STOR 565 Spring 2019 Homework 3

Coleman Breen

Remark. This homework aims to help you further understand the model selection techniques in linear model. Credits for **Theoretical Part** and **Computational Part** are in total 100 pt. For **Computational Part** , please complete your answer in the **RMarkdown** file and summit your printed PDF homework created by it.

Computational Part

Hint. Before starting your work, carefully read Textbook Chapter 6.5-6.7 (Lab 1-3). Mimic the related analyses you learn from it. Related packages have been loaded in setup.

1. (Model Selection, Textbook 6.8, *18 pt*) In this exercise, we will generate simulated data, and will then use this data to perform model selection.

a. Use the `rnorm` function to generate a predictor $\bm X$ of length $n = 100$, as well as a noise vector $\bm \epsilon$ of length $n = 100$. Do not print the entire vector.

```
set.seed(919) # Raleigh
x <- rnorm(100)
error <- rnorm(100)
```

Hint. Before generating random numbers, fix your favourite random seed by `set.seed` so that your result is reproducible as you carry forward your exploration.

b. Generate a response vector $\bm Y$ of length $n = 100$ according to the model

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon,$$

where $\beta_0 = 3$, $\beta_1 = 2$, $\beta_2 = -3$, $\beta_3 = 0.3$. Do not print the entire vector.

```
y <- 3 + 2*x - 3*(x^2) + .3*(x^3) + error
head(y)
```

```
## [1]  -0.06476829   2.44988692 -11.64387519   3.10499851   0.89510639
## [6]  -1.34996850
```

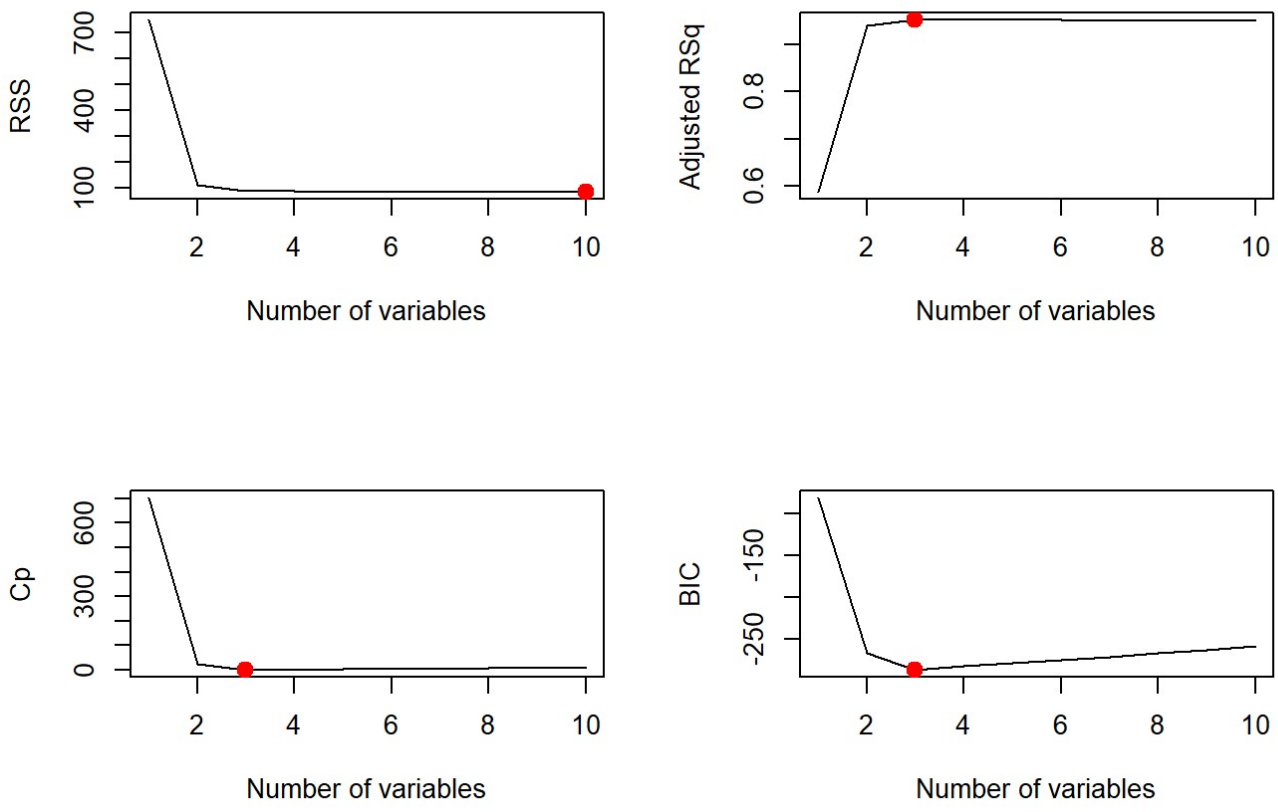
c. Use the `regsubsets` function from `leaps` package to perform best subset selection in order to choose the best model containing the predictors $(X, X^2, \cdots, X^{10})$.

What is the best model obtained according to C_p, BIC, and adjusted R^2? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained.

```
#--> Create a data frame with x, x^2, x^3,...
data2 <- as.data.frame(cbind(y, x)) %>%
  mutate(x2=x^2, x3=x^3, x4=x^4, x5=x^5, x6=x^6, x7=x^7, x8=x^8, x9=x^9, x10=x^10)

#--> Plug in with nvmax=10
regfit.full <- regsubsets(y~., data=data2, nvmax=10)
regfit.summary <- summary(regfit.full)

regfit.summary
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data2, nvmax = 10)
## 10 Variables  (and intercept)
##     Forced in Forced out
## x       FALSE      FALSE
## x2      FALSE      FALSE
## x3      FALSE      FALSE
## x4      FALSE      FALSE
## x5      FALSE      FALSE
## x6      FALSE      FALSE
## x7      FALSE      FALSE
## x8      FALSE      FALSE
## x9      FALSE      FALSE
## x10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           x   x2  x3  x4  x5  x6  x7  x8  x9  x10
## 1  ( 1 )  " " "*" " " " " " " " " " " " " " " " "
## 2  ( 1 )  "*" "*" " " " " " " " " " " " " " " " "
## 3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " "
## 4  ( 1 )  "*" "*" "*" " " " " " " " " " " "*" " "
## 5  ( 1 )  "*" "*" "*" "*" " " "*" " " " " " " " "
## 6  ( 1 )  "*" "*" " " "*" "*" " " "*" " " "*" " "
## 7  ( 1 )  "*" "*" "*" "*" " " "*" " " "*" " " "*"
## 8  ( 1 )  "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 9  ( 1 )  "*" "*" " " "*" "*" "*" "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
#--> Create plots function
plotFour <- function(summary) {
  par(mfrow=c(2,2))

  plot(summary$rss, xlab="Number of variables", ylab="RSS", type="l")
  points(which.min(summary$rss), summary$rss[which.min(summary$rss)],
         col="red", cex=2,pch=20)

  plot(summary$adjr2, xlab="Number of variables", ylab="Adjusted RSq", type="l")
  points(which.max(summary$adjr2), summary$adjr2[which.max(summary$adjr2)],
         col="red", cex=2,pch=20)

  plot(summary$cp, xlab="Number of variables", ylab="Cp", type="l")
  points(which.min(summary$cp), summary$cp[which.min(summary$cp)],
         col="red", cex=2,pch=20)

  plot(summary$bic, xlab="Number of variables", ylab="BIC", type="l")
  points(which.min(summary$bic), summary$bic[which.min(summary$bic)],
         col="red", cex=2,pch=20)
}

#--> Plot
plotFour(regfit.summary)
```



```
#--> 3 variable model
coef(regfit.full, 3)
```

```
## (Intercept)           x          x2          x3
##   3.1143667   2.0089255  -3.0078037   0.3165343
```

We looked at four different metrics to test how many variables to include in our model. The first, R^2, predictably gives that 10 variables is the best model. As we know, R^2 by itself does not account for model complexity, so more variables will always be better because they include more degrees of freedom. Therefore, the other three metrics do a better job of helping us find a solid model. Adjusted R^2, C_p and the Bayesian Information Criterion (BIC) all support that we should include 3 variables in our model.

