Business Analytics :: PS2

Tongxin Zhu

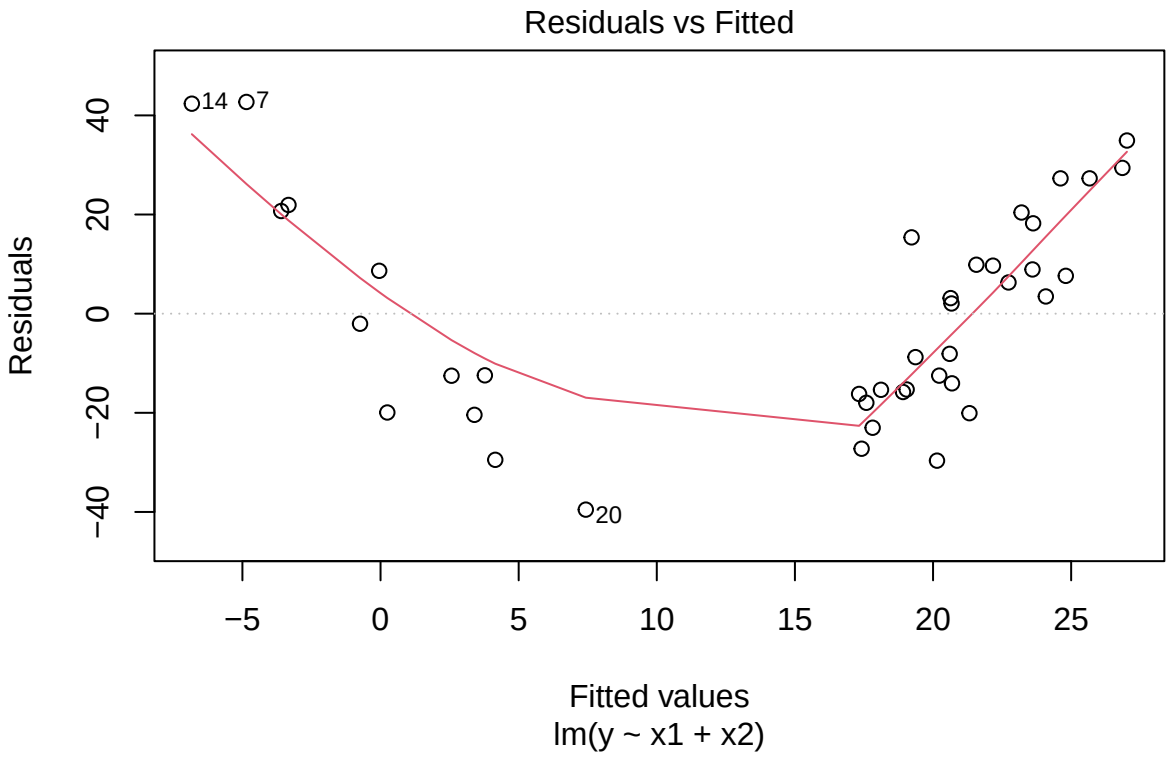
1. Linear Regression 1

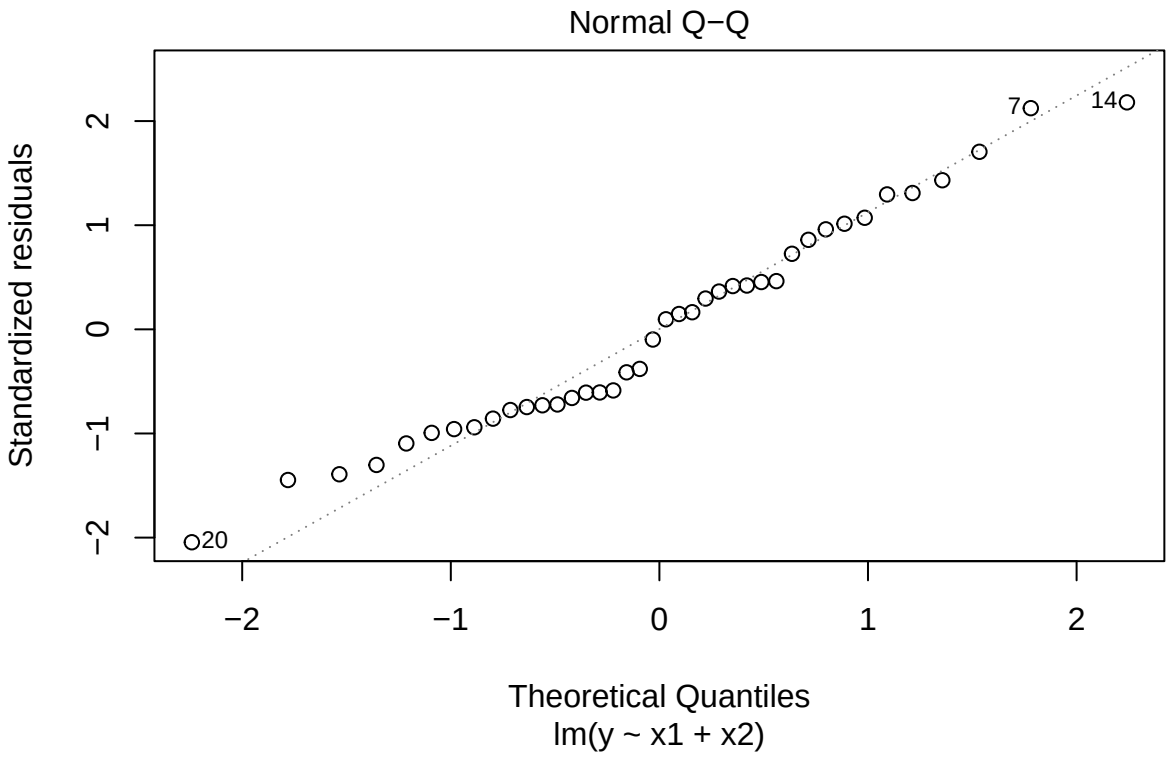
```
knitr::opts_chunk$set(error = TRUE)
# Read data
data1 <- read.csv("~/Desktop/PS2_EX1_Data Set.csv")
# Fit the linear regression model for first forty rows
fit1.1 <- lm(y ~ x1 + x2, data = data1[1:40, ])
# Summarize the model
summary(fit1.1)
```

