Assignment 2

Dave Anderson

February 10, 2019

4.6, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13

4.6)

a)***38%***

b)***10 More hours, 50 total***

4.8)

With KNN=1, the training error rate will be 0% because the nearest neighbor is the response itself. That gives KNN a 38% test rate, so I would chose logistic with a lower test error rate

4.9)

a)

$P(x)/1 - P(x) = .37$ $P(X) = 0.27$ *27% chance of defaulting*

b)

$0.16/(1 - 0.16) = 0.19$ *The odds are 0.19*

4.10

```r
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.5.2
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(MASS)
```

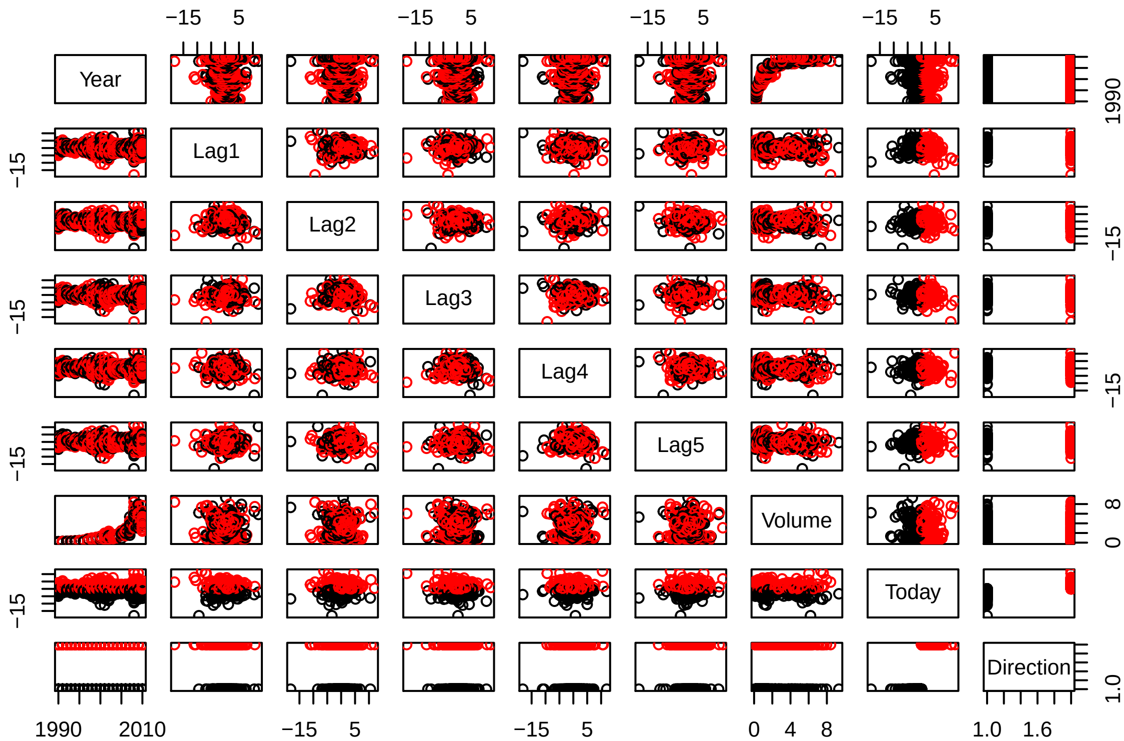
```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

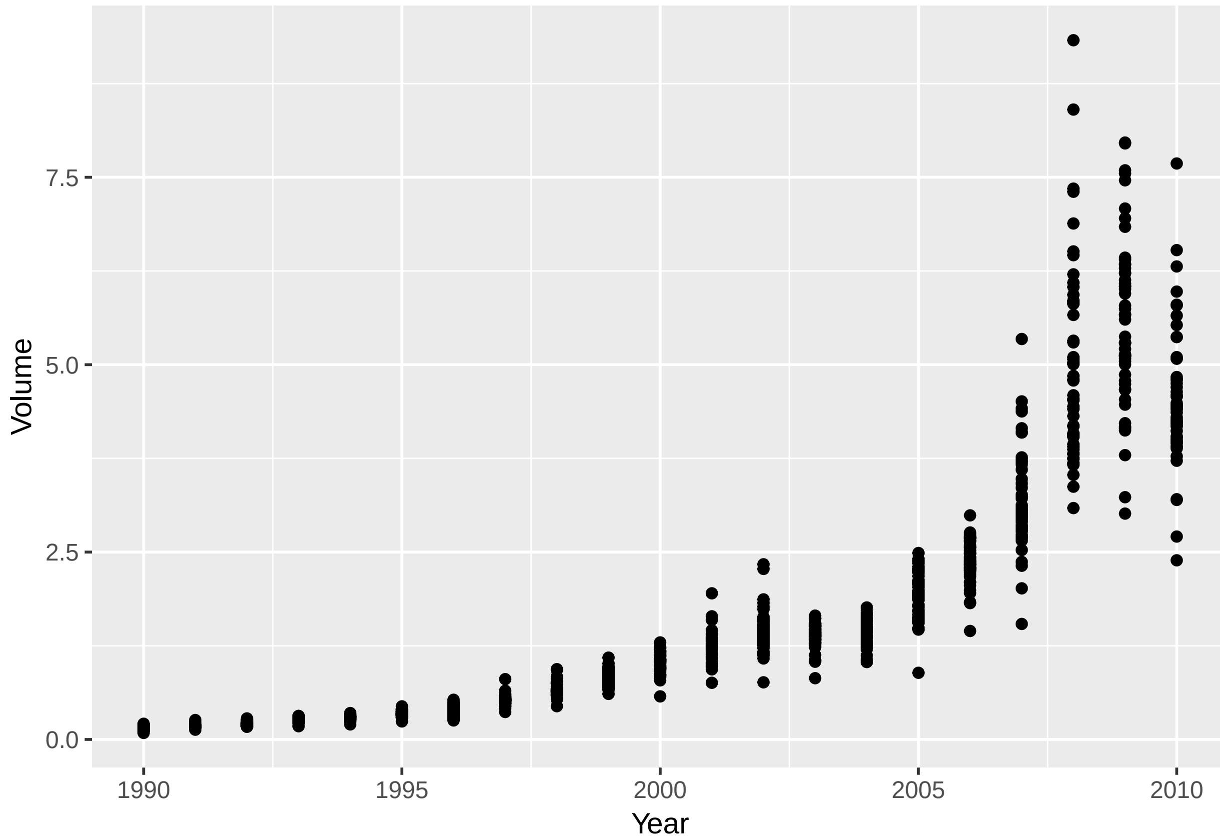
```r
library(class)

