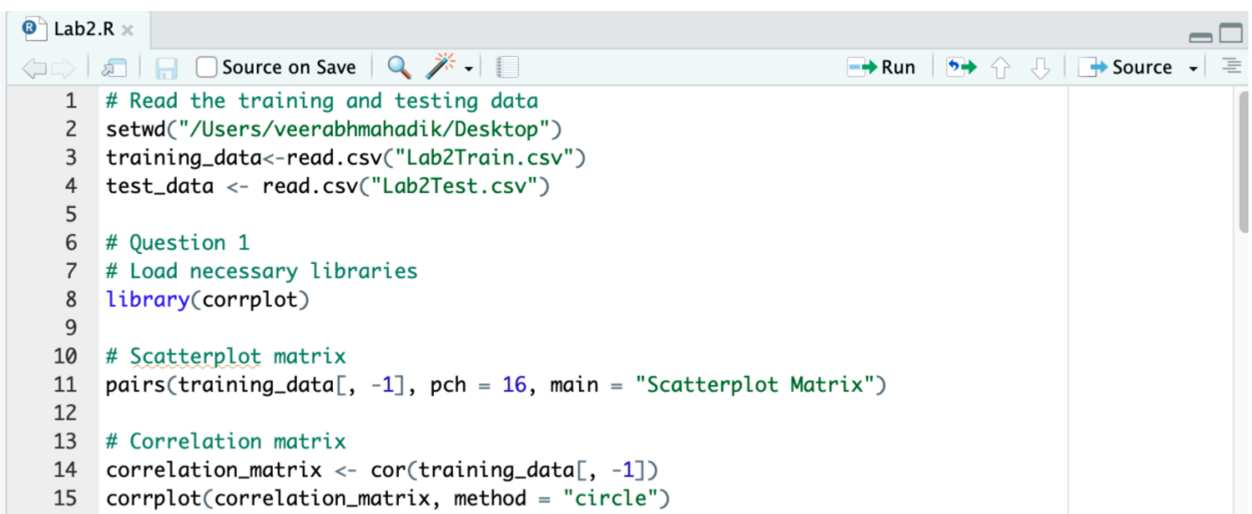
1.

```
# Read the training and testing data

setwd("/Users/veerabhmahadik/Desktop")

training_data<-read.csv("Lab2Train.csv")

test_data <- read.csv("Lab2Test.csv")


# Question 1

# Load necessary libraries

library(corrplot)


# Scatterplot matrix

pairs(training_data[, -1], pch = 16, main = "Scatterplot Matrix")


# Correlation matrix

correlation_matrix <- cor(training_data[, -1])

corrplot(correlation_matrix, method = "circle")
```



