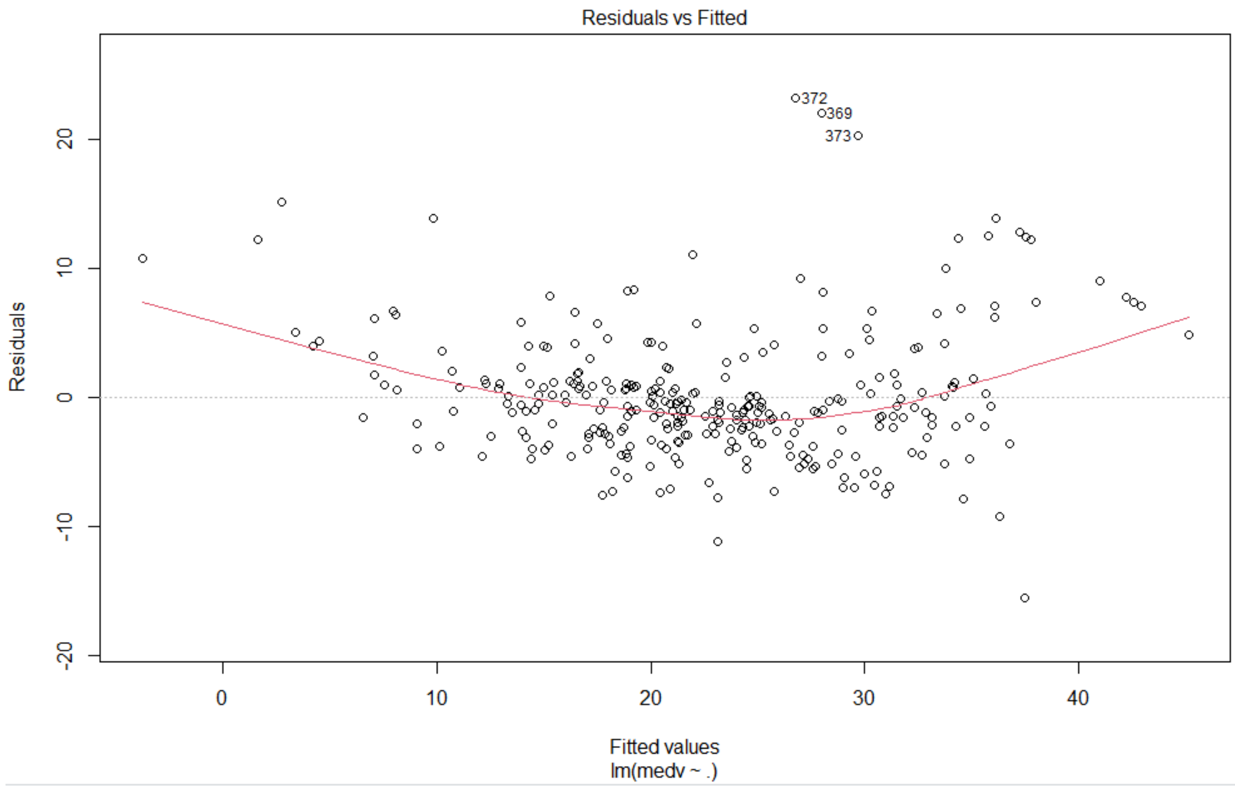
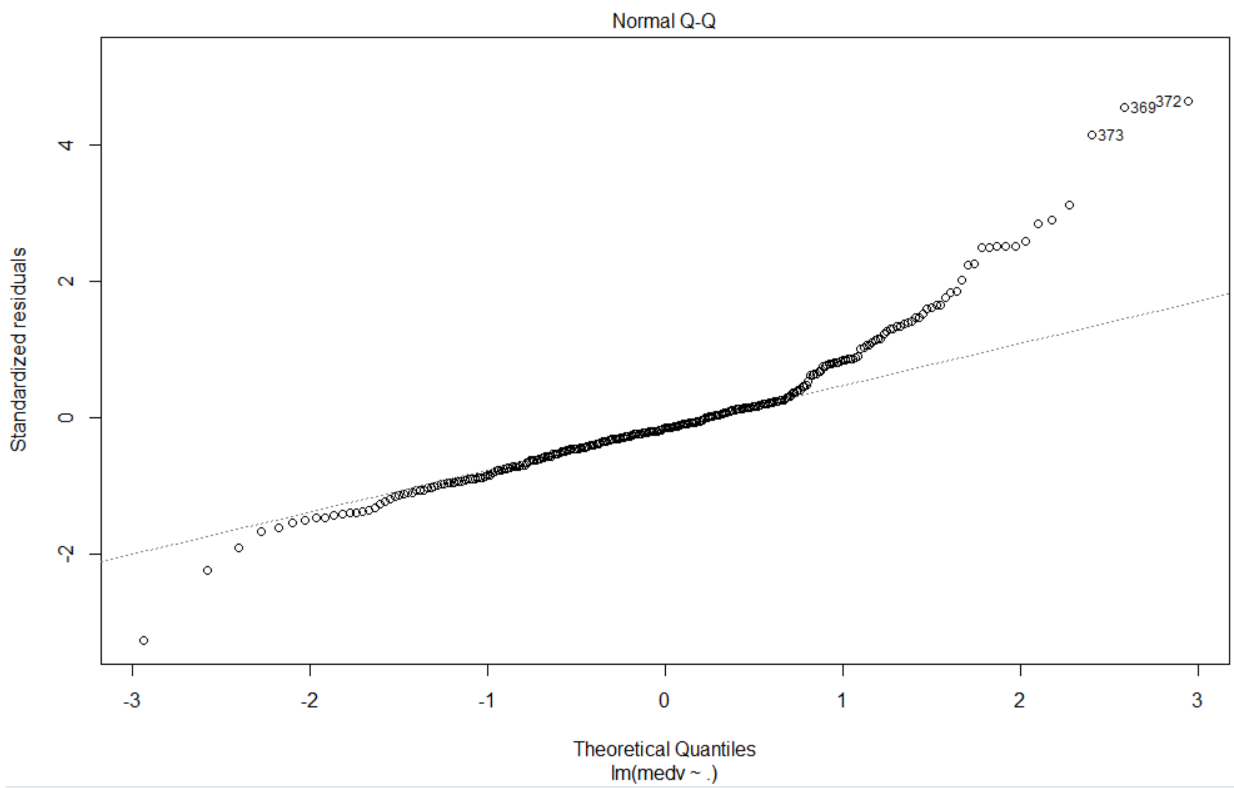
```
> ## clear console and environment ##
> rm(list = ls())
> cat("\014")
>
> ## packages to use ##
> library(ISLR)
> library(glmnet)
> library(MASS)
> str(Boston)
'data.frame':	506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
> names(Boston)
 [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"     "dis"     "rad"
[10] "tax"     "ptratio" "black"   "lstat"   "medv"
> library(leaps)
> library(ggplot2)
```

