Exercise 4

Advanced Methods for Regression and Classification

Rita Selimi

11/11/2024

Load the necessary libraries

Load Data

```
# Load the dataset
load("building.RData")
# str(df)
```

Data Splitting

```
set.seed(12332281)

# Split the data
sample_index <- sample(1:nrow(df), size = 2/3 * nrow(df))
train_data <- df[sample_index, ]
test_data <- df[-sample_index, ]
```

Fit regression model on the training data

```
# Fit linear regression model
model <- lm(y ~ ., data = train_data)
# summary(model)
```

Compute RMSE (training data)

```
train_pred <- predict(model, newdata = train_data)

# Compute RMSE for training data
train_rmse <- sqrt(mean((train_data$y - train_pred)^2))
cat("Training RMSE:", train_rmse, "\n")
```

```
## Training RMSE: 0.1928942
```

1. Ridge Regression

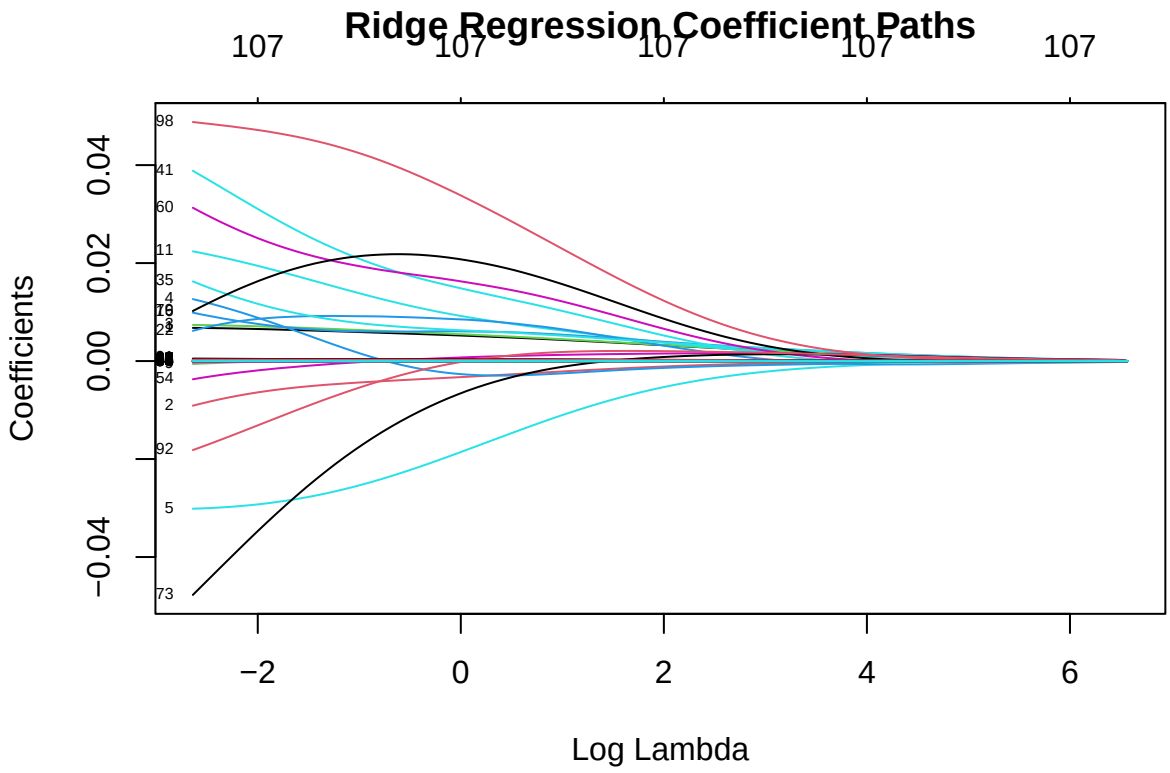
a) Fit Ridge Regression Model

Prepare the data

```
# Prepare the predictor and response matrices for the training data
x_train <- as.matrix(train_data[, -1])
y_train <- train_data$y

# Prepare test data matrices
x_test <- as.matrix(test_data[, -1])
y_test <- test_data$y
```

Fit Ridge Regression Model

```
# Fit Ridge Regression model
ridge_model <- glmnet(x_train, y_train, alpha = 0)

# Plot the ridge model to visualize the coefficient paths
plot(ridge_model, xvar = "lambda", label = TRUE, main = "Ridge Regression Coefficient Paths")
```



How can you interpret the plot?

The plot displays how the coefficients for each predictor change as `lambda` increases. As `lambda` increases, coefficients shrink towards zero, which helps control overfitting by reducing the impact of multicollinearity. Each line represents a different predictor's coefficient.

With low `lambda` (left side), coefficients are larger, meaning the model fits more closely to the data. As `lambda` increases, less important predictors shrink faster, while important ones hold their influence longer. Eventually, at very high `lambda` values, all coefficients get close to zero, which can lead to underfitting.

Which default parameters are used for lambda?

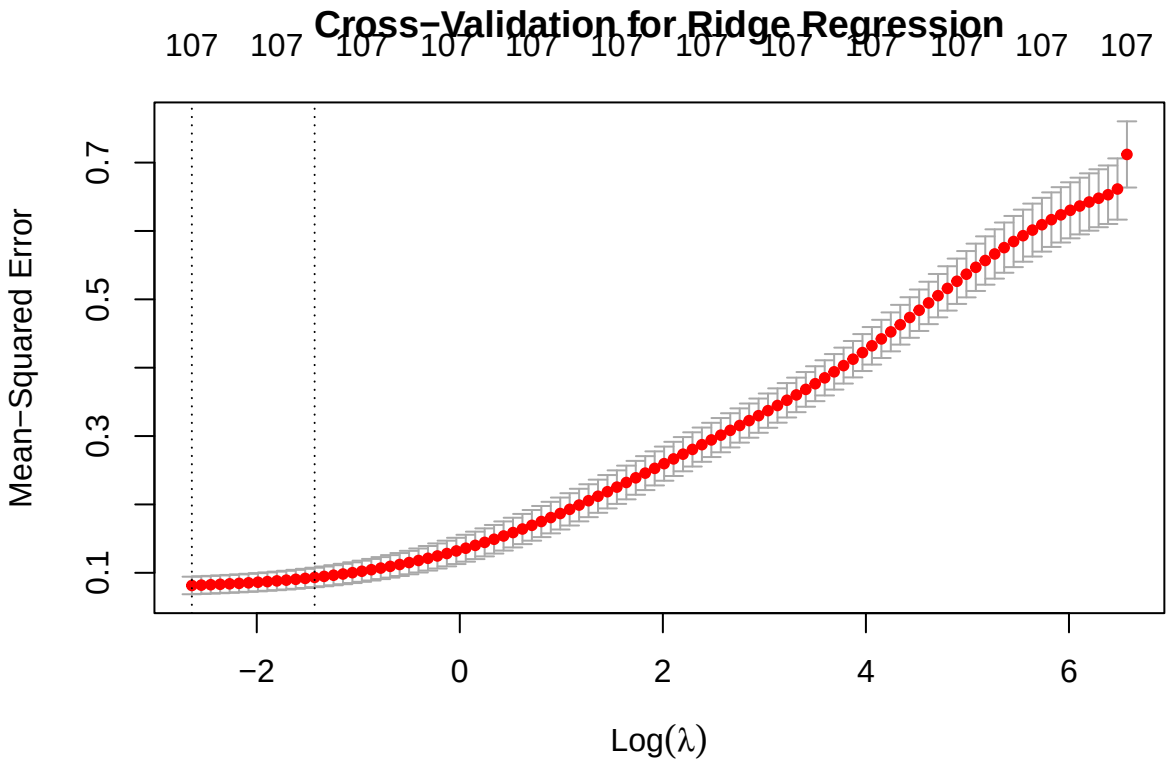
In this function, the default `lambda` sequence is automatically generated based on the data. This sequence typically starts from a maximum value, which is high enough to shrink all coefficients to nearly zero, and then decreases logarithmically to a minimum fraction of the maximum. The plot shows this progression, with `Log Lambda` on the x-axis. As `lambda` increases, the regularization effect strengthens, and the coefficients shrink toward zero.

What is the meaning of the parameter alpha?

The `alpha` parameter controls the type of regularization: - `alpha = 0`: Ridge Regression (L2 penalty) shrinks coefficients but keeps all of them, useful for handling multicollinearity.

b) Cross-Validation for Optimal Lambda

```r
# Perform cross-validation with Ridge Regression (alpha = 0)
ridge_cv <- cv.glmnet(x_train, y_train, alpha = 0)

# Plot the cross-validation results
plot(ridge_cv, main = "Cross-Validation for Ridge Regression")
```




```r
# Print the optimal lambda values
cat("Lambda with minimum MSE:", ridge_cv$lambda.min, "\n")
```

```
## Lambda with minimum MSE: 0.07141336
```

```r
cat("Lambda with 1-SE rule:", ridge_cv$lambda.1se, "\n")
```

```
## Lambda with 1-SE rule: 0.2393492
```

```r
cat("Log Lambda with minimum MSE:", log(ridge_cv$lambda.min), "\n")
```

```
## Log Lambda with minimum MSE: -2.63927
```

```r
cat("Log Lambda with 1-SE rule:", log(ridge_cv$lambda.1se), "\n")
```

```
## Log Lambda with 1-SE rule: -1.429832
```

The plot shows the mean squared error (MSE) for each value of `lambda` used during cross-validation. Dashed lines on the plot indicate `lambda.min` and `lambda.1se`, helping identify the best `lambda` values for balancing model performance. 1. **`lambda.min`**: This is the `lambda` value that achieves the minimum cross-validated MSE. This `lambda` results in the best predictive accuracy among all tested values. 2. **`lambda.1se`**: This is the largest `lambda` within one standard error of the minimum MSE. Using this value often leads to a simpler model with slightly more regularization, reducing overfitting risk.

