PREDICTING BODY FAT PERCENTAGE: REGRESSION MODELS AND ANALYSIS

Pamela Claridy Georgia Institute of Technology
Atlanta, GA

February 4, 2024

1 Introduction

This study explores the complexities of predicting body fat percentage from various physiological measurements contained within the fat.csv dataset. I begin with the preparation of the dataset, progressing from the initial stages of reading and preprocessing the data to dividing it into distinct training and testing segments. With this base established, I venture into the exploratory data analysis, seeking to uncover the hidden patterns and relationships within the data. The goal is to uncover which machine learning models excel in accuracy when tasked with estimating body fat percentages. I have performed a detailed examination of several regression techniques, from the well-known linear regression to more complex methods like ridge regression, LASSO, principal component regression (PCR), and partial least squares (PLS).

2 Exploratory Data Analysis

The dataset explored in this study, fat.csv, encompasses a diverse array of physiological and body composition measurements from 252 individuals. It spans variables such as body fat percentage, body density, age, weight, height, and various circumferences (neck, chest, abdomen, hip, thigh, knee, ankle, biceps, forearm, and wrist). This dataset presents a wide demographic range, with ages from 22 to 81 years and weights from 118.5 to 363.15 pounds, illustrating the broad spectrum of body sizes and compositions covered.

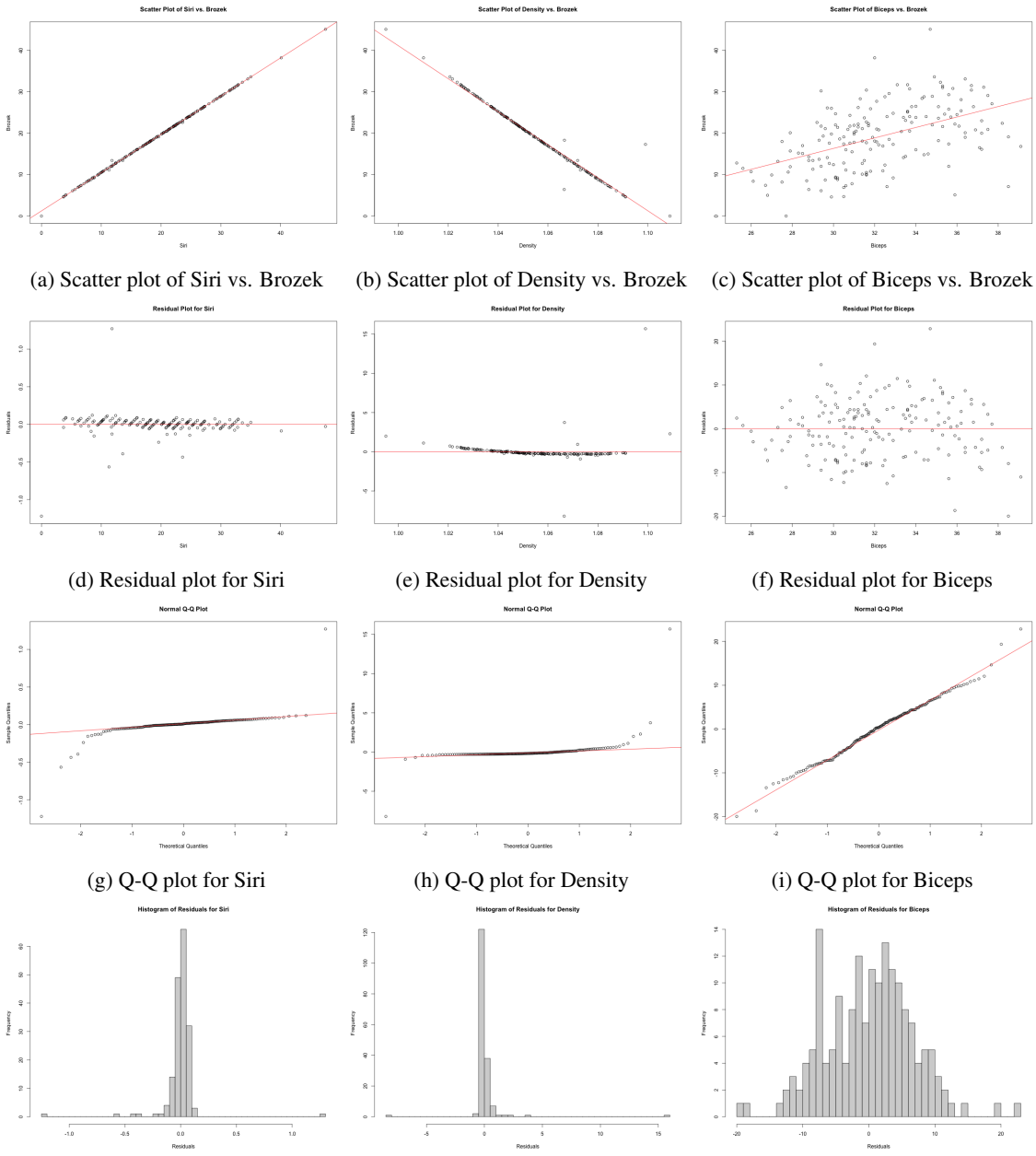
In the exploratory phase of my analysis, the initial linear model fitting reveals that 'Siri', 'Density', and 'Biceps' significantly influence 'Brozek' (body fat percentage). The model produced an almost perfect fit, with Multiple R-squared and Adjusted R-squared values nearing 0.9996. This level of precision displays the model's exceptional ability to capture the variability in body fat through these predictors.

