Objective Wine Evaluation

KyungjinSohn_SeongjinLee

2021 11 30

```
if(!require(tibble))
        install.packages('tibble', repos = "http://cran.us.r-project.org")
if(!require(dplyr))
        install.packages('dplyr', repos = "http://cran.us.r-project.org")
if(!require(tidyr))
        install.packages('tidyr', repos = "http://cran.us.r-project.org")
if(!require(ggplot2))
        install.packages('ggplot2', repos = "http://cran.us.r-project.org")
if(!require(leaps))
        install.packages('leaps', repos = "http://cran.us.r-project.org")
if(!require(pls))
        install.packages('pls', repos = "http://cran.us.r-project.org")
if(!require(glmnet))
        install.packages('glmnet', repos = "http://cran.us.r-project.org")
library(tibble)
library(dplyr)
library(tidyr)
library(ggplot2)
library(leaps)
library(pls)
library(glmnet)
```

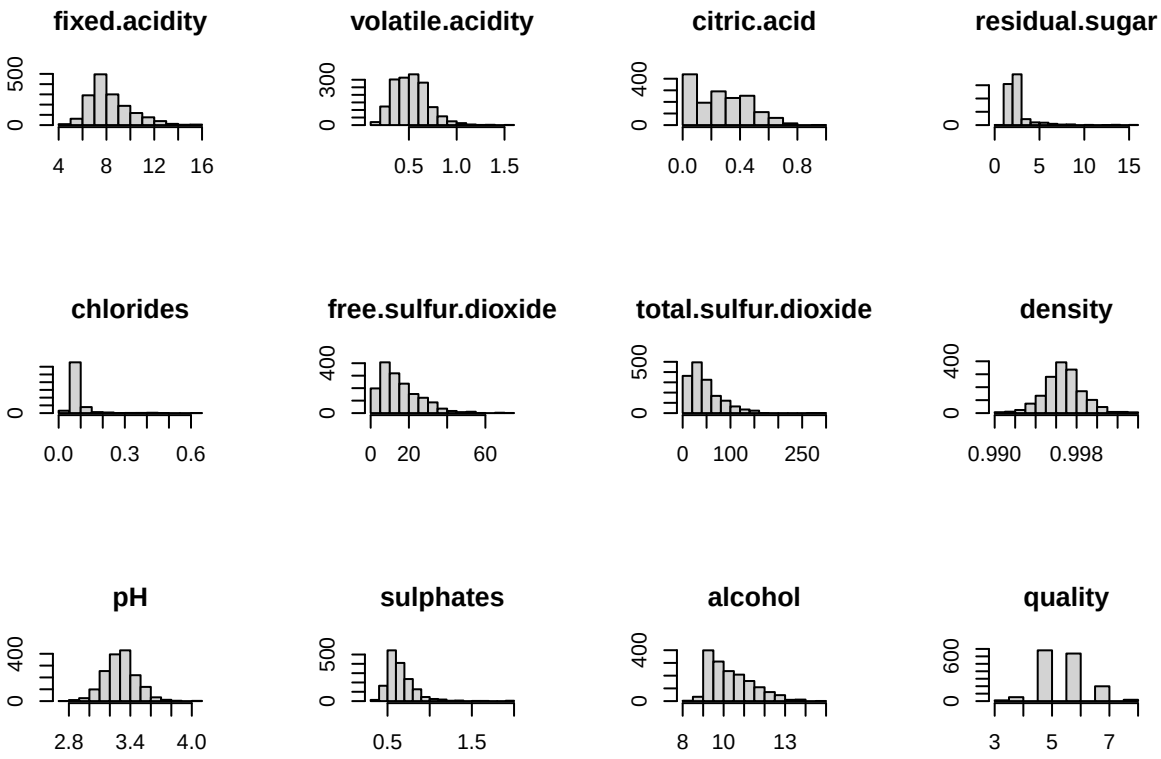
1. Overview of Data

```
red = read.csv("../data/winequality-red.csv", header = TRUE, sep = ";")
# white = read.csv("../data/winequality-white.csv", header = TRUE, sep = ";")

# 1) summary of the data
summary(red)
```

```
##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
```

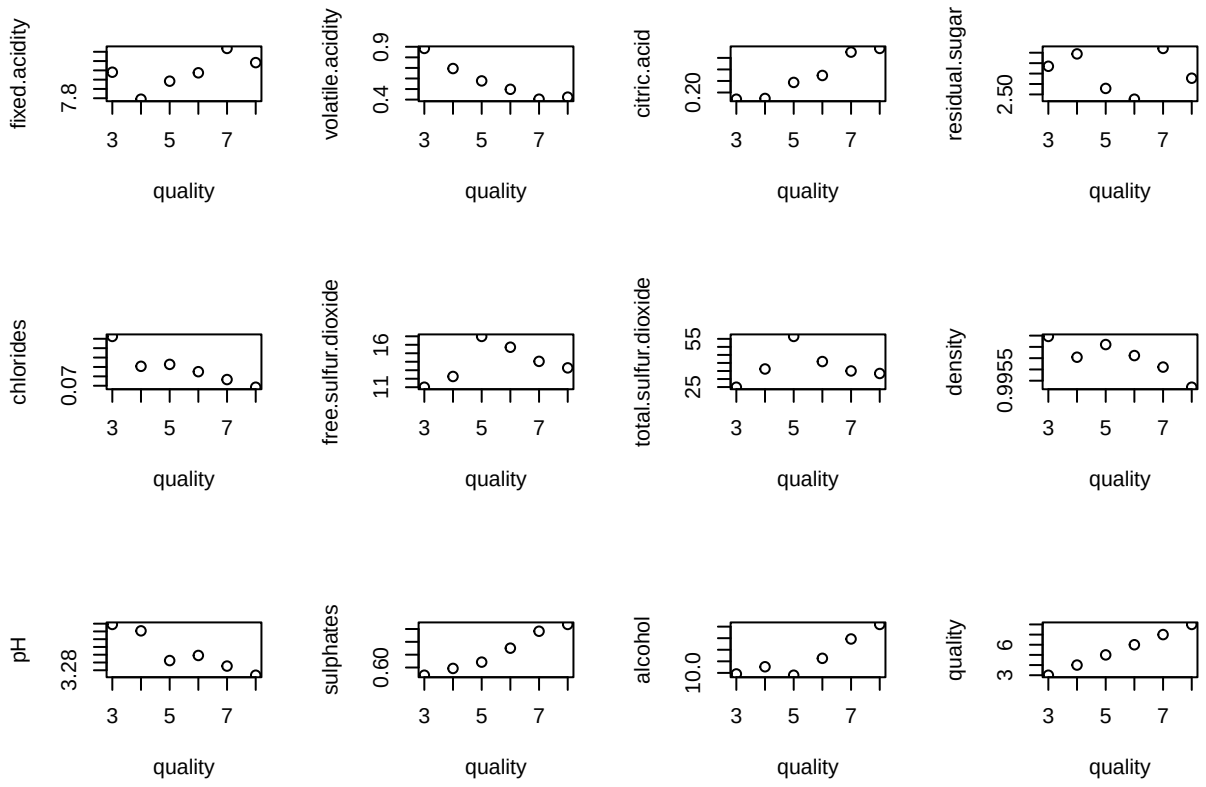
```
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
##        pH          sulphates         alcohol         quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
```

```r
# 2) histogram: distribution of each variable
varnames = colnames(red)

par(mfrow = c(3, 4))

for(name in varnames){
        hist(red[[name]], main = name, xlab = NULL, ylab = NULL )
}
```

