hw1_code

Jiaqi Chen

2/13/2022

```
library(RNHANES)
library(tidyverse)
library(summarytools)
library(leaps)
library(corrplot)
library(glmnet)
library(caret)
library(plotmo)
library(ISLR)
library(pls)
```

(a)

Import and clean data

```
housing_test = read.csv("./housing_test.csv") %>%
  janitor::clean_names()

housing_training = read.csv("./housing_training.csv") %>%
  janitor::clean_names()
```

```
st_options(plain.ascii = FALSE,
           style = "rmarkdown",
           dfSummary.silent = TRUE,
           footnote = NA,
           subtitle.emphasis = FALSE)
```

Fit a linear model using least squares on the training data

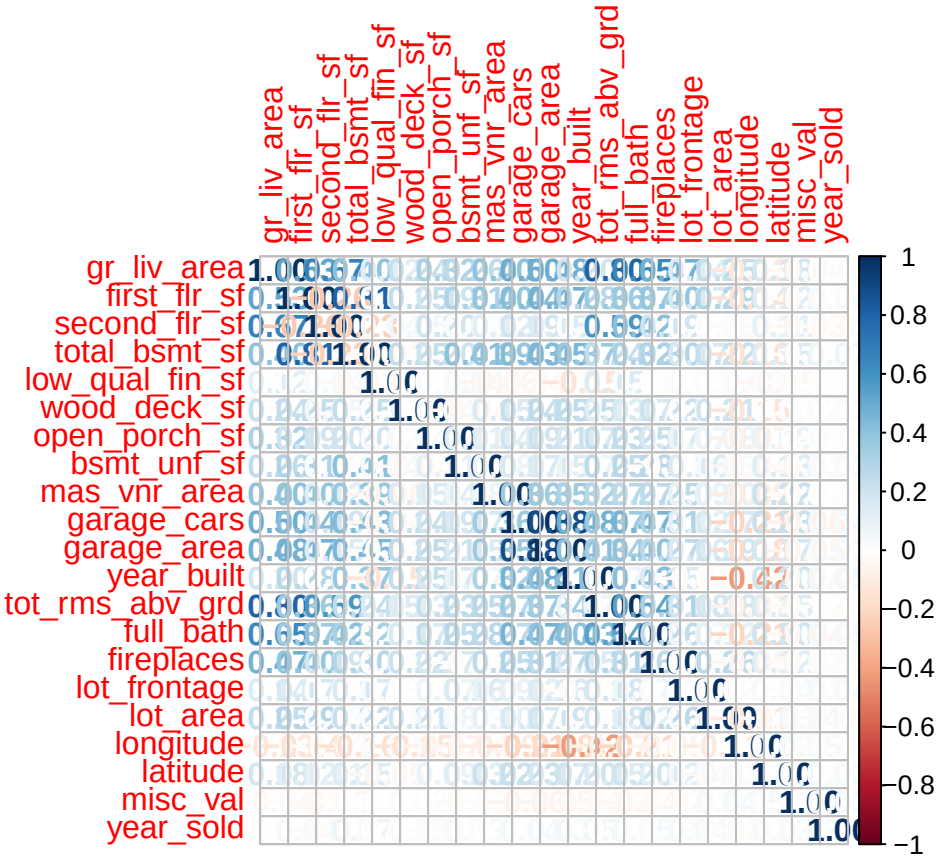
```
set.seed(1)

ctrl1 <- trainControl(method = "repeatedcv", number = 10, repeats = 5)

training2 <- model.matrix(sale_price ~., data = housing_training)[ ,-1]
test2 <- model.matrix(sale_price ~., data = housing_test)[ ,-1]

# matrix of predictors (glmnet uses input matrix)
x <- training2
# vector of response
y <- housing_training$sale_price
y2 <- housing_test$sale_price
```

```
lm.fit <- train(x, y, method = "lm", trControl = ctrl1)

pred1 <- predict(lm.fit, newdata = test2)
# test error
test_error1 = mean((pred1 - y2)^2)
```

The test error for least square model is 4.4728765×10^8.

