LASSO and Ridge Regression amsterdam

Laurens van der Maas

12/1/2019

Shrinkage Methods

In this section shrinkage methods are applied to the AirBnB data. These methods constrain the coefficient estimates, shrinking them towards zero. This approach can significantly reduce the variance.

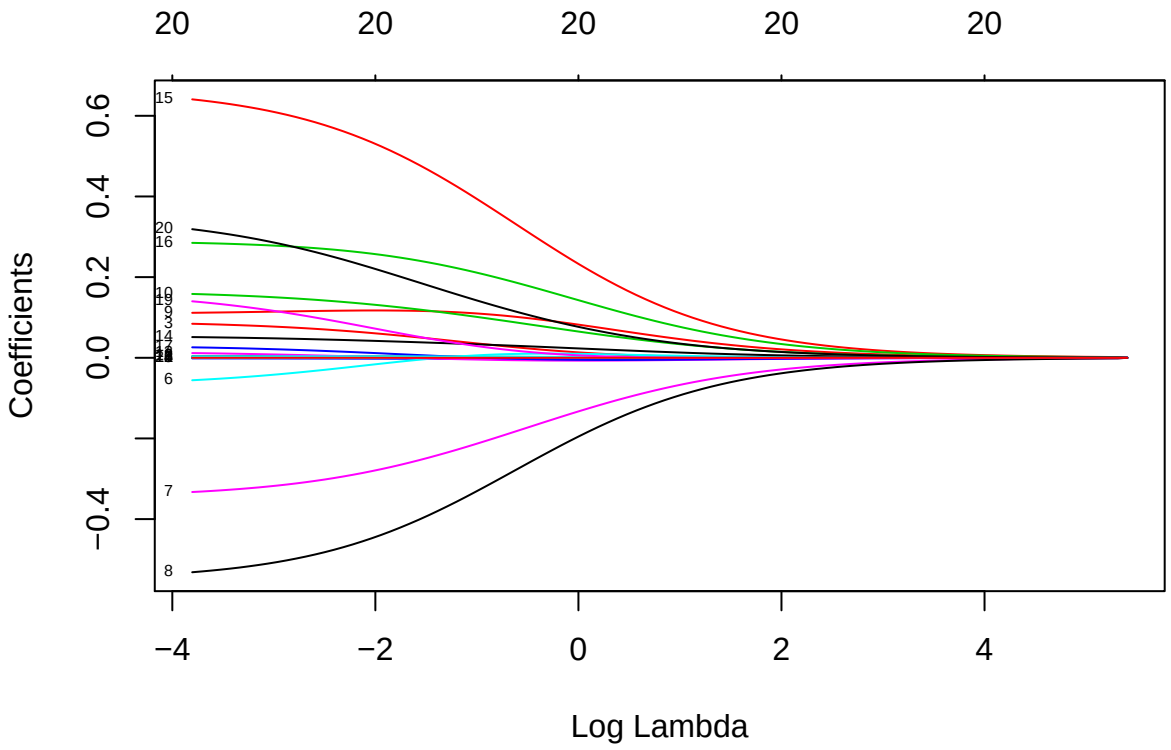
Ridge Regression

In a ridge regression, instead of using the least squares fitting procedure typical of OLS a penalty term is included in the estimation of the coefficients. The parameter λ determines the size of this penalty. The larger the λ the more the oefficients are shrunk.

