# Project #2

Trennon,Jessie, Jose

5/4/2020

Project Number 2 (due for your group by 1:30pm on May 6th, the end of the final exam period for the class). Working on the same teams as last time, working with the same data as last time, and working with the same response variable as last time, do the following steps:

```
library(tidyverse)
```

```
## ── Attaching packages ───────────────────────────────────────────────────────
───────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.0     ✓ purrr   0.3.4
## ✓ tibble  3.0.1     ✓ dplyr   0.8.5
## ✓ tidyr   1.0.2     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────────────────────────────
──────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(leaps)
library(dplyr)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(readxl)
library(ggplot2)
library(GGally)
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
library(DT)
library(caTools)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loaded glmnet 2.0-18
```

```
library(leaps)
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

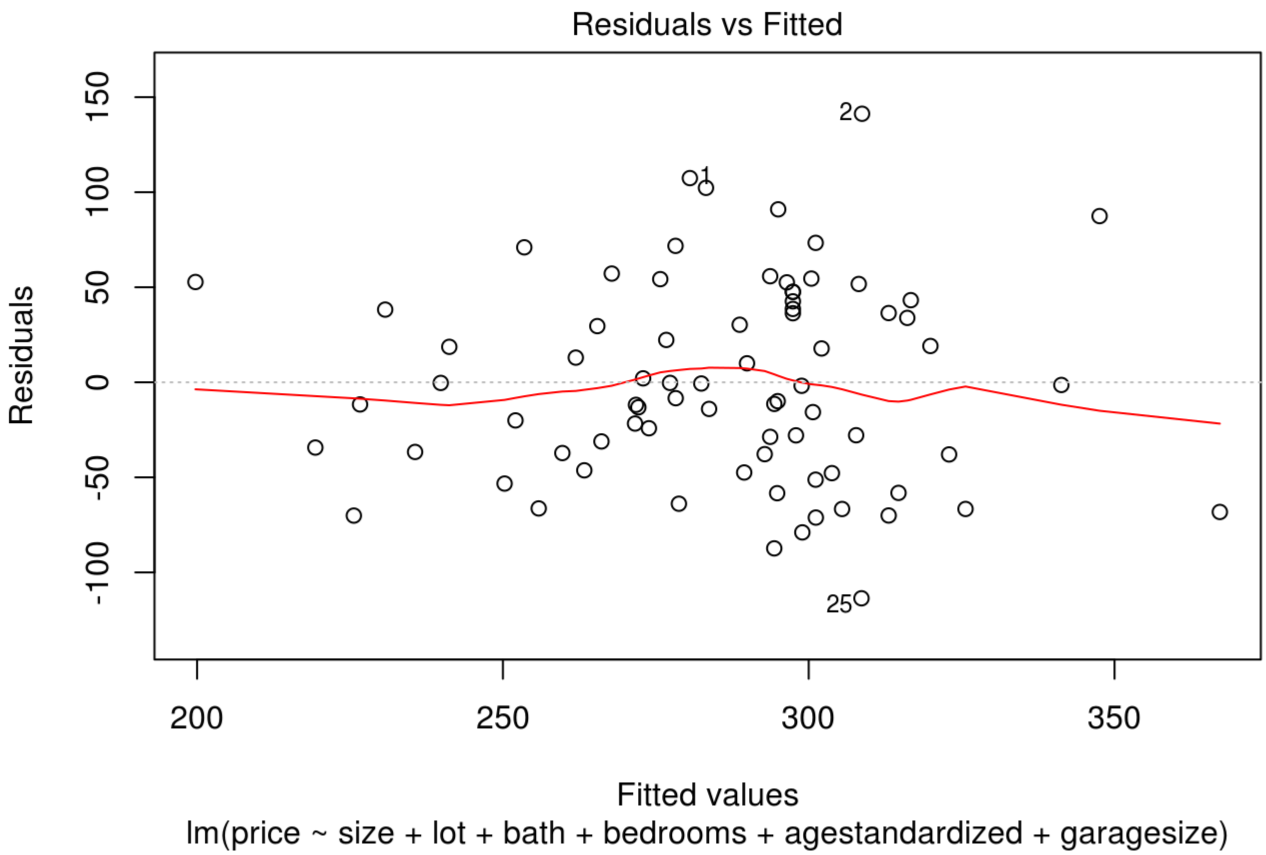
```
