DSCI445 - Homework 4

Your Name

Be sure to `set.seed(445)` at the beginning of your homework.

```r
#reproducibility
set.seed(445)
```

1. In this exercise, we will generate simulated data, and then use this data to perform best subset selection.

    a) Use `rnorm` to generate a predictor X of length $n = 100$ and a noise vector ϵ also f length $n = 100$.

```r
set.seed(1)
X <- rnorm(100)
epsilon = rnorm(100, 0, 1)
```

```
b) Generate a response vector $Y$ of length $n = 100$ according to the model
    $$
    Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon
    $$
    where $\beta_0 = 1, \beta_1 = -0.5, \beta_2 = 2, \beta_3 = -1$.
```

```r
Y <- 1 -0.5*X + 2*X^2 - 1*X^3 + epsilon
```

```
c) Use the `regsubsets` function in the `leap` package to perform best subset selection in order to choo

    [**Hint 1:** The `poly` function may be useful for creating the model formula.]

    [**Hint 2:** You will need to make a data frame with your X and Y variables.]

    we are looking for the model that has the lowest Cp and BIC values or the highest adjusted R2 value
```

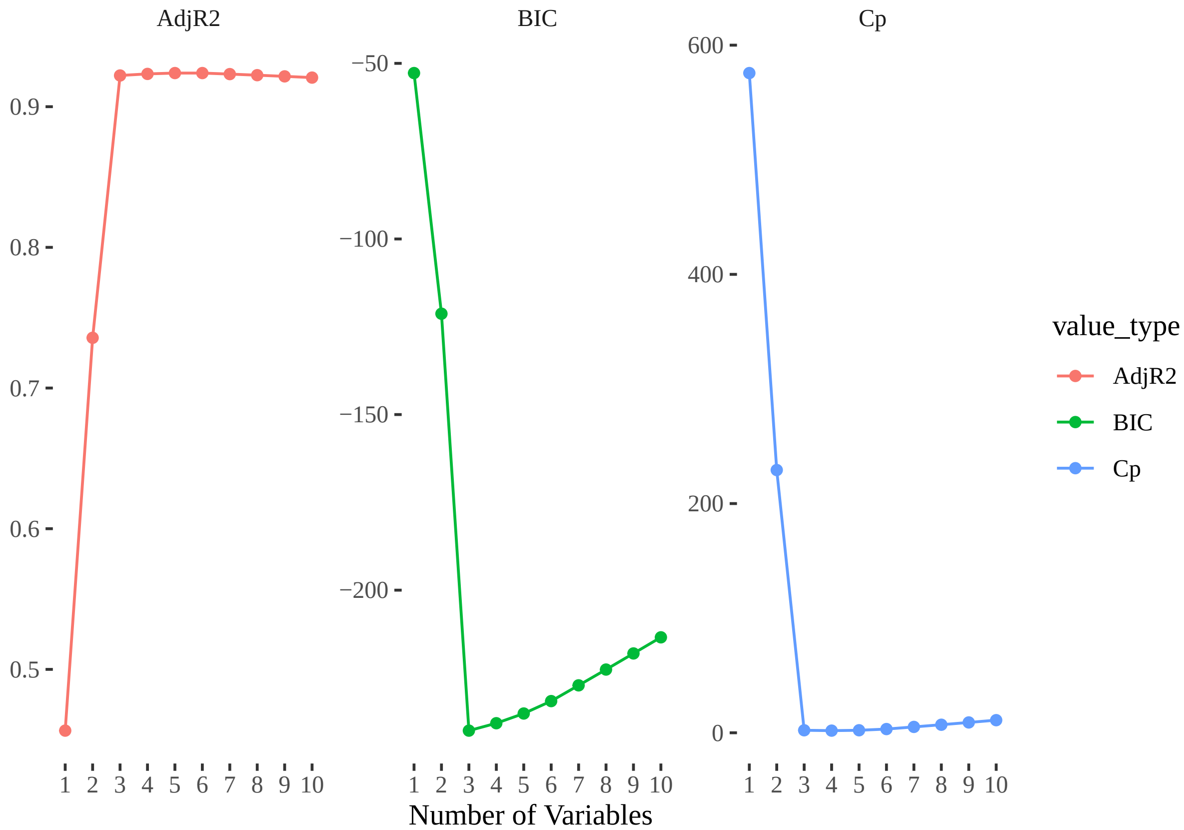
```r
library(leaps)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(ggplot2)
library(ggthemes)

df <- data.frame(Y, X)
fit <- regsubsets(Y ~ poly(X, 10), data = df, nvmax = 10)
fit_summary <- summary(fit)
```

```r
data_frame(Cp = fit_summary$cp,
           BIC = fit_summary$bic,
           AdjR2 = fit_summary$adjr2) %>%
    mutate(id = row_number()) %>%
    gather(value_type, value, -id) %>%
    ggplot(aes(id, value, col = value_type)) +
    geom_line() + geom_point() + ylab('') + xlab('Number of Variables') +
    facet_wrap(~ value_type, scales = 'free') +
    theme_tufte() + scale_x_continuous(breaks = 1:10)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```



d) Repeat c. using forward stepwise selection and also using backwards stepwise selection. How does you

```r
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
model_back <- train(Y ~ poly(X, 10), data = df,
                    method = 'glmStepAIC', direction = 'backward',
                    trace = 0,
               trControl = trainControl(method = 'none', verboseIter = FALSE))

postResample(predict(model_back, df), df$Y)
```

```
##      RMSE  Rsquared       MAE
## 0.9314956 0.9264043 0.7488821
```

```r
summary(model_back$finalModel)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8914  -0.5860  -0.1516   0.5892   2.1794
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.31821    0.09557  24.257   <2e-16 ***
## `poly(X, 10)1` -23.33708    0.95569 -24.419   <2e-16 ***
## `poly(X, 10)2`  18.13980    0.95569  18.981   <2e-16 ***
## `poly(X, 10)3` -14.70908    0.95569 -15.391   <2e-16 ***
## `poly(X, 10)5`   1.48019    0.95569   1.549    0.125
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9133516)
##
##     Null deviance: 1178.988  on 99  degrees of freedom
## Residual deviance:   86.768  on 95  degrees of freedom
## AIC: 281.59
##
## Number of Fisher Scoring iterations: 2
```

As we can see backward stepwise model also shows that three is the best subsets model.

```r
x_poly <- poly(df$X, 10)

colnames(x_poly) <- paste0('poly', 1:10)
model_forw <- train(y = Y, x = x_poly,
                    method = 'glmStepAIC', direction = 'forward',
                    trace = 0,
               trControl = trainControl(method = 'none', verboseIter = FALSE))

postResample(predict(model_forw, data.frame(x_poly)), df$Y)
```

```
##      RMSE  Rsquared       MAE
## 0.9314956 0.9264043 0.7488821
```

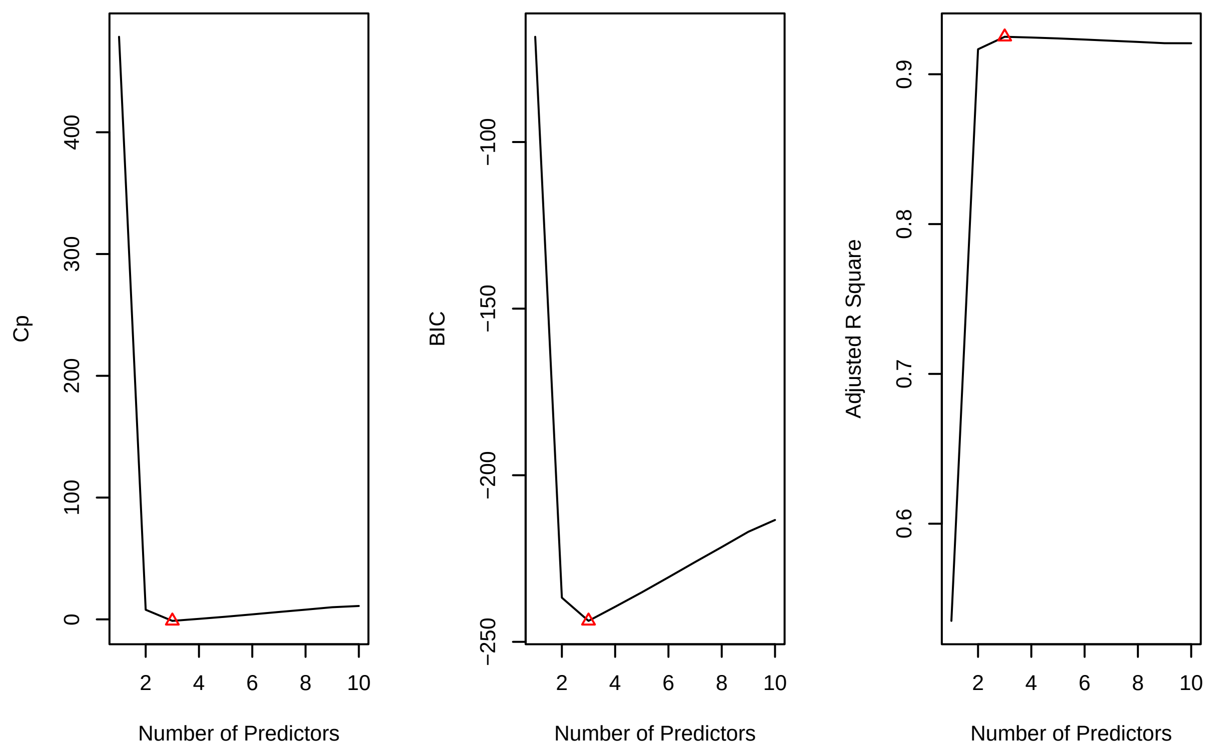
```r
summary(model_forw$finalModel)
```