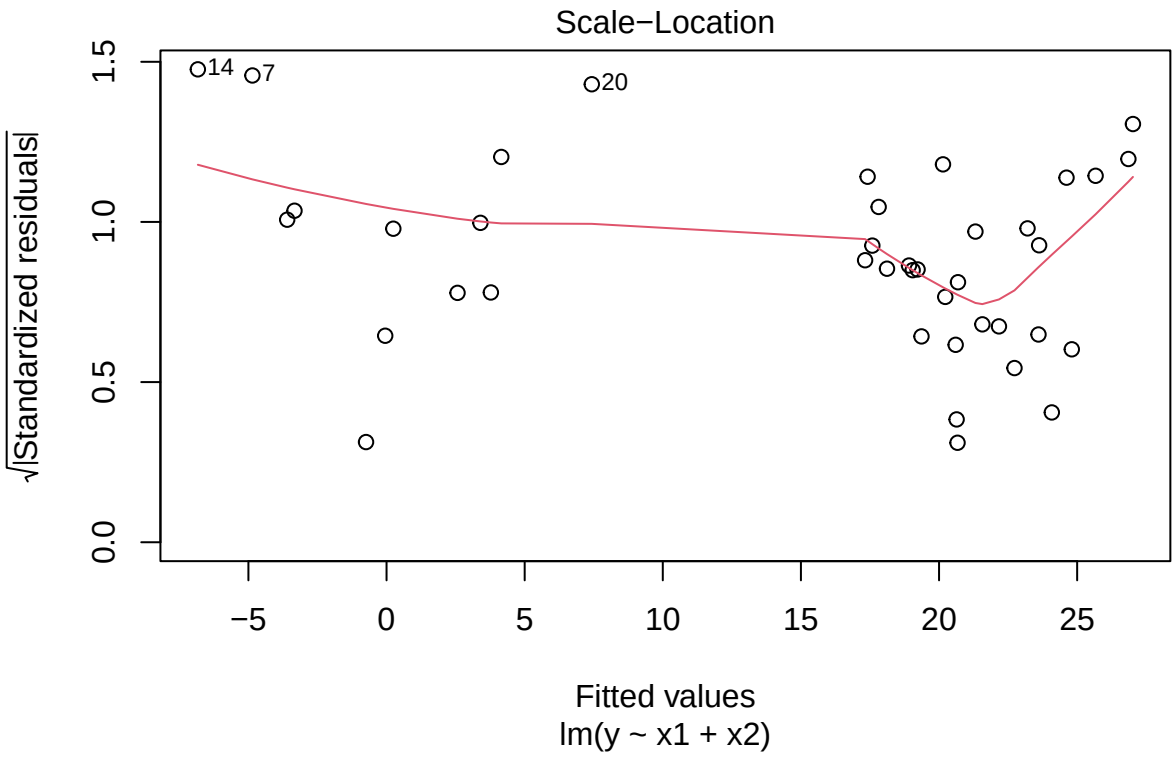
1.1

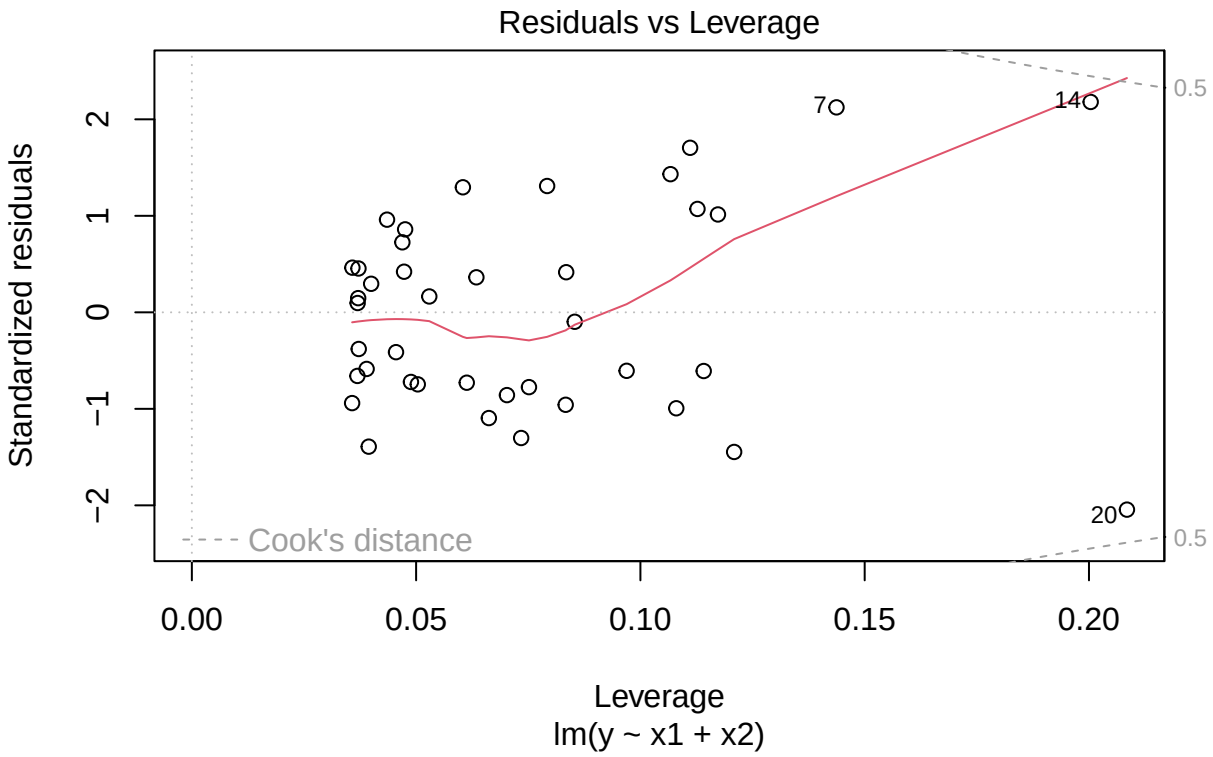
```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data1[1:40, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.529 -15.903   0.012  16.100  42.707
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  21.0846     4.1200   5.118  9.8e-06 ***
## x1            0.8601     0.9121   0.943  0.35179
## x2TRUE      -21.2042     7.4982  -2.828  0.00752 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.73 on 37 degrees of freedom
## Multiple R-squared:  0.1938, Adjusted R-squared:  0.1502
## F-statistic: 4.446 on 2 and 37 DF,  p-value: 0.0186
```

```
# Plot the model
plot(fit1.1)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x1 + x2)
14
7
20
40
20
0
−20
−40
−5
0
5
10
15
20
25


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x1 + x2)
20
7
14
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
1.5
1.0
0.5
0.0
14
7
20
−5
0
5
10
15
20
25
Fitted values
lm(y ~ x1 + x2)