## The following object is masked from 'package:stats':
##
##     loadings
```

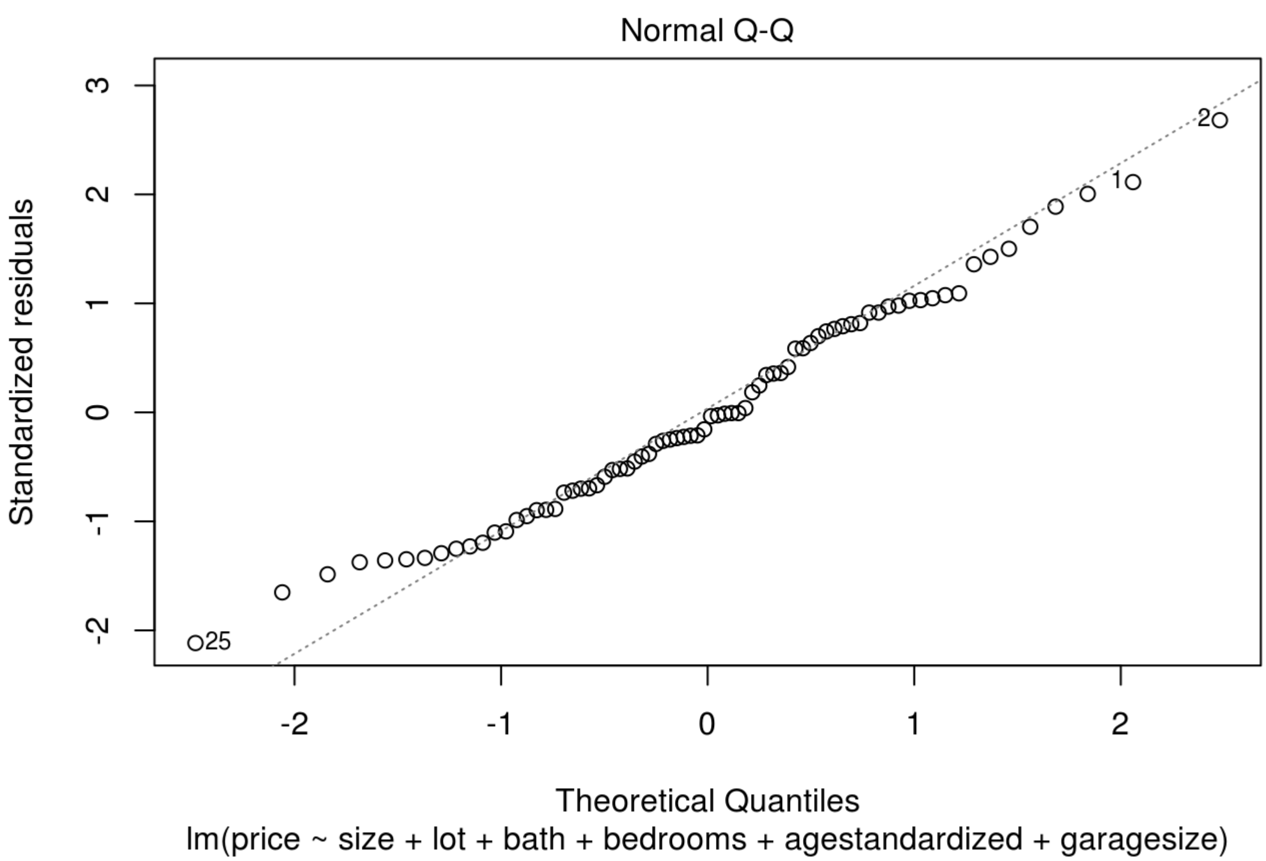
```
housing <- read_excel("Housing.xlsx")
attach(housing)
set.seed(1)
```

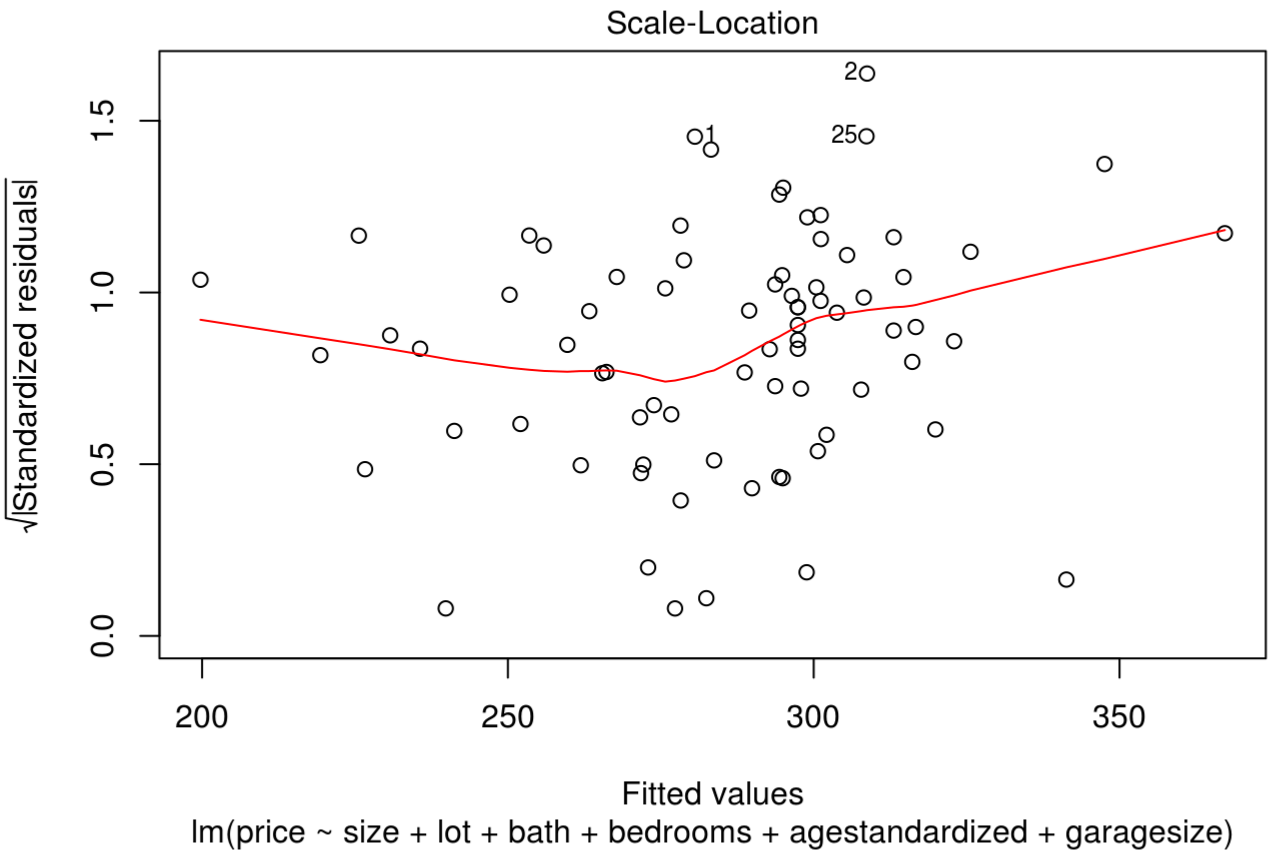
# a.

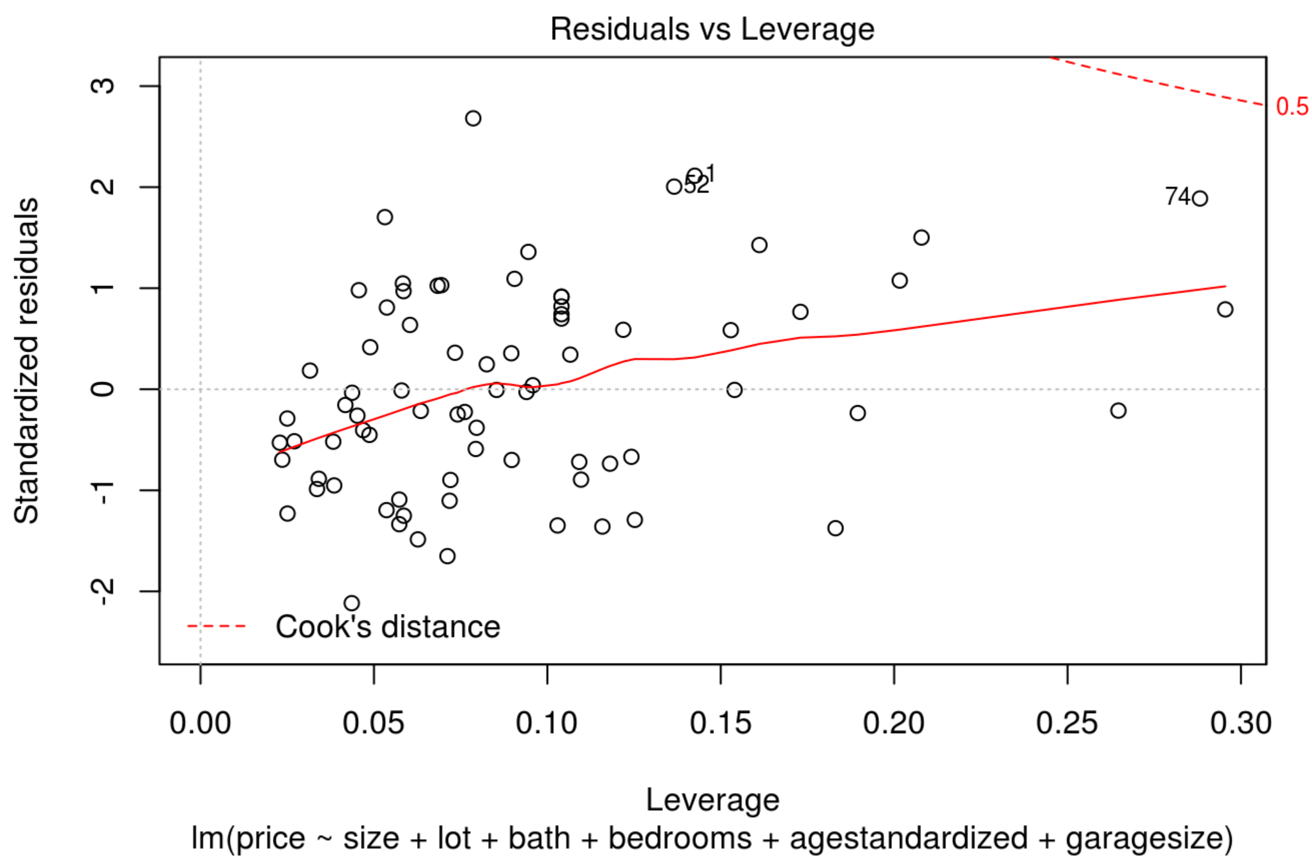
Consider the model that you arrived at in the previous project as the first candidate model.