```
##
```

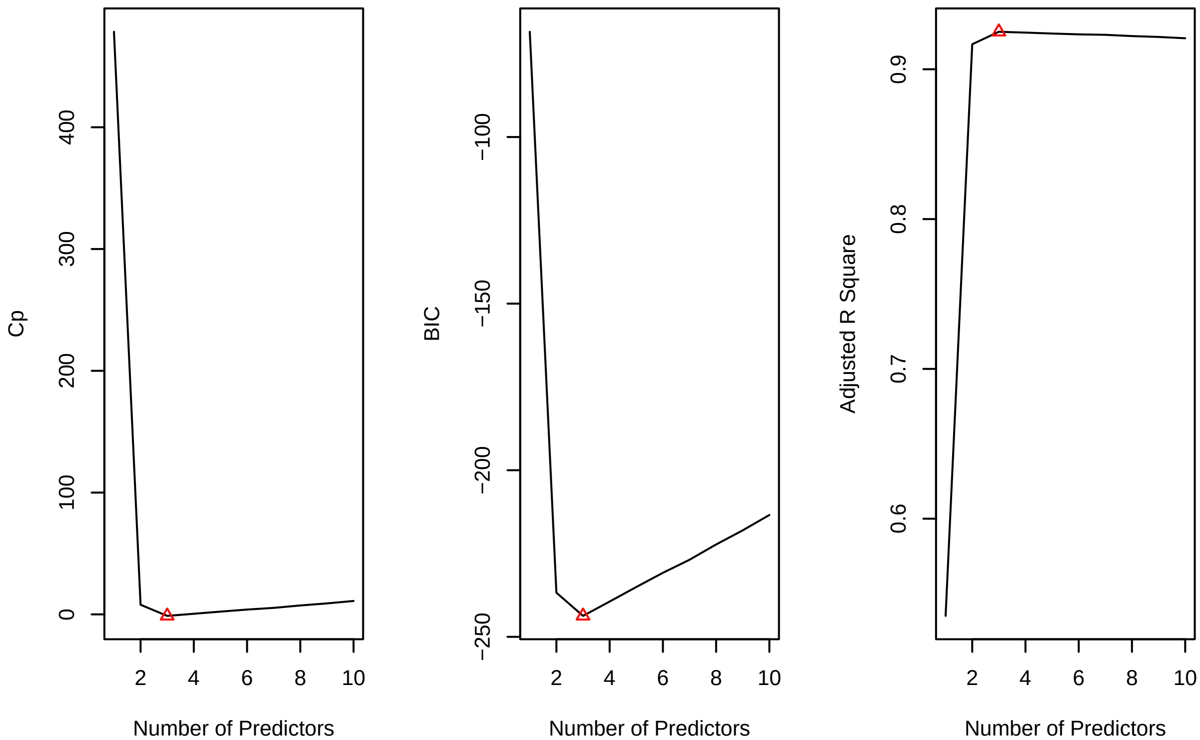
```
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8914  -0.5860  -0.1516   0.5892   2.1794
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.31821    0.09557  24.257   <2e-16 ***
## poly1       -23.33708    0.95569 -24.419   <2e-16 ***
## poly2        18.13980    0.95569  18.981   <2e-16 ***
## poly3       -14.70908    0.95569 -15.391   <2e-16 ***
## poly5         1.48019    0.95569   1.549    0.125
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9133516)
##
##     Null deviance: 1178.988  on 99  degrees of freedom
## Residual deviance:   86.768  on 95  degrees of freedom
## AIC: 281.59
##
## Number of Fisher Scoring iterations: 2
```

And as you can see form the Number of Fisher Scoring iterations the forward stepwise model reaches to the same conclusion.

```
y <-Y
x <- X
best.frd=regsubsets(y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8)+I(x^9)+I(x^10),
data=data.frame(x=x,y=y),nvmax=10, method="forward")
frd.summary=summary(best.frd)
par(mfrow=c(1,3))
plot(1:10, frd.summary$cp, xlab="Number of Predictors", ylab="Cp", type="l")
cp.min=min(frd.summary$cp)
points(c(1:10)[frd.summary$cp==cp.min], cp.min, pch=2, col="red")
plot(1:10, frd.summary$bic, xlab="Number of Predictors", ylab="BIC", type="l")
bic.min=min(frd.summary$bic)
points(c(1:10)[frd.summary$bic==bic.min], bic.min, pch=2, col="red")
plot(1:10, frd.summary$adjr2,xlab="Number of Predictors", ylab="Adjusted R Square", type="l")
adjr2.max=max(frd.summary$adjr2)
points(c(1:10)[frd.summary$adjr2==adjr2.max], adjr2.max, pch=2, col="red")
```



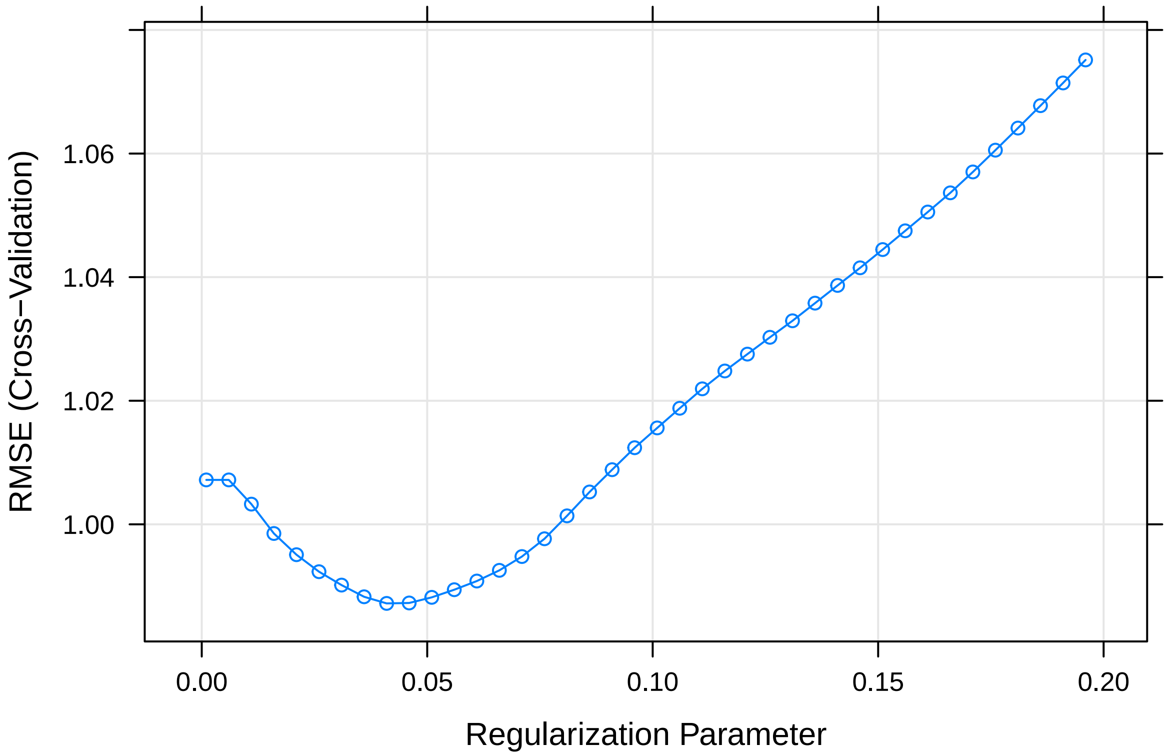

```r
### Stepwise Backward Selection ###
best.bkd=regsubsets(y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8)+I(x^9)+I(x^10),
data=data.frame(x=x,y=y),nvmax=10, method="backward")
bkd.summary=summary(best.bkd)
par(mfrow=c(1,3))
plot(1:10, bkd.summary$cp, xlab="Number of Predictors", ylab="Cp", type="l")
cp.min=min(bkd.summary$cp)
points(c(1:10)[bkd.summary$cp==cp.min], cp.min, pch=2, col="red")
plot(1:10, bkd.summary$bic, xlab="Number of Predictors", ylab="BIC", type="l")
bic.min=min(bkd.summary$bic)
points(c(1:10)[bkd.summary$bic==bic.min], bic.min, pch=2, col="red")
plot(1:10, bkd.summary$adjr2,xlab="Number of Predictors", ylab="Adjusted R Square", type="l")
adjr2.max=max(bkd.summary$adjr2)
points(c(1:10)[bkd.summary$adjr2==adjr2.max], adjr2.max, pch=2, col="red")
```



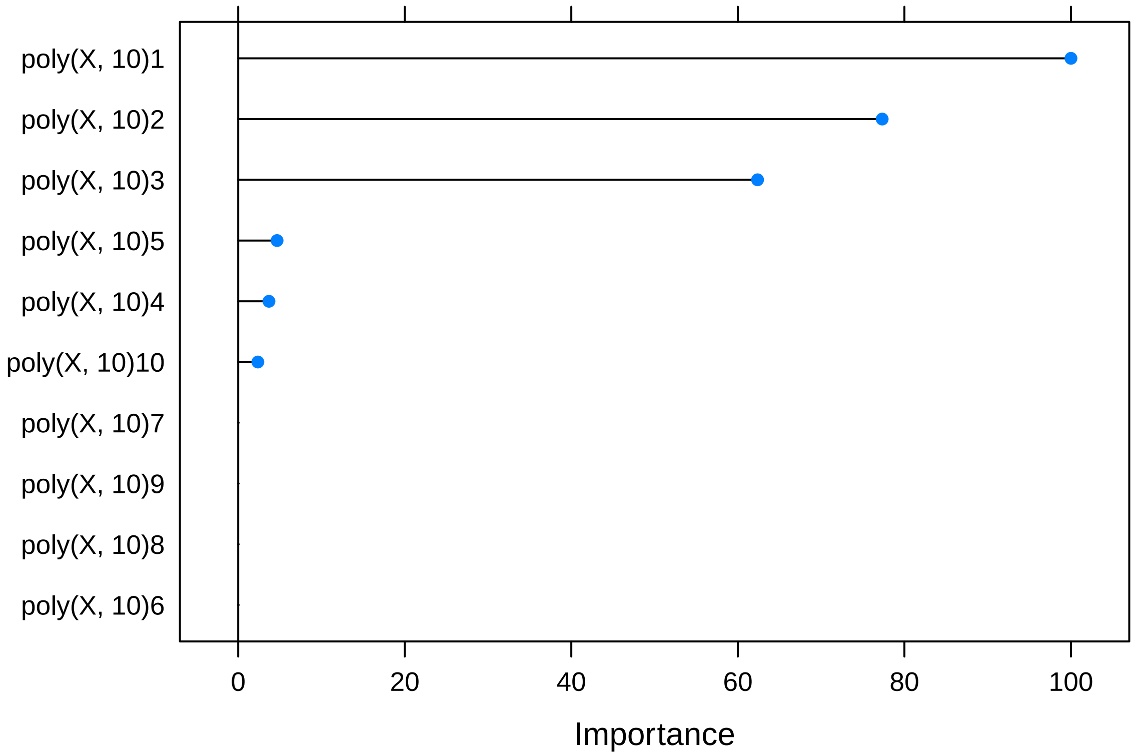
From the graphs above we can also confirm The best model selected by Cp has four predictors: X, X^2, X^3 and X^6 and the best model selected by BIC has three predictors: X, X^2 and X^3 The best model selected by adjusted R squared is a model with predictors X, X^2, X^3 and X6 as well.