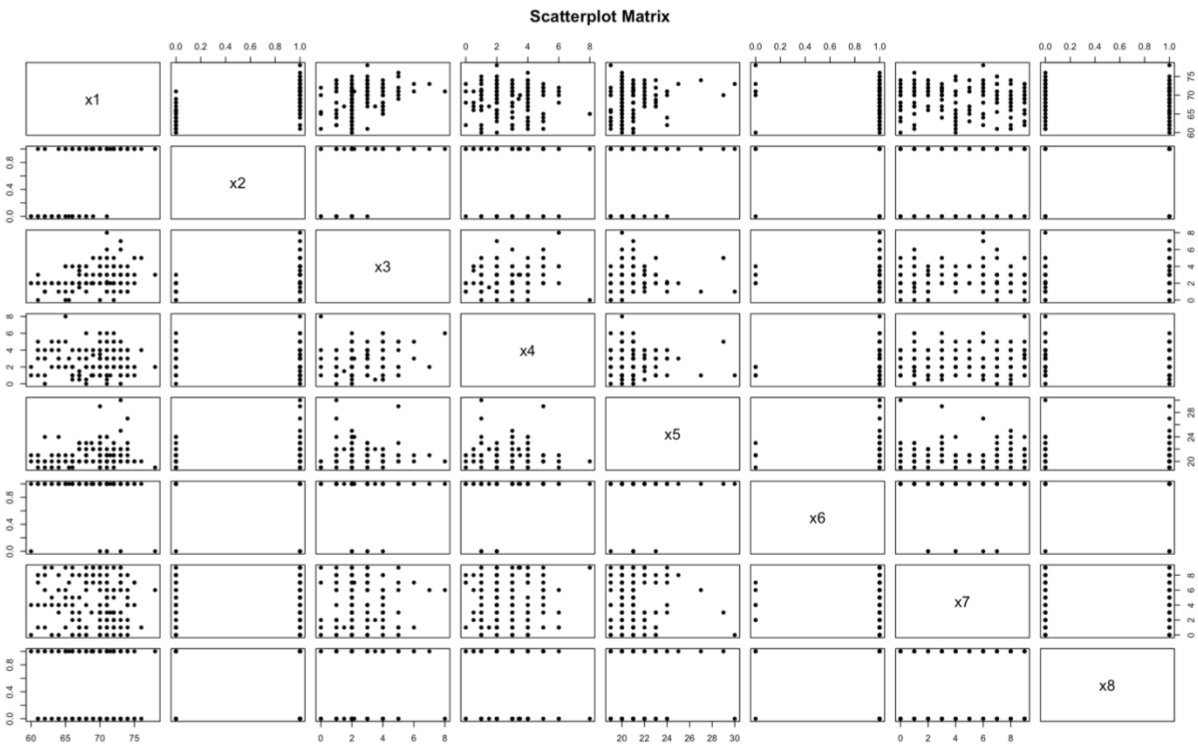
Lab2.R

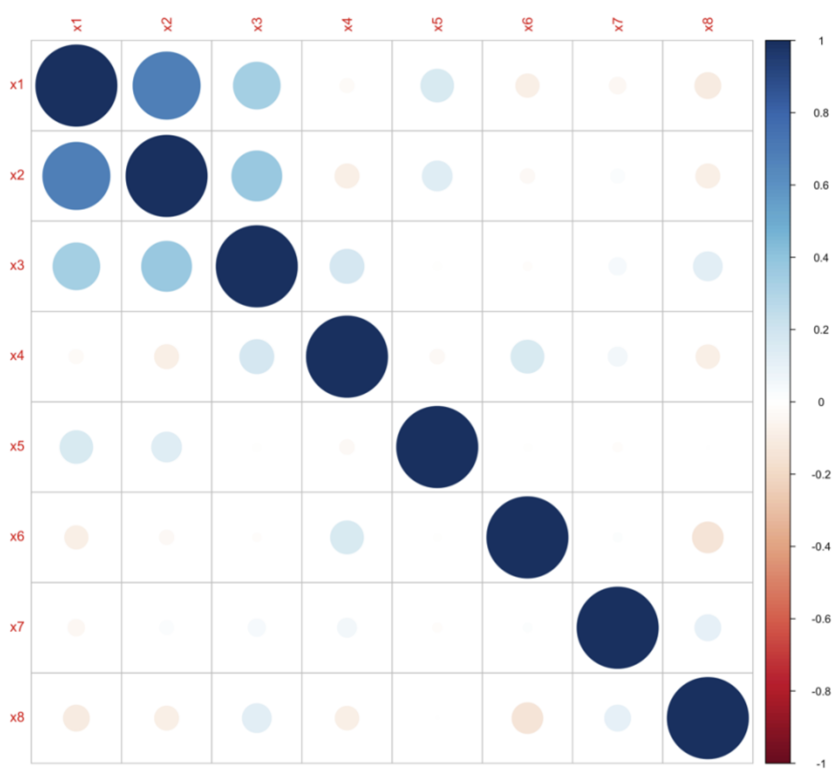
Source on Save

Run

Source

```
# Read the training and testing data
setwd("/Users/veerabhmahadik/Desktop")
training_data<-read.csv("Lab2Train.csv")
test_data <- read.csv("Lab2Test.csv")

# Question 1
# Load necessary libraries
library(corrplot)

# Scatterplot matrix
pairs(training_data[, -1], pch = 16, main = "Scatterplot Matrix")

# Correlation matrix
correlation_matrix <- cor(training_data[, -1])
corrplot(correlation_matrix, method = "circle")
```


Scatterplot Matrix
x1
x2
x3
x4
x5
x6
x7
x8


x1
x2
x3
x4
x5
x6
x7
x8
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

2.

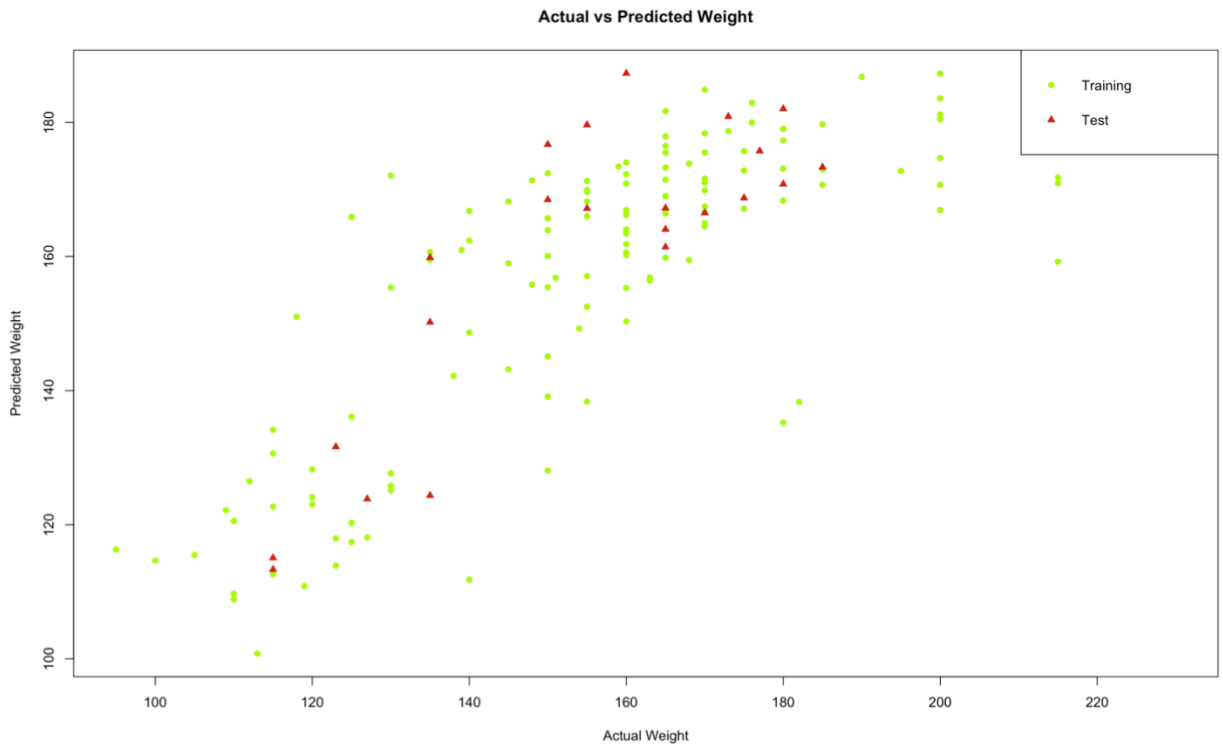
# Question 2

```
# Fit a linear regression model using all predictor variables
model <- lm(y ~ ., data = training_data)

# Make predictions for both training and test sets
training_predictions <- predict(model, newdata = training_data)
test_predictions <- predict(model, newdata = test_data)

# Create a data frame to combine actual and predicted values along with group labels
combined_data <- data.frame(
  Group = c(rep("Training", nrow(training_data)), rep("Test", nrow(test_data))),
  Actual = c(training_data$y, test_data$y),
  Predicted = c(training_predictions, test_predictions)
)

# Scatter plot of actual vs predicted values, distinguishing training and test groups
plot(
  Predicted ~ Actual,
  data = combined_data,
  pch = ifelse(combined_data$Group == "Training", 16, 17),
  col = ifelse(combined_data$Group == "Training", "green", "red"),
  xlab = "Actual Weight",
  ylab = "Predicted Weight",
  main = "Actual vs Predicted Weight"
)

# Add a legend
```

