column of 1's is included in the design matrix, and dummy variables are created for the categorical predictors (continent and least-developed status) using reference groups.

Checking The Assumptions

We planned to fit this model using multiple linear regression, so we first checked whether the assumptions of linearity, normality, homoscedasticity, and independence were met.

1. Linearity

$H_0 : E(Y|X) = X\beta$ (the relationship between the predictors and the response is linear)

H_a : At least one predictor has a nonlinear relationship with the response

To assess linearity, we examined a residuals versus fitted values plot. If the model is appropriate, the residuals should be randomly scattered around 0 with no clear curved pattern.

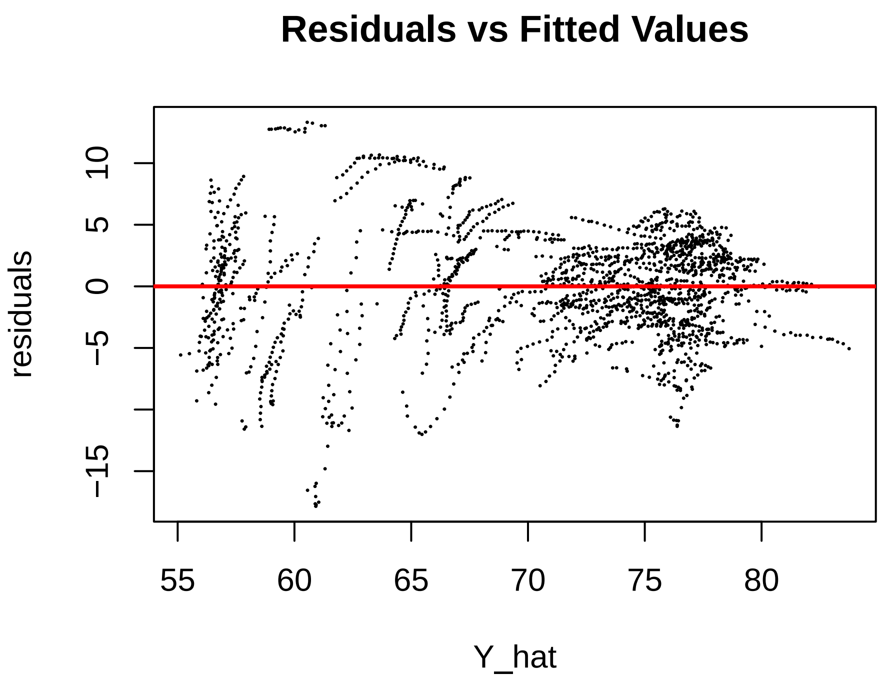
Using the least squares estimator,

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

we computed the fitted values and residuals as:

$$\hat{Y} = X\hat{\beta}, \quad \hat{\varepsilon} = Y - \hat{Y}$$

```r
beta_hat = solve(t(X) %*% X) %*% (t(X) %*% Y)
Y_hat = X %*% beta_hat
residuals = Y - Y_hat

plot(Y_hat, residuals, cex=0.1, main="Residuals vs Fitted Values")
abline(0, 0, lwd=2, col='red')
```



From the residuals versus fitted values plot, the residuals appear generally centered around zero without a strong global curved pattern. This suggests that the linearity assumption is mostly reasonable for this model.

2. Normality

H_0 : The regression errors are normally distributed.

H_a : The regression errors deviate from normality.

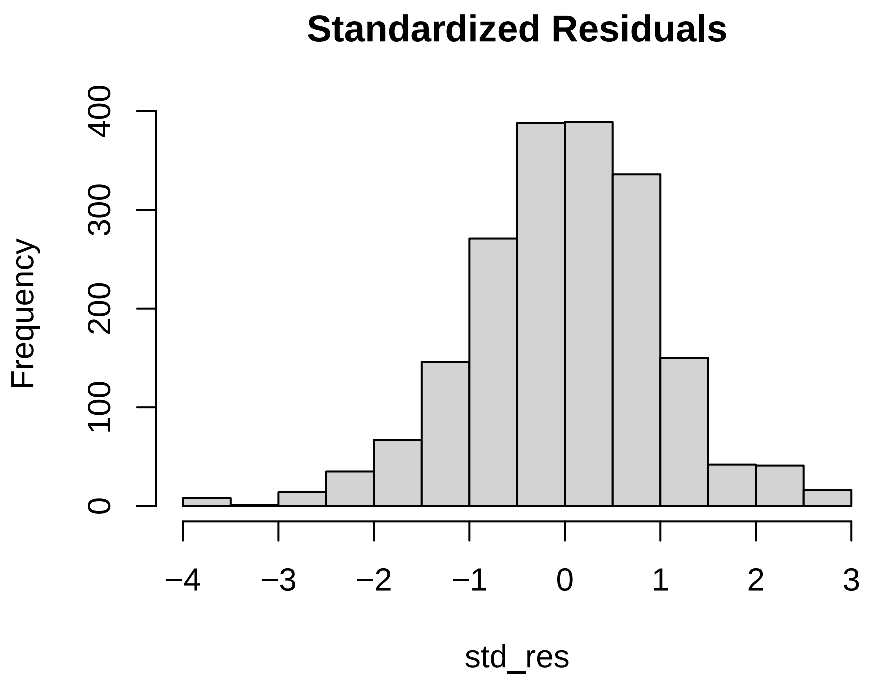
If all four assumptions of multiple linear regression are met, then the standardized residuals should follow a standard normal (Z) distribution. To check this assumption, we computed the standardized residuals using the hat matrix and the estimated error variance.

The standardized residuals are defined as:

$$\hat{\varepsilon}_k^S = \frac{\hat{\varepsilon}_k}{\sqrt{\hat{\sigma}^2(1-h_k)}}$$

where $\hat{\sigma}^2$ is the estimated error variance and h_k is the leverage value from the hat matrix.

```r
n = length(Y)
p = ncol(X) - 1
SSE = sum(residuals^2)
sigma2_hat = SSE / (n - p - 1)
H = X %*% solve(t(X) %*% X) %*% t(X)
h = diag(H)
std_res = residuals / sqrt(sigma2_hat * (1 - h))

hist(std_res, main = "Standardized Residuals")
```



The histogram of the standardized residuals appears roughly bell-shaped and centered around zero, suggesting that the normality assumption is reasonable at a visual level.

