HW4

Hanao Li

October 3, 2019

Question 1

```
if(!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
p_load(MASS, pls, mctest, glmnet, knitr, kableExtra)

fit <- lm(medv ~ crim + zn + indus + nox + rm + age + tax, Boston)
sumsq <- anova(fit)$"Sum Sq"
VIF <- (1 - sumsq[-length(sumsq)] / sum(sumsq))^(-1)
VIF
```

```
## [1] 1.177552 1.090760 1.063519 1.000671 1.381436 1.001737 1.009707
```

```
mean(VIF)
```

```
## [1] 1.103626
```

```
omcdiag(Boston[, -14], Boston[, 14])
```

```
## 
## Call:
## omcdiag(x = Boston[, -14], y = Boston[, 14])
## 
## 
## Overall Multicollinearity Diagnostics
## 
##                        MC Results detection
## Determinant |X'X|:         0.0001         1
## Farrar Chi-Square:      4486.4729         1
## Red Indicator:             0.4432         0
## Sum of Lambda Inverse:    45.1229         0
## Theil's Method:           -1.2643         0
## Condition Number:         87.3183         1
## 
## 1 --> COLLINEARITY is detected by the test
## 0 --> COLLINEARITY is not detected by the test
```

```
# Since VIF < 10, there is no multicollinearity. But we have VIF bar is larger than 1, so there
 is little multicollinearity.
# From the diagnostics, we could see that condition number method showing there is collinearity
 while some other methods do not. So we can conclude that there exists little but not serious mu
lticollinearity.

# Remedial measures: we could use ridge regression or we can use PCR to avoid multicollinearity.
```

Question 2

```
variables <- c("crim", "zn", "indus", "nox", "rm", "age", "tax")
varmatrix <- as.matrix(Boston[variables])
depmatrix <- as.matrix(Boston["medv"])

summary(fit)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + indus + nox + rm + age + tax,
##     data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.625  -3.161  -0.833   2.089  41.042
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -19.615259   3.221482  -6.089 2.27e-09 ***
## crim         -0.132538   0.038482  -3.444 0.000621 ***
## zn            0.022103   0.014823   1.491 0.136547
## indus        -0.014980   0.072282  -0.207 0.835909
## nox           0.010643   4.230468   0.003 0.997994
## rm            7.606508   0.418424  18.179  < 2e-16 ***
## age          -0.023198   0.014893  -1.558 0.119964
## tax          -0.009006   0.002662  -3.384 0.000772 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.989 on 498 degrees of freedom
## Multiple R-squared:  0.5818, Adjusted R-squared:  0.576
## F-statistic: 98.99 on 7 and 498 DF,  p-value: < 2.2e-16
```

```
pred_linear <- predict(fit, newx=varmatrix)
mse_linear <- mean((depmatrix - pred_linear) ^ 2)
mse_linear
```

```
## [1] 35.30022
```

```
fit_pcr <- pcr(medv ~ crim + zn + indus + nox + rm + age + tax, data = Boston)
summary(fit_pcr)
```

