STOR 565 Spring 2019 Homework 3

Due on 02/14/2018 in Class

Alexandra Beja-Glasser

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 $\boldsymbol{X}$ of length $n = 100$, as well as a noise vector $\boldsymbol{\epsilon}$ of length $n = 100$. Do not print the entire vector.

```r
set.seed(1)
X = rnorm(100)
eps = 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 $\boldsymbol{Y}$ 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.

```r
beta0 = 3
beta1 = 2
beta2 = -3
beta3 = 0.3
Y = beta0 + beta1 * X + beta2 * X^2 + beta3 * X^3 + eps
```

(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.

```r
data = data.frame(y = Y, x = X)
model = regsubsets(y ~ poly(x, 10, raw = T), data = data, nvmax = 10)
model_summary = summary(model)


which.min(model_summary$cp)
```

```
## [1] 3
```

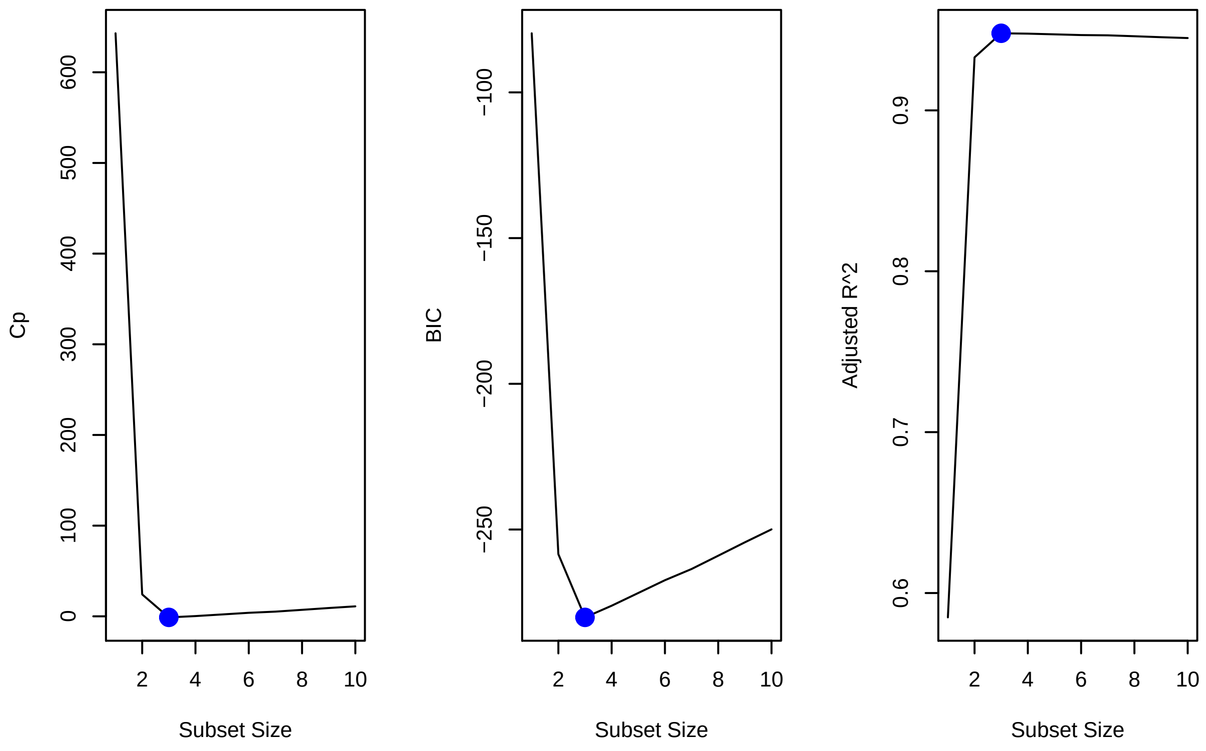
```r
which.min(model_summary$bic)
```

```
## [1] 3
```

```r
which.max(model_summary$adjr2)
```

```
## [1] 3
```

```r
#the best model according to Cp, BIC, and Adjusted R^2 is the one with a subset of size 3

par(mfrow = c(1, 3))
plot(model_summary$cp, xlab = "Subset Size", ylab = "Cp", type = "l")
points(3, model_summary$cp[3], col = "blue", lwd=5)

plot(model_summary$bic, xlab = "Subset Size", ylab = "BIC", type = "l")
points(3, model_summary$bic[3], col = "blue", lwd=5)

plot(model_summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R^2", type = "l")
points(3, model_summary$adjr2[3], col = "blue", lwd=5)
```



```
#coefficients of the best model
coef(model, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            3.07627412            2.35623596           -3.16514887
## poly(x, 10, raw = T)7
##            0.01046843
```

(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.

```
forward_model = regsubsets(y ~ poly(x, 10, raw = T), data = data, nvmax = 10, method = "forward")
forward_summary = summary(forward_model)
which.min(forward_summary$cp)
```

```
## [1] 3
```

```
which.min(forward_summary$bic)
```

```
## [1] 3
```

```r
which.max(forward_summary$adjr2)
```

```
## [1] 3
```

```r
#the best model using forward-stepwise regression is the one with a subset of size 3

backward_model = regsubsets(y ~ poly(x, 10, raw = T), data = data, nvmax = 10, method = "backward")
backward_summary = summary(backward_model)
which.min(backward_summary$cp)
```