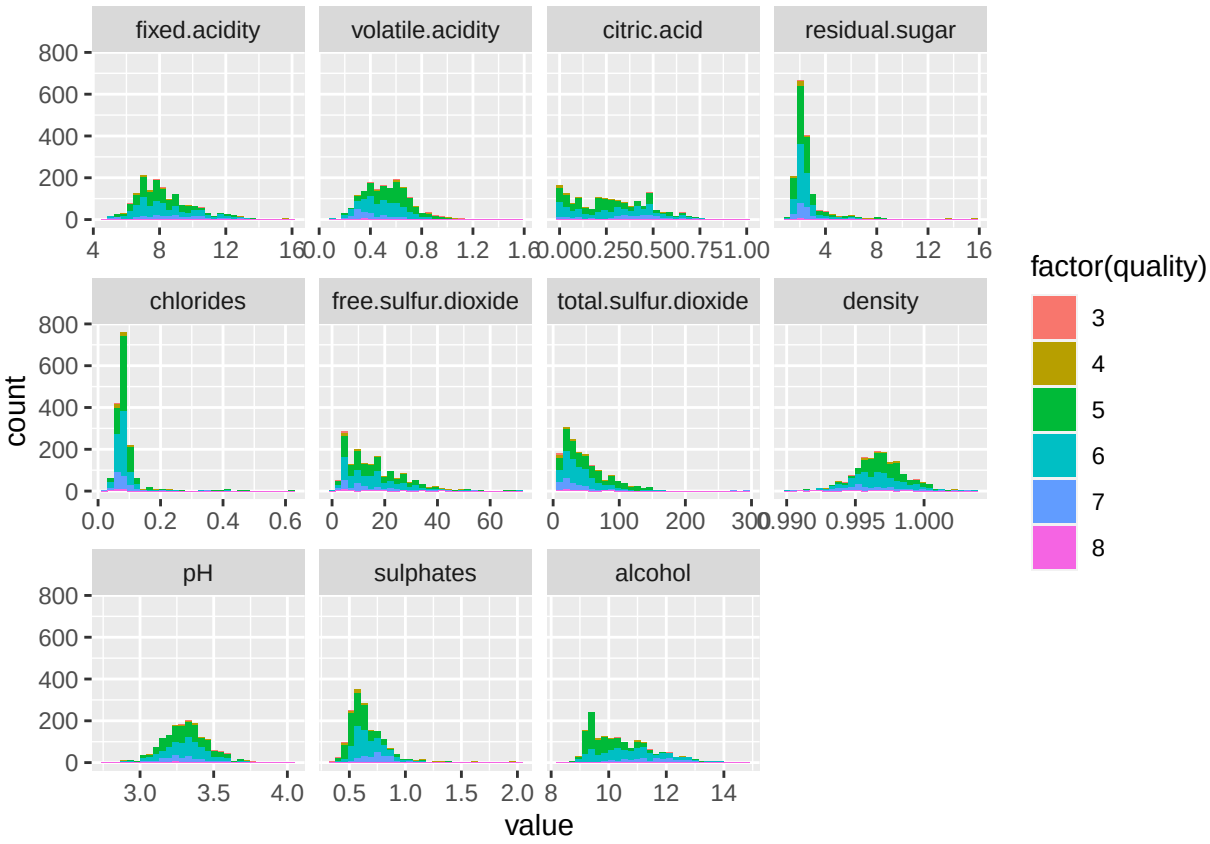



```r
# 3) scatter plot:
#       average distribution of each variable based on the quality
mean_summary = red %>% group_by(quality) %>% summarise_all(mean)

par(mfrow = c(3, 4))

for(name in varnames){
```

```
        plot(mean_summary$quality,
             mean_summary[[name]],
             xlab = "quality",
             ylab = name
        )
}
```




```
# 4) histogram: distribution of each variable grouped by quality
ggplot(gather(red, key, value, -c(quality), factor_key = TRUE),
       aes(value, fill = factor(quality))) +
        geom_histogram(bins = 30) +
        facet_wrap(~ key, scales = 'free_x')
```

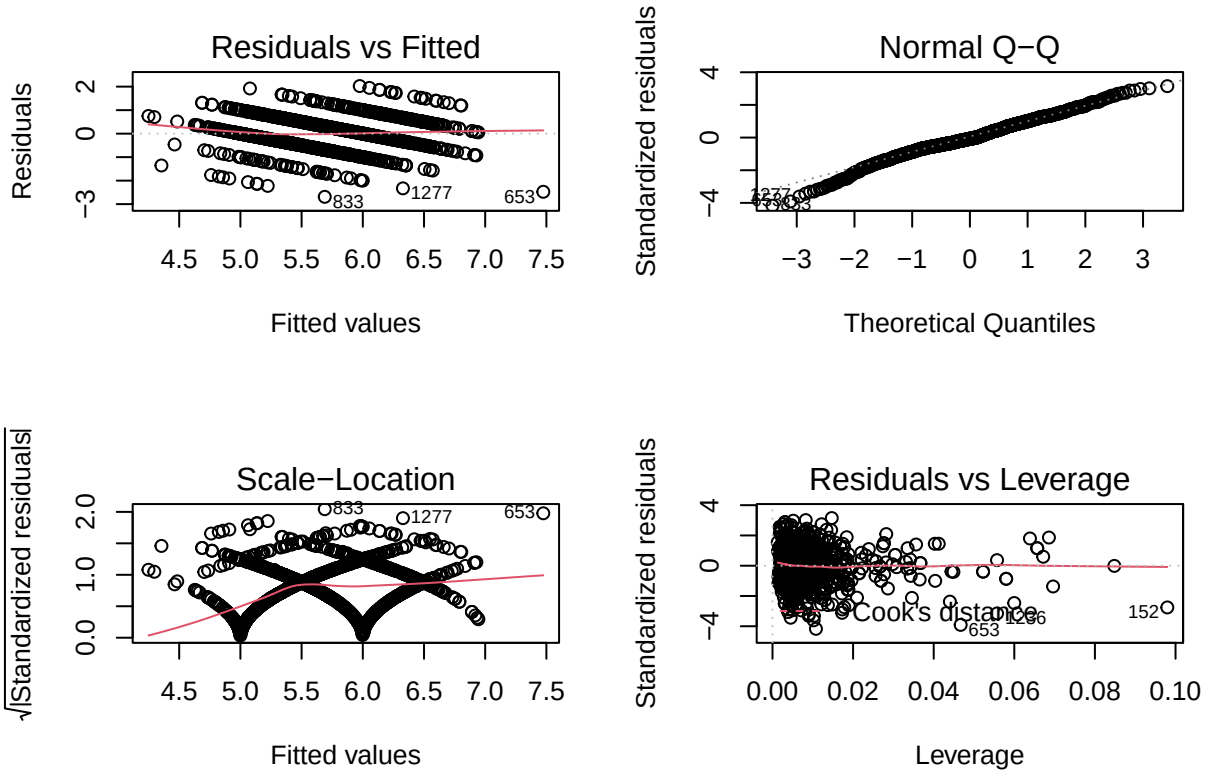



```
# 5) linear regression analysis of original data
lm_ori = lm(quality ~ ., red)
summary(lm_ori)
```

```
##
## Call:
## lm(formula = quality ~ ., data = red)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.68911 -0.36652 -0.04699  0.45202  2.02498
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.197e+01  2.119e+01   1.036   0.3002
## fixed.acidity         2.499e-02  2.595e-02   0.963   0.3357
## volatile.acidity     -1.084e+00  1.211e-01  -8.948  < 2e-16 ***
## citric.acid          -1.826e-01  1.472e-01  -1.240   0.2150
## residual.sugar        1.633e-02  1.500e-02   1.089   0.2765
## chlorides            -1.874e+00  4.193e-01  -4.470 8.37e-06 ***
## free.sulfur.dioxide   4.361e-03  2.171e-03   2.009   0.0447 *
## total.sulfur.dioxide -3.265e-03  7.287e-04  -4.480 8.00e-06 ***
## density              -1.788e+01  2.163e+01  -0.827   0.4086
## pH                   -4.137e-01  1.916e-01  -2.159   0.0310 *
## sulphates             9.163e-01  1.143e-01   8.014 2.13e-15 ***
## alcohol               2.762e-01  2.648e-02  10.429  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.648 on 1587 degrees of freedom
## Multiple R-squared:  0.3606, Adjusted R-squared:  0.3561
## F-statistic: 81.35 on 11 and 1587 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))
plot(lm_ori)
```




```
# 6) Check multicolinearity
cor(red)
```

```
##                       fixed.acidity volatile.acidity citric.acid residual.sugar
## fixed.acidity            1.00000000     -0.256130895  0.67170343    0.114776724
## volatile.acidity        -0.25613089      1.000000000 -0.55249568    0.001917882
## citric.acid              0.67170343     -0.552495685  1.00000000    0.143577162
## residual.sugar           0.11477672      0.001917882  0.14357716    1.000000000
## chlorides                0.09370519      0.061297772  0.20382291    0.055609535
## free.sulfur.dioxide     -0.15379419     -0.010503827 -0.06097813    0.187048995
## total.sulfur.dioxide    -0.11318144      0.076470005  0.03553302    0.203027882
## density                  0.66804729      0.022026232  0.36494718    0.355283371
## pH                      -0.68297819      0.234937294 -0.54190414   -0.085652422
## sulphates                0.18300566     -0.260986685  0.31277004    0.005527121
## alcohol                 -0.06166827     -0.202288027  0.10990325    0.042075437
## quality                  0.12405165     -0.390557780  0.22637251    0.013731637
##                         chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity         0.093705186        -0.153794193          -0.11318144
```

```
## volatile.acidity      0.061297772        -0.010503827           0.07647000
## citric.acid           0.203822914        -0.060978129           0.03553302
## residual.sugar        0.055609535         0.187048995           0.20302788
## chlorides             1.000000000         0.005562147           0.04740047
## free.sulfur.dioxide   0.005562147         1.000000000           0.66766645
## total.sulfur.dioxide  0.047400468         0.667666450           1.00000000
## density               0.200632327        -0.021945831           0.07126948
## pH                   -0.265026131         0.070377499          -0.06649456
## sulphates             0.371260481         0.051657572           0.04294684
## alcohol              -0.221140545        -0.069408354          -0.20565394
## quality              -0.128906560        -0.050656057          -0.18510029
##                         density          pH    sulphates     alcohol
## fixed.acidity         0.66804729 -0.68297819  0.183005664 -0.06166827
## volatile.acidity      0.02202623  0.23493729 -0.260986685 -0.20228803
## citric.acid           0.36494718 -0.54190414  0.312770044  0.10990325
## residual.sugar        0.35528337 -0.08565242  0.005527121  0.04207544
## chlorides             0.20063233 -0.26502613  0.371260481 -0.22114054
## free.sulfur.dioxide  -0.02194583  0.07037750  0.051657572 -0.06940835
## total.sulfur.dioxide  0.07126948 -0.06649456  0.042946836 -0.20565394
## density               1.00000000 -0.34169933  0.148506412 -0.49617977
## pH                   -0.34169933  1.00000000 -0.196647602  0.20563251
## sulphates             0.14850641 -0.19664760  1.000000000  0.09359475
## alcohol              -0.49617977  0.20563251  0.093594750  1.00000000
## quality              -0.17491923 -0.05773139  0.251397079  0.47616632
##                         quality
## fixed.acidity         0.12405165
## volatile.acidity     -0.39055778
## citric.acid           0.22637251
## residual.sugar        0.01373164
## chlorides            -0.12890656
## free.sulfur.dioxide  -0.05065606
## total.sulfur.dioxide -0.18510029
## density              -0.17491923
## pH                   -0.05773139
## sulphates             0.25139708
## alcohol               0.47616632
## quality               1.00000000
```