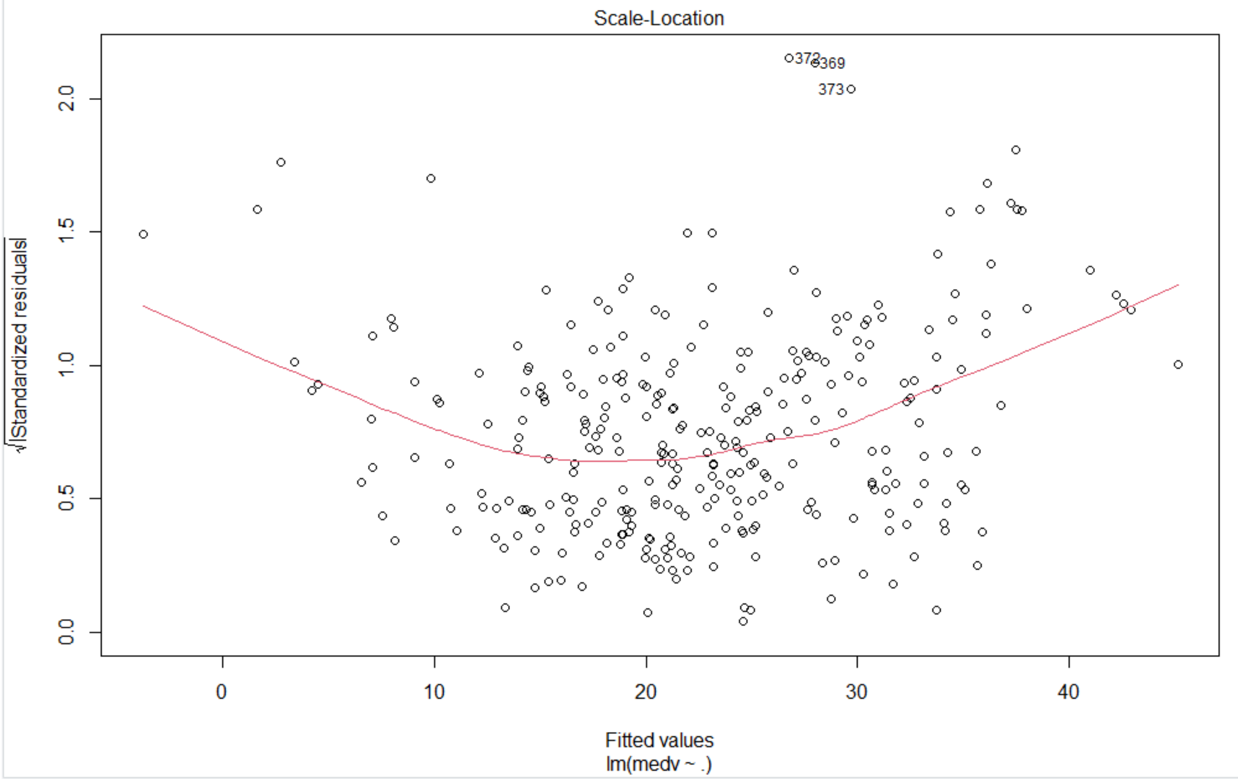
```
> library(glmnet)
> require(caret)
> library(tidyverse)
> library(ggthemes)
>
> ## Tasks ##
>
> ## 0) Please use set.seed(1) for all operations that involve user-induced randomness ##
> set.seed(1)
> ###### ###### ###### ###### ###### ######
```

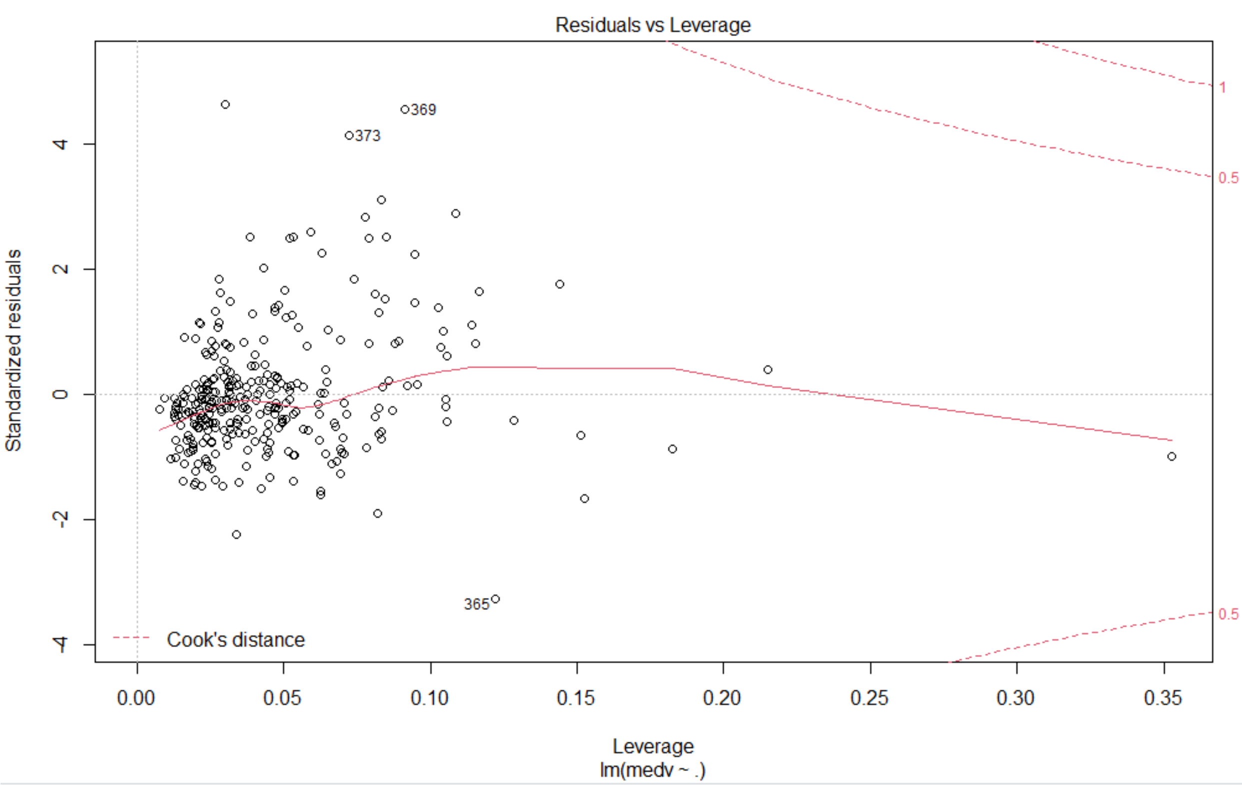
```
> ## 1) Please randomly split (using the sample command) the observations into a training set and a validation set, so that the training set can be used to fit a linear model, and the validation set can be used to evaluate the prediction accuracy of the fitted model ##

> train<-sample(nrow(Boston), 306)

> Boston.train<-Boston[train,]

> Boston.valid<-Boston[-train,]

> preObj <- preProcess(Boston.train, method = c('center', 'scale'))

> training <- predict(preObj, Boston.train)

> testing <- predict(preObj, Boston.valid)

> y.train <- training$medv

> y.test <- testing$medv

>

> encoding <- dummyVars(medv ~ ., data = training)

> x.train <- predict(encoding, training)

> x.test <- predict(encoding, testing)

>

> lin.model<-lm(medv ~ ., data = Boston.train)

> summary(lin.model)


Call:

lm(formula = medv ~ ., data = Boston.train)


Residuals:

    Min       1Q   Median       3Q      Max

-15.5731  -2.8349  -0.7741   1.3258  23.2142


Coefficients:

             Estimate Std. Error t value Pr(>|t|)

(Intercept)  49.119269   6.911156   7.107 9.11e-12 ***
```

```
crim        -0.102294   0.037420  -2.734 0.006646 **
zn           0.049984   0.018732   2.668 0.008047 **
indus        0.095080   0.078809   1.206 0.228617
chas         4.304232   1.198179   3.592 0.000384 ***
nox        -22.746062   5.687691  -3.999 8.06e-05 ***
rm           2.527001   0.562515   4.492 1.02e-05 ***
age          0.009605   0.018020   0.533 0.594415
dis         -1.618095   0.274378  -5.897 1.02e-08 ***
rad          0.351480   0.085358   4.118 4.99e-05 ***
tax         -0.012809   0.004684  -2.735 0.006623 **
ptratio     -1.018225   0.179711  -5.666 3.51e-08 ***
black        0.009063   0.003746   2.419 0.016154 *
lstat       -0.649383   0.064962  -9.996  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.085 on 292 degrees of freedom
Multiple R-squared:  0.7234,	Adjusted R-squared:  0.7111
F-statistic: 58.74 on 13 and 292 DF,  p-value: < 2.2e-16

> pred<-predict(lin.model, Boston.valid)
> MSE=mean((Boston.valid$medv-pred)^2)
> MSE
[1] 20.01941
> par(mfrow = c(1,1))
> plot(lin.model)
```