Based on the summary of fit1, we can conclude that x2 is statistical significant for predicting y value since the standard error value of x2 is small in comparison to the coefficient, and the p-value of x2 is smaller than 0.05. The adjusted R-squared value is about 15% which means the model explains 15% of the variability of the response data around its mean which is relative low, so our model is not fitting the data very well.

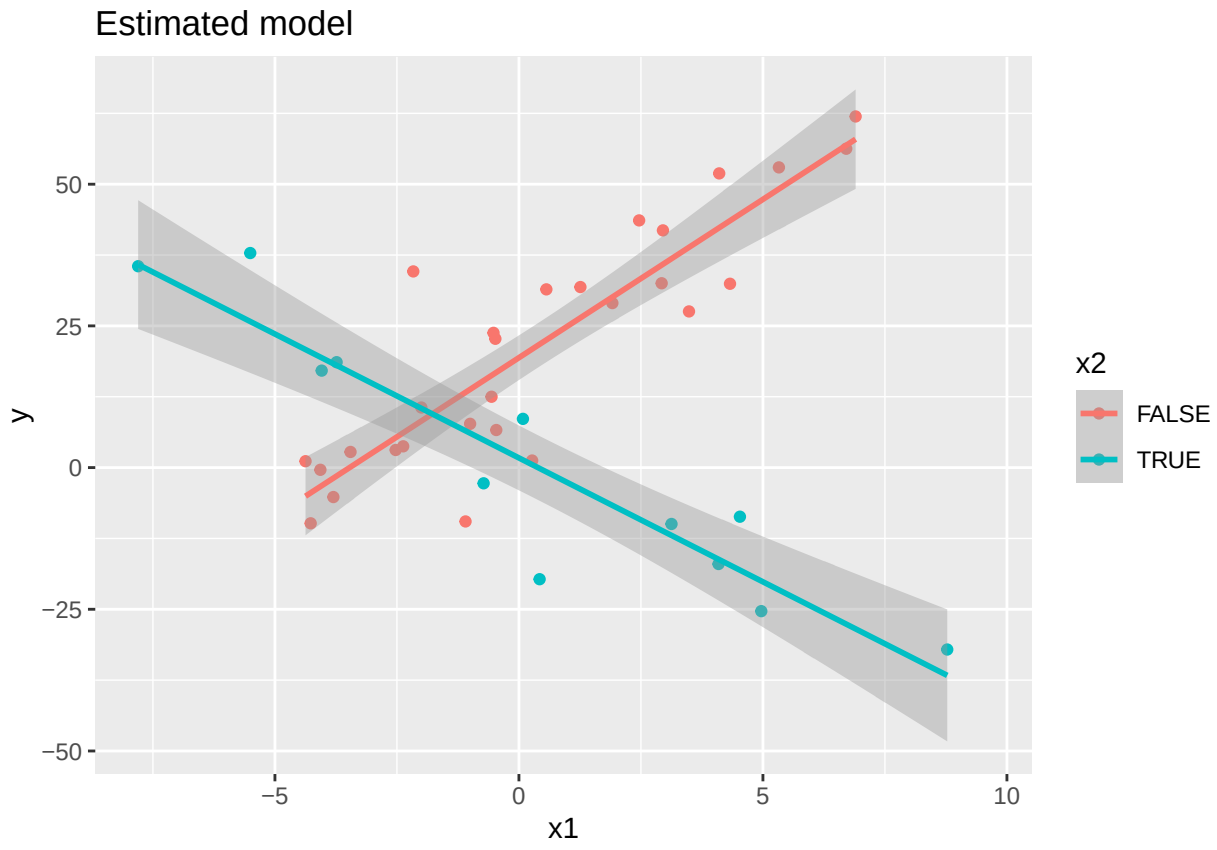
```
# update.packages("ggplot2")
# install.packages("scales")
library(ggplot2)
# Plot the estimated model
p1.2 <- ggplot(data1, aes(x1, y, color = x2)) +
  geom_point() +
  geom_smooth(method = "lm")
# Add title and labels
p1.2 + labs(title="Estimated model",
        x ="x1", y = "y")
```

1.2

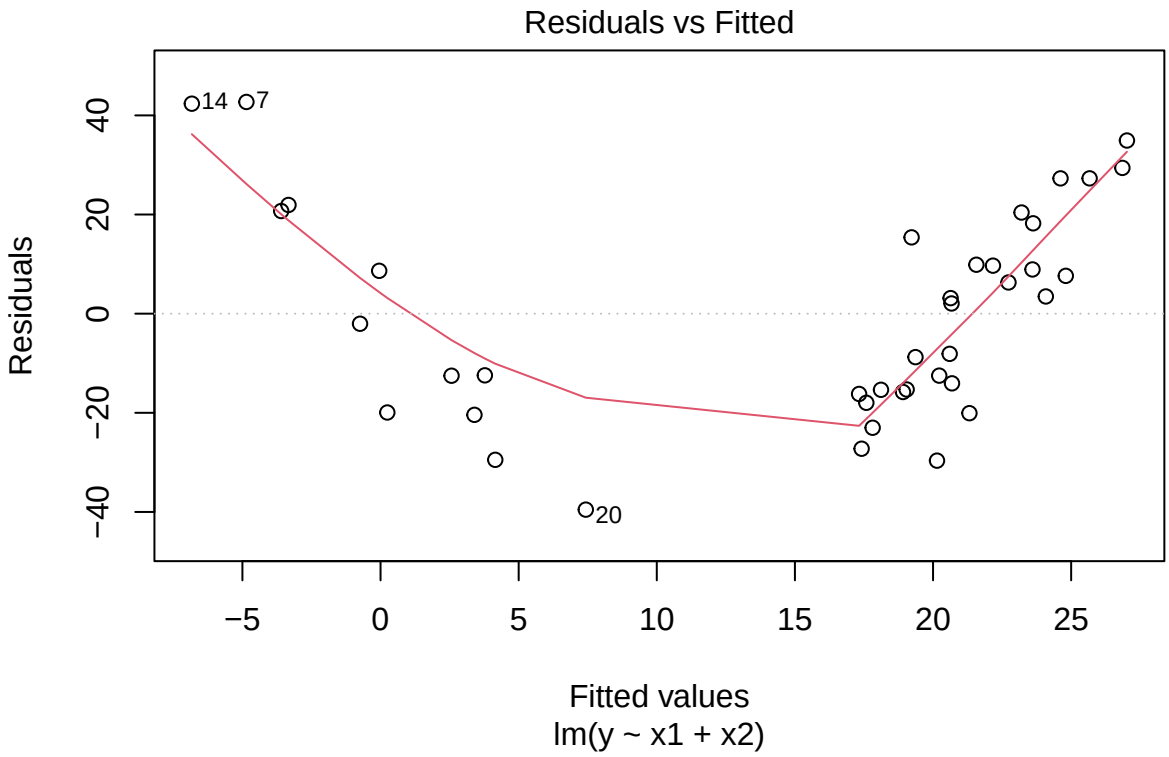
```
## `geom_smooth()` using formula = 'y ~ x'
```

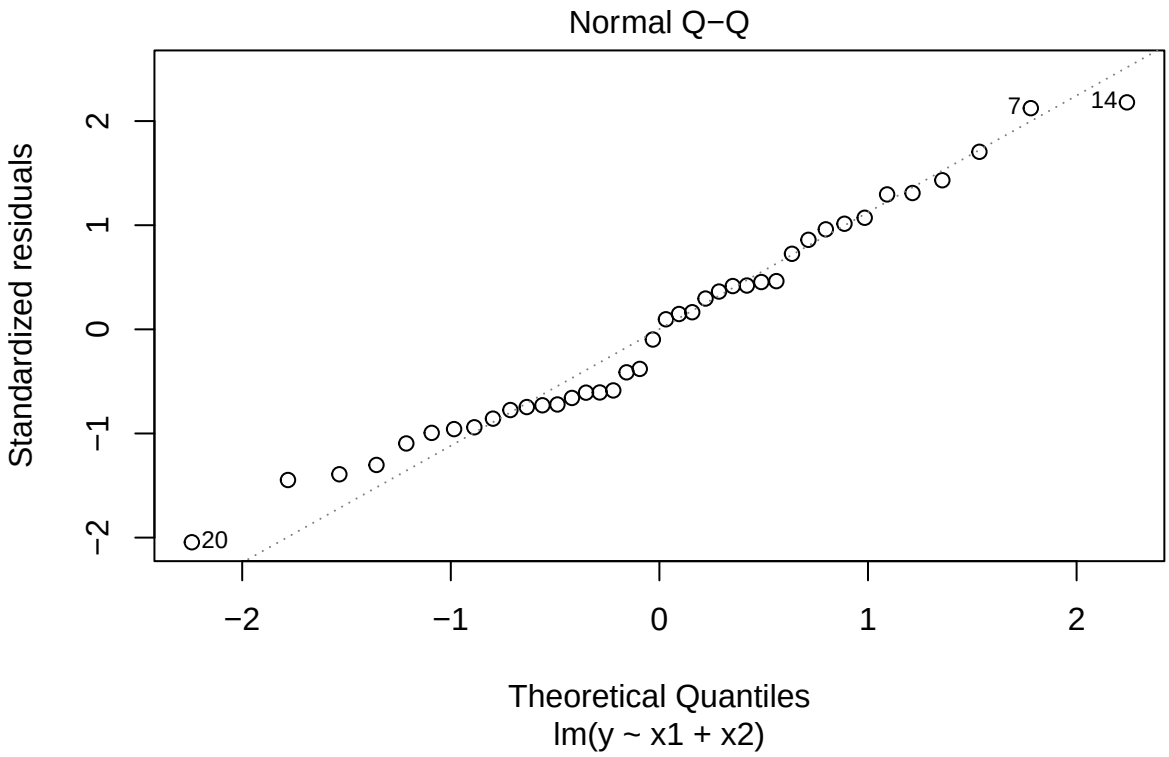
```
## Warning: Removed 20 rows containing non-finite values (‘stat_smooth()‘).
```

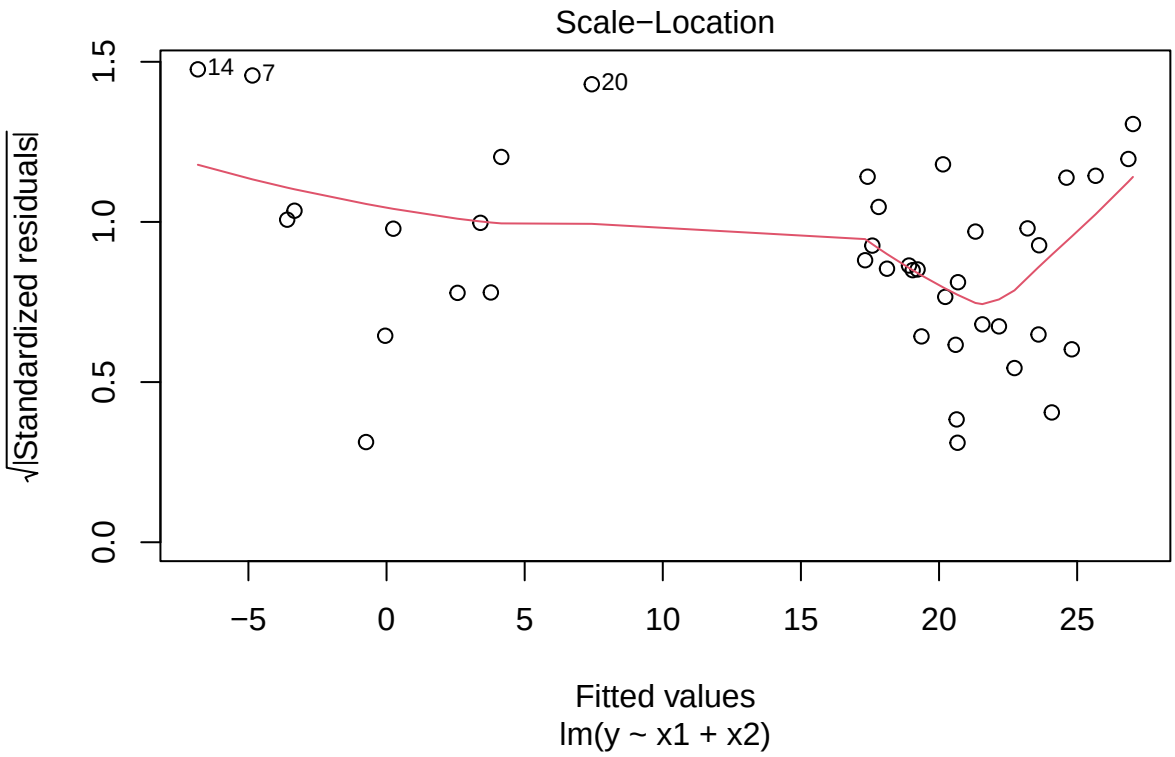
```
## Warning: Removed 20 rows containing missing values (‘geom_point()‘).
```

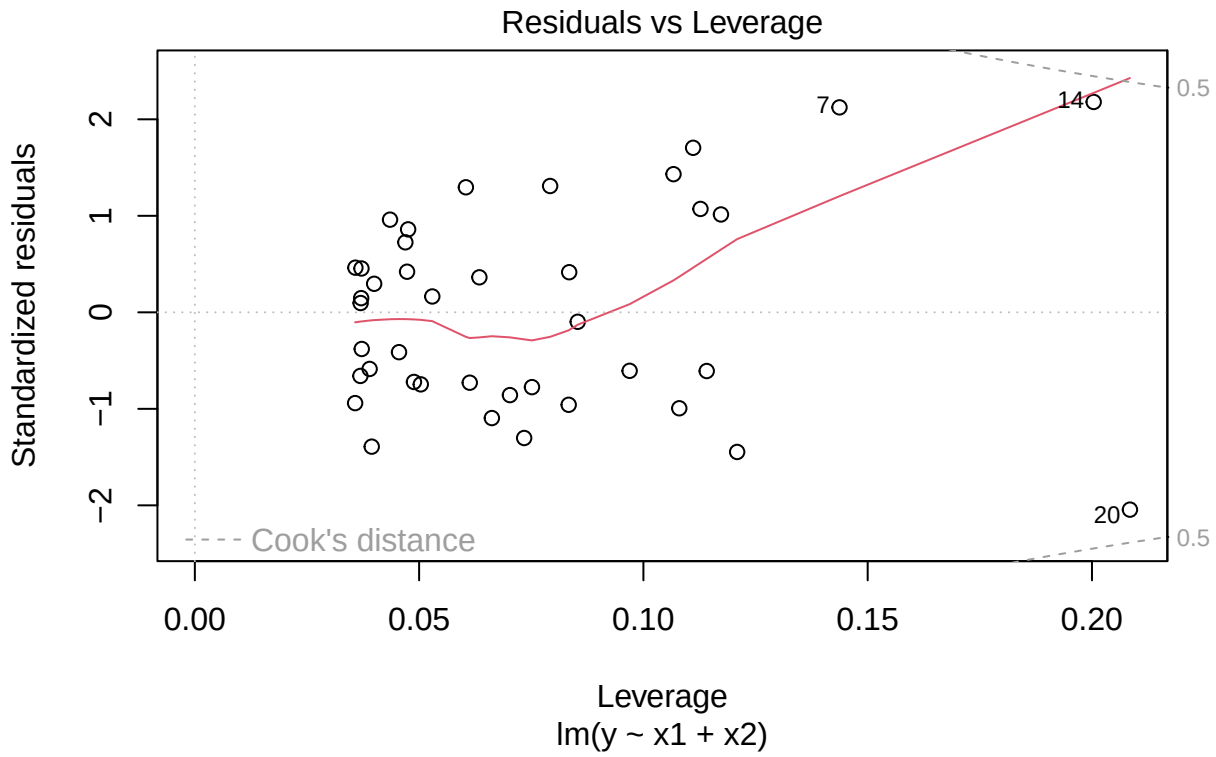



