Statistical Learning, Tutorato #5

Veronica Vinciotti, Marco Chierici

April 12, 2021

Exercise 1

The `College` dataset (contained in the `ISLR` library) collects statistics measured on 18 variables for 777 US colleges. Some of the variables include whether the college is a private or public institution, the number of application received, the number of applications accepted, etc. (for full details: `?College`)

Here, we want to predict the number of applications received using the other variables.

a. Split the data into a training/test set.
b. Fit a least squares linear model on the training set set and evaluate the error on the test set.
c. Fit a ridge regression model on the training set, choosing λ by cross-validation. Report the test error.
d. Fit a lasso model on the training set, choosing λ by cross-validation. Report the test error and the number of non-zero coefficient estimates.
e. Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from the different approaches? Draw a plot of the predictions vs the true response values on the test data for all of the models.

Hints:

- To perform ridge regression and the lasso, use the `glmnet` package;
- The syntax is `ridge_mod <- glmnet(x, y, alhpha, lambda)`:
  - note that **no formula notation** is supported;
  - `alpha=0` performs ridge regression;
  - `alpha=1` performs the lasso
  - the optional parameter `lambda` is used to pass a grid of possible values of λ; if this parameter is omitted, the `glmnet()` function performs ridge regression for an automatically selected range of λ values.
- It may be convenient to use `model.matrix()` to prepare the `x` data for `glmnet()`;
- In order to get the predictions on new data, given a value L of λ: `predict(ridge_mod, s=L, newx=x_test)`
- In order to obtain the coefficients given a value L of λ: `predict(ridge_mod, s=L, type="coefficients")`
- Instead of arbitrarily choosing λ, it would be better to use cross-validation to choose the optimal λ. Use the built-in cross-validation function `cv_mod <- cv.glmnet(x, y, alpha, lambda)`, with the same syntax as `glmnet()`. The optimal lambda can be found in `cv_mod$lambda.min`.

```r
data(College)
summary(College)
```

```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
```

```
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
##     Terminal       S.F.Ratio      perc.alumni        Expend
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
##    Grad.Rate
##  Min.   : 10.00
##  1st Qu.: 53.00
##  Median : 65.00
##  Mean   : 65.46
##  3rd Qu.: 78.00
##  Max.   :118.00
```

```r
n_obs <- nrow(College)

### a) data partitioning
set.seed(11)
# training set percentage
train_perc <- 0.75
train <- sample(1:n_obs, train_perc * n_obs)
data_train <- College[train, ]
data_test <- College[-train, ]

### b) least squares linear regression
fit_lm <- lm(Apps ~ ., data = data_train)
summary(fit_lm)
```

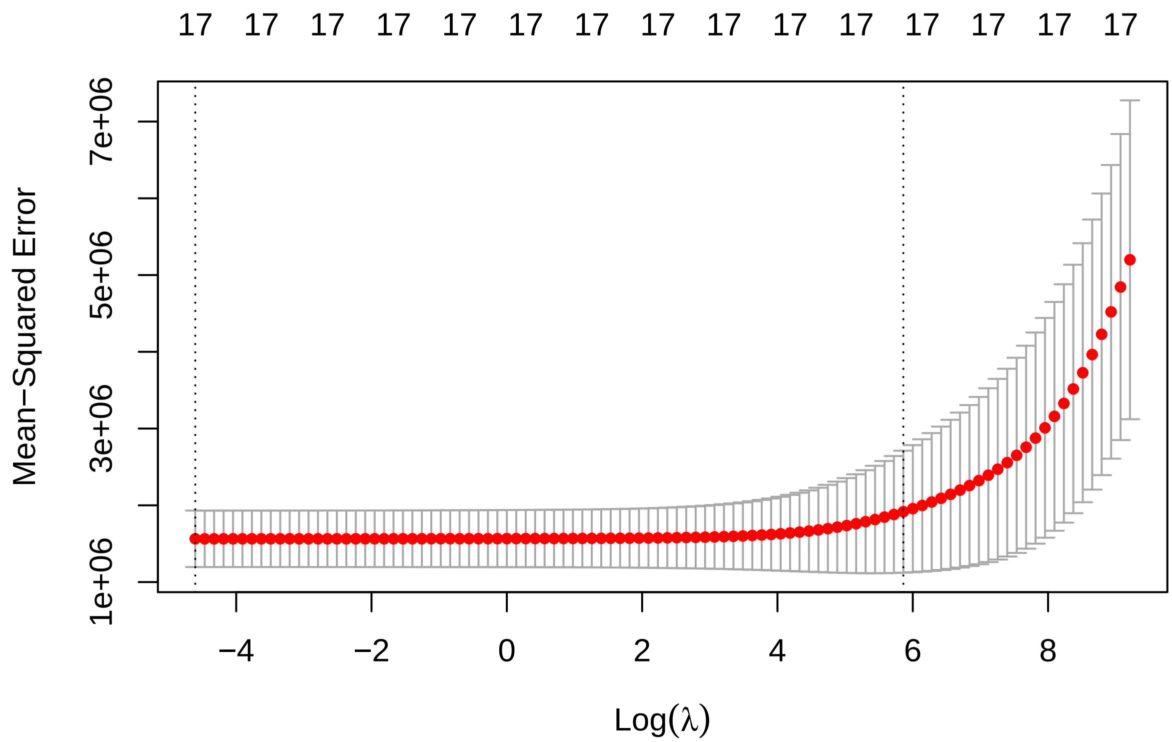
```
##
## Call:
## lm(formula = Apps ~ ., data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5010.4  -411.1   -44.6   331.3  7516.5
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -252.14312  481.48173  -0.524 0.600706
## PrivateYes  -513.83863  166.29747  -3.090 0.002101 **
```

```
## Accept         1.61207    0.04786  33.680  < 2e-16 ***
## Enroll        -1.02685    0.23535  -4.363 1.53e-05 ***
## Top10perc     54.45456    6.80808   7.999 7.18e-15 ***
## Top25perc    -17.99742    5.62943  -3.197 0.001466 **
## F.Undergrad    0.07407    0.04097   1.808 0.071139 .
## P.Undergrad    0.05437    0.03658   1.486 0.137753
## Outstate      -0.08471    0.02312  -3.664 0.000272 ***
## Room.Board     0.12870    0.06082   2.116 0.034779 *
## Books         -0.17938    0.27905  -0.643 0.520606
## Personal       0.01944    0.07372   0.264 0.792116
## PhD           -9.03931    5.78759  -1.562 0.118886
## Terminal      -4.54601    6.38395  -0.712 0.476697
## S.F.Ratio     23.58273   15.53971   1.518 0.129681
## perc.alumni    1.54769    5.10719   0.303 0.761970
## Expend         0.08171    0.01500   5.446 7.69e-08 ***
## Grad.Rate      9.10731    3.63123   2.508 0.012420 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1085 on 564 degrees of freedom
## Multiple R-squared:  0.9316, Adjusted R-squared:  0.9296
## F-statistic: 452.1 on 17 and 564 DF,  p-value: < 2.2e-16
```

```r
pred_lm <- predict(fit_lm, data_test)
# evaluate test error
err_lm <- mean((data_test$Apps - pred_lm)^2)
print(err_lm)
```

```
## [1] 838294.4
```

```r
### c) ridge regression convert to model matrices
x_train <- model.matrix(Apps ~ ., data = data_train)[, -1]
x_test <- model.matrix(Apps ~ ., data = data_test)[, -1]
# labels
y_train <- data_train$Apps
y_test <- data_test$Apps
# ridge regression
grid <- 10^seq(4, -2, length = 100)
mod_ridge <- glmnet(x_train, y_train, alpha = 0, lambda = grid)
fit_ridge <- cv.glmnet(x_train, y_train, alpha = 0, lambda = grid)
plot(fit_ridge)
```




```r
# optimal lambda
lambda <- fit_ridge$lambda.min

pred_ridge <- predict(mod_ridge, s = lambda, newx = x_test)
# test error
err_ridge <- mean((y_test - pred_ridge)^2)
print(err_ridge)
```

```
## [1] 838225.3
```

```r
### d) lasso
mod_lasso <- glmnet(x_train, y_train, alpha = 1, lambda = grid)
fit_lasso <- cv.glmnet(x_train, y_train, alpha = 1, lambda = grid)
lambda <- fit_lasso$lambda.min
pred_lasso <- predict(mod_lasso, s = lambda, newx = x_test)
err_lasso <- mean((y_test - pred_lasso)^2)
print(err_lasso)
```

```
## [1] 810496.4
```

```r
coef_lasso <- predict(fit_lasso, type = "coefficients", s = lambda)[1:ncol(College),
    ]
coef_lasso[coef_lasso != 0]
```