legend("topright", legend = c("Training", "Test"), pch = c(16, 17), col = c("green", "red"))

Lab2.R

```r

# Correlation matrix
correlation_matrix <- cor(training_data[, -1])
corrplot(correlation_matrix, method = "circle")

# Question 2
# Fit a linear regression model using all predictor variables
model <- lm(y ~ ., data = training_data)

# Make predictions for both training and test sets
training_predictions <- predict(model, newdata = training_data)
test_predictions <- predict(model, newdata = test_data)

# Create a data frame to combine actual and predicted values along with group labels
combined_data <- data.frame(
  Group = c(rep("Training", nrow(training_data)), rep("Test", nrow(test_data))),
  Actual = c(training_data$y, test_data$y),
  Predicted = c(training_predictions, test_predictions)
)

# Scatter plot of actual vs predicted values, distinguishing training and test groups
plot(
  Predicted ~ Actual,
  data = combined_data,
  pch = ifelse(combined_data$Group == "Training", 16, 17),
  col = ifelse(combined_data$Group == "Training", "green", "red"),
  xlab = "Actual Weight",
  ylab = "Predicted Weight",
  main = "Actual vs Predicted Weight"
)

# Add a legend
legend("topright", legend = c("Training", "Test"), pch = c(16, 17), col = c("green", "red"))
```




3.

```
# Question 3

# Make predictions for the test set

test_predictions <- predict(model, newdata = test_data, interval = "prediction", level = 0.95)


# Extract predicted values and prediction intervals

predicted_values <- test_predictions[, 1]

prediction_intervals <- test_predictions[, c(2, 3)]


# Calculate prediction errors

prediction_errors <- test_data$y - predicted_values


# Output prediction intervals and prediction errors

print("Prediction Intervals:")
```

print(prediction_intervals)

print("Prediction Errors:")

print(prediction_errors)

Lab2.R*

Source on Save    Run    Source

```r

# Question 3
# Make predictions for the test set
test_predictions <- predict(model, newdata = test_data, interval = "prediction", level = 0.95)

# Extract predicted values and prediction intervals
predicted_values <- test_predictions[, 1]
prediction_intervals <- test_predictions[, c(2, 3)]

# Calculate prediction errors
prediction_errors <- test_data$y - predicted_values

# Output prediction intervals and prediction errors
print("Prediction Intervals:")
print(prediction_intervals)
print("Prediction Errors:")
print(prediction_errors)
```

46:13 (Top Level) R Script

Console Terminal Background Jobs

R 4.2.1 · ~/Desktop/

```
5  123.88557 195.7577
6  127.70647 205.2711
7   76.40633 150.2313
8  132.05538 204.8932
9  140.42306 212.9950
10 143.33967 215.9131
11  78.39535 151.7043
12 150.84750 223.7724
13 113.75799 186.6301
14 131.03984 203.3750
15  95.08791 168.1358
16 139.48053 211.9656
17 128.21593 199.8877
18 131.33489 202.9951
19 145.97155 218.0690
20 132.58052 204.8730
21 125.14799 197.6686
22 134.75912 206.8255
> print("Prediction Errors:")
[1] "Prediction Errors:"
> print(prediction_errors)
          1           2           3           4           5           6           7           8           9          10          11          12          13 
-7.88198505 11.68719853 10.65570257  3.16364494 -24.82162231  3.51121492  1.68116413 -18.47428599 -26.70905199 -24.62636229 -0.04984664 -27.30993150 -15.19404159 
         14          15          16          17          18          19          20          21          22 
-2.20741754 -8.61187550  1.27694572  0.94817650 -12.16499120 -2.02026224  6.27326134  3.59168646  9.20771322 
>
```

