CME 250 HW

Anna Khazenzon

February 16, 2015

Conceptual Exercises

(Introduction) section 2.4

Question 1

Part a

If the sample size n is extremely large, and the number of predictors p is small, we would expect a flexible method to work better. A flexible method is unlikely to overfit the data with a large n, and would instead better fit the data than an inflexible model would.

Part b

If the number of predictors p is extremely large, and the number of observations is small, we would expect an inflexible method to work better. A flexible method would likely overfit the data.

Part c

If the relationship between the predictors and response is highly non-linear, we would expect a flexible method to work better. Flexible methods are able to reveal more complex shapes than inflexible methods (such as linear regression).

Part d

If the variance of the error terms, i.e. σ2 = Var(), is extremely high, we would expect an inflexible method to work better. A flexible method would be likely to follow the errors too closely, and thus overfit the data.

Question 2

Part a

a. We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.

Regression, inference

n=500 (top firms) p=3 (profit, number of employes, industry)

Part b

b. We are considering launching a new product and wish to know whether it will be a success or a failure. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.

Classification, prediction

n=20 (similar products) p=13 (price charged, marketing budget, competition price, +10)

Part c

c. We are interested in predicting the % change in the US dollar in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2012. For each week we record the % change in the dollar, the % change in the US market, the % change in the British market, and the % change in the German market.

Regression, prediction

n=52 (weeks) p=3 (% change in dollar, % change in US market, % change in British market, % change in German market)

Question 6

A parametric statistical learning approach involves constructing a model that can be parameterized by a finite number of parameters. Parametric procedures rely on assumptions about the shape of the distribution in the underlying population, and about the form, or parameters, of the assumed distribution. On the other hand, a nonparametric model cannot be parameterized by a fixed number of parameters.

An advantage of nonparametric methods over parametric is that they can handle nonnormal data, because they do not rely on assumptions about the shape or form of the probability distribution from which the data are drawn. Using a parametric approach to analyze data which do not meet underlying assumptions can lead to misleading results.

A disadvantage of nonparametric approaches is that they can be less powerful than their paramatric counterparts when the data truly are normally distributed, so there is greater incidence of Type II errors, or false negatives. A larger sample size will be necessary to rectify this issue. Another disadvantage is that results can be more difficult to interpret.

More specifically, parametric statistical learning approaches involve a two-step model-based approach: first we make an assumption about the shape of the function f to select a model, then we use training data to fit the model. This approach is called parametric because it reduces the problem of estimating the function down to one of estimating a set of parameters. An advantage of this approach is that it simplifies the problem of estimating f with a set of parameters, rather than fitting an entirely arbitrary f. The disadvantage is that the model we choose might not match the true unknown form of f. To ameliorate this problem, we might fit a more flexible model, but this may lead to overfitting of the data (ie following the noise in the data too closely).

Nonparametric statistical learning approaches do not make explicit assumptions about the functional form of f, but rather seek an estimate for f that is reasonably close. Without relying on assumptions, nonparametric approaches can accurately fit a wider range of possible shapes for f.

(Linear Regression) section 3.7

Question 3

Part a

iii. For a fixed value of IQ and GPA, males earn more on average than females provided that the GPA is high enough.

Even with a fixed value of IQ and GPA, because of the interaction effect between GPA and gender, males earn more on average than females at a high GPA, eg a 4.0.

Part b

$$y = 50 + 20 * 4 + .07 * 110 + 35 * 1 + .01 * 110 * 4 + (-10) * 4 * 1 = 137.1$$

The predicted starting salary for a female with IQ 110 and GPA 4.0 is $137,100.

(Logistic Regression) section 4.7

Question 8

Based on these results, we should prefer to use logistic regression for classification. Even though 1-nearest neighbors has a lower average error rate than either training or test error rates for logistic regression, this is misleading, because 1-nearest neighbors is drastically overfitting the data. It is unlikely that it would do well on a training set. The classifier likely had 0% error at training, and 36% error at test, because the decision boundary is fit closely to every single data point, and would not generalize well to other data.

(Regularization) section 6.8

Question 2

Part a

The lasso, relative to least squares, is less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance. With a higher tuning parameter lambda, lasso becomes less flexible. Lasso only incorporates predictors with nonzero coefficients in the final model, so there are fewer predictors. It becomes less prone to overfitting.

Part b

Ridge regression, relative to least squares, is less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance. As the tuning parameter for lambda increases, ridge regression becomes less flexible. When lambda is equal to zero, it is equivalent to least squares.

Applied Exercises

```
library(ggplot2)
```

(Introduction) section 2.4

Question 9

Part a

```
a <- read.csv("http://www-bcf.usc.edu/~gareth/ISL/Auto.csv", header=T)
```

Quantitative parts of the predictors: mpg, cylinders, displacement, horsepower, weight, acceleration, year
Qualitative parts of the predictors: origin, name

Part b

Range of quantitative predictors:

```
summary(a)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   150    : 22
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.0   90     : 20
##  Median :23.00   Median :4.000   Median :146.0   88     : 19
##  Mean   :23.52   Mean   :5.458   Mean   :193.5   110    : 18
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0   100    : 17
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   75     : 14
##                                                  (Other):287
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2223   1st Qu.:13.80   1st Qu.:73.00   1st Qu.:1.000
##  Median :2800   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2970   Mean   :15.56   Mean   :75.99   Mean   :1.574
##  3rd Qu.:3609   3rd Qu.:17.10   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##
##              name
##  ford pinto    :  6
##  amc matador   :  5
##  ford maverick :  5
##  toyota corolla:  5
##  amc gremlin   :  4
##  amc hornet    :  4
##  (Other)       :368
```

mpg: 9-46.6 cylinders: 3-8 displacement: 68-455 horsepower: 75-150 weight: 1613-5140 acceleration: 8-24.8 year: 70-82

Part c

Mean and standard deviation of quantitative predictors:

```
## [1] mean mpg:
```

```
## [1] 23.51587
```

```
## [1] mpg sd:
```

```
## [1] 7.825804
```

```
## [1] mean cylinders:
```

```
## [1] 5.458438
```

```
## [1] cylinders sd:
```

```
## [1] 1.701577
```

```
## [1] mean displacement:
```

```
## [1] 193.5327
```

```
## [1] displacement sd:
```

```
## [1] 104.3796
```

```
## [1] mean horsepower:
```

```
## [1] 51.51637
```

```
## [1] horsepower sd:
```

```
## [1] 29.8627
```

```
## [1] mean weight:
```

```
## [1] 2970.262
```

```
## [1] weight sd:
```

```
## [1] 847.9041
```

```
## [1] mean acceleration:
```

```
## [1] 15.55567
```

```
## [1] acceleration sd:
```

```
## [1] 2.749995
```

```
## [1] mean year:
```

```
## [1] 75.99496
```

```
## [1] year sd:
```

```
## [1] 3.690005
```

Part d

```
#remove 10th-85th obs
a_subset <- a[10:85,]

#range & mean
summary(a_subset)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 70.0   150    : 6
##  1st Qu.:14.00   1st Qu.:4.000   1st Qu.:109.2   88     : 5
##  Median :19.00   Median :6.000   Median :199.5   90     : 5
##  Mean   :19.62   Mean   :5.829   Mean   :220.9   95     : 5
##  3rd Qu.:25.00   3rd Qu.:8.000   3rd Qu.:323.5   100    : 4
##  Max.   :35.00   Max.   :8.000   Max.   :455.0   175    : 3
##                                                  (Other):48
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2249   1st Qu.:13.00   1st Qu.:70.00   1st Qu.:1.000
##  Median :2883   Median :14.50   Median :71.00   Median :1.000
##  Mean   :3124   Mean   :14.85   Mean   :71.11   Mean   :1.474
##  3rd Qu.:4106   3rd Qu.:16.50   3rd Qu.:72.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :23.50   Max.   :72.00   Max.   :3.000
##
##                  name
##  amc gremlin       : 2
##  chevrolet impala  : 2
##  datsun pl510      : 2
##  ford galaxie 500  : 2
##  plymouth fury iii : 2
##  amc ambassador dpl: 1
##  (Other)           :65
```

```
#MPG
sd(a_subset$mpg)
```

```
## [1] 6.123108
```

```
#Cylinders
sd(a_subset$cylinders)
```

```
## [1] 1.857512
```

```
#Displacement
sd(a_subset$displacement)
```

```
## [1] 119.2912
```

```
#Horsepower
sd(as.numeric(a_subset$horsepower))
```

```
## [1] 29.03277
```

```
#Weight
sd(a_subset$weight)
```

```
## [1] 981.1961
```

```
#Acceleration
sd(a_subset$acceleration)
```

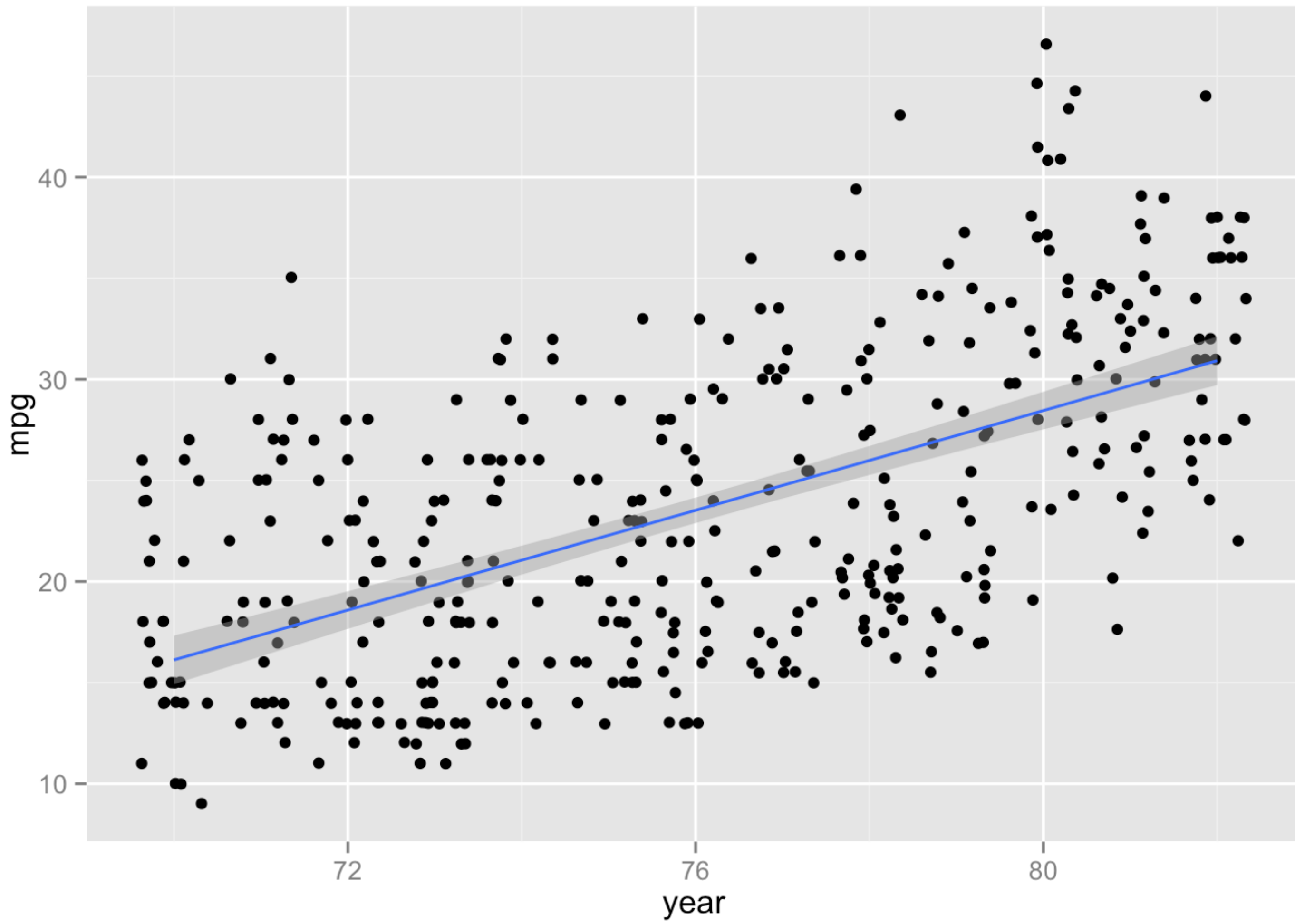
```
## [1] 2.940544
```

```
#Year
sd(a_subset$year)
```