```
## [1] 3
```

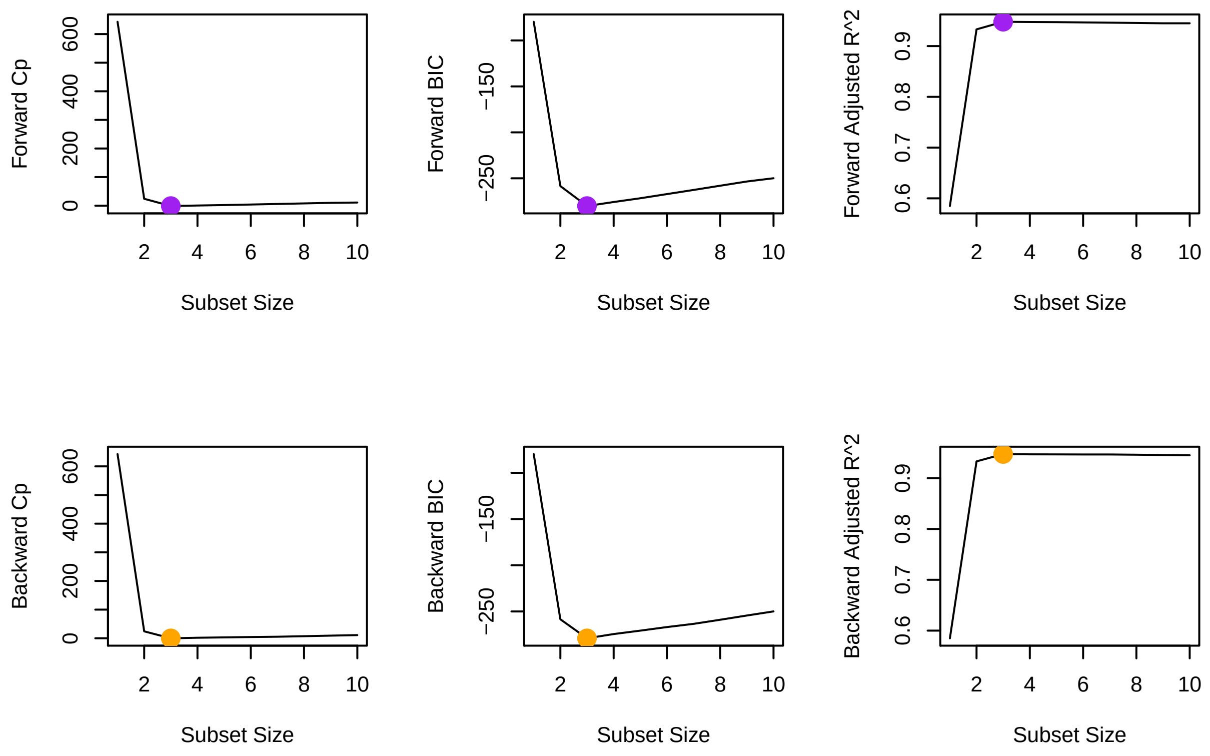
```r
which.min(backward_summary$bic)
```

```
## [1] 3
```

```r
which.max(backward_summary$adjr2)
```

```
## [1] 3
```

```r
#the best model using backward-stepwise regression is also the one with a subset of size 3


par(mfrow = c(2, 3))
plot(forward_summary$cp, xlab = "Subset Size", ylab = "Forward Cp", type = "l")
points(3, forward_summary$cp[3], col = "purple", lwd=5)

plot(forward_summary$bic, xlab = "Subset Size", ylab = "Forward BIC", type = "l")
points(3, forward_summary$bic[3], col = "purple", lwd=5)

plot(forward_summary$adjr2, xlab = "Subset Size", ylab = "Forward Adjusted R^2", type = "l")
points(3, forward_summary$adjr2[3], col = "purple", lwd=5)


plot(backward_summary$cp, xlab = "Subset Size", ylab = "Backward Cp", type = "l")
points(3, backward_summary$cp[3], col = "orange", lwd=5)

plot(backward_summary$bic, xlab = "Subset Size", ylab = "Backward BIC", type = "l")
points(3, backward_summary$bic[3], col = "orange", lwd=5)

plot(backward_summary$adjr2, xlab = "Subset Size", ylab = "Backward Adjusted R^2", type = "l")
points(3, backward_summary$adjr2[3], col = "orange", lwd=5)
```

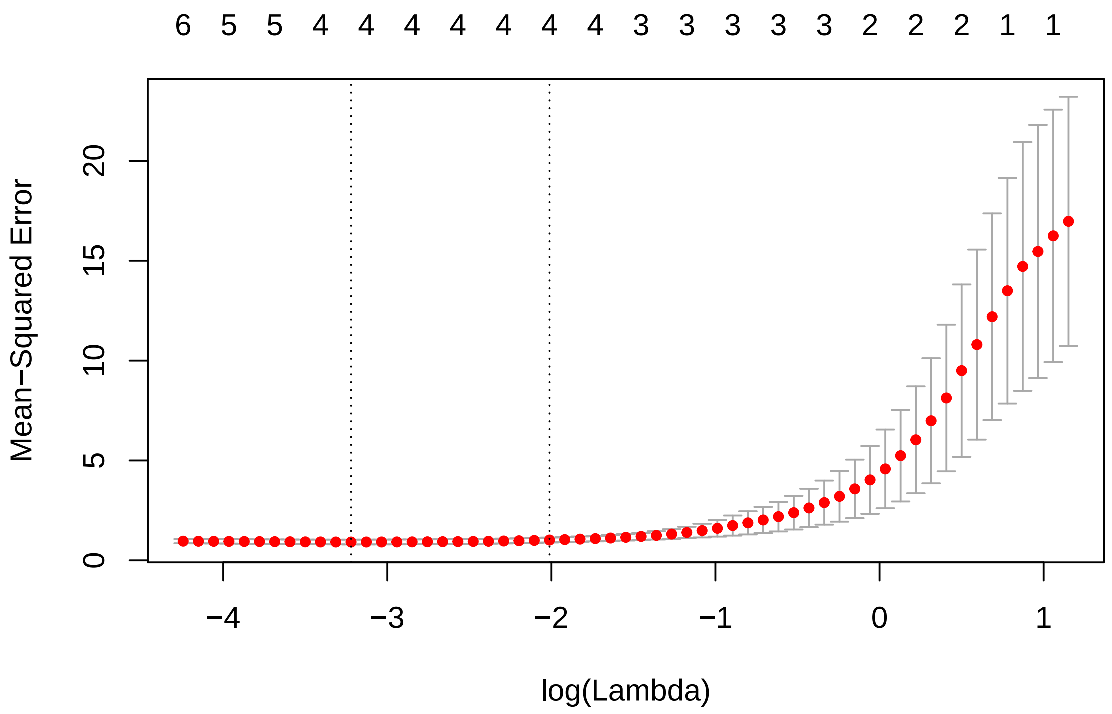


(e) Now fit a LASSO model with `glmnet` function from `glmnet` package to the simulated data, again using $(X, X^2, \cdots, X^{10})$ as predictors. Use 5-fold cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient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

