ISE 529 - Final

Vaishnavi Guntupalli

2022-04-28

```
set.seed(999)

#Q1

house_votes <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/hous
           header=FALSE)

#(a)
class(house_votes) #check if the loaded data is a data frame
```

```
## [1] "data.frame"
```

```
#since it has all data in one column, split it using separate()
house_votes_full <- separate(house_votes,1,c("class_name",
                                             "handicapped",
                                             "water",
                                             "adoption",
                                             "physician",
                                             "el",
                                             "religious",
                                             "anti",
                                             "aid",
                                             "mx",
                                             "immigration",
                                             "synfuels",
                                             "education",
                                             "superfund",
                                             "crime",
                                             "duty",
                                             "export"),sep=",")
#getting to know the data
head(house_votes_full)
```

```
##   class_name handicapped water adoption physician el religious anti aid mx
## 1 republican           n     y        n         y  y         y    n   n  n
## 2 republican           n     y        n         y  y         y    n   n  n
## 3   democrat           ?     y        y         ?  y         y    n   n  n
## 4   democrat           n     y        y         n  ?         y    n   n  n
## 5   democrat           y     y        y         n  y         y    n   n  n
## 6   democrat           n     y        y         n  y         y    n   n  n
##   immigration synfuels education superfund crime duty export
## 1           y        ?         y         y     y    n      y
## 2           n        n         y         y     y    n      ?
```

```
## 3           n        y         n         y     y    n      n
## 4           n        y         n         y     n    n      y
## 5           n        y         ?         y     y    y      y
## 6           n        n         n         y     y    y      y
```

```
summary(house_votes_full)
```

```
##   class_name        handicapped          water             adoption
##  Length:435         Length:435         Length:435         Length:435
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##   physician              el             religious             anti
##  Length:435         Length:435         Length:435         Length:435
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##      aid                mx             immigration          synfuels
##  Length:435         Length:435         Length:435         Length:435
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##   education          superfund             crime               duty
##  Length:435         Length:435         Length:435         Length:435
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##     export
##  Length:435
##  Class :character
##  Mode  :character
```

```
apply(house_votes_full,2,function(x) class(x))
```

```
##  class_name handicapped       water    adoption   physician          el
## "character" "character" "character" "character" "character" "character"
##   religious        anti         aid          mx immigration    synfuels
## "character" "character" "character" "character" "character" "character"
##   education   superfund       crime        duty      export
## "character" "character" "character" "character" "character"
```

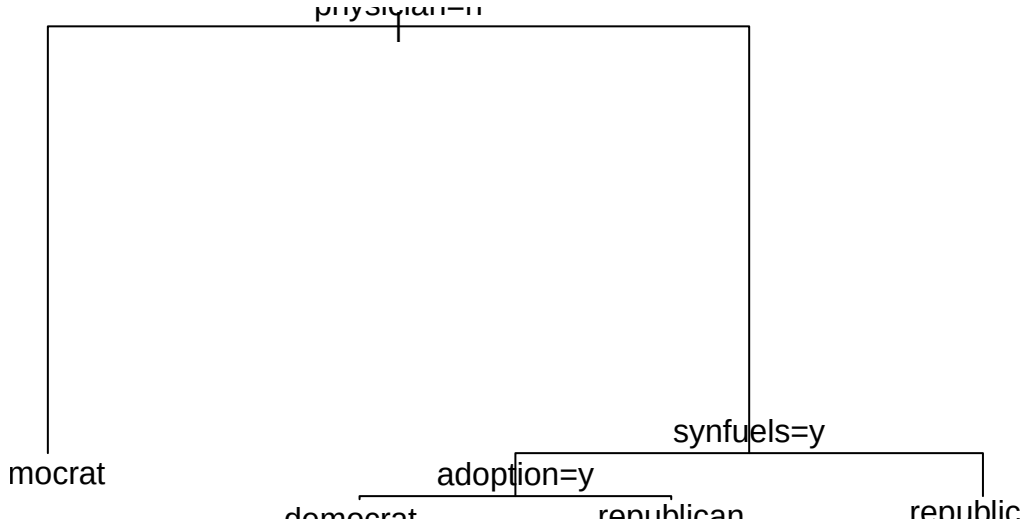
```
#treating the missing values with voting
house_votes_full <- as.data.frame(apply(house_votes_full,2,function(x)
  ifelse(x=="?",names(which(table(x) == max(table(x)[-1]))),x)))

#converting the columns from character to factor
house_votes_factors<-house_votes_full
for(i in 1:ncol(house_votes_full))
house_votes_factors[,i] <- as.factor(house_votes_full[,i])

#divide the set into train and test
#using 0.9 so that we have enough to train on and capture variance
train <- sample(1:nrow(house_votes_factors),0.9*nrow(house_votes_factors),FALSE)
house_votes_train <- house_votes_factors[train,]

#(b)
#built control with cp = 0 to have as many branches as possible
#used 1000 cvs
#used maximum depth possible
rpart_pars <- rpart.control(20,6,0,4,5,2,1000,0,ncol(house_votes_factors))
```

```r
#built tree using the control
rpart_house_votes <- rpart(class_name~.,house_votes_train,method="class",
                           control=rpart_pars)
summary(rpart_house_votes)
```

```
## Call:
## rpart(formula = class_name ~ ., data = house_votes_train, method = "class",
##     control = rpart_pars)
##   n= 391
##
##            CP nsplit rel error    xerror       xstd
## 1 0.889610390      0 1.0000000 1.0000000 0.06273724
## 2 0.003246753      1 0.1103896 0.1103896 0.02618491
## 3 0.000000000      3 0.1038961 0.1298701 0.02828739
##
## Variable importance
## physician  adoption education        el       aid        mx  synfuels
##        24        17        15        15        15        14         1
##
## Node number 1: 391 observations,    complexity param=0.8896104
##   predicted class=democrat    expected loss=0.3938619  P(node) =1
##     class counts:   237   154
##    probabilities: 0.606 0.394
##   left son=2 (230 obs) right son=3 (161 obs)
##   Primary splits:
##       physician splits as  LR, improve=154.69670, (0 missing)
##       adoption  splits as  RL, improve= 96.01968, (0 missing)
##       el        splits as  LR, improve= 86.01369, (0 missing)
##       education splits as  LR, improve= 79.63033, (0 missing)
##       mx        splits as  RL, improve= 70.71148, (0 missing)
##   Surrogate splits:
##       adoption  splits as  RL, agree=0.867, adj=0.677, (0 split)
##       el        splits as  LR, agree=0.841, adj=0.615, (0 split)
##       education splits as  LR, agree=0.841, adj=0.615, (0 split)
##       aid       splits as  RL, agree=0.839, adj=0.609, (0 split)
##       mx        splits as  RL, agree=0.826, adj=0.578, (0 split)
##
## Node number 2: 230 observations
##   predicted class=democrat    expected loss=0.02173913  P(node) =0.5882353
##     class counts:   225     5
##    probabilities: 0.978 0.022
##
## Node number 3: 161 observations,    complexity param=0.003246753
##   predicted class=republican  expected loss=0.07453416  P(node) =0.4117647
##     class counts:    12   149
##    probabilities: 0.075 0.925
##   left son=6 (30 obs) right son=7 (131 obs)
##   Primary splits:
##       synfuels    splits as  RL, improve=3.7485850, (0 missing)
##       adoption    splits as  RL, improve=1.3239620, (0 missing)
##       duty        splits as  RL, improve=1.2212260, (0 missing)
##       education   splits as  LR, improve=0.9554983, (0 missing)
##       immigration splits as  LR, improve=0.8403867, (0 missing)
```

```
##   Surrogate splits:
##       crime splits as  LR, agree=0.832, adj=0.1, (0 split)
##
## Node number 6: 30 observations,    complexity param=0.003246753
##   predicted class=republican  expected loss=0.3  P(node) =0.07672634
##     class counts:     9    21
##    probabilities: 0.300 0.700
##   left son=12 (7 obs) right son=13 (23 obs)
##   Primary splits:
##       adoption    splits as  RL, improve=1.3453420, (0 missing)
##       immigration splits as  LR, improve=0.9800905, (0 missing)
##       education   splits as  LR, improve=0.8727273, (0 missing)
##       water       splits as  RL, improve=0.4509317, (0 missing)
##       export      splits as  LR, improve=0.3000000, (0 missing)
##
## Node number 7: 131 observations
##   predicted class=republican  expected loss=0.02290076  P(node) =0.3350384
##     class counts:     3   128
##    probabilities: 0.023 0.977
##
## Node number 12: 7 observations
##   predicted class=democrat    expected loss=0.4285714  P(node) =0.01790281
##     class counts:     4     3
##    probabilities: 0.571 0.429
##
## Node number 13: 23 observations
##   predicted class=republican  expected loss=0.2173913  P(node) =0.05882353
##     class counts:     5    18
##    probabilities: 0.217 0.783
```

```
plot(rpart_house_votes) #print tree
text(rpart_house_votes,pretty=0)  #add labels
```




```
#predict probabilities using test set
probs_house_votes <- predict(rpart_house_votes,house_votes_factors[-train,-1])

#classification using probabilities
predicted_house_votes <- as.data.frame(house_votes_factors[-train,1])
names(predicted_house_votes) <- c("actual")

for(i in 1:nrow(probs_house_votes))
{
  predicted_house_votes[i,"predicted"] <- ifelse(probs_house_votes[i,1] > 0.5,"democrat","republican" )
}

#classification accuracy
rpart_predicted_table <- table(predicted_house_votes[,1],predicted_house_votes[,2])
(rpart_predicted_table[1,1]+rpart_predicted_table[2,2])/sum(rpart_predicted_table)
```

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

```
#test classification accuracy is 90%, which proves it captured the variance well
(rpart_predicted_table[1,2]+rpart_predicted_table[2,1])/sum(rpart_predicted_table)
```

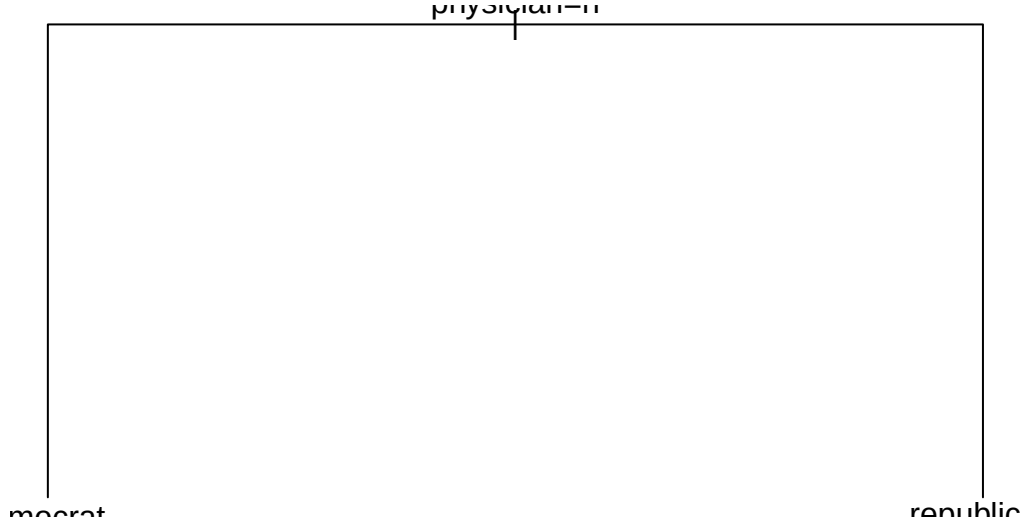
```
## [1] 0
```

```
#misclassification error is small

#(c)
#we prune using cost complexity method
prune_house_votes <- prune.rpart(rpart_house_votes,0.01)
```

```
prune_house_votes #pruned tree
```

```
## n= 391
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 391 154 democrat (0.60613811 0.39386189)
##   2) physician=n 230   5 democrat (0.97826087 0.02173913) *
##   3) physician=y 161  12 republican (0.07453416 0.92546584) *
```

```
plot(prune_house_votes)
text(prune_house_votes,pretty=0)
```




```
#This tree is pruned to have only 1 split because of the cp of 0.01

probs_prune_house_votes <- predict(prune_house_votes,house_votes_factors[-train,-1])

#classification using probabilities
predicted_house_votes <- as.data.frame(house_votes_factors[-train,1])
names(predicted_house_votes) <- c("actual")

for(i in 1:nrow(probs_prune_house_votes))
{
  predicted_house_votes[i,"predicted"] <- ifelse(probs_house_votes[i,1] > 0.5,"democrat","republican" )
}
```

```
#classification accuracy
rpart_predicted_table <- table(predicted_house_votes[,1],predicted_house_votes[,2])
(rpart_predicted_table[1,1]+rpart_predicted_table[2,2])/sum(rpart_predicted_table)
```