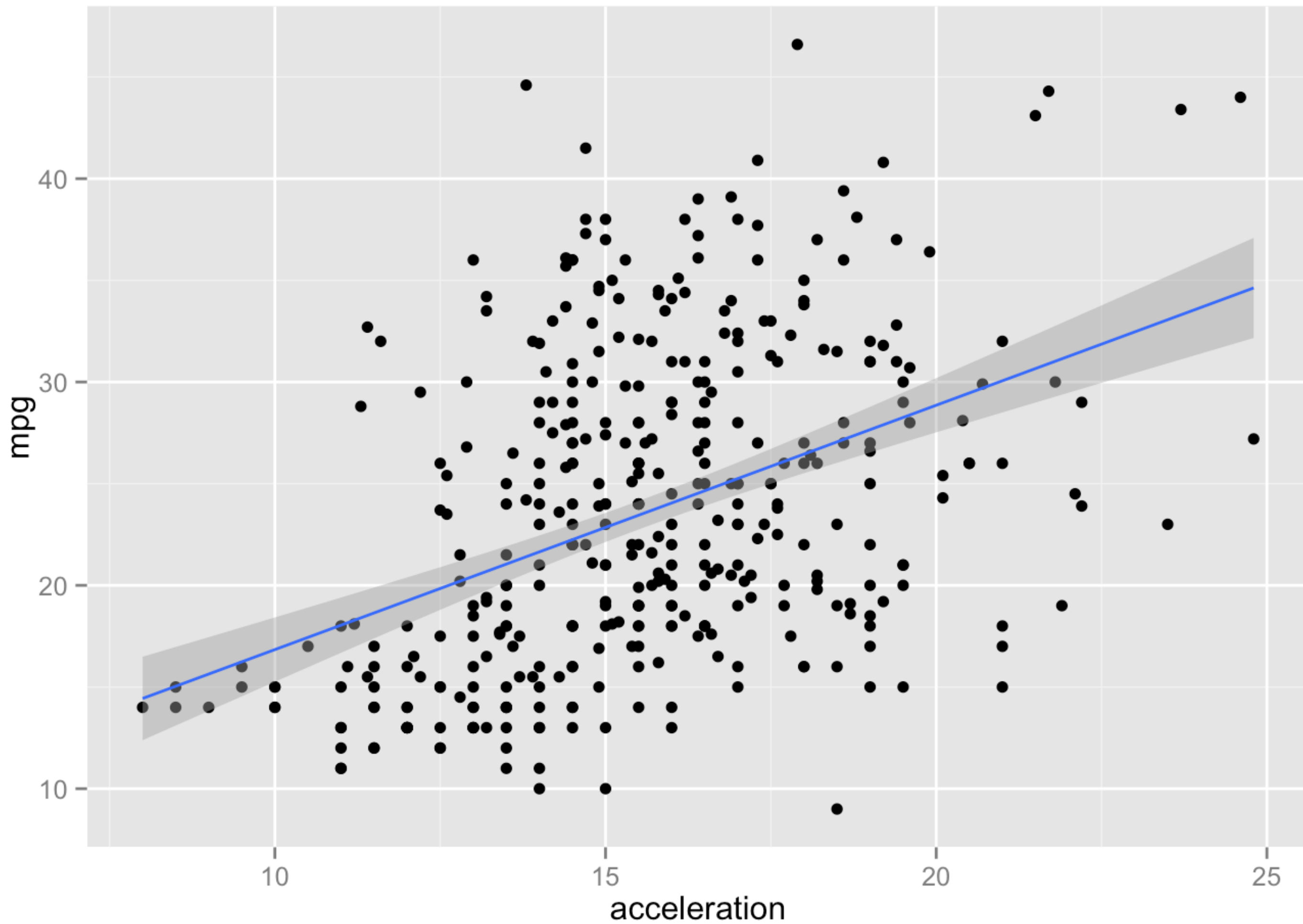
```
## [1] 0.7929514
```

Part e

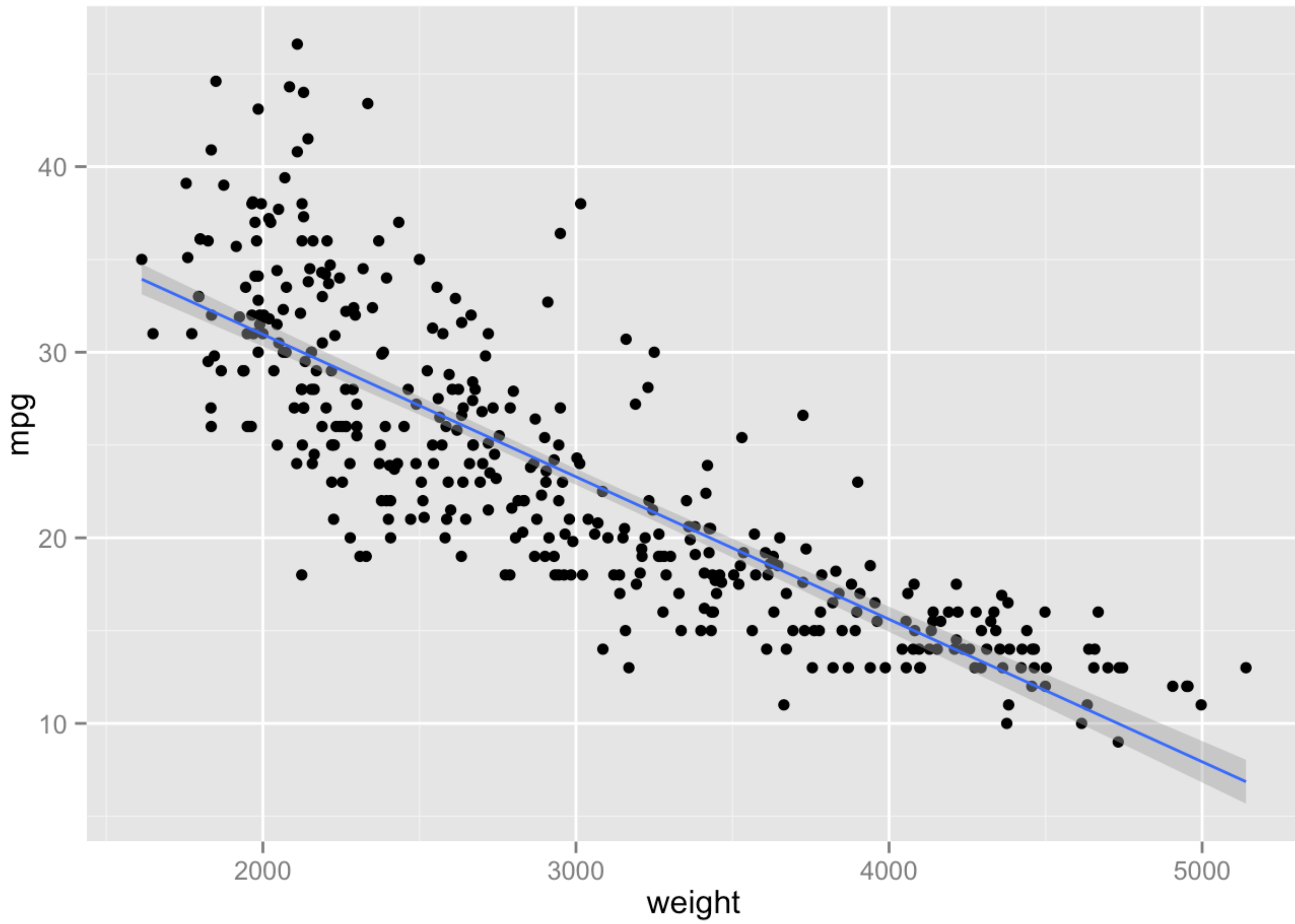
```
#plot data (scatter)
ggplot(a,aes(x=year,y=mpg)) + geom_jitter() + stat_smooth(method="lm")
```



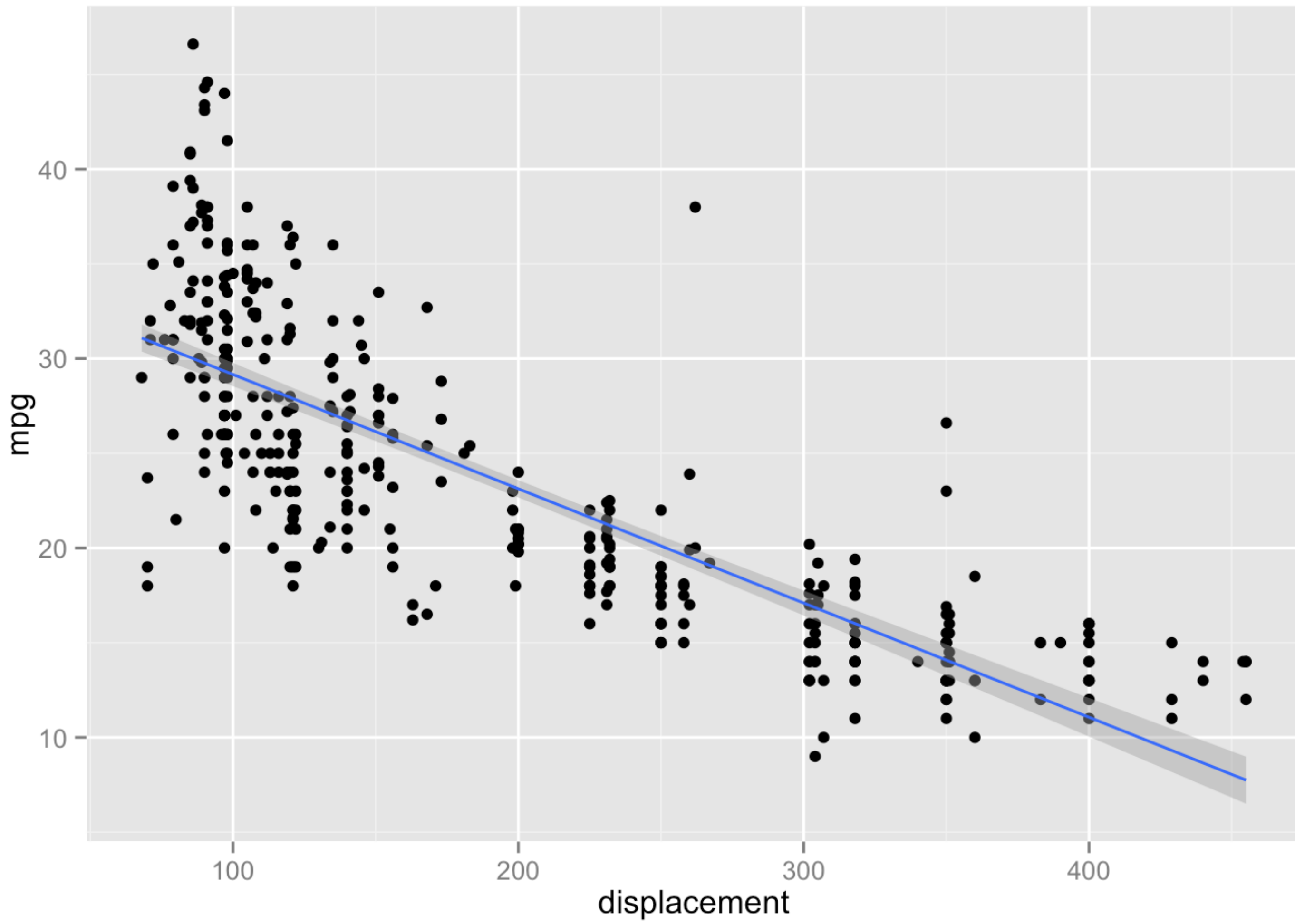

```
ggplot(a,aes(x=acceleration,y=mpg)) + geom_point() + stat_smooth(method="lm")
```



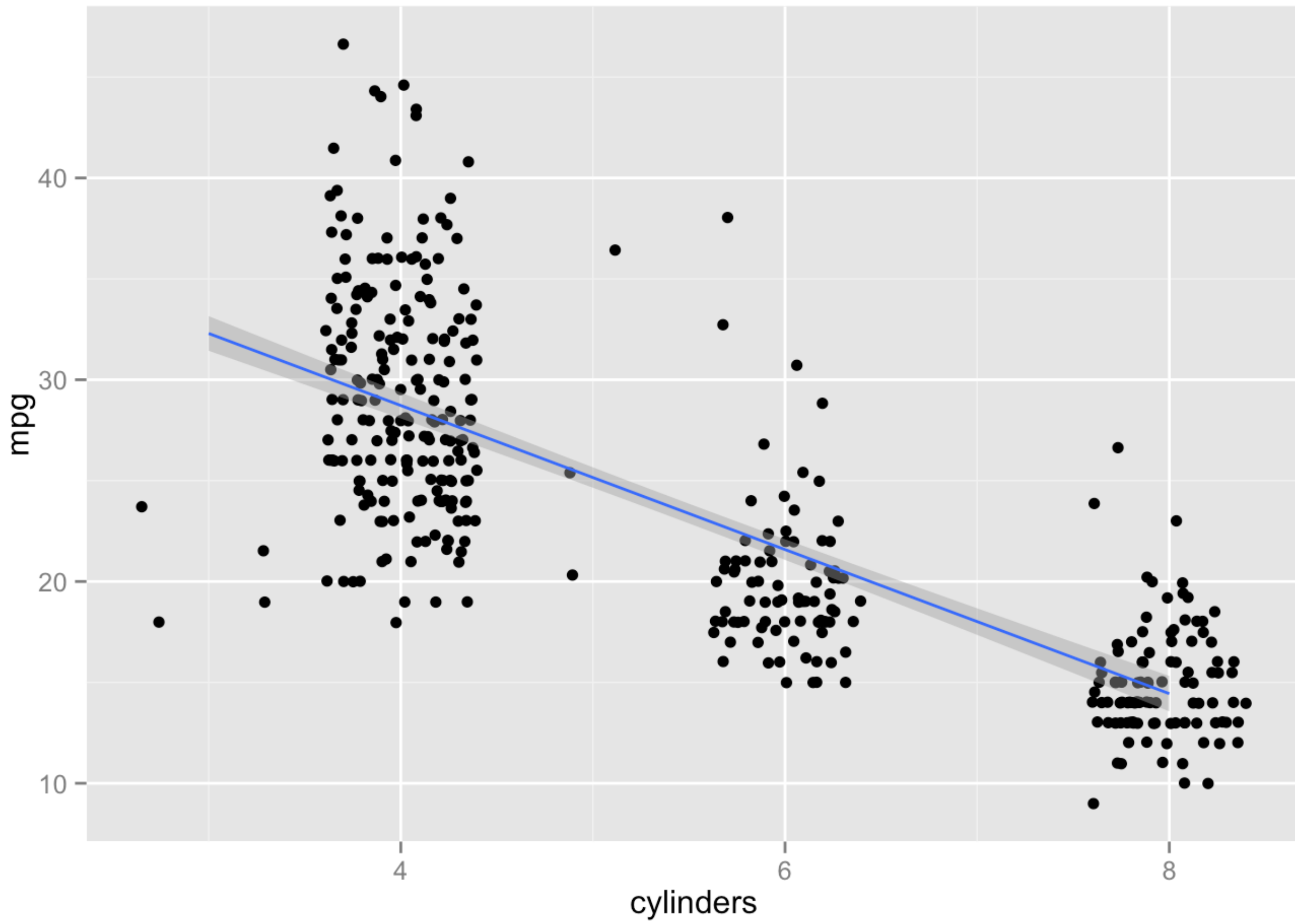

```
ggplot(a,aes(x=weight,y=mpg)) + geom_point() + stat_smooth(method="lm")
```



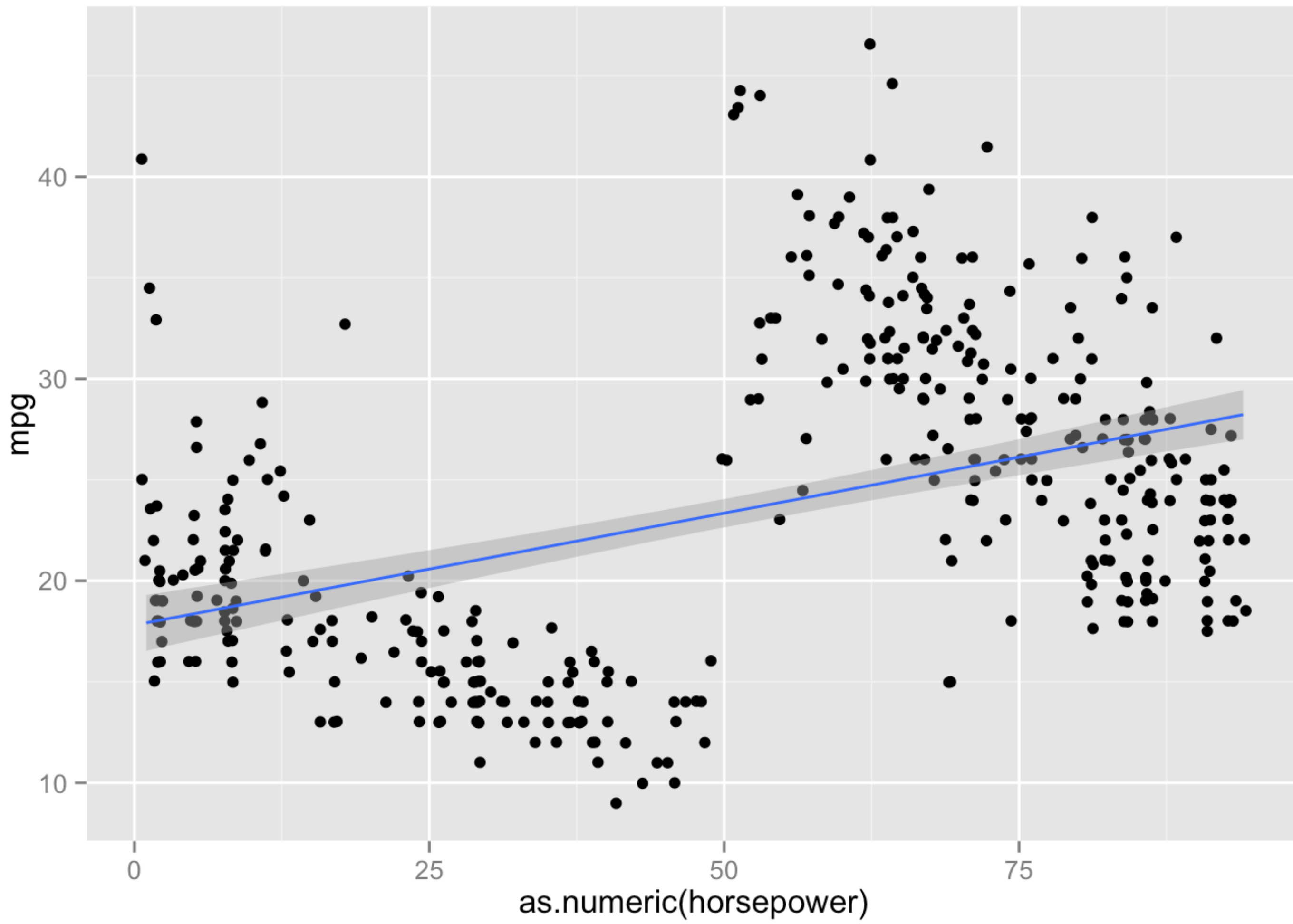

```
ggplot(a,aes(x=displacement,y=mpg)) + geom_point() + stat_smooth(method="lm")
```



