Stats 101C HW2 - Hayley Todd 904637605

Hayley Todd

Question 1:

Use ggplot2 to create a graphic (trying to predict the house price), based on the LArealestate.csv data from week 3 that shows us 3 (or more) variables on the same plot. What questions does the graphic answer?

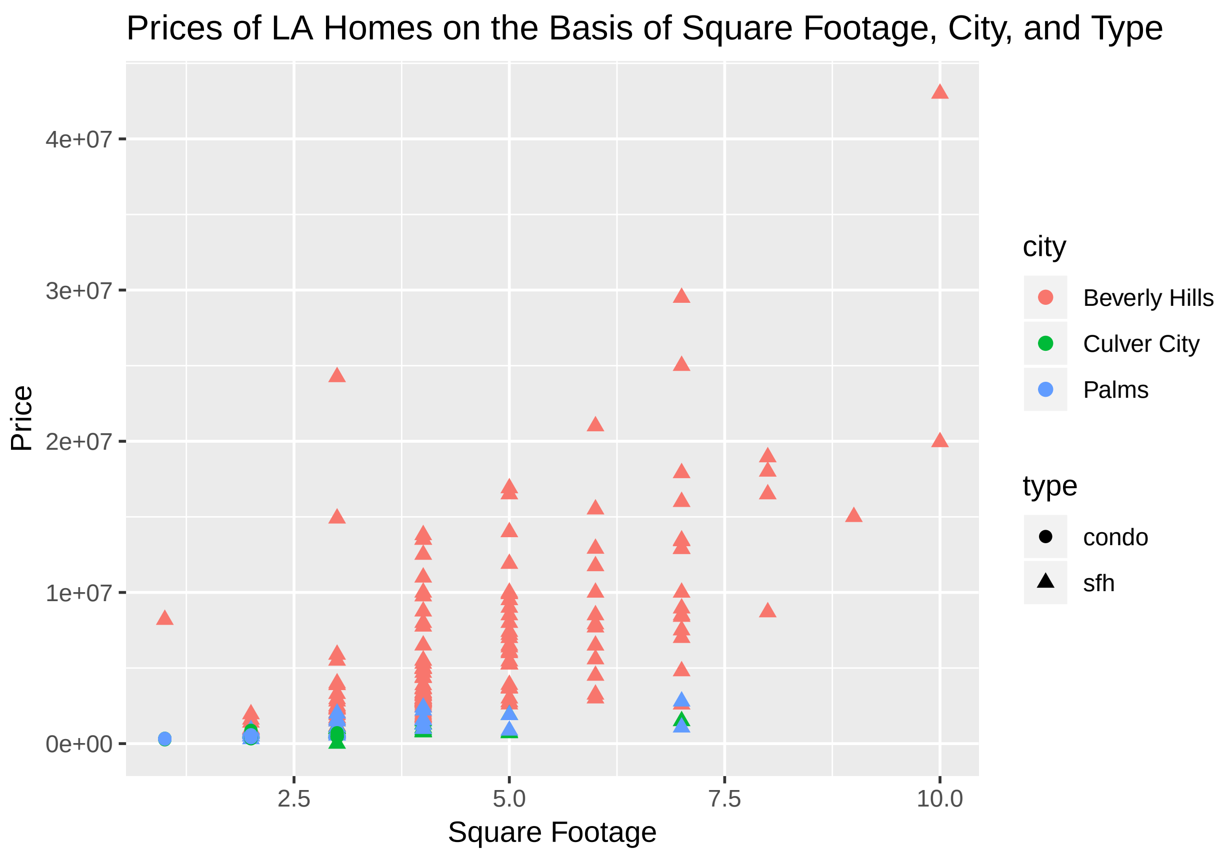
```
LAdata <- read.csv("LArealestate.csv")
library("ggplot2")
attach(LAdata)
head(LAdata)
```

```
##                      address beds baths  sqft
## 1   1005 Benedict Canyon Dr    8  11.0 10379
## 2        10084 Westwanda Dr    2   3.0  2013
## 3         1009 N Beverly Dr    4   5.0  3476
## 4         1010 N Rexford Dr    5   6.5  7718
## 5      10101 Angelo View Dr    3   1.5  4000
## 6         1013 N Beverly Dr    7   8.0  6365
##                                                                      date
## 1                                      03/08/2014Hilton & HylandFeatured
## 2 12/11/2013Coldwell Banker Residential Brokerage - Sherman OaksFeatured
## 3                                              03/18/2014Hilton & Hyland
## 4                                             03/01/2014John Aaroe Group
## 5                                            03/24/2014Rodeo Realty Inc.
## 6                                              03/18/2014Hilton & Hyland
##      price          city type
## 1 18000000 Beverly Hills  sfh
## 2   950000 Beverly Hills  sfh
## 3  9750000 Beverly Hills  sfh
## 4 16500000 Beverly Hills  sfh
## 5 24250000 Beverly Hills  sfh
## 6 13450000 Beverly Hills  sfh
```

```
for(i in 1:255){
  if(LAdata$city[i]=="culver city") LAdata$city[i] <- "Culver City"
}
```

```
ggplot(LAdata, aes(x=beds, y=price, color = city, shape = type)) + geom_point(size = 2, fill = "white")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



This graphic answers the questions of whether or not City, Type, or Square Footage would be good practical predictors for determining the price of a Los Angeles home. It appears that based on these observations, city plays a role in price, with those in Beverly Hills being the most expensive, but also seems to have the largest number of homes overall. The relationship between type and price seems to be a little more unclear considering we can see that there are significantly more homes than condos.

Question 2:

a)

Using the cdc.csv data, make a plot that helps us understand the association between people's desired weight and their current weight, given their gender and whether or not they exercise. Your plot should include least squares lines to show the linear relation between desired weight and current weight for each of the four subgroups. Interpret these plots.

