test-linear.R

rocka

2023-12-11

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.3.2

## Loaded glmnet 4.1-8
```

```r
library(ggcorrplot)
```

```
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.2
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(dplyr)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(leaps)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(coefplot)
library(car)
```

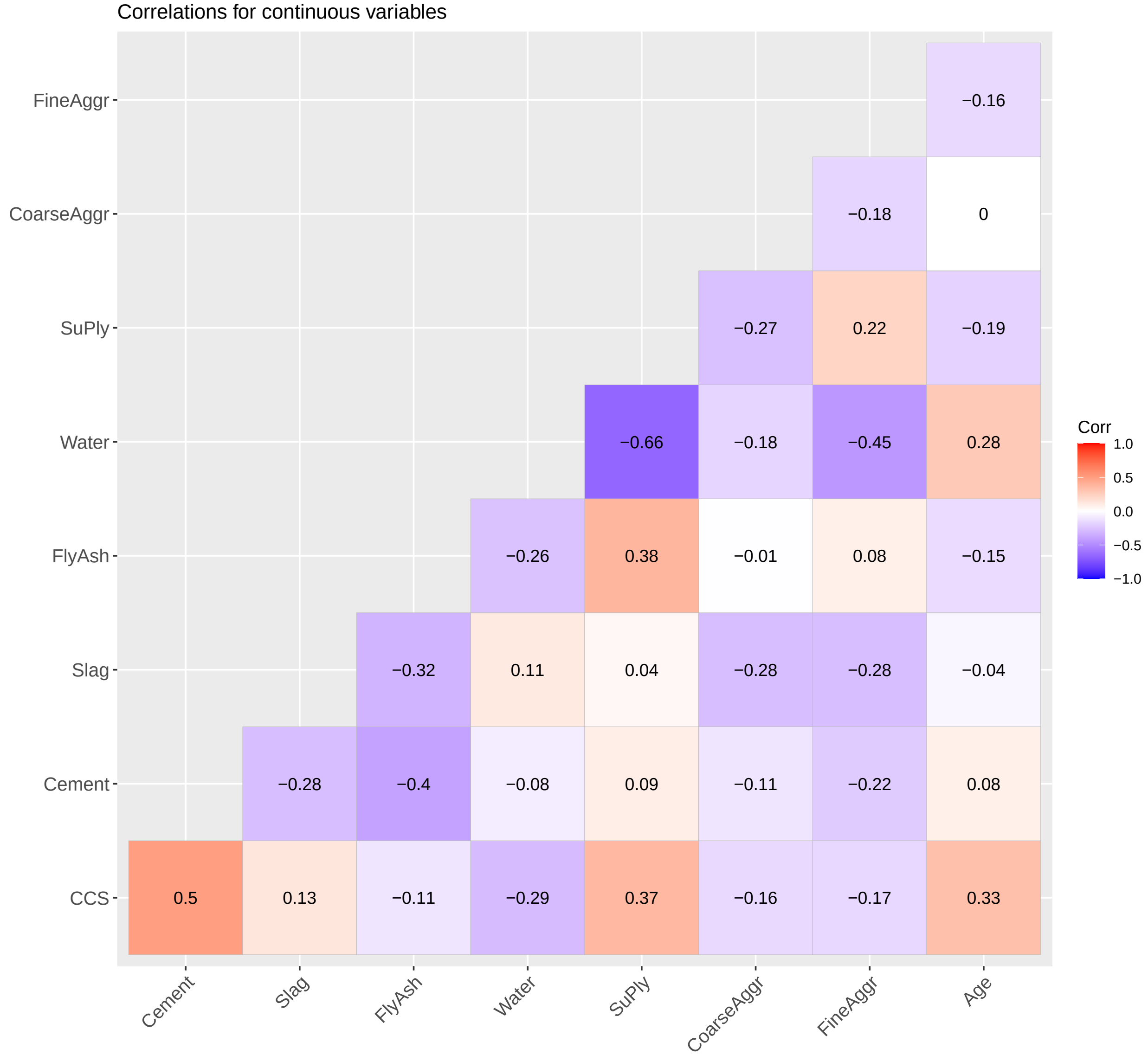
```
## Warning: package 'car' was built under R version 4.3.2

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.3.2

##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```r
df<-read.csv("C:/Users/rocka/OneDrive/Documents/output_file.csv")

na_count <- colSums(is.na(df))
print(na_count)
```

```
##     Cement       Slag     FlyAsh      Water      SuPly CoarseAggr   FineAggr
##          0          0          0          0          0          0          0
##        Age        CCS
##          0          0
```

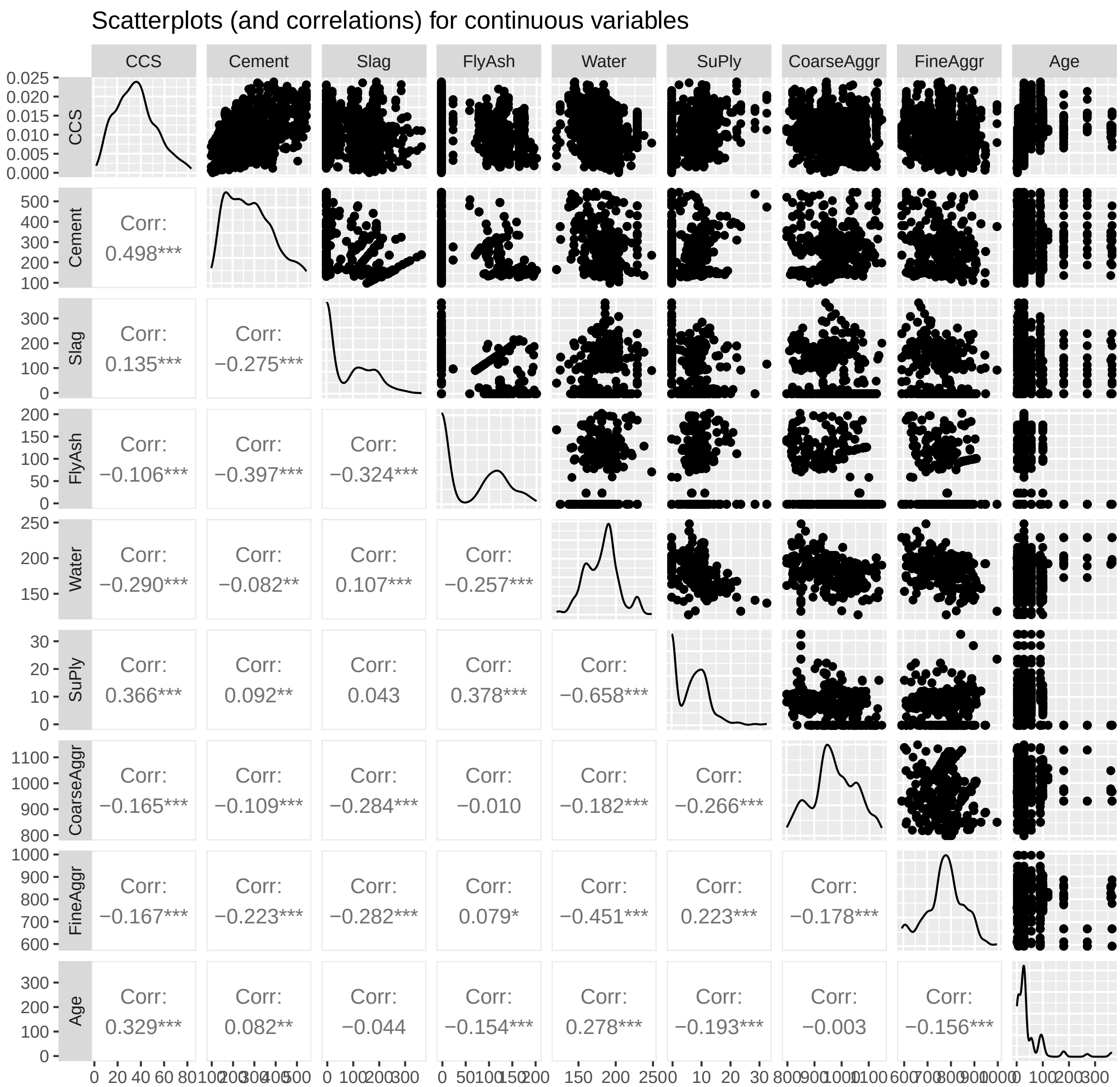
```r
#Correlation Plots