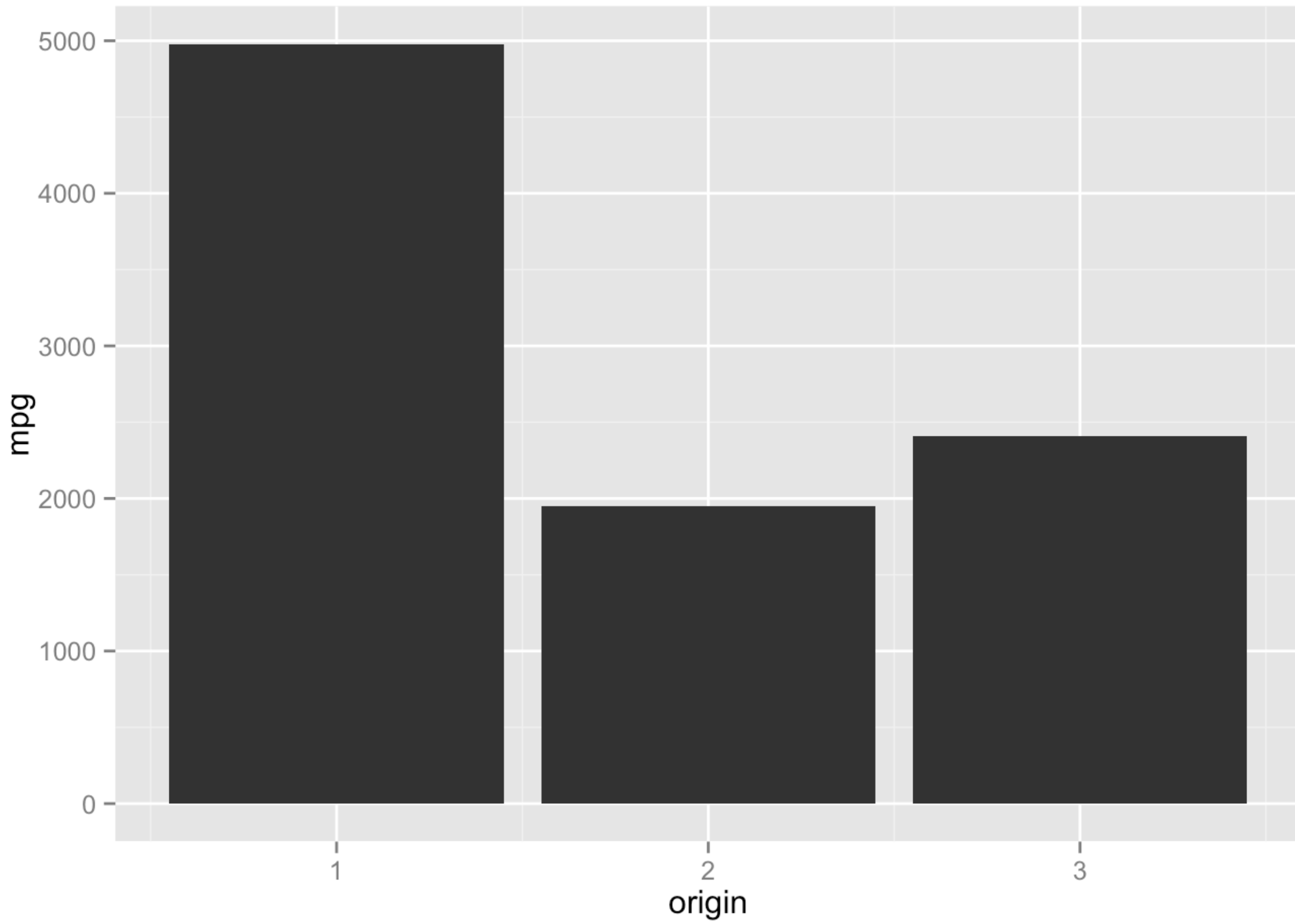

```
ggplot(a,aes(x=cylinders,y=mpg)) + geom_jitter() + stat_smooth(method="lm")
```




```
ggplot(a,aes(x=as.numeric(horsepower),y=mpg)) + geom_jitter() + stat_smooth(method="lm")
```




```
ggplot(a,aes(x=origin,y=mpg)) + geom_bar(stat="identity")
```



Year, acceleration, horsepower appear to have a positive relationship with mpg. Weight, displacement, cylinders appear to have a negative relationship with mpg. American cars have higher mpg than European or Japanese.

Part f

The plot of effect of year on mpg suggests that more recent models of cars have higher miles per gallon, because we see an increasing slope across time. acceleration

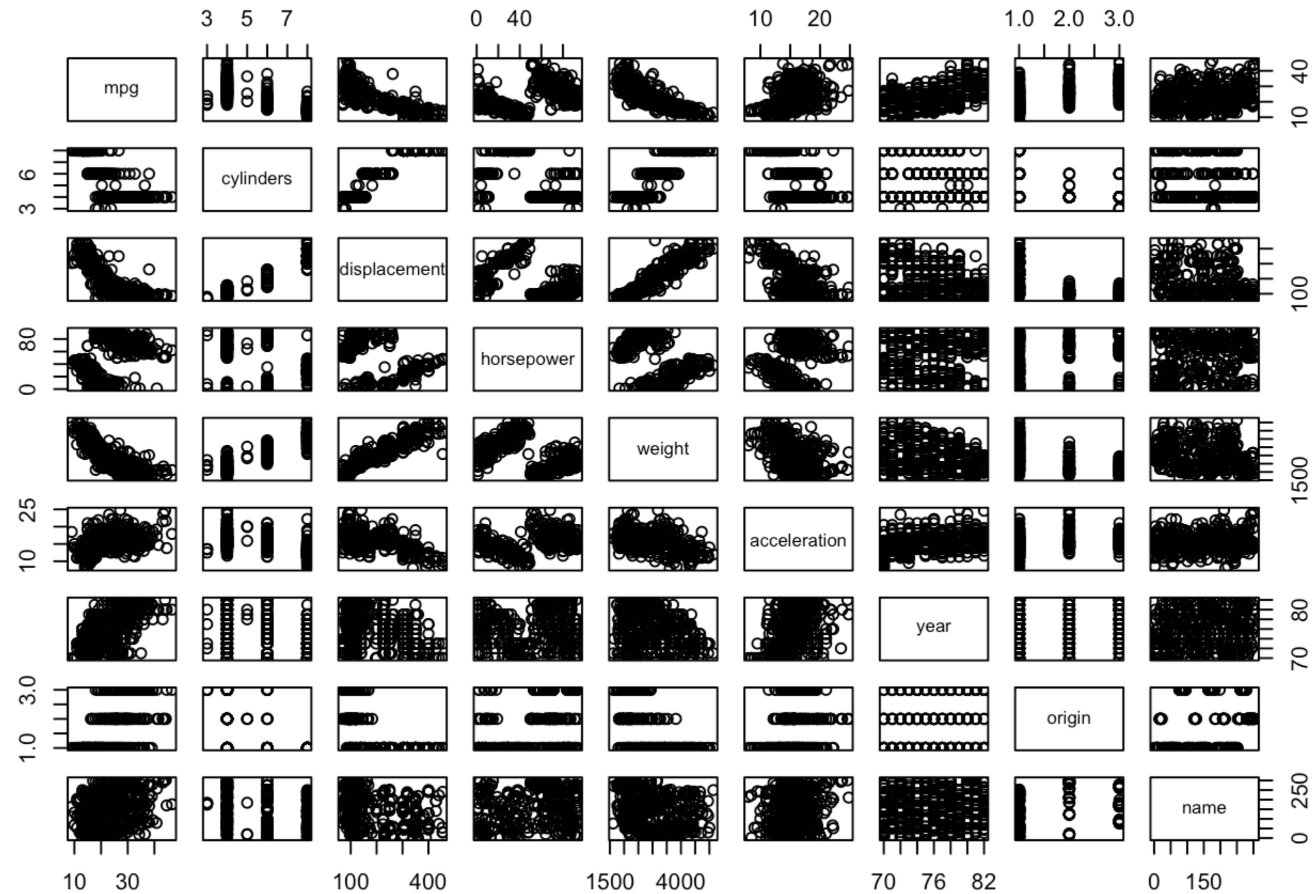
The plot of the effect of weight on mpg suggests that heavier cars have lower miles per gallon. This is plausible, since it would take more gasoline to move a heavier car.

(Linear Regression) section 3.7

Question 9

Part a

```
pairs(a)
```



Part b

```
a_num <- a[,1:7]
a_num$horsepower <- as.numeric(a_num$horsepower)
cor(a_num)
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7762599   -0.8044430  0.4228227 -0.8317389
## cylinders    -0.7762599  1.0000000    0.9509199 -0.5466585  0.8970169
## displacement -0.8044430  0.9509199    1.0000000 -0.4820705  0.9331044
## horsepower    0.4228227 -0.5466585   -0.4820705  1.0000000 -0.4821507
## weight       -0.8317389  0.8970169    0.9331044 -0.4821507  1.0000000
## acceleration  0.4222974 -0.5040606   -0.5441618  0.2662877 -0.4195023
## year          0.5814695 -0.3467172   -0.3698041  0.1274167 -0.3079004
##              acceleration       year
## mpg             0.4222974  0.5814695
## cylinders      -0.5040606 -0.3467172
## displacement   -0.5441618 -0.3698041
## horsepower      0.2662877  0.1274167
## weight         -0.4195023 -0.3079004
## acceleration    1.0000000  0.2829009
## year            0.2829009  1.0000000
```

Part c

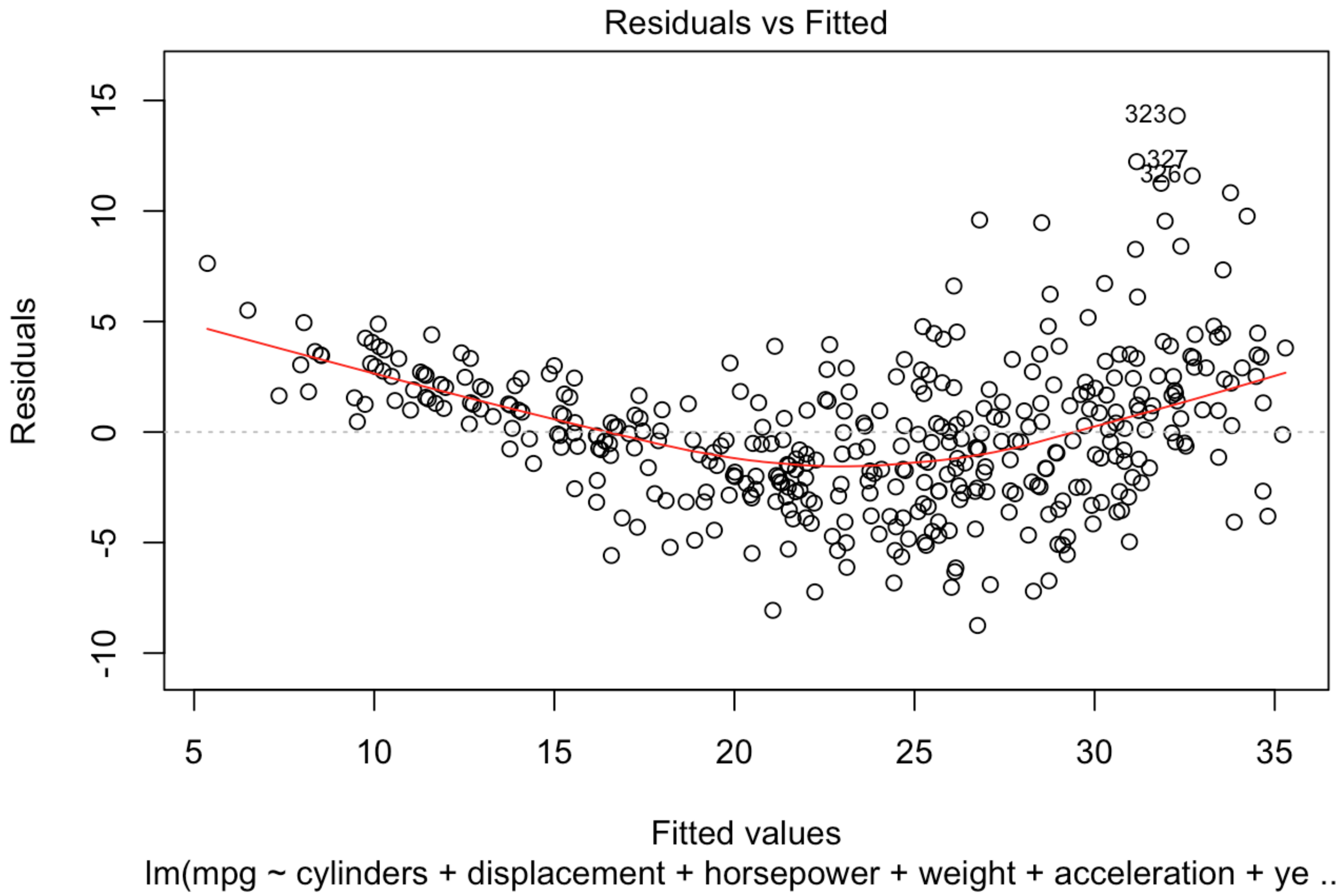
```
rs <- lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + year, data=a_n
um)
summary(rs)
```