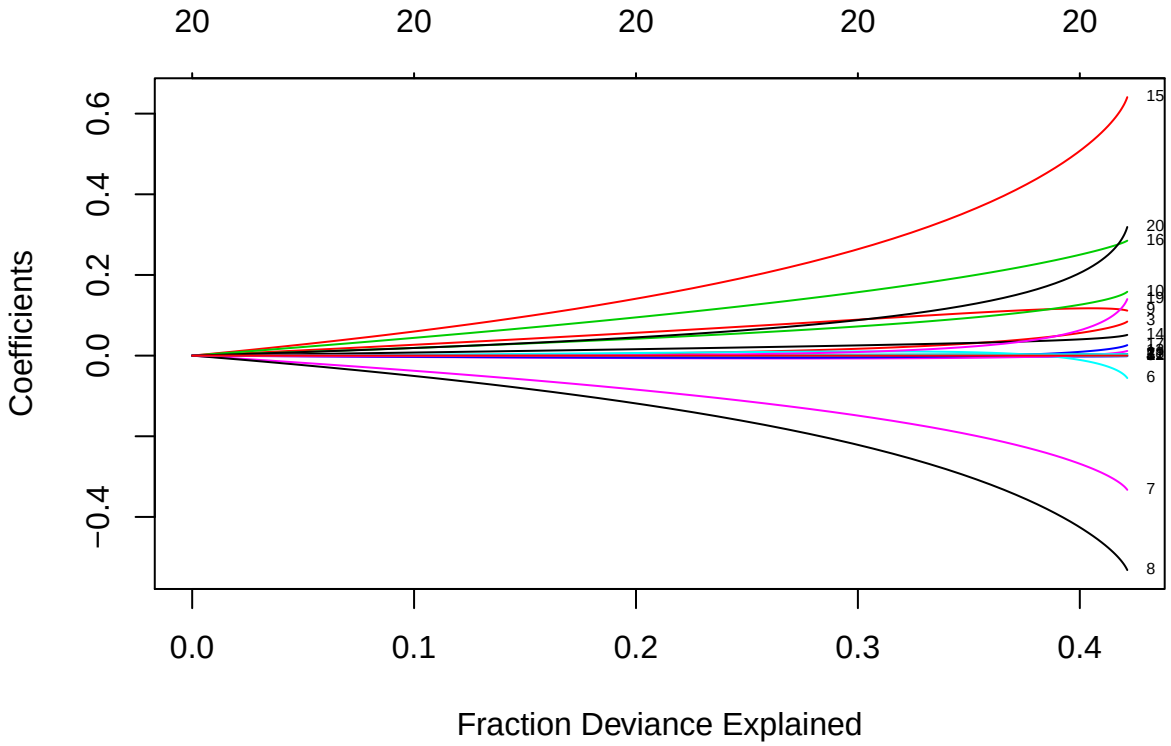
In terms of fractional deviance explained, it turns out that the (dummy) variables Private Room, Shared Room, super_strict_30, super_strict_60, near_centre are most important. Some of these, however, converge to zero faster than other variables that explain less of the fractional deviance.

In the optimal setting, the log of lambda is near -4. The optimal lambda is equal to 0.022 and the MSE on the test set is equal to 0.380. This optimal lambda is very small and therefore the model is not very different from the MLR model. Even more, the test MSE of the ridge regression with a lambda optimal for the training set is larger than that of the MLR. This method is therefore not relevant on this dataset.

