homework3

Yi-Ling Chen

May 17, 2021

```
## Loading required package: Matrix

## Loaded glmnet 4.1-1

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: lattice

##
## Attaching package: 'MASS'

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

Task2.1

```r
df <- read.table("prostate_data.txt", header=T)
sub_df <- df %>% filter(train==TRUE)
test_df <- df %>% filter(train==FALSE)

train_X = as.matrix(sub_df[1:8])
train_y = sub_df$lpsa
test_X = as.matrix(test_df[1:8])
test_y = test_df$lpsa
```

```r
model1 <- lm(lpsa ~ lcavol, data=sub_df)
model2 <- lm(lpsa ~ lcp, data=sub_df)
model3 <- lm(lpsa ~ lcavol + lcp, data=sub_df)
MS1 <- summary(model1)$coef
MS2 <- summary(model2)$coef
MS3 <- summary(model3)$coef
print("Model1:")
```

```
## [1] "Model1:"
```

```r
MS1
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 1.5163048 0.14772483 10.264387 3.123765e-15
## lcavol      0.7126351 0.08199036  8.691694 1.733134e-12
```

```r
print("Model2:")
```

```
## [1] "Model2:"
```

```r
MS2
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2.5427013 0.13121183 19.378598 1.041939e-28
## lcp         0.4218252 0.09327981  4.522149 2.659180e-05
```

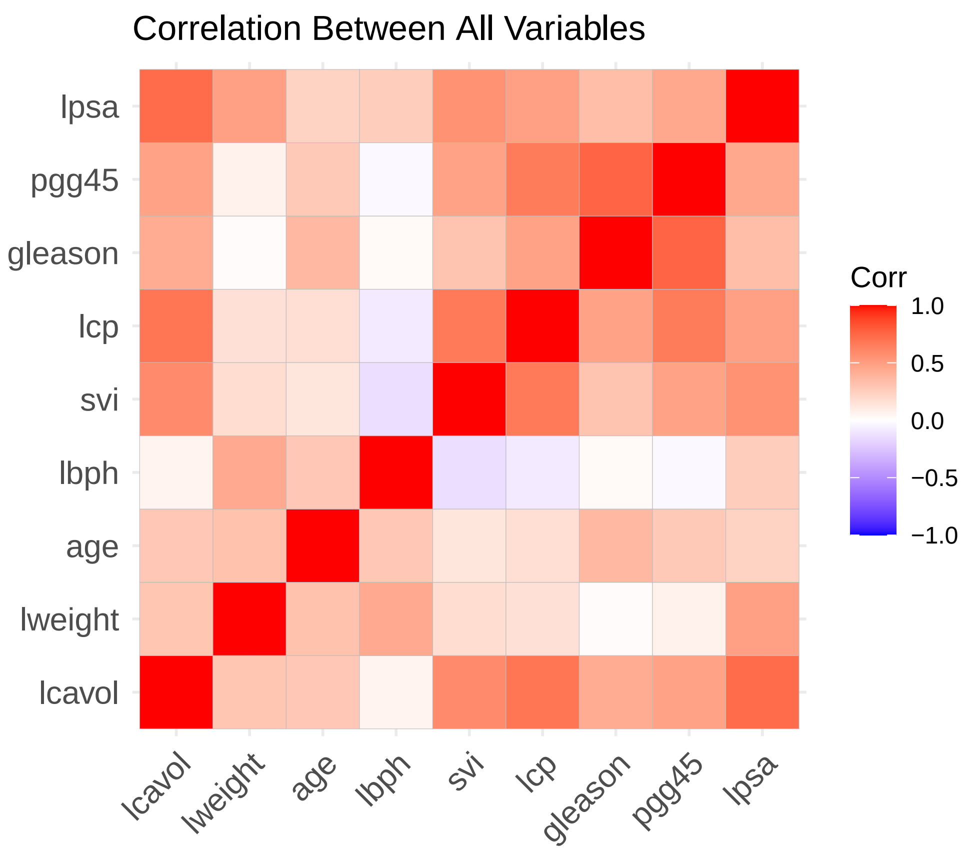
```r
print("Model3:")
```