The vertical error bars show the standard error of the MSE across cross-validation folds for each lambda value. Shorter error bars indicate more consistent MSE values across folds, while longer bars suggest greater variability.

The plot shows that as lambda decreases, the regularization effect weakens, and the MSE decreases to reach its minimum around lambda.min. The optimal tuning parameters from the cross-validation results are: the **lambda.min** value is 0.07141336 and the **lambda.1se** value is 0.1810589

```
# Obtain coefficients for the model using lambda.min
coef_min <- coef(ridge_cv, s = "lambda.min")
# Obtain coefficients for the model using lambda.1se
coef_1se <- coef(ridge_cv, s = "lambda.1se")

# Convert the sparse matrix to a data frame
coef_min_df <- as.data.frame(as.matrix(coef_min))
coef_1se_df <- as.data.frame(as.matrix(coef_1se))
print(coef_min_df)
```

```
##                              s1
## (Intercept)         2.064868e+00
## START.YEAR          6.773860e-03
## START.QUARTER      -9.106207e-03
## COMPLETION.YEAR     7.375795e-03
## COMPLETION.QUARTER  1.264652e-02
## PhysFin1           -3.014116e-02
## PhysFin2            2.025580e-05
## PhysFin3           -5.554051e-05
## PhysFin4           -3.527582e-05
## PhysFin5           -5.857419e-04
## PhysFin6            2.548419e-04
## PhysFin7            2.240144e-02
## PhysFin8            3.486082e-04
## Econ1              -5.952670e-06
## Econ2               1.297971e-04
## Econ3               3.488637e-04
## Econ4               9.882388e-03
## Econ5               1.842354e-08
## Econ6               8.186419e-07
## Econ7               1.179136e-04
## Econ8               1.927953e-05
## Econ9               3.324336e-07
## Econ10              6.164708e-03
## Econ11             -7.648616e-06
## Econ12              4.017106e-06
## Econ13              2.864845e-06
## Econ14              2.157881e-05
## Econ15              3.158719e-04
## Econ16              1.495569e-04
## Econ17              3.812913e-06
## Econ18              8.698462e-07
## Econ19              5.203772e-08
## Econ1.lag1          1.291298e-05
## Econ2.lag1          1.380293e-05
## Econ3.lag1         -1.317764e-04
```

```
## Econ4.lag1          1.625909e-02
## Econ5.lag1          1.701285e-08
## Econ6.lag1         -6.495074e-06
## Econ7.lag1          1.049669e-05
## Econ8.lag1          1.849144e-04
## Econ9.lag1          5.340490e-07
## Econ10.lag1         3.884703e-02
## Econ11.lag1        -1.778834e-05
## Econ12.lag1         6.919671e-06
## Econ13.lag1         5.491944e-06
## Econ14.lag1         5.746214e-06
## Econ15.lag1         2.776931e-04
## Econ16.lag1         1.695921e-04
## Econ17.lag1         2.211414e-06
## Econ18.lag1         1.006409e-07
## Econ19.lag1         1.308758e-08
## Econ1.lag2         -4.497651e-06
## Econ2.lag2          5.266017e-05
## Econ3.lag2         -1.533792e-04
## Econ4.lag2         -3.718570e-03
## Econ5.lag2          2.514858e-08
## Econ6.lag2         -6.482133e-06
## Econ7.lag2         -1.676108e-04
## Econ8.lag2          1.518860e-04
## Econ9.lag2          1.258403e-06
## Econ10.lag2         3.129852e-02
## Econ11.lag2         1.829139e-05
## Econ12.lag2         8.919879e-07
## Econ13.lag2        -7.198795e-07
## Econ14.lag2         1.272201e-05
## Econ15.lag2         1.893105e-04
## Econ16.lag2         2.013250e-04
## Econ17.lag2         7.586893e-08
## Econ18.lag2         7.871515e-07
## Econ19.lag2        -1.933130e-08
## Econ1.lag3          3.498546e-05
## Econ2.lag3          1.124135e-04
## Econ3.lag3          9.813411e-05
## Econ4.lag3         -4.773526e-02
## Econ5.lag3          1.851171e-08
## Econ6.lag3         -5.756891e-06
## Econ7.lag3         -5.021630e-05
## Econ8.lag3          2.373648e-04
## Econ9.lag3          1.315355e-07
## Econ10.lag3         1.022295e-02
## Econ11.lag3         1.115674e-05
## Econ12.lag3         9.341606e-06
## Econ13.lag3         9.570124e-07
## Econ14.lag3         2.457913e-05
## Econ15.lag3         3.385168e-04
## Econ16.lag3         2.914323e-04
## Econ17.lag3        -3.036519e-06
## Econ18.lag3         2.847741e-07
## Econ19.lag3         1.572744e-08
```

```
## Econ1.lag4          3.749051e-05
## Econ2.lag4          2.260209e-04
## Econ3.lag4          5.102211e-04
## Econ4.lag4         -1.817344e-02
## Econ5.lag4          1.348488e-08
## Econ6.lag4          5.459856e-06
## Econ7.lag4          3.934593e-05
## Econ8.lag4         -2.413299e-04
## Econ9.lag4          1.348938e-07
## Econ10.lag4         4.879473e-02
## Econ11.lag4         1.084777e-05
## Econ12.lag4         1.299363e-05
## Econ13.lag4         1.338420e-07
## Econ14.lag4         1.368359e-05
## Econ15.lag4         3.872322e-04
## Econ16.lag4         2.925019e-04
## Econ17.lag4         3.452229e-06
## Econ18.lag4         9.129923e-07
## Econ19.lag4         2.256251e-08
```

```
print(coef_1se_df)
```

```
##                             s1
## (Intercept)         2.432805e+00
## START.YEAR          6.211527e-03
## START.QUARTER      -5.015412e-03
## COMPLETION.YEAR     6.690077e-03
## COMPLETION.QUARTER  4.682784e-03
## PhysFin1           -2.745227e-02
## PhysFin2            9.669766e-06
## PhysFin3           -2.352381e-05
## PhysFin4           -3.784910e-06
## PhysFin5            1.729183e-04
## PhysFin6            1.979074e-04
## PhysFin7            1.627678e-02
## PhysFin8            2.515008e-04
## Econ1               2.477827e-06
## Econ2               6.199776e-05
## Econ3               1.640411e-04
## Econ4               6.414360e-03
## Econ5               1.073608e-08
## Econ6               4.351404e-07
## Econ7               8.592621e-06
## Econ8               5.240672e-05
## Econ9               4.005693e-07
## Econ10              9.181003e-03
## Econ11              1.615633e-06
## Econ12              3.052118e-06
## Econ13              3.175212e-06
## Econ14              1.953244e-05
## Econ15              2.288092e-04
## Econ16              1.315194e-04
## Econ17              2.688564e-06
## Econ18              1.702589e-07
```

```
## Econ19              2.280546e-08
## Econ1.lag1          1.084263e-05
## Econ2.lag1          1.835835e-05
## Econ3.lag1          2.701164e-06
## Econ4.lag1          9.013807e-03
## Econ5.lag1          9.701494e-09
## Econ6.lag1         -3.969162e-06
## Econ7.lag1         -2.553241e-05
## Econ8.lag1          1.935261e-04
## Econ9.lag1          5.317371e-07
## Econ10.lag1         2.472991e-02
## Econ11.lag1        -3.596960e-06
## Econ12.lag1         1.974894e-06
## Econ13.lag1         3.853757e-06
## Econ14.lag1         1.369356e-05
## Econ15.lag1         2.322288e-04
## Econ16.lag1         1.537616e-04
## Econ17.lag1         1.857927e-06
## Econ18.lag1         4.901056e-07
## Econ19.lag1         5.380302e-09
## Econ1.lag2          2.215374e-06
## Econ2.lag2          3.941583e-05
## Econ3.lag2          3.409327e-05
## Econ4.lag2         -8.440700e-04
## Econ5.lag2          1.108381e-08
## Econ6.lag2         -4.199833e-06
## Econ7.lag2         -1.142909e-04
## Econ8.lag2          2.089432e-04
## Econ9.lag2          5.485206e-07
## Econ10.lag2         2.130304e-02
## Econ11.lag2         7.276992e-06
## Econ12.lag2        -2.178625e-07
## Econ13.lag2         8.203958e-07
## Econ14.lag2         1.466632e-05
## Econ15.lag2         2.146689e-04
## Econ16.lag2         1.863167e-04
## Econ17.lag2         1.343138e-06
## Econ18.lag2         1.229094e-06
## Econ19.lag2        -4.788735e-09
## Econ1.lag3          1.822453e-05
## Econ2.lag3          8.307329e-05
## Econ3.lag3          1.964556e-04
## Econ4.lag3         -2.419081e-02
## Econ5.lag3          8.708639e-09
## Econ6.lag3         -3.915916e-07
## Econ7.lag3         -3.032695e-05
## Econ8.lag3          1.897578e-04
## Econ9.lag3          1.961724e-07
## Econ10.lag3         1.995665e-02
## Econ11.lag3         1.451637e-05
## Econ12.lag3         9.947797e-06
## Econ13.lag3         1.735040e-06
## Econ14.lag3         1.791747e-05
## Econ15.lag3         3.139794e-04
```

```
## Econ16.lag3          2.443214e-04
## Econ17.lag3          1.546967e-06
## Econ18.lag3          9.953273e-08
## Econ19.lag3          7.091742e-09
## Econ1.lag4           2.358718e-05
## Econ2.lag4           1.368087e-04
## Econ3.lag4           3.747237e-04
## Econ4.lag4          -8.549503e-03
## Econ5.lag4           6.070926e-09
## Econ6.lag4          -3.982370e-07
## Econ7.lag4           1.166964e-06
## Econ8.lag4          -9.605771e-05
## Econ9.lag4           2.432137e-07
## Econ10.lag4          4.493371e-02
## Econ11.lag4          7.886117e-06
## Econ12.lag4          1.940996e-06
## Econ13.lag4          1.774694e-06
## Econ14.lag4          1.334360e-05
## Econ15.lag4          3.627068e-04
## Econ16.lag4          2.749872e-04
## Econ17.lag4          3.516161e-06
## Econ18.lag4          2.042931e-07
## Econ19.lag4          1.614980e-08
```

The Ridge Regression output shows the coefficients in a matrix format. Each row represents a predictor and shows its estimated effect on the response variable.