In our model with 3 coefficients, we get $\hat{Y}$ = 3.11 + 2.01X - 3.01X^2 + 0.32X^3 + ϵ

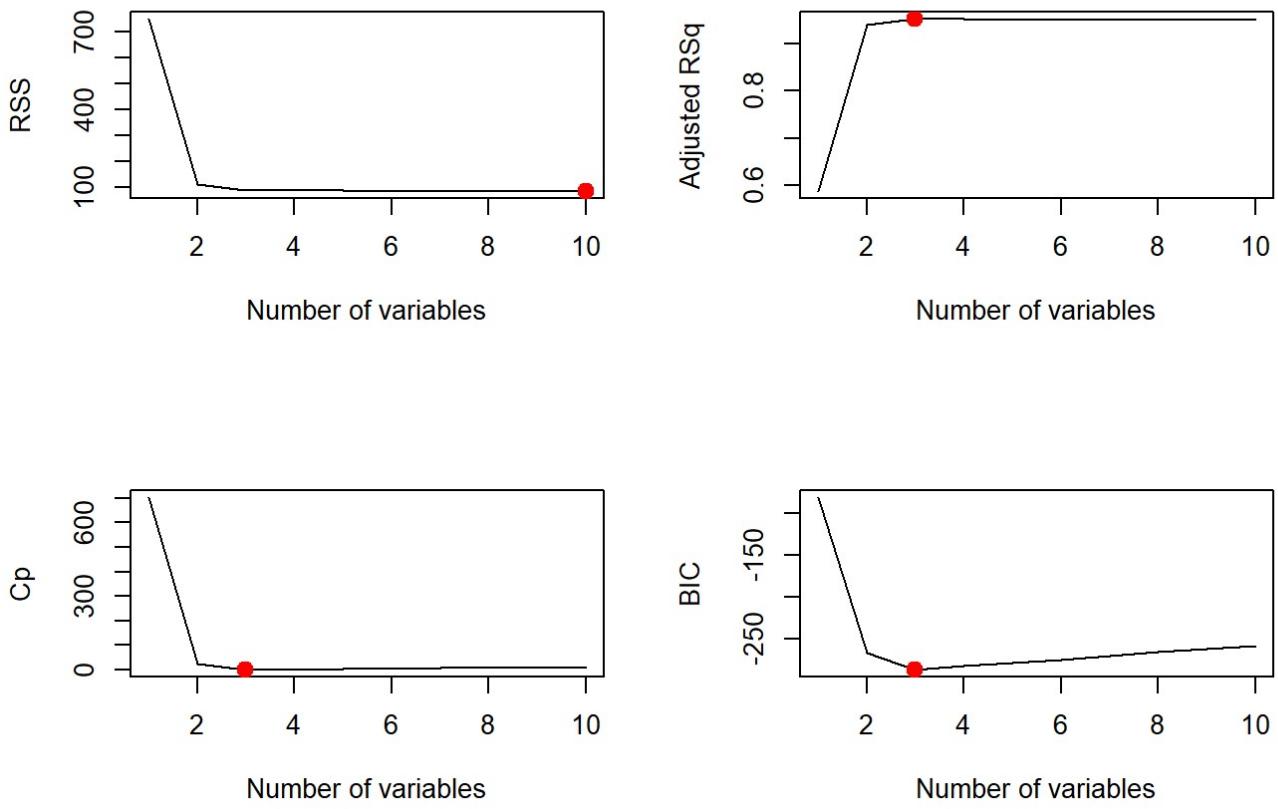
Note that we do not need to split into train and test sets when doing best subset selection.

d. Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)? You must show a summary of the selected model or other evidence to support your statements.

```
#--> Plug in forward stepwise with nvmax=10
fwdfit.full <- regsubsets(y~., data=data2, nvmax=10, method="forward")
fwdfit.summary <- summary(fwdfit.full)

fwdfit.summary
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data2, nvmax = 10, method = "forward")
## 10 Variables  (and intercept)
##     Forced in Forced out
## x       FALSE      FALSE
## x2      FALSE      FALSE
## x3      FALSE      FALSE
## x4      FALSE      FALSE
## x5      FALSE      FALSE
## x6      FALSE      FALSE
## x7      FALSE      FALSE
## x8      FALSE      FALSE
## x9      FALSE      FALSE
## x10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: forward
##           x   x2  x3  x4  x5  x6  x7  x8  x9  x10
## 1  ( 1 )  " " "*" " " " " " " " " " " " " " " " "
## 2  ( 1 )  "*" "*" " " " " " " " " " " " " " " " "
## 3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " "
## 4  ( 1 )  "*" "*" "*" " " " " " " " " " " "*" " "
## 5  ( 1 )  "*" "*" "*" " " "*" " " " " " " "*" " "
## 6  ( 1 )  "*" "*" "*" "*" "*" " " " " " " "*" " "
## 7  ( 1 )  "*" "*" "*" "*" "*" " " "*" " " "*" " "
## 8  ( 1 )  "*" "*" "*" "*" "*" " " "*" " " "*" "*"
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" " " "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
#--> Create plots to find which model is best
plotFour(fwdfit.summary)
```




```
coef(fwdfit.full, 3)
```

```
## (Intercept)           x          x2          x3
##   3.1143667   2.0089255  -3.0078037   0.3165343
```

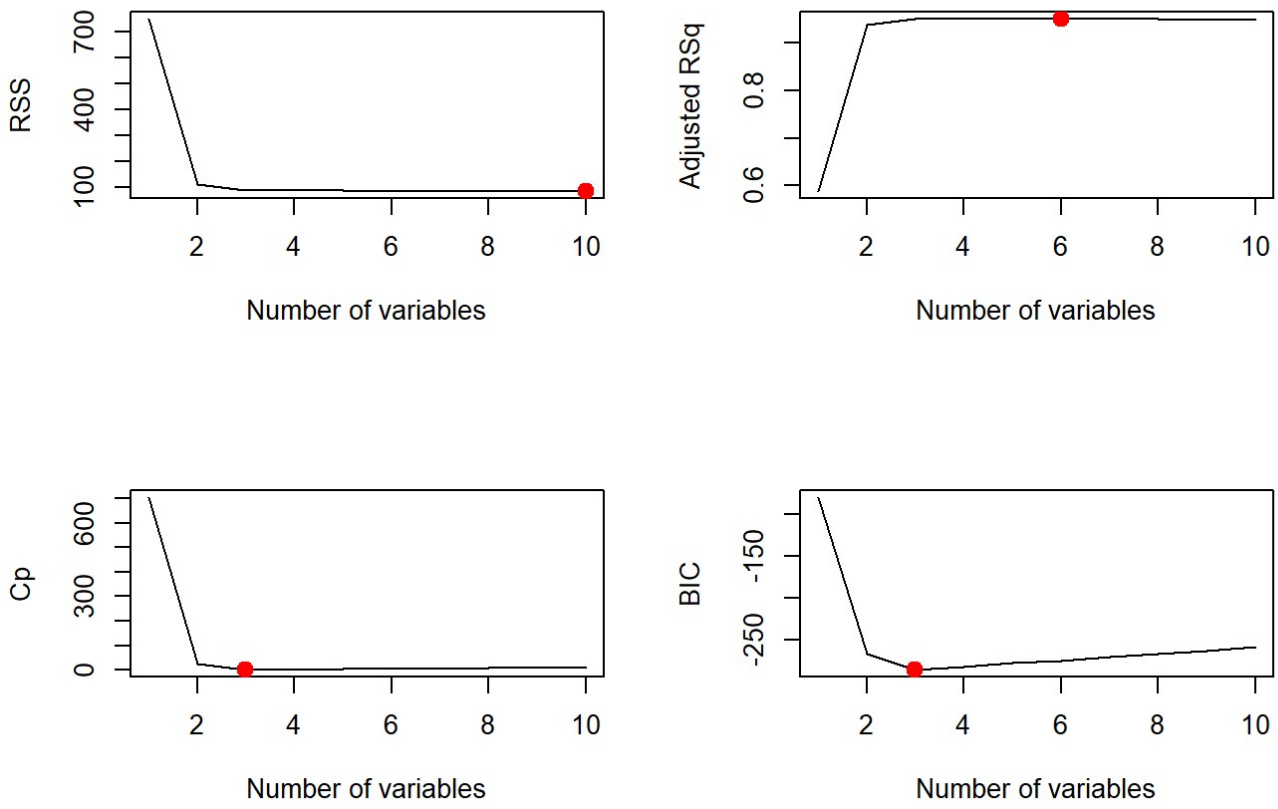
Like in the last method, forward stepwise selection shows that we should include 3 variables. We actually get the same coefficients as before because the optimum linear model is unique when there are more observations that there are predictors.

In our model with 3 coefficients, we get (for the second time) $\hat{Y}$ = 3.11 + 2.01X - 3.01X^2 + 0.32X^3 + ϵ

```
#--> Plug in backward with nvmax=10
backfit.full <- regsubsets(y~., data=data2, nvmax=10, method="backward")
backfit.summary <- summary(backfit.full)

backfit.summary
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data2, nvmax = 10, method = "backward")
## 10 Variables  (and intercept)
##     Forced in Forced out
## x       FALSE      FALSE
## x2      FALSE      FALSE
## x3      FALSE      FALSE
## x4      FALSE      FALSE
## x5      FALSE      FALSE
## x6      FALSE      FALSE
## x7      FALSE      FALSE
## x8      FALSE      FALSE
## x9      FALSE      FALSE
## x10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: backward
##           x   x2  x3  x4  x5  x6  x7  x8  x9  x10
## 1  ( 1 )  " " "*" " " " " " " " " " " " " " " " "
## 2  ( 1 )  "*" "*" " " " " " " " " " " " " " " " "
## 3  ( 1 )  "*" "*" " " " " " "*" " " " " " " " " "
## 4  ( 1 )  "*" "*" " " " " " "*" " " " "*" " " " "
## 5  ( 1 )  "*" "*" " " " " " "*" " " " "*" " " "*" " "
## 6  ( 1 )  "*" "*" " " " "*" "*" " " " "*" " " "*" " "
## 7  ( 1 )  "*" "*" " " " "*" "*" " " " "*" "*" "*" " "
## 8  ( 1 )  "*" "*" " " " "*" "*" "*" "*" "*" "*" " "
## 9  ( 1 )  "*" "*" " " " "*" "*" "*" "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
plotFour(backfit.summary)
```




```
coef(backfit.full, 3)
```