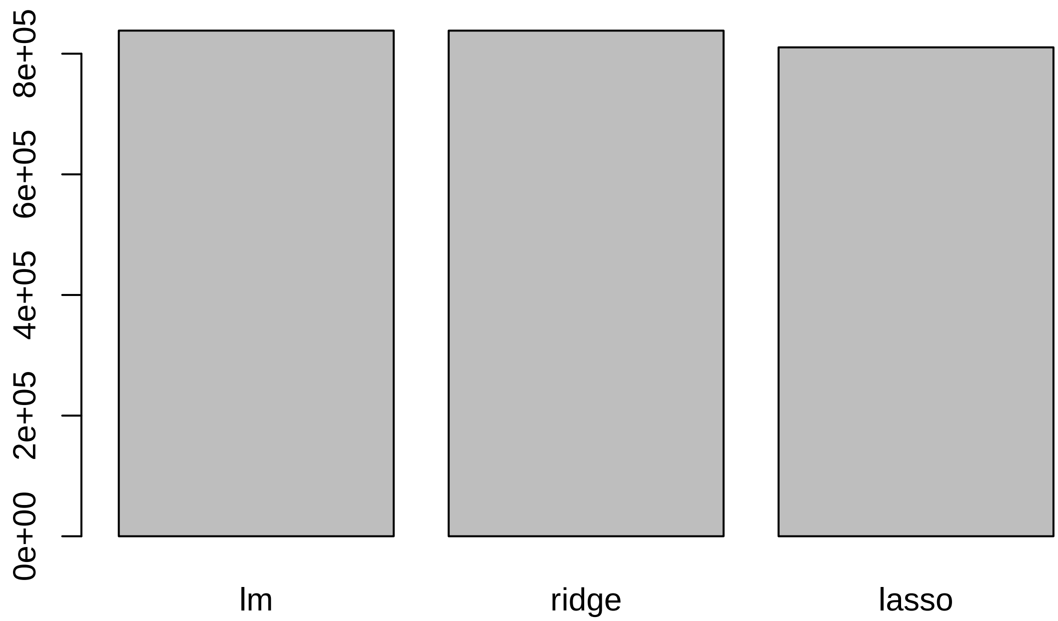
```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc 
## -499.83372113 -467.72805134    1.48930341   -0.27983943   39.03560000 
##     Top25perc   P.Undergrad      Outstate    Room.Board           PhD 
##   -6.55166194    0.03207624   -0.05929834    0.10890418   -6.32928038
```

```
##      Terminal     S.F.Ratio        Expend     Grad.Rate
##   -4.64456097   14.44260772    0.07439999    5.49267772
```

```r
length(coef_lasso[coef_lasso != 0])
```

```
## [1] 14
```

```r
### e)
err_all <- c(err_lm, err_ridge, err_lasso)
names(err_all) <- c("lm", "ridge", "lasso")
barplot(err_all)
```



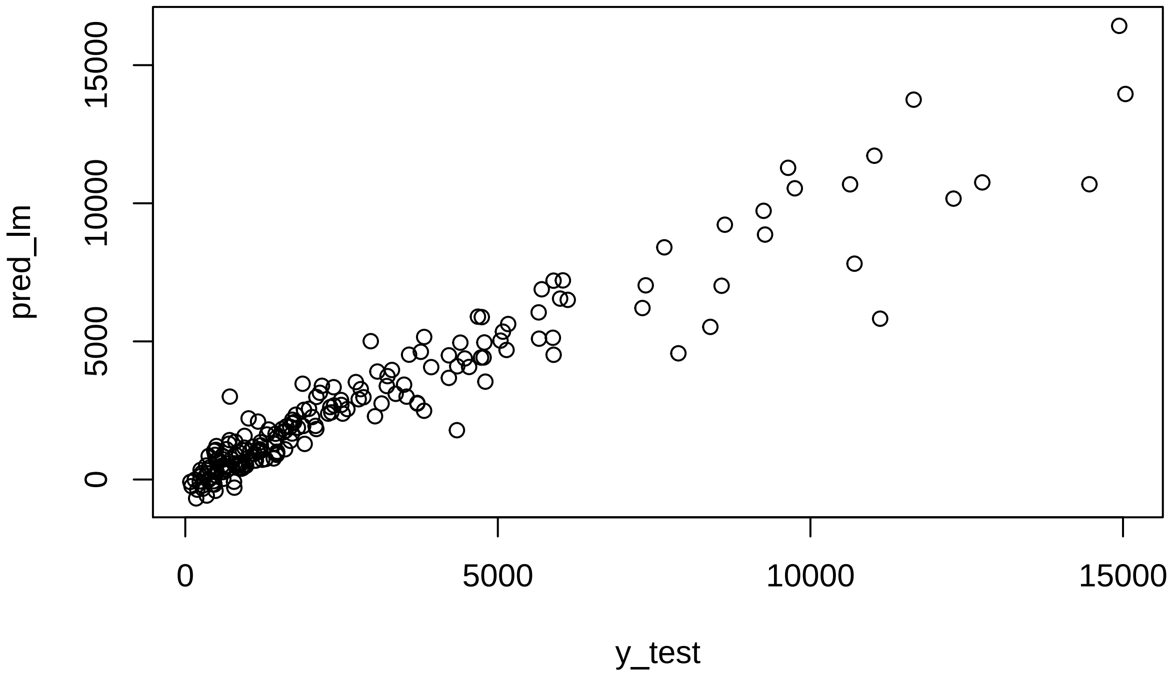

```r
# we can also compare the results in terms of R2
test_avg <- mean(y_test)
tss <- mean((y_test - test_avg)^2)

r2_lm <- 1 - err_lm/tss
r2_ridge <- 1 - err_ridge/tss
r2_lasso <- 1 - err_lasso/tss
```

