AMRC: Exercise 4 - 11912007

Analysis

Final performance comparison of the models:

Model	Training RMSE	Test RMSE
Full Linear Model	0.1910	2.9346
Best Subset (5 vars)	0.2950	0.3072
PCR (37 components)	0.2660	0.2390
PLS (7 components)	0.2859	0.2416
Ridge Regression	0.2243	0.2564
Lasso Regression	0.2014	0.2571
Adaptive Lasso Regression	0.2269	0.2441

Ridge Regression Analysis For Ridge regression, I used `glmnet` with `alpha=0`. The plot shows how the coefficients shrink towards zero as lambda increases. The default lambda sequence ranges from 0.072 to 16.072, with 100 values. The parameter `alpha=0` specifies Ridge regression, which uses L2 regularization.

Using cross-validation with `cv.glmnet()`, I found the optimal lambda (minimum MSE) to be 0.0716. This lambda value provides the best balance between bias and variance. The model achieved a training RMSE of 0.2243 and test RMSE of 0.2564.

Lasso Regression Analysis For Lasso regression (`alpha=1`), the coefficient paths show more variables being set exactly to zero as lambda increases. The optimal lambda from cross-validation was 0.006228, resulting in a training RMSE of 0.2014 and test RMSE of 0.2571.

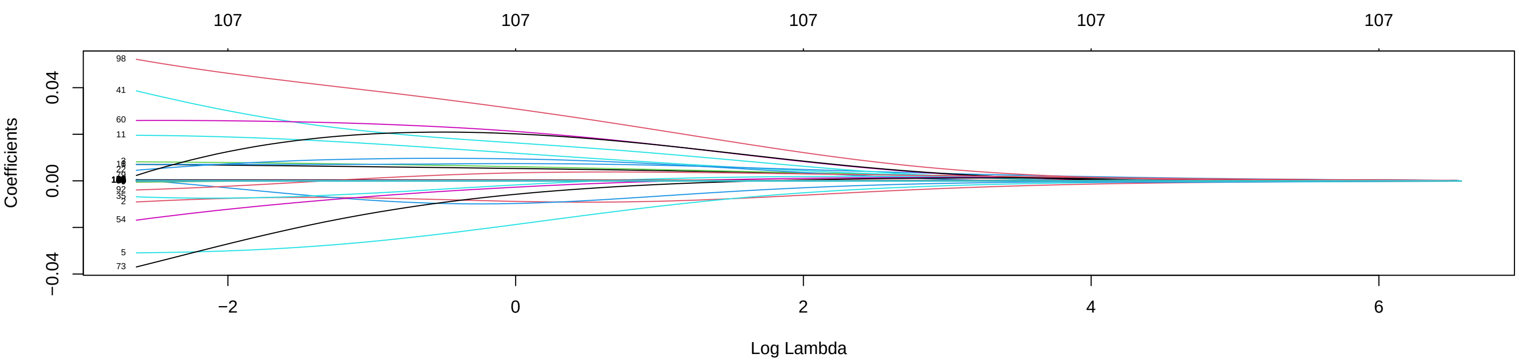
Adaptive Lasso Analysis The Adaptive Lasso used weights derived from the Ridge regression coefficients. This approach combines the stability of Ridge regression with Lasso's variable selection properties. The optimal lambda was 14.4992, yielding a training RMSE of 0.2269 and test RMSE of 0.2441.

Model Comparison Comparing the methods, Ridge regression retained all 107 variables with non-zero coefficients, while Lasso and Adaptive Lasso showed stronger variable selection with only 21 and 22 non-zero coefficients respectively. The Adaptive Lasso performed best on the test set with the lowest RMSE of 0.2441, suggesting it found a good balance between model complexity and prediction accuracy. The Adaptive Lasso seems most suitable for this dataset as it combines effective variable selection with good predictive performance.