2. Transformations

```
colnames(red)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

```
logred = red
# 1) add 11 new columns that transform each column data into log(data)
logred$logFA = log(red$fixed.acidity)
logred$logVA = log(red$volatile.acidity)
```

```
logred$logCA = red$citric.acid
logred$logRS = log(red$residual.sugar)
logred$logCL = log(red$chlorides)
logred$logFS = log(red$free.sulfur.dioxide)
logred$logTS = log(red$total.sulfur.dioxide)
logred$logDE = log(red$density)
logred$logPH = red$pH
logred$logSP = log(red$sulphates)
logred$logAL = log(red$alcohol)

# 2) compare the AIC (lm_ori vs lm using the log(data) as a predictor)
lm_log = lm(quality ~ ., logred[,12:23])

comp_AIC = data.frame(lm = c("Ori.", "Log."),
                      AIC = c(AIC(lm_ori), AIC(lm_log)))
comp_AIC$AIC = round(comp_AIC$AIC, 3)
comp_AIC
```

```
##     lm      AIC
## 1 Ori. 3164.277
## 2 Log. 3154.790
```

```
# 3) select the predictor to use with the log transformation.
boolExpand = expand.grid(c(0,1), c(0,1), c(0,1), c(0,1), c(0,1),
                         c(0,1), c(0,1), c(0,1), c(0,1))
boolExpand = cbind(boolExpand[,1:2], 0, boolExpand[,3:7], 0, boolExpand[,8:9])

df1 = logred[,1:12] # ori. data
lm1 = lm(quality ~ ., df1) # lm model
aic1 = AIC(lm1) # ori. AIC

for (i in 1:nrow(boolExpand)) {
        bool = as.numeric(boolExpand[i,])
        df_trans = logred[, c(bool*12 + 1:11, 12)]

        lm2 = lm(quality ~ ., df_trans)
        aic2 = AIC(lm2)
        if(aic2 < aic1){
                df1 = df_trans
                lm1 = lm2
                aic1 = aic2
                print(paste0('switching ', i))
        }
}
```

```
## [1] "switching 2"
## [1] "switching 6"
## [1] "switching 17"
## [1] "switching 18"
## [1] "switching 129"
## [1] "switching 130"
## [1] "switching 134"
## [1] "switching 145"
```

```
## [1] "switching 146"
## [1] "switching 150"
## [1] "switching 214"
```

```
head(df1)
```

```
##      logFA volatile.acidity citric.acid     logRS chlorides    logFS
## 1 2.001480             0.70        0.00 0.6418539     0.076 2.397895
## 2 2.054124             0.88        0.00 0.9555114     0.098 3.218876
## 3 2.054124             0.76        0.04 0.8329091     0.092 2.708050
## 4 2.415914             0.28        0.56 0.6418539     0.075 2.833213
## 5 2.001480             0.70        0.00 0.6418539     0.076 2.397895
## 6 2.001480             0.66        0.00 0.5877867     0.075 2.564949
##   total.sulfur.dioxide        logDE   pH      logSP alcohol quality
## 1                   34 -0.002202424 3.51 -0.5798185     9.4       5
## 2                   67 -0.003205131 3.20 -0.3856625     9.8       5
## 3                   54 -0.003004509 3.26 -0.4307829     9.8       5
## 4                   60 -0.002002003 3.16 -0.5447272     9.8       6
## 5                   34 -0.002202424 3.51 -0.5798185     9.4       5
## 6                   40 -0.002202424 3.51 -0.5798185     9.4       5
```

```
names(df1) # final columns selected
```

```
##  [1] "logFA"               "volatile.acidity"    "citric.acid"
##  [4] "logRS"               "chlorides"           "logFS"
##  [7] "total.sulfur.dioxide" "logDE"              "pH"
## [10] "logSP"               "alcohol"             "quality"
```

```
# 4) compare the AIC (lm_ori vs lm_log vs final model after transformation)
comp_AIC = rbind(comp_AIC, c("Trans.", round(AIC(lm1), 3)))
comp_AIC
```

```
##       lm      AIC
## 1   Ori. 3164.277
## 2   Log.  3154.79
## 3 Trans.   3127.7
```

```
# 5) Map dependent variable to the real line.
logitlm = lm(log(quality/(11-quality)) ~., data = df1)

summary(logitlm)
```

```
##
## Call:
## lm(formula = log(quality/(11 - quality)) ~ ., data = df1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.06846 -0.13385 -0.01661  0.15906  0.79613
##
## Coefficients:
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -5.045e-01  3.712e-01  -1.359  0.17434
## logFA                 1.664e-01  8.524e-02   1.952  0.05106 .
## volatile.acidity     -3.760e-01  4.487e-02  -8.380  < 2e-16 ***
## citric.acid          -7.908e-02  5.418e-02  -1.460  0.14455
## logRS                 4.823e-02  2.365e-02   2.039  0.04161 *
## chlorides            -7.059e-01  1.523e-01  -4.635 3.86e-06 ***
## logFS                 3.255e-02  1.231e-02   2.643  0.00829 **
## total.sulfur.dioxide -1.297e-03  2.735e-04  -4.740 2.33e-06 ***
## logDE                -1.890e+01  8.461e+00  -2.234  0.02563 *
## pH                   -1.310e-01  7.319e-02  -1.790  0.07363 .
## logSP                 3.151e-01  3.244e-02   9.713  < 2e-16 ***
## alcohol               8.989e-02  1.021e-02   8.801  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2398 on 1587 degrees of freedom
## Multiple R-squared:  0.373,  Adjusted R-squared:  0.3686
## F-statistic: 85.81 on 11 and 1587 DF,  p-value: < 2.2e-16
```