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

```
#test classification accuracy is 90%, which proves it captured the variance well
(rpart_predicted_table[1,2]+rpart_predicted_table[2,1])/sum(rpart_predicted_table)
```

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

```
#mis classification error is same as above

#(d)
house_votes_factors_new <- data.frame("n","n","n","y","y","y","y","n","n","y",
                              "n","y","n","y","n","n")
names(house_votes_factors_new) <- names(house_votes_factors)[-1]

new_predict_1 <- predict(rpart_house_votes,house_votes_factors_new)
new_predicted_1 <- ifelse(new_predict_1[,1] > 0.5,"democrat","republican")
new_predicted_1 #predicted using unpruned tree
```

```
## [1] "republican"
```

```
new_predicted_2 <- predict(prune_house_votes,house_votes_factors_new)
new_predicted_2 <- ifelse(new_predict_1[,1] > 0.5,"democrat","republican")
new_predicted_2 #predicted using pruned tree
```

```
## [1] "republican"
```

```
##############################################################################################


## Q2

abalone <- as.vector(read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abal
                         header=FALSE))

#(a)
#splitting teh data into different columns
abalone_full <- separate(abalone,1,c("sex",
                                     "Length",
                                     "Diameter",
                                     "Height",
                                     "Whole_weight",
                                     "Shucked_weight",
                                     "Viscera_weight",
                                     "Shell_weight",
                                     "Rings"),sep=",")
#explore the data
class(abalone_full)
```

```
## [1] "data.frame"
```

```
head(abalone_full)
```

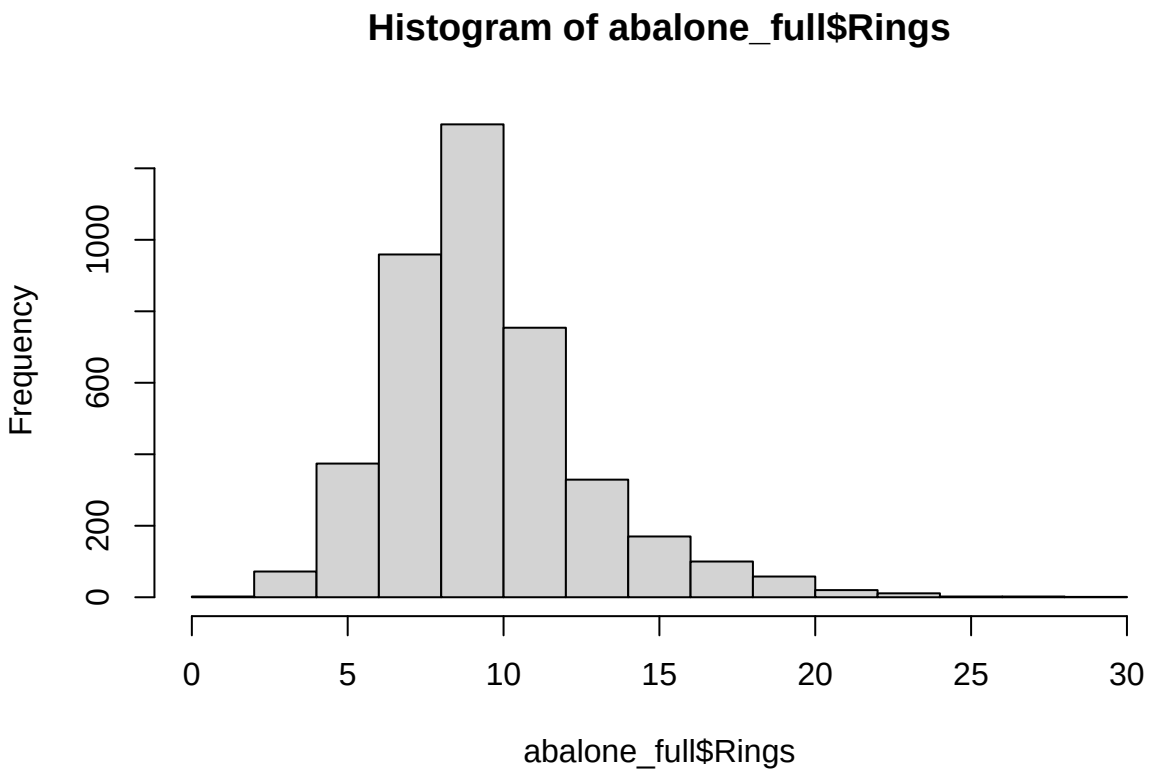
```
##   sex Length Diameter Height Whole_weight Shucked_weight Viscera_weight
## 1   M  0.455    0.365  0.095        0.514         0.2245          0.101
```

```
## 2   M   0.35    0.265   0.09       0.2255         0.0995         0.0485
## 3   F   0.53     0.42  0.135        0.677         0.2565         0.1415
## 4   M   0.44    0.365  0.125        0.516         0.2155          0.114
## 5   I   0.33    0.255   0.08        0.205         0.0895         0.0395
## 6   I  0.425      0.3  0.095       0.3515          0.141         0.0775
##   Shell_weight Rings
## 1         0.15    15
## 2         0.07     7
## 3         0.21     9
## 4        0.155    10
## 5        0.055     7
## 6         0.12     8
```

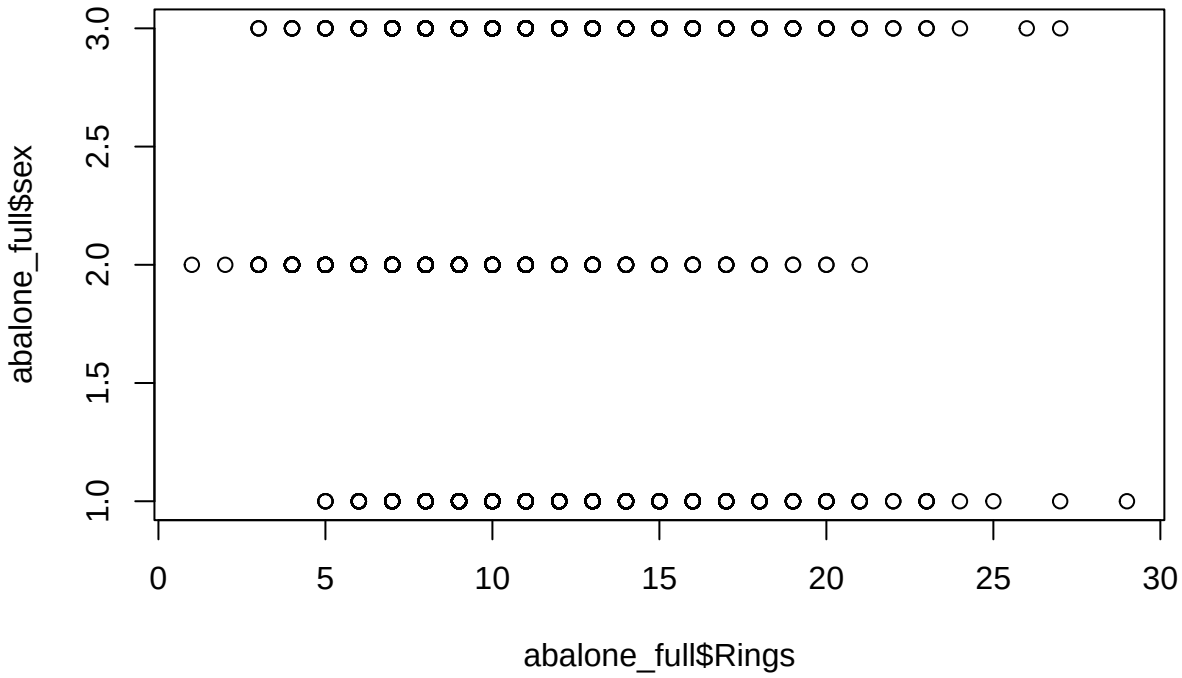
```
summary(abalone_full)
```

```
##      sex               Length           Diameter           Height
##  Length:4177        Length:4177        Length:4177        Length:4177
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  Whole_weight       Shucked_weight     Viscera_weight     Shell_weight
##  Length:4177        Length:4177        Length:4177        Length:4177
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##     Rings
##  Length:4177
##  Class :character
##  Mode  :character
```

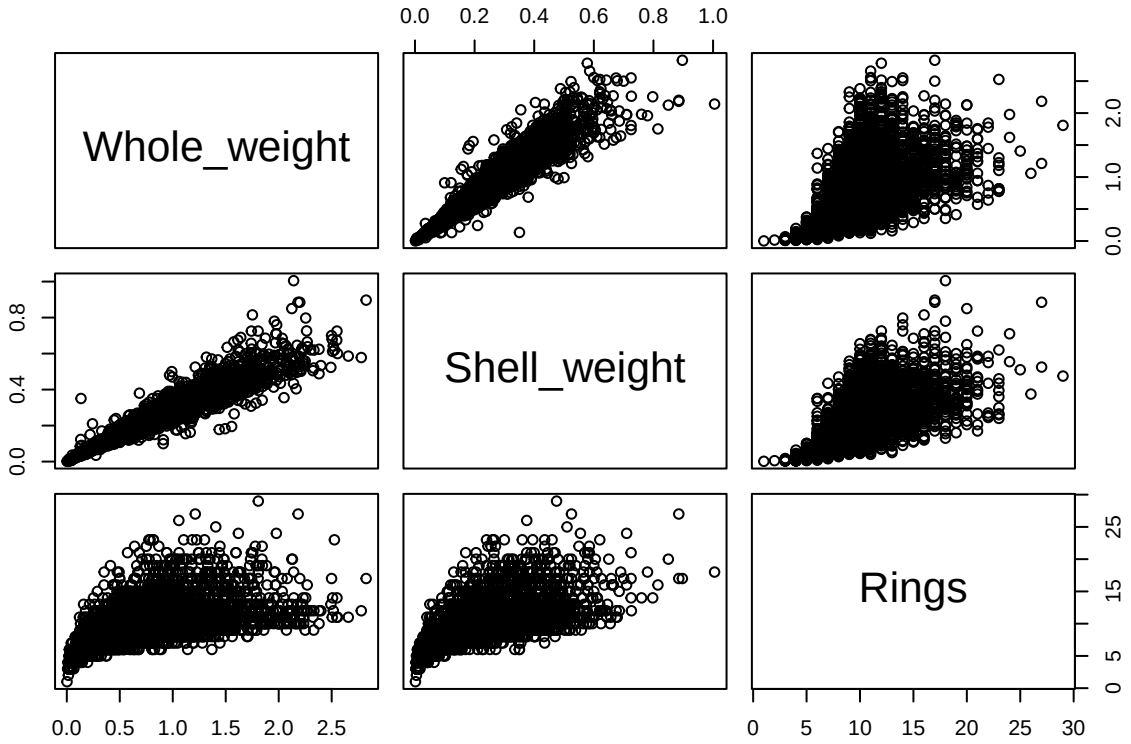
```
#converting to factor
abalone_full[,1] <- as.factor(abalone_full[,1])
#converting to numerical
abalone_full[,-1]<- as.data.frame(apply(abalone_full[,-1],2,function(x) as.double(x)))

#train test split
split_sample <- sample(1:4177,3133,replace=FALSE) #random split
abalone_train <- abalone_full[split_sample,]
abalone_test_x <- abalone_full[-split_sample,-9]

#(b)
#explore the spread of Rings
hist(abalone_full$Rings)
```