Code

```r
# a) download
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00437/Residential-Building-Data-Set.xlsx"
temp <- tempfile(fileext = ".xlsx")
download.file(url, temp, mode = "wb")
df <- read_excel(temp)
unlink(temp)

# b) use provided .RData file
load("building.RData")
```

```r
# random 2/3 train, 1/3 test split
split_and_validate_data <- function(df) {
    n <- nrow(df)
    n_train <- round(2/3 * n)
    n_test <- n - n_train
    train_indices <- sample(1:n, n_train)
    train_data <- df[train_indices, ]
    test_data <- df[-train_indices, ]
    y_train <- train_data$y
    X_train <- train_data[, setdiff(names(train_data), "y")]
    y_test <- test_data$y
    X_test <- test_data[, setdiff(names(test_data), "y")]

    assert_that((n_train + n_test == n) && nrow(train_data) == n_train && nrow(test_data) == n_test, msg="split sizes don't add up")
    assert_that(length(intersect(train_indices, which(!1:n %in% train_indices))) == 0, msg="train and test sets overlap")
    assert_that(all(names(X_train) == names(X_test)), msg="feature names don't match between train and test")
    assert_that(length(y_train) == nrow(X_train) && length(y_test) == nrow(X_test), msg="response and feature dimensions mismatch")
    assert_that(!any(is.na(X_train)) && !any(is.na(X_test)) && !any(is.na(y_train)) && !any(is.na(y_test)), msg="missing values found in data")
    return(list(
        X_train = X_train,
        y_train = y_train,
        X_test = X_test,
        y_test = y_test
    ))
}
split_data <- split_and_validate_data(df)
X_train <- split_data$X_train
y_train <- split_data$y_train # already log transformed
X_test <- split_data$X_test
y_test <- split_data$y_test

#
# 1) ridge regression
#

# a) fit ridge regression model
X_train_matrix <- as.matrix(X_train)
X_test_matrix <- as.matrix(X_test)
ridge_model <- glmnet(X_train_matrix, y_train, alpha = 0)
plot(ridge_model, xvar="lambda", label=TRUE)
```



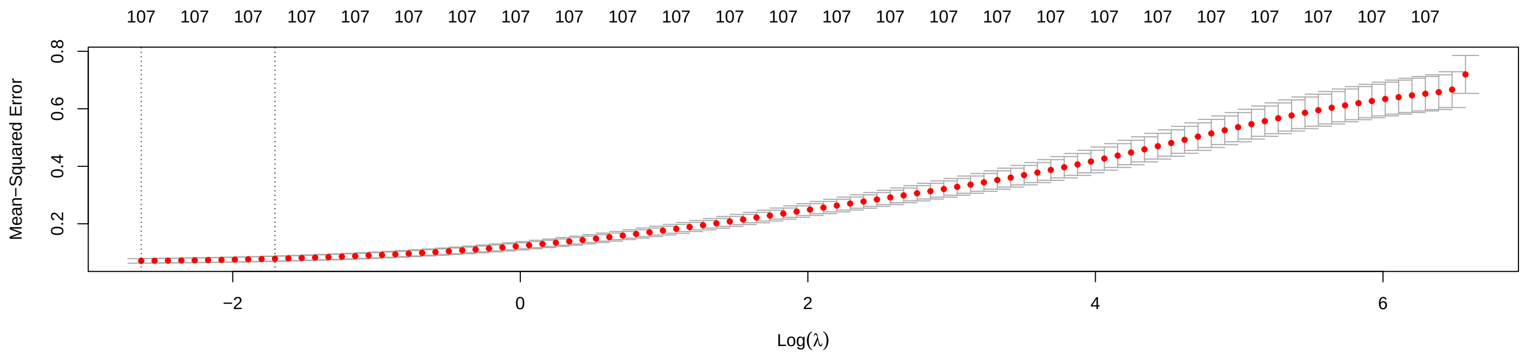

```r
cat("number of lambda values:", length(ridge_model$lambda), "\n")
```

```
## number of lambda values: 100
```

```r
cat("lambda range:", range(ridge_model$lambda), "\n")
```

```
## lambda range: 0.0716072 716.072
```

```r
# b) 10-fold cv to find optimal lambda
cv_ridge <- cv.glmnet(X_train_matrix, y_train, alpha = 0, nfolds = 10)
plot(cv_ridge)
```




```r
cat("optimal lambda (minimum MSE):", cv_ridge$lambda.min, "\n")
```

```
## optimal lambda (minimum MSE): 0.0716072
```

```r
cat("lambda within 1 SE (1 standard error rule):", cv_ridge$lambda.1se, "\n")
```

```
## lambda within 1 SE (1 standard error rule): 0.1815504
```

