Advanced Methods for Regression and Classification

Exercise 1

Bosse Behrens , st.id: 12347333

Task 1

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

```
## Lade nötiges Paket: lattice
```

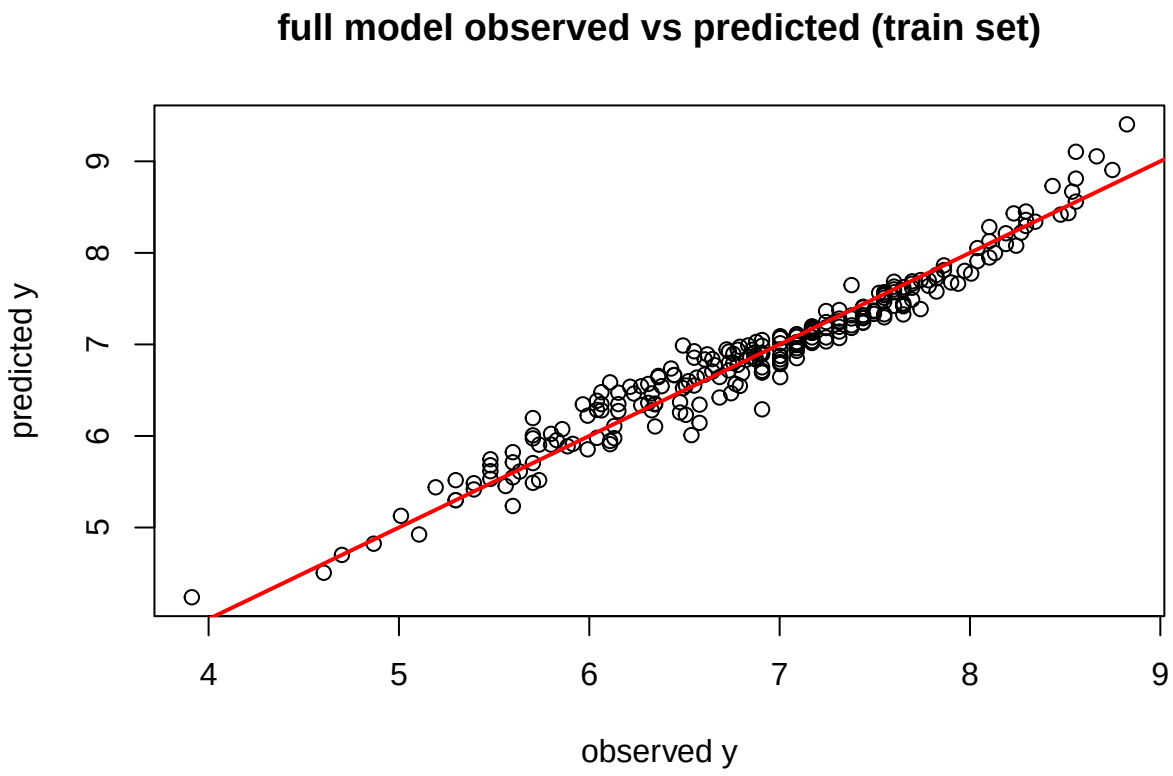
```
## Lade nötiges Paket: robustbase
```

```
library(leaps)
```

a)

We split the data into training and testing set and create the full model using all 108 predictors. Then we plot the values of the resposnse vs. the predicted values and calculate the RMSE for the training data.

