STP598-Assignment 1

Hao Wang
hwang306

September 14, 2017

1 Question 1

Load the data first, with a brief inspection.

```
mydata <- read.csv("https://raw.githubusercontent.com/haowang666/Computational-Stats/mas
summary(mydata)
```

```
##      HtVol             Male              CT              Age
##  Min.   : 112.2   Min.   :0.0000   Min.   :0.000   Min.   : 13.0
##  1st Qu.: 340.7   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:110.5
##  Median : 539.9   Median :1.0000   Median :1.000   Median :174.5
##  Mean   : 535.5   Mean   :0.6034   Mean   :0.569   Mean   :156.8
##  3rd Qu.: 680.7   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:203.2
##  Max.   :1340.2   Max.   :1.0000   Max.   :1.000   Max.   :359.0
##        Ht               Wt              BMI             BSA
##  Min.   : 71.0   Min.   :  7.90   Min.   :13.51   Min.   :0.390
##  1st Qu.:128.2   1st Qu.: 28.85   1st Qu.:16.62   1st Qu.:1.005
##  Median :157.2   Median : 54.30   Median :21.37   Median :1.550
##  Mean   :148.0   Mean   : 53.60   Mean   :22.72   Mean   :1.458
##  3rd Qu.:165.7   3rd Qu.: 71.90   3rd Qu.:27.07   3rd Qu.:1.810
##  Max.   :186.0   Max.   :139.20   Max.   :46.51   Max.   :2.590
```

1.1 Predictive model

I build a simple linear model first

$$\hat{HtVol} = \beta_0 + \beta_1 Male + \beta_2 Age + \beta_3 Ht + \beta_4 Wt$$

Use `lm` function to obtain the LS coefficients

```
lm <- lm(HtVol ~ Male + Age + Ht + Wt, data = mydata)
summary(lm)
```

```
##
## Call:
```

```
## lm(formula = HtVol ~ Male + Age + Ht + Wt, data = mydata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -209.56  -57.02    2.58   41.83  199.03
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -263.6665    90.2773  -2.921  0.00512 **
## Male          41.3395    23.1534   1.785  0.07991 .
## Age           -0.3373     0.3797  -0.888  0.37835
## Ht             3.3829     0.9713   3.483  0.00100 **
## Wt             6.0891     0.6795   8.962 3.37e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 82.28 on 53 degrees of freedom
## Multiple R-squared:  0.8904, Adjusted R-squared:  0.8821
## F-statistic: 107.6 on 4 and 53 DF,  p-value: < 2.2e-16
```

With the `lm` function, the fitting line is

$$\hat{HtVol} = -263.67 + 41.34 * Male - 0.34 * Age + 3.38 * Ht + 0.68 * Wt$$

I use package 'L1pack' to perform least absolute deviation regression

Unlike Least Squares, Least Absolute Deviation minimize

$$S = \sum_{i=1}^{n} |y_i - f(x_i)|$$

```
library("L1pack")
lad  <-  lad(HtVol ~ Male + Age + Ht + Wt, data = mydata)
summary(lad)
```

```
## Call:
## lad(formula = HtVol ~ Male + Age + Ht + Wt, data = mydata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -133.32  -37.07    0.00   36.72  339.75
##
## Coefficients:
##             Estimate  Std.Error Z value   p-value
## (Intercept) -213.7607    0.7758 -275.5348    0.0000
```

```
## Male          48.1662    0.1990  242.0777    0.0000
## Age            0.1834    0.0033   56.2063    0.0000
## Ht             2.9940    0.0083  358.7031    0.0000
## Wt             4.4133    0.0058  755.8221    0.0000
##
## Degrees of freedom: 58 total; 53 residual
## Scale estimate: 80.61168
## Log-likelihood: -332.7006 on 6 degrees of freedom
```

The fitting line is

$$\hat{HtVol} = -213.76 + 48.17 * Male + 0.18 * Age + 2.99 * Ht + 4.41 * Wt$$

1.2 Changing Models

Steps are very similar, just change the explainatory variables

```
lm <- lm(HtVol ~ Male + Age + BMI + BSA, data = mydata)
summary(lm)
```

```
##
## Call:
## lm(formula = HtVol ~ Male + Age + BMI + BSA, data = mydata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -184.474  -51.125   -2.295   35.538  265.924
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -121.9456    41.7940  -2.918  0.00516 **
## Male          37.0077    24.0400   1.539  0.12965
## Age           -0.6737     0.3966  -1.699  0.09520 .
## BMI           -5.2994     2.9600  -1.790  0.07912 .
## BSA          590.6002    73.5415   8.031 9.99e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 84.36 on 53 degrees of freedom
## Multiple R-squared:  0.8848, Adjusted R-squared:  0.8761
## F-statistic: 101.8 on 4 and 53 DF,  p-value: < 2.2e-16
```