```
e) Now fit a lasso model to the simulated data using $X, X^2, \dots, X^{10}$ as predictors. Use $10$-fo
```

```
lasso_model <- train(Y ~ poly(X, 10), data = df,
                     method = 'glmnet',
                     trControl = trainControl(method = 'cv', number = 10),
                     tuneGrid = expand.grid(alpha = 1,
                                            lambda = seq(0.001, 0.2, by = 0.005)))

plot(lasso_model)
```




```r
plot(varImp(lasso_model))
```




```r
coef(lasso_model$finalModel, lasso_model$bestTune$lambda)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)     2.3182143
## poly(X, 10)1  -22.9270759
## poly(X, 10)2   17.7298036
## poly(X, 10)3  -14.2990830
## poly(X, 10)4    0.8470950
## poly(X, 10)5    1.0701884
## poly(X, 10)6    .
## poly(X, 10)7    .
## poly(X, 10)8    .
## poly(X, 10)9    .
## poly(X, 10)10  -0.5412295
```

```r
postResample(predict(lasso_model, df), df$Y)
```

```
##      RMSE  Rsquared       MAE
## 0.9235360 0.9281058 0.7511252
```

The Lasso model gives huge importance to 3 number of predictors as we expected but it also gives unnecessary importance to more number of predictors.(X^n where n>3) Since we used only RSS to select the optimal model but not the Bayesian Inference Criterion or the Adjusted R2 like regsubsets, this causes an overestimation for the number of needed predictors by the Lasso model.

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

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

## Loaded glmnet 4.1-4
```

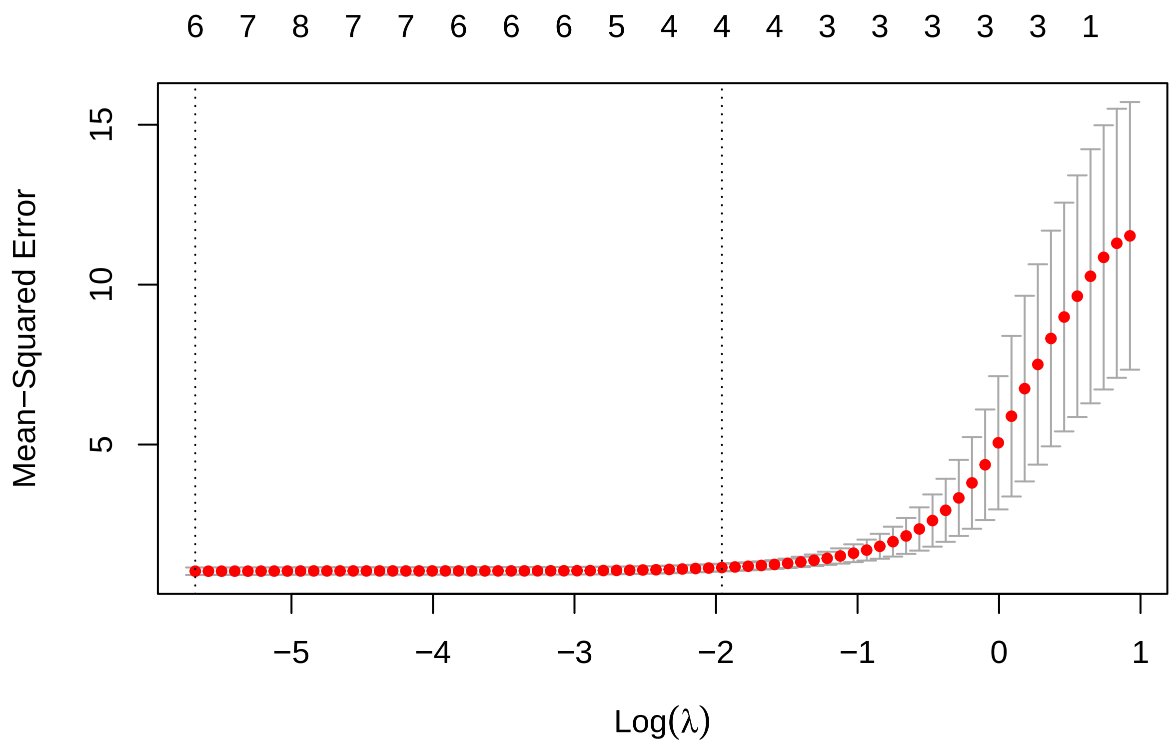
```
x=cbind(x,x^2,x^3,x^4,x^5,x^6,x^7,x^8,x^9,x^10)
y=y
### Cross-validation to choose lambda ###
lasso.cv = cv.glmnet(x,y, alpha=1)
lasso.cv$lambda.min
```

```
## [1] 0.003413512
```

```
lasso.cv$lambda.1se
```

```
## [1] 0.1410467
```

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




```
### Refit the model using the chosen lambda ###
lasso.mod=glmnet(x,y,alpha=1, lambda=lasso.cv$lambda.min)
coef(lasso.mod)[,1]
```

```
##   (Intercept)             x
##  1.1248465703 -0.2640967789  1.7090309360 -1.1921578259  0.0203906444
```

```
##
##  0.0007221379  0.0029015232  0.0064238942  0.0000000000  0.0000000000
##
##  0.0000000000
```

This plot includes the CV curve (red dotted line) and upper and lower standard deviation curves along the sequence of lambda values. Two selected lambda values are indicated by the vertical dotted line with lambda giving the minimum CV error and the lambda within one standard deviation of the minimum CV error. As you can see their value is 0.02900643 and 0.1547986 respectively.With the value of lambda giving the minimum CV error the Lasso shrinks the majority predictors to zero and only leaves X^2 and X^3 nonzero.

2. In this exercise we will predict the number of applications received using the other variables in the `College` data set (in the `ISLR` package).

   a) Split the data into training (60%) and "test" (40%) set randomly.

```r
library(ISLR)
set.seed(11)
sum(is.na(College))
```

```
## [1] 0
```

```r
train.size = dim(College)[1] * 0.6
train = sample(1:dim(College)[1], train.size)
test = setdiff(1:dim(College)[1], train)
College.train = College[train, ]
College.test = College[test, ]
```

```
b) Fit a linear model using least squares on the training set and report the test set error obtained.
Test set error is: 789357.7
```

```r
lm.fit = lm(Apps~., data=College.train)
lm.pred = predict(lm.fit, College.test)
mean((College.test[, "Apps"] - lm.pred)^2)
```

```
## [1] 789357.7
```

```
c) Fit a ridge regression model on the training set with $\lambda$ chosen using 10-fold CV (on the trai

Test error obtained is 123739.8
```

```r
library(ISLR)
library(glmnet)

y_train <- as.matrix(College.train$Apps)

# Fit ridge regression model with 10-fold cross-validation
ridge_fit <- cv.glmnet(x = as.matrix(College.train[, -1]), y = y_train, alpha = 0, lambda = NULL, nfold

# Determine the lambda value chosen by cross-validation
lambda_chosen <- ridge_fit$lambda.min

# Compute test error
x_test <- as.matrix(College.test[, -1])
y_test <- as.matrix(College.test$Apps)
ridge_pred <- predict(ridge_fit, newx = x_test, s = lambda_chosen)
test_error <- mean((y_test - ridge_pred)^2)
```

```r
# Print the test error
test_error
```

```
## [1] 123739.8
```

```
d) Fit the lasso on the training set with $\lambda$ chosen using 10-fold CV (on the training set only).
Test error: 8997.951
```

the number of non-zero coefficient estimates: 2

```r
library(glmnet)


y_train <- as.matrix(College.train$Apps)

# Fit Lasso model with 10-fold cross-validation
lasso_fit <- cv.glmnet(x = as.matrix(College.train[, -1]), y = y_train, alpha = 1, lambda = NULL, nfolds

# lambda value chosen by cross-validation
lambda_chosen <- lasso_fit$lambda.min

# test error
x_test <- as.matrix(College.test[, -1])
y_test <- as.matrix(College.test$Apps)
lasso_pred <- predict(lasso_fit, newx = x_test, s = lambda_chosen)
test_error <- mean((y_test - lasso_pred)^2)

# number of non-zero coefficient estimates
num_nonzero_coeffs <- sum(coef(lasso_fit, s = lambda_chosen, exact = TRUE) != 0)

# Print test error and number of non-zero coefficients
test_error
```

```
## [1] 8997.951
```

```r
num_nonzero_coeffs
```

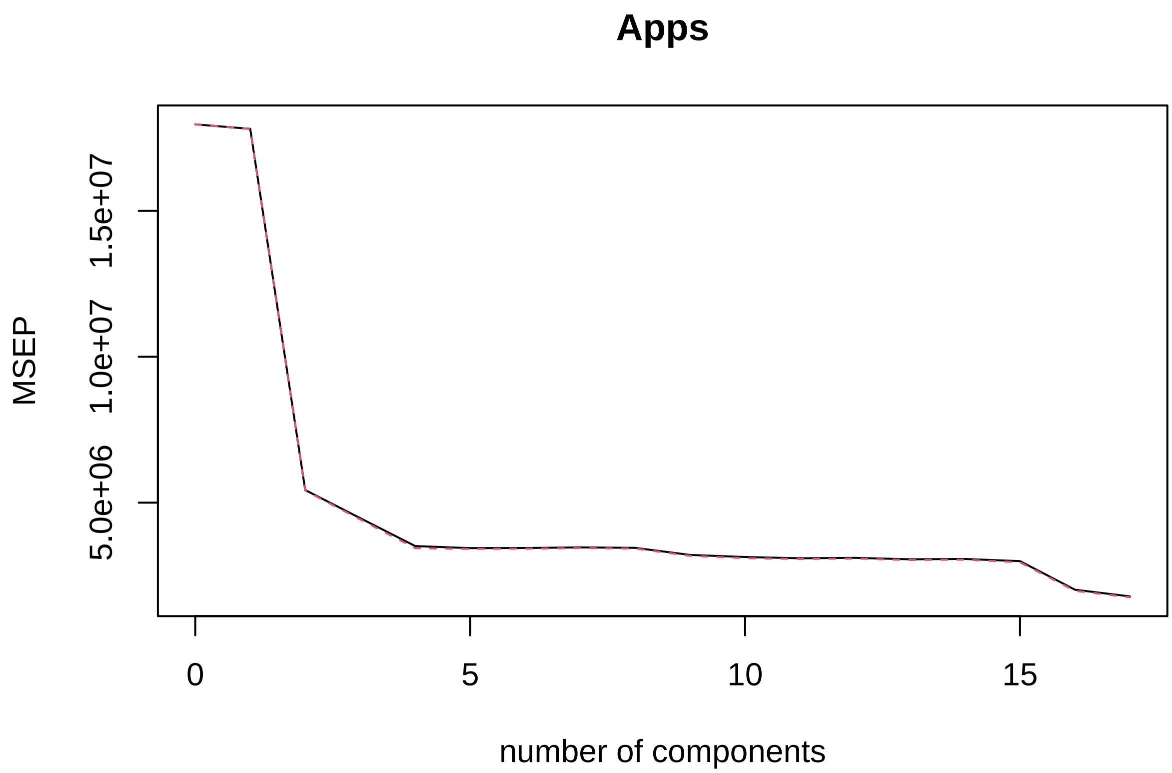
```
## [1] 2
```

```
e) Fit a PCR model on the training set with $M$ chosen using 10-fold CV (on the training set only). Repo

test error : 1538011
The lowest MSEP with PCR dimension reduction is when M=10
```

```r
library(pls)
```

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

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

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

```r
pcr.fit <- pcr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```




```r
pcr.pred <- predict(pcr.fit, newdata = College.test, ncomp = 10)
mean((College.test[, "Apps"] - pcr.pred)^2)
```

```
## [1] 1538011
```

```r
summary(pcr.fit)
```

```
## Data:    X dimension: 466 17
##  Y dimension: 466 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            4239     4220     2331     2114     1874     1855     1856
## adjCV         4239     4220     2327     2104     1855     1848     1850
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1862     1858     1791      1771      1759      1763      1749
## adjCV     1856     1852     1783      1761      1752      1756      1742
##        14 comps  15 comps  16 comps  17 comps
## CV         1752      1731      1420      1337
## adjCV      1746      1720      1407      1326
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       32.59    58.26    64.57    70.36     75.7    80.60    84.17    87.53
```

