chapter 6

Linear Model Selection and Regularization

22/10/2024

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.3.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3
```

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

```
## Loaded glmnet 4.1-8
```

```
library(leaps)
```

#chp 6(9)

a. Split the data set into a training set and a test set.

```
set.seed(123)
tr=sample(nrow(College),nrow(College)*.70)
train1=College[tr,]
test1=College[-tr,]
dim(test1)
```

```
## [1] 234  18
```

```
dim(train1)
```

```
## [1] 543  18
```

INTERPRETATION #This shows that the training set (train1) has 543 rows and 18 columns

b. Fit a linear model using least squares on the training set, and report the test error obtained.

```
model_linear = lm(Apps ~ ., data = train1)
summary(model_linear)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3097.8  -455.8   -46.5   343.8  6452.5
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -310.17331  481.30075  -0.644 0.519566
## PrivateYes  -681.96465  164.08211  -4.156 3.78e-05 ***
## Accept         1.22130    0.05921  20.626  < 2e-16 ***
## Enroll         0.08046    0.21794   0.369 0.712155
## Top10perc     49.33503    6.18296   7.979 9.31e-15 ***
## Top25perc    -16.11744    5.02717  -3.206 0.001428 **
## F.Undergrad    0.02284    0.03985   0.573 0.566831
## P.Undergrad    0.03541    0.03529   1.003 0.316139
## Outstate      -0.05446    0.02132  -2.555 0.010910 *
## Room.Board     0.18967    0.05275   3.596 0.000354 ***
## Books          0.21366    0.28099   0.760 0.447381
## Personal      -0.03685    0.07279  -0.506 0.612876
## PhD           -6.00401    5.34580  -1.123 0.261897
## Terminal      -5.01712    5.77787  -0.868 0.385609
## S.F.Ratio     -2.18927   14.83898  -0.148 0.882766
## perc.alumni   -8.01836    4.67330  -1.716 0.086792 .
## Expend         0.07614    0.01340   5.681 2.23e-08 ***
## Grad.Rate     10.63461    3.38228   3.144 0.001760 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 992.3 on 525 degrees of freedom
## Multiple R-squared:  0.9175, Adjusted R-squared:  0.9148
## F-statistic: 343.2 on 17 and 525 DF,  p-value: < 2.2e-16
```

INTERPRETATION #It explains about 91.75% of the variability, which is high and indicates a strong model fit #992.3 indicates the average deviation of observed values from the model's predictions #Variables such as Enroll, F.Undergrad, P.Undergrad, Books, Personal, PhD, Terminal, and S.F.Ratio do not show statistically significant effects (high p-values). #Being a private college is associated with a decrease in applications

```
pred =predict(model_linear, test1)
error = mean((pred - test1$Apps)^2)
sqrt(error)
```

```
## [1] 1317.134
```

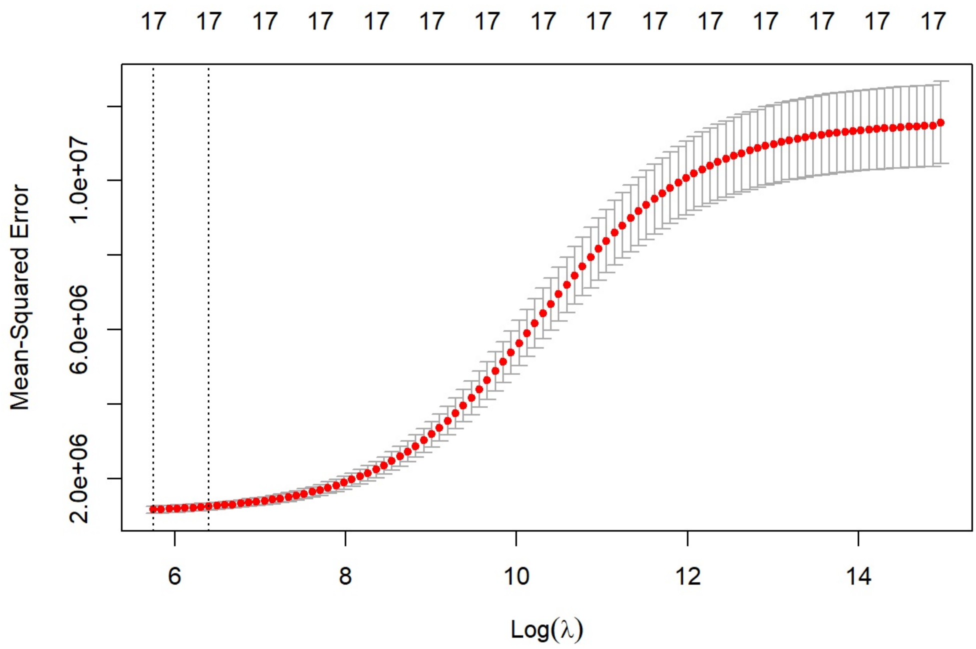
INTERPRETATION #an RMSE of 1317.134 means that, on average, the model's predictions deviate from the actual Apps #1317 applications might be high or low depending on the average and range of applications in the dataset