```
finalmod <-lm(price ~ size + lot + bath + bedrooms + agestandardized + garagesize, data = housin
g)
plot(finalmod)
```


Residuals vs Fitted
Residuals
150
100
50
0
-50
-100
200
250
300
350
2
1
25
Fitted values
lm(price ~ size + lot + bath + bedrooms + agestandardized + garagesize)


Normal Q-Q
Standardized residuals
3
2
1
0
-1
-2
-2
-1
0
1
2
25
Theoretical Quantiles
lm(price ~ size + lot + bath + bedrooms + agestandardized + garagesize)


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
200
250
300
350
2
1
25
Fitted values
lm(price ~ size + lot + bath + bedrooms + agestandardized + garagesize)


Residuals vs Leverage
Standardized residuals
-2
-1
0
1
2
3
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.5
1
52
74
Cook's distance
Leverage
lm(price ~ size + lot + bath + bedrooms + agestandardized + garagesize)

```
summary(finalmod)
```

```
##
## Call:
## lm(formula = price ~ size + lot + bath + bedrooms + agestandardized +
##     garagesize, data = housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -113.63  -37.84   -5.10   39.59  141.28
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      181.482     64.574   2.810  0.00643 **
## size              49.079     35.733   1.373  0.17405
## lot                5.291      4.186   1.264  0.21051
## bath              17.330     13.665   1.268  0.20899
## bedrooms         -23.167     10.643  -2.177  0.03292 *
## agestandardized   -3.870      3.531  -1.096  0.27682
## garagesize        17.811     10.650   1.672  0.09898 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54.9 on 69 degrees of freedom
## Multiple R-squared:  0.2383, Adjusted R-squared:  0.1721
## F-statistic: 3.598 on 6 and 69 DF,  p-value: 0.003672
```

# b.

Create a second candidate model by using regsubsets over the entire data set. You can decide whether you prefer overall selection, forward selection, or backward selection, and you can decide which statistic you will use to determine the best model from the regsubsets process. Just conduct a justifiable model selection process and report the predictors in your final model.

```
regfit.full <- regsubsets(price ~ id + size + lot + bath + bedrooms + agestandardized + garagesi
ze + status + elem, data = housing, nvmax = 14)
reg.summary <- summary(regfit.full)
names(reg.summary)
```

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

```
reg.summary$bic
```

```
##  [1]  -1.7863476 -10.3081110 -15.2600323 -19.7220814 -22.4837938 -22.6105510
##  [7] -20.3906504 -17.6757586 -15.0643496 -11.6418283  -7.7234094  -3.6261882
## [13]   0.5383984   4.8059017
```

```
par(mfrow = c(2, 2))
plot(reg.summary$rss, xlab = "Number of Variables", ylab = "RSS")

plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq")
which.max(reg.summary$adjr2)
```

```
## [1] 9
```

```
points(9, reg.summary$adjr2[9], col = "red", cex = 2, pch = 20)

plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp")
which.min(reg.summary$cp)
```

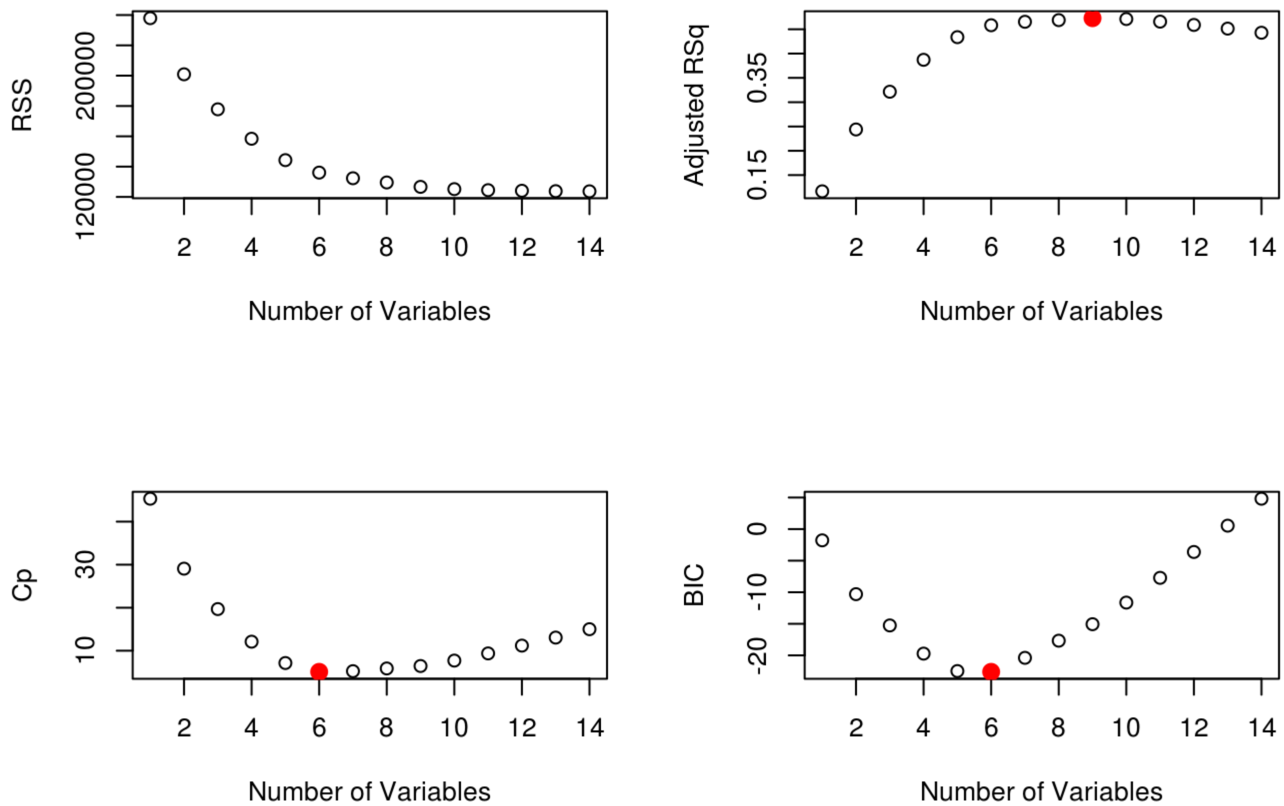
```
## [1] 6
```

```
points(6, reg.summary$cp[6], col = "red", cex = 2, pch = 20)

plot(reg.summary$bic, xlab = "Number of Variables", ylab = "BIC")
which.min(reg.summary$bic)
```

```
## [1] 6
```

```
points(6, reg.summary$bic[6], col = "red", cex = 2, pch = 20)
```




```
coef(regfit.full, 6)
```

```
## (Intercept)        size         lot    bedrooms   statussld   elemedison
##  127.556047   85.475310    9.995031  -16.004639  -34.907872    79.665432
##  elemharris
##   54.392695
```

The $C_p$ and BIC both reach their minimum at 6 variables. Adjusted $R^2$ reaches its minimum at 9 variables, but that method "is not as well motivated by statistical theory" as the ISLR textbook would put it. Therefore, the best model is chosen to be that with 6 variables. These variables are size, lot, bedrooms, statussld, elemedison and elemharris.

# C.

Create a training/test split of the data by which roughly half of the 76 observations are training data and half are test data.

```
set.seed(1)
train <- sample(c(TRUE, FALSE), nrow(housing), rep = TRUE)
test <- (!train)
test
```

```
##  [1] FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [13]  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE
## [25] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE
## [37]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [49]  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
## [61]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE
## [73] FALSE FALSE FALSE  TRUE
```

# d.

Now use regsubsets over only the training data to determine the number of predictors that should be in your final model. Then use regsubsets over the entire data set with the determined number of variables to determine your third candidate model.

```
set.seed(1)

regfit.best <- regsubsets(price ~ id + size + lot + bath + bedrooms + agestandardized + garagesi
ze + status + elem, data = housing[train, ], nvmax = 14)

test.mat <- model.matrix(price ~ id + size + lot + bath + bedrooms + agestandardized + garagesiz
e + status + elem, data = housing[test, ])

val.errors <- rep(NA, 14)
for(i in 1: 14){
  coefi = coef(regfit.best, id = i)
  pred = test.mat[, names(coefi)]%*%coefi
  val.errors[i] = mean((housing$price[test] - pred)^2)
}

val.errors
```

```
##  [1] 3291.975 3655.479 3713.480 3044.076 3366.699 2911.513 2484.384 2650.321
##  [9] 3520.744 3099.686 2942.913 2954.630 2939.537 2948.394
```

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

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