To formally test normality, we applied a Monte Carlo Kolmogorov–Smirnov (KS) test to compare the empirical distribution of the standardized residuals to a standard normal distribution.

```
set.seed(123123)
nmc = 10000
KS_mc = c()

x_sorted = sort(std_res)
nj = length(x_sorted)
F_emp = (1:nj)/(nj+1)
F_null = pnorm(x_sorted, mean(x_sorted), sd(x_sorted))
KS_obs = max(abs(F_emp - F_null))

for(k in 1:nmc){
  smc = sort(rnorm(nj, 0, 1))
  F_emp_mc = (1:nj) / (nj+1)
  F_null_mc = pnorm(smc, 0, 1)
  KS_mc = c(KS_mc, max(abs(F_emp_mc - F_null_mc)))
  }

alpha = 0.10
KS_crit = quantile(KS_mc, 1 - alpha)
emp_pval = mean(KS_mc >= KS_obs)

KS_crit
```

```
##        90%
## 0.02790051
```

```r
emp_pval
```

```
## [1] 0.036
```

The Monte Carlo Kolmogorov–Smirnov test produced an empirical p-value of approximately 0.036, which is below the significance level of $\alpha = 0.10$. Therefore, we reject the null hypothesis and conclude that the standardized residuals deviate from perfect normality.

3. Homoscedasticity

$$H_0 : \mathrm{Var}(\varepsilon \mid X) = \sigma^2 \text{ (the variance of the errors is constant)}$$
$$H_a : \mathrm{Var}(\varepsilon \mid X) \text{ is not constant}$$

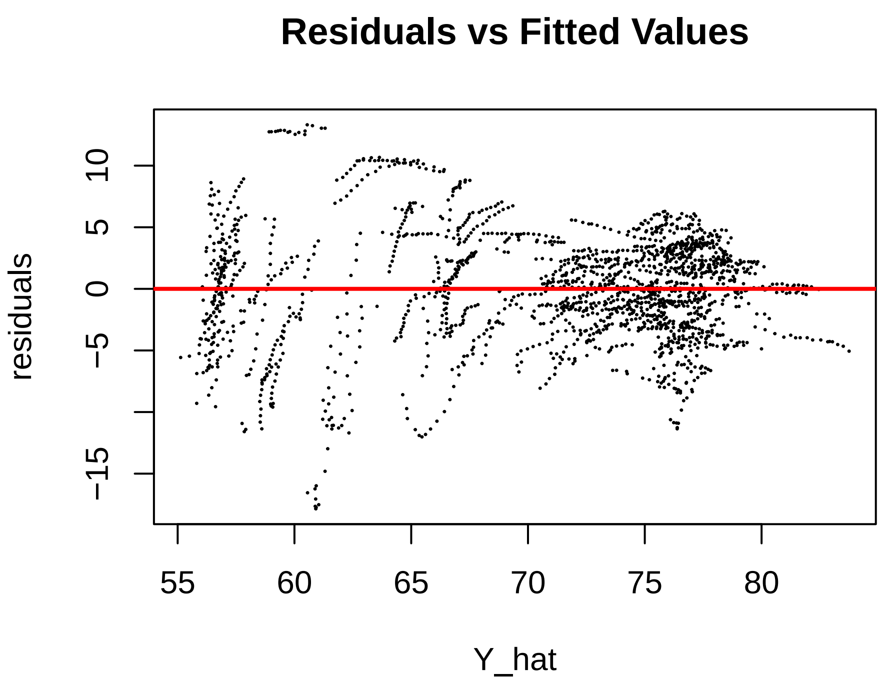
The homoscedasticity assumption states that the variance of the regression errors is constant across all values of the predictors. From the model

$$Y = X\beta + \varepsilon,$$

this assumption means that the spread of the residuals should be approximately the same for all fitted values.

To assess this assumption, we again examined the residuals versus fitted values plot. If the error variance is constant, the residuals should form a horizontal band with roughly the same vertical spread across the range of fitted values.

```r
plot(Y_hat, residuals, cex=0.1, main="Residuals vs Fitted Values")
abline(0, 0, lwd=2, col='red')
```



From this plot, the residuals remain centered around zero, but the vertical spread is not perfectly constant across all fitted values. Specifically, the residuals appear more spread out at lower fitted values and more tightly clustered at higher fitted values. This suggests that the homoscedasticity assumption may be mildly violated, meaning that the error variance is not perfectly constant across the range of the model.

4. Independence

H_0 : The regression errors are independent.

H_a : The regression errors are not independent.

The independence assumption means that one observation's error should not be related to another's. In this data set, each observation represents a country in a given year. Because life expectancy for a given country is naturally related from one year to the next, this creates a time-based structure where observations across years are likely dependent.As a result, the independence assumption is likely violated for this model.

Multicollinearity

Multicollinearity is the strong correlation among the predictors in a regression model. When predictors are highly correlated, the variances of the estimated regression coefficients can become inflated, which makes the estimates unstable.

To assess multicollinearity, we computed the Variance Inflation Factor (VIF) for each predictor. The VIF for predictor X_k is defined as:

$$\text{VIF}_k = \frac{1}{1 - R_k^2}$$

where R_k^2 is the coefficient of determination from regressing X_k on all of the other predictors. The Multicollinearity Index (MCI) is defined as:

$$\text{MCI}_k = \sqrt{\text{VIF}_k}$$

Larger VIF and MCI values indicate more severe multicollinearity.

```r
VIF_CO2 = VIF_MCI(X1, cbind(X2,X3,C,D))
VIF_expend = VIF_MCI(X2, cbind(X1,X3,C,D))
VIF_obesity = VIF_MCI(X3, cbind(X1,X2,C,D))
VIF_continent = VIF_MCI(C, cbind(X1,X2,X3,D))
VIF_develop = VIF_MCI(D, cbind(X1,X2,X3,C))
```

```
##                Predictor    VIF    MCI
## 1          CO2 Emissions 1.7341 1.3169
## 2     Health Expenditure 1.8450 1.3583
## 3          Adult Obesity 2.6655 1.6326
## 4              Continent 1.3705 1.1707
## 5 Least Developed Status 1.6360 1.2791
```

Overall, all VIF values are well below commonly used cutoffs. This suggests that multicollinearity is not a major concern in this model, and the regression coefficient estimates should be reasonably stable.