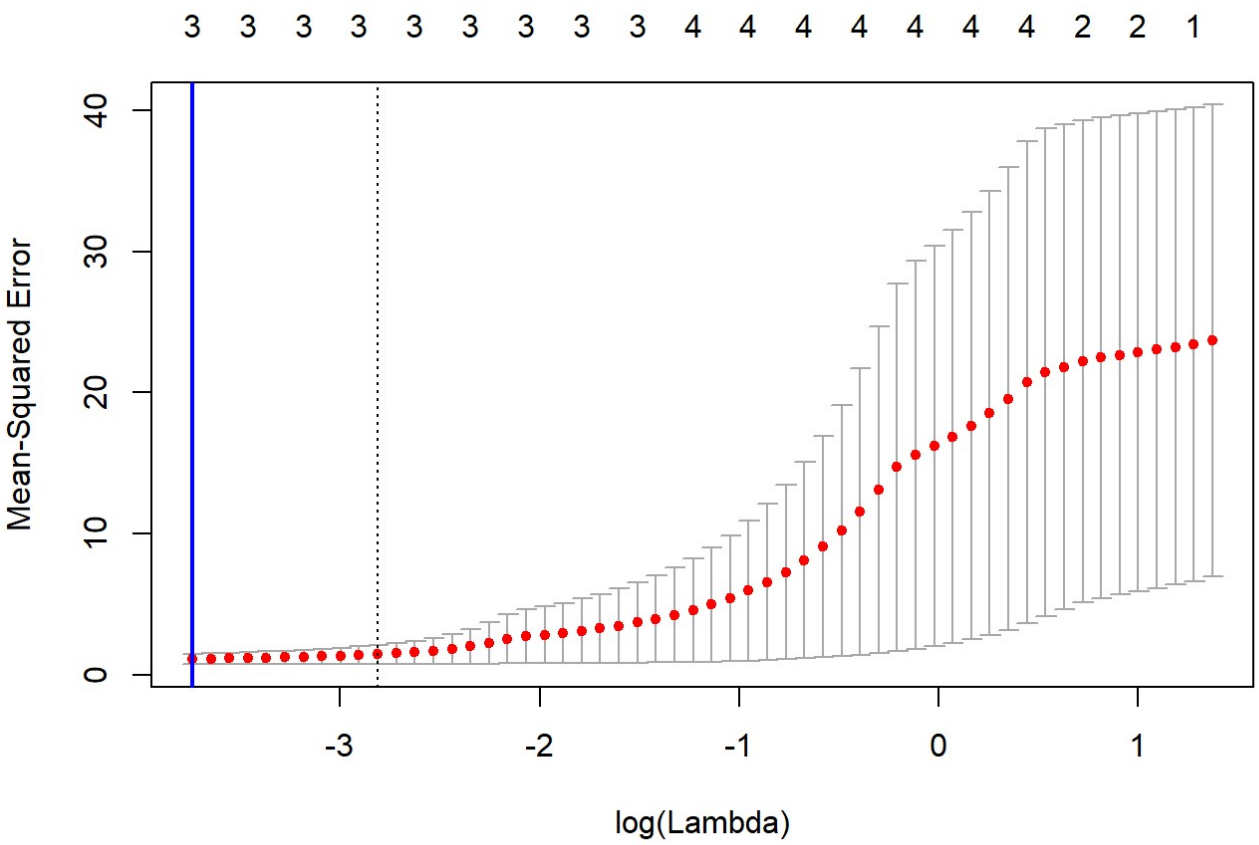
```
## (Intercept)           x          x2          x5
##  3.10746815  2.35971773 -2.97663604  0.04515634
```

The backward stepwise selection model creates some trouble. In its first round of pruning, it eliminates X^3 as a variable. Because these data are simulated, we know that X^3 is part of our "true"system, but this method cut it out early on. As a result, the three metrics that we're interested in (Adjusted R^2, C_p, and BIC) give different results. Adjusted R^2 wants us to use 6 variables whereas the other two want us to use 3. I'll use three variables in my final model because that's what the majority of the metrics suggest.

The three variable model is $\hat{Y}$ = 3.11 + 2.31X - 2.98X^2 + 0.05X^5 + ϵ

e. Now fit a LASSO model with `glmnet` function from `glmnet` package to the simulated data, again using $(X, X^2, \cdots, X^{10})$ as predictors. Use 5-fold cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained.

```
#--> Coerce data
set.seed(919) # Represent
train <- sample(c(T, F), nrow(data2), rep=T)
test <- (!train)

x <- as.matrix(select(data2, -y))
y <- as.matrix(select(data2, y))

#--> Cross validate for best lambda
lasso.cv <- cv.glmnet(x[train, ], y[train, ], nfolds=5, alpha=1) # 1 for LASSO

#--> Plot lambda vs MSE
plot(lasso.cv)
abline(v=log(lasso.cv$lambda.min), col="blue", lwd=2)
```




```
#--> What is the best lambda?
lasso.cv$lambda.min
```

```
## [1] 0.02369704
```

```
#--> What is our MSPE
lasso.pred <- predict(lasso.cv, s=lasso.cv$lambda, newx=x[test, ])
mean((lasso.pred - y[test])^2)
```

```
## [1] 3.441524
```

```
#--> Fit model to whole data set
full.lasso <- glmnet(x,y,alpha=1,lambda=lasso.cv$lambda.min)
coef(full.lasso)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  3.0935694965
## x            2.0218351369
## x2          -2.9766794689
## x3           0.2895219841
## x4           .
## x5           .
## x6           .
## x7           .
## x8           .
## x9           0.0001001679
## x10          .
```

The choice of lambda in our LASSO regression that minimizes mean squared error in our 5-fold cross validation is λ = 0.024.

The resulting model is $\hat{Y}$ = 3.09 + 2.02X - 2.98X^2 + 0.0001X^9 + ϵ

f. Now generate a response vector Y according to the model

$$Y = \beta_0 + \beta_7 X^7 + \epsilon,$$

where $\beta_7 = 7$, and perform best subset selection and the LASSO. Discuss the results obtained.