```
# Plot the model
plot(fit1.1)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x1 + x2)
14
7
20
40
20
0
−20
−40
−5
0
5
10
15
20
25


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x1 + x2)
7
14
20
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
1.5
1.0
0.5
0.0
14
7
20
−5
0
5
10
15
20
25
Fitted values
lm(y ~ x1 + x2)




```
# Predict response variables of rest rows
prediction <- predict(fit1.1, data1[41:60, ])
prediction
```

1.3

```
##        41        42        43        44        45        46        47        48
## -5.622415 21.059512  2.027875 -6.836206  0.509351 25.046804  8.175549 -6.318698
##        49        50        51        52        53        54        55        56
## 18.529359  3.536420 22.372925 14.559949 25.247372 -4.813607 -3.247319 -1.656302
##        57        58        59        60
## 20.234041 -3.699678  3.752380 19.840260
```

I am not confident about these results because the R squared value we got from the first step is not high which means this model does not fit our dataset very well.

2. Linear Regression 2

```r
# Simulate two variables randomly
set.seed(100)
var1 <- rnorm(1000, 1, 2)
```

2.1

```r
var2 <- rnorm(1000, 1, 2)
```

2.2

```r
# Fit the linear regression model
fit2.3 <- lm(var1 ~ var2)
summary(fit2.3)
```

2.3

```
##
## Call:
## lm(formula = var1 ~ var2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6569 -1.3356  0.0298  1.3904  6.6282
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.01868    0.07332  13.894   <2e-16 ***
## var2         0.01480    0.03325   0.445    0.656
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.062 on 998 degrees of freedom
## Multiple R-squared:  0.0001986,  Adjusted R-squared:  -0.0008032
## F-statistic: 0.1982 on 1 and 998 DF,  p-value: 0.6562
```

2.4 Based on the summary from 2.2, we can conclude that var2 is not statistical significant for predicting var1 since the p-value of var2 is not less than 0.05, the 5% significance level. Moreover, The standard error value is even larger than the estimate value. The multiple R-squared value is super small which means this model does not fit the dataset well.

2.5 The multiple R-squared value measures the strength of the relationship between the model and the dependent variable on a convenient 0-100% scale. For example, if R-squared value is 100%, the variation of dependent value can be explained by the variation of independent variable completely. So, based on the results we got from last step, we can conclude that 0.019% variation the model can explain.

3. Linear Regression 3

```
# The Manhattan Housing data contains missing values indicated by "0"
# so we are using the argument na.strings="0" to treat them as missing values.
# Read dataset
data2 <- read.csv("~/Desktop/Rolling_Sales_Manhattan_Data Set_ver4.csv")
head(data2)
```

3.1

```
##                Neighborhood TotalUnits LandSqFt GrossSqFt YearBuilt TaxClass
## 1 MIDTOWN WEST                       2     7532    112850      2007  Class 4
## 2 MIDTOWN WEST                       2     7532    112850      2007  Class 4
## 3 UPPER EAST SIDE (59-79)            0        0         0      1955  Class 2
## 4 MIDTOWN WEST                       1        0         0      1950  Class 2
## 5 MIDTOWN WEST                       2     7532    112850      2007  Class 4
## 6 HARLEM-CENTRAL                     0        0         0      1920  Class 2
##   SalePrice
## 1    100100
## 2    100780
## 3    101400
## 4    101825
## 5    102971
## 6    103000
```

```
# Check the structure of dataset
dim(data2)
```

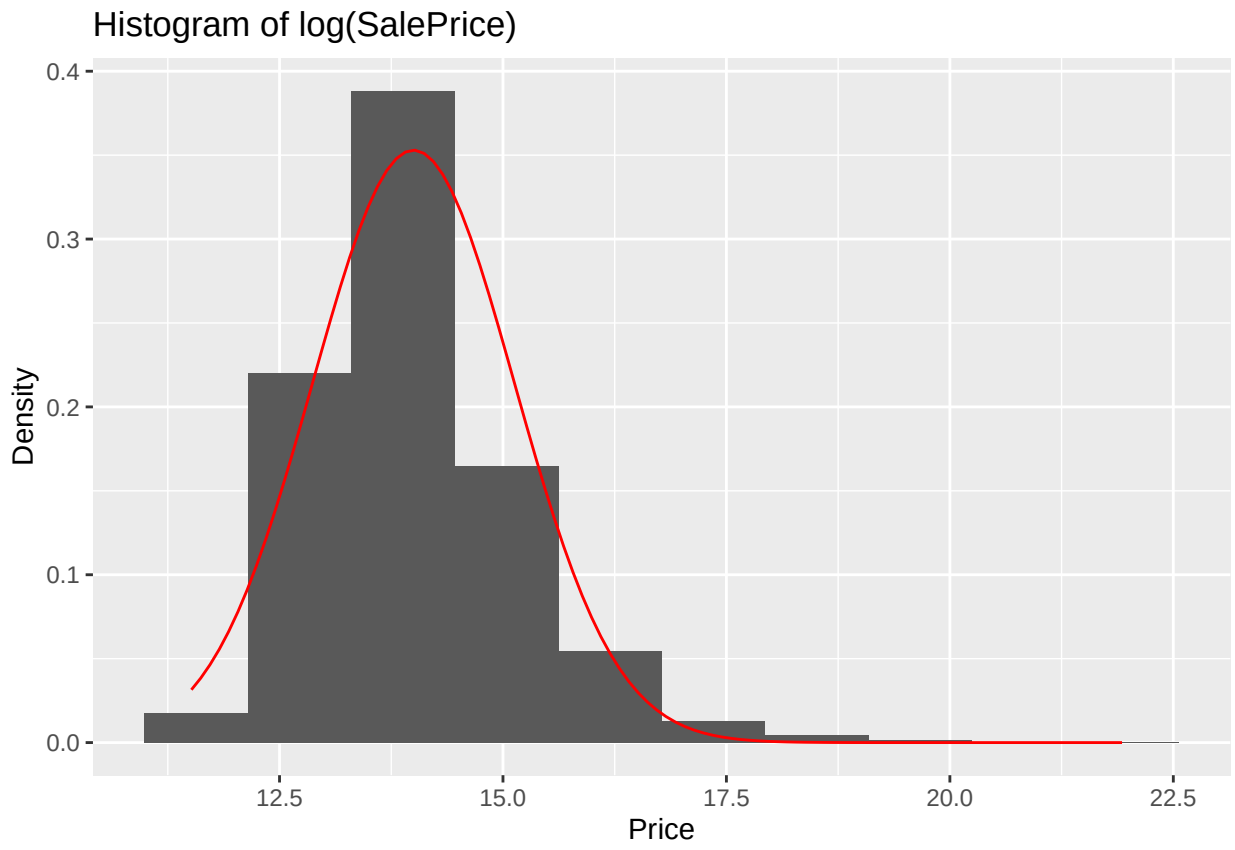
```
## [1] 15156     7
```

The dataset has 15156 rows and 7 columns. The dataset lists properties that sold in last twelve month period in Manhattan for tax class 1, 2, 3, and 4. Each row represents the property has been sold, and each column represents the neighborhood name, total unit number, land area, total area, year the property built, tax class based on the use of property, and price paid for the property.

```r
# Make SalePrice looks more normal distributed
data2$newSalePrice <- log(data2$SalePrice)
# Plot histogram of SalePrice to check the distribution
q <- ggplot(data2, aes(newSalePrice)) +
  geom_histogram(aes(y = ..density..), bins = 10) +
  geom_function(fun = dnorm, args = list(mean = mean(data2$newSalePrice), sd = sd(data2$newSale
q + ggtitle("Histogram of log(SalePrice)") +
  xlab("Price") + ylab("Density")
```

3.2

```
## Warning: The dot-dot notation (‘..density..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(density)‘ instead.
```




```r
# Fit the linear regression model
# I think all variables can effect the price since different neighbor has different price rang
fit3.2 <- lm(newSalePrice ~ TotalUnits + LandSqFt + GrossSqFt + YearBuilt, data = data2)
summary(fit3.2)
```

```
##
```

```
## Call:
## lm(formula = newSalePrice ~ TotalUnits + LandSqFt + GrossSqFt +
##     YearBuilt, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2887 -0.7480 -0.1949  0.5852  6.8896
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.410e+01  2.664e-02 529.398  < 2e-16 ***
## TotalUnits   2.159e-03  4.821e-04   4.479 7.54e-06 ***
## LandSqFt    -2.032e-05  1.890e-06 -10.747  < 2e-16 ***
## GrossSqFt    5.353e-06  3.042e-07  17.596  < 2e-16 ***
## YearBuilt   -6.818e-05  1.453e-05  -4.691 2.74e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.111 on 15151 degrees of freedom
## Multiple R-squared:  0.03434,    Adjusted R-squared:  0.03409
## F-statistic: 134.7 on 4 and 15151 DF,  p-value: < 2.2e-16
```