Potential disadvantage of this model

```
cor_df=
  housing_training %>%
  dplyr::select(-sale_price & -overall_qual & -kitchen_qual & -fireplace_qu & -exter_qual)

cor_df=apply(cor_df, 2, as.numeric)

corrplot::corrplot(cor(cor_df), method = 'number')
```



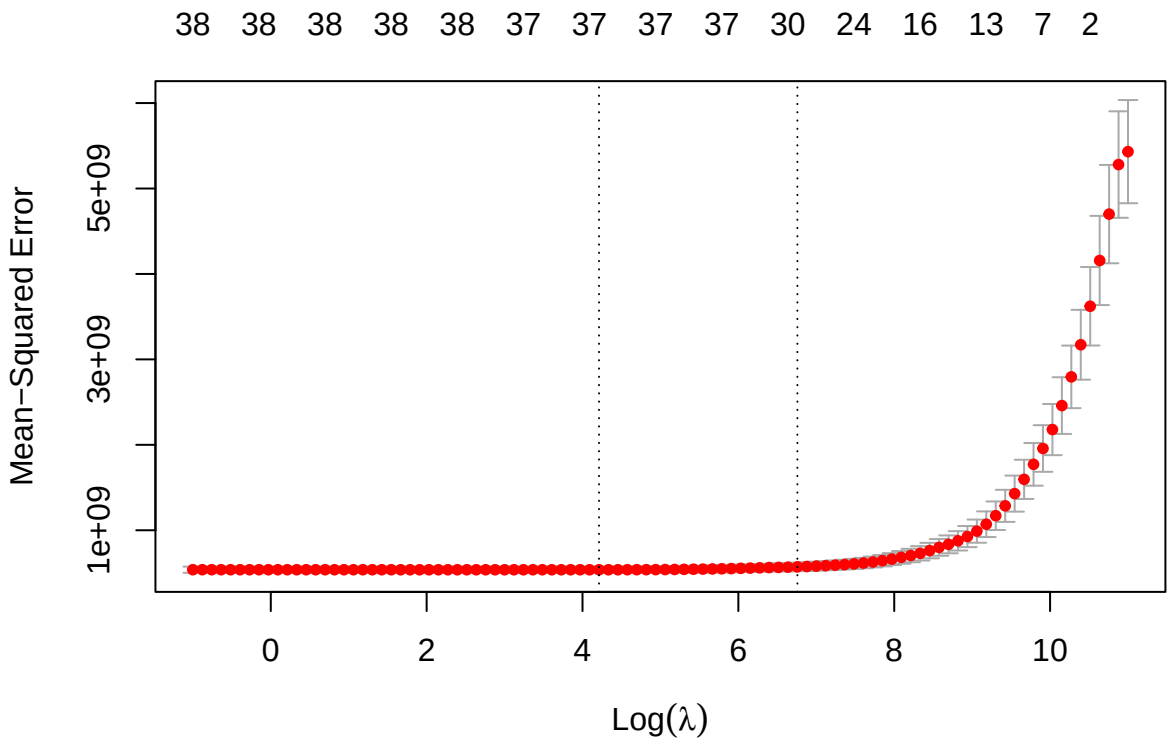
This linear model has multiple potential disadvantages: * There are too many factors to consider. * Some of these factors have correlations between each other, shown as above correlation plot. For example, the correlation between `Total rooms above grade (TotRms_AbvGrd)` and `Above grade (ground) living area square feet (Gr_Liv_Area)` is as high as 0.8.

(b)

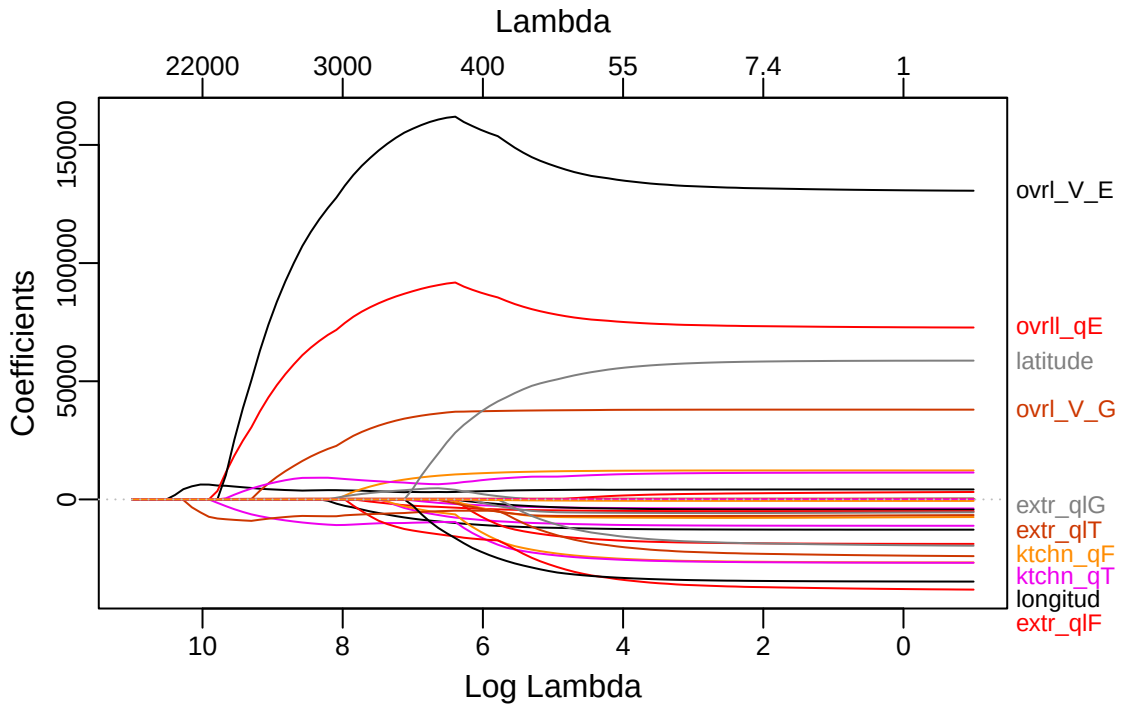
```
cv.lasso <- cv.glmnet(x, y,
                      alpha = 1,
                      lambda = exp(seq(11, -1, length = 100)))

cv.lasso$lambda.1se
```

```
## [1] 860.5535
```

```
plot(cv.lasso)
```




```
plot_glmnet(cv.lasso$glmnet.fit)
```

