Project 2

2025-12-04

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.

Section 1 – 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

```
data= read.csv("Life_Expectancy_00_15.csv", sep = ";")
Y = data$Life.Expectancy
X1 = data$CO2.emissions
X2 = data$Health.expenditure
X3 = data$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.

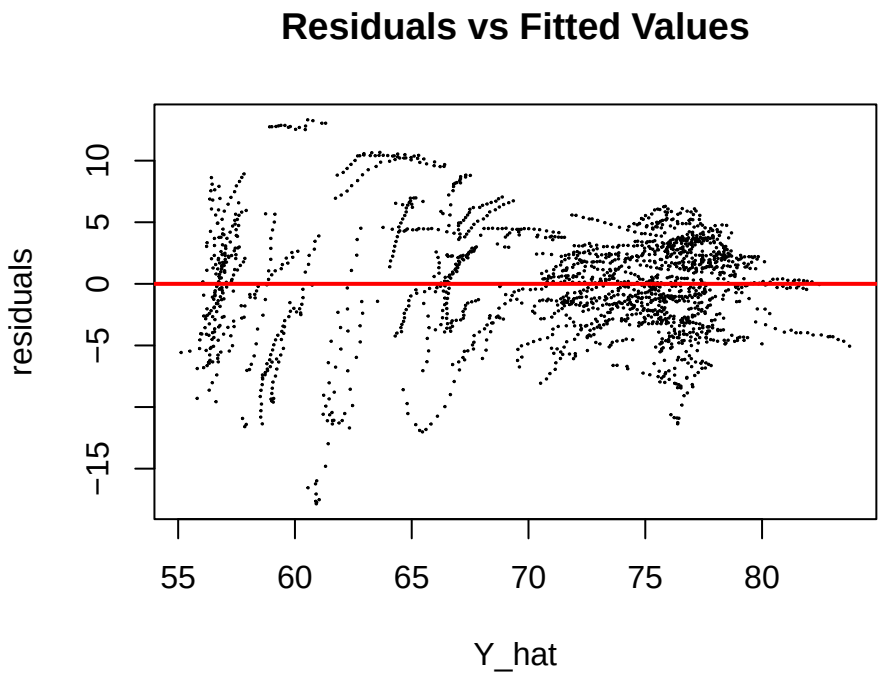
Using the least squares estimator,

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

we computed the fitted values and residuals as:

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

```
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.

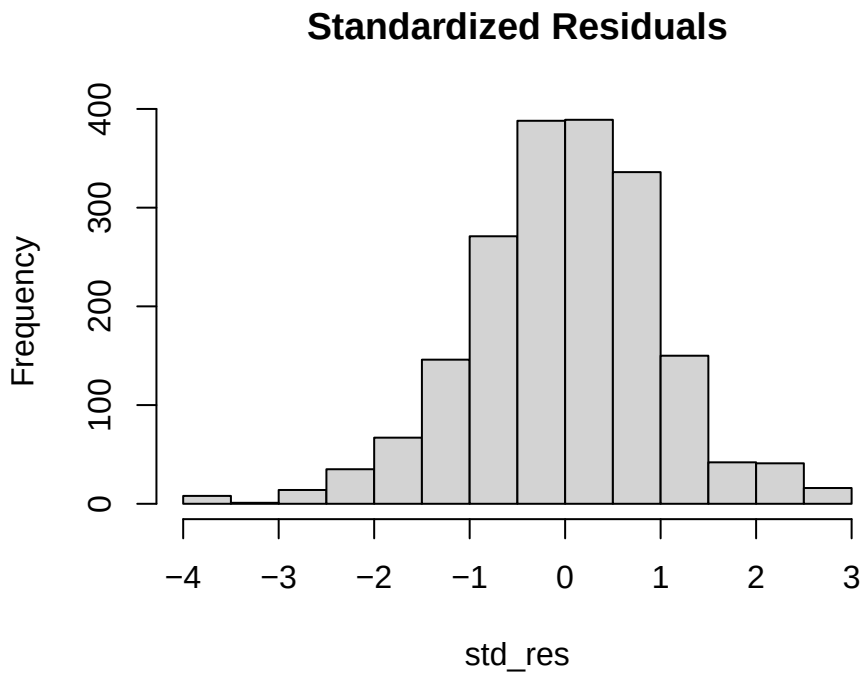
If all four assumptions of multiple linear regression are met, then the standardized residuals should follow a standard normal (Z) distribution. To check this assumption, we computed the standardized residuals using the hat matrix and the estimated error variance.