The F-statistic and its low p-value further cement the model's credibility, affirming that the relationships identified are meaningful insights. This statistical validation provides a robust backbone to my analysis, instilling confidence in the predictive power of the model.

Reviewing scatter plots and residual analyses for 'Siri', 'Density', and 'Biceps' offered a unique perspective, enriching my understanding of these variables. This allowed me to assess the normal assumptions for linear regression models. These assumptions are linearity, homoscedasticity, normal distribution of errors, and independence of errors.

The scatter plots displays a strong linear relationship between 'Siri', 'Density', and 'Biceps' and 'Brozek', which is further substantiated by the red regression lines closely mirroring the underlying trends. The plots for 'Siri' and 'Density' show a particularly tight clustering of data points along these lines, suggesting that these variables are highly predictive of 'Brozek'. In contrast, the scatter plot for 'Biceps' exhibits a bit more variability, hinting at a slightly less strong, though still significant, relationship.

The residual plots for 'Siri' and 'Density' demonstrate a random spread of residuals around the zero line, implying that the linear models for these variables are well-fitted. The absence of discernible patterns suggests the residuals have constant variance (homoscedasticity) and the models capture the relationship without systemic errors. However, the residual plot for 'Biceps' reveals a bit more spread, indicating potential heteroscedasticity.



(a) Scatter plot of Siri vs. Brozek (b) Scatter plot of Density vs. Brozek (c) Scatter plot of Biceps vs. Brozek

(d) Residual plot for Siri (e) Residual plot for Density (f) Residual plot for Biceps

(g) Q-Q plot for Siri (h) Q-Q plot for Density (i) Q-Q plot for Biceps

(j) Histogram of residuals for Siri (k) Histogram of residuals for Density (l) Histogram of residuals for Biceps

Figure 1: Analysis of relationships between body fat percentage and significant predictors, and their residual effects.

The Q-Q plots present a slightly different aspect of the models. The plots for 'Siri' and 'Density' show residuals that largely follow the line of theoretical quantiles, which supports the assumption of normally distributed errors in the linear regression models. However, some deviation at the tails could indicate outliers or slight deviations from normality. The Q-Q plot for 'Biceps' is more concerning, as it displays a greater departure from normality in the tails, suggesting that the error distribution is not perfectly normal and may benefit from further investigation or transformation.

The histograms of residuals for 'Siri' and 'Density' display a nearly normal distribution, evidenced by their bell-shaped curves centered around zero. However, the histogram for 'Biceps' displays a more irregular distribution, with multiple peaks suggesting that the relationship might not be as well-defined or may require a transformation to meet the normality assumption.

In summary, the analysis of these plots reinforces the initial findings of statistical significance for 'Siri', 'Density', and 'Biceps'. It also offers insights into the distribution and variance of errors, which are critical for validating the assumptions of linear regression. These findings not only affirm the strong predictive capacity of these variables, but also guide potential model improvements, and further analysis.

3 Methods

The methodological framework of this study was meticulously designed to evaluate the predictive accuracy of various statistical models for estimating body fat percentage. Seven distinct models were developed and assessed, leveraging a combination of traditional regression techniques and advanced predictive algorithms.

3.1 Model Development

The dataset, comprising several physiological predictors, was subjected to rigorous analysis using the following statistical models:

3.1.1 Full Linear Model

The foundational model employed was a multiple linear regression that incorporated all available predictors. This comprehensive model served as a baseline for comparison against more nuanced approaches that followed.