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

lm_model <- lm(y ~ ., data = df, subset = train)

pred_lm_train <- predict(lm_model, newdata = df[train, ])
obs_train <- df[train,"y"]

plot(obs_train, pred_lm_train,
     main = "full model observed vs predicted (train set)",
     xlab = "observed y",
     ylab = "predicted y")
abline(0, 1, col = "red", lwd = 2)
```




```
n_train <- length(obs_train)
RMSE_train <- sqrt((1/n_train)*sum((obs_train - pred_lm_train)^2))


cat("RMSE of training data:", RMSE_train)
```

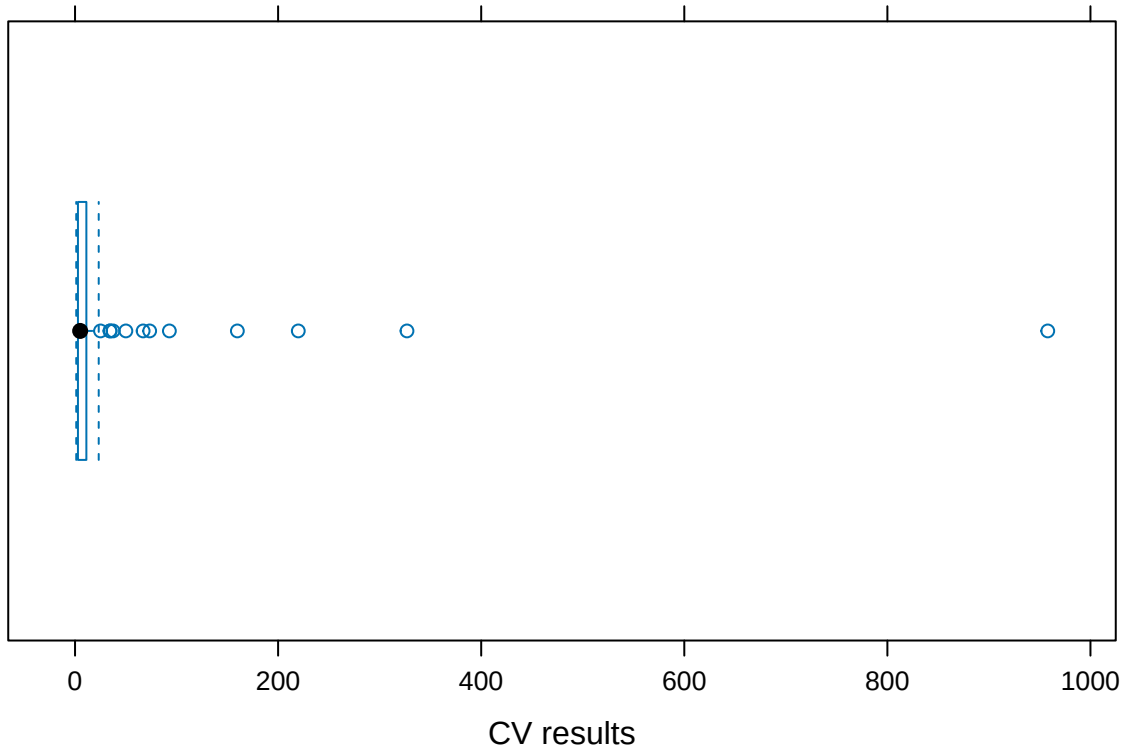
```
## RMSE of training data: 0.1884366
```