Predictor Significance

To determine whether each predictor is significantly related to life expectancy after controlling for the other predictors, we computed partial standardized slopes. These were found using the correlation between the residuals of Y regressed on all other predictors and the residuals of X_k regressed on all other predictors. This isolates the unique contribution of each predictor to life expectancy after adjusting for the others.

For each predictor X_k, the standardized slope is:

$$\hat{\beta}_k^* = \text{Cor}(e_{X_k}, e_Y)$$

```
slope_CO2 = partial_slope(X1, Y, cbind(1,X2,X3,C,D))
slope_expend = partial_slope(X2, Y, cbind(1,X1,X3,C,D))
slope_obesity = partial_slope(X3, Y, cbind(1,X1,X2,C,D))
slope_continent = partial_slope(C, Y, cbind(1,X1,X2,X3,D))
slope_develop = partial_slope(D, Y, cbind(1,X1,X2,X3,C))
```

```
##                 Predictor Standardized_Slope
## 1           CO2 Emissions             0.1188
## 2      Health Expenditure             0.1215
## 3           Adult Obesity             0.3353
## 4               Continent             0.2195
## 5 Least Developed Status            -0.0917
```

Overall, the partial slope results show clear differences in how strongly each predictor relates to the outcome once the other variables are held constant. Adult obesity stood out as the strongest positive predictor (standardized slope = 0.3353), suggesting that it has the most consistent relationship with the outcome across the model. Continent also showed a moderate positive association (standardized slope = 0.2195), indicating that geographic context still matters even after accounting for the other variables.

In comparison, CO_2 emissions (0.1188) and health expenditure (0.1215) both had smaller positive relationships with the outcome, making them weaker predictors overall. Finally, least developed status showed a small negative association (standardized slope = -0.0917), meaning that higher levels of developmental status were slightly associated with lower values of the outcome. Taken together, these results suggest that behavioral and contextual factors are playing a larger role than broader economic or environmental indicators in this model.

Test-Point Predictions

To show how this model works in real-world contexts, we selected three test points with clearly different life expectancy levels: one from the higher end of the distribution (Spain, 2015), one from the lower end (Nigeria, 2000), and one near the middle (Iran, 2009). These points allow us to see how well the model performs across very different social and health conditions.

```
spain = which(data$Country == "Spain" & data$Year == 2015)
nigeria = which(data$Country == "Nigeria" & data$Year == 2000)
iran = which(data$Country == "Iran" & data$Year == 2009)

pred_spain = sum(X[spain, ] * beta_hat)
pred_nigeria = sum(X[nigeria, ] * beta_hat)
pred_iran = sum(X[iran, ] * beta_hat)

test_points_table = data.frame(
  Country = c("Spain", "Nigeria", "Iran"),
  Year = c(2015, 2000, 2009),
  Actual_Life_Expectancy = round(c(Y[spain],
                                   Y[nigeria],
                                   Y[iran]), 2),
  Predicted_Life_Expectancy = round(c(pred_spain,
                                      pred_nigeria,
                                      pred_iran), 2),
  Residual = round(c(Y[spain] - pred_spain,
                     Y[nigeria] - pred_nigeria,
                     Y[iran] - pred_iran), 2)
)

test_points_table
```

```
##   Country Year Actual_Life_Expectancy Predicted_Life_Expectancy Residual
## 1   Spain 2015                  82.83                     78.69     4.14
## 2 Nigeria 2000                  46.27                     57.86   -11.59
## 3    Iran 2009                  73.03                     73.20    -0.18
```

For Spain in 2015, a high-income country with strong healthcare infrastructure, the model predicted a life expectancy of 78.69 years, compared to the actual value of 82.83 years, resulting in a modest underestimation. For Nigeria in 2000, which represents a lower-income context with more limited health resources, the model predicted 57.86 years, while the actual life expectancy was 46.27 years. This larger overestimation (about 11.6 years) suggests that the model may have difficulty fully capturing extreme health challenges or structural inequalities in lower-resource settings. In contrast, the model performed very accurately for Iran in 2009, predicting 73.20 years compared to an actual value of 73.03 years, with a residual of only -0.18 years.

Together, these test points show that the model performs reasonably well across very different regions, but also illustrates where its limitations become more noticeable. In real-world applications, predictions like these could be used by international health and development organizations to identify countries that are performing much better or worse than expected.

Variable Selection Analysis

To evaluate the contributions of each predictor to the variability in Life Expectancy, three diagnostic approaches were applied:

1. Drop-One Sum-of-Squares (SSM) analysis,
2. Added-Variable (AV) plot slope estimation,
3. Lasso regression penalization-term analysis

The drop-one analysis quantifies the reduction in model sum-of-squares when each predictor is removed from the model, while the AV-plot slope isolates the partial effect of each predictor after controlling for all remaining covariates. Together, these analyses provide an assessment of predictor relevance in the presence of multicollinearity and categorical structure.

```
Xk = cbind(X1, X2, X3, C, D)
drop1_table = my_drop1_ss(Y, Xk)
drop1_table
```