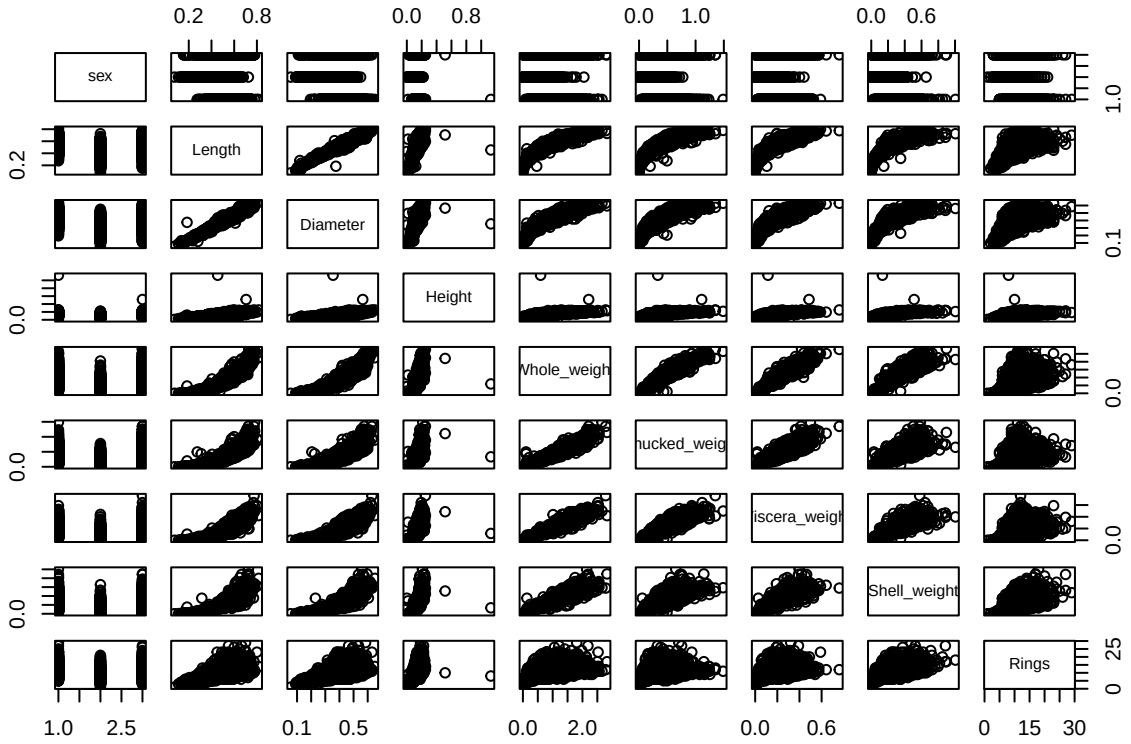

```
#explore correlation and covariance between the variables
data_correlation <- cor(abalone_full[,-1])
data_covariance <- cov(abalone_full[,-1])
plot(abalone_full$Rings,abalone_full$sex)
```




```
#look at pairwise scatter plots of Y with strongly correlated ones
pairs(abalone_full[,c(5,8,9)])
```




```
#pairwise plot of all variables
pairs(abalone_full)
```




```
#(c)
abalone_lm_fit <-lm(Rings~sex+Length+Diameter+Height+Whole_weight, data=abalone_train)
summary(abalone_lm_fit)
```

```
##
## Call:
## lm(formula = Rings ~ sex + Length + Diameter + Height + Whole_weight,
##     data = abalone_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.1936  -1.5801  -0.5278   0.8742  15.7575
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.82196    0.37885  10.088  < 2e-16 ***
## sexI          -1.04792    0.13391  -7.825 6.87e-15 ***
## sexM          -0.07249    0.11054  -0.656    0.512
## Length       -11.86677    2.33690  -5.078 4.04e-07 ***
## Diameter      26.59260    2.86391   9.285  < 2e-16 ***
## Height        15.54173    1.86107   8.351  < 2e-16 ***
## Whole_weight  -0.38981    0.25489  -1.529    0.126
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.516 on 3126 degrees of freedom
```

```
## Multiple R-squared:  0.3813, Adjusted R-squared:  0.3801
## F-statistic: 321.1 on 6 and 3126 DF,  p-value: < 2.2e-16
```

```
abalone_lm_predict <- predict(abalone_lm_fit,abalone_train[,-9])

#explain ability of our model
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_lm_predict-abalone_train[,9])^2)
(TSS-RSS)/TSS
```

```
## [1] 0.3813178
```

```
abalone_lm_fit <-lm(Rings~sex+Length+Diameter+Height, data=abalone_train)
summary(abalone_lm_fit)
```

```
##
## Call:
## lm(formula = Rings ~ sex + Length + Diameter + Height, data = abalone_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.6156  -1.5777  -0.5312   0.8830  15.6510
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.22433    0.27265  15.494  < 2e-16 ***
## sexI         -1.03173    0.13352  -7.727 1.47e-14 ***
## sexM         -0.07741    0.11051  -0.700    0.484
## Length      -12.57176    2.29147  -5.486 4.43e-08 ***
## Diameter     25.90726    2.82924   9.157  < 2e-16 ***
## Height       14.96846    1.82332   8.209 3.21e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.517 on 3127 degrees of freedom
## Multiple R-squared:  0.3809, Adjusted R-squared:  0.3799
## F-statistic: 384.7 on 5 and 3127 DF,  p-value: < 2.2e-16
```

```
abalone_lm_predict <- predict(abalone_lm_fit,abalone_train[,-9])

#explain ability of our model
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_lm_predict-abalone_train[,9])^2)
(TSS-RSS)/TSS
```

```
## [1] 0.380855
```

```
#(d)
#fit using all 8 predictors
abalone_lm_fit_all <- lm(Rings~.,data=abalone_train)
summary(abalone_lm_fit_all)
```

```
##
## Call:
## lm(formula = Rings ~ ., data = abalone_train)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -8.9016 -1.2944 -0.3132  0.8700 14.0069
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.70263    0.32923  11.246  < 2e-16 ***
## sexI            -0.76690    0.11611  -6.605 4.66e-11 ***
## sexM             0.09727    0.09533   1.020    0.308
## Length          -1.62978    2.03771  -0.800    0.424
## Diameter        13.66453    2.50407   5.457 5.22e-08 ***
## Height           9.24002    1.61514   5.721 1.16e-08 ***
## Whole_weight     8.37343    0.81403  10.286  < 2e-16 ***
## Shucked_weight -19.45850    0.91591 -21.245  < 2e-16 ***
## Viscera_weight  -9.22558    1.49035  -6.190 6.79e-10 ***
## Shell_weight     9.02562    1.25172   7.211 6.96e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.163 on 3123 degrees of freedom
## Multiple R-squared:  0.5435, Adjusted R-squared:  0.5422
## F-statistic: 413.1 on 9 and 3123 DF,  p-value: < 2.2e-16
```

```
abalone_lm_predict_all <- predict(abalone_lm_fit_all,abalone_train[,-9])

TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_lm_predict_all-abalone_train[,9])^2)
RSS
```

```
## [1] 14606.31
```

```
(TSS-RSS)/TSS
```

```
## [1] 0.5435025
```

```
#Low explainability but higher than previous.

abalone_lm_fit_all <- lm(Rings~sex+Diameter+Height+Whole_weight+Shucked_weight
                         +Viscera_weight+Shell_weight,data=abalone_train)
summary(abalone_lm_fit_all)
```

```
##
## Call:
## lm(formula = Rings ~ sex + Diameter + Height + Whole_weight +
##     Shucked_weight + Viscera_weight + Shell_weight, data = abalone_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8392 -1.3013 -0.3165  0.8686 14.0563
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.61728    0.31143  11.615  < 2e-16 ***
## sexI            -0.77321    0.11584  -6.675 2.91e-11 ***
## sexM             0.09582    0.09531   1.005    0.315
## Diameter        11.87001    1.11178  10.677  < 2e-16 ***
## Height           9.18846    1.61376   5.694 1.36e-08 ***
```

```
## Whole_weight     8.38671    0.81382  10.305  < 2e-16 ***
## Shucked_weight -19.52246    0.91236 -21.398  < 2e-16 ***
## Viscera_weight  -9.34421    1.48287  -6.301 3.36e-10 ***
## Shell_weight     9.05445    1.25113   7.237 5.75e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.163 on 3124 degrees of freedom
## Multiple R-squared:  0.5434, Adjusted R-squared:  0.5422
## F-statistic: 464.8 on 8 and 3124 DF,  p-value: < 2.2e-16
```

```
abalone_lm_predict_all <- predict(abalone_lm_fit_all,abalone_train[,-9])

TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_lm_predict_all-abalone_train[,9])^2)
(TSS-RSS)/TSS
```

```
## [1] 0.543409
```

```
#(e)
#5 predictors###############
abalone_quad_5 <- lm(Rings~poly(Diameter,2)+poly(Length,2)+poly(Height,2)
                     +poly(Whole_weight,2)+I(sex=="I"), data=abalone_train)
summary(abalone_quad_5)
```

```
##
## Call:
## lm(formula = Rings ~ poly(Diameter, 2) + poly(Length, 2) + poly(Height,
##     2) + poly(Whole_weight, 2) + I(sex == "I"), data = abalone_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6061 -1.5662 -0.4498  0.9103 16.1922
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                10.18314    0.05869 173.514  < 2e-16 ***
## poly(Diameter, 2)1         92.30684   17.49967   5.275 1.42e-07 ***
## poly(Diameter, 2)2         17.63214    9.98008   1.767  0.07737 .
## poly(Length, 2)1         -101.70400   17.61809  -5.773 8.57e-09 ***
## poly(Length, 2)2          -38.82058    9.87864  -3.930 8.69e-05 ***
## poly(Height, 2)1           67.91458    5.99414  11.330  < 2e-16 ***
## poly(Height, 2)2          -34.72478    3.64390  -9.530  < 2e-16 ***
## poly(Whole_weight, 2)1     32.34237   12.53270   2.581  0.00991 **
## poly(Whole_weight, 2)2     -3.16795    4.15629  -0.762  0.44599
## I(sex == "I")TRUE          -0.84408    0.11842  -7.128 1.26e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.451 on 3123 degrees of freedom
## Multiple R-squared:  0.4138, Adjusted R-squared:  0.4122
## F-statistic:   245 on 9 and 3123 DF,  p-value: < 2.2e-16
```

```
abalone_quad_5_predict <- predict(abalone_quad_5,abalone_train)

TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
```

```
RSS = sum((abalone_quad_5_predict-abalone_train[,9])^2)
RSS
```

```
## [1] 18754.85
```

```
(TSS-RSS)/TSS
```

```
## [1] 0.4138461
```

```
#8 predictors###############
abalone_quad_8 <- lm(Rings~poly(Length,2)+
                     poly(Diameter,2)+
                       poly(Height)+
                       poly(Whole_weight,2)+
                       poly(Shucked_weight,2)+
                       poly(Viscera_weight,2)+
                       poly(Shell_weight,2)+sex, data=abalone_train)
summary(abalone_quad_8)
```

```
##
## Call:
## lm(formula = Rings ~ poly(Length, 2) + poly(Diameter, 2) + poly(Height) +
##     poly(Whole_weight, 2) + poly(Shucked_weight, 2) + poly(Viscera_weight,
##     2) + poly(Shell_weight, 2) + sex, data = abalone_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5481 -1.2778 -0.3037  0.8685 15.5356
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  10.07727    0.07165 140.651  < 2e-16 ***
## poly(Length, 2)1            -40.15498   15.21011  -2.640  0.00833 **
## poly(Length, 2)2            -25.99195    8.48267  -3.064  0.00220 **
## poly(Diameter, 2)1           44.67550   15.15727   2.947  0.00323 **
## poly(Diameter, 2)2           -0.89353    8.61108  -0.104  0.91736
## poly(Height)                 12.02139    3.84182   3.129  0.00177 **
## poly(Whole_weight, 2)1      342.61716   25.35186  13.514  < 2e-16 ***
## poly(Whole_weight, 2)2      -77.80552   11.85459  -6.563 6.14e-11 ***
## poly(Shucked_weight, 2)1 -284.50577   12.53847 -22.691  < 2e-16 ***
## poly(Shucked_weight, 2)2   65.51676    6.96410   9.408  < 2e-16 ***
## poly(Viscera_weight, 2)1  -66.62445   10.57904  -6.298 3.44e-10 ***
## poly(Viscera_weight, 2)2   16.93098    5.80074   2.919  0.00354 **
## poly(Shell_weight, 2)1     88.88790   11.44756   7.765 1.10e-14 ***
## poly(Shell_weight, 2)2     -3.27467    5.18177  -0.632  0.52746
## sexI                       -0.57392    0.11579  -4.956 7.57e-07 ***
## sexM                        0.04653    0.09230   0.504  0.61421
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.091 on 3117 degrees of freedom
## Multiple R-squared:  0.5739, Adjusted R-squared:  0.5719
## F-statistic: 279.9 on 15 and 3117 DF,  p-value: < 2.2e-16
```

```
abalone_quad_8_predict <- predict(abalone_quad_8,abalone_train)
```