varsnum <-select(df, where(is.numeric)) |>
  relocate(CCS)
```

```r
ggcorrplot(corr=cor(varsnum), type="lower",
           ggtheme=ggplot2::theme_gray, lab=TRUE,
           title="Correlations for continuous variables")
```




```
gg <- ggpairs(varsnum,
              switch="y", upper=list(continuous="points"),
              lower=list(continuous="cor"))
```

```
gg + labs(title="Scatterplots (and correlations) for continuous variables")
```




```r
source("C:/Users/rocka/OneDrive/Documents/rss_regress_funcs_v5.R")

df <- rename(df, response=CCS)

lmodel<-lm(response~ . ,df)
summary(lmodel)
```

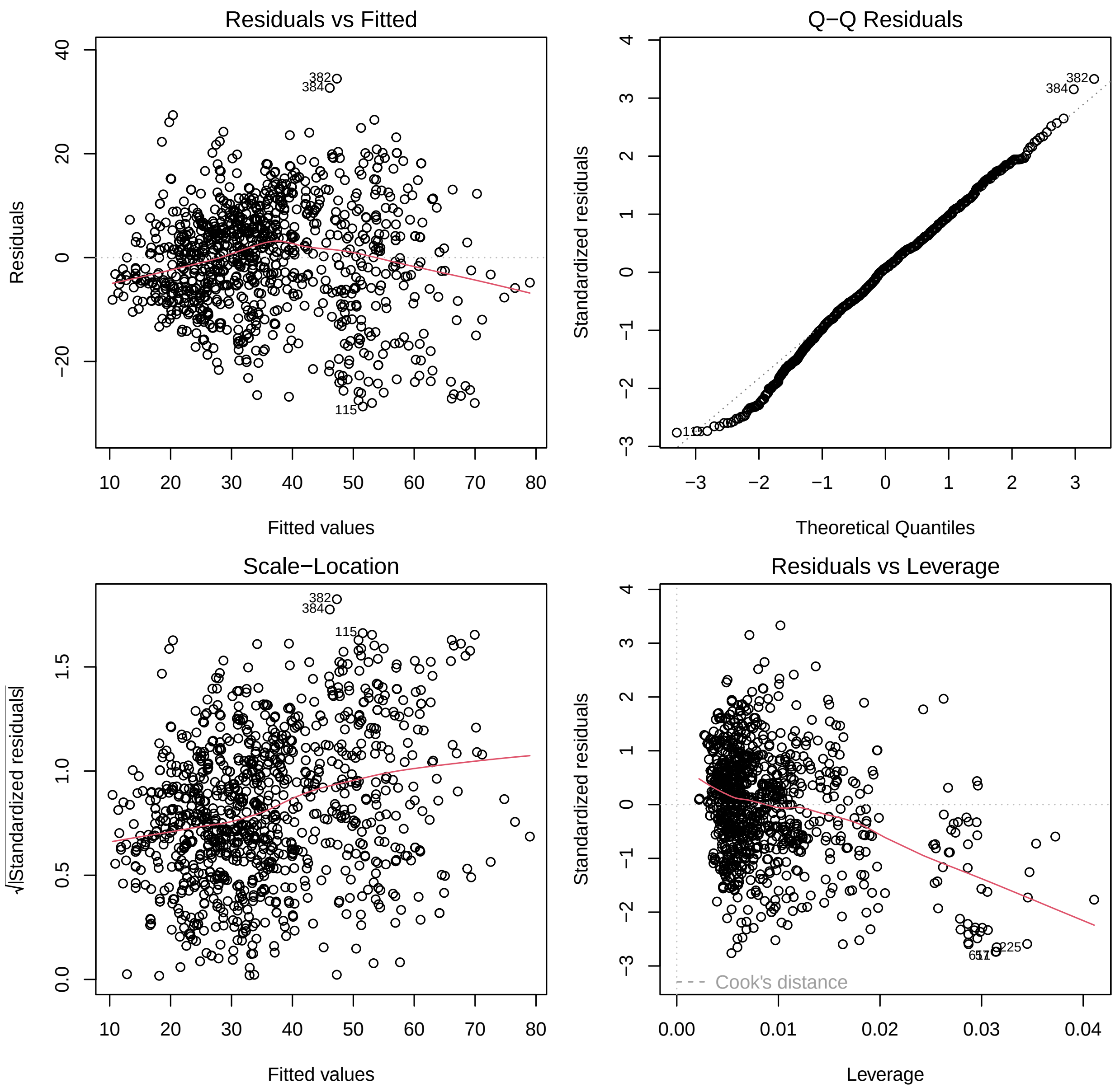
```
##
## Call:
## lm(formula = response ~ ., data = df)
##
## Residuals:
##      Min       1Q  Median       3Q      Max
## -28.654  -6.302   0.703   6.569  34.450
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -23.331214  26.585504  -0.878 0.380372
## Cement        0.119804   0.008489  14.113  < 2e-16 ***
## Slag          0.103866   0.010136  10.247  < 2e-16 ***
## FlyAsh        0.087934   0.012583   6.988 5.02e-12 ***
## Water        -0.149918   0.040177  -3.731 0.000201 ***
## SuPly         0.292225   0.093424   3.128 0.001810 **
## CoarseAggr    0.018086   0.009392   1.926 0.054425 .
## FineAggr      0.020190   0.010702   1.887 0.059491 .
## Age           0.114222   0.005427  21.046  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.4 on 1021 degrees of freedom
## Multiple R-squared:  0.6155, Adjusted R-squared:  0.6125
## F-statistic: 204.3 on 8 and 1021 DF,  p-value: < 2.2e-16
```