data <- as.data.frame(Weekly)
summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
##      Today          Direction
##  Min.   :-18.1950   Down:484
##  1st Qu.: -1.1540   Up  :605
##  Median :  0.2410
##  Mean   :  0.1499
##  3rd Qu.:  1.4050
##  Max.   : 12.0260
```

```r
pairs(data, col = data$Direction)
```




```
ggplot(data)+
  geom_point(aes(Year,Volume))
```



There aren't really any patterns within the variables beyond the correlation between year and volume.

b)

```
glm1 <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = data, family = binomial)
summary(glm1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

Lag2 is the only statistically significant coefficient with a p-value of 0.03

c)

```r
probs <- predict(glm1, type = "response")
pred.glm <- rep("Down", length(probs))
pred.glm[probs > 0.5] <- "Up"
table(pred.glm, data$Direction)
```

```
##
## pred.glm Down  Up
##     Down   54  48
##     Up    430 557
```

Our training error rate is (430+48)/1089 = 43%. So our model did not work very well on our training data. The model over-predicted "up" by quite a bit. ie large false positive rate.

d)

```r
train <- data %>% filter(Year <= 2008)
test <- data %>% filter(Year >= 2009)

glm2 <- glm(Direction ~ Lag2, data = train, family = binomial)
summary(glm2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
```

```
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

```r
probs2 <- predict(glm2, test, type = "response")
pred.glm2 <- rep("Down", length(probs2))
pred.glm2[probs2 > 0.5] <- "Up"
table(pred.glm2, test$Direction)
```

```
##
## pred.glm2 Down Up
##      Down    9  5
##      Up     34 56
```

Here we have a better test error rate of about 38%, but we still see a high number of false positives

e)

```r
lda <- lda(Direction ~ Lag2, data = train)
summary(lda)
```

```
##         Length Class  Mode
## prior   2      -none- numeric
## counts  2      -none- numeric
## means   2      -none- numeric
## scaling 1      -none- numeric
## lev     2      -none- character
## svd     1      -none- numeric
## N       1      -none- numeric
## call    3      -none- call
## terms   3      terms  call
## xlevels 0      -none- list
```

```r
pred.lda <- predict(lda, test)
table(pred.lda$class, test$Direction)
```

```
##
##        Down Up
##   Down    9  5
##   Up     34 56
```

The lda approach actually resulted in the same confusion matrix as the logistic regression.

f)

```r
qda <- qda(Direction ~ Lag2, data = train)
pred.qda <- predict(qda, test)
table(pred.qda$class, test$Direction)
```

```
##
##        Down Up
##   Down    0  0
##   Up     43 61
```

The QDA model predicts the up direction 100% of the time. The error rate doesn't seem bad on paper, but it will be wrong everytime the market goes down.

g)

```
set.seed(2019)
```

h)

Logistic and LDA both provide smaller, similar test rates

4.11)

a)