```
summary(lm1)
```

```
##
## Call:
## lm(formula = quality ~ ., data = df_trans)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7410 -0.3635 -0.0474  0.4389  2.0094
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.050e+00  9.918e-01   4.083 4.66e-05 ***
## logFA                 4.622e-01  2.277e-01   2.030  0.04255 *
## volatile.acidity     -1.007e+00  1.199e-01  -8.399  < 2e-16 ***
## citric.acid          -2.166e-01  1.447e-01  -1.496  0.13476
## logRS                 1.301e-01  6.319e-02   2.059  0.03963 *
## chlorides            -1.868e+00  4.069e-01  -4.590 4.79e-06 ***
## logFS                 8.940e-02  3.290e-02   2.717  0.00666 **
## total.sulfur.dioxide -3.559e-03  7.308e-04  -4.869 1.23e-06 ***
## logDE                -5.111e+01  2.260e+01  -2.261  0.02388 *
## pH                   -3.331e-01  1.955e-01  -1.703  0.08870 .
## logSP                 8.455e-01  8.668e-02   9.755  < 2e-16 ***
## alcohol               2.408e-01  2.729e-02   8.824  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6406 on 1587 degrees of freedom
## Multiple R-squared:  0.375,  Adjusted R-squared:  0.3707
## F-statistic: 86.57 on 11 and 1587 DF,  p-value: < 2.2e-16
```

```
AIC_correction = sum(log(11/df1$quality/(11-df1$quality))) *2
```

```
comp_AIC = rbind(comp_AIC, c("Logit.", round(AIC(logitlm) - AIC_correction, 3)))
comp_AIC
```

```
##       lm      AIC
## 1   Ori. 3164.277
## 2   Log.  3154.79
## 3 Trans.   3127.7
## 4 Logit. 3146.288
```

3. Variable Selection

```
colnames(df1)
```

```
##  [1] "logFA"                "volatile.acidity"     "citric.acid"
##  [4] "logRS"                "chlorides"            "logFS"
##  [7] "total.sulfur.dioxide" "logDE"                "pH"
## [10] "logSP"                "alcohol"              "quality"
```

```
quadred = df1
# 1) add 11 new columns that transform each column data into (data)^2
quadred$quadFA = (quadred$logFA)^2
quadred$quadVA = (quadred$volatile.acidity)^2
quadred$quadCA = (quadred$citric.acid)^2
quadred$quadRS = (quadred$logRS)^2
quadred$quadCL = (quadred$chlorides)^2
quadred$quadFS = (quadred$logFS)^2
quadred$quadTS = (quadred$total.sulfur.dioxide)^2
quadred$quadDE = (quadred$logDE)^2
quadred$quadPH = (quadred$pH)^2
quadred$quadSP = (quadred$logSP)^2
quadred$quadAL = (quadred$alcohol)^2


lm_trans = lm(quality ~ ., quadred)
summary(lm_trans)
```

```
##
## Call:
## lm(formula = quality ~ ., data = quadred)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.70481 -0.38031 -0.01694  0.42765  1.99261
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -2.209e+00  5.280e+00  -0.418  0.67568
## logFA                 5.032e+00  1.777e+00   2.832  0.00469 **
## volatile.acidity     -6.082e-01  4.271e-01  -1.424  0.15460
## citric.acid          -5.141e-01  3.197e-01  -1.608  0.10808
```

```
## logRS                 1.685e-01  1.864e-01   0.904  0.36612
## chlorides            -1.460e+00  1.103e+00  -1.323  0.18594
## logFS                 2.745e-01  1.460e-01   1.880  0.06028 .
## total.sulfur.dioxide -5.568e-03  1.768e-03  -3.149  0.00167 **
## logDE                -7.760e+00  3.047e+01  -0.255  0.79897
## pH                    4.691e-01  3.404e+00   0.138  0.89040
## logSP                 1.009e-01  1.690e-01   0.597  0.55062
## alcohol               2.772e-01  2.749e-01   1.009  0.31330
## quadFA               -1.098e+00  4.173e-01  -2.631  0.00860 **
## quadVA               -2.763e-01  3.273e-01  -0.844  0.39863
## quadCA                5.996e-01  4.741e-01   1.265  0.20612
## quadRS               -2.873e-02  7.263e-02  -0.396  0.69246
## quadCL               -2.501e-01  2.407e+00  -0.104  0.91727
## quadFS               -3.715e-02  2.943e-02  -1.262  0.20711
## quadTS                1.496e-05  9.375e-06   1.596  0.11080
## quadDE                7.943e+03  3.560e+03   2.231  0.02583 *
## quadPH               -1.415e-01  5.105e-01  -0.277  0.78171
## quadSP               -1.135e+00  2.071e-01  -5.480 4.95e-08 ***
## quadAL               -2.508e-03  1.272e-02  -0.197  0.84372
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6331 on 1576 degrees of freedom
## Multiple R-squared:  0.3938, Adjusted R-squared:  0.3853
## F-statistic: 46.54 on 22 and 1576 DF,  p-value: < 2.2e-16
```

```
# 1) AIC
require(leaps)
b = regsubsets(quality ~ ., quadred, nvmax = 22)
rs = summary(b)

rs$which
```

```
##    (Intercept) logFA volatile.acidity citric.acid logRS chlorides logFS
## 1         TRUE FALSE            FALSE       FALSE FALSE     FALSE FALSE
## 2         TRUE FALSE             TRUE       FALSE FALSE     FALSE FALSE
## 3         TRUE FALSE             TRUE       FALSE FALSE     FALSE FALSE
## 4         TRUE FALSE             TRUE       FALSE FALSE     FALSE FALSE
## 5         TRUE FALSE             TRUE       FALSE FALSE      TRUE FALSE
## 6         TRUE FALSE             TRUE       FALSE FALSE      TRUE FALSE
## 7         TRUE FALSE             TRUE       FALSE FALSE      TRUE  TRUE
## 8         TRUE FALSE             TRUE       FALSE FALSE      TRUE  TRUE
## 9         TRUE FALSE             TRUE       FALSE FALSE      TRUE  TRUE
## 10        TRUE  TRUE             TRUE       FALSE FALSE      TRUE  TRUE
## 11        TRUE  TRUE            FALSE       FALSE FALSE      TRUE  TRUE
## 12        TRUE  TRUE            FALSE       FALSE  TRUE      TRUE  TRUE
## 13        TRUE  TRUE             TRUE       FALSE  TRUE      TRUE  TRUE
## 14        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
## 15        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
## 16        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
## 17        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
## 18        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
## 19        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
## 20        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
```