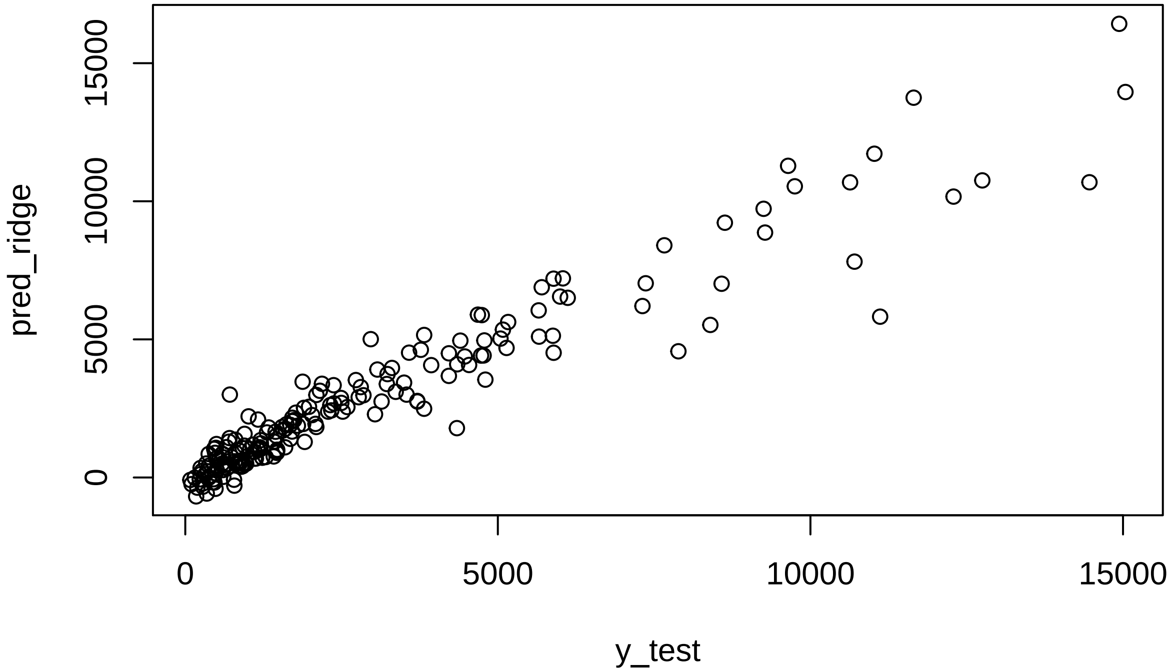
According to this specific seed and train/test partitioning, the test error for lasso is slightly less than full linear regression. Ridge regression performs similarly to linear regression. Lasso reduces `F.Undergrad`, `Books`, `Personal`, and `perc.alumni` to zero and shrinks the coefficients of the other variables. Note that the variables whose coefficients are shrinked to zero were not significant in the lm model.

All three R^2 values are above 0.9, meaning that the three models predict college applications with high accuracy.

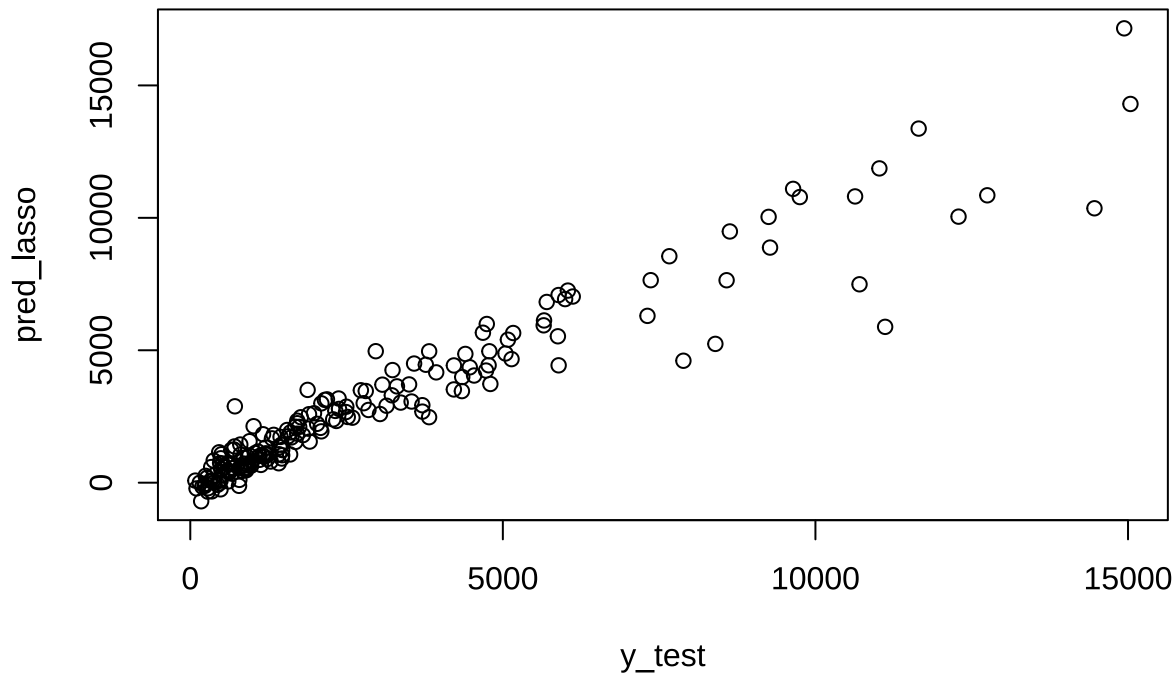
```r
plot(y_test, pred_lm)
```




```r
plot(y_test, pred_ridge)
```




```
plot(y_test, pred_lasso)
```



Exercise 2

The `Boston` data (`MASS` library) contains housing values in the suburbs of Boston for a sample of 506 observations. The aim is to predict per capita crime rate (`crim`) from the other variables.

