Advanced Methods for Regression and Classification

Exercise 3

Bosse Behrens , st.id: 12347333

```
load("building.RData")
library(glmnet)
```

```
## Lade nötiges Paket: Matrix
```

```
## Loaded glmnet 4.1-8
```

Setting up the training and testing data split the same as in exercise 2.

```
set.seed(1234)
n <- nrow(df)
train <- sample(1:n, round((2/3) * n))
test<-(1:n)[-train]

y_train <- df[train,"y"]
y_test <- df[test,"y"]
```

Task 1

a)

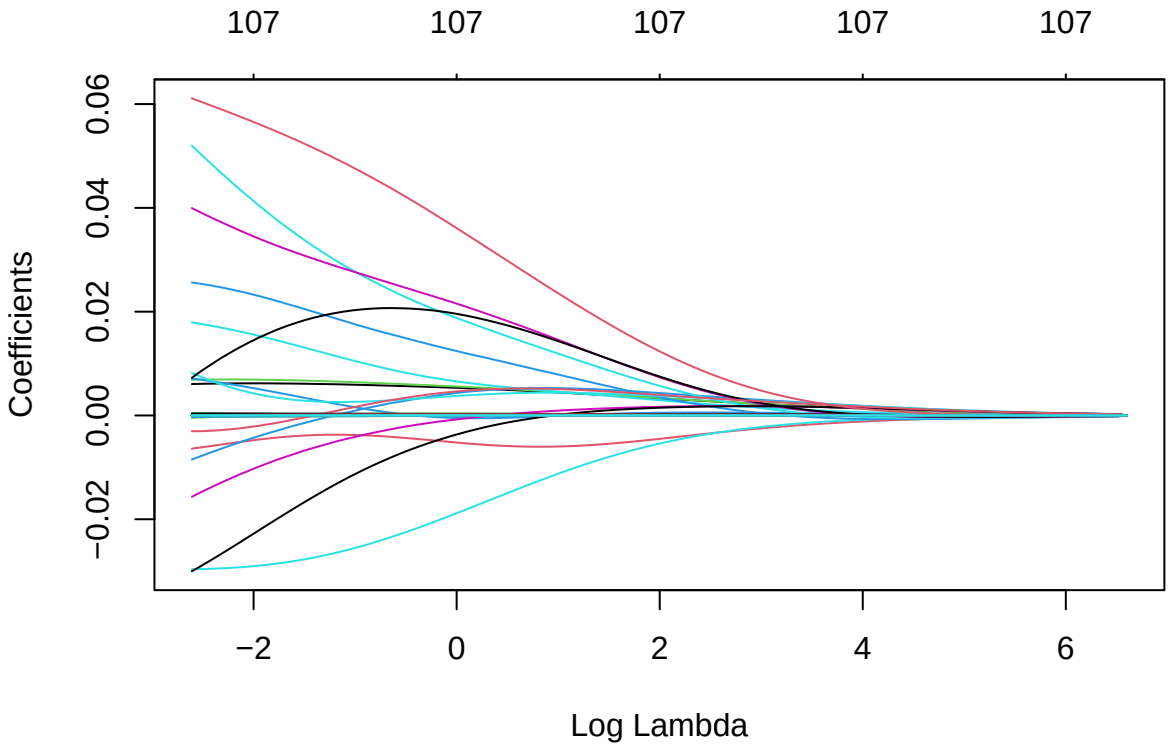
Setting up the model with the spcified parameters.

```
ridge_model <- glmnet(as.matrix(df[train,-1]),df[train,"y"],alpha=0)

head(ridge_model$lambda)
```

```
## [1] 736.6616 671.2186 611.5894 557.2575 507.7522 462.6449
```

```
plot(ridge_model, xvar="lambda")
```