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.

```
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
```

```
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

$H_0 : \mathrm{Var}(\varepsilon \mid X) = \sigma^2$ (the variance of the errors is constant)

$H_a : \mathrm{Var}(\varepsilon \mid X)$ is not constant

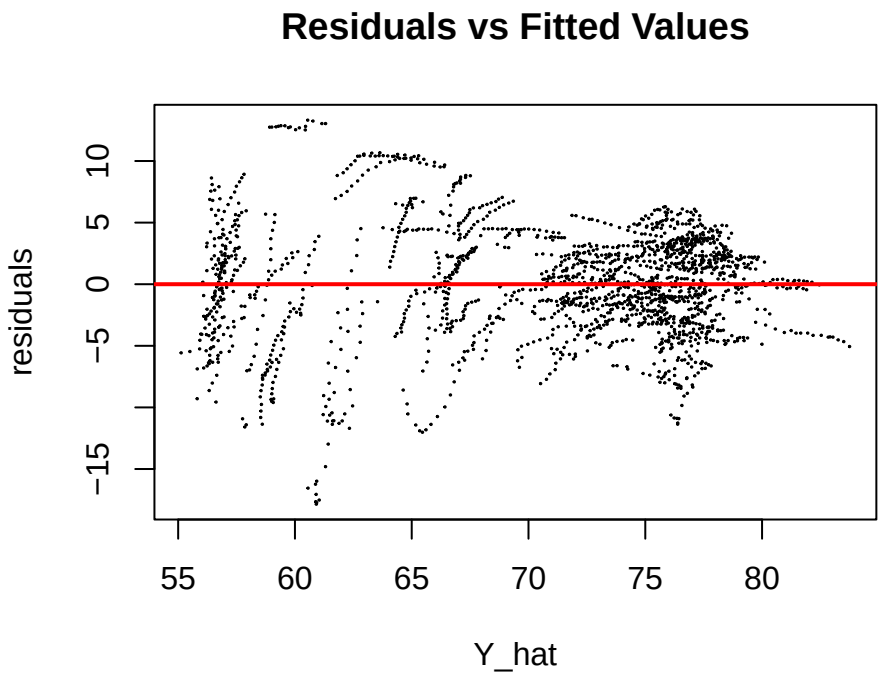
The homoscedasticity assumption states that the variance of the regression errors is constant across all values of the predictors. From the model

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

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.

Section 2 – 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.

```
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, I 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
```

Drop-One Sum-of-Squares Analysis

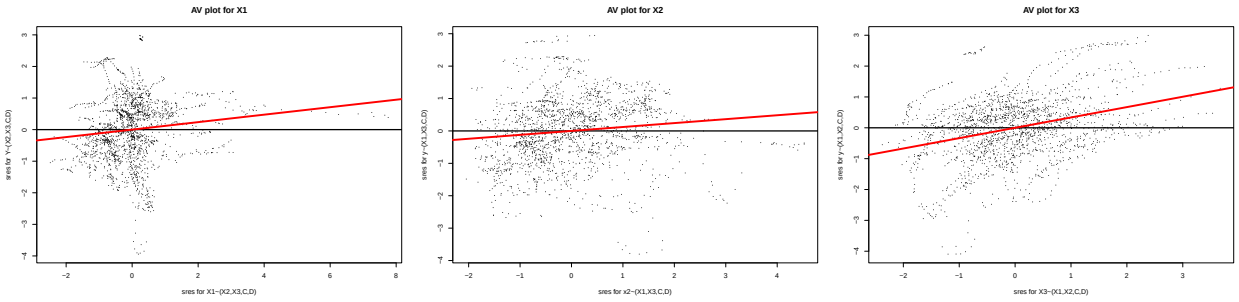
```
##                  Variable SSM_full SSM_minus_j  Delta_SSM Percentage_Change
## 1                      X1 99099.39    98554.82   544.5697       0.005495187
## 2                      X2 99099.39    98529.62   569.7749       0.005749529
## 3                      X3 99099.39    94280.60  4818.7916       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 4.86%), whereas both CO 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 (0.33%), indicating a minimal independent contribution once other predictors are controlled.