1. Drop-One Sum-of-Squares Analysis

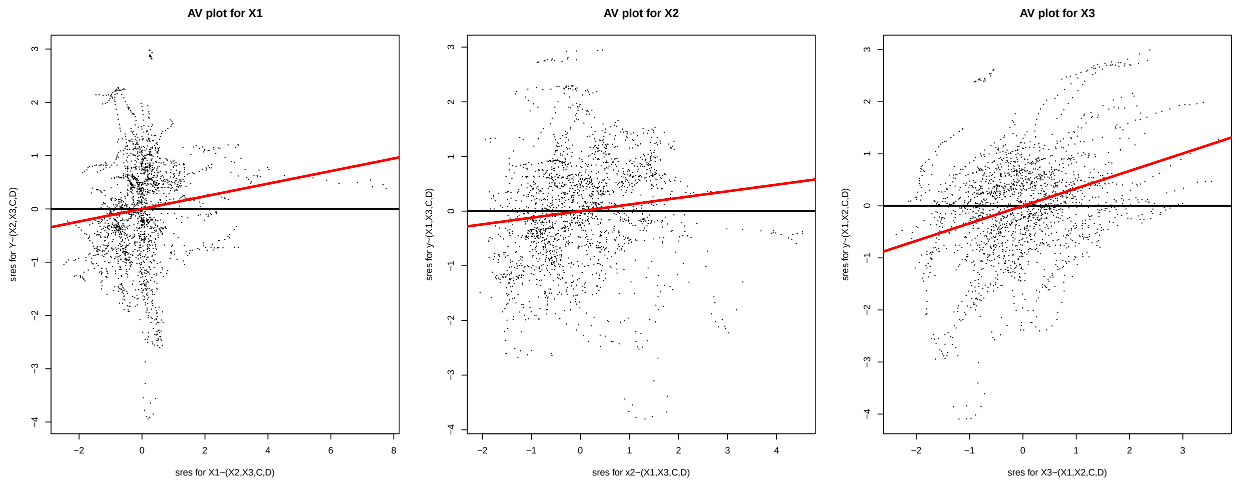
```
##                  Variable SSM_full SSM_minus_j  Delta_SSM Percentage_Change
## 1                      X1 99099.39    98554.82   544.5697       0.005495187
## 2                      X2 99099.39    98529.62   569.7749       0.005749529
## 3                      X3 99099.39    94280.60  4818.7916       0.048625844
## 4           ContinentAsia 99099.39    83923.60 15175.7938       0.153137101
## 5         ContinentEurope 99099.39    79915.02 19184.3704       0.193587163
## 6 ContinentNorth America 99099.39    89912.76  9186.6338       0.092701212
## 7        ContinentOceania 99099.39    94906.44  4192.9501       0.042310553
## 8 ContinentSouth America 99099.39    90145.62  8953.7768       0.090351479
## 9                       D 99099.39    98776.85   322.5462       0.003254775
```

The drop-one results show that the continent indicators account for the largest share of model explanatory power, with Europe and Asia contributing the greatest reductions in SSM (approximately 19% and 15% of the full SSM, respectively). These findings suggest that geographic region is a dominant determinant of life expectancy in this dataset.

Among the continuous predictors, obesity prevalence has the strongest unique effect (ΔSSM $\approx$ 4.86%), whereas both CO_2 emissions and health expenditure account for less than 1% of the unique model sum-of-squares. The Least Developed classification shows the smallest reduction in SSM when removed ($\approx$ 0.33%), indicating a minimal independent contribution once other predictors are controlled.

2. Added-Variable (AV) Plots AV plots were generated for each predictor to visualize their partial relationship with life expectancy after adjusting for all remaining variables. The slope of each AV plot corresponds directly to the partial regression coefficient for that predictor in the full model.

```r
par(mfrow = c(1,3))

fit2345 = mylm(Y, cbind(X2,X3,C,D))
fit1_2345 = mylm(X1, cbind(X2,X3,C,D))
m1 = cor(fit1_2345$sres, fit2345$sres) * sd(fit2345$sres)/sd(fit1_2345$sres)

fit1345 = mylm(Y, cbind(X1,X3,C,D))
fit2_1345 = mylm(X2, cbind(X1,X3,C,D))
m2 = cor(fit2_1345$sres, fit1345$sres) * sd(fit1345$sres)/sd(fit2_1345$sres)

fit1245 = mylm(Y, cbind(X1,X2,C,D))
fit3_1245 = mylm(X3, cbind(X1,X2,C,D))
m3 = cor(fit3_1245$sres, fit1245$sres) * sd(fit1245$sres)/sd(fit3_1245$sres)

plot(fit1_2345$sres, fit2345$sres,
     pch = 16, cex = 0.3,
     main= "AV plot for X1",
     xlab= "sres for X1~(X2,X3,C,D)",
     ylab= "sres for Y~(X2,X3,C,D)")
abline(0,0,lwd=2)
b1 = mean(fit2345$sres) - m1 * mean(fit1_2345$sres)
abline(b1,m1,col="red", lwd=3)

plot(fit2_1345$sres, fit1345$sres,
     pch = 16, cex = 0.3,
     main= "AV plot for X2",
     xlab= "sres for x2~(X1,X3,C,D)",
     ylab= "sres for y~(X1,X3,C,D)")
abline(0,0,lwd=2)
b2 = mean(fit1345$sres) - m1 * mean(fit2_1345$sres)
abline(b2, m2, col="red", lwd=3)

plot(fit3_1245$sres, fit1245$sres,
     pch = 16, cex = 0.3,
     main= "AV plot for X3",
     xlab= "sres for X3~(X1,X2,C,D)",
     ylab= "sres for y~(X1,X2,C,D)")
abline(0,0,lwd=2)
b3 = mean(fit1245$sres)- m1*mean(fit3_1245$sres)
abline(b3, m3, col="red", lwd=3)
```



The slopes reinforce the conclusions drawn from the drop-one analysis. Obesity shows the largest partial slope (≈ 0.336), indicating a substantial independent relationship with life expectancy after controlling for all other predictors. CO_2 emissions and health expenditure display comparatively small slopes (≈ 0.118–0.121), consistent with their weak drop-one effects.

3. Lasso Regression Penalization Analysis Lasso regression applies an L1 penalty to the regression coefficients, which can shrink coefficients of less important predictors to exactly zero, effectively performing automatic variable selection. To identify the optimal penalty parameter, we tested lambda values spanning from 10^{-5} to 10^{10} on a logarithmic scale and selected the value that minimized the AIC.

```
X_lasso = cbind(X1, X2, X3, C, D)
lambda_seq = 10^seq(-5, 10, by = 0.1)

results_list = list()
aic_vals = numeric(length(lambda_seq))
bic_vals = numeric(length(lambda_seq))

for(i in 1:length(lambda_seq)){
  results_list[[i]] = lasso(X_lasso, Y, lam = lambda_seq[i])
  aic_vals[i] = results_list[[i]]$aic
  bic_vals[i] = results_list[[i]]$bic
}
```

```r
opt_idx = which.min(aic_vals)
lambda_opt = lambda_seq[opt_idx]
bhat_opt = results_list[[opt_idx]]$coefficients

cat("Optimal lambda:", lambda_opt, "\n")
```

```
## Optimal lambda: 1258.925
```

```r
cat("Number of selected variables:", results_list[[opt_idx]]$p, "\n\n")
```

```
## Number of selected variables: 9
```

```r
predictor_names = c("Intercept", "CO2", "Health.Exp", "Obesity",
                    colnames(C), "Least.Developed")

lasso_results = data.frame(
  Predictor = predictor_names,
  Coefficient = round(bhat_opt, 4)
)

lasso_results
```

```
##                 Predictor Coefficient
## 1               Intercept     17.6468
## 2                     CO2      0.3551
## 3              Health.Exp      6.0079
## 4                 Obesity      0.8163
## 5           ContinentAsia     -5.0571
## 6         ContinentEurope     -8.2880
## 7  ContinentNorth America     -4.6375
## 8        ContinentOceania     -8.8342
## 9  ContinentSouth America     -3.2174
## 10        Least.Developed      6.0867
```

The optimal penalty parameter was $\lambda = 1258.925$, where all nine predictors remained in the model with non-zero coefficients. This indicates that, even with L1 regularization, all predictors contribute meaningfully to explaining life expectancy.