```
summary_lm <- summary(lm_model)
```

b)

We use the cvFit() function from the cvTools package to perform 5-fold cross-validation with 100 replications and RMSE as cost function on the model with the training data set and plot the resulting distribution of the RMSE. As we can see the 50% interquartile and median are close to zero as we would expect for the RMSE from the data with a response that has value sin range 5-9 but there seem to be quite a few very big outliers with one even over 600. This means the model performs very poorly on some subsets due to overfitting or extreme outliers in the data. A solution would be to reduce the number of predictors and choose a better subset of predictors.

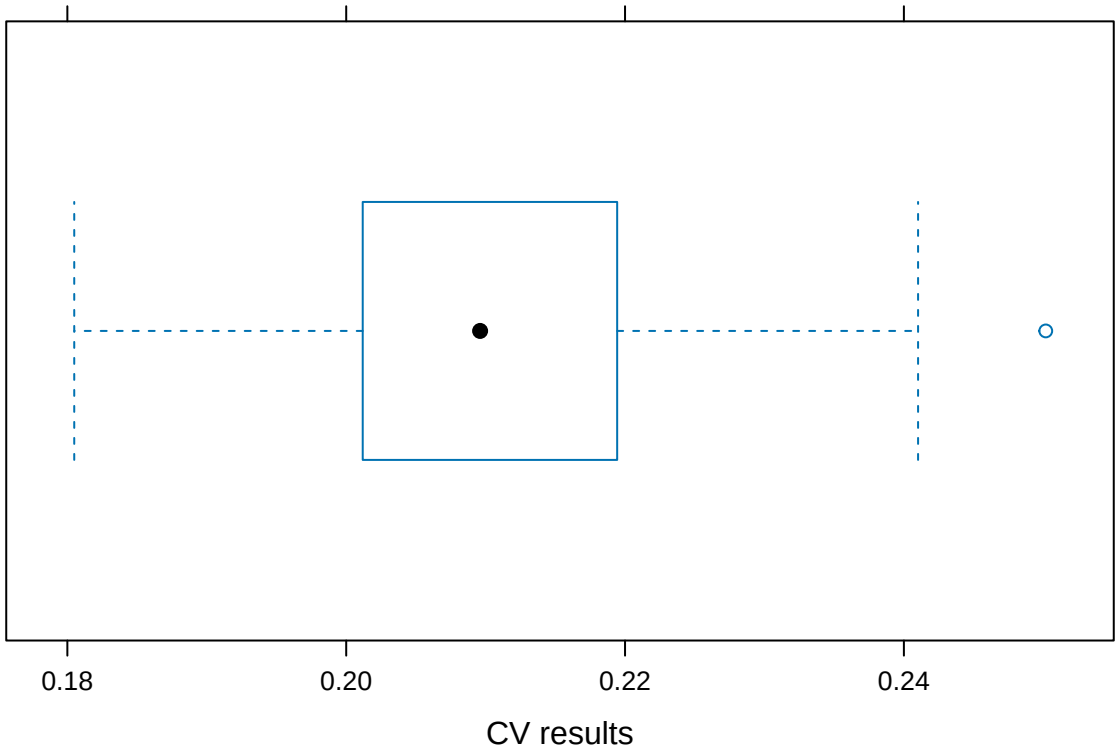
```
set.seed(1234)
cv_res <- cvFit(lm_model, data = df[train, ], y = obs_train, cost = rmspe, K = 5, R = 100)

plot(cv_res)
```



c)

We perform the same cross validation again but with a trim of the 0.1 highest RMSE results. As we can observe in the plot the RMSE distribution is way better now ranging from ~0.15 to ~0.21. This si what we would expect from a model that is a good fit on the data. Also there are no outliers at all this time, meaning less than 0.1 are extreme big values.

```
set.seed(1234)
cv_rest <- cvFit(lm_model, data = df[train, ], y = obs_train, cost = rtmspe, K = 5, R = 100)

