Final_Project_Wine_Quality

Michael Knight

6/18/2022

I am using the Wine Quality Data Set (https://archive.ics.uci.edu/ml/datasets/Wine+Quality) for both my regression and classification model sets.

```
#red_wine <- read.csv("winequality-red.csv", sep=";", header=T)

white_wine <- read.csv("winequality-white.csv", sep=";",header=T)
```

For each of the Regression and Classification tasks, complete the following: 2. Prepare your data set(s): (a) Briefly define the variables in the data set and the overall goal of your analysis. Clearly identify the response variable and predictor variables in your data set. Identify whether these are quantitative or categorical. State if this corresponds to a regression or classification setting and why.

```
colnames(white_wine)
```

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

(b) Remove any observations that have missing values on any variables. Treat the remaining observations as your full (complete cases) data set. How many observations did you remove? What is the sample size of your remaining full data set?

```
#red_wine <- read.csv("winequality-red.csv", sep=";", header=T)

white_wine <- read.csv("winequality-white.csv", sep=";",header=T)


#apply(is.na(red_wine),2,sum) # no missing values
apply(is.na(white_wine),2,sum) # no missing values
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##                    0                    0                    0
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    0                    0                    0
## total.sulfur.dioxide              density                   pH
##                    0                    0                    0
##            sulphates              alcohol              quality
##                    0                    0                    0
```

```
# add a qualitative version of the quality variable, to be used for the classification models
#red_wine$quality.factor <- as.factor(red_wine$quality)
white_wine$quality.factor <- as.factor(white_wine$quality)

# add a qualitative factor of quality as a binary 'good/bad' where 0=bad, 1=good
quality.good <- rep(0, length(white_wine$quality))
quality.good[white_wine$quality > 5] <- 1
white_wine <- data.frame(white_wine, quality.good)
white_wine$quality.good <- as.factor(white_wine$quality.good)


# quality.good <- rep(0, length(red_wine$quality))
# quality.good[red_wine$quality > 5] <- 1
# red_wine <- data.frame(red_wine, quality.good)


#red_wine <- na.omit(red_wine)

white_wine <- na.omit(white_wine)

#nrow(red_wine)

nrow(white_wine)
```

```
## [1] 4898
```

Ans: There were 0 null valuations, hence no observations required removal.

(c) Randomly select a test data set that is approximately 10% of your full data set. Separate this out from your full data set. Treat the remaining 90% of your data set as your training data.

```
set.seed(20) # for reproducibility

# red wine
#n <- length(red_wine$quality)
#Z <- sample(n,n*0.9)
#train.red <- red_wine[sort(Z),]
#test.red <- red_wine[-sort(Z),]

# 1st 6 observations of training data
#head(train.red)
#nrow(train.red)

# 1st 6 observations of testing data
#head(test.red)
#nrow(test.red)


# white wine
n <- length(white_wine$quality)
Z <- sample(n,n*0.9)
train.white <- white_wine[sort(Z),]
```

```
test.white <- white_wine[-sort(Z),]

# 1st 6 observations of training data
head(train.white)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4           7.2             0.23        0.32            8.5     0.058
## 6           8.1             0.28        0.40            6.9     0.050
## 7           6.2             0.32        0.16            7.0     0.045
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 6                  30                   97  0.9951 3.26      0.44    10.1
## 7                  30                  136  0.9949 3.18      0.47     9.6
##   quality quality.factor quality.good
## 1       6              6            1
## 2       6              6            1
## 3       6              6            1
## 4       6              6            1
## 6       6              6            1
## 7       6              6            1
```

```
nrow(train.white)
```

```
## [1] 4408
```

```
# 1st 6 observations of testing data
head(test.white)
```

```
##    fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 5            7.2             0.23        0.32            8.5     0.058
## 17           6.3             0.48        0.04            1.1     0.046
## 19           7.4             0.34        0.42            1.1     0.033
## 62           6.0             0.19        0.26           12.4     0.048
## 67           6.4             0.26        0.24            6.4     0.040
## 72           6.8             0.30        0.23            4.6     0.061
##    free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 5                 47.0                186.0  0.9956 3.19      0.40     9.9
## 17                30.0                 99.0  0.9928 3.24      0.36     9.6
## 19                17.0                171.0  0.9917 3.12      0.53    11.3
## 62                50.0                147.0  0.9972 3.30      0.36     8.9
## 67                27.0                124.0  0.9903 3.22      0.49    12.6
## 72                50.5                238.5  0.9958 3.32      0.60     9.5
##    quality quality.factor quality.good
## 5        6              6            1
## 17       6              6            1
## 19       6              6            1
```

```
## 62       6             6             1
## 67       7             7             1
## 72       5             5             0
```

```
nrow(test.white)
```

```
## [1] 490
```

3. Identify and conduct your analysis on the training data:

(a) Conduct an exploratory data analysis on your training data and briefly summarize any interesting features of your data set.

```
#summary(train.red)
summary(train.white)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.080    Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.210    1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.260    Median :0.3200   Median : 5.175
##  Mean   : 6.852   Mean   :0.277    Mean   :0.3342   Mean   : 6.370
##  3rd Qu.: 7.300   3rd Qu.:0.320    3rd Qu.:0.3900   3rd Qu.: 9.812
##  Max.   :14.200   Max.   :1.100    Max.   :1.6600   Max.   :65.800
##
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04572   Mean   : 35.31      Mean   :137.9        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##
##        pH          sulphates         alcohol         quality     quality.factor
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.00   3:  17
##  1st Qu.:3.087   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.00   4: 147
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.00   5:1302
##  Mean   :3.188   Mean   :0.4897   Mean   :10.51   Mean   :5.88   6:1991
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.00   7: 788
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.00   8: 158
##                                                                  9:   5
##  quality.good
##  0:1466
##  1:2942
##
##
##
##
##
```

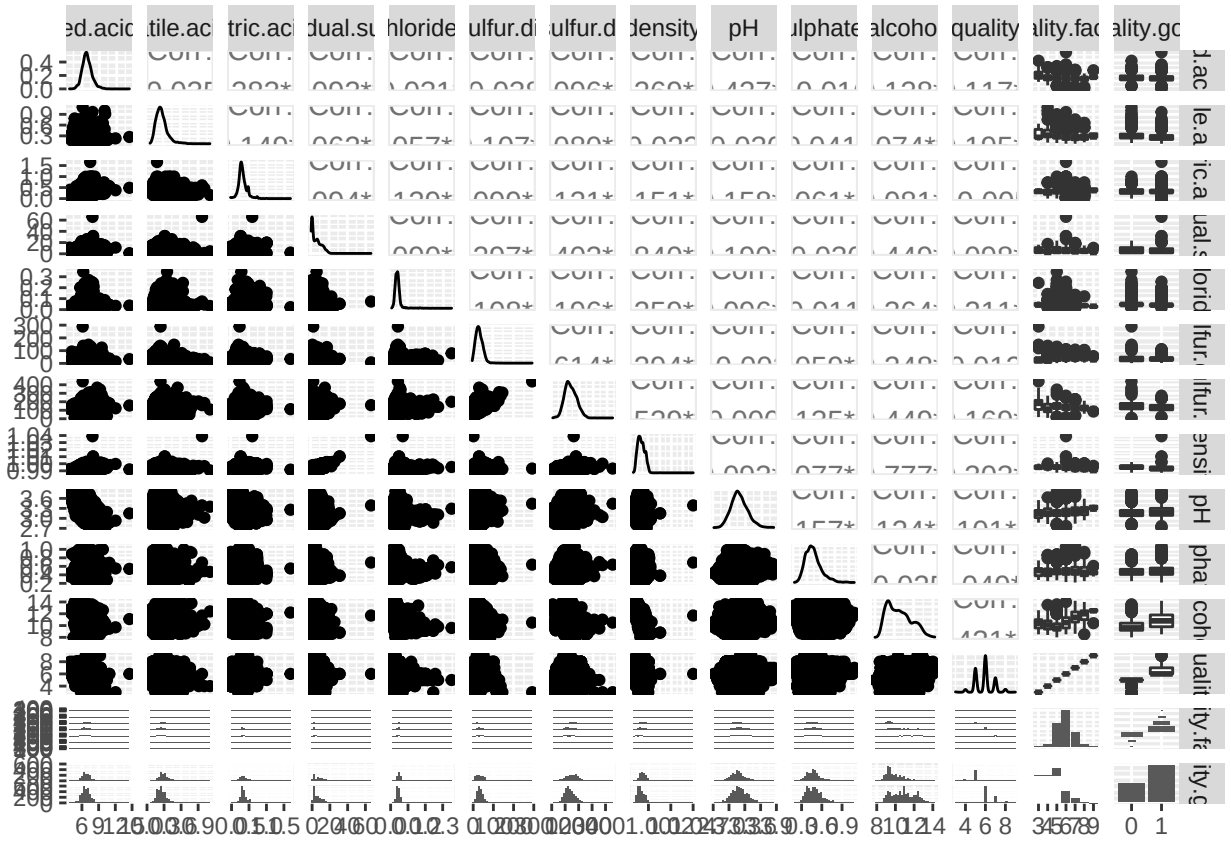
Ans: It is interesting to me that for red wines, there are no wines with a rating less then 3 (and there were only 10 reds with a rating of 3) and no reds with a rating higher than 8 (and there were only 16 wines with a rating of 8), and that for white wines, there are no wines with a rating less then 3 (and there were only

19 white with a rating of 3) and no whites with a rating higher than 9 (and there were only 5 wines with a rating of 9).

Checking for multicolinearity:

```
ggpairs(data = train.white) # we see a 0.833 correlation between density and residual.sugars, so we wil
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



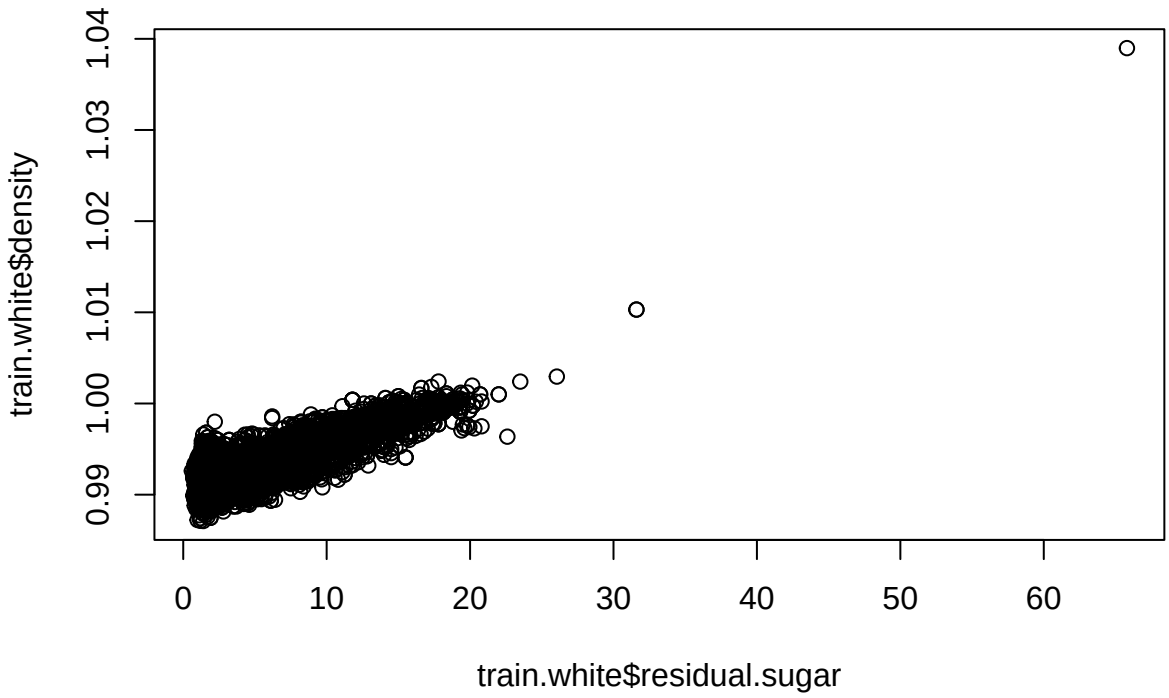

```
# note that Correlation coefficients whose magnitude are between 0.9 and 1.0 indicate variables which c
```

since residual sugar would determine how sweet a wine is, and since a wine can only be so dense, I decided to remove density from my MLR model (and any other models where non-multicolinearity was an assumption).