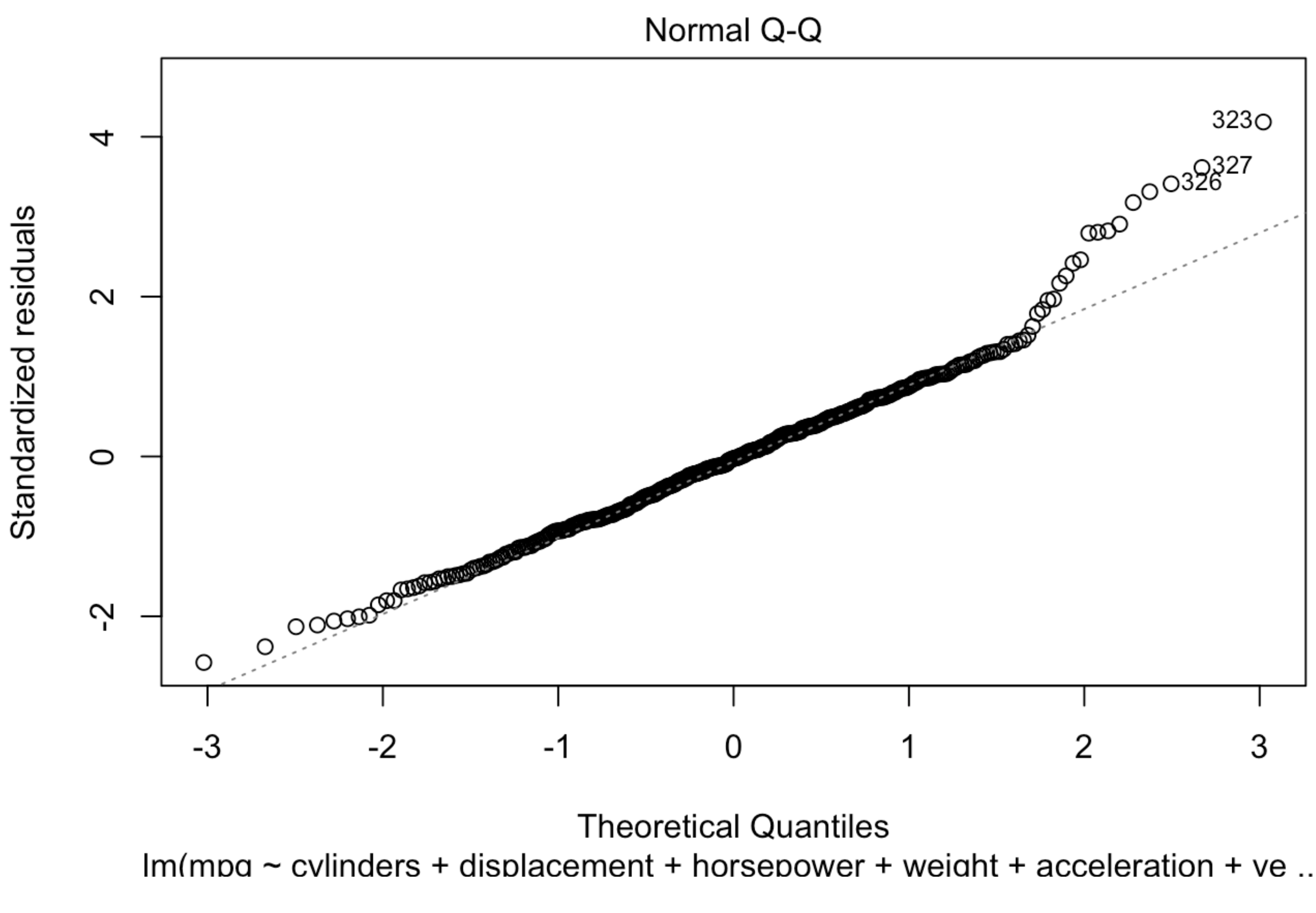
```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + year, data = a_num)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7503 -2.4208 -0.0873  1.9823 14.3087
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.640e+01  4.271e+00  -3.839 0.000144 ***
## cylinders    -1.569e-01  3.473e-01  -0.452 0.651789
## displacement  6.299e-03  7.233e-03   0.871 0.384355
## horsepower    8.980e-03  7.013e-03   1.280 0.201131
## weight       -6.827e-03  5.975e-04 -11.427  < 2e-16 ***
## acceleration  8.384e-02  7.854e-02   1.067 0.286408
## year          7.640e-01  5.085e-02  15.025  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.432 on 390 degrees of freedom
## Multiple R-squared:  0.8106, Adjusted R-squared:  0.8076
## F-statistic: 278.1 on 6 and 390 DF,  p-value: < 2.2e-16
```

There is a significant negative relationship between a car's weight and its mpg, $t(390)=-11.43$, $p<.001$. Converseley, there is a significant positive relationship between the year a car was made and its mpg, meaning modern cars have higher mpg, $t(390)=15.025$, $p<.001$

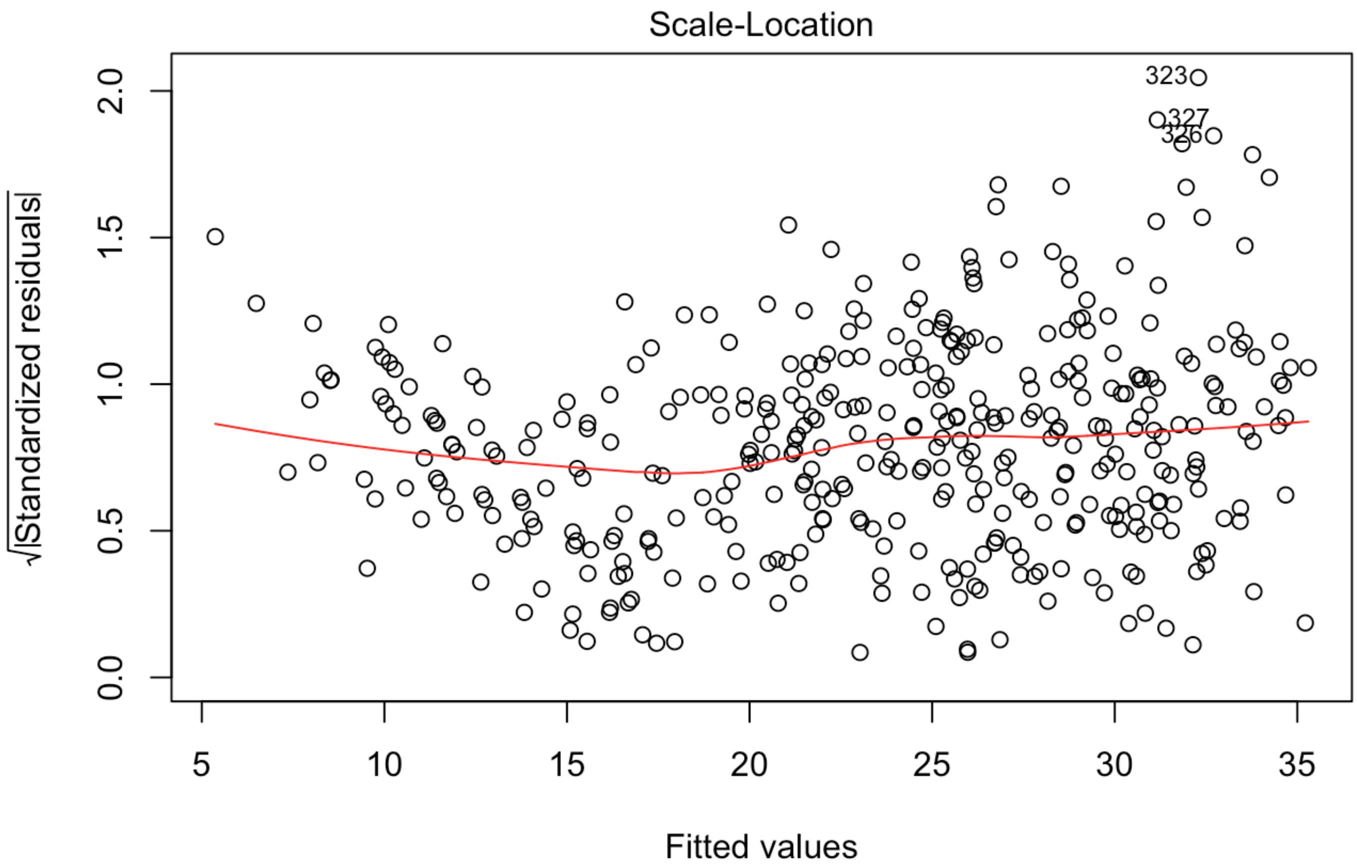
Part d

```
#diagnostic plots of linear reg fit
plot(rs)
```

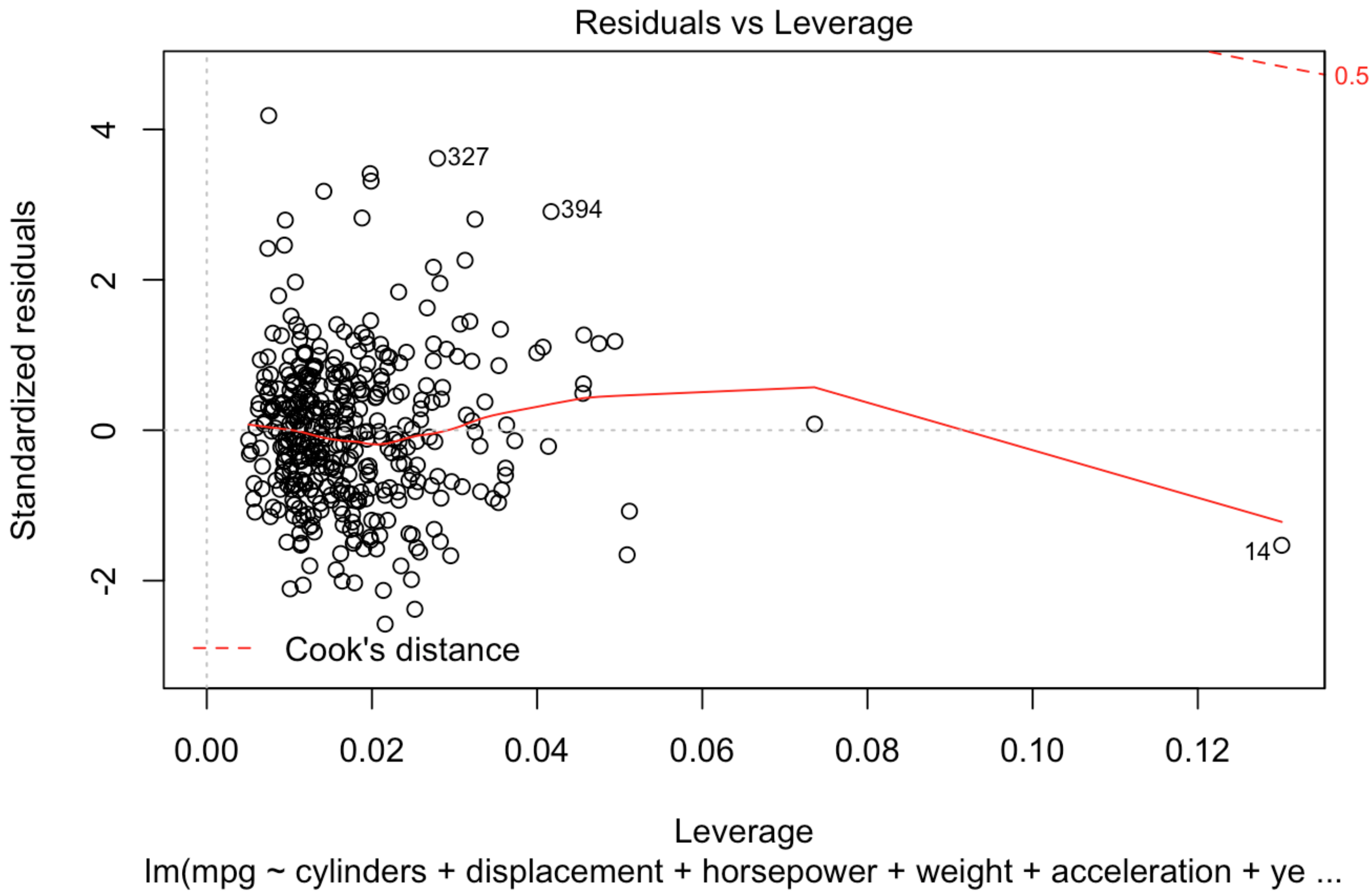

Residuals vs Fitted
Residuals
Fitted values
lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + ye ...
323
327
326
15
10
5
0
-5
-10
5
10
15
20
25
30
35


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + ye ...
323
327
326
4
2
0
-2
-3
-2
-1
0
1
2
3

lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + ye ...



lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + ye ...



The residual plot shows heteroscedasticity, as there is a dependency between the residuals and the fitted values, indicating that our data could be fit better with another model. Data points 323, 326, and 327 appear to be large outliers, and the leverage plot seems to identify point 14 as having unusually high leverage.

Part e

```
rs1 <- lm(mpg ~ weight * acceleration, data=a_num)
summary(rs1)
```

```
##
## Call:
## lm(formula = mpg ~ weight * acceleration, data = a_num)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.5831  -2.7125  -0.3628   2.3091  15.6577
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.855e+01  4.878e+00   5.854 1.01e-08 ***
## weight               -3.254e-03  1.464e-03  -2.222 0.026844 *
## acceleration          1.098e+00  3.098e-01   3.544 0.000442 ***
## weight:acceleration -2.753e-04  9.704e-05  -2.837 0.004789 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.271 on 393 degrees of freedom
## Multiple R-squared:  0.7044, Adjusted R-squared:  0.7021
## F-statistic: 312.1 on 3 and 393 DF,  p-value: < 2.2e-16
```

```
rs2<- lm(mpg ~ weight * origin, data=a)
summary(rs2)
```

```
##
## Call:
## lm(formula = mpg ~ weight * origin, data = a)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5036  -2.8495  -0.4089   2.2353  15.5098
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   38.7748668  2.1982519  17.639  < 2e-16 ***
## weight        -0.0054943  0.0007847  -7.002  1.1e-11 ***
## origin         4.2924911  1.4993493   2.863  0.00442 **
## weight:origin -0.0013306  0.0006258  -2.126  0.03409 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.267 on 393 degrees of freedom
## Multiple R-squared:  0.7049, Adjusted R-squared:  0.7027
## F-statistic: 312.9 on 3 and 393 DF,  p-value: < 2.2e-16
```

There is a significant interaction between weight and acceleration in predicting mpg. There is also a significant interaction between weight and origin in predicting mpg.

Part f

```
rs1 <- lm(mpg ~ log(displacement) + displacement, data=a_num)
summary(rs1)
```

```
##
## Call:
## lm(formula = mpg ~ log(displacement) + displacement, data = a_num)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.7532  -2.4074  -0.3931   2.1257  20.1257
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        96.12128    8.54628  11.247  < 2e-16 ***
## log(displacement) -14.65619    2.05260  -7.140 4.52e-12 ***
## displacement        0.01284    0.01046   1.227     0.22
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.385 on 394 degrees of freedom
## Multiple R-squared:  0.6876, Adjusted R-squared:  0.686
## F-statistic: 433.5 on 2 and 394 DF,  p-value: < 2.2e-16
```

```
rs2 <- lm(mpg ~ poly(weight, 2), data=a_num)
summary(rs2)
```

```
##
## Call:
## lm(formula = mpg ~ poly(weight, 2), data = a_num)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.6632  -2.7081  -0.3426   1.8221  16.0931
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        23.5159     0.2103 111.838  < 2e-16 ***
## poly(weight, 2)1 -129.5280     4.1895 -30.917  < 2e-16 ***
## poly(weight, 2)2   23.6488     4.1895   5.645 3.17e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.19 on 394 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7134
## F-statistic: 493.9 on 2 and 394 DF,  p-value: < 2.2e-16
```

There is a significant effect of log displacement on mpg. There is also a significant quadratic effect of weight on mpg, indicating that there might be a nonlinear relationship between how heavy a car is and how gas-efficient it is.