The linear fitting is

$$\hat{HtVol} = -121.95 + 37 * Male - 0.67 * Age - 5.30 * BMI + 590 * BSA$$

For the LAD method:

```
lad <- lad(HtVol ~ Male + Age + BMI + BSA, data = mydata)
summary(lad)
```

```
## Call:
## lad(formula = HtVol ~ Male + Age + BMI + BSA, data = mydata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -130.22  -41.36    0.00   42.30  394.91
##
## Coefficients:
##             Estimate  Std.Error Z value   p-value
## (Intercept)  -40.9899    0.3503 -117.0057    0.0000
## Male          37.8707    0.2015  187.9367    0.0000
## Age           -0.2234    0.0033  -67.1994    0.0000
## BMI           -6.6558    0.0248 -268.2551    0.0000
## BSA          501.5719    0.6164  813.6613    0.0000
##
## Degrees of freedom: 58 total; 53 residual
## Scale estimate: 80.12083
## Log-likelihood: -332.3463 on 6 degrees of freedom
```

$$\hat{HtVol} = -40.99 + 37.87 * Male - 0.22 * Age - 6.66 * BMI + 501.57 * BSA$$

1.3 Ten fold cross validation

I did 10-cv for question 1.1 first (predictors are Male, Age, Ht and Wt).

```
#set seed
k = 10
library(boot)
set.seed(99)
folds <- sample(1:k, nrow(mydata), replace = TRUE)
```

```
for (i in 1:10) {
  #ls fit
  ls.fit <- lm(HtVol ~ Male + Age + Ht + Wt, data = mydata[folds != i, ])
  pred.ls <- predict(ls.fit, mydata[folds == i, ])
  #rmse and mae and smdape
  print(paste0(i,") RMSE of Question 1.1 (LS fit): ",
```

```
              sqrt(mean((mydata$HtVol[folds == i] - pred.ls)^2))
              ))
  print(paste0(i,") MAE of Question 1.1 (LS fit): ",
              mean(abs(mydata$HtVol[folds == i] - pred.ls))
              ))
  print(paste0(i,") sMdAPE of Question 1.1 (LS fit): ",
              median(200*(
                abs(mydata$HtVol[folds == i] - pred.ls) /
                  (mydata$HtVol[folds == i] = pred.ls)))
              ))
}
```

```
## [1] "1) RMSE of Question 1.1 (LS fit): 79.2572579503131"
## [1] "1) MAE of Question 1.1 (LS fit): 72.5978339070048"
## [1] "1) sMdAPE of Question 1.1 (LS fit): 29.5465011192757"
## [1] "2) RMSE of Question 1.1 (LS fit): 73.8192720920171"
## [1] "2) MAE of Question 1.1 (LS fit): 69.6972690177388"
## [1] "2) sMdAPE of Question 1.1 (LS fit): 29.342953592183"
## [1] "3) RMSE of Question 1.1 (LS fit): 77.8045787396735"
## [1] "3) MAE of Question 1.1 (LS fit): 75.7837179217279"
## [1] "3) sMdAPE of Question 1.1 (LS fit): 26.4576974139865"
## [1] "4) RMSE of Question 1.1 (LS fit): 128.379379826768"
## [1] "4) MAE of Question 1.1 (LS fit): 78.8670025057081"
## [1] "4) sMdAPE of Question 1.1 (LS fit): 15.2815975127876"
## [1] "5) RMSE of Question 1.1 (LS fit): 39.4939367424547"
## [1] "5) MAE of Question 1.1 (LS fit): 36.0114091736559"
## [1] "5) sMdAPE of Question 1.1 (LS fit): 11.8447982230939"
## [1] "6) RMSE of Question 1.1 (LS fit): 68.0490060157573"
## [1] "6) MAE of Question 1.1 (LS fit): 54.9183785428069"
## [1] "6) sMdAPE of Question 1.1 (LS fit): 22.9141350432453"
## [1] "7) RMSE of Question 1.1 (LS fit): 110.721459230973"
## [1] "7) MAE of Question 1.1 (LS fit): 78.6692229675634"
## [1] "7) sMdAPE of Question 1.1 (LS fit): 15.6163646181016"
## [1] "8) RMSE of Question 1.1 (LS fit): 25.6464503060201"
## [1] "8) MAE of Question 1.1 (LS fit): 22.4791142032178"
## [1] "8) sMdAPE of Question 1.1 (LS fit): 8.69176471170605"
## [1] "9) RMSE of Question 1.1 (LS fit): 88.0642588058668"
## [1] "9) MAE of Question 1.1 (LS fit): 43.3427970766537"
## [1] "9) sMdAPE of Question 1.1 (LS fit): 4.14863468666304"
## [1] "10) RMSE of Question 1.1 (LS fit): 77.6038519202542"
## [1] "10) MAE of Question 1.1 (LS fit): 65.501569084523"
## [1] "10) sMdAPE of Question 1.1 (LS fit): 22.8729741211142"
```