```
checkfullmodel <- lm(quality ~ . - quality.factor - quality.good, data = train.white)
tidycheck <- tidy(checkfullmodel)
tidycheck
```

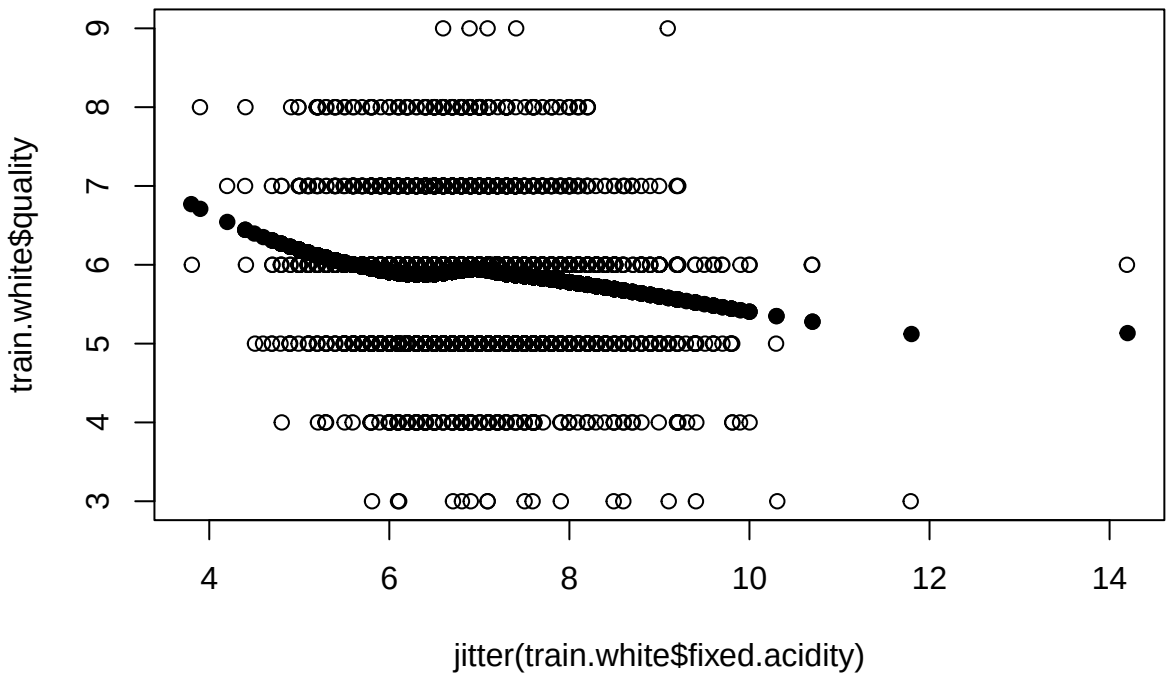
```
## # A tibble: 12 x 5
##    term                  estimate std.error statistic  p.value
##    <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)           135.       19.6          6.88  6.95e-12
##  2 fixed.acidity           0.0457    0.0218       2.09  3.65e- 2
##  3 volatile.acidity       -1.89      0.121      -15.7   8.36e-54
##  4 citric.acid             0.0804    0.101        0.799 4.24e- 1
##  5 residual.sugar          0.0747    0.00787      9.49  3.72e-21
##  6 chlorides              -0.535     0.594       -0.902 3.67e- 1
##  7 free.sulfur.dioxide     0.00365   0.000887     4.11  4.03e- 5
##  8 total.sulfur.dioxide   -0.000200  0.000399    -0.502 6.15e- 1
##  9 density              -134.       19.9         -6.77  1.49e-11
## 10 pH                     0.621     0.111        5.60  2.21e- 8
## 11 sulphates              0.593     0.106        5.58  2.55e- 8
## 12 alcohol                0.209     0.0253       8.27  1.80e-16
```

```
plot(train.white$residual.sugar, train.white$density)
```

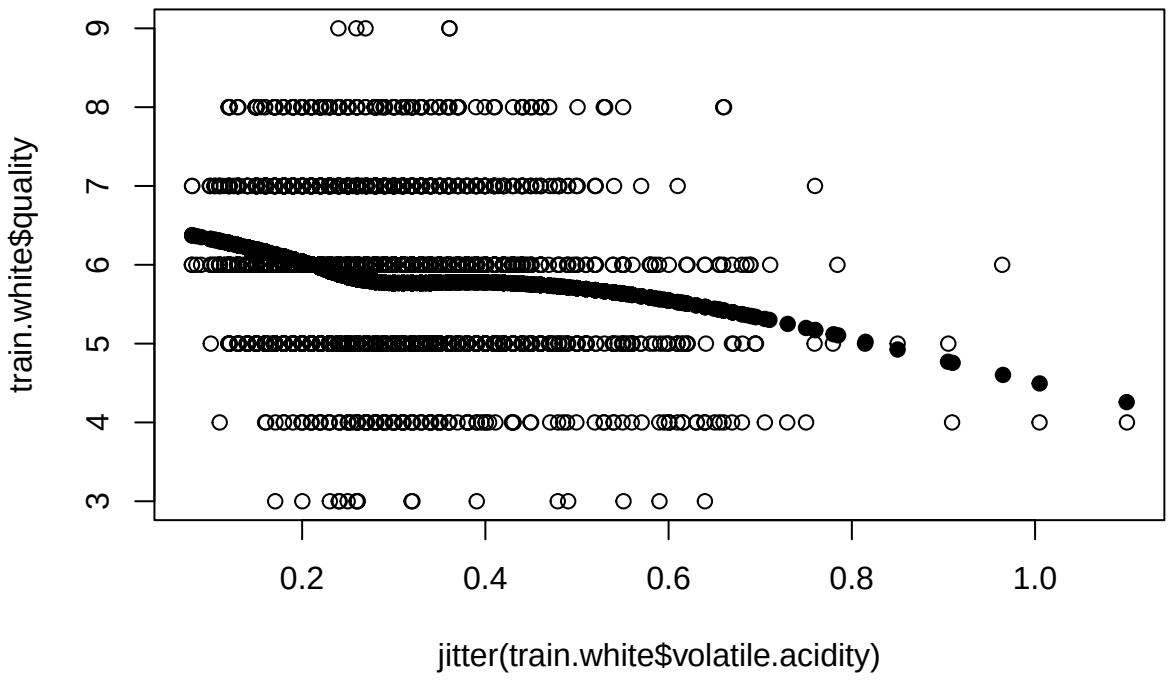


Scatter Plots of residual vs fitted for train.white

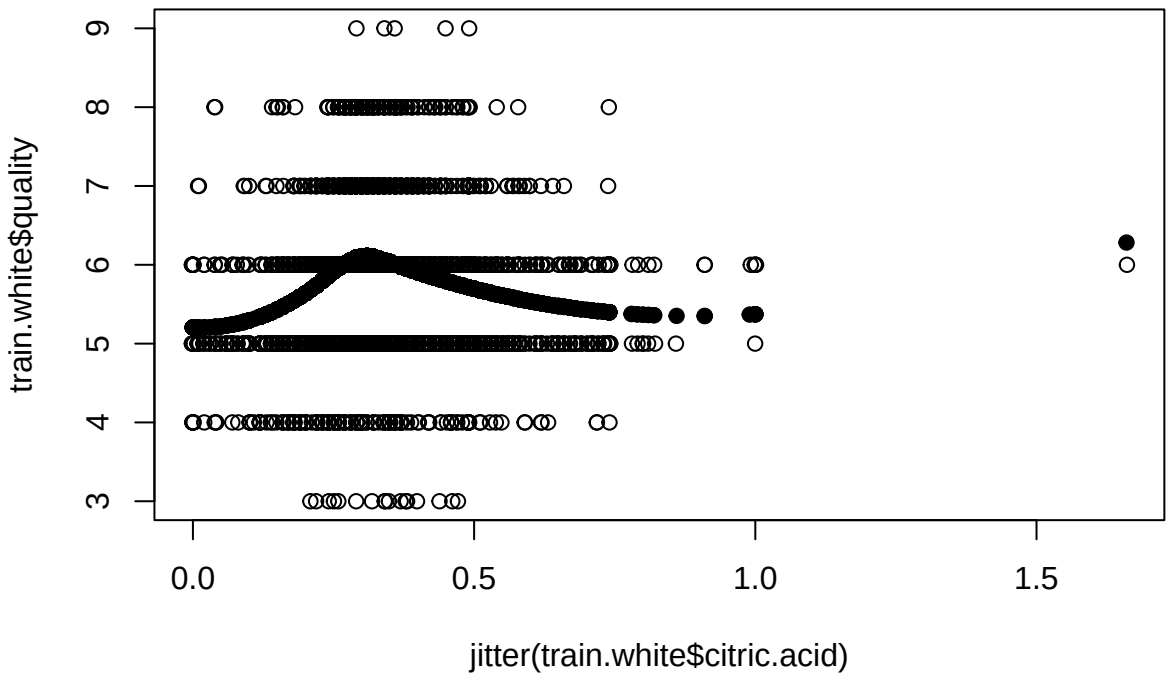
```
plot(jitter(train.white$fixed.acidity), train.white$quality)
points(train.white$fixed.acidity, loess(train.white$quality~train.white$fixed.acidity)$fitted, pch=19)
```



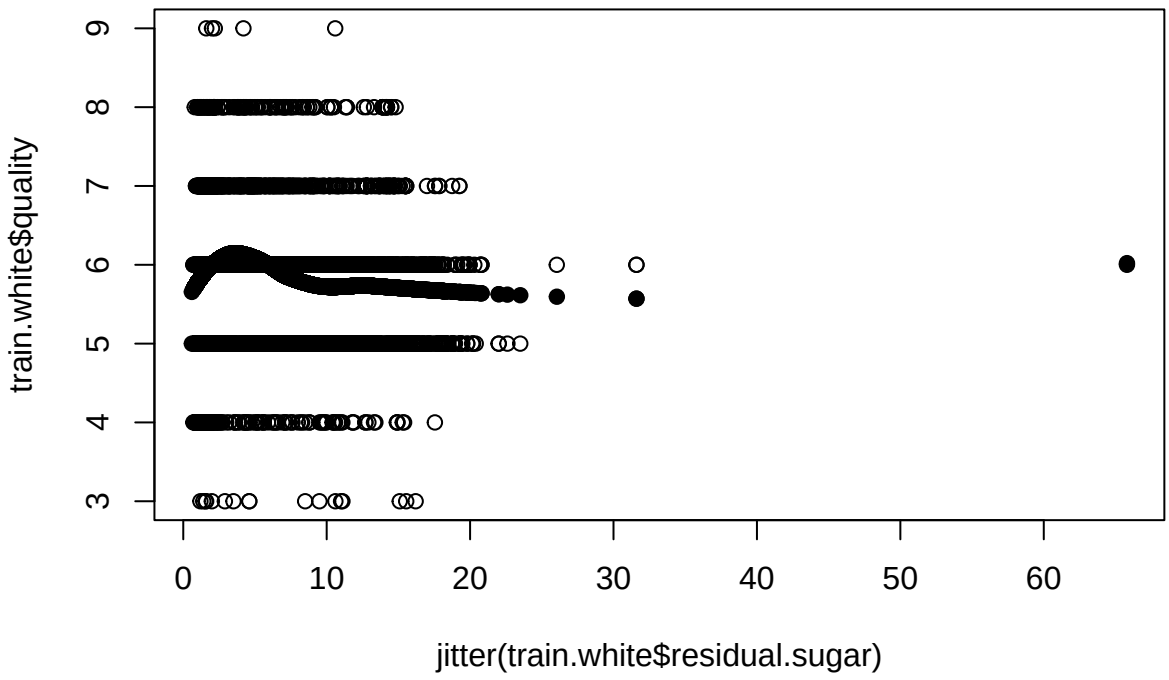