Question 13

Part a

```
set.seed(2)

x <- rnorm(n = 100, mean = 0, sd = 1)
```

Part b

```
eps <- rnorm(n = 100, mean = 0, sd = .25)
```

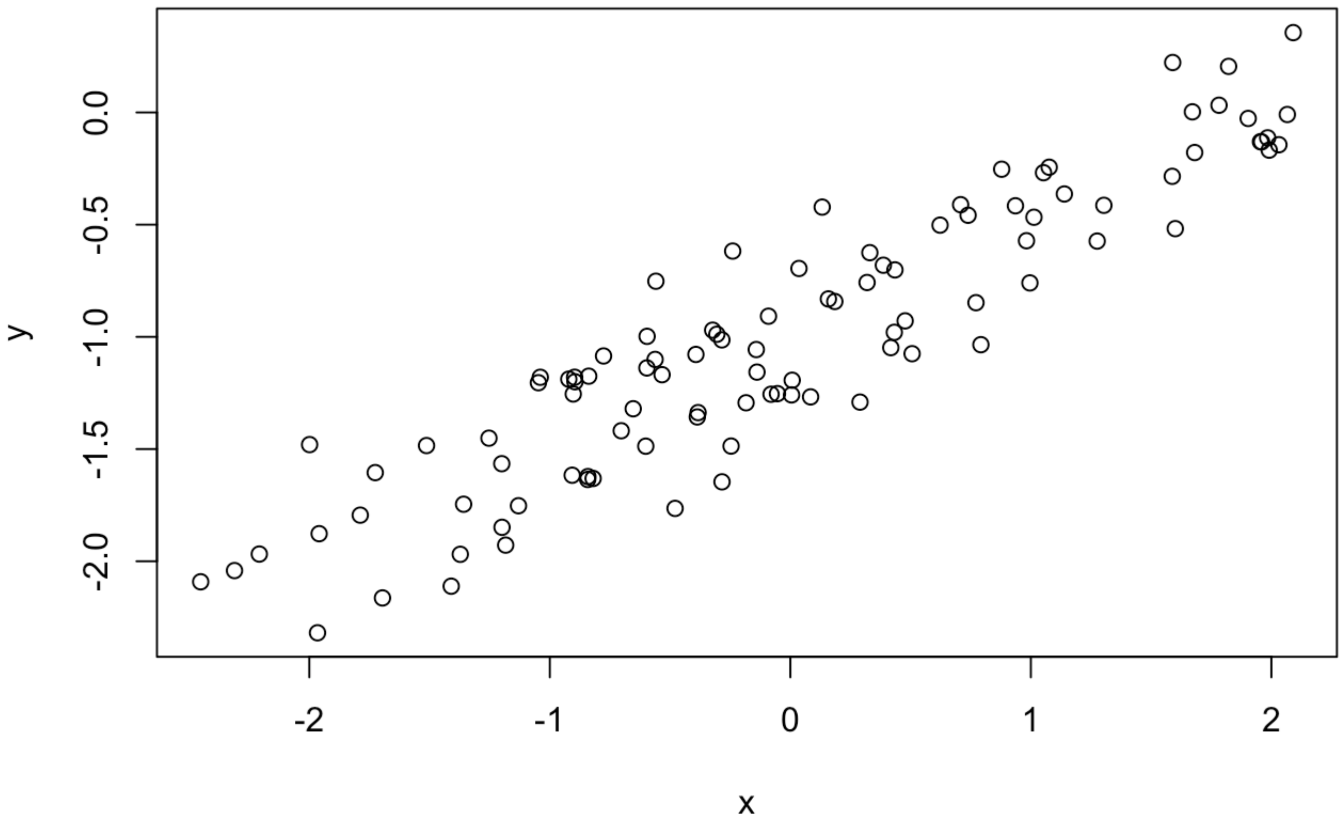
Part c

```
y <- -1 + .5 * x + eps
```

Vector y has length 100; there are as many values of y as there are of x. In this linear model B0 is -1, and B1 is 0.5.

Part d

```
plot(x, y)
```



There appears to be a strong positive linear relationship between x and y.

Part e

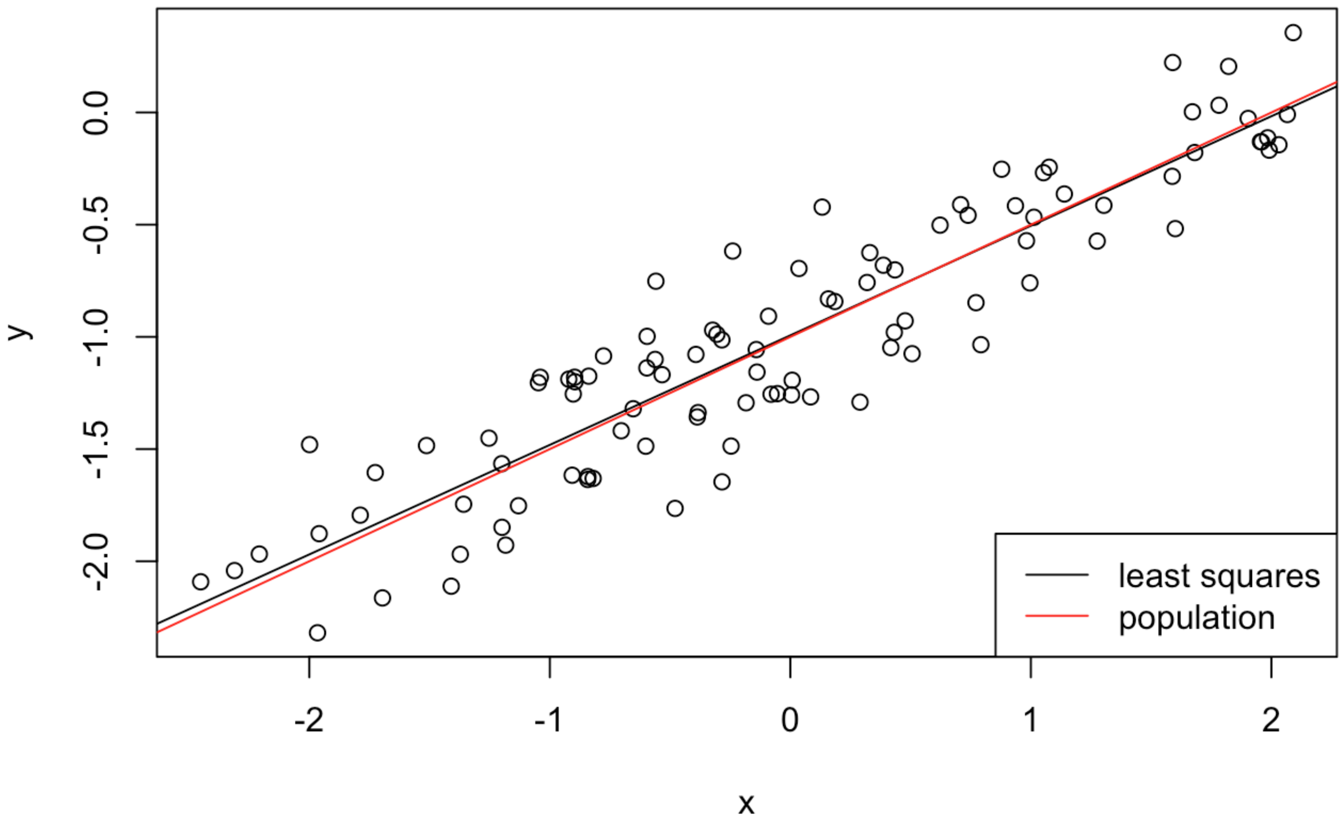
```
rs <- lm(y ~ x)
summary(rs)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.53739 -0.20490  0.03425  0.17986  0.51399
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.99306    0.02473  -40.16   <2e-16 ***
## x            0.48822    0.02142   22.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2472 on 98 degrees of freedom
## Multiple R-squared:  0.8414, Adjusted R-squared:  0.8397
## F-statistic: 519.8 on 1 and 98 DF,  p-value: < 2.2e-16
```

The observed least squares linear model shows a highly significant positive relationship between x and y. In this model, B0 is -1.01 and B1 is .500, nearly identical to the B0 and B1 above.

Part f

```
plot(x, y)
abline(rs)
abline(a = -1, b = .5, col = "red")
legend("bottomright", lty=1, c("least squares", "population"), col=c(1,2))
```



Part g

```
rs_poly <- lm(y ~ poly(x,2))
summary(rs_poly)
```

```
##
## Call:
## lm(formula = y ~ poly(x, 2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52305 -0.19662  0.02365  0.18825  0.52715
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.00805    0.02477 -40.701   <2e-16 ***
## poly(x, 2)1  5.63593    0.24767  22.756   <2e-16 ***
## poly(x, 2)2  0.19753    0.24767   0.798    0.427
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2477 on 97 degrees of freedom
## Multiple R-squared:  0.8424, Adjusted R-squared:  0.8391
## F-statistic: 259.2 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
anova(rs, rs_poly)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ x
## Model 2: y ~ poly(x, 2)
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     98 5.989
## 2     97 5.950  1  0.039018 0.6361 0.4271
```

There is no evidence that the quadratic term improves the model fit. The quadratic term x^2 is not significantly predictive of y, *t*(97) = -1.40), *p* = 0.164. When comparing the linear and polynomial models directly, we find that the model is not significantly improved, *F*(1, 97) = 1.97, *p* = 0.164.

(Regularization and Cross-validation) section 6.8

Question 9

Part a

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 1.9-8
```

```
library(ISLR)

#going to predict number applications received given other factors

x = model.matrix(Apps ~ ., College)[,-1]
y = College$Apps
dim(College) # 777 rows, 18 cols
```

```
## [1] 777  18
```

```
#split into training/test
set.seed(1)
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
y.test = y[test]
```

Part b

```
summary(lm(y ~ x, subset = train))
```

```
##
## Call:
## lm(formula = y ~ x, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5276.1  -473.2   -63.9   351.9  6574.0
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    78.15204  600.84427   0.130 0.896581
## xPrivateYes  -757.22843  205.47577  -3.685 0.000263 ***
## xAccept         1.67981    0.05196  32.329  < 2e-16 ***
## xEnroll        -0.62380    0.27629  -2.258 0.024544 *
## xTop10perc     67.45654    8.45231   7.981 1.84e-14 ***
## xTop25perc    -22.37500    6.57093  -3.405 0.000734 ***
## xF.Undergrad   -0.06126    0.05468  -1.120 0.263258
## xP.Undergrad    0.04745    0.06248   0.760 0.448024
## xOutstate      -0.09227    0.02889  -3.194 0.001524 **
## xRoom.Board     0.24513    0.07300   3.358 0.000867 ***
## xBooks          0.09086    0.36826   0.247 0.805254
## xPersonal       0.05886    0.09260   0.636 0.525455
## xPhD           -8.89027    7.20890  -1.233 0.218271
## xTerminal      -1.71947    8.22589  -0.209 0.834539
## xS.F.Ratio     -5.75201   21.32871  -0.270 0.787554
## xperc.alumni   -1.46681    6.28702  -0.233 0.815652
## xExpend         0.03487    0.01928   1.808 0.071361 .
## xGrad.Rate      7.57567    4.69602   1.613 0.107551
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1087 on 370 degrees of freedom
## Multiple R-squared:  0.9397, Adjusted R-squared:  0.9369
## F-statistic: 339.3 on 17 and 370 DF,  p-value: < 2.2e-16
```

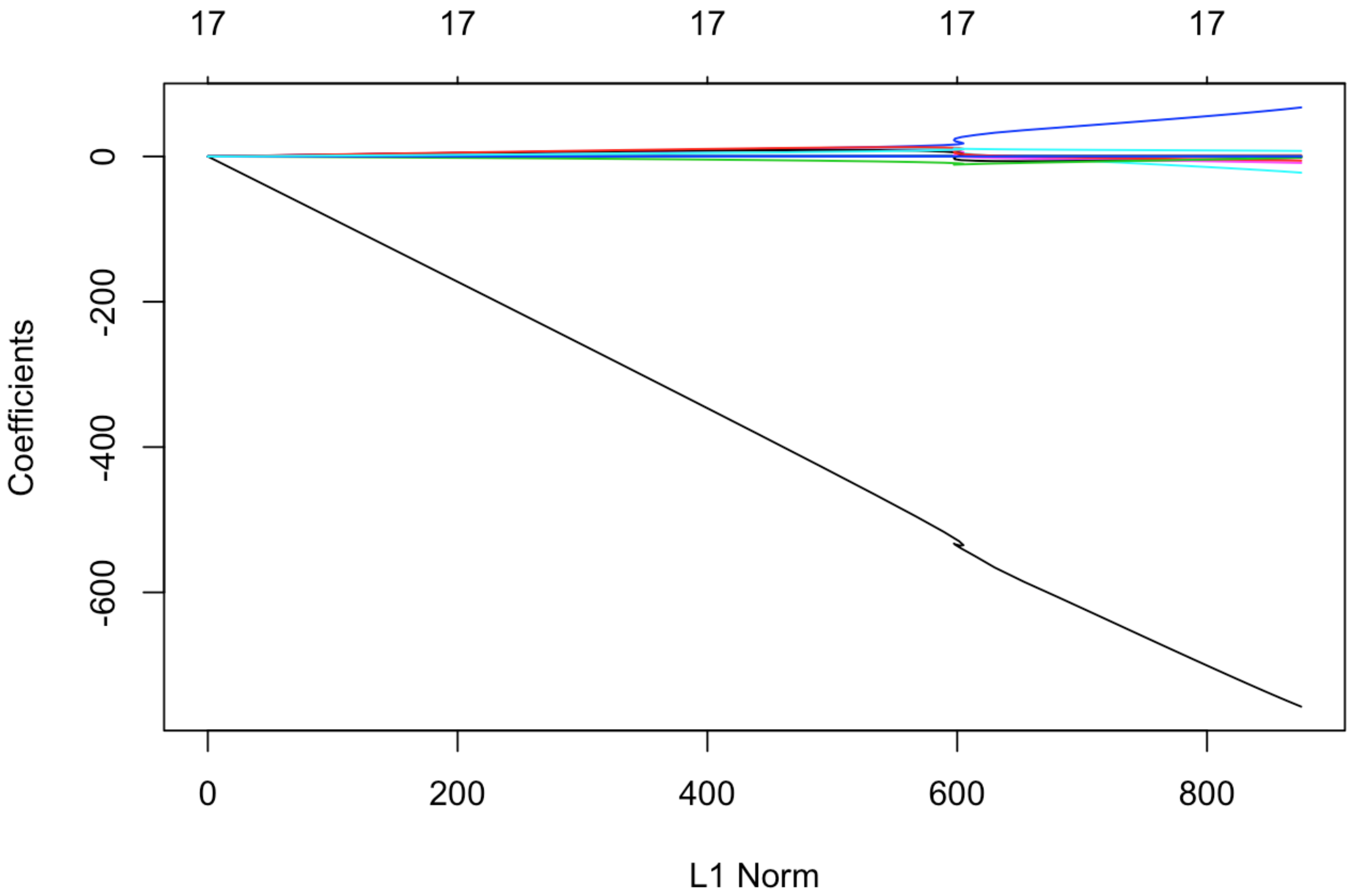
```
lm.mod = glmnet(x[train,], y[train], alpha=0, lambda=0, thresh=1e-12)
lm.pred = predict(lm.mod, s=0, newx=x[test,], exact=T)
mean((lm.pred-y.test)^2) #1075351
```