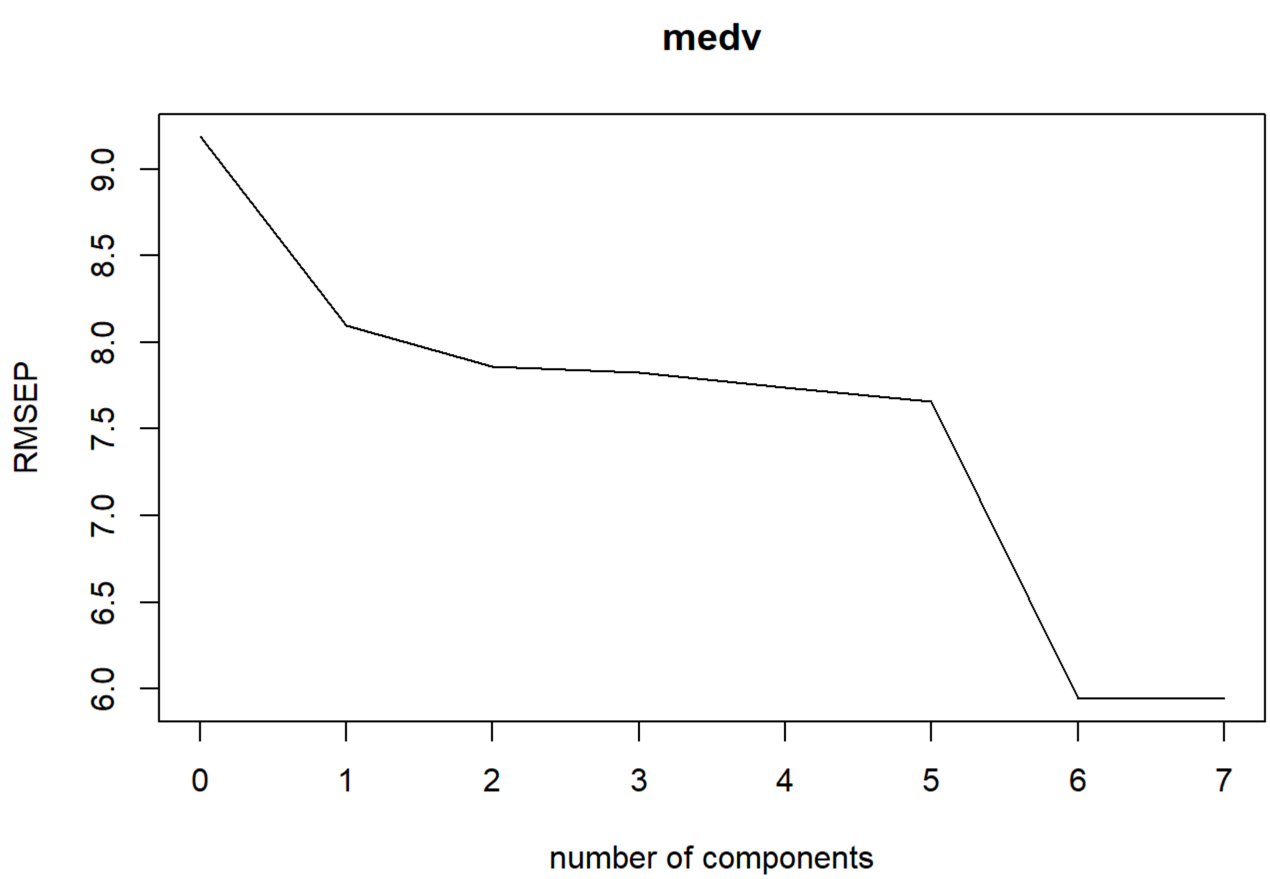
```
## Data:    X dimension: 506 7
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 7
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       96.17    98.89    99.78    99.95   100.00   100.00   100.00
## medv    22.29    26.77    27.40    29.06    30.52    58.18    58.18
```

```
fit_pcr$coefficients
```