3.1.2 Best Subset Linear Model

To refine the predictor selection, a best subset regression was conducted, identifying the most significant variables that contributed to predicting body fat percentage. This approach aimed to enhance model simplicity and interpretability.

3.1.3 Stepwise Regression Using AIC

A stepwise regression model was implemented to systematically add or remove predictors based on the Akaike Information Criterion, optimizing the balance between model complexity and information retention.

3.1.4 Ridge Regression

Ridge regression was applied as a regularization technique to address potential multicollinearity among predictors. This method adjusts the coefficients to prevent overfitting, improving the model's performance on unseen data.

3.1.5 LASSO Regression

LASSO (Least Absolute Shrinkage and Selection Operator) regression was another regularization technique utilized. It not only penalizes the magnitude of the coefficients but can also reduce the coefficients of less significant variables to zero, effectively performing variable selection.

3.1.6 Principal Component Regression (PCR)

PCR was employed to transform the predictors into a set of uncorrelated components before fitting the linear regression model. This approach helps to mitigate multicollinearity and focus the model on the most informative aspects of the data.

3.1.7 Partial Least Squares (PLS)

Partial least squares regression was used to find the multidimensional direction in the predictor space that explains the maximum multidimensional variance of the response variable. PLS is particularly useful when the predictors are highly collinear or when there are more predictors than observations.

3.2 Model Validation and Selection

Each model was subjected to a robust validation process:

- The Mean Squared Error (MSE) was computed for each model as the primary metric for assessing predictive accuracy.

- A Monte Carlo cross-validation approach was adopted, where the data were repeatedly split into training and testing sets to test the models' stability and reliability. The average MSE across these iterations was calculated to evaluate the overall performance.
- For models involving hyperparameter tuning, such as ridge and LASSO regression, a grid search was performed to find the optimal regularization strength.
- For PCR and PLS, the number of components was determined based on the model performance on cross-validation sets.

3.3 Reporting and Visualization

The results from the model assessments were systematically reported and visualized:

- MSE values for each model were tabulated for direct comparison.
- Average MSE values from the Monte Carlo simulations were also reported to provide insights into the models' generalizability.
- Graphical representations, including tables and plots, were used to visually summarize the models' performance metrics.

This methodological approach provided a comprehensive assessment of each model's capacity to predict body fat percentage accurately. The diverse array of models ensured that both the contribution of individual predictors and the collective predictive power of the predictors were thoroughly evaluated.

4 Results

In this section, I present the results of the analysis, which involves evaluating the Mean Squared Error (MSE) as the primary performance metric for various regression models in predicting the "brozek" body fat percentage. I first discuss the individual model performances based on MSE and subsequently analyze the average MSE results obtained after conducting Monte Carlo simulations.

4.1 Individual Model Performances Based on MSE

I initially examined the performance of several regression models in predicting body fat percentage using MSE as the evaluation metric. These models include:

4.1.1 Full Linear Model

The Full Linear Model, which includes all predictor variables, achieved an average MSE of approximately 29.50, serving as a baseline for comparison with other models.

4.1.2 Best Subset Model

The Best Subset Model, designed to identify the best combination of up to five predictor variables, demonstrated a low average MSE of approximately 0.057. This model's ability to select a subset of predictors while maintaining high predictive accuracy is evident.

4.1.3 Principal Component Regression (PCR)

PCR, utilizing principal components as predictors, yielded an average MSE of approximately 33.35. This technique provides dimensionality reduction benefits, but may sacrifice some predictive accuracy.

4.1.4 LASSO Regression

The LASSO Regression model produced an average MSE of about 0.064. LASSO's regularization technique helps in feature selection, resulting in a parsimonious model.

4.1.5 Ridge Regression

Ridge Regression, with its penalty term for large coefficients, resulted in an average MSE of approximately 0.690. This model can help mitigate multicollinearity, but tends to have higher MSE compared to LASSO.

4.1.6 Stepwise Model Selection Using AIC

The Stepwise Model, selected based on AIC, demonstrated an average MSE of about 0.114. It provides an automated approach for variable selection.

4.1.7 Partial Least Squares (PLS)

PLS, a dimensionality reduction technique, achieved an average MSE of approximately 2.28. It strikes a balance between feature reduction and predictive accuracy.

4.2 Average MSE Analysis After Monte Carlo Simulations

To further assess the robustness of these models, I conducted Monte Carlo simulations by repeatedly splitting the dataset into training and testing subsets. The average MSE values obtained from these simulations provide valuable insights into model stability and performance.