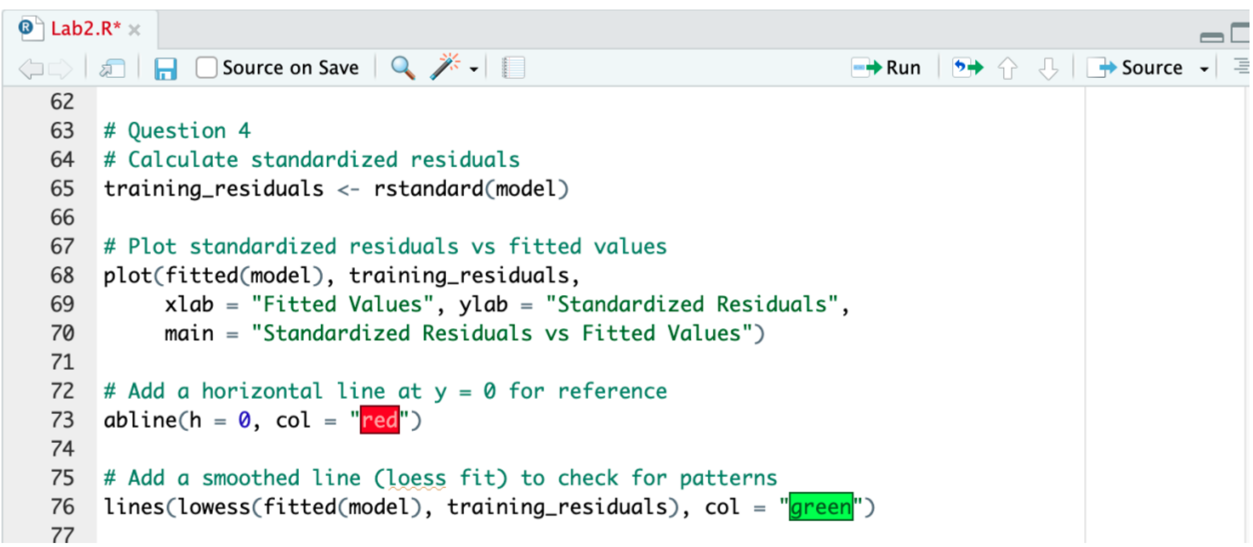
Console Terminal Background Jobs

R 4.2.1 · ~/Desktop/

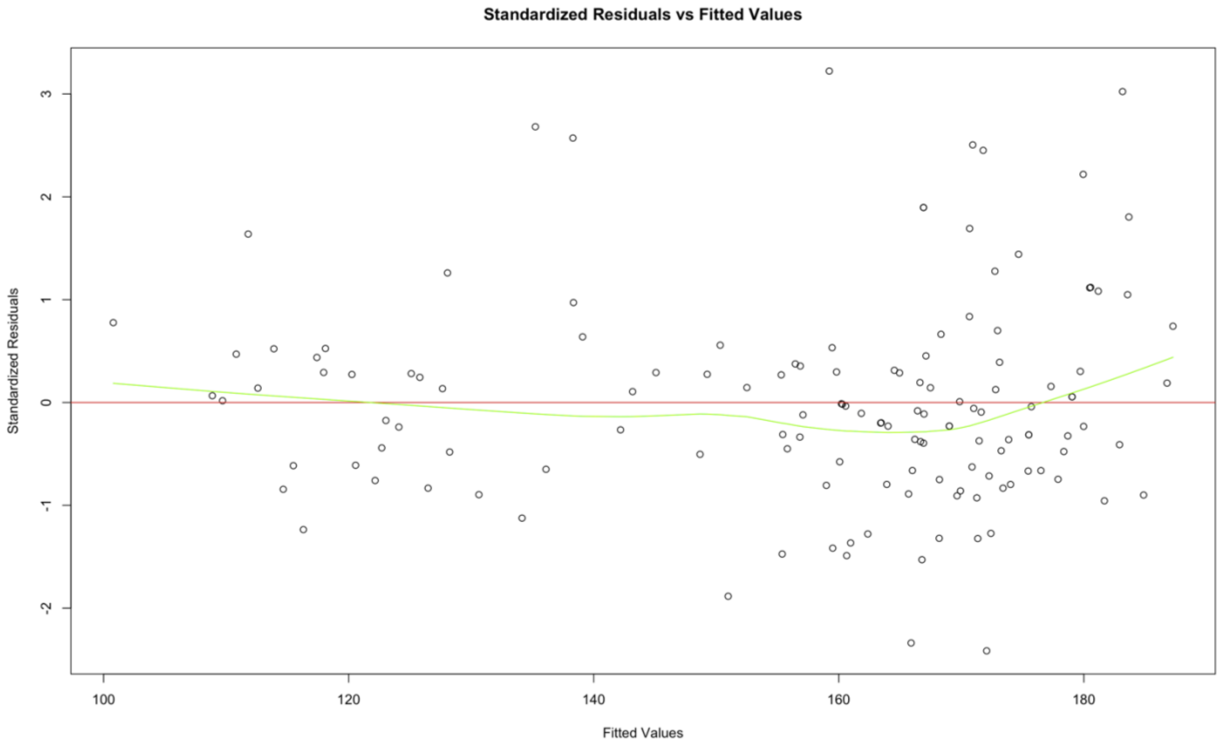
```
> # Calculate prediction errors
> prediction_errors <- test_data$y - predicted_values
> # Output prediction intervals and prediction errors
> print("Prediction Intervals:")
[1] "Prediction Intervals:"
> print(prediction_intervals)
         lwr      upr
1  144.70866 217.0553
2  137.41433 209.2113
3   88.14774 160.5409
4   87.40075 160.2720
5  123.88557 195.7577
6  127.70647 205.2711
7   76.40633 150.2313
8  132.05538 204.8932
9  140.42306 212.9950
10 143.33967 215.9131
11  78.39535 151.7043
12 150.84750 223.7724
13 113.75799 186.6301
14 131.03984 203.3750
15  95.08791 168.1358
16 139.48053 211.9656
17 128.21593 199.8877
18 131.33489 202.9951
19 145.97155 218.0690
20 132.58052 204.8730
21 125.14799 197.6686
22 134.75912 206.8255
> print("Prediction Errors:")
[1] "Prediction Errors:"
> print(prediction_errors)
          1           2           3           4           5           6           7           8           9          10          11          12          13 
-7.88198505 11.68719853 10.65570257  3.16364494 -24.82162231  3.51121492  1.68116413 -18.47428599 -26.70905199 -24.62636229 -0.04984664 -27.30993150 -15.19404159 
         14          15          16          17          18          19          20          21          22 
-2.20741754 -8.61187550  1.27694572  0.94817650 -12.16499120 -2.02026224  6.27326134  3.59168646  9.20771322 
>
```