```
## 21        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
## 22        TRUE  TRUE             TRUE        TRUE  TRUE      TRUE  TRUE
##    total.sulfur.dioxide logDE    pH logSP alcohol quadFA quadVA quadCA quadRS
## 1                 FALSE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 2                 FALSE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 3                 FALSE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 4                 FALSE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 5                 FALSE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 6                  TRUE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 7                  TRUE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 8                  TRUE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 9                  TRUE FALSE FALSE FALSE    TRUE  FALSE  FALSE  FALSE  FALSE
## 10                 TRUE FALSE FALSE FALSE    TRUE   TRUE  FALSE  FALSE  FALSE
## 11                 TRUE FALSE FALSE FALSE    TRUE   TRUE   TRUE  FALSE  FALSE
## 12                 TRUE FALSE FALSE FALSE    TRUE   TRUE   TRUE  FALSE  FALSE
## 13                 TRUE FALSE FALSE FALSE    TRUE   TRUE  FALSE  FALSE  FALSE
## 14                 TRUE FALSE FALSE FALSE    TRUE   TRUE  FALSE  FALSE  FALSE
## 15                 TRUE FALSE FALSE FALSE    TRUE   TRUE  FALSE   TRUE  FALSE
## 16                 TRUE FALSE FALSE FALSE    TRUE   TRUE   TRUE   TRUE  FALSE
## 17                 TRUE FALSE FALSE  TRUE    TRUE   TRUE   TRUE   TRUE  FALSE
## 18                 TRUE FALSE FALSE  TRUE    TRUE   TRUE   TRUE   TRUE   TRUE
## 19                 TRUE  TRUE FALSE  TRUE    TRUE   TRUE   TRUE   TRUE   TRUE
## 20                 TRUE  TRUE FALSE  TRUE    TRUE   TRUE   TRUE   TRUE   TRUE
## 21                 TRUE  TRUE  TRUE  TRUE    TRUE   TRUE   TRUE   TRUE   TRUE
## 22                 TRUE  TRUE  TRUE  TRUE    TRUE   TRUE   TRUE   TRUE   TRUE
##    quadCL quadFS quadTS quadDE quadPH quadSP quadAL
## 1   FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
## 2   FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
## 3   FALSE  FALSE  FALSE  FALSE  FALSE   TRUE  FALSE
## 4   FALSE  FALSE  FALSE  FALSE   TRUE   TRUE  FALSE
## 5   FALSE  FALSE  FALSE  FALSE   TRUE   TRUE  FALSE
## 6   FALSE  FALSE  FALSE  FALSE   TRUE   TRUE  FALSE
## 7   FALSE  FALSE  FALSE  FALSE   TRUE   TRUE  FALSE
## 8   FALSE  FALSE  FALSE   TRUE   TRUE   TRUE  FALSE
## 9   FALSE  FALSE   TRUE   TRUE   TRUE   TRUE  FALSE
## 10  FALSE  FALSE  FALSE   TRUE   TRUE   TRUE  FALSE
## 11  FALSE  FALSE   TRUE   TRUE   TRUE   TRUE  FALSE
## 12  FALSE  FALSE   TRUE   TRUE   TRUE   TRUE  FALSE
## 13  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE  FALSE
## 14  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE  FALSE
## 15  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE  FALSE
## 16  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE  FALSE
## 17  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE  FALSE
## 18  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE  FALSE
## 19  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE  FALSE
## 20  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
## 21  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
## 22   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
```

```r
par(mfrow = c(1, 3))

p = 22
eval_aic = nrow(red)*log(rs$rss/nrow(red)) + (2:(p+1))*2
plot(eval_aic ~ I(1:p),
```

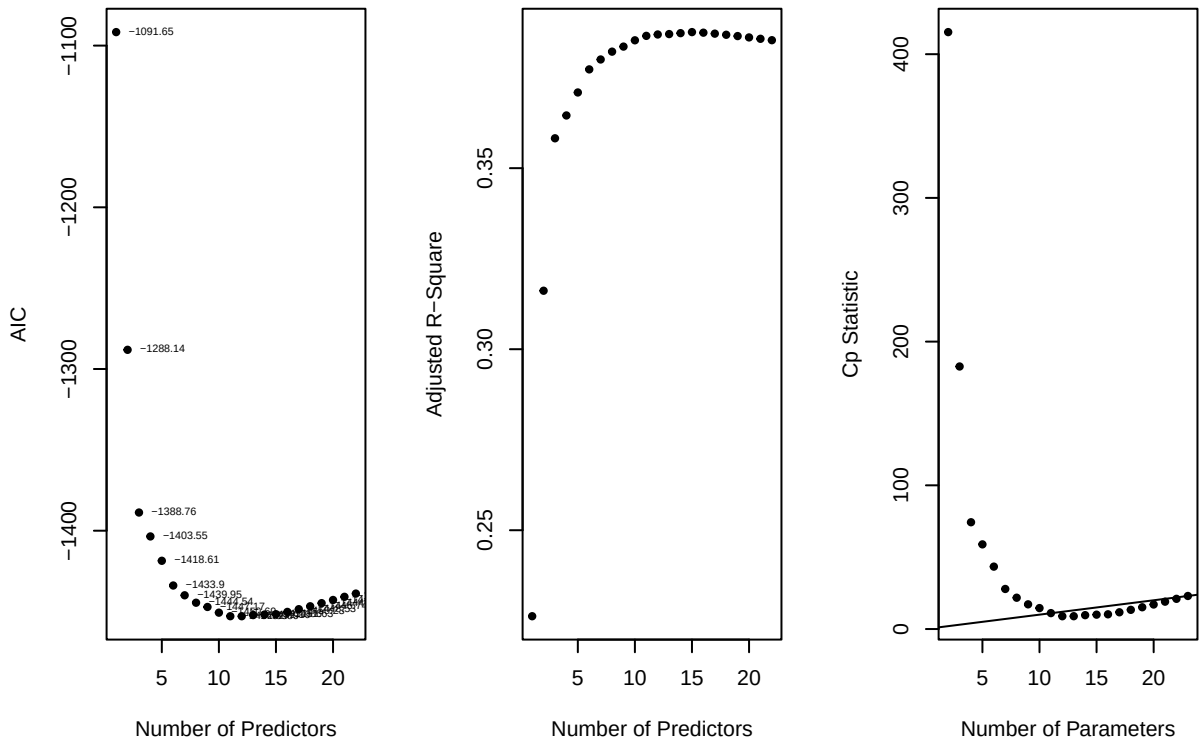
```
     ylab = 'AIC',
     xlab = 'Number of Predictors', pch = 20)
text(I(1:p), eval_aic, round(eval_aic, 2), cex = 0.5, pos = 4)
paste("min AIC: ", which.min(eval_aic))
```

```
## [1] "min AIC:  12"
```

```
min_aic = which.min(eval_aic)
boolCol_step = as.vector(rs$which[min_aic, -1]) # columns selected by min AIC

# 2) Adjusted R-Squre
plot(1:p, rs$adjr2,
     xlab = 'Number of Predictors',
     ylab='Adjusted R-Square', pch = 20)
paste("max Adjusted R-Squre: ", which.max(rs$adjr2))
```

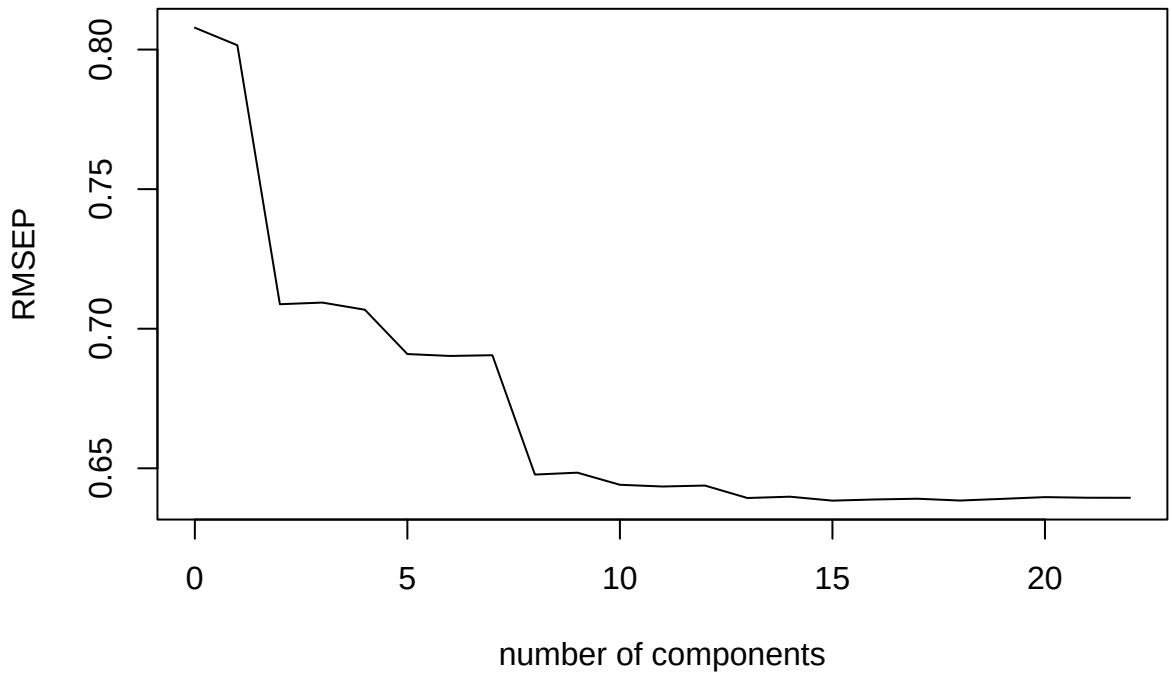
```
## [1] "max Adjusted R-Squre:  15"
```