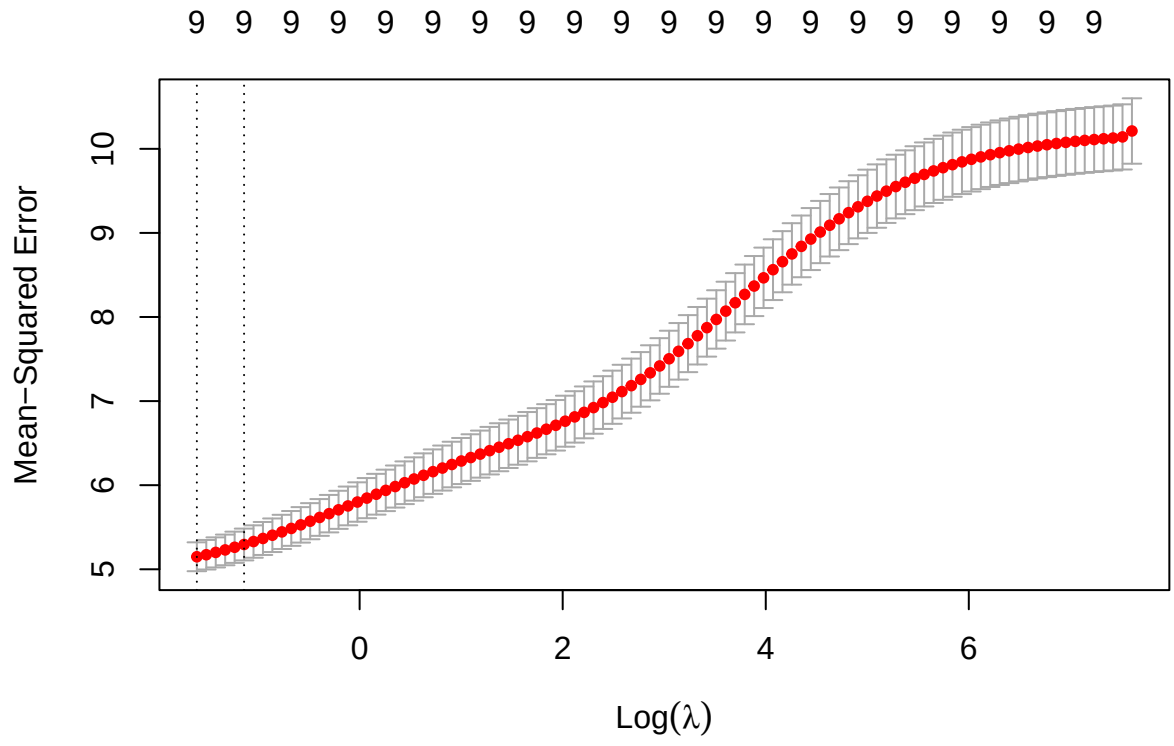
```
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_quad_8_predict-abalone_train[,9])^2)
RSS
```

```
## [1] 13632.52
```

```
(TSS-RSS)/TSS
```

```
## [1] 0.5739368
```

```
#(f)
x=model.matrix(Rings~.,abalone_train) #convert 'sex' to dummy vars
test_x <- model.matrix(Rings~.,abalone_full[-split_sample,])
#ridge model with default parameters
ridge_glmnet <- glmnet(x,abalone_train[,9],alpha=0)
#cv to fit best lambda
cv_ridge <- cv.glmnet(x,abalone_train[,9],alpha=0,nfolds=4,standardize=TRUE)
plot(cv_ridge)
```




```
summary(cv_ridge)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
```

```
## call         6    -none- call
## name         1    -none- character
## glmnet.fit  12    elnet  list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```
cv_ridge$lambda.min
```

```
## [1] 0.2012065
```

```
#(g)
forward <- function(data)
{
  step = 1
  model = "Rings ~ "
  predictor = c()
  variable = c(colnames(data)[-9],"poly(Length,2)","poly(Diameter,2)",
               "poly(Height,2)","poly(Whole_weight,2)",
               "poly(Shucked_weight,2)","poly(Viscera_weight,2)",
               "poly(Shell_weight,2)")
  while(TRUE)
  {
    p_value = c()
    for(i in variable)
    {
      curmodel = paste0("Rings ~ ", paste(c(predictor,i),collapse = " + "))
      fit = lm(curmodel, data = data)
      p = summary(fit)$coefficients[,4][step+1]
      p_value = c(p_value, p)
    }
    newPred = which.min(p_value)
    if(p_value[newPred] <= 0.05)
    {
      predictor = c(predictor, variable[newPred])
      model = paste0("Rings ~ ", paste(predictor,collapse = " + "))
      variable = variable[-newPred]
      # print step wise result
      print(sprintf("=========Step %i: =========", step))
      print("p_value of each element:")
      print(p_value)
      print(sprintf("Model in step %i: ", step))
      print(model)
      step = step + 1
    }else break
  }
  # Show final results
  print(sprintf("===== The final model is: %s =====", model))
  final.fit = lm(model, data = data)
  summary(final.fit)
}

forward(abalone_train)
```

```
## [1] "=========Step 1: ========="
```

```
## [1] "p_value of each element:"
##                       sexI                     Length                   Diameter
##             1.571770e-126             2.128961e-265             3.548247e-289
##                     Height              Whole_weight            Shucked_weight
##             3.018258e-247             3.209224e-241             1.123144e-138
##            Viscera_weight              Shell_weight          poly(Length, 2)1
##             3.761234e-208              0.000000e+00             6.358523e-268
##       poly(Diameter, 2)1          poly(Height, 2)1   poly(Whole_weight, 2)1
##             7.979909e-291             1.079919e-272             3.156546e-255
## poly(Shucked_weight, 2)1 poly(Viscera_weight, 2)1   poly(Shell_weight, 2)1
##             1.464188e-149             3.238842e-223              0.000000e+00
## [1] "Model in step 1: "
## [1] "Rings ~ Shell_weight"
## [1] "=========Step 2: ========="
## [1] "p_value of each element:"
##                       sexI                     Length                   Diameter
##              4.760512e-15              9.985254e-01              8.473396e-03
##                     Height              Whole_weight            Shucked_weight
##              1.762808e-08              9.172727e-46              5.904285e-98
##            Viscera_weight          poly(Length, 2)1       poly(Diameter, 2)1
##              1.738569e-27              2.500767e-26              4.475506e-13
##         poly(Height, 2)1   poly(Whole_weight, 2)1 poly(Shucked_weight, 2)1
##              3.961720e-19              1.392159e-37              5.191709e-77
## poly(Viscera_weight, 2)1   poly(Shell_weight, 2)2
##              1.418236e-16              7.523963e-22
## [1] "Model in step 2: "
## [1] "Rings ~ Shell_weight + Shucked_weight"
## [1] "=========Step 3: ========="
## [1] "p_value of each element:"
##                       sexI                     Length                   Diameter
##              4.371487e-22              5.272999e-37              3.880639e-48
##                     Height              Whole_weight            Viscera_weight
##              2.362513e-25              1.264484e-28              8.078071e-05
##         poly(Length, 2)1       poly(Diameter, 2)1         poly(Height, 2)1
##              2.673870e-03              2.181483e-10              2.277380e-54
##   poly(Whole_weight, 2)1 poly(Shucked_weight, 2)2 poly(Viscera_weight, 2)1
##              3.710618e-34              3.693220e-16              3.003265e-10
##   poly(Shell_weight, 2)2
##              7.176464e-59
## [1] "Model in step 3: "
## [1] "Rings ~ Shell_weight + Shucked_weight + poly(Shell_weight,2)"
## [1] "=========Step 4: ========="
## [1] "p_value of each element:"
##                       sexI                     Length                   Diameter
##              2.812813e-10              2.971982e-02              6.233029e-06
##                     Height              Whole_weight            Viscera_weight
##              4.806521e-09              3.697746e-23              1.162045e-01
##         poly(Length, 2)1       poly(Diameter, 2)1         poly(Height, 2)1
##              2.418025e-01              2.623832e-02              1.348526e-19
##   poly(Whole_weight, 2)1 poly(Shucked_weight, 2)2 poly(Viscera_weight, 2)1
##              1.110680e-21              3.289760e-06              1.540991e-01
## [1] "Model in step 4: "
## [1] "Rings ~ Shell_weight + Shucked_weight + poly(Shell_weight,2) + Whole_weight"
## [1] "=========Step 5: ========="
```

```
## [1] "p_value of each element:"
##                      sexI                    Length                  Diameter
##             1.267821e-07             4.040578e-01             9.783822e-04
##                    Height           Viscera_weight         poly(Length, 2)1
##             3.529178e-07             1.033115e-09             6.063750e-05
##       poly(Diameter, 2)1         poly(Height, 2)1   poly(Whole_weight, 2)2
##             9.013851e-01             2.678168e-15             8.550275e-01
## poly(Shucked_weight, 2)2 poly(Viscera_weight, 2)1
##             1.019791e-05             9.331948e-09
## [1] "Model in step 5: "
## [1] "Rings ~ Shell_weight + Shucked_weight + poly(Shell_weight,2) + Whole_weight + poly(Height,2)"
## [1] "=========Step 6: ========="
## [1] "p_value of each element:"
## poly(Height, 2)2 poly(Height, 2)2 poly(Height, 2)2 poly(Height, 2)2
##     9.309546e-10     4.061249e-10     8.980693e-09     6.005606e-10
## poly(Height, 2)2 poly(Height, 2)2 poly(Height, 2)2 poly(Height, 2)2
##     7.528023e-11     3.565644e-05     2.211961e-05     3.567327e-10
## poly(Height, 2)2 poly(Height, 2)2
##     3.938270e-11     3.297091e-11
## [1] "Model in step 6: "
## [1] "Rings ~ Shell_weight + Shucked_weight + poly(Shell_weight,2) + Whole_weight + poly(Height,2) + p
## [1] "=========Step 7: ========="
## [1] "p_value of each element:"
## poly(Viscera_weight, 2)1 poly(Viscera_weight, 2)1 poly(Viscera_weight, 2)1
##             4.754111e-13             5.931349e-11             6.418190e-12
## poly(Viscera_weight, 2)1 poly(Viscera_weight, 2)1 poly(Viscera_weight, 2)1
##             2.561311e-11             2.561311e-11             5.166858e-09
## poly(Viscera_weight, 2)1 poly(Viscera_weight, 2)1 poly(Viscera_weight, 2)1
##             1.810062e-11             2.320011e-09             1.819938e-06
## [1] "Model in step 7: "
## [1] "Rings ~ Shell_weight + Shucked_weight + poly(Shell_weight,2) + Whole_weight + poly(Height,2) + p
## [1] "=========Step 8: ========="
## [1] "p_value of each element:"
## poly(Viscera_weight, 2)2 poly(Viscera_weight, 2)2 poly(Viscera_weight, 2)2
##             1.784494e-02             3.687838e-03             2.732775e-02
## poly(Viscera_weight, 2)2 poly(Viscera_weight, 2)2 poly(Viscera_weight, 2)2
##             2.732775e-02             2.520125e-04             4.759043e-05
## poly(Viscera_weight, 2)2 poly(Viscera_weight, 2)2
##             2.922789e-01             2.619689e-02
## [1] "Model in step 8: "
## [1] "Rings ~ Shell_weight + Shucked_weight + poly(Shell_weight,2) + Whole_weight + poly(Height,2) + p
## [1] "=========Step 9: ========="
## [1] "p_value of each element:"
##         sexI         sexI         sexI         sexI         sexI         sexI
## 1.657219e-08 1.029289e-09 1.029289e-09 1.029289e-09 1.350760e-08 9.037636e-10
##         sexI
## 6.504389e-09
## [1] "Model in step 9: "
## [1] "Rings ~ Shell_weight + Shucked_weight + poly(Shell_weight,2) + Whole_weight + poly(Height,2) + p
## [1] "===== The final model is: Rings ~ Shell_weight + Shucked_weight + poly(Shell_weight,2) + Whole_w

##
## Call:
## lm(formula = model, data = data)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5340 -1.2866 -0.2980  0.8882 14.9804
##
## Coefficients: (2 not defined because of singularities)
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  6.40128    0.42663  15.004  < 2e-16 ***
## Shell_weight                13.91733    1.45845   9.543  < 2e-16 ***
## Shucked_weight             -18.91774    0.89934 -21.035  < 2e-16 ***
## poly(Shell_weight, 2)1           NA         NA      NA       NA
## poly(Shell_weight, 2)2    -25.32608    4.57359  -5.537 3.32e-08 ***
## Whole_weight                 8.72175    0.86608  10.070  < 2e-16 ***
## poly(Height, 2)1            31.42866    5.38720   5.834 5.97e-09 ***
## poly(Height, 2)2           -14.87287    3.24566  -4.582 4.78e-06 ***
## poly(Viscera_weight, 2)1   -66.28783   10.61278  -6.246 4.78e-10 ***
## poly(Viscera_weight, 2)2    8.10051    5.79955   1.397    0.163
## sexI                       -0.71405    0.11621  -6.145 9.04e-10 ***
## sexM                        0.04745    0.09359   0.507    0.612
## poly(Diameter, 2)1          9.91521    9.37217   1.058    0.290
## poly(Diameter, 2)2        -20.20168    3.41549  -5.915 3.68e-09 ***
## poly(Whole_weight, 2)1           NA         NA      NA       NA
## poly(Whole_weight, 2)2     11.44598    7.43764   1.539    0.124
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.121 on 3119 degrees of freedom
## Multiple R-squared:  0.5616, Adjusted R-squared:  0.5598
## F-statistic: 307.3 on 13 and 3119 DF,  p-value: < 2.2e-16
```

```
#(h)
abalone_spline <- gam(Rings~s(Shell_weight,6)+s(Shucked_weight,6)+s(Diameter,6)
                      +s(Whole_weight,6)+s(Height,6)+poly(Length,2)+
                      poly(Length,2)+poly(Diameter,2)+
                      poly(Whole_weight,2)+sex,
                      abalone_train,family="gaussian")

abalone_spline_predict <- predict(abalone_spline,abalone_train)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_spline_predict-abalone_train[,9])^2)
RSS
```

```
## [1] 13587.37
```

```
(TSS-RSS)/TSS
```

```
## [1] 0.5753479
```

```
#(i)
abalone_lm_predict <- predict(abalone_lm_fit,abalone_test_x)
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_lm_predict-abalone_full[-split_sample,9])^2)
(TSS-RSS)/TSS
```

```
## [1] 0.7629324
```

```
mean((abalone_lm_predict-abalone_full[-split_sample,9])^2)
```

```
## [1] 7.265638
```

```
abalone_lm_predict_all <- predict(abalone_lm_fit_all,abalone_test_x)
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_lm_predict_all-abalone_full[-split_sample,9])^2)
(TSS-RSS)/TSS
```

```
## [1] 0.8290767
```

```
mean((abalone_lm_predict_all-abalone_full[-split_sample,9])^2)
```

```
## [1] 5.238449
```

```
abalone_quad_5_predict <- predict(abalone_quad_5,abalone_test_x)
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_quad_5_predict-abalone_full[-split_sample,9])^2)
(TSS-RSS)/TSS
```

```
## [1] 0.7714268
```

```
mean((abalone_quad_5_predict-abalone_full[-split_sample,9])^2)
```

```
## [1] 7.005302
```

```
abalone_quad_8_predict <- predict(abalone_quad_8,abalone_test_x)
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_quad_8_predict-abalone_full[-split_sample,9])^2)
(TSS-RSS)/TSS
```

```
## [1] 0.8412828
```

```
mean((abalone_quad_8_predict-abalone_full[-split_sample,9])^2)
```

```
## [1] 4.864358
```

```
cv_ridge_predict <- predict(ridge_glmnet,s=cv_ridge$lambda.min,test_x)
mean((cv_ridge_predict-abalone_full[-split_sample,9])^2)
```

```
## [1] 5.653694
```

```
#abalone_forward_predict <- predict(abalone_forward_predict,abalone_test_x)
#TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
#RSS = sum((abalone_forward_predict-abalone_full[-split_sample,9])^2)
#(TSS-RSS)/TSS
#mean((abalone_forward_predict-abalone_full[-split_sample,9])^2)

abalone_spline_predict <- predict(abalone_spline,abalone_test_x)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

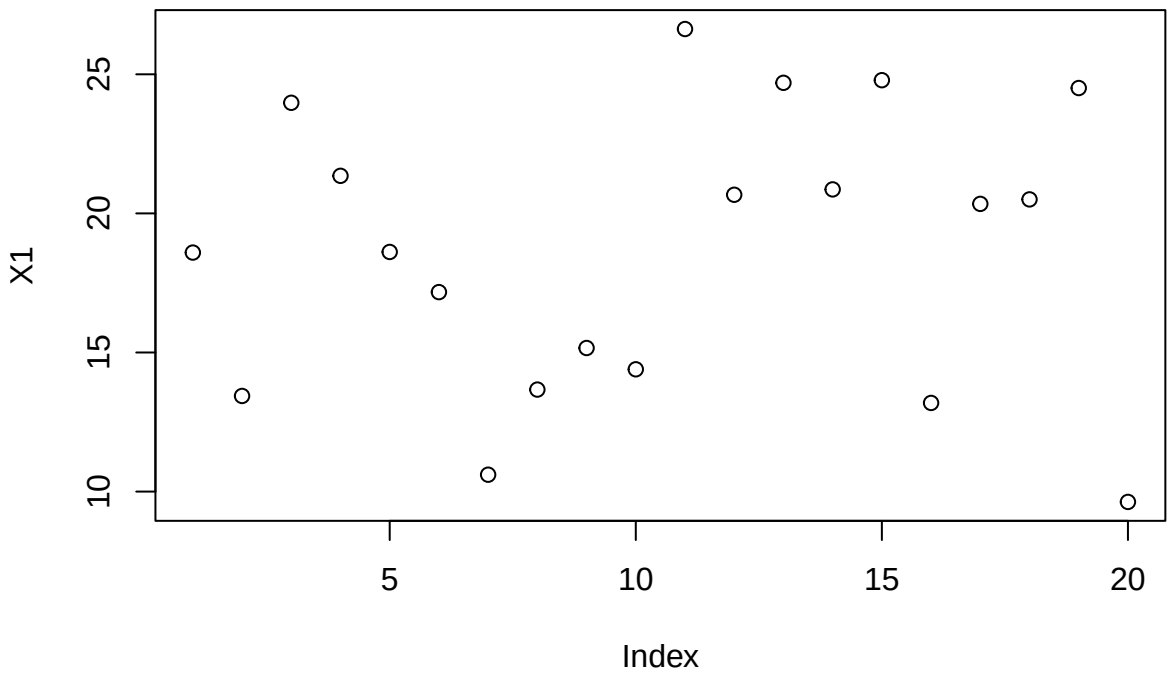
```
TSS = sum((abalone_train[,9]-mean(abalone_train[,9]))^2)
RSS = sum((abalone_spline_predict-abalone_full[-split_sample,9])^2)
(TSS-RSS)/TSS
```