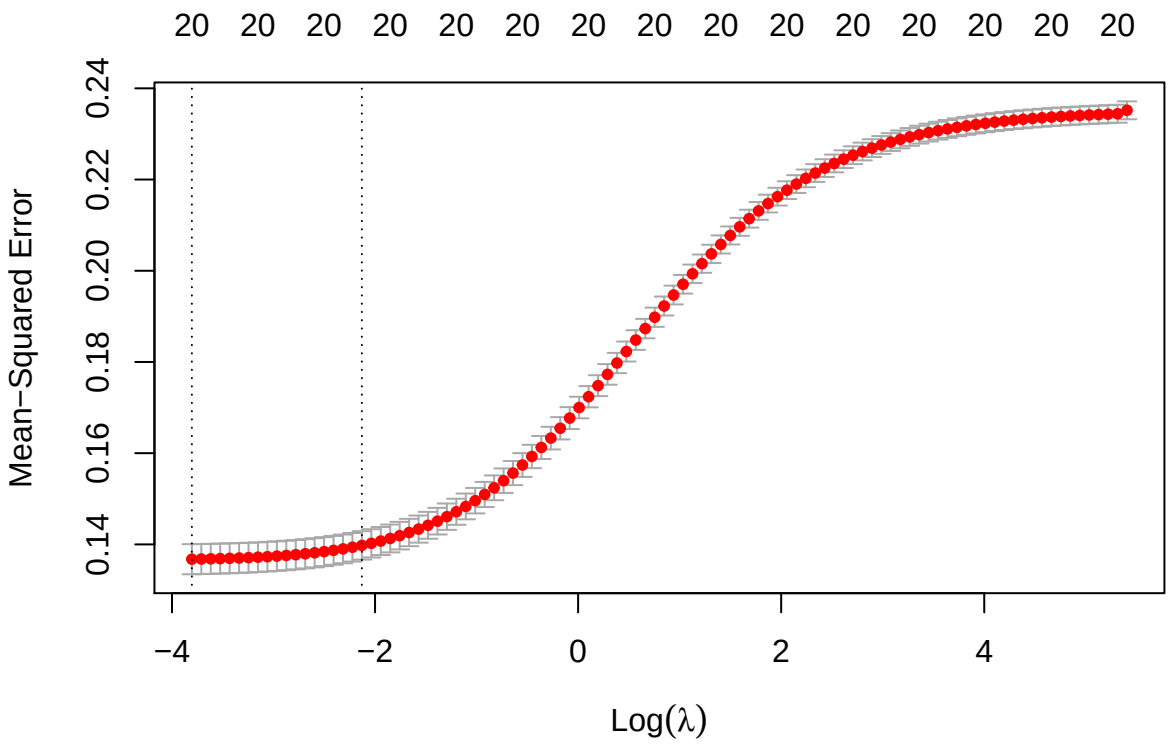
```
suppressMessages(library(readr))
suppressMessages(library(glmnet))

amsterdam <- read_csv('st443_final_data')
amsterdam <- amsterdam[,-c(1,15)]

# glmnet does not use formula language
x <- model.matrix(logprice ~ ., data = amsterdam)
y <- amsterdam$logprice

fit.ridge <-glmnet(x, y, alpha=0)

# 8, 7, 15, 20, 16 most important vars
plot(fit.ridge, xvar="lambda", label= TRUE)
```




```
plot(fit.ridge, xvar="dev", label= TRUE)
```




```
cv.ridge <-cv.glmnet(x, y, alpha=0)

## Plot of CV mse vs log (lambda), small lambda is best
plot(cv.ridge)
```




```
## Coefficent vector corresponding to the mse which is within
# one standard error of the lowest mse using the best lambda.
coef(cv.ridge)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                          1
## (Intercept)                                   4.082406e+00
## (Intercept)                                   .
## review_scores_rating                          3.227879e-03
## host_is_superhost                             6.336638e-02
## host_listings_count                          -3.531530e-04
## host_identity_verified                       -2.162182e-03
## room_typeHotel room                          -1.971261e-02
## room_typePrivate room                        -2.857848e-01
## room_typeShared room                         -4.554177e-01
## bathrooms                                     1.172363e-01
## bedrooms                                      1.343557e-01
## minimum_nights                               -1.874507e-04
## number_of_reviews                            -3.575878e-04
## cancellation_policymoderate                   2.877033e-03
## cancellation_policystrict_14_with_grace_period  4.281647e-02
## cancellation_policysuper_strict_30            5.439223e-01
## cancellation_policysuper_strict_60            2.609669e-01
## instant_bookableTRUE                          1.308324e-02
## cleaning_fee                                  3.490979e-03
## location_3waysModerate                        7.842598e-02
## location_3waysnear_centre                     2.299614e-01
## host_since_duration                          -9.967789e-06
```