Residuals vs Fitted
372
369
373
20
10
0
-10
-20
Residuals
0
10
20
30
40
Fitted values
lm(medv ~ .)

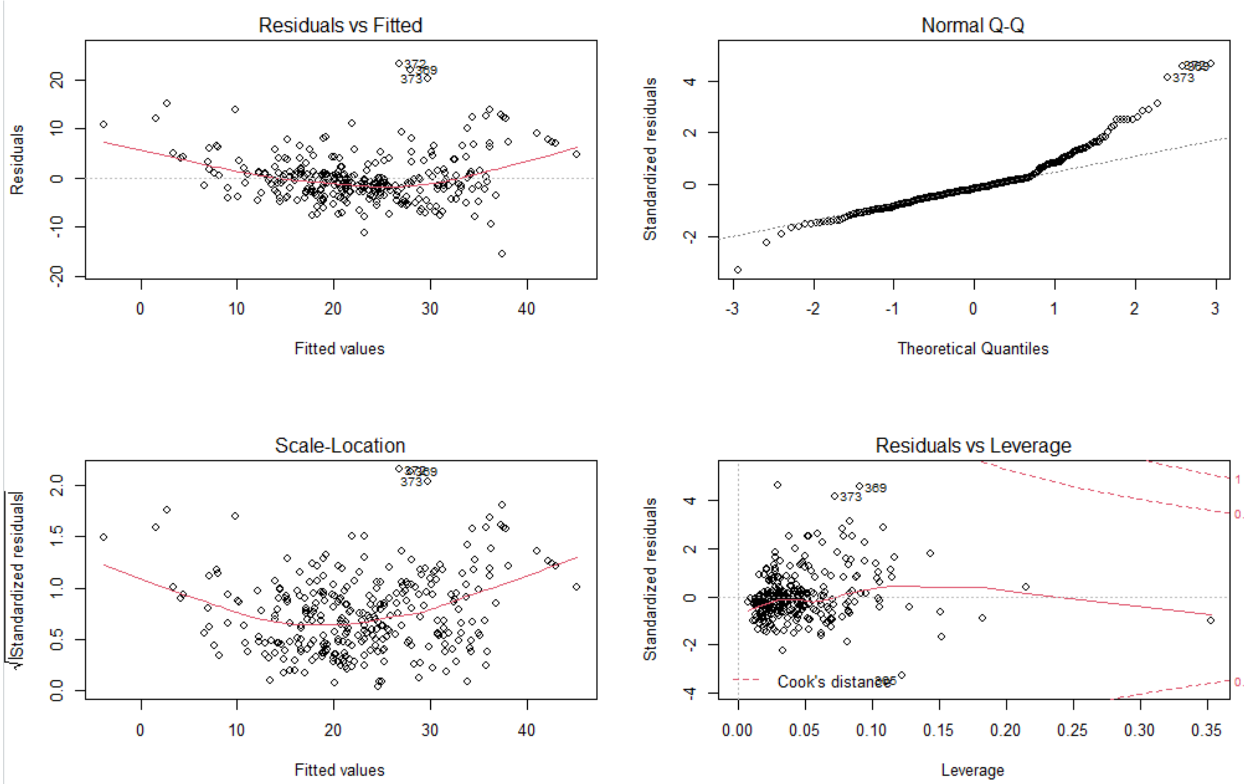

Normal Q-Q
369
372
373
4
2
0
-2
Standardized residuals
-3
-2
-1
0
1
2
3
Theoretical Quantiles
lm(medv ~ .)


Scale-Location
√|Standardized residuals|
Fitted values
lm(medv ~ .)
372
369
373


Residuals vs Leverage
Standardized residuals
Leverage
lm(medv ~ .)
369
373
365
Cook's distance
1
0.5
0.5

> par(mfrow = c(2,2))

> plot(lin.model)



> ###### ###### ###### ###### ###### ######

> ## 2) Apply best subset selection on all potential predictors without interactions between them, report the best model and its fitted model, perform model diagnostics on the model, conduct hypothesis tests on some coefficients of the model and report your findings, and assess the prediction accuracy of the fitted model and report your findings. ##

> reglist <- regsubsets(medv~., data=Boston, method = "forward")

> sum <- summary(reglist)