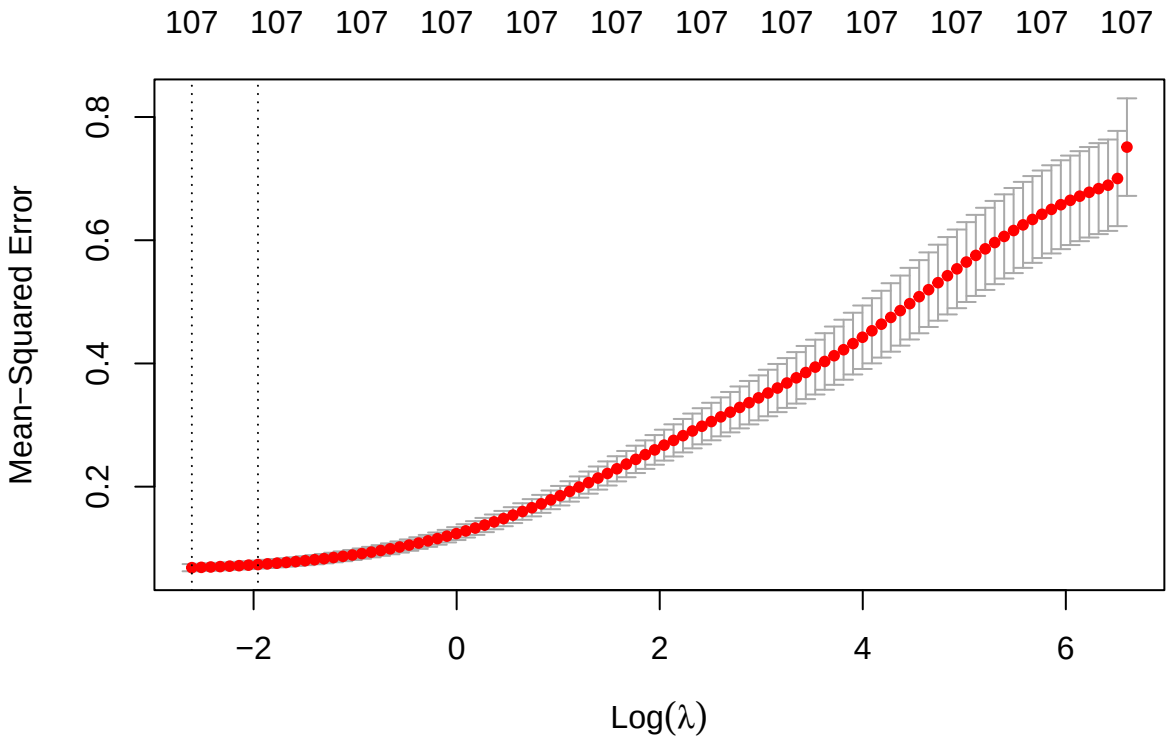


The default values for the lambda parameter is a computed sequence of values that decrease, starting from a large value where all coefficients are nearly zero down to a way smaller value to cover a good range of models with varying strengths. In the Elastic Net regression glmnet() uses, when alpha α is zero it adds only an L" penalty to the regression which makes all coefficients go towards zero but not completely, so does no variable selection. This reduces multicolinearity among predictors and gives reduced influence on predictors that are less correlated with the response.
In the plot we can see the Ridge regression coefficients for varying values of the tuning parameter λ. With increasing λ, the coefficients are more and more shrunk towards zero. The solution at the very left is the LS solution. On top we can see the number of variables in the model.

b)

Setting up the cv.glmnet().

```
ridge_cv <- cv.glmnet(as.matrix(df[train,-1]),df[train,"y"],alpha=0)
plot(ridge_cv)
```



The plot shows the Mean-Squared Error (red dots) with the standard errors (grey intervals) over the log-transformed λ parameter values. The left vertical line is the smallest MSE and the right one is the optimal λ for which the MSE is below the smallest MSE plus its standard error. By default this λ is selected with the one-standard error rule. We now apply this rule ("1se") to the model and get the selected coefficients.

```
coef_ridge <- coef(ridge_cv,s="lambda.1se")
head(coef_ridge)
```