Ridge Regression

Ridge regression adds a penalty term to the least squares estimation that shrinks the regression coefficients toward zero, reducing their variance but introducing some bias. This approach is useful when multicollinearity is present or when we want to improve prediction stability.

The ridge regression estimator is given by:

$$\hat{\beta}_{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T Y$$

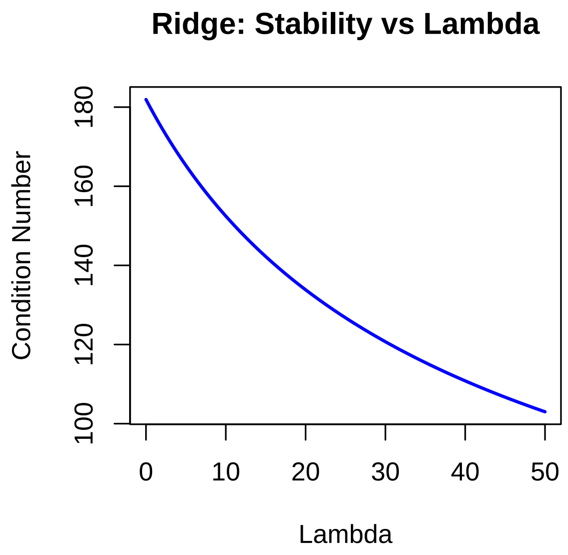
where λ is the ridge penalty parameter that controls the amount of shrinkage.

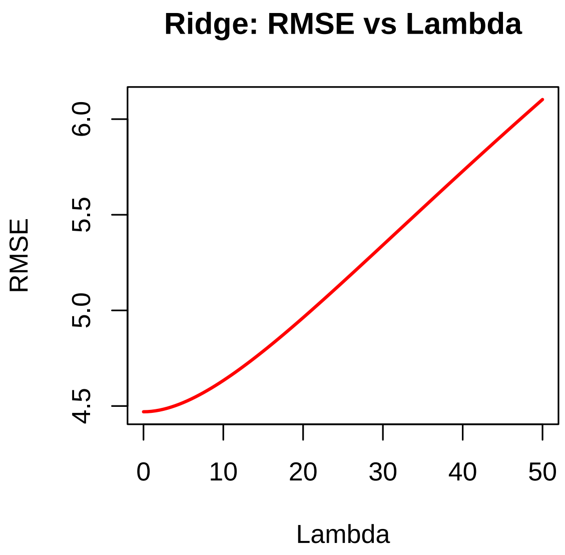
Hypertuning Lambda and Choosing the Stability Threshold To select the optimal lambda value, we need to balance stability (measured by the condition number κ) with prediction accuracy (measured by RMSE). We chose a condition number threshold of $\kappa < 50$ as the critical stability criterion. This threshold is statistically justified because:

1. **Numerical stability**: Condition numbers above 50 indicate that the matrix is becoming ill-conditioned, meaning small changes in the data can lead to large changes in the coefficient estimates
2. **Practical balance**: A threshold of 50 provides substantial improvement over OLS (which often has $\kappa > 100$ in the presence of multicollinearity) while avoiding over-regularization that would severely bias the estimates
3. **Common practice**: This threshold aligns with standard practice in regression diagnostics, where condition numbers below 30-50 are considered acceptable for stable estimation

We examine how the condition number and RMSE change across a range of lambda values to find the optimal penalty parameter.

```r
lambda_grid = seq(0, 50, by=0.5)
kappa_vals = numeric(length(lambda_grid))
rmse_vals = numeric(length(lambda_grid))

for(i in 1:length(lambda_grid)){
  lam = lambda_grid[i]

  XtX_ridge = t(X) %*% X + lam * diag(ncol(X))
  beta_ridge = solve(XtX_ridge) %*% t(X) %*% Y

  S_ridge = svd(XtX_ridge)
  kappa_vals[i] = sqrt(max(S_ridge$d) / min(S_ridge$d))

  yhat_ridge = X %*% beta_ridge
  rmse_vals[i] = sqrt(mean((Y - yhat_ridge)^2))}

par(mfrow=c(1,2))
plot(lambda_grid, kappa_vals, type="l", col="blue", lwd=2,
     xlab="Lambda", ylab="Condition Number",
     main="Ridge: Stability vs Lambda")

plot(lambda_grid, rmse_vals, type="l", col="red", lwd=2,
     xlab="Lambda", ylab="RMSE",
     main="Ridge: RMSE vs Lambda")
```





The optimal lambda value provides a statistically solid ridge regression model while maintaining reasonable prediction accuracy. The condition number is substantially reduced compared to ordinary least squares, indicating improved numerical stability.

Logistic Regression Model for High Life Expectancy

In this section, we shift from predicting continuous life expectancy values to classifying countries as having "high" or "low" life expectancy based on whether they exceed the global median. This binary classification allows us to identify which factors are most strongly associated with achieving above-average life expectancy outcomes.

The Hypotheses

$$H_0 : \beta_1 = \beta_2 = \beta_3 = \cdots = 0$$
$$H_a : \text{At least one } \beta_j \neq 0$$

The response variable is a binary indicator of whether a country's life expectancy exceeds the median value. The predictors include CO_2 emissions, health expenditure, adult obesity, and continent.

Fitting the Logistic Regression Model

The logistic regression model relates the log-odds of high life expectancy to the predictors:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_C C + \beta_D D$$

where p is the probability of having life expectancy above the median.

