Exercise 4 Advanced Methods for Regression and Classification

12433732 - Stefan Merdian

2024-11-11

```
load("building.RData")
require(pls)
```

```
## Loading required package: pls

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```
library(pls)
```

```
set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.7,0.3))
train_data <- df[sample, ]
test_data <- df[!sample, ]
```

1) Ridge Regression:

a)

```
library(glmnet)
```

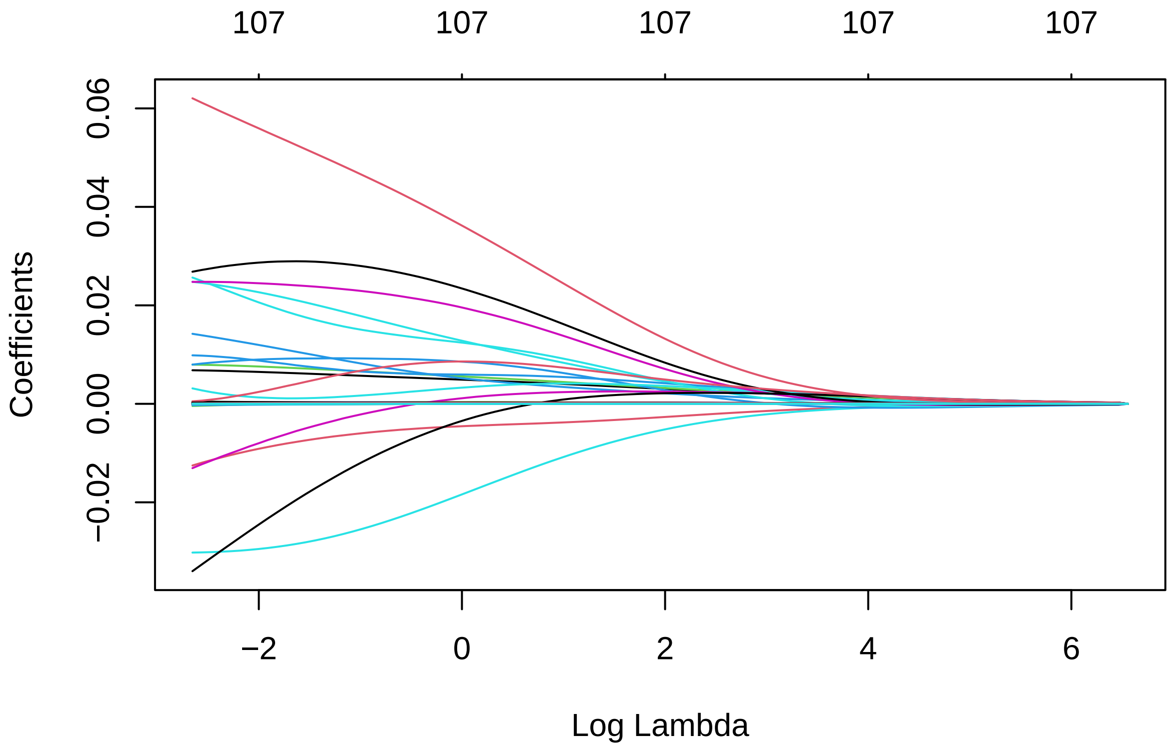
```
## Warning: package 'glmnet' was built under R version 4.4.2

## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```
ridge <- glmnet(train_data[,-1],train_data$y,alpha=0)
print(ridge)
```

```
##
## Call:  glmnet(x = train_data[, -1], y = train_data$y, alpha = 0)
##
##      Df  %Dev Lambda
## 1   107  0.00 704.10
## 2   107  8.38 641.50
## 3   107  9.09 584.50
## 4   107  9.86 532.60
## 5   107 10.67 485.30
## 6   107 11.54 442.20
## 7   107 12.47 402.90
## 8   107 13.45 367.10
## 9   107 14.49 334.50
## 10  107 15.59 304.80
## 11  107 16.75 277.70
## 12  107 17.96 253.00
## 13  107 19.22 230.60
## 14  107 20.54 210.10
## 15  107 21.90 191.40
## 16  107 23.32 174.40
## 17  107 24.78 158.90
## 18  107 26.27 144.80
## 19  107 27.78 131.90
## 20  107 29.32 120.20
## 21  107 30.88 109.50
## 22  107 32.44  99.80
## 23  107 34.01  90.94
## 24  107 35.57  82.86
## 25  107 37.12  75.50
## 26  107 38.65  68.79
## 27  107 40.16  62.68
## 28  107 41.63  57.11
## 29  107 43.08  52.04
## 30  107 44.50  47.41
## 31  107 45.87  43.20
## 32  107 47.20  39.36
## 33  107 48.49  35.87
## 34  107 49.74  32.68
## 35  107 50.95  29.78
## 36  107 52.14  27.13
## 37  107 53.29  24.72
## 38  107 54.41  22.53
## 39  107 55.51  20.52
## 40  107 56.59  18.70
## 41  107 57.65  17.04
## 42  107 58.70  15.53
## 43  107 59.75  14.15
## 44  107 60.78  12.89
## 45  107 61.81  11.74
## 46  107 62.84  10.70
## 47  107 63.87   9.75
## 48  107 64.89   8.88
## 49  107 65.92   8.10
## 50  107 66.94   7.38
```

```
## 51  107 67.97   6.72
## 52  107 68.99   6.12
## 53  107 70.00   5.58
## 54  107 71.01   5.08
## 55  107 72.00   4.63
## 56  107 72.99   4.22
## 57  107 73.96   3.85
## 58  107 74.91   3.50
## 59  107 75.85   3.19
## 60  107 76.76   2.91
## 61  107 77.64   2.65
## 62  107 78.50   2.42
## 63  107 79.34   2.20
## 64  107 80.14   2.00
## 65  107 80.91   1.83
## 66  107 81.65   1.67
## 67  107 82.36   1.52
## 68  107 83.04   1.38
## 69  107 83.69   1.26
## 70  107 84.30   1.15
## 71  107 84.89   1.05
## 72  107 85.45   0.95
## 73  107 85.97   0.87
## 74  107 86.47   0.79
## 75  107 86.94   0.72
## 76  107 87.39   0.66
## 77  107 87.81   0.60
## 78  107 88.20   0.55
## 79  107 88.58   0.50
## 80  107 88.93   0.45
## 81  107 89.26   0.41
## 82  107 89.57   0.38
## 83  107 89.86   0.34
## 84  107 90.13   0.31
## 85  107 90.39   0.28
## 86  107 90.63   0.26
## 87  107 90.86   0.24
## 88  107 91.07   0.22
## 89  107 91.27   0.20
## 90  107 91.46   0.18
## 91  107 91.64   0.16
## 92  107 91.81   0.15
## 93  107 91.97   0.14
## 94  107 92.13   0.12
## 95  107 92.27   0.11
## 96  107 92.41   0.10
## 97  107 92.54   0.09
## 98  107 92.66   0.08
## 99  107 92.78   0.08
## 100 107 92.89   0.07
```

```r
plot(ridge, xvar="lambda")
```