>

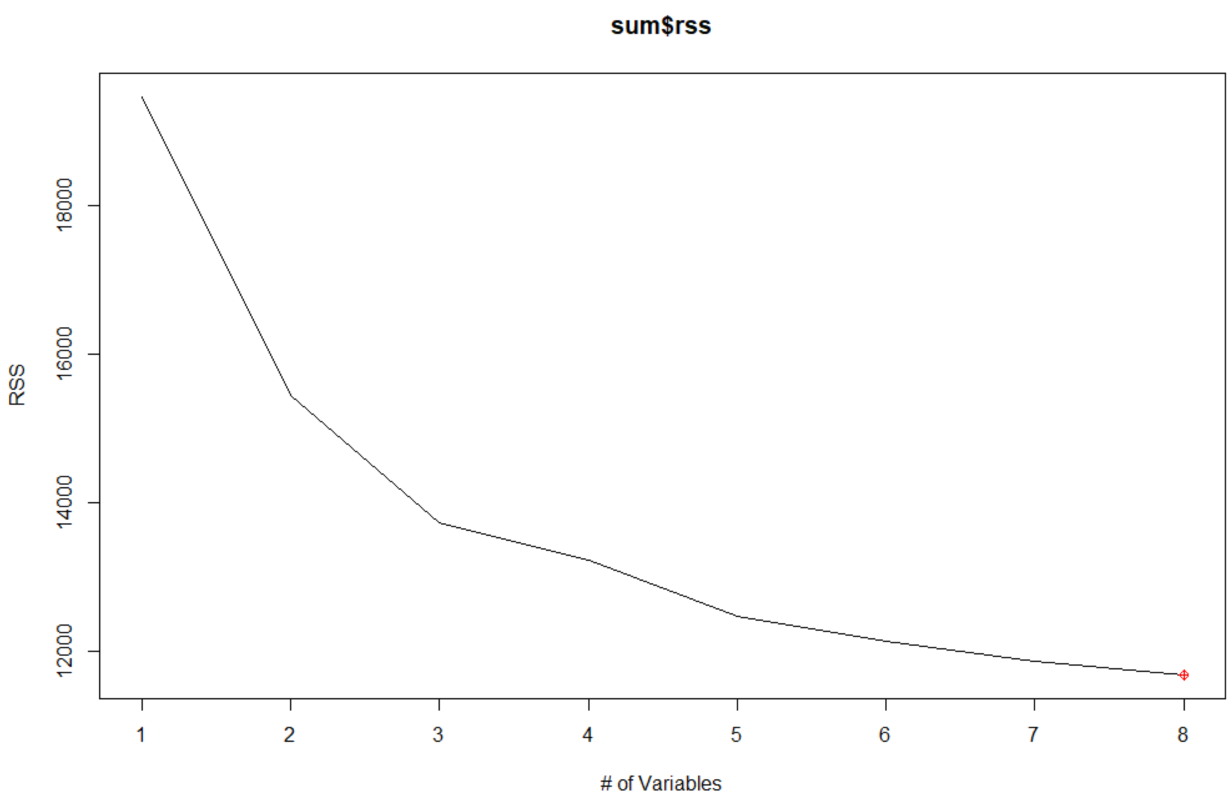
> ## Plots ##

> plot(sum$rss, main = "", xlab="# of Variables", ylab="RSS", type = 'l')

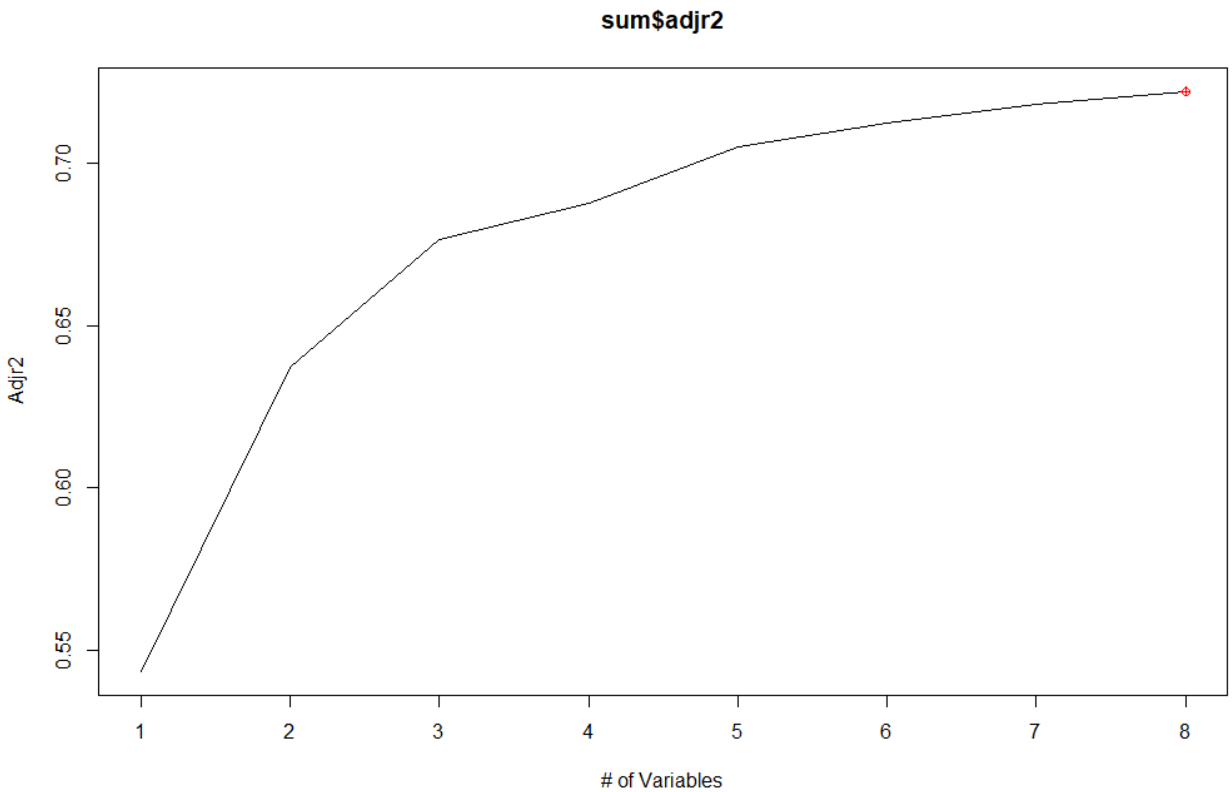
> min_rss <- which.min(sum$rss)

> points(min_rss, sum$rss[min_rss], pch=10, col="red")

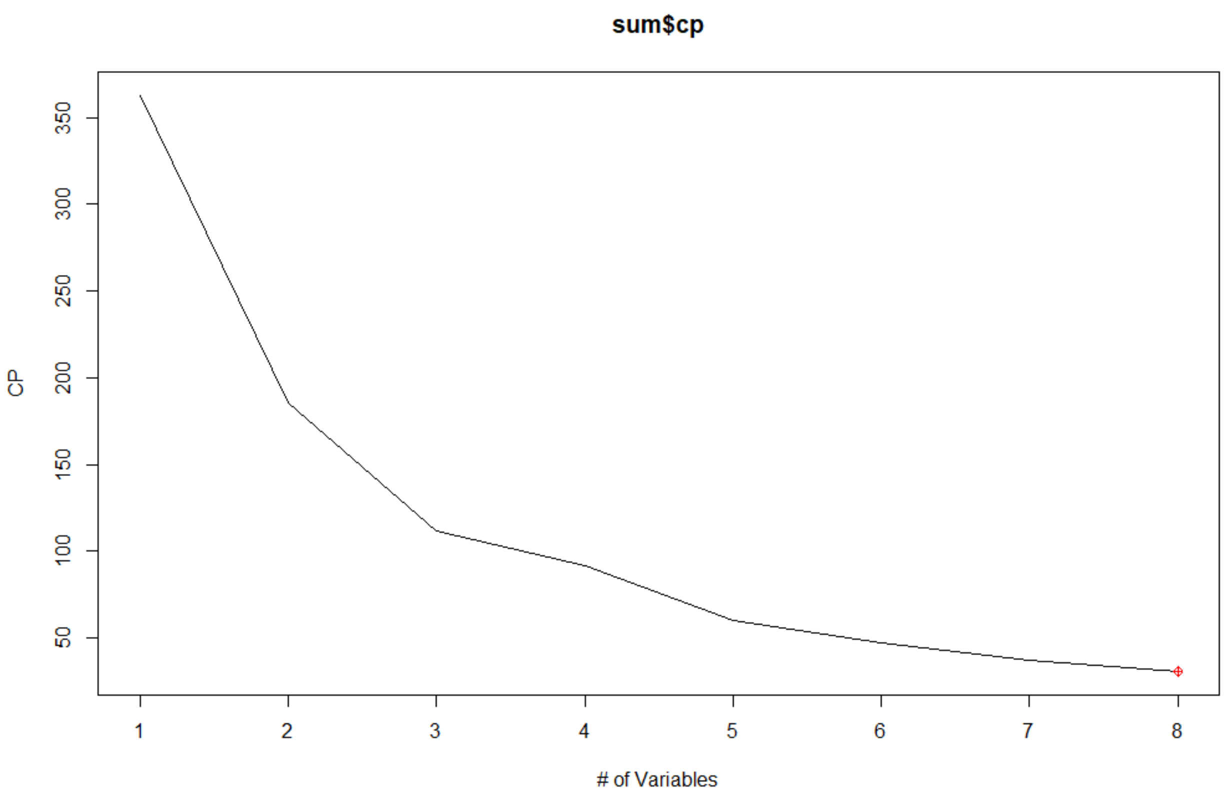
> title(main = "sum$rss")