```
## [1] 1108526
```

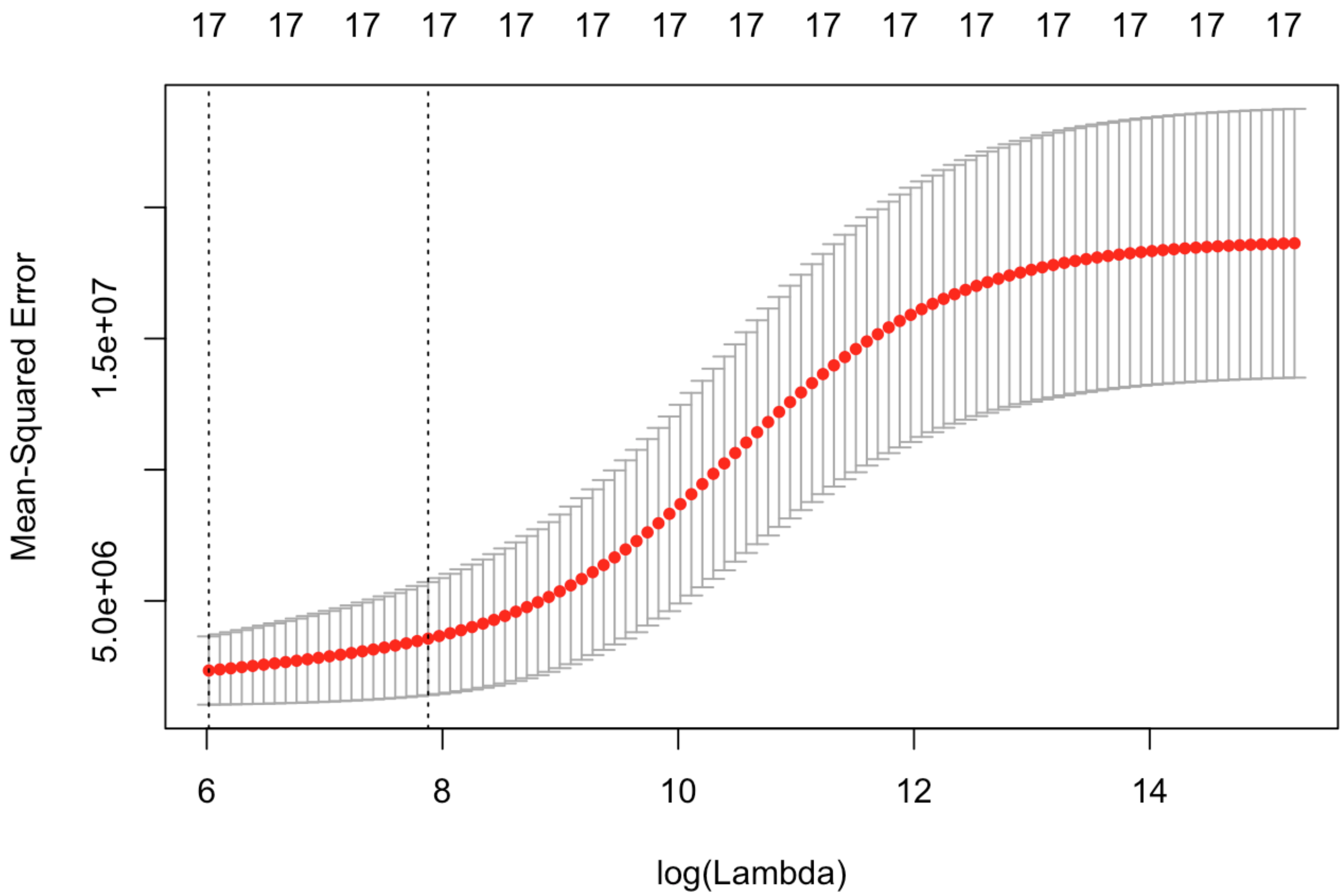
The test error obtained by fitting a linear model using least squares on the training set is 1075351.

Part c

```
grid = 10^seq(10,-2,length=100)
ridge.mod = glmnet(x[train,], y[train], alpha=0, lambda=grid, thresh=1e-12)
plot(ridge.mod)
```




```
set.seed(1)
cv.out = cv.glmnet(x[train,], y[train], alpha=0)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min; bestlam #411
```

```
## [1] 410.7007
```

```
# what is the test MSE assoc w/ lambda = 411?
ridge.pred = predict(ridge.mod, s=bestlam, newx=x[test,])
mean((ridge.pred-y.test)^2) #1043745
```

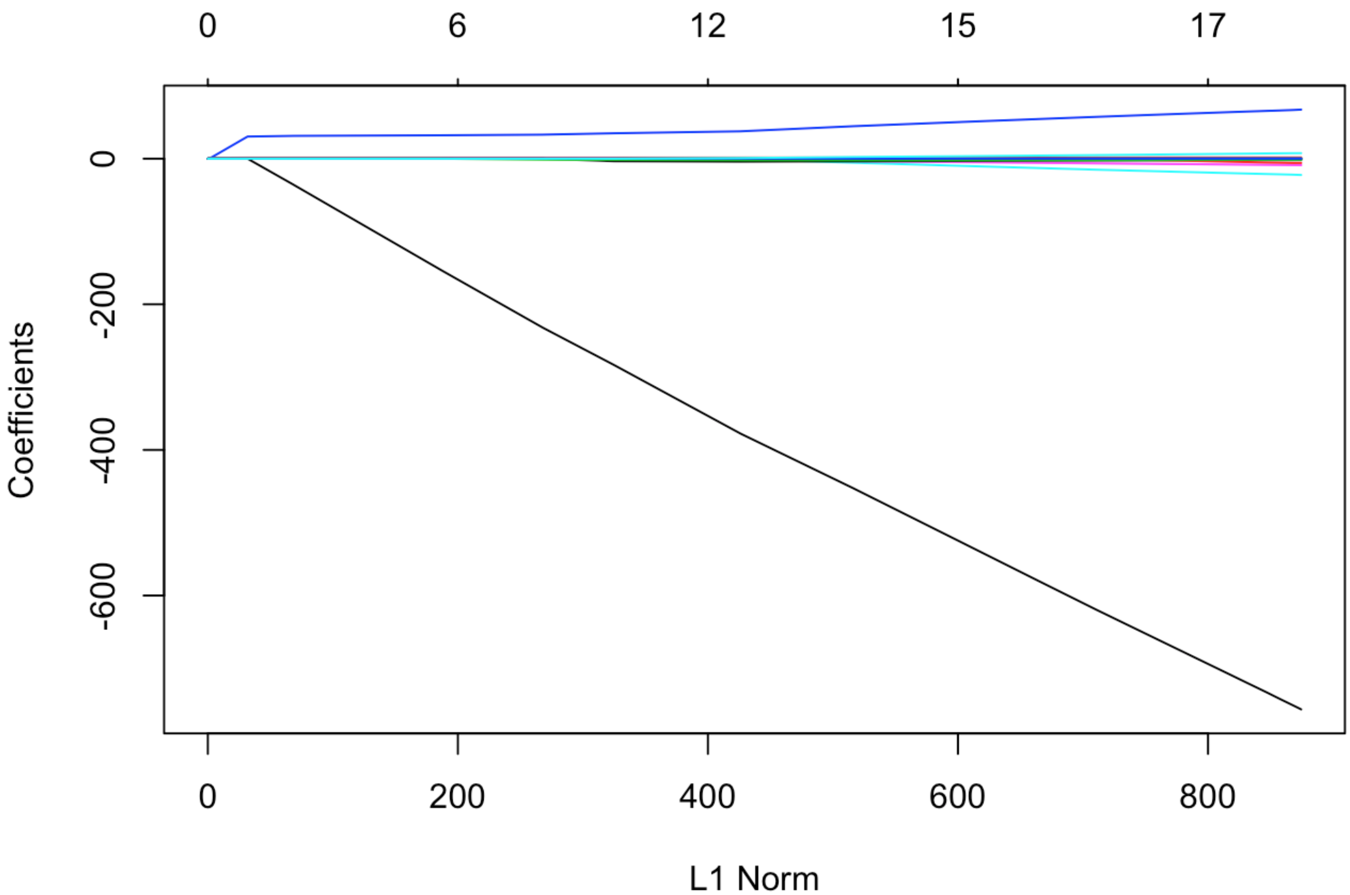
```
## [1] 1029090
```

With a λ of 411 chosen by cross-validation, we fit a ridge regression model on the training set, and obtain test error 1043745, lower than the MSE of our linear model using least squares.

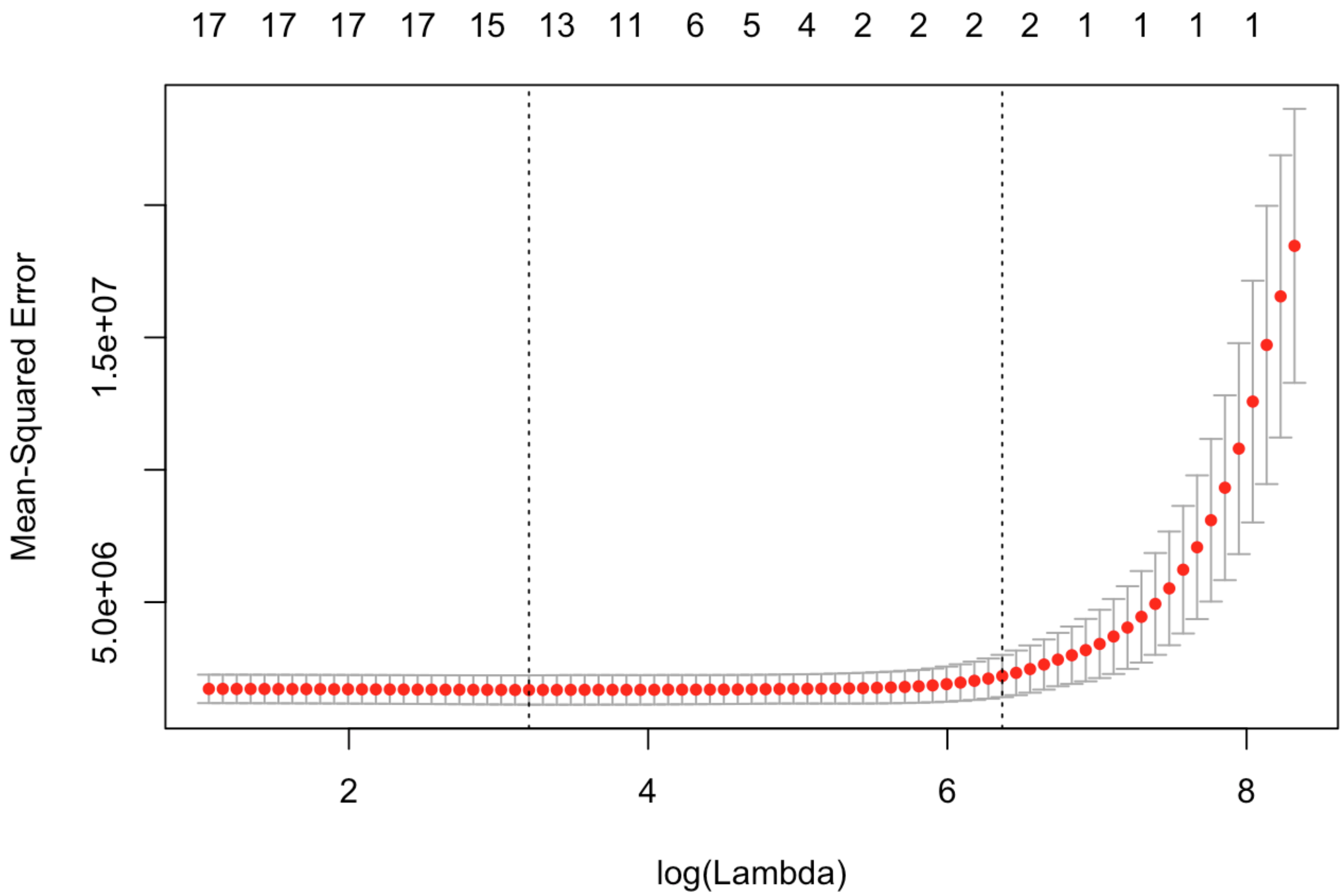
Part d

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

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




```
set.seed(1)
cv.out = cv.glmnet(x[train,], y[train], alpha=1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min; bestlam #27
```

```
## [1] 24.62086
```

```
lasso.pred = predict(lasso.mod, s=bestlam, newx=x[test,])
mean((lasso.pred-y.test)^2) #MSE 104082
```

```
## [1] 1032128
```

```
out = glmnet(x ,y, alpha=1, lambda=grid)
lasso.coef = predict(out, type="coefficients", s=bestlam)[1:17,]
lasso.coef
```

```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -6.321166e+02 -4.088980e+02  1.437087e+00 -1.418240e-01  3.146071e+01
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
## -8.818529e-01  0.000000e+00  1.488050e-02 -5.348474e-02  1.206366e-01
##         Books      Personal           PhD      Terminal     S.F.Ratio
##  0.000000e+00  6.054932e-05 -5.127428e+00 -3.370371e+00  2.739664e+00
##   perc.alumni        Expend
## -1.038499e+00  6.839807e-02
```