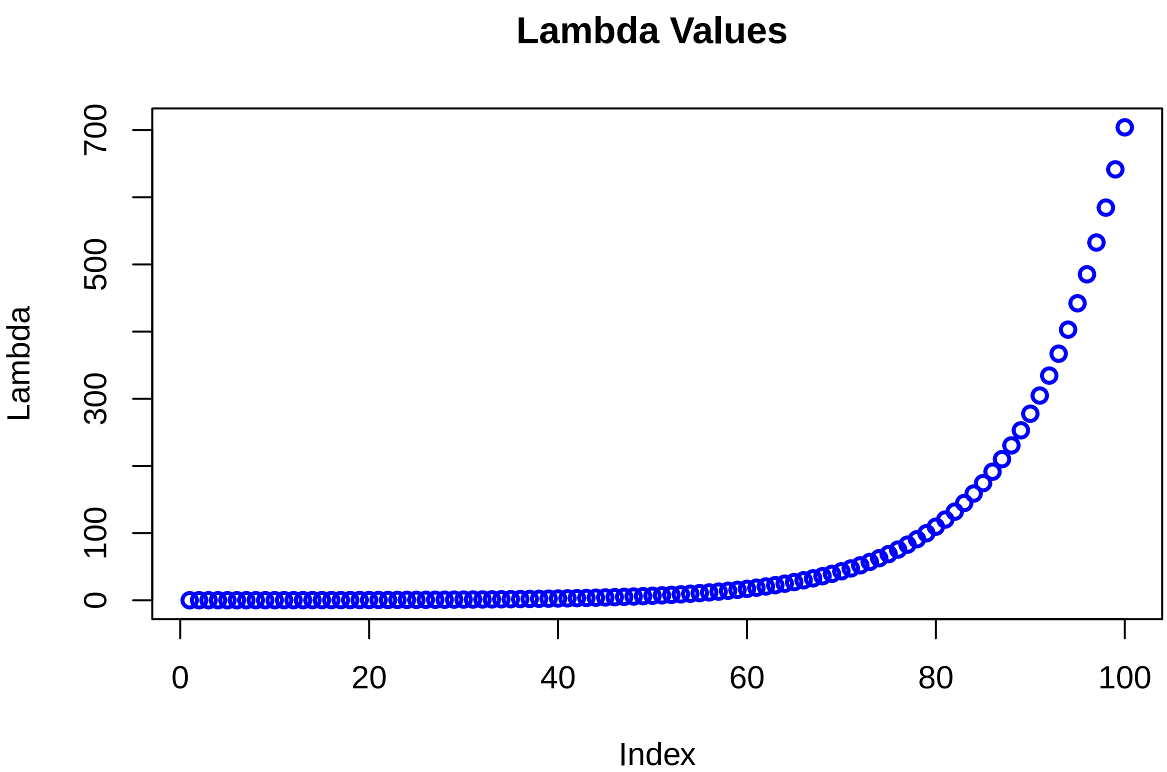
Interpretation:

We see how the coefficients in a Ridge Regression model change as λ varies. On the left side of the plot, where `log(lambda)` is between -2 and 0, λ is small (around 0.1 to 1), meaning the regularization effect is weak, so the coefficients remain relatively large. As λ increases, moving to the right on the plot, the regularization strength grows, adding a penalty that causes the coefficients to shrink toward zero. This shrinking effect happens because Ridge Regression penalizes large coefficients to prevent overfitting. With higher λ values, the model becomes less sensitive to individual predictors, focusing only on the most impactful variables and creating a simpler, more stable model. The plot demonstrates how Ridge Regression balances between fitting the data and keeping coefficients small, with stronger regularization leading to more significant shrinkage.

Which default parameters are used for lambda?:

glmnet fits the model for 100 values of lambda by default.

```r
plot(1:100, rev(ridge$lambda),xlab = "Index", ylab = "Lambda",
     main = "Lambda Values ",
     col = "blue", lwd = 2)
```



What is the meaning of the parameter alpha?:

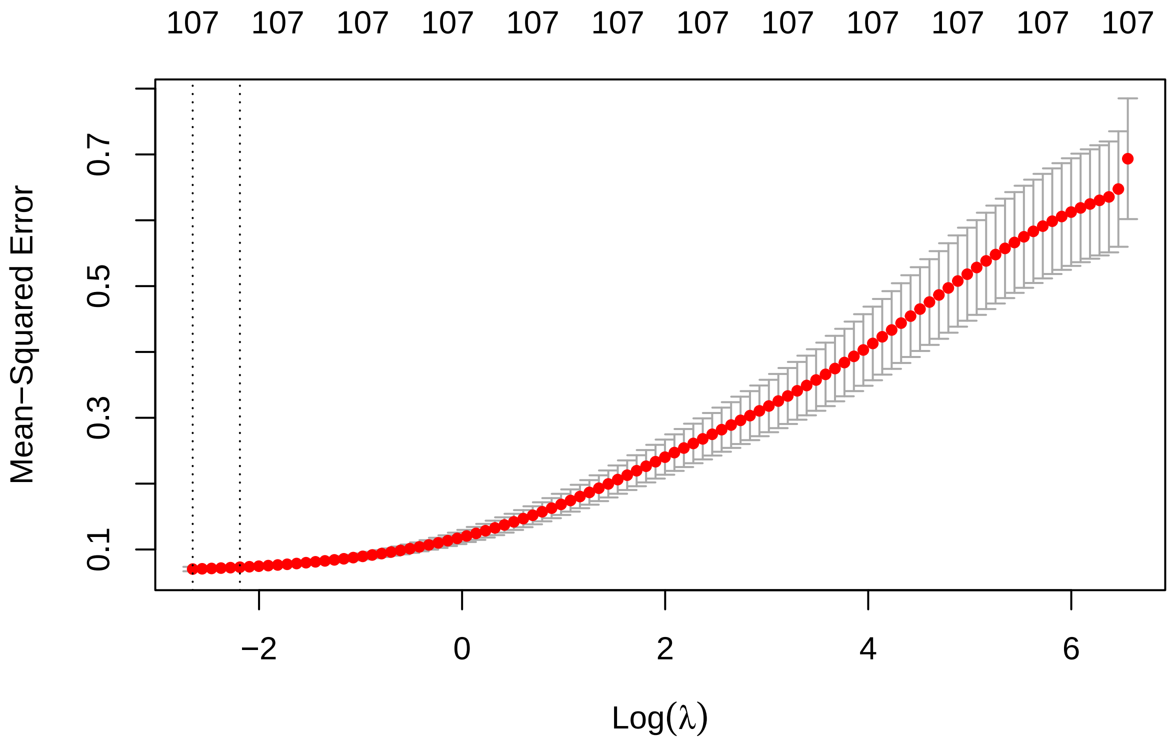
When alpha is set to 0 ($= 0$), the model applies only Ridge Regression. If $= 1$ it is Lasso Regression.

b)

```
x <- as.matrix(train_data[,-1])
cv_fit <- cv.glmnet(x,train_data$y,alpha=0)
print(cv_fit)
```

```
##
## Call:  cv.glmnet(x = x, y = train_data$y, alpha = 0)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure       SE Nonzero
## min 0.07041   100 0.07028 0.003357     107
## 1se 0.11211    95 0.07304 0.003111     107
```

```
plot(cv_fit)
```



How do you obtain the optimal tuning parameter and the regression coefficients?:

Intuitively, we might choose the model with the lowest MSE score, which corresponds to the smallest value, essentially resembling a least squares model. However, to avoid overfitting, we select the largest value within one standard error of the minimum cross-validation error. This choice provides a more generalized model that is likely to perform better on unseen data.

In the plot, the first vertical line represents the value with the lowest MSE. The second vertical line represents the within one standard error of the minimum MSE.

So we do:

```r
optimal_lambda_min <- cv_fit$lambda.min
optimal_lambda_1se <- cv_fit$lambda.1se

cat("lambda (min):", optimal_lambda_min, "\n")
```

```
## lambda (min): 0.0704078
```

```r
cat("Optimal lambda (1se), the one we will pick:", optimal_lambda_1se, "\n")
```

```
## Optimal lambda (1se), the one we will pick: 0.1121091
```

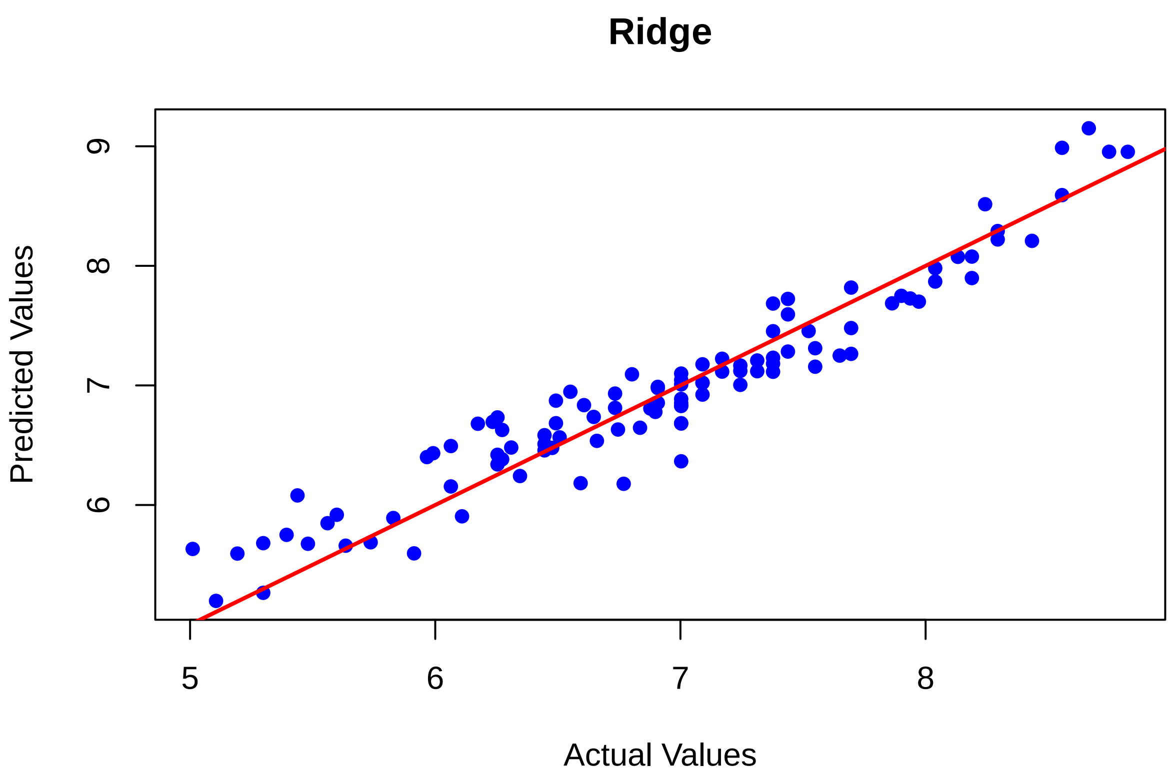
```r
coef_1se <- coef(cv_fit, s = "lambda.1se")
```

c)

```r
x_test <- as.matrix(test_data[,-1])
pred.ridge <- predict(cv_fit, newx = x_test, s = "lambda.1se")
sqrt(mean((test_data$y-pred.ridge)^2))
```

```
## [1] 0.258288
```

```r
plot(test_data$y, pred.ridge,
     xlab = "Actual Values", ylab = "Predicted Values",
     main = "Ridge",
     pch = 16, col = "blue")

abline(0, 1, col = "red", lwd = 2)
```




```r
rmse <- sqrt(mean((test_data$y-pred.ridge)^2))
cat('Ridge RMSE for test data:',rmse)
```

```
## Ridge RMSE for test data: 0.258288
```