```r
x_matrix = model.matrix(y ~ poly(x, 10, raw = T), data = data)[, -1]
lasso_model = cv.glmnet(x_matrix, Y, alpha = 1)
best_lambda = lasso_model$lambda.min
best_lambda
```

```
## [1] 0.03991416
```

```r
plot(lasso_model)
```



```r
best_lambda_model = glmnet(x_matrix, Y, alpha = 1)
predict(best_lambda_model, s = best_lambda, type = "coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                   1
## (Intercept)             3.039815105
## poly(x, 10, raw = T)1   2.230337134
## poly(x, 10, raw = T)2  -3.103319268
## poly(x, 10, raw = T)3   .
## poly(x, 10, raw = T)4   .
## poly(x, 10, raw = T)5   0.049841076
## poly(x, 10, raw = T)6   .
## poly(x, 10, raw = T)7   0.000806843
## poly(x, 10, raw = T)8   .
## poly(x, 10, raw = T)9   .
## poly(x, 10, raw = T)10  .
```

```r
#the LASSO model picks X5 instead of X3. It also picks X7, but has a very small coefficient.
```

(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.

```r
beta7 = 7
Y = beta0 + beta7 * X^7 + eps

# Predict using regsubsets
data = data.frame(y = Y, x = X)
model2 = regsubsets(y ~ poly(x, 10, raw = T), data = data, nvmax = 10)
model2_summary = summary(model2)

which.min(model2_summary$cp) #best model has subset of size 2
```

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

```r
which.min(model2_summary$bic) #best model has subset of size 1
```

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

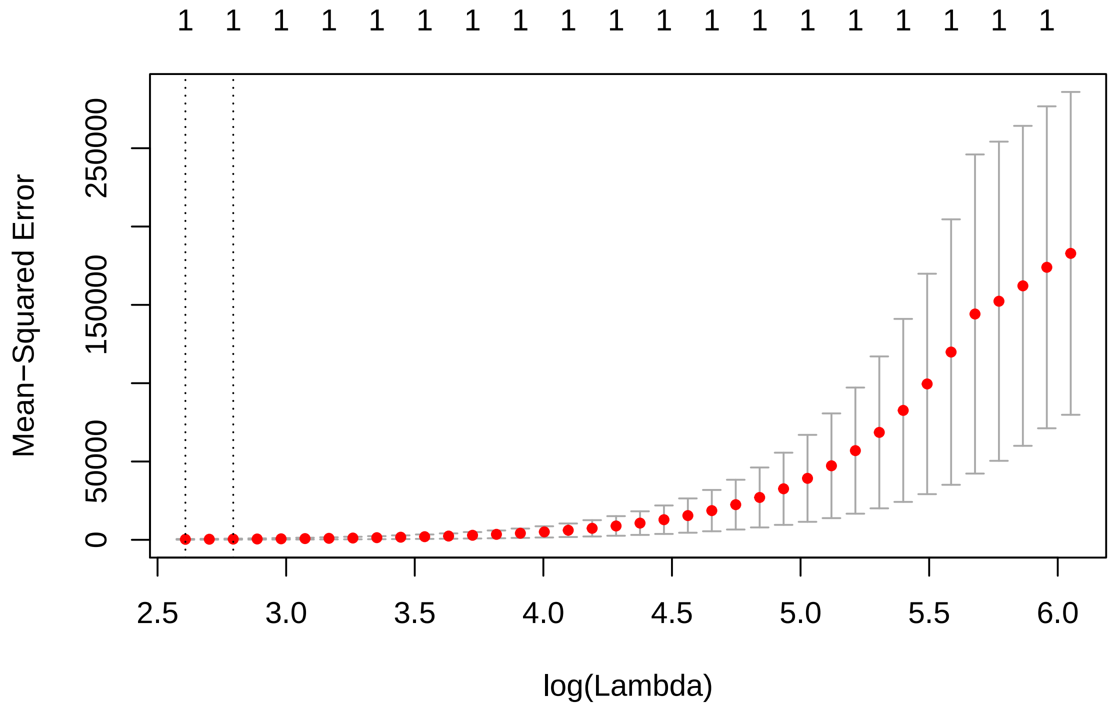
```r
which.max(model2_summary$adjr2) #best model has subset of size 4
```

```
## [1] 4
```

```r
coef(model2, id=c(1, 2, 4))
```

```
## [[1]]
##           (Intercept) poly(x, 10, raw = T)7
##               2.95894               7.00077
##
## [[2]]
##           (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
##             3.0704904            -0.1417084             7.0015552
##
## [[3]]
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             3.0762524             0.2914016            -0.1617671
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
##            -0.2526527             7.0091338
```