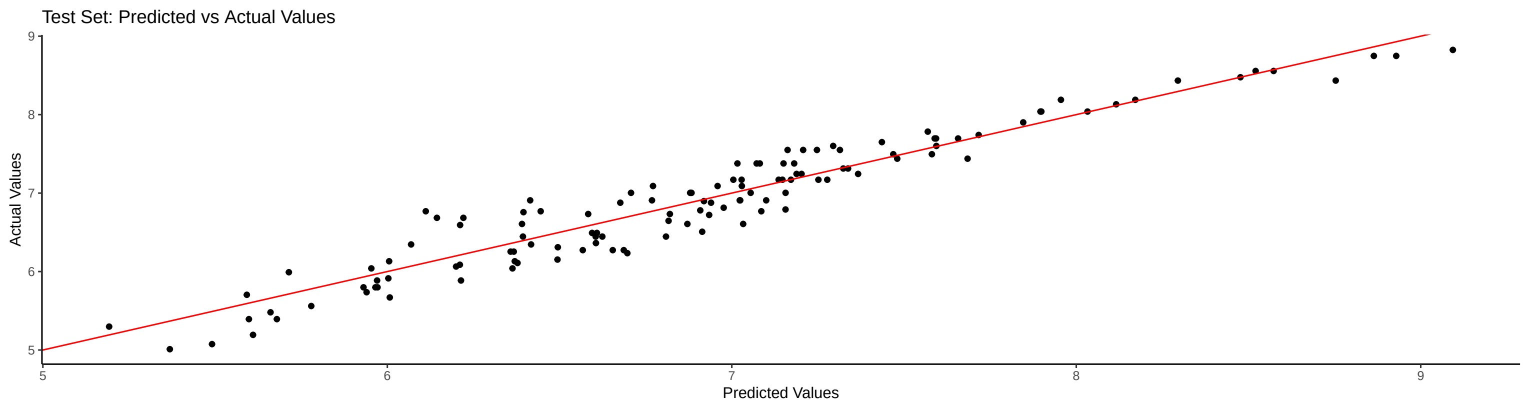
```r
ridge_coef <- coef(cv_ridge, s="lambda.min")

# c) test set performance
train_preds <- predict(ridge_model, s = cv_ridge$lambda.min, newx = X_train_matrix)
test_preds <- predict(ridge_model, s = cv_ridge$lambda.min, newx = X_test_matrix)
cat("train RMSE:", round(rmse(train_preds, y_train), 4), "\n")
```

```
## train RMSE: 0.2311
```

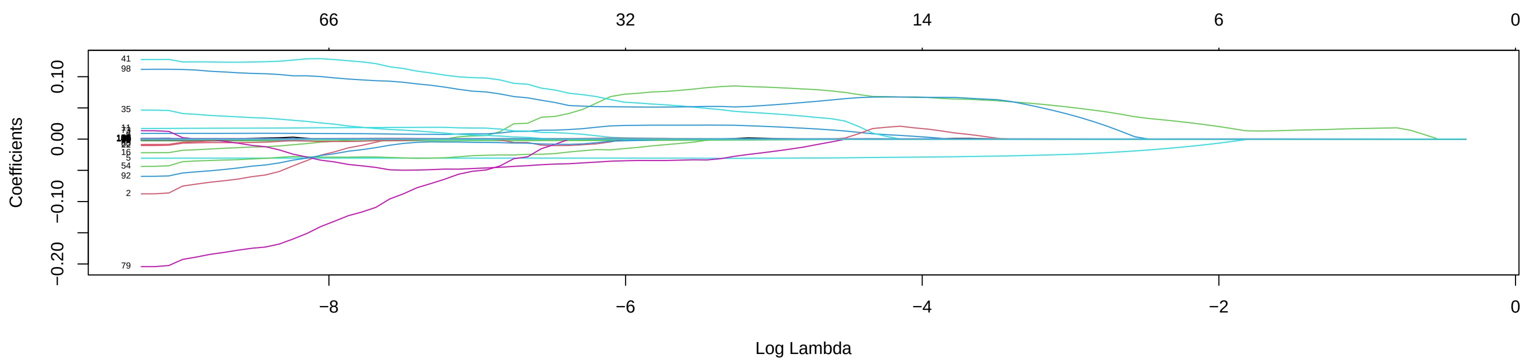
```r
cat("test RMSE:", round(rmse(test_preds, y_test), 4), "\n")
```

```
## test RMSE: 0.2358
```

```r
plot_data <- data.frame(predicted = as.vector(test_preds),actual = y_test)
ggplot(plot_data, aes(x = predicted, y = actual)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, color = "red") +
    labs(x = "Predicted Values", y = "Actual Values", title = "Test Set: Predicted vs Actual Values") +
    theme_classic()
```




```r
#
# 2) lasso regression
#

# a) fit lasso regression model
lasso_model <- glmnet(X_train_matrix, y_train, alpha=1) # alpha=1 for lasso, 0 for ridge
plot(lasso_model, xvar="lambda", label=TRUE)
```