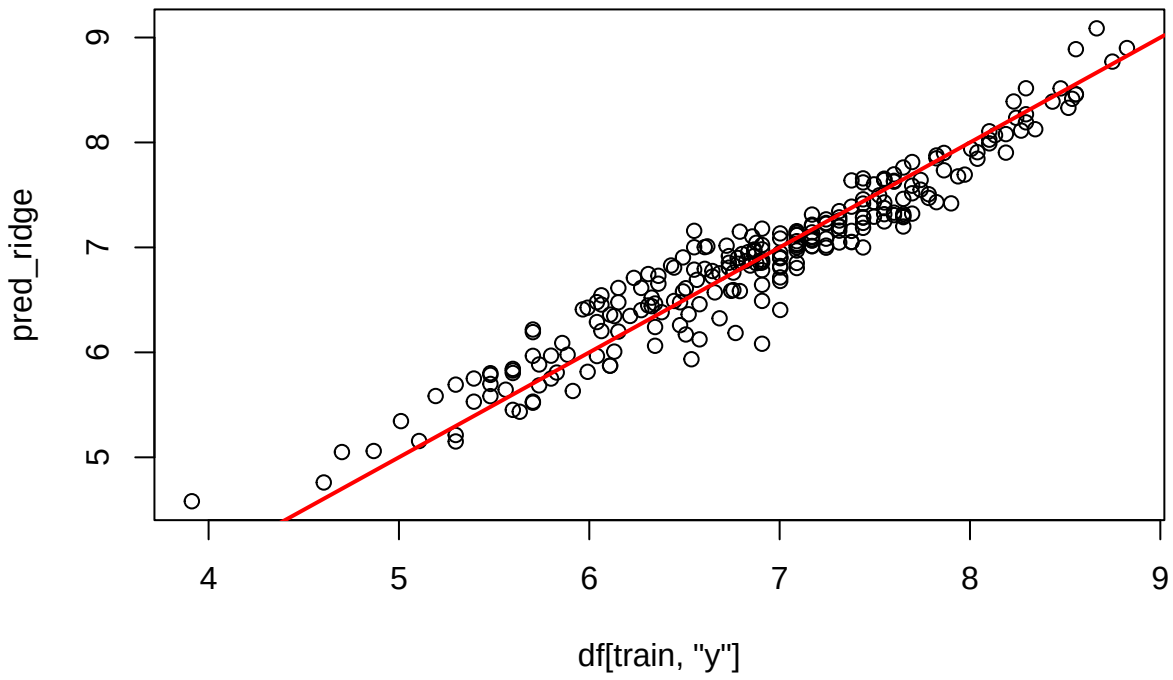
```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)         1.621419082
## START.YEAR          0.006191409
## START.QUARTER      -0.004657751
## COMPLETION.YEAR     0.006898178
## COMPLETION.QUARTER  0.005102613
## PhysFin1           -0.028959072
```

c)

We now use the optimal model we selected to compute the the predicted response for y as well as visualize them against the observed y-data and compute the RMSE.

```
pred_ridge <- predict(ridge_cv, newx=as.matrix(df[train,-1]),s="lambda.1se")

plot(df[train,"y"], pred_ridge)
abline(0, 1, col = "red", lwd = 2)
```




```
n_train <- length(y_train)

RMSE_train_ridge <- sqrt((1/n_train)*sum((y_train - pred_ridge)^2))
print(RMSE_train_ridge)
```

```
## [1] 0.2390407
```

As we can see the data points seem closely and randomly distributed around the xy-line which indicates a well-suited model to the data. The RMSE is 0.2390407 and therefore small enough we can call it good as well as similar to the one from Exercise 2 where we used regsubset() and slightly better than in Exercise 3 where we used pls() and pcr().

Task 2

a)