```r
x_matrix2 = model.matrix(y ~ poly(x, 10, raw = T), data = data)[, -1]
lasso_model2 = cv.glmnet(x_matrix, Y, alpha = 1)
plot(lasso_model2)
```



```r
best_lambda2 = lasso_model2$lambda.min
best_lambda2
```

```
## [1] 13.57478
```

```r
model3 = glmnet(x_matrix, Y, alpha = 1)
predict(model3, s = best_lambda2, type = "coefficients")
```

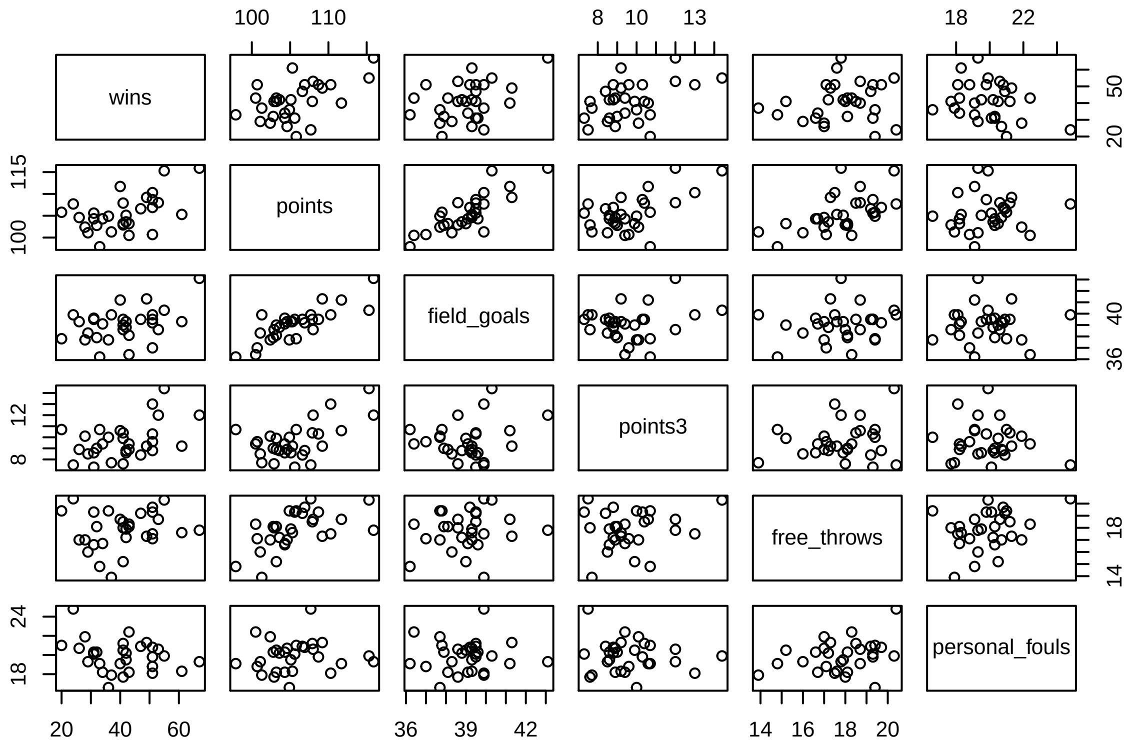
```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)            3.904188
## poly(x, 10, raw = T)1  .
## poly(x, 10, raw = T)2  .
## poly(x, 10, raw = T)3  .
## poly(x, 10, raw = T)4  .
## poly(x, 10, raw = T)5  .
## poly(x, 10, raw = T)6  .
## poly(x, 10, raw = T)7  6.776797
## poly(x, 10, raw = T)8  .
## poly(x, 10, raw = T)9  .
## poly(x, 10, raw = T)10 .
```

```r
#the LASSO model gives us an intercept of 3.9,
#which is different from the intercept of 3 we got from best subset selection
#(from all models containing best Cp, BIC, Adjusted R^2)
```

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!):

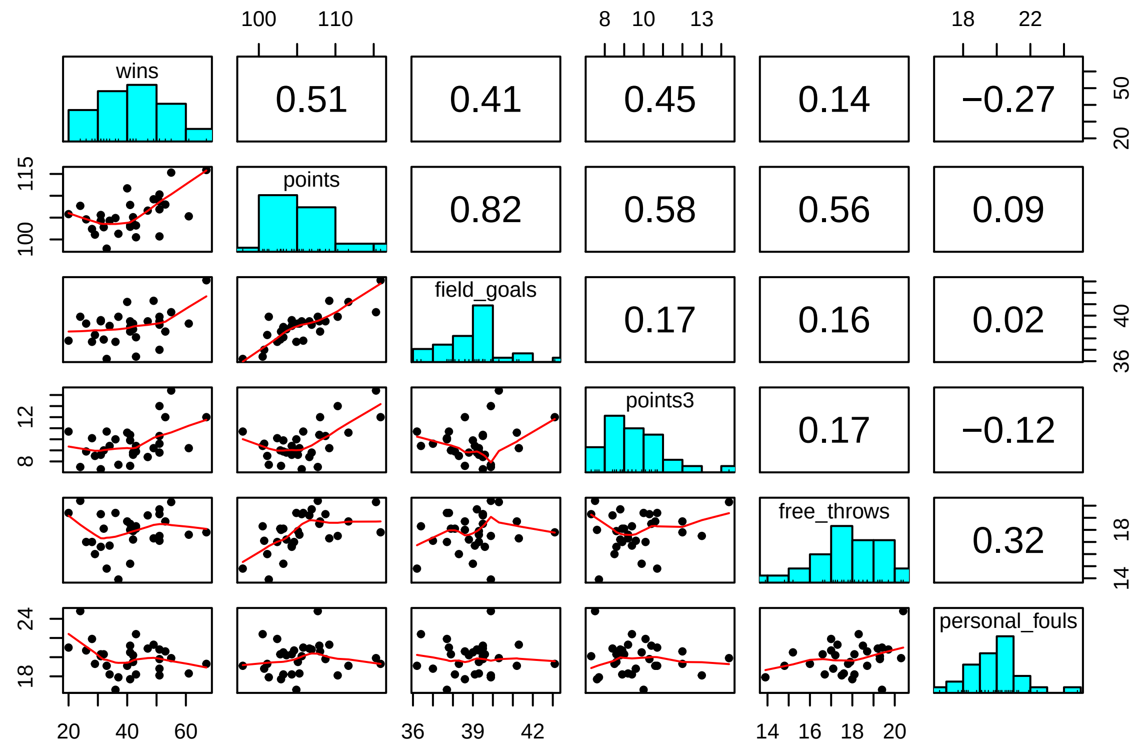
```r
bball <- read.csv("nba-teams-2017.csv")