```
plot(jitter(train.white$volatile.acidity), train.white$quality)
points(train.white$volatile.acidity, loess(train.white$quality~train.white$volatile.acidity)$fitted, pch
```



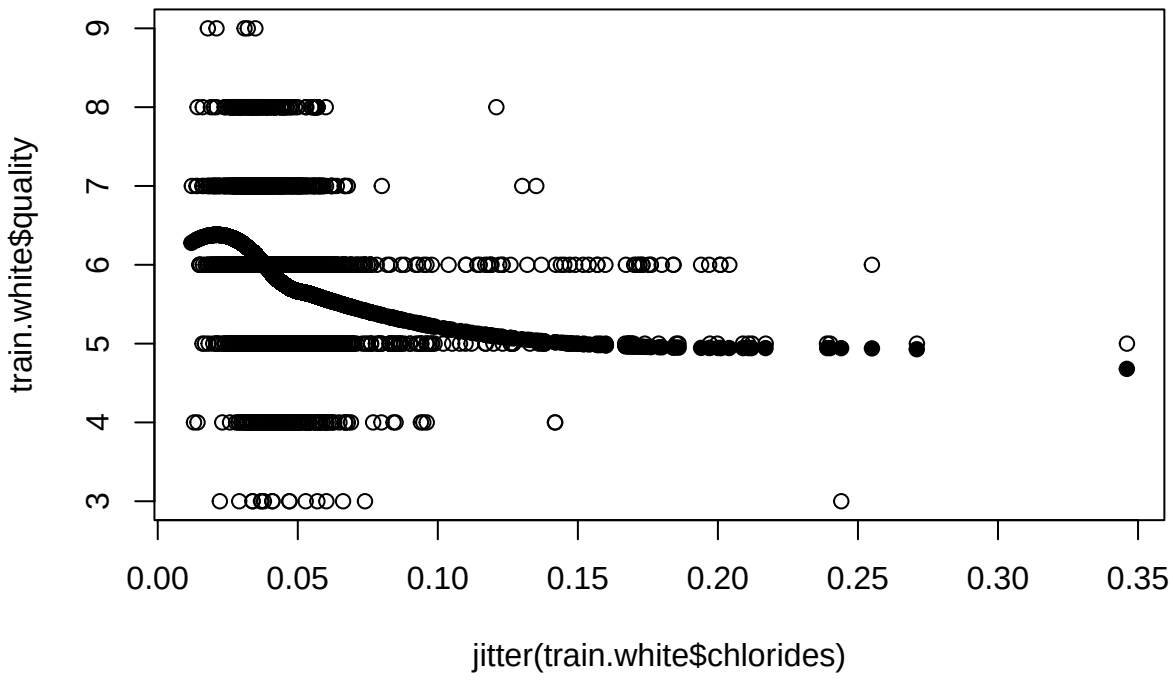

```
plot(jitter(train.white$citric.acid), train.white$quality)
points(train.white$citric.acid, loess(train.white$quality~train.white$citric.acid)$fitted, pch=19)
```



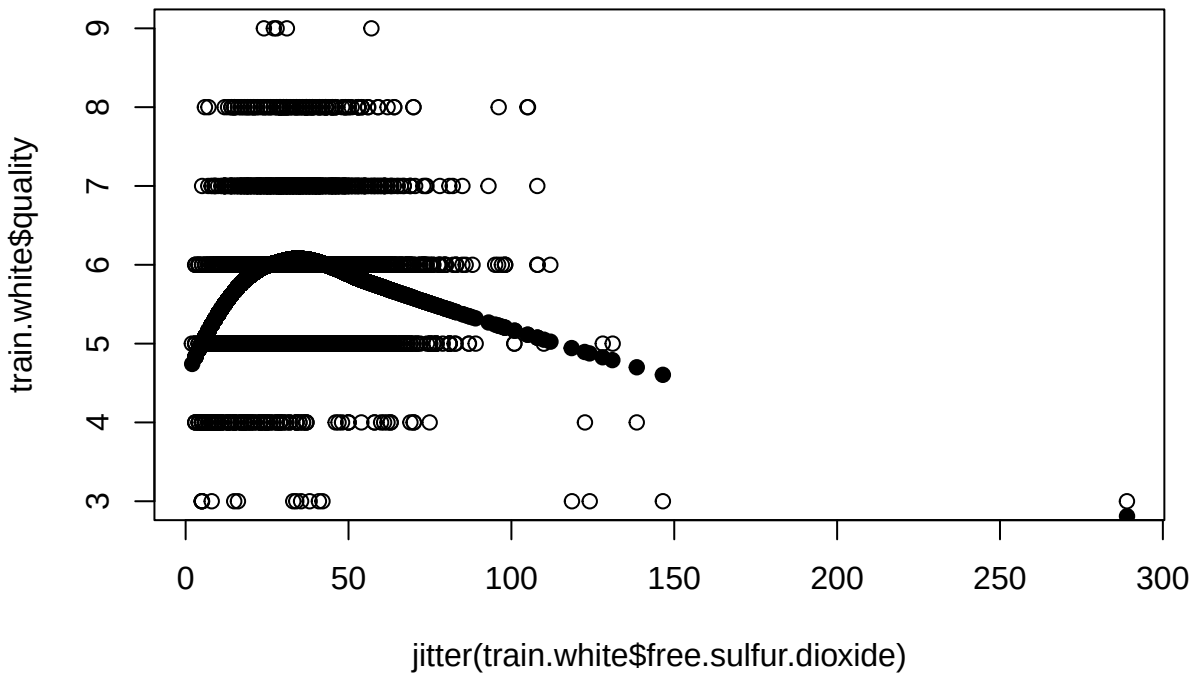

```
plot(jitter(train.white$residual.sugar), train.white$quality)
points(train.white$residual.sugar, loess(train.white$quality~train.white$residual.sugar)$fitted, pch=19
```



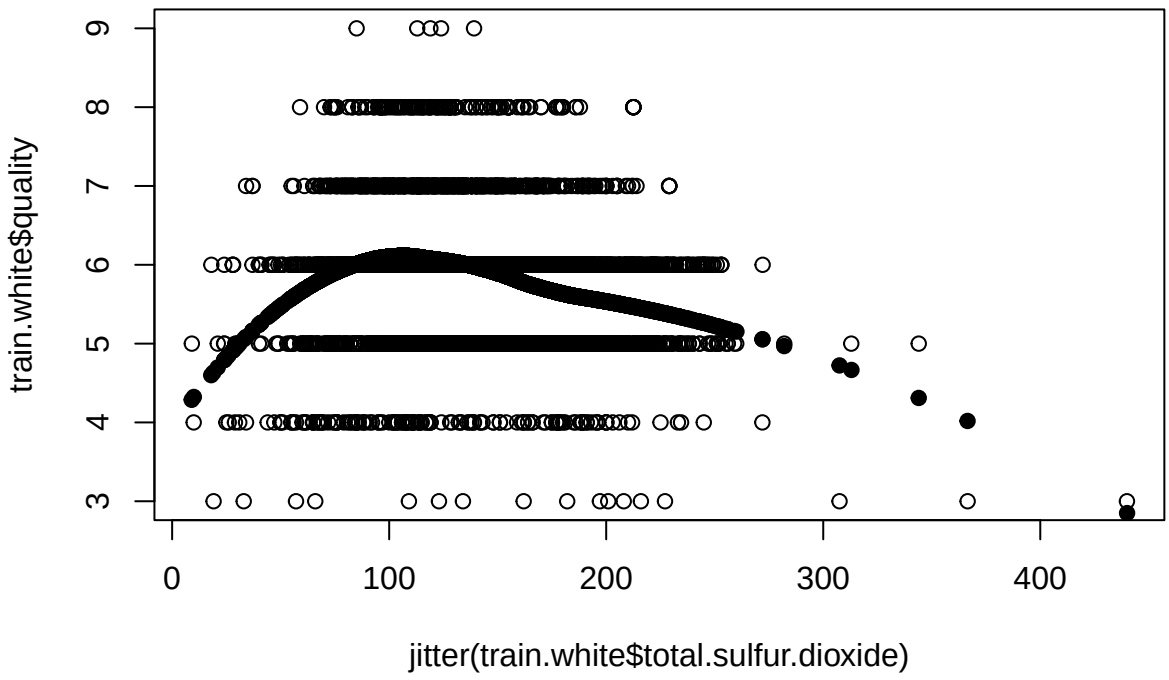

```
plot(jitter(train.white$chlorides), train.white$quality)
points(train.white$chlorides, loess(train.white$quality~train.white$chlorides)$fitted, pch=19)
```



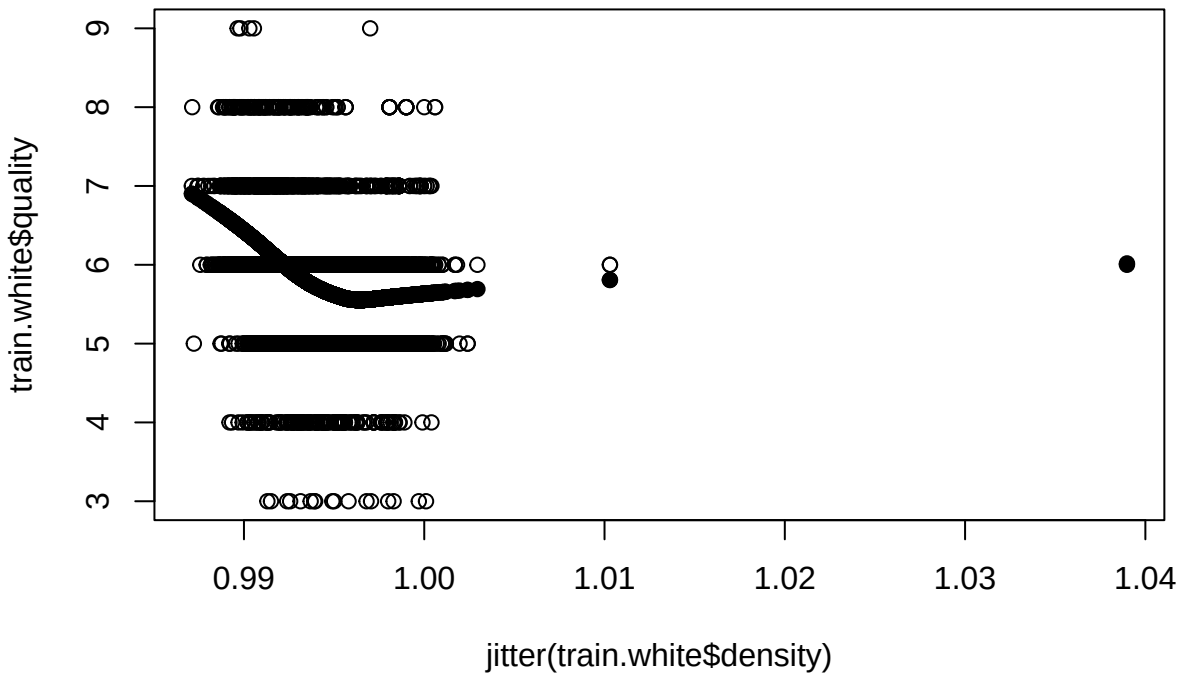

```
plot(jitter(train.white$free.sulfur.dioxide), train.white$quality)
points(train.white$free.sulfur.dioxide, loess(train.white$quality~train.white$free.sulfur.dioxide)$fitt
```



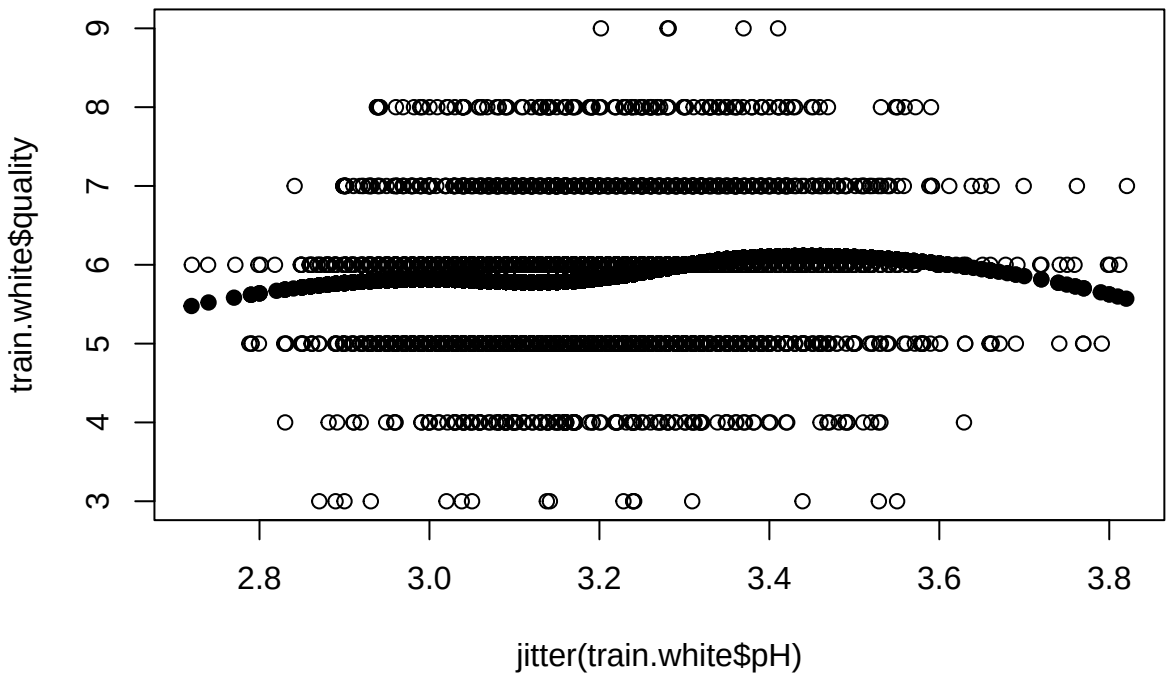