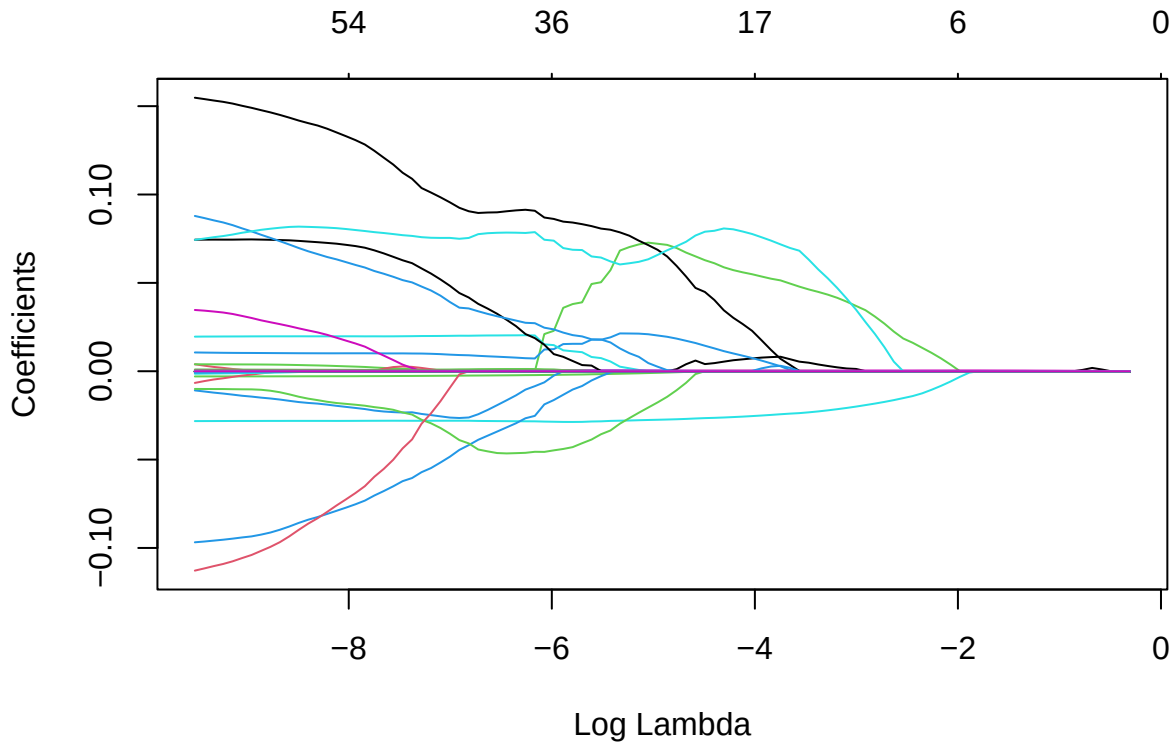
We do the same as in task 1 but using $\alpha = 1$ as parameter. this has th effect that the penalty term in the Elastic Net regression in glmnet() now only adds a L1 penalty. This means we now got a Lasso-regression that shrinks some coefficients to zero and therefore creates a variable selection. the lambda again takes some decreasing values as default sequence.

```
lasso_model <- glmnet(as.matrix(df[train,-1]),df[train,"y"],alpha=1)

head(lasso_model$lambda)
```

```
## [1] 0.7366616 0.6712186 0.6115894 0.5572575 0.5077522 0.4626449
```

```
plot(lasso_model, xvar="lambda")
```

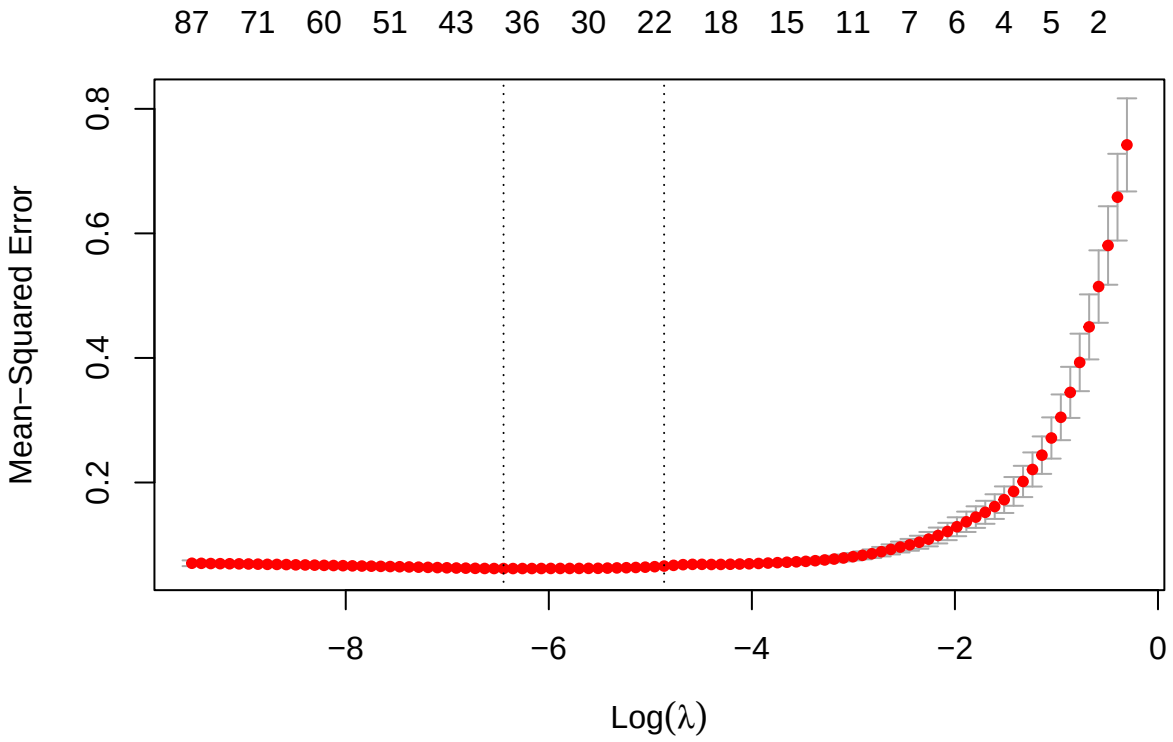


As we can see now for increasing lambda the coefficients are not driven close to zero but actually become zero one after another. It behaves similar to Ridge-regression in terms how it works, with the difference in the penalty term that also allows for variable selection and does not keep all original predicotrs in the model. On the top we can again see the number of predictors still in the model.