```
names(train1)
```

```
##  [1] "Private"     "Apps"        "Accept"      "Enroll"      "Top10perc"
##  [6] "Top25perc"   "F.Undergrad" "P.Undergrad" "Outstate"    "Room.Board"
## [11] "Books"       "Personal"    "PhD"         "Terminal"    "S.F.Ratio"
## [16] "perc.alumni" "Expend"      "Grad.Rate"
```

c. Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
xtrain = model.matrix(train1$Apps ~ ., data = train1[,-2])
fit2 <- cv.glmnet(xtrain, train1$Apps, alpha = 0)
plot(fit2)
```



INTERPRETATION #The x-axis typically represents the values of the regularization parameter (log(lambda)). #The y-axis shows the cross-validated mean squared error (MSE) for each lambda. # BELOW six the line indicates the lambda min and the second line indicates the 1se(above 6)

```
fit2$lambda.min
```

```
## [1] 314.2524
```

```
coef(fit2,fit2$lambda.min)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -1.108384e+03
## (Intercept)  .
## PrivateYes  -6.344812e+02
## Accept       7.799468e-01
## Enroll       7.025220e-01
## Top10perc    2.888869e+01
## Top25perc   -2.480330e+00
## F.Undergrad  1.000467e-01
## P.Undergrad  7.509154e-03
## Outstate    -1.062903e-02
## Room.Board   2.160271e-01
## Books        2.845690e-01
## Personal    -6.359640e-02
## PhD         -2.824524e+00
## Terminal    -5.297962e+00
## S.F.Ratio   -2.570072e+00
## perc.alumni -1.259409e+01
## Expend       7.644084e-02
## Grad.Rate    1.090898e+01
```

interpretation #Positive contributions come from Room.Board, Expend, and Grad.Rate (10.91) #Ridge regression shrinks smaller coefficients like P.Undergrad and Outstate toward zero #a higher percentage of students in the top 10% of their high school class is associated with more applications, whereas the top 25% has a slight negative influence.

```
xtest=model.matrix(test1$Apps ~ ., data = test1[,-2])
p <- predict(fit2, xtest, s = fit2$lambda.min)
mse_ridge <- mean((p - test1$Apps)^2)
sqrt(mse_ridge)
```

```
## [1] 1725.214
```

interpretation #This RMSE value is higher than the linear regression model's RMSE (1317.134) #ridge regression would be preferred if it stabilizes coefficients effectively without much compromise on prediction accuracy #an RMSE of 1725.214 meaning that the model's predictions differ from the actual number of applications (Apps) by about 1725 applications on average.

d. Fit a lasso model on the training set, with λ chosen by cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
xtrain = model.matrix(train1$Apps ~ ., data = train1[,-2])
fit2 <- cv.glmnet(xtrain, train1$Apps, alpha = 1)
```

```
xtest=model.matrix(test1$Apps ~ ., data = test1[,-2])
p <- predict(fit2, xtest, s = fit2$lambda.min)
mse_lasso <- mean((p - test1$Apps)^2)
sqrt(mse_lasso)
```

```
## [1] 1315.667
```

interpretation #An RMSE of 1315.667 means that the Lasso model's predictions deviate from the actual number of applications (Apps) by about 1316 applications on average. #Lasso Regression imposes regularization by shrinking some coefficients to zero #Lasso performs as well as or slightly better than linear regression

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.3.3
```