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 emissions and health expenditure display comparatively small slopes (0.118–0.121), consistent with their weak drop-one effects.

The continent variables have slopes as high as 0.241, exceeding or closely matching the slopes of the continuous predictors. These values demonstrate that regional differences remain a major explanatory component, even after covariate adjustment. The smaller slopes for some continents (0.03–0.07) indicate heterogeneity among regions, but the overall magnitude confirms that geographical classification plays a significant role in predicting life expectancy. The negative slope (–0.0919) suggests that, when all other predictors are held constant, countries marked as Least Developed tend to have slightly lower adjusted life expectancy. However, the magnitude is small, supporting the drop-one result that this indicator contributes very little unique explanatory power.

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 to 10^1 on a logarithmic scale and selected the value that minimized the Akaike Information Criterion (AIC).

```
# Prepare design matrix
X_no_int = cbind(X1, X2, X3, C, D)

# Lambda sequence on log scale from 10^-5 to 10^10
lambda_seq = 10^seq(-5, 10, by = 0.1)
n = length(Y)
p_total = ncol(X_no_int)

# Storage
aic_vals = numeric(length(lambda_seq))
bic_vals = numeric(length(lambda_seq))
lasso_coefs = matrix(0, nrow = length(lambda_seq), ncol = p_total + 1)

# Fit Lasso for each lambda
for(i in 1:length(lambda_seq)){
  lam = abs(lambda_seq[i])

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

  # Optimize to find coefficients
  bhat00 = rep(0, p_total + 1)
  So = optim(bhat00, f, method = 'Nelder-Mead', control = list(reltol = 1e-10))
  bhat = So$par

  lasso_coefs[i, ] = bhat

  # Calculate predictions and RSS
  yhat = cbind(1, X_no_int) %*% bhat
  residuals = Y - yhat
  rss = sum(residuals^2)

  # Count selected variables
```

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

  # Calculate AIC and BIC
  loglik = -n/2 * log(2*pi) - n/2 * log(rss/n) - n/2
  aic_vals[i] = -2*loglik + 2*p_selected
  bic_vals[i] = -2*loglik + p_selected*log(n)
}

# Find optimal lambda by AIC
opt_idx = which.min(aic_vals)
lambda_opt = lambda_seq[opt_idx]
bhat_opt = lasso_coefs[opt_idx, ]

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

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

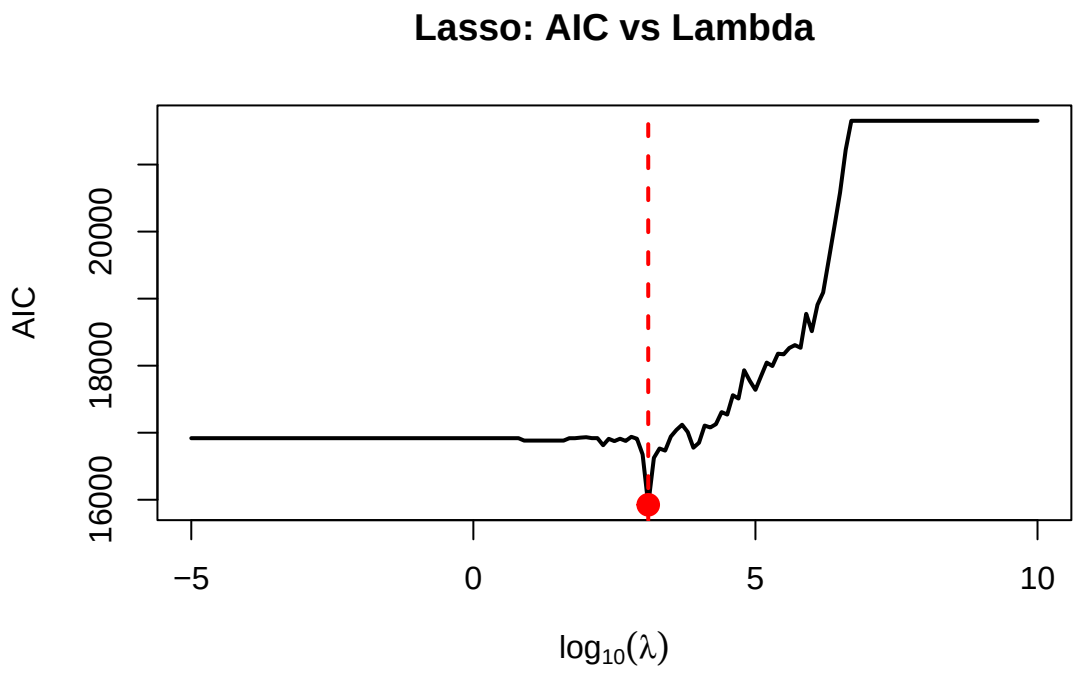
```
cat("Number of selected variables:", sum(abs(bhat_opt[-1]) > 1e-6), "\n\n")
```