4.

```
# Question 4

# Calculate standardized residuals

training_residuals <- rstandard(model)


# Plot standardized residuals vs fitted values

plot(fitted(model), training_residuals,

    xlab = "Fitted Values", ylab = "Standardized Residuals",

    main = "Standardized Residuals vs Fitted Values")


# Add a horizontal line at y = 0 for reference

abline(h = 0, col = "red")


# Add a smoothed line (loess fit) to check for patterns

lines(lowess(fitted(model), training_residuals), col = "green")
```




```

# Question 4
# Calculate standardized residuals
training_residuals <- rstandard(model)

# Plot standardized residuals vs fitted values
plot(fitted(model), training_residuals,
     xlab = "Fitted Values", ylab = "Standardized Residuals",
     main = "Standardized Residuals vs Fitted Values")

# Add a horizontal line at y = 0 for reference
abline(h = 0, col = "red")

# Add a smoothed line (loess fit) to check for patterns
lines(lowess(fitted(model), training_residuals), col = "green")

```




5.

# Question 5

# Problem 2: Fit Regression Models

# Fit regression model with only predictor variables x1 and x2

model_two_predictors <- lm(y ~ x1 + x2, data = training_data)

# Make predictions for the test set using both models

test_predictions_eight_predictors <- predict(model, newdata = test_data)

test_predictions_two_predictors <- predict(model_two_predictors, newdata = test_data)

# Problem 3: Calculate Prediction Intervals and Errors

# Calculate prediction intervals and errors for both models

prediction_intervals_eight_predictors <- predict(model, newdata = test_data, interval = "prediction", level = 0.95)

```r
prediction_errors_eight_predictors <- test_data$y - prediction_intervals_eight_predictors[,
1]

prediction_intervals_two_predictors <- predict(model_two_predictors, newdata =
test_data, interval = "prediction", level = 0.95)
prediction_errors_two_predictors <- test_data$y - prediction_intervals_two_predictors[, 1]

# Problem 4: Plot Standardized Residuals vs Fitted Values
# Calculate standardized residuals for both models
training_residuals_two_predictors <- rstandard(model_two_predictors)

# Plot standardized residuals vs fitted values for the two-predictor model
plot(fitted(model_two_predictors), training_residuals_two_predictors,
    xlab = "Fitted Values", ylab = "Standardized Residuals",
    main = "Standardized Residuals vs Fitted Values (Two-Predictor Model)")

abline(h = 0, col = "red")
lines(lowess(fitted(model_two_predictors), training_residuals_two_predictors), col =
"green")

# Construct side-by-side box plots of the two sets of prediction errors for the test data
# Combine prediction errors into a data frame
prediction_errors_data <- data.frame(
  Model = c(rep("Eight Predictors", length(prediction_errors_eight_predictors)),
         rep("Two Predictors", length(prediction_errors_two_predictors))),
  Prediction_Error = c(prediction_errors_eight_predictors,
prediction_errors_two_predictors)
```