```
attach(Auto)
```

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

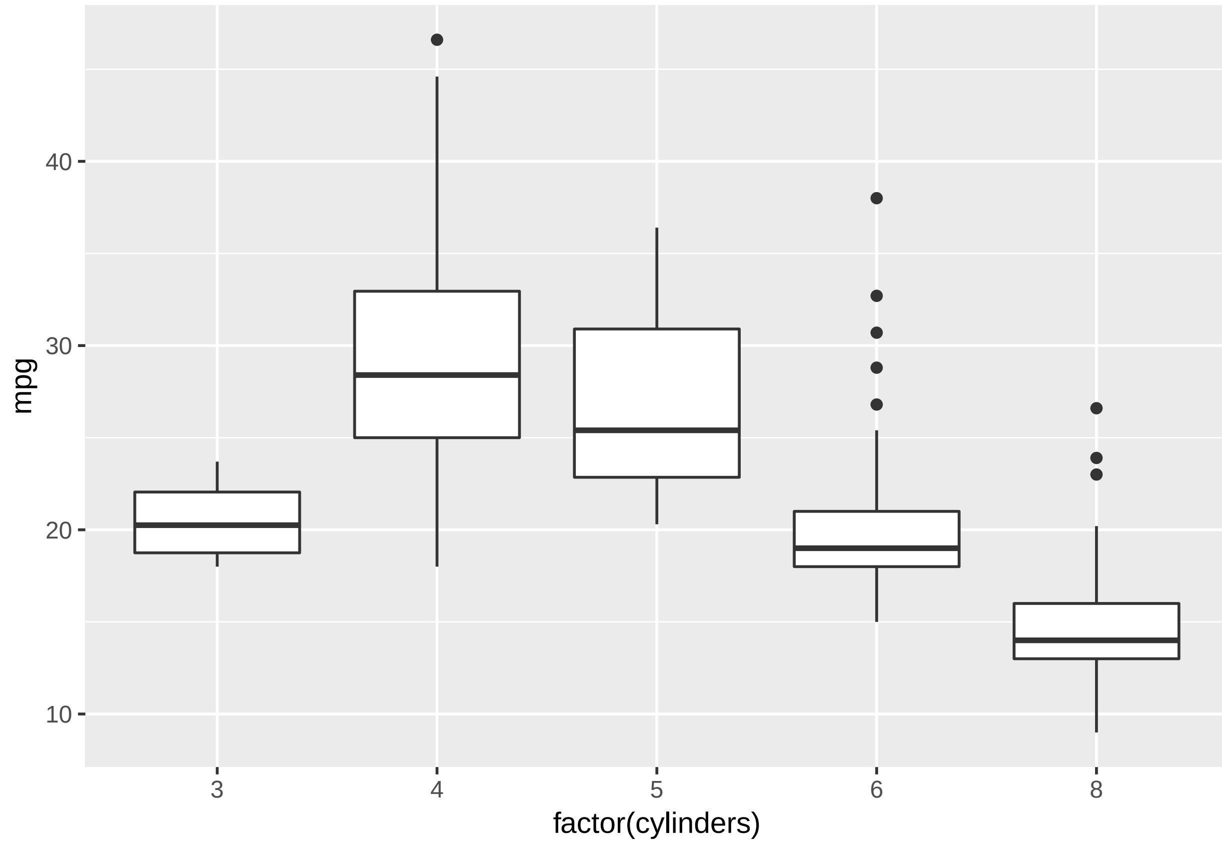
```
mpg01 <- rep(0,length(mpg))
mpg01[mpg > median(mpg)] = 1
auto <- data.frame(Auto,mpg01)
```

b)

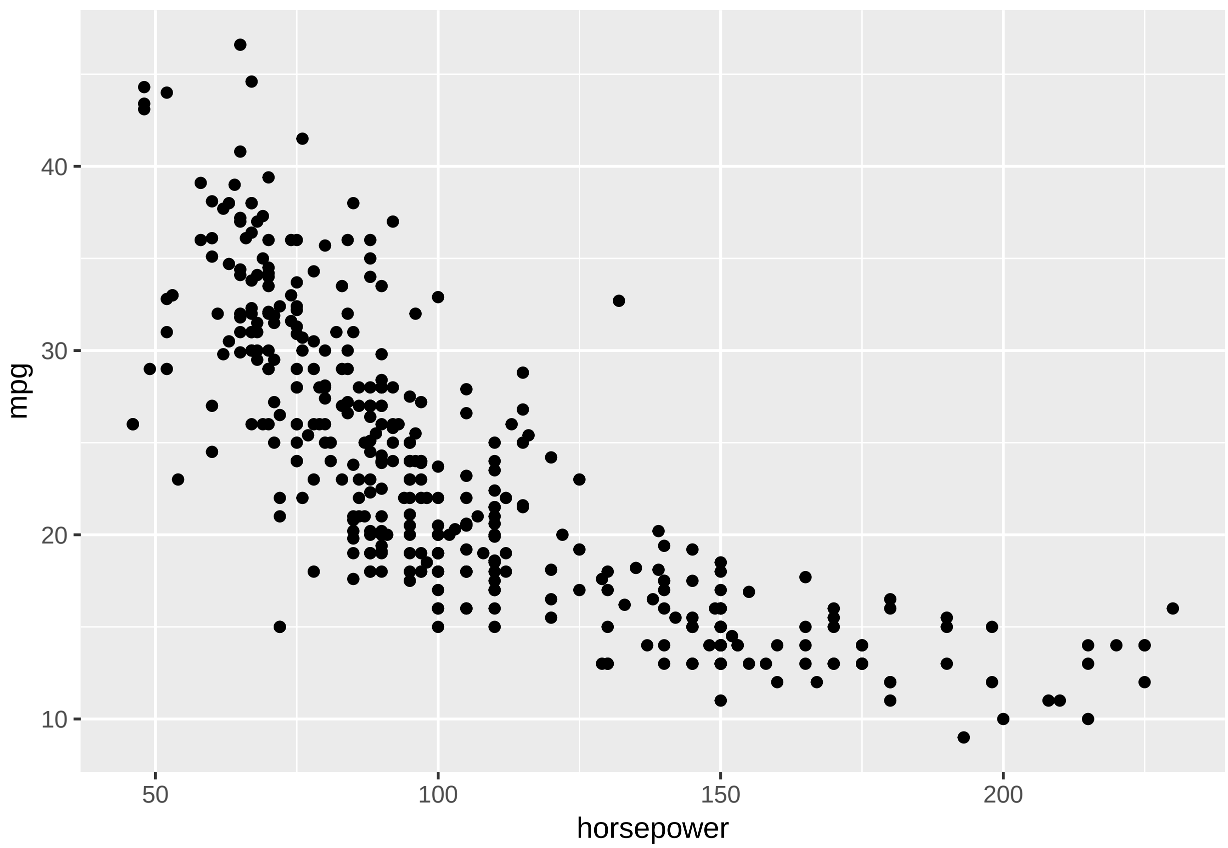
```
cor(auto[,-9])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
## mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

```r
ggplot(auto)+geom_boxplot(aes(factor(cylinders),mpg, group = factor(cylinders)))
```




```r
ggplot(auto)+geom_point(aes(horsepower,mpg))
```