```
## [1] 0.8379224
```

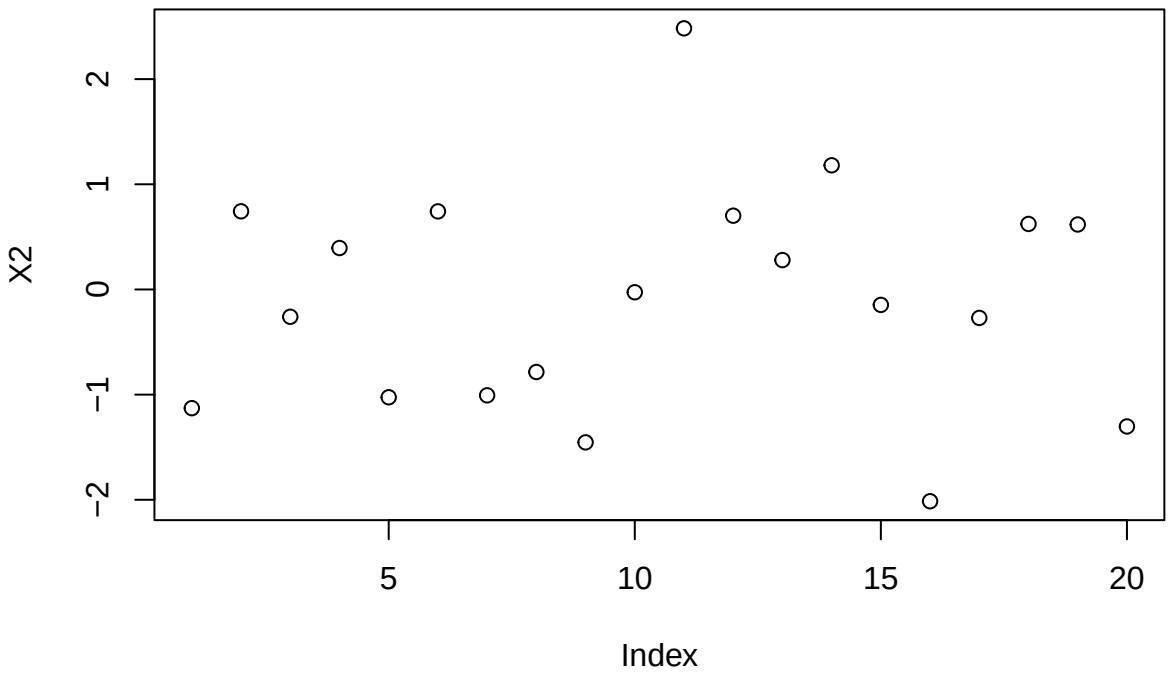
```
mean((abalone_spline_predict-abalone_full[-split_sample,9])^2)
```