```
cdc <- read.csv("cdc.csv")
dim(cdc)
```

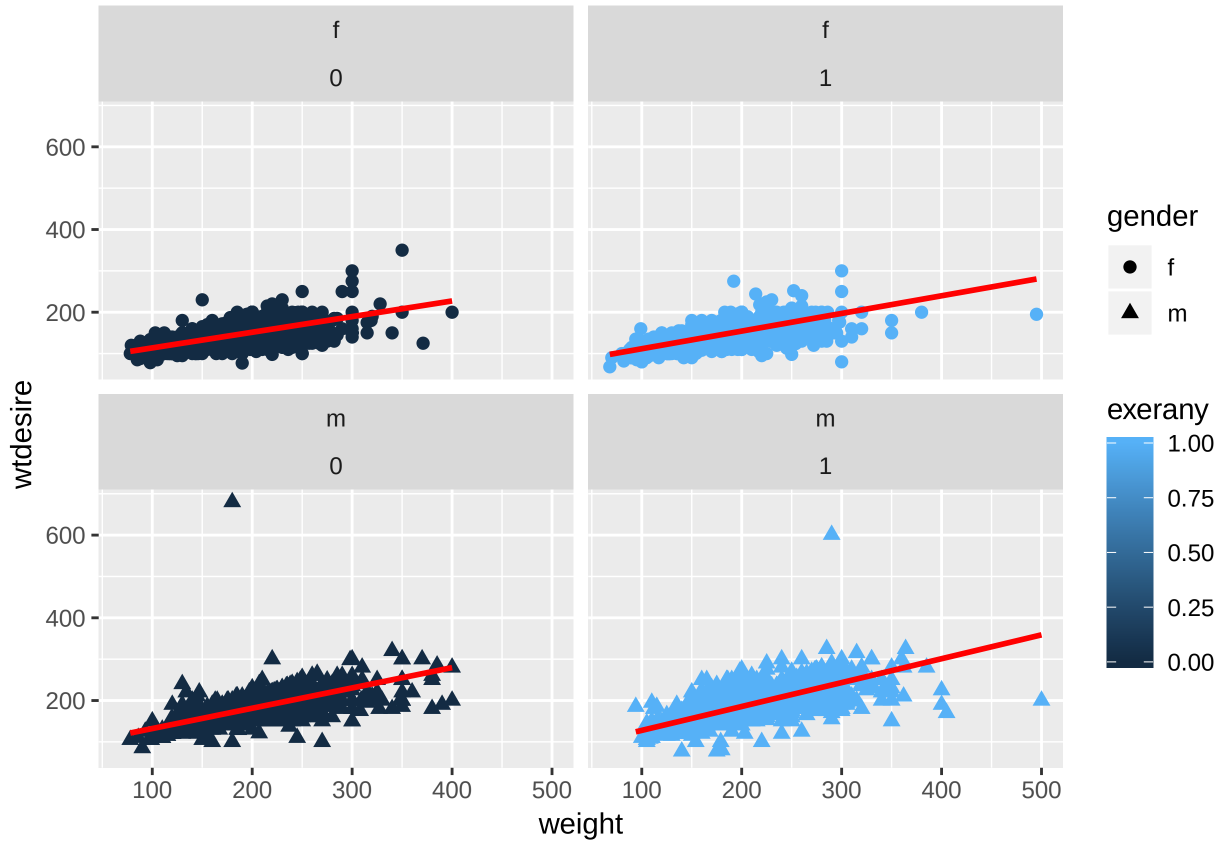
```
## [1] 20000    11
```

```
head(cdc)
```

```
##   state   genhlth physhlth exerany hlthplan smoke100 height weight
## 1    22      good        0       0        1        0     70    175
## 2    25      good       30       0        1        1     64    125
## 3     6      good        2       1        1        1     60    105
## 4     6      good        0       1        1        0     66    132
## 5    39 very good        0       0        1        0     61    150
```

```
## 6     42 very good         0        1         1         0      64     114
##   wtdesire age gender
## 1      175  77      m
## 2      115  33      f
## 3      105  49      f
## 4      124  42      f
## 5      130  55      f
## 6      114  55      f
```