Some parts of results do not make sense since LandSqFt cannot have negitive relationship with SalePrice. Same as YearBuilt. So I analyzed each of them individually shown as following:

```
fit3.2unit <- lm(newSalePrice ~ TotalUnits, data = data2)
summary(fit3.2unit)
```

```
##
## Call:
## lm(formula = newSalePrice ~ TotalUnits, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9315 -0.7586 -0.2041  0.5750  6.9973
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.399e+01  9.125e-03 1533.65   <2e-16 ***
## TotalUnits  1.689e-03  1.182e-04   14.29   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.123 on 15154 degrees of freedom
## Multiple R-squared:  0.01329,    Adjusted R-squared:  0.01322
## F-statistic: 204.1 on 1 and 15154 DF,  p-value: < 2.2e-16
```

```r
fit3.2land <- lm(newSalePrice ~ LandSqFt, data = data2)
summary(fit3.2land)
```

```
##
## Call:
## lm(formula = newSalePrice ~ LandSqFt, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3134 -0.7592 -0.2048  0.5759  6.9983
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.400e+01  9.131e-03 1532.65   <2e-16 ***
## LandSqFt    5.325e-06  3.981e-07   13.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.124 on 15154 degrees of freedom
## Multiple R-squared:  0.01167,    Adjusted R-squared:  0.0116
## F-statistic: 178.9 on 1 and 15154 DF,  p-value: < 2.2e-16
```

```r
fit3.2gross <- lm(newSalePrice ~ GrossSqFt, data = data2)
summary(fit3.2gross)
```

```
##
## Call:
## lm(formula = newSalePrice ~ GrossSqFt, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.9596 -0.7542 -0.1998  0.5793  7.0033
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.399e+01  9.088e-03 1539.38   <2e-16 ***
## GrossSqFt   2.001e-06  1.067e-07   18.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.117 on 15154 degrees of freedom
## Multiple R-squared:  0.02268,    Adjusted R-squared:  0.02262
## F-statistic: 351.7 on 1 and 15154 DF,  p-value: < 2.2e-16
```

```r
fit3.2year <- lm(newSalePrice ~ YearBuilt, data = data2)
summary(fit3.2year)
```

```
##
## Call:
## lm(formula = newSalePrice ~ YearBuilt, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5271 -0.7585 -0.2077  0.5706  7.9402
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.410e+01  2.709e-02 520.486  < 2e-16 ***
## YearBuilt   -5.809e-05  1.477e-05  -3.933 8.44e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.13 on 15154 degrees of freedom
## Multiple R-squared:  0.001019,   Adjusted R-squared:  0.0009535
## F-statistic: 15.46 on 1 and 15154 DF,  p-value: 8.443e-05
```

From the results above we can conclude following: 1. TotalUnits is positive relevant to SalePrice. For every one unit increase in unit number, price increases by 0.1690427%. TolUnits is statistical signifant in predicting SalePrice. 2. LandSqFt is positive relevant to SalePrice. For every one unit increase in land area, price increases by 0.0005325014%. LandSqFt is statistical signifant in predicting SalePrice. 3. GrossSqFt is positive relevant to SalePrice. For every one unit increase in total area, price increases by 0.0002001002%. GrossSqFt is statistical signifant in predicting SalePrice. 4. YearBuilt is negative relevant to SalePrice. For every one unit increase in Year the property was built, price decreases by 0.005809169%. YearBuilt is statistical signifant in predicting SalePrice. This conclusion is contrary to my hypothesis since it means the older the strutures of a property was built, the more expensive the property is.

```r
# unique(data2$Neighborhood)
# unique(data2$SalePrice)
# install.packages("dplyr")
library(dplyr)
```

3.3

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

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

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

```
# Use group by to analyze the average price of properties of each neighborhood
grp_neighborhood = data2 %>% group_by(Neighborhood)  %>%
                   summarise(avg_price = mean(SalePrice),
                             .groups = 'drop')
# Ascending order of average price
grp_neighborhood <- grp_neighborhood[order(grp_neighborhood$avg_price), ]
print(grp_neighborhood)
```

```
## # A tibble: 38 x 2
##    Neighborhood                avg_price
##    <chr>                           <dbl>
##  1 "MORNINGSIDE HEIGHTS      "   452143.
##  2 "INWOOD                   "  1087096.
##  3 "WASHINGTON HEIGHTS UPPER "  1412176.
##  4 "LOWER EAST SIDE          "  1498233.
##  5 "UPPER WEST SIDE (96-116) "  1593609.
##  6 "HARLEM-CENTRAL           "  1656040.
##  7 "MANHATTAN VALLEY         "  1680225.
##  8 "WASHINGTON HEIGHTS LOWER "  1851784.
##  9 "HARLEM-UPPER             "  1889266.
## 10 "MIDTOWN EAST             "  1894725.
## # ... with 28 more rows
```

We can check which neighborhood we can afford based on the table above, and from the analysis from last step, if our own price range is not high, we also can choose some properties with low area and unit numbers or properties that the structures were built later.

3.4 We are using linear regression in this problem because the response variable is quantitative. Liner regression model can help us to find the relationship between price and other variables to find appropriate properties.

3.5 As a city planner, I can know which neighborhood has the lowest or highest average pricing, and I will get information of some common features of these properties belong to different neighborhood. For example, I can go investigate whether the cheapest community is also poor in natural environment and living convenience of residents. Then I will use this data to improve the whole city look. Also, since we know older properties tend to value more, I would suggest government to check and repair these old houses to ensure the safety and facilities are perfect.

4. Linear Regression 4

```
data3 <- read.csv("~/Desktop/Auto.csv")
fit4.1 <- lm(mpg ~ cylinders + weight + year, data = data3)
summary(fit4.1)
```

4.1

```
##
## Call:
## lm(formula = mpg ~ cylinders + weight + year, data = data3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9913 -2.3038 -0.0807  2.0158 14.3283
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.430e+01  4.042e+00  -3.539  0.00045 ***
## cylinders   -1.169e-01  2.328e-01  -0.502  0.61583
## weight      -6.450e-03  4.605e-04 -14.007  < 2e-16 ***
## year         7.581e-01  4.986e-02  15.204  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.434 on 393 degrees of freedom
## Multiple R-squared:  0.8089, Adjusted R-squared:  0.8074
## F-statistic: 554.4 on 3 and 393 DF,  p-value: < 2.2e-16
```

4.2 From the summary above, we can conclude that weight and year are statistical significant for predicting mpg since their p-value are smaller than 0.05.

```
fit4.3 <- lm(cylinders ~ weight + year, data = data3)
summary(fit4.3)
```

4.3

```
##
## Call:
## lm(formula = cylinders ~ weight + year, data = data3)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.98387 -0.51594 -0.01943  0.45903  2.20234
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.9848600  0.8617181   3.464 0.000591 ***
## weight       0.0017520  0.0000463  37.838  < 2e-16 ***
## year        -0.0359275  0.0106397  -3.377 0.000806 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7433 on 394 degrees of freedom
## Multiple R-squared:  0.8101, Adjusted R-squared:  0.8092
## F-statistic: 840.6 on 2 and 394 DF,  p-value: < 2.2e-16
```

From the summary above, we can conclude that both weight and year are significant in predicting cylinders which means cylinders, weight, and year are highly correlated. Thus, these three variables are not independent with each other as they cannot predict mpg accurately.

4.4 Based on the results above, I will not include variable cylinders in predicting mpg to avoid collinearity since it reduces the accuracy of our model.

5. Linear Regression 5

5.1 The coefficient of age is positive which means the older the house is, the higher the price it worth. The p-value, 0.00 is smaller than 0.05, so age is statistical significant for predicting value.

5.2 I thought of several reasons making older house more expensive: 1. Older houses tend to be located in more established areas. 2. Older houses always have stronger sense of community. 3. Some architectural styles are not replicated now. 4. Some older houses come with larger yards. 5. Owners are not willing to sell their historic houses.