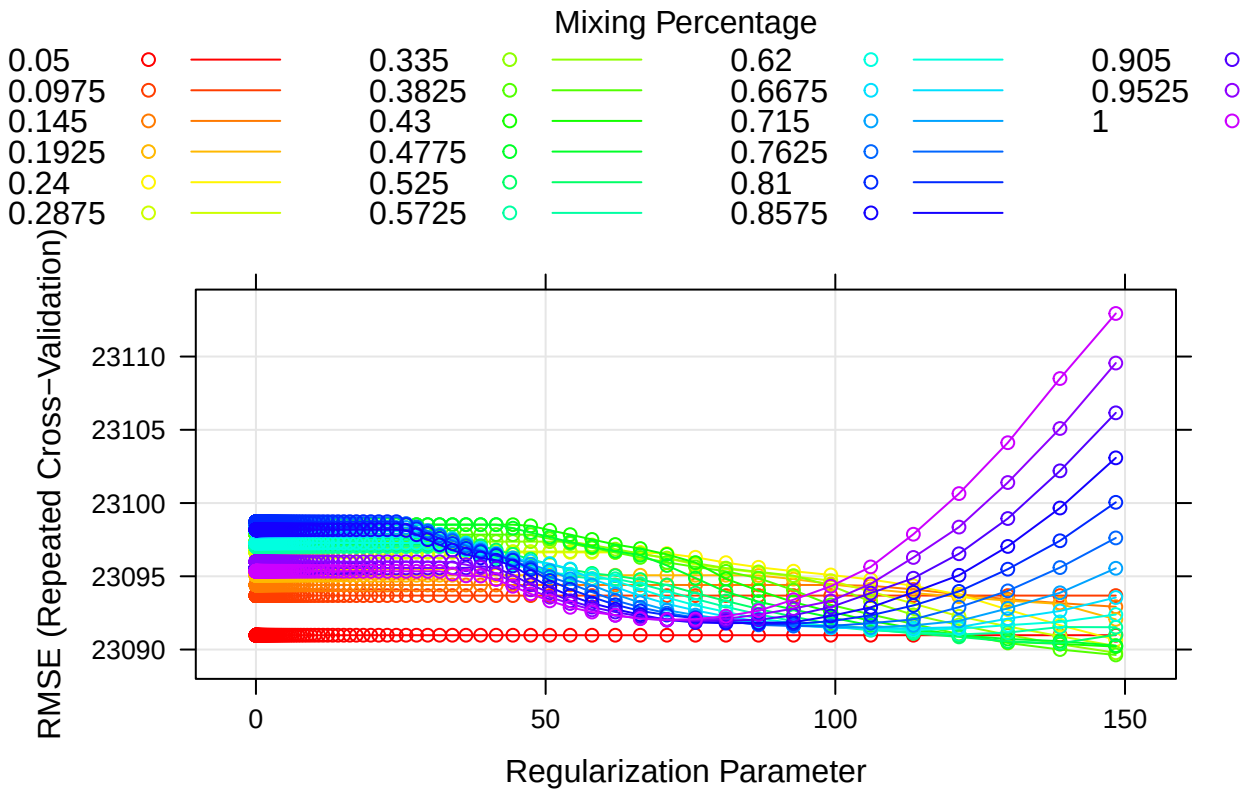



```
pred <- predict(cv.lasso, newx = test2, s = "lambda.1se", type = "response")

test_error2 = mean((pred - y2)^2)
```

There are 29 predictors included in this model. The test error is 4.2129638×10^8.