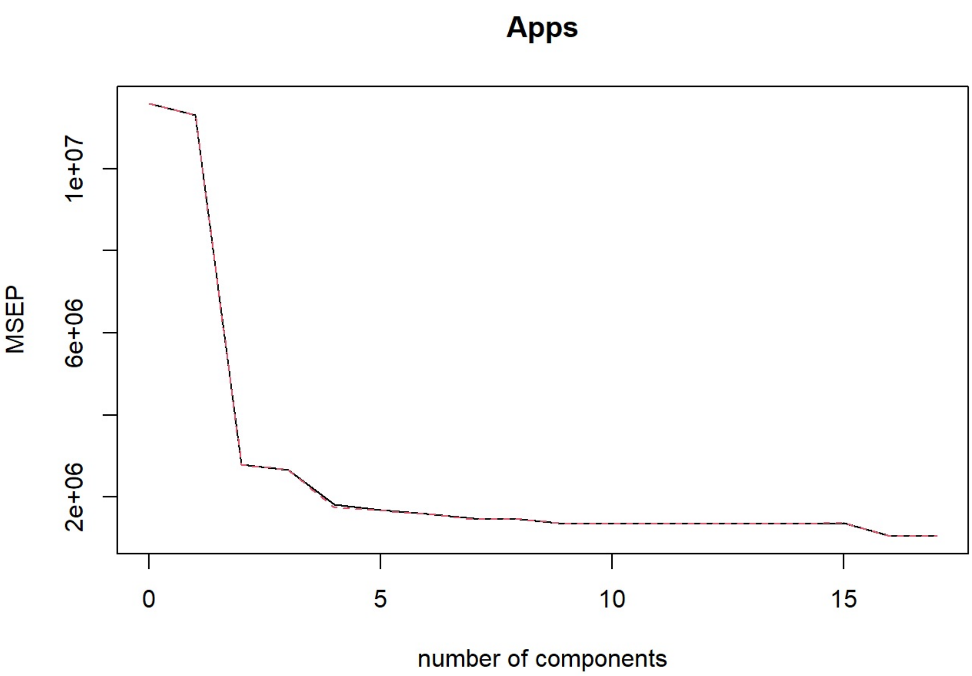
```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

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

e. Fit a PCR model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
fit4 <- pcr(Apps ~ ., data = train1, scale = TRUE, validation = "CV")
validationplot(fit4, val.type = "MSEP")
```




```
summary(fit4)
```

```
## Data:    X dimension: 543 17
##  Y dimension: 543 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3402     3361     1675     1634     1347     1299     1264
## adjCV         3402     3361     1673     1632     1325     1289     1262
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1218     1210     1163      1163      1164      1166      1165
## adjCV     1207     1208     1161      1162      1163      1164      1163
##        14 comps  15 comps  16 comps  17 comps
## CV         1166      1171      1027      1028
## adjCV      1164      1170      1024      1025
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      31.797    57.68    64.58     70.2    75.49    80.41    84.00    87.43
## Apps    3.037    76.20    77.49     85.1    86.15    86.83    87.92    88.01
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.62     93.05     95.12     96.96     98.04     98.89     99.42
## Apps    88.85     88.87     88.94     88.98     89.01     89.01     89.03
##       16 comps  17 comps
## X        99.83    100.00
## Apps     91.68     91.75
```

interpretation #The RMSEP stabilizes as more components are added, with the largest improvement occurring up to 5 or 6 components #the RMSEP decreasing substantially up to around 5–6 components, where it reaches around 1264 and then stabilizes #the RMSEP is 3361, which shows some improvement over the intercept-only model.