```
## [1] 4.967346
```

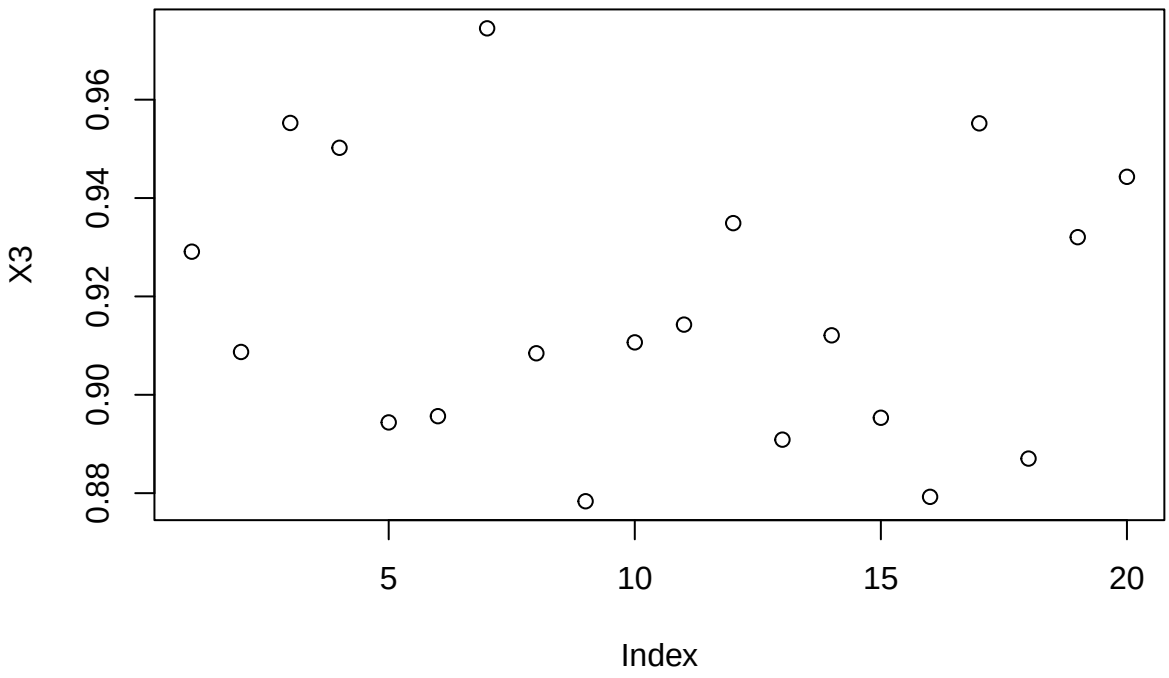
```
#Q3
set.seed(999)
X1 <- rnorm(20,20,5)
plot(X1)
```



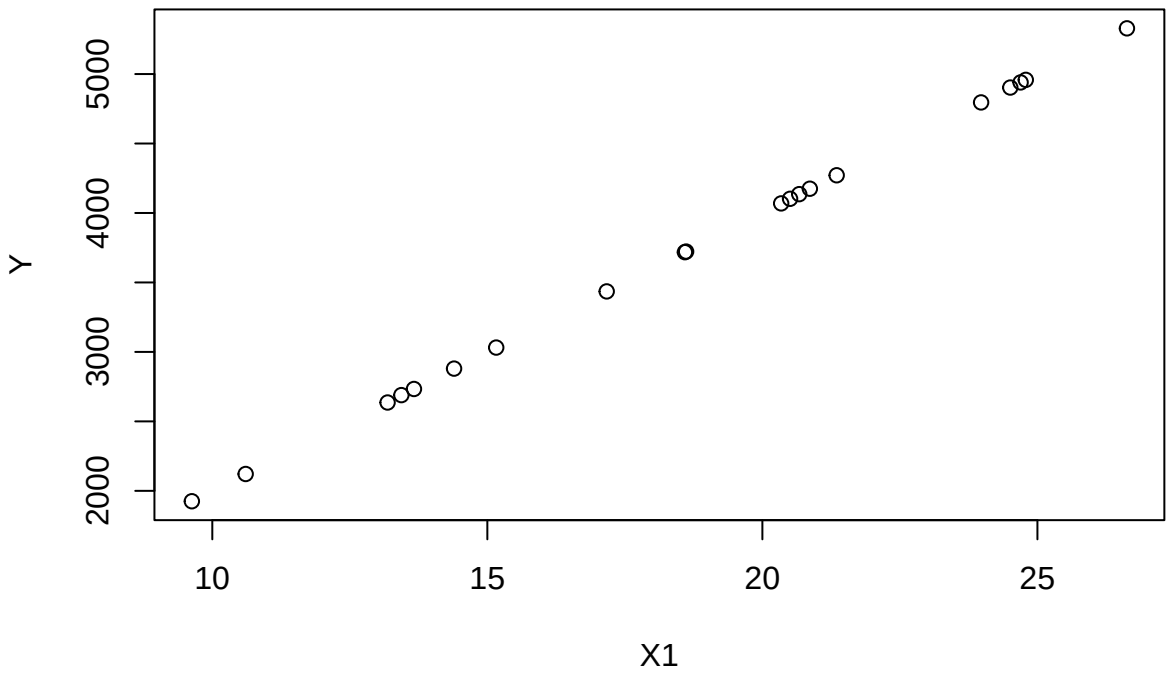

```
X2 <- rnorm(20,0.1)
plot(X2)
```



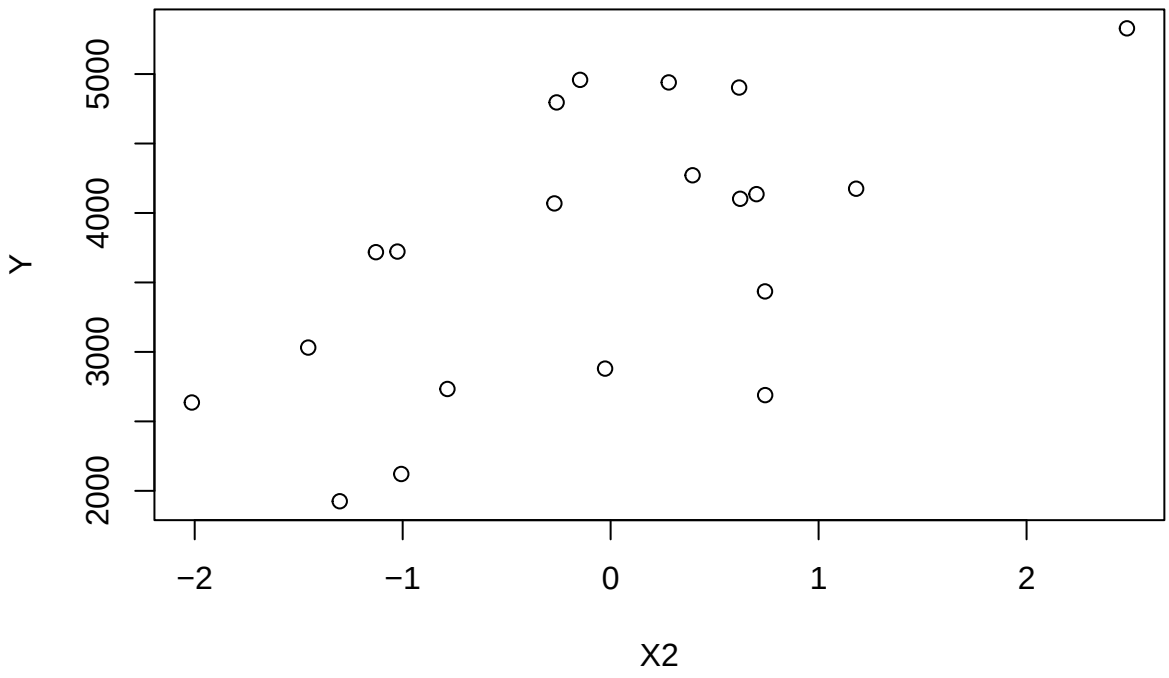

```
X3 <- rbeta(20,50,5)
plot(X3)
```



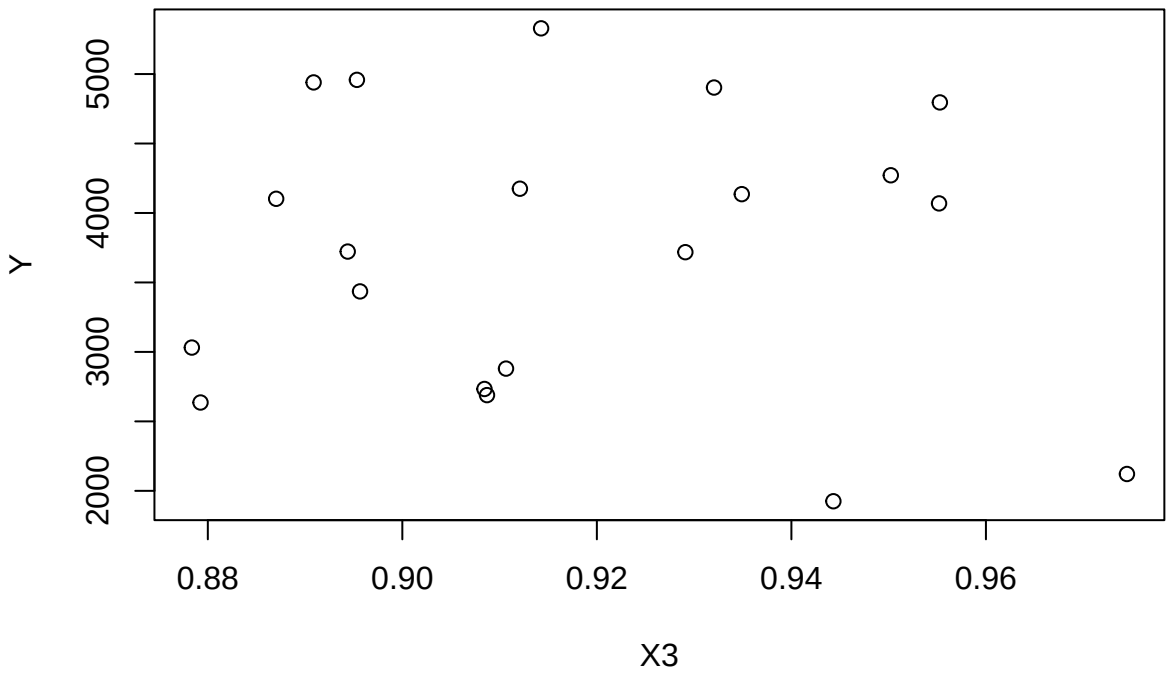

```
#Y <- (X2+X3)^2+X1+X3+10
Y <- X1*200 + X2/X3 + X3^2
#Y <- X1*200 + X2/X3 + X3^2
plot(X1,Y)
```




```
plot(X2,Y)
```




```
plot(X3,Y)
```




```
lm__fit <- lm(Y~X1+X2+X3)
summary(lm__fit)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.047295 -0.013818 -0.001936  0.024374  0.038568
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept)  -1.438972   0.210017     -6.852 3.89e-06 ***
## X1          199.997994   0.001581 126471.358  < 2e-16 ***
## X2            1.105732   0.007408    149.266  < 2e-16 ***
## X3            2.527351   0.224032     11.281 5.01e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02728 on 16 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 8.677e+09 on 3 and 16 DF,  p-value: < 2.2e-16
```

```
lm_fit_1 <- lm(Y~X1)
summary(lm_fit_1)
```

```
##
## Call:
## lm(formula = Y ~ X1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48857 -0.90338 -0.00433  0.60098  1.65801
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  -1.91427    0.84984   -2.253    0.037 *
## X1          200.14300    0.04409 4539.710   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9694 on 18 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 2.061e+07 on 1 and 18 DF,  p-value: < 2.2e-16
```