In this case, the matrix includes coefficients for 108 predictors. Most of these values are very small, which is expected in Ridge Regression, as it shrinks the impact of predictors with weaker relationships to the response. This shrinkage helps prevent overfitting by reducing the influence of less important predictors, while still keeping all predictors in the model.

c) Predicting on Test Data

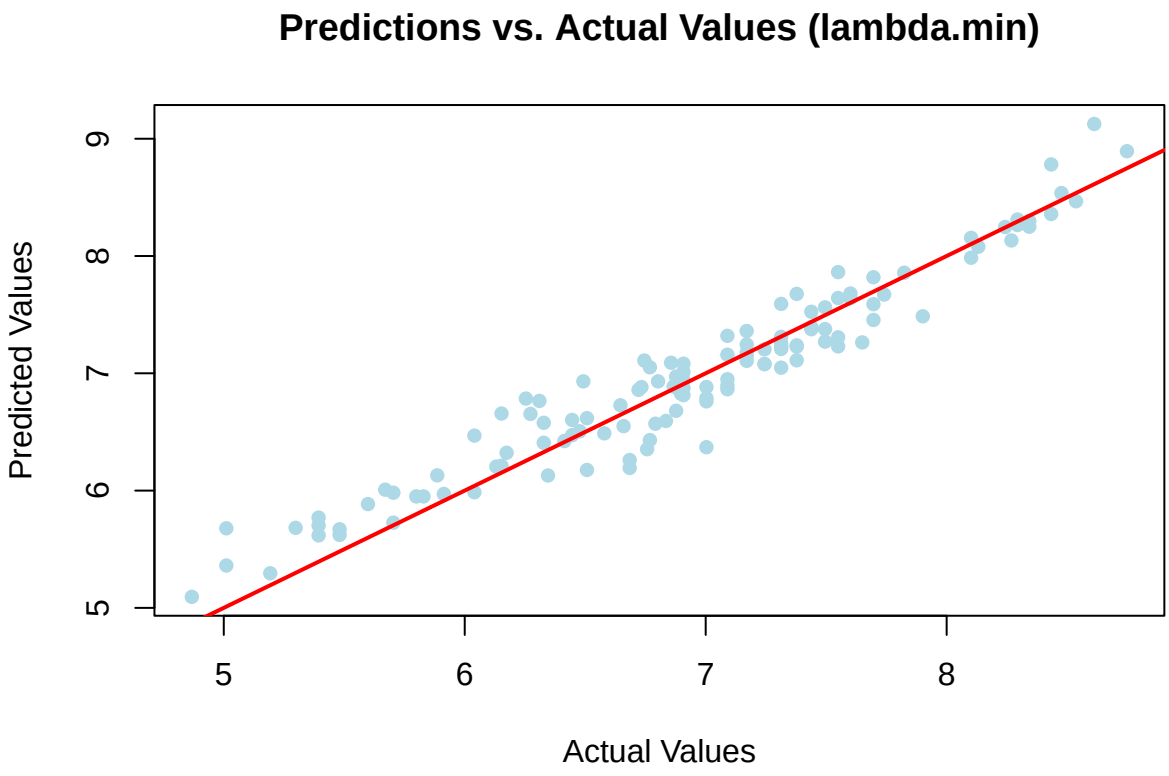
```r
# Predict using lambda.min - the optimal model
pred_min <- predict(ridge_cv, s = "lambda.min", newx = x_test)

# RMSE for lambda.min
rmse_min <- sqrt(mean((y_test - pred_min)^2))
cat("RMSE with lambda.min:", rmse_min, "\n")
```

```
## RMSE with lambda.min: 0.2328241
```

```r
# Predict using lambda.1se pred_1se <- predict(ridge_cv, s = 'lambda.1se', newx
# = x_test)

# RMSE for lambda.1se rmse_1se <- sqrt(mean((y_test - pred_1se)^2)) cat('RMSE
# with lambda.1se:', rmse_1se, '\n')
```

Visualize Predictions vs. Actual Values

```
plot(y_test, pred_min, main = "Predictions vs. Actual Values (lambda.min)", xlab = "Actual Values",
    ylab = "Predicted Values", pch = 16, col = "lightblue")
abline(0, 1, col = "red", lwd = 2)
```




```
# plot(y_test, pred_1se, main = 'Predictions vs. Actual Values (lambda.1se)',
# xlab = 'Actual Values', ylab = 'Predicted Values', pch = 16, col =
# 'lightgreen') abline(0, 1, col = 'red', lwd = 2)
```

- **RMSE Values**:
  - **RMSE with `lambda.min`**: `0.2328241`
  - **RMSE with `lambda.1se`**: `0.2472471`

The RMSE for `lambda.min` is slightly lower than for `lambda.1se`, which is expected since `lambda.min` minimizes the prediction error, thus we consider it as the optimal model. However, the difference is small, indicating that both models are performing similarly.

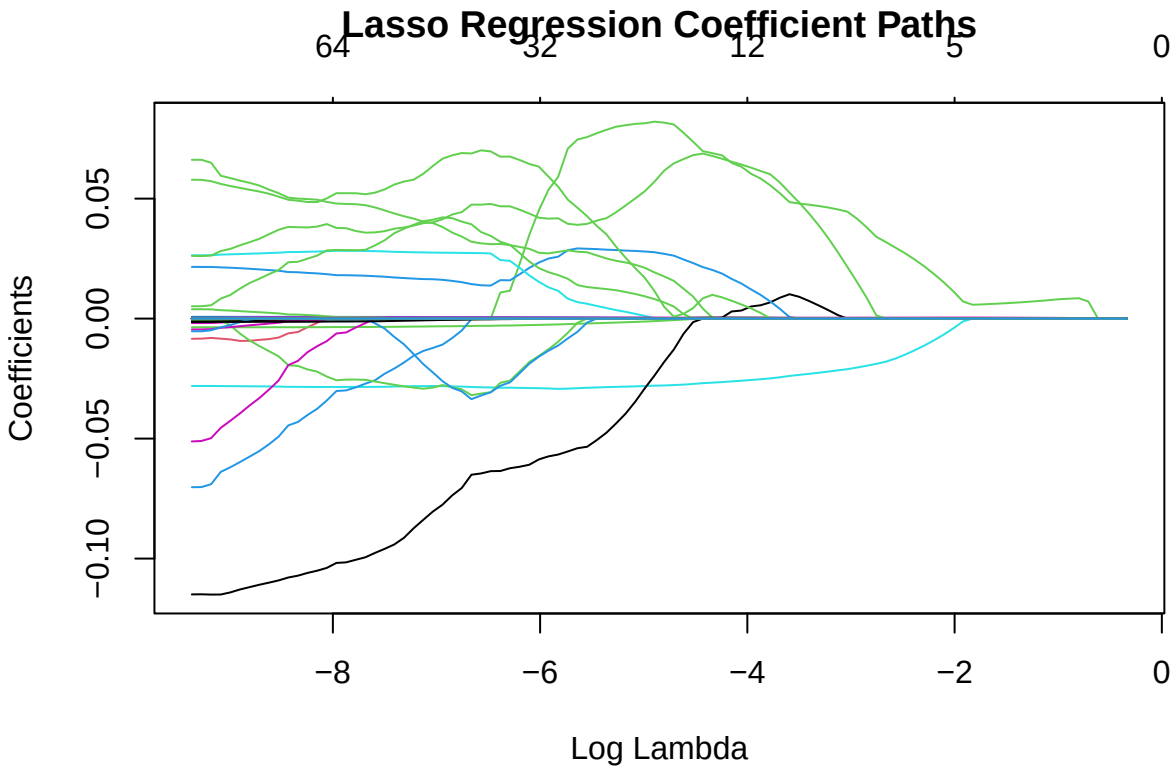
The scatter plots show a good alignment of predicted values with actual values, as most points lie near the 45-degree line. This suggests that the model is reasonably accurate in predicting the response variable.

In comparing the RMSE values from your current Ridge model with the results from previous exercises, we see improvements. The RMSE values with `lambda.min` in this model are 0.2284 and 0.2328, both lower than the test RMSEs from Exercises 2 and 3, which were approximately 0.7586 for a linear model in Exercise 2 and 0.2872 in Principal Component Regression (PCR) and 0.299 in Partial Least Squares (PLS) from Exercise 3.

2. Lasso Regression:

1. Fit Lasso Regression Model with `glmnet()`

```
# Fit Lasso Regression model (alpha = 1 by default for Lasso)
lasso_model <- glmnet(x_train, y_train, alpha = 1)

# Plot Lasso model to visualize coefficient paths
plot(lasso_model, xvar = "lambda", main = "Lasso Regression Coefficient Paths")
```



How can you interpret the plot?