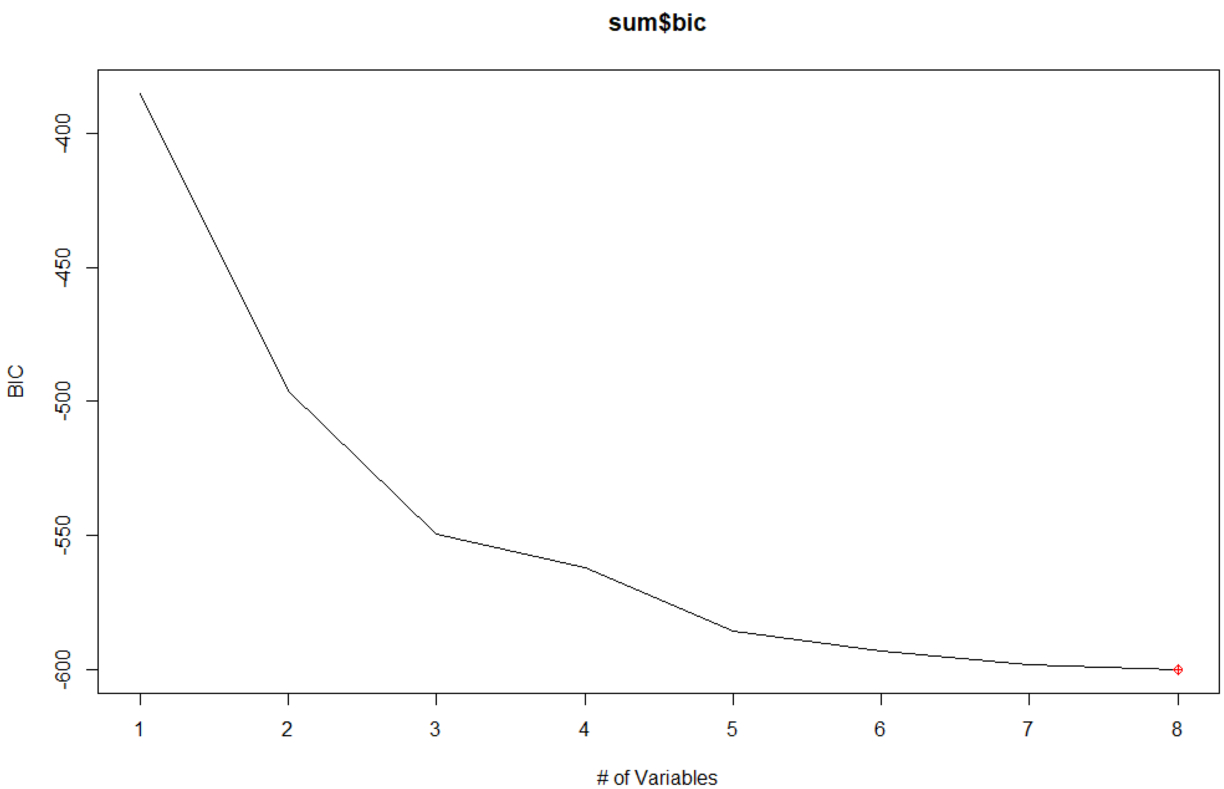
```
> plot(sum$adjr2, xlab="# of Variables", ylab="Adjr2", type = 'l')
> max_adjr2 <- which.max(sum$adjr2)
> points(max_adjr2, sum$adjr2[max_adjr2], pch=10, col="red")
> title(main = "sum$adjr2")
```

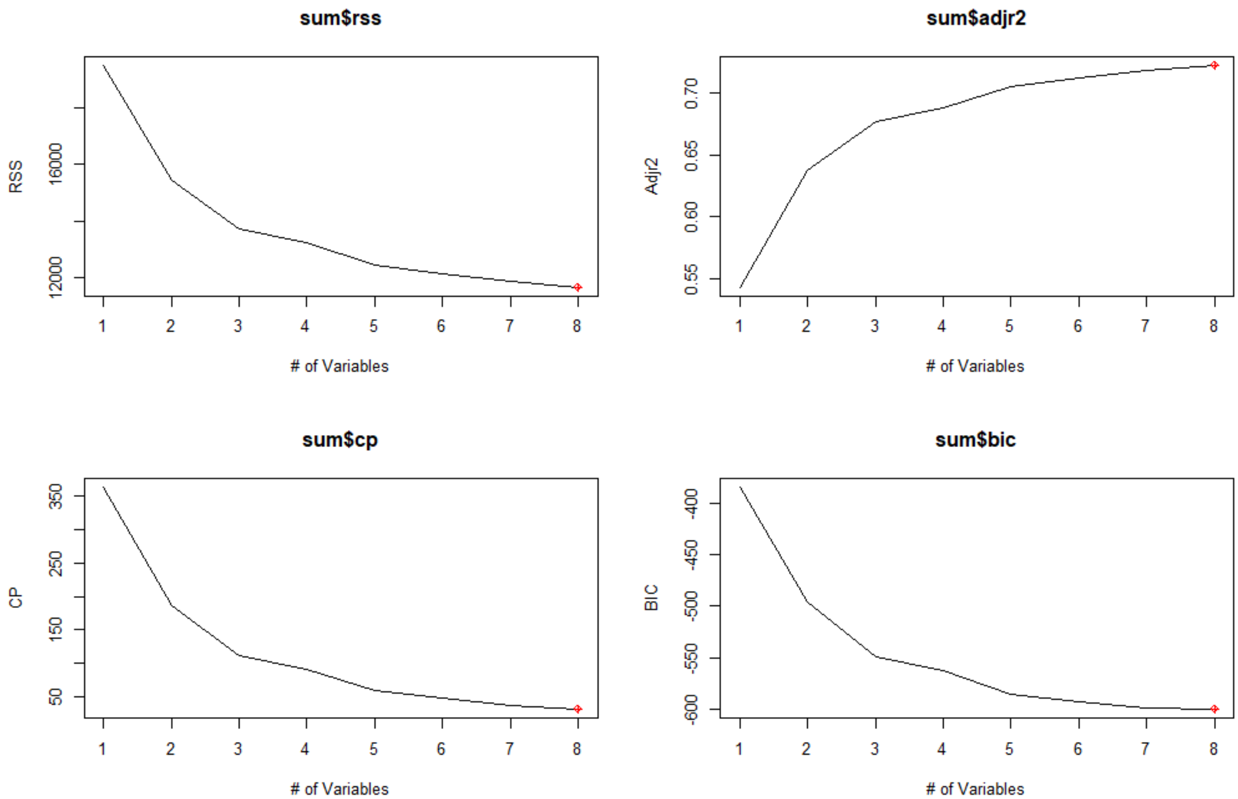


```
> plot(sum$cp, xlab="# of Variables", ylab="CP", type = 'l')
> min_cp <- which.min(sum$cp)
> points(min_cp, sum$cp[min_cp], pch=10, col="red")
> title(main = "sum$cp")
```



```
> plot(sum$bic, xlab="# of Variables", ylab="BIC", type = 'l')

> min_bic <- which.min(sum$bic)

> points(min_bic, sum$bic[min_bic], pch=10, col="red")

> title(main = "sum$bic")
```






```
> ## Get the best coefficients ##

> coef(reglist, 8)

 (Intercept)          zn        chas         nox          rm         dis     ptratio

 30.316950269   0.037808067   3.111061718 -16.687427958   4.116082349  -1.382714038  -
0.881851067

       black        lstat

  0.009403764  -0.543125369

> lm_model <- lm(medv ~ zn + chas + nox + rm + dis + ptratio + black + lstat, data=Boston)

>

> pred <- predict(lm_model, testing)

> (lin.info <- postResample(pred, testing$medv))

    RMSE Rsquared      MAE

36.37803  0.43877 32.61980

>

> summary(lm_model)
```

```
Call:
lm(formula = medv ~ zn + chas + nox + rm + dis + ptratio + black +
    lstat, data = Boston)

Residuals:
     Min       1Q   Median       3Q      Max
-15.6996  -2.7925  -0.5477   1.7005  27.6510

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  30.316950   4.870856   6.224 1.03e-09 ***
zn            0.037808   0.013298   2.843 0.004652 **
chas          3.111062   0.870076   3.576 0.000384 ***
nox         -16.687428   3.228873  -5.168 3.43e-07 ***
rm            4.116082   0.408594  10.074  < 2e-16 ***
dis          -1.382714   0.187604  -7.370 7.15e-13 ***
ptratio      -0.881851   0.115718  -7.621 1.29e-13 ***
black         0.009404   0.002639   3.563 0.000401 ***
lstat        -0.543125   0.047652 -11.398  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.847 on 497 degrees of freedom
Multiple R-squared:  0.7266,	Adjusted R-squared:  0.7222
F-statistic: 165.1 on 8 and 497 DF,  p-value: < 2.2e-16

> mse <- mean(lm_model$residuals ^ 2)
> mse
```

```
[1] 23.07964
>
>
> ###### ###### ###### ###### ###### ######
```