```
lasso.coef[lasso.coef != 0]
```

```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -6.321166e+02 -4.088980e+02  1.437087e+00 -1.418240e-01  3.146071e+01
##     Top25perc   P.Undergrad      Outstate    Room.Board      Personal
## -8.818529e-01  1.488050e-02 -5.348474e-02  1.206366e-01  6.054932e-05
##           PhD      Terminal     S.F.Ratio   perc.alumni        Expend
## -5.127428e+00 -3.370371e+00  2.739664e+00 -1.038499e+00  6.839807e-02
```

Using a LASSO model with a λ of 27 chosen by cross-validation, we obtain a slightly lower test error – 104082. However, we now have fewer non-zero coefficient estimates – 14 instead of 17.

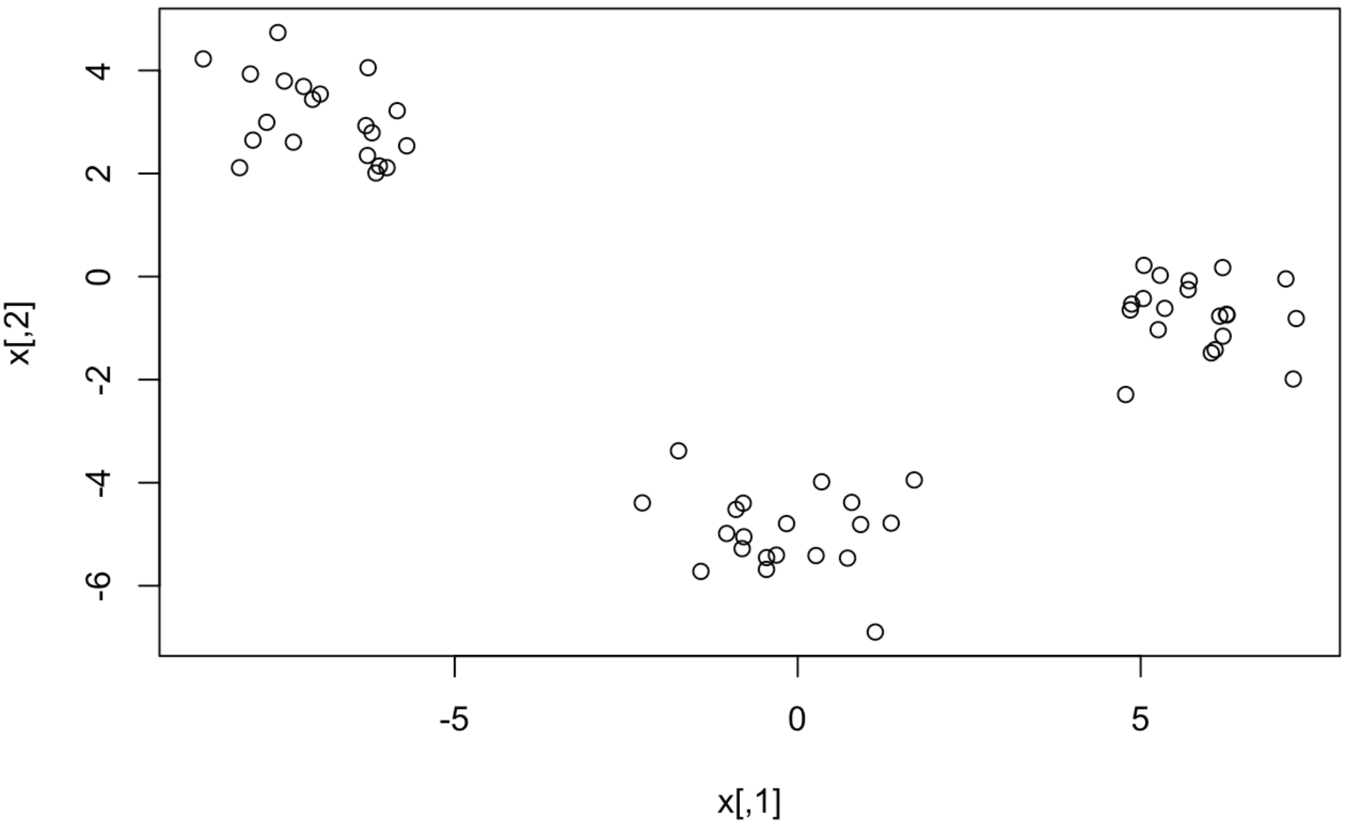
(Clustering and PCA) section 10.7

Question 10

Part a

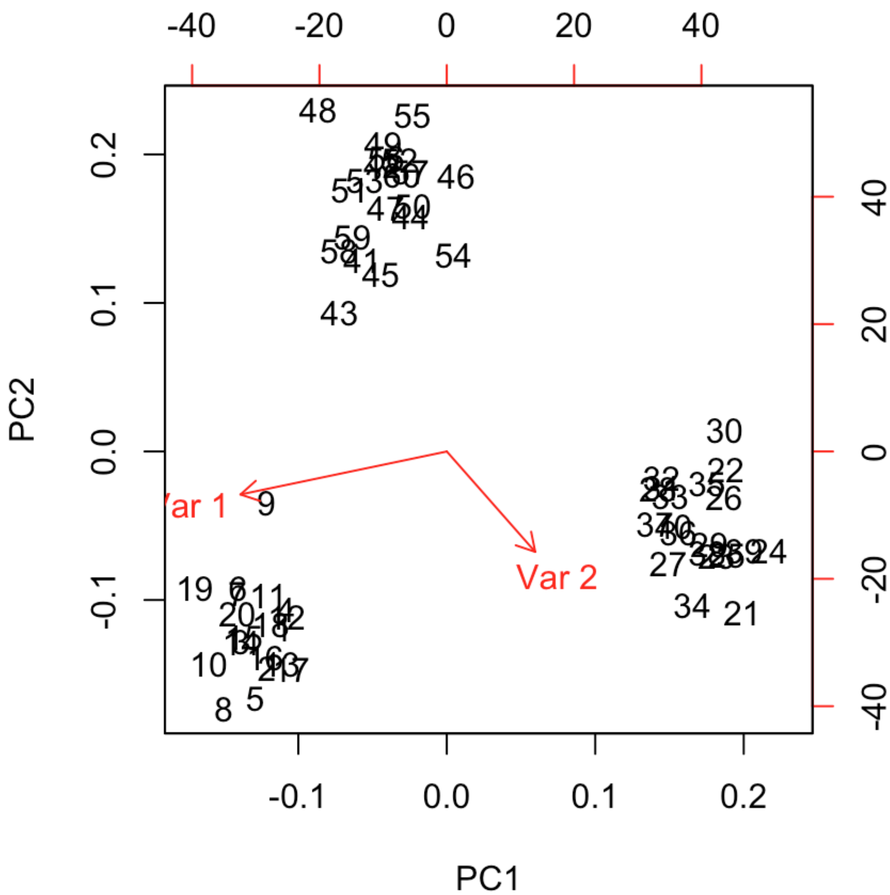
```
set.seed(3)
x = matrix(rnorm(60*2), ncol=2)
x[1:20,1] = x[1:20,1] + 6
x[1:20,2] = x[1:20,2] - 1
x[21:40,1] = x[21:40,1] - 7
x[21:40,2] = x[21:40,2] + 3
x[41:60,2] = x[41:60,2] - 5

plot(x)
```

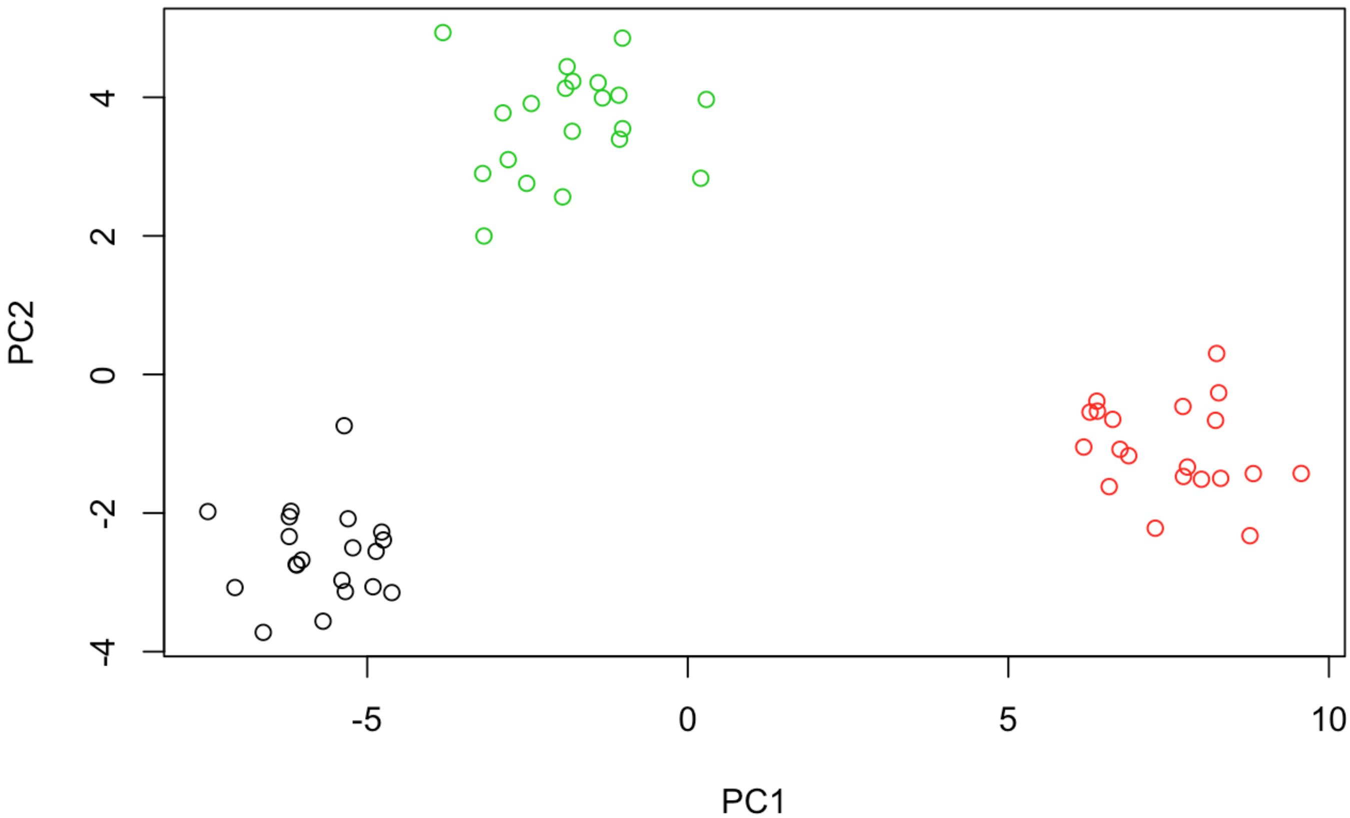


Part b

```
x_clus = c(rep(1,20), rep(2,20), rep(3,20))
pr.out = prcomp(x)
biplot(pr.out)
```




```
plot(predict(pr.out), col=x_clus)
```

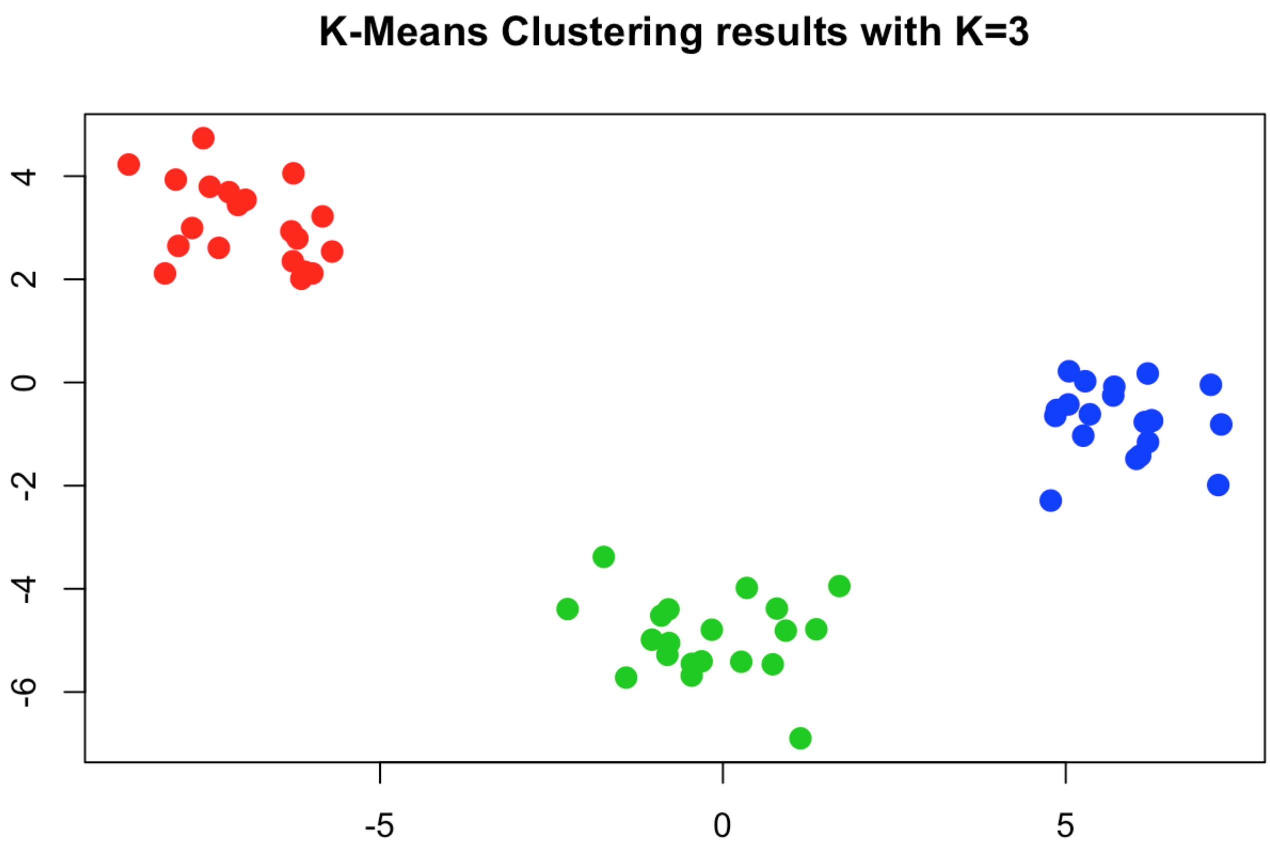


Part c

```
set.seed(4)
km.out = kmeans(x, 3, nstart = 20)
km.out
```

```
## K-means clustering with 3 clusters of sizes 20, 20, 20
##
## Cluster means:
##         [,1]      [,2]
## 1 -6.9464804  3.093462
## 2 -0.1940736 -4.937102
## 3  5.8328096 -0.730540
##
## Clustering vector:
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 27.14631 33.48953 20.51274
##  (between_SS / total_SS =  96.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(x, col=(km.out$cluster + 1), main="K-Means Clustering results with K=3", xlab="", ylab="
", pch=20, cex=2)
```



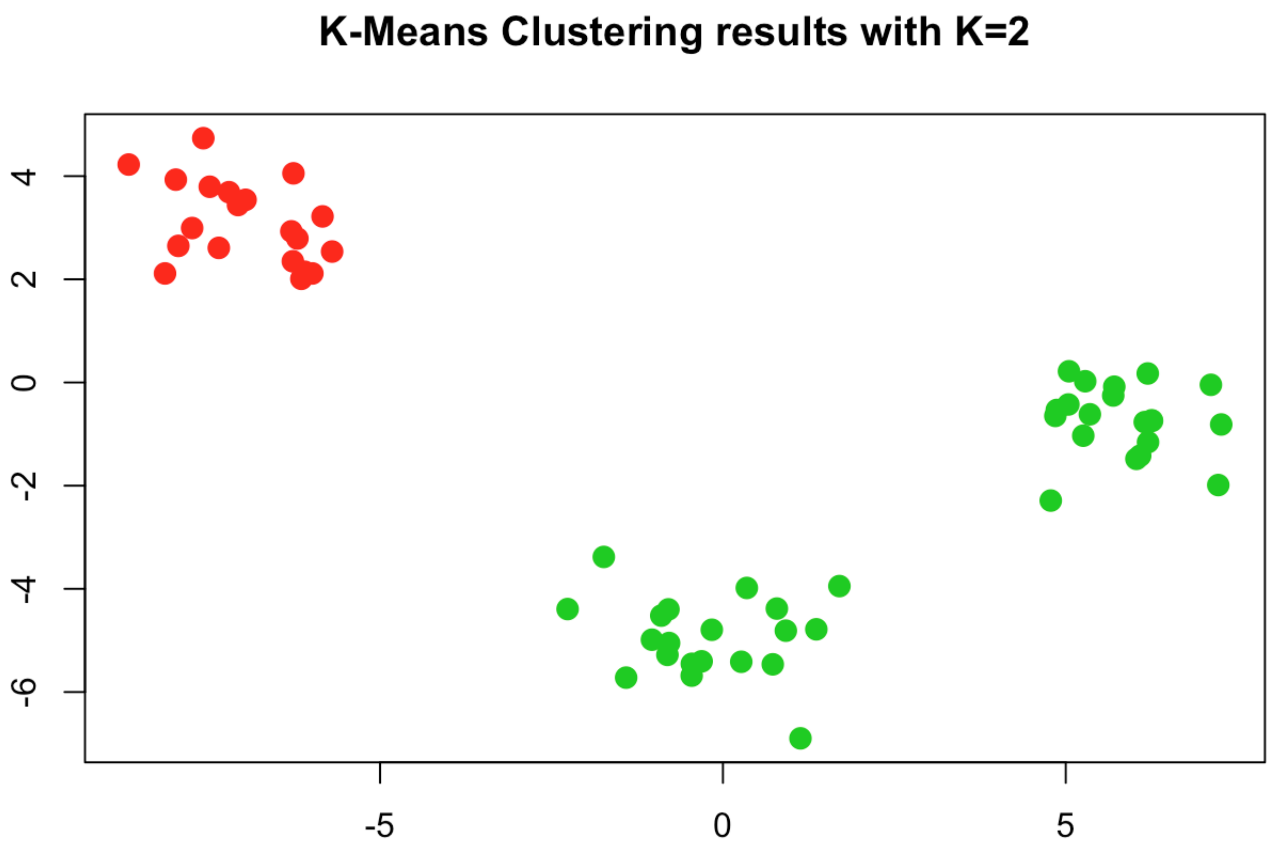
The K-means cluster labels are identical to the true class labels.

Part d