plot(cv_rest)
```



d)

We plot the response vs the fitted values of the test data and calculate the RMSE. As we can see the RMSE is way worse than for the training data and would fall into the outliers of the RMSE in part c). In the plot we can clearly see that some points are far away from the $x = y$ line.

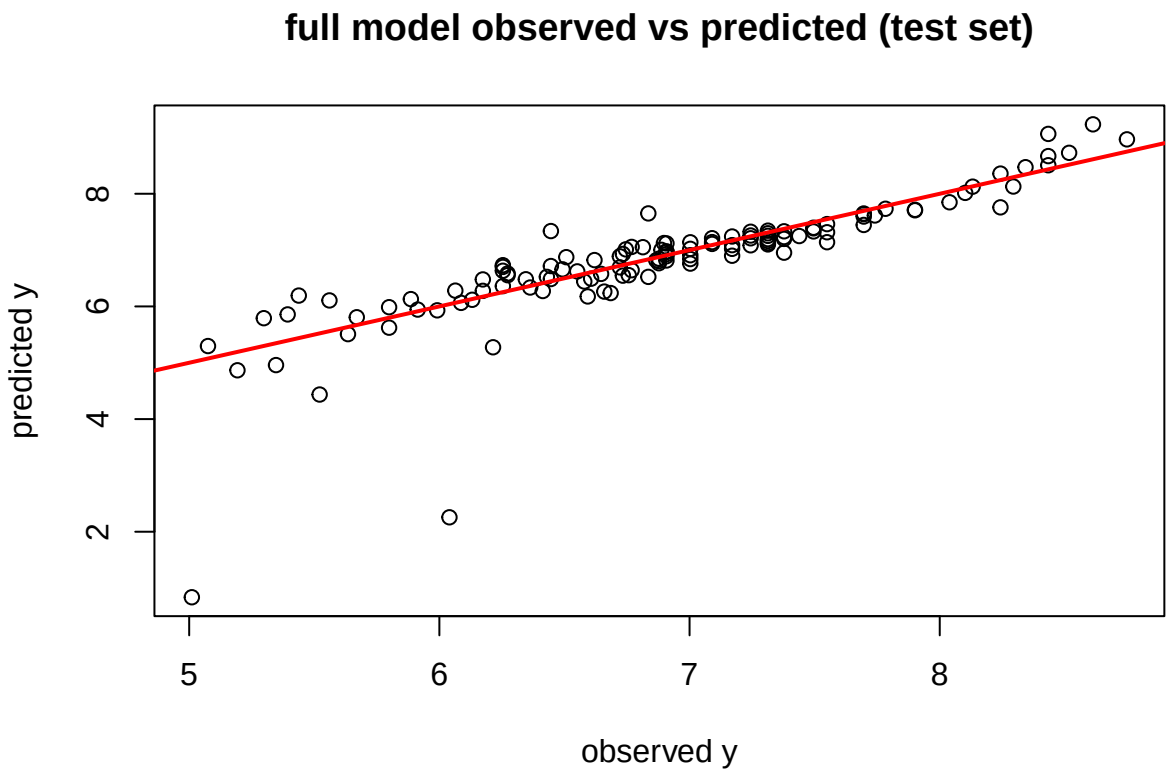
```
pred_lm_test <- predict(lm_model, newdata = df[test, ])
```

```
## Warning in predict.lm(lm_model, newdata = df[test, ]): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
obs_test <- df[test,"y"]

n_test <- length(obs_test)
n_test
```

```
## [1] 124
```

```
RMSE_test <- sqrt((1/n_test)*sum((obs_test - pred_lm_test)^2))


plot(obs_test, pred_lm_test,
     main = "full model observed vs predicted (test set)",
     xlab = "observed y",
     ylab = "predicted y")
abline(0, 1, col = "red", lwd = 2)
```




```
cat("RMSE for test data:", RMSE_test)
```

```
## RMSE for test data: 0.5810165
```

Task 2

a)

As a very simple (and possibly sloppy) way to reduce the predictors we simply only include the predictors from the original full model that were significant at the 10 level. This leaves us with only 31 predictors instead of 108. On this we can now use regsubsets() with nvmax = 10 to get the best subsets for 1 to 10 predictors in a reasonable time.

```
sign_p <- rownames(summary_lm$coefficients)[summary_lm$coefficients[, "Pr(>|t|)"] < 0.1]
sign_p <- sign_p[sign_p != "(Intercept)"]
print(sign_p)
```