5.3 I would suggest my intern to include following characteristics of houses: 1. Inside area and yard area 2. How established, convenient, completed the location of the house is 3. Material for building the house and inside structures 4. Room numbers and space utilization I will also suggest my intern to analyze how independent these variables are to avoid collinearity

6. Support-Vector Machines

```
# install.packages('plyr', repos = "http://cran.us.r-project.org")
# install.packages( c("e1071", "kernlab") )
library(e1071)
library(kernlab)
```

6.1

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
data(spam)
dim(spam)
```

```
## [1] 4601   58
```

```
head(spam)
```

```
##   make address  all num3d  our over remove internet order mail receive will
## 1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00  0.00 0.00    0.00 0.64
## 2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07  0.00 0.94    0.21 0.79
## 3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38 0.45
## 4 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31 0.31
## 5 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31 0.31
## 6 0.00    0.00 0.00     0 1.85 0.00   0.00     1.85  0.00 0.00    0.00 0.00
##   people report addresses free business email  you credit your font num000
## 1   0.00   0.00      0.00 0.32     0.00  1.29 1.93   0.00 0.96    0   0.00
## 2   0.65   0.21      0.14 0.14     0.07  0.28 3.47   0.00 1.59    0   0.43
## 3   0.12   0.00      1.75 0.06     0.06  1.03 1.36   0.32 0.51    0   1.16
## 4   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0   0.00
## 5   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0   0.00
## 6   0.00   0.00      0.00 0.00     0.00  0.00 0.00   0.00 0.00    0   0.00
##   money hp hpl george num650 lab labs telnet num857 data num415 num85
## 1  0.00  0   0      0      0   0    0      0      0    0      0     0
## 2  0.43  0   0      0      0   0    0      0      0    0      0     0
## 3  0.06  0   0      0      0   0    0      0      0    0      0     0
## 4  0.00  0   0      0      0   0    0      0      0    0      0     0
## 5  0.00  0   0      0      0   0    0      0      0    0      0     0
## 6  0.00  0   0      0      0   0    0      0      0    0      0     0
##   technology num1999 parts pm direct cs meeting original project   re  edu
## 1          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 2          0    0.07     0  0   0.00  0       0     0.00       0 0.00 0.00
## 3          0    0.00     0  0   0.06  0       0     0.12       0 0.06 0.06
## 4          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 5          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 6          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
##   table conference charSemicolon charRoundbracket charSquarebracket
## 1     0          0          0.00            0.000                 0
```

```
## 2     0          0          0.00            0.132                 0
## 3     0          0          0.01            0.143                 0
## 4     0          0          0.00            0.137                 0
## 5     0          0          0.00            0.135                 0
## 6     0          0          0.00            0.223                 0
##   charExclamation charDollar charHash capitalAve capitalLong capitalTotal type
## 1           0.778      0.000    0.000      3.756          61          278 spam
## 2           0.372      0.180    0.048      5.114         101         1028 spam
## 3           0.276      0.184    0.010      9.821         485         2259 spam
## 4           0.137      0.000    0.000      3.537          40          191 spam
## 5           0.135      0.000    0.000      3.537          40          191 spam
## 6           0.000      0.000    0.000      3.000          15           54 spam
```

```
?spam
set.seed(02115)
sample <- sample( c(TRUE, FALSE), nrow(spam), replace=TRUE)
train <- spam[sample,]
test <- spam[!sample,]
```

```
library(e1071)
# Fit the support vector machine model on the training data
fit6.2 <- svm(type ~ ., data = train, type='C-classification', kernel='linear', scale = FALSE)
```

6.2

```
# Use our model to predict on the testing data
pred_value <- predict(fit6.2, test)
true_value <- test$type
# Make a confusion matrix
tt <- table(pred_value, true_value)
tt
```

6.3

```
##           true_value
## pred_value nonspam spam
##    nonspam    1261   63
##    spam        137  870
```

```r
# Calculate the error rate
error_rate <- (tt[1,2] + tt[2,1])/nrow(test)
error_rate
```

```
## [1] 0.08580009
```

The classification error rate is 9%.

```r
# Change the kernal and cost of new model
fit6.4 <- svm(type ~ ., data = train, type='C-classification', kernel='sigmoid', cost = 100, s
# Predict using new model
new_pred_value <- predict(fit6.4, test)
new_true_value <- test$type
# Calculate updated error rate
tt <- table(new_pred_value, new_true_value)
tt
```

6.4

```
##               new_true_value
## new_pred_value nonspam spam
##        nonspam     669  807
##        spam        729  126
```

```r
new_error_rate <- (tt[1,2] + tt[2,1])/nrow(test)
new_error_rate
```

```
## [1] 0.6589447
```

The updated classification error rate is 66%. The updated error rate is much larger than the original suggests me the updated cost is very large which leads to a narrower margin leading to more misclassifications because there will be less support vectors on the or violating the margin. Also, we changed kernal leads to a larger error suggests that sigmoid is not proper to this dataset.

6.5 I think interpreting svm result is much harder than regression model since svm summary does not provide sophisticated and interpretable reports. For example, svm does not provide p-value or R squared value so we cannot know how significant a predictor is or how well the model perform; it does not provide if the dependent variable is positive or negative to independent variables; however, regression model can provide these information to help us analyze. Also the cost of svm is much larger and it is very easy to overfit. So in conclusion, svm is harder to interpret than regression model.

```
library(tree)
set.seed(2022)
kFolds <- 10
folds <- sample(1:kFolds, nrow(spam), replace = TRUE)
table(folds)
```

6.6

```
## folds
##   1   2   3   4   5   6   7   8   9  10
## 454 435 475 436 465 467 431 455 469 514
```

```
results <- data.frame( Fold = 1:kFolds, HoldoutAccuracy = NA)
for(i in 1:kFolds) {
  train <- spam[ ! folds == i,]
  test  <- spam[ folds == i,]
  fit6.5 <- tree(type ~ . ,data = train)
  fit6.5.2 <- predict(fit6.5, test, type = "class")
  tab <- table(fit6.5.2, test$type)
  result$HoldoutAccuracy[i] <- (tab[1,1] + tab[2,2]) / nrow(test)}
```

```
## Error in eval(expr, envir, enclos): object 'result' not found
```

```
result
```

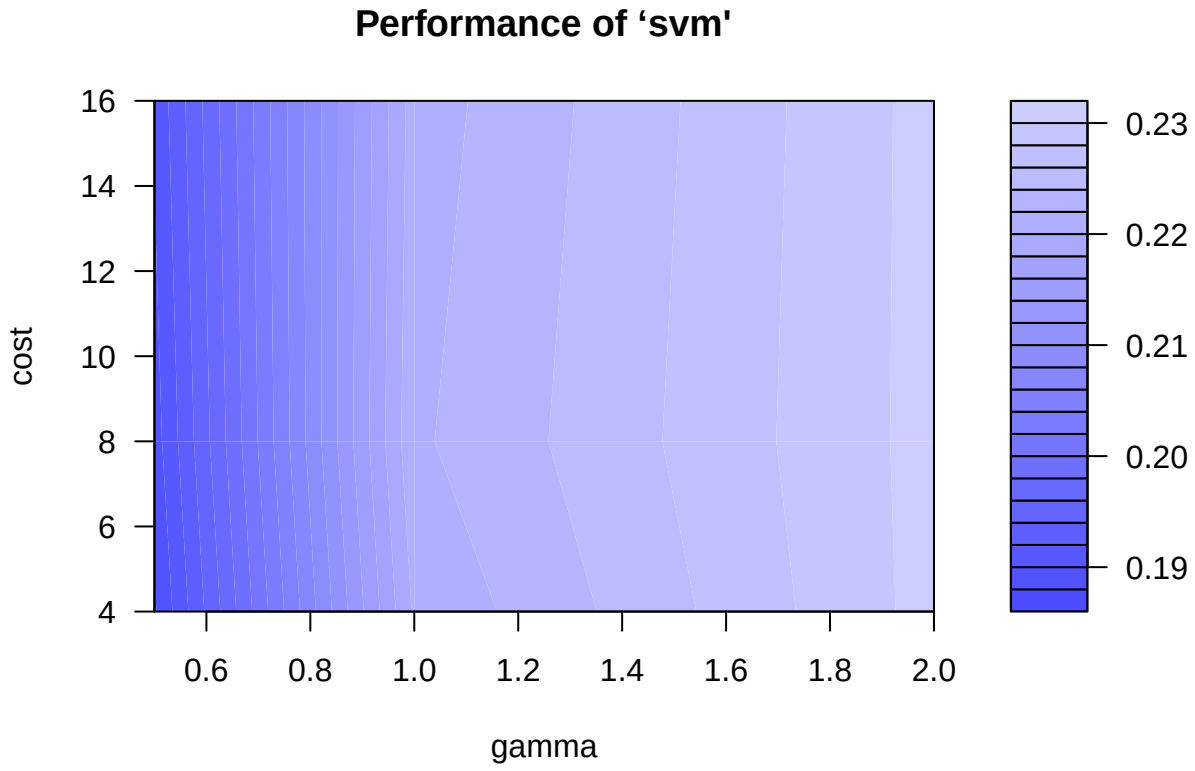
```
## Error in eval(expr, envir, enclos): object 'result' not found
```

```
# ?tune
bestpar <- tune(svm, type ~ ., data = spam,
              ranges = list(gamma = 2^(-1:1), cost = 2^(2:4)),
              tunecontrol = tune.control(sampling = "fix")
             )
summary(bestpar)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: fixed training/validation set
##
## - best parameters:
##  gamma cost
##    0.5    4
##
```

```
## - best performance: 0.1877445
##
## - Detailed performance results:
##   gamma cost     error dispersion
## 1   0.5    4 0.1877445         NA
## 2   1.0    4 0.2203390         NA
## 3   2.0    4 0.2307692         NA
## 4   0.5    8 0.1890482         NA
## 5   1.0    8 0.2216428         NA
## 6   2.0    8 0.2307692         NA
## 7   0.5   16 0.1903520         NA
## 8   1.0   16 0.2209909         NA
## 9   2.0   16 0.2307692         NA
```

```
plot(bestpar)
```



7. Clustering & PCA

```
data4 <- read.csv("~/Desktop/Mall_Customers.csv")
# Drop CustomerID
```

```
data4$CustomerID <- NULL
# Change Gender variavle to binary since string values cannot be used in clustering
data4$Gender <- ifelse(data4$Gender == "Male", 1, 0)
head(data4)
```

7.1