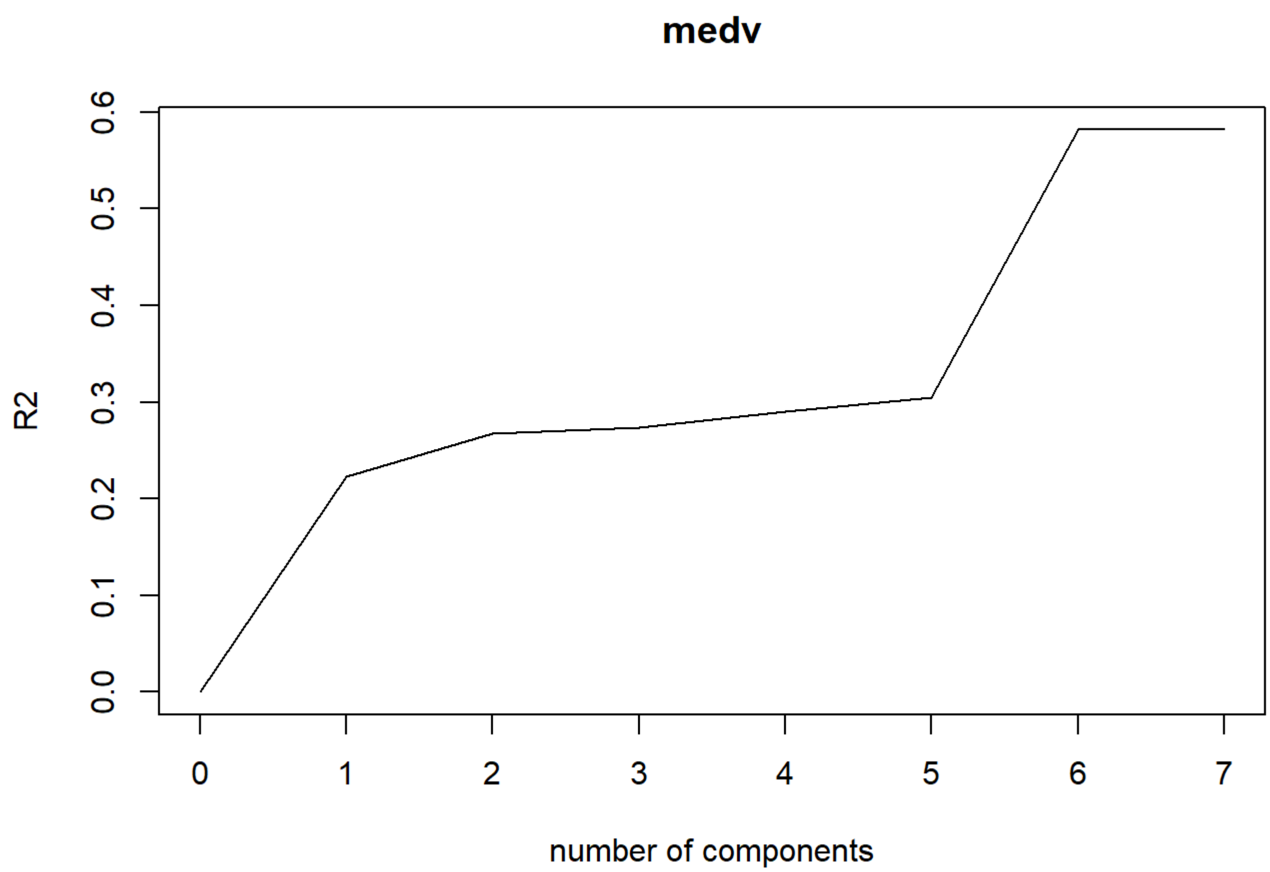
```
## , , 1 comps
##
##                medv
## crim  -7.602935e-04
## zn     1.150884e-03
## indus -7.542334e-04
## nox   -1.184132e-05
## rm     3.133808e-05
## age   -2.213204e-03
## tax   -2.547918e-02
##
## , , 2 comps
##
##                medv
## crim  -0.0016853343
## zn     0.0446402402
## indus -0.0068021172
## nox   -0.0001366212
## rm     0.0003397777
## age   -0.0540646790
## tax   -0.0188037200
##
## , , 3 comps
##
##                medv
## crim  -0.0004213788
## zn     0.0788779816
## indus -0.0073933769
## nox   -0.0001059403
## rm     0.0006096457
## age   -0.0254242782
## tax   -0.0197649097
##
## , , 4 comps
##
##                medv
## crim  -1.700791e-01
## zn     8.153525e-02
## indus  2.692739e-03
## nox   -8.996815e-05
## rm     1.489102e-03
## age   -2.076327e-02
## tax   -1.528469e-02
##
## , , 5 comps
##
##               medv
## crim  -0.186159901
## zn     0.062658498
## indus -0.274628965
## nox   -0.001638117
## rm     0.011031329
## age   -0.003089130
```

```
## tax   -0.008971014
##
## , , 6 comps
##
##              medv
## crim  -0.132536071
## zn     0.022100922
## indus -0.014931710
## nox    0.002233208
## rm     7.606510824
## age   -0.023185138
## tax   -0.009004800
##
## , , 7 comps
##
##              medv
## crim  -0.132538486
## zn     0.022103317
## indus -0.014979611
## nox    0.010642684
## rm     7.606508171
## age   -0.023197907
## tax   -0.009006005
```

```
validationplot(fit_pcr)
```



```
validationplot(fit_pcr, "R2")
```




```
pred_pcr5 <- predict(fit_pcr, ncomp=5, newx=varmatrix)
mse_pcr5 <- mean((depmatrix - as.numeric(pred_pcr5)) ^ 2)
mse_pcr5
```

```
## [1] 58.65863
```

```
pred_pcr6 <- predict(fit_pcr, ncomp=6, newx=varmatrix)
mse_pcr6 <- mean((depmatrix - as.numeric(pred_pcr6)) ^ 2)
mse_pcr6
```

```
## [1] 35.30022
```