plot(bball[,c("wins", "points", "field_goals", "points3", "free_throws", "personal_fouls")])
```



```r
library(psych)
```

```
## Warning: package 'psych' was built under R version 3.4.4
```

```r
pairs.panels(bball[,c("wins", "points", "field_goals", "points3", "free_throws", "personal_fouls")],
             method = "pearson", density = F, ellipses = F)
```



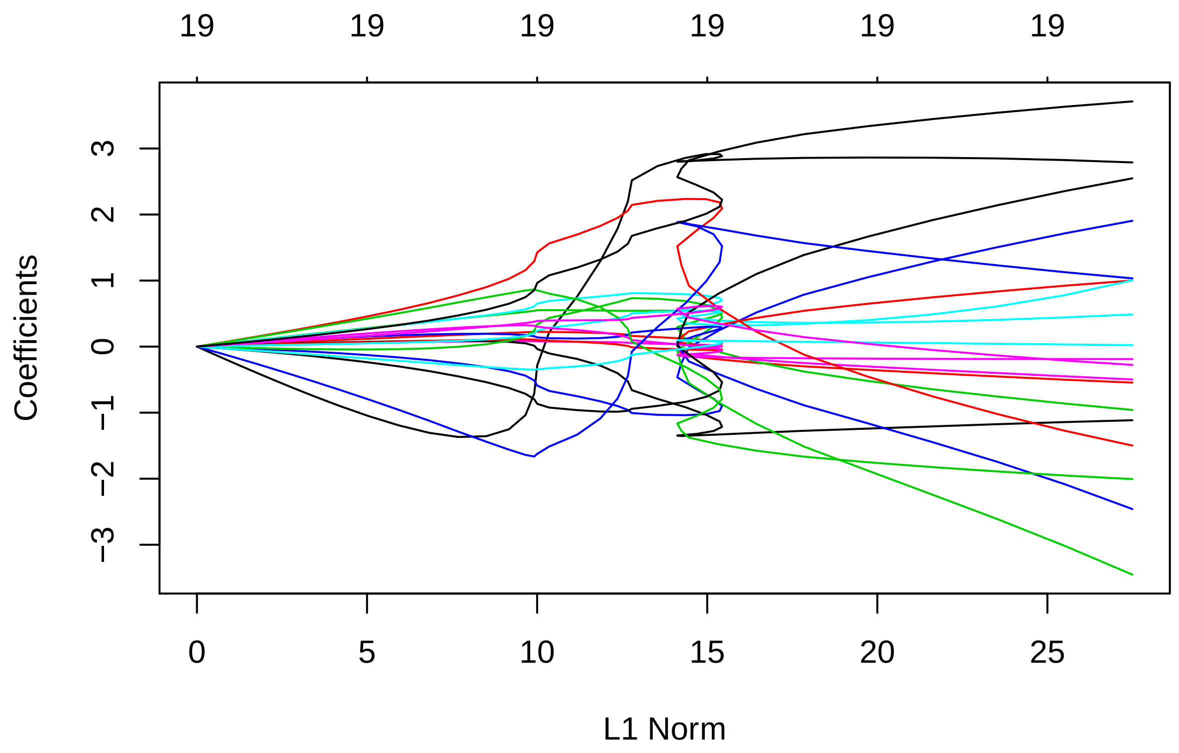
NOTE: You may remove the includegraphics statements below when knitting your own response, if they are giving you trouble

(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'm removing the losses because it  is dependent on "wins", which is our response variable.
#Losses is equal to games_played - wins.
#I'm also removing all proportional variables (ie, "win_prop", "field_goals_prop", "points3_prop",
#and "free_throws_prop"), since they are all proportions of variables that are already included.
#And finally, I'm removing "team", since the name of the team shouldn't have any effect on wins.

bball <- bball[,c(-1, -4, -5, -10, -13, -16)]
#View(bball)
```

(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.