a. Try out some of the regression methods you learned so far, such as subset selection (forward and backward), lasso, ridge regression. Present and discuss results for the approaches that you consider.
b. Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross-validation, or some other reasonable alternative, as opposed to using training error.
c. Inspect your selected model. Does it involve all of the features in the data set?

```
data(Boston)
summary(Boston)
```

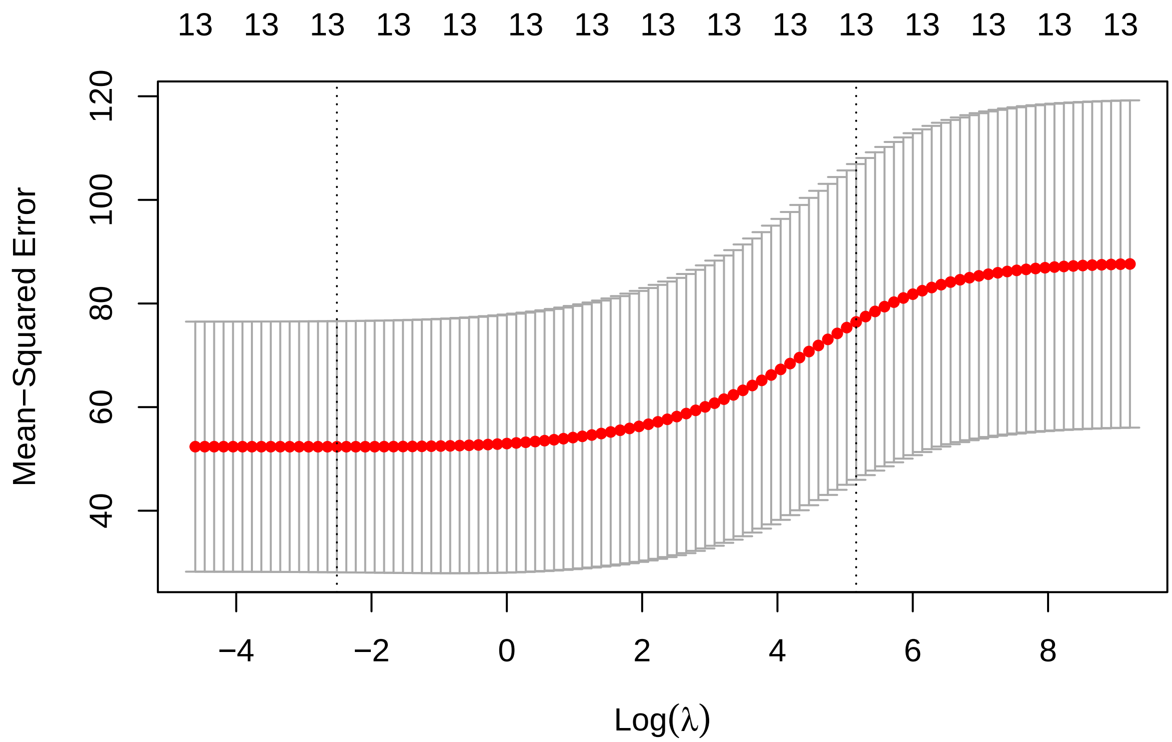
```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
```

```
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          black
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

```r
n_obs <- nrow(Boston)

set.seed(2021)
train_perc <- 0.75
train <- sample(1:n_obs, train_perc * n_obs)
data_train <- Boston[train, ]
data_test <- Boston[-train, ]

### ridge regression model matrices
x_train <- model.matrix(crim ~ ., data = data_train)[, -1]
x_test <- model.matrix(crim ~ ., data = data_test)[, -1]
# labels
y_train <- data_train$crim
y_test <- data_test$crim
# ridge regression
grid <- 10^seq(4, -2, length = 100)
mod_ridge <- glmnet(x_train, y_train, alpha = 0, lambda = grid)
fit_ridge <- cv.glmnet(x_train, y_train, alpha = 0, lambda = grid)
# optimal lambda
lambda <- fit_ridge$lambda.min
plot(fit_ridge)
```




```
pred_ridge <- predict(mod_ridge, s = lambda, newx = x_test)
# test error. Note: I select lambda by CV on training data and then evaluate
# accuracy of optimal model on (unseen) test data. This can be developed also
# into a nested CV.
err_ridge <- mean((y_test - pred_ridge)^2)
err_ridge
```

```
## [1] 15.73557
```

```
# coefficients
coefs_ridge <- predict(fit_ridge, s = lambda, type = "coefficients")[1:ncol(Boston),
    ]

### lasso
mod_lasso <- glmnet(x_train, y_train, alpha = 1, lambda = grid)
fit_lasso <- cv.glmnet(x_train, y_train, alpha = 1, lambda = grid)
lambda <- fit_lasso$lambda.min
pred_lasso <- predict(mod_lasso, s = lambda, newx = x_test)
err_lasso <- mean((y_test - pred_lasso)^2)
err_lasso
```

```
## [1] 15.23144
```

```
coef_lasso <- predict(fit_lasso, s = lambda, type = "coefficients")[1:ncol(Boston),
    ]
coef_lasso[coef_lasso != 0]
```

```
##   (Intercept)            zn          indus           chas            nox
```

```
##  19.770740861   0.046758239  -0.087145286  -0.171006112 -12.388565818
##            rm           dis           rad       ptratio         black
##   0.395258521  -1.120148742   0.546643913  -0.334680162  -0.006795943
##         lstat          medv
##   0.126293265  -0.229378830
```