b)

We now set up again cv.glmnet with our Lasso-model.

```
lasso_cv <- cv.glmnet(as.matrix(df[train,-1]),df[train,"y"],alpha=1)
plot(lasso_cv)
```



The plot description is the same as before with the MSE, the two vertical lines and on top the number of predictors in the model (that actually decrease now). As we can see using the 1se rule again the optimal model would result in (around?) 17 predictors.

```
coef_lasso <- coef(lasso_cv,s="lambda.1se")
head(coef_lasso)
```

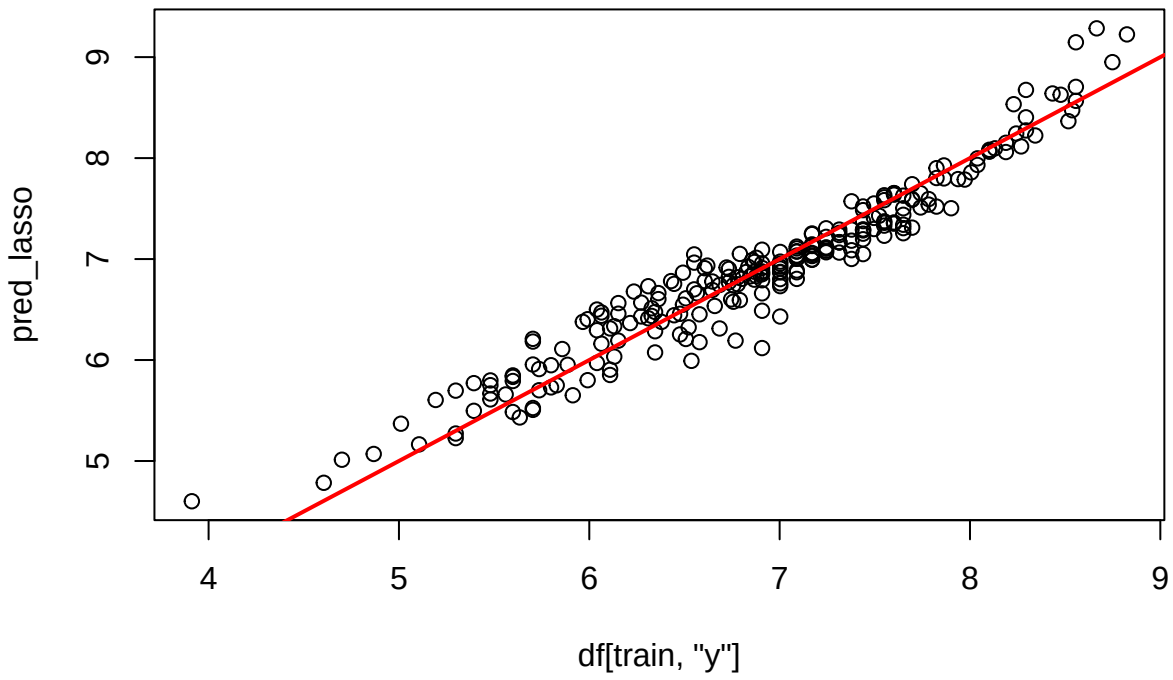
```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)        -2.00815431
## START.YEAR          .
## START.QUARTER       .
## COMPLETION.YEAR     0.07150392
## COMPLETION.QUARTER  0.01950070
## PhysFin1           -0.02733476
```

c)

We again plot the predicted response vs the true and compute the RMSE.

```
pred_lasso <- predict(lasso_cv, newx=as.matrix(df[train,-1]),s="lambda.1se")

plot(df[train,"y"], pred_lasso)
abline(0, 1, col = "red", lwd = 2)
```




```
n_train <- length(y_train)

RMSE_train_lasso <- sqrt((1/n_train)*sum((y_train - pred_lasso)^2))
print(RMSE_train_lasso)
```

```
## [1] 0.2277203
```

The plot of the true vs predicted response again looks well randomly and small dsitributed around the xy-line only with some slight derivations for the higher and lower values and one more sever outlier as the lowest value. This also results in the RMSE being slightly worse than for the RIdge regression, but better than in Exercise 3 with pls() and pcr() and around the same as in Exercise 2 with regsubset().

