Homework 1

Tina Zhang

2019/4/21

```
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

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

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

```
library(ggplot2)
library(broom)
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16
```

```
library(e1071)

setwd("C:/Users/tzwhi/Desktop/Northwestern/311-2/R")
```

Regression using OLS

```
#a
sick <- read.csv("sick_data.csv")
sick$binaryr <-ifelse(sick$result=="Positive", 1, 0)
afit <- lm(binaryr ~ temp+bp, sick)
summary(afit)
```

```
##
## Call:
## lm(formula = binaryr ~ temp + bp, data = sick)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.32785 -0.09918 -0.02229  0.05700  0.82096
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.2134563  0.5141439  -10.14   <2e-16 ***
## temp         0.0628185  0.0050579   12.42   <2e-16 ***
## bp          -0.0082865  0.0004702  -17.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1695 on 997 degrees of freedom
## Multiple R-squared:  0.3966, Adjusted R-squared:  0.3954
## F-statistic: 327.7 on 2 and 997 DF,  p-value: < 2.2e-16
```

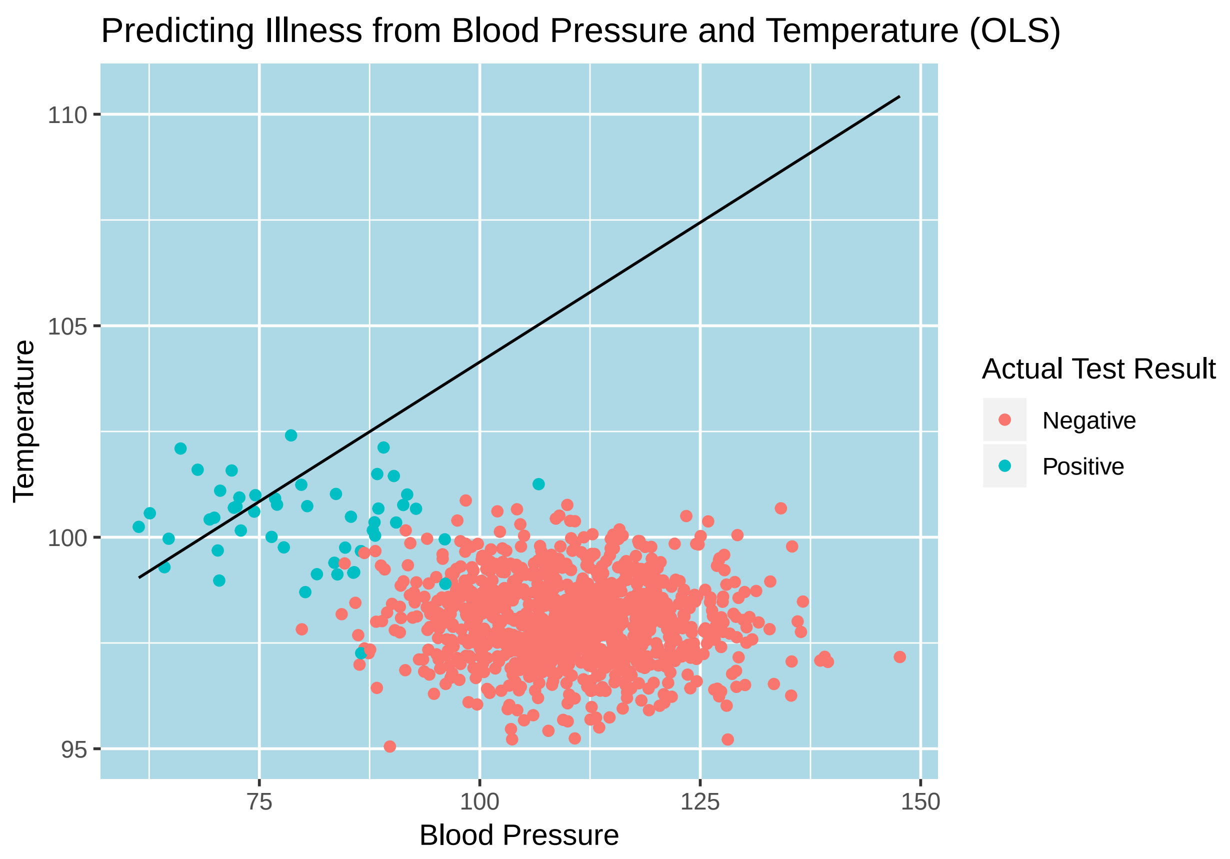
```
#b
sick$fitted <- ifelse(fitted.values(afit)>=0.5, 1, 0)
sick$rightfit <- ifelse(sick$binaryr == sick$fitted, 1, 0)
print(mean(sick$rightfit))
```

```
## [1] 0.964
```

b: The OLS regression predicts the test results reasonably well, with an accuracy rate of 96.4%.

c: The equation of the line where y-hat = 0.5 is approximately 5.713 = 0.0628temp - 0.00829bp.