```r
length(coef_lasso[coef_lasso != 0])
```

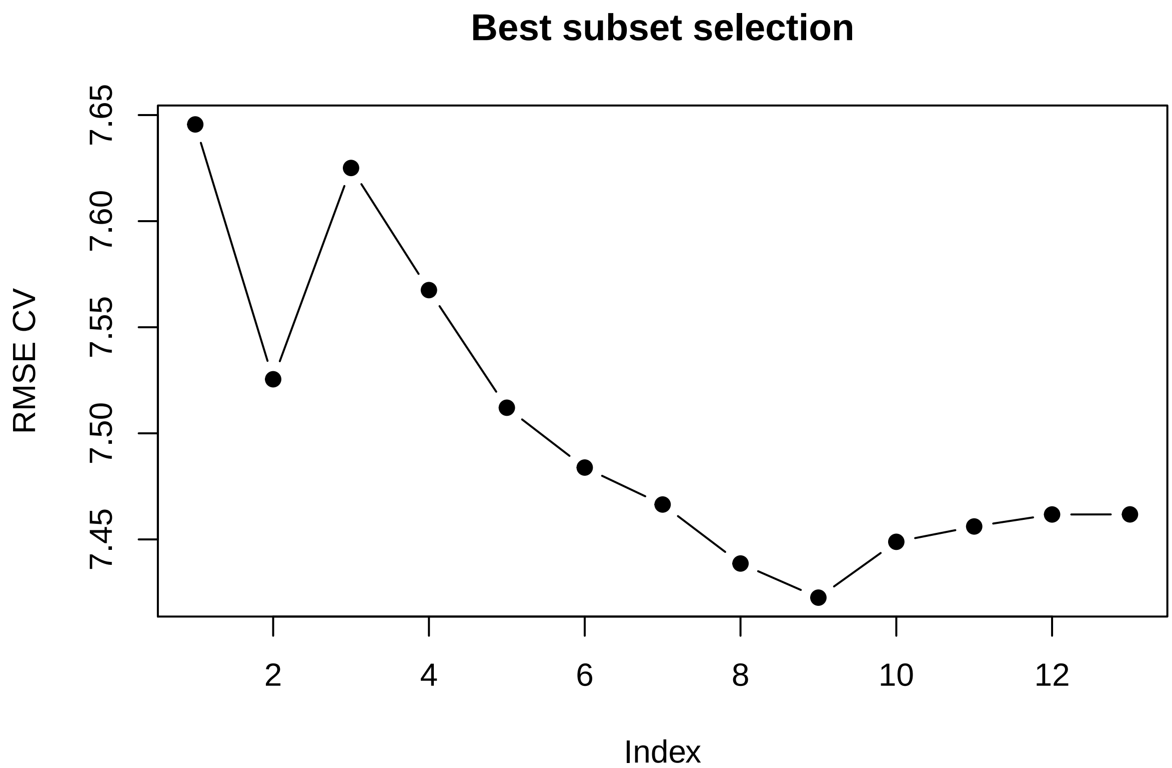
```
## [1] 12
```

```r
### subset selection custom predict method for regsubsets (see Lab3)
predict.regsubsets <- function(object, newdata, id, ...) {
    form <- as.formula(object$call[[2]])
    # this extracts the formula used in the call to regsubsets()
    mat <- model.matrix(form, newdata)
    coefi <- coef(object, id = id)
    xvars <- names(coefi)
    mat[, xvars] %*% coefi
}

# (max) number of variables
nv <- ncol(Boston) - 1

# best subset selection by k-fold cross-validation

# we perform model selection on the training data, as with ridge/lasso. So we
# leave the test data for the final comparison between the different methods.
# This can be developed into a nested CV.
set.seed(1)
k <- 5  # try also 10
n_obs_train <- nrow(data_train)
folds <- sample(1:k, n_obs_train, replace = TRUE)
cv.errors <- matrix(NA, k, nv)
for (j in 1:k) {
    best.fit <- regsubsets(crim ~ ., data = data_train[folds != j, ], nvmax = nv)
    for (i in 1:nv) {
        pred <- predict(best.fit, data_train[folds == j, ], id = i)
        cv.errors[j, i] <- mean((data_train$crim[folds == j] - pred)^2)
    }
}
mse.cv <- apply(cv.errors, MARGIN = 2, FUN = mean)
rmse.cv <- sqrt(mse.cv)
plot(rmse.cv, type = "b", pch = 19, main = "Best subset selection", ylab = "RMSE CV")
```



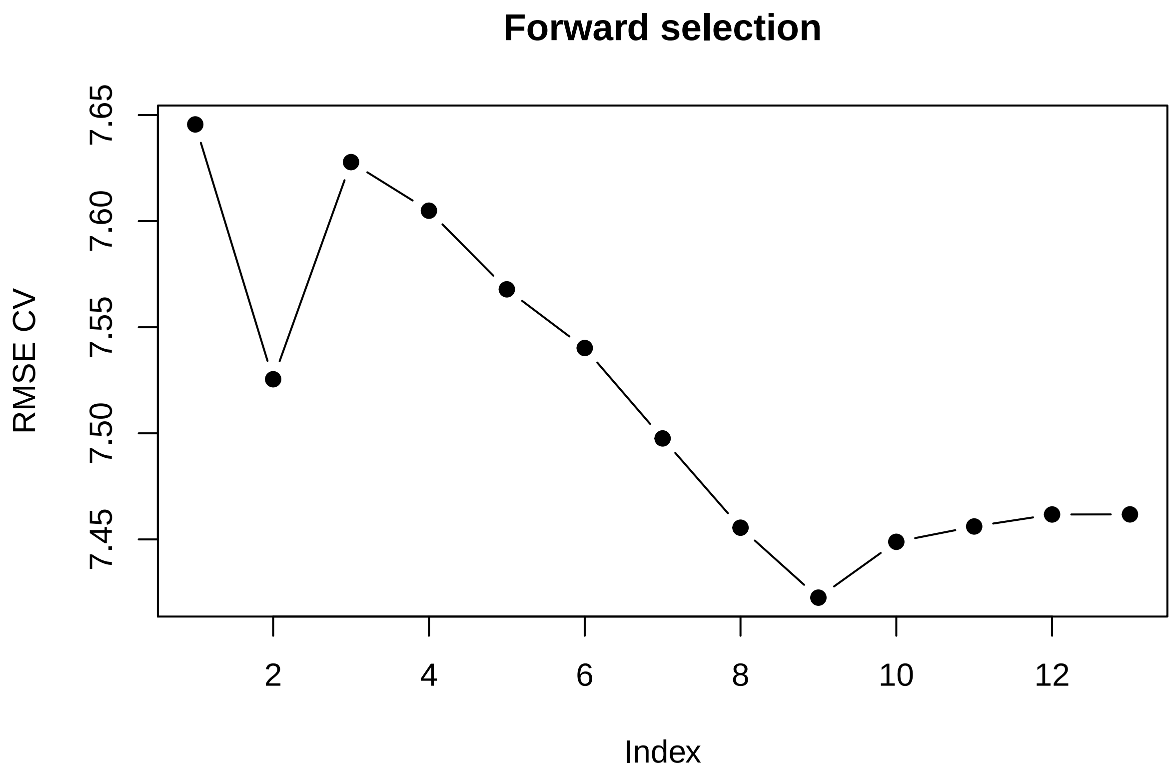

```
reg.best <- regsubsets(crim ~ ., data = data_train, nvmax = which.min(rmse.cv))
coef(reg.best, which.min(rmse.cv))
```

```
##   (Intercept)            zn         indus           nox           dis
##  27.297067145   0.053092801  -0.115957265 -15.585948410  -1.336059273
##           rad       ptratio         black         lstat          medv
##   0.571870103  -0.429956435  -0.007187638   0.112149525  -0.246084748
```

```
pred <- predict(reg.best, data_test, id = which.min(rmse.cv))
err_best <- mean((y_test - pred)^2)
err_best
```