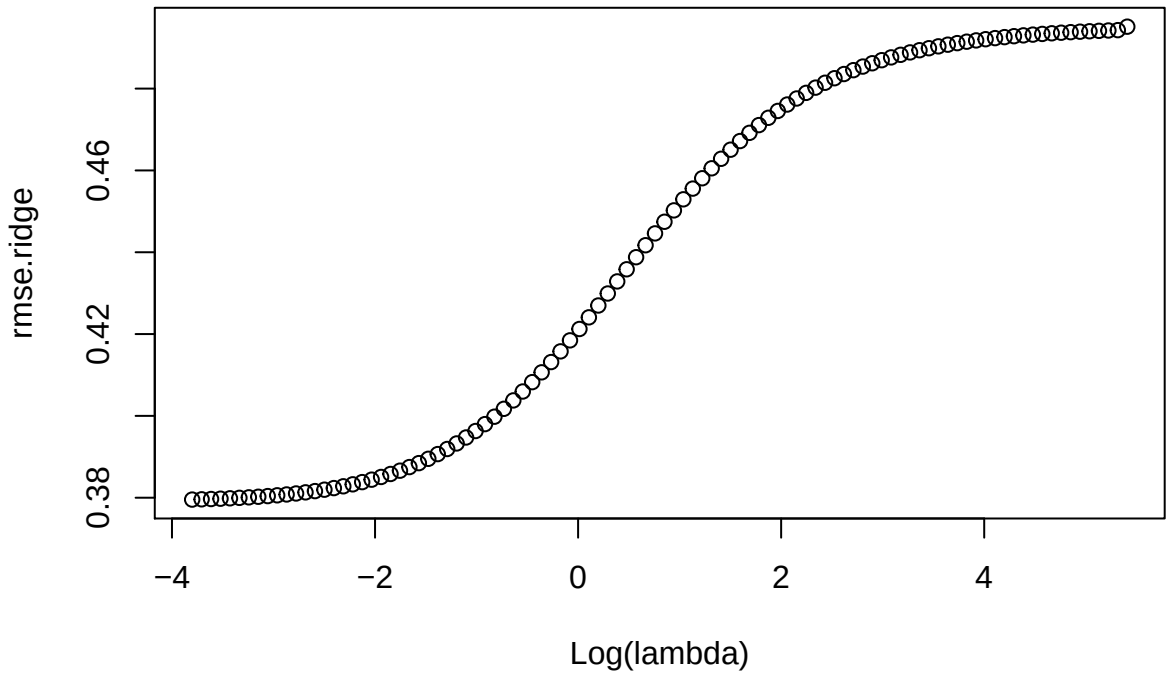
```
## Coefficient vector corresponding to the lowest mse using the best lambda
coef(glmnet(x,y,alpha=0, lambda=cv.ridge$lambda.min))
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                          s0
## (Intercept)                                    3.967646e+00
## (Intercept)                                    .
## review_scores_rating                           3.592352e-03
## host_is_superhost                              8.430644e-02
## host_listings_count                           -5.073438e-04
## host_identity_verified                        -1.472142e-03
## room_typeHotel room                           -5.567573e-02
## room_typePrivate room                         -3.328267e-01
## room_typeShared room                          -5.317271e-01
## bathrooms                                      1.115002e-01
## bedrooms                                       1.582633e-01
## minimum_nights                                -2.600664e-04
## number_of_reviews                             -3.480520e-04
## cancellation_policymoderate                    1.165951e-02
## cancellation_policystrict_14_with_grace_period 5.136956e-02
## cancellation_policysuper_strict_30             6.407773e-01
## cancellation_policysuper_strict_60             2.849320e-01
## instant_bookableTRUE                           2.598437e-02
## cleaning_fee                                   3.518221e-03
## location_3waysModerate                         1.400964e-01
## location_3waysnear_centre                      3.188280e-01
## host_since_duration                           -1.652107e-05
```

```
# finding MSE
traingsize = floor(0.7*nrow(amsterdam))
set.seed(123)
train = sample(seq_len(nrow(amsterdam)),size = traingsize)

ridge.train <-glmnet(x[train,], y[train], alpha = 0)
pred.test.ridge <-predict(ridge.train, x[-train,])
dim(pred.test.ridge)
```

```
## [1] 4506  100
```

```
rmse.ridge <-sqrt(apply((y[-train]-pred.test.ridge)^2,2,mean))
plot(log(ridge.train$lambda), rmse.ridge, type="b", xlab="Log(lambda)")
```




```
lambda.best.ridge <- ridge.train$lambda[order(rmse.ridge)[1]]
lambda.best.ridge
```

```
## [1] 0.02234597
```

```
mseRidge <- min(rmse.ridge)
mseRidge
```