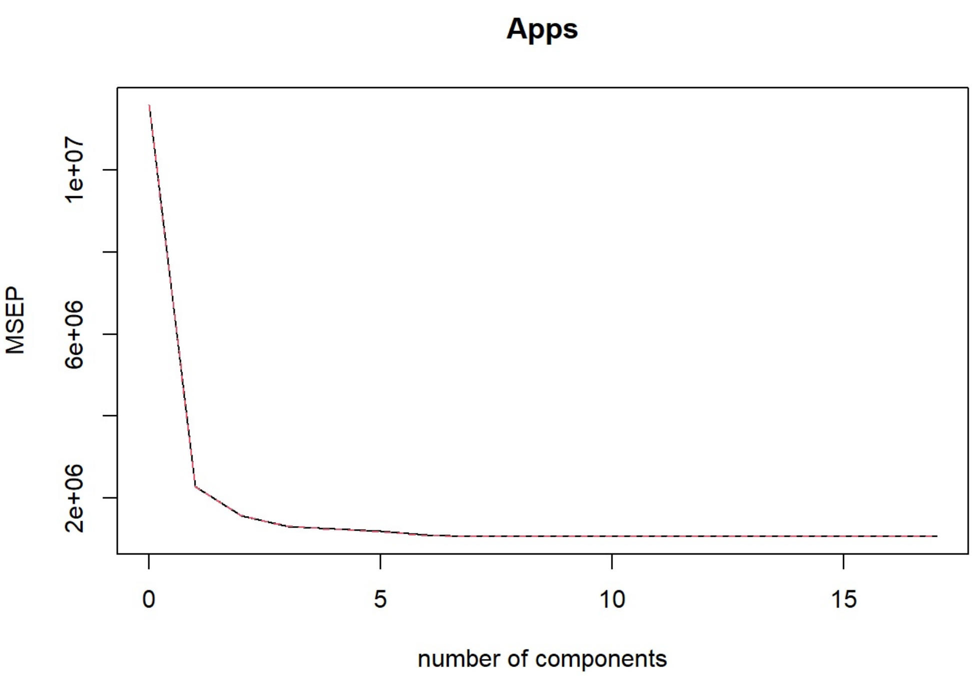
```
p <- predict(fit4, test1, ncomp = 9)
mse_pcr <- mean((p - test1$Apps)^2)
sqrt(mse_pcr)
```

```
## [1] 1993.632
```

INTERPRETATION #RMSEP of 1993.632 indicates that, on average, the model’s predictions differ from the actual observed values #Exploring interactions or nonlinear relationships among variables.

f. Fit a PLS model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
fit5 <- plsr(Apps ~ ., data = train1, scale = TRUE, validation = "CV")
validationplot(fit5, val.type = "MSEP")
```




```
summary(fit5)
```

```
## Data:    X dimension: 543 17
##  Y dimension: 543 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3402     1515     1253     1150     1120     1095     1052
## adjCV         3402     1512     1256     1148     1117     1090     1048
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1038     1037     1038      1038      1037      1038      1038
## adjCV     1035     1034     1035      1035      1034      1035      1035
##        14 comps  15 comps  16 comps  17 comps
## CV         1038      1038      1038      1038
## adjCV      1035      1035      1035      1035
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       26.09    41.97    63.14    67.44    71.36    74.05    77.72    80.98
## Apps    80.83    86.94    89.29    90.10    90.94    91.65    91.71    91.73
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       83.77     86.46     89.83     91.07     93.08     95.14     97.06
## Apps    91.73     91.74     91.74     91.74     91.75     91.75     91.75
##       16 comps  17 comps
## X        99.09    100.00
## Apps     91.75     91.75
```

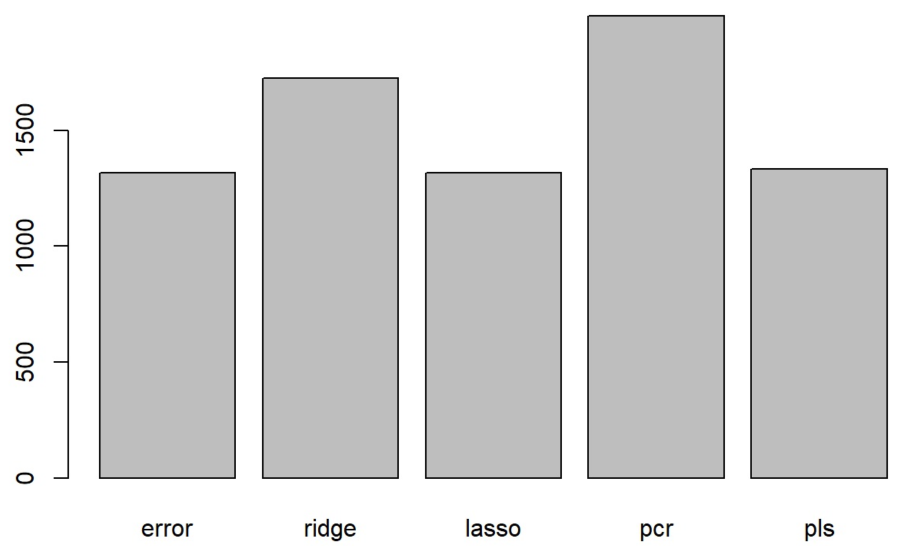
INTERPRETATION #The RMSEP stabilizes around 7 to 9 components, where the RMSEP hovers just below 1040 #The RMSEP values indicate a consistent decrease in prediction error as more components are added, suggesting that the model becomes more accurate