```
## [1] "Model3:"
```

```r
MS3
```

```
##                 Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  1.47905119  0.1947748  7.5936468 1.678372e-10
## lcavol       0.73609361  0.1143882  6.4350510 1.802808e-08
## lcp         -0.03007034  0.1014736 -0.2963367 7.679325e-01
```

```r
ggcorrplot(cor(sub_df[1:9]), title = "Correlation Between All Variables")
```



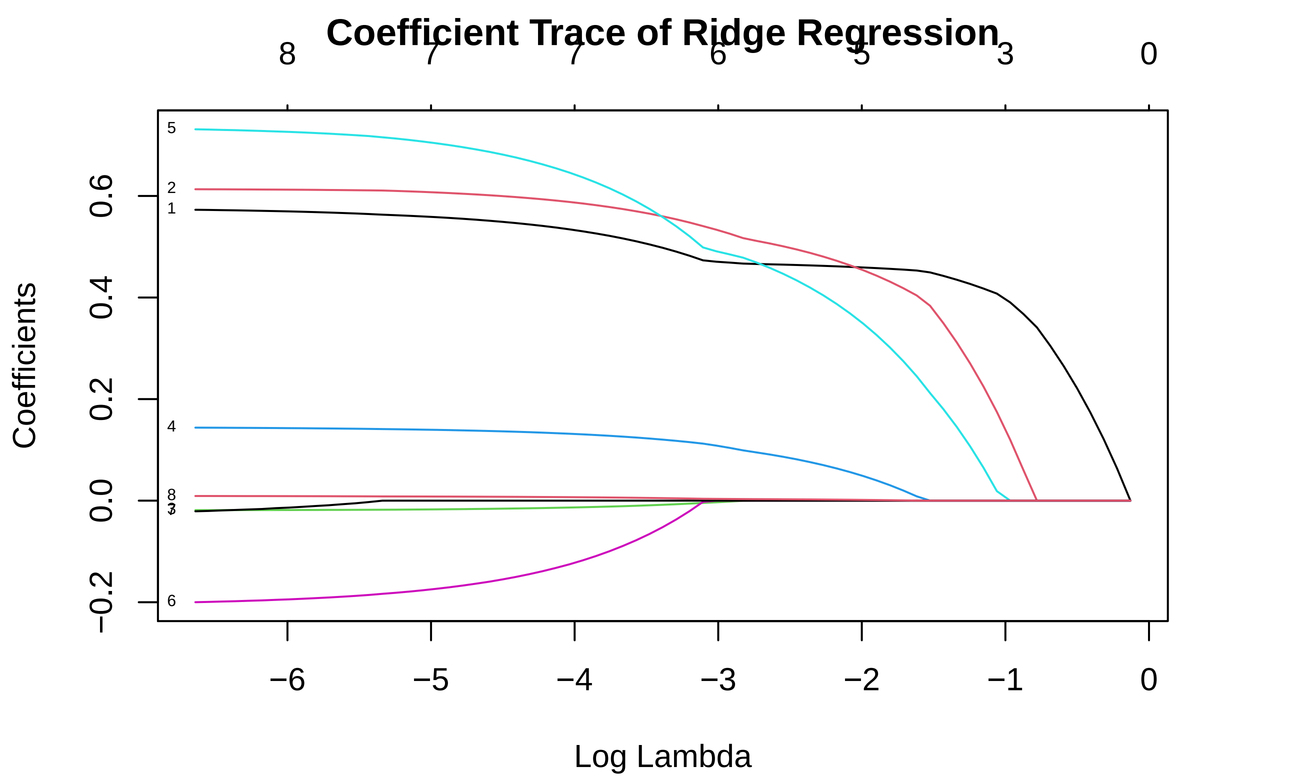
Since predictors lcavol(log cancer volume) and lcp(log of capsular penetration) are highly correlated. The more highly correlated the independent variables are, the more difficult to determine how much variation in Y each X is responsible for. Therefore, the standard errors for both variables become larger.

Task2.2

```r
# least-square with all variables
least_square <- lm(lpsa~., data = df, subset = train)