```
set.seed(4)
km.out = kmeans(x, 2, nstart = 20)
km.out
```

```
## K-means clustering with 2 clusters of sizes 20, 40
##
## Cluster means:
##        [,1]      [,2]
## 1 -6.946480  3.093462
## 2  2.819368 -2.833821
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]  27.14631 594.18715
##  (between_SS / total_SS =  73.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(x, col=(km.out$cluster + 1), main="K-Means Clustering results with K=2", xlab="", ylab="
", pch=20, cex=2)
```



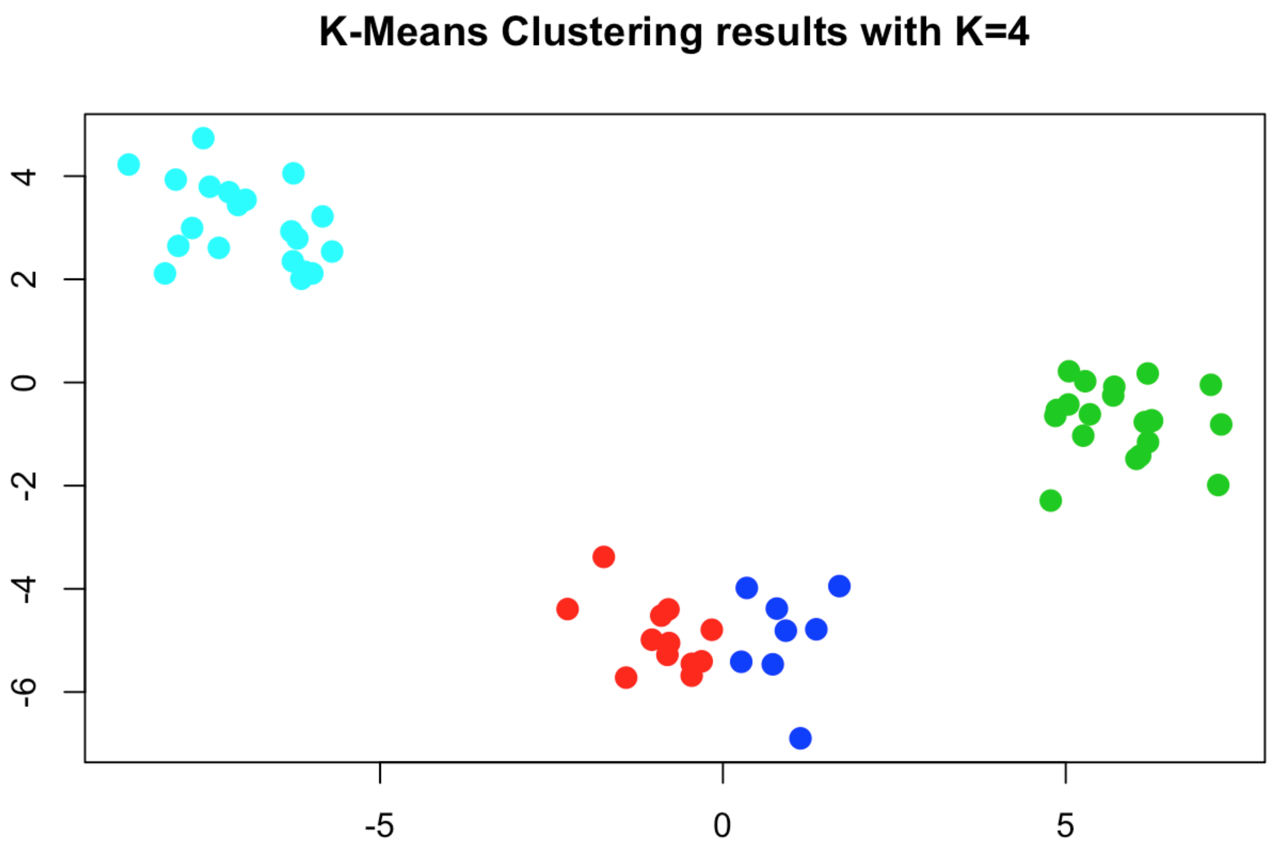
With K = 2, K-means clustering now clusters the two closest clusters together.

Part e

```
set.seed(4)
km.out = kmeans(x, 4, nstart = 20)
km.out
```

```
## K-means clustering with 4 clusters of sizes 12, 20, 8, 20
##
## Cluster means:
##         [,1]      [,2]
## 1 -0.9266408 -4.921679
## 2  5.8328096 -0.730540
## 3  0.9047773 -4.960237
## 4 -6.9464804  3.093462
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [36] 4 4 4 4 4 3 1 3 1 3 1 1 3 1 1 3 1 3 1 1 1 1 3 3 1
##
## Within cluster sum of squares by cluster:
## [1]  9.148013 20.512736  8.234741 27.146309
##  (between_SS / total_SS =  97.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(x, col=(km.out$cluster + 1), main="K-Means Clustering results with K=4", xlab="", ylab="
", pch=20, cex=2)
```



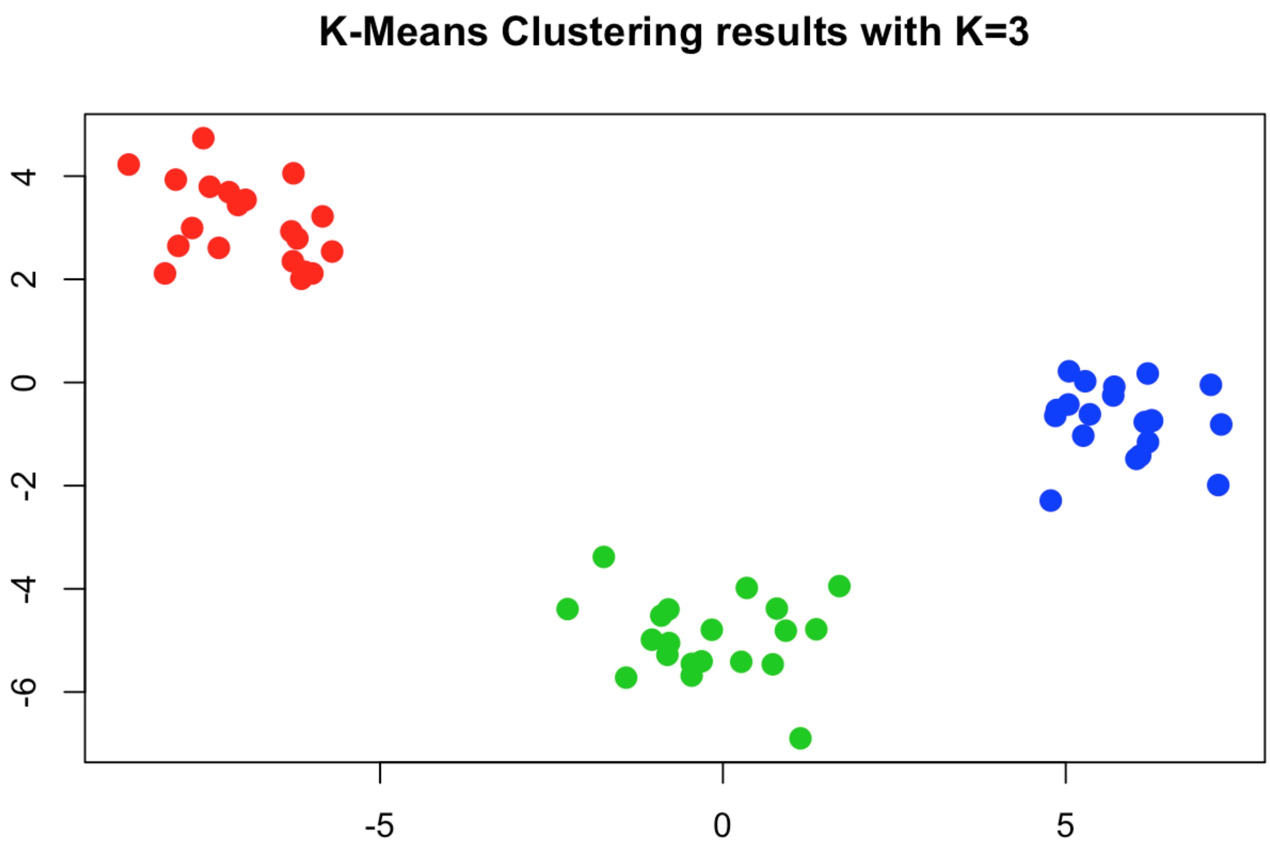
With K = 4, K-means clustering still clusters two of the true clusters correctly, but it splits one cluster into two based on a boundary on the y-axis.

Part f

```
set.seed(4)
km.out = kmeans(pr.out$x, 3, nstart = 20)
km.out
```

```
## K-means clustering with 3 clusters of sizes 20, 20, 20
##
## Cluster means:
##         PC1       PC2
## 1  7.540771 -1.067127
## 2 -1.829226  3.653903
## 3 -5.711546 -2.586776
##
## Clustering vector:
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 27.14631 33.48953 20.51274
##  (between_SS / total_SS =  96.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(x, col=(km.out$cluster + 1), main="K-Means Clustering results with K=3", xlab="", ylab="
", pch=20, cex=2)
```



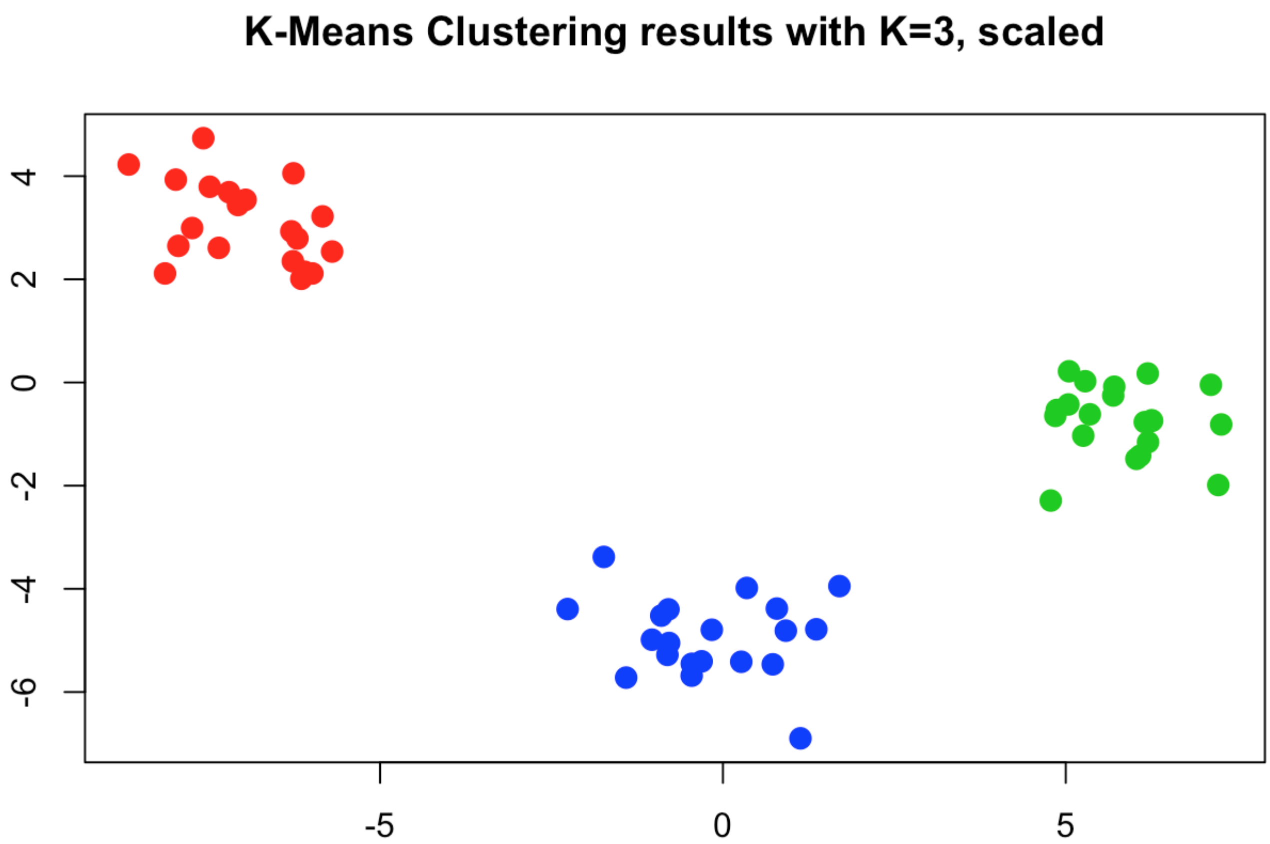
The results for K-means clustering with K = 3 on the first two principal component score vectors is identical to that with the raw data, because the principal component score vectors perfectly labeled the three clusters.

Part g

```
set.seed(4)
km.out = kmeans(scale(x, scale=T), 3, nstart = 20)
km.out
```

```
## K-means clustering with 3 clusters of sizes 20, 20, 20
##
## Cluster means:
##          [,1]        [,2]
## 1 -1.21887255  1.16562755
## 2  1.17359635  0.03761612
## 3  0.04527619 -1.20324367
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##
## Within cluster sum of squares by cluster:
## [1] 1.585322 1.180660 1.775489
##  (between_SS / total_SS =  96.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(x, col=(km.out$cluster + 1), main="K-Means Clustering results with K=3, scaled", xlab=""
, ylab="", pch=20, cex=2)
```



Results appear similar to those obtained in (b). Because our data were drawn from a normal distribution with an SD of 1, it is unsurprising that clustering should not change upon scaling.