For LAD fit, it can be done similarly

```
for (i in 1:10) {
  #lad fit
  lad.fit <- lad(HtVol ~ Male + Age + Ht + Wt, data = mydata[folds != i, ])
  pred.lad <- predict(lad.fit, mydata[folds == i, ])
  #rmse and mae and smdape
  print(paste0(i,") RMSE of Question 1.1 (LAD fit): ",
               sqrt(mean((mydata$HtVol[folds == i] - pred.lad)^2))
               ))
  print(paste0(i,") MAE of Question 1.1 (LAD fit): ",
               mean(abs(mydata$HtVol[folds == i] - pred.lad))
               ))
  print(paste0(i,") sMdAPE of Question 1.1 (LAD fit): ",
               median(200*(
                 abs(mydata$HtVol[folds == i] - pred.lad) /
                   (mydata$HtVol[folds == i] = pred.lad)))
               ))
}
```

```
## [1] "1) RMSE of Question 1.1 (LAD fit): 14.3453518385207"
## [1] "1) MAE of Question 1.1 (LAD fit): 11.9961309633368"
## [1] "1) sMdAPE of Question 1.1 (LAD fit): 5.64465635421732"
## [1] "2) RMSE of Question 1.1 (LAD fit): 15.7020361867056"
## [1] "2) MAE of Question 1.1 (LAD fit): 12.2976451600937"
## [1] "2) sMdAPE of Question 1.1 (LAD fit): 3.23507965587468"
## [1] "3) RMSE of Question 1.1 (LAD fit): 14.8885252726359"
## [1] "3) MAE of Question 1.1 (LAD fit): 10.7352757696485"
## [1] "3) sMdAPE of Question 1.1 (LAD fit): 2.14928758343317"
## [1] "4) RMSE of Question 1.1 (LAD fit): 5.19363389238994"
## [1] "4) MAE of Question 1.1 (LAD fit): 2.81149142181363"
## [1] "4) sMdAPE of Question 1.1 (LAD fit): 0.167256237496758"
## [1] "5) RMSE of Question 1.1 (LAD fit): 4.60003946020518"
## [1] "5) MAE of Question 1.1 (LAD fit): 4.20206940089508"
## [1] "5) sMdAPE of Question 1.1 (LAD fit): 1.5525045880044"
## [1] "6) RMSE of Question 1.1 (LAD fit): 6.02762316651441"
## [1] "6) MAE of Question 1.1 (LAD fit): 4.07753931555348"
## [1] "6) sMdAPE of Question 1.1 (LAD fit): 0.719956450000851"
## [1] "7) RMSE of Question 1.1 (LAD fit): 3.36965813966359"
## [1] "7) MAE of Question 1.1 (LAD fit): 3.10097771133032"
## [1] "7) sMdAPE of Question 1.1 (LAD fit): 0.985513823476876"
## [1] "8) RMSE of Question 1.1 (LAD fit): 3.1477322175539"
## [1] "8) MAE of Question 1.1 (LAD fit): 2.20108713680344"
## [1] "8) sMdAPE of Question 1.1 (LAD fit): 0.373846757015664"
## [1] "9) RMSE of Question 1.1 (LAD fit): 9.59114288466579"
## [1] "9) MAE of Question 1.1 (LAD fit): 7.91740084949112"
```

```
## [1] "9) sMdAPE of Question 1.1 (LAD fit): 2.81383598249695"
## [1] "10) RMSE of Question 1.1 (LAD fit): 1.89227266469138"
## [1] "10) MAE of Question 1.1 (LAD fit): 1.46983730507167"
## [1] "10) sMdAPE of Question 1.1 (LAD fit): 0.535006538078385"
```