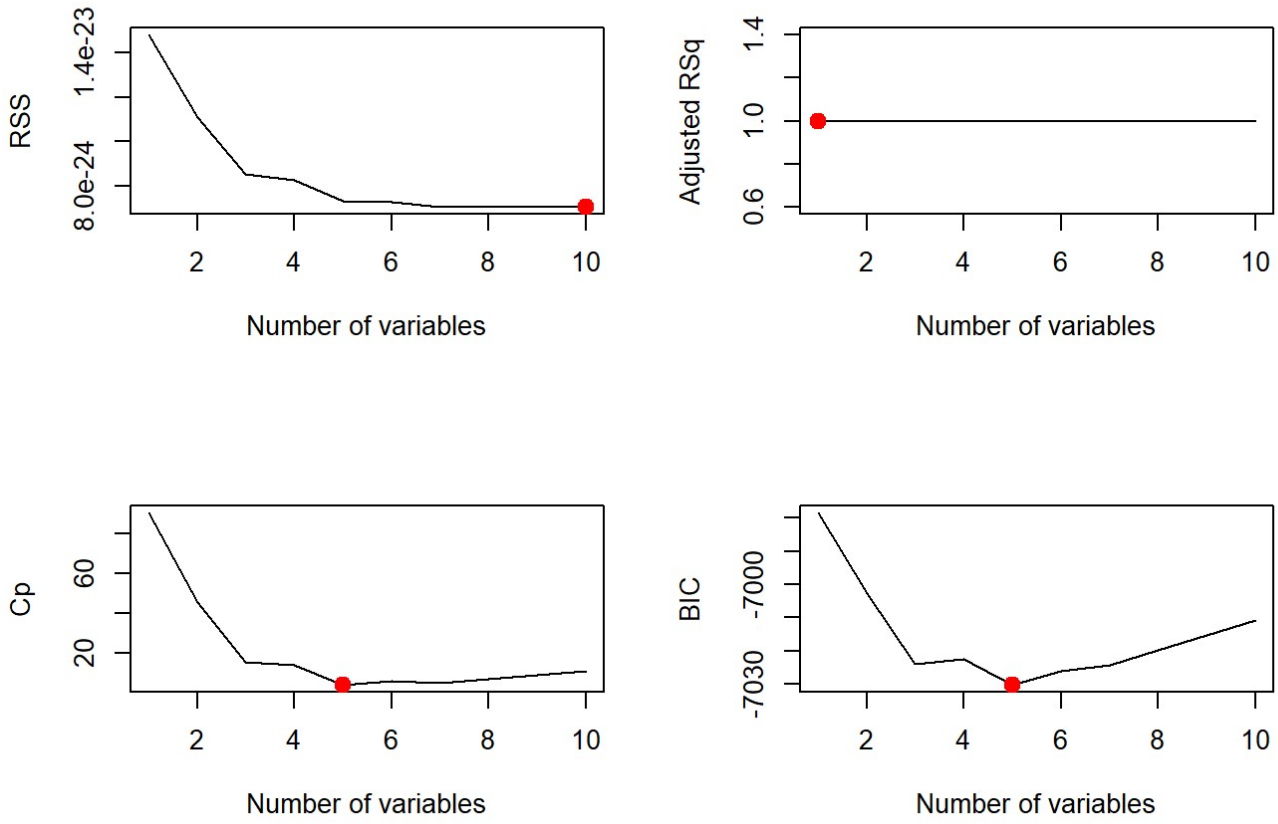
```
#--> Generate new y
data3 <- data2 %>%
  mutate(y = 7*x7+3)

head(data3)
```

```
##             y          x        x2           x3           x4            x5
## 1  183.182494  1.5904366 2.5294886  4.022991308 6.3983126669  1.017611e+01
## 2    3.133700  0.5681115 0.3227506  0.183358340 0.1041679759  5.917902e-02
## 3 -312.765379 -1.7231528 2.9692555 -5.116480757 8.8164779718 -1.519214e+01
## 4    2.999981 -0.1598950 0.0255664 -0.004087938 0.0006536406 -1.045138e-04
## 5   88.321843  1.4293463 2.0430308  2.920198446 4.1739747809  5.966055e+00
## 6    1.396166 -0.8101776 0.6563877 -0.531790583 0.4308448011 -3.490608e-01
##             x6            x7           x8            x9          x10
## 1 1.618446e+01  2.574036e+01 4.093840e+01  6.510994e+01 1.035532e+02
## 2 3.362028e-02  1.910007e-02 1.085097e-02  6.164559e-03 3.502157e-03
## 3 2.617838e+01 -4.510934e+01 7.773028e+01 -1.339412e+02 2.308011e+02
## 4 1.671124e-05 -2.672042e-06 4.272461e-07 -6.831449e-08 1.092314e-08
## 5 8.527559e+00  1.218883e+01 1.742207e+01  2.490216e+01 3.559382e+01
## 6 2.828012e-01 -2.291192e-01 1.856272e-01 -1.503910e-01 1.218434e-01
```

```
#--> Best subset selection
regfit.full2 <- regsubsets(y~., data=data3, nvmax=10)
regfit.sum2 <- summary(regfit.full2)

regfit.sum2
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data3, nvmax = 10)
## 10 Variables  (and intercept)
##     Forced in Forced out
## x       FALSE      FALSE
## x2      FALSE      FALSE
## x3      FALSE      FALSE
## x4      FALSE      FALSE
## x5      FALSE      FALSE
## x6      FALSE      FALSE
## x7      FALSE      FALSE
## x8      FALSE      FALSE
## x9      FALSE      FALSE
## x10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           x   x2  x3  x4  x5  x6  x7  x8  x9  x10
## 1  ( 1 )  " " " " " " " " " " " " "*" " " " " " "
## 2  ( 1 )  " " " " " " " " " " " " "*" " " "*" " "
## 3  ( 1 )  "*" " " " " " " " " " " "*" " " "*" " "
## 4  ( 1 )  "*" "*" "*" " " " " " " " "*" " " " " " "
## 5  ( 1 )  "*" " " " " " " " " " " "*" "*" "*" "*" " "
## 6  ( 1 )  "*" " " " " " " " " " " "*" "*" "*" "*" "*"
## 7  ( 1 )  "*" " " " " " "*" " " " "*" "*" "*" "*" "*"
## 8  ( 1 )  "*" " " " " " "*" "*" "*" "*" "*" "*" "*"
## 9  ( 1 )  "*" "*" " " " "*" "*" "*" "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
plotFour(regfit.sum2)
```




```
print(list(c("With 5 predictors... ", coef(regfit.full2, 5))))
```

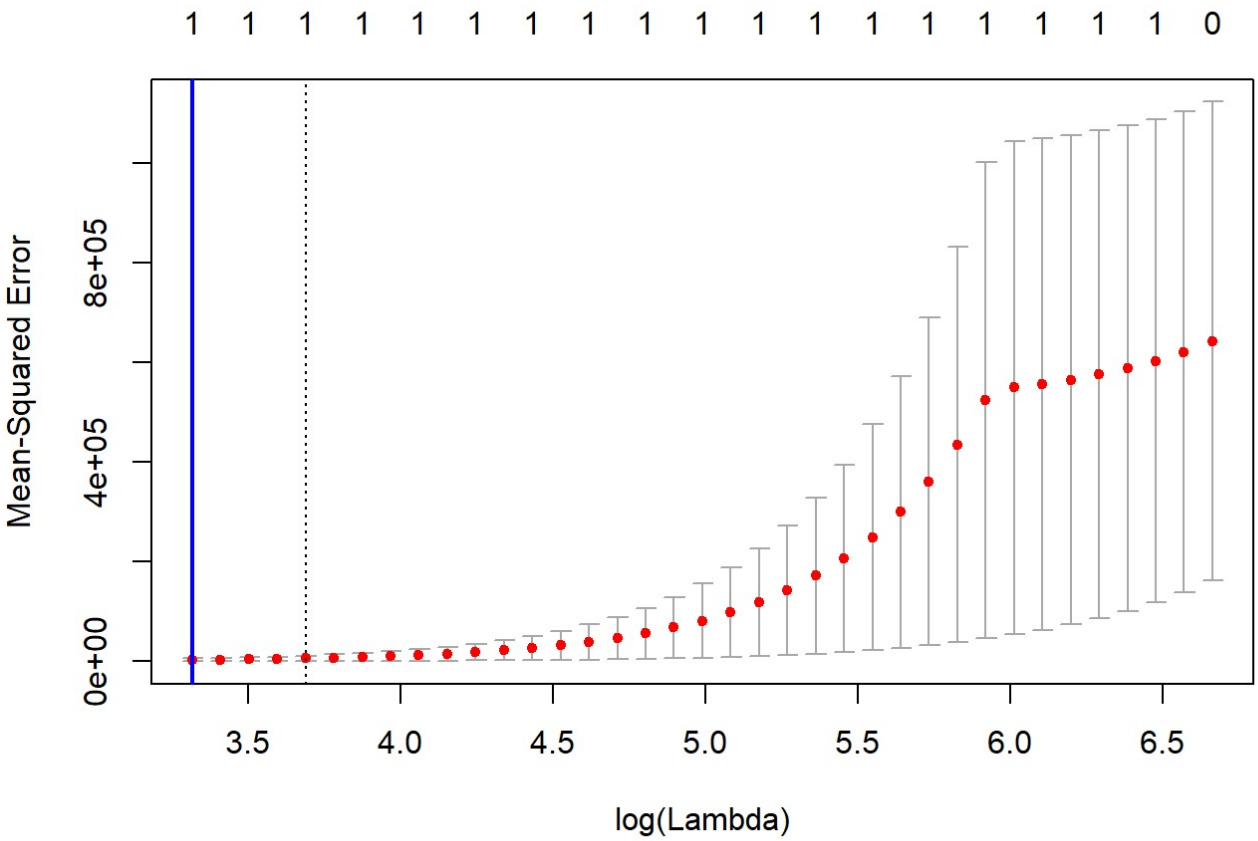
```
## [[1]]
##                                  (Intercept)                           x
## "With 5 predictors... "      "3.00000000000006"  "2.05264103612812e-13"
##                        x6                     x7                     x8
## "-4.89043987960847e-14"      "6.99999999999995"  "1.00018045749077e-14"
##                        x9
##  "8.09757574514965e-15"
```

```
print(list(c("With 1 predictor... ", coef(regfit.full2, 1))))
```

```
## [[1]]
##                                 (Intercept)                      x7
## "With 1 predictor... "     "2.99999999999998"                    "7"
```

We are getting quite a bit of ambiguity by the best subset method alone. In our usual three metrics, we are seeing that we should include one (Adjusted R^2) or five (BIC and C_p) variables in our model. We'll look at the LASSO results before we make a decision.

```
set.seed(919) # Represent
train <- sample(c(T, F), nrow(data3), rep=T)
test <- (!train)

x <- as.matrix(select(data3, -y))
y <- as.matrix(select(data3, y))

#--> Cross validate for best lambda
lasso.cv <- cv.glmnet(x[train, ], y[train, ], nfolds=5, alpha=1) # 1 for LASSO

#--> Plot lambda vs MSE
plot(lasso.cv)
abline(v=log(lasso.cv$lambda.min), col="blue", lwd=2)
```




```
#--> What is the best lambda?
lasso.cv$lambda.min
```

```
## [1] 27.52045
```

```
#--> Fit model to whole data set
full.lasso <- glmnet(x,y,alpha=1,lambda=lasso.cv$lambda.min)
coef(full.lasso)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## (Intercept) 2.063382
## x           .
## x2          .
## x3          .
## x4          .
## x5          .
## x6          .
## x7          6.664046
## x8          .
## x9          .
## x10         .
```

Recall that our "ground truth" is the model Y = 3 + 7X^7. Now, out first round of testing, where we perform best subset selection, does not help us discern how many variables to include (one and five are both candidates). Five-fold cross validation gives the optimum λ = 27.52 for our penalty term. Using that, we fit a model that includes one non-zero coefficient for our predictors. Our final LASSO model is $\hat{y}$ = 2.06 + 6.66X^7. Our final model from best subset selection with one predictor is $\hat{y}$ = 3 + 7X^7 (Wow!).

Note that I assumed $\beta_0 = 3$ as in the original setup.