# ridge regression
# tracing lambda estimation
ridge_trace <- glmnet(train_X, train_y, family = "gaussian", alpah=0)
# choosing best lambda with cross-validation
ridge_cv <- cv.glmnet(train_X, train_y, family = "gaussian", alpah=0)

plot(ridge_trace, label = TRUE,xvar = "lambda", main="Coefficient Trace of Ridge Regression")
```




```r
coef(least_square)
```

```
##  (Intercept)       lcavol      lweight          age         lbph          svi
##  0.429170133  0.576543185  0.614020004 -0.019001022  0.144848082  0.737208645
##          lcp      gleason        pgg45    trainTRUE
## -0.206324227 -0.029502884  0.009465162           NA
```

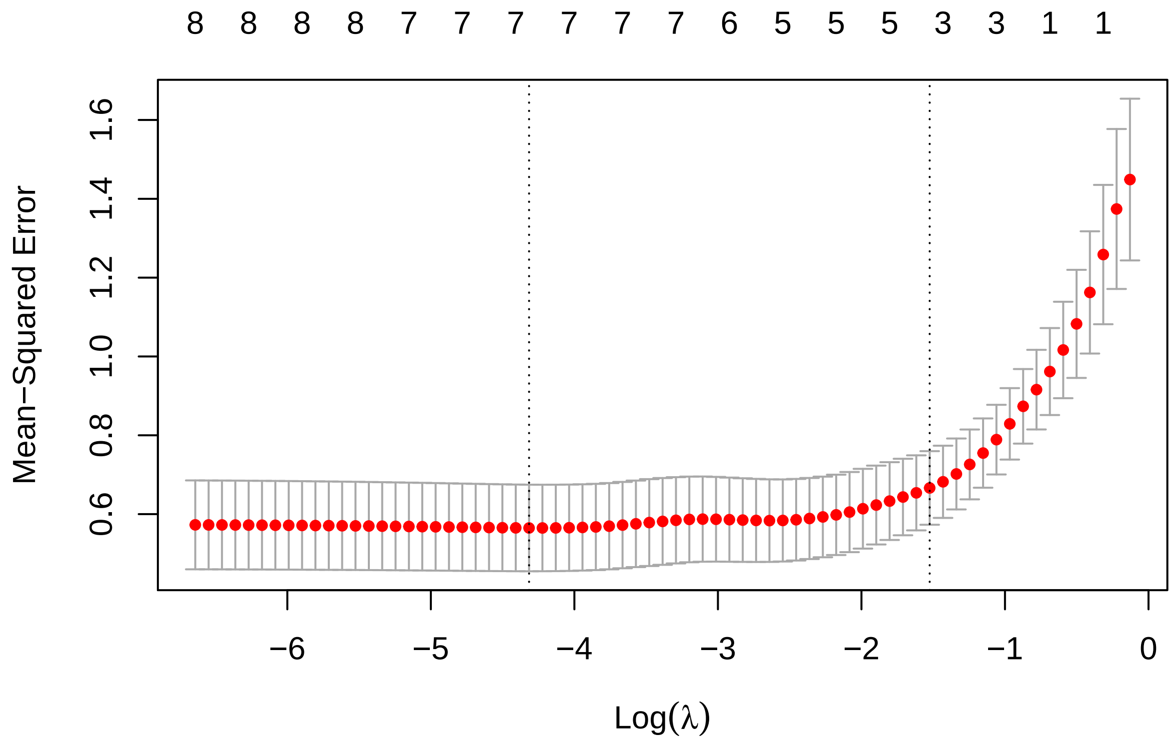
When lambda is zero, we can see the coefficients at the left side for each variables are the same as least-square model.

```r
print(ridge_cv)
```

```
##
## Call:  cv.glmnet(x = train_X, y = train_y, family = "gaussian", alpah = 0)
##
```

```
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.01336    46  0.5649 0.10983       7
## 1se 0.21771    16  0.6664 0.09337       3
```

```r
plot(ridge_cv)
```



According to lambda.min and lambda.1se, we can choose lambda.min for our model, which means when lmbda is 0.01609, the cross-validation result has the lowest error. However, to the future unseen data, it's better to have a more rgularized model because we don't know the distribution of the unseen data. Therefore, choosing the result of lambda.1se for the model.