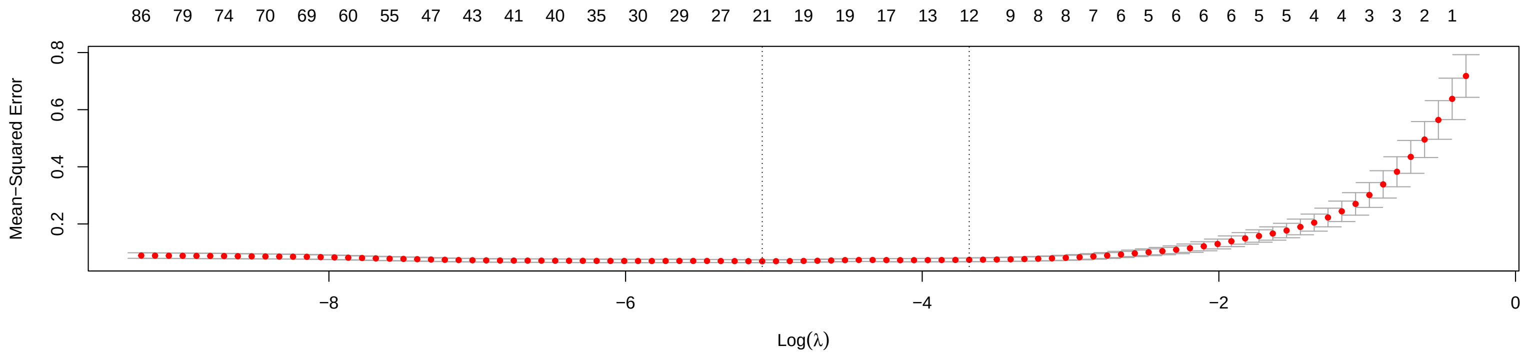
```r
cat("number of lambda values:", length(lasso_model$lambda), "\n")
```

```
## number of lambda values: 97
```

```r
cat("lambda range:", range(lasso_model$lambda), "\n")
```

```
## lambda range: 0.0000946605 0.716072
```

```r
# b) 10-fold cv to find optimal lambda
cv_lasso <- cv.glmnet(X_train_matrix, y_train, alpha=1, nfolds=10)
plot(cv_lasso)
```




```r
cat("optimal lambda (minimum MSE):", cv_lasso$lambda.min, "\n")
```

```
## optimal lambda (minimum MSE): 0.006228029
```

```r
cat("lambda within 1 SE (1 standard error rule):", cv_lasso$lambda.1se, "\n")
```

```
## lambda within 1 SE (1 standard error rule): 0.02514266
```

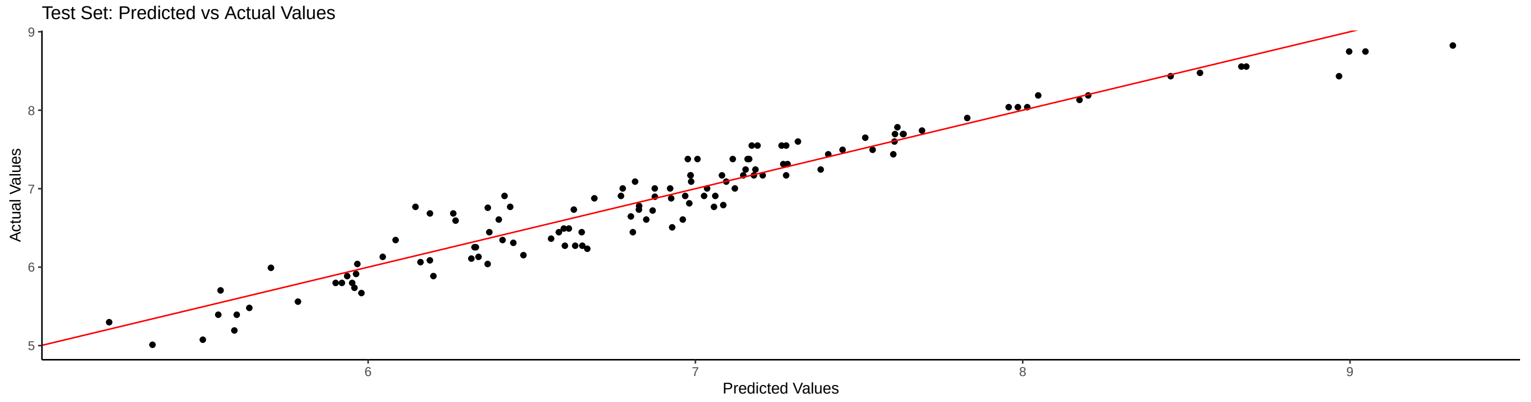
```r
coef_min <- coef(cv_lasso, s="lambda.min")

# c) test set performance
train_preds <- predict(cv_lasso, newx=X_train_matrix, s="lambda.min")
test_preds <- predict(cv_lasso, newx=X_test_matrix, s="lambda.min")
cat("train RMSE:", round(rmse(train_preds, y_train), 4), "\n")
```

```
## train RMSE: 0.2294
```