```
plot(jitter(train.white$total.sulfur.dioxide), train.white$quality)
points(train.white$total.sulfur.dioxide, loess(train.white$quality~train.white$total.sulfur.dioxide)$fi
```



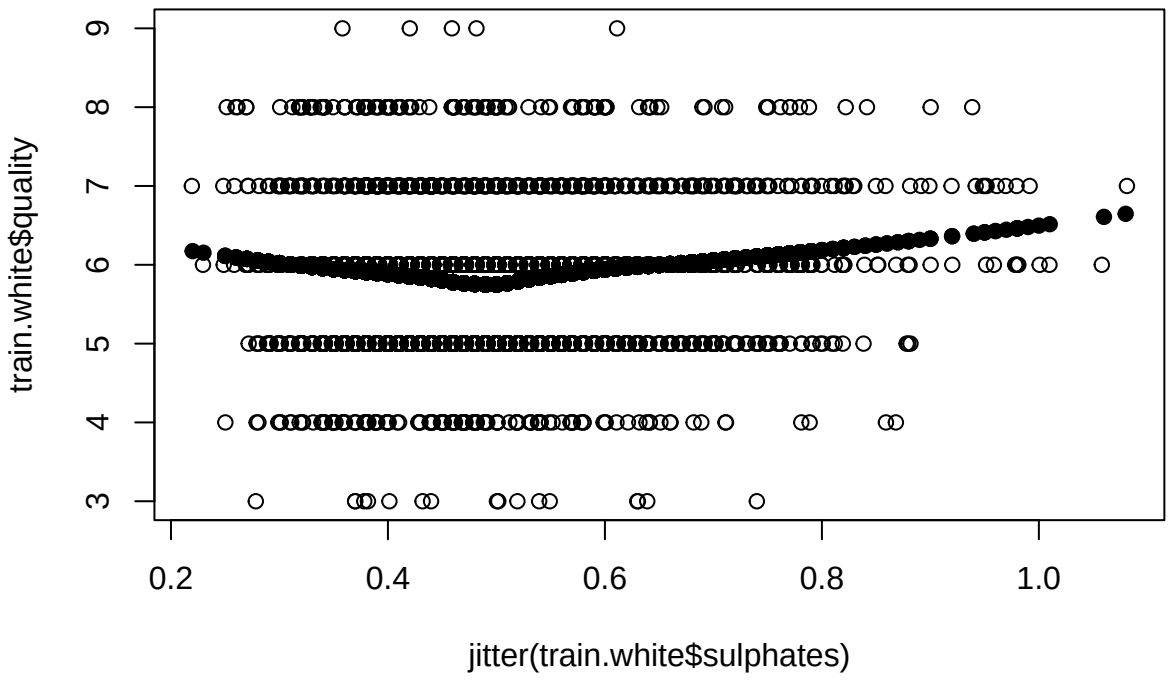

```
plot(jitter(train.white$density), train.white$quality)
points(train.white$density, loess(train.white$quality~train.white$density)$fitted, pch=19)
```



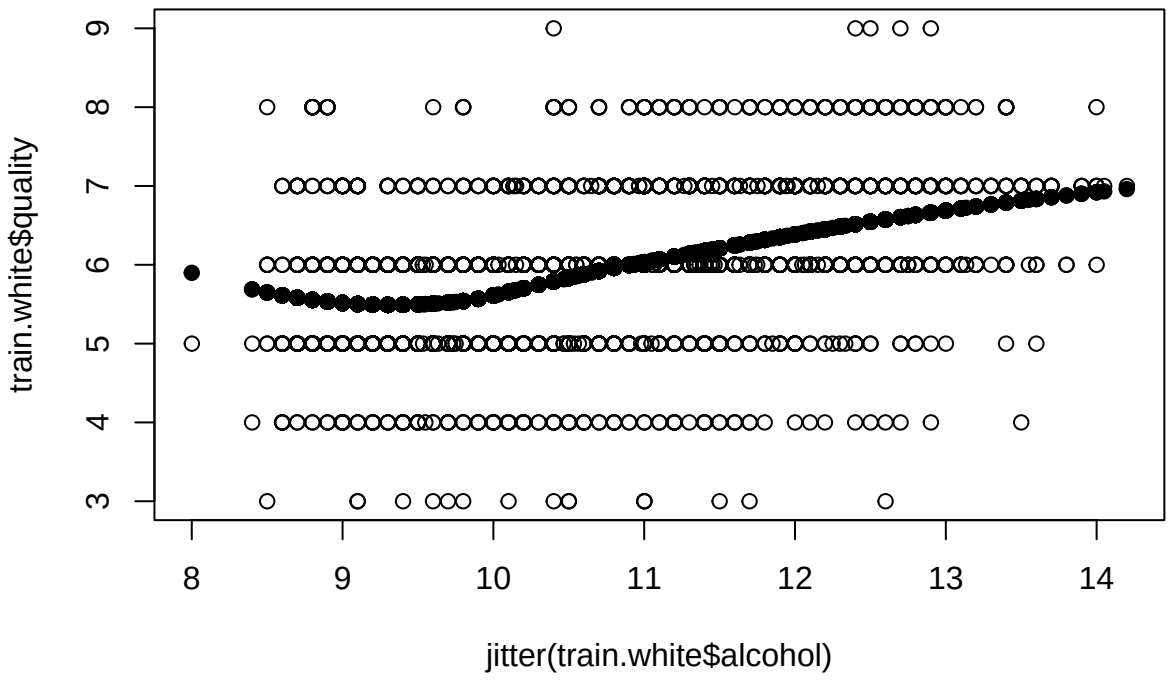

```
plot(jitter(train.white$pH), train.white$quality)
points(train.white$pH, loess(train.white$quality~train.white$pH)$fitted, pch=19)
```




```
plot(jitter(train.white$sulphates), train.white$quality)
points(train.white$sulphates, loess(train.white$quality~train.white$sulphates)$fitted, pch=19)
```




```
plot(jitter(train.white$alcohol), train.white$quality)
points(train.white$alcohol, loess(train.white$quality~train.white$alcohol)$fitted, pch=19)
```



I am seeing some outliers that are clearly skewing the data for fixed.acidity, citric.acid, residual.sugar, free.sulfur.dioxide, and density.

```
# Outlier deterction

#max(train.white$free.sulfur.dioxide)

train.white %>%
  filter(fixed.acidity >12) # one outlier
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1          14.2             0.27        0.49            1.1     0.037
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  33                  156   0.992 3.15      0.54    11.1
##   quality quality.factor quality.good
## 1       6              6            1
```

```
train.white %>%
  filter(citric.acid >1.2) # two outliers
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4              0.2        1.66            2.1     0.022
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  34                  113 0.99165 3.26      0.55    12.2
##   quality quality.factor quality.good
## 1       6              6            1
```

```
train.white %>%
  filter(residual.sugar >35) # one outlier
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.8            0.965         0.6           65.8     0.074
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                   8                  160 1.03898 3.39      0.69    11.7
##   quality quality.factor quality.good
## 1       6              6            1
```

```
train.white %>%
  filter(free.sulfur.dioxide >250) # one outlier
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           6.1             0.26        0.25            2.9     0.047
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                 289                  440 0.99314 3.44      0.64    10.5
##   quality quality.factor quality.good
## 1       3              3            0
```

```
train.white %>%
  filter(density >1.01) # two outliers, but notice that one of these two points is the same wine with r
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.9            0.330        0.28           31.6     0.053
## 2           7.9            0.330        0.28           31.6     0.053
## 3           7.8            0.965        0.60           65.8     0.074
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  35                  176 1.01030 3.15      0.38     8.8
## 2                  35                  176 1.01030 3.15      0.38     8.8
## 3                   8                  160 1.03898 3.39      0.69    11.7
##   quality quality.factor quality.good
## 1       6              6            1
## 2       6              6            1
## 3       6              6            1
```

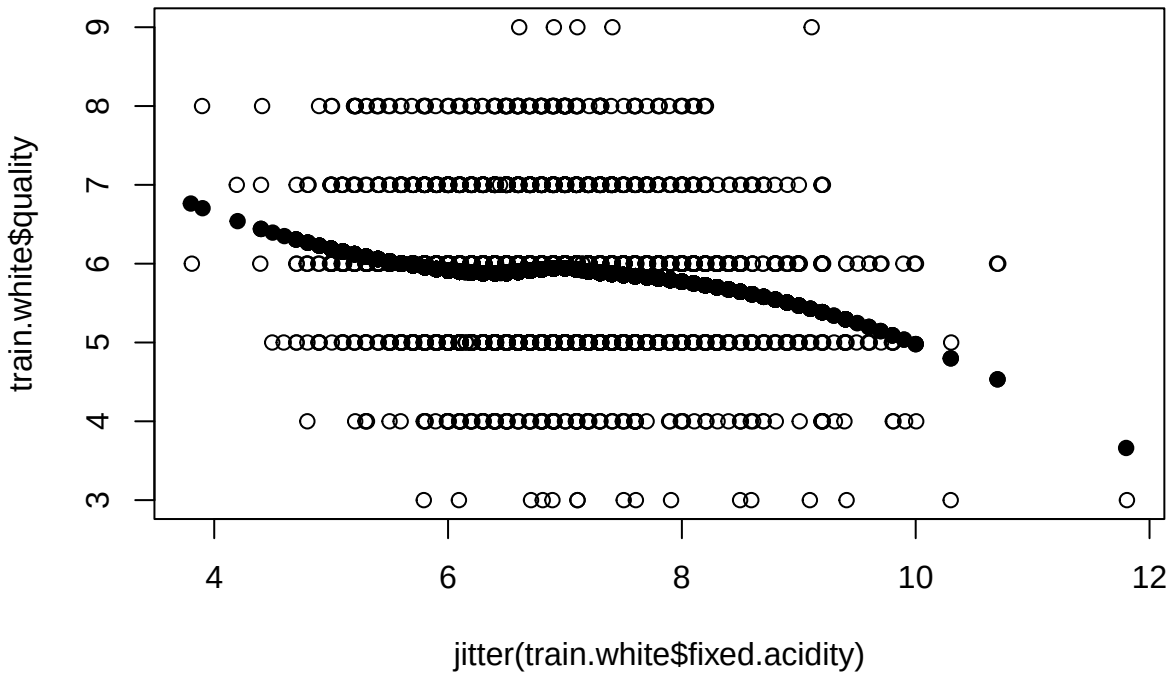
I will remove these six observations and re-examine the plots

```
train.white %>%
  filter(fixed.acidity <12) %>%
  filter(citric.acid < 1.2) %>%
  filter(residual.sugar < 35) %>%
  filter(free.sulfur.dioxide < 250) %>%
  filter(density < 1.01) -> train.white