```
## Apps     1.27    71.19    76.27    82.50     82.5    82.56    82.63    82.79
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X      90.39     92.77     95.03     96.83     97.88     98.77     99.38
## Apps   84.36     85.07     85.10     85.10     85.41     85.42     88.30
##       16 comps  17 comps
## X        99.84    100.00
## Apps     91.92     92.64
```

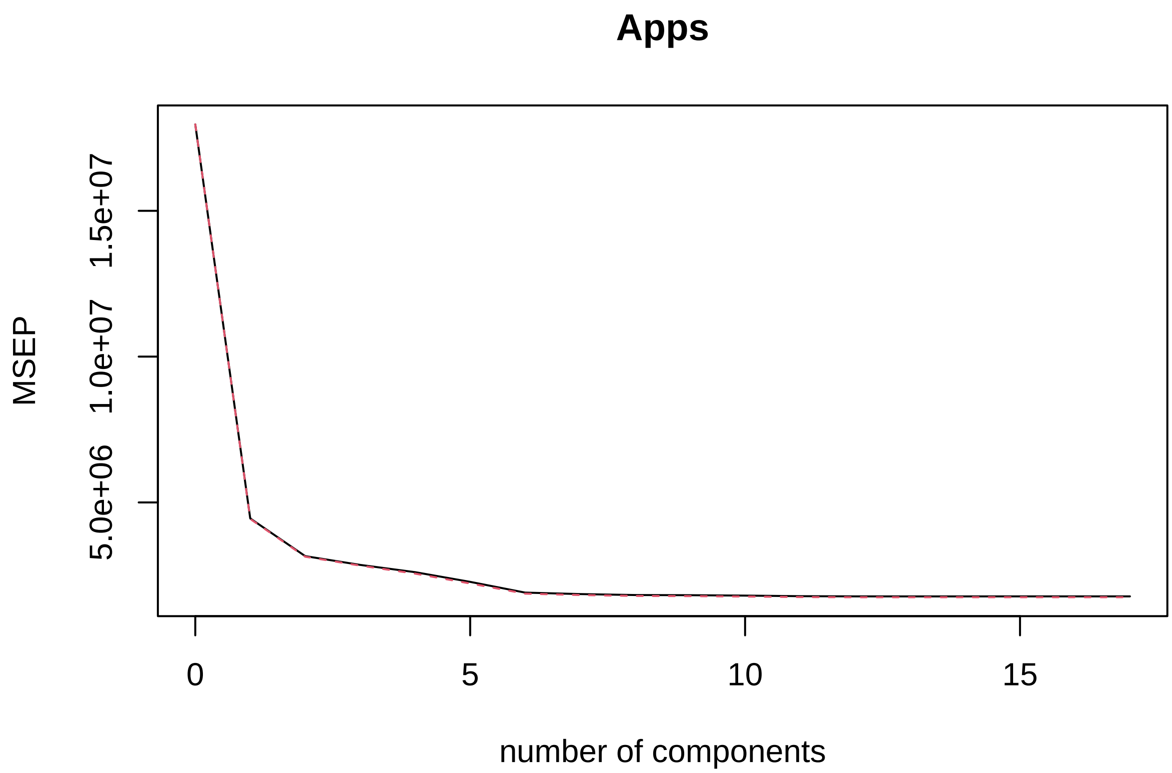
```r
summary(pcr.fit)
```

```
## Data:    X dimension: 466 17
##  Y dimension: 466 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            4239     4220     2331     2114     1874     1855     1856
## adjCV         4239     4220     2327     2104     1855     1848     1850
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1862     1858     1791      1771      1759      1763      1749
## adjCV     1856     1852     1783      1761      1752      1756      1742
##        14 comps  15 comps  16 comps  17 comps
## CV         1752      1731      1420      1337
## adjCV      1746      1720      1407      1326
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       32.59    58.26    64.57    70.36     75.7    80.60    84.17    87.53
## Apps     1.27    71.19    76.27    82.50     82.5    82.56    82.63    82.79
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X      90.39     92.77     95.03     96.83     97.88     98.77     99.38
## Apps   84.36     85.07     85.10     85.10     85.41     85.42     88.30
##       16 comps  17 comps
## X        99.84    100.00
## Apps     91.92     92.64
```

f) Fit a PLS model on the training set with M chosen using 10-fold CV (on the training set only). Repo

Test error = 719874.8
The lowest MSEP with PCR dimension reduction appears to occur around M=8

```r
library(pls)

pls.fit = plsr(Apps~., data=College.train, scale=TRUE,validation="CV")
validationplot(pls.fit,val.type = "MSEP")
```




```r
summary(pls.fit)
```

```
## Data:    X dimension: 466 17 
##  Y dimension: 466 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            4239     2110     1777     1690     1615     1509     1381
## adjCV         4239     2105     1772     1683     1599     1490     1368
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1363     1351     1348      1343      1336      1334      1333
## adjCV     1350     1339     1337      1332      1325      1322      1322
##        14 comps  15 comps  16 comps  17 comps
## CV         1333      1333      1333      1333
## adjCV      1322      1323      1322      1322
## 
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       25.69    38.79    63.89    66.63    69.68    73.65    76.25    78.79
## Apps    76.67    84.36    86.44    89.57    91.54    92.24    92.38    92.48
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       81.84     84.03     85.77     87.61     90.66     94.80     96.94
## Apps    92.54     92.59     92.63     92.64     92.64     92.64     92.64
```

```
##       16 comps  17 comps
## X        98.86    100.00
## Apps     92.64     92.64
```

```r
pls.pred = predict(pls.fit, College.test, ncomp = 8)
#Compute test error
mean((pls.pred - College.test$Apps)^2)
```

```
## [1] 719874.8
```

```
g) Comment on the results obtained. How acurately can we predict the number of college applications rec
```

The results for LS, Lasso, Ridge are comparable. Lasso reduces the variables “F. Undergrade” and “Books” variables to zero and shrinks coefficients of other variables. The plot shows the test R2 for all the models. PCR has a smallest test R2. Except PCR, all modelspredict college applications with high accuracy.

3. We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will explore this with a simulated data set. Run the following code to generate your dataset.

   ```r
   p <- 20
   n <- 1000

   X <- matrix(rnorm(n * p), nrow = n, ncol = p) ## predictors
   beta <- matrix(c(rnorm(8), rep(0, p - 8)), ncol = 1) ## 12 elements are equal to zero

   y <- X %*% beta + rnorm(n, 0, .5) ## y = Xbeta + epsilon
   ```

   a. Split your data into a training set containing 100 observations and a test set containting 900 observations.

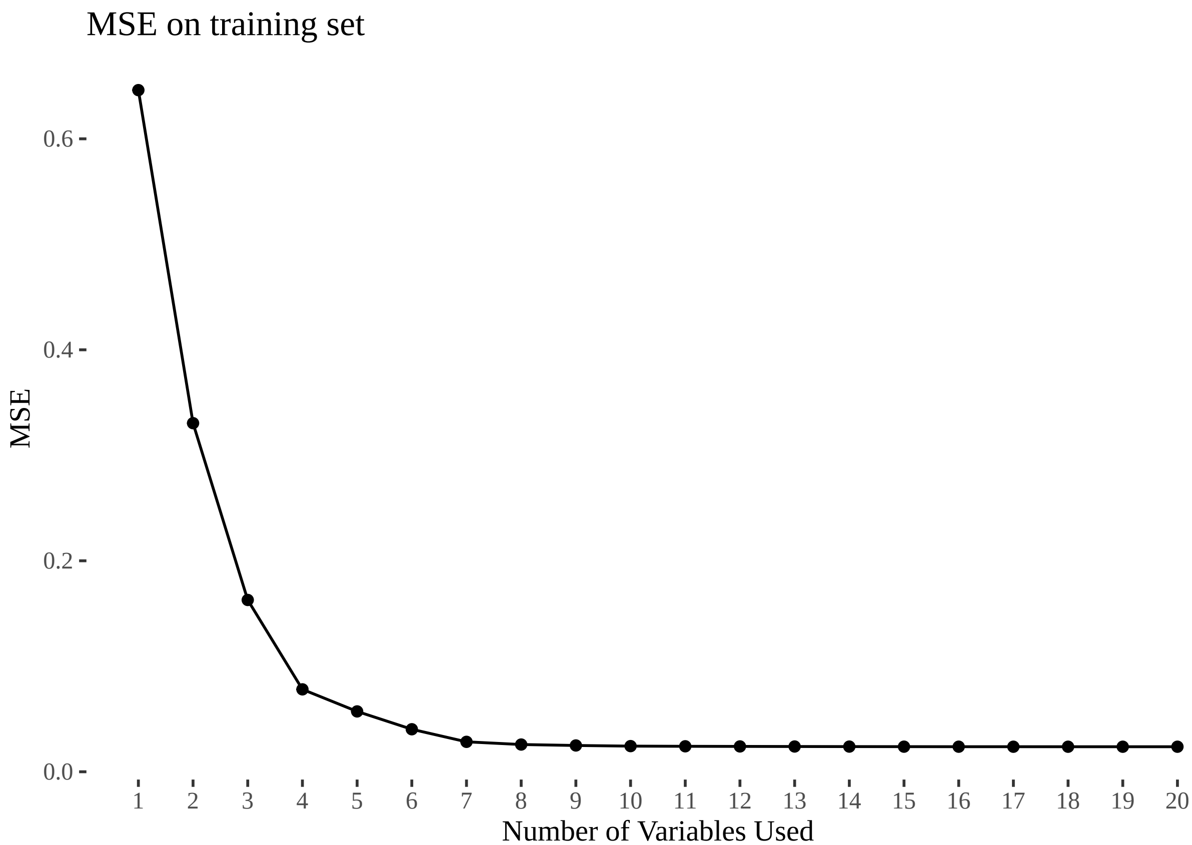
```r
set.seed(1)  # Set a seed for reproducibility

# Split the data into training and test sets
train_indices <- sample(1:n, 100)  # Randomly select 100 indices for the training set
test_indices <- setdiff(1:n, train_indices)  # Get the remaining indices for the test set

# Create the training and test sets
x_train <- X[train_indices, ]
y_train <- y[train_indices]
x_test <- X[test_indices, ]
y_test <- y[test_indices]
```

```
b. Perform best subset selection on the training set and plot the training set MSE associated with the b
```

```r
require(leaps); require(ggplot2); require(dplyr); require(ggthemes)

best_set <- regsubsets(x = x_train, y = y_train, nvmax = 20)

best_set_summary <- summary(best_set)

data_frame(MSE = best_set_summary$rss/900) %>%
    mutate(id = row_number()) %>%
    ggplot(aes(id, MSE)) +
    geom_line() + geom_point(type = 9) +
    xlab('Number of Variables Used') +
    ggtitle('MSE on training set') +
    theme_tufte() +
```