```
ggplot(cdc, aes(x=weight, y=wtdesire, color = exerany, shape = gender)) + geom_point(size = 2) + facet_
```



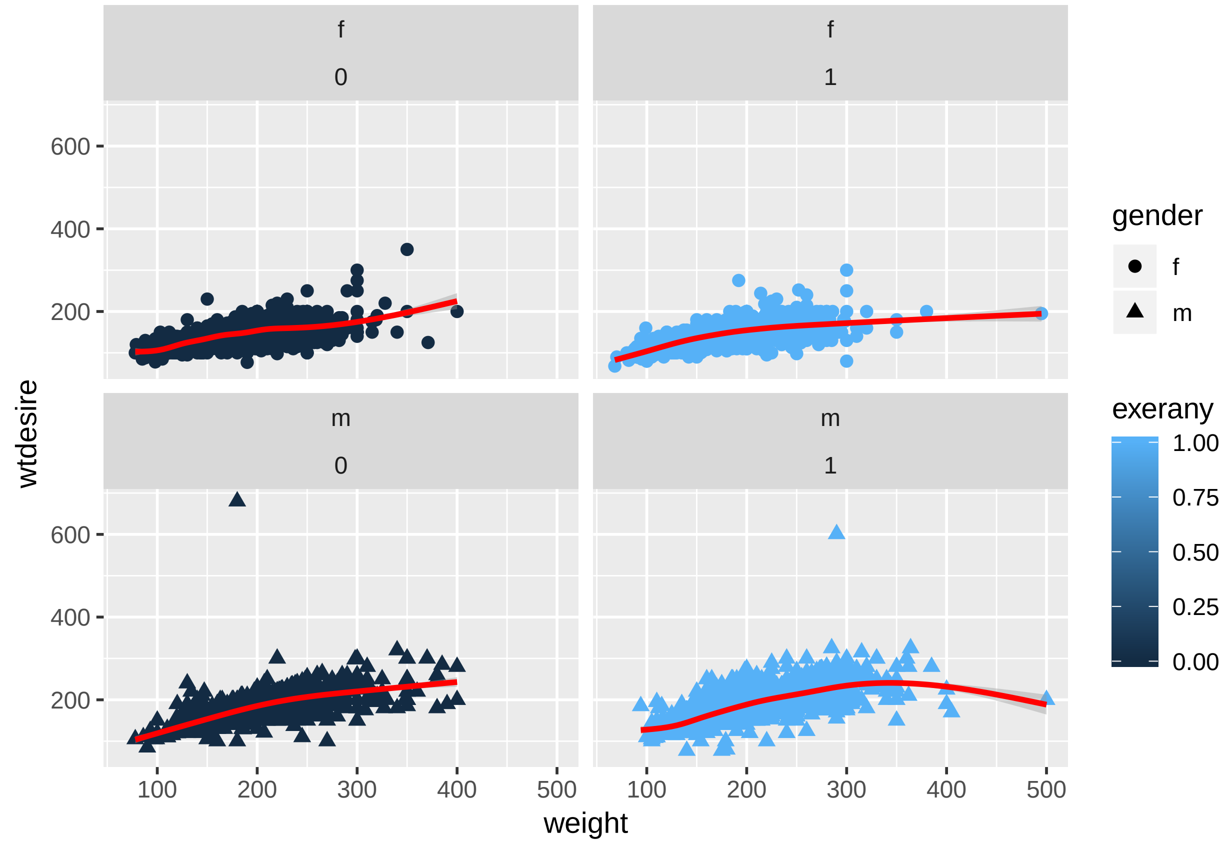
These plots show that regardless of gender or exercise, all four plots show similar slightly positive trends. All points also seem to be clustered around the same weights.

b)

Instead of a regression line, use a smoother. Explain how the results differ from (a). Note: You can learn more about it at http://www.cdc.gov/brfss. The data come from the Behavioral Risk Factor Survey System.

```
ggplot(cdc, aes(x=weight, y=wtdesire, color = exerany, shape = gender)) + geom_point(size = 2) + facet_
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



The lines are actually very similar despite the connections with outliers. The graph with males who exercise seems to change the most with a slight downward curve on the right side of the graph.

Question 3

The goal is to use both the Logistic Regression and KNN classification algorithms to classify bank notes based on the following features. i.e. Develop a rule to tell them apart.