```
regfit.best2 <- regsubsets(price ~ id + size + lot + bath + bedrooms + agestandardized + garages
ize + status + elem, data = housing[train, ], nvmax = 7)

coef(regfit.best2, 7)
```

```
##     (Intercept)            size             lot        bedrooms agestandardized
##      116.816303       86.751449       13.789252      -20.382807        9.226705
##       statussld      elemedison      elemharris
##      -22.258492      119.587051       43.724871
```

After running regsubsets for best subset selecton with a training and test group, the best number of coefficients is determined by the for loop. This number becomes the number of coeffieients used to pick the best model when regsubsets is run over the entire data set. THat final model comes out to have the variables size, lot, bedrooms, agesandardized, statussld, elemdison and elemharris. The specific coefficients can be seen directly above, as the output of coef(regfit.best2, 7).

# e.

Next, use either Ridge Regression or Lasso Regression with the training data, and use cross validation via the cv.glmnet function to determine the best λ value. The model from this step with the best λ value will be your fourth candidate model.

```
#library(glmnet)
set.seed(1)

train.mat <- model.matrix(price ~ ., data = housing[train,])

test.mat <- model.matrix(price ~ ., data = housing[test,])

grid <- 10 ^ seq(4, -2, length = 100)

ridge.mod <- glmnet(train.mat, housing$price[train], alpha = 0, lambda = grid, thresh = 1e-12)

ridge.cv <- cv.glmnet(train.mat, housing$price[train], alpha = 0, lambda = grid, thresh = 1e-12)

ridge.bestlam <- ridge.cv$lambda.min

ridge.bestlam
```

```
## [1] 65.79332
```

```
ridge.pred <- predict(ridge.mod, s = ridge.bestlam, newx = test.mat)

mean((ridge.pred - housing$price[test])^2)
```

```
## [1] 1701.457
```

```
out = glmnet(train.mat <- model.matrix(price ~ ., data = housing), housing$price, alpha = 0)

ridge.final <- predict(out, type = "coefficients", s = ridge.bestlam)[1:17,]

ridge.final
```

```
##     (Intercept)     (Intercept)              id            size             lot
##    128.86766873      0.00000000     -0.02321362     27.43317017      4.28202556
##            bath        bedrooms       yearbuilt agestandardized      garagesize
##      6.10229778     -7.39977992      0.04601017      0.46002637      8.58346870
##       statuspen       statussld       elemcrest        elemedge      elemedison
##      2.16564369    -12.23638115     -0.82845567    -10.42567388     27.80419213
##      elemharris      elemparker
##     15.06341898    -15.59797103
```

We found that our best lambda from the ridge regression is 65.79332.

# f.

Finally, use either principal components regression or partial least squares regression for the training data. Use cross validation (see the class notes or the Chapter 6 Lab from the text) to help you determine the number of components in the model and briefly explain your choice. This model will be your 5th candidate model.

```
library(pls)
set.seed(1)

plsr.train <- plsr(price ~ ., data = housing, subset = train, scale = TRUE, validation = "CV")

summary(plsr.train)
```

```
## Data:    X dimension: 40 15
##  Y dimension: 40 1
## Fit method: kernelpls
## Number of components considered: 15
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           68.63    63.52    61.50    68.27    73.49    74.11    75.37
## adjCV        68.63    62.74    60.62    66.87    71.72    72.31    73.46
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       73.70    74.31    74.26     75.34     75.61     75.72     75.68
## adjCV    71.97    72.57    72.42     73.38     73.62     73.72     73.68
##         14 comps   15 comps
## CV     3.765e+13  3.785e+13
## adjCV  3.572e+13  3.591e+13
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X        14.87    32.50    39.57    46.85    55.32    60.62    69.19    76.00
## price    49.20    57.99    60.86    62.01    62.72    63.37    63.78    64.45
##        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X        80.39     85.24     90.31     94.35     97.68    100.00    101.08
## price    65.36     65.72     65.83     65.85     65.85     65.85     65.85
```

```
#validationplot(pcr.train, val.type = "MSEP")
#coefplot(pcr.train)
```

We see that our smallest adjusted cross-validation error occurs when M = 8 partial least squarres direcctions are used, so that is the model which is chosen.