```
##  [1] "COMPLETION.YEAR"    "COMPLETION.QUARTER" "PhysFin1"
##  [4] "PhysFin2"           "PhysFin3"           "PhysFin5"
##  [7] "PhysFin6"           "PhysFin8"           "Econ3"
## [10] "Econ7"              "Econ8"              "Econ9"
## [13] "Econ10"             "Econ11"             "Econ16"
## [16] "Econ19"             "Econ1.lag1"         "Econ2.lag1"
## [19] "Econ4.lag1"         "Econ10.lag1"        "Econ11.lag1"
## [22] "Econ15.lag1"        "Econ16.lag1"        "Econ19.lag1"
## [25] "Econ1.lag2"         "Econ2.lag2"         "Econ9.lag2"
## [28] "Econ12.lag2"        "Econ19.lag2"        "Econ1.lag3"
```

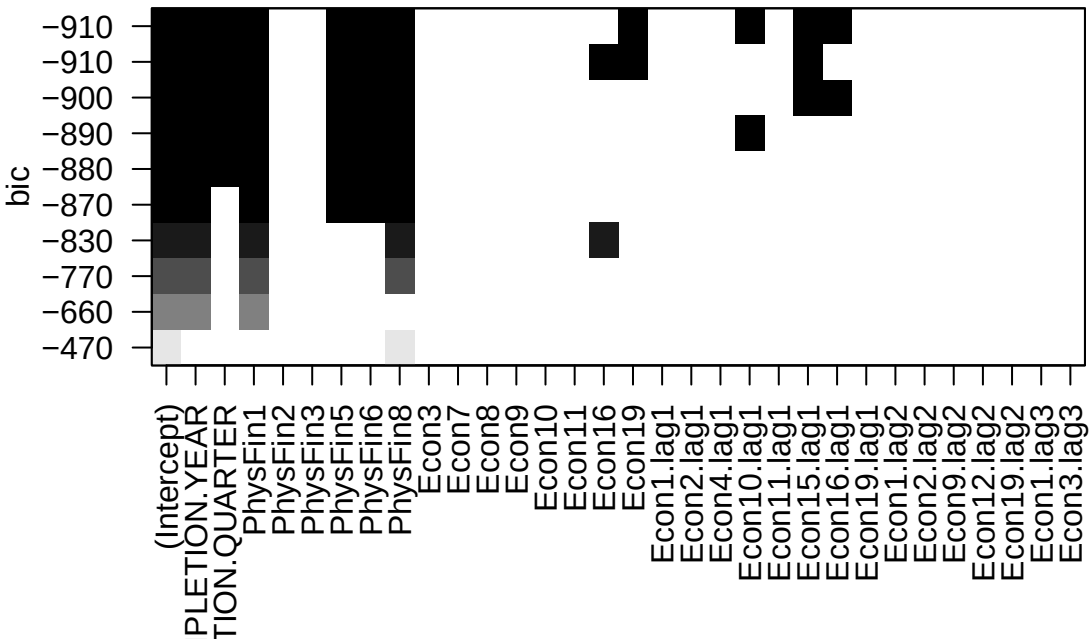
```
## [31] "Econ3.lag3"
```

```
sign_data <- df[, c("y", sign_p)]

best_subset <- regsubsets(y ~ ., data = sign_data, nvmax = 10, really.big = TRUE)
```

b)

We now plot the result object of the regsubset(). We gain a heat map that shows us the best combinations for 1 to 10 predictors and the correspondin g BIC value. Furthermore we can look at the adjusted R squared value of each model. The best model in terms of BIC/AdjR2 isa the model with 10 predictors. The models with 5 to 9 predictors perform only slightly worse so it is possible to argue to also select on of these since there is a balncing between the most promising in termns of evaluation statistics (BIC/AdjR2) and simplicity (fewer predictors). In this case we choose to go with the largest 10 predictors model.

```
plot(best_subset)
```




```
summary_best <- summary(best_subset)

print(summary_best$adjr2)
```

```
##  [1] 0.7255156 0.8359883 0.8821476 0.8999229 0.9109041 0.9156355 0.9188052
##  [8] 0.9216773 0.9233161 0.9249719
```

```
print(summary_best$bic)
```

```
##  [1] -470.1104 -656.7621 -774.8006 -830.7131 -869.0457 -884.4436 -893.7912
##  [8] -902.2927 -905.2660 -908.4967
```

c)

We select the names of the predictors of our chosen model and delete the intercept from them.