EX2:

- RMSE all predictors: 0.6334959
- RMSE value for the subReg (10 Predictors) model: 0.2527903

EX3:

- RMSE PCR 32 Components : 0.2582652
- RMSE PLS 13 Components : 0.2749538 EX4:
- Ridge RMSE for test data: 0.258288

Task 2)

a)

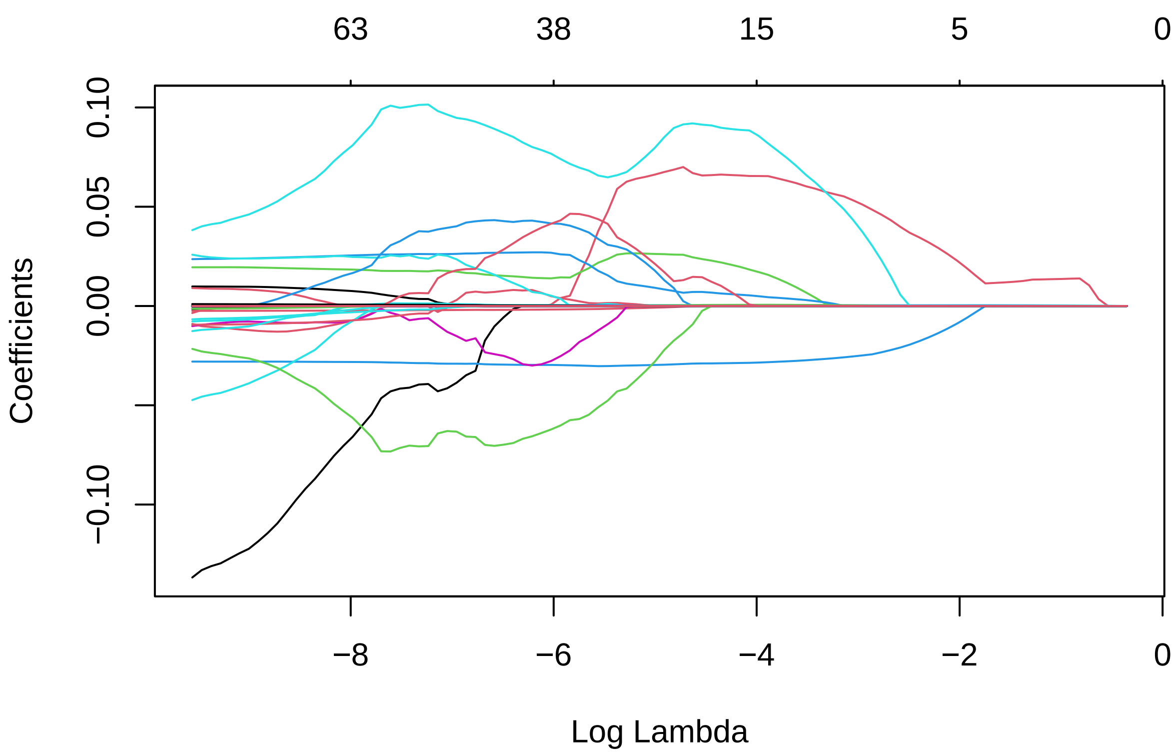
```
lasso <- glmnet(train_data[,-1], train_data$y, alpha = 1)
print(lasso)
```

```
##
## Call:  glmnet(x = train_data[, -1], y = train_data$y, alpha = 1)
##
##     Df  %Dev  Lambda
## 1    0   0.00 0.70410
## 2    1  12.16 0.64150
## 3    1  22.25 0.58450
## 4    2  31.52 0.53260
## 5    2  40.12 0.48530
## 6    3  47.49 0.44220
## 7    3  53.83 0.40290
## 8    3  59.08 0.36710
## 9    3  63.45 0.33450
## 10   3  67.07 0.30480
## 11   4  70.08 0.27770
## 12   4  72.58 0.25300
## 13   4  74.66 0.23060
## 14   4  76.40 0.21010
## 15   4  77.83 0.19140
## 16   4  79.03 0.17440
## 17   5  80.78 0.15890
## 18   5  82.30 0.14480
## 19   5  83.56 0.13190
## 20   5  84.61 0.12020
## 21   5  85.47 0.10950
## 22   5  86.20 0.09980
## 23   5  86.80 0.09094
## 24   5  87.29 0.08286
## 25   7  87.87 0.07550
## 26   7  88.46 0.06879
## 27   7  88.94 0.06268
## 28   8  89.35 0.05711
## 29   8  89.71 0.05204
## 30   8  90.01 0.04741
## 31   9  90.26 0.04320
## 32   9  90.47 0.03936
## 33  10  90.66 0.03587
## 34  11  90.85 0.03268
```

```
## 35  11 91.01 0.02978
## 36  12 91.15 0.02713
## 37  12 91.28 0.02472
## 38  12 91.38 0.02253
## 39  13 91.47 0.02052
## 40  14 91.54 0.01870
## 41  15 91.61 0.01704
## 42  16 91.68 0.01553
## 43  16 91.74 0.01415
## 44  17 91.79 0.01289
## 45  18 91.84 0.01174
## 46  19 91.90 0.01070
## 47  20 92.05 0.00975
## 48  21 92.24 0.00888
## 49  21 92.44 0.00810
## 50  23 92.60 0.00738
## 51  24 92.74 0.00672
## 52  24 92.86 0.00612
## 53  25 92.96 0.00558
## 54  26 93.04 0.00508
## 55  28 93.13 0.00463
## 56  29 93.24 0.00422
## 57  32 93.34 0.00385
## 58  32 93.46 0.00350
## 59  33 93.55 0.00319
## 60  37 93.64 0.00291
## 61  36 93.71 0.00265
## 62  38 93.78 0.00242
## 63  35 93.84 0.00220
## 64  36 93.88 0.00200
## 65  37 93.93 0.00183
## 66  40 93.98 0.00166
## 67  40 94.02 0.00152
## 68  41 94.06 0.00138
## 69  43 94.10 0.00126
## 70  45 94.16 0.00115
## 71  47 94.19 0.00105
## 72  48 94.24 0.00095
## 73  51 94.29 0.00087
## 74  50 94.36 0.00079
## 75  50 94.46 0.00072
## 76  53 94.47 0.00066
## 77  53 94.51 0.00060
## 78  55 94.56 0.00055
## 79  56 94.60 0.00050
## 80  56 94.65 0.00045
## 81  56 94.72 0.00041
## 82  60 94.77 0.00038
## 83  61 94.81 0.00034
## 84  63 94.84 0.00031
## 85  66 94.87 0.00028
## 86  66 94.91 0.00026
## 87  66 94.94 0.00024
## 88  65 94.96 0.00021
```

```
## 89   65 94.98 0.00020
## 90   67 95.00 0.00018
## 91   68 95.02 0.00016
## 92   70 95.04 0.00015
## 93   72 95.05 0.00014
## 94   73 95.06 0.00012
## 95   75 95.07 0.00011
## 96   75 95.08 0.00010
## 97   75 95.08 0.00009
## 98   76 95.09 0.00008
## 99   79 95.10 0.00008
## 100 81 95.11 0.00007
```

```r
plot(lasso, xvar='lambda')
```