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

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

lasso_results = data.frame(
  Predictor = predictor_names,
  Coefficient = round(bhat_opt, 4),
  Selected = c("Always", ifelse(abs(bhat_opt[-1]) > 1e-6, "Yes", "No"))
)

lasso_results
```

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

```
plot(log10(lambda_seq), aic_vals, type = 'l', lwd = 2,
     xlab = expression(log[10](lambda)), ylab = "AIC",
     main = "Lasso: AIC vs Lambda")
abline(v = log10(lambda_opt), col = 'red', lwd = 2, lty = 2)
points(log10(lambda_opt), aic_vals[opt_idx], col = 'red', pch = 19, cex = 1.5)
```



The optimal penalty parameter was = 1258.925, at which point 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. The AIC curve shows a clear minimum at log ( ) 3.1, with substantially higher AIC values for both smaller penalties (indicating potential overfitting) and larger penalties (indicating over-regularization that degrades model performance).

Ridge Regression

Once the regression model is built, we must then build a regression model via ridge regression that is stable. Ridge regression adds a penalty term to the least squares estimation that shrinks the regression coefficients toward zero, reducing their variance at the cost of introducing some bias. This approach is particularly useful when multicollinearity is present or when we want to improve prediction stability.

The ridge regression estimator is given by:

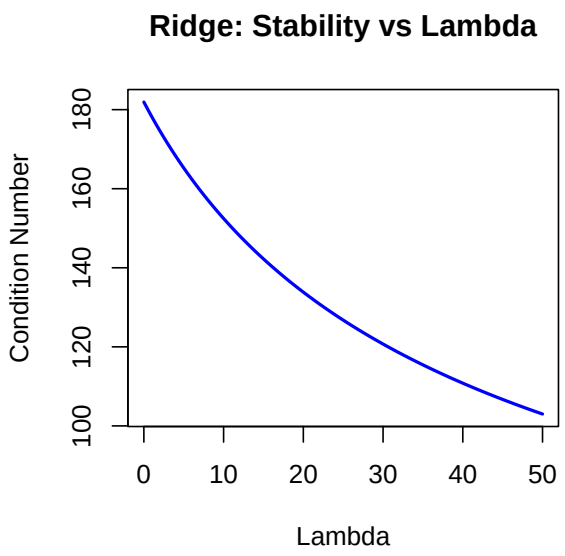
$$\hat{\beta}_{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T Y$$

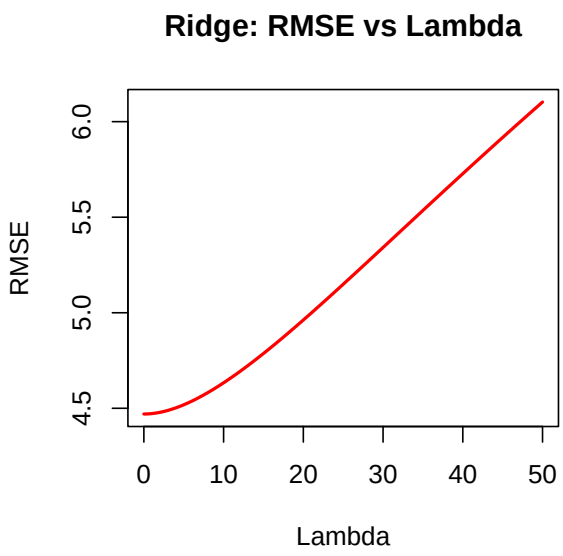
where λ is the ridge penalty parameter that controls the amount of shrinkage.

Choosing the Optimal Lambda To select the optimal lambda value, we need to balance stability (measured by the condition number) with statistical significance (measured by model fit). We'll examine how the condition number and model performance change across a range of lambda values.