```
p <- predict(fit5, test1, ncomp = 8)
mse_pls <- mean((p - test1$Apps)^2)
sqrt(mse_pls)
```

```
## [1] 1332.112
```

INTERPRETATION #an RMSEP of 1332.112 indicates that the model's predictions deviate from actual values by approximately 1332 applications on average #if the actual application values are much lower than this RMSEP (g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

```
mse=c(error=sqrt(error),ridge=sqrt(mse_ridge),lasso=sqrt(mse_lasso),pcr=sqrt(mse_pcr),pls=sqrt(mse_pls))
barplot(mse)
```



INTERPRETATION #Shorter bars indicate better predictive performance, as they signify lower RMSE values #Taller bars indicate poorer predictive performance, as they signify higher RMSE values

#chp 6(11) 11. We will now try to predict per capita crime rate in the Boston data set.

```
library(MASS)
?Boston
```

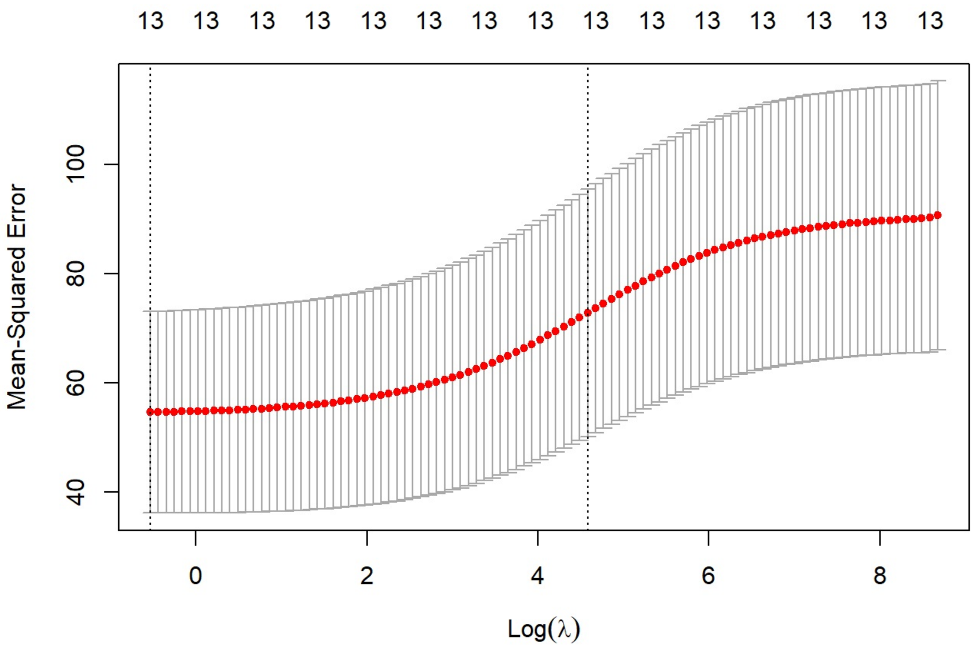
```
## starting httpd help server ... done
```

a. Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

```
set.seed(123)
tr=sample(nrow(Boston),nrow(Boston)*.70)
train2=Boston[tr,]
test2=Boston[-tr,]
```

```
library(pls)
```

```
xtrain = model.matrix(train2$crim~., data = train2[,-1])
fit22 <- cv.glmnet(xtrain, train2$crim, alpha = 0)
plot(fit22)
```



INTERPRETATION #The vertical line in the plot marks this point, providing guidance on which lambda value to select for further analysis or predictions. #As lambda increases (moving to the right), the model becomes more regularized

```
xtest=model.matrix(test2$crim ~ ., data = test2[,-1])
p1 <- predict(fit22, xtest, s = fit22$lambda.min)
mse_ridg <- mean((p - test2$crim)^2)
```

```
## Warning in p - test2$crim: longer object length is not a multiple of shorter
## object length
```