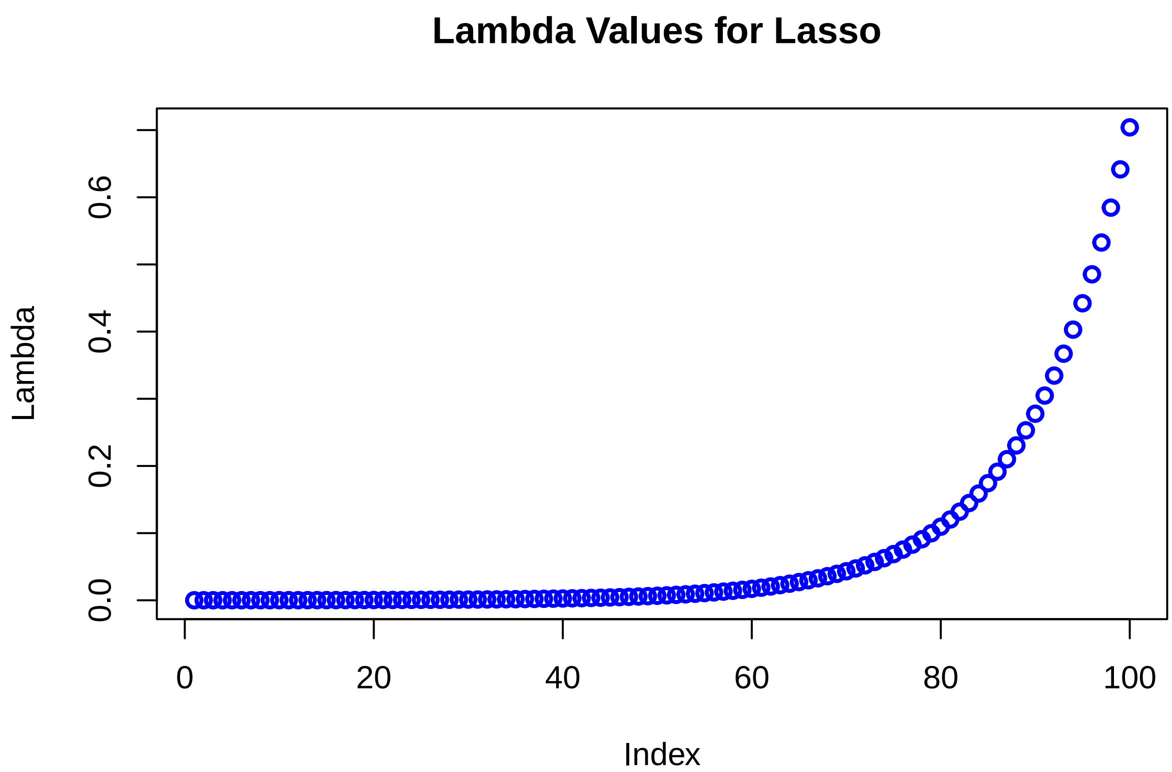
Interpretation:

Lasso can shrinks coefficients to zero, as   increase. Higher lambda values lead to more aggressive regularization, while lower values allow the model to be less constrained. As we move further to the right, the model simplifies by selecting only the most important predictors, meaning some coefficients shrink in importance or become zero.

Which default parameters are used for lambda?:

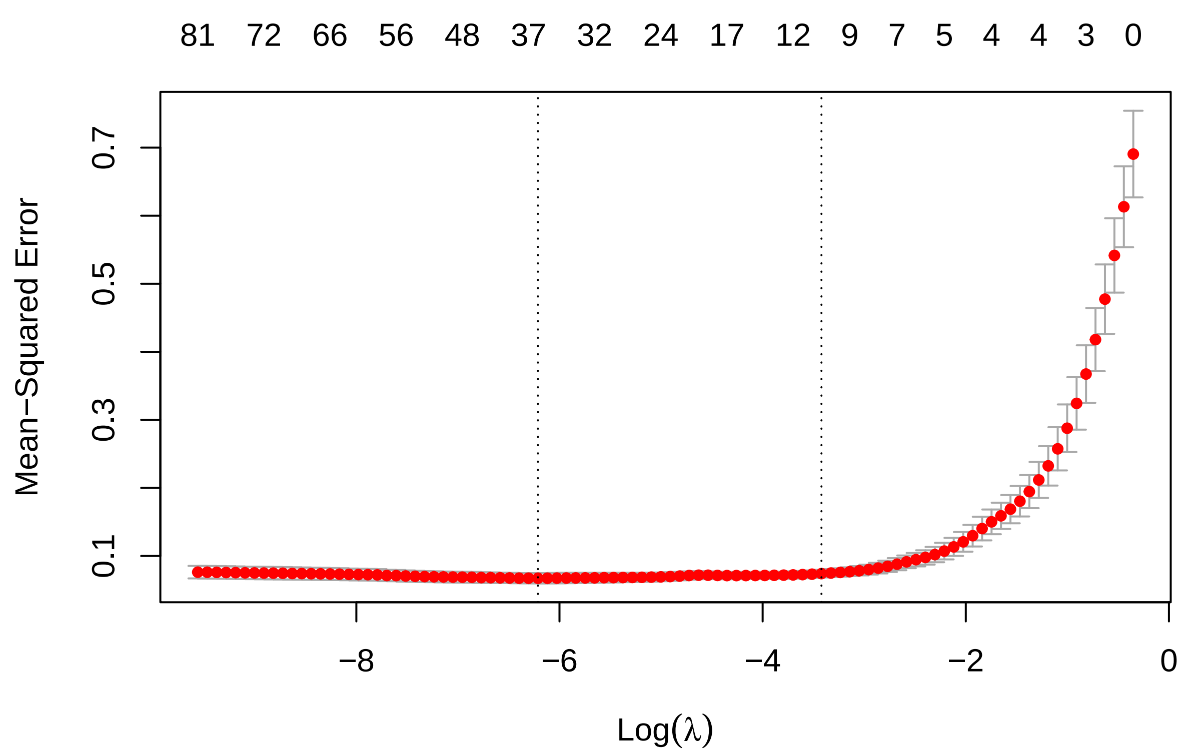
By default, glmnet creates 100 lambda values, giving a fine-grained path to explore the effect of different regularization strengths on the model.

```r
plot(1:100, rev(lasso$lambda),xlab = "Index", ylab = "Lambda",
     main = "Lambda Values for Lasso ",
     col = "blue", lwd = 2)
```



b)

```r
cv_fit_lasso <- cv.glmnet(x, train_data$y, alpha = 1)
plot(cv_fit_lasso)
```



Moving from left to right, decreases, meaning the regularization strength weakens. Smaller values allow more flexibility in the model, resulting in more complex models with more non-zero coefficients. The left dashed line corresponds to lambda.min, the value that gives the minimum MSE. The right dashed line corresponds to lambda.1se, the largest within one standard error of the minimum MSE. The same like in the Ridge Plot.

How do you obtain the optimal tuning parameter and the regression coefficients?:

Same like in Ridge regression we take the largest within one standard error of the minimum MSE. This choice provides a more generalized model that is likely to perform better on unseen data. So we will take the regression coefficients that Lasso has for lambda.1se

So we do:

```r
optimal_lambda_min_lasso <- cv_fit_lasso$lambda.min
optimal_lambda_1se_lasso <- cv_fit_lasso$lambda.1se

cat("lambda (min):", optimal_lambda_min_lasso, "\n")
```

```
## lambda (min): 0.002005239
```

```r
cat("Optimal lambda (1se), the one we will pick:", optimal_lambda_1se_lasso, "\n")
```

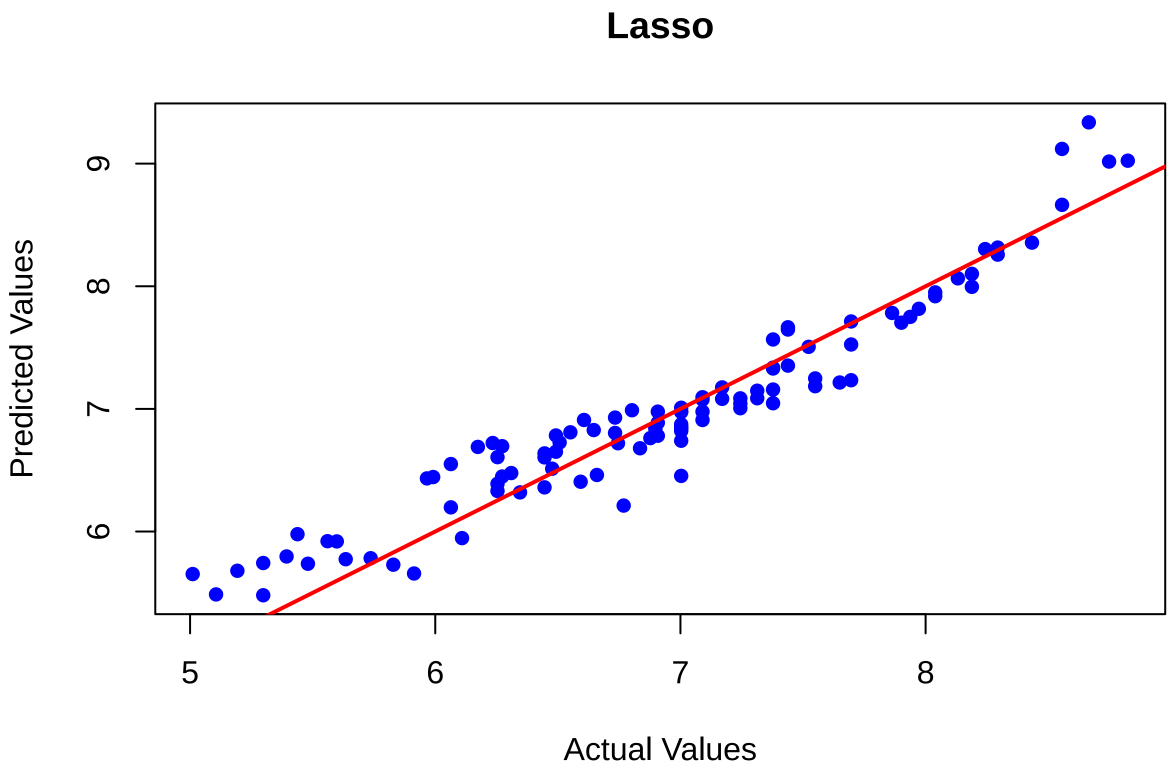
```
## Optimal lambda (1se), the one we will pick: 0.0326804
```

c)

```r
pred.lasso <- predict(cv_fit_lasso, newx = x_test, s = "lambda.1se")
sqrt(mean((test_data$y-pred.lasso)^2))
```

```
## [1] 0.2607314
```

```r
plot(test_data$y, pred.lasso,
     xlab = "Actual Values", ylab = "Predicted Values",
     main = "Lasso",
     pch = 16, col = "blue")

abline(0, 1, col = "red", lwd = 2)
```