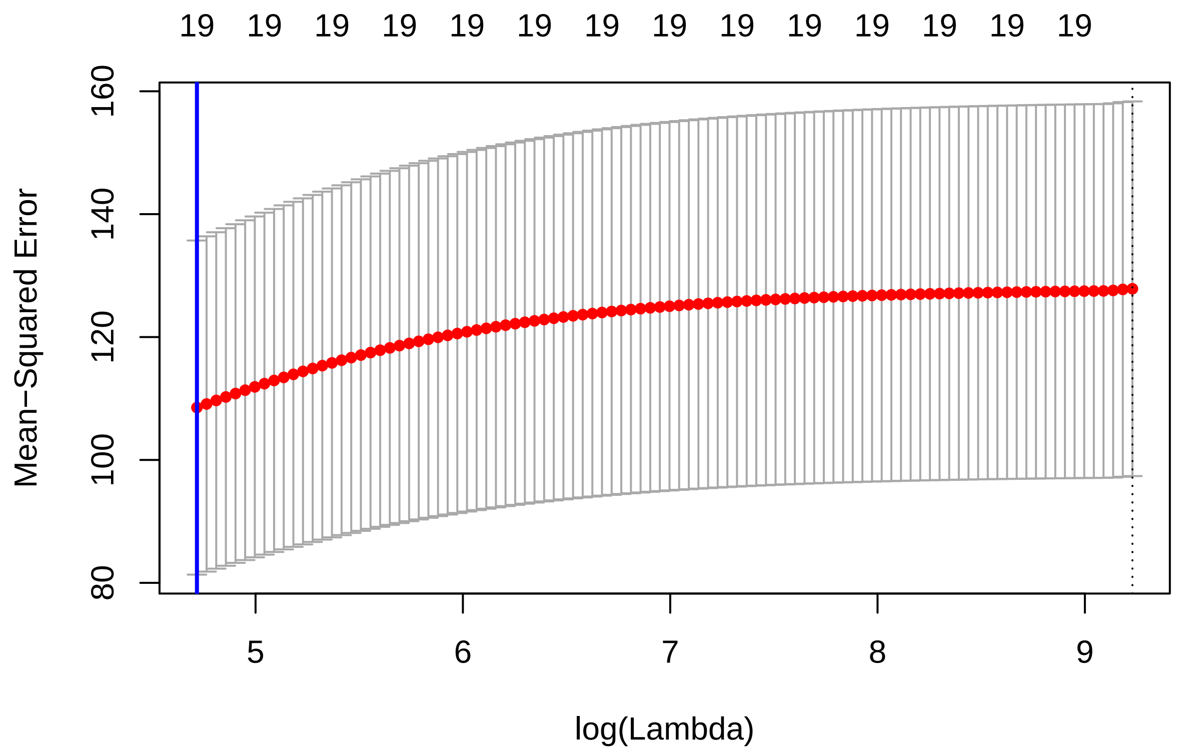
```
x = model.matrix(wins ~., bball)[,-1]
y = bball$wins

grid <- 10^seq(10, -2, length = 100)
ridge_model <- glmnet(x, y, alpha = 0, lambda = grid)
plot(ridge_model)
```



```
ell2.norm <- numeric()
for(i in 1:length(grid)){
ell2.norm[i] <- sqrt(sum(coef(ridge_model)[-1, i]^2))
}
#plot(x = grid, y = ell2.norm, xlab = expression(lambda), ylab = "L2 Norm", xlim = c(10,10000))
train <- sample(1:nrow(x), nrow(x) / 2)
test <- (-train)
y.train <- y[train]
y.test <- y[test]
ridge_model_train <- glmnet(x[train, ], y.train, alpha = 0, lambda = grid)
cv.out <- cv.glmnet(x[train, ], y.train, alpha = 0)
plot(cv.out)
best.lambda <- cv.out$lambda.min
best.lambda
```

```
## [1] 111.8553
```

```
plot(cv.out)
abline(v = log(best.lambda), col = "blue", lwd = 2)
```



```r
ridge.pred <- predict(ridge_model_train, s = best.lambda, newx = x[test, ])
mspe.ridge <- mean((ridge.pred - y.test)^2)
mspe.ridge
```

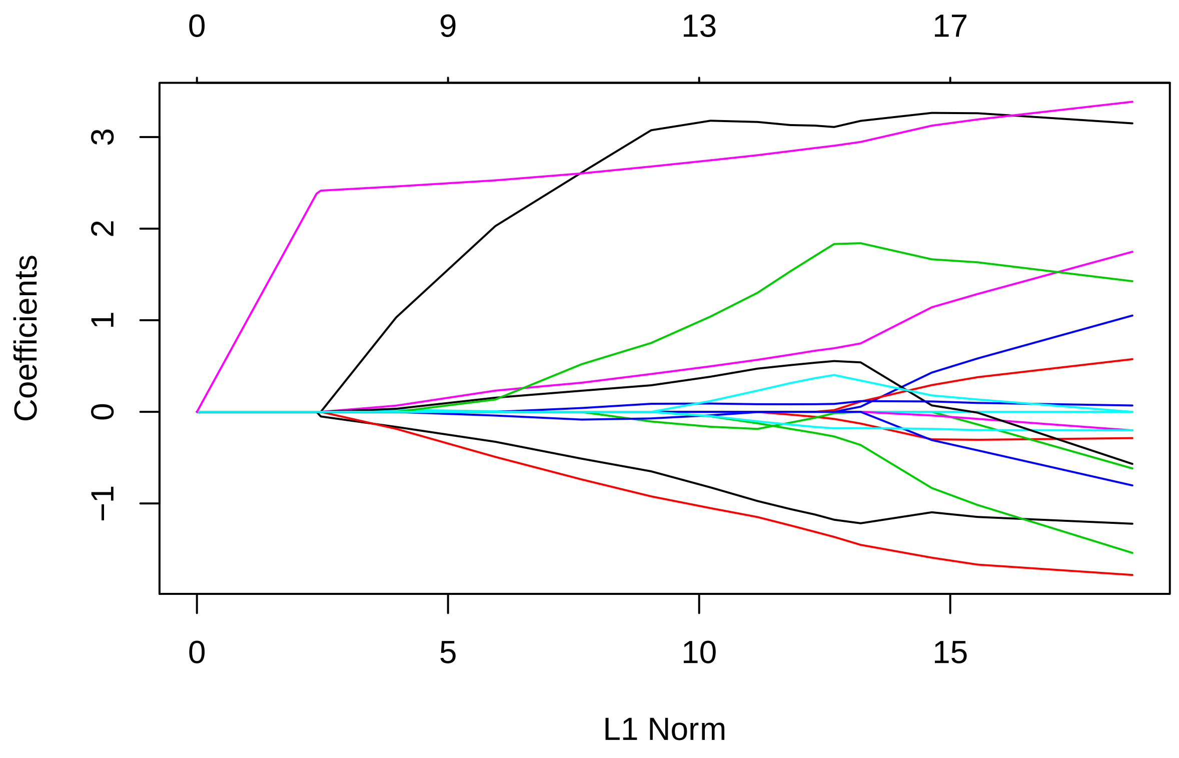
```
## [1] 110.7338
```

```r
final.model <- glmnet(x, y, alpha = 0, lambda = best.lambda)
Coef.Ridge <- coef(final.model)[1:20, ]
Coef.Ridge
```

```
##           (Intercept)          games_played                minutes
##           57.35239860            0.00000000            -0.83455153
##                points           field_goals field_goals_attempted
##            0.09306339            0.21881467            -0.09019787
##               points3      points3_attempted           free_throws
##            0.21280801            0.04639706             0.05983121
##        free_throws_att          off_rebounds          def_rebounds
##            0.05383682           -0.04107572             0.12562803
##              rebounds               assists             turnovers
##            0.03706376            0.15693151            -0.16857996
##                steals                blocks             block_fga
##            0.34384969            0.32279723            -0.59860335
##        personal_fouls  personal_fouls_drawn
##           -0.13710865            0.12489964
```