```
mse_ridg
```

```
## [1] 27410091
```

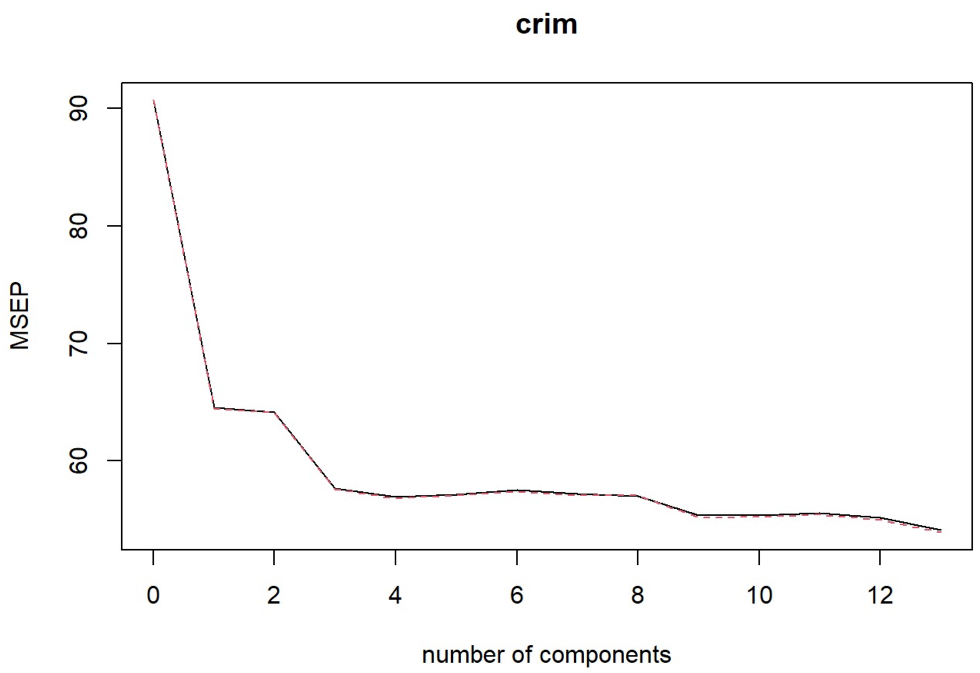
```
xtrain = model.matrix(train2$crim ~ ., data = train2[,-1])
fit222 <- cv.glmnet(xtrain, train2$crim, alpha = 1)
```

```
xtest=model.matrix(test2$crim ~ ., data = test2[,-1])
p <- predict(fit222, xtest, s = fit222$lambda.min)
mse_lasso <- mean((p - test2$crim)^2)
mse_lasso
```

```
## [1] 18.07153
```

INTERPRETATION #his coefficient indicates that for a one-unit increase in the associated predictor variable, the predicted crime rate (crim) increases by approximately 4.251062 units

```
fit5 <- pcr(crim ~ ., data = train2, scale = TRUE, validation = "CV")
validationplot(fit5, val.type = "MSEP")
```




```
summary(fit5)
```

```
## Data:    X dimension: 354 13
##  Y dimension: 354 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           9.525    8.032    8.009    7.595    7.545    7.560    7.583
## adjCV        9.525    8.028    8.005    7.587    7.539    7.555    7.575
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       7.564    7.552    7.439      7.44     7.453     7.427     7.356
## adjCV    7.556    7.554    7.429      7.43     7.444     7.413     7.342
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       48.58    61.36    70.50    77.35    83.36    88.23    91.30    93.45
## crim    29.82    30.41    37.81    38.42    38.54    38.72    39.17    40.09
##       9 comps  10 comps  11 comps  12 comps  13 comps
## X       95.54     97.16     98.50     99.54    100.00
## crim    41.44     41.62     41.62     43.09     44.25
```

INTERPRETATION #the optimal number of components appears to be around 4 to 5, after which the improvement in RMSEP is marginal #The lowest RMSEP of 7.439 occurs at 9 components, indicating this model configuration offers the best predictive performance

```
p <- predict(fit5, test2, ncomp = 4)
mse_pcr <- mean((p - test2$crim)^2)
mse_pcr
```

```
## [1] 21.06829
```

INTERPRETATION #an RMSEP of 4.590021 indicates that the model's predictions deviate from actual values by about 4.59 units on average

b. Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross-validation, or some other reasonable alternative, as opposed to using training error. We will try to fit models to `log(Boston$crim)` which is closer to a normal distribution.