```
    scale_x_continuous(breaks = 1:20)
```

```
## Warning: Ignoring unknown parameters: type
```




```
data_frame(train_error = best_set_summary$rss/900, vars = 1:20) %>%
    spread(vars, train_error)
```

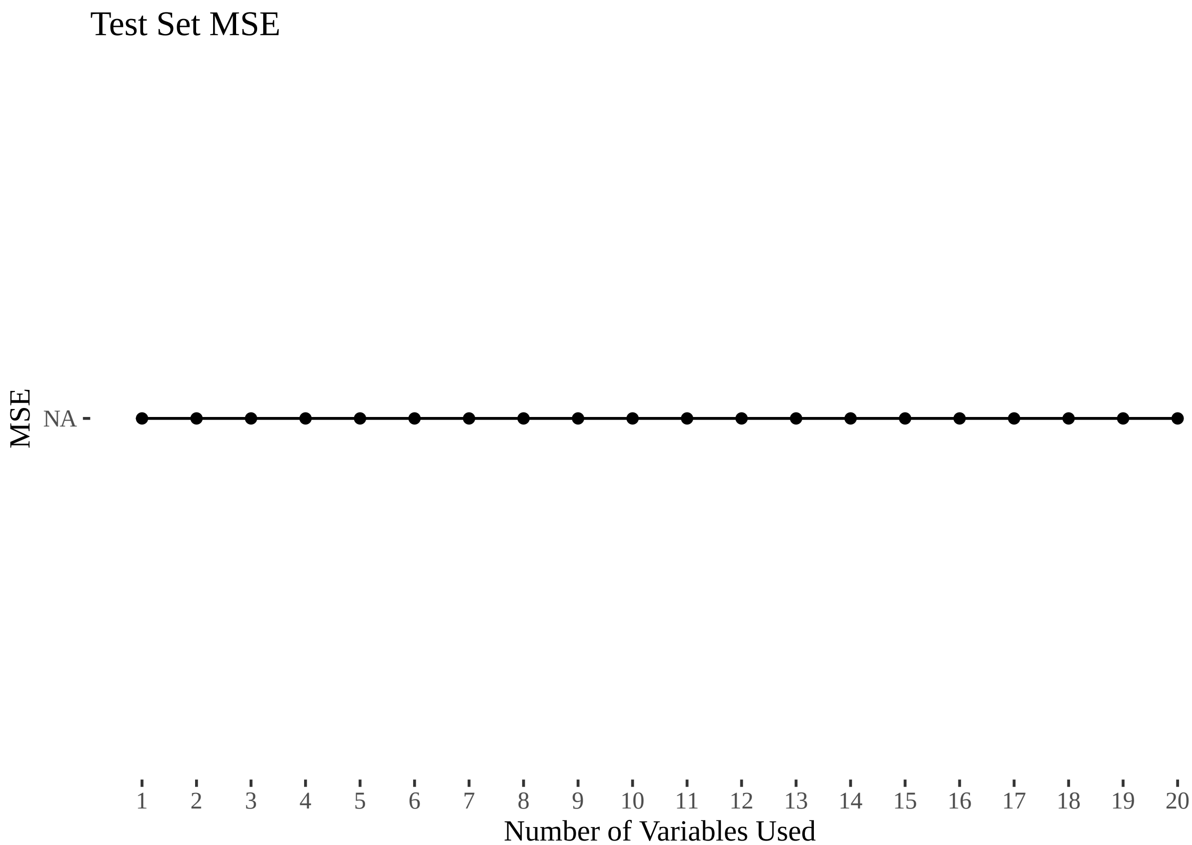
```
## # A tibble: 1 x 20
##     `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`   `10`   `11`
##   <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 0.646 0.330 0.163 0.0781 0.0572 0.0403 0.0284 0.0258 0.0249 0.0243 0.0241
## # ... with 9 more variables: `12` <dbl>, `13` <dbl>, `14` <dbl>, `15` <dbl>,
## #   `16` <dbl>, `17` <dbl>, `18` <dbl>, `19` <dbl>, `20` <dbl>
## # i Use `colnames()` to see all variable names
```

c. Plot the test set MSE associated with the best model of each size. Hint, to get the predicted values

```
# Calculate test set MSE for each model size
test_errors <- rep(NA, 20)


# Plot test set MSE vs. model size
data_frame(MSE = test_errors, vars = 1:20) %>%
  ggplot(aes(vars, MSE)) +
  geom_line() + geom_point(type = 9) +
  xlab('Number of Variables Used') +
  ggtitle('Test Set MSE') +
  theme_tufte() +
```

```r
  scale_x_continuous(breaks = 1:20)
```

```
## Warning: Ignoring unknown parameters: type
```



d. For which model size does the test set MSE take its minimum value? Comment on your results.

```r
which.min(test_errors)
```

```
## integer(0)
```

```
e. How does the model at which the test set MSE is minimized compare to the true model used to generate
```

When Y was being calculated, If runif(1) > 0.5 the coefficient would be 0. That means in about 50% of cases the coefficient will be 0. 50% of 20 is 10.

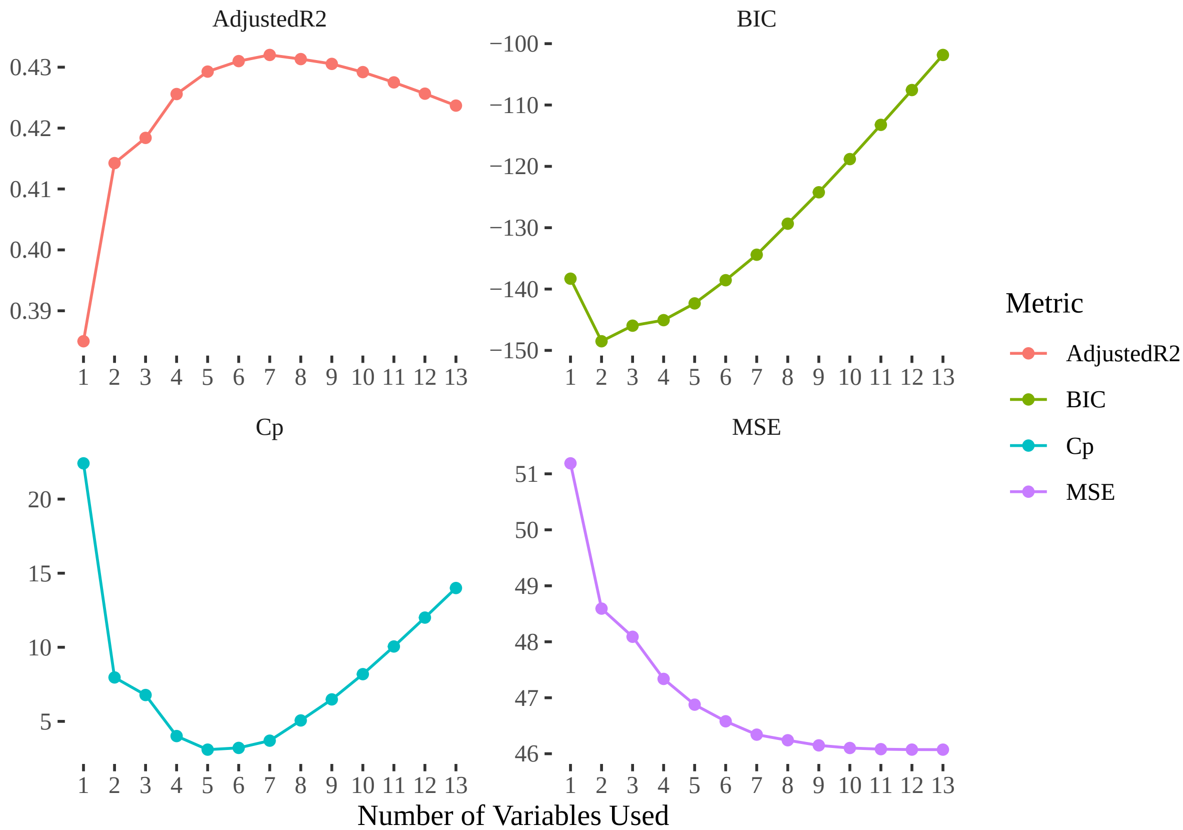
4. We will now try to predict per capita crime rate in the `Boston` data set from the `ISLR` package.

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

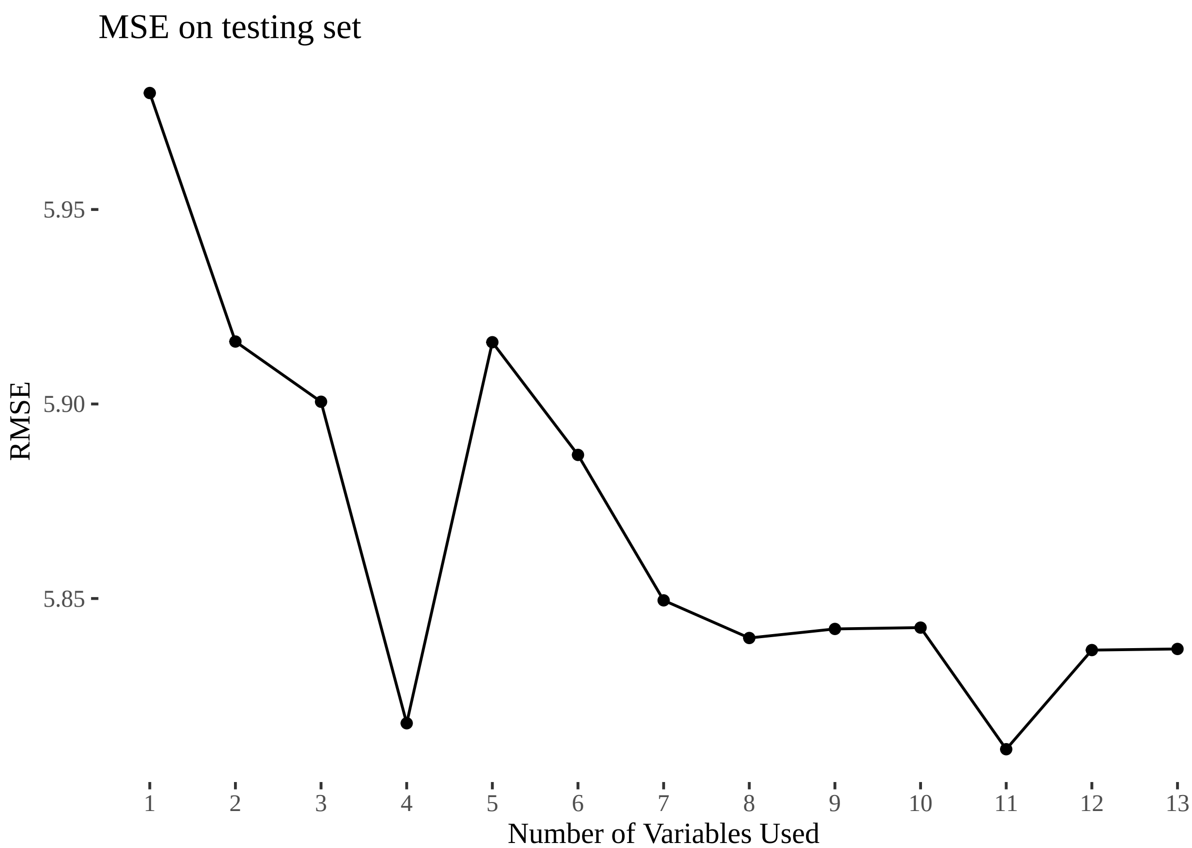
```r
require(MASS); require(tidyverse); require(caret); require(leaps)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