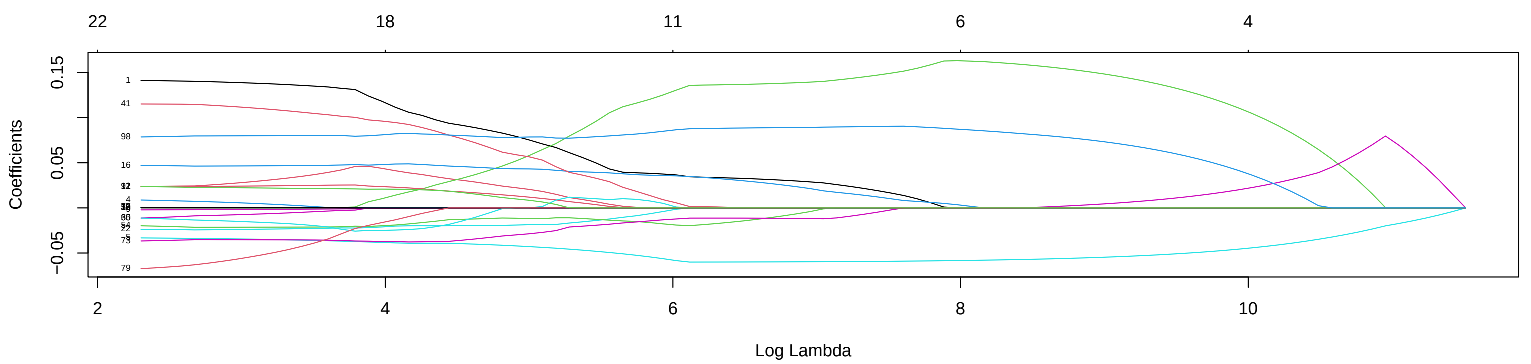
```r
cat("test RMSE:", round(rmse(test_preds, y_test), 4), "\n")
```

```
## test RMSE: 0.2301
```

```r
plot_data <- data.frame(predicted = as.vector(test_preds),actual = y_test)
ggplot(plot_data, aes(x = predicted, y = actual)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, color = "red") +
    labs(x = "Predicted Values", y = "Actual Values", title = "Test Set: Predicted vs Actual Values") +
    theme_classic()
```




```r
#
# 3) adaptive lasso regression
#

# a) fit adaptive lasso model with weights from ridge regression
ridge_coef_vector <- as.vector(coef(cv_ridge, s="lambda.min"))[-1] # get weights from ridge regression coefficients, exclude intercept
weights <- 1/abs(ridge_coef_vector) # inverse of absolute coefficients
weights[is.infinite(weights)] <- max(weights[!is.infinite(weights)]) * 100 # handle zero coefficients
adaptive_lasso_model <- glmnet(X_train_matrix, y_train, alpha=1, penalty.factor=weights) # incorporate weights
plot(adaptive_lasso_model, xvar="lambda", label=TRUE)
```



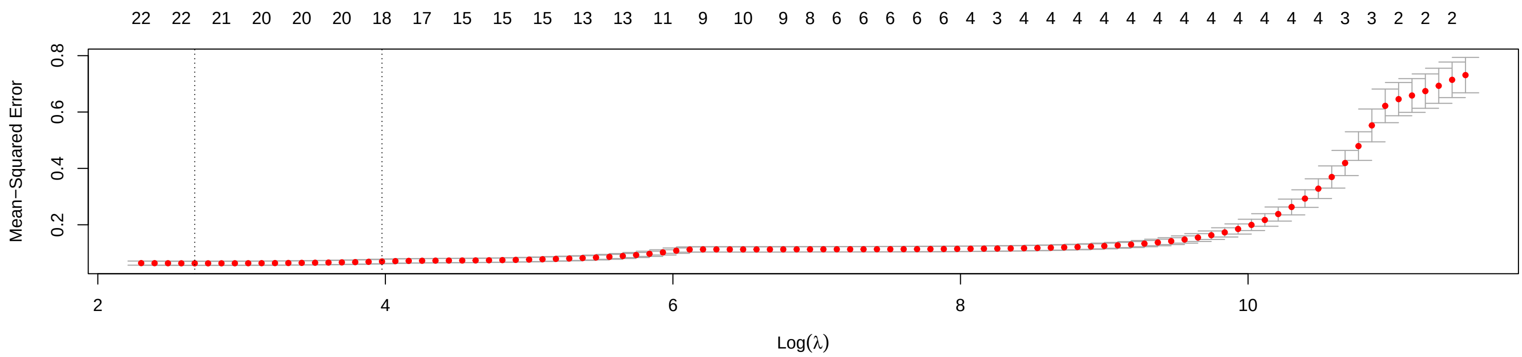
```r
cat("number of lambda values:", length(adaptive_lasso_model$lambda), "\n")
```

```
## number of lambda values: 100
```

```r
cat("lambda range:", range(adaptive_lasso_model$lambda), "\n")
```

```
## lambda range: 9.993739 99937.39
```

```r
# b) 10-fold cv to find optimal lambda
cv_adaptive_lasso <- cv.glmnet(X_train_matrix, y_train, alpha=1, penalty.factor=weights, nfolds=10)
plot(cv_adaptive_lasso)
```




```r
cat("optimal lambda (minimum MSE):", cv_adaptive_lasso$lambda.min, "\n")
```

```
## optimal lambda (minimum MSE): 14.4992
```

```r
cat("lambda within 1 SE (1 standard error rule):", cv_adaptive_lasso$lambda.1se, "\n")
```

```
## lambda within 1 SE (1 standard error rule): 53.33358
```