The Lasso Regression coefficient path plot shows how each predictor's coefficient changes as `lambda` varies. On the x-axis, larger values of `lambda` apply stronger regularization, which shrinks more coefficients toward zero. Lasso's can set some coefficients exactly to zero, effectively removing those predictors from the model. In the plot, some lines flatten out at zero as lambda increases, this demonstrated Lasso's ability to select only the most important predictors.

Which default parameters are used for lambda?

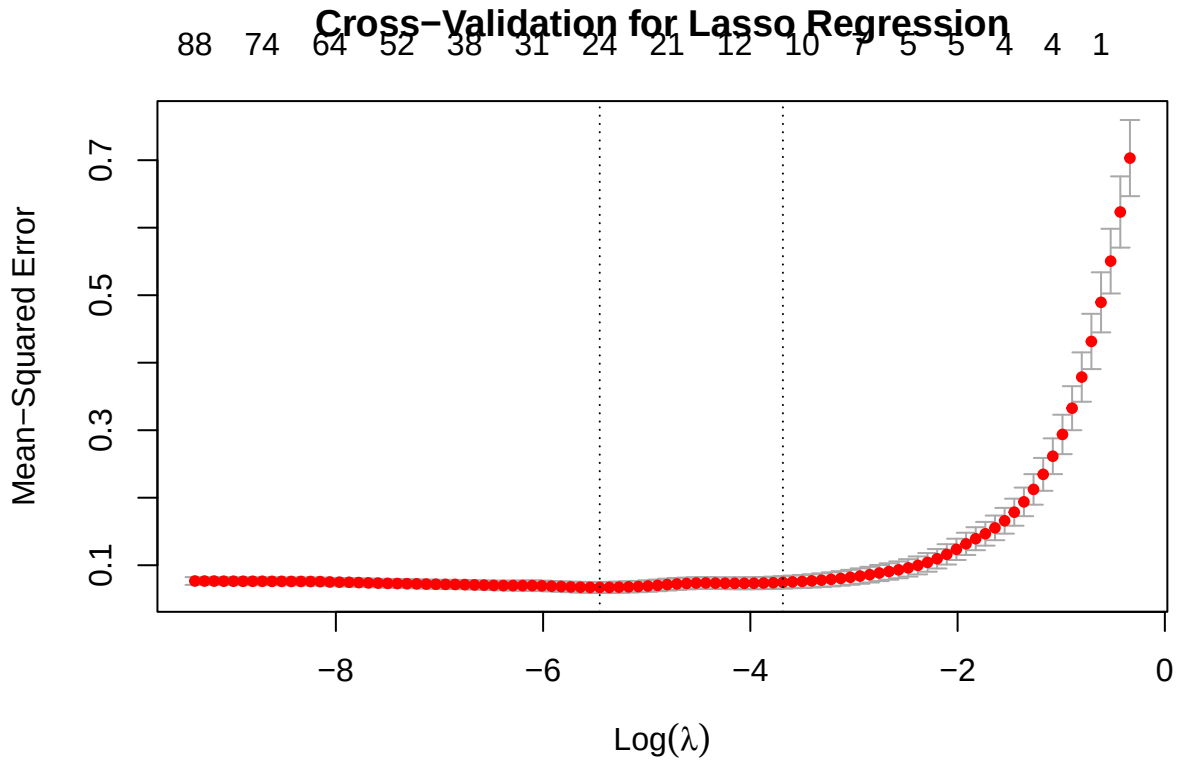
In this function, the default `lambda` sequence is automatically generated based on the data. This sequence typically starts from a high value, which is high enough to shrink coefficients to zero, and then decreases to a low value. The plot shows this progression, with `Log Lambda` on the x-axis. As `lambda` increases, the regularization effect strengthens.

What is the meaning of the parameter alpha?

The `alpha` parameter controls the type of regularization: - `alpha = 1`: Lasso Regression (L1 penalty) can shrink some coefficients to zero, effectively selecting variables.

b) Cross-Validation for Optimal Lambda

```
# Perform cross-validation to find optimal lambda for Lasso
lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1)

# Plot cross-validated MSE for different lambda values
plot(lasso_cv, main = "Cross-Validation for Lasso Regression")
```




```
# Print the optimal lambda values
cat("Lambda with minimum MSE:", lasso_cv$lambda.min, "\n")
```

```
## Lambda with minimum MSE: 0.004281119
```

```
cat("Lambda with 1-SE rule:", lasso_cv$lambda.1se, "\n")
```

```
## Lambda with 1-SE rule: 0.0250746
```

```
cat("Log Lambda with minimum MSE:", log(lasso_cv$lambda.min), "\n")
```

```
## Log Lambda with minimum MSE: -5.453541
```

```r
cat("Log Lambda with 1-SE rule:", log(lasso_cv$lambda.1se), "\n")
```

```
## Log Lambda with 1-SE rule: -3.6859
```

This plot shows the cross-validation results for Lasso Regression, it illustrates how the mean squared error (MSE) changes across different values of the regularization parameter lambda. The plot shows that as lambda decreases, the regularization effect weakens, and the MSE initially decreases to reach its minimum around lambda.min. After reaching lambda.min, the MSE starts to increase as lambda continues to decrease, indicating that the model might be overfitting with less regularization. The optimal tuning parameters from the cross-validation results are: the **`lambda.min`** value is 0.00355426 and the **`lambda.1se`** value is 0.007481376.