```
#d
ggplot(sick) +
  geom_point(aes(bp, temp, color = result)) +
  stat_function(fun = function(bp)
    (-1/afit$coefficients[2])*(afit$coefficients[1]-0.5 + afit$coefficients[3]*bp)) + labs(x = "Blood P
  theme(panel.background = element_rect("lightblue"))
```



Regression using Logit

```
#a
logit <- glm(binaryr ~ temp+bp, data = sick, family = "binomial")
summary(logit)
```

```
##
## Call:
## glm(formula = binaryr ~ temp + bp, family = "binomial", data = sick)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.62332  -0.02253  -0.00462  -0.00093   3.02311
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -199.3267    46.8077  -4.258 2.06e-05 ***
## temp           2.3140     0.4923   4.700 2.60e-06 ***
## bp            -0.3499     0.0638  -5.485 4.14e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 397.030  on 999  degrees of freedom
## Residual deviance:  53.837  on 997  degrees of freedom
## AIC: 59.837
##
## Number of Fisher Scoring iterations: 10
```

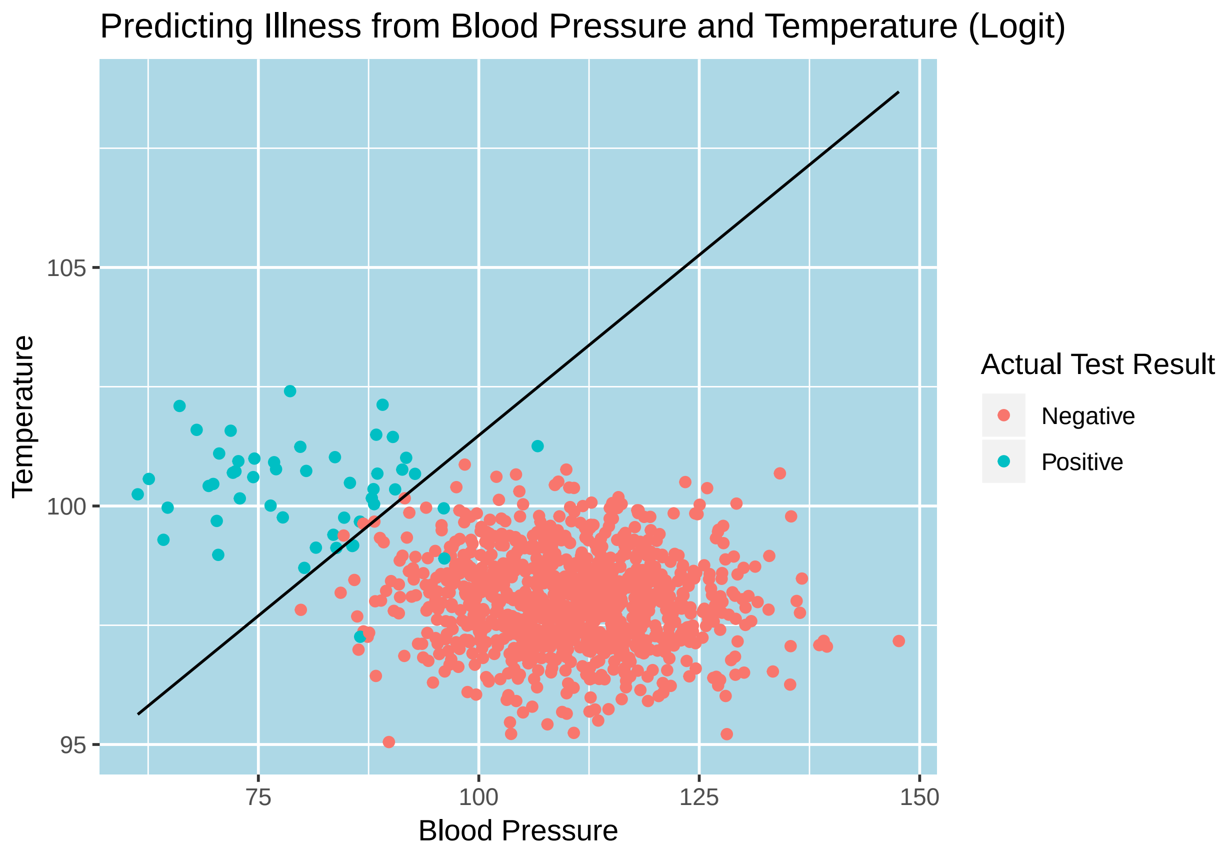
```
#b
sick$logfitted <- ifelse(fitted.values(logit)>=0.5, 1, 0)
sick$logrightfit <- ifelse(sick$binaryr == sick$logfitted, 1, 0)
print(mean(sick$logrightfit))
```