```r
vif_results <- vif(lmodel)

print(vif_results)
```

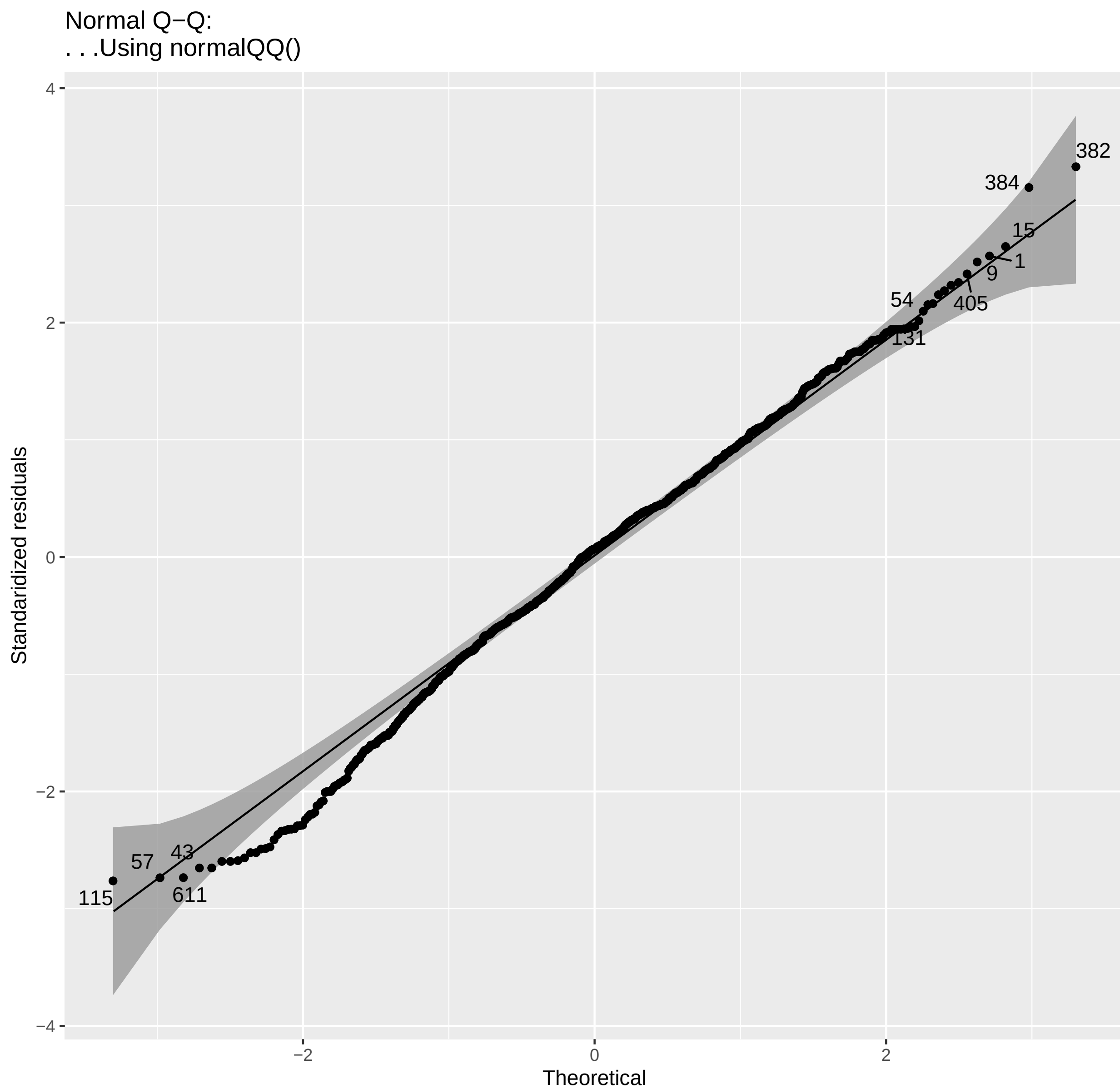
```
##     Cement       Slag     FlyAsh      Water      SuPly CoarseAggr   FineAggr
##   7.488944   7.276963   6.170634   7.003957   2.963776   5.074617   7.005081
##        Age
##   1.118367
```

```r
par(mar=c(4, 4, 2, 1))
par(mfrow=c(2,2))
plot(lmodel)
```




```r
normalQQ(lmodel,title2=". . .Using normalQQ()")
```

```
## Warning: ggrepel: 36 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```r
# Linear regression using Best subset
```

```r
SEED<-1234
set.seed(SEED)

npredictors<-8

regfit.full <- regsubsets(response~., data=df) #nvmax not essential(8)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(response ~ ., data = df)
## 8 Variables  (and intercept)
##            Forced in Forced out
```