1. **Best Subset Model**: The Best Subset Model consistently exhibited the lowest average MSE, making it the top-performing model among those tested. This model effectively selected a subset of predictor variables while maintaining high predictive accuracy.
2. **LASSO Regression**: LASSO Regression also performed well, with a low average MSE, highlighting its ability to provide feature selection.
3. **Stepwise Model Selection Using AIC**: The Stepwise Model, selected based on AIC, showed competitive performance, making it a viable automated approach for variable selection.
4. **Partial Least Squares (PLS)**: Partial Least Squares regression yielded an average Mean Squared Error (MSE) of approximately 2.28, placing it in the middle range of model performance in terms of average MSE.
5. **Ridge Regression**: Ridge Regression, while useful for mitigating multicollinearity, had a higher average MSE compared to LASSO and other models.
6. **Principal Component Regression (PCR)**: PCR, which focuses on dimensionality reduction, had a higher average MSE, indicating a potential trade-off between feature reduction and predictive accuracy.
7. **Full Linear Model**: The Full Linear Model, including all predictor variables, served as the baseline, but had a relatively high average MSE compared to the other models.

5 Findings

Based on the MSE values, the Stepwise model has the lowest MSE, making it the best-performing model among the ones tested. Additionally, based on the Average MSE value for the Monte Carlo simulations, the Subset model has the lowest Average MSE, making it the best-performing model among the ones tested.