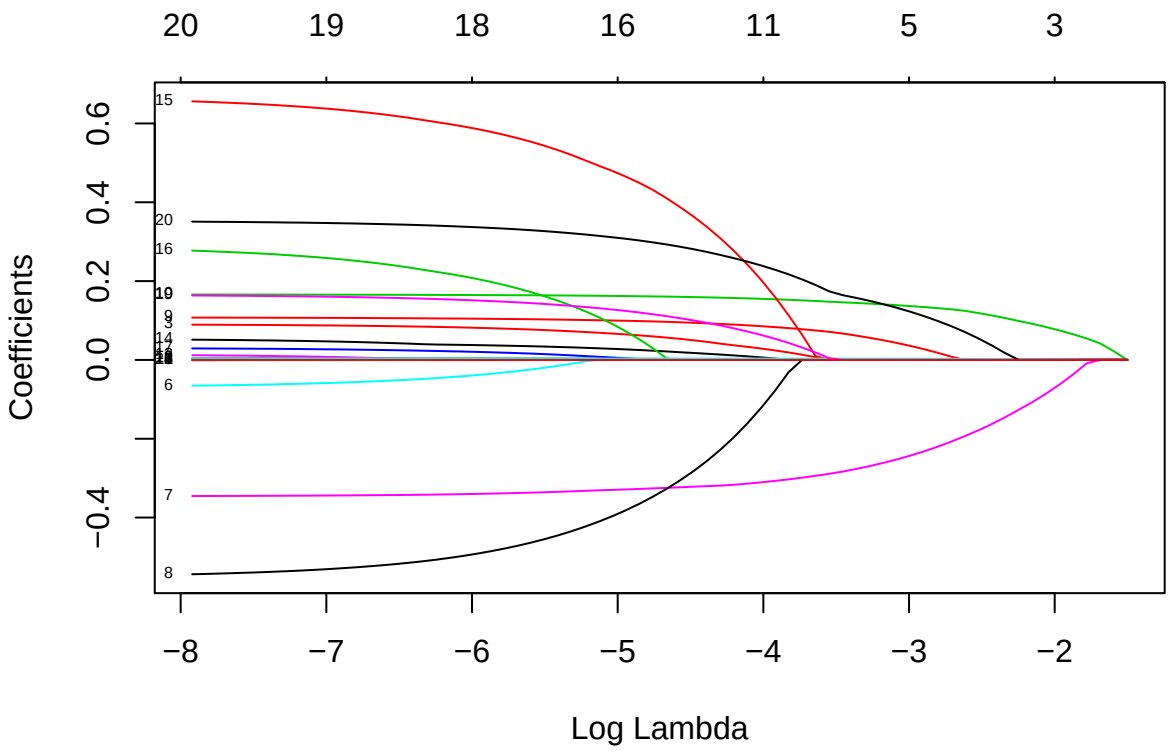
```
## [1] 0.3795437
```

LASSO Regression

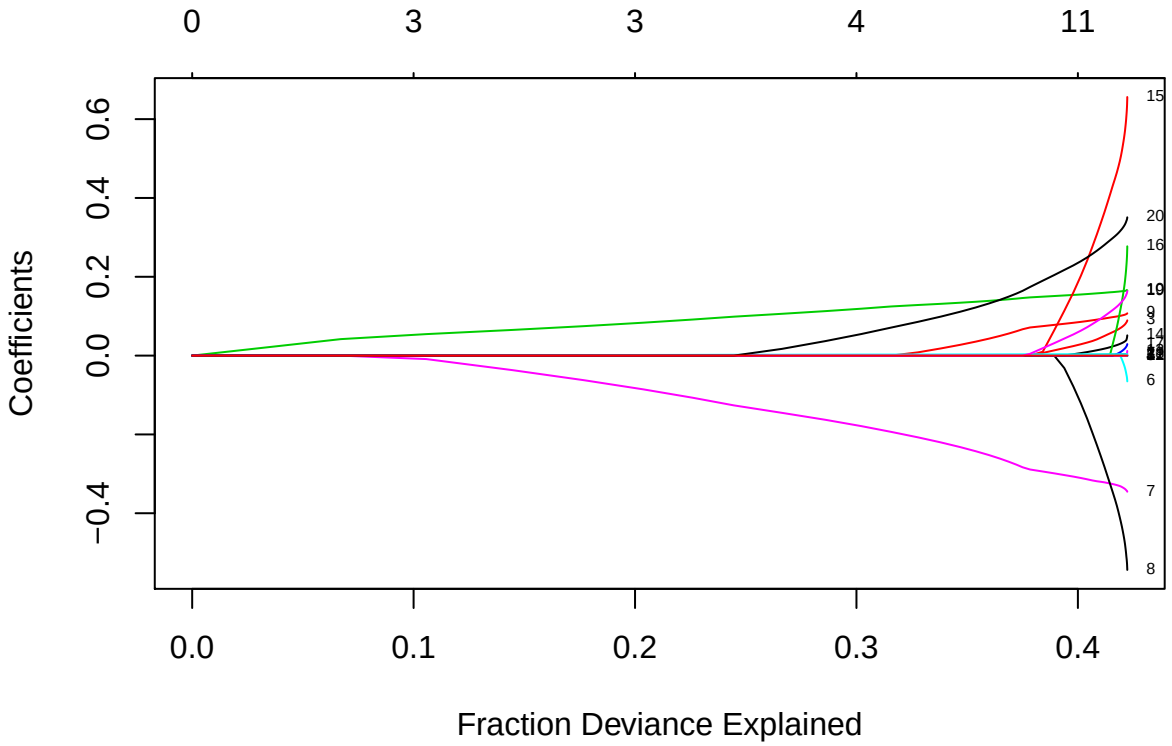
A lasso regression is similar to a ridge regression in that it is a shrinkage method, but uses a different type of penalty term. In this type of model, some coefficient are shrunk to 0. The same five variables explain the largest part of the fractional deviance.

The test MSE of the optimal LASSO regression is again very similar to that of the MLR. This method is therefore neither relevant on this dataset. The log lambda at optimum is near -8. Lambda equals 0.00036 and the test MSE equals 0.379. Again, this lambda is too low to have any significant impact on the estimation and none of the coefficients are shrunk to 0 at this lambda.