(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.

```
lasso.model <- glmnet(x, y, alpha = 1, lambda = grid)
plot(lasso.model)
```



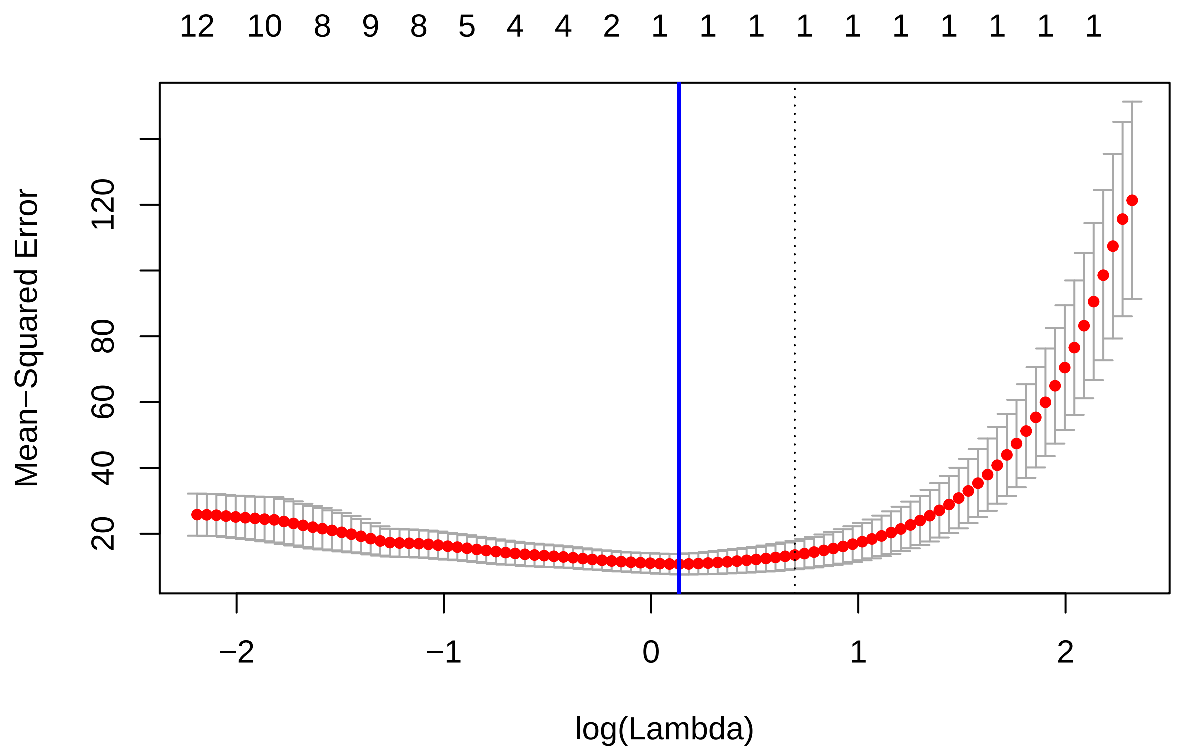
```
lasso.model.train <- glmnet(x[train, ], y.train, alpha = 1, lambda = grid)
set.seed(1) #in case R forgot
cv.out <- cv.glmnet(x[train, ], y.train, alpha = 1)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations
## per fold
```

```
plot(cv.out)
best.lambda <- cv.out$lambda.min
best.lambda
```

```
## [1] 1.144874
```

```
plot(cv.out)
abline(v = log(best.lambda), col = "blue", lwd = 2)
```



```r
lasso.pred <- predict(lasso.model.train, s = best.lambda, newx = x[test,])
mspe.lasso <- mean((lasso.pred - y.test)^2)
mspe.lasso
```

```
## [1] 10.02344
```

```r
final.model <- glmnet(x, y, alpha = 1, lambda = best.lambda)
Coef.Lasso <- coef(final.model)[1:20,]
Coef.Lasso
```

```
##           (Intercept)          games_played               minutes
##                    41                     0                     0
##                points           field_goals field_goals_attempted
##                     0                     0                     0
##               points3     points3_attempted           free_throws
##                     0                     0                     0
##       free_throws_att          off_rebounds          def_rebounds
##                     0                     0                     0
##              rebounds               assists             turnovers
##                     0                     0                     0
##                steals                blocks             block_fga
##                     0                     0                     0
##        personal_fouls  personal_fouls_drawn
##                     0                     0
```