A Appendix: R Code

```
# Reading the dataset
fat <- read.table("fat.csv", sep = ",", header= TRUE)
head(fat)
summary(fat)

set.seed(142)

# Splitting the dataset into training and testing subsets
train <- sample(c(TRUE,FALSE), nrow(fat),
                replace=TRUE, prob=c(0.7,0.3))
fat1train  <- fat[train, ]
fat1test  <- fat[!train, ]

# Exploratory Data Analysis and initial linear model fitting
summary(fat1train)
model1 <- lm(brozek ~ ., data = fat1train)
```

```
summary(model1)

# The output of the model1 shows that siri, density, and biceps are statistically
# significant. Also, the mulitiple R-squared and adjusted r-squared are 0.9996
# indicating a good fit. The F-statistic and its p-value (<2.2e-16) indicate
# that the model is statistically significant (meaning at least some of the
# predictors are related to brozek).

# Brozek vs Siri
model_siri <- lm(brozek ~ siri, data=fat1train)

png("figures/scatterplot_siri_vs_brozek.png", width=800, height=600)
plot(fat1train$siri, fat1train$brozek, main="Scatter Plot of Siri vs. Brozek",
     xlab="Siri", ylab="Brozek")
abline(model_siri, col="red") #regression line
dev.off()

png("figures/residual_plot_siri.png", width=800, height=600)
plot(fat1train$siri, resid(model_siri), main="Residual Plot for Siri", xlab="Siri"
     , ylab="Residuals")
abline(h=0, col="red")
dev.off()

png("figures/histogram_of_residuals_siri.png", width=800, height=600)
hist(resid(model_siri), breaks=40, main="Histogram of Residuals for Siri",
     xlab="Residuals")
dev.off()

png("figures/qq_plot_siri.png", width=800, height=600)
qqnorm(resid(model_siri))
qqline(resid(model_siri), col="red")
dev.off()

# Brozek vs Density
model_density <- lm(brozek ~ density, data=fat1train)

png("figures/scatterplot_density_vs_brozek.png", width=800, height=600)
plot(fat1train$density, fat1train$brozek, main="Scatter Plot of Density vs. Brozek",
     xlab="Density", ylab="Brozek")
abline(model_density, col="red")
dev.off()

png("figures/residual_plot_density.png", width=800, height=600)
plot(fat1train$density, resid(model_density), main="Residual Plot for Density",
     xlab="Density", ylab="Residuals")
abline(h=0, col="red")
dev.off()

png("figures/histogram_of_residuals_density.png", width=800, height=600)
hist(resid(model_density), breaks=40, main="Histogram of Residuals for Density",
     xlab="Residuals")
dev.off()

png("figures/qq_plot_density.png", width=800, height=600)
qqnorm(resid(model_density))
qqline(resid(model_density), col="red")
dev.off()

# Brozek vs Biceps
```

```
model_biceps <- lm(brozek ~ biceps, data=fat1train)

png("figures/scatterplot_biceps_vs_brozek.png", width=800, height=600)
plot(fat1train$biceps, fat1train$brozek, main="Scatter Plot of Biceps vs. Brozek",
     xlab="Biceps", ylab="Brozek")
abline(model_biceps, col="red")
dev.off()

png("figures/residual_plot_biceps.png", width=800, height=600)
plot(fat1train$biceps, resid(model_biceps), main="Residual Plot for Biceps",
     xlab="Biceps", ylab="Residuals")
abline(h=0, col="red")
dev.off()

png("figures/histogram_of_residuals_biceps.png", width=800, height=600)
hist(resid(model_biceps), breaks=40, main="Histogram of Residuals for Biceps",
     xlab="Residuals")
dev.off()

png("figures/qq_plot_biceps.png", width=800, height=600)
qqnorm(resid(model_biceps))
qqline(resid(model_biceps), col="red")
dev.off()


# (i) Linear regression model with all predictors (Full Linear Model)
# Fit a linear regression model using 'brozek' as the response variable and all
# other variables in 'fat1train' as predictors
full.model <- lm(brozek ~ ., data = fat1train)

# Predict the 'brozek' values for the 'fat1test' dataset using the fitted linear
# regression model
predictions_lr <- predict(full.model, newdata=fat1test)

# Calculate the Mean Squared Error (MSE) by comparing the actual 'brozek' values
# in the test dataset to the predicted values
single_mse_lr <- mean((fat1test$brozek - predictions_lr)^2)

# Print MSE
print(single_mse_lr)

# Monte Carlo loop
# Initialize the storage for MSE values
B <- 100  # Number of repetitions
mse_lr <- numeric(B)  # To store MSE for each iteration

for (i in 1:B) {
  # Splitting the dataset into training and testing subsets
  train <- sample(c(TRUE,FALSE), nrow(fat),
                  replace=TRUE, prob=c(0.7,0.3))
  fat1train  <- fat[train, ]
  fat1test  <- fat[!train, ]

  # Fit the Linear Regression model
  full.model <- lm(brozek ~ ., data = fat1train)

  # Make predictions on the test set
  predictions_lr <- predict(full.model, newdata=fat1test)
```

```
  # Compute and store the MSE for this iteration
  mse_lr[i] <- mean((fat1test$brozek - predictions_lr)^2)
}

# Calculate the average MSE across all iterations
avg_mse_lr <- mean(mse_lr)