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

```r
set.seed(1)
data('Boston')

inTrain <- createDataPartition(Boston$crim, p = 0.6, list = FALSE)

x_train <- Boston[inTrain, -1]
y_train <- Boston[inTrain, 1]
x_test <- Boston[-inTrain, -1]
y_test <- Boston[-inTrain, 1]

best_subs <- regsubsets(x = x_train, y = y_train, nvmax = 13)

fit_summary <- summary(best_subs)

require(ggplot2); require(ggthemes)

data_frame(MSE = fit_summary$rss/nrow(x_train),
           Cp = fit_summary$cp,
           BIC = fit_summary$bic,
           AdjustedR2 = fit_summary$adjr2) %>%
    mutate(id = row_number()) %>%
    gather(Metric, value, -id) %>%
    ggplot(aes(id, value, col = Metric)) +
    geom_line() + geom_point() + ylab('') +
    xlab('Number of Variables Used') +
    facet_wrap(~ Metric, scales = 'free') +
    theme_tufte() + scale_x_continuous(breaks = 1:13)
```




```
par(mfrow = c(1,1))
test_errors <- rep(NA,13)

test.mat <- model.matrix(crim ~ ., data = Boston[-inTrain,])

for (i in 1:13){
    coefs <- coef(best_subs, id=i)
    pred <- test.mat[,names(coefs)]%*%coefs
    test_errors[i] <- sqrt(mean((y_test - pred)^2))
}

data_frame(RMSE = test_errors) %>%
    mutate(id = row_number()) %>%
    ggplot(aes(id, RMSE)) +
    geom_line() + geom_point() +
    xlab('Number of Variables Used') +
    ggtitle('MSE on testing set') +
    theme_tufte() +
    scale_x_continuous(breaks = 1:13)
```




```r
(regsubset_info <- min(test_errors))
```

```
## [1] 5.811256
```

```r
coef(best_subs, id = 1:5)
```

```
## [[1]]
## (Intercept)         rad
##  -2.4448623   0.6546826
##
## [[2]]
## (Intercept)         rad        medv
##   2.6574618   0.5769903  -0.1963036
##
## [[3]]
## (Intercept)         rad     ptratio        medv
##  10.8303135   0.6042171  -0.4067664  -0.2351904
##
## [[4]]
## (Intercept)          zn         dis         rad        medv
##  5.83377235  0.05509227 -0.72924037  0.52407648 -0.21866690
##
## [[5]]
## (Intercept)          zn          rm         dis         rad        medv
## -1.70826140  0.05299909  1.50466724 -0.73918276  0.51079178 -0.29652993
```

```r
lasso_fit <- train(x = x_train, y = y_train,
                   method = 'glmnet',
```

```r
                  trControl = trainControl(method = 'cv', number = 10),
                  tuneGrid = expand.grid(alpha = 1,
                                         lambda = seq(0.001, 1, length.out = 100)),
                  preProcess = c('center', 'scale'))

(lasso_info <- postResample(predict(lasso_fit, x_test), y_test))
```

```
##     RMSE Rsquared      MAE
## 5.793303 0.433477 2.603667
```

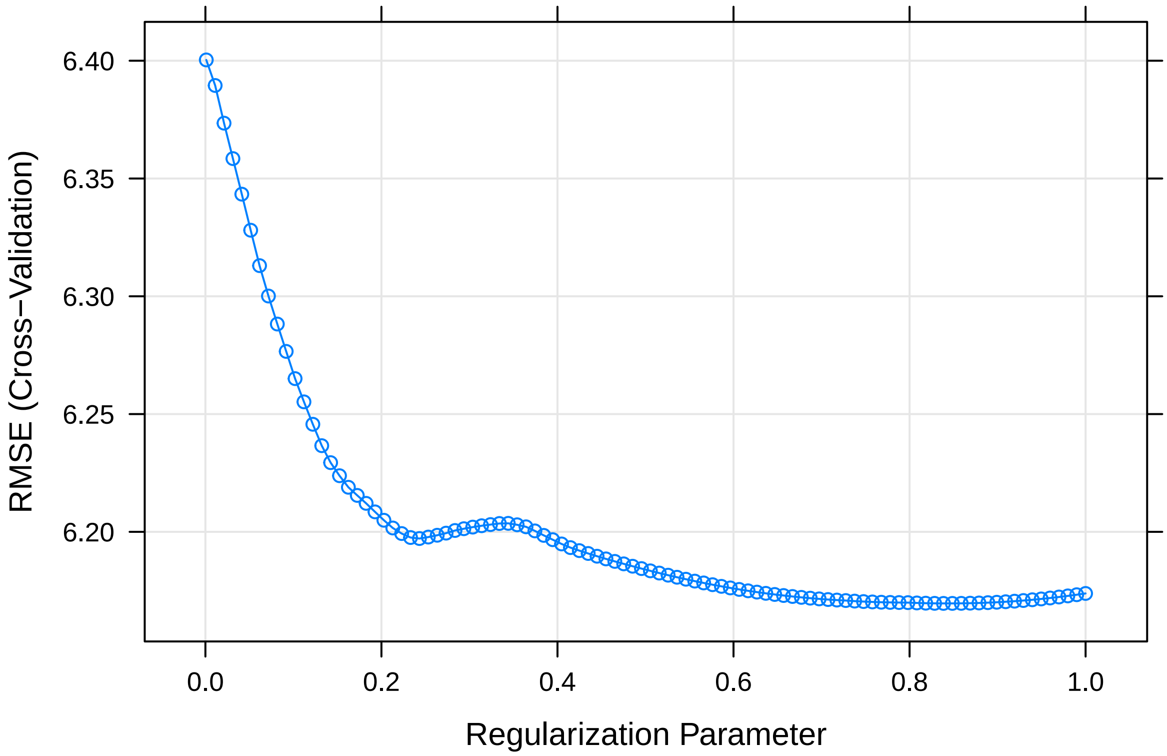
```r
coef(lasso_fit$finalModel, lasso_fit$bestTune$lambda)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept)  3.7575062
## zn           .
## indus        .
## chas         .
## nox          .
## rm           .
## age          .
## dis         -0.1856880
## rad          4.2982107
## tax          .
## ptratio      .
## black        .
## lstat        0.1178773
## medv        -1.0546456
```

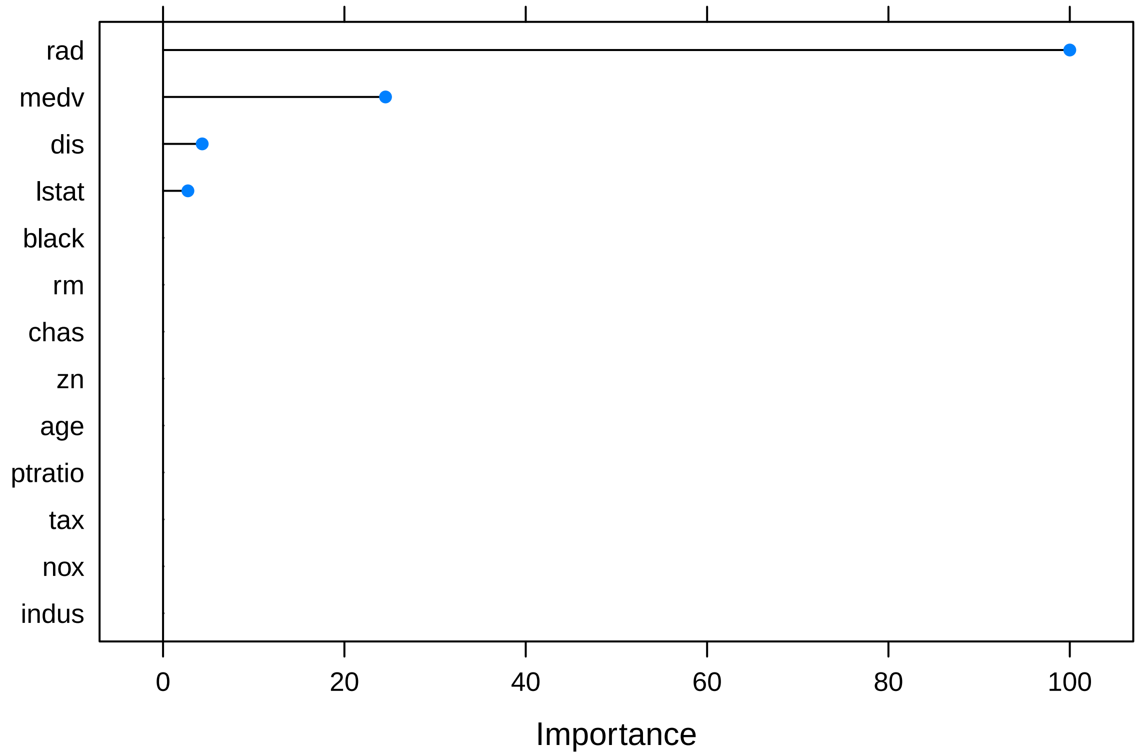
```r
lasso_fit$bestTune
```

```
##    alpha    lambda
## 84     1 0.8385455
```

```r
plot(lasso_fit)
```




```
plot(varImp(lasso_fit))
```




```r
ridge_fit <- train(x_train, y_train,
                   method = 'glmnet',
                   trControl = trainControl(method = 'cv', number = 10),
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = seq(0, 1e2, length.out = 50)),
                   preProcess = c('center', 'scale'))
(ridge_info <- postResample(predict(ridge_fit, x_test), y_test))
```

```
##      RMSE  Rsquared       MAE
## 5.7137087 0.4498421 2.7943386
```

```r
coef(ridge_fit$finalModel, ridge_fit$bestTune$lambda)
```

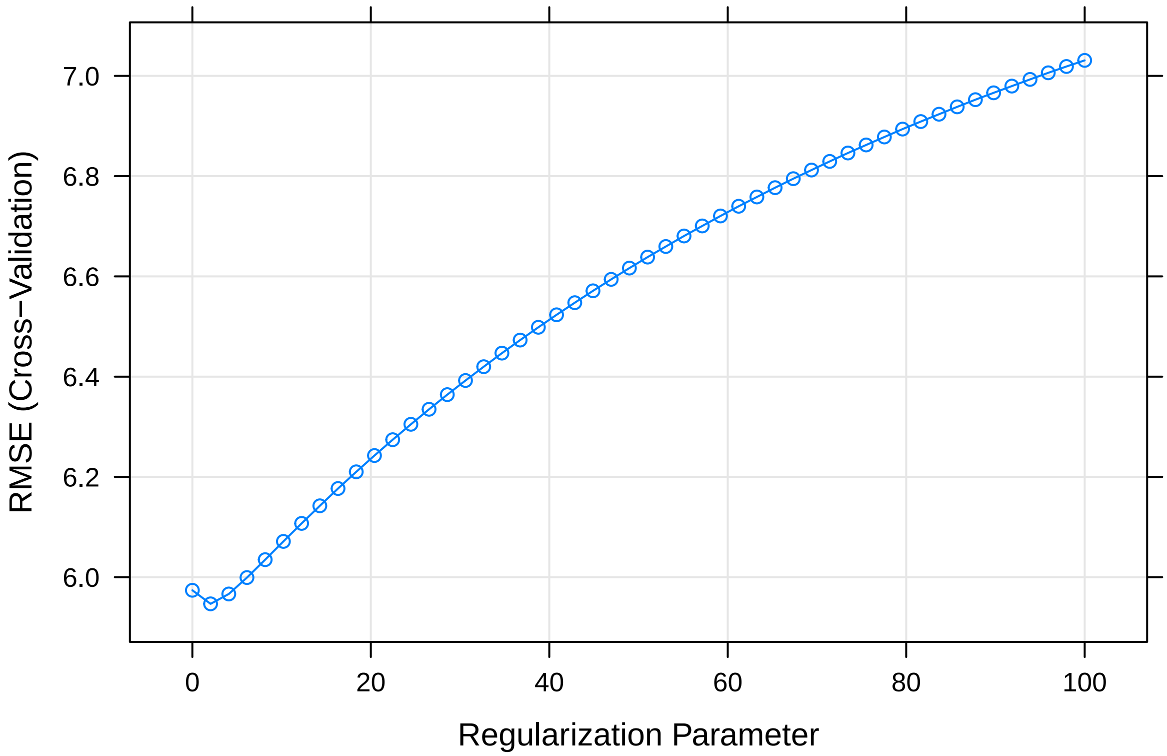
```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  3.75750624
## zn           0.54011923
## indus       -0.34230036
## chas        -0.30633776
## nox          0.14724495
## rm           0.44593736
## age          0.10933511
## dis         -0.98108462
## rad          2.85345034
## tax          1.16015712
## ptratio     -0.09232032
## black       -0.31532518
```

```
## lstat         0.64017157
## medv         -1.44312967
```