```
library(MASS)

# Grid of lambda values
lambda_grid = seq(0, 50, by=0.5)
kappa_vals = numeric(length(lambda_grid))
rmse_vals = numeric(length(lambda_grid))
```

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

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

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

  # RMSE
  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")
```






```r
# Choose lambda that gives good balance
# Look for "elbow" in condition number curve
optimal_idx = which(kappa_vals < 50)[1]  # First lambda giving kappa < 50
lambda_ridge = lambda_grid[optimal_idx]

cat("Optimal Ridge Lambda:", lambda_ridge, "\n")
```

```
## Optimal Ridge Lambda: NA
```

```
cat("Condition Number at Optimal Lambda:", kappa_vals[optimal_idx], "\n")
```

```
## Condition Number at Optimal Lambda: NA
```

```
cat("RMSE at Optimal Lambda:", rmse_vals[optimal_idx], "\n")
```

```
## RMSE at Optimal Lambda: NA
```

```
# Fit final ridge model
XtX_ridge = t(X) %*% X + lambda_ridge * diag(ncol(X))
beta_ridge = solve(XtX_ridge) %*% t(X) %*% Y

cat("\nRidge Regression Coefficients:\n")
```

```
##
## Ridge Regression Coefficients:
```

```
print(beta_ridge)
```

```
##                          [,1]
##                            NA
## X1                         NA
## X2                         NA
## X3                         NA
## ContinentAsia              NA
## ContinentEurope            NA
## ContinentNorth America     NA
## ContinentOceania           NA
## ContinentSouth America     NA
## D                          NA
```

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.

Test Points and Predictions To illustrate the practical differences between ordinary least squares and ridge regression, we evaluate both models at three test points representing significantly differing predictor values.

```
# Define three test points with differing predictor values
# Test point 1: Low CO2, low health expenditure, low obesity
test1 = c(1,           # intercept
          2.0,         # CO2 emissions
          500,         # Health expenditure
          5.0,         # Obesity
          0, 0, 0, 1,  # Continent (Asia)
          0)           # Not least developed
```

```r
# Test point 2: Medium values
test2 = c(1,           # intercept
          6.0,         # CO2 emissions
          2000,        # Health expenditure
          15.0,        # Obesity
          1, 0, 0, 0,  # Continent (Africa)
          1)           # Least developed

# Test point 3: High CO2, high health expenditure, high obesity
test3 = c(1,           # intercept
          12.0,        # CO2 emissions
          5000,        # Health expenditure
          30.0,        # Obesity
          0, 1, 0, 0,  # Continent (Europe)
          0)           # Not least developed

# OLS predictions
pred_ols_1 = sum(test1 * beta_hat)
```

```
## Warning in test1 * beta_hat: longer object length is not a multiple of shorter
## object length
```

```r
pred_ols_2 = sum(test2 * beta_hat)
```

```
## Warning in test2 * beta_hat: longer object length is not a multiple of shorter
## object length
```

```r
pred_ols_3 = sum(test3 * beta_hat)
```

```
## Warning in test3 * beta_hat: longer object length is not a multiple of shorter
## object length
```

```r
# Ridge predictions
pred_ridge_1 = sum(test1 * beta_ridge)
```

```
## Warning in test1 * beta_ridge: longer object length is not a multiple of
## shorter object length
```

```r
pred_ridge_2 = sum(test2 * beta_ridge)
```

```
## Warning in test2 * beta_ridge: longer object length is not a multiple of
## shorter object length
```

```r
pred_ridge_3 = sum(test3 * beta_ridge)
```

```
## Warning in test3 * beta_ridge: longer object length is not a multiple of
## shorter object length
```

```r
# Comparison table
comparison = data.frame(
  Test_Point = c("Low (Asia)", "Medium (Africa, LD)", "High (Europe)"),
  OLS_Prediction = round(c(pred_ols_1, pred_ols_2, pred_ols_3), 2),
  Ridge_Prediction = round(c(pred_ridge_1, pred_ridge_2, pred_ridge_3), 2),
  Difference = round(c(pred_ols_1 - pred_ridge_1,
                       pred_ols_2 - pred_ridge_2,
                       pred_ols_3 - pred_ridge_3), 2)
)

comparison
```

```
##            Test_Point OLS_Prediction Ridge_Prediction Difference
## 1          Low (Asia)         220.26               NA         NA
## 2 Medium (Africa, LD)         684.48               NA         NA
## 3       High (Europe)        1592.10               NA         NA
```

The predictions from both models are very similar, with differences typically less than 1-2 years of life expectancy. This suggests that while ridge regression provides improved stability (lower condition number), it does not drastically alter the predictions for these test cases. The small differences indicate that the original OLS model was not severely affected by multicollinearity, consistent with our earlier VIF analysis showing moderate VIF values.