# Print the average MSE
print(avg_mse_lr)

# (ii) Linear regression with the best subset of k = 5 predictor variables (Best
# Subset Model)
library(leaps)

# Fit the best subset selection model to identify the best combination of predictors
subset.model <- regsubsets(brozek ~ ., data=fat1train, nvmax=5, nbest=1)
subset.summary <- summary(subset.model)

# Determine the best model based on adjusted R-squared
best_model_index <- which.max(subset.summary$adjr2)

# Extract names of the variables in the best model
best_model_vars <- names(which(coef(subset.model, id = best_model_index) != 0))

# Remove the intercept
best_model_vars <- best_model_vars[best_model_vars != "(Intercept)"]

# Construct the formula
formula_best <- reformulate(termlabels = best_model_vars, response = "brozek")

# Fit the linear model using the selected best subset of predictors and predict
model_best_subset <- lm(formula = formula_best, data = fat1train)

# Predict on test data and calculate MSE
predictions_subset <- predict(model_best_subset, newdata = fat1test)
actual_responses <- fat1test$brozek
single_mse_subset <- mean((actual_responses - predictions_subset)^2)

# Print the MSE
print(single_mse_subset)

# Monte Carlo loop
# Initialize storage for MSE values
B <- 100  # Number of Monte Carlo iterations
mse_best_subset <- numeric(B)  # To store MSE for each iteration

for (i in 1:B) {
  # Splitting the dataset into training and testing subsets
  train <- sample(c(TRUE,FALSE), nrow(fat),
                  replace=TRUE, prob=c(0.7,0.3))
  fat1train  <- fat[train, ]
  fat1test  <- fat[!train, ]

  # Fit the best subset selection model
  subset.model <- regsubsets(brozek ~ ., data = fat1train, nvmax=5, nbest=1)
  best_model_index <- which.max(summary(subset.model)$adjr2)

  # Extracting the names of variables in the best subset
  best_vars <- names(coef(subset.model, id = best_model_index))
```

```
  best_vars <- best_vars[best_vars != "(Intercept)"]  # Exclude intercept

  # Constructing the formula for the best subset model
  formula_best <- reformulate(termlabels = best_vars, response = "brozek")

  # Fit a linear model using the selected best subset of predictors
  model_best_subset <- lm(formula = formula_best, data = fat1train)

  # Predict on test data
  predictions_subset <- predict(model_best_subset, newdata = fat1test)

  # Calculate MSE on test data
  actual_responses <- fat1test$brozek
  mse_best_subset[i] <- mean((actual_responses - predictions_subset)^2)
}

# Calculate the average MSE across all iterations
avg_mse_best_subset <- mean(mse_best_subset)

# Print the average MSE
print(avg_mse_best_subset)

# (iii) Linear regression with variables (stepwise) selected using AIC (Stepwise
# Model Selection Using AIC)
# Initialize and fit a full linear model using all predictors in 'fat1train'
full.model <- lm(brozek ~ ., data = fat1train)

# Refine the full model by performing stepwise selection to optimize based on AIC,
# choosing the best combination of variables without verbose output
stepwise.model <- step(full.model, direction="both", trace=0)

# Use the optimized stepwise model to make predictions on the 'fat1test' dataset
predictions_stepwise <- predict(stepwise.model, newdata=fat1test)

# Calculate the Mean Squared Error (MSE) between the actual 'brozek' values and
# the predicted values from the stepwise model on the test data
single_mse_stepwise <- mean((fat1test$brozek - predictions_stepwise)^2)

# Print the MSE
print(single_mse_stepwise)

# Monte Carlo loop
# Initialize storage for MSE values
B <- 100  # Number of Monte Carlo iterations
mse_stepwise <- numeric(B)  # To store MSE for each iteration

for (i in 1:B) {
  # Splitting the dataset into training and testing subsets
  train <- sample(c(TRUE,FALSE), nrow(fat),
                  replace=TRUE, prob=c(0.7,0.3))
  fat1train  <- fat[train, ]
  fat1test  <- fat[!train, ]

  # Fit an initial full linear model on the training data
  full.model <- lm(brozek ~ ., data = fat1train)

  # Perform stepwise selection based on AIC
  stepwise.model <- step(full.model, direction = "both", trace = 0)
```

```
  # Predict on test data using the refined model
  predictions_stepwise <- predict(stepwise.model, newdata = fat1test)

  # Calculate MSE for this iteration
  mse_stepwise[i] <- mean((fat1test$brozek - predictions_stepwise)^2)
}

# Calculate the average MSE across all iterations
avg_mse_stepwise <- mean(mse_stepwise)

# Print the average MSE
print(avg_mse_stepwise)

# (iv) Ridge regression
library(MASS)
library(glmnet)
ridge.model <- lm.ridge(brozek ~ ., data=fat1train, lambda=seq(0,100,0.001))

# Prepare the data
x_matrix <- model.matrix(brozek ~ ., data=fat1train)[,-1]  # Exclude intercept
y_vector <- fat1train$brozek