```
## [1] 0.992
```

b: The Logit regression predicts the test results very well, with an accuracy rate of 99.2%, which is an improvement upon the accuracy rate of 96.4% of the OLS model. Although, we must note that these accuracy rates are computed using the training set data.

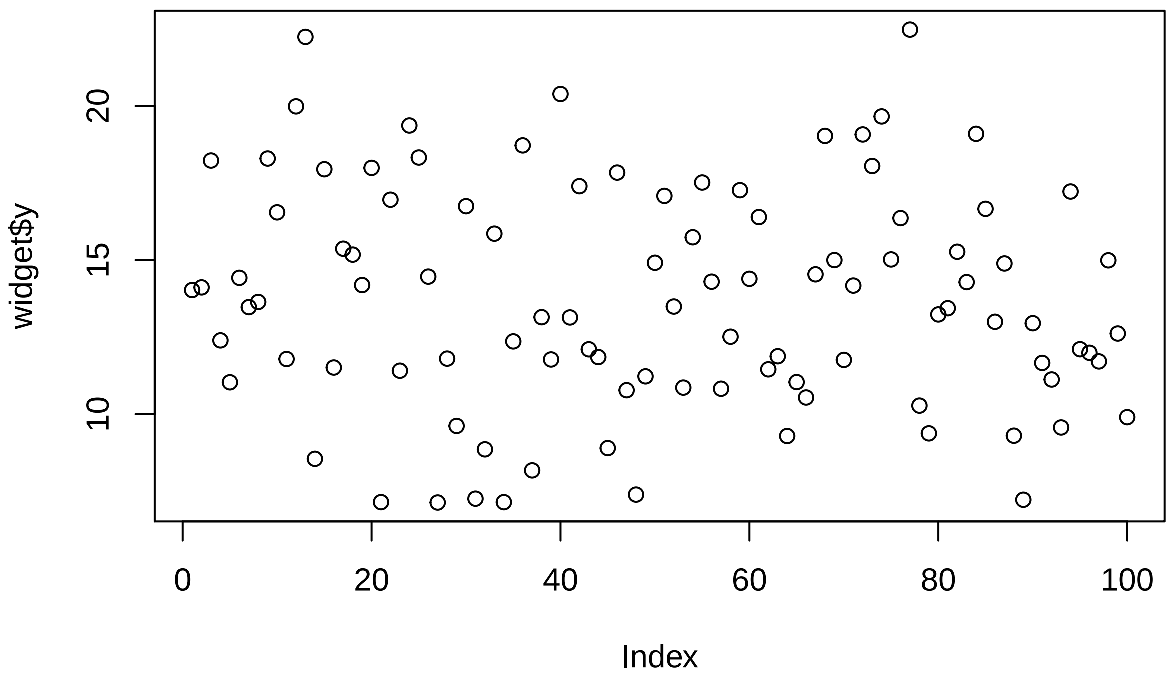
c: The equation of the line where y-hat = 0.5 is approximately 199.8267 = 2.314temp - 0.3499bp.