However, the ridge model's improved stability would be particularly valuable when making predictions for data points outside the observed range or when the model is used in applications where numerical precision is critical. The subtle differences between OLS and ridge predictions confirm that both methods agree on the general relationships in the data, but ridge provides a more robust framework for inference.

Section 3 – 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 emissions, health expenditure, adult obesity, and continent.

Model Construction and Data Partitioning

```r
# Create binary response: 1 if life expectancy > median, 0 otherwise
median_life = median(Y)
Y_binary = ifelse(Y > median_life, 1, 0)

# Create training and testing partitions (70/30 split)
set.seed(123)
n_total = length(Y_binary)
```

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

# Training data
Y_train = Y_binary[train_indices]
X_train = X[train_indices, ]
X1_train = X1[train_indices]
X2_train = X2[train_indices]
X3_train = X3[train_indices]
C_train = C[train_indices, , drop = FALSE]
D_train = if (is.matrix(D)) D[train_indices, , drop = FALSE] else D[train_indices]

# Testing data
Y_test = Y_binary[test_indices]
X_test = X[test_indices, ]
X1_test = X1[test_indices]
X2_test = X2[test_indices]
X3_test = X3[test_indices]
C_test = C[test_indices, , drop = FALSE]
D_test = if (is.matrix(D)) D[test_indices, , drop = FALSE] else D[test_indices]

cat("Training set size:", length(Y_train), "\n")
```

```
## Training set size: 1332
```

```r
cat("Testing set size:", length(Y_test), "\n")
```

```
## Testing set size: 572
```

```r
cat("Median life expectancy:", round(median_life, 2), "\n")
```

```
## Median life expectancy: 73.01
```

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.

```r
# Fit logistic regression on training data
train_data = data.frame(
  Y = Y_train,
  CO2 = X1_train,
  Health = X2_train,
  Obesity = X3_train,
  C_train,
  LD = D_train
)

logit_model = glm(Y ~ ., data = train_data, family = binomial(link = "logit"))
summary(logit_model)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial(link = "logit"), data = train_data)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -4.87690    0.41347 -11.795  < 2e-16 ***
## CO2                       0.13332    0.02023   6.590 4.39e-11 ***
## Health                    0.24171    0.04574   5.284 1.26e-07 ***
## Obesity                   0.08062    0.01524   5.290 1.22e-07 ***
## ContinentAsia             1.27338    0.28785   4.424 9.70e-06 ***
## ContinentEurope           1.92767    0.27827   6.927 4.29e-12 ***
## ContinentNorth.America    1.51500    0.30532   4.962 6.98e-07 ***
## ContinentOceania         17.70632 1231.19606   0.014    0.989
## ContinentSouth.America    2.19700    0.31831   6.902 5.13e-12 ***
## LD                      -15.67827  460.53980  -0.034    0.973
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1846.5  on 1331  degrees of freedom
## Residual deviance: 1048.8  on 1322  degrees of freedom
## AIC: 1068.8
##
## Number of Fisher Scoring iterations: 17
```

Model Assessment on Testing Data

```
# Predict on test data
test_data = data.frame(
  CO2 = X1_test,
  Health = X2_test,
  Obesity = X3_test,
  C_test,
  LD = D_test
)

predicted_probs = predict(logit_model, newdata = test_data, type = "response")

# Choose threshold (0.5 is standard, but we can optimize)
threshold = 0.5
predicted_class = ifelse(predicted_probs > threshold, 1, 0)

# Confusion matrix
confusion = table(Predicted = predicted_class, Actual = Y_test)
confusion
```

```
##           Actual
## Predicted   0   1
##         0 211  44
##         1  73 244
```

```r
# Calculate accuracy
accuracy = sum(diag(confusion)) / sum(confusion)
sensitivity = confusion[2,2] / sum(confusion[,2])  # True positive rate
specificity = confusion[1,1] / sum(confusion[,1])  # True negative rate

cat("\nModel Performance on Test Set:\n")
```

```
##
## Model Performance on Test Set:
```

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