for question 1.2, all I need to do is to change model specification.

```
for (i in 1:10) {
  #ls fit
  ls.fit <- lm(HtVol ~ Male + Age + BMI + BSA, data = mydata[folds != i, ])
  pred.ls <- predict(ls.fit, mydata[folds == i, ])
  #rmse and mae and smdape
  print(paste0(i,") RMSE of Question 1.1 (LS fit): ",
               sqrt(mean((mydata$HtVol[folds == i] - pred.ls)^2))
               ))
  print(paste0(i,") MAE of Question 1.1 (LS fit): ",
               mean(abs(mydata$HtVol[folds == i] - pred.ls))
               ))
  print(paste0(i,") sMdAPE of Question 1.1 (LS fit): ",
               median(200*(
                 abs(mydata$HtVol[folds == i] - pred.ls) /
                   (mydata$HtVol[folds == i] = pred.ls)))
               ))
}
```

```
## [1] "1) RMSE of Question 1.1 (LS fit): 8.14155918652547"
## [1] "1) MAE of Question 1.1 (LS fit): 5.23538765295797"
## [1] "1) sMdAPE of Question 1.1 (LS fit): 1.41723082445628"
## [1] "2) RMSE of Question 1.1 (LS fit): 7.10432252363834"
## [1] "2) MAE of Question 1.1 (LS fit): 5.82091463704941"
## [1] "2) sMdAPE of Question 1.1 (LS fit): 2.11904815749066"
## [1] "3) RMSE of Question 1.1 (LS fit): 11.9153267592014"
## [1] "3) MAE of Question 1.1 (LS fit): 11.3484655206159"
## [1] "3) sMdAPE of Question 1.1 (LS fit): 4.20198197608745"
## [1] "4) RMSE of Question 1.1 (LS fit): 17.9358770579246"
## [1] "4) MAE of Question 1.1 (LS fit): 11.6558591842775"
## [1] "4) sMdAPE of Question 1.1 (LS fit): 2.43692160105206"
## [1] "5) RMSE of Question 1.1 (LS fit): 9.63043457790252"
## [1] "5) MAE of Question 1.1 (LS fit): 9.11995342049252"
## [1] "5) sMdAPE of Question 1.1 (LS fit): 2.45009299694824"
## [1] "6) RMSE of Question 1.1 (LS fit): 13.5543334747355"
## [1] "6) MAE of Question 1.1 (LS fit): 9.75190146201487"
## [1] "6) sMdAPE of Question 1.1 (LS fit): 2.8793579022763"
## [1] "7) RMSE of Question 1.1 (LS fit): 6.99148670614194"
## [1] "7) MAE of Question 1.1 (LS fit): 6.01971586282796"
## [1] "7) sMdAPE of Question 1.1 (LS fit): 2.7450863923397"
```

```
## [1] "8) RMSE of Question 1.1 (LS fit): 6.09471752204587"
## [1] "8) MAE of Question 1.1 (LS fit): 5.74075962398751"
## [1] "8) sMdAPE of Question 1.1 (LS fit): 2.01004294103703"
## [1] "9) RMSE of Question 1.1 (LS fit): 12.248700032907"
## [1] "9) MAE of Question 1.1 (LS fit): 10.3521637922481"
## [1] "9) sMdAPE of Question 1.1 (LS fit): 3.96446166318085"
## [1] "10) RMSE of Question 1.1 (LS fit): 5.90511714948728"
## [1] "10) MAE of Question 1.1 (LS fit): 4.67887793521267"
## [1] "10) sMdAPE of Question 1.1 (LS fit): 1.53497658238639"
```

Similarly for LAD

I kept got warning in formating with `rmarkdown` *here: the solutions are not unique.*