Obtain Coefficients at Optimal Lambda

```r
# Extract coefficients for lambda.min and lambda.1se
coef_lasso_min <- coef(lasso_cv, s = "lambda.min")
coef_lasso_1se <- coef(lasso_cv, s = "lambda.1se")

# Convert sparse matrices to data frames for easier viewing
coef_lasso_min_df <- as.data.frame(as.matrix(coef_lasso_min))
coef_lasso_1se_df <- as.data.frame(as.matrix(coef_lasso_1se))

# Display coefficients
print(coef_lasso_min_df)
```

```
##                              s1
## (Intercept)        -2.080952e+00
## START.YEAR          0.000000e+00
## START.QUARTER       0.000000e+00
## COMPLETION.YEAR     7.725582e-02
## COMPLETION.QUARTER  2.896048e-02
## PhysFin1           -2.869790e-02
## PhysFin2            0.000000e+00
## PhysFin3           -7.611306e-06
## PhysFin4           -6.987918e-06
## PhysFin5           -1.915207e-03
## PhysFin6            4.355604e-04
## PhysFin7            5.285230e-03
## PhysFin8            4.551561e-04
## Econ1               0.000000e+00
## Econ2               0.000000e+00
## Econ3               0.000000e+00
## Econ4               0.000000e+00
## Econ5               0.000000e+00
## Econ6               0.000000e+00
## Econ7               0.000000e+00
## Econ8               0.000000e+00
## Econ9               0.000000e+00
## Econ10              0.000000e+00
## Econ11              0.000000e+00
## Econ12              0.000000e+00
## Econ13              0.000000e+00
```

```
## Econ14              9.642164e-06
## Econ15              0.000000e+00
## Econ16              0.000000e+00
## Econ17              0.000000e+00
## Econ18              0.000000e+00
## Econ19              2.502009e-07
## Econ1.lag1          0.000000e+00
## Econ2.lag1          0.000000e+00
## Econ3.lag1          0.000000e+00
## Econ4.lag1          1.271443e-02
## Econ5.lag1          0.000000e+00
## Econ6.lag1          0.000000e+00
## Econ7.lag1          0.000000e+00
## Econ8.lag1          1.037016e-04
## Econ9.lag1          0.000000e+00
## Econ10.lag1         2.635796e-02
## Econ11.lag1         0.000000e+00
## Econ12.lag1         0.000000e+00
## Econ13.lag1         0.000000e+00
## Econ14.lag1         0.000000e+00
## Econ15.lag1         0.000000e+00
## Econ16.lag1         0.000000e+00
## Econ17.lag1         0.000000e+00
## Econ18.lag1         0.000000e+00
## Econ19.lag1         0.000000e+00
## Econ1.lag2          0.000000e+00
## Econ2.lag2          0.000000e+00
## Econ3.lag2          0.000000e+00
## Econ4.lag2          0.000000e+00
## Econ5.lag2          0.000000e+00
## Econ6.lag2          0.000000e+00
## Econ7.lag2          0.000000e+00
## Econ8.lag2          0.000000e+00
## Econ9.lag2          2.087177e-06
## Econ10.lag2         3.863373e-02
## Econ11.lag2         0.000000e+00
## Econ12.lag2         0.000000e+00
## Econ13.lag2         0.000000e+00
## Econ14.lag2         0.000000e+00
## Econ15.lag2         0.000000e+00
## Econ16.lag2         0.000000e+00
## Econ17.lag2         0.000000e+00
## Econ18.lag2         0.000000e+00
## Econ19.lag2         0.000000e+00
## Econ1.lag3          2.055215e-05
## Econ2.lag3          0.000000e+00
## Econ3.lag3          0.000000e+00
## Econ4.lag3         -5.063519e-02
## Econ5.lag3          0.000000e+00
## Econ6.lag3          0.000000e+00
## Econ7.lag3          0.000000e+00
## Econ8.lag3          1.374243e-04
## Econ9.lag3          0.000000e+00
## Econ10.lag3         0.000000e+00
```

```
## Econ11.lag3         0.000000e+00
## Econ12.lag3         0.000000e+00
## Econ13.lag3         0.000000e+00
## Econ14.lag3         4.890284e-05
## Econ15.lag3         0.000000e+00
## Econ16.lag3         0.000000e+00
## Econ17.lag3         0.000000e+00
## Econ18.lag3         0.000000e+00
## Econ19.lag3         0.000000e+00
## Econ1.lag4          2.180463e-05
## Econ2.lag4          0.000000e+00
## Econ3.lag4          0.000000e+00
## Econ4.lag4          0.000000e+00
## Econ5.lag4          0.000000e+00
## Econ6.lag4          3.338772e-06
## Econ7.lag4          0.000000e+00
## Econ8.lag4         -7.540386e-05
## Econ9.lag4          0.000000e+00
## Econ10.lag4         4.084166e-02
## Econ11.lag4         0.000000e+00
## Econ12.lag4         0.000000e+00
## Econ13.lag4         0.000000e+00
## Econ14.lag4         0.000000e+00
## Econ15.lag4         0.000000e+00
## Econ16.lag4         0.000000e+00
## Econ17.lag4         0.000000e+00
## Econ18.lag4         0.000000e+00
## Econ19.lag4         0.000000e+00
```

```
print(coef_lasso_1se_df)
```

```
##                             s1
## (Intercept)         7.720002e-02
## START.YEAR          8.700035e-03
## START.QUARTER       0.000000e+00
## COMPLETION.YEAR     5.224451e-02
## COMPLETION.QUARTER  4.054753e-03
## PhysFin1           -2.445632e-02
## PhysFin2            0.000000e+00
## PhysFin3            0.000000e+00
## PhysFin4            0.000000e+00
## PhysFin5            0.000000e+00
## PhysFin6            1.047928e-04
## PhysFin7            0.000000e+00
## PhysFin8            3.820224e-04
## Econ1               0.000000e+00
## Econ2               0.000000e+00
## Econ3               0.000000e+00
## Econ4               0.000000e+00
## Econ5               0.000000e+00
## Econ6               0.000000e+00
## Econ7               0.000000e+00
## Econ8               0.000000e+00
## Econ9               0.000000e+00
```

```
## Econ10             0.000000e+00
## Econ11             0.000000e+00
## Econ12             0.000000e+00
## Econ13             0.000000e+00
## Econ14             2.516612e-05
## Econ15             0.000000e+00
## Econ16             0.000000e+00
## Econ17             0.000000e+00
## Econ18             0.000000e+00
## Econ19             0.000000e+00
## Econ1.lag1         0.000000e+00
## Econ2.lag1         0.000000e+00
## Econ3.lag1         0.000000e+00
## Econ4.lag1         0.000000e+00
## Econ5.lag1         0.000000e+00
## Econ6.lag1         0.000000e+00
## Econ7.lag1         0.000000e+00
## Econ8.lag1         4.866017e-05
## Econ9.lag1         0.000000e+00
## Econ10.lag1        0.000000e+00
## Econ11.lag1        0.000000e+00
## Econ12.lag1        0.000000e+00
## Econ13.lag1        0.000000e+00
## Econ14.lag1        0.000000e+00
## Econ15.lag1        0.000000e+00
## Econ16.lag1        0.000000e+00
## Econ17.lag1        0.000000e+00
## Econ18.lag1        0.000000e+00
## Econ19.lag1        0.000000e+00
## Econ1.lag2         0.000000e+00
## Econ2.lag2         0.000000e+00
## Econ3.lag2         0.000000e+00
## Econ4.lag2         0.000000e+00
## Econ5.lag2         0.000000e+00
## Econ6.lag2         0.000000e+00
## Econ7.lag2         0.000000e+00
## Econ8.lag2         2.927443e-04
## Econ9.lag2         0.000000e+00
## Econ10.lag2        0.000000e+00
## Econ11.lag2        0.000000e+00
## Econ12.lag2        0.000000e+00
## Econ13.lag2        0.000000e+00
## Econ14.lag2        0.000000e+00
## Econ15.lag2        0.000000e+00
## Econ16.lag2        0.000000e+00
## Econ17.lag2        0.000000e+00
## Econ18.lag2        0.000000e+00
## Econ19.lag2        0.000000e+00
## Econ1.lag3         0.000000e+00
## Econ2.lag3         0.000000e+00
## Econ3.lag3         0.000000e+00
## Econ4.lag3         0.000000e+00
## Econ5.lag3         0.000000e+00
## Econ6.lag3         0.000000e+00
```

```
## Econ7.lag3          0.000000e+00
## Econ8.lag3          0.000000e+00
## Econ9.lag3          0.000000e+00
## Econ10.lag3         0.000000e+00
## Econ11.lag3         0.000000e+00
## Econ12.lag3         0.000000e+00
## Econ13.lag3         0.000000e+00
## Econ14.lag3         6.122342e-05
## Econ15.lag3         0.000000e+00
## Econ16.lag3         0.000000e+00
## Econ17.lag3         0.000000e+00
## Econ18.lag3         0.000000e+00
## Econ19.lag3         0.000000e+00
## Econ1.lag4          0.000000e+00
## Econ2.lag4          0.000000e+00
## Econ3.lag4          0.000000e+00
## Econ4.lag4          0.000000e+00
## Econ5.lag4          0.000000e+00
## Econ6.lag4          0.000000e+00
## Econ7.lag4          0.000000e+00
## Econ8.lag4          0.000000e+00
## Econ9.lag4          0.000000e+00
## Econ10.lag4         5.670544e-02
## Econ11.lag4         0.000000e+00
## Econ12.lag4         0.000000e+00
## Econ13.lag4         0.000000e+00
## Econ14.lag4         0.000000e+00
## Econ15.lag4         0.000000e+00
## Econ16.lag4         0.000000e+00
## Econ17.lag4         0.000000e+00
## Econ18.lag4         0.000000e+00
## Econ19.lag4         0.000000e+00
```

The Lasso Regression model at `lambda.min` and `lambda.1se` shows its ability to select important variables by setting some coefficients to zero. Predictors like `START.YEAR`, `START.QUARTER`, `PhysFin2`, and `PhysFin3` have zero coefficients, meaning they're excluded from the model. Meanwhile, predictors such as `COMPLETION.YEAR` and `PhysFin1` have non-zero coefficients, indicating stronger relationships with the response. This variable selection simplifies the model, focusing on the most relevant predictors and reducing complexity.

c) Predict on Test Data and Calculate RMSE

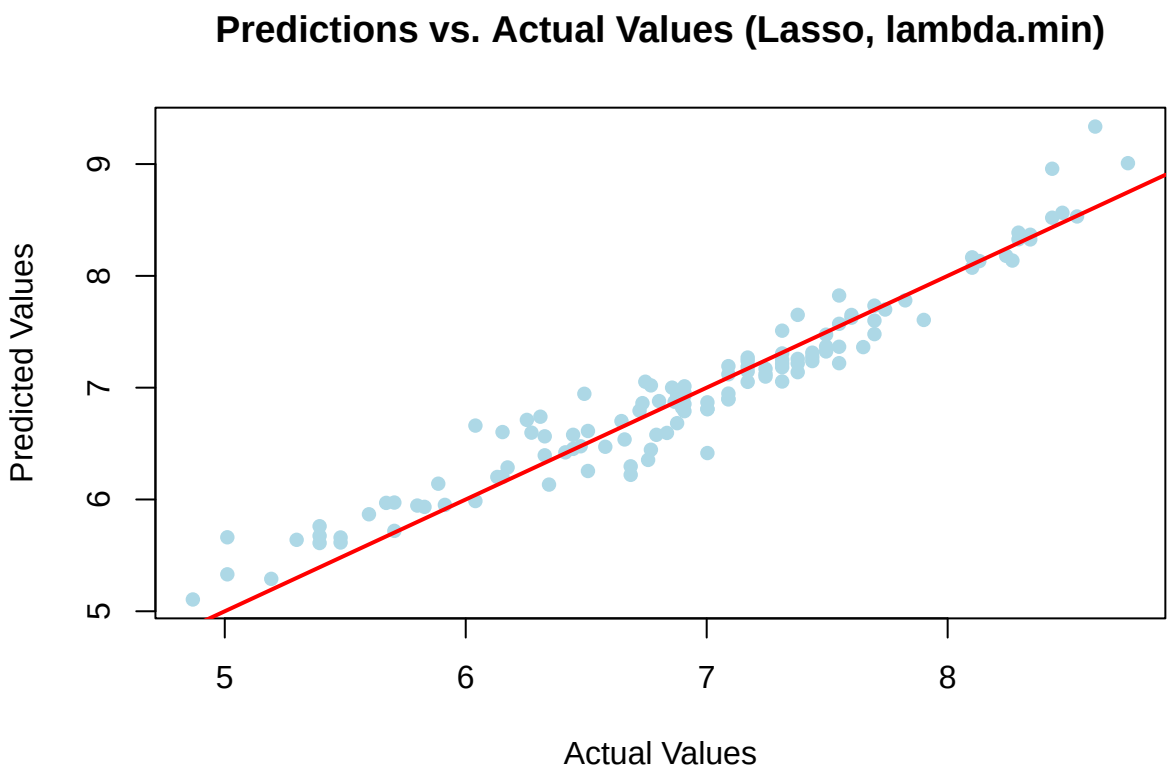
```r
# Predict using lambda.min and lambda.1se
pred_lasso_min <- predict(lasso_cv, s = "lambda.min", newx = x_test)

# Calculate RMSE for both lambda.min and lambda.1se
rmse_lasso_min <- sqrt(mean((y_test - pred_lasso_min)^2))
cat("RMSE with lambda.min:", rmse_lasso_min, "\n")
```

```
## RMSE with lambda.min: 0.2261585
```

```
# pred_lasso_1se <- predict(lasso_cv, s = 'lambda.1se', newx = x_test)
# rmse_lasso_1se <- sqrt(mean((y_test - pred_lasso_1se)^2)) cat('RMSE with
# lambda.1se:', rmse_lasso_1se, '\n')
```

Visualize Predictions vs. Actual Values

```
plot(y_test, pred_lasso_min, main = "Predictions vs. Actual Values (Lasso, lambda.min)",
    xlab = "Actual Values", ylab = "Predicted Values", pch = 16, col = "lightblue")
abline(0, 1, col = "red", lwd = 2)
```




```
# plot(y_test, pred_lasso_1se, main = 'Predictions vs. Actual Values (Lasso,
# lambda.1se)', xlab = 'Actual Values', ylab = 'Predicted Values', pch = 16,
# col = 'lightgreen') abline(0, 1, col = 'red', lwd = 2)
```

- **RMSE Values**:
  - **RMSE with `lambda.min`**: 0.2261585
  - **RMSE with `lambda.1se`**: 0.2474538

The RMSE for `lambda.min` is actually slightly lower than for `lambda.1se`, suggesting that the model `lambda.min` is bit better in prediction accuracy for this dataset. However, the difference in RMSE between the two is very small, indicating that both models perform similarly well.