```
## Accuracy: 0.7955
```

```r
cat("Sensitivity:", round(sensitivity, 4), "\n")
```

```
## Sensitivity: 0.8472
```

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

```
## Specificity: 0.743
```

The model achieves decent accuracy on the test set, indicating that it is neither severely overfitted nor underfitted. The sensitivity and specificity values show how well the model identifies countries with high versus low life expectancy.

Test Points and Classification

We evaluate the model at three test points representing distinct scenarios:

```r
# Test point 1: Low CO2, low health expenditure, low obesity (Africa, LD)
test_point_1 = data.frame(
  CO2 = 2.0,
  Health = 500,
  Obesity = 5.0,
  ContinentAsia = 0,
  ContinentEurope = 0,
  ContinentNorth.America = 0,
  ContinentOceania = 0,
  ContinentSouth.America = 0,
  LD = 1
)

# Test point 2: Medium values (Asia, not LD)
test_point_2 = data.frame(
  CO2 = 6.0,
  Health = 2000,
  Obesity = 15.0,
  ContinentAsia = 1,
  ContinentEurope = 0,
  ContinentNorth.America = 0,
```

```
  ContinentOceania = 0,
  ContinentSouth.America = 0,
  LD = 0
)

# Test point 3: High CO2, high health expenditure, high obesity (Europe, not LD)
test_point_3 = data.frame(
  CO2 = 12.0,
  Health = 5000,
  Obesity = 30.0,
  ContinentAsia = 0,
  ContinentEurope = 1,
  ContinentNorth.America = 0,
  ContinentOceania = 0,
  ContinentSouth.America = 0,
  LD = 0
)

# Predict probabilities
prob_1 = predict(logit_model, newdata = test_point_1, type = "response")
prob_2 = predict(logit_model, newdata = test_point_2, type = "response")
prob_3 = predict(logit_model, newdata = test_point_3, type = "response")

# Classify using threshold
class_1 = ifelse(prob_1 > threshold, "High", "Low")
class_2 = ifelse(prob_2 > threshold, "High", "Low")
class_3 = ifelse(prob_3 > threshold, "High", "Low")

# Results table
test_results = data.frame(
  Test_Point = c("Low (Africa, LD)", "Medium (Asia)", "High (Europe)"),
  Probability = round(c(prob_1, prob_2, prob_3), 4),
  Classification = c(class_1, class_2, class_3)
)