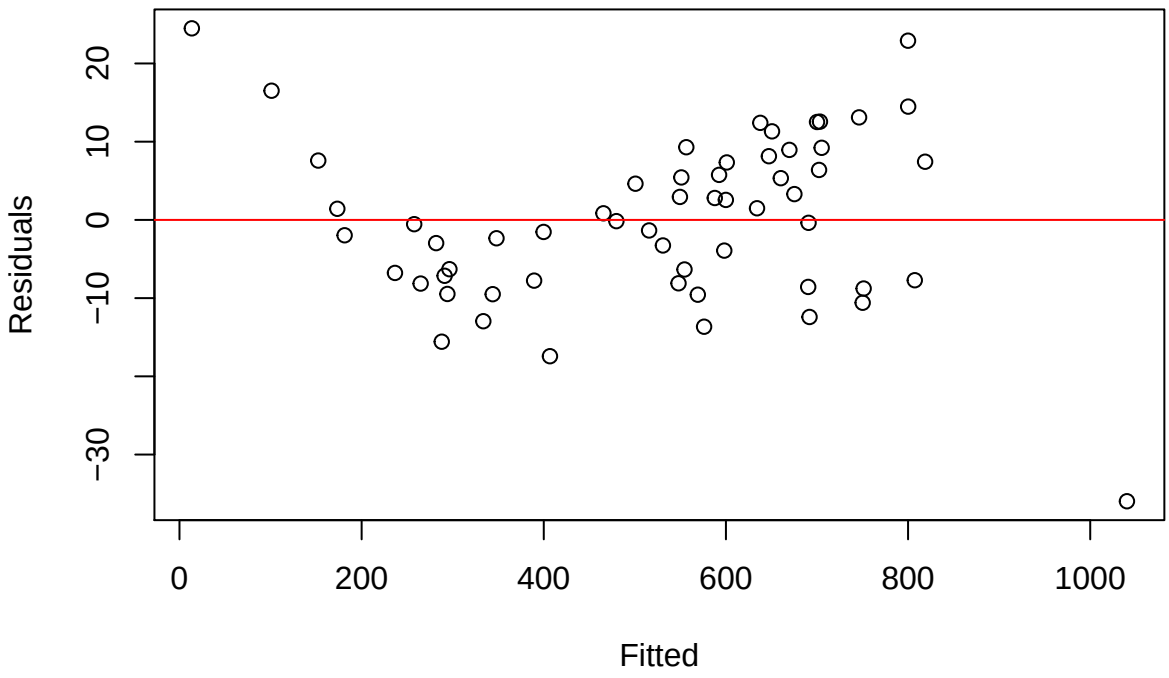
```
for (i in 1:10) {
  #lad fit
  lad.fit <- lad(HtVol ~ Male + Age + BMI + BSA, data = mydata[folds != i, ], print.it
 = FALSE)
  pred.lad <- predict(lad.fit, mydata[folds == i, ])
  #rmse and mae and smdape
  print(paste0(i,") RMSE of Question 1.1 (LAD fit): ",
               sqrt(mean((mydata$HtVol[folds == i] - pred.lad)^2))
               ))
  print(paste0(i,") MAE of Question 1.1 (LAD fit): ",
               mean(abs(mydata$HtVol[folds == i] - pred.lad))
               ))
  print(paste0(i,") sMdAPE of Question 1.1 (LAD fit): ",
               median(200*(
                 abs(mydata$HtVol[folds == i] - pred.lad) /
                   (mydata$HtVol[folds == i] = pred.lad)))
               ))
}
```

1.4 Residul fitting plots

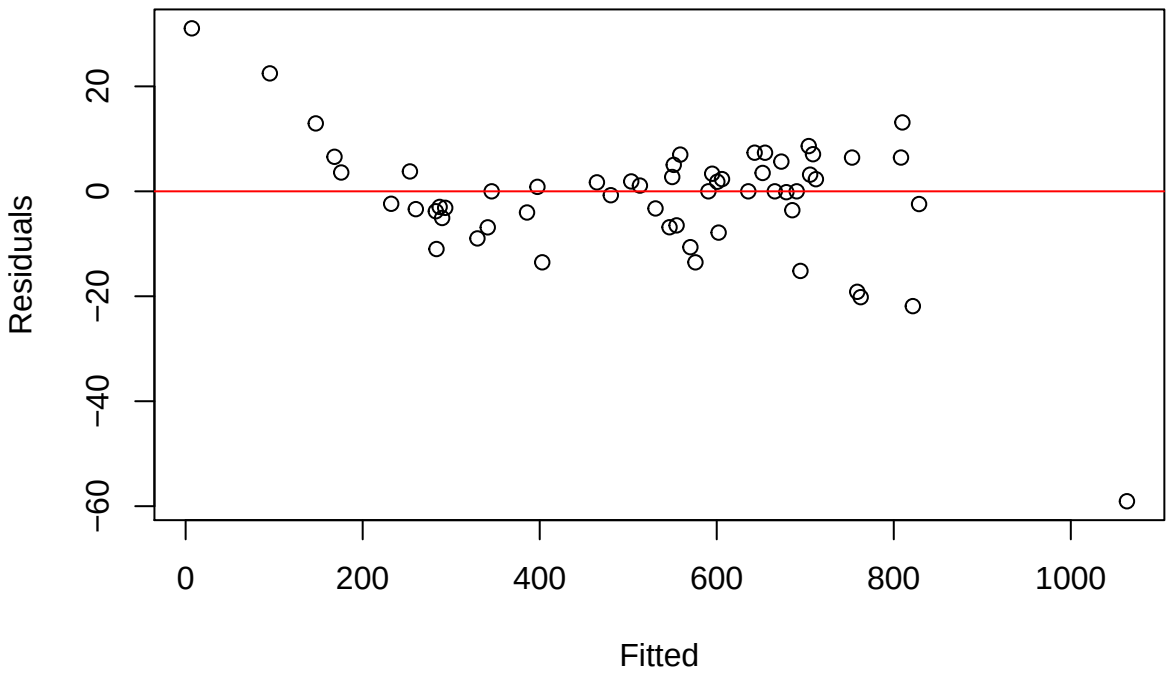
It seems to me that LS fitting is better than LAD fitting. To illustrate my point I draw residual fitted plot.

```
library(ggplot2)
lm <- lm(HtVol ~ Male + Age + Ht + Wt, data = mydata)
lad <- lad(HtVol ~ Male + Age + Ht + Wt, data = mydata)

plot(fitted(lm), residuals(lm),  xlab="Fitted", ylab="Residuals")
abline(h=0, col="red") # draws a horizontal red line at y = 0
```