```
## Cement          FALSE      FALSE
## Slag            FALSE      FALSE
## FlyAsh          FALSE      FALSE
## Water           FALSE      FALSE
## SuPly           FALSE      FALSE
## CoarseAggr      FALSE      FALSE
## FineAggr        FALSE      FALSE
## Age             FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          Cement Slag FlyAsh Water SuPly CoarseAggr FineAggr Age
## 1  ( 1 ) "*"    " "  " "    " "   " "   " "        " "      " "
## 2  ( 1 ) "*"    " "  " "    " "   "*"   " "        " "      " "
## 3  ( 1 ) "*"    " "  " "    " "   "*"   " "        " "      "*"
## 4  ( 1 ) "*"    "*"  " "    "*"   " "   " "        " "      "*"
## 5  ( 1 ) "*"    "*"  "*"    "*"   " "   " "        " "      "*"
## 6  ( 1 ) "*"    "*"  "*"    "*"   "*"   " "        " "      "*"
## 7  ( 1 ) "*"    "*"  "*"    "*"   "*"   "*"        " "      "*"
## 8  ( 1 ) "*"    "*"  "*"    "*"   "*"   "*"        "*"      "*"
```

```
reg.summary <- summary(regfit.full)
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

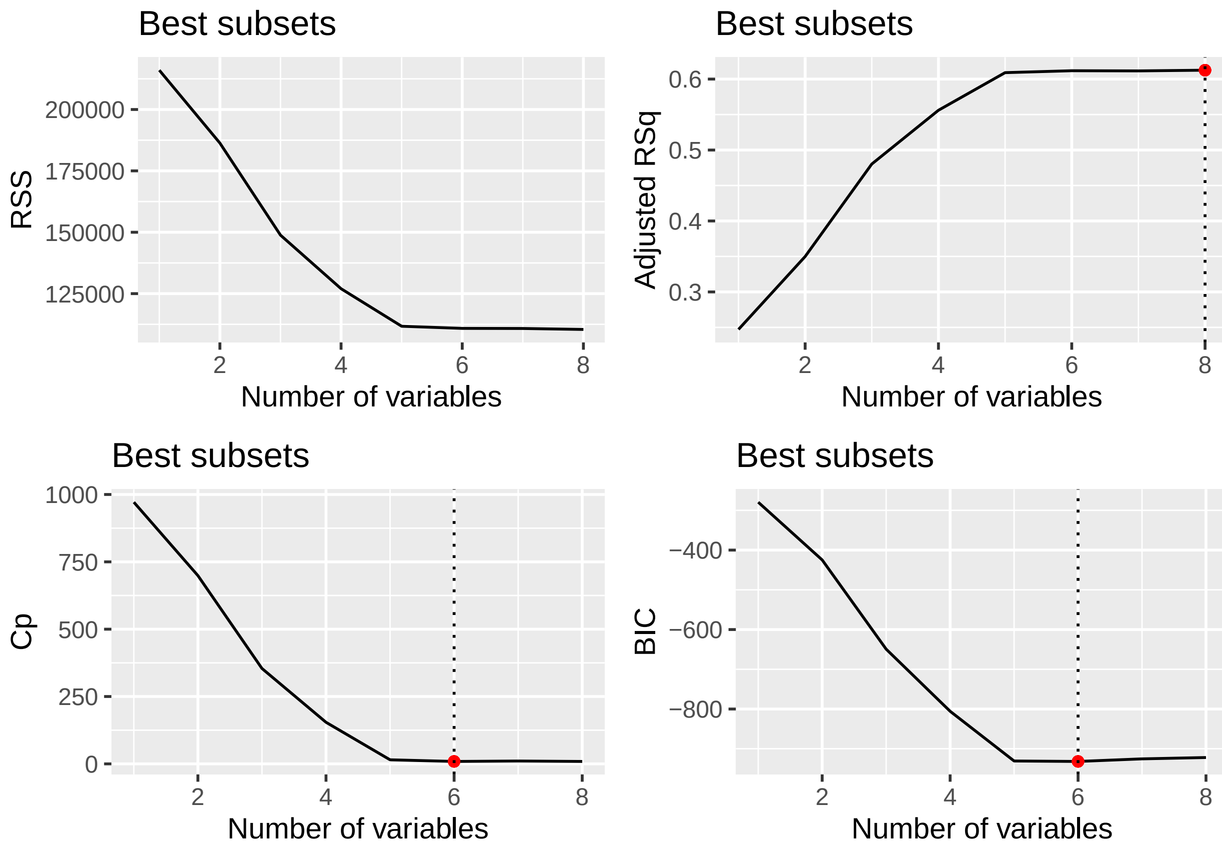
```
reg.summary$rsq
```

```
## [1] 0.2478366 0.3511737 0.4817538 0.5577500 0.6109756 0.6140224 0.6141795
## [8] 0.6155199
```

```
best.plot <- function(varName, varLabel, minmax=" ") {
  gg <- ggplot(data.frame(varName), aes(x=seq_along(varName), y=varName)) +
    geom_line() +
    labs(x="Number of variables", y=varLabel, title="Best subsets")
  if (minmax=="min") {
    gg <- gg + geom_point(aes(x=which.min(varName), y=min(varName)),
                          color="red") +
      geom_vline(aes(xintercept=which.min(varName)), linetype=
                   "dotted")
  }
  if (minmax=="max") {
    gg <- gg + geom_point(aes(x=which.max(varName), y=max(varName)),
                          color="red") +
      geom_vline(aes(xintercept=which.max(varName)), linetype=
                   "dotted")
  }
  return(gg)
}

results <- with(reg.summary, data.frame(rss,adjr2,cp,bic))
names(results)
```

```
## [1] "rss"   "adjr2" "cp"    "bic"
```

```r
grid.arrange(best.plot(results$rss, "RSS"),
             best.plot(results$adjr2, "Adjusted RSq"
                       , "max"),
             best.plot(results$cp, "Cp", minmax="min"),
             best.plot(results$bic, "BIC", minmax="min"),
             ncol=2)
```




```r
#Use plot.regsubsets() to plot accuracy measures vs selected predictors
dev.off()
```

```
## null device
##           1
```

```r
plot(regfit.full,scale="r2")

plot(regfit.full,scale="adjr2")

plot(regfit.full,scale="Cp")

plot(regfit.full,scale="bic")



# Display coefficients for best 6- and 8-variables subsets
round(coef(regfit.full,6), 3)
```

```
## (Intercept)      Cement        Slag      FlyAsh       Water       SuPly
##      28.993       0.105       0.086       0.069      -0.218       0.240
##         Age
##       0.113
```

```r
round(coef(regfit.full,8), 3)
```

```
## (Intercept)      Cement        Slag      FlyAsh       Water       SuPly
##     -23.331       0.120       0.104       0.088      -0.150       0.292
##  CoarseAggr    FineAggr         Age
##       0.018       0.020       0.114
```

```r
# Compute training-data MSEs for best 6 & 8 variable models
# for comparison with later cross-validated results

(MSE6 <- results$rss[6]/nrow(df))
```

```
## [1] 107.6148
```

```r
(MSE8 <- results$rss[8]/nrow(df))
```

```
## [1] 107.1972
```

```r
# Results so far:
#                     # predictors  training-set MSE
#   Best BIC & Cp model:      6         107.6148
#   Best Adj Rsq model:       8         107.1972



# Use a validation set to choose among models

# Use sampling with replacement to select an approx 50% training sample
set.seed(SEED)
train <- sample(c(TRUE,FALSE), nrow(df), replace=TRUE)
str(train)
```

```
##  logi [1:1030] FALSE FALSE FALSE FALSE TRUE FALSE ...
```

```r
test <- (!train)

# Compute test-set mean sum-of-squares for use in computing
# test-set R-squared
mean <- mean(df$response[test])
MSS <- mean((df$response[test]-mean)^2)

# Use the training sample to determine best subsets (for RSS)
regfit.best <- regsubsets(response~., data=df[train,], nvmax=npredictors)

# Use the validation set to compute test-set MSEs.  There is no predict()
# function for regsubsets(,) so need code to compute predictions. This is
```

```r
# done by creating an X matrix and then using matrix multiplication.
test.mat=model.matrix(response~.,data=df[test,]) # Create X matrix
val.errors=rep(NA,npredictors)                    # Reserves space to hold MSE's