(c)

```
enet.fit <- train(x, y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0.05, 1, length = 21),
                                         lambda = exp(seq(5, -5, length = 150))),
                  trControl = ctrl1)

parameter2 = enet.fit$bestTune

myCol<- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
                    superpose.line = list(col = myCol))

plot(enet.fit, par.settings = myPar)
```




```
coef(enet.fit$finalModel, enet.fit$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -4.881045e+06
## gr_liv_area                 6.104683e+01
## first_flr_sf                5.012330e+00
## second_flr_sf               4.017683e+00
## total_bsmt_sf               3.527050e+01
## low_qual_fin_sf            -3.696515e+01
## wood_deck_sf                1.178829e+01
## open_porch_sf               1.576615e+01
## bsmt_unf_sf                -2.086491e+01
## mas_vnr_area                1.104465e+01
## garage_cars                 4.081968e+03
## garage_area                 8.290829e+00
## year_built                  3.223307e+02
## tot_rms_abv_grd            -3.573240e+03
## full_bath                  -3.805021e+03
## overall_qualAverage        -4.918851e+03
## overall_qualBelow_Average  -1.252490e+04
## overall_qualExcellent       7.547622e+04
## overall_qualFair           -1.094325e+04
## overall_qualGood            1.211570e+04
## overall_qualVery_Excellent  1.357339e+05
## overall_qualVery_Good       3.786985e+04
## kitchen_qualFair           -2.472437e+04
## kitchen_qualGood           -1.709156e+04
## kitchen_qualTypical        -2.519037e+04
```

```
## fireplaces                  1.062604e+04
## fireplace_quFair           -7.748265e+03
## fireplace_quGood            .
## fireplace_quNo_Fireplace    1.522719e+03
## fireplace_quPoor           -5.706632e+03
## fireplace_quTypical        -7.023415e+03
## exter_qualFair             -3.336157e+04
## exter_qualGood             -1.507474e+04
## exter_qualTypical          -1.954704e+04
## lot_frontage                9.978185e+01
## lot_area                    6.041661e-01
## longitude                  -3.336222e+04
## latitude                    5.569194e+04
## misc_val                    8.373167e-01
## year_sold                  -5.655100e+02
```

```
enet.pred <- predict(enet.fit, newdata = test2)
# test error
test_error3 = mean((enet.pred - y2)^2)
```

Elastic net mixing parameter alpha is 0.3825, lambda is 148.413159102577. Test error is 4.3983643×10^8.