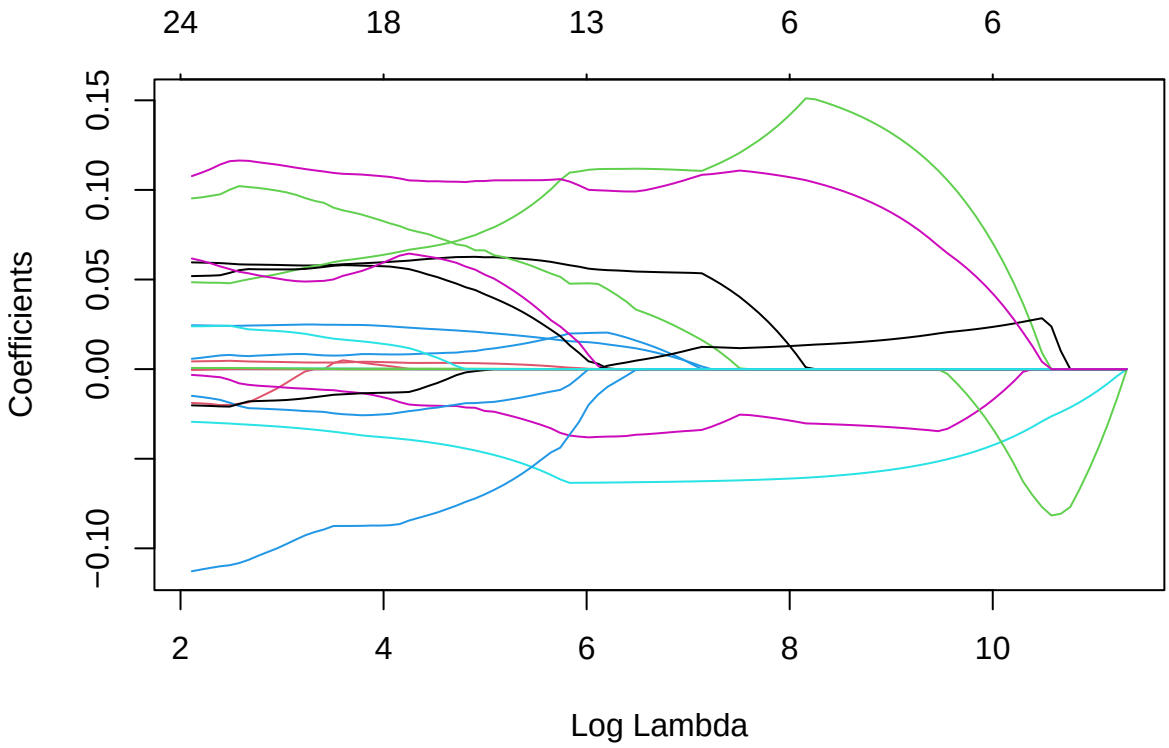
Task 3

a)

We again set up the adaptive lasso model. Default value for α is zero, meaning we have a lasso-model but we do specify a penalty factor with the inverse Ridge coefficients as weights.

```
alasso_model <- glmnet(as.matrix(df[train,-1]),df[train,"y"],penalty.factor=1/abs(coef_ridge[-1]))

plot(alasso_model,xvar="lambda")
```