First we must input the data and standardize the features.

```
webdata <- c(214.8,131,131.1,   9,  9.7,    141,    0,
             214.6, 129.7,  129.7,  8.1,    9.5,    141.7,  0,
             214.8, 129.7,  129.7,  8.7,    9.6,    142.2,  0,
             214.8, 129.7,  129.6,  7.5,    10.4,   142,    0,
             215,   129.6,  129.7,  10.4,   7.7,    141.8,  0,
             215.7, 130.8,  130.5,  9,  10.1,   141.4,  0,
             215.5, 129.5,  129.7,  7.9,    9.6,    141.6,  0,
             214.5, 129.6,  129.2,  7.2,    10.7,   141.7,  0,
             214.9, 129.4,  129.7,  8.2,    11, 141.9,  0,
             215.2, 130.4,  130.3,  9.2,    10, 140.7,  0,
             215.3, 130.4,  130.3,  7.9,    11.7,   141.8,  0,
             215.1, 129.5,  129.6,  7.7,    10.5,   142.2,  0,
             215.2, 130.8,  129.6,  7.9,    10.8,   141.4,  0,
             214.7, 129.7,  129.7,  7.7,    10.9,   141.7,  0,
             215.1, 129.9,  129.7,  7.7,    10.8,   141.8,  0,
             214.5, 129.8,  129.8,  9.3,    8.5,    141.6,  0,
             214.6, 129.9,  130.1,  8.2,    9.8,    141.7,  0,
```

```
              215,   129.9,  129.7,  9,  9,  141.9,  0,
              215.2, 129.6,  129.6,  7.4,    11.5,   141.5,  0,
              214.7, 130.2,  129.9,  8.6,    10, 141.9,  0,
              215,   129.9,  129.3,  8.4,    10, 141.4,  0,
              215.6, 130.5,  130,    8.1,    10.3,   141.6,  0,
              215.3, 130.6,  130,    8.4,    10.8,   141.5,  0,
              215.7, 130.2,  130,    8.7,    10, 141.6,  0,
              215.1, 129.7,  129.9,  7.4,    10.8,   141.1,  0,
              215.3, 130.4,  130.4,  8,  11, 142.3,  0,
              215.5, 130.2,  130.1,  8.9,    9.8,    142.4,  0,
              215.1, 130.3,  130.3,  9.8,    9.5,    141.9,  0,
              215.1, 130,    130,    7.4,    10.5,   141.8,  0,
              214.8, 129.7,  129.3,  8.3,    9,  142,    0,
              215.2, 130.1,  129.8,  7.9,    10.7,   141.8,  0,
              214.8, 129.7,  129.7,  8.6,    9.1,    142.3,  0,
              215,   130,    129.6,  7.7,    10.5,   140.7,  0,
              215.6, 130.4,  130.1,  8.4,    10.3,   141,    0,
              215.9, 130.4,  130,    8.9,    10.6,   141.4,  0,
              214.6, 130.2,  130.2,  9.4,    9.7,    141.8,  0,
              215.5, 130.3,  130,    8.4,    9.7,    141.8,  0,
              215.3, 129.9,  129.4,  7.9,    10, 142,    0,
              215.3, 130.3,  130.1,  8.5,    9.3,    142.1,  0,
              213.9, 130.3,  129,    8.1,    9.7,    141.3,  0,
              214.4, 129.8,  129.2,  8.9,    9.4,    142.3,  0,
214.8,  130.1,  129.6,  8.8,    9.9,    140.9,  0,
214.9,  129.6,  129.4,  9.3,    9,  141.7,  0
,214.9, 130.4,  129.7,  9,  9.8,    140.9,  0
,214.8, 129.4,  129.1,  8.2,    10.2,   141,    0
,214.3, 129.5,  129.4,  8.3,    10.2,   141.8,  0
,214.8, 129.9,  129.7,  8.3,    10.2,   141.5,  0
,214.8, 129.9,  129.7,  7.3,    10.9,   142,    0
,214.6, 129.7,  129.8,  7.9,    10.3,   141.1,  0
,214.5, 129,    129.6,  7.8,    9.8,    142,    0
,214.6, 129.8,  129.4,  7.2,    10, 141.3,  0
,215.3, 130.6,  130,    9.5,    9.7,    141.1,  0
,214.5, 130.1,  130,    7.8,    10.9,   140.9,  0
,215.4, 130.2,  130.2,  7.6,    10.9,   141.6,  0
,214.5, 129.4,  129.5,  7.9,    10, 141.4,  0
,215.2, 129.7,  129.4,  9.2,    9.4,    142,    0
,215.7, 130,    129.4,  9.2,    10.4,   141.2,  0
,215, 129.6,    129.4,  8.8,    9,  141.1,  0
,215.1, 130.1,  129.9,  7.9,    11, 141.3,  0
,215.1, 130,    129.8,  8.2,    10.3,   141.4,  0
,215.1, 129.6,  129.3,  8.3,    9.9,    141.6,  0
,215.3, 129.7,  129.4,  7.5,    10.5,   141.5,  0
,215.4, 129.8,  129.4,  8,  10.6,   141.5,  0
,214.5, 130,    129.5,  8,  10.8,   141.4,  0
,215,   130,    129.8,  8.6,    10.6,   141.5,  0
,215.2, 130.6,  130,    8.8,    10.6,   140.8,  0
,214.6, 129.5,  129.2,  7.7,    10.3,   141.3,  0
,214.8, 129.7,  129.3,  9.1,    9.5,    141.5,  0
,215.1, 129.6,  129.8,  8.6,    9.8,    141.8,  0
,214.9, 130.2,  130.2,  8,  11.2,   139.6,  0
```

```
,213.8, 129.8,  129.5,  8.4,    11.1,   140.9,  0
,215.2, 129.9,  129.5, 8.2, 10.3,   141.4,  0
,215,   129.6,  130.2,  8.7,    10, 141.2,  0
,214.4, 129.9,  129.6,  7.5,    10.5,   141.8,  0
,215.2, 129.9,  129.7,  7.2,    10.6,   142.1,  0
,214.1, 129.6,  129.3,  7.6,    10.7,   141.7,  0
,214.9, 129.9,  130.1,  8.8,    10, 141.2,  0
,214.6, 129.8,  129.4,  7.4,    10.6,   141,    0
,215.2, 130.5,  129.8,  7.9,    10.9,   140.9,  0
,214.6, 129.9,  129.4,  7.9,    10, 141.8,  0
,215.1, 129.7,  129.7,  8.6,    10.3,   140.6,  0
,214.9, 129.8,  129.6,  7.5,    10.3,   141,    0
,215.2, 129.7,  129.1,  9,  9.7,    141.9,  0
,215.2, 130.1,  129.9,  7.9,    10.8,   141.3,  0
,215.4, 130.7,  130.2,  9,  11.1,   141.2,  0
,215.1, 129.9,  129.6,  8.9,    10.2,   141.5,  0
,215.2, 129.9,  129.7,  8.7,    9.5,    141.6,  0
,215,   129.6,  129.2,  8.4,    10.2,   142.1,  0
,214.9, 130.3,  129.9,  7.4,    11.2,   141.5,  0
,215,   129.9,  129.7,  8,  10.5,   142,    0
,214.7, 129.7,  129.3,  8.6,    9.6,    141.6,  0
,215.4, 130,    129.9,  8.5,    9.7,    141.4,  0
,214.9, 129.4,  129.5,  8.2,    9.9,    141.5,  0
,214.5, 129.5,  129.3,  7.4,    10.7,   141.5,  0
,214.7, 129.6,  129.5,  8.3,    10, 142,    0
,215.6, 129.9,  129.9,  9,  9.5,    141.7,  0
,215,   130.4,  130.3,  9.1,    10.2,   141.1,  0
,214.4, 129.7,  129.5,  8,  10.3,   141.2,  0
,215.1, 130,    129.8,  9.1,    10.2,   141.5,  0
,214.7, 130,    129.4,  7.8,    10, 141.2,  0
,214.4, 130.1,  130.3,  9.7,    11.7,   139.8,  1
,214.9, 130.5,  130.2,  11, 11.5,   139.5,  1
,214.9, 130.3,  130.1,  8.7,    11.7,   140.2,  1
,215,   130.4,  130.6,  9.9,    10.9,   140.3,  1
,214.7, 130.2,  130.3,  11.8,   10.9,   139.7,  1
,215,   130.2,  130.2,  10.6,   10.7,   139.9,  1
,215.3, 130.3,  130.1,  9.3,    12.1,   140.2,  1