There seems to be a negative relationship between cylinders, weight, displacement, horsepower with MPG

c)

```
train <-  (year%%2 == 0)  # if the year is even
test <- !train
auto.train <-  auto[train, ]
auto.test <-  auto[test, ]
mpg01.test <-  mpg01[test]
```

d)

```
lda.fit <- lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = auto.train)
lda.pred <-  predict(lda.fit, auto.test)
mean(lda.pred$class != mpg01.test)
```

```
## [1] 0.1263736
```

12.6% test error rate

e)

```r
qda.fit <-  qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = auto.train)
qda.pred <-  predict(qda.fit, auto.test)
mean(qda.pred$class != mpg01.test)
```

```
## [1] 0.1318681
```

13% Test error rate

f)

```r
glm.fit <-  glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = auto.train, family = bin
glm.probs <- predict(glm.fit, auto.test, type = "response")
glm.pred <-  rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != mpg01.test)
```

```
## [1] 0.1208791
```

12.1% test error rate

g)

```r
train.X <-  cbind(cylinders, weight, displacement, horsepower)[train, ]
test.X <-  cbind(cylinders, weight, displacement, horsepower)[test, ]
train.mpg01 <-  mpg01[train]
set.seed(1)
# KNN(k=1)
knn.pred <-  knn(train.X, test.X, train.mpg01, k = 1)
mean(knn.pred != mpg01.test)
```

```
## [1] 0.1538462
```

```r
#KNN(k = 10)
knn.pred2 <-  knn(train.X, test.X, train.mpg01, k = 10)
mean(knn.pred2 !=  mpg01.test)
```

```
## [1] 0.1648352
```

```r
#KNN(k = 100)
knn.pred3 <- knn(train.X, test.X, train.mpg01, k = 100)
mean(knn.pred3 != mpg01.test)
```

```
## [1] 0.1428571
```

$K = 1$ performs better than $K = 10$, but $K = 100$ is the best with a test error rate of 14.3% Logistic regression had the lowest test error of them all.