```
#d
ggplot(sick) +
  geom_point(aes(bp, temp, color = result)) +
  stat_function(fun = function(bp)
    (-1/logit$coefficients[2])*(logit$coefficients[1]-0.5 + logit$coefficients[3]*bp))+
  labs(x = "Blood Pressure", y = "Temperature", title = "Predicting Illness from Blood Pressure and Temp
  theme(panel.background = element_rect("lightblue"))
```

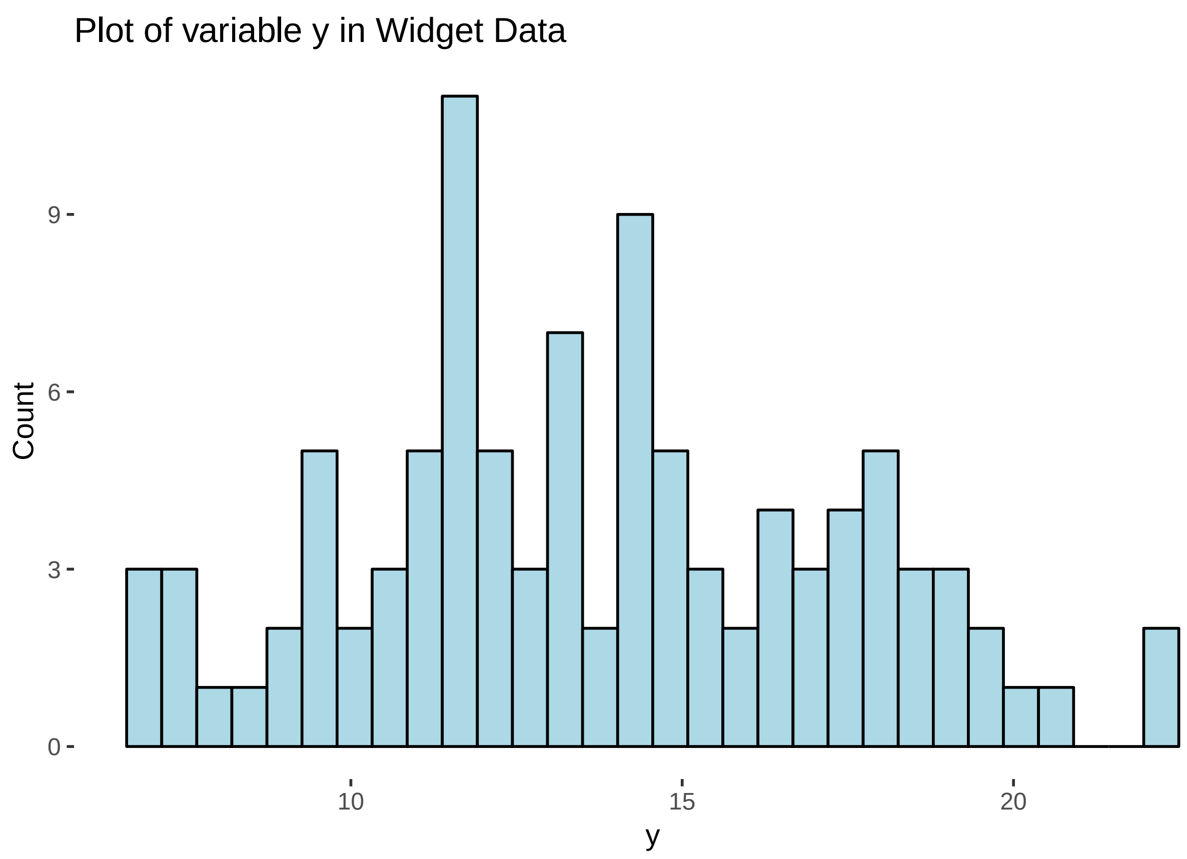


Ridge Regularization/Selection