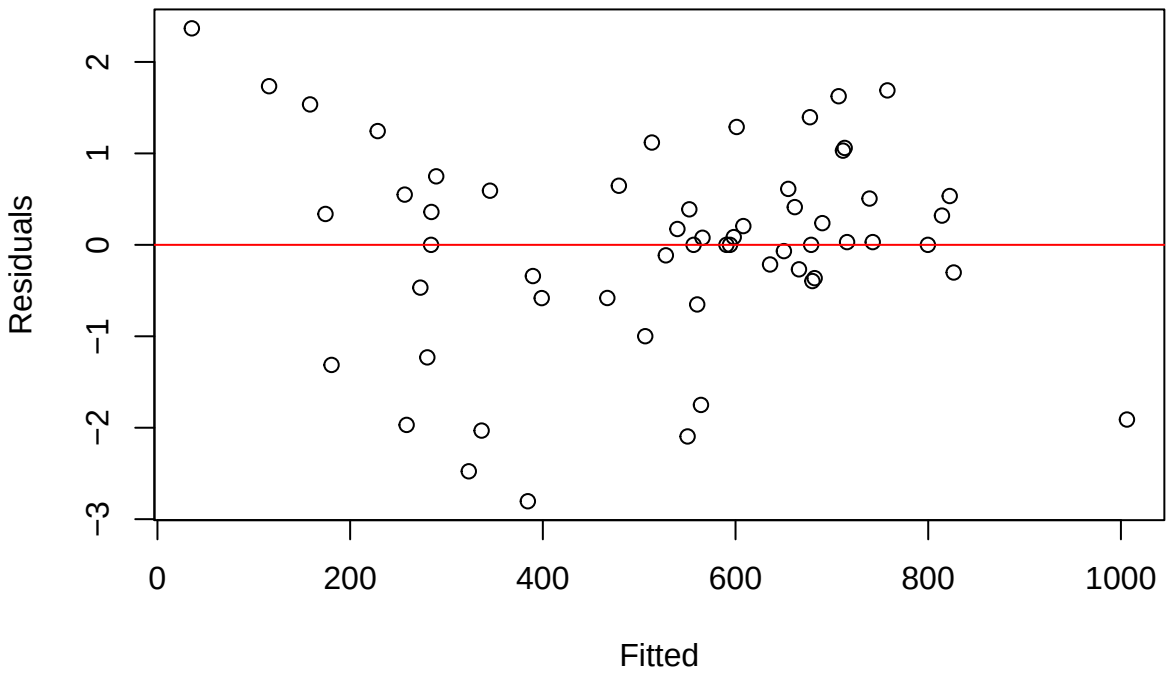



```
plot(fitted(lad), residuals(lad),  xlab="Fitted", ylab="Residuals")
abline(h=0, col="red") # draws a horizontal red line at y = 0
```