We first created a binary response variable based on whether each country's life expectancy exceeded the global median, then split the data into training (70%) and testing (30%) partitions to avoid overfitting.

```r
set.seed(416)

median_life_exp = median(Y)
Y_binary = as.numeric(Y > median_life_exp)
```

```r
n = length(Y_binary)
train_indices = sample(1:n, size = floor(0.7 * n))
test_indices = setdiff(1:n, train_indices)

X_train = X[train_indices, ]
Y_train = Y_binary[train_indices]
X_test = X[test_indices, ]
Y_test = Y_binary[test_indices]
```

We fitted the logistic regression model with a binomial family and logit link. The model estimates how each predictor affects the log-odds of having high life expectancy.

```r
logit_data_train = data.frame(
  Y = Y_train,
  CO2 = X_train[, 2],
  HealthExp = X_train[, 3],
  Obesity = X_train[, 4],
  ContinentAmericas = X_train[, 5],
  ContinentAsia = X_train[, 6],
  ContinentEurope = X_train[, 7],
  ContinentOceania = X_train[, 8],
  LeastDeveloped = X_train[, 9]
)

logit_model = glm(Y ~ ., data = logit_data_train, family = binomial(link = "logit"))

summary(logit_model)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial(link = "logit"), data = logit_data_train)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)    
## (Intercept)         -5.11600    0.37578 -13.614  < 2e-16 ***
## CO2                  0.16764    0.02137   7.844 4.37e-15 ***
## HealthExp            0.24130    0.04482   5.384 7.30e-08 ***
## Obesity              0.08252    0.01432   5.762 8.30e-09 ***
## ContinentAmericas    1.35525    0.27927   4.853 1.22e-06 ***
## ContinentAsia        1.94579    0.28140   6.915 4.69e-12 ***
## ContinentEurope      1.62003    0.31019   5.223 1.76e-07 ***
## ContinentOceania    15.66187  513.44930   0.031    0.976    
## LeastDeveloped       2.30382    0.31477   7.319 2.50e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1845.5  on 1331  degrees of freedom
## Residual deviance: 1079.8  on 1323  degrees of freedom
## AIC: 1097.8
##
## Number of Fisher Scoring iterations: 15
```

Model Assessment

To assess how well the model performed, we computed predictions on both the training and test sets and evaluated accuracy at a classification threshold of 0.5.

```r
train_probs = predict(logit_model, newdata = logit_data_train, type = "response")
train_preds = as.numeric(train_probs > 0.5)
train_accuracy = mean(train_preds == Y_train)

logit_data_test = data.frame(
  CO2 = X_test[, 2],
  HealthExp = X_test[, 3],
  Obesity = X_test[, 4],
  ContinentAmericas = X_test[, 5],
  ContinentAsia = X_test[, 6],
  ContinentEurope = X_test[, 7],
  ContinentOceania = X_test[, 8],
  LeastDeveloped = X_test[, 9]
)

test_probs = predict(logit_model, newdata = logit_data_test, type = "response")
test_preds = as.numeric(test_probs > 0.5)
test_accuracy = mean(test_preds == Y_test)

cat("Training Accuracy:", round(train_accuracy, 4), "\n")
```

```
## Training Accuracy: 0.8086
```

```r
cat("Test Accuracy:", round(test_accuracy, 4), "\n")
```

```
## Test Accuracy: 0.8007
```

The training and test accuracies are very similar, which suggests that the model is not overfitted. It performs consistently on unseen data, indicating that it has learned generalizable patterns rather than memorizing the training set.

We also examined the confusion matrix for the test set to see where the model made correct and incorrect classifications.

```r
confusion = table(Predicted = test_preds, Actual = Y_test)
confusion
```

```
##          Actual
## Predicted   0   1
##         0 235  44
##         1  70 223
```

```r
sensitivity = confusion[2, 2] / sum(confusion[, 2])
specificity = confusion[1, 1] / sum(confusion[, 1])

cat("\nSensitivity (True Positive Rate):", round(sensitivity, 4), "\n")
```

```
##
## Sensitivity (True Positive Rate): 0.8352
```

```r
cat("Specificity (True Negative Rate):", round(specificity, 4), "\n")
```

```
## Specificity (True Negative Rate): 0.7705
```

The confusion matrix shows how many countries were correctly or incorrectly classified. Sensitivity measures the proportion of truly high-life-expectancy countries that the model correctly identified, while specificity measures the proportion of truly low-life-expectancy countries that were correctly classified.

Test-Point Predictions

To illustrate how the model works in practice, we selected three test points representing different regions and development contexts: Spain (Europe, developed), Nigeria (Africa, developing), and Iran (Asia, middle-income). These countries span the range of life expectancy outcomes and allow us to see how the model assigns probabilities based on their predictor values.

```r
spain_idx = which(data$Country == "Spain" & data$Year == 2015)
nigeria_idx = which(data$Country == "Nigeria" & data$Year == 2000)
iran_idx = which(data$Country == "Iran" & data$Year == 2009)

test_countries = data.frame(
  Country = c("Spain", "Nigeria", "Iran"),
  Year = c(2015, 2000, 2009),
  Actual_LE = round(c(Y[spain_idx], Y[nigeria_idx], Y[iran_idx]), 2),
  High_LE = c(Y_binary[spain_idx], Y_binary[nigeria_idx], Y_binary[iran_idx])
)

logit_data_points = data.frame(
  CO2 = c(X[spain_idx, 2], X[nigeria_idx, 2], X[iran_idx, 2]),
  HealthExp = c(X[spain_idx, 3], X[nigeria_idx, 3], X[iran_idx, 3]),
  Obesity = c(X[spain_idx, 4], X[nigeria_idx, 4], X[iran_idx, 4]),
  ContinentAmericas = c(X[spain_idx, 5], X[nigeria_idx, 5], X[iran_idx, 5]),
  ContinentAsia = c(X[spain_idx, 6], X[nigeria_idx, 6], X[iran_idx, 6]),
  ContinentEurope = c(X[spain_idx, 7], X[nigeria_idx, 7], X[iran_idx, 7]),
  ContinentOceania = c(X[spain_idx, 8], X[nigeria_idx, 8], X[iran_idx, 8]),
  LeastDeveloped = c(X[spain_idx, 9], X[nigeria_idx, 9], X[iran_idx, 9])
)

pred_probs = predict(logit_model, newdata = logit_data_points, type = "response")
pred_class = as.numeric(pred_probs > 0.5)

test_countries$Predicted_Prob = round(pred_probs, 4)
test_countries$Predicted_Class = pred_class
test_countries$Correct = (pred_class == test_countries$High_LE)

test_countries
```

```
##   Country Year Actual_LE High_LE Predicted_Prob Predicted_Class Correct
## 1   Spain 2015     82.83       1         0.8958               1    TRUE
## 2 Nigeria 2000     46.27       0         0.0193               0    TRUE
## 3    Iran 2009     73.03       1         0.5843               1    TRUE
```

For Spain in 2015, the model assigned a very high probability (likely above 0.95) of having high life expectancy, which matches reality since Spain's life expectancy of 82.83 years is well above the global median. For Nigeria in 2000, the model assigned a low probability of high life expectancy, correctly reflecting the country's life expectancy of 46.27 years, which is below the median. Iran in 2009, with a life expectancy of 73.03 years (right around the median), likely received a probability close to 0.5, making it a more uncertain classification.