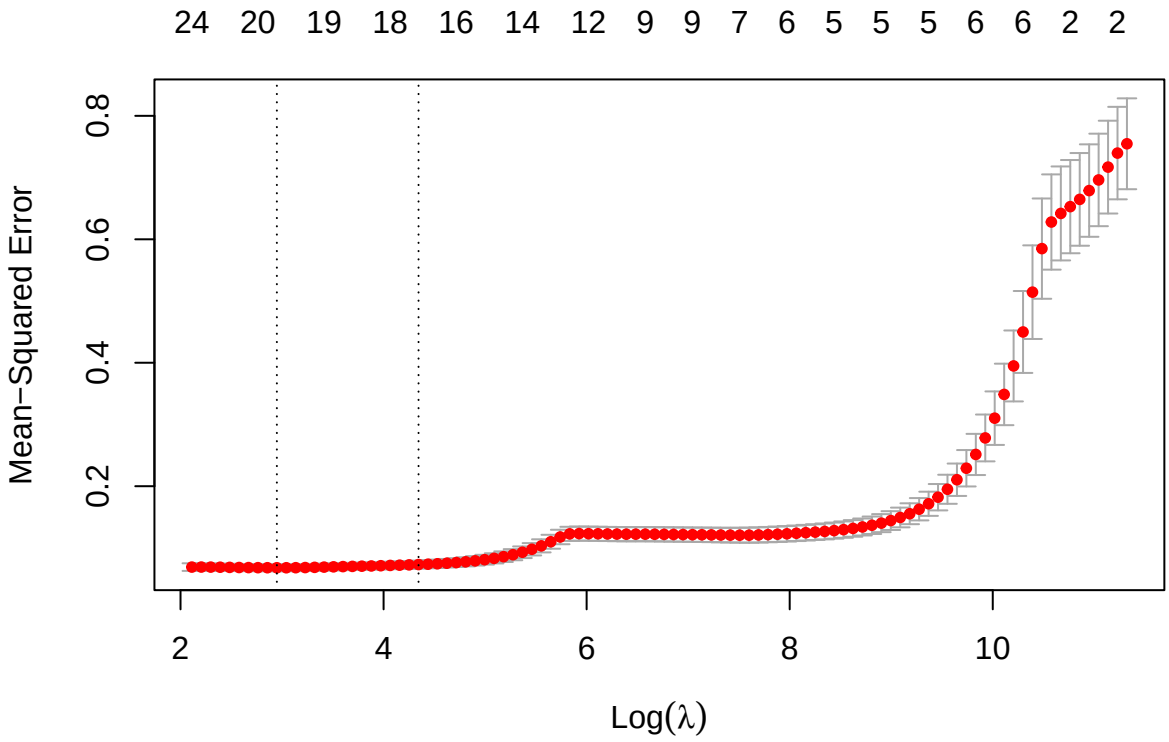


Since we a Lasso regression, again the coefficients shrink to zero, but they do take longer to do so and need higher λ values in this adaptive lasso model than in the normal Lasso-model. Besides that the meta-data of the plot is the same before.

b)

We now use cv-glmnet() again.

```
alasso_cv <- cv.glmnet(as.matrix(df[train,-1]),df[train,"y"], penalty.factor=1/abs(coef_ridge[-1]))
plot(alasso_cv)
```



The plot again shows MSE, number of coefficients,a s well as the two vertical lines indicating lowestMSE/optimal MSE by 1se rule. The points this time follow a line that is less smooth. Also the optimal λ value is significantly larger than in the normal lasso-model and the optimal model has less coefficeints.
We now again apply the 1se rule to get the optimal model-coefficients.

```
coef_alasso <- coef(alasso_cv,s="lambda.1se")
head(coef_alasso)
```

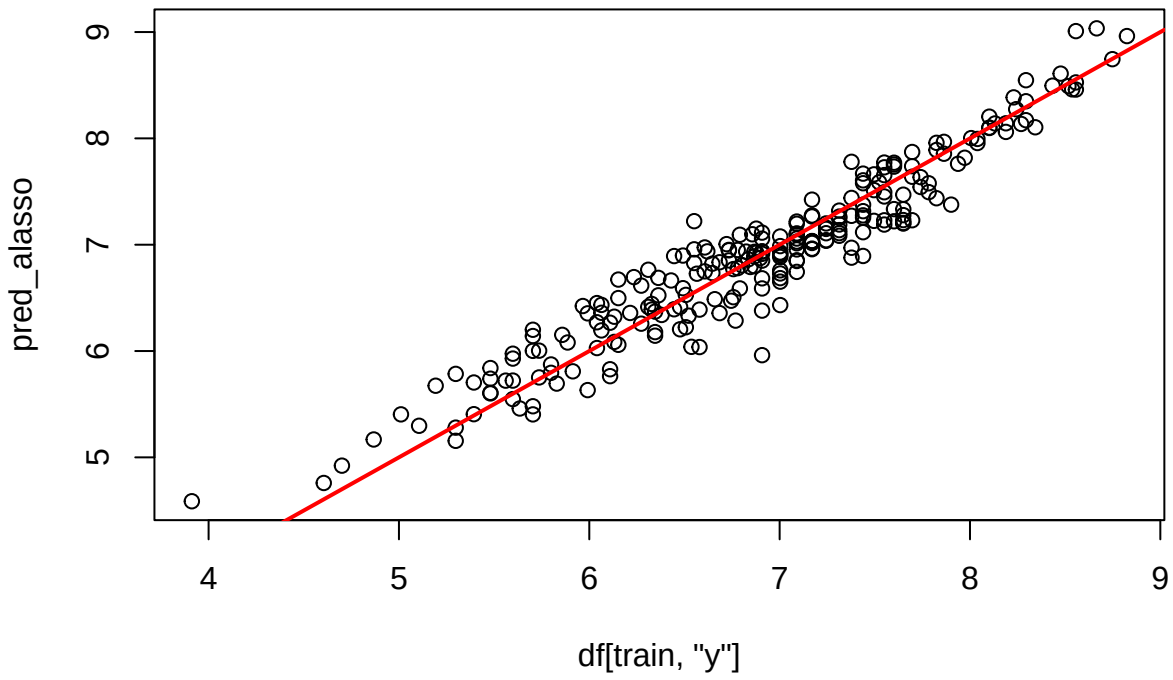
```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)        -5.85239916
## START.YEAR          0.06104224
## START.QUARTER       .
## COMPLETION.YEAR     0.06737805
## COMPLETION.QUARTER  0.02282256
## PhysFin1           -0.04001819
```