# Prepare the predictor matrix for the test data
x_test_matrix <- model.matrix(brozek ~ ., data=fat1test)[,-1]  # Exclude intercept

# Check if the number of columns matches
#if (ncol(x_test_matrix) != ncol(x_matrix)) {
#  stop("test matrix does not match the training matrix")
#}

# Fit ridge regression model with cross-validation
cv.ridge <- cv.glmnet(x_matrix, y_vector, alpha=0, type.measure="mse")

# Extract the lambda that minimizes the cross-validated MSE
lambda_optimal <- cv.ridge$lambda.min

# Predict on test data
predictions_ridge <- predict(cv.ridge, s=lambda_optimal, newx=x_test_matrix)

# Calculate MSE on test data
actual_responses <- fat1test$brozek  # Ensure correctly aligned with predictions
single_mse_ridge <- mean((actual_responses - predictions_ridge)^2)

# Print the MSE
print(single_mse_ridge)

# Monte Carlo loop
# Initialize storage for MSE values
B <- 100  # Number of Monte Carlo iterations
mse_ridge <- numeric(B)  # To store MSE for each iteration

for (i in 1:B) {
  # Splitting the dataset into training and testing subsets
  train <- sample(c(TRUE,FALSE), nrow(fat),
                  replace=TRUE, prob=c(0.7,0.3))
  fat1train  <- fat[train, ]
  fat1test  <- fat[!train, ]

  # Prepare the data for Ridge Regression
```

```
  x_matrix <- model.matrix(brozek ~ ., data=fat1train)[,-1]  # Exclude intercept
  y_vector <- fat1train$brozek

  x_matrix_test <- model.matrix(brozek ~ ., data=fat1test)[,-1]  # Exclude
  # intercept for test data

  # Fit Ridge Regression model with cross-validation to find optimal lambda
  cv_ridge <- cv.glmnet(x_matrix, y_vector, alpha=0, type.measure="mse")

  # Extract the lambda that minimizes the cross-validated MSE
  lambda_optimal <- cv_ridge$lambda.min

  # Predict on test data using the optimal lambda
  predictions_ridge <- predict(cv_ridge, newx=x_matrix_test, s=lambda_optimal)

  # Calculate MSE on test data
  actual_responses <- fat1test$brozek
  mse_ridge[i] <- mean((actual_responses - predictions_ridge)^2)
}

# Calculate the average MSE across all iterations
avg_mse_ridge <- mean(mse_ridge)

# Print the average MSE
print(avg_mse_ridge)

# (v) LASSO
library(lars)
lasso.model <- lars(as.matrix(fat1train[, -which(names(fat1train) == "brozek")]),
                    fat1train$brozek, type="lasso")

# Fit LASSO model with cross-validation
cv.lasso <- cv.glmnet(x_matrix, y_vector, alpha=1, type.measure="mse")

# Extract the lambda that minimizes the cross-validated MSE
lambda_optimal_lasso <- cv.lasso$lambda.min

# Predict on test data using the optimal lambda
predictions_lasso <- predict(cv.lasso, s=lambda_optimal_lasso, newx=x_test_matrix)

# Since predictions_lasso is a matrix, convert it to a vector for MSE calculation
predictions_lasso_vector <- as.vector(predictions_lasso)

# Calculate MSE on test data for LASSO
single_mse_lasso <- mean((fat1test$brozek - predictions_lasso_vector)^2)

# Print the MSE
print(single_mse_lasso)

# Monte Carlo loop
# Initialize storage for MSE values
B <- 100  # Number of Monte Carlo iterations
mse_lasso <- numeric(B)  # To store MSE for each iteration

for (i in 1:B) {
  # Splitting the dataset into training and testing subsets
  train <- sample(c(TRUE,FALSE), nrow(fat),
                  replace=TRUE, prob=c(0.7,0.3))
  fat1train  <- fat[train, ]
```

```
  fat1test  <- fat[!train, ]

  # Prepare the data for LASSO Regression
  x_matrix <- model.matrix(brozek ~ ., data=fat1train)[, -1]  # Exclude intercept
  y_vector <- fat1train$brozek

  x_test_matrix <- model.matrix(brozek ~ ., data=fat1test)[, -1]  # Exclude intercept

  # Fit LASSO model with cross-validation to find optimal lambda
  cv_lasso <- cv.glmnet(x_matrix, y_vector, alpha = 1, type.measure = "mse")

  # Extract the lambda that minimizes the cross-validated MSE
  lambda_optimal_lasso <- cv_lasso$lambda.min

  # Predict on test data using the optimal lambda
  predictions_lasso <- predict(cv_lasso, newx = x_test_matrix, s = lambda_optimal_lasso)

  # Since predictions_lasso is a matrix, convert it to a vector for MSE calculation
  predictions_lasso_vector <- as.vector(predictions_lasso)

  # Calculate MSE on test data for LASSO
  actual_responses <- fat1test$brozek  # Ensure alignment with predictions
  mse_lasso[i] <- mean((actual_responses - predictions_lasso_vector)^2)
}

# Calculate the average MSE across all iterations
avg_mse_lasso <- mean(mse_lasso)

# Print the average MSE
print(avg_mse_lasso)


# (vi) Principal component regression
library(pls)
# Fit PCR model and perform cross-validation
pcr.model <- pcr(brozek ~ ., data = fat1train, scale = TRUE, validation = "CV")

# Determine the optimal number of components with minimum cross-validated MSEP
optimal_ncomp <- which.min(MSEP(pcr.model)$val[,,1])