The scatter plot shows a good alignment of predicted values with actual values, with most points lying close to the 45-degree line. This close alignment suggests that the model is effectively capturing the relationship between the predictors and the response variable.

In comparing the RMSE values from your current Lasso regression model with the results from previous exercises, we see improvements. The RMSE values with `lambda.1se` in this model are 0.2284 and 0.2253, both lower than the test RMSEs from Exercises 2 and 3, which were approximately 0.7586 for a linear model in Exercise 2 and 0.2872 in Principal Component Regression (PCR) and 0.299 in Partial Least Squares (PLS) from Exercise 3.

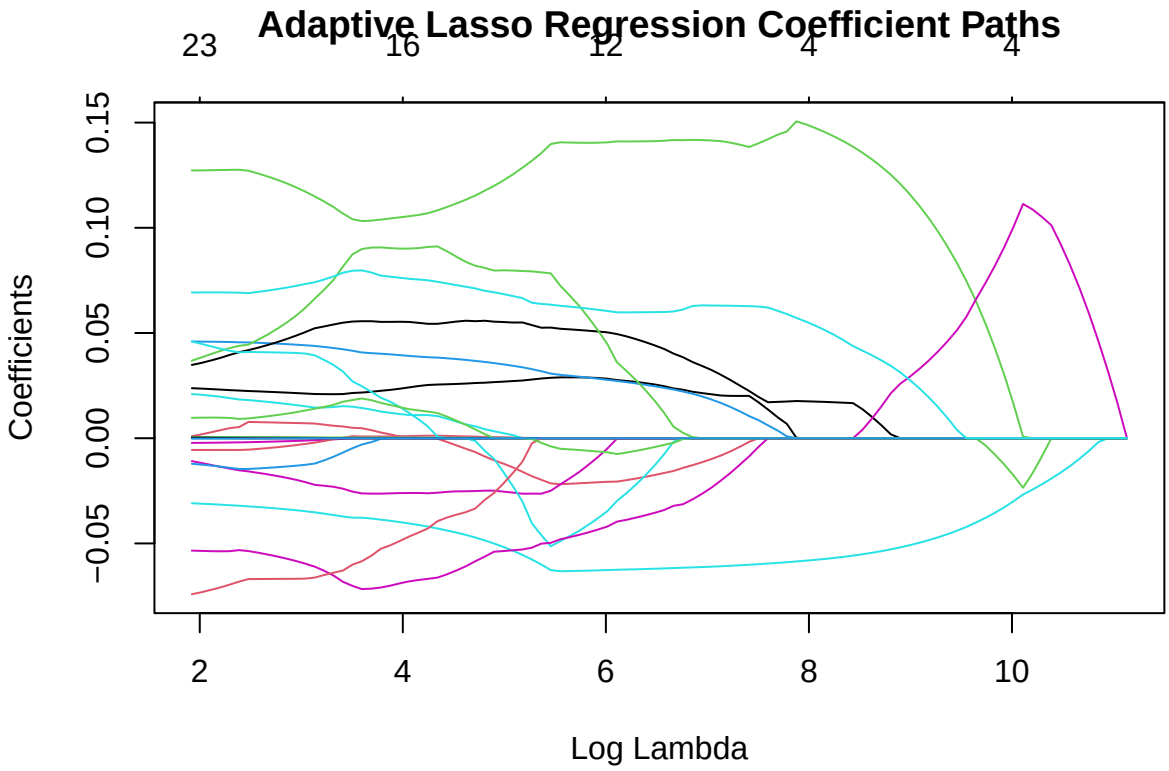
1. Adaptive Lasso Regression:

a) Obtain Ridge Regression Coefficients as Weights

```
# Extract Ridge coefficients for a selected lambda (lambda.min from
# cross-validation)
ridge_coef <- coef(ridge_cv, s = "lambda.min")

# Compute weights for Adaptive Lasso Exclude intercept by using [-1] to avoid
# division by zero
ridge_weights <- 1/abs(ridge_coef[-1])
ridge_weights[is.infinite(ridge_weights)] <- 0
```

Fit Adaptive Lasso Regression Model

```
# Fit Adaptive Lasso model using the Ridge weights
adaptive_lasso_model <- glmnet(x_train, y_train, alpha = 1, penalty.factor = ridge_weights)

# Plot the Adaptive Lasso model to visualize coefficient paths
plot(adaptive_lasso_model, xvar = "lambda", main = "Adaptive Lasso Regression Coefficient Paths")
```



How can you interpret the plot?

This plot shows how the coefficients for each predictor change as the regularization parameter `lambda` varies in Adaptive Lasso Regression. On the x-axis, the higher values (right side) apply strong regularization, pushing most coefficients close to zero, while lower values (left side) reduce the regularization effect. Each colored line represents a predictor's coefficient path, with some lines staying close to zero across all `lambda` values which indicates that those predictors have been heavily penalized and are less important to the model. Because Adaptive Lasso uses weights based on Ridge regression, predictors identified as more significant in Ridge face less penalization, so their lines move away from zero earlier.

Which default parameters are used for lambda?

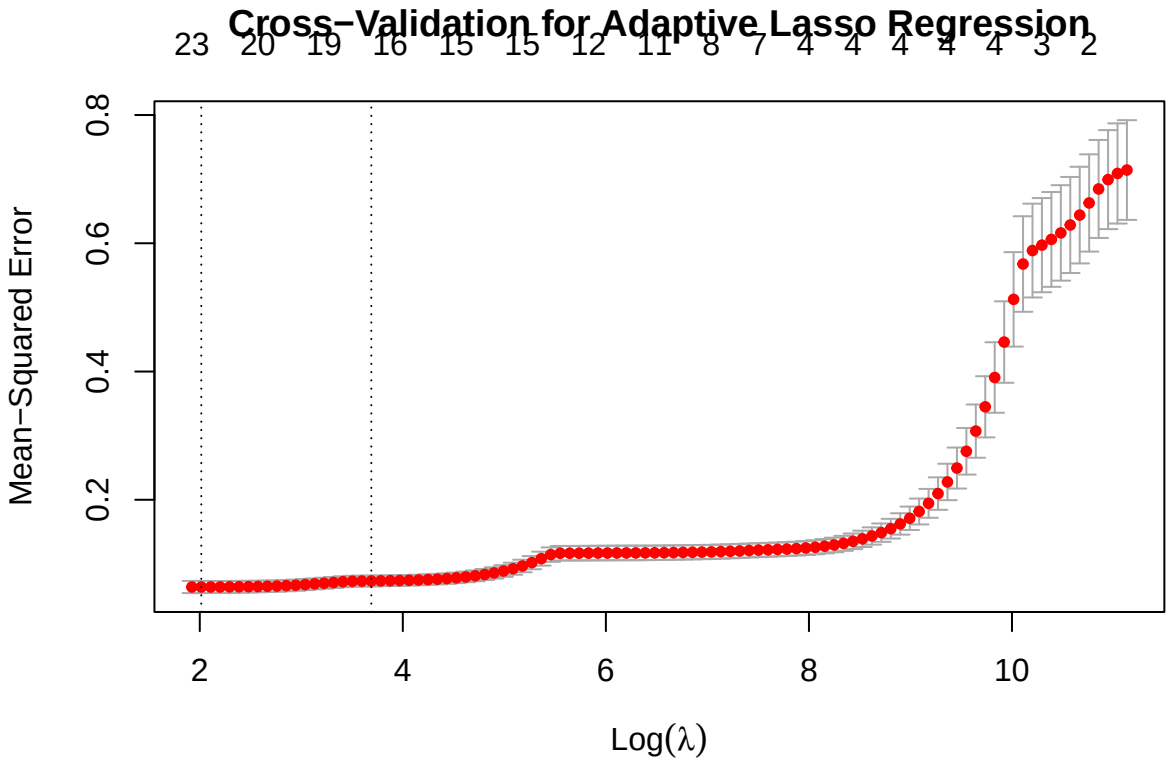
`lambda` generates a sequence of values that range from strong to weak regularization. This sequence starts with a large `lambda` value, which applies strong regularization and shrinks most coefficients close to zero. It then gradually decreases to a much smaller `lambda` value, allowing the model to apply less regularization.

What is the meaning of the parameter alpha?