```
set.seed(1)
train <- sample(nrow(Boston), nrow(Boston) * 2 / 3)
test <- setdiff(seq_len(nrow(Boston)), train)
```

INTERPRETATION #the Boston dataset into a training set (67%) and a test set (33%) with reproducible random sampling by setting a seed.

```
fit <- lm(log(crim) ~ ., data = Boston[train, ])
mean((predict(fit, Boston[test, ]) - log(Boston$crim[test]))^2)
```

```
## [1] 0.6779016
```

INTERPRETATION This code fits a linear regression model on the log of crim using all predictors on the training data, then calculates the mean squared error (MSE) of predictions on the test data.

```
mm <- model.matrix(log(crim) ~ ., data = Boston[train, ])
fit2 <- cv.glmnet(mm, log(Boston$crim[train]), alpha = 0)
ridge <- predict(fit2, model.matrix(log(crim) ~ ., data = Boston[test, ]), s = fit2$lambda.min)
mean((ridge - log(Boston$crim[test]))^2)
```

```
## [1] 0.665665
```

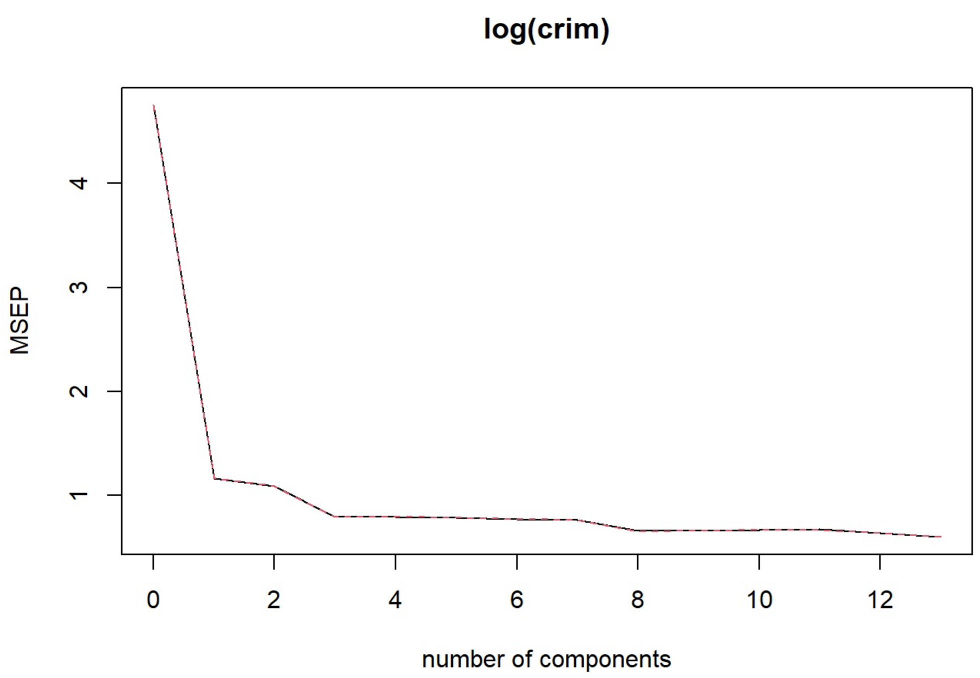
INTERPRETATION This indicates the average squared difference between the predicted and actual log-transformed crime rates (crim) in the test data. Lower MSE values generally imply better predictive accuracy, so an MSE of 0.665665 suggests the model's fit accuracy on unseen data.

```
mm <- model.matrix(log(crim) ~ ., data = Boston[train, ])
fit3 <- cv.glmnet(mm, log(Boston$crim[train]), alpha = 1)
lesso <- predict(fit3, model.matrix(log(crim) ~ ., data = Boston[test, ]), s = fit3$lambda.min)
mean((lesso - log(Boston$crim[test]))^2)
```

```
## [1] 0.6541562
```

INTERPRETATION the lasso regression model's predictions on the test set. This MSE value indicates the average squared difference between the predicted and actual log-transformed crime rates (crim). An MSE of 0.6541562 suggests that the lasso model's predictions are reasonably close to the actual values, reflecting the model's accuracy on unseen data. Lower MSE compared to previous models indicates improved predictive performance.