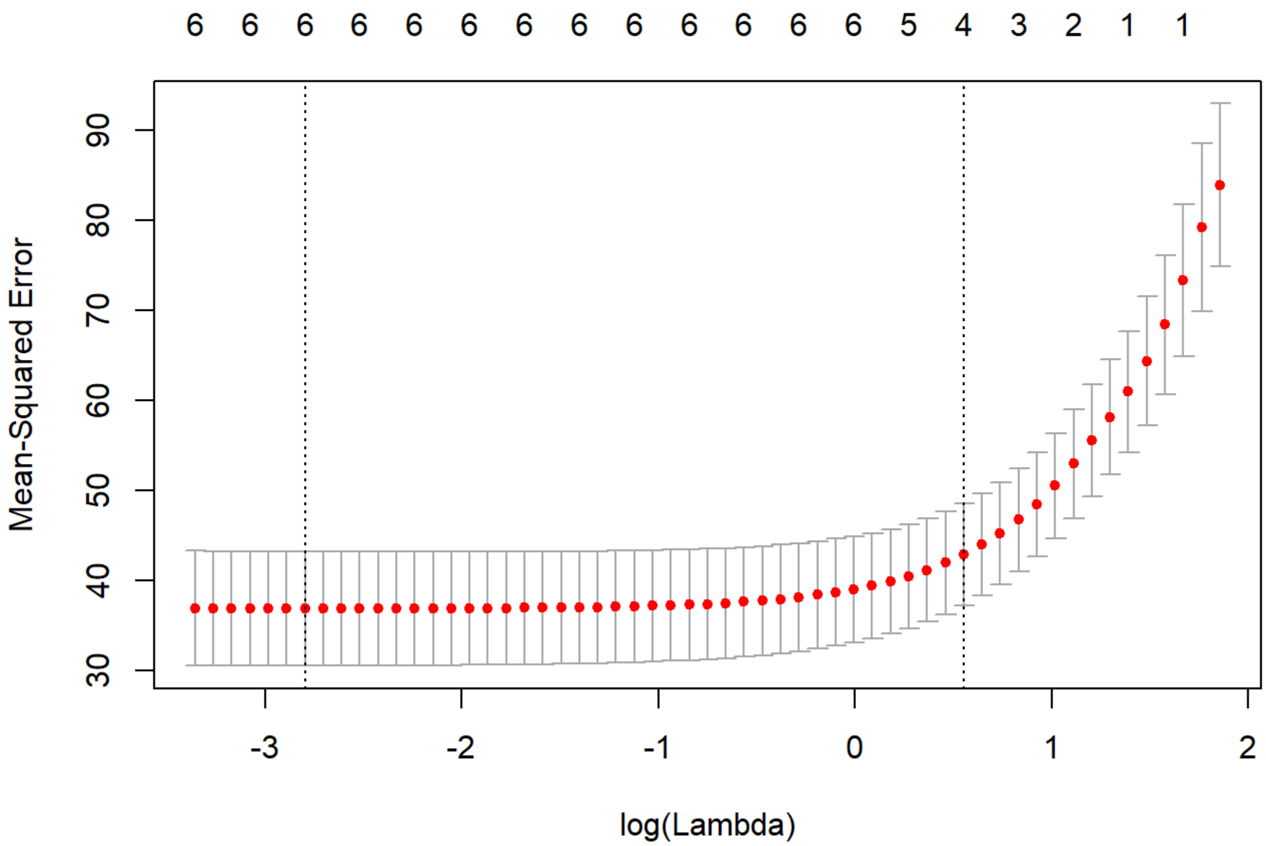
```
pred_pcr7 <- predict(fit_pcr, ncomp=7, newx=varmatrix)
mse_pcr7 <- mean((depmatrix - as.numeric(pred_pcr7)) ^ 2)
mse_pcr7
```

```
## [1] 35.30022
```

```
# From these two methods, we could see that when PCR has 6 or more comp, it will have the same M
SE as the linear regression.
```