```
lm_fit_2 <- lm(Y~X2)
summary(lm_fit_2)
```

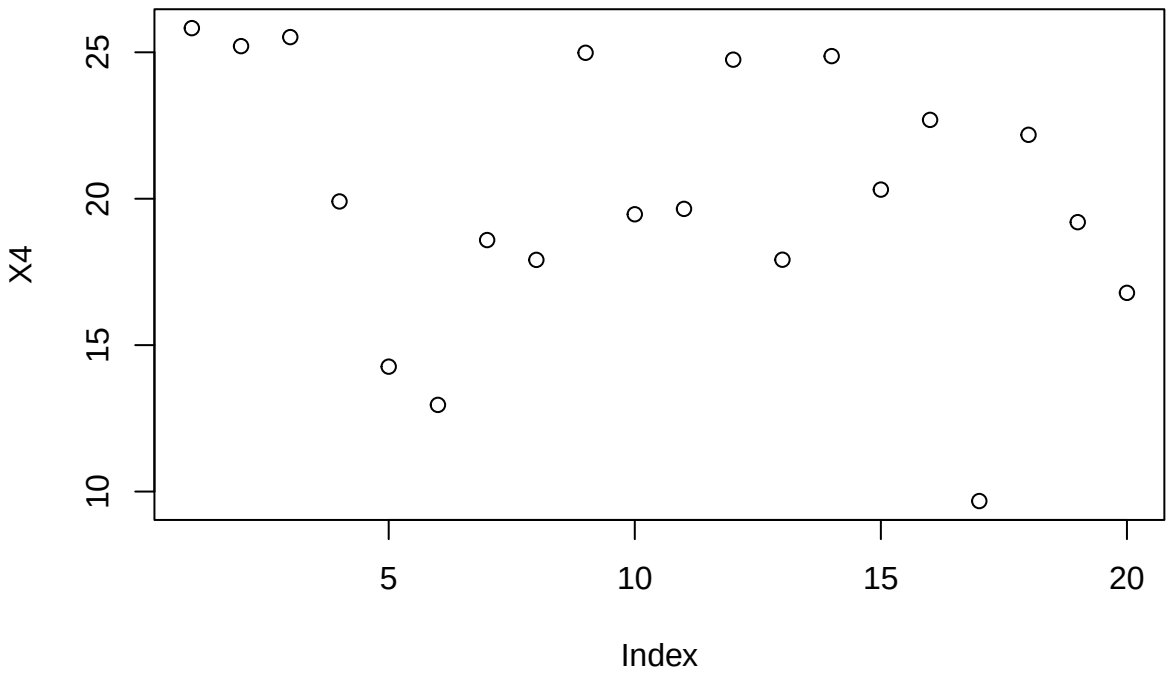
```
##
## Call:
## lm(formula = Y ~ X2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1518.37  -633.98    62.66   554.31  1266.92
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3776.5      182.9  20.649 5.55e-14 ***
## X2             580.0      173.8   3.336  0.00367 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 815.4 on 18 degrees of freedom
## Multiple R-squared:  0.3821, Adjusted R-squared:  0.3478
## F-statistic: 11.13 on 1 and 18 DF,  p-value: 0.003675
```

```
lm_fit_3 <- lm(Y~X3)
summary(lm_fit_3)
```

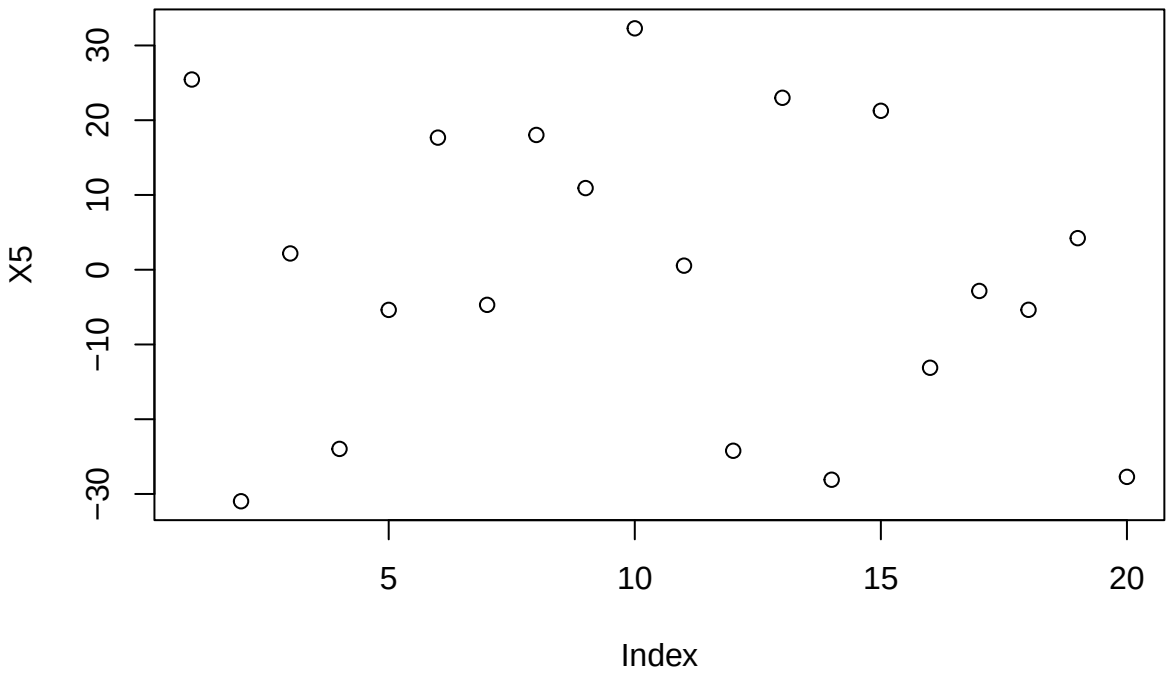
```
##
## Call:
## lm(formula = Y ~ X3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1760.4  -897.3   166.3   728.5  1595.3
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5201       7789   0.668    0.513
```

```
## X3             -1604       8485  -0.189    0.852
##
## Residual standard error: 1036 on 18 degrees of freedom
## Multiple R-squared:  0.001982,   Adjusted R-squared:  -0.05346
## F-statistic: 0.03575 on 1 and 18 DF,  p-value: 0.8521
```

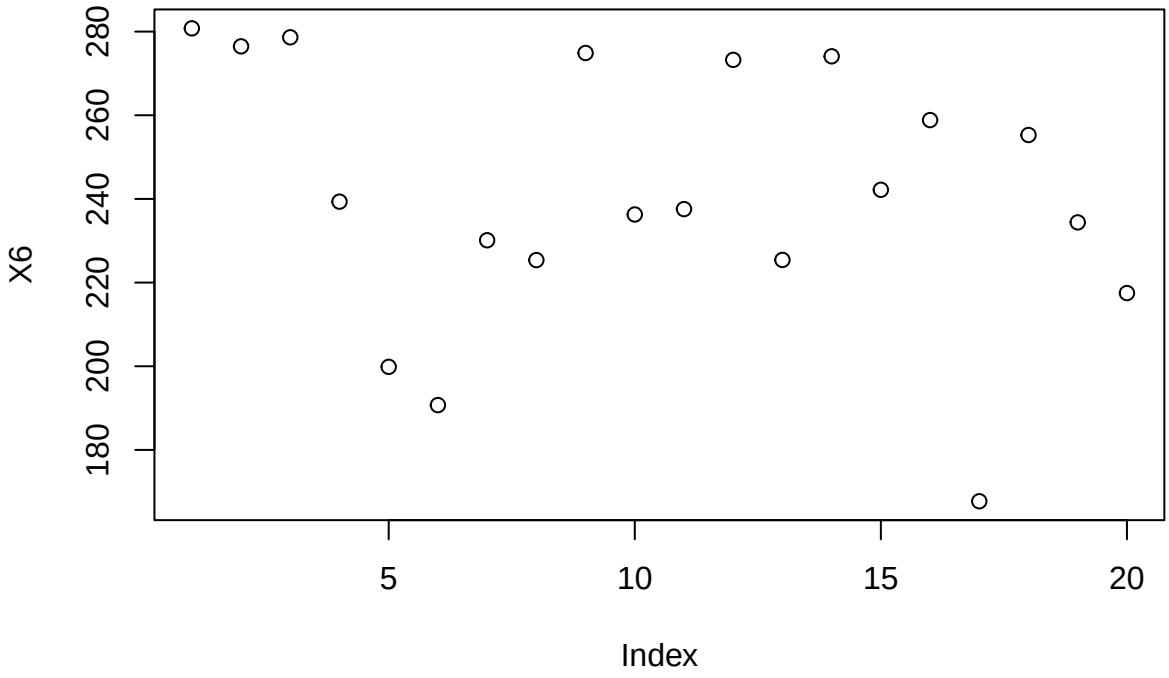
```
X4 <- rnorm(20,20,5)
plot(X4)
```



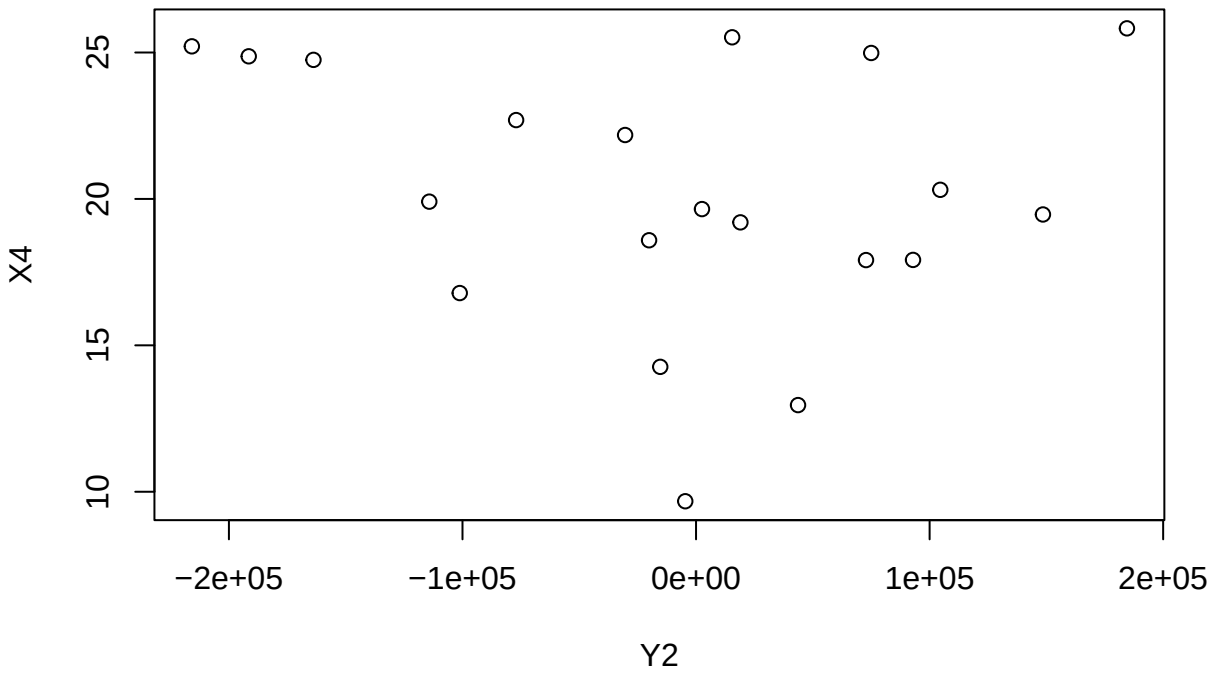