,214.8, 130.1,  130.4,  9.8,    11.5,   139.9,  1
,215,   130.2,  129.9,  10, 11.9,   139.4,  1
,215.2, 130.6,  130.8,  10.4,   11.2,   140.3,  1
,215.2, 130.4,  130.3,  8,  11.5,   139.2,  1
,215.1, 130.5,  130.3,  10.6,   11.5,   140.1,  1
,215.4, 130.7,  131.1,  9.7,    11.8,   140.6,  1
,214.9, 130.4,  129.9,  11.4,   11, 139.9,  1
,215.1, 130.3,  130,    10.6,   10.8,   139.7,  1
,215.5, 130.4,  130,    8.2,    11.2,   139.2,  1
,214.7, 130.6,  130.1,  11.8,   10.5,   139.8,  1
,214.7, 130.4,  130.1,  12.1,   10.4,   139.9,  1
,214.8, 130.5,  130.2,  11, 11, 140,    1
,214.4, 130.2,  129.9,  10.1,   12, 139.2,  1
,214.8, 130.3,  130.4,  10.1,   12.1,   139.6,  1
,215.1, 130.6,  130.3,  12.3,   10.2,   139.6,  1
,215.3, 130.8,  131.1,  11.6,   10.6,   140.2,  1
```

```
,215.1, 130.7,  130.4,  10.5,   11.2,   139.7,  1
,214.7, 130.5,  130.5,  9.9,    10.3,   140.1,  1
,214.9, 130,    130.3,  10.2,   11.4,   139.6,  1
,215,   130.4,  130.4,  9.4,    11.6,   140.2,  1
,215.5, 130.7,  130.3,  10.2,   11.8,   140,    1
,215.1, 130.2,  130.2,  10.1,   11.3,   140.3,  1
,214.5, 130.2,  130.6,  9.8,    12.1,   139.9,  1
,214.3, 130.2,  130,    10.7,   10.5,   139.8,  1
,214.5, 130.2,  129.8,  12.3,   11.2,   139.2,  1
,214.9, 130.5,  130.2,  10.6,   11.5,   139.9,  1
,214.6, 130.2,  130.4,  10.5,   11.8,   139.7,  1
,214.2, 130,    130.2,  11, 11.2,   139.5,  1
,214.8, 130.1,  130.1,  11.9,   11.1,   139.5,  1
,214.6, 129.8,  130.2,  10.7,   11.1,   139.4,  1
,214.9, 130.7,  130.3,  9.3,    11.2,   138.3,  1
,214.6, 130.4,  130.4,  11.3,   10.8,   139.8,  1
,214.5, 130.5,  130.2,  11.8,   10.2,   139.6,  1
,214.8, 130.2,  130.3,  10, 11.9,   139.3,  1
,214.7, 130,    129.4,  10.2,   11, 139.2,  1
,214.6, 130.2,  130.4,  11.2,   10.7,   139.9,  1
,215,   130.5,  130.4,  10.6,   11.1,   139.9,  1
,214.5, 129.8,  129.8,  11.4,   10, 139.3,  1
,214.9, 130.6,  130.4,  11.9,   10.5,   139.8,  1
,215,   130.5,  130.4,  11.4,   10.7,   139.9,  1
,215.3, 130.6,  130.3,  9.3,    11.3,   138.1,  1
,214.7, 130.2,  130.1,  10.7,   11, 139.4,  1
,214.9, 129.9,  130,    9.9,    12.3,   139.4,  1
,214.9, 130.3,  129.9,  11.9,   10.6,   139.8,  1
,214.6, 129.9,  129.7,  11.9,   10.1,   139,    1
,214.6, 129.7,  129.3,  10.4,   11, 139.3,  1
,214.5, 130.1,  130.1,  12.1,   10.3,   139.4,  1
,214.5, 130.3,  130,    11, 11.5,   139.5,  1
,215.1, 130,    130.3,  11.6,   10.5,   139.7,  1
,214.2, 129.7,  129.6,  10.3,   11.4,   139.5,  1
,214.4, 130.1,  130,    11.3,   10.7,   139.2,  1
,214.8, 130.4,  130.6,  12.5,   10, 139.3,  1
,214.6, 130.6,  130.1,  8.1,    12.1,   137.9,  1
,215.6, 130.1,  129.7,  7.4,    12.2,   138.4,  1
,214.9, 130.5,  130.1,  9.9,    10.2,   138.1,  1
,214.6, 130.1,  130,    11.5,   10.6,   139.5,  1
,214.7, 130.1,  130.2,  11.6,   10.9,   139.1,  1
,214.3, 130.3,  130,    11.4,   10.5,   139.8,  1
,215.1, 130.3,  130.6,  10.3,   12, 139.7,  1
,216.3, 130.7,  130.4,  10, 10.1,   138.8,  1
,215.6, 130.4,  130.1,  9.6,    11.2,   138.6,  1
,214.8, 129.9,  129.8,  9.6,    12, 139.6,  1
,214.9, 130,    129.9,  11.4,   10.9,   139.7,  1
,213.9, 130.7,  130.5,  8.7,    11.5,   137.8,  1
,214.2, 130.6,  130.4,  12, 10.2,   139.6,  1
,214.8, 130.5,  130.3,  11.8,   10.5,   139.4,  1
,214.8, 129.6,  130,    10.4,   11.6,   139.2,  1
,214.8, 130.1,  130,    11.4,   10.5,   139.6,  1
,214.9, 130.4,  130.2,  11.9,   10.7,   139,    1
```

```
,214.3, 130.1,  130.1,  11.6,   10.5,   139.7,  1
,214.5, 130.4,  130,    9.9,    12, 139.6,  1
,214.8, 130.5,  130.3,  10.2,   12.1,   139.1,  1
,214.5, 130.2,  130.4,  8.2,    11.8,   137.8,  1
,215,   130.4,  130.1,  11.4,   10.7,   139.1,  1
,214.8, 130.6,  130.6,  8,  11.4,   138.7,  1
,215,   130.5,  130.1,  11, 11.4,   139.3,  1
,214.6, 130.5,  130.4,  10.1,   11.4,   139.3,  1
,214.7, 130.2,  130.1,  10.7,   11.1,   139.5,  1
,214.7, 130.4,  130,    11.5,   10.7,   139.4,  1
,214.5, 130.4,  130,    8,  12.2,   138.5,  1
,214.8, 130,    129.7,  11.4,   10.6,   139.2,  1
,214.8, 129.9,  130.2,  9.6,    11.9,   139.4,  1
,214.6, 130.3,  130.2,  12.7,   9.1,    139.2,  1
,215.1, 130.2,  129.8,  10.2,   12, 139.4,  1
,215.4, 130.5,  130.6,  8.8,    11, 138.6,  1
,214.7, 130.3,  130.2,  10.8,   11.1,   139.2,  1
,215,   130.5,  130.3,  9.6,    11, 138.5,  1
,214.9, 130.3,  130.5,  11.6,   10.6,   139.8,  1
,215,   130.4,  130.3,  9.9,    12.1,   139.6,  1
,215.1, 130.3,  129.9,  10.3,   11.5,   139.7,  1
,214.8, 130.3,  130.4,  10.6,   11.1,   140,    1
,214.7, 130.7,  130.8,  11.2,   11.2,   139.4,  1
,214.3, 129.9,  129.9,  10.2,   11.5,   139.6,  1)

lengthdata <- c(webdata[seq(1,1400, by = 7)])
leftdata <- c(webdata[seq(2,1400, by = 7)])
rightdata <- c(webdata[seq(3,1400, by = 7)])
bottomdata <- c(webdata[seq(4,1400, by = 7)])
topdata <- c(webdata[seq(5,1400, by = 7)])
diagonaldata <- c(webdata[seq(6,1400, by = 7)])
ydata <- factor(c(webdata[seq(7,1400, by = 7)]))

banknote <- data.frame(lengthdata,leftdata,rightdata,bottomdata,topdata,diagonaldata,ydata)
head(banknote)
```

```
##   lengthdata leftdata rightdata bottomdata topdata diagonaldata ydata
## 1      214.8    131.0     131.1        9.0     9.7        141.0     0
## 2      214.6    129.7     129.7        8.1     9.5        141.7     0
## 3      214.8    129.7     129.7        8.7     9.6        142.2     0
## 4      214.8    129.7     129.6        7.5    10.4        142.0     0
## 5      215.0    129.6     129.7       10.4     7.7        141.8     0
## 6      215.7    130.8     130.5        9.0    10.1        141.4     0
```

```
library(class)
newlength <- scale(lengthdata)
newleft <- scale(leftdata)
newright <- scale(rightdata)
newbottom <- scale(bottomdata)
newtop <- scale(topdata)
newdiagonal <- scale(diagonaldata)

banknote <- data.frame(newlength,newleft,newright,newbottom,newtop,newdiagonal,ydata)
```