Question 3

```
fit_lasso <- cv.glmnet(varmatrix, depmatrix)
plot(fit_lasso)
```




```
lambda <- fit_lasso$lambda.min
lambda
```

```
## [1] 0.06098586
```

```
coef <- coef(fit_lasso, "lambda.min")
coef
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -19.346619579
## crim         -0.128613180
## zn            0.020751561
## indus        -0.016321199
## nox           .
## rm            7.552663663
## age          -0.022484475
## tax          -0.008901037
```

```
pred_lasso <- predict(fit_lasso, newx=varmatrix, s="lambda.min")
mse_lasso <- mean((depmatrix - pred_lasso) ^ 2)
mse_lasso
```

```
## [1] 35.3082
```

```
fit_forward <- lm(medv ~ 1, data=Boston)
fit_backward <- lm(medv ~ crim + zn + indus + nox + rm +age + tax, data=Boston)

forward <- stepAIC(fit_forward, direction="forward", scope=list(upper=fit_backward, lower=fit_fo
rward), trace=F)
forward
```

```
##
## Call:
## lm(formula = medv ~ rm + tax + crim + age + zn, data = Boston)
##
## Coefficients:
## (Intercept)           rm          tax         crim          age
##  -19.713176     7.625253    -0.009323    -0.131852    -0.024121
##          zn
##    0.022947
```

```
backward <- stepAIC(fit_backward, direction="backward", scope=list(upper=fit_backward, lower=fit
_forward), trace=F)
backward
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + rm + age + tax, data = Boston)
##
## Coefficients:
## (Intercept)         crim           zn           rm          age
##  -19.713176    -0.131852     0.022947     7.625253    -0.024121
##         tax
##   -0.009323
```

```
pred_forward <- predict(forward, newx=varmatrix)
mse_forward <- mean((depmatrix - pred_forward) ^ 2)
mse_forward
```

```
## [1] 35.30361
```

```
pred_backward <- predict(backward, newx=varmatrix)
mse_backward <- mean((depmatrix - pred_backward) ^ 2)
mse_backward
```

```
## [1] 35.30361
```

```
# From these two methods, we could see that their MSE are very close, so their performance of pr
edicting are pretty much the same. But since stepwise procedure used less variables, it is bette
r since Lasso used more variables and might lead to overfitting.
```

Table

```
df <- data.frame(Cat = c("Performance n>>p", "Performance Multicollinearity", "Unbiased Estimato
rs", "Model Selection", "Simplicity"),
                 OLS = linebreak(c("3", "3", "1", "2", "1")),
                 Ridge = linebreak(c("2", "2", "3", "3", "2")),
                 Lasso = linebreak(c("2", "2", "3", "1", "1")),
                 Elastic_Net = linebreak(c("1", "1", "3", "2" , "3")))
kable(df, col.names = c("", "OLS", "Ridge", "Lasso", "Elastic Net"), escape = F, caption = "Tabl
e") %>%
  kable_styling(latex_options = "hold_position")
```

Table

	OLS	**Ridge**	**Lasso**	**Elastic Net**
Performance n>>p	3	2	2	1
Performance Multicollinearity	3	2	2	1
Unbiased Estimators	1	3	3	3
Model Selection	2	3	1	2
Simplicity	1	2	1	3

Performance n>>p

Performance Multicollinearity

Unibased Estimators

Model Selection

Simplicity