4.12)

a)

```r
Power <-  function() {
    2^3
}
print(Power())
```

```
## [1] 8
```

b)

```r
Power2 <-  function(x, a) {
    x^a
}
Power2(3, 8)
```

```
## [1] 6561
```

c)

```r
Power2(10,3)
```

```
## [1] 1000
```

```r
Power2(8,17)
```

```
## [1] 2.2518e+15
```

```r
Power2(131,3)
```

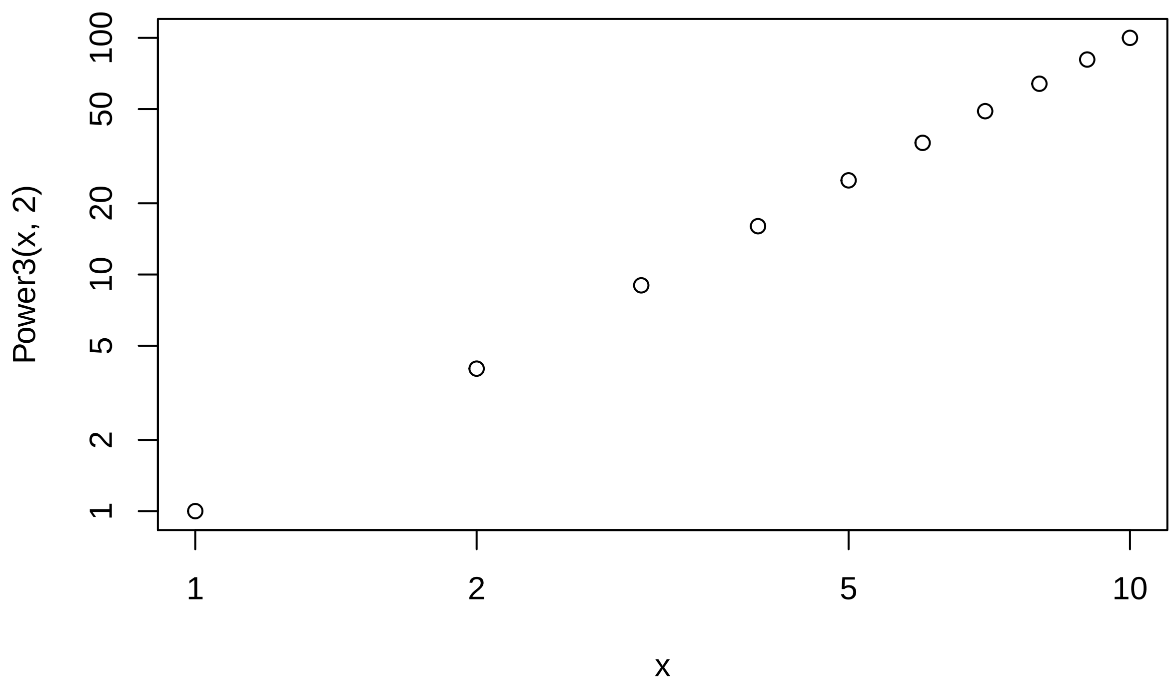
```
## [1] 2248091
```

d)

```r
Power3 <-  function(x, a) {
    result <-  x^a
    return(result)
}
```

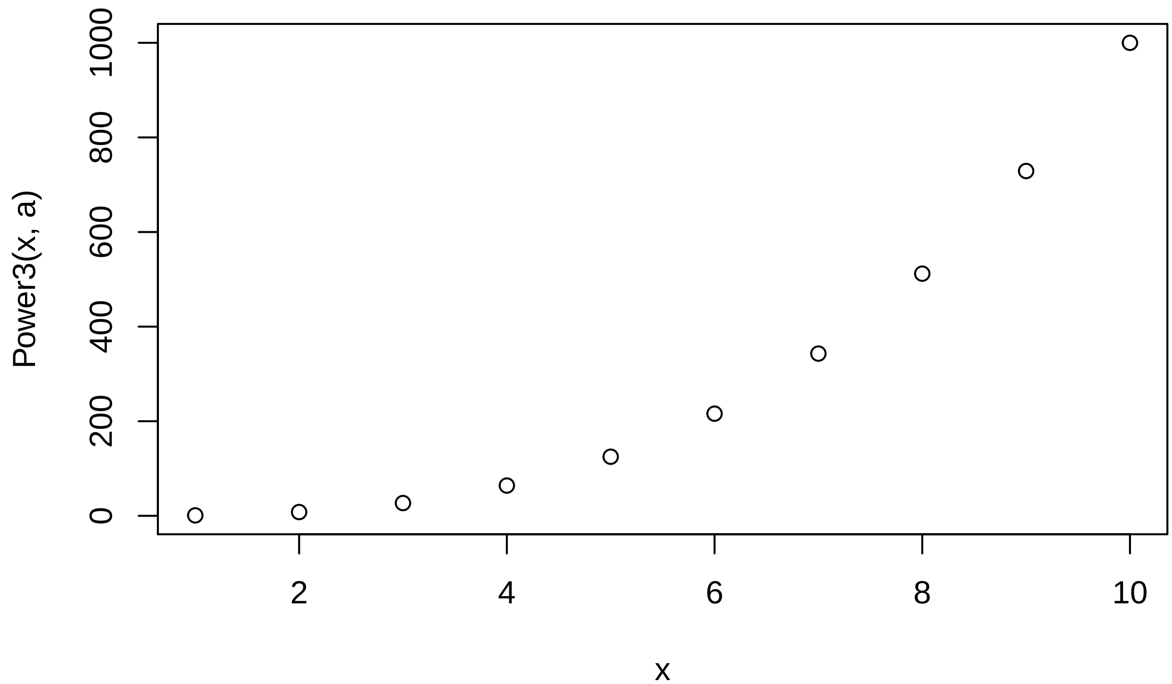
e)

```r
x <- 1:10
plot(x,Power3(x,2), log = "xy")
```



###f)

```
PlotPower <-  function(x, a) {
    plot(x, Power3(x, a))
}
PlotPower(x, 3)
```



4.13)