c)

We now again plot true vs predicted response and compute the RMSE.

```
pred_alasso <- predict(alasso_cv, newx=as.matrix(df[train,-1]),s="lambda.1se")

plot(df[train,"y"], pred_alasso)
abline(0, 1, col = "red", lwd = 2)
```




```
n_train <- length(y_train)

RMSE_train_alasso <- sqrt((1/n_train)*sum((y_train - pred_alasso)^2))
print(RMSE_train_alasso)
```

```
## [1] 0.2519979
```

```
options(scipen = 999)
df <- cbind(round(coef_alasso, 8), round(coef_lasso, 8))
colnames(df) <- c("adaptive Lasso coef", "Lasso coef")
df <- df[!(is.na(df[, "adaptive Lasso coef"]) & is.na(df[, "Lasso coef"]) |
           df[, "adaptive Lasso coef"] == 0 & df[, "Lasso coef"] == 0), ]
print(df)
```

```
## 31 x 2 sparse Matrix of class "dgCMatrix"
##                     adaptive Lasso coef  Lasso coef
## (Intercept)                -5.85239916 -2.00815431
## START.YEAR                  0.06104224  .
## COMPLETION.YEAR             0.06737805  0.07150392
## COMPLETION.QUARTER          0.02282256  0.01950070
## PhysFin1                   -0.04001819 -0.02733476
## PhysFin4                    .          -0.00001883
## PhysFin5                    .          -0.00073346
## PhysFin6                    .           0.00027854
## PhysFin7                    0.00344385  .
## PhysFin8                    0.00029303  0.00041930
## Econ4                       0.00851467  .
```

```
## Econ10                    -0.02014969  .
## Econ14                     .           0.00001718
## Econ19                     .           0.00000010
## Econ4.lag1                 0.05440952  .
## Econ8.lag1                 .           0.00009308
## Econ9.lag1                 .           0.00000014
## Econ10.lag1                0.07653338  0.06496360
## Econ4.lag2                -0.02283316  .
## Econ8.lag2                 .           0.00003478
## Econ9.lag2                 .           0.00000337
## Econ10.lag2                0.06368414  0.00061800
## Econ1.lag3                 .           0.00001999
## Econ4.lag3                -0.01131010 -0.01448406
## Econ8.lag3                 .           0.00007401
## Econ10.lag3               -0.08298913  .
## Econ14.lag3                .           0.00003338
## Econ1.lag4                 .           0.00002564
## Econ4.lag4                 0.01013357  .
## Econ10.lag4                0.10513842  0.06841797
## Econ14.lag4                .           0.00001129
```

The resulting plot looks like the data points are a bit better normally distributed around the xy-line, but only very slightly. The outliers for the high and small valeus are also very slightly less off, but overall it is very similar to the normal Lasso-model. The RMSE is also slightly higher, but not by much. When comparing the resulting coefficients we can observe the ones for the normal Lasso regression are way smaller on average in terms of absolute values. Overall adaptive Lasso when weighted by the Ridge weights emphasizes more the reliable predictors since due to the weights it puts less emphasis on predictors with less influence and is less likely to eliminate the influential predictors identified by Ridge. When looking at our results though, we cannot observe that big of a difference. Adaptive Lasso performs slightly worse in terms of RMSE but not by much so we cannot really say in this case which one is the better model or more plausible than the other.