```
## [1] 16.26828
```

```
# forward selection by k-fold cross-validation
cv.errors <- matrix(NA, k, nv)
for (j in 1:k) {
    fit_fwd <- regsubsets(crim ~ ., data = data_train[folds != j, ], nvmax = nv,
        method = "forward")
    for (i in 1:nv) {
        pred <- predict(fit_fwd, data_train[folds == j, ], id = i)
        cv.errors[j, i] <- mean((data_train$crim[folds == j] - pred)^2)
    }
}
mse.cv <- apply(cv.errors, MARGIN = 2, FUN = mean)
rmse.cv <- sqrt(mse.cv)
plot(rmse.cv, type = "b", pch = 19, main = "Forward selection", ylab = "RMSE CV")
```



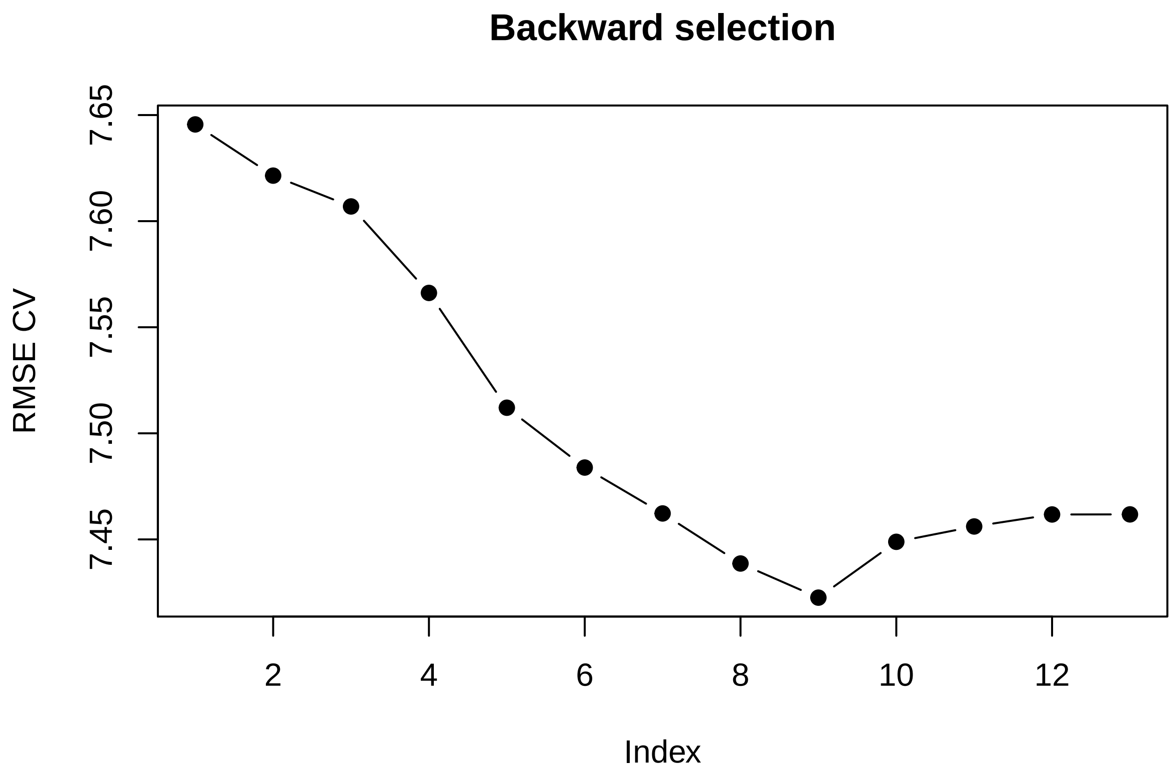

```
reg.fwd <- regsubsets(crim ~ ., data = data_train, nvmax = which.min(rmse.cv), method = "forward")
coef(reg.fwd, which.min(rmse.cv))
```

```
##   (Intercept)            zn         indus           nox           dis
##  27.297067145   0.053092801  -0.115957265 -15.585948410  -1.336059273
##           rad       ptratio         black         lstat          medv
##   0.571870103  -0.429956435  -0.007187638   0.112149525  -0.246084748
```

```
pred <- predict(reg.fwd, data_test, id = which.min(rmse.cv))
err_fwd <- mean((y_test - pred)^2)
err_fwd
```

```
## [1] 16.26828
```

```
# backward selection by k-fold cross-validation
cv.errors <- matrix(NA, k, nv)
for (j in 1:k) {
    fit_bwd <- regsubsets(crim ~ ., data = data_train[folds != j, ], nvmax = nv,
        method = "backward")
    for (i in 1:nv) {
        pred <- predict(fit_bwd, data_train[folds == j, ], id = i)
        cv.errors[j, i] <- mean((data_train$crim[folds == j] - pred)^2)
    }
}
mse.cv <- apply(cv.errors, MARGIN = 2, FUN = mean)
rmse.cv <- sqrt(mse.cv)
plot(rmse.cv, type = "b", pch = 19, main = "Backward selection", ylab = "RMSE CV")
```




```r
reg.bkw <- regsubsets(crim ~ ., data = data_train, nvmax = which.min(rmse.cv), method = "backward")
coef(reg.bkw, which.min(rmse.cv))
```

```
##   (Intercept)            zn         indus           nox           dis
##  27.297067145   0.053092801  -0.115957265 -15.585948410  -1.336059273
##           rad       ptratio         black         lstat          medv
##   0.571870103  -0.429956435  -0.007187638   0.112149525  -0.246084748
```

```r
pred <- predict(reg.bkw, data_test, id = which.min(rmse.cv))
err_bwd <- mean((y_test - pred)^2)
err_bwd
```