for(i in 1:npredictors){
  coefi <- coef(regfit.best,id=i) # Coefficients for selected variables
  pred <- test.mat[,names(coefi)]%*%coefi # multiply X matrix by coefficients
  val.errors[i] <- mean((df$response[test]-pred)^2)  # mean squared error
}

val.errors
```

```
## [1] 205.7370 178.0775 144.5313 133.3471 111.8174 112.6201 115.3067 113.5548
```

```r
which.min(val.errors)
```

```
## [1] 5
```

```r
round(coef(regfit.best, 5),3)  # Display coefficients for this model
```

```
## (Intercept)      Cement        Slag      FlyAsh       Water         Age
##      29.435       0.109       0.100       0.082      -0.231       0.117
```

```r
# Repeat without having to view the console output to see # of variables
(pmin <- which.min(val.errors))
```

```
## [1] 5
```

```r
round(coef(regfit.best,pmin), 3)
```

```
## (Intercept)      Cement        Slag      FlyAsh       Water         Age
##      29.435       0.109       0.100       0.082      -0.231       0.117
```

```r
# Display MSE, root-mean-squared error (RMSE), and test-set R-squared
(MSE <-val.errors[pmin])
```

```
## [1] 111.8174
```

```r
(RMSE <- sqrt(MSE))
```

```
## [1] 10.57438
```

```r
(Rsq <- 1 - MSE/MSS)
```

```
## [1] 0.5957348
```

```r
# Finally, use the full data set to calculate the coefficients for the
# best 5 variable model
reg.best <- regsubsets(response~., data=df, nvmax=npredictors)
round(coef(reg.best,5),3)
```

```
## (Intercept)      Cement       Slag     FlyAsh      Water        Age
##      34.826       0.110      0.092      0.080     -0.255      0.114
```

```r
# Results so far:
#                     # predictors  training-set MSE   Test MSE
#   Best BIC & Cp model:      6          107.6148
#   Best Adj Rsq model:       8          107.1972
#
#   Validation set:
#     Best test-set model     5                         111.8174

# 4. ----------------------------------------------------------------------
#  Use cross validation to choose among models.

# First, create a user-defined predict function for regsubsets().
# This function uses syntax not taught in class.
predict.regsubsets <- function(object, newdata, id, ...){
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form,newdata)
  coefi <- coef(object, id=id)
  xvars <- names(coefi)
  mat[,xvars] %*% coefi
}

# Create random assignments for  k=10 folds
k <- 10
n <- nrow(df)
set.seed(SEED)
folds=sample(rep(1:k, length=n))

# Reserve space for a matrix to store the cross-validated MSEs
# for k test folds and all of the best models.  Initialize to NA.
cv.errors=matrix(NA,k,npredictors,
                 dimnames=list(NULL, paste(1:npredictors)))

# For each test fold, use the k-1 other folds to determine best models and
# use the user-defined predict() function to compute each model's test-fold MSE.
for(j in 1:k){
  best.fit=regsubsets(response~.,data=df[folds!=j,],nvmax=npredictors)
  for(i in 1:npredictors){
    pred=predict(best.fit,df[folds==j,], id=i)
    cv.errors[j,i] <- mean( (df$response[folds==j] - pred)^2)
  }
}

# Compute the cross-validated MSE for each model.
mean.cv.errors = rep(0, npredictors)
for (i in 1:npredictors) {
```

```r
  mean.cv.errors[i] <- mean(cv.errors[,i])
}
round(mean.cv.errors,0)
```

```
## [1] 211 182 145 125 110 109 110 110
```

```r
# Use the user-defined best.plot function to display results
best.plot(mean.cv.errors, "Cross-validated MSE", "min")

# Display cross-validated MSE and RMSE for best 6-variable model
(MSE <- mean.cv.errors[6])
```

```
## [1] 109.1922
```

```r
(RMSE <- sqrt(MSE))
```

```
## [1] 10.4495
```

```r
mean <- mean(df$response)
MSS <- mean((df$response-mean)^2)
(Rsq <- 1 - MSE/MSS)
```

```
## [1] 0.6083648
```

```r
# # Finally, use the full data set to calculate the coefficients for the
# best 6 variable model
reg.best <- regsubsets(response~., data=df, nvmax=npredictors)
round(coef(reg.best,6),3)
```

```
## (Intercept)      Cement        Slag      FlyAsh       Water       SuPly
##      28.993       0.105       0.086       0.069      -0.218       0.240
##         Age
##       0.113
```

```r
# Results so far:
#                      # predictors  training-set MSE   Test MSE
#   Best BIC & Cp model:      6         107.6148
#   Best Adj Rsq model:       8         107.1972
#
#   Validation set:
#   Best test-set model       5                           111.8174
#   K-fold CV best model      6                           109.1922
```

```r
#Lasso regression
```

```r
y<-df$response
x<-model.matrix(response~.,df)[,-1]

# Create training set
```