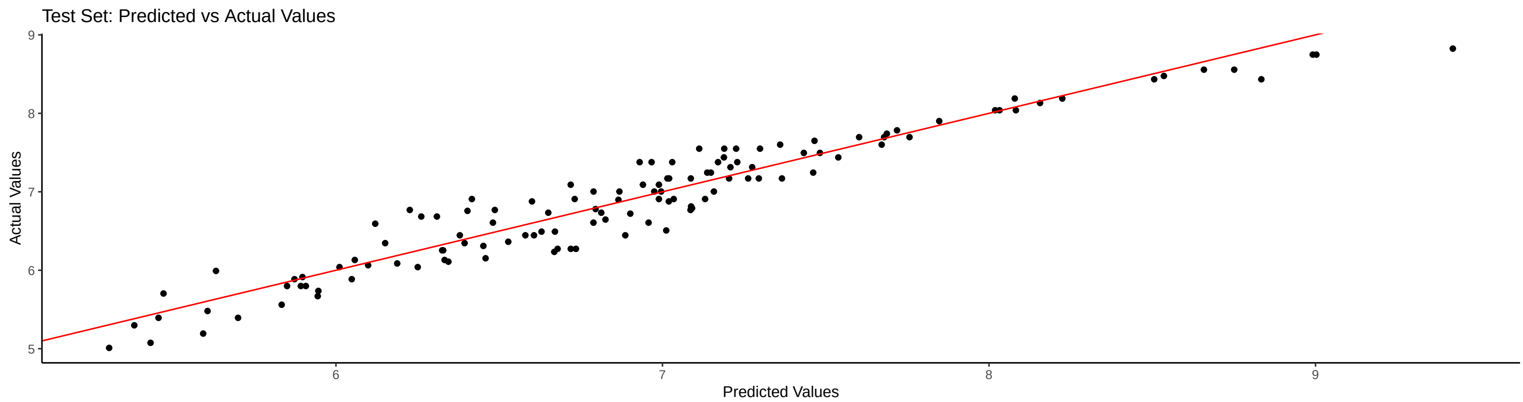
```r
coef_adaptive <- coef(cv_adaptive_lasso, s="lambda.min")

# c) test set performance
train_preds <- predict(cv_adaptive_lasso, newx=X_train_matrix, s="lambda.min")
test_preds <- predict(cv_adaptive_lasso, newx=X_test_matrix, s="lambda.min")
cat("train RMSE:", round(rmse(train_preds, y_train), 4), "\n")
```

```
## train RMSE: 0.2287
```

```r
cat("test RMSE:", round(rmse(test_preds, y_test), 4), "\n")
```

```
## test RMSE: 0.2375
```

```r
plot_data <- data.frame(predicted = as.vector(test_preds), actual = y_test)
ggplot(plot_data, aes(x = predicted, y = actual)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, color = "red") +
    labs(x = "Predicted Values", y = "Actual Values", title = "Test Set: Predicted vs Actual Values") +
    theme_classic()
```




```r
#
# compare methods
#

# compare number of non-zero coefficients between methods
cat("ridge:", sum(abs(coef(cv_ridge, s="lambda.min")) > 0) - 1, "\n") # -1 for intercept
```

```
## ridge: 107
```

```r
cat("lasso:", sum(abs(coef(cv_lasso, s="lambda.min")) > 0) - 1, "\n")
```

```
## lasso: 21
```

```r
cat("adaptive lasso:", sum(abs(coef_adaptive) > 0) - 1, "\n")
```

```
## adaptive lasso: 22
```

```r
# compare variable selection across methods
coef_comparison <- data.frame(
    Variable = rownames(coef_adaptive),
    Ridge = as.vector(coef(cv_ridge, s="lambda.min")),
    Lasso = as.vector(coef(cv_lasso, s="lambda.min")),
    Adaptive_Lasso = as.vector(coef_adaptive)
)
print(coef_comparison)
```

```
##                 Variable               Ridge           Lasso  Adaptive_Lasso
## 1            (Intercept)  2.126320100003685 -2.3871318620474 -6.3023988130661
## 2             START.YEAR  0.007072252358656  0.0013689781045  0.1403587190704
## 3          START.QUARTER -0.009064090699867  0.0000000000000  0.0243506645474
## 4        COMPLETION.YEAR  0.008149588932348  0.0834632435441  0.0001043260107
## 5     COMPLETION.QUARTER  0.000598087938405  0.0205903024111  0.0073135640571
## 6               PhysFin1 -0.030912009355813 -0.0305863777283 -0.0337932741778
## 7               PhysFin2  0.000032469363031  0.0000002503804  0.0000000000000
## 8               PhysFin3 -0.000079881355277  0.0000000000000  0.0000000000000
## 9               PhysFin4 -0.000042665760183  0.0000000000000  0.0000000000000
## 10              PhysFin5 -0.000594310998825 -0.0013075067546 -0.0020089638770
## 11              PhysFin6  0.000220748210329  0.0002803557314  0.0003557716648
## 12              PhysFin7  0.019571294882453  0.0000000000000  0.0237760623162
## 13              PhysFin8  0.000348726765650  0.0004294914392  0.0004493770053
## 14                 Econ1  0.000005622941512  0.0000000000000  0.0000000000000
## 15                 Econ2  0.000098081113169  0.0000000000000  0.0000000000000
## 16                 Econ3  0.000336106143117  0.0000000000000  0.0000000000000
## 17                 Econ4  0.007020542035221  0.0000000000000  0.0463541036616
## 18                 Econ5  0.000000005813180  0.0000000000000  0.0000000000000
## 19                 Econ6  0.000004442553688  0.0000000000000  0.0000000000000
## 20                 Econ7 -0.000055150101447  0.0000000000000  0.0000000000000
## 21                 Econ8  0.000029923414635  0.0000000000000  0.0000000000000
## 22                 Econ9 -0.000000200557919  0.0000000000000  0.0000000000000
## 23                Econ10  0.004539500919538  0.0000000000000 -0.0243722932682
## 24                Econ11  0.000011942699367  0.0000000000000  0.0000000000000
## 25                Econ12  0.000004078385215  0.0000000000000  0.0000000000000
## 26                Econ13  0.000002903647502  0.0000000000000  0.0000000000000
## 27                Econ14  0.000014229356455  0.0000070503478  0.0000000000000
## 28                Econ15  0.000275905320223  0.0000000000000  0.0000000000000
## 29                Econ16  0.000112306708060  0.0000000000000  0.0000000000000
## 30                Econ17  0.000003890004107  0.0000000000000  0.0000000000000
## 31                Econ18  0.000002155175211  0.0000019885228  0.0000000000000
## 32                Econ19  0.000000018134470  0.0000000000000  0.0000000000000
## 33            Econ1.lag1  0.000009685209256  0.0000000000000  0.0000000000000
## 34            Econ2.lag1  0.000032458550451  0.0000000000000  0.0000000000000
## 35            Econ3.lag1  0.000068319859339  0.0000000000000  0.0000000000000
## 36            Econ4.lag1 -0.006845758182389  0.0000000000000 -0.0088007051754
## 37            Econ5.lag1  0.000000005223755  0.0000000000000  0.0000000000000
## 38            Econ6.lag1 -0.000006393584022  0.0000000000000  0.0000000000000
## 39            Econ7.lag1 -0.000067111717961  0.0000000000000  0.0000000000000
## 40            Econ8.lag1  0.000284904682830  0.0001453979805  0.0000007575666
## 41            Econ9.lag1  0.000000487197552  0.0000007632263  0.0000000000000
## 42           Econ10.lag1  0.038635926812528  0.0417357016562  0.1147935376188
## 43           Econ11.lag1 -0.000014079753861  0.0000000000000  0.0000000000000
## 44           Econ12.lag1  0.000001296410694  0.0000000000000  0.0000000000000
## 45           Econ13.lag1 -0.000001430605650  0.0000000000000  0.0000000000000
## 46           Econ14.lag1  0.000014397042062  0.0000000000000  0.0000000000000
## 47           Econ15.lag1  0.000309373681746  0.0000000000000  0.0000000000000
## 48           Econ16.lag1  0.000156359731111  0.0000000000000  0.0000000000000
## 49           Econ17.lag1  0.000000101343119  0.0000000000000  0.0000000000000
## 50           Econ18.lag1 -0.000000994332853  0.0000000000000  0.0000000000000
## 51           Econ19.lag1  0.000000024162988  0.0000000000000  0.0000000000000
## 52            Econ1.lag2  0.000005128940561  0.0000000000000  0.0000000000000
## 53            Econ2.lag2  0.000084993593500  0.0000000000000  0.0000000000000
## 54            Econ3.lag2  0.000024719732990  0.0000000000000  0.0000000000000
## 55            Econ4.lag2 -0.016883545401540  0.0000000000000 -0.0214479059655
## 56            Econ5.lag2  0.000000011530356  0.0000000000000  0.0000000000000
## 57            Econ6.lag2 -0.000002389297157  0.0000000000000  0.0000000000000
## 58            Econ7.lag2 -0.000191828677967  0.0000000000000  0.0000000000000
## 59            Econ8.lag2  0.000507339081244  0.0002438803003  0.0003383473504
## 60            Econ9.lag2  0.000001686563952  0.0000033083017  0.0000000000000
## 61           Econ10.lag2  0.025915101907233  0.0000000000000 -0.0134209337191
## 62           Econ11.lag2  0.000003640645741  0.0000000000000  0.0000000000000
## 63           Econ12.lag2  0.000008823454368  0.0000000000000  0.0000000000000
## 64           Econ13.lag2 -0.000000963489976  0.0000000000000  0.0000000000000
## 65           Econ14.lag2  0.000020721745087  0.0000000000000  0.0000000000000
## 66           Econ15.lag2  