```
best_predictors <- summary_best$which[10, ]
selected_predictors <- names(best_predictors)[best_predictors]
selected_predictors <- selected_predictors[selected_predictors != "(Intercept)"]
```

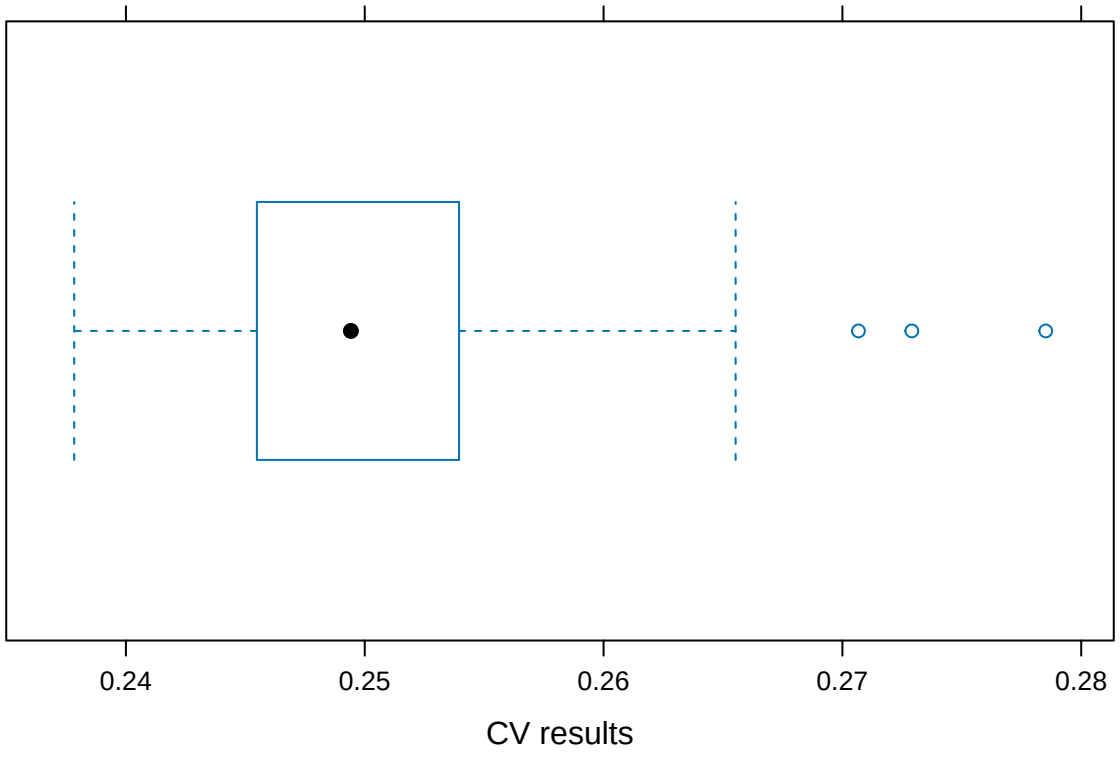
Now we create the new reduced model with chosen best subset.

```
selected_data <- df[ , c("y", selected_predictors)]

best_model <- lm(y ~ ., data = selected_data, subset = train)