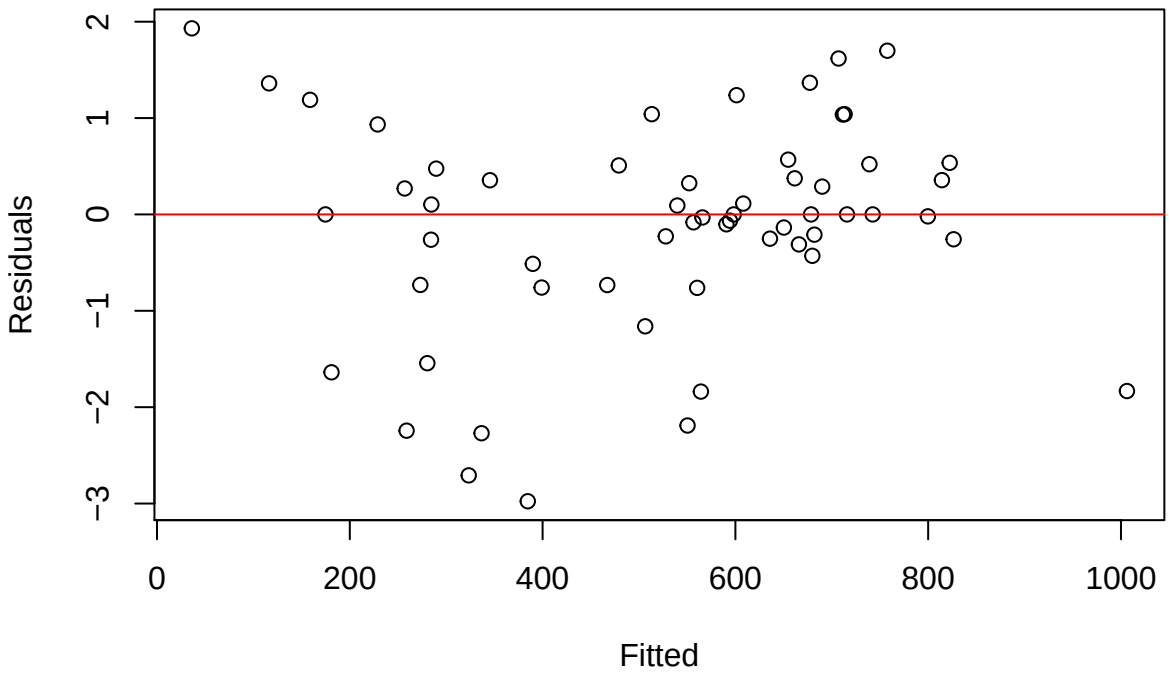



```
library(ggplot2)
lm <- lm(HtVol ~ Male + Age + BMI + BSA, data = mydata)
lad <- lad(HtVol ~ Male + Age + BMI + BSA, data = mydata)

plot(fitted(lm), residuals(lm),  xlab="Fitted", ylab="Residuals")
abline(h=0, col="red") # draws a horizontal red line at y = 0
```




```
plot(fitted(lad), residuals(lad),  xlab="Fitted", ylab="Residuals")
abline(h=0, col="red") # draws a horizontal red line at y = 0
```



2 Question 2

2.1 10 fold CV

```
mydata <- read.csv("https://raw.githubusercontent.com/haowang666/Computational-Stats/mas
library(dplyr)
mydata <- mydata %>%
  select(Smoking, Age, Gender, Diabetes, Death)

mydata2 <- mydata %>%
  mutate_each_(funs(scale(.) %>% as.vector),
                              vars = c("Age"))
```

I pick `Smoking`, `Age`, `Gender`, `Diabetes` as my predictors. I use package `caret` to do 10 fold cross validation. I use package `dplyr` to do a subselection. The first thing I did is to inspect the data scales. I found `Age` is not in the same dimension with other variables. So I did a rescaling first. `mydata2` contains all the preditors I used (age scaled with mean 0 and std 1).

As an example, I pick the number of nearest neighbors as 5.

```
library(class)
set.seed(99)
k = 10
folds <- sample(1:k, nrow(mydata2), replace = TRUE)

mse.knn = rep(0, times = k)
predict.knn = rep(0, times = nrow(mydata2))

for (i in 1:10) {
  pred.knn <- knn(mydata2[folds != i, ],
                  mydata2[folds == i, ],
                  as.factor(mydata2$Death[folds != i]), k = 5)
  print(paste0(i,") error rate: ",
               round(mean(mydata2$Death[folds == i ] != pred.knn),3)))
}
```

```
## [1] "1) error rate: 0.13"
## [1] "2) error rate: 0.05"
## [1] "3) error rate: 0"
## [1] "4) error rate: 0"
## [1] "5) error rate: 0.05"
## [1] "6) error rate: 0.057"
## [1] "7) error rate: 0"
## [1] "8) error rate: 0"
## [1] "9) error rate: 0"
## [1] "10) error rate: 0.043"
```