---

2. (Prediction, *20 pt*) In this exercise, we will try to develop a prediction model for wins in a basketball season for a team based on a host of other factors. The starting point is to load the nba-teams-2017 data set (which was scraped by Gaston Sanchez at Berkeley).

a. Do some exploratory data analysis by picking 6-7 features that you think might be interesting and explore relationship between these features by making a scatterplot matrix like the following (you **do not** have to use the same features I am using!):

```
nba.data <- read.csv("nba-teams-2017.csv")

head(nba.data)
```

```
##                  team games_played wins losses win_prop minutes points
## 1 Golden State Warriors           82   67     15    0.817    48.2  115.9
## 2     San Antonio Spurs           82   61     21    0.744    48.3  105.3
## 3       Houston Rockets           82   55     27    0.671    48.2  115.3
## 4        Boston Celtics           82   53     29    0.646    48.2  108.0
## 5             Utah Jazz           82   51     31    0.622    48.2  100.7
## 6       Toronto Raptors           82   51     31    0.622    48.2  106.9
##   field_goals field_goals_attempted field_goals_prop points3
## 1        43.1                 87.1             49.5    12.0
## 2        39.3                 83.7             46.9     9.2
## 3        40.3                 87.2             46.2    14.4
## 4        38.6                 85.1             45.4    12.0
## 5        37.0                 79.5             46.6     9.6
## 6        39.2                 84.4             46.4     8.8
##   points3_attempted points3_prop free_throws free_throws_att
## 1              31.2         38.3        17.8            22.6
## 2              23.5         39.1        17.6            22.0
## 3              40.3         35.7        20.3            26.5
## 4              33.4         35.9        18.7            23.2
## 5              26.0         37.2        17.1            22.9
## 6              24.3         36.3        19.7            24.7
##   free_throws_prop off_rebounds def_rebounds rebounds assists turnovers
## 1             78.8          9.4         35.0     44.4    30.4      14.8
## 2             79.7         10.0         33.9     43.9    23.8      13.4
## 3             76.6         10.9         33.5     44.4    25.2      15.1
## 4             80.7          9.1         32.9     42.0    25.2      13.3
## 5             74.7          9.4         33.8     43.2    20.1      13.6
## 6             79.6         10.6         32.6     43.3    18.5      12.7
##   steals blocks block_fga personal_fouls personal_fouls_drawn plus_minus
## 1    9.6    6.8       3.8           19.3                19.4       11.6
## 2    8.0    5.9       4.1           18.3                19.8        7.2
## 3    8.2    4.3       5.0           19.9                20.4        5.8
## 4    7.5    4.1       5.2           20.6                20.3        2.6
## 5    6.7    5.0       3.8           18.8                20.2        3.9
## 6    8.3    4.9       4.8           20.8                20.3        4.2
```

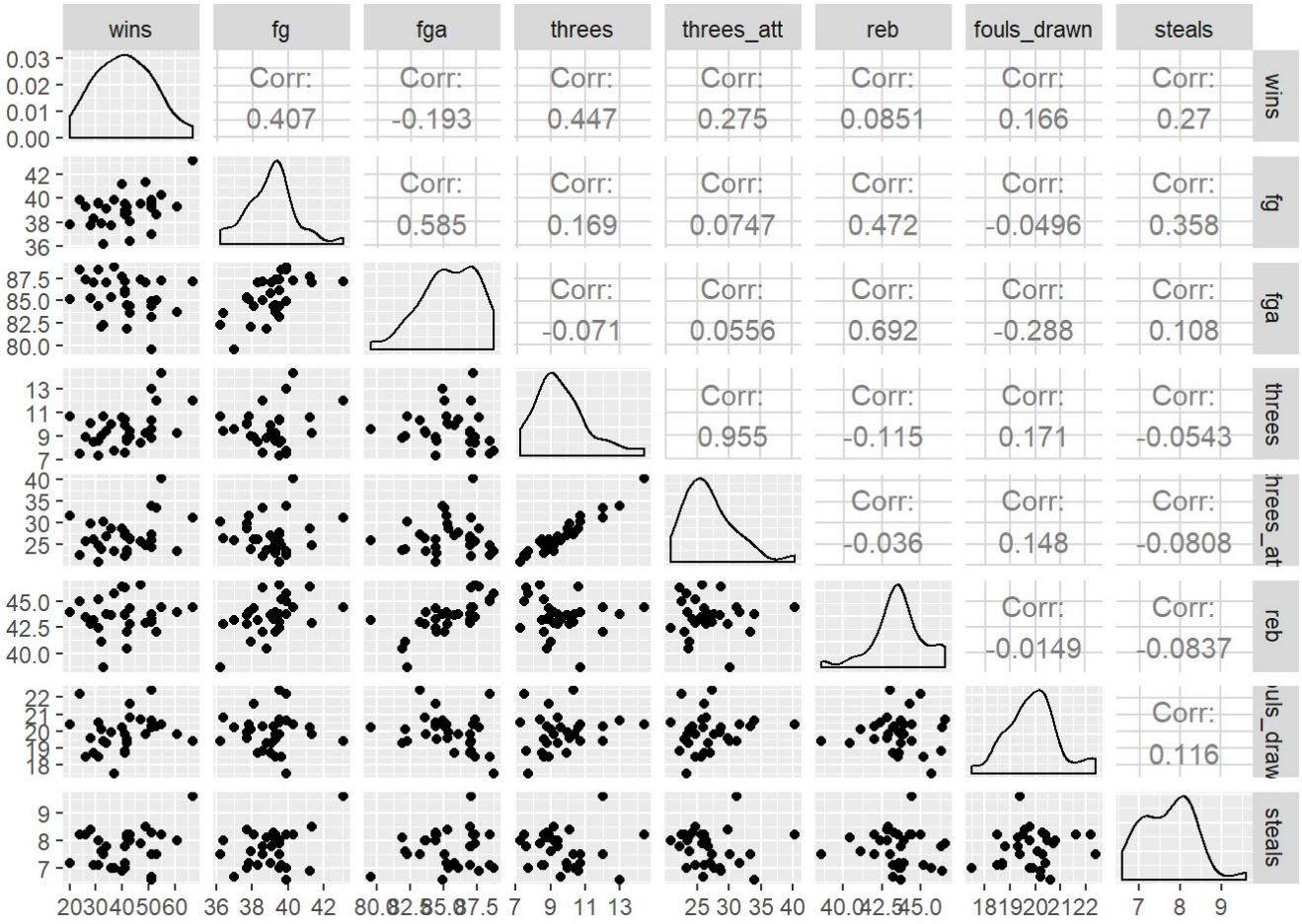
NOTE: You may remove the includegraphics statements below when knitting your own response, if they are giving you trouble

b. The aim is now to predict *wins* based on the other features. First explain why you would remove the "losses" column from the above data set? Would you necessarily remove any other columns?

I am removing `losses` because it has a deterministic relationship with `wins` (`wins` = 82 - `losses`). I will remove `win_prop` for the same reason. In other words, we know these variables are not independent form one another.

The variables I will include are `field_goals`, `field_goals_attempted`, `points3`, `points3_attempted`, `rebounds`, `personal_fouls_drawn`, `steals`. My theory is that a team needs to be strong on offense by both shooting a lot and making a lot of their shots. It needs to get rebounds and steals to keep the other team from shooting. Finally, they can get more points and send the other players to the bench if they draw a lot of fouls (looking at you James Harden).

```
#--> Create new data set
nba.data2 <- nba.data %>%
  transmute(wins=wins, fg=field_goals, fga=field_goals_attempted,
         threes=points3, threes_att=points3_attempted, reb=rebounds,
         fouls_drawn=personal_fouls_drawn, steals=steals)

#--> Plot
library(GGally)
ggpairs(nba.data2, progress=FALSE)
```