Then we split the data into training and testing.

```r
set.seed(1975397)
trainsamp <- sample(1:200, 200*0.7)
banktraining <- banknote[trainsamp,]
banktest <- banknote[-trainsamp,]
```

Then we will use the knn algorithm with k = 1, 3, and 7, also showing the confusion matrix for each.

```r
output1 <- knn(banktraining[1:6],banktest[1:6],banktraining$ydata, k = 1)
mean(output1!=banktest$ydata)
```

```
## [1] 0.01666667
```

```r
table(output1,banktest$ydata)
```

```
##
## output1  0  1
##       0 30  1
##       1  0 29
```

```r
output3 <- knn(banktraining[1:6],banktest[1:6],banktraining$ydata, k = 3)
mean(output3!=banktest$ydata)
```

```
## [1] 0.01666667
```

```r
table(output3,banktest$ydata)
```

```
##
## output3  0  1
##       0 30  1
##       1  0 29
```

```r
output7 <- knn(banktraining[1:6],banktest[1:6],banktraining$ydata, k = 7)
mean(output7!=banktest$ydata)
```

```
## [1] 0.01666667
```

```r
table(output7,banktest$ydata)
```

```
##
## output7  0  1
##       0 30  1
##       1  0 29
```

Use of logistic regression instead:

```r
attach(banknote)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     newbottom, newdiagonal, newleft, newlength, newright, newtop,
##     ydata
```

```r
bankmod <- glm(ydata~lengthdata+leftdata+rightdata+bottomdata+topdata+diagonaldata, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```r
predictbank <- predict(bankmod, banknote[,c(1:6)], type = 'response')
predictbank <- predictbank > 0.5
table(predictbank,ydata)
```