> ## 3) Implement LASSO (with cross-validation to select the optimal tuning parameter) on all potential predictors without interactions between them, report the best model (that is based on the optimal tuning parameter) and its fitted model, conduct hypothesis tests on some coefficients of the model and report your findings, and assess the prediction accuracy of the fitted model and report your findings. ##

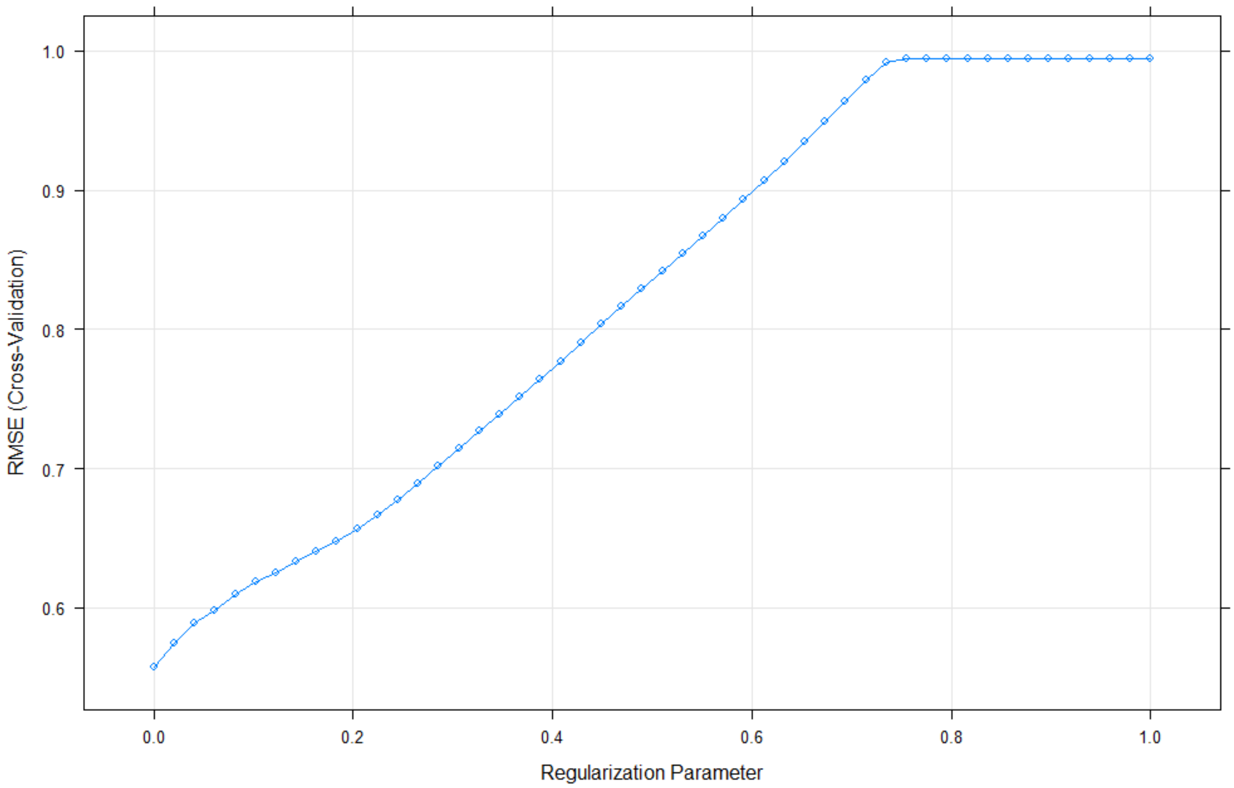
```
> lasso.fit <- train(x = x.train, y = y.train,
+                  method = 'glmnet',
+                  trControl = trainControl(method = 'cv', number = 10),
+                  tuneGrid = expand.grid(alpha = 1,
+                                         lambda = seq(0.0001, 1, length.out = 50)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  :
  There were missing values in resampled performance measures.
>
> (lasso.info <- postResample(predict(lasso.fit, x.test), y.test))
     RMSE  Rsquared       MAE
0.4721135 0.7465157 0.3645205
> coef(lasso.fit$finalModel, lasso.fit$bestTune$lambda)
14 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept)  2.408810e-16
crim        -1.062609e-01
zn           1.219673e-01
indus        6.639662e-02
chas         1.154210e-01
nox         -2.619868e-01
rm           1.879603e-01
age          2.588099e-02
dis         -3.650837e-01
rad          3.187337e-01
tax         -2.235858e-01
```