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. This data is about how to determine the "best" candy. It is from FiveThirtyEight. Link: https://github.com/fivethirtyeight/data/blob/master/candy-power-ranking/candy-data.csv

```r
candy <- read_csv("candy-data.csv")
```

```
## Parsed with column specification:
## cols(
##   competitorname = col_character(),
##   chocolate = col_double(),
##   fruity = col_double(),
##   caramel = col_double(),
##   peanutyalmondy = col_double(),
##   nougat = col_double(),
##   crispedricewafer = col_double(),
##   hard = col_double(),
##   bar = col_double(),
##   pluribus = col_double(),
##   sugarpercent = col_double(),
##   pricepercent = col_double(),
##   winpercent = col_double()
## )
```

```r
#View(candy)

#Trying to predict the overall "win" percentage (ie, how much people like it from a scale of 1 to 100)
#of each candy
candy <- data.frame(candy[,2:13])

summary(lm(winpercent ~ ., data=candy))  #general linear model has  R^2 of 0.54
```

```
##
## Call:
## lm(formula = winpercent ~ ., data = candy)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.2244  -6.6247   0.1986   6.8420  23.8680
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       34.5340     4.3199   7.994 1.44e-11 ***
## chocolate         19.7481     3.8987   5.065 2.96e-06 ***
## fruity             9.4223     3.7630   2.504  0.01452 *
## caramel            2.2245     3.6574   0.608  0.54493
## peanutyalmondy    10.0707     3.6158   2.785  0.00681 **
## nougat             0.8043     5.7164   0.141  0.88849
## crispedricewafer   8.9190     5.2679   1.693  0.09470 .
## hard              -6.1653     3.4551  -1.784  0.07852 .
## bar                0.4415     5.0611   0.087  0.93072
## pluribus          -0.8545     3.0401  -0.281  0.77945
## sugarpercent       9.0868     4.6595   1.950  0.05500 .
## pricepercent      -5.9284     5.5132  -1.075  0.28578
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.7 on 73 degrees of freedom
## Multiple R-squared:  0.5402, Adjusted R-squared:  0.4709
## F-statistic: 7.797 on 11 and 73 DF,  p-value: 9.504e-09
```

```
x = candy[,1:11]
y = candy[,12]
candy_model = regsubsets(x=x, y=y, data = candy, nvmax = 10)
candy_summary = summary(candy_model)
candy_summary
```

```
## Subset selection object
## 11 Variables  (and intercept)
##                  Forced in Forced out
## chocolate            FALSE      FALSE
## fruity               FALSE      FALSE
## caramel              FALSE      FALSE
## peanutyalmondy       FALSE      FALSE
## nougat               FALSE      FALSE
## crispedricewafer     FALSE      FALSE
## hard                 FALSE      FALSE
## bar                  FALSE      FALSE
## pluribus             FALSE      FALSE
## sugarpercent         FALSE      FALSE
## pricepercent         FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           chocolate fruity caramel peanutyalmondy nougat crispedricewafer
## 1  ( 1 )  "*"       " "    " "     " "            " "    " "
## 2  ( 1 )  "*"       " "    " "     "*"            " "    " "
## 3  ( 1 )  "*"       "*"    " "     "*"            " "    " "
## 4  ( 1 )  "*"       "*"    " "     "*"            " "    "*"
## 5  ( 1 )  "*"       "*"    " "     "*"            " "    "*"
## 6  ( 1 )  "*"       "*"    " "     "*"            " "    "*"
## 7  ( 1 )  "*"       "*"    " "     "*"            " "    "*"
## 8  ( 1 )  "*"       "*"    "*"     "*"            " "    "*"
## 9  ( 1 )  "*"       "*"    "*"     "*"            " "    "*"
## 10  ( 1 ) "*"       "*"    "*"     "*"            "*"    "*"
##           hard bar pluribus sugarpercent pricepercent
## 1  ( 1 )  " "  " " " "      " "          " "
## 2  ( 1 )  " "  " " " "      " "          " "
## 3  ( 1 )  " "  " " " "      " "          " "
## 4  ( 1 )  " "  " " " "      " "          " "
## 5  ( 1 )  " "  " " " "      "*"          " "
## 6  ( 1 )  "*"  " " " "      "*"          " "
## 7  ( 1 )  "*"  " " " "      "*"          "*"
## 8  ( 1 )  "*"  " " " "      "*"          "*"
## 9  ( 1 )  "*"  " " "*"      "*"          "*"
## 10  ( 1 ) "*"  " " "*"      "*"          "*"
```

```
#it looks like the best 1-variable model is whether or not the candy contains chocolate

which.min(candy_summary$cp) #the lowest Cp is the model with 6 variables
```

```
## [1] 6
```

```
which.min(candy_summary$bic) #the lowest BIC is the model with 3 variables
```

```
## [1] 3
```

```
which.max(candy_summary$adjr2) #the lowest Adjusted R^2 is the model with 7 variables
```

```
## [1] 7
```

```
grid <- 10^seq(10, -2, length = nrow(candy))
#ridge_model <- glmnet(x, y, alpha = 0, lambda = grid)
#lasso.model <- glmnet(x, y, alpha = 1, lambda = grid)
```