nrow(train.white) # training set now has 4,403 observations
```

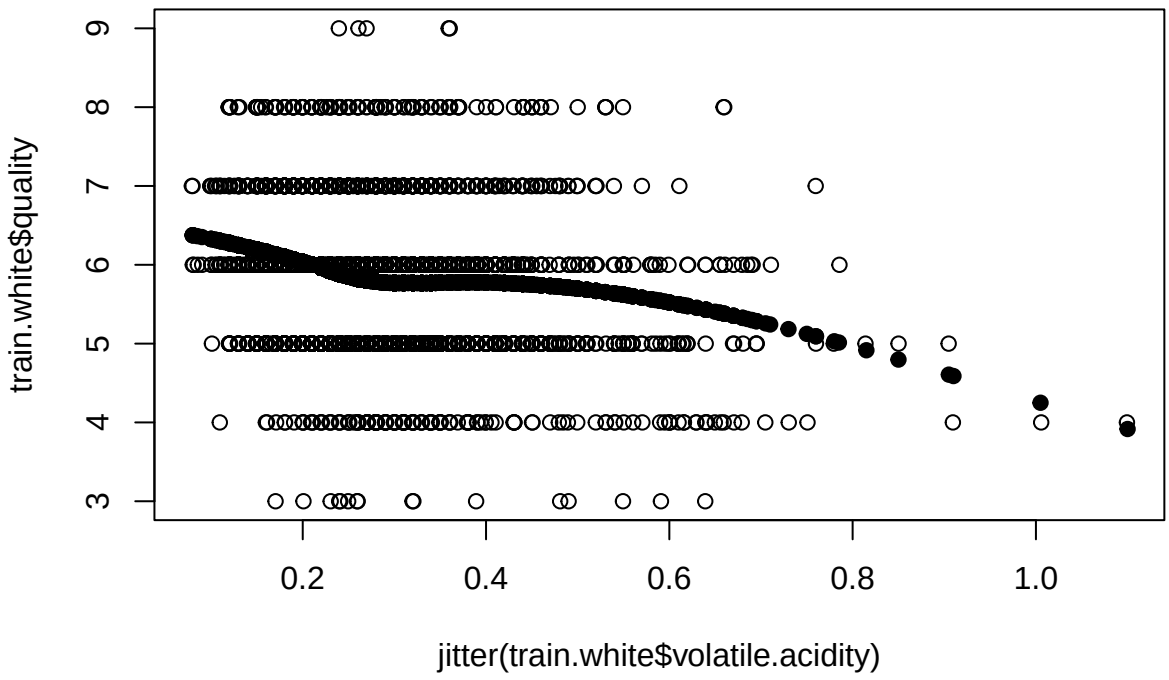
```
## [1] 4402
```

Lets reobserve the graphs from before

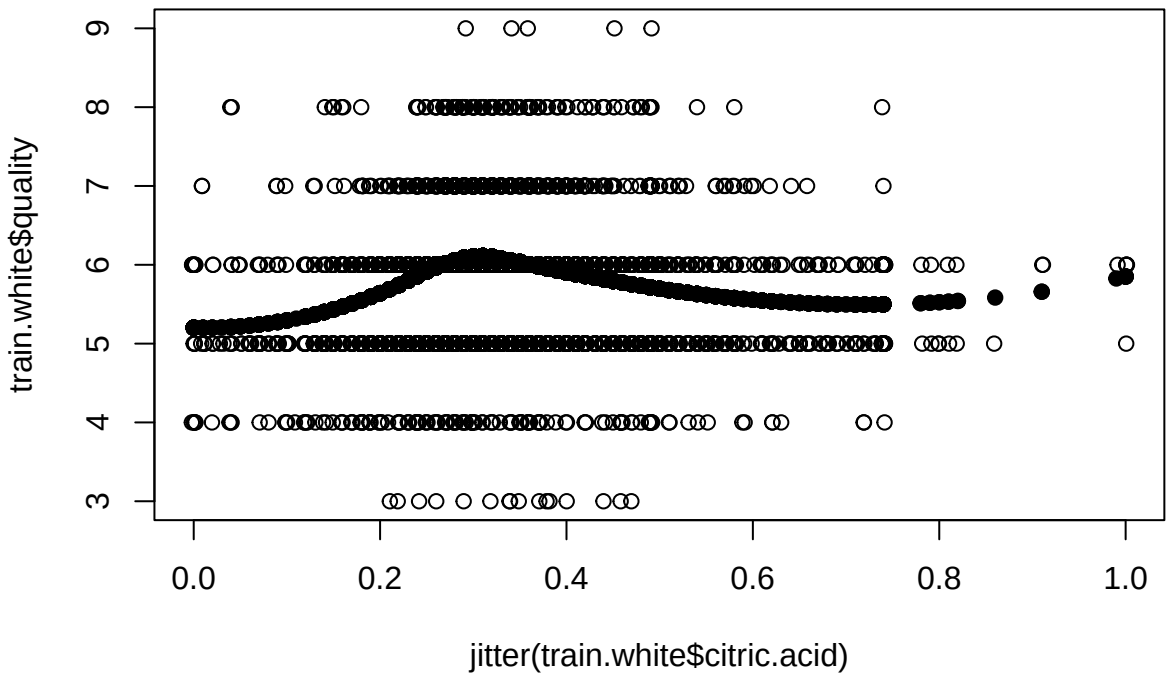
```
# Lets reobserve the graphs from before

plot(jitter(train.white$fixed.acidity), train.white$quality)
points(train.white$fixed.acidity, loess(train.white$quality~train.white$fixed.acidity)$fitted, pch=19)
```



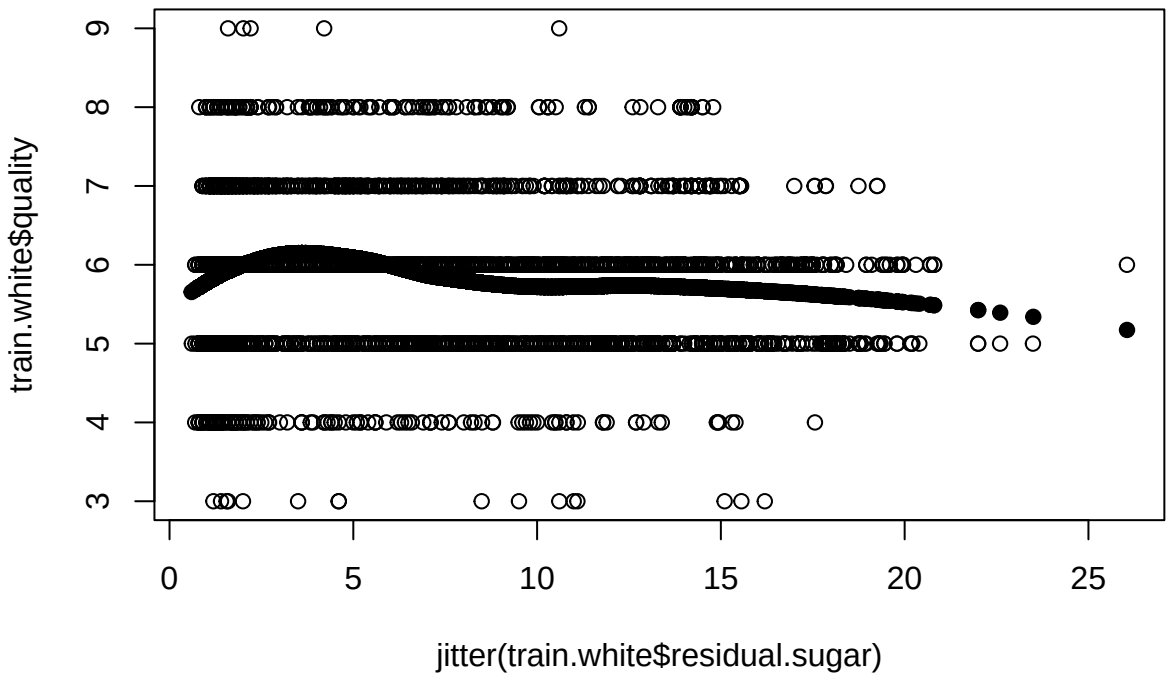

```
plot(jitter(train.white$volatile.acidity), train.white$quality)
points(train.white$volatile.acidity, loess(train.white$quality~train.white$volatile.acidity)$fitted, pch
```



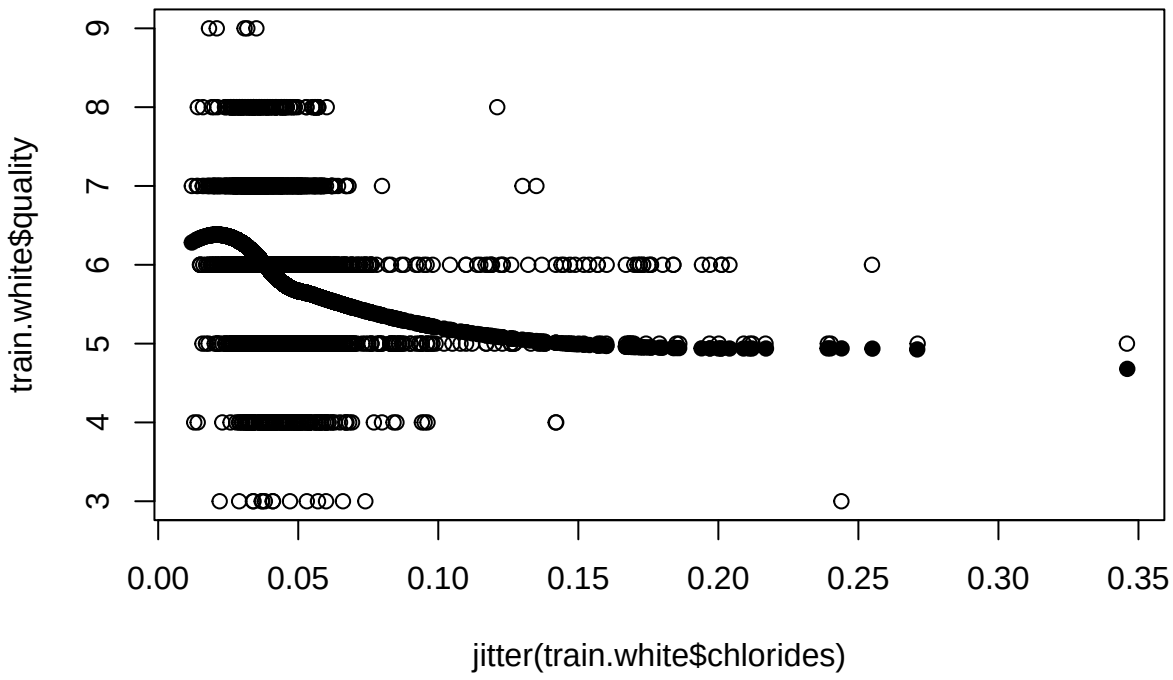

```
plot(jitter(train.white$citric.acid), train.white$quality)
points(train.white$citric.acid, loess(train.white$quality~train.white$citric.acid)$fitted, pch=19)
```



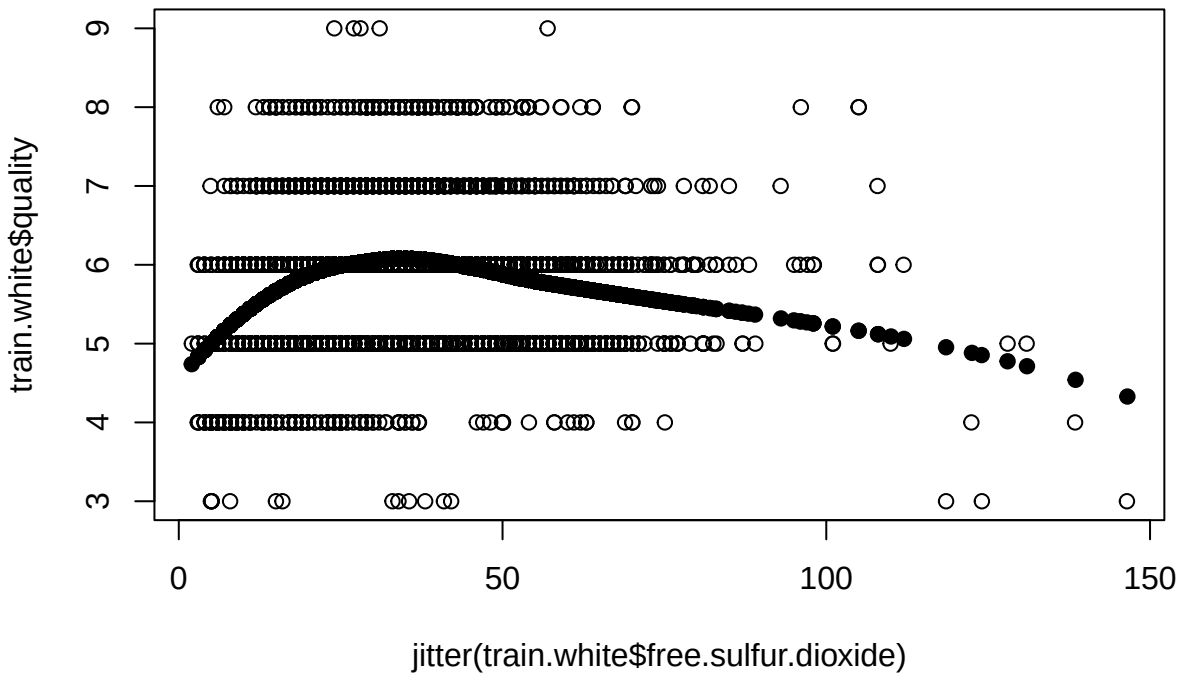