```
## [1] 16.26828
```

```r
# Summary of results
err_ridge
```

```
## [1] 15.73557
```

```r
err_lasso
```

```
## [1] 15.23144
```

```r
err_best
```

```
## [1] 16.26828
```

```r
err_bwd
```

```
## [1] 16.26828
```

```
err_fwd
```

```
## [1] 16.26828
```

On this dataset and for the chosen seed, lasso performed the best and led to the selection of 12 variables. The errors of the different methods are very similar, so repeated train-test splits (or nested CV) can be used for further evaluation.

Exercise 3

For this exercise, we explore how lasso performs on simulated data.

a. For this first exercise, we will use again the simulated data of Exercise 3, Tutorato #4, ie:

- Generate a predictor x of length $n = 100$ and a noise vector ϵ of the same size, using the `rnorm()` function.
- Generate a response vector y of length $n = 100$ from the model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$$

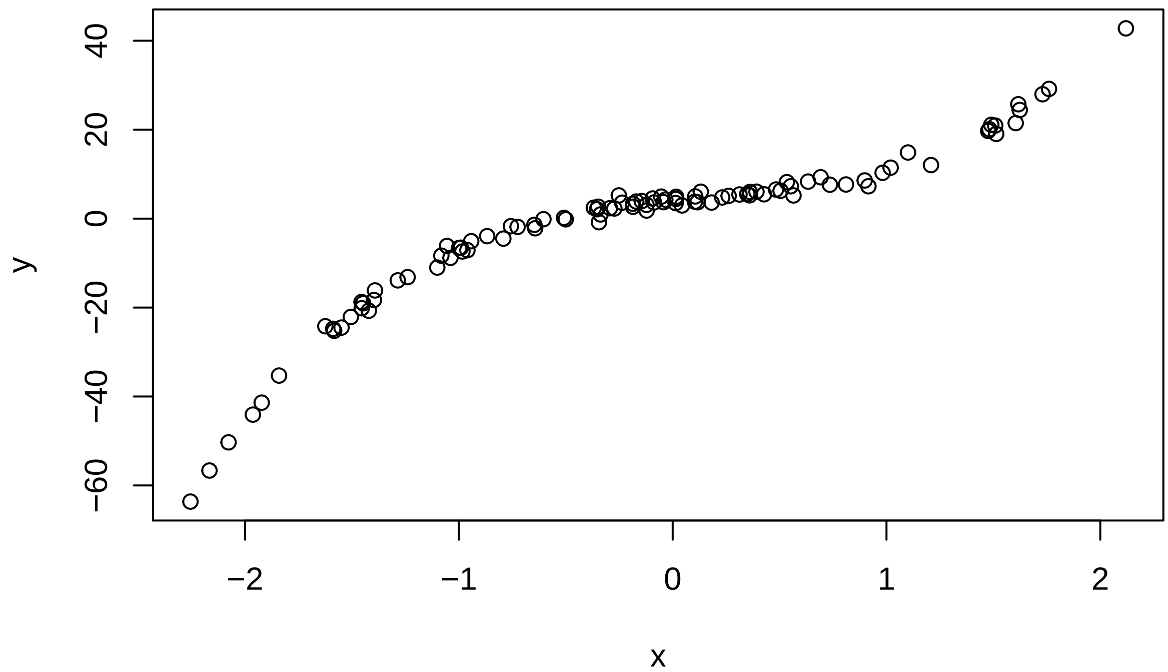
  where you can freely choose the values for the constants β_i.
  We now pretend that we do not know the true model and fit a polynomial model with degree 10 on the simulated data. Fit a lasso model using the 10 predictors. Select the optimal λ by cross-validation (CV), specifying a grid of possible λ values (see the beginning of 6.6.1); plot the CV error as a function of λ. Report and discuss the coefficient estimates. Is the selected model close to the true model, i.e. the model that you used for simulating the data?

b. As a second simulation, generate a new response vector y according to the model

$$y = \beta_0 + \beta_7 x^7 + \epsilon$$

   where you choose a value for β_7 (and reuse the previous value for β_0). Perform best subset selection and the lasso, again with a polynomial model of degree 10. For lasso, select the optimal λ by cross-validation, letting the function choose its own grid of values. Discuss the results obtained.

```
set.seed(2021)
x <- rnorm(100)
epsilon <- rnorm(100)
# using custom betas
beta0 <- 4
beta1 <- 5
beta2 <- -2
beta3 <- 4
y <- beta0 + beta1 * x + beta2 * x^2 + beta3 * x^3 + epsilon
# visually inspect the (true) relationship between y and x
plot(x, y)
```

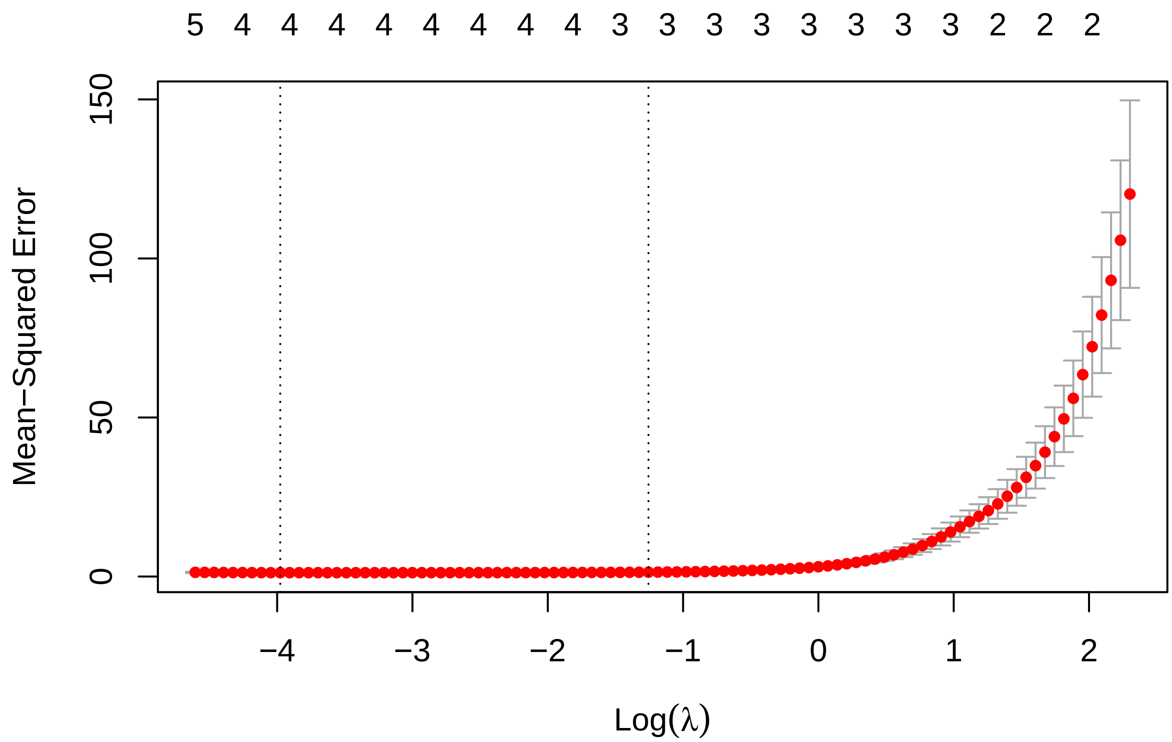



```
df <- data.frame(y, x)

# lasso
x_mat <- model.matrix(y ~ poly(x, 10, raw = TRUE), data = df)[, -1]
grid <- 10^seq(1, -2, length = 100)
mod_lasso <- glmnet(x_mat, y, alpha = 1, lambda = grid)
fit_lasso <- cv.glmnet(x_mat, y, alpha = 1, lambda = grid)
best_lambda <- fit_lasso$lambda.min
best_lambda
```

```
## [1] 0.01873817
```

```
plot(fit_lasso)
```




```
coef_lasso <- predict(mod_lasso, s = best_lambda, type = "coefficients")[1:ncol(x_mat),
    ]
coef_lasso[coef_lasso != 0]
```