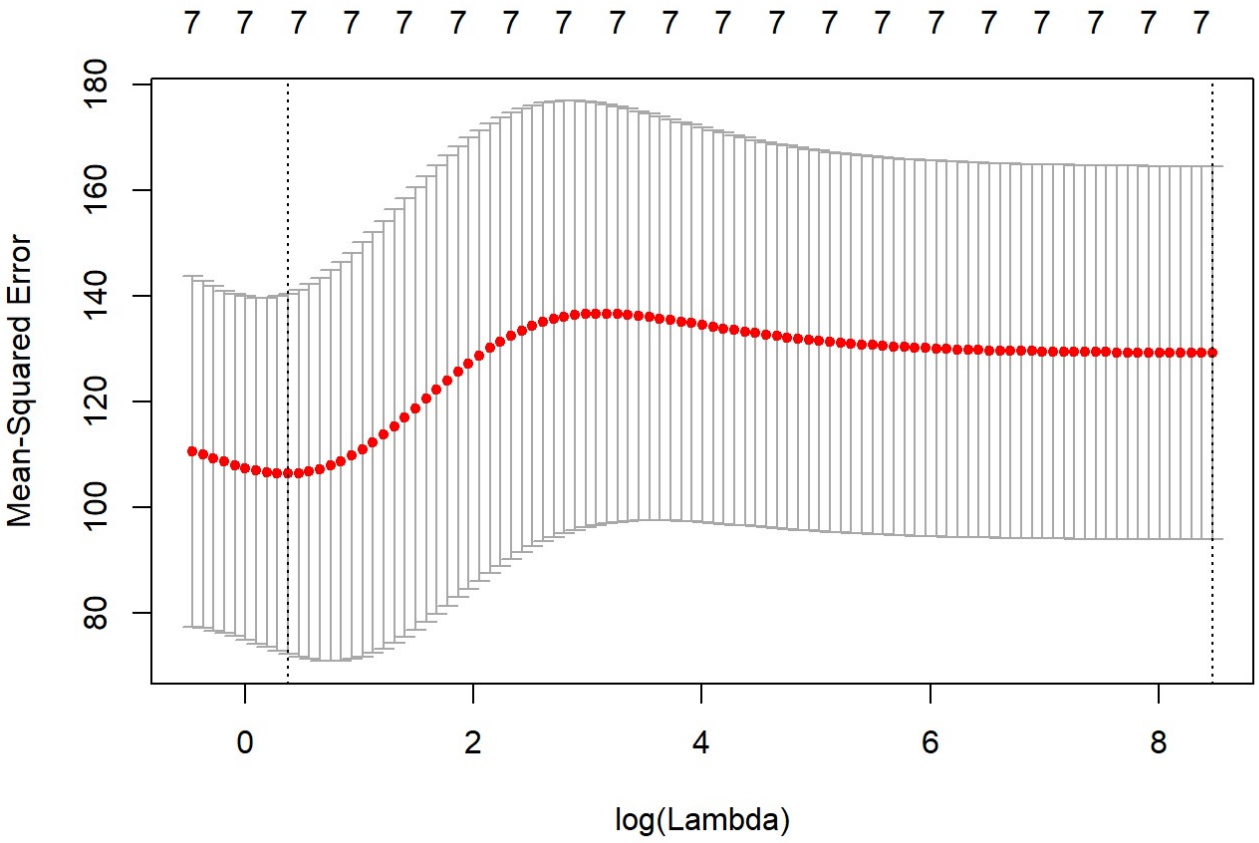


c. Use ridge regression with 5 fold cross-validation to choose the optimal tuning parameter and report your model along with your test error as found by cross-validation for that choice of λ.

```
#--> Sampling
set.seed(919)
train=sample(1:nrow(nba.data2), nrow(nba.data2)/2)
test=(-train)

#--> Split data
x=as.matrix(nba.data2[, -1])
y=nba.data2$wins

#--> Fit a linear model
lm.mod <- lm(wins~., data=nba.data2[train, ])
lm.pred <- predict(lm.mod, nba.data2[test, ])
print(list("...MSE...", mean((lm.pred - y[test])^2)))
```

```
## [[1]]
## [1] "...MSE..."
##
## [[2]]
## [1] 227.5641
```

```
#--> Optimal lambda
cv.out <- cv.glmnet(x[train, ], y[train], alpha=0, nfolds=10) # 5 too small
bestlam <- cv.out$lambda.min

plot(cv.out)
```




```
#--> Output
print(list("...Best lambda..", bestlam))
```

```
## [[1]]
## [1] "...Best lambda.."
##
## [[2]]
## [1] 1.455692
```

```
#--> MSE on test
ridge.mod <- glmnet(x[train, ], y[train], alpha=0, lambda=bestlam)
ridge.pred <- predict(ridge.mod, s=bestlam, newx=x[test, ])
print(list("...MSE...", mean((ridge.pred - y[test])^2)))
```

```
## [[1]]
## [1] "...MSE..."
##
## [[2]]
## [1] 84.32717
```

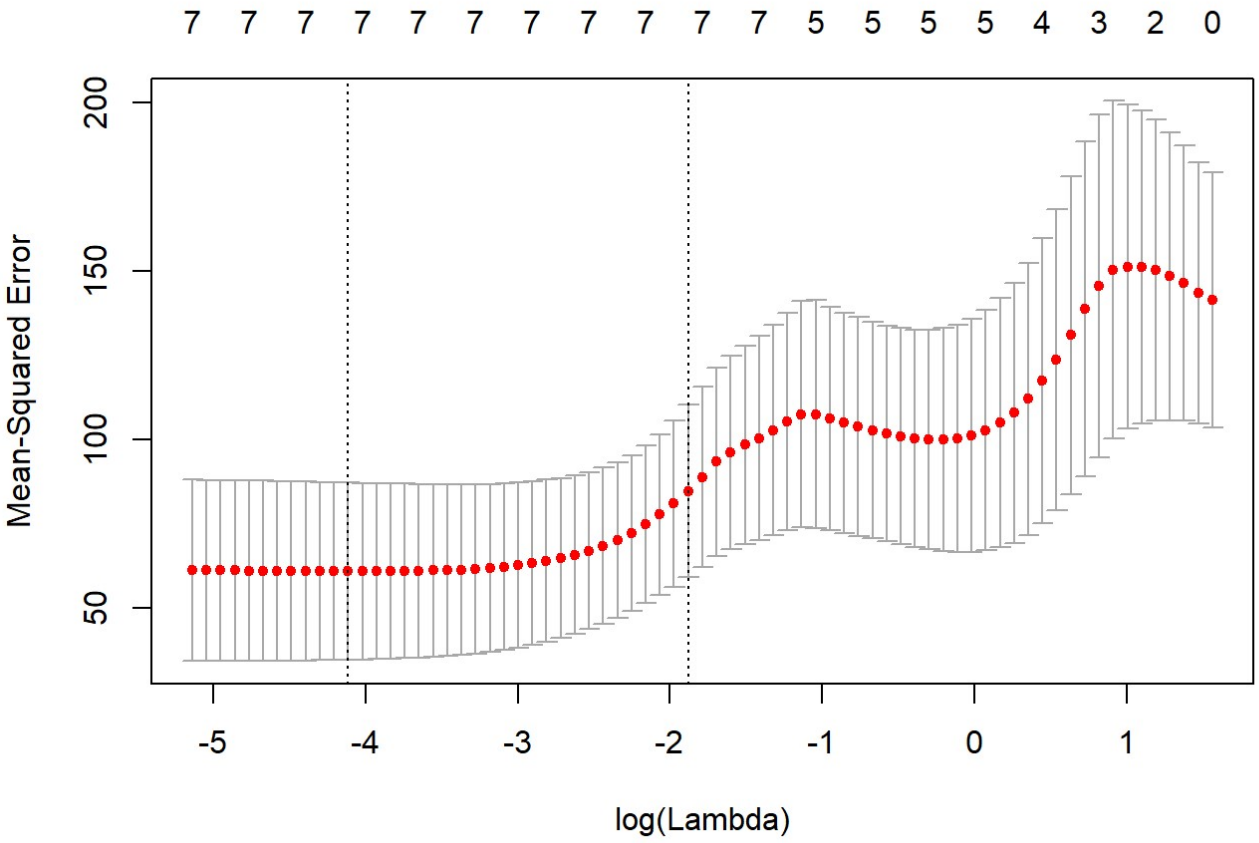
```
#--> Refit model with properly selected lambda and full data set
nba.mod <- glmnet(x, y, lambda=bestlam)
predict(nba.mod, type="coefficients", s=bestlam)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 19.314913
## fg           2.953501
## fga         -1.348343
## threes       1.616979
## threes_att   .
## reb          .
## fouls_drawn  .
## steals       0.767577
```

As reported above, the best choice of tuning parameter for our ridge regression is λ=1.46. Our model is thus $\hat{Wins}$ = 19.31 + 2.95 `fg` - 1.35 `fga` + 1.62 `threes` + 0.77 `steals` . Note that attempted threes, rebounds, and fouls drawn are so near zero that they are not included. Despite using α = 0 (to use ridge regression) we still inadvertently end up dropping variables. See below for MSPE.

d. Fit a LASSO model on the training set, with λ chosen by 5-fold cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
#--> Optimal lambda
cv.out2 <- cv.glmnet(x[train, ], y[train], alpha=1, nfolds=10) # 5 too small
bestlam2 <- cv.out2$lambda.min

plot(cv.out2)
```




```
#--> Output
print(list("...Best lambda...", bestlam2))
```

```
## [[1]]
## [1] "...Best lambda..."
##
## [[2]]
## [1] 0.01635214
```

```
#--> MSE on lasso
lasso.mod <- glmnet(x[train, ], y[train], alpha=1, lambda=bestlam2)
lasso.pred <- predict(lasso.mod, s=bestlam2, newx=x[test, ])
print(list("...MSPE...", mean((lasso.pred - y[test])^2)))
```