```
plot(jitter(train.white$residual.sugar), train.white$quality)
points(train.white$residual.sugar, loess(train.white$quality~train.white$residual.sugar)$fitted, pch=19
```



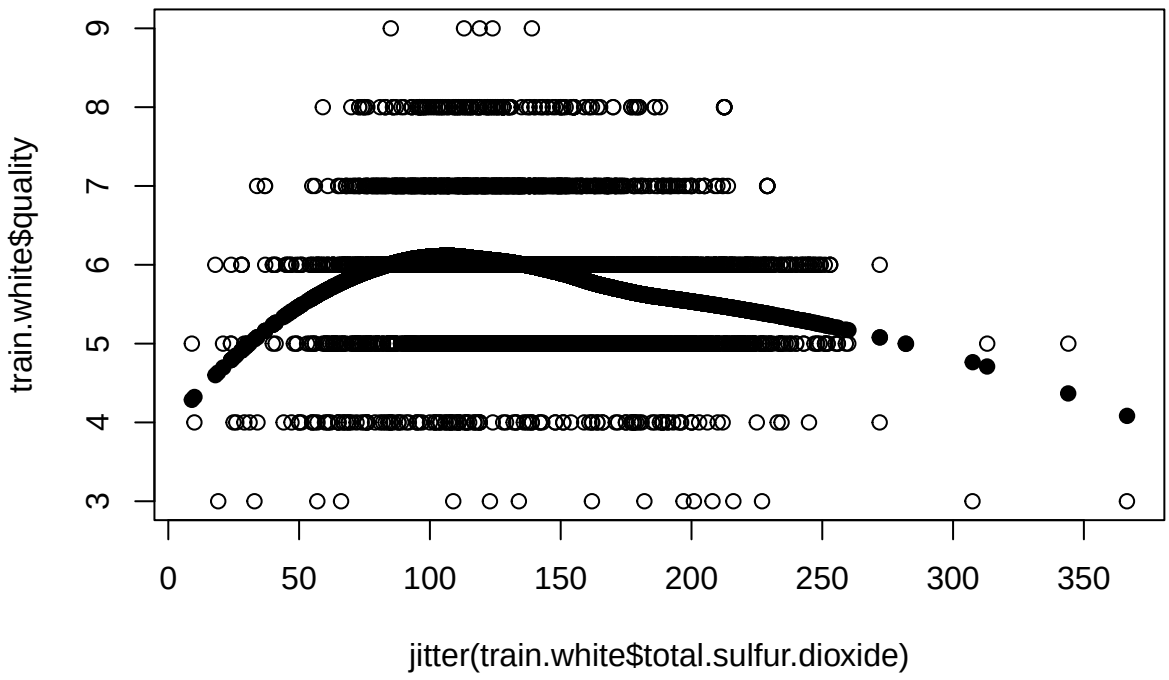

```
plot(jitter(train.white$chlorides), train.white$quality)
points(train.white$chlorides, loess(train.white$quality~train.white$chlorides)$fitted, pch=19)
```



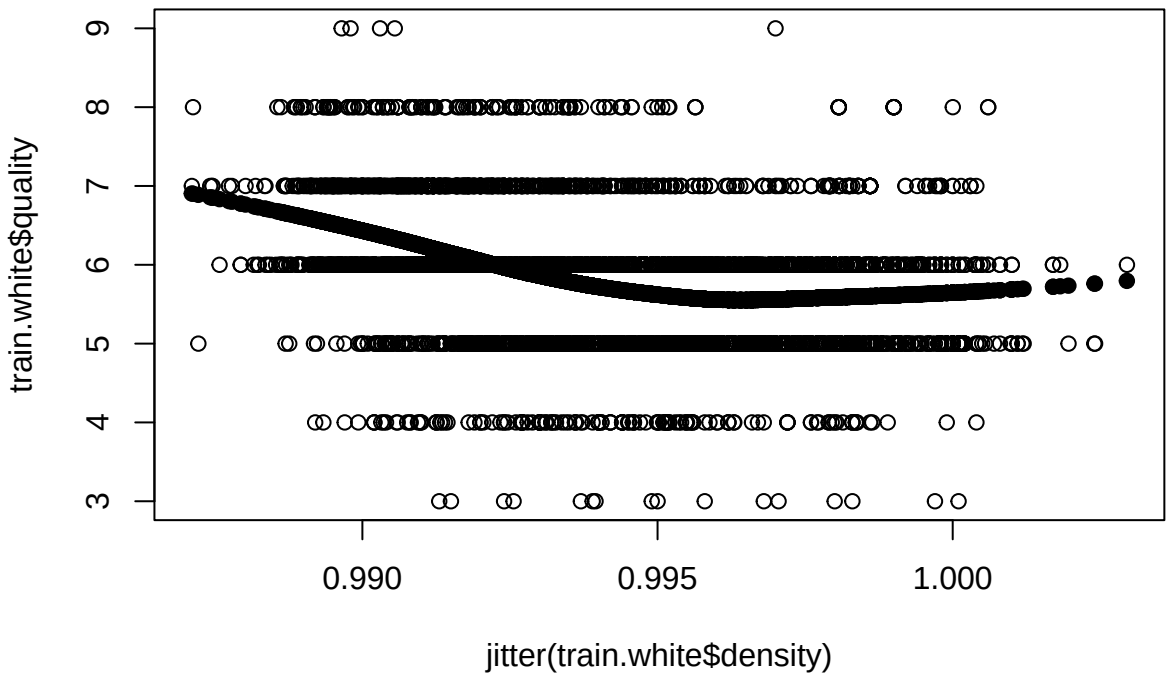

```
plot(jitter(train.white$free.sulfur.dioxide), train.white$quality)
points(train.white$free.sulfur.dioxide, loess(train.white$quality~train.white$free.sulfur.dioxide)$fitt
```



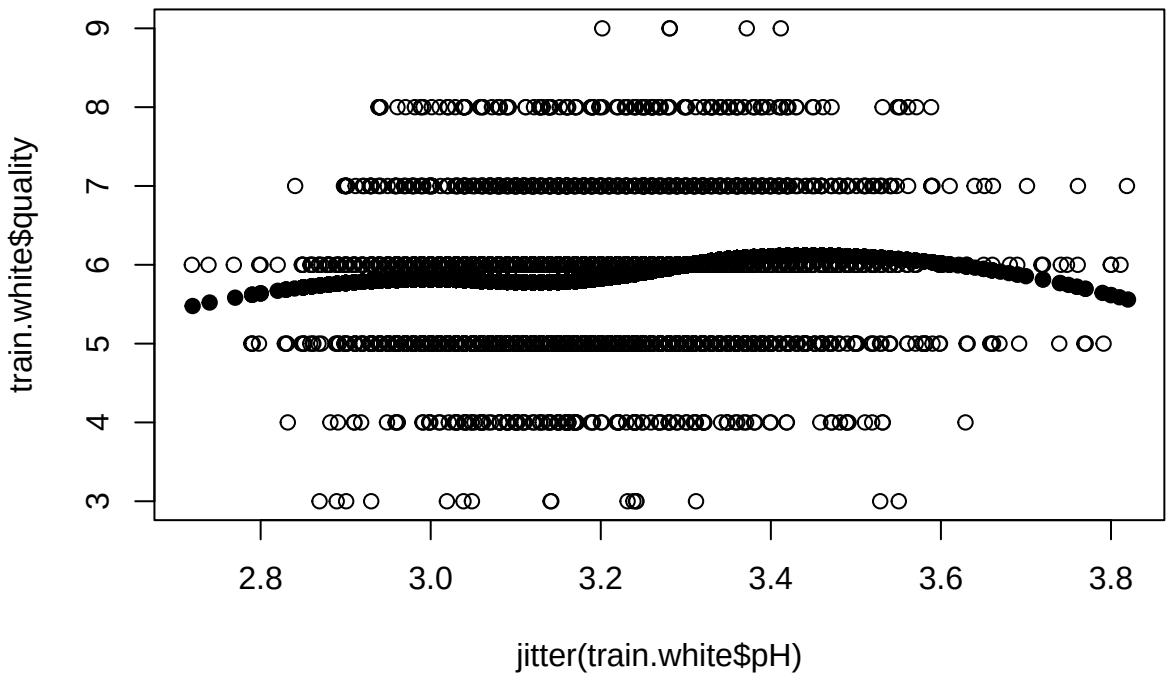

```
plot(jitter(train.white$total.sulfur.dioxide), train.white$quality)
points(train.white$total.sulfur.dioxide, loess(train.white$quality~train.white$total.sulfur.dioxide)$fi
```



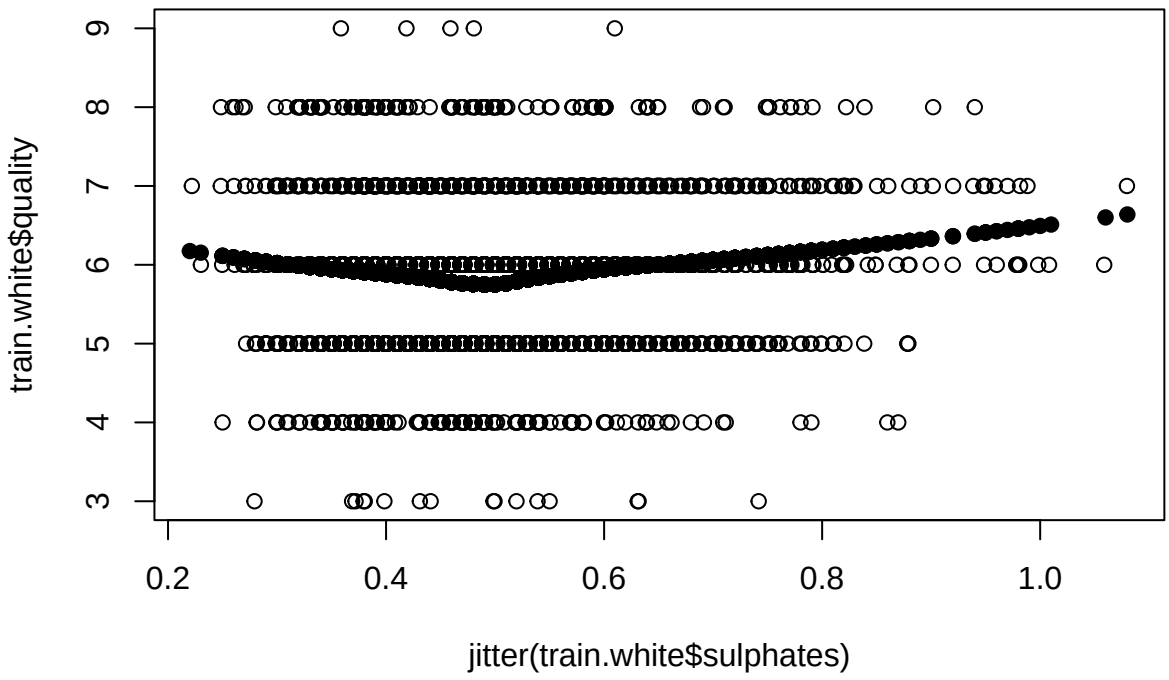

```
plot(jitter(train.white$density), train.white$quality)
points(train.white$density, loess(train.white$quality~train.white$density)$fitted, pch=19)
```



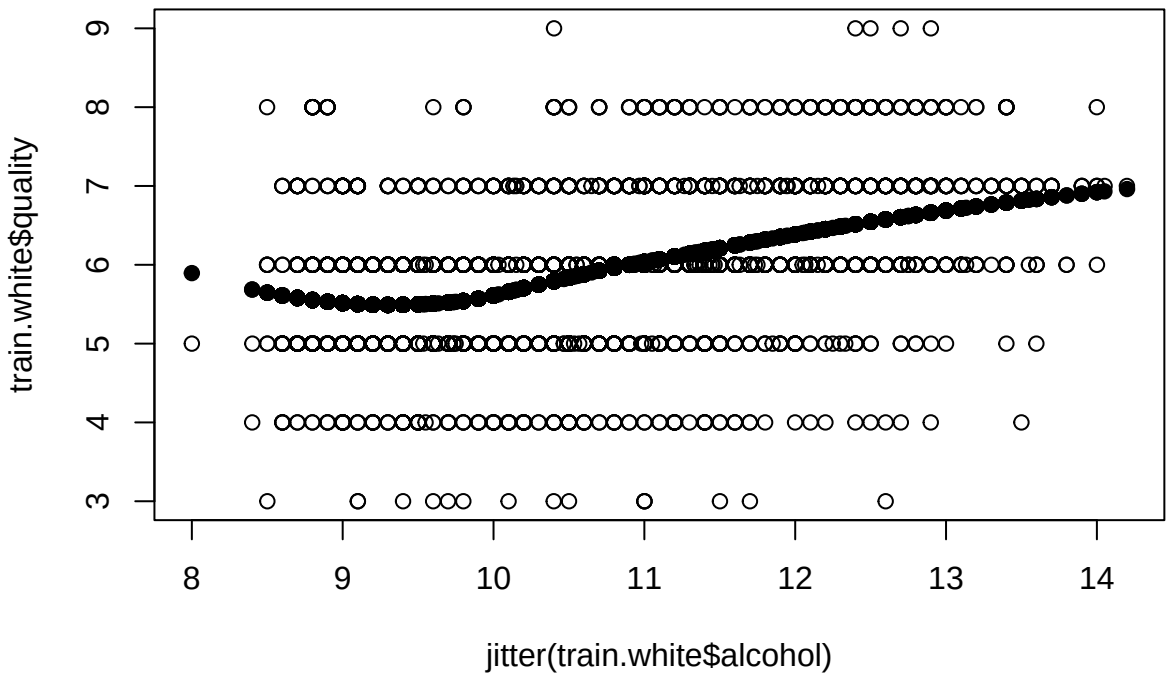

```
plot(jitter(train.white$pH), train.white$quality)
points(train.white$pH, loess(train.white$quality~train.white$pH)$fitted, pch=19)
```