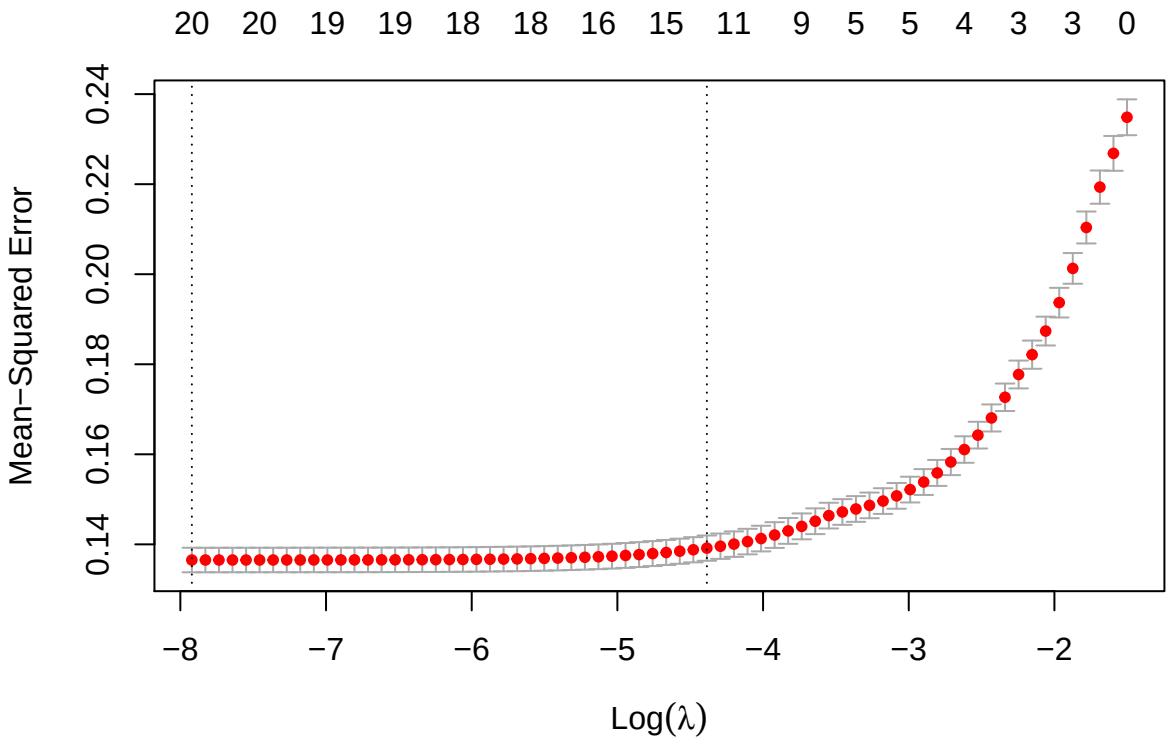
```
fit.lasso <- glmnet(x,y)
plot(fit.lasso, xvar="lambda", label= TRUE)
```




```
plot(fit.lasso, xvar="dev", label= TRUE)
```




```
cv.lasso <-cv.glmnet(x, y)
# Again, 8, 15, 7, 20.

plot(cv.lasso)
```




```
# Use very small lambda, again
## coefficent vector corresponding to the mse which is within
# one standard error of the lowest mse using the best lambda.
coef(cv.lasso)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                          1
## (Intercept)                                   4.165290e+00
## (Intercept)                                   .
## review_scores_rating                          2.067228e-03
## host_is_superhost                             4.570986e-02
## host_listings_count                           .
## host_identity_verified                        .
## room_typeHotel room                           .
## room_typePrivate room                        -3.202302e-01
## room_typeShared room                         -2.559487e-01
## bathrooms                                     9.280449e-02
## bedrooms                                      1.586917e-01
## minimum_nights                                .
## number_of_reviews                            -5.087404e-05
## cancellation_policymoderate                   .
## cancellation_policystrict_14_with_grace_period 1.563968e-02
## cancellation_policysuper_strict_30            3.370446e-01
## cancellation_policysuper_strict_60            .
## instant_bookableTRUE                          .
## cleaning_fee                                  3.411724e-03
## location_3waysModerate                        9.394693e-02
## location_3waysnear_centre                     2.738160e-01
## host_since_duration                           .
```