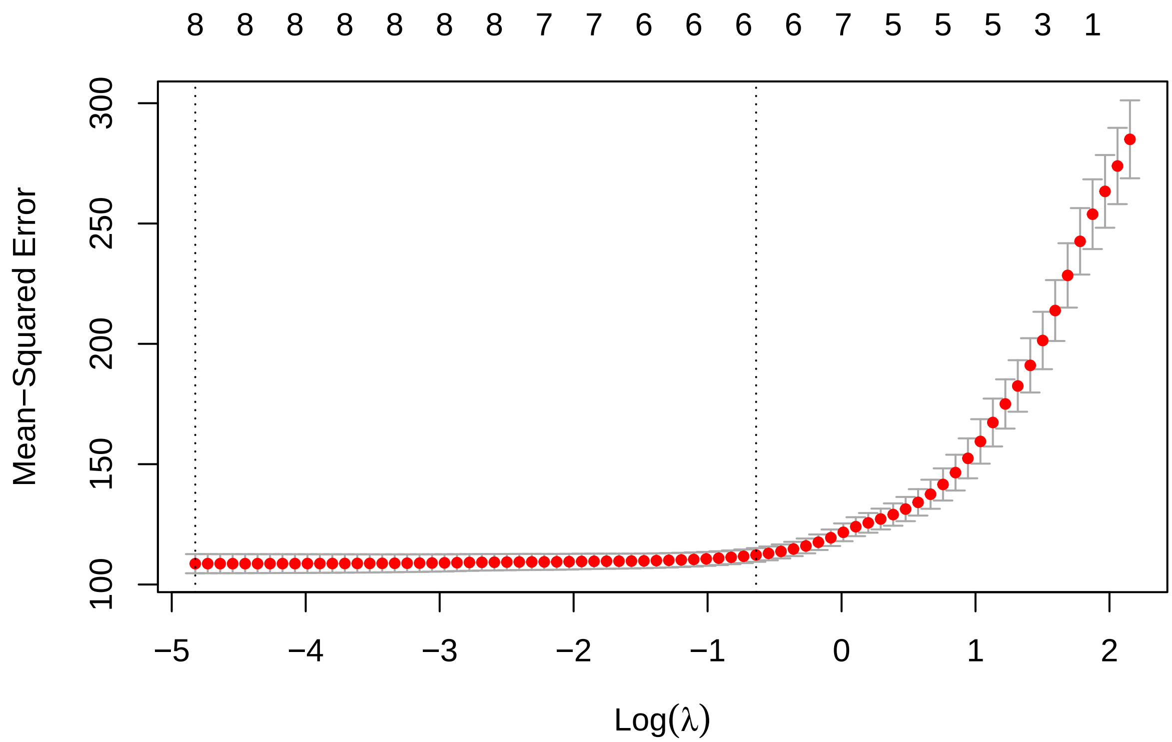
```r
set.seed(SEED)
train <- sample(1:nrow(x), size=nrow(x)/2)
test <- (-train)

# Recompute test-set mean sum-of-squares for use in computing
# test-set R-squared
mean <- mean(df$response[test])
MSS <- mean((df$response[test]-mean)^2)

# Use training set to perform lasso regression for 100 lambda values
# from 10^10 to 1/10^189.  alpha=1 for lasso(default)
grid <- 10^seq(10,-2,length=100)
lasso.mod <- glmnet(x[train,],y[train], alpha=1, lambda=grid)

# Cross validation of the training set determines the lambda minimizing MSE.
# Also, determines lambda for the 1-standard-error rule.
set.seed(SEED)
cv.out <- cv.glmnet(x[train,],y[train],alpha=1, nfold=10)
(bestlam <- cv.out$lambda.min)
```

```
## [1] 0.008031182
```

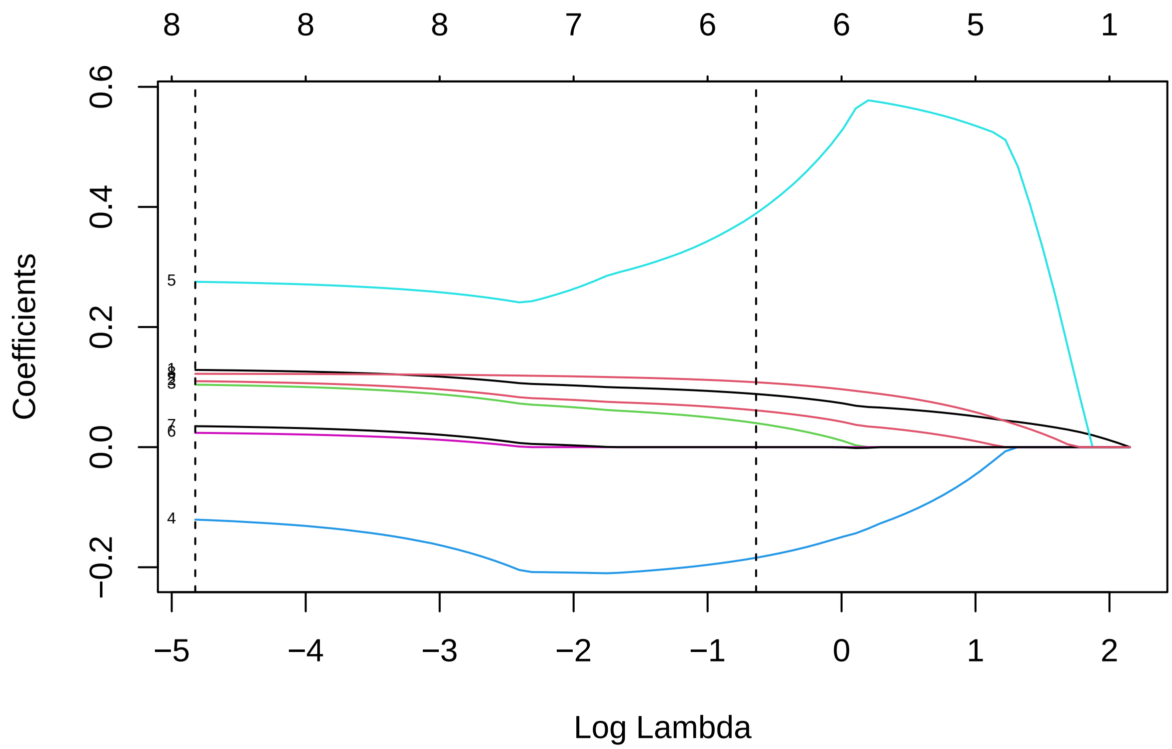
```r
(bestlam.1se <- cv.out$lambda.1se)
```

```
## [1] 0.5283982
```

```r
plot(cv.out)
```




```
# Plot coefficient paths using plot() function in glmnet package
par(mfrow=c(1,1))
plot(cv.out$glmnet.fit, xvar="lambda", label=TRUE) # function in glmnet package
abline(v=log(c(bestlam, bestlam.1se)), lty=2)
```




```
# Determine test-set MSE for lambda minimizing cross-validated training-set MSE
y.test <- y[test]
lasso.pred.min=predict(lasso.mod, s=bestlam, newx=x[test,])
(MSE.lasso.min <- mean((lasso.pred.min-y.test)^2))
```

```
## [1] 113.5209
```

```
(RMSE <- sqrt(MSE.lasso.min))
```