EX2:

- RMSE all predictors: 0.6334959
- RMSE value for the subReg (10 Predictors) model: 0.2527903

EX3:

- RMSE PCR 32 Components : 0.2582652
- RMSE PLS 13 Components : 0.2749538

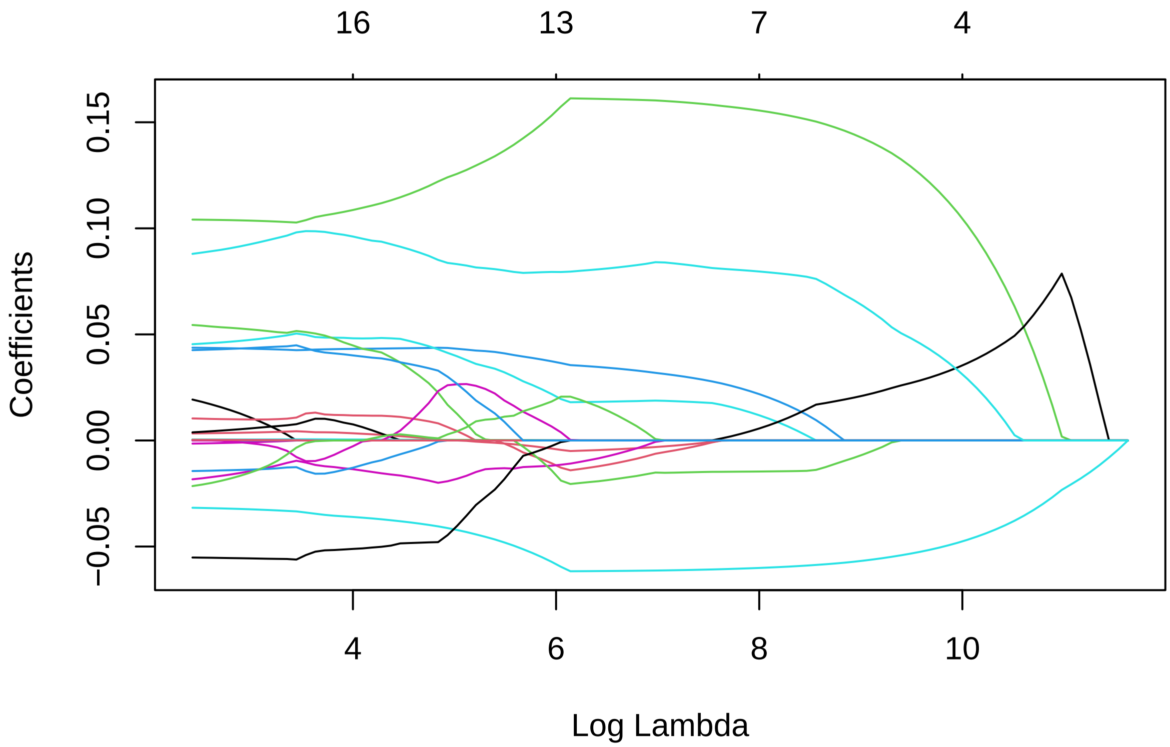
EX4:

- Ridge RMSE for test data: 0.258288
- Lasso RMSE for test data: 0.260731

3)

a)

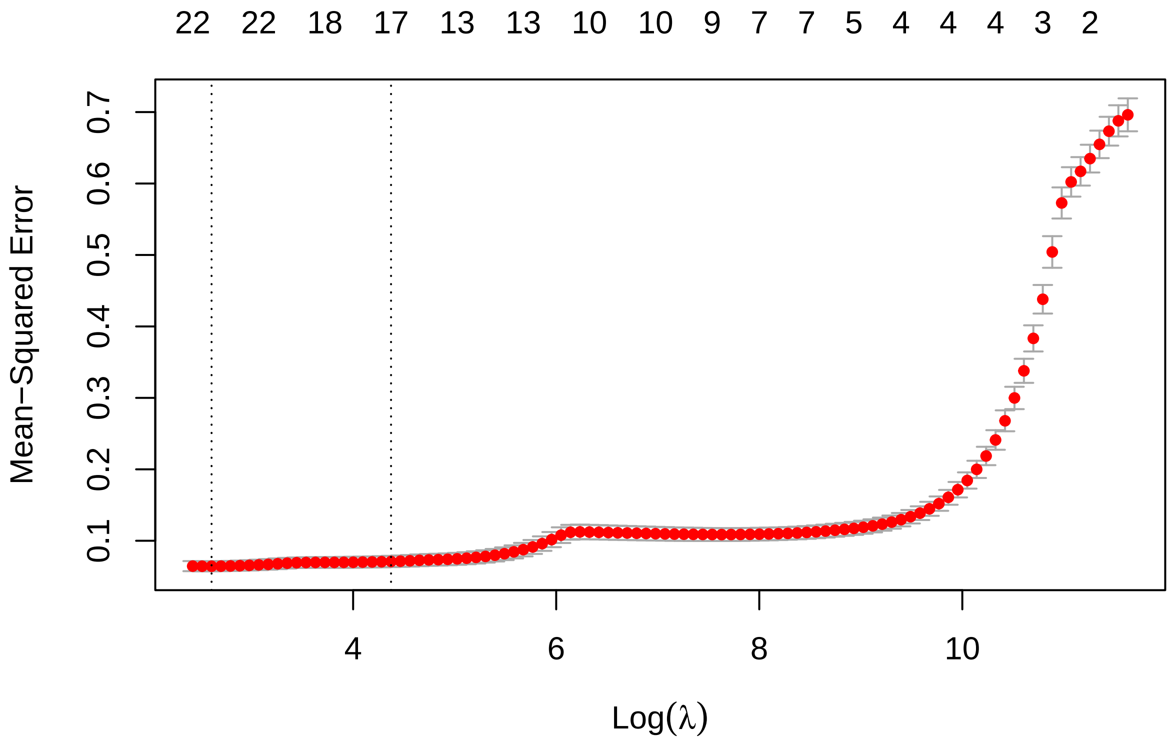
```
coef.ridge <- coef(cv_fit, s = "lambda.1se")
alasso <- glmnet(x,train_data$y,penalty.factor = 1 / abs(coef.ridge[-1]))
plot(alasso, xvar="lambda")
```