```r
#a
widget <- read.csv("widget_data.csv")
plot(widget$y)
```

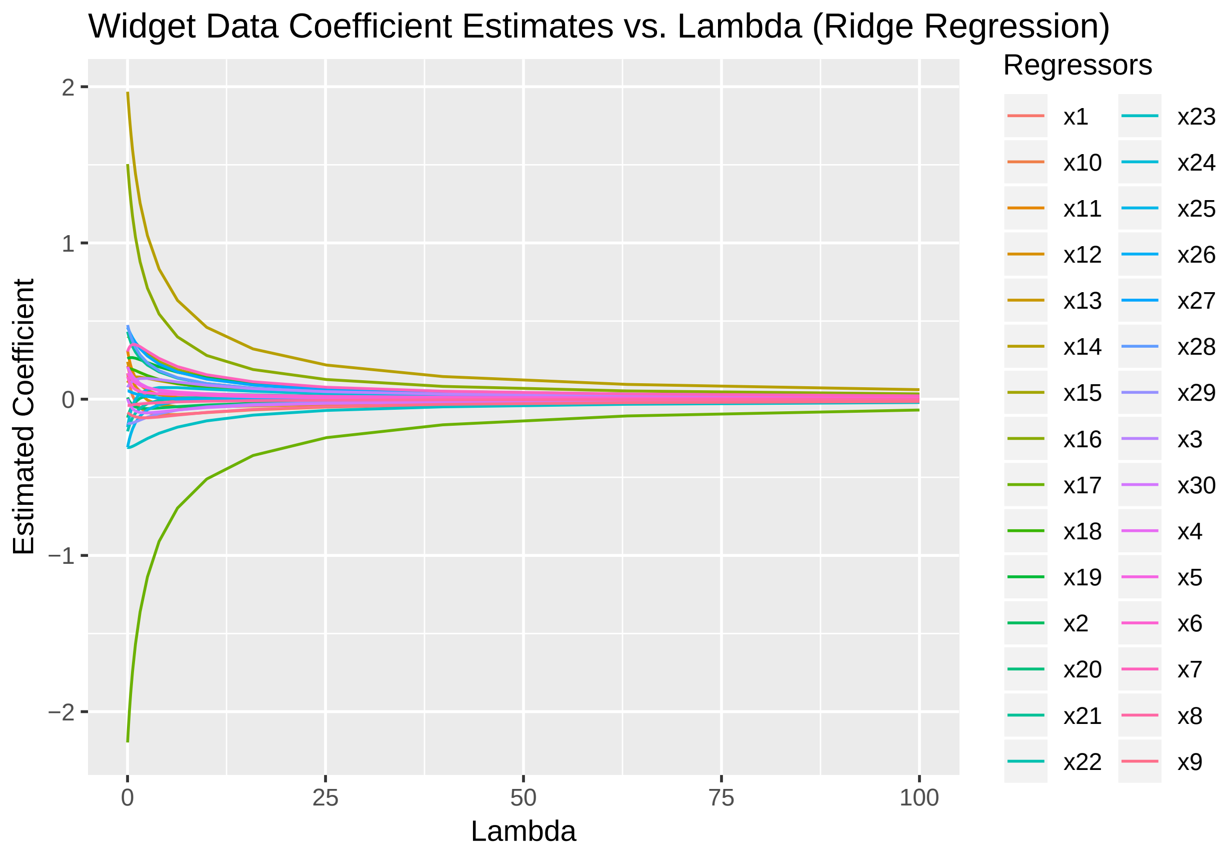


```r
#perhaps a more useful plot of y
ggplot(widget, aes(x = y)) + geom_histogram(bins = 30, color = "black", fill = "lightblue") +  labs(x =
```



```
#b
grid <- 10^seq(from = -2, to = 2, by = 0.2)
x <- as.matrix(widget[,-1])
y <- widget$y
ridge <- glmnet::glmnet(x , y, alpha = 0, lambda = grid)

#c
coeff <- broom::tidy(ridge)
nointercept <- subset(coeff, term!="(Intercept)")
ggplot(nointercept) +
  geom_line(aes(lambda, estimate, color = term)) +
  labs(x = "Lambda", y = "Estimated Coefficient", title = "Widget Data Coefficient Estimates vs. Lambda
```



```r
#Separate data into half training set and half test set
set.seed(15)
train <- sample(1:nrow(widget),nrow(widget)/2)
test <- (-train)
y.test <- y[test]

#d
ridge.mod <- glmnet::glmnet(x[train, ], y[train], alpha = 0, lambda = grid,
                            thresh =1e-12)
cv <- cv.glmnet(x[train, ], y[train], alpha = 0)
bestlam <- cv$lambda.min
bestridge <- glmnet::glmnet(x , y, alpha = 0, lambda = bestlam)
print(bestlam)
```

```
## [1] 0.9971406
```

d: Using the training set, we found that the lambda value that should minimize MSE is 0.9971406. When you run the ridge regression on the full dataset with this lambda, the coefficients estimated are:

```r
print(coefficients(bestridge))
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  5.2367716307
## x1           0.0822451506
## x2          -0.0495006188
## x3           0.1247750487
## x4           0.1253285172
## x5           0.1059687622
## x6           0.0613883891
## x7           0.3498179185
## x8          -0.0334567332
## x9          -0.1129070503
## x10         -0.0182760514
## x11          0.1353519838
## x12          0.0804436692
## x13          0.1424980297
## x14          1.4435804099
## x15          0.3610905091
## x16          1.0380657746
## x17         -1.5687103711
## x18          0.1833893832
## x19          0.2624214952
## x20         -0.0169268090
## x21          0.3023169858
## x22         -0.0893857260
## x23         -0.2928737151
## x24         -0.0009530317
## x25         -0.1500565065
## x26          0.0344821005
## x27          0.3628454357
## x28          0.3311759530
## x29         -0.1456025293
## x30         -0.0760109854
```

```r
ridge.pred <- predict(ridge.mod, s=bestlam, newx = x[test, ])
mean((ridge.pred-y.test)^2)
```

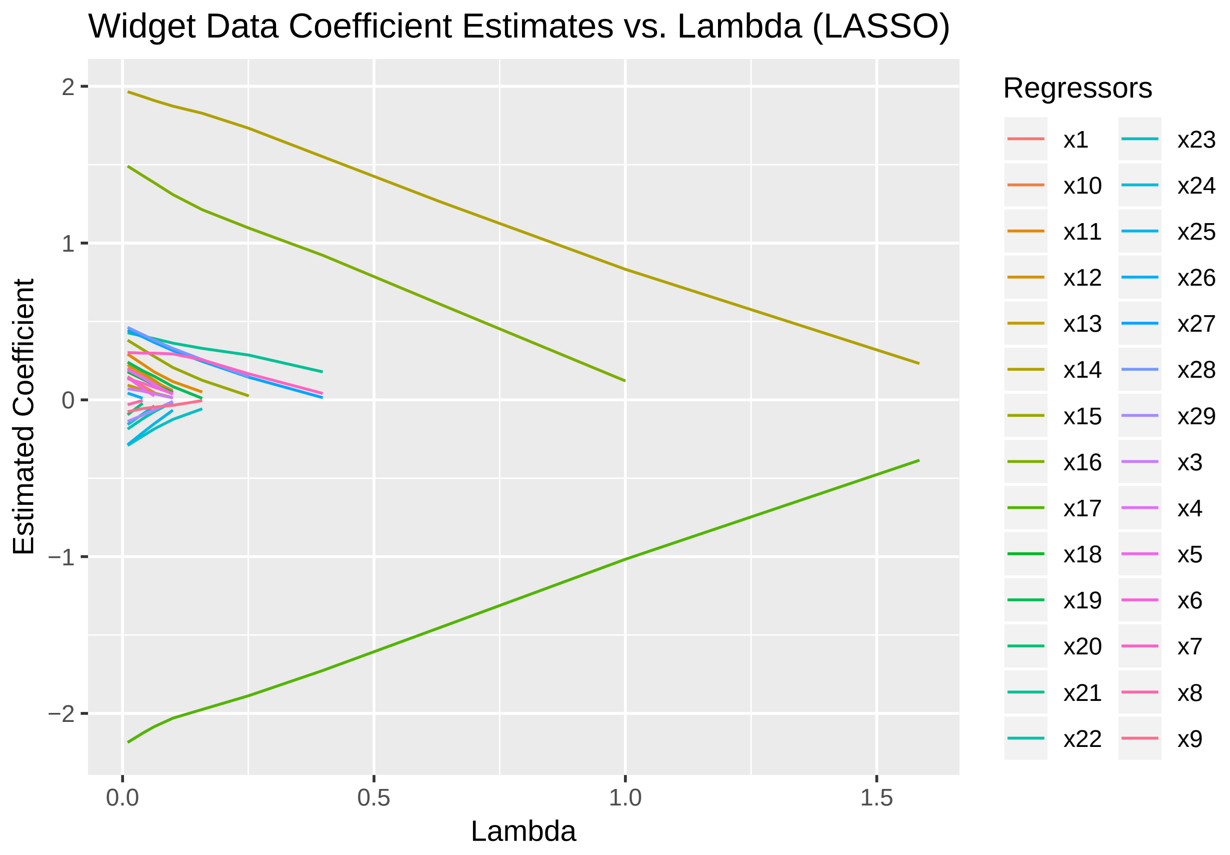
```
## [1] 6.567986
```

As shown above, for the training set and testing set generated in this question, the test MSE from the ridge regression using the "MSE-minimizing" lambda is 6.567986.

LASSO Regularization/Selection