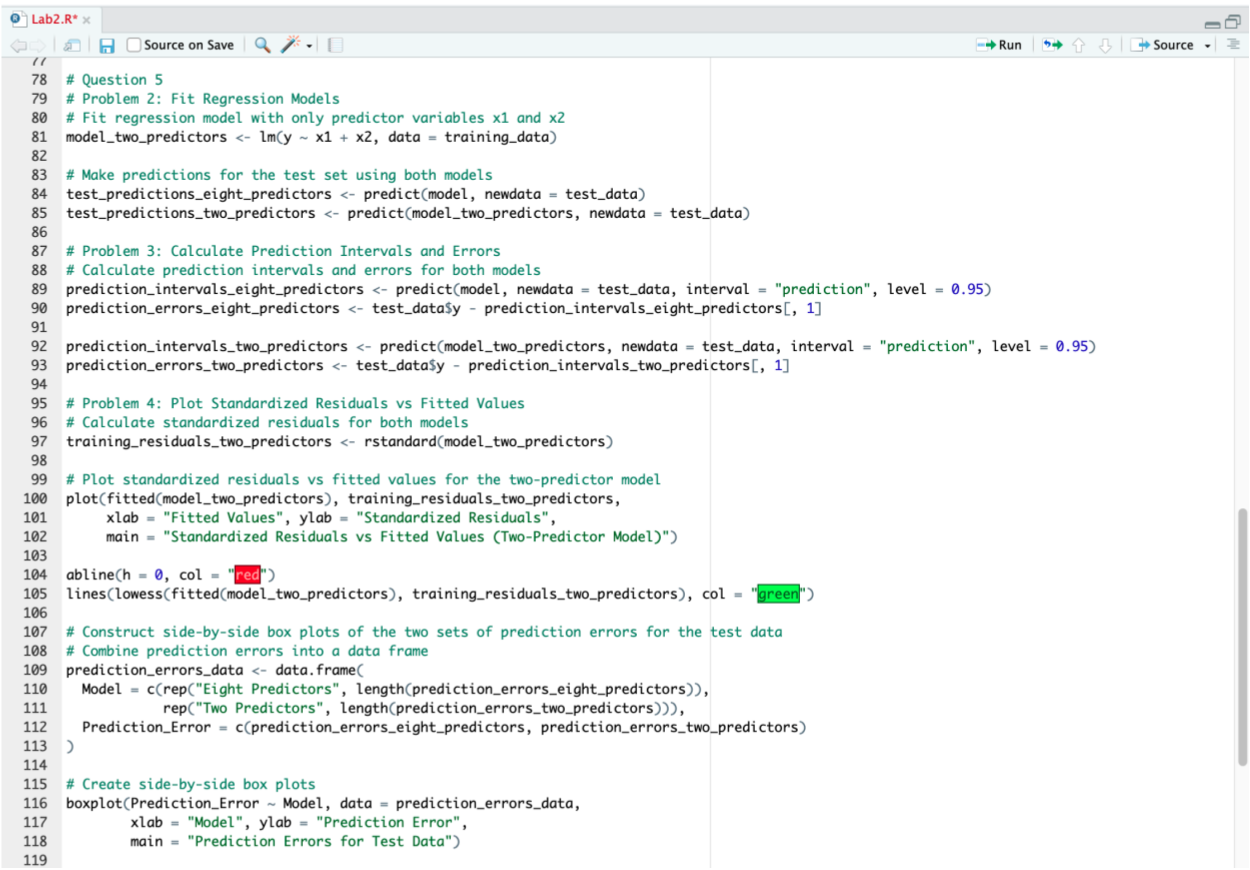
)

# Create side-by-side box plots

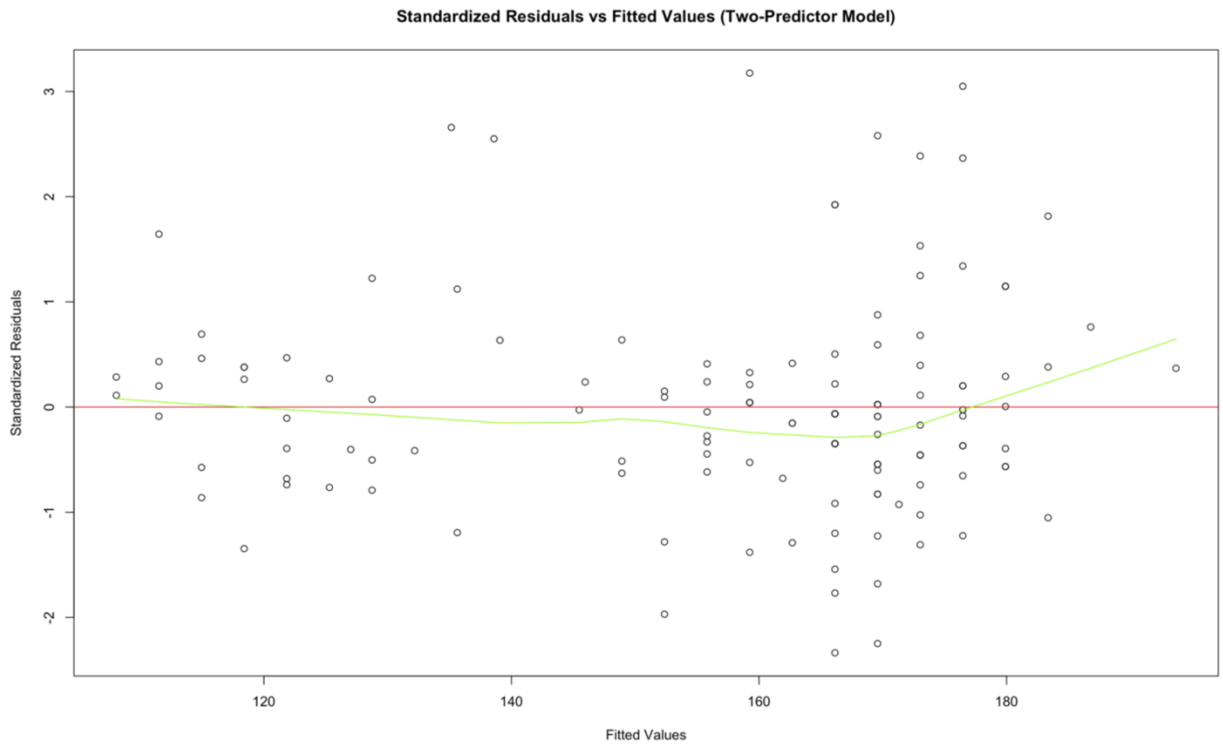
boxplot(Prediction_Error ~ Model, data = prediction_errors_data,