```
##   Gender Age AnnualIncome SpendingScore
## 1      1  19           15            39
## 2      1  21           15            81
## 3      0  20           16             6
## 4      0  23           16            77
## 5      0  31           17            40
## 6      0  22           17            76
```

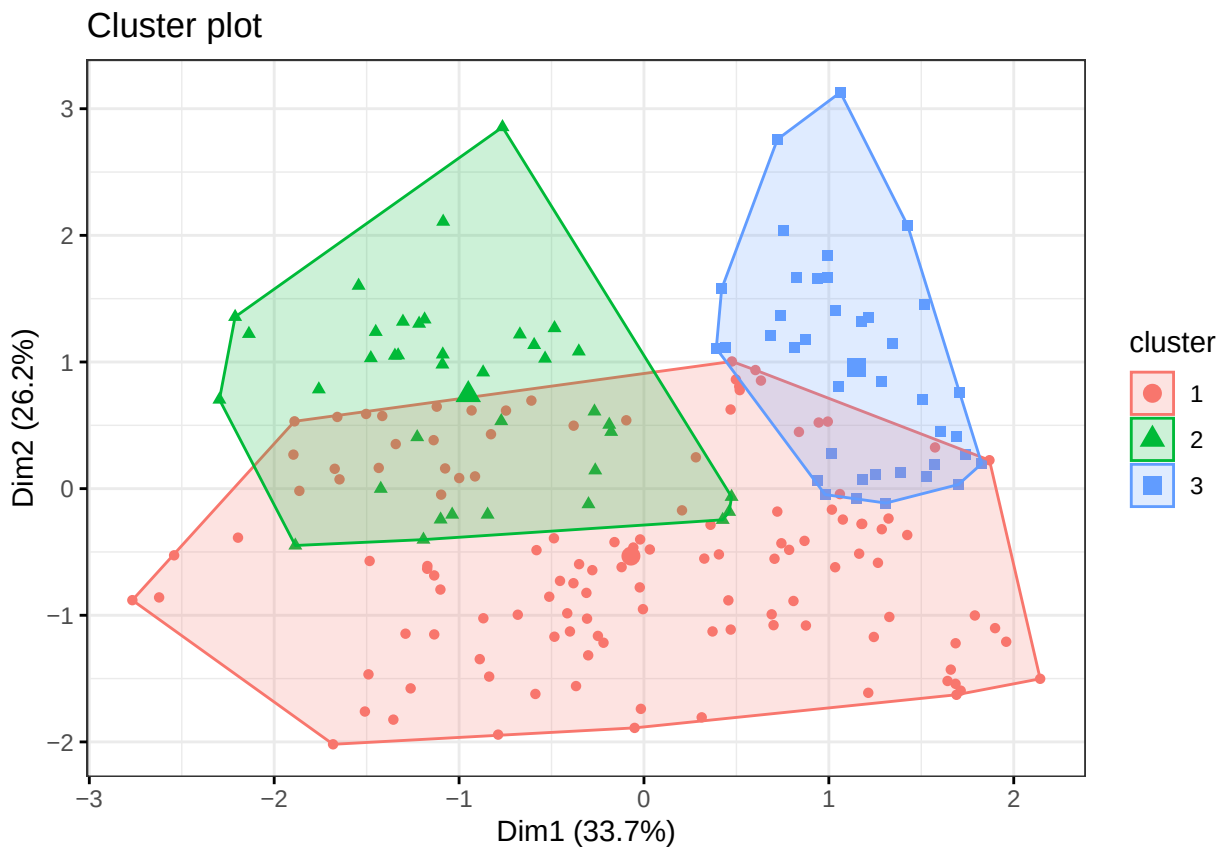
```
# Fit k-means clustering model with k=3
fit7.1 <- kmeans(data4, centers = 3, nstart = 20)
fit7.1$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3
## [149] 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3
## [186] 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3
```

```
# install.packages("factoextra")
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_cluster(fit7.1, data = data4,
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
             )
```



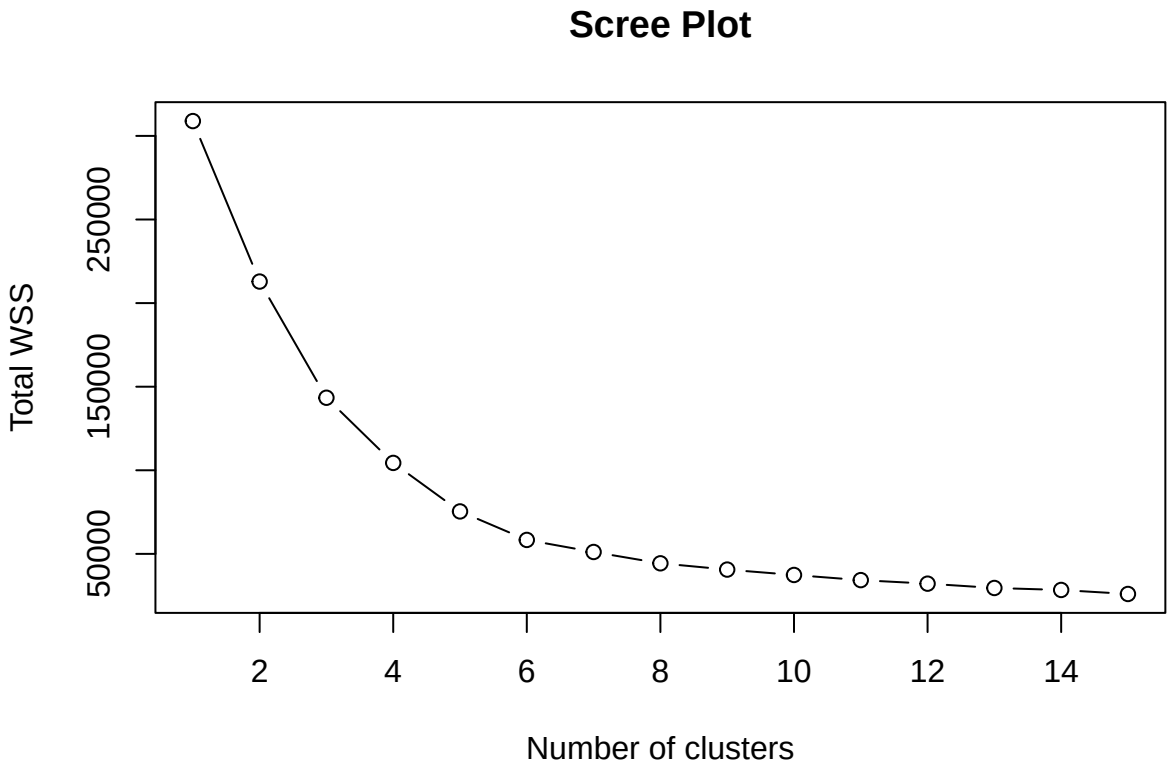

```
Within_cluster_sum <- numeric(15)
# Use for loop to perform k-means with different k values
for(i in 1:15)
{
  #Extract total with-in cluster sum of square
  Within_cluster_sum[i] <- kmeans(data4, centers = i, nstart = 20)$tot.withinss
}
cbind(No.of.Cluters = 1:15, Within_cluster_sum)
```

7.2

```
##       No.of.Cluters Within_cluster_sum
##  [1,]             1          308862.06
##  [2,]             2          212889.44
##  [3,]             3          143391.59
##  [4,]             4          104414.68
##  [5,]             5           75399.62
##  [6,]             6           58348.64
##  [7,]             7           51130.69
##  [8,]             8           44355.31
```

```
##  [9,]              9          40615.15
## [10,]             10          37391.43
## [11,]             11          34311.16
## [12,]             12          32195.12
## [13,]             13          29609.31
## [14,]             14          28416.91
## [15,]             15          26020.91
```

```
# Plot number of clusters with total WSS
plot(1:15, Within_cluster_sum, type="b", xlab = "Number of clusters", ylab = "Total WSS", main
```



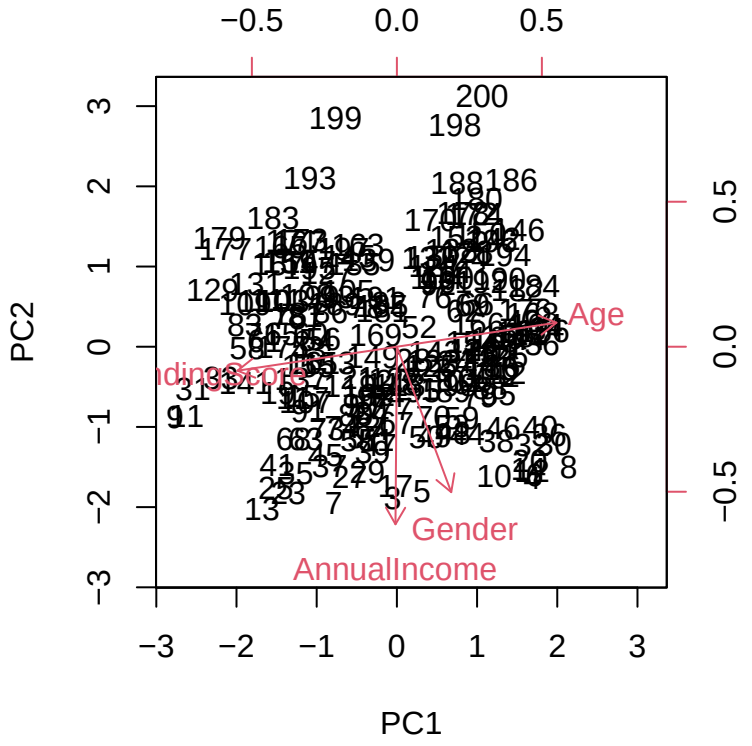
I suggest to use 6 clusters to group our customers since the proper elbow is at 6 which means the larger k value will not increase efficiency.