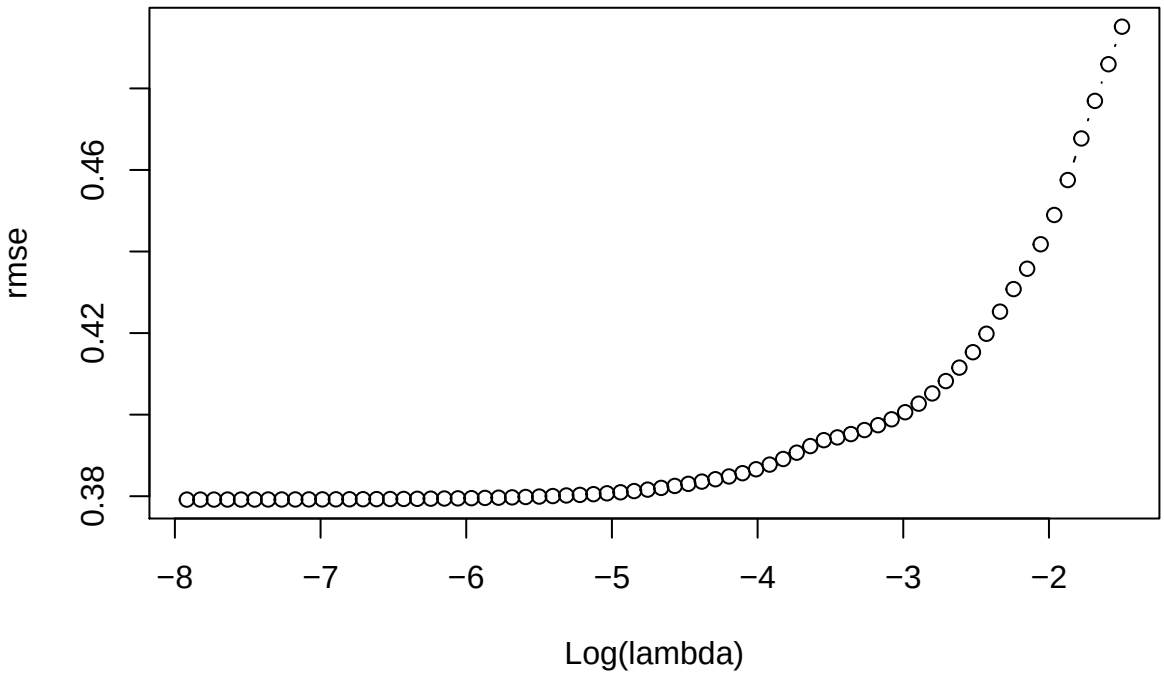
```r
## coefficient vector corresponding to the lowest mse using the best lambda
coef(glmnet(x,y, lambda=cv.lasso$lambda.min))
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                          s0
## (Intercept)                                    3.940950e+00
## (Intercept)                                    .
## review_scores_rating                           3.632109e-03
## host_is_superhost                              8.941803e-02
## host_listings_count                           -5.363794e-04
## host_identity_verified                        -5.333925e-04
## room_typeHotel room                           -6.555163e-02
## room_typePrivate room                         -3.453835e-01
## room_typeShared room                          -5.437931e-01
## bathrooms                                      1.074298e-01
## bedrooms                                       1.658676e-01
## minimum_nights                                -2.592541e-04
## number_of_reviews                             -3.362666e-04
## cancellation_policymoderate                    1.237059e-02
## cancellation_policystrict_14_with_grace_period 5.173575e-02
## cancellation_policysuper_strict_30             6.563447e-01
## cancellation_policysuper_strict_60             2.775788e-01
## instant_bookableTRUE                           2.921495e-02
## cleaning_fee                                   3.470366e-03
## location_3waysModerate                         1.636307e-01
## location_3waysnear_centre                      3.509282e-01
## host_since_duration                           -1.841013e-05
```

```r
## test MSE
lasso.train <-glmnet(x[train,], y[train])
pred.test <-predict(lasso.train, x[-train,])
dim(pred.test)
```

```
## [1] 4506   70
```

```r
rmse <-sqrt(apply((y[-train]-pred.test)^2,2,mean))
plot(log(lasso.train$lambda), rmse, type="b", xlab="Log(lambda)")
```




```
lambda.best <-lasso.train$lambda[order(rmse)[1]]
lambda.best
```

```
## [1] 0.0003641836
```

```
mseLasso <- min(rmse)
mseLasso
```

```
## [1] 0.3791684
```