summary(best_model)
```

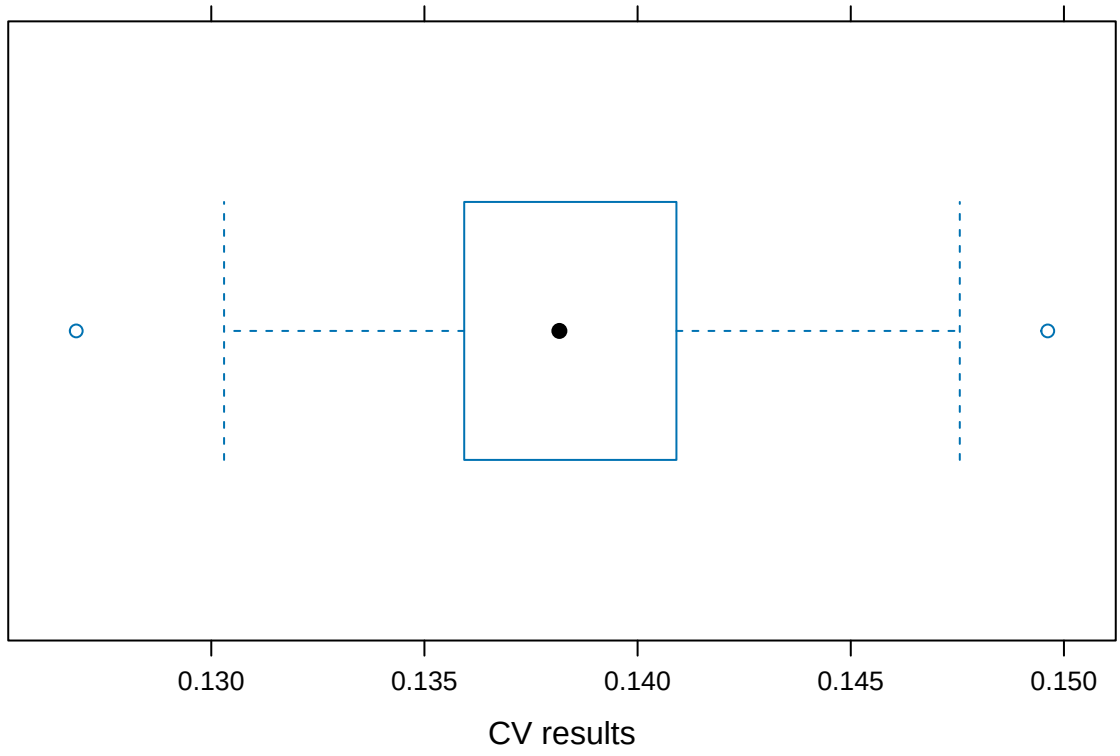
```
##
## Call:
## lm(formula = y ~ ., data = selected_data, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69706 -0.13078  0.01737  0.15278  0.81886
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -6.252e+00  1.109e+00  -5.640 4.83e-08 ***
## COMPLETION.YEAR      1.377e-01  1.650e-02   8.348 5.80e-15 ***
## COMPLETION.QUARTER   4.248e-02  1.395e-02   3.045 0.002591 **
## PhysFin1            -3.021e-02  3.677e-03  -8.215 1.38e-14 ***
## PhysFin5            -3.174e-03  4.727e-04  -6.715 1.38e-10 ***
## PhysFin6             6.629e-04  1.078e-04   6.149 3.28e-09 ***
## PhysFin8             5.099e-04  3.079e-05  16.561  < 2e-16 ***
## Econ19               3.704e-07  1.073e-07   3.452 0.000659 ***
## Econ10.lag1          6.919e-02  1.931e-02   3.582 0.000413 ***
## Econ15.lag1          2.614e-02  8.243e-03   3.171 0.001721 **
## Econ16.lag1         -2.486e-02  7.220e-03  -3.443 0.000679 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2339 on 237 degrees of freedom
## Multiple R-squared:  0.9299, Adjusted R-squared:  0.927
## F-statistic: 314.4 on 10 and 237 DF,  p-value: < 2.2e-16
```

Now we repeat the steps from 1b and 1c using cvFit() once with cost = rmspe and once with the trimmed rtmspe on the subset of predicotrs with the training data. As we can observe now for both the normal RMSE and the trimmed cost functions both distributions of the RMSE are very low. From these consistent and good values we can conclude that the model now fits the data way better than before. Also when looking at the model summary we see that the R squared and F-statistic are very high which supports this conclusion.