0.000205137720537  0.0000000000000  0.0000000000000
## 67           Econ16.lag2  0.000100998071480  0.0000000000000  0.0000000000000
## 68           Econ17.lag2  0.000001541126711  0.0000000000000  0.0000000000000
## 69           Econ18.lag2 -0.000000201814175  0.0000000000000  0.0000000000000
## 70           Econ19.lag2  0.000000013947699  0.0000000000000  0.0000000000000
## 71            Econ1.lag3  0.000029645500770  0.0000000000000  0.0000000000000
## 72            Econ2.lag3  0.000058671821049  0.0000000000000  0.0000000000000
## 73            Econ3.lag3  0.000042489811561  0.0000000000000  0.0000000000000
## 74            Econ4.lag3 -0.036971865998876 -0.0219553020899 -0.0352683892165
## 75            Econ5.lag3  0.000000008418478  0.0000000000000  0.0000000000000
## 76            Econ6.lag3 -0.000008197510773  0.0000000000000  0.0000000000000
## 77            Econ7.lag3 -0.000142814128065  0.0000000000000  0.0000000000000
## 78            Econ8.lag3  0.000342206839381  0.0002770804701  0.0006000815240
## 79            Econ9.lag3  0.000000550070756  0.0000005957505  0.0000000000000
## 80           Econ10.lag3  0.002321618163323  0.0000000000000 -0.0630448723431
## 81           Econ11.lag3  0.000021231623137  0.0000000000000  0.0000000000000
## 82           Econ12.lag3  0.000008244114293  0.0000000000000  0.0000000000000
## 83           Econ13.lag3  0.000002033355085  0.0000000000000  0.0000000000000
## 84           Econ14.lag3  0.000022937043499  0.0000565616900  0.0000000000000
## 85           Econ15.lag3  0.000269114575893  0.0000000000000  0.0000000000000
## 86           Econ16.lag3  0.000137687964809  0.0000000000000  0.0000000000000
## 87           Econ17.lag3  0.000000322961802  0.0000000000000  0.0000000000000
## 88           Econ18.lag3  0.000000647684802  0.0000000000000  0.0000000000000
## 89           Econ19.lag3  0.000000025722844  0.0000000000000  0.0000000000000
## 90            Econ1.lag4  0.000032851594205  0.0000194556489  0.0000000000000
## 91            Econ2.lag4  0.000108019681604  0.0000000000000  0.0000000000000
## 92            Econ3.lag4  0.000331272771847  0.0000000000000  0.0000000000000
## 93            Econ4.lag4 -0.003906439343284  0.0000000000000  0.0227334816992
## 94            Econ5.lag4  0.000000003367242  0.0000000000000  0.0000000000000
## 95            Econ6.lag4  0.000002320541989  0.0000000000000  0.0000000000000
## 96            Econ7.lag4 -0.000149433412741  0.0000000000000  0.0000000000000
## 97            Econ8.lag4 -0.000199977065055  0.0000000000000  0.0000000000000
## 98            Econ9.lag4 -0.000000270738926  0.0000000000000  0.0000000000000
## 99           Econ10.lag4  0.052194931573460  0.0536738405669  0.0797714420764
## 100          Econ11.lag4  0.000001110629272  0.0000000000000  0.0000000000000
## 101          Econ12.lag4  0.000022289881677  0.0000000000000  0.0000000000000
## 102          Econ13.lag4  0.000002848680282  0.0000000000000  0.0000000000000
## 103          Econ14.lag4  0.000012786032727  0.0000000000000  0.0000000000000
## 104          Econ15.lag4  0.000274717548719  0.0000000000000  0.0000000000000
## 105          Econ16.lag4  0.000133817834652  0.0000000000000  0.0000000000000
## 106          Econ17.lag4  0.000004832674551  0.0000000000000  0.0000000000000
## 107          Econ18.lag4  0.000002132043307  0.0000000000000  0.0000000000000
## 108          Econ19.lag4  0.000000018977588  0.0000000000000  0.0000000000000
```