```r
#b
lasso <- glmnet::glmnet(x , y, alpha = 1, lambda = grid)

#c
lcoeff <- broom::tidy(lasso)
lnointercept <- subset(lcoeff, term!="(Intercept)")
ggplot(lnointercept) +
  geom_line(aes(lambda, estimate, color = term)) +
  labs(x = "Lambda", y = "Estimated Coefficient", title = "Widget Data Coefficient Estimates vs. Lambda
```



```
lasso.mod <- glmnet::glmnet(x[train, ], y[train], alpha = 1, lambda = grid)

#d
lcv <- cv.glmnet(x[train, ], y[train], alpha = 1)
lbestlam <- lcv$lambda.min
print(lbestlam)
```

```
## [1] 0.2413208
```

```
bestlasso <- glmnet::glmnet(x , y, alpha = 1, lambda = lbestlam)
```

d: Using the training set, we found that the lambda value that minimizes MSE is 0.2413208. When you run LASSO on the full dataset with this lambda, the coefficients estimated are:

```
print(coefficients(bestlasso))
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept)  4.1162555
## x1           .
## x2           .
## x3           .
## x4           .
## x5           .
## x6             .
## x7             0.1748278
## x8             .
## x9             .
## x10            .
## x11            .
## x12            .
## x13            .
## x14            1.7433018
## x15            0.0346543
## x16            1.1079891
## x17           -1.8966854
## x18            .
## x19            .
## x20            .
## x21            0.2902535
## x22            .
## x23            .
## x24            .
## x25            .
## x26            .
## x27            0.1543479
## x28            0.1601486
## x29            .
## x30            .
```

```r
lasso.pred <- predict(lasso.mod, s=lbestlam, newx=x[test, ])
mean((lasso.pred-y.test)^2)
```

```
## [1] 5.282438
```

As shown above, for the training set and testing set generated in this question, the test MSE from LASSO using the "MSE-minimizing" lambda is 5.282438.

f: When comparing the coefficient estimates produced by ridge and LASSO (using the optimal lambdas), one major difference is that while ridge shrinks the coefficients of relatively nonsignificant regressors toward 0, it won't eliminate any regressor (i.e. it produces coefficients that are very close to 0 but not actually 0). This contrasts LASSO, which often produces coefficient estimates exactly equal to 0. In fact, of the 30 regressors, only 8 had nonzero coefficient estimates under LASSO.

Another observation is that in the plots of estimated coefficient vs. lambda, for the ridge regression, the lines are smooth curves, while for LASSO, the lines look straight and jagged. This may be a reflection of the fact that, in general, ridge regressions tend to shrink every dimension of the data by the same proportion, while LASSO more or less shrinks coefficients by similar incremental amounts, and sufficiently small coefficients are shrunk directly to 0.

Finally, in this case, LASSO gave a lower test set MSE than ridge ($5.28 < 6.57$), which may suggest that LASSO produces a better fit model for this data (i.e. a subset of the regressors are truly irrelevant to y and should be eliminated).

Classification

```
#a
pol <- read.csv("pol_data.csv")
set.seed(123)
poltrain <- sample(1:nrow(pol),nrow(pol)*(2/3))
traindata <- pol[poltrain, ]
testdata <- pol[-poltrain, ]
```

Naive Bayes Classification

```
#b
nb <- naiveBayes(group ~., data = traindata )
#c
nbpred <- predict(nb, testdata)
#d
table(Predicted = nbpred, Actual = testdata[ ,1])
```

```
##               Actual
## Predicted      Politicalist Socialcrat
##   Politicalist           44          1
##   Socialcrat              1         54
```

Support Vector Classification

```
#b
set.seed (1)
tune.out = tune(svm, group~., data = traindata, kernel ="linear",
                ranges =list(cost=c(0.001 , 0.01, 0.1, 1,5,10,100) ))
bestmod <- tune.out$best.model
#c
svmpred <- predict(bestmod, testdata)
#d
table(Predicted = svmpred, Actual = testdata[ ,1])
```

```
##               Actual
## Predicted      Politicalist Socialcrat
##   Politicalist           44          1
##   Socialcrat              1         54
```