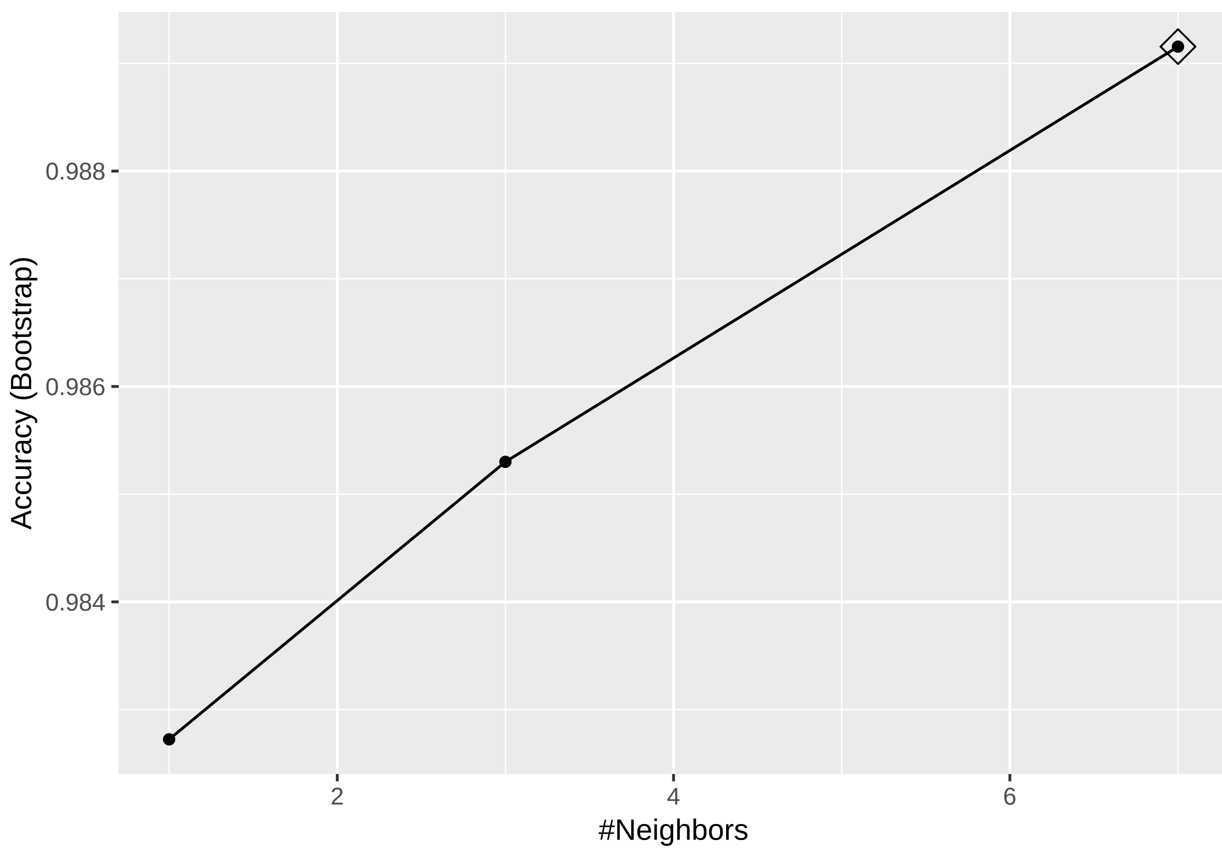
```
##             ydata
## predictbank   0   1
##       FALSE 100   0
##       TRUE    0 100
```

My error rates for each k in the knn approach were about 1.67%, however it is 0% for the logistic regression.

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
train_knn <- train(ydata ~ ., method = "knn", data = banktraining, tuneGrid = data.frame(k=c(1,3,7)))
ggplot(train_knn, highlight = TRUE)
```



k = 7 is the best k with the lowest RMSE.

Question 4

a)

Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

```r
library(ISLR)
median(Auto$mpg)
```

```
## [1] 22.75
```

```r
mpg01 <- c()
for(i in 1:392){
  if(Auto$mpg[i] > median(Auto$mpg)){
    mpg01[i] <- 1
  }
  if(Auto$mpg[i] < median(Auto$mpg)){
    mpg01[i] <- 0
  }
}
newAuto <- data.frame(Auto, mpg01)
head(newAuto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name mpg01
## 1 chevrolet chevelle malibu     0
## 2         buick skylark 320     0
## 3        plymouth satellite     0
## 4             amc rebel sst     0
## 5               ford torino     0
## 6          ford galaxie 500     0
```

b)

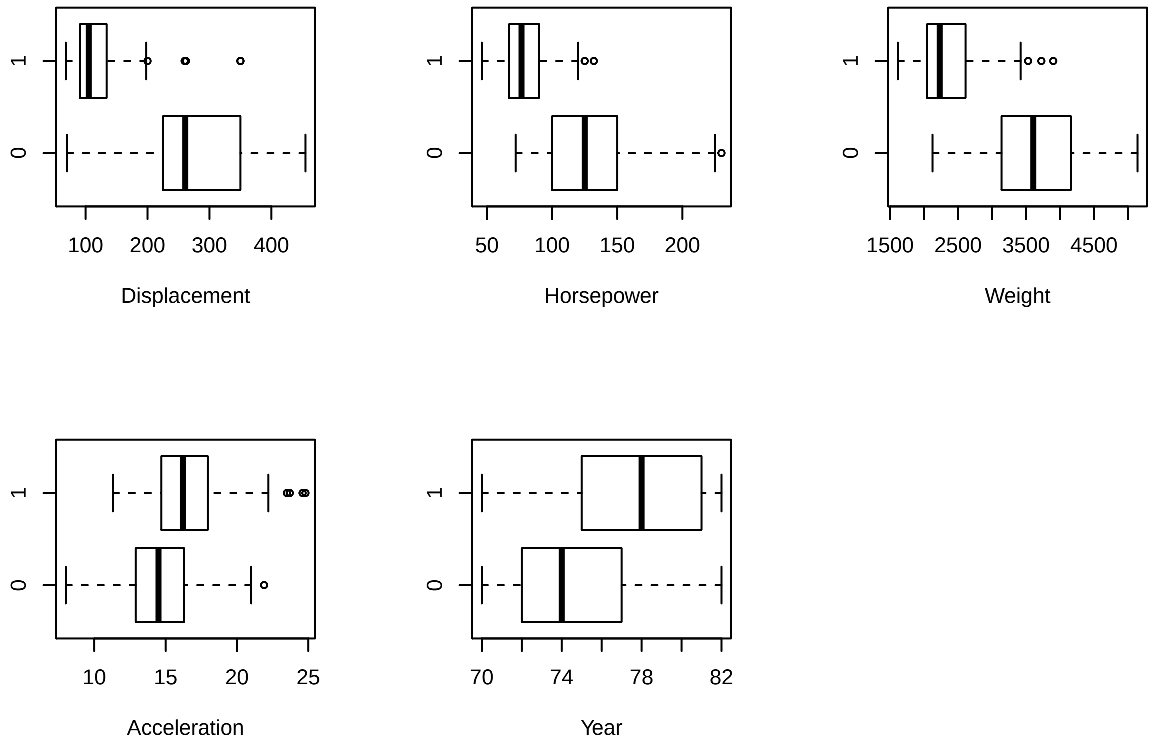
Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```r
attach(newAuto)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     mpg01
```