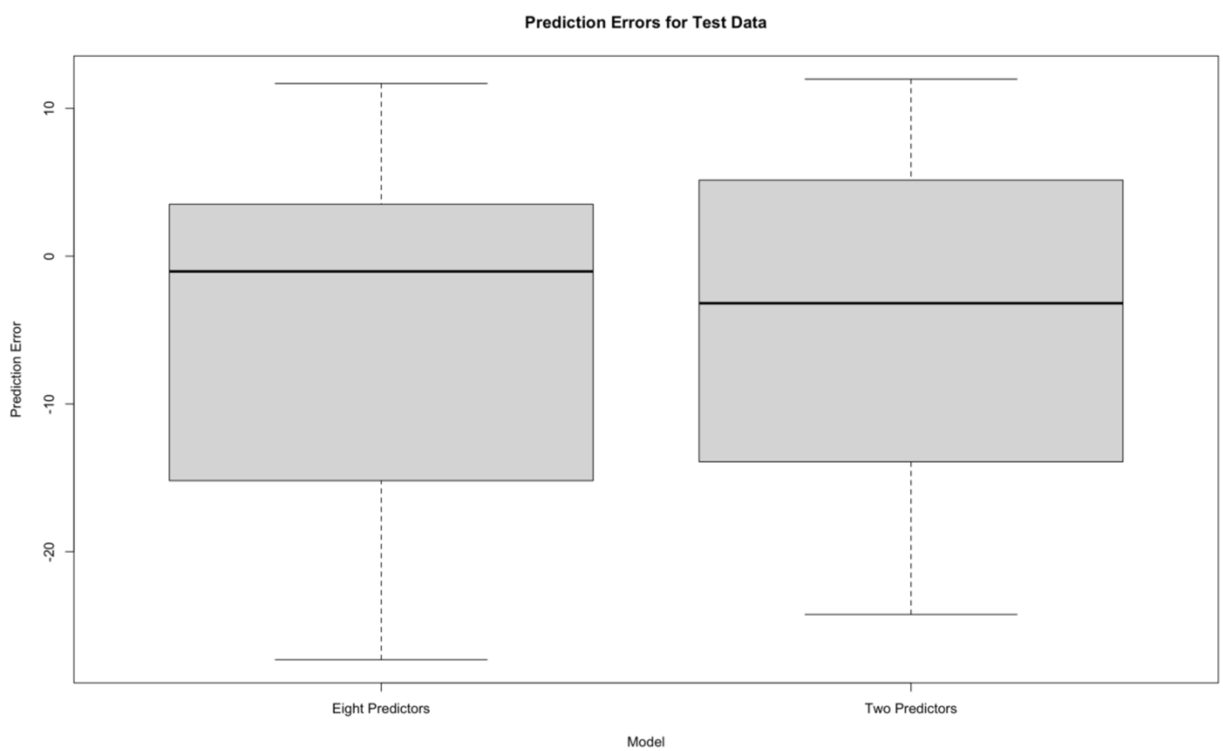
xlab = "Model", ylab = "Prediction Error",

main = "Prediction Errors for Test Data")




```r
# Question 5
# Problem 2: Fit Regression Models
# Fit regression model with only predictor variables x1 and x2
model_two_predictors <- lm(y ~ x1 + x2, data = training_data)

# Make predictions for the test set using both models
test_predictions_eight_predictors <- predict(model, newdata = test_data)
test_predictions_two_predictors <- predict(model_two_predictors, newdata = test_data)

# Problem 3: Calculate Prediction Intervals and Errors
# Calculate prediction intervals and errors for both models
prediction_intervals_eight_predictors <- predict(model, newdata = test_data, interval = "prediction", level = 0.95)
prediction_errors_eight_predictors <- test_data$y - prediction_intervals_eight_predictors[, 1]

prediction_intervals_two_predictors <- predict(model_two_predictors, newdata = test_data, interval = "prediction", level = 0.95)
prediction_errors_two_predictors <- test_data$y - prediction_intervals_two_predictors[, 1]

# Problem 4: Plot Standardized Residuals vs Fitted Values
# Calculate standardized residuals for both models
training_residuals_two_predictors <- rstandard(model_two_predictors)

# Plot standardized residuals vs fitted values for the two-predictor model
plot(fitted(model_two_predictors), training_residuals_two_predictors,
     xlab = "Fitted Values", ylab = "Standardized Residuals",
     main = "Standardized Residuals vs Fitted Values (Two-Predictor Model)")

abline(h = 0, col = "red")
lines(lowess(fitted(model_two_predictors), training_residuals_two_predictors), col = "green")

# Construct side-by-side box plots of the two sets of prediction errors for the test data
# Combine prediction errors into a data frame
prediction_errors_data <- data.frame(
  Model = c(rep("Eight Predictors", length(prediction_errors_eight_predictors)),
            rep("Two Predictors", length(prediction_errors_two_predictors))),
  Prediction_Error = c(prediction_errors_eight_predictors, prediction_errors_two_predictors)
)

# Create side-by-side box plots
boxplot(Prediction_Error ~ Model, data = prediction_errors_data,
        xlab = "Model", ylab = "Prediction Error",
        main = "Prediction Errors for Test Data")
```


Standardized Residuals vs Fitted Values (Two-Predictor Model)
Standardized Residuals
Fitted Values
120
140
160
180
-2
-1
0
1
2
3


Prediction Errors for Test Data
Prediction Error
Model
Eight Predictors
Two Predictors
10
0
-10
-20

6.