```
# 3) Mellow's Cp
plot(2:(p+1), rs$cp,
     xlab = 'Number of Parameters',
     ylab = 'Cp Statistic', pch = 20)
abline(0,1)
```

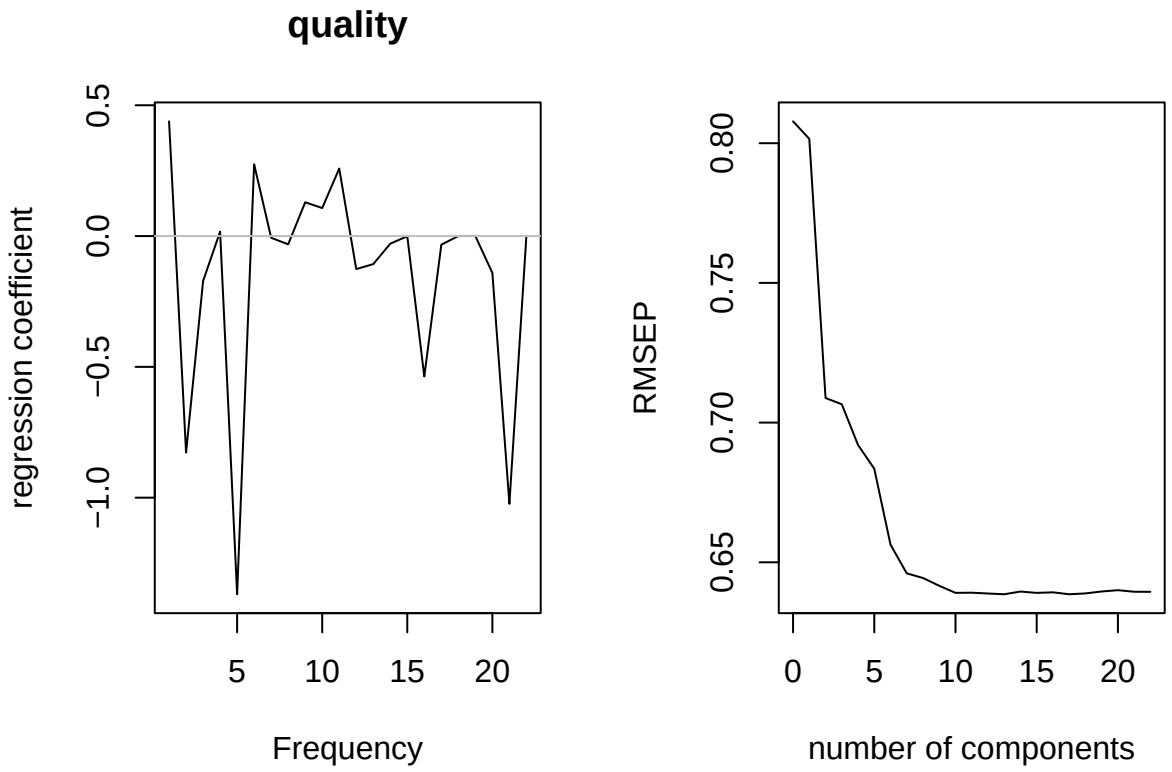



```
par(mfrow = c(1, 1))

# 4) Principal component regression
require(pls)
```

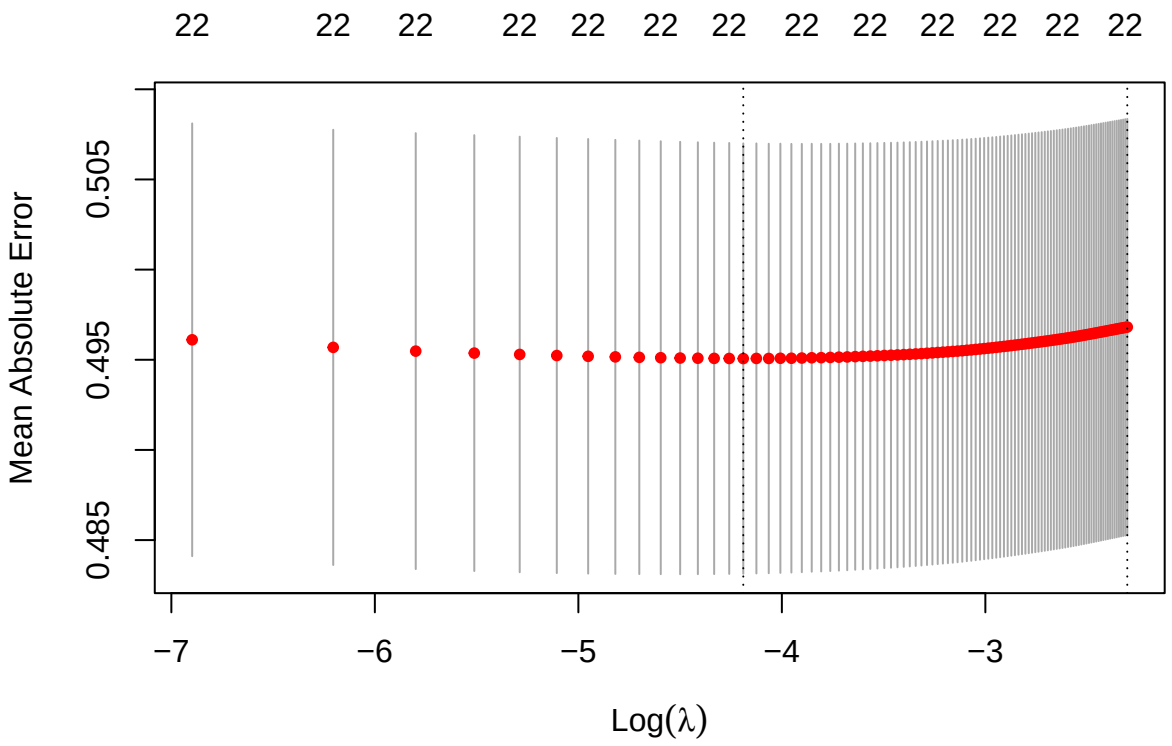
```r
set.seed(664)
lm_pcr = pcr(quality ~ ., data = quadred, validation = "CV", ncomp = 22)
cv_pcr = RMSEP(lm_pcr, estimate = "CV")

min_pcr = which.min(cv_pcr$val) - 1

plot(cv_pcr, main = "")
```




```r
# 5) Partial least squares regression
set.seed(664)
lm_pls = plsr(quality ~ ., data = quadred, validation = "CV", ncomp = 22)
cv_pls = RMSEP(lm_pls, estimate = "CV")