ptratio   -2.364365e-01

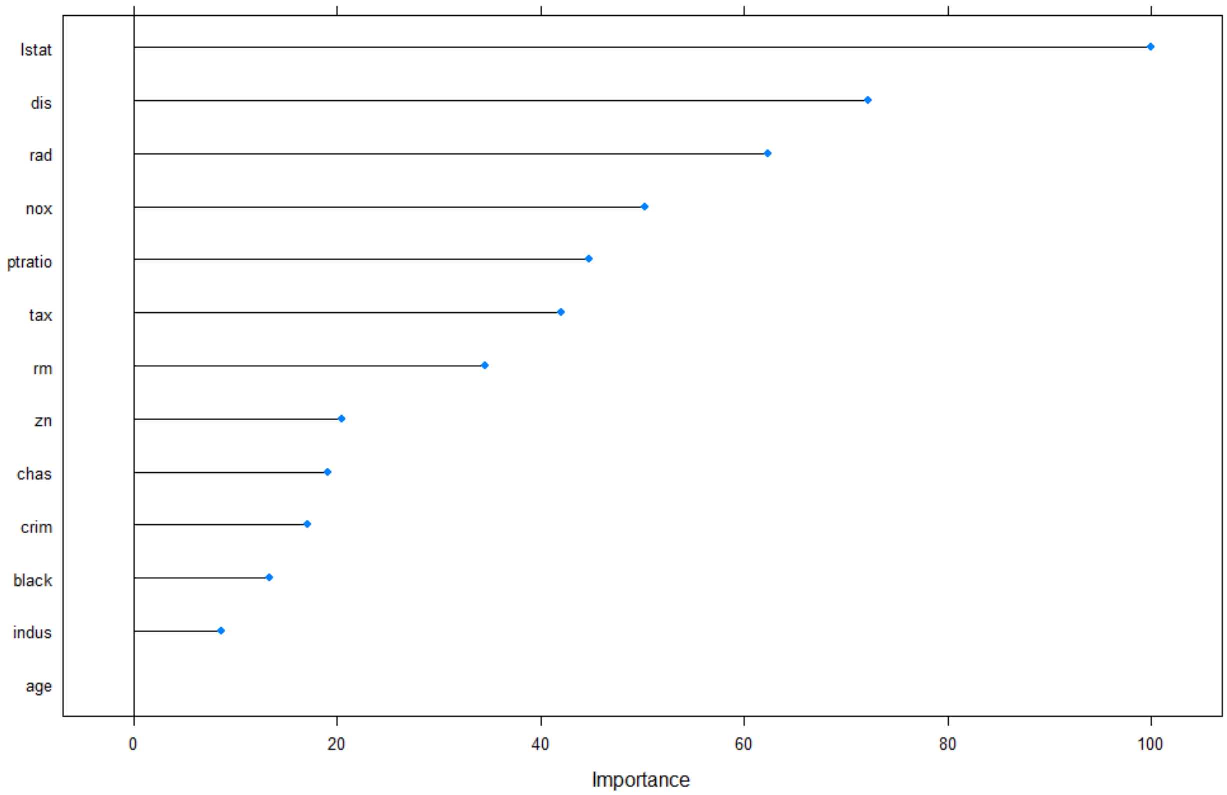
black   8.875764e-02

lstat   -4.958159e-01

> plot(lasso.fit)



```
> plot(varImp(lasso.fit))
```




```
> ###### ###### ###### ###### ###### ######
```

```
> ## 4) Implement ridge regression (with cross-validation to select the optimal tuning parameter) without interactions between them, report the best model (that is based on the optimal tuning parameter) and its fitted model, conduct hypothesis tests on some coefficients of the model and report your findings, and assess the prediction accuracy of the fitted model and report your findings. ##
>
> ridge.fit <- train(x = x.train, y = y.train,
+                 method = "glmnet",
+                 trControl = trainControl(
+                 method = "cv", number = 10),
+                 tuneGrid = expand.grid(alpha = 0,
+                                 lambda = seq(0, 10e2, length.out = 20)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  :
  There were missing values in resampled performance measures.
>
> (ridge.info <- postResample(predict(ridge.fit, x.test), y.test))
     RMSE  Rsquared       MAE
0.4603729 0.7633665 0.3513930
>
> coef(ridge.fit$finalModel, ridge.fit$bestTune$lambda)
14 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept)  2.445863e-16
crim        -8.823110e-02
zn           8.553090e-02
indus        1.631845e-02
chas         1.185078e-01
nox         -1.701335e-01
rm           2.166071e-01
age          8.986583e-03
```

dis   -2.749644e-01

rad   1.741917e-01

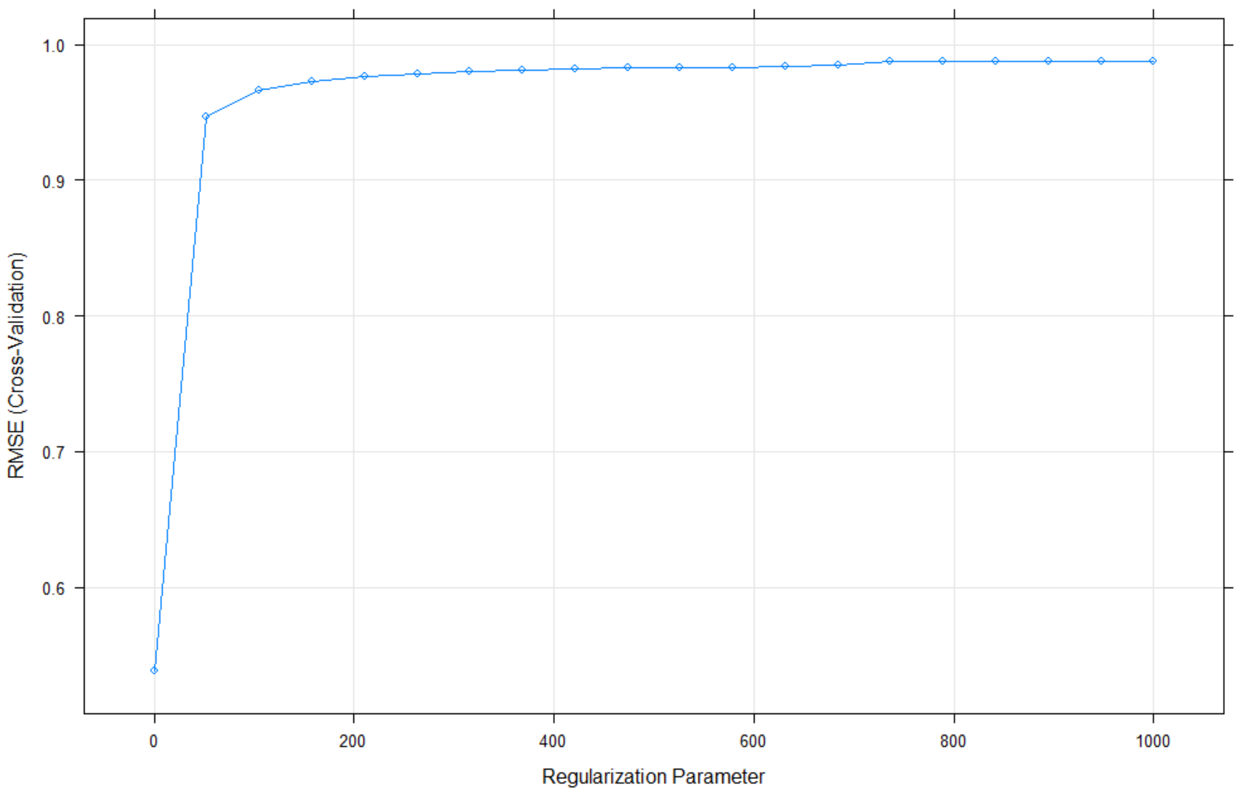
tax   -1.102646e-01

ptratio   -2.098198e-01

black   8.591837e-02

lstat   -4.450896e-01

> plot(ridge.fit)