```r
# RMSE
best_ridge <- glmnet(train_X, train_y, family = "gaussian", alpah=0, lambda = ridge_cv$lambda.1se)
sqrt(mean((predict(best_ridge, test_X)-test_y)^2))
```

```
## [1] 0.6953873
```

```r
min_ridge <- glmnet(train_X, train_y, family = "gaussian", alpah=0, lambda = ridge_cv$lambda.min)
sqrt(mean((predict(min_ridge, test_X)-test_y)^2))
```

```
## [1] 0.7023585
```

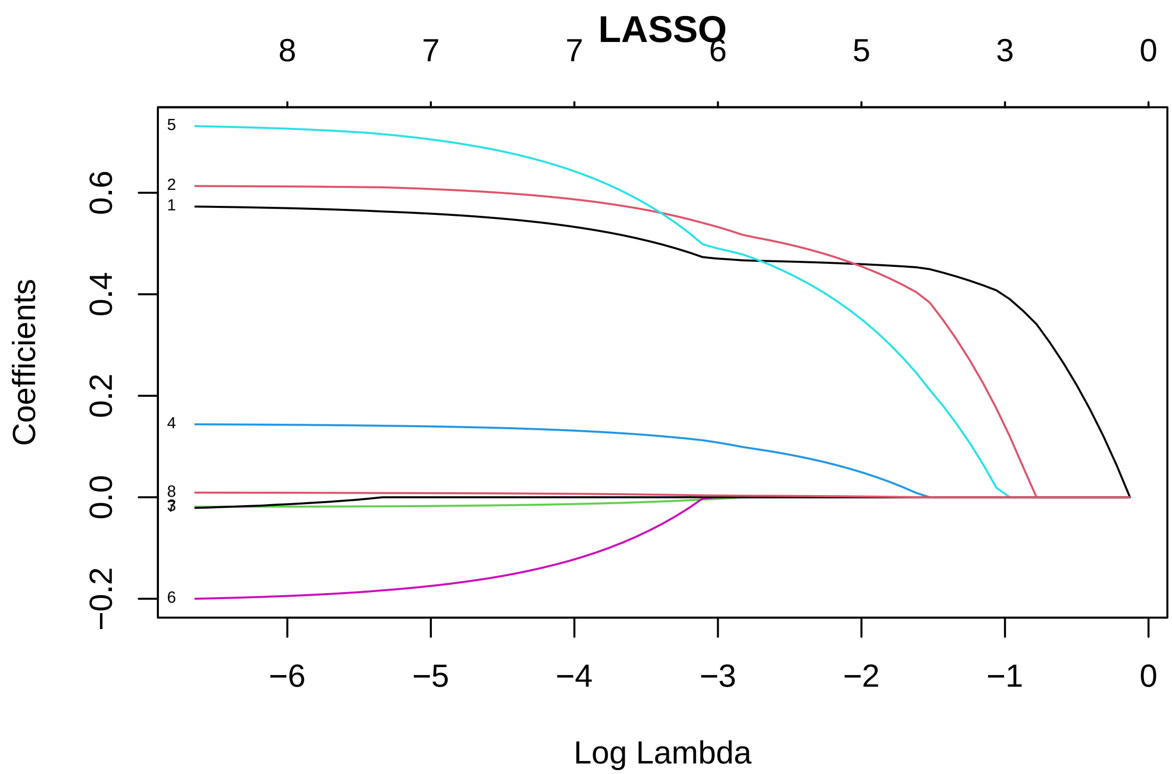
The first value is the lambda from lambda.1se, which represents a more general model with less model complexity, the rmse is 0.6729088. The second one is the model with lambda.min, the rmse is 0.6993649. The result shows that the model lambda has a better performance.

Task2.3

```r
# glmnet default alpha =1
lasso = cv.glmnet(train_X, train_y,family = "gaussian", alpah=1)
lasso_trace = glmnet(train_X, train_y,family = "gaussian", alpah=1)
print(lasso)
```

```
##
## Call:  cv.glmnet(x = train_X, y = train_y, family = "gaussian", alpah = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure     SE Nonzero
## min 0.01466    45  0.5733 0.1199       7
## 1se 0.21771    16  0.6901 0.1016       3
```