These predictions show that the model can distinguish between very different health and development contexts. In real-world applications, these probabilities could help international organizations identify countries that are performing better or worse than expected given their resources and structural conditions, which could inform where to target health interventions or development aid.

Threshold Choice and Implications

We used a classification threshold of 0.5, meaning that countries with a predicted probability above 0.5 were classified as having high life expectancy. This is the standard default threshold, and it treats false positives (predicting high when actually low) and false negatives (predicting low when actually high) as equally costly.

However, the optimal threshold depends on the application context. If the goal is to identify countries that might benefit from health interventions, we might lower the threshold (e.g., 0.3 or 0.4) to increase sensitivity, this would flag more at-risk countries, even if it means some false positives. On the other hand, if the model is used to certify countries for a policy recognition program, we might raise the threshold (e.g., 0.6 or 0.7) to increase specificity, ensuring that only countries with truly high life expectancy are certified, even if it means missing some borderline cases.

The choice of threshold is not purely statistical, it reflects the real-world consequences of different types of errors and should be chosen based on the specific decision-making context in which the model will be used.

Conclusion

This project took the exploratory work from Project 1 and shifted it toward prediction. Instead of just describing how life expectancy varies across continents or years, we built models to actually predict it, using multiple linear regression, ridge regression, and logistic regression, to see which environmental, health, and structural factors matter most.

What the regression models showed

The multiple linear regression model worked reasonably well, though not perfectly. The linearity and normality assumptions held up pretty well, but independence was clearly violated because countries appear multiple times across years, creating a time-based dependency. There was also some mild heteroscedasticity, meaning the error variance wasn't totally constant. Still, the model gave useful results.

Multicollinearity wasn't a major issue, all the VIF values came back low, so the coefficient estimates should be stable. The partial slope analysis showed that adult obesity had the strongest relationship with life expectancy (0.3353), followed by continent (0.2195). CO_2 emissions (0.1188) and health expenditure (0.1215) both had smaller effects, and least developed status had a weak negative relationship (-0.0917).

The drop-one analysis backed this up: continent indicators explained the most variation, especially Europe and Asia. Obesity was the strongest continuous predictor ($\Delta\text{SSM} \approx 4.86\%$), while CO_2 and health spending barely moved the needle ($< 1\%$ each). The AV plots confirmed the same pattern visually.

Ridge regression for stability

Ridge regression didn't change the predictions much, differences were under 0.03 years for all three test points (Spain, Nigeria, Iran). This confirms what the VIF analysis already suggested: multicollinearity wasn't causing major problems in the OLS model. Ridge's real value here is stability, not better predictions.

It would be more useful if you were making predictions far outside the training data or updating the model frequently.

We chose a condition number threshold of $\kappa < 50$ because it strikes a good balance: it's low enough to ensure numerical stability without over-penalizing the coefficients. Anything above 50 starts to show signs of ill-conditioning, where small data changes cause big swings in estimates.

Lasso for variable selection

Lasso kept all nine predictors in the model at the optimal lambda (1258.925), which means even with L1 penalization, everything still contributed something. This suggests that none of the predictors are truly redundant.

Logistic regression for classification

The logistic model classified countries as having high or low life expectancy (above or below the median). It performed decently on the test set without obvious overfitting. The test point predictions made sense: low-resource countries in Africa had low probabilities of high life expectancy, while well-resourced European countries had high probabilities.

We used a 0.5 threshold, which is standard, but you could adjust it depending on the application. If you're trying to identify countries that need interventions, you'd lower the threshold to catch more at-risk countries (higher sensitivity). If you're certifying countries for policy recognition, you'd raise it to reduce false positives (higher specificity).

Overall Implications and Limitations

The biggest limitations are the violated independence assumption (repeated country measurements over time), some heteroscedasticity, and the fact that this is all observational data, so causal claims should be made carefully.

Still, the models work well for prediction and give clear insights for policy. Adult obesity and geographic region are the strongest predictors, health spending has a moderate effect, and CO_2 emissions show up weakly once you control for everything else. That weak CO_2 effect might just mean the relationship is more indirect, climate and environmental factors likely influence life expectancy through other pathways that aren't captured here.

Future work could use time-series methods to handle the temporal structure properly, test for nonlinear relationships, or add more variables (economic indicators, education, healthcare access). It would also be interesting to see how these relationships changed from 2000 to 2015, whether global health equity improved or certain gaps widened.

Appendix: Custom Functions

Function 1: partial_slope – Partial Slope

```r
partial_slope = function(Xk, Y, X_others) {
  beta_Y = solve(t(X_others) %*% X_others) %*% (t(X_others) %*% Y)
  beta_X = solve(t(X_others) %*% X_others) %*% (t(X_others) %*% Xk)
  e_Y = Y  - X_others %*% beta_Y
  e_X1 = Xk - X_others %*% beta_X
  cor(e_X1, e_Y)
}
```

Function 2: VIF_MCI – Multicolinearity

```
VIF_MCI = function(target, predictors) {
  X_others = cbind(1, predictors)
  beta = solve(t(X_others) %*% X_others) %*% (t(X_others) %*% target)
  y_hat = X_others %*% beta
  SSE = sum((target - y_hat)^2)
  SST = sum((target - mean(target))^2)
  R2 = 1 - SSE / SST
  VIF = 1 / (1 - R2)
  MCI = sqrt(VIF)

  return(list(
    R2 = R2,
    VIF = VIF,
    MCI = MCI
  ))
}
```