With respect to logistic regression. I did the similar steps.

```
library(class)
set.seed(99)
k = 10
folds <- sample(1:k, nrow(mydata2), replace = TRUE)

mse.knn = rep(0, times = k)
predict.knn = rep(0, times = nrow(mydata2))

mydata2$Death <- as.factor(mydata2$Death)


for (i in 1:10) {
  glm.fit <- glm(Death ~., data = mydata2[folds != i, ], family = "binomial")
  pred.probs <- predict(glm.fit, mydata2[folds == i, ], type = "response")
  glm.pred <- rep(0, length(pred.probs))
  glm.pred[pred.probs > .5] = 1
```

```
  print(paste0(i,") error rate: ",
               mean(mydata2$Death[folds == i ] != glm.pred)))
}
```

```
## [1] "1) error rate: 0.217391304347826"
## [1] "2) error rate: 0.1"
## [1] "3) error rate: 0.0588235294117647"
## [1] "4) error rate: 0"
## [1] "5) error rate: 0.1"
## [1] "6) error rate: 0.0857142857142857"
## [1] "7) error rate: 0.08"
## [1] "8) error rate: 0.05"
## [1] "9) error rate: 0.0625"
## [1] "10) error rate: 0.130434782608696"
```

2.2 Validation set approach

Again use mydata2 given in the previous section

```
set.seed(1)

#creat a train with 75% of the data
trainRows = sample(1:nrow(mydata2), 0.75*nrow(mydata2))
length(trainRows)
```

```
## [1] 162
```

```
train <- mydata2[trainRows, ]
test <- mydata2[-trainRows, ]
```

The nest step is fitting glm and knn with train

```
#glm fitting
glm.fit <- glm(Death ~., data = train, family = "binomial")
#knn fitting
knn.fit <- knn(train = train, test = test, cl = train$Death, k = 5)
```

The validation error can be calculated through the following

```
  pred.probs <- predict(glm.fit, data = test, type = "response")
  glm.pred <- rep(0, length(pred.probs))
  glm.pred[pred.probs > .5] = 1
  print(paste(" glm error rate: ",
               round(mean(test$Death != glm.pred), 4)))
```

```
## [1] " glm error rate:  0.0185"
```

```
  print(paste(" knn error rate: ",
              mean(test$Death != knn.fit)))
```

```
## [1] " knn error rate:  0"
```

To repeat the process, all I need to change is the seed.

```
set.seed(2)

#creat a train with 75% of the data
trainRows = sample(1:nrow(mydata2), 0.75*nrow(mydata2))
train <- mydata2[trainRows, ]
test <- mydata2[-trainRows, ]

#glm fitting
glm.fit <- glm(Death ~., data = train, family = "binomial")
#knn fitting
knn.fit <- knn(train = train, test = test, cl = train$Death, k = 5)

#glm fitting
glm.fit <- glm(Death ~., data = train, family = "binomial")
#knn fitting
knn.fit <- knn(train = train, test = test, cl = train$Death, k = 5)

#error rate
pred.probs <- predict(glm.fit, data = test, type = "response")
  glm.pred <- rep(0, length(pred.probs))
  glm.pred[pred.probs > .5] = 1
  print(paste(" glm error rate: ",
              round(mean(test$Death != glm.pred), 4)))
```

```
## [1] " glm error rate:  0.1667"
```

```
  print(paste(" knn error rate: ",
              mean(test$Death != knn.fit)))
```

```
## [1] " knn error rate:  0.148148148148148"
```

```
set.seed(3)

#creat a train with 75% of the data
trainRows = sample(1:nrow(mydata2), 0.75*nrow(mydata2))
train <- mydata2[trainRows, ]
test <- mydata2[-trainRows, ]

#glm fitting
glm.fit <- glm(Death ~., data = train, family = "binomial")
```

```r
#knn fitting
knn.fit <- knn(train = train, test = test, cl = train$Death, k = 5)

#glm fitting
glm.fit <- glm(Death ~., data = train, family = "binomial")
#knn fitting
knn.fit <- knn(train = train, test = test, cl = train$Death, k = 5)

#error rate
pred.probs <- predict(glm.fit, data = test, type = "response")
  glm.pred <- rep(0, length(pred.probs))
  glm.pred[pred.probs > .5] = 1
  print(paste(" glm error rate: ",
              round(mean(test$Death != glm.pred), 4)))
```

```
## [1] " glm error rate:  0.0926"
```

```r
  print(paste(" knn error rate: ",
              mean(test$Death != knn.fit)))
```

```
## [1] " knn error rate:  0.0740740740740741"
```

By changing the `seed` alone, I get three different error rates. This indicates that when using different train sets, we may have different predictions over the testing sets. Cross-validation is needed to get an optimized error rate.