min_pls = which.min(cv_pls$val) -1

par(mfrow = c(1,2))
coefplot(lm_pls, ncomp = min_pls, xlab = "Frequency")
plot(cv_pls, main = "")
```



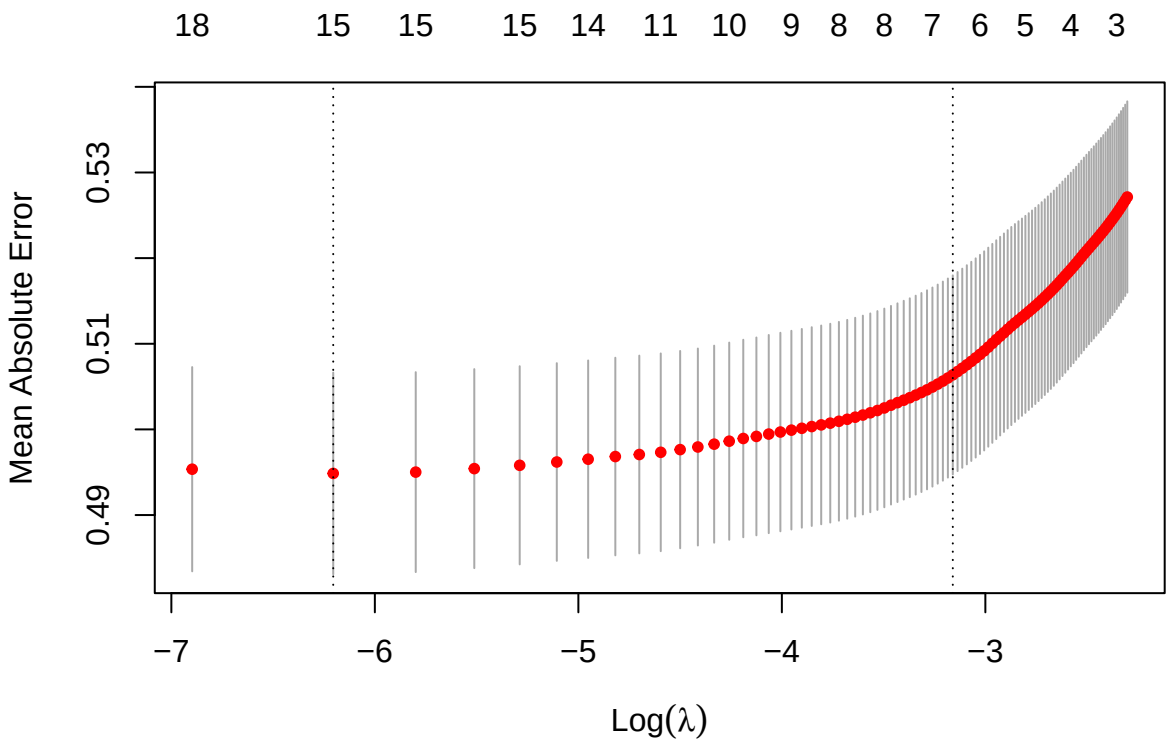

```
par(mfrow = c(1,1))


# 6) Ridge
set.seed(664)
cv_ridge = cv.glmnet(as.matrix(quadred[,-12]), quadred[,12],
                     alpha = 0, type.measure = 'mae',
                     lambda = seq(0, 0.1, len = 100))
plot(cv_ridge)
```




```
opt_lambda_ridge = cv_ridge$lambda.min
lm_ridge = cv_ridge$glmnet.fit

opt_lambda_ridge # optimum lambda by ridge regression
```

```
## [1] 0.01515152
```

```
# 7) LASSO
set.seed(664)
cv_lasso = cv.glmnet(as.matrix(quadred[,-12]), quadred[,12],
                     alpha = 1, type.measure = 'mae',
                     lambda = seq(0, 0.1, len = 100))
plot(cv_lasso)
```




```
opt_lambda_lasso = cv_lasso$lambda.min
lm_lasso = cv_lasso$glmnet.fit

opt_lambda_lasso # optimum lambda by LASSO
```

```
## [1] 0.002020202
```

```
lasso_beta = lm_lasso$beta[,which(lm_lasso$lambda == opt_lambda_lasso)]
boolCol_lasso = as.vector(lasso_beta != 0) # column selected by LASSO

# 8) comparing final model selected using AIC vs LASSO
find_hierarchical <- function(boolCol, p){
        return(c(boolCol[1:p]|boolCol[(p+1):(2*p)], T, boolCol[(p+1):(2*p)]))
}

boolCol_step = find_hierarchical(boolCol_step, 11)
boolCol_lasso = find_hierarchical(boolCol_lasso, 11)

df_step_hier = quadred[,boolCol_step]
df_lasso_hier = quadred[,boolCol_lasso]

lm_step_hier = lm(quality ~., data = df_step_hier)
lm_lasso_hier = lm(quality ~., data = df_lasso_hier)

summary(lm_step_hier)
```

```
##
## Call:
## lm(formula = quality ~ ., data = df_step_hier)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.76915 -0.38617 -0.01881  0.42996  2.00074
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.103e+00  5.064e+00  -0.218 0.827688
## logFA                 4.533e+00  1.707e+00   2.656 0.007984 **
## volatile.acidity     -3.375e-01  3.913e-01  -0.863 0.388526
## logRS                 9.094e-02  6.413e-02   1.418 0.156382
## chlorides            -1.587e+00  4.019e-01  -3.948 8.23e-05 ***
## logFS                 1.068e-01  3.574e-02   2.989 0.002845 **
## total.sulfur.dioxide -6.278e-03  1.664e-03  -3.774 0.000166 ***
## logDE                -1.022e+01  2.998e+01  -0.341 0.733337
## pH                    3.490e-01  3.357e+00   0.104 0.917228
## logSP                 1.077e-01  1.647e-01   0.654 0.513094
## alcohol               2.246e-01  2.881e-02   7.798 1.13e-14 ***
## quadFA               -9.943e-01  3.989e-01  -2.493 0.012769 *
## quadVA               -4.104e-01  3.149e-01  -1.303 0.192642
## quadTS                1.611e-05  9.035e-06   1.783 0.074830 .
## quadDE                7.177e+03  3.261e+03   2.201 0.027864 *
## quadPH               -1.209e-01  5.035e-01  -0.240 0.810333
## quadSP               -1.115e+00  2.019e-01  -5.523 3.89e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6328 on 1582 degrees of freedom
## Multiple R-squared:  0.3921, Adjusted R-squared:  0.386
## F-statistic: 63.78 on 16 and 1582 DF,  p-value: < 2.2e-16
```

```
summary(lm_lasso_hier)
```

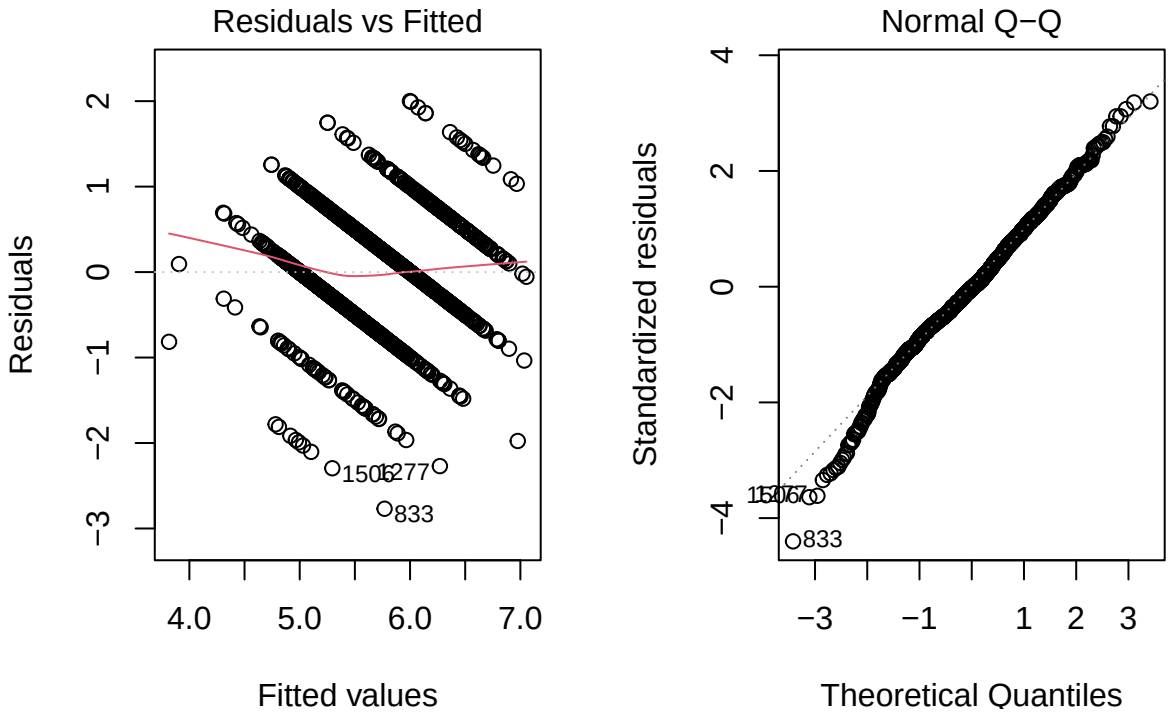
```
##
## Call:
## lm(formula = quality ~ ., data = df_lasso_hier)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.78658 -0.37651 -0.02757  0.43257  2.00030
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -2.917e+00  5.002e+00  -0.583 0.559866
## logFA                 3.900e-01  2.315e-01   1.685 0.092263 .
## volatile.acidity     -5.387e-01  4.194e-01  -1.285 0.199112
## citric.acid          -1.861e-01  1.472e-01  -1.264 0.206362
## logRS                 1.010e-01  6.416e-02   1.575 0.115567
## chlorides            -1.331e+00  4.132e-01  -3.222 0.001300 **
## logFS                 1.016e-01  3.612e-02   2.814 0.004955 **
## total.sulfur.dioxide -6.087e-03  1.678e-03  -3.627 0.000296 ***
## logDE                -3.633e+01  2.802e+01  -1.297 0.194940
## pH                    4.095e+00  2.951e+00   1.388 0.165420
## logSP                 1.411e-01  1.642e-01   0.859 0.390476
```