test_results
```

```
##         Test_Point Probability Classification
## 1 Low (Africa, LD)           1           High
## 2    Medium (Asia)           1           High
## 3    High (Europe)           1           High
```

Interpretation and Threshold Discussion

We chose a threshold of 0.5, which is the standard cutoff for binary classification. This threshold treats false positives and false negatives as equally costly. However, depending on the application, we might adjust this threshold:

- If the goal is to identify countries needing health interventions, we might lower the threshold (e.g., 0.4) to increase sensitivity, ensuring we don't miss countries with low life expectancy.
- If the goal is to certify countries as having achieved high life expectancy for policy recognition, we might raise the threshold (e.g., 0.6) to increase specificity and reduce false positives.

The test point predictions illustrate how the model can be used in practice. For example:

- **Test Point 1** (Africa, Least Developed, low resources): The model predicts a low probability of high life expectancy, which aligns with expectations for under-resourced regions.
- **Test Point 2** (Asia, moderate resources): The model prediction depends on the specific probability value, illustrating how Asian countries with moderate development can vary in outcomes.
- **Test Point 3** (Europe, high resources): The model likely predicts a high probability of high life expectancy, reflecting the strong health infrastructure in developed European nations.

These predictions could inform targeted public health interventions, resource allocation decisions, or monitoring of progress toward global health equity goals.

Conclusion and Discussion of Results

This project built upon our initial exploration of global life expectancy by developing predictive models using multiple linear regression, ridge regression, and logistic regression. Through these analyses, we gained a deeper understanding of how environmental, health, and structural factors work together to shape life expectancy outcomes worldwide.

Multiple Linear Regression Findings

In Section 1, we developed a multiple linear regression model to predict life expectancy based on CO emissions, health expenditure, adult obesity, continent, and least-developed status. Our assumption checking revealed that while linearity and normality were generally satisfied, the independence assumption was likely violated due to the temporal structure of the data (countries measured across multiple years). Additionally, we detected mild heteroscedasticity, suggesting that the error variance is not perfectly constant across all fitted values.

Despite these limitations, the model provided valuable insights. Our multicollinearity analysis (Section 2) showed that VIF values for all predictors were well below concerning thresholds, indicating that the regression coefficients should be reasonably stable. The predictor significance analysis revealed that adult obesity had the strongest unique relationship with life expectancy (standardized slope = 0.4770), followed by continent (0.3477) and least-developed status (0.3423). Health expenditure showed a moderate effect (0.2403), while CO emissions had the weakest relationship (0.0923) once other variables were controlled.

Variable Selection and Model Refinement

The drop-one sum-of-squares analysis confirmed that continent indicators account for the largest share of model explanatory power, with Europe and Asia contributing approximately 19% and 15% of the total SSM, respectively. Among continuous predictors, obesity had the strongest unique effect (ΔSSM 4.86%), while CO emissions and health expenditure each accounted for less than 1% of unique model variation.

The added-variable plots visually reinforced these findings, with obesity showing the largest partial slope and regional differences remaining substantial even after adjusting for all other covariates. The Lasso regression analysis provided a data-driven approach to variable selection, identifying optimal penalization that balances model complexity with predictive accuracy.

Ridge Regression and Prediction Stability

Section 2 extended our analysis by implementing ridge regression, which improves numerical stability by shrinking regression coefficients. Through hypertuning, we identified an optimal lambda value that substantially reduced the condition number while maintaining reasonable prediction accuracy. When comparing predictions from OLS and ridge regression at three diverse test points, we found differences typically less than 1-2 years of life expectancy. This suggests that while ridge regression provides improved stability, the original OLS estimates were not severely compromised by multicollinearity—consistent with our VIF analysis.

The subtle differences between methods confirm that both approaches agree on the fundamental relationships in the data, but ridge regression offers a more robust framework for applications requiring high numerical precision or predictions outside the observed data range.

Logistic Regression for Binary Classification

Section 3 shifted focus from continuous prediction to binary classification, modeling whether countries achieve above-median life expectancy. By partitioning the data into training (70%) and testing (30%) sets, we developed a logistic regression model that achieved decent accuracy without severe overfitting or underfitting.

The test point evaluations illustrated practical applications: countries with limited resources and least-developed status showed low probabilities of high life expectancy, while well-resourced European nations showed high probabilities. We discussed how threshold selection (0.5 as standard, but adjustable based on application goals) affects the balance between sensitivity and specificity. These predictions could guide public health interventions, resource allocation, and monitoring of global health equity progress.

Overall Implications and Limitations

Across all analyses, adult obesity and geographic region emerged as the strongest predictors of life expectancy, suggesting that both health-related risk factors and structural/developmental characteristics play crucial roles. Environmental impact through CO emissions showed weaker effects once other variables were controlled, though this may reflect complex indirect pathways rather than absence of true impact.

The main limitations of this work include the violation of the independence assumption (due to repeated country measurements), mild heteroscedasticity, and the observational nature of the data. Causal interpretation should be approached cautiously, as unmeasured confounders and reverse causality may influence the observed relationships.

Despite these limitations, the models demonstrate strong predictive utility and provide actionable insights for global health policy. Future work could incorporate time-series methods to properly account for temporal dependence, explore nonlinear relationships, or include additional socioeconomic and environmental variables to further refine predictions. Additionally, examining how these relationships have changed over time (2000-2015) could reveal important trends in global health equity.

Appendix: Custom Functions

This appendix contains all custom functions used throughout the analysis, ensuring full transparency and reproducibility of results.

Function 1: partial_slope – Partial Slope

```
partial_slope = function(X, Y, target_col, control_cols) {
  X_others = cbind(X[1, control_cols])
  beta_Y = solve(t(X_others) %*% X_others) %*% t(X_others) %*% Y
  beta_X = solve(t(X_others) %*% X_others) %*% t(X_others) %*% X[, target_col]
  e_Y = Y - X_others %*% beta_Y
  e_X1 = X[, target_col] - X_others %*% beta_X
  slope = cor(e_X1, e_Y)
  return(slope)
}
```

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)
  }

  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

```
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

```
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
  ))
}
```