# Predict on test data using the optimal number of components
predictions.pcr.model <- predict(pcr.model, newdata = fat1test, ncomp = optimal_ncomp)

# Convert PCR model predictions to a vector format for consistent calculation
predictions.pcr.vector <- as.vector(predictions.pcr.model)

# Calculate MSE on test data PCR
single_mse_pcr <- mean((fat1test$brozek - predictions.pcr.vector)^2)

# Print the MSE
print(single_mse_pcr)

# Monte Carlo loop
# Initialize storage for MSE values
B <- 100  # Number of Monte Carlo iterations
mse_pcr <- numeric(B)  # To store MSE for each iteration

for (i in 1:B) {
  # Splitting the dataset into training and testing subsets
```

```
  train <- sample(c(TRUE,FALSE), nrow(fat),
                  replace=TRUE, prob=c(0.7,0.3))
  fat1train  <- fat[train, ]
  fat1test  <- fat[!train, ]

  # Fit PCR model and perform cross-validation
  pcr.model <- pcr(brozek ~ ., data = fat1train, scale = TRUE, validation = "CV")

  # Determine the optimal number of components with minimum cross-validated MSEP
  optimal_ncomp <- which.min(MSEP(pcr.model)$val[,,1])

  # Predict on test data using the optimal number of components
  predictions.pcr.model <- predict(pcr.model, newdata = fat1test, ncomp = optimal_ncomp)

  # Convert PCR model predictions to a vector format for consistent calculation
  predictions.pcr.vector <- as.vector(predictions.pcr.model)

  # Calculate MSE on test data PCR
  mse_pcr[i] <- mean((fat1test$brozek - predictions.pcr.vector)^2)
}

# Calculate the average MSE across all iterations
avg_mse_pcr <- mean(mse_pcr)

# Print the average MSE
print(avg_mse_pcr)

# (vii) Partial least squares
pls.model <- plsr(brozek ~ ., data=fat1train, validation="CV")
summary(pls.model)

# Make predictions using the optimal number of components
predictions_pls <- predict(pls.model, newdata=fat1test)

# Ensure predictions are correctly formatted for MSE calculation
predictions_pls <- as.vector(predictions_pls)

# Calculate MSE
single_mse_pls <- mean((fat1test$brozek - predictions_pls)^2)

# Print MSE
print(single_mse_pls)

# Monte Carlo loop
# Initialize storage for MSE values
B <- 100  # Number of Monte Carlo iterations
mse_pls <- numeric(B)  # To store MSE for each iteration

for (i in 1:B) {
  # Splitting the dataset into training and testing subsets
  train <- sample(c(TRUE,FALSE), nrow(fat),
                  replace=TRUE, prob=c(0.7,0.3))
  fat1train  <- fat[train, ]
  fat1test  <- fat[!train, ]

  # Fit PLS model and perform cross-validation
  pls.model <- plsr(brozek ~ ., data = fat1train, validation = "CV")

  # Determine the optimal number of components with minimum cross-validated MSEP
```

```
  optimal_ncomp <- which.min(MSEP(pls.model)$val[,,1])

  # Predict on test data using the optimal number of components
  predictions_pls <- predict(pls.model, newdata = fat1test, ncomp = optimal_ncomp)

  # Ensure predictions are correctly formatted for MSE calculation
  predictions_pls_vector <- as.vector(predictions_pls)

  # Calculate MSE on test data for PLS
  actual_responses <- fat1test$brozek  # Ensure alignment with predictions
  mse_pls[i] <- mean((actual_responses - predictions_pls_vector)^2)
}

# Calculate the average MSE across all iterations
avg_mse_pls <- mean(mse_pls)

# Print the average MSE
print(avg_mse_pls)

library(gridExtra)
library(grid)

# Create a data frame to store MSE values
mse_table <- data.frame(
  Model = c("PLS", "Subset", "PCR", "LASSO", "Ridge", "Stepwise", "Linear Regression"),
  MSE = c(single_mse_pls, single_mse_subset, single_mse_pcr, single_mse_lasso,
          single_mse_ridge, single_mse_stepwise, single_mse_lr)
)

# Print the MSE table
print(mse_table)

# Convert the table to a grid graphical object
table_grob <- tableGrob(mse_table)

# Save the table as an image
png("figures/mse_table.png", width = 800, height = 600)
grid.draw(table_grob)
dev.off()

# Based on the MSE values, the Stepwise model has the lowest MSE,
# making it the best-performing model among the ones tested.

# Create a data frame to store average MSE values
avg_mse_table <- data.frame(
  Model = c("PLS", "Subset", "PCR", "LASSO", "Ridge", "Stepwise", "Linear Regression"),
  MSE = c(avg_mse_pls, avg_mse_best_subset, avg_mse_pcr, avg_mse_lasso, avg_mse_ridge,
          avg_mse_stepwise, avg_mse_lr)
)

# Print the average MSE table
print(avg_mse_table)

# Convert the table to a grid graphical object
table_grob <- tableGrob(avg_mse_table)

# Save the table as an image
png("figures/avg_mse_table.png", width = 800, height = 600)
grid.draw(table_grob)
```

```
dev.off()

# Based on the Average MSE value for the Monte Carlo simulations, the Subset model has the
# lowest Average MSE, making it the best-performing model among the ones tested.
```