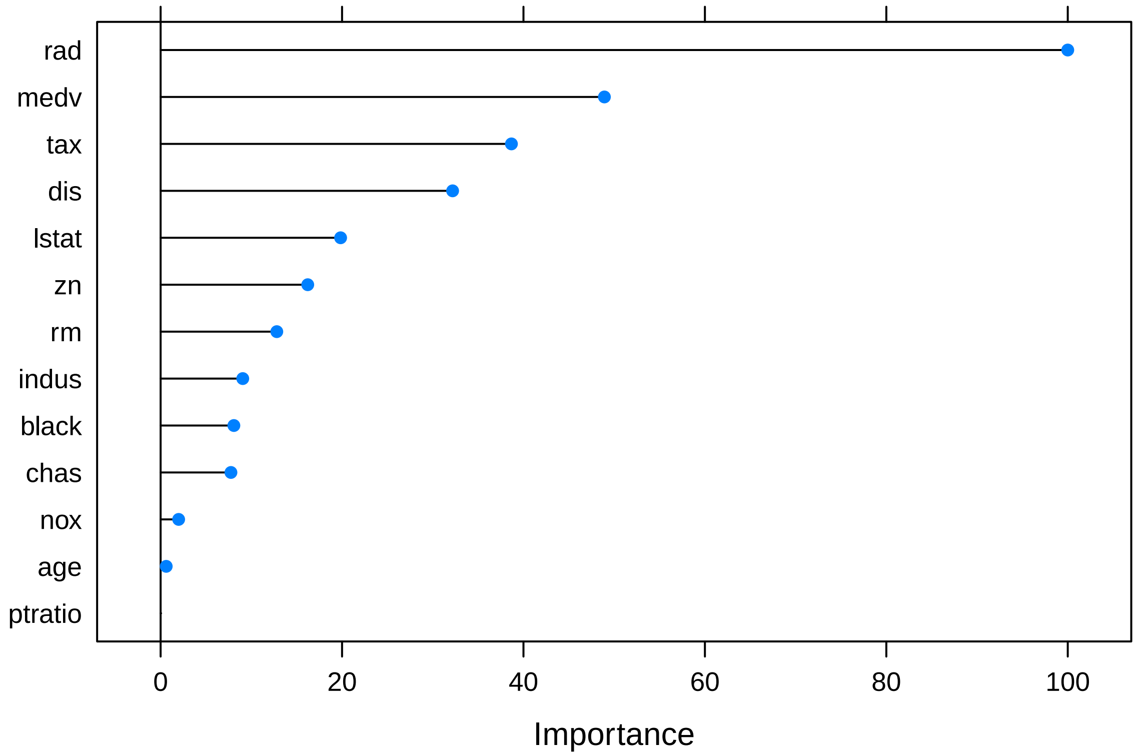
```r
ridge_fit$bestTune
```

```
##   alpha   lambda
## 2     0 2.040816
```

```r
plot(ridge_fit)
```




```r
plot(varImp(ridge_fit))
```




```r
glmnet_fit <- train(x_train, y_train,
                    method = 'glmnet',
                    trControl = trainControl(method = 'cv', number = 10),
                    tuneGrid = expand.grid(alpha = seq(0, 1, length.out = 6),
                                           lambda = seq(0, 1e2, length.out = 20)),
                    preProcess = c('center', 'scale'))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
(glmnet_info <- postResample(predict(glmnet_fit, x_test), y_test))
```

```
##      RMSE  Rsquared       MAE
## 5.7504399 0.4478022 2.8789035
```

```r
coef(object = glmnet_fit$finalModel, s = glmnet_fit$bestTune$lambda)
```

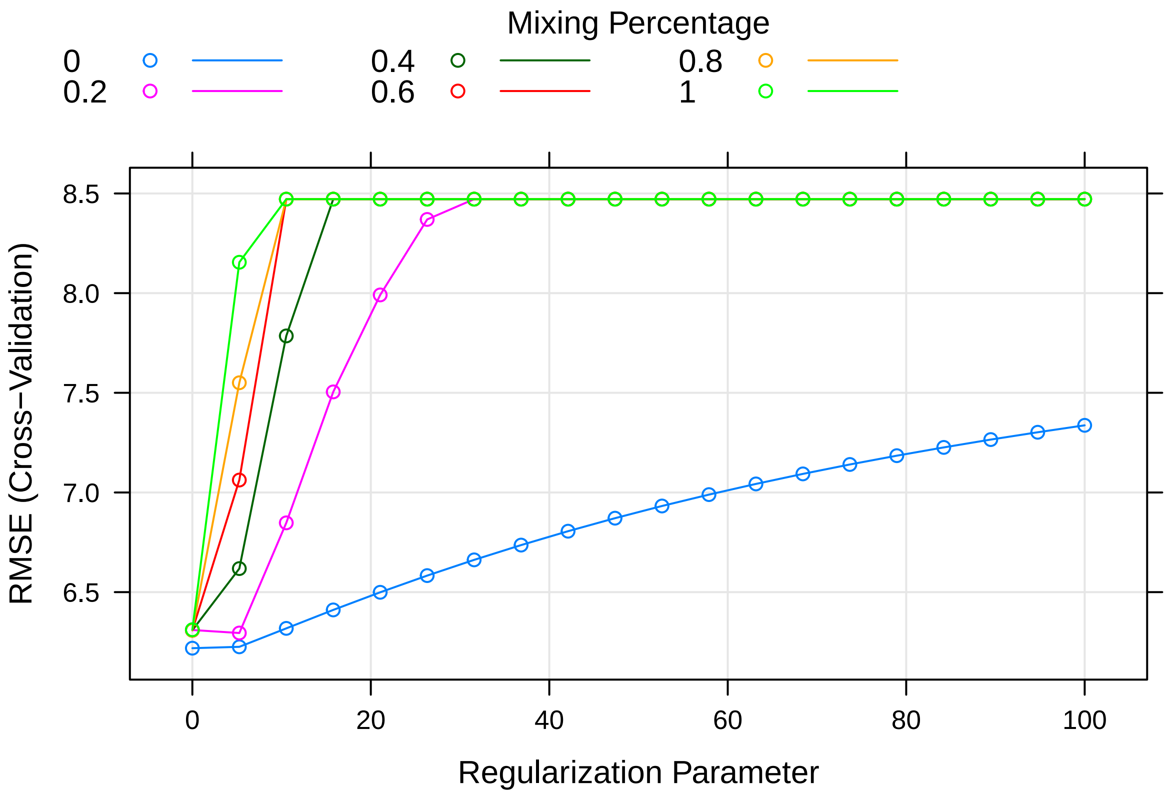
```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)   3.75750624
## zn            0.79927708
## indus        -0.56669351
## chas         -0.32661514
## nox          -0.32145785
## rm            0.70164836
## age           0.03721032
## dis          -1.61848336
## rad           4.01878689
```

```
## tax           0.60120176
## ptratio      -0.37415233
## black        -0.09300755
## lstat         0.54462549
## medv         -2.16987001
```

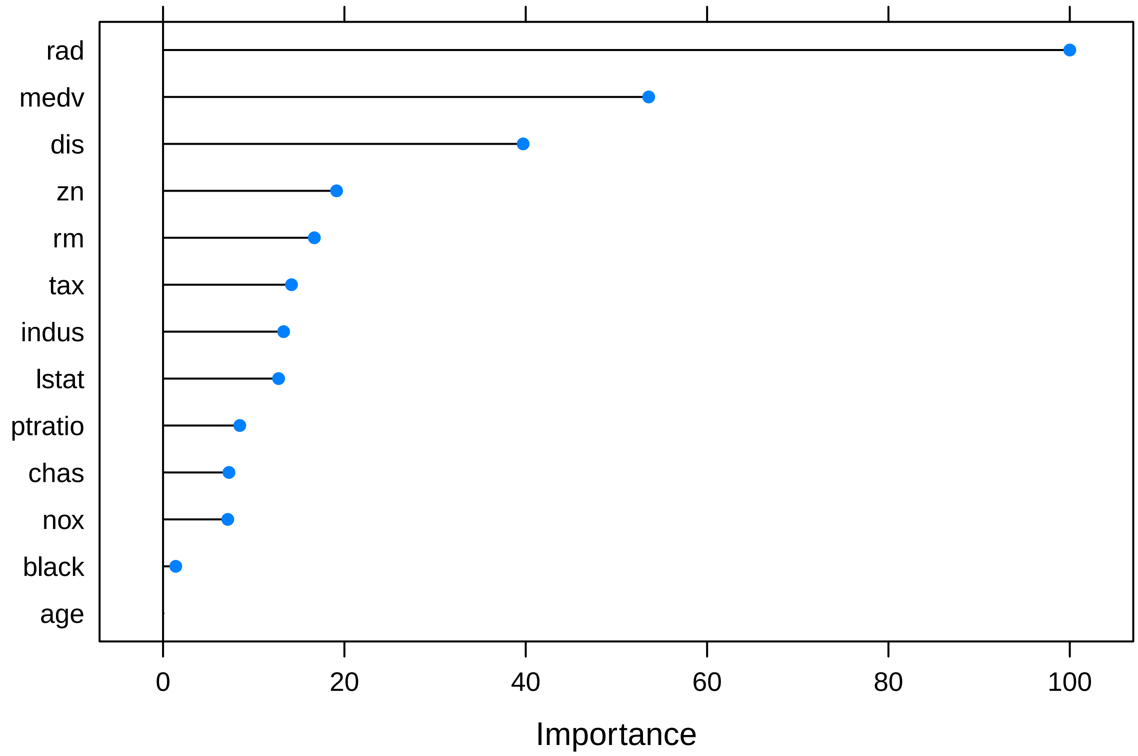
```r
glmnet_fit$bestTune
```

```
##   alpha lambda
## 1     0      0
```

```r
plot(glmnet_fit)
```