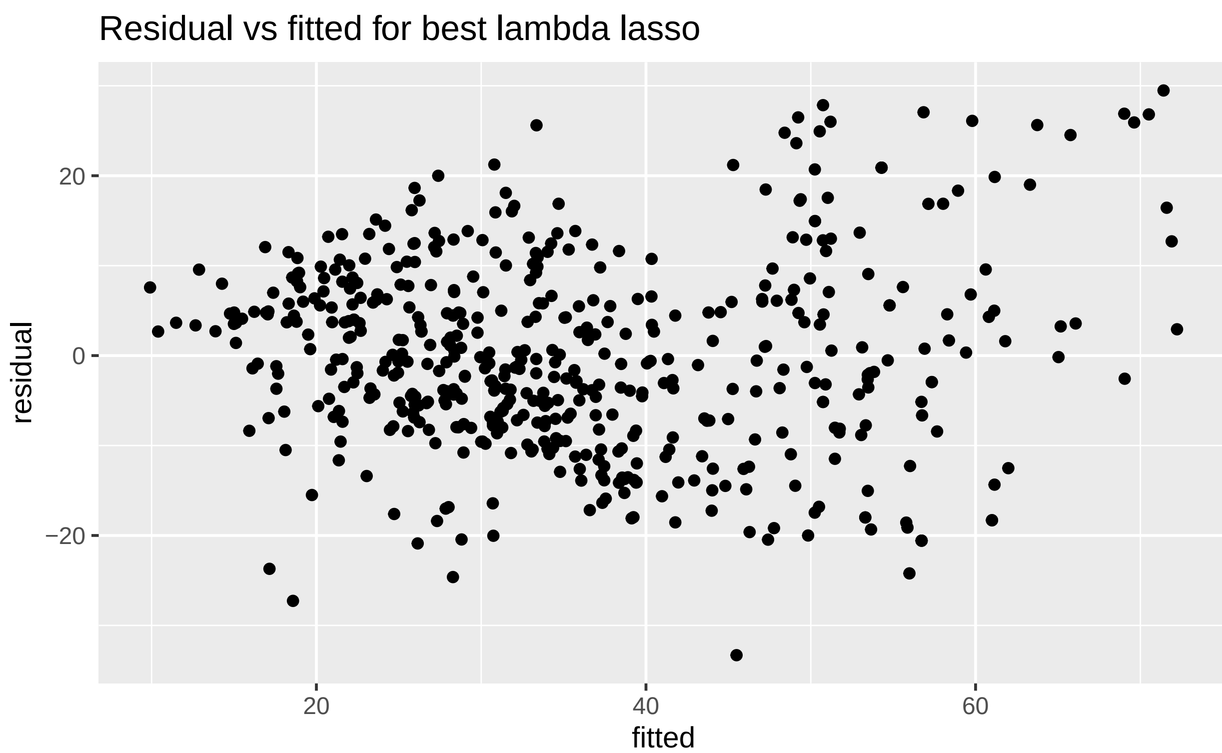
```
## [1] 10.65462
```

```
(Rsq <- 1 - MSE.lasso.min/MSS)
```

```
## [1] 0.5834291
```

```
#Plot residuals vs fitted for best-lambda lasso
```

```
ggplot(data.frame(y.test,lasso.pred.min),
       aes(x=lasso.pred.min,
           y=lasso.pred.min-y.test)) +
  geom_point() +
  labs(x="fitted", y="residual",
       title="Residual vs fitted for best lambda lasso")
```




```
# Determine test-set MSE for lambda from 1-standard-error rule
lasso.pred.1se=predict(lasso.mod, s=bestlam.1se, newx=x[test,])
(MSE.lasso.1se <-mean((lasso.pred.1se-y.test)^2))
```

```
## [1] 114.7777
```

```
(RMSE <- sqrt(MSE.lasso.1se))
```

```
## [1] 10.71343
```

```
(Rsq <- 1 - MSE.lasso.1se/MSS)
```

```
## [1] 0.5788173
```