```
# PCA analysis
pcaFit <- prcomp(data4, scale = TRUE)
pcaFit$rotation <- -pcaFit$rotation
# Check PCs
pcaFit$rotation
```

7.3

```
##                        PC1        PC2        PC3          PC4
## Gender          0.234301560 -0.6268855 -0.7430091 -0.007405324
## Age             0.687900253  0.1036895  0.1223844  0.707858590
## AnnualIncome   -0.006082173 -0.7652519  0.6436671  0.006721332
## SpendingScore -0.686919957 -0.1032111 -0.1365732  0.706283372
```

```
biplot(pcaFit, scale=0)
```

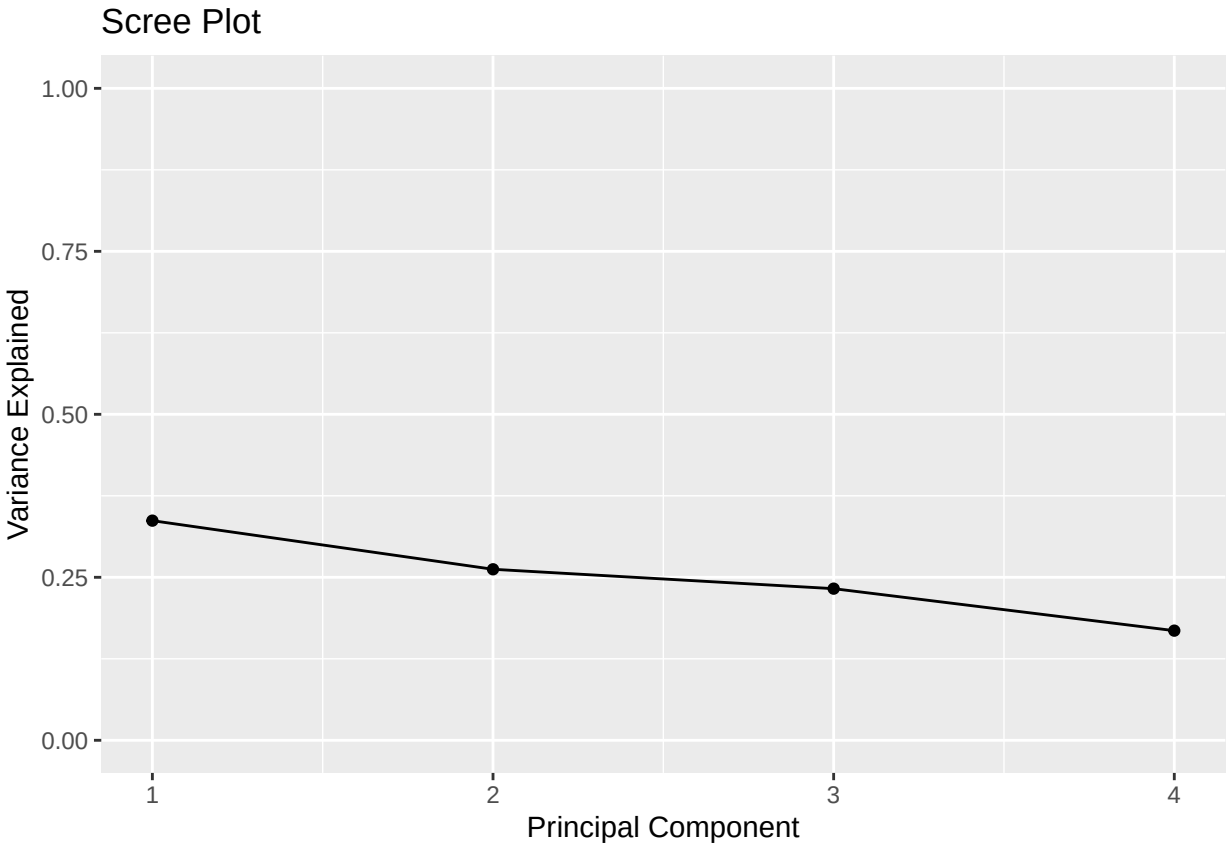



```
var_explained = pcaFit$sdev^2 / sum(pcaFit$sdev^2)
var_explained
```

```
## [1] 0.3369005 0.2623064 0.2326064 0.1681867
```

```
qplot(c(1:4), var_explained) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```



PC1 has large positive associations with Age and large negative associations with SpendingScore, so PC1 can be named as Spending_Ability. PC2 has large negative association with AnnualIncome and large negative associations with Gender, so PC2 can be named as Financial_Stability.

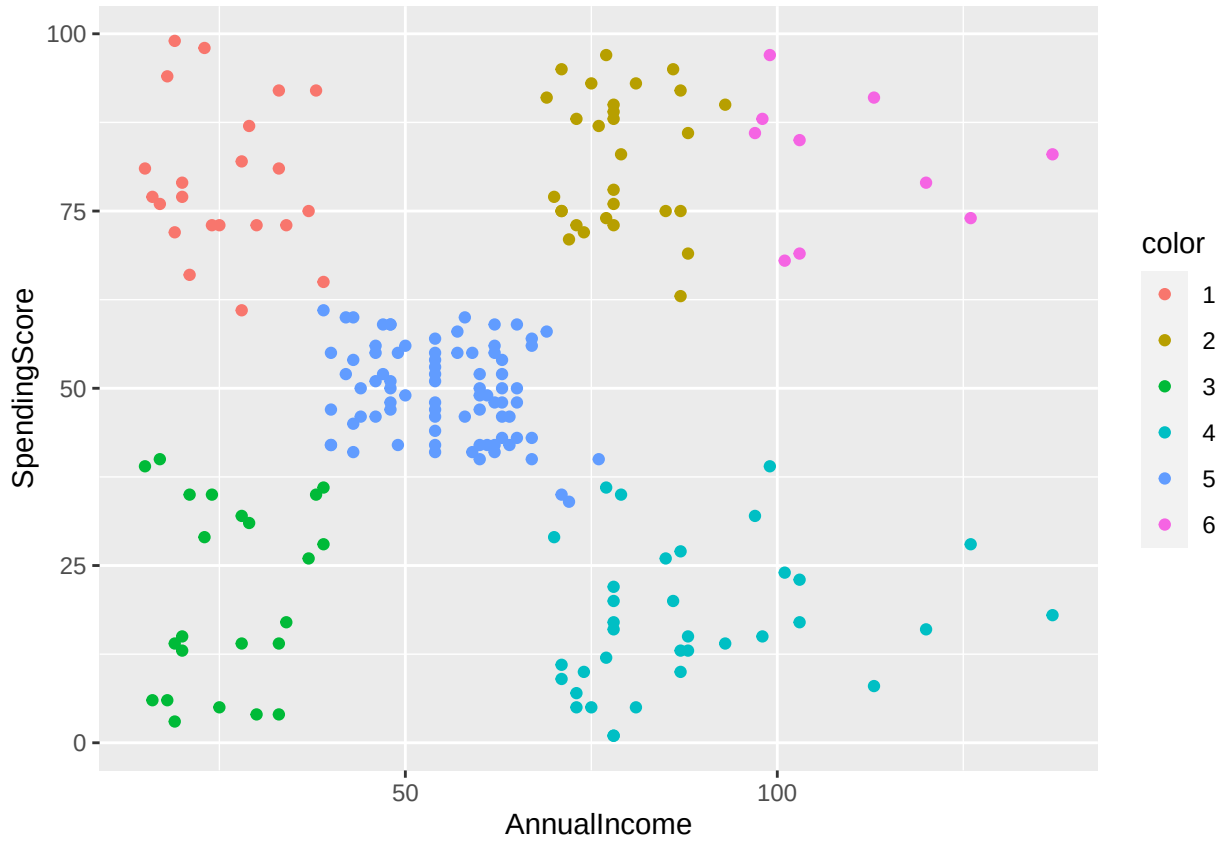
```
# Subset a new dataset with columns AnnualIncome and SpendingScore since these two should be t
newdataset <- data4[ , -c(1, 2)]
fit7.4 <- kmeans(newdataset, centers = 6, nstart = 20)
fit7.4$cluster
```

7.4

```
##   [1] 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3
##  [38] 1 3 1 3 1 3 5 3 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [75] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [112] 5 5 5 5 5 5 5 5 5 5 5 5 2 4 2 5 2 4 2 4 2 5 2 4 2 4 2 4 2 4 2 5 2 4 2 4 2
## [149] 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 6 4 6 4
## [186] 6 4 6 4 6 4 6 4 6 4 6 4 6 4 6
```

```
newdataset$color <- as.factor(fit7.4$cluster)
# Cluster customers by AnnualIncome and SpendingScore
```

```
ggplot(newdataset, aes(x = AnnualIncome, y = SpendingScore, color = color)) +
  geom_point()
```



The cluster at right top coner should be the most valuable customers.