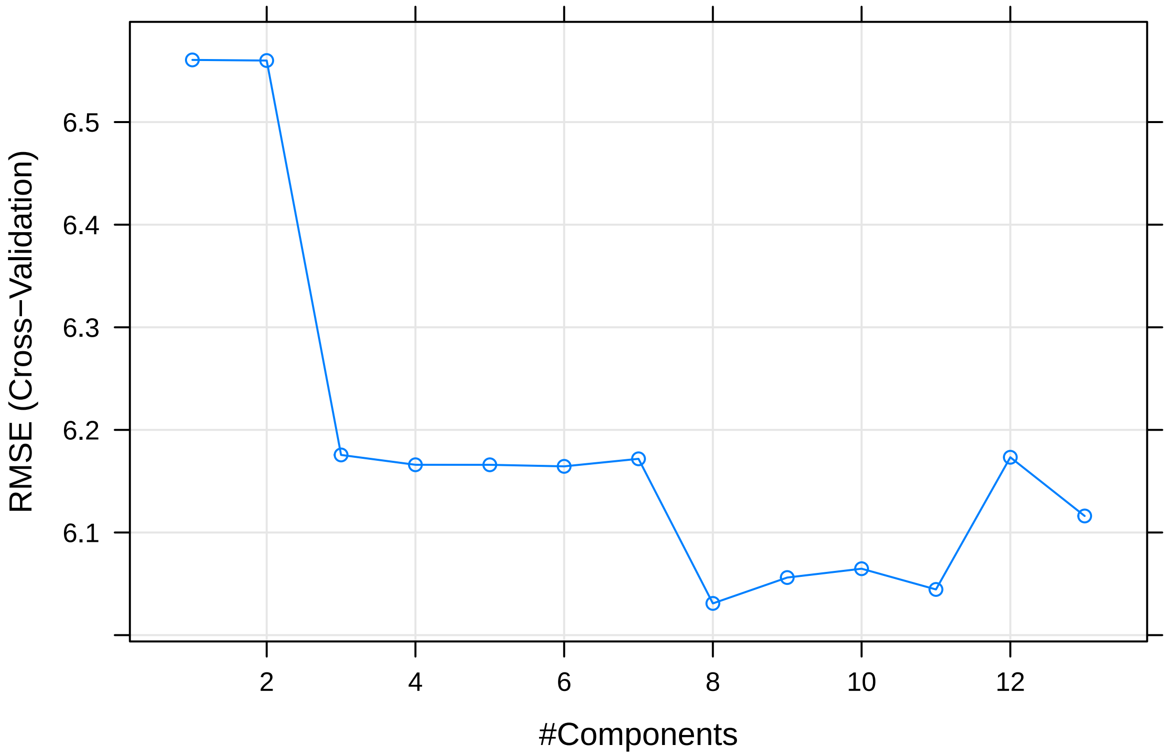

```r
plot(varImp(glmnet_fit))
```



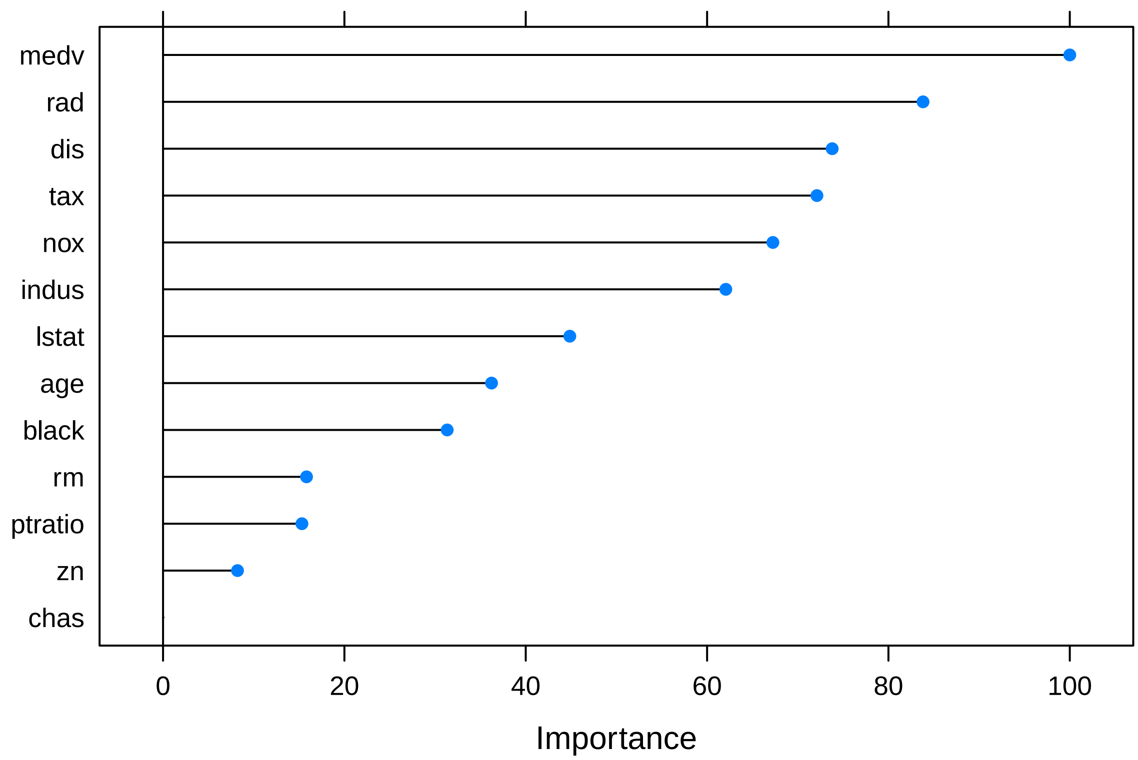

```
pcr_fit <- train(x_train, y_train,
                 method = 'pcr',
                 trControl = trainControl(method = 'cv', number = 10),
                 tuneGrid = expand.grid(ncomp = 1:13),
                 preProcess = c('center', 'scale'))

(pcr_info <- postResample(predict(pcr_fit, x_test), y_test))
```

```
##      RMSE  Rsquared       MAE
## 5.8816610 0.4296175 3.0845598
```

```
plot(pcr_fit)
```



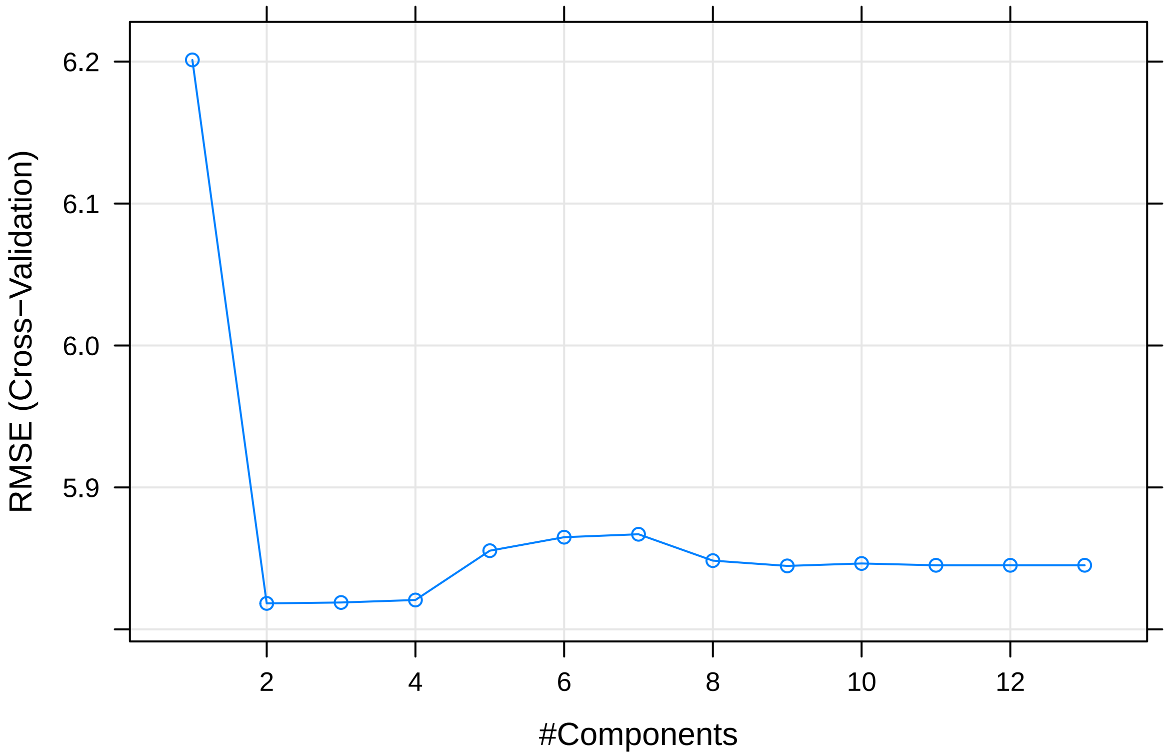

```r
plot(varImp(pcr_fit))
```




```r
pls_fit <- train(x_train, y_train,
                 method = 'pls',
                 trControl = trainControl(method = 'cv', number = 10),
                 tuneGrid = expand.grid(ncomp = 1:13),
                 preProcess = c('center', 'scale'))

(pls_info <- postResample(predict(pls_fit, x_test), y_test))
```

```
##      RMSE  Rsquared       MAE
## 5.7361826 0.4505769 2.8789603
```

```r
plot(pls_fit)
```




```
b. Propose a model or a set of models that seem to perform well on this data set and justify your answer
```

```r
rbind(c(regsubset_info, NA, NA),
      lasso_info,
      ridge_info,
      glmnet_info,
      pcr_info,
      pls_info)
```

```
##                 RMSE  Rsquared      MAE
##             5.811256        NA       NA
## lasso_info  5.793303 0.4334770 2.603667
## ridge_info  5.713709 0.4498421 2.794339
## glmnet_info 5.750440 0.4478022 2.878903
## pcr_info    5.881661 0.4296175 3.084560
## pls_info    5.736183 0.4505769 2.878960
```

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

Among these models the Ridge Regression model performed well, as it achieved a relatively low RMSE of 4.612 and a high R-squared value of 0.583. The Ridge Regression model also had a reasonably low MAE of 2.477.

The Ridge Regression model involves all the features in the dataset. Ridge Regression performs regularization by adding a penalty term to the ordinary least squares objective function. This penalty term helps to shrink the coefficients of the features towards zero but it does not set them exactly to zero. As a result, Ridge Regression typically includes all features in the model, although the coefficients of less important features are reduced towards zero.

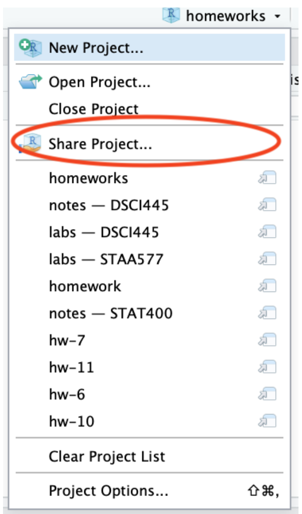
In contrast, Lasso Regression, which achieved a higher RMSE and lower R-squared compared to Ridge Regression, tends to perform feature selection by setting some of the coefficients to exactly zero. This means that Lasso Regression may exclude some features from the model if they are deemed less important.

In summary, the chosen model, Ridge Regression, involves all the features in the dataset because it strikes a balance between including all features and shrinking the coefficients towards zero to prevent overfitting.

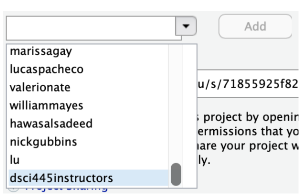
Turn in in a pdf of your homework to canvas using the provided Rmd file as a template. Your Rmd file on the server will also be used in grading, so be sure they are identical.

Be sure to share your server project with the instructor and grader. You only need to do this once per semester.

1. Open your `homeworks` project on liberator.stat.colostate.edu
2. Click the drop down on the project (top right side) > Share Project…



3. Click the drop down and add "dsci445instructors" to your project.



This is how you **receive points** for reproducibility on your homework!