```
set.seed(1)

plsr.full <- plsr(price ~ ., data = housing, scale = TRUE, validation = "CV")
```

The final model is fit over the entire data set, and the number of components chosen from the training cross validation is used. This is the modle that will have its MSE computed and compared to the other models.

# g.

For each of the five candidate models, calculate the mean square error for predicting the outcomes in the test data set that you created in part c. Based on this comparison, which model do you prefer for this situation?

```
error1 <- mean((housing$price-predict.lm(finalmod, data = housing[test]))^2)

error1
```

```
## [1] 2736.128
```

```
val.errors <- rep(NA, 14)

for(i in 1: 14){
  coefi = coef(regfit.full, id = i)
  pred = test.mat[, names(coefi)]%*%coefi
  val.errors[i] = mean((housing$price[test] - pred)^2)
}

val.errors
```

```
##  [1] 2076.025 2012.350 1763.017 1743.163 1489.681 1508.909 1437.044 1414.495
##  [9] 1347.138 1335.175 1369.972 1363.977 1397.658 1380.818
```

```
MSE.fin.regfit.full <- 1508.909

MSE.fin.regfit.full  #When tested against the test data set which is included in the training da
ta set for this model because the whole data set ws used to fit it, the MSE for the selected 6 v
ariable model is 1508.909.
```

```
## [1] 1508.909
```

```
val.errors <- rep(NA, 7)

for(i in 1: 7){
  coefi = coef(regfit.best2, id = i)
  pred = test.mat[, names(coefi)]%*%coefi
  val.errors[i] = mean((housing$price[test] - pred)^2)
}

val.errors
```

```
## [1] 3291.975 3655.479 3713.480 3044.076 3366.699 2911.513 2484.384
```

```
MSE.fin.regfit2 <- 2484.384

MSE.fin.regfit2  #When tested against the test data set which is included in the training data s
et for this model because the whole data set ws used to fit it, the MSE for the selected 7 varia
ble model is 2484.384.
```

```
## [1] 2484.384
```

```
ridge.pred <- predict(ridge.mod, s = ridge.final, newx = test.mat)

MSE.fin.ridge <- mean((ridge.pred - housing$price[test])^2)

MSE.fin.ridge #When tested against the test data set which is included in the training data set
 for this model because the whole data set ws used to fit it, the MSE for the selected $\lambda
 = 65.79332$ is 2484.384.
```

```
## [1] 2462.732
```

```
MSE.pls <- mean((housing$price[test]-predict(plsr.full, housing[test,], ncomp = 7))^2)

MSE.pls
```

```
## [1] 1377.501
```

```
#pred <- predict(plsr.train, test.mat, ncomp = 7)

errors <- matrix(c(error1, MSE.fin.regfit.full, MSE.fin.regfit2, MSE.fin.ridge, MSE.pls))

errors
```

```
##          [,1]
## [1,] 2736.128
## [2,] 1508.909
## [3,] 2484.384
## [4,] 2462.732
## [5,] 1377.501
```

```
rownames(errors) <- c("error1", "MSE.fin.regfit.full", "MSE.fin.regfit2", "MSE.fin.ridge", "MSE.
pls")

errors
```

```
##                         [,1]
## error1              2736.128
## MSE.fin.regfit.full 1508.909
## MSE.fin.regfit2     2484.384
## MSE.fin.ridge       2462.732
## MSE.pls             1377.501
```

Our final model, PLS Regression has the lowest MSE at 1,377.501. Indicating it is be our prefered model in predicting house prices.