```r
plot(lasso_trace, label = TRUE,xvar = "lambda", main="LASSO")
```




```r
lasso_1se = glmnet(train_X, train_y,family = "gaussian", alpah=1, lambda = lasso$lambda.1se)
print("The rmse estimate of lasso:")
```

Comparison of the rmse performance

```
## [1] "The rmse estimate of lasso:"
```

```r
sqrt(mean((predict(lasso_1se, test_X)-test_y)^2))
```

```
## [1] 0.6953873
```

```r
print("The rmse estimate of ridge regression:")
```

```
## [1] "The rmse estimate of ridge regression:"
```

```r
sqrt(mean((predict(best_ridge, test_X)-test_y)^2))
```

```
## [1] 0.6953873
```

```r
coef(lasso_1se)
```

LASSO coefficients

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## (Intercept) 0.4228791
## lcavol      0.4494666
## lweight     0.3837892
## age         .
## lbph        .
## svi         0.2118731
## lcp         .
## gleason     .
## pgg45       .
```

```r
coef(best_ridge)
```

Ridge coefficients

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## (Intercept) 0.4228791
## lcavol      0.4494666
## lweight     0.3837892
## age         .
## lbph        .
## svi         0.2118731
## lcp         .
## gleason     .
## pgg45       .
```

After penalizing, both models has 5 predictors, all the predictors have the similar estimation. And LASSO has a lightly better result on rmse estimation.

Task2.4

Comparison of three models

```r
# Search for the best elasticnet(combination of alpha and lambda)
for(i in 0:10){
  assign(paste("fit", i, sep=""), cv.glmnet(train_X, train_y, type.measure="mse", alpha=i/10,family="ga
}
```

```r
# Show the best one
fit9
```

```
##
## Call:  cv.glmnet(x = train_X, y = train_y, type.measure = "mse", alpha = i/10,      family = "gaussia
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure     SE Nonzero
## min 0.01352    47   0.618 0.1276       7
## 1se 0.26548    15   0.733 0.1711       3
```

```r
alpha = 0.9
lambda = 0.18298
L1 = alpha*lambda
L2 = ((1-alpha)/2)*lambda
L1
```

```
## [1] 0.164682
```

```r
L2
```

```
## [1] 0.009149
```

The elasticnet with alpha=0.3, lambda.1se=0.31413 has the best estimation of MSE on training data, and we can calculate the lambda1 and lambda2 according to the loss function. So lambda1(L1)=0.164682, lambda2(L2)=0.009149.

```r
cv_ridge <- cv.glmnet(train_X, train_y,family = "gaussian", alpah=0)
cv_lasso <- cv.glmnet(train_X, train_y,family = "gaussian", alpah=1)
cv_elnet <- cv.glmnet(train_X, train_y,family = "gaussian", alpah=0.9)

cv_ridge
```

CV-RMSE: Ridge Regression

```
##
## Call:  cv.glmnet(x = train_X, y = train_y, family = "gaussian", alpah = 0)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.00527    56  0.5639 0.07711       7
## 1se 0.13673    21  0.6409 0.06600       5
```

```r
cv_lasso
```

CV-RMSE: LASSO

```
##
## Call:  cv.glmnet(x = train_X, y = train_y, family = "gaussian", alpah = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.00189    67  0.5531 0.09769       8
```

```
## 1se 0.18074    18  0.6505 0.08842       5
```

```
cv_elnet
```

CV-RMSE: Elasticnet

```
##
## Call:  cv.glmnet(x = train_X, y = train_y, family = "gaussian", alpah = 0.9)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min 0.0101    49  0.6151 0.1246       7
## 1se 0.2177    16  0.7281 0.1608       3
```

```
# ridge
sqrt(0.6462)
```

```
## [1] 0.8038657
```

```
# lasso
sqrt(0.6430)
```

```
## [1] 0.8018728
```

```
# elasticnet
sqrt(0.7039)
```

```
## [1] 0.8389875
```

According to the cv-rmse, LASSO is the best and then ridge regression, the worst one is elasticnet.