# Question 6

```r
# Fit the full model with all predictors
full_model <- lm(y ~ ., data = training_data)

# Fit the reduced model excluding predictors x3 through x8
reduced_model <- lm(y ~ x1 + x2, data = training_data)

# Calculate the sums of squares for both models
ss_full <- sum(residuals(full_model)^2)
ss_reduced <- sum(residuals(reduced_model)^2)

# Calculate the extra sum of squares
extra_ss <- ss_reduced - ss_full

# Calculate degrees of freedom for the F-test
df_full <- df.residual(full_model)
df_reduced <- df.residual(reduced_model)
df_extra <- df_reduced - df_full

# Perform the F-test
F_stat <- (extra_ss / df_extra) / (ss_full / df_full)
p_value <- pf(F_stat, df_extra, df_full, lower.tail = FALSE)

# Results
cat("Extra Sum of Squares:", extra_ss, "\n")
cat("F-statistic:", F_stat, "\n")
cat("p-value:", p_value, "\n")
```

```r
# Question 6
# Fit the full model with all predictors
full_model <- lm(y ~ ., data = training_data)

# Fit the reduced model excluding predictors x3 through x8
reduced_model <- lm(y ~ x1 + x2, data = training_data)

# Calculate the sums of squares for both models
ss_full <- sum(residuals(full_model)^2)
ss_reduced <- sum(residuals(reduced_model)^2)

# Calculate the extra sum of squares
extra_ss <- ss_reduced - ss_full

# Calculate degrees of freedom for the F-test
df_full <- df.residual(full_model)
df_reduced <- df.residual(reduced_model)
df_extra <- df_reduced - df_full

# Perform the F-test
F_stat <- (extra_ss / df_extra) / (ss_full / df_full)
p_value <- pf(F_stat, df_extra, df_full, lower.tail = FALSE)

# Results
cat("Extra Sum of Squares:", extra_ss, "\n")
cat("F-statistic:", F_stat, "\n")
cat("p-value:", p_value, "\n")
```

```
> # Calculate degrees of freedom for the F-test
> df_full <- df.residual(full_model)
> df_reduced <- df.residual(reduced_model)
> df_extra <- df_reduced - df_full
> # Perform the F-test
> F_stat <- (extra_ss / df_extra) / (ss_full / df_full)
> p_value <- pf(F_stat, df_extra, df_full, lower.tail = FALSE)
> # Results
> cat("Extra Sum of Squares:", extra_ss, "\n")
Extra Sum of Squares: 1159.286
> cat("F-statistic:", F_stat, "\n")
F-statistic: 0.6062496
> cat("p-value:", p_value, "\n")
p-value: 0.7249332
>
```

7.

**Performance:**

- Prediction Intervals: The 8-predictor model yields broader prediction intervals compared to the 2-predictor model, signaling increased uncertainty in predictions.

- Extra Sum of Squares F-test: The relatively low F-statistic (0.606) and high p-value (0.725) from the F-test suggest that the incorporation of predictors x3 through x8 does not significantly enhance the model beyond using only x1 and x2.

**Quantitative and Qualitative Analysis:**

- Visual inspection of scatter plots from Problem 2 helps gauge the fit of models to the data. A clearer alignment of the 8-predictor model with data points might imply superior model performance.

- A quantitative examination of prediction errors in Problem 5 is crucial. If the 8-predictor model consistently yields lower errors, it indicates enhanced predictive capability.

- The F-test outcome in Problem 6 implies that the inclusion of additional predictors may not contribute substantially to the model's predictive prowess, endorsing the sufficiency of a simpler model with only x1 and x2.

**Recommendation:**

- Considering the evidence, it appears that the 2-predictor model (utilizing only height and gender) is preferable for predicting the weight of a new person.

- The 8-predictor model does not exhibit a significant performance advantage over the 2-predictor model in terms of predictive accuracy, as indicated by the F-test and potentially reflected in the comparison of prediction errors.

- The 2-predictor model, with its simplicity and interpretability, provides reasonable weight predictions.

**Conclusion:**

- The selection of a model should weigh both predictive accuracy and model complexity. While the 8-predictor model captures more data nuances, the additional predictors may not justify the increased model complexity.

- In practical terms, the 2-predictor model is favored due to its simplicity and comparable predictive performance. However, further validation on independent datasets is advisable to affirm the model's robustness.

By considering both quantitative factors (e.g., prediction errors, F-test outcomes) and qualitative aspects (e.g., model simplicity, interpretability), the recommendation leans towards the 2-predictor model for predicting the weight of a new person.