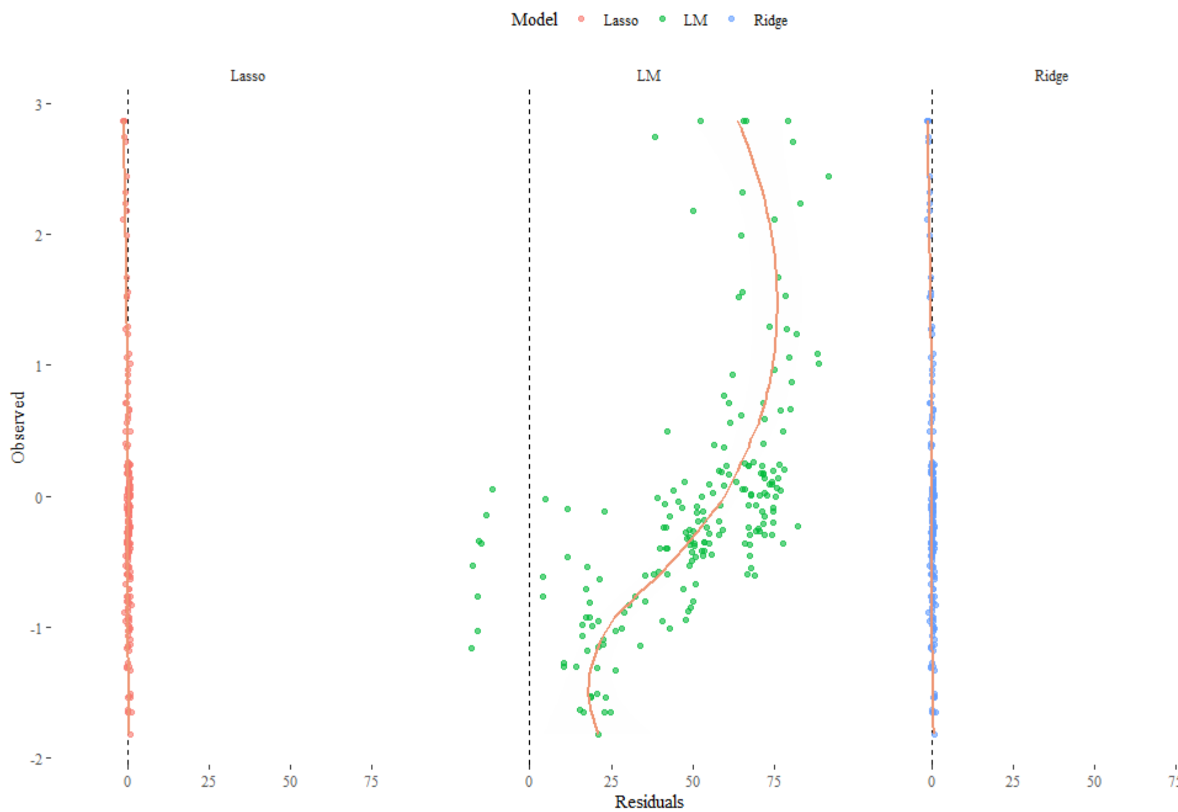
> ###### ###### ###### ###### ###### ######

```
> ## 4) Implement ridge regression (with cross-validation to select the optimal tuning parameter) without interactions between them, report the best model (that is based on the optimal tuning parameter) and its fitted model, conduct hypothesis tests on some coefficients of the model and report your findings, and assess the prediction accuracy of the fitted model and report your findings. ##
>
> ridge.fit <- train(x = x.train, y = y.train,
+                    method = "glmnet",
+                    trControl = trainControl(
+                    method = "cv", number = 10),
+                    tuneGrid = expand.grid(alpha = 0,
+                                           lambda = seq(0, 10e2, length.out = 20)))
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  :
  There were missing values in resampled performance measures.
>
> (ridge.info <- postResample(predict(ridge.fit, x.test), y.test))
     RMSE  Rsquared       MAE
0.4603729 0.7633665 0.3513930
>
> coef(ridge.fit$finalModel, ridge.fit$bestTune$lambda)
14 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept)  2.445863e-16
crim        -8.823110e-02
zn           8.553090e-02
indus        1.631845e-02
chas         1.185078e-01
nox         -1.701335e-01
rm           2.166071e-01
age          8.986583e-03
```

```
dis        -2.749644e-01

rad         1.741917e-01

tax        -1.102646e-01

ptratio    -2.098198e-01

black       8.591837e-02

lstat      -4.450896e-01

> plot(ridge.fit)

> ###### ###### ###### ###### ###### ######

> ## 5) Among the best/optimal models you would find in (2), (3) and (4) respectively, which one has the best prediction accuracy? If you consider a trade-off between the number of predictors in a model and its prediction accuracy, which among the best models you found in (2), (3) and (4) would you prefer? ##

>

> as_data_frame(rbind(lin.info,

+      ridge.info,

+      lasso.info))

# A tibble: 3 x 3

   RMSE Rsquared    MAE

  <dbl>    <dbl>  <dbl>

1 36.4      0.439 32.6

2  0.460    0.763  0.351

3  0.472    0.747  0.365

>

>

> testing %>%

+    summarize(sd = sd(medv))

        sd

1 0.9285333

> The Lasso and Ridge models performed similarly. R2≥70 for them all and RMSE≤53. However LM performed very differently, with R2≥36 for them all and RMSE≤33.8 (or 33800) I suspect this is because
```

of my code and not completely accurate results. When we compare the RMSE scores with the mean and standard deviation of the response variable we see that the models all have phenomenal accuracy.

```
Error: unexpected symbol in "The Lasso"
>
> residfunc <- function(fit, data) {
+  predict(fit, data) - testing$medv
+ }
>
> data_frame(Observed = testing$medv,
+           LM = residfunc(lin.model, testing),
+           Ridge = residfunc(ridge.fit, x.test),
+           Lasso = residfunc(lasso.fit, x.test)) %>%
+    gather(Model, Residuals, -Observed) %>%
+    ggplot(aes(Observed, Residuals, col = Model)) +
+    geom_hline(yintercept = 0, lty = 2) +
+    geom_point(alpha = 0.6) +
+    geom_smooth(method = 'loess', alpha = 0.01, col = 'lightsalmon2') +
+    facet_wrap(~ Model, ncol = 5) +
+    theme_tufte() +
+    theme(legend.position = 'top') +
+    coord_flip()
`geom_smooth()` using formula 'y ~ x'
```



Model
Lasso
LM
Ridge
Lasso
LM
Ridge
Observed
Residuals
3
2
1
0
-1
-2
0
25
50
75

> ###### ###### ###### ###### ###### ######