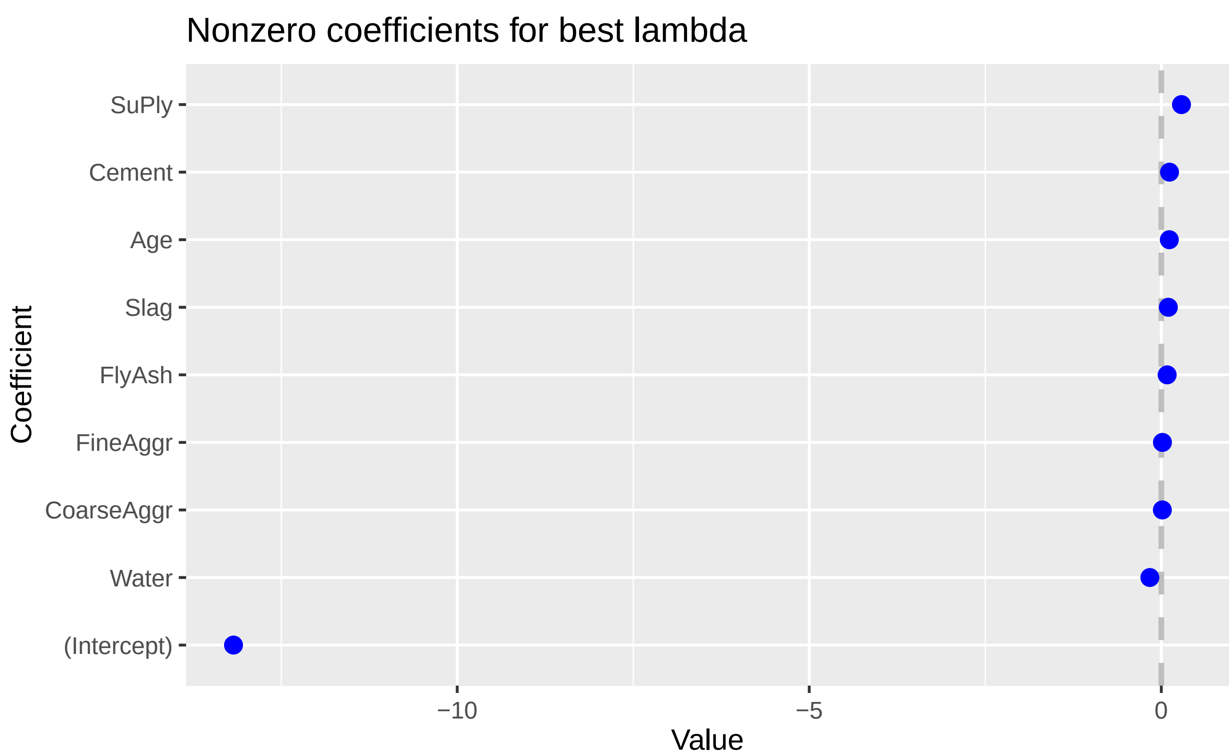
```
# Finally, use the full data set to perform lasso regression and
# display non-zero coefficients for best lambda determined from
# cross validation of the training set
out <- glmnet(x,y, alpha=1, lambda=grid)
lasso.coef.min <- predict(out, type="coefficients",
                          s=bestlam)[1:(ncol(x)+1),]
round(lasso.coef.min[lasso.coef.min != 0], 3)
```

```
## (Intercept)      Cement        Slag      FlyAsh       Water       SuPly
##     -13.177       0.117       0.100       0.083      -0.162       0.287
##  CoarseAggr    FineAggr         Age
##       0.015       0.016       0.114
```

```r
# Number of predictors with non-zero coefficients for best lambda
(nvars <- length(lasso.coef.min[lasso.coef.min!=0])-1)
```

```
## [1] 8
```

```r
# Use coefplot() in coefplot package to plot coefficients for best lambda
coefplot(out, lambda=bestlam, sort='magnitude',
         title="Nonzero coefficients for best lambda")
```



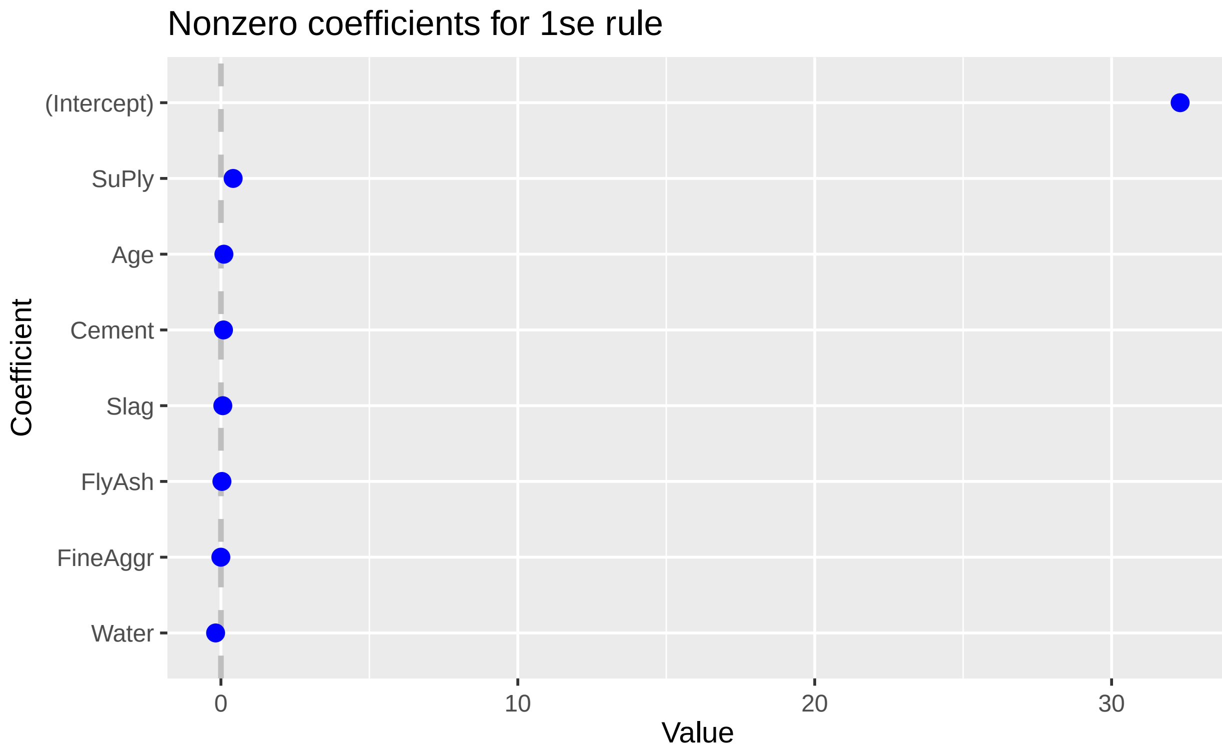

```r
# Also, display non-zero coefficients for lambda from 1se rule
lasso.coef.1se <- predict(out,type="coefficients",s=bestlam.1se)[1:(ncol(x)+1),]
round(lasso.coef.1se[lasso.coef.1se!=0], 3)
```

```
## (Intercept)      Cement        Slag      FlyAsh       Water       SuPly
##      32.309       0.086       0.063       0.033      -0.179       0.411
##    FineAggr         Age
##      -0.002       0.100
```

```r
# Number of predictors with non-zero coefficients for 1se rule
(nvars <- length(lasso.coef.1se[lasso.coef.1se!=0])-1)
```

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

```r
# Use coefplot() in coefplot package to plot coefficients for 1se rule
coefplot(out, lambda=bestlam.1se, sort='magnitude',
         title="Nonzero coefficients for 1se rule")
```




```
# Results so far:
#                     # predictors  training-set MSE   Test MSE
#   Best BIC & Cp model:      6         107.6148
#   Best Adj Rsq model:       8         107.1972
#
#   Validation set:
#   Best test-set model       5                          111.8174
#   K-fold CV best model      6                          109.1922
#
#   Best-lambda Lasso         8                          113.5209
#   1se-lambda Lasso          7                          114.7777
```