```
plot(jitter(train.white$sulphates), train.white$quality)
points(train.white$sulphates, loess(train.white$quality~train.white$sulphates)$fitted, pch=19)
```

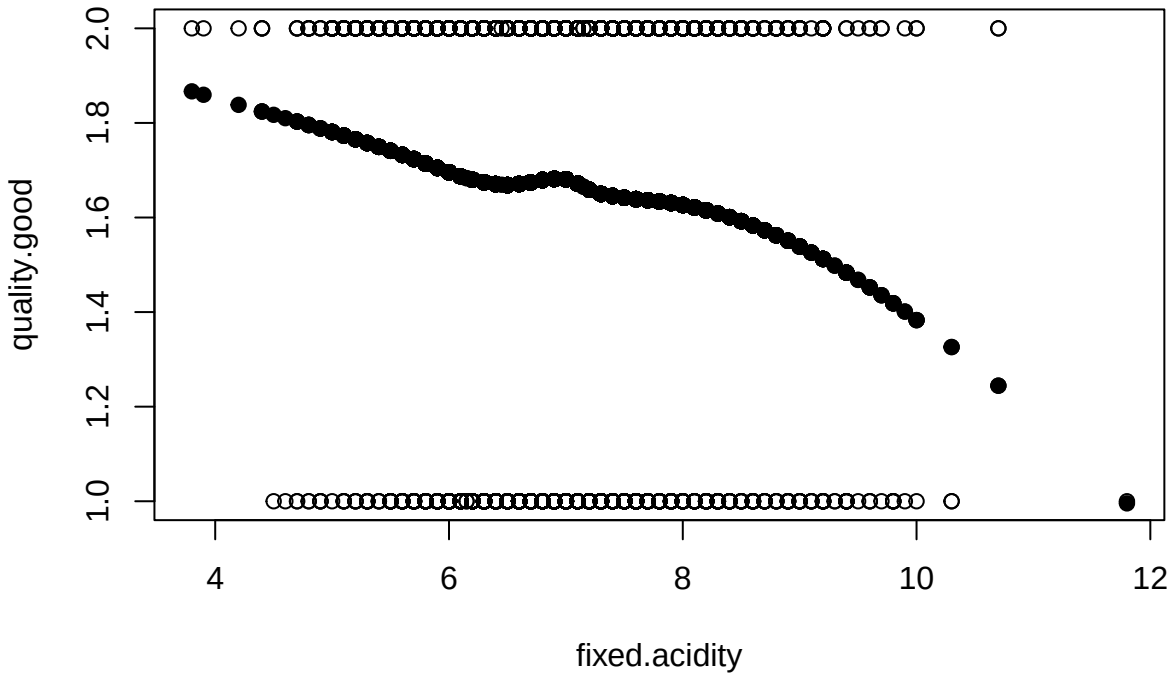



```
plot(jitter(train.white$alcohol), train.white$quality)
points(train.white$alcohol, loess(train.white$quality~train.white$alcohol)$fitted, pch=19)
```

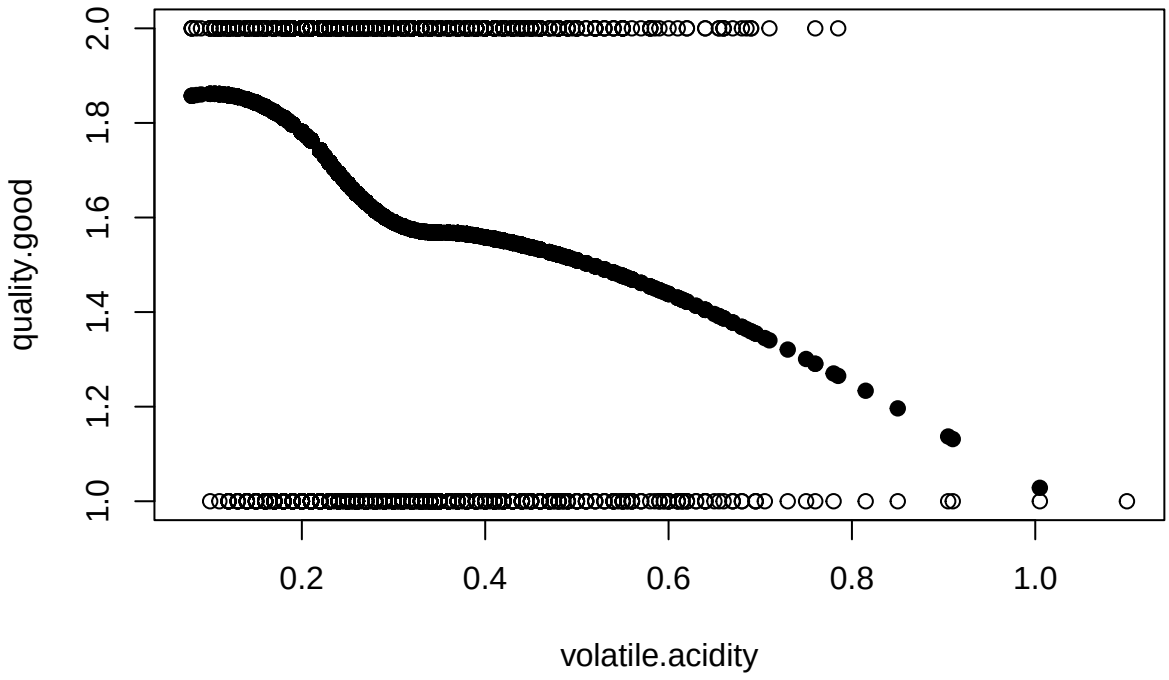


plotting to see possible pattern between eash predictor and the binary classification response 'quality.good.' Curves Look somewhat similar to plotting when using regression response, a continuous variable focusing on 7 ratings of quality rather than 2

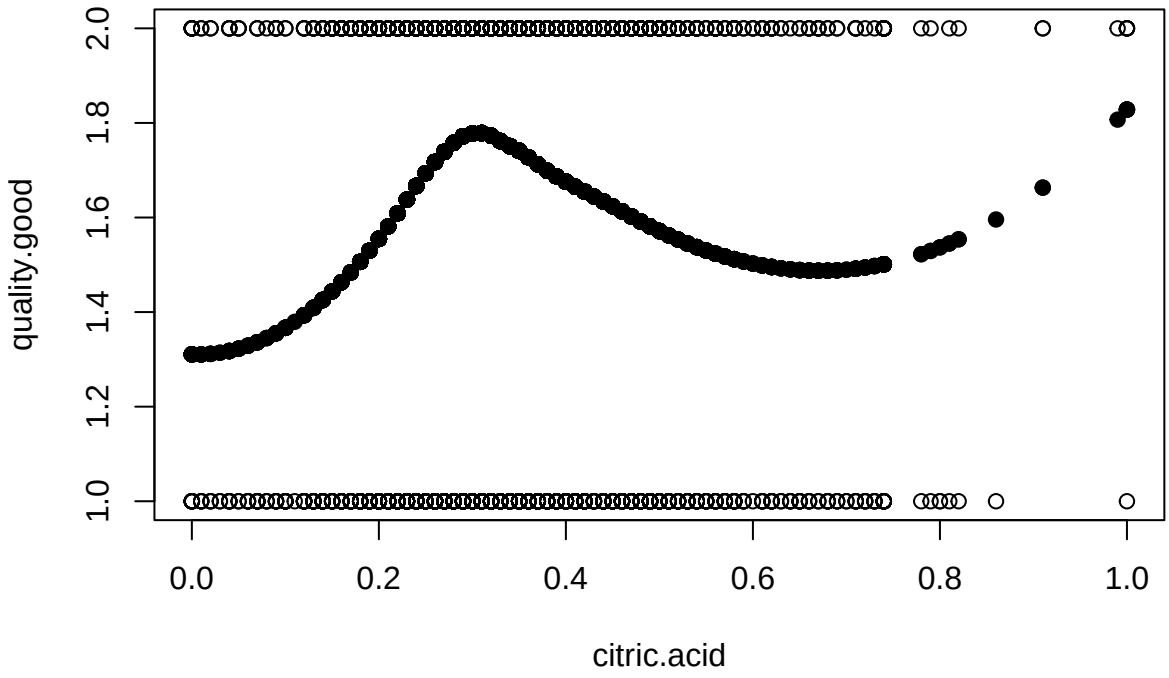
```
plot(train.white$fixed.acidity, train.white$quality.good, ylab = 'quality.good', xlab = 'fixed.acidity'
points(train.white$fixed.acidity, loess(as.numeric(train.white$quality.good)~train.white$fixed.acidity)
```




```
plot(train.white$volatile.acidity, train.white$quality.good, ylab = 'quality.good', xlab = 'volatile.aci
points(train.white$volatile.acidity, loess(as.numeric(train.white$quality.good)~train.white$volatile.aci
```

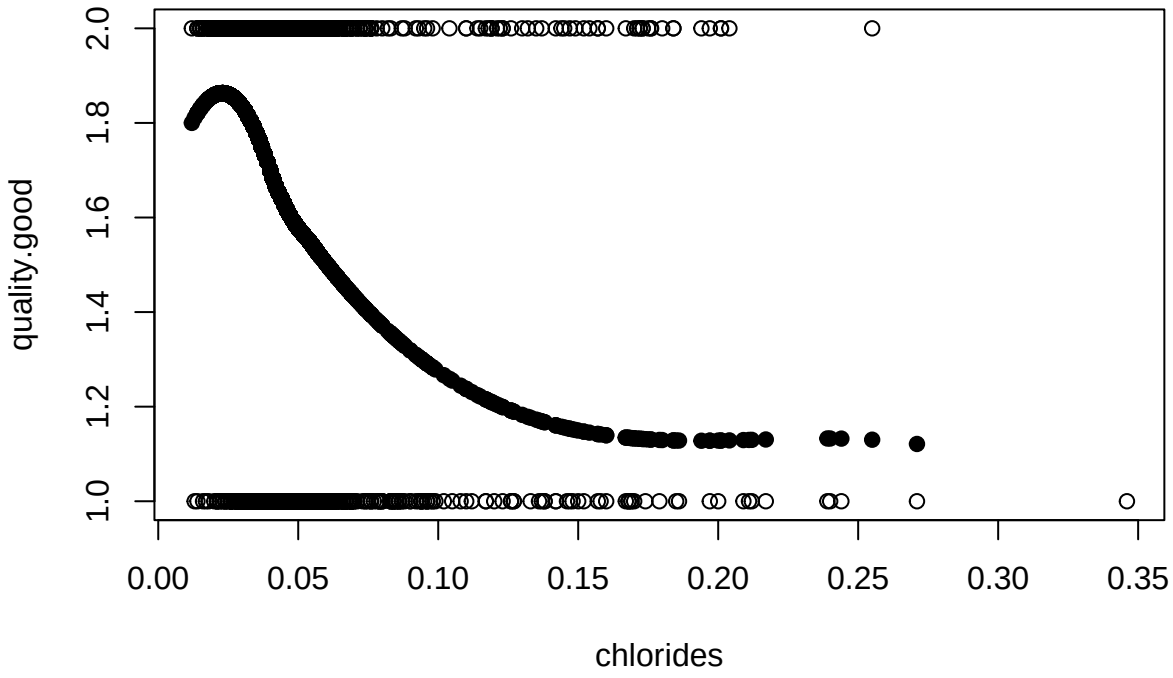



```
plot(train.white$citric.acid, train.white$quality.good, ylab = 'quality.good', xlab = 'citric.acid')
points(train.white$citric.acid, loess(as.numeric(train.white$quality.good)~train.white$citric.acid)$fit
```

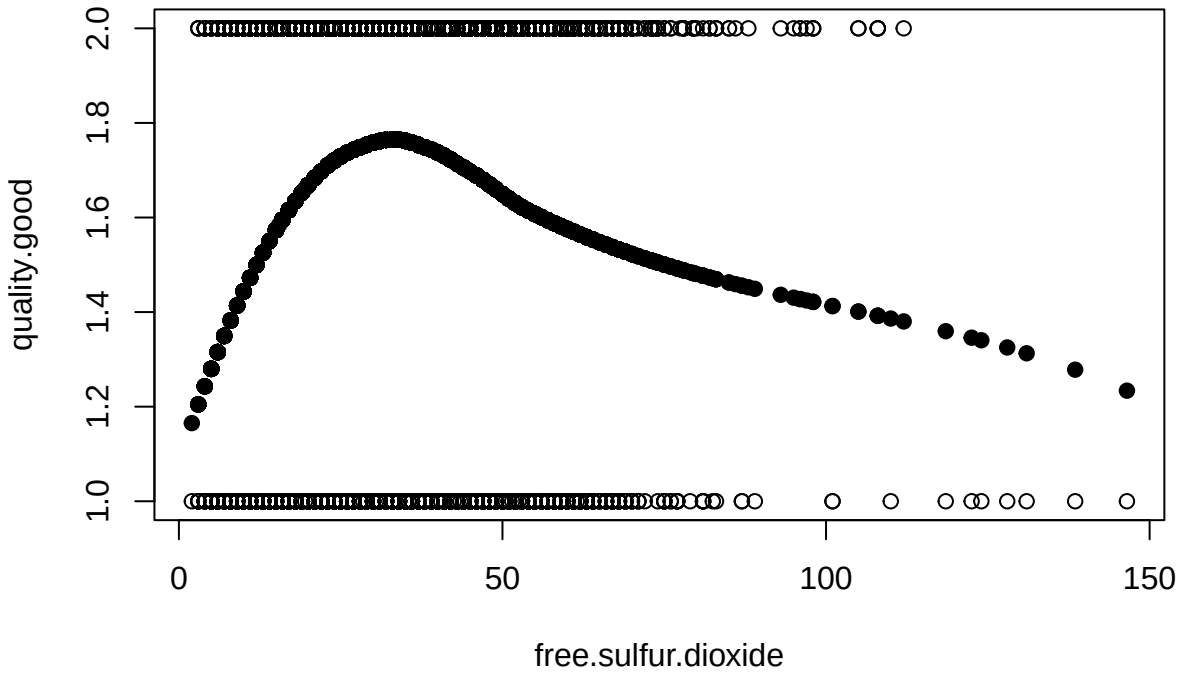