We are seening now the plot for adaptive lasso. The interpration ist the same. The key diffrence here is, we changed the penalty factor to **penalty.factor = 1 / abs(coef.ridge**[**-1**]). The idea is to penalize less important variables more heavily, which increases their likelihood of being shrunk to zero, while penalizing more important variables less, allowing them to remain in the model. **1 / abs(coef.ridge**[**-1**]) creates adaptive weights for each predictor, where the weight for each predictor is the inverse of its absolute Ridge coefficient. This means that predictors with larger Ridge coefficients will have smaller penalties penalty, making them less likely to be shrunk to zero.

b)

```
alasso.cv <- cv.glmnet(x,train_data$y,penalty.factor = 1 / abs(coef.ridge[-1]))
plot(alasso.cv)
```



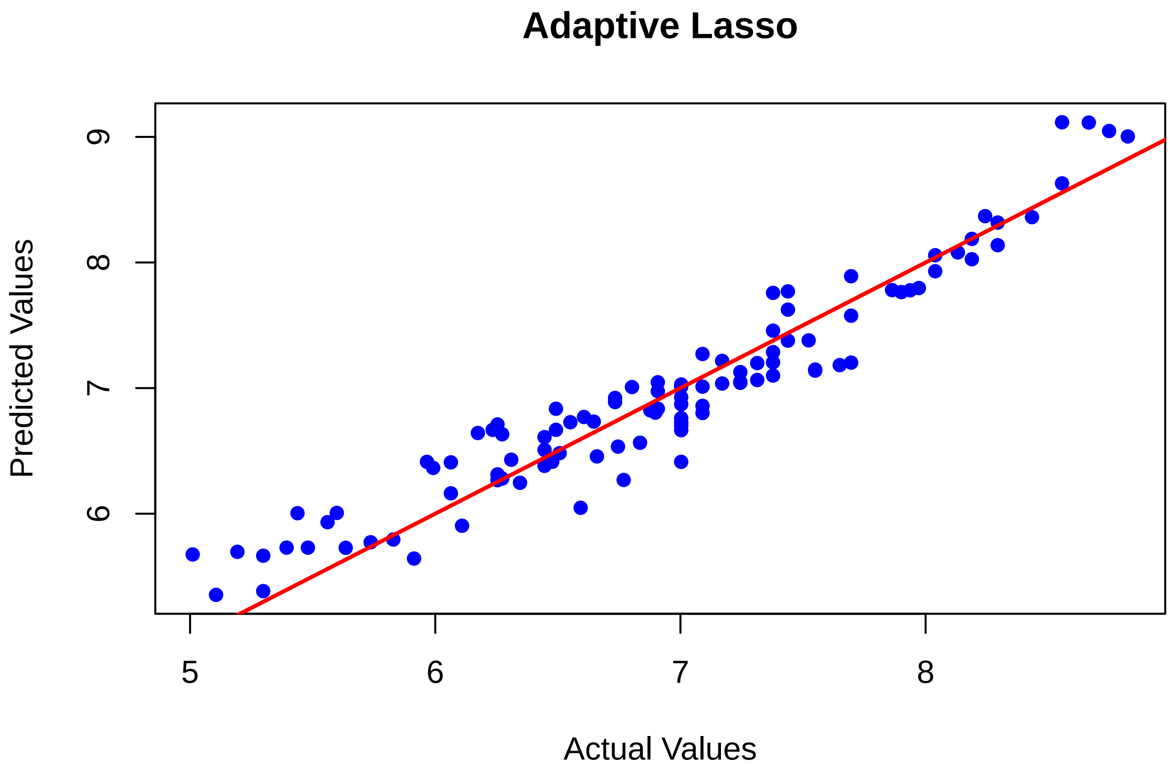
Here the same again. The left dashed line corresponds to lambda.min, the value that gives the minimum MSE. The right dashed line corresponds to lambda.1se, the largest within one standard error of the minimum MSE.

c)

```
pred.alasso <- predict(alasso.cv, newx = x_test, s = "lambda.1se")
sqrt(mean((test_data$y-pred.alasso)^2))
```

```
## [1] 0.264656
```

```
plot(test_data$y, pred.alasso,
     xlab = "Actual Values", ylab = "Predicted Values",
     main = "Adaptive Lasso",
     pch = 16, col = "blue")

abline(0, 1, col = "red", lwd = 2)
```



EX2: - RMSE all predictors: 0.6334959 - RMSE value for the subReg (10 Predictors) model: 0.2527903 EX3: - RMSE PCR 32 Components : 0.2582652 - RMSE PLS 13 Components : 0.2749538 EX4: - Ridge RMSE for test data: 0.258288 - Lasso RMSE for test data: 0.260731 - Adaptive Lasso RMSE for test data: 0.264656

Is the model more plausible for the interpretation?:

Yes, by using Ridge Regression coefficients as weights, Adaptive Lasso can assign different penalties to different predictors. Variables with higher Ridge coefficients are penalized less, making it more likely that they will be included in the model. This results in a model that is more likely to retain important variables while shrinking or eliminating less relevant ones, improving interpretability. Standard Lasso can be overly aggressive in shrinking coefficients, sometimes removing variables that might be important. So compared to standard lasso, this one reduces unnecessary shrinkage on significant predictors. Especially for interpretation, Adaptive Lasso provides a more focused model by highlighting the most important variables. However, this approach relies on the assumption that the Ridge coefficients were selected appropriately. If the Ridge model selected variables poorly, the Adaptive Lasso model may also lack accuracy.