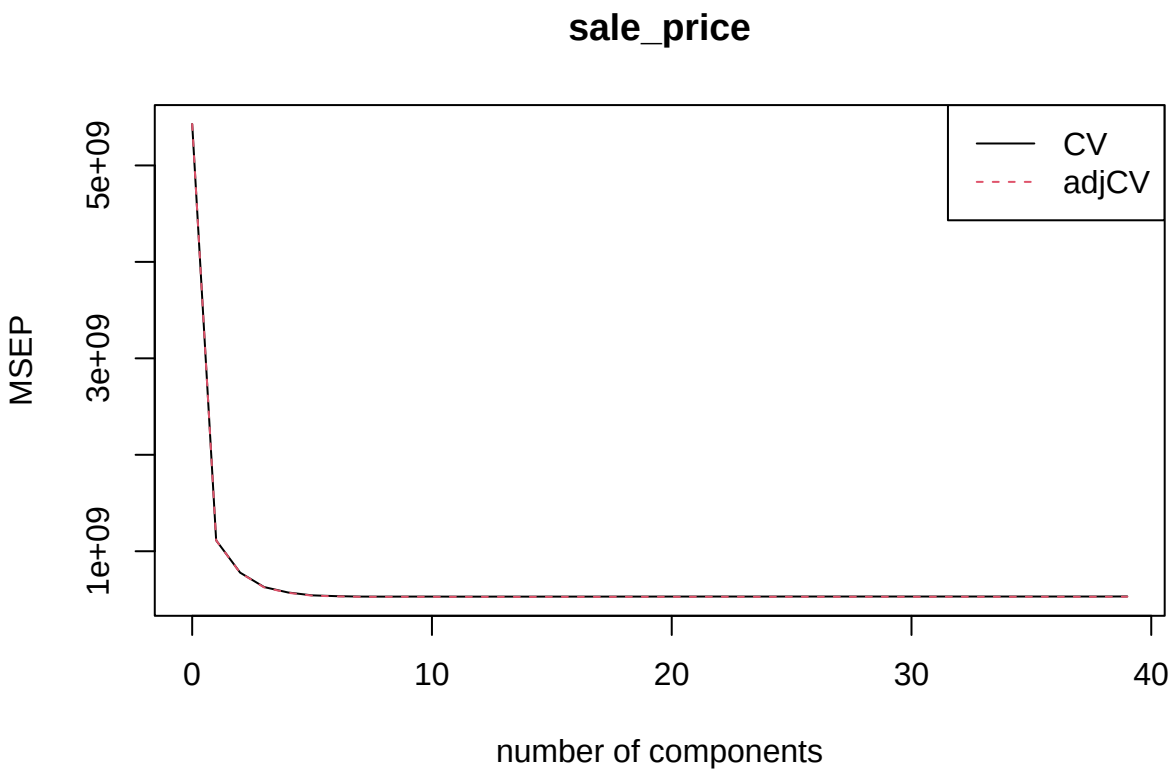
(d)

```
set.seed(2)

pls.mod <- plsr(sale_price~., data = housing_training, scale = TRUE, validation = "CV")

summary(pls.mod)
```

```
## Data:    X dimension: 1440 39
##  Y dimension: 1440 1
## Fit method: kernelpls
## Number of components considered: 39
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           73685    33357    27895    25075    23920    23294    23114
## adjCV        73685    33354    27863    25006    23858    23233    23061
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       23020    23007    23021     23022     23011     23011     23013
## adjCV    22968    22954    22965     22964     22953     22952     22953
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        23010     23016     23014     23014     23019     23022     23024
## adjCV     22951     22956     22954     22954     22959     22961     22964
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        23027     23031     23029     23030     23030     23032     23033
## adjCV     22967     22970     22968     22969     22969     22971     22972
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV        23034     23034     23034     23034     23034     23034     23034
## adjCV     22973     22973     22972     22973     22973     22973     22973
##        35 comps  36 comps  37 comps  38 comps  39 comps
## CV        23034     23034     23034     23034     23052
```

```
## adjCV     22973     22973     22973     22973     22990
##
## TRAINING: % variance explained
##             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X             20.02    25.93    29.67    33.59    37.01    40.03    42.49
## sale_price    79.73    86.35    89.36    90.37    90.87    90.99    91.06
##             8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X             45.53    47.97     50.15     52.01     53.69     55.35     56.86
## sale_price    91.08    91.10     91.13     91.15     91.15     91.16     91.16
##             15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## X              58.64     60.01     62.18     63.87     65.26     67.10
## sale_price     91.16     91.16     91.16     91.16     91.16     91.16
##             21 comps  22 comps  23 comps  24 comps  25 comps  26 comps
## X              68.44     70.12     71.72     73.35     75.20     77.27
## sale_price     91.16     91.16     91.16     91.16     91.16     91.16
##             27 comps  28 comps  29 comps  30 comps  31 comps  32 comps
## X              78.97     80.10     81.83     83.55     84.39     86.34
## sale_price     91.16     91.16     91.16     91.16     91.16     91.16
##             33 comps  34 comps  35 comps  36 comps  37 comps  38 comps
## X              88.63     90.79     92.79     95.45     97.49    100.00
## sale_price     91.16     91.16     91.16     91.16     91.16     91.16
##             39 comps
## X             100.67
## sale_price     91.16
```

```
validationplot(pls.mod, val.type="MSEP", legendpos = "topright")
```




```
cv.mse <- RMSEP(pls.mod)
ncomp.cv <- which.min(cv.mse$val[1,,])-1
ncomp.cv
```

```
## 8 comps
##       8
```

```
predy2.pls <- predict(pls.mod, newdata = test2,
ncomp = ncomp.cv)
# test MSE
test_error4 = mean((y2 - predy2.pls)^2)
```

Test error is 4.4021794×10^8. There are 8 components included in my model.

(e)

Combining all calculations above, test error for linear model is 4.4728765×10^8; test error for lasso model is 4.2129638×10^8; test error for elastic net model is 4.3983643×10^8; test error for partial least squares model is 4.4021794×10^8. The test error of lasso model is the smallest, so I will choose lasso model for predicting the response.