```
## alcohol               2.240e-01  2.916e-02   7.682 2.72e-14 ***
## quadVA               -2.985e-01  3.231e-01  -0.924 0.355720
## quadTS                1.722e-05  9.035e-06   1.906 0.056895 .
## quadDE                3.675e+03  2.984e+03   1.231 0.218321
## quadPH               -6.893e-01  4.415e-01  -1.561 0.118719
## quadSP               -1.075e+00  2.013e-01  -5.341 1.06e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6337 on 1582 degrees of freedom
## Multiple R-squared:  0.3903, Adjusted R-squared:  0.3842
## F-statistic: 63.31 on 16 and 1582 DF,  p-value: < 2.2e-16
```

```
comp_AIC = rbind(comp_AIC, c("quad_step", round(AIC(lm_step_hier), 3)))
comp_AIC = rbind(comp_AIC, c("quad_lasso", round(AIC(lm_lasso_hier), 3)))
comp_AIC
```

```
##           lm      AIC
## 1       Ori. 3164.277
## 2       Log.  3154.79
## 3     Trans.   3127.7
## 4     Logit. 3146.288
## 5  quad_step 3093.325
## 6 quad_lasso  3097.98
```

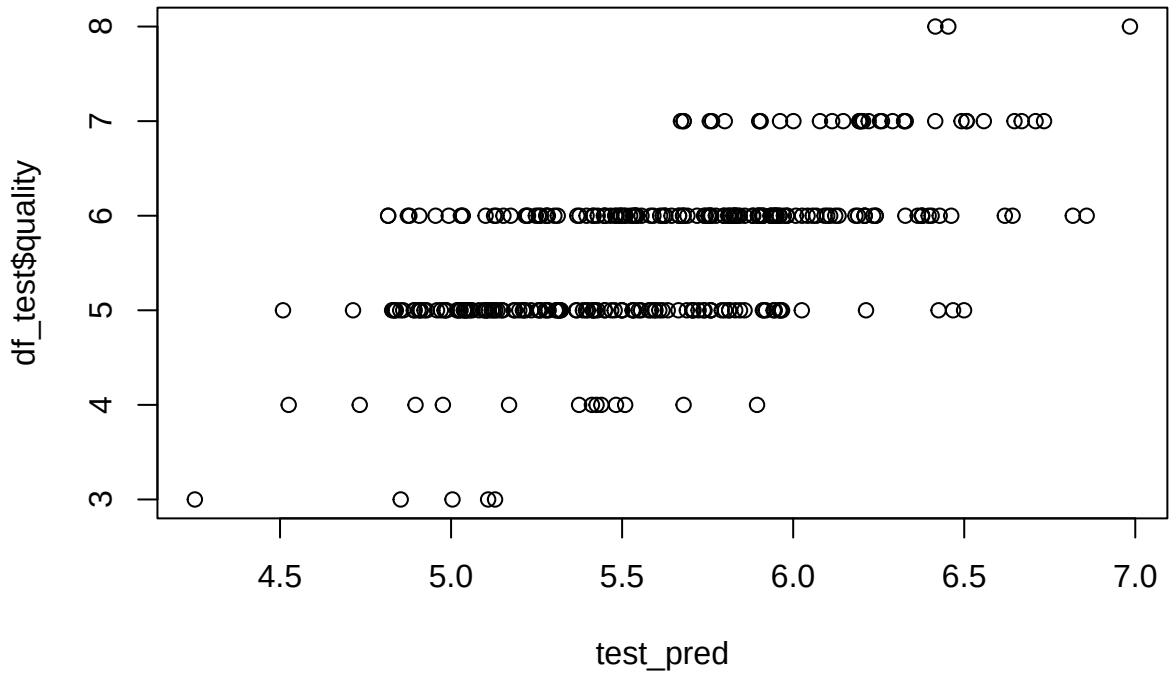
```
par(mfrow = c(1, 2))

plot(lm_step_hier, which = 1)
plot(lm_step_hier, which = 2)
```



```
par(mfrow = c(1, 1))
```

4. Estimation of Error

```
df_final = df_step_hier
n = nrow(df_final)
df_train = df_final[(1:n)%%5 != 0,]
df_test = df_final[(1:n)%%5 == 0,]

lm_train = lm(quality ~ ., data = df_train)
test_pred = predict(lm_train, df_test)
plot(test_pred, df_test$quality)
```




```
mean((test_pred-df_test$quality)^2)
```

```
## [1] 0.4533842
```

```
mean((test_pred-df_test$quality)^2) - 1/12
```

```
## [1] 0.3700508
```

```
table(round(test_pred), df_test$quality)
```

```
##
##      3  4  5  6  7  8
##   4  1  0  0  0  0  0
##   5  4 10 79 41  0  0
##   6  0  3 45 94 28  2
##   7  0  0  0  4  7  1
```

```
lm_final = lm(quality ~ ., data = df_final)
table(round(predict(lm_final, data = df_final)), df_final$quality)
```

```
##
##       3   4   5   6   7   8
##   4   1   3   5   0   0   0
##   5   8  34 468 170   6   0
##   6   1  16 207 441 140   9
##   7   0   0   1  27  53   9
```

5. Weighted Least Squares for imbalanced data

```
wt1 = 1/sqrt(table(df_train$quality))
wts1 = wt1[df_train$quality-2]

wt2 = 1/table(df_train$quality)
wts2 = wt2[df_train$quality-2]

lm_train = lm(quality~., data = df_train)
table(round(predict(lm_train, newdata = df_test)), df_test$quality)
```

```
##
##      3  4  5  6  7  8
##   4  1  0  0  0  0  0
##   5  4 10 79 41  0  0
##   6  0  3 45 94 28  2
##   7  0  0  0  4  7  1
```

```
lm_wt1 = lm(quality~., data = df_train, weights = wts1)
table(round(predict(lm_wt1, newdata = df_test)), df_test$quality)
```

```
##
##      3  4  5  6  7  8
##   3  1  0  0  0  0  0
##   4  1  3  3  2  0  0
##   5  3  7 79 46  0  0
##   6  0  3 39 77 19  0
##   7  0  0  3 14 16  2
##   8  0  0  0  0  0  1
```

```
lm_wt2 = lm(quality~., data = df_train, weights = wts2)
table(round(predict(lm_wt2, newdata = df_test)), df_test$quality)
```

```
##
##      3  4  5  6  7  8
##   3  2  1  1  0  0  0
##   4  2  3 34 13  0  0
##   5  1  7 54 45  5  0
##   6  0  2 29 58 12  0
##   7  0  0  6 18 16  2
##   8  0  0  0  5  2  1
```