```
##              (Intercept) poly(x, 10, raw = TRUE)1 poly(x, 10, raw = TRUE)2
##                 3.926717                 4.942705                -1.930583
## poly(x, 10, raw = TRUE)3
##                 3.978422
```

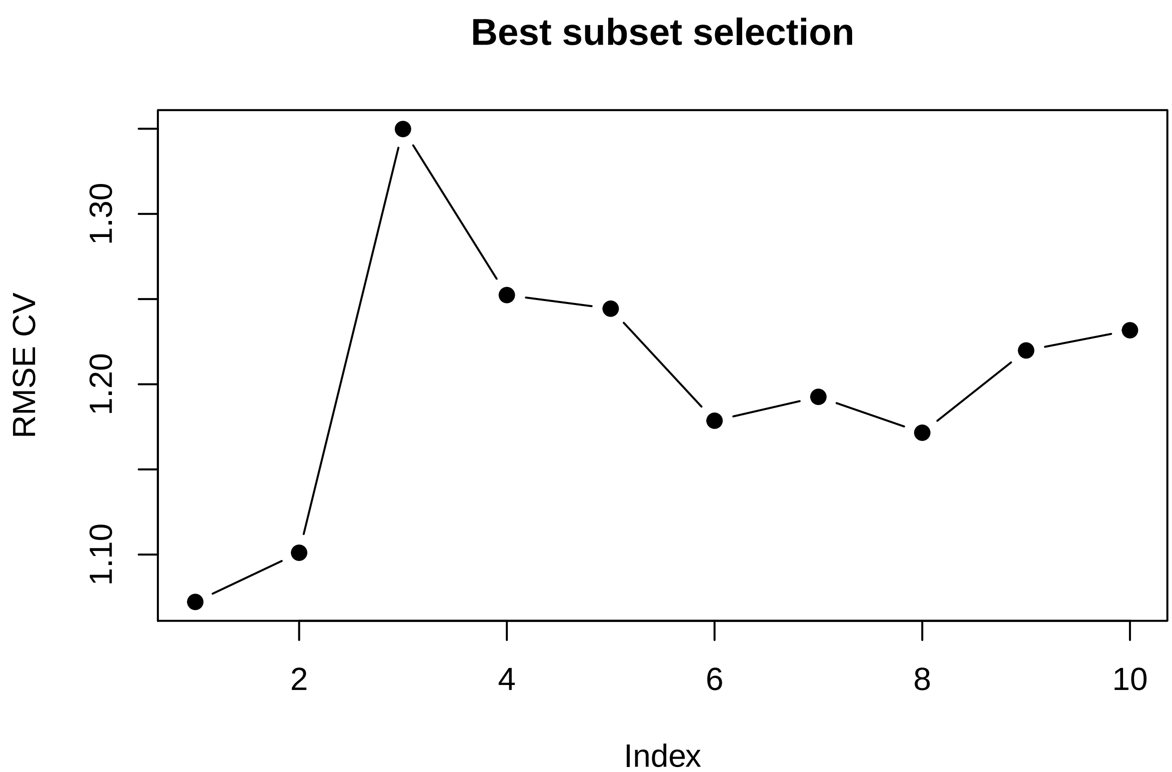
The lasso coefficients are zero for degrees 4 to 10.

```
beta7 <- 7
y = beta0 + beta7 * x^7 + epsilon
df <- data.frame(y, x)

# best subset selection here the k-fold CV approach is shown you are free to use
# the full dataset and evaluate the indicators Cp, BIC, adjr2

set.seed(1)
n_obs <- nrow(df)  # number of observations
nv <- 10  # number of variables
k <- 5
folds <- sample(1:k, n_obs, replace = TRUE)
cv.errors <- matrix(NA, k, nv)
for (j in 1:k) {
    best.fit <- regsubsets(y ~ poly(x, nv, raw = TRUE), data = df[folds != j, ],
        nvmax = nv)
```

```r
    for (i in 1:nv) {
        pred <- predict(best.fit, df[folds == j, ], id = i)
        cv.errors[j, i] <- mean((df$y[folds == j] - pred)^2)
    }
}
mse.cv <- apply(cv.errors, MARGIN = 2, FUN = mean)
rmse.cv <- sqrt(mse.cv)
plot(rmse.cv, type = "b", pch = 19, main = "Best subset selection", ylab = "RMSE CV")
```




```r
which.min(mse.cv)
```

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

A 1-variable model has the lowest MSE (RMSE) according to best subset selection evaluated in a 5-fold cross-validation. The following code shows how the correct predictor is selected and how the parameters are close to the true ones:

```r
reg.best <- regsubsets(y ~ poly(x, nv, raw = TRUE), data = df, nvmax = which.min(rmse.cv))
coef(reg.best, which.min(rmse.cv))
```

```
##              (Intercept) poly(x, nv, raw = TRUE)7
##                 3.988194                 6.998570
```

```r
# true parameters
c(beta0, beta7)
```

```
## [1] 4 7
```

Similarly for lasso:

```r
set.seed(1)
n_obs <- nrow(df)  # number of observations
nv <- 10  # number of variables
x <- model.matrix(y ~ poly(x, nv, raw = TRUE), data = df)[, -1]
y <- df$y

### lasso
fit_lasso <- cv.glmnet(x, y, alpha = 1)
lambda <- fit_lasso$lambda.min
coef_lasso <- predict(fit_lasso, s = lambda, type = "coefficients")[1:ncol(x), ]
coef_lasso[coef_lasso != 0]
```

```
##              (Intercept) poly(x, nv, raw = TRUE)7
##                 2.529031                 6.794558
```

```r
coef(fit_lasso)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)              2.386765
## poly(x, nv, raw = TRUE)1  .
## poly(x, nv, raw = TRUE)2  .
## poly(x, nv, raw = TRUE)3  .
## poly(x, nv, raw = TRUE)4  .
## poly(x, nv, raw = TRUE)5  .
## poly(x, nv, raw = TRUE)6  .
## poly(x, nv, raw = TRUE)7  6.774668
## poly(x, nv, raw = TRUE)8  .
## poly(x, nv, raw = TRUE)9  .
## poly(x, nv, raw = TRUE)10 .
```

Both procedures perform very well and lead to the true model. The estimates are less accurate (shrinked) for lasso. It is generally regarded as good practice to use lasso only for variable selection, i.e. identify the predictors with non-zero coefficients, and to then refit the model by least squares just using the selected predictors. This typically leads to more accurate estimates than the lasso ones.

```r
# refit model on the selected predictor
fit_lm <- lm(y ~ I(x^7), data = df)
coef(fit_lm)
```

```
## (Intercept)      I(x^7)
##    3.988194    6.998570
```