```
## [[1]]
## [1] "...MSPE..."
##
## [[2]]
## [1] 216.4527
```

```
#--> Refit model with properly selected lambda
nba.mod2 <- glmnet(x, y, lambda=bestlam2)
predict(nba.mod2, type="coefficients", s=bestlam2)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 45.554393
## fg           1.771497
## fga         -3.163654
## threes       9.643864
## threes_att  -2.648851
## reb          3.672911
## fouls_drawn -1.319634
## steals       5.367881
```

As reported above, the best choice of tuning parameter for our LASSO regression is λ=0.016. Our model is thus $\hat{Wins}$ = 45.55 + 1.77 `fg` - 3.16 `fga` + 9.64 `threes` - 2.64 `threes_att` + 3.67 `reb` - 1.32 `fouls_drawn` + 5.37 `steals` . The MSPE is 7.87 and we dropped eight variables. See below for MSPE.

Out of curiosity, I fit an unpenalized/normal linear model in the code chunk called “lm_nba.” The mean squared error is 228 for the standard linear model, 216 for LASSO, and 82 for Ridge regression. Therefore, ridge regression performs the best for these data. The model can be found above.

---

3. (Optional *12 pt*) Find a data set online (different from those in the book!). Put a link to where **we** can find this data set and describe why you were interested in this data set. Carry out multivariate linear regression (as well as any general exploratory data analysis you think is relevant for example generating plots as in problem 2 (a)). Use subset selection as well as ridge regression and lasso to obtain models and interpret your results. Describe your findings to someone who might know no math or statistics. At the end of the day, you will have more fun if you find data that you truly care about!

This question is optional if you do some of the optional questions in the theory portion.

I would like to do further analysis on the data set I used for linear regression.

My sister Paige, a sage, sentinel, and soon-to-be-sommelier, is living in Belgium and playing carrilon (https://en.wikipedia.org/wiki/Carillon). She even has a blog (https://beerbellsandbelgium.com/) about it. While there, she has developed a taste for not only Belgian beer, but French wine (I personally feel you can only like one or the other). Her love of oenology is so deep that she aspires to work in wine tourism. To make her life easier, I would like to look at predictors of wine quality (i.e. alcohol content). Is there a way to predict, say, alcohol content if you know other measurements? I obtained this data from the University of California at Irvine Machine Learning Data Collection (https://archive.ics.uci.edu/ml/datasets/Wine).

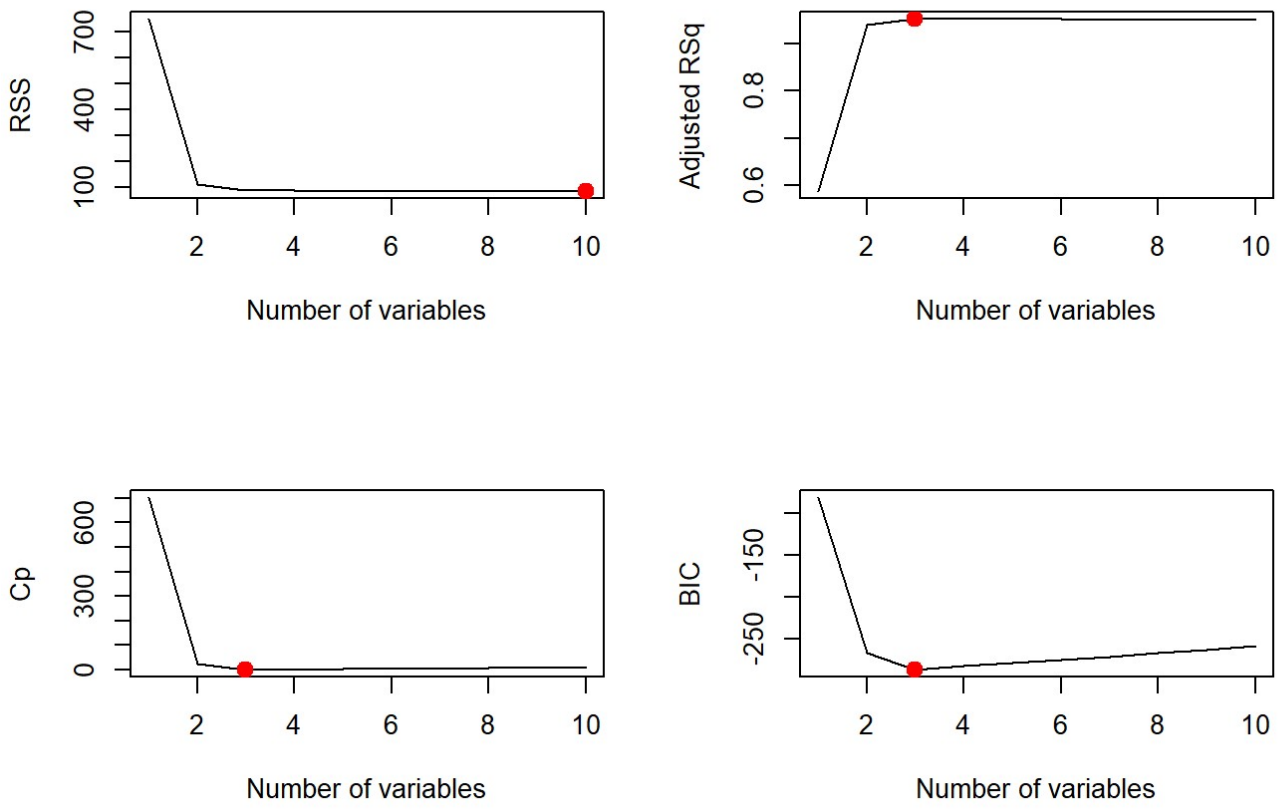
```
#--> Read and add variable names
wine <- read.csv("wine.data", header=FALSE)
wine <- wine[, -1]
names(wine) <- c("Alcohol", "MalicAcid", "Ash", "AlcalinityAsh",
                 "Magnesium", "TotalPhenols", "Flavanoids",
                 "NonflavPhenols", "Proanthocyanins", "ColorIntensity",
                 "Hue", "OD280", "Proline")

head(wine)
```

```
##   Alcohol MalicAcid  Ash AlcalinityAsh Magnesium TotalPhenols Flavanoids
## 1   14.23      1.71 2.43          15.6       127         2.80       3.06
## 2   13.20      1.78 2.14          11.2       100         2.65       2.76
## 3   13.16      2.36 2.67          18.6       101         2.80       3.24
## 4   14.37      1.95 2.50          16.8       113         3.85       3.49
## 5   13.24      2.59 2.87          21.0       118         2.80       2.69
## 6   14.20      1.76 2.45          15.2       112         3.27       3.39
##   NonflavPhenols Proanthocyanins ColorIntensity  Hue OD280 Proline
## 1           0.28            2.29           5.64 1.04  3.92    1065
## 2           0.26            1.28           4.38 1.05  3.40    1050
## 3           0.30            2.81           5.68 1.03  3.17    1185
## 4           0.24            2.18           7.80 0.86  3.45    1480
## 5           0.39            1.82           4.32 1.04  2.93     735
## 6           0.34            1.97           6.75 1.05  2.85    1450
```

```
#--> Perform standard subset selection
wine.full <- regsubsets(Alcohol~., data=wine, nvmax=12)
wine.summary <- summary(regfit.full)

#--> Tell us results
wine.summary
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data2, nvmax = 10)
## 10 Variables  (and intercept)
##     Forced in Forced out
## x       FALSE      FALSE
## x2      FALSE      FALSE
## x3      FALSE      FALSE
## x4      FALSE      FALSE
## x5      FALSE      FALSE
## x6      FALSE      FALSE
## x7      FALSE      FALSE
## x8      FALSE      FALSE
## x9      FALSE      FALSE
## x10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           x   x2  x3  x4  x5  x6  x7  x8  x9  x10
## 1  ( 1 )  " " "*" " " " " " " " " " " " " " " " "
## 2  ( 1 )  "*" "*" " " " " " " " " " " " " " " " "
## 3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " "
## 4  ( 1 )  "*" "*" "*" " " " " " " " " " " "*" " "
## 5  ( 1 )  "*" "*" "*" "*" " " "*" " " " " " " " "
## 6  ( 1 )  "*" "*" " " "*" "*" " " "*" " " "*" " "
## 7  ( 1 )  "*" "*" "*" "*" " " "*" " " "*" " " "*"
## 8  ( 1 )  "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 9  ( 1 )  "*" "*" " " "*" "*" "*" "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