```
X5 <- X4*rnorm(20,0,1)
plot(X5)
```



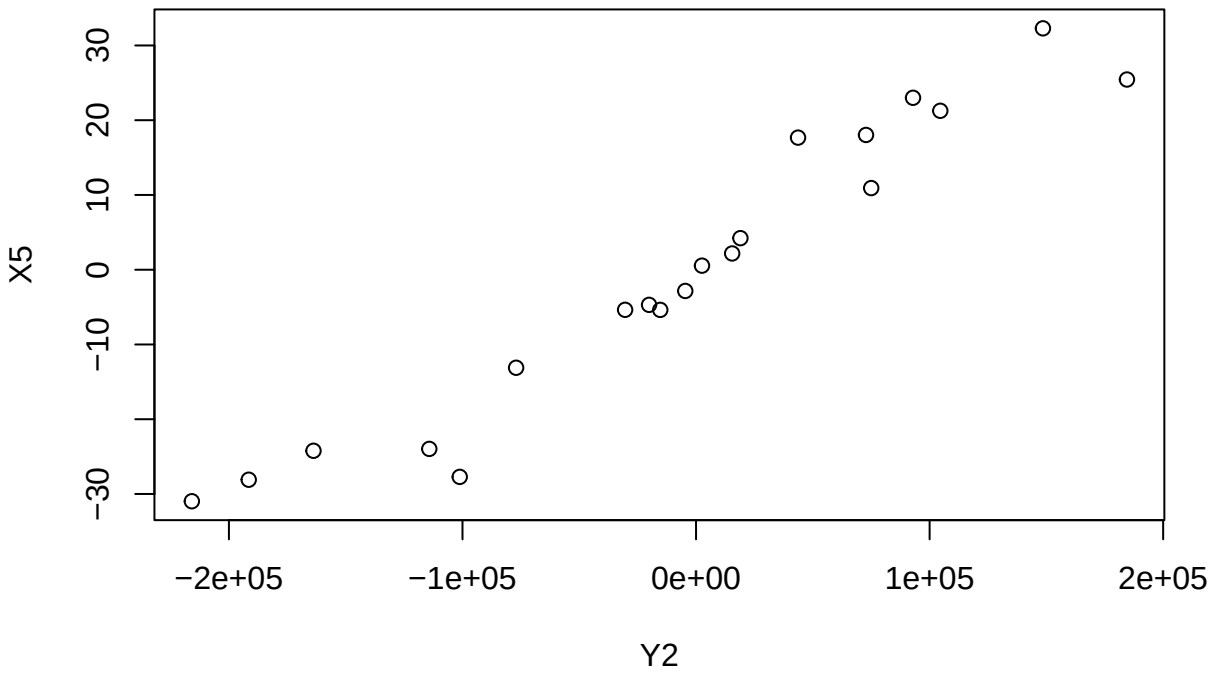

```
X6 <- 7*X4+100
plot(X6)
```



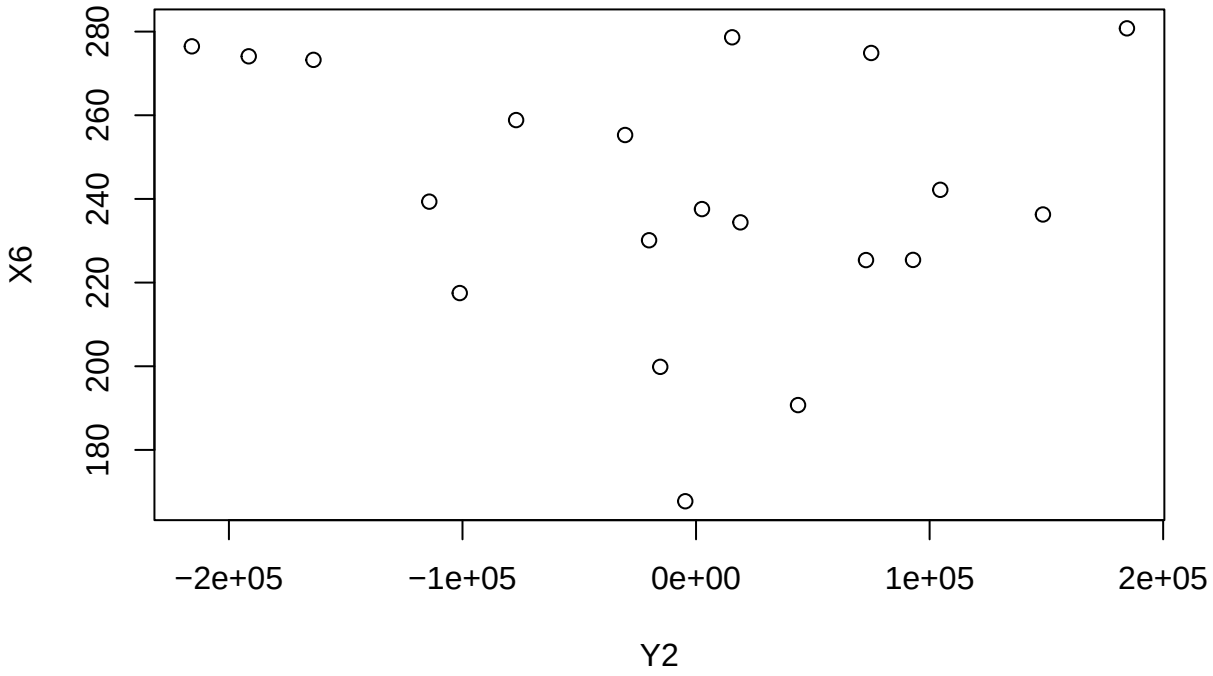

```
Y2 <- X4*X5*X6
plot(Y2,X4)
```




```
plot(Y2,X5)
```




```
plot(Y2,X6)
```




```
lm_Y2 <- lm(Y2~X4+X5+X6)
summary(lm_Y2)
```

```
##
## Call:
## lm(formula = Y2 ~ X4 + X5 + X6)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -47252 -20679   3377  14739  55270
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -487.9    31464.9  -0.016    0.988
## X4            -266.4     1529.7  -0.174    0.864
## X5            5368.3      349.6  15.357 2.13e-11 ***
## X6                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29520 on 17 degrees of freedom
## Multiple R-squared:  0.9351, Adjusted R-squared:  0.9275
## F-statistic: 122.6 on 2 and 17 DF,  p-value: 7.969e-11
```

```
lm_X4 <- lm(Y2~X4)
summary(lm_X4)
```

```
##
## Call:
## lm(formula = Y2 ~ X4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -183850  -69035   -1567   76375  219346
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    83556     116128   0.720    0.481
## X4             -4585       5635  -0.814    0.427
##
## Residual standard error: 110600 on 18 degrees of freedom
## Multiple R-squared:  0.03547,    Adjusted R-squared:  -0.01812
## F-statistic: 0.6619 on 1 and 18 DF,  p-value: 0.4265
```

```
lm_X5 <- lm(Y2~X5)
summary(lm_X5)
```

```
##
## Call:
## lm(formula = Y2 ~ X5)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -45545 -20740   3229  17252  53750
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5844.6     6423.9   -0.91    0.375
## X5            5379.5      334.2   16.09 3.94e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28720 on 18 degrees of freedom
## Multiple R-squared:  0.935,  Adjusted R-squared:  0.9314
## F-statistic: 259.1 on 1 and 18 DF,  p-value: 3.942e-12
```

```
lm_X6 <- lm(Y2~X6)
summary(lm_X6)
```

```
##
## Call:
## lm(formula = Y2 ~ X6)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -183850  -69035   -1567   76375  219346
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 149053.2   195539.4   0.762    0.456
## X6            -655.0      805.1  -0.814    0.427
##
```

```
## Residual standard error: 110600 on 18 degrees of freedom
## Multiple R-squared:  0.03547,    Adjusted R-squared:  -0.01812
## F-statistic: 0.6619 on 1 and 18 DF,  p-value: 0.4265
```

Include commented exam work

Q1(b)

##in the summary of teh tree output, we can see all the split variables along ##with next important variables.

interpreting the tree

The tree output proves that the most important variable was physician.

##That explains the first split. If a candidate is not physician, ##the vote goes to democrat but when a canditate is physician, further ## splits are required. If the candidate is a physician and not synfuels, the ## vote goes to republic. In case of synfuels with no education, ## the vote is for democrat. Otherwise, republican.

##Q1(c) ##comparing both trees ## From the pruned tree we notice that only one split remains thats because ## of the cp factor we added. The splits in the previous tree were not ## decreasing the lack of fit by this factor. But since we set factor to 0, ## the whole tree was given as output. ## Hence it means that having those additional splits was not contributing ## to the explainability significantly. ## This is evident from the same test misclassification error.

##Q1(d) ## both the trees are used to predict and both predict the same (republican)

For unpruned tree, we notice that the tree requires the input not

to be a physician for "democrat" vote. But our input is a physician and also

the new candidiate is synfuels and educated. Hence vote goes to "republican"

For Pruned tree, since our new candidate is a physician, it classifies

as "republican"

end Q1

##Q2(b) ## the data is explored graphically ## the spread of the output variable is explored ## looks like normal distribution approximately ## covarainace and correlation matrix are explored to look at the relationship between ## the variables. ## Look at the scatter plots of all predictor variables with output. ## For readability, lets see the ones that are strongly correlated ## Some variables seem to be strongly correlated to the output variable ## Look at the plot of such variables with Y ## We can intuitively say that these variables will contribute to the ## explainability of the output varaibles more than others.

##Q2(c) ## Here we built a model using only the train set and also only the first 5 ## variables. ## The nominal variable does not need any special care as dummy variables are ##created by lm() automatically. ## the first model has all 5 variables and the second model doesn't have ## Whole_weight included as it was not significant from the first model. ##From teh output, we see that abalone being M or not is not

significant. ## Being I is significant by factor of -1.11 and F by intercept 4.28 ## For both these, we have calculated the TSS,RSS and proportion explained ## The explainability of the model is only 37% which is pretty low. ## HeNce, better models are required.

##Q2(d) ## Here we built a model using only the train set and all 8 variables. ## the first model has all 8 variables and the second model doesn't have ## Length included as it was not significant from the first model. ## For both these, we have calculated the TSS,RSS and proportion explained ## The explainability of the model is only 53% which higher than previous model ## This means all 8 variables are important for the model. ## However, better models are required.

##Q2(e) ## Here we fit a quadratic model to the data using 5 variables. ## But the variable 'sex' cannot be quadratic. Hence, we either use sex directly. ## I have created a variable using I() dor sex==I because from previous ##we know that it is significant. ## This model has a very low accuarcy. We can see that from the RSS ## and explainability. It is much lower than previous models. ##Hence, we need to add more variables or remove quadratic behaviour.

##Next, we use all 8 predictors with quadratic regression ## Here, we see that we have used 'sex' as is without any transformation ## We see that the results are better than previous models ## This proves that we need all the variables and that quadratic behaviour ## is present in the data ##RSS is much lower compared to the 5 variable model

##Q2(f) ## To fit a ridge regression we first have to use model.matrix() ## to convert 'sex' to dummy vars ## Next we have to divide the train data into 4 to parts, fit a ridge model ## on 3 parts and test the selected lambda on teh 4th part. We have to also ## capture the squared error while selecting the best lambda. ## ALl this can be done by using cv.glmnet with alpha=0 ## First we fit a plain ridge with default parameters. ##This will help in final testing. ## Next, fit a coross validated ridge with 4 folds. ## The plot of this output shows us how squared error varies by lambda ## fianlly we obtain the lambda for which the error is lowest.

##Q2(g) ## Here we are asked to use the stepwise model selection method to choose ## the model. However, we are asked to use only 3133 points. So no test data ## can be used. ## We use the forward method because we saw previosuly that almost all varaibles ## to explain teh variability in this data. ## We also add the squared predictor variables to see if they get selected ##So forward selection will eleminate ## only those that are not required. ## From the resulst we see that the selected model is ##combination of linear and quadratic variable models.

##Q2(h) ## I added smooth splines and quadratic terms in teh model and ##it gives slightly better results compared to the other models. ## I added quadratic terms because they were chosen to be the best ## in previous model selection ## I chose smooth splines because just quadratic was not enough to capture data ## and the non linear behaviour is best explained by cubic splines ## and they are continuous too

##Q2(i) ## Here we test all the models that we have built ## we notice that the quadtratic with all variables is still the best model ## we can see taht from the squared error and the explainability ## Even in the test data we see that quadratic had good performance too. ## Thsi means that both the bias and the variance of this model is good ## the division of test train captured teh variability well while having ## to test on.

##Q3 ## To get a data set with no vars significant in regression but individually ## significant with the response, all the variables must be highly correlated. ## and teh response must be a linear combination of all variables ## in that case together they will not have any explainability, but individually ## they can explain the variance in reponse

for the opposite case,

all the variables must have no relationship to each other but the response should be a multiplicative combination of the variables

Tn this case no variable can individually be significant in explaining

##the variance in the data but together they will be important and significant