```
plot(train.white$residual.sugar, train.white$quality.good, ylab = 'quality.good', xlab = 'residual.sugar')
points(train.white$residual.sugar, loess(as.numeric(train.white$quality.good)~train.white$residual.sugar)
```

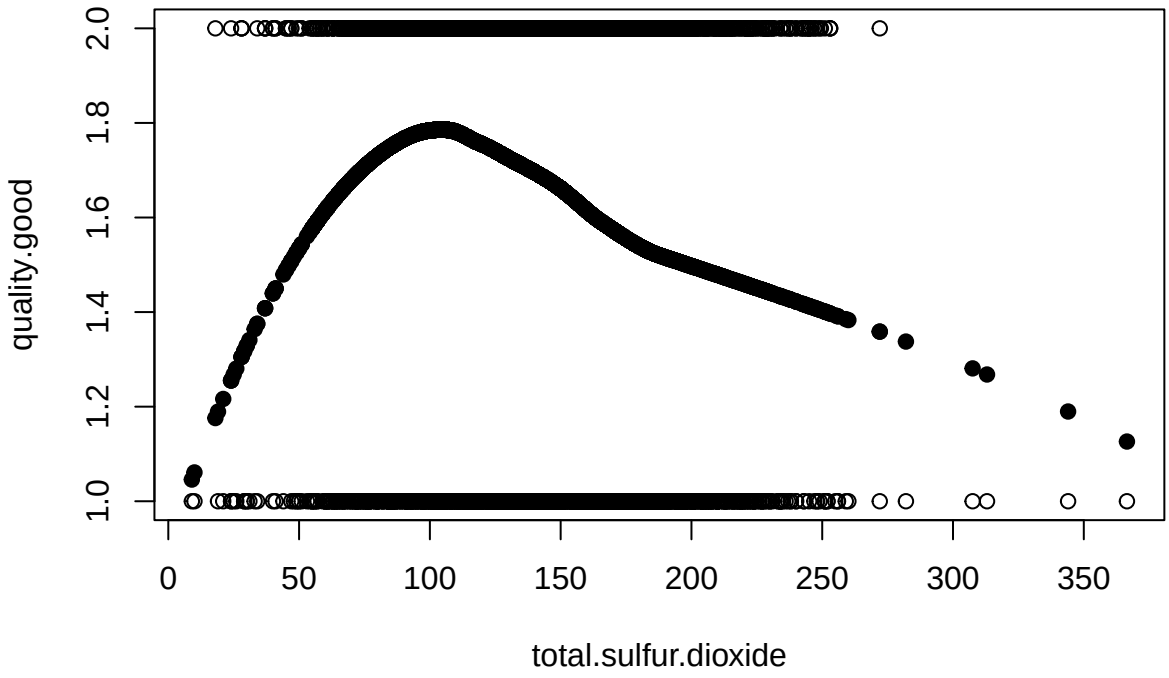

```
plot(train.white$chlorides, train.white$quality.good, ylab = 'quality.good', xlab = 'chlorides')
points(train.white$chlorides, loess(as.numeric(train.white$quality.good)~train.white$chlorides)$fitted,
```



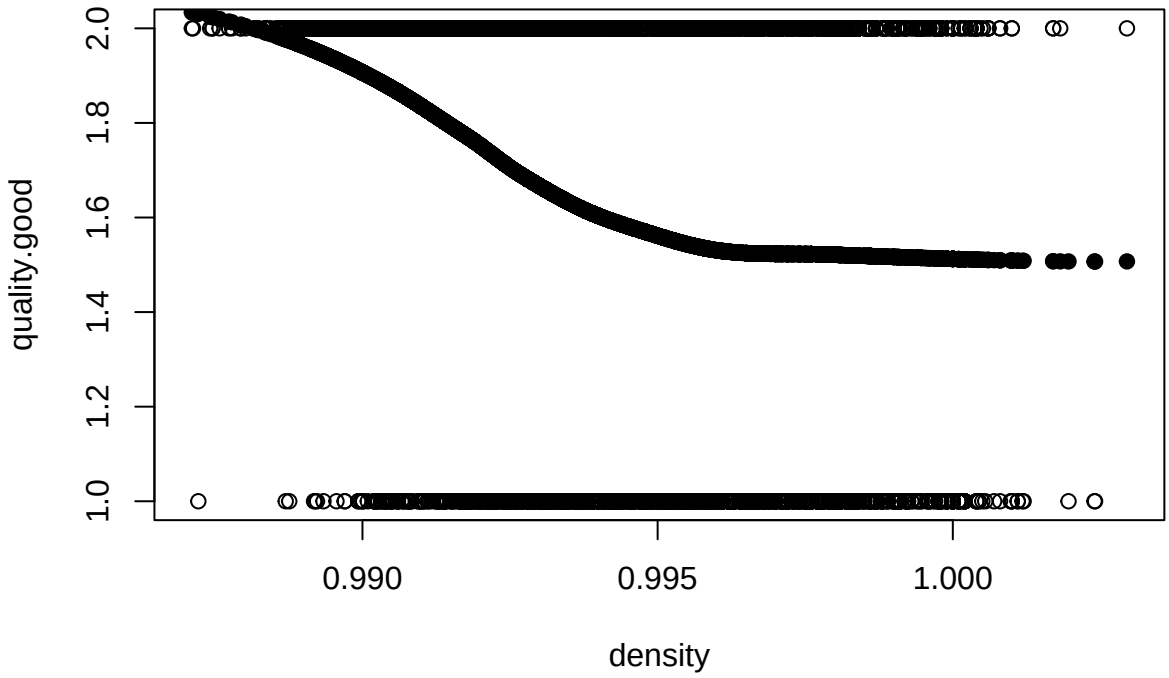

```
plot(train.white$free.sulfur.dioxide, train.white$quality.good, ylab = 'quality.good', xlab = 'free.sul
points(train.white$free.sulfur.dioxide, loess(as.numeric(train.white$quality.good)~train.white$free.sul
```



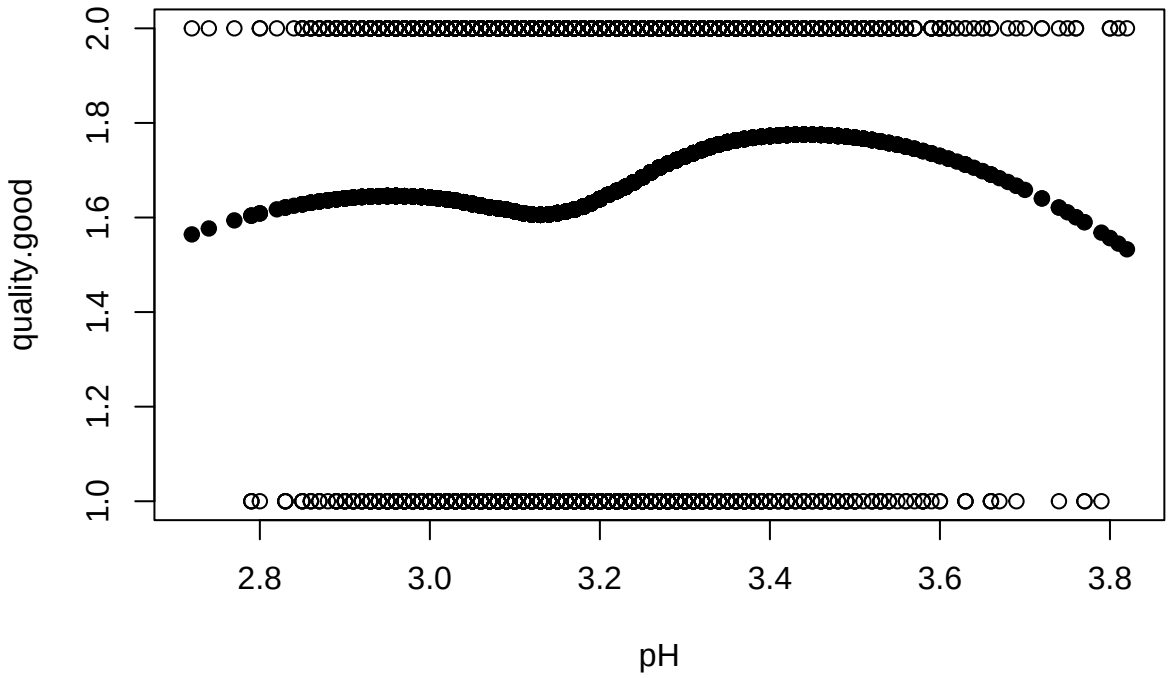

```
plot(train.white$total.sulfur.dioxide, train.white$quality.good, ylab = 'quality.good', xlab = 'total.su
points(train.white$total.sulfur.dioxide, loess(as.numeric(train.white$quality.good)~train.white$total.su
```



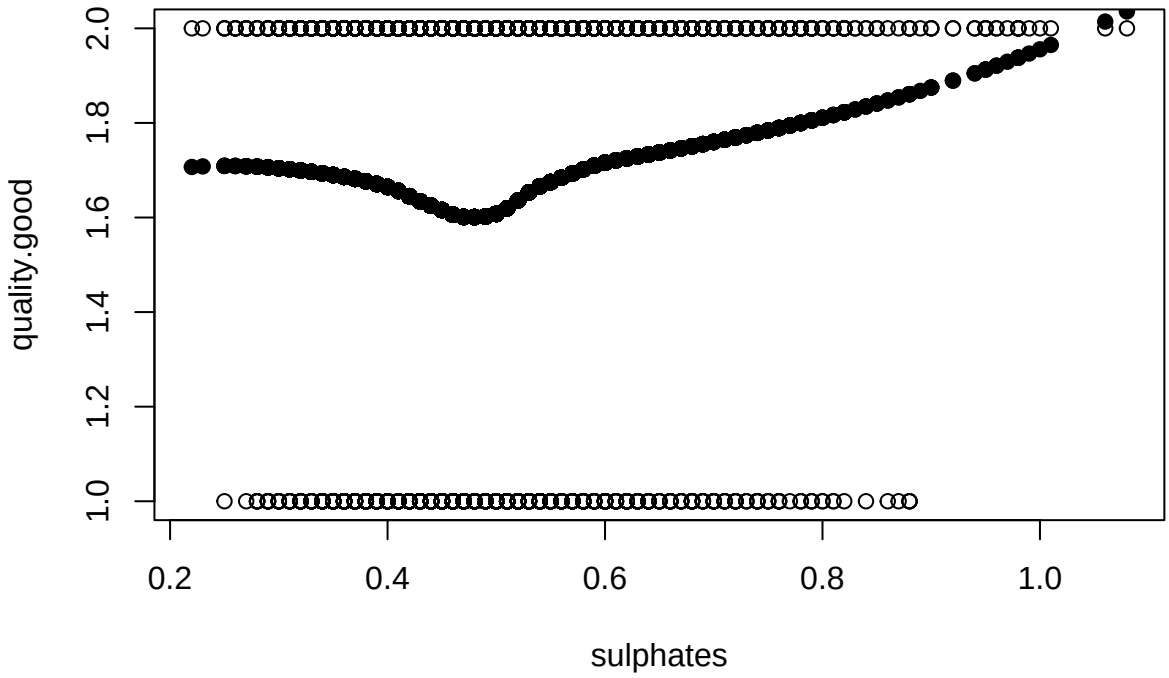

```
plot(train.white$density, train.white$quality.good, ylab = 'quality.good', xlab = 'density')
points(train.white$density, loess(as.numeric(train.white$quality.good)~train.white$density)$fitted, pch=
```