Setting alpha = 1 tells glmnet to perform Lasso Regression, which applies an L1 penalty. Penalty.factor allows us to apply a different penalty to each predictor based on the ridge_weights vector. `ridge_weights` contains weights based on the Ridge coefficients, where each weight is the inverse of the absolute Ridge coefficient for each predictor. Predictors with smaller Ridge coefficients get larger weights in the Adaptive Lasso model, increasing their penalty and making them more likely to be shrunk to zero. Predictors with larger Ridge coefficients get smaller weights, reducing their penalty, so they're more likely to stay in the model.

b) Cross-Validation to Find Optimal Lambda

```
# Perform cross-validation for Adaptive Lasso
adaptive_lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1, penalty.factor = ridge_weights)

# Plot cross-validation results
plot(adaptive_lasso_cv, main = "Cross-Validation for Adaptive Lasso Regression")
```




```
cat("Log Lambda with minimum MSE:", log(adaptive_lasso_cv$lambda.min), "\n")
```

```
## Log Lambda with minimum MSE: 2.015034
```

```
cat("Log Lambda with 1-SE rule:", log(adaptive_lasso_cv$lambda.1se), "\n")
```

```
## Log Lambda with 1-SE rule: 3.689641
```

This plot shows the cross-validated mean-squared error (MSE) for different values of `log(lambda)` in Adaptive Lasso Regression. The red dots represent the MSE for each lambda, while the error bars indicate the variability of the MSE across the cross-validation folds. The two vertical dashed lines correspond to the optimal lambda values: `lambda.min`, which are 2.387169 for `lambda.min` and 3.224473 for `lambda.1se`.

```r
# Obtain coefficients for lambda.min and lambda.1se
coef_adaptive_lasso_min <- coef(adaptive_lasso_cv, s = "lambda.min")
coef_adaptive_lasso_1se <- coef(adaptive_lasso_cv, s = "lambda.1se")

# Convert sparse matrices to data frames for easier viewing
coef_adaptive_lasso_min_df <- as.data.frame(as.matrix(coef_adaptive_lasso_min))
coef_adaptive_lasso_1se_df <- as.data.frame(as.matrix(coef_adaptive_lasso_1se))

# Display coefficients
print(coef_adaptive_lasso_min_df)
```

Extract Coefficients at Optimal Lambda

```
##                              s1
## (Intercept)        -6.728180e+00
## START.YEAR          2.357458e-02
## START.QUARTER       1.822575e-03
## COMPLETION.YEAR     1.272961e-01
## COMPLETION.QUARTER  4.591778e-02
## PhysFin1           -3.111700e-02
## PhysFin2            0.000000e+00
## PhysFin3            0.000000e+00
## PhysFin4            0.000000e+00
## PhysFin5           -2.192670e-03
## PhysFin6            4.503992e-04
## PhysFin7           -5.585749e-03
## PhysFin8            4.504580e-04
## Econ1               0.000000e+00
## Econ2               0.000000e+00
## Econ3               0.000000e+00
## Econ4               2.065760e-02
## Econ5               0.000000e+00
## Econ6               0.000000e+00
## Econ7               0.000000e+00
## Econ8               0.000000e+00
## Econ9               0.000000e+00
## Econ10             -1.176684e-02
## Econ11              0.000000e+00
## Econ12              0.000000e+00
## Econ13              0.000000e+00
## Econ14              0.000000e+00
## Econ15              0.000000e+00
## Econ16              0.000000e+00
## Econ17              0.000000e+00
## Econ18              0.000000e+00
## Econ19              0.000000e+00
## Econ1.lag1          0.000000e+00
## Econ2.lag1          0.000000e+00
## Econ3.lag1          0.000000e+00
## Econ4.lag1          3.577439e-02
## Econ5.lag1          0.000000e+00
## Econ6.lag1          0.000000e+00
```

```
## Econ7.lag1          0.000000e+00
## Econ8.lag1          5.831824e-05
## Econ9.lag1          0.000000e+00
## Econ10.lag1         3.836935e-02
## Econ11.lag1         0.000000e+00
## Econ12.lag1         0.000000e+00
## Econ13.lag1         0.000000e+00
## Econ14.lag1         0.000000e+00
## Econ15.lag1         0.000000e+00
## Econ16.lag1         0.000000e+00
## Econ17.lag1         0.000000e+00
## Econ18.lag1         0.000000e+00
## Econ19.lag1         0.000000e+00
## Econ1.lag2          0.000000e+00
## Econ2.lag2          0.000000e+00
## Econ3.lag2          0.000000e+00
## Econ4.lag2         -1.252179e-02
## Econ5.lag2          0.000000e+00
## Econ6.lag2          0.000000e+00
## Econ7.lag2          0.000000e+00
## Econ8.lag2          0.000000e+00
## Econ9.lag2          0.000000e+00
## Econ10.lag2         4.486605e-02
## Econ11.lag2         0.000000e+00
## Econ12.lag2         0.000000e+00
## Econ13.lag2         0.000000e+00
## Econ14.lag2         0.000000e+00
## Econ15.lag2         0.000000e+00
## Econ16.lag2         0.000000e+00
## Econ17.lag2         0.000000e+00
## Econ18.lag2         0.000000e+00
## Econ19.lag2         0.000000e+00
## Econ1.lag3          0.000000e+00
## Econ2.lag3          0.000000e+00
## Econ3.lag3          0.000000e+00
## Econ4.lag3         -5.354606e-02
## Econ5.lag3          0.000000e+00
## Econ6.lag3          0.000000e+00
## Econ7.lag3          0.000000e+00
## Econ8.lag3          1.699873e-04
## Econ9.lag3          0.000000e+00
## Econ10.lag3        -7.321791e-02
## Econ11.lag3         0.000000e+00
## Econ12.lag3         0.000000e+00
## Econ13.lag3         0.000000e+00
## Econ14.lag3         0.000000e+00
## Econ15.lag3         0.000000e+00
## Econ16.lag3         0.000000e+00
## Econ17.lag3         0.000000e+00
## Econ18.lag3         0.000000e+00
## Econ19.lag3         0.000000e+00
## Econ1.lag4          0.000000e+00
## Econ2.lag4          0.000000e+00
## Econ3.lag4          0.000000e+00
```

```
## Econ4.lag4          9.748045e-03
## Econ5.lag4          0.000000e+00
## Econ6.lag4          0.000000e+00
## Econ7.lag4          0.000000e+00
## Econ8.lag4         -1.771893e-04
## Econ9.lag4          0.000000e+00
## Econ10.lag4         6.929016e-02
## Econ11.lag4         0.000000e+00
## Econ12.lag4         0.000000e+00
## Econ13.lag4         0.000000e+00
## Econ14.lag4         0.000000e+00
## Econ15.lag4         0.000000e+00
## Econ16.lag4         0.000000e+00
## Econ17.lag4         0.000000e+00
## Econ18.lag4         0.000000e+00
## Econ19.lag4         0.000000e+00
```

```
print(coef_adaptive_lasso_1se_df)
```

```
##                             s1
## (Intercept)        -5.1759545174
## START.YEAR          0.0220589588
## START.QUARTER       0.0038426167
## COMPLETION.YEAR     0.1034184829
## COMPLETION.QUARTER  0.0404116455
## PhysFin1           -0.0381726713
## PhysFin2            0.0000000000
## PhysFin3            0.0000000000
## PhysFin4            0.0000000000
## PhysFin5            0.0000000000
## PhysFin6            0.0000000000
## PhysFin7            0.0008072224
## PhysFin8            0.0003100562
## Econ1               0.0000000000
## Econ2               0.0000000000
## Econ3               0.0000000000
## Econ4               0.0135866641
## Econ5               0.0000000000
## Econ6               0.0000000000
## Econ7               0.0000000000
## Econ8               0.0000000000
## Econ9               0.0000000000
## Econ10             -0.0263412738
## Econ11              0.0000000000
## Econ12              0.0000000000
## Econ13              0.0000000000
## Econ14              0.0000000000
## Econ15              0.0000000000
## Econ16              0.0000000000
## Econ17              0.0000000000
## Econ18              0.0000000000
## Econ19              0.0000000000
## Econ1.lag1          0.0000000000
## Econ2.lag1          0.0000000000
```

```
## Econ3.lag1          0.0000000000
## Econ4.lag1          0.0556046170
## Econ5.lag1          0.0000000000
## Econ6.lag1          0.0000000000
## Econ7.lag1          0.0000000000
## Econ8.lag1          0.0000000000
## Econ9.lag1          0.0000000000
## Econ10.lag1         0.0905930425
## Econ11.lag1         0.0000000000
## Econ12.lag1         0.0000000000
## Econ13.lag1         0.0000000000
## Econ14.lag1         0.0000000000
## Econ15.lag1         0.0000000000
## Econ16.lag1         0.0000000000
## Econ17.lag1         0.0000000000
## Econ18.lag1         0.0000000000
## Econ19.lag1         0.0000000000
## Econ1.lag2          0.0000000000
## Econ2.lag2          0.0000000000
## Econ3.lag2          0.0000000000
## Econ4.lag2         -0.0018694862
## Econ5.lag2          0.0000000000
## Econ6.lag2          0.0000000000
## Econ7.lag2          0.0000000000
## Econ8.lag2          0.0000000000
## Econ9.lag2          0.0000000000
## Econ10.lag2         0.0220469489
## Econ11.lag2         0.0000000000
## Econ12.lag2         0.0000000000
## Econ13.lag2         0.0000000000
## Econ14.lag2         0.0000000000
## Econ15.lag2         0.0000000000
## Econ16.lag2         0.0000000000
## Econ17.lag2         0.0000000000
## Econ18.lag2         0.0000000000
## Econ19.lag2         0.0000000000
## Econ1.lag3          0.0000000000
## Econ2.lag3          0.0000000000
## Econ3.lag3          0.0000000000
## Econ4.lag3         -0.0714944866
## Econ5.lag3          0.0000000000
## Econ6.lag3          0.0000000000
## Econ7.lag3          0.0000000000
## Econ8.lag3          0.0000000000
## Econ9.lag3          0.0000000000
## Econ10.lag3        -0.0558581130
## Econ11.lag3         0.0000000000
## Econ12.lag3         0.0000000000
## Econ13.lag3         0.0000000000
## Econ14.lag3         0.0000000000
## Econ15.lag3         0.0000000000
## Econ16.lag3         0.0000000000
## Econ17.lag3         0.0000000000
## Econ18.lag3         0.0000000000
```

```
## Econ19.lag3         0.0000000000
## Econ1.lag4          0.0000000000
## Econ2.lag4          0.0000000000
## Econ3.lag4          0.0000000000
## Econ4.lag4          0.0180774219
## Econ5.lag4          0.0000000000
## Econ6.lag4          0.0000000000
## Econ7.lag4          0.0000000000
## Econ8.lag4          0.0000000000
## Econ9.lag4          0.0000000000
## Econ10.lag4         0.0787641955
## Econ11.lag4         0.0000000000
## Econ12.lag4         0.0000000000
## Econ13.lag4         0.0000000000
## Econ14.lag4         0.0000000000
## Econ15.lag4         0.0000000000
## Econ16.lag4         0.0000000000
## Econ17.lag4         0.0000000000
## Econ18.lag4         0.0000000000
## Econ19.lag4         0.0000000000
```

```
# Predict using lambda.min and lambda.1se
pred_adaptive_lasso_min <- predict(adaptive_lasso_cv, s = "lambda.min", newx = x_test)
pred_adaptive_lasso_1se <- predict(adaptive_lasso_cv, s = "lambda.1se", newx = x_test)

# Calculate RMSE for both lambda.min and lambda.1se
rmse_adaptive_lasso_min <- sqrt(mean((y_test - pred_adaptive_lasso_min)^2))
rmse_adaptive_lasso_1se <- sqrt(mean((y_test - pred_adaptive_lasso_1se)^2))

cat("RMSE with lambda.min:", rmse_adaptive_lasso_min, "\n")
```