a)

```
attach(Boston)
crime01 <-  rep(0, length(crim))
crime01[crim > median(crim)] = 1
boston <-  data.frame(Boston, crime01)

train <-  1:(dim(boston)[1]/2)
test <-  (dim(boston)[1]/2 + 1):dim(boston)[1]
boston.train <-  boston[train, ]
boston.test <-  boston[test, ]
crime01.test = crime01[test]

cor(boston)
```

```
##                crim          zn       indus         chas         nox
## crim     1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## zn      -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## indus    0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## chas    -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
## nox      0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
## rm      -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
```

```
## age      0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## dis     -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## rad      0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## tax      0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## ptratio  0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## black   -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
## lstat    0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## medv    -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077
## crime01  0.40939545 -0.43615103  0.60326017  0.070096774  0.72323480
##                 rm         age         dis          rad         tax
## crim    -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431
## zn       0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332
## indus   -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018
## chas     0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652
## nox     -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320
## rm       1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783
## age     -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559
## dis      0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158
## rad     -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819
## tax     -0.29204783  0.50645559 -0.53443158  0.910228189  1.00000000
## ptratio -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304
## black    0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801
## lstat   -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341
## medv     0.69535995 -0.37695457  0.24992873 -0.381626231 -0.46853593
## crime01 -0.15637178  0.61393992 -0.61634164  0.619786249  0.60874128
##            ptratio       black      lstat       medv     crime01
## crim     0.2899456 -0.38506394  0.4556215 -0.3883046  0.40939545
## zn      -0.3916785  0.17552032 -0.4129946  0.3604453 -0.43615103
## indus    0.3832476 -0.35697654  0.6037997 -0.4837252  0.60326017
## chas    -0.1215152  0.04878848 -0.0539293  0.1752602  0.07009677
## nox      0.1889327 -0.38005064  0.5908789 -0.4273208  0.72323480
## rm      -0.3555015  0.12806864 -0.6138083  0.6953599 -0.15637178
## age      0.2615150 -0.27353398  0.6023385 -0.3769546  0.61393992
## dis     -0.2324705  0.29151167 -0.4969958  0.2499287 -0.61634164
## rad      0.4647412 -0.44441282  0.4886763 -0.3816262  0.61978625
## tax      0.4608530 -0.44180801  0.5439934 -0.4685359  0.60874128
## ptratio  1.0000000 -0.17738330  0.3740443 -0.5077867  0.25356836
## black   -0.1773833  1.00000000 -0.3660869  0.3334608 -0.35121093
## lstat    0.3740443 -0.36608690  1.0000000 -0.7376627  0.45326273
## medv    -0.5077867  0.33346082 -0.7376627  1.0000000 -0.26301673
## crime01  0.2535684 -0.35121093  0.4532627 -0.2630167  1.00000000
```

```r
#logistic
glm.fit <- glm(crime01 ~ . -crime01 -crim, family = binomial, data = boston.train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(glm.fit)
```

```
##
## Call:
## glm(formula = crime01 ~ . - crime01 - crim, family = binomial,
##     data = boston.train)
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -2.83229  -0.06593   0.00000   0.06181   2.61513
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -91.319906  19.490273  -4.685 2.79e-06 ***
## zn           -0.815573   0.193373  -4.218 2.47e-05 ***
## indus         0.354172   0.173862   2.037  0.04164 *
## chas          0.167396   0.991922   0.169  0.86599
## nox          93.706326  21.202008   4.420 9.88e-06 ***
## rm           -4.719108   1.788765  -2.638  0.00833 **
## age           0.048634   0.024199   2.010  0.04446 *
## dis           4.301493   0.979996   4.389 1.14e-05 ***
## rad           3.039983   0.719592   4.225 2.39e-05 ***
## tax          -0.006546   0.007855  -0.833  0.40461
## ptratio       1.430877   0.359572   3.979 6.91e-05 ***
## black        -0.017552   0.006734  -2.606  0.00915 **
## lstat         0.190439   0.086722   2.196  0.02809 *
## medv          0.598533   0.185514   3.226  0.00125 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 329.367  on 252  degrees of freedom
## Residual deviance:  69.568  on 239  degrees of freedom
## AIC: 97.568
##
## Number of Fisher Scoring iterations: 10
```

```r
glm.probs <-  predict(glm.fit, boston.test, type = "response")
glm.pred <-  rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != crime01.test)
```

```
## [1] 0.1818182
```

```r
#logistic2
glm.fit2 <- glm(crime01 ~ nox + age + rad + ptratio + black + medv, data = boston.train, family = binomi
summary(glm.fit2)
```

```
##
## Call:
## glm(formula = crime01 ~ nox + age + rad + ptratio + black + medv,
##     family = binomial, data = boston.train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.85430  -0.32230  -0.07623   0.23434   2.52859
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -34.167180   7.173211  -4.763 1.91e-06 ***
## nox          40.214386   7.957219   5.054 4.33e-07 ***
## age           0.006875   0.012795   0.537   0.5910
```

```
## rad           0.762453   0.164294   4.641 3.47e-06 ***
## ptratio       0.577746   0.142777   4.046 5.20e-05 ***
## black        -0.012204   0.007338  -1.663   0.0963 .
## medv          0.091497   0.039595   2.311   0.0208 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 329.37  on 252  degrees of freedom
## Residual deviance: 136.54  on 246  degrees of freedom
## AIC: 150.54
##
## Number of Fisher Scoring iterations: 7
```

```
glm.probs2 <-  predict(glm.fit2, boston.test, type = "response")
glm.pred2 <-  rep(0, length(glm.probs2))
glm.pred2[glm.probs2 > 0.5] = 1
mean(glm.pred2 != crime01.test)
```

```
## [1] 0.1067194
```

```
#LDA
lda.fit <- lda(crime01 ~ nox + age + rad + ptratio + black + medv, data = boston.train)
lda.pred <- predict(lda.fit,boston.test)
mean(lda.pred$class != crime01.test)
```

```
## [1] 0.1185771
```

Logistic regression with selected variables performed the best