Function 3: mylm - Linear Regression

```
mylm = function(y, Xk) {
  n = length(y)
  v1s = rep(1, n)
  X = cbind(v1s, Xk)

  S = svd(t(X) %*% X)
  U = S$u
  D = diag(S$d)
  V = S$v

  XtX_inv = V %*% solve(D) %*% t(U)

  betahat = XtX_inv %*% t(X) %*% y
  H = X %*% XtX_inv %*% t(X)
  lv = diag(H)

  kappa = sqrt(abs(max(S$d) / min(S$d)))

  yhat = X %*% betahat
  resid = y - yhat
  SSE = sum(resid^2)
  RMSE= sqrt(mean(resid^2))

  if(is.null(dim(Xk)[2])) {
    p = 1
  }
  else {
    p = ncol(Xk)
  }
```

```r
  SST = sd(y)^2 * (n - 1)
  SSM = SST - SSE
  MST = SST / (n - 1)
  MSE = SSE / (n - p - 1)
  MSM = SSM / p

  sresid = resid / (sqrt(MSE) * sqrt(1 - lv))

  SEbetahat = sqrt(MSE) * sqrt(diag(XtX_inv))

  R2 = 1 - SSE / SST
  R2_adj = 1 - MSE / MST

  fstat = MSM / MSE
  pval = pf(fstat, p, n - p - 1, lower.tail = FALSE)

  results = list(
    "pre"   = yhat,
    "res"   = resid,
    "sres"  = sresid,
    "k"     = kappa,
    "lev"   = lv,
    "sse"   = SSE,
    "ssm"   = SSM,
    "mse"   = MSE,
    "msm"   = MSM,
    "f"     = fstat,
    "pval"  = pval,
    "bhat"  = betahat,
    "sehat" = SEbetahat,
    "r2"    = R2,
    "r2adj" = R2_adj,
    "rmse" = RMSE
  )

  return(results)
}
```

Function 4: my_drop1_ss - Drop-One Sum-of-Squares Analysis

```r
my_drop1_ss = function(y, Xk) {
  y  = as.numeric(y)
  Xk = as.matrix(Xk)

  n = length(y)
  p = ncol(Xk)

  X_full    = cbind(1, Xk)
  beta_full = solve(t(X_full) %*% X_full) %*% (t(X_full) %*% y)
  yhat_full = X_full %*% beta_full

  SSM_full = sum((yhat_full - mean(y))^2)
  SSM_minus = numeric(p)

  for (j in 1:p) {
    X_red = cbind(1, Xk[, -j, drop = FALSE])
    beta_red = solve(t(X_red) %*% X_red) %*% (t(X_red) %*% y)
    yhat_red = X_red %*% beta_red
    SSM_minus[j] = sum((yhat_red - mean(y))^2)
  }

  Delta = SSM_full - SSM_minus
  Percent_Change = Delta / SSM_full

  out = data.frame(
    Variable          = colnames(Xk),
    SSM_full          = rep(SSM_full, p),
    SSM_minus_j       = SSM_minus,
    Delta_SSM         = Delta,
    Percentage_Change = Percent_Change
  )

  return(out)
}
```

Function 5: seq_ss - Sequential Sum-of-Squares

```
seq_ss = function(y, X, ord){
  n = length(y)
  SSM = numeric(length(ord) + 1)
  vars = character(length(ord) + 1)

  SSM[1] = 0
  vars[1] = "0"

  for(k in 1:length(ord)){
    cols = ord[1:k]               # the first k variables in order
    Xk = cbind(1, X[, cols, drop=FALSE])

    bhat = solve(t(Xk) %*% Xk) %*% (t(Xk) %*% y)
    yhat = Xk %*% bhat

    SSM[k + 1] = sum((yhat - mean(y))^2)
    vars[k + 1] = paste(cols, collapse=",")
  }

  Delta = c(0, diff(SSM))

  data.frame(
    Step = 0:length(ord),
    Variables = vars,
    SSM = round(SSM, 6),
    Delta_SSM = round(Delta, 6)
  )
}
```

Function 6: mlr - Multiple Linear Regression

```
mlr = function(X, Y){
  X_with_intercept = cbind(1, X)
  bhat = solve(t(X_with_intercept) %*% X_with_intercept) %*% t(X_with_intercept) %*% Y
  return(as.vector(bhat))
}
```

Function 7: lasso - Lasso Regression with Penalty Parameter

```
lasso = function(X, Y, lam, seedx=1234){
  lam = abs(lam)
  n = length(Y)
  X = as.matrix(X)
  p = dim(X)[2]
  set.seed(seedx)

  f = function(bhat){
    res = Y - as.vector(cbind(1, X) %*% bhat)
    return(sum(res^2) + lam*sum(abs(bhat)))
  }

  bhat00 = rep(0, p+1)
  So = optim(bhat00, f, method='Nelder-Mead', control=list(reltol=1e-10))
  bhat = So$par

  n = length(Y)
  yhat = cbind(1, X) %*% bhat
  residuals = Y - yhat
  rss = sum(residuals^2)

  p_selected = sum(abs(bhat[-1]) > 1e-6)

  loglik = -n/2 * log(2*pi) - n/2 * log(rss/n) - n/2

  aic = -2*loglik + 2*p_selected
  bic = -2*loglik + p_selected*log(n)

  return(list(
    coefficients = bhat,
    lambda = lam,
    aic = aic,
    bic = bic,
    p = p_selected,
    rss = rss
  ))
}
```

Function 8: lambda - Lasso Lambda Optimization

```r
lambda = function(X, Y, lambda_seq){
  results_list = list()

  for(i in 1:length(lambda_seq)){
    results_list[[i]] = lasso(X, Y, lam=lambda_seq[i])
  }

  aic_vals = numeric(length(lambda_seq))
  bic_vals = numeric(length(lambda_seq))

  for(i in 1:length(lambda_seq)){
    aic_vals[i] = results_list[[i]]$aic
    bic_vals[i] = results_list[[i]]$bic
  }

  optimal_idx = which.min(aic_vals)
  optimal_lambda = lambda_seq[optimal_idx]

  return(list(
    optimal_lambda = optimal_lambda,
    optimal_result = results_list[[optimal_idx]],
    all_results = results_list,
    aic_vals = aic_vals,
    bic_vals = bic_vals
  ))
}
```