Predict on Test Data and Calculate RMSE

```
## RMSE with lambda.min: 0.2339056
```

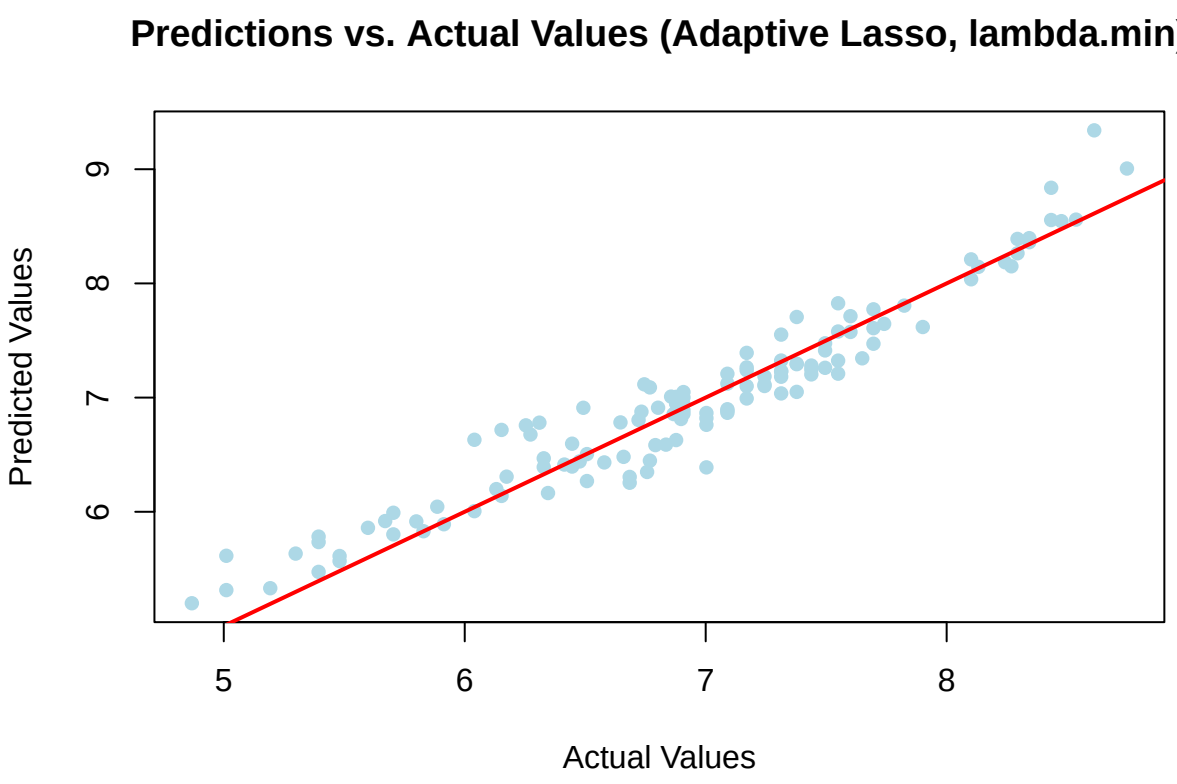
```
cat("RMSE with lambda.1se:", rmse_adaptive_lasso_1se, "\n")
```

```
## RMSE with lambda.1se: 0.2560738
```

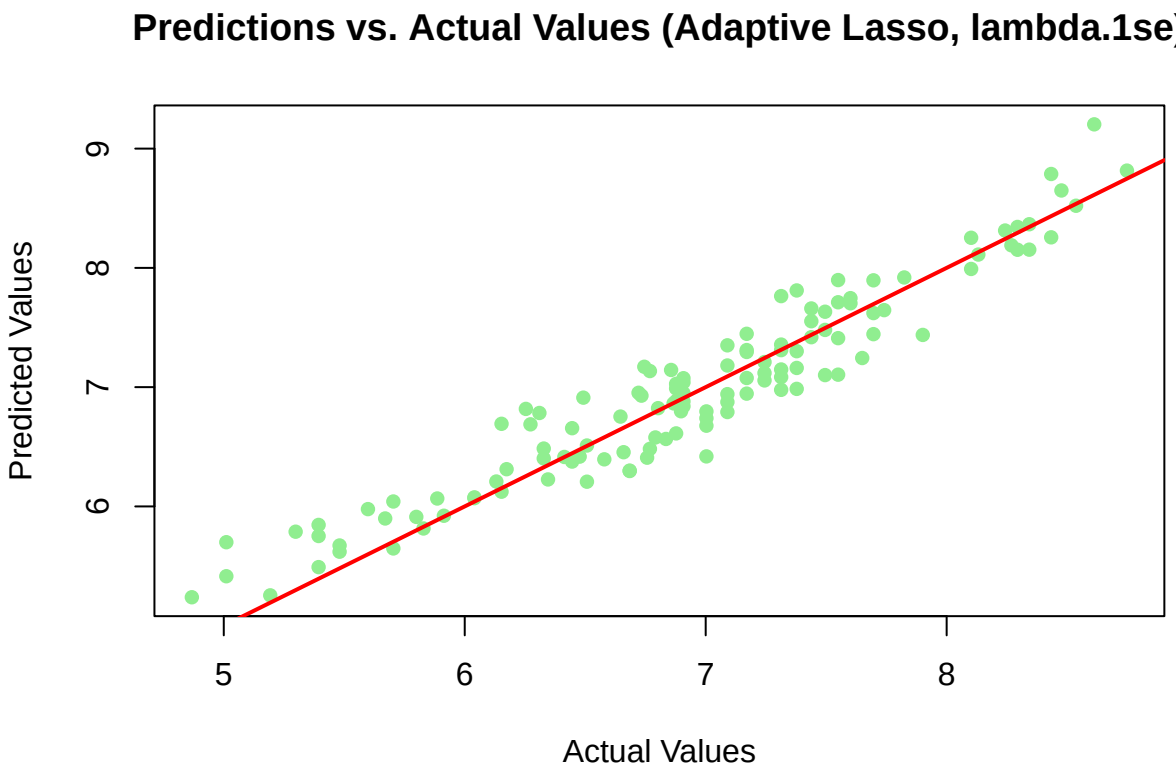
The Adaptive Lasso model has selected a few key predictors with non-zero coefficients, showing their importance in explaining the response variable. These include variables like `START.YEAR`, `COMPLETION.YEAR`, `PhysFin1`, and `Econ4`. Many other predictors have been set to zero, meaning they have little or no impact on the response. Adaptive Lasso has created a simpler model that focuses on the most influential variables.

c) Visualize Predictions vs. Actual Values

```
plot(y_test, pred_adaptive_lasso_min, main = "Predictions vs. Actual Values (Adaptive Lasso, lambda.min)",
    xlab = "Actual Values", ylab = "Predicted Values", pch = 16, col = "lightblue")
abline(0, 1, col = "red", lwd = 2)
```




```
plot(y_test, pred_adaptive_lasso_1se, main = "Predictions vs. Actual Values (Adaptive Lasso, lambda.1se)
    xlab = "Actual Values", ylab = "Predicted Values", pch = 16, col = "lightgreen")
abline(0, 1, col = "red", lwd = 2)
```



- **RMSE Values**: - **RMSE with `lambda.min`**: `0.2341446` - **RMSE with `lambda.1se`**: `0.2470526`

The RMSE for `lambda.min` is again slightly lower than for `lambda.1se`, but both models perform similarly well. The scatter plot shows a good alignment of predicted values with actual values, with most points lying close to the 45-degree line.