```
fit4 <- pcr(log(crim) ~ ., data = Boston[train, ], scale = TRUE, validation = "CV")
validationplot(fit4, val.type = "MSEP")
```

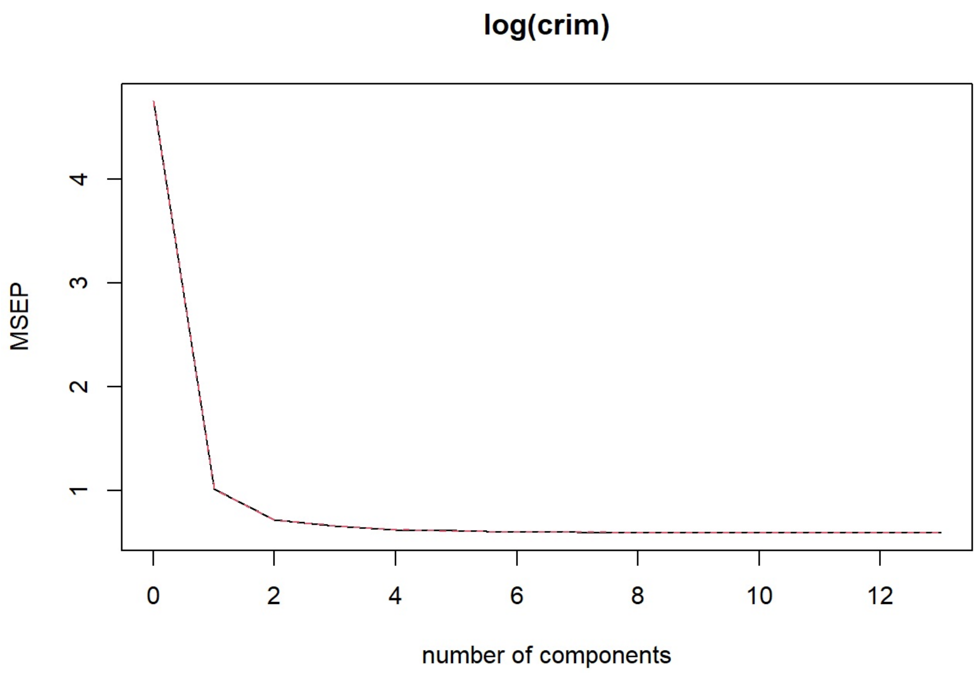



```
pcr <- predict(fit4, Boston[test, ], ncomp = 8)
mean((pcr - log(Boston$crim[test]))^2)
```

```
## [1] 0.6609357
```

INTERPRETATION the average squared difference between the predicted and actual log-transformed crime rates, allowing assessment of the PCR model's performance with 8 components.

```
fit5 <- plsr(log(crim) ~ ., data = Boston[train, ], scale = TRUE, validation = "CV")
validationplot(fit5, val.type = "MSEP")
```




```
plsr<- predict(fit5, Boston[test, ], ncomp = 6)
mean((plsr - log(Boston$crim[test]))^2)
```

```
## [1] 0.6911389
```

INTERPRETATION the PLSR model's performance using 6 components. A lower MSE indicates better predictive accuracy, suggesting how well the model generalizes to new data.

```
coef(fit3, s = fit3$lambda.min)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -4.029305640
## (Intercept)  .
## zn          -0.011299858
## indus        0.022032467
## chas         .
## nox          3.766724465
## rm          -0.025289189
## age          0.004397170
## dis          .
## rad          0.139103776
## tax          .
## ptratio     -0.026355504
## black       -0.001732709
## lstat        0.034798406
## medv         0.009282387
```

INTERPRETATION ###In this case lasso (`alpha = 1`) seems to perform very slightly better than un-penalized regression. Some coefficients have been dropped: the lasso regression model has selected a few key predictors (like nox, age, rad, and lstat) as significant for predicting the log of crime rates, while excluding others. This simplifies the model and emphasizes the most impactful features in understanding the relationship between predictors and crime rates in the Boston dataset.

#As computed above the model with the lower cross-validation error is the one chosen by the Ridge method.

c. Does your chosen model involve all of the features in the data set? Why or why not?

###Not all features are included due to the lasso penalization. The subset models, especially lasso regression, often result in sparse models, meaning only a few features may have non-zero coefficients. Ridge regression, however, uses all features but applies a penalty to prevent overfitting. For best performance, a lasso model might be preferable since it yields a simpler model, excluding irrelevant predictors and improving interpretability without sacrificing accuracy.