```
set.seed(1234)
cv_res_b <- cvFit(best_model, data = df[train, c("y", selected_predictors)], y = obs_train, cost = rmsp

plot(cv_res_b)
```




```
set.seed(1234)
cv_res_bt <- cvFit(best_model, data = df[train, c("y", selected_predictors)], y = obs_train, cost = rtms

plot(cv_res_bt)
```



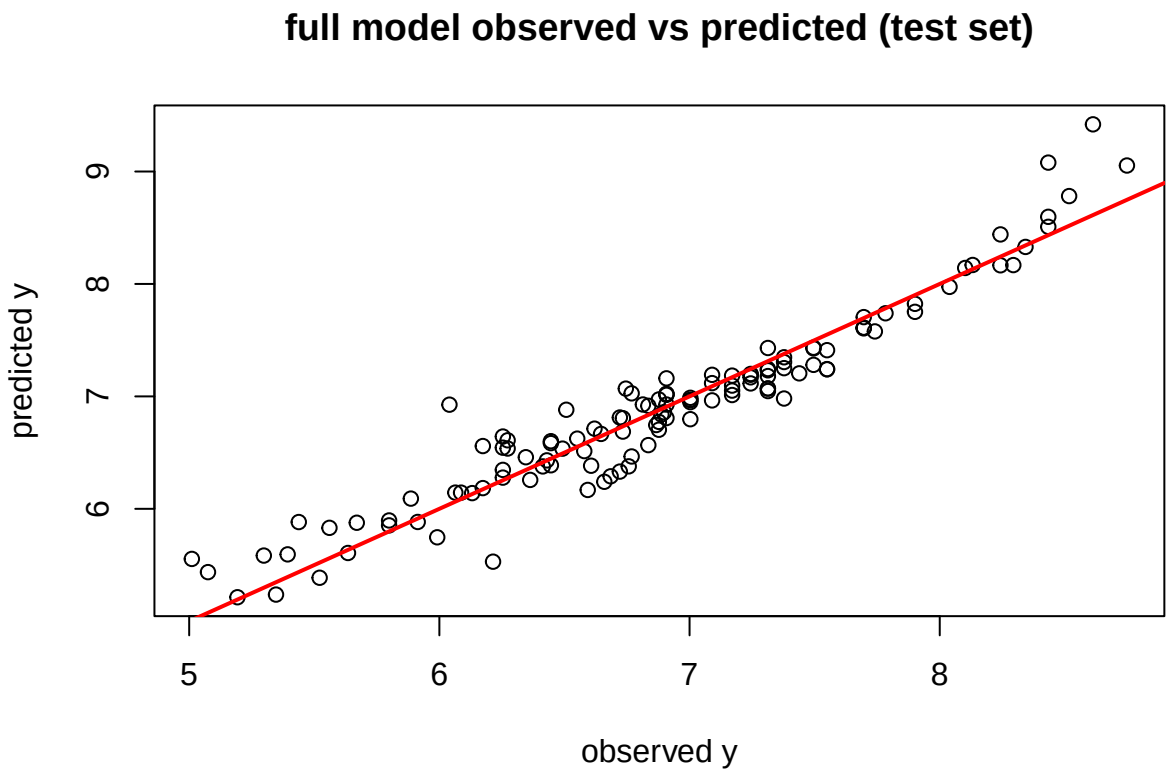
d)

As we can see the RMSE is much lower than on the previous full model using the test data. It also is similar now to the RMSE of the training data which again supports the conclusion that the model is now now well suited to the data. The plot furthermore shows that the resposne vs fitted values are randomly scattered around $x = y$ without extreme outleirs as before.

```
pred_lmb_test <- predict(best_model, newdata = df[test, ])
obs_test <- df[test,"y"]

n_test <- length(obs_test)
n_test
```

```
## [1] 124
```

```
RMSE_testb <- sqrt((1/n_test)*sum((obs_test - pred_lmb_test)^2))


plot(obs_test, pred_lmb_test,
     main = "full model observed vs predicted (test set)",
     xlab = "observed y",
     ylab = "predicted y")
abline(0, 1, col = "red", lwd = 2)
```




```
print(RMSE_testb)
```

```
## [1] 0.233771
```