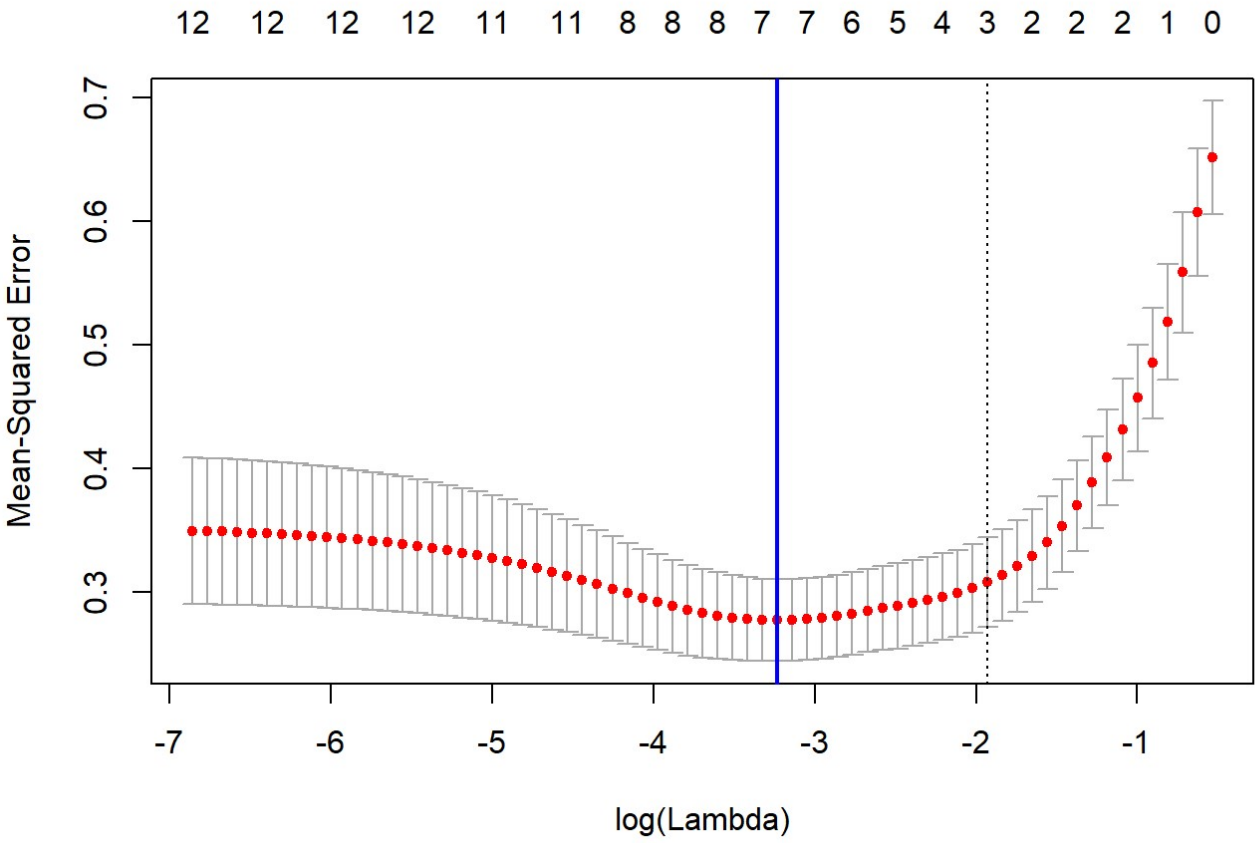
```
#--> Graph
plotFour(wine.summary)
```




```
#--> Coefficients for 3-variable model
coef(wine.full, 3)
```

```
##    (Intercept)      MalicAcid ColorIntensity        Proline
##   11.123529619    0.084858620    0.119332750    0.001439608
```

```
#--> Coerce data
set.seed(919) # Represent
train <- sample(c(T, F), nrow(wine), rep=T)
test <- (!train)

predictors <- as.matrix(select(wine, -Alcohol))
alcohol <- as.matrix(select(wine, Alcohol))

#--> Cross validate for best lambda
wine.lasso.cv <- cv.glmnet(predictors[train, ], alcohol[train, ], nfolds=5, alpha=1) #
1 for LASSO

#--> Plot lambda vs MSE
plot(wine.lasso.cv)
abline(v=log(wine.lasso.cv$lambda.min), col="blue", lwd=2)
```




```
#--> What is the best lambda?
wine.lasso.cv$lambda.min
```

```
## [1] 0.03949453
```

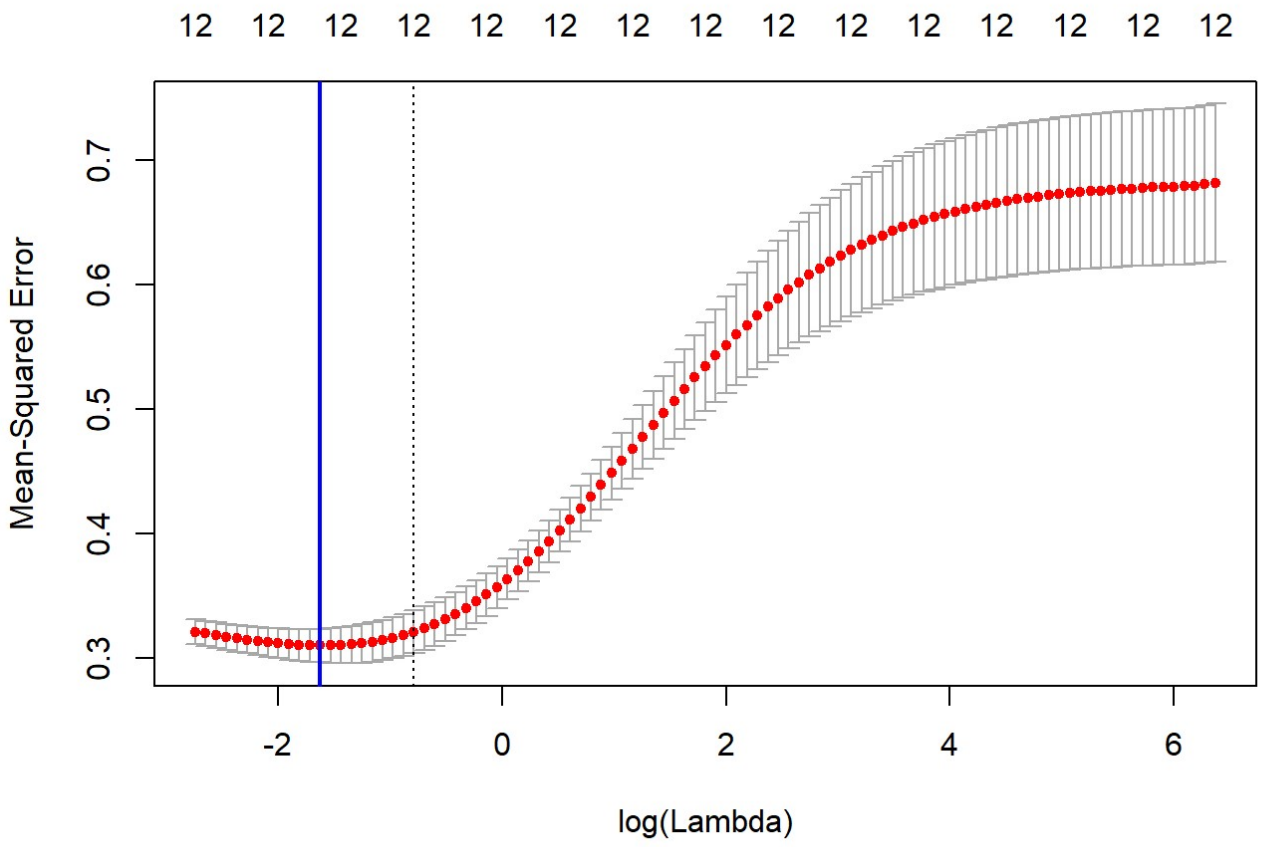
```
#--> What is our MSPE
lasso.pred <- predict(wine.lasso.cv, s=wine.lasso.cv$lambda.min, newx=predictors[test,
])
mean((lasso.pred - alcohol[test])^2)
```

```
## [1] 0.3434003
```

```
#--> Fit model to whole data set
wine.full.lasso <- glmnet(x=predictors,y=alcohol,alpha=1,lambda=wine.lasso.cv$lambda.m
in)
coef(wine.full.lasso)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)     11.618503302
## MalicAcid        0.061526642
## Ash              .
## AlcalinityAsh   -0.019913598
## Magnesium        .
## TotalPhenols     0.023868624
## Flavanoids       .
## NonflavPhenols   .
## Proanthocyanins  .
## ColorIntensity   0.123121718
## Hue              .
## OD280            0.035176156
## Proline          0.001147648
```

```
#--> Cross validate for best lambda
wine.ridge.cv <- cv.glmnet(predictors[train, ], alcohol[train, ], nfolds=5, alpha=0) # 0 for ridge

#--> Plot lambda vs MSE
plot(wine.ridge.cv)
abline(v=log(wine.ridge.cv$lambda.min), col="blue", lwd=2)
```



```
#--> What is the best lambda?
wine.ridge.cv$lambda.min
```

```
## [1] 0.1965652
```

```
#--> What is our MSPE
wine.pred <- predict(wine.ridge.cv, s=wine.ridge.cv$lambda.min, newx=predictors[test,
])
mean((lasso.pred - alcohol[test])^2)
```

```
## [1] 0.3434003
```

```
#--> Fit model to whole data set
wine.full.ridge <- glmnet(x=predictors,y=alcohol,alpha=0,lambda=wine.ridge.cv$lambda.m
in)
coef(wine.full.ridge)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)     11.4956338896
## MalicAcid        0.0909546114
## Ash              0.1934748464
## AlcalinityAsh   -0.0366814737
## Magnesium        0.0013307218
## TotalPhenols     0.0822552822
## Flavanoids       0.0327233409
## NonflavPhenols  -0.1891251809
## Proanthocyanins -0.0819487658
## ColorIntensity   0.1183951197
## Hue             -0.0240781493
## OD280            0.0536864175
## Proline          0.0008630452
```

Paige, here's what we learned from this analysis. There are some interesting relationships between the measurements you collected and the alcohol content of your wine. To simplify this model, I'm going to stick with the LASSO regression model–this one is useful because it knocks out measurements which we don't necessarily need. There is a slighty positive relationship between Malic Acid, Total Phenols, Color Intensity, OD280, and Proline. So if there is an increase in one of those, we expect an increase in Alcohol Content. There is a negative relationship with Alcalinity Ash, so if Alcalinity Ash increases, we expect Alcohol Content to decrease. These are not hard and fast rules but general trends based on our assumptions.

We don't need to get into the nitty-gritty, but if you'd like to predict alcohol, I'd suggest plugging this formula into Excel:

$\hat{Alcohol}$ = 11.62 + 0.06 `MalicAcid` - 0.02 `AlcalinityAsh` + 0.02 `TotalPhenols` + 0.12 `ColorIntensity` + 0.04 `OD280` + 0.001 `Proline`