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

```r
##plot(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year + origin)
par(mfrow = c(2:3))
boxplot(displacement ~ mpg01, xlab = "Displacement", horizontal = T)
boxplot(horsepower ~ mpg01, xlab = "Horsepower", horizontal = T)
boxplot(weight ~ mpg01, xlab = "Weight", horizontal = T)
boxplot(acceleration ~ mpg01, xlab = "Acceleration", horizontal = T)
boxplot(year ~ mpg01, xlab = "Year", horizontal = T)
```

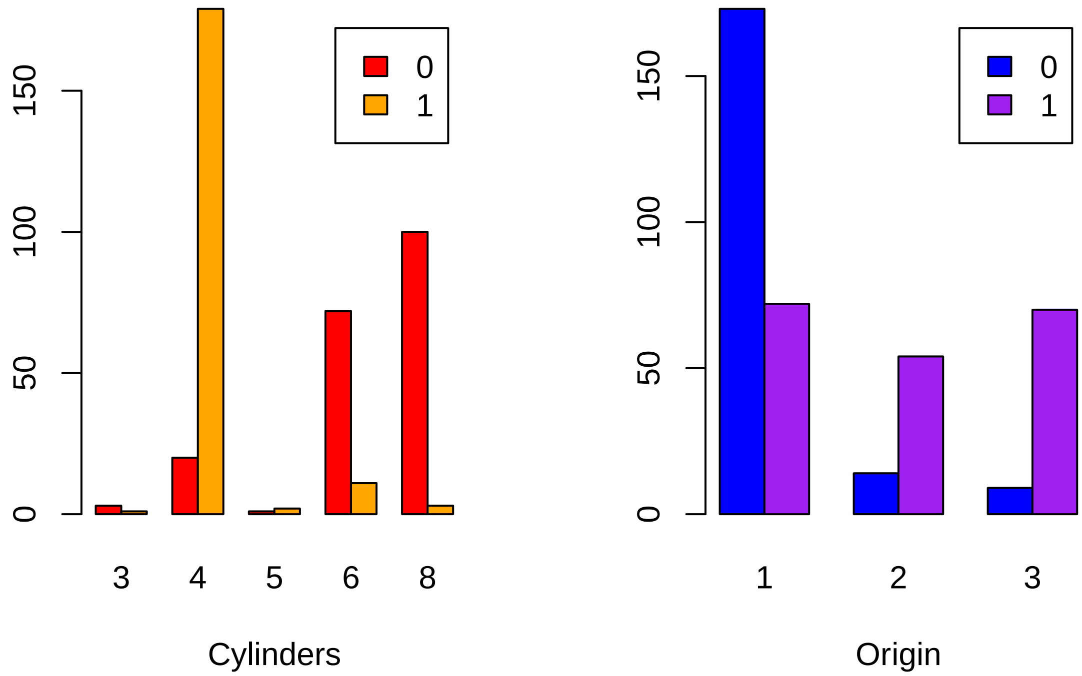


It appears that horsepower and weight have the most association with mpg01, and I will also include displacement and year because it appears that they may be an association there as well.

I believe that barplots would best represent the discrete variables.

```
par(mfrow = c(1:2))
cylcounts <- table(newAuto$mpg01, newAuto$cylinders)
barplot(cylcounts, xlab = "Cylinders", beside = TRUE, legend = rownames(cylcounts), col = c("red","orang

origincounts <- table(newAuto$mpg01, newAuto$origin)
barplot(origincounts, xlab = "Origin", beside = TRUE, legend = rownames(origincounts), col = c("blue","
```



Neither of the discrete variables seem to show strong evidence of association.

c)

Split the data into a training set and a test set. (70% and 30% respectively) Set.seed(1975397)

```
set.seed(1975397)
trainsamp <- sample(1:392, 392*0.7)
Autotraining <- newAuto[trainsamp,]
Autotest <- newAuto[-trainsamp,]
```

d)

Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
m1 <- glm(mpg01~horsepower+weight+displacement, family = binomial)
summary(m1)
```

```
##
## Call:
## glm(formula = mpg01 ~ horsepower + weight + displacement, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4385  -0.1920   0.0473   0.3565   3.3805
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  11.776685   1.624216   7.251 4.15e-13 ***
## horsepower   -0.042257   0.013571  -3.114  0.00185 **
## weight       -0.001943   0.000689  -2.821  0.00479 **
## displacement -0.013217   0.005304  -2.492  0.01271 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 543.43  on 391  degrees of freedom
## Residual deviance: 207.27  on 388  degrees of freedom
## AIC: 215.27
##
## Number of Fisher Scoring iterations: 7
```

```
predictmpg <- predict(m1, newAuto[,c(2:9)], type = 'response')
predictmpg <- predictmpg > 0.5
table(predictmpg,factor(newAuto$mpg01))
```

```
##
## predictmpg   0   1
##      FALSE 172  16
##      TRUE   24 180
```

```
testerror <- (24+16)/(172+16+24+180)
testerror
```

```
## [1] 0.1020408
```

The test error rate of my model is around 10.2%

e)

Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

```
knn_means <- numeric()

for(i in 1:10){
  output <- knn(Autotraining[3:5],Autotest[3:5],Autotraining$mpg01,k=i)
  knn_means[i] <- mean(output!=Autotest$mpg01)
  #table(output,Autotest$mpg01)
}

#Test errors
knn_means
```

```
##  [1] 0.1525424 0.1610169 0.1186441 0.1101695 0.1016949 0.1271186 0.1271186
##  [8] 0.1186441 0.1355932 0.1271186
```

```
#k = 5 is the lowest/best

bestoutput <- knn(Autotraining[3:5],Autotest[3:5],Autotraining$mpg01,k=5)
mean(bestoutput!=Autotest$mpg01)
```

```
## [1] 0.1016949
```

```r
table(bestoutput,Autotest$mpg01)
```

```
##
## bestoutput  0  1
##          0 57  5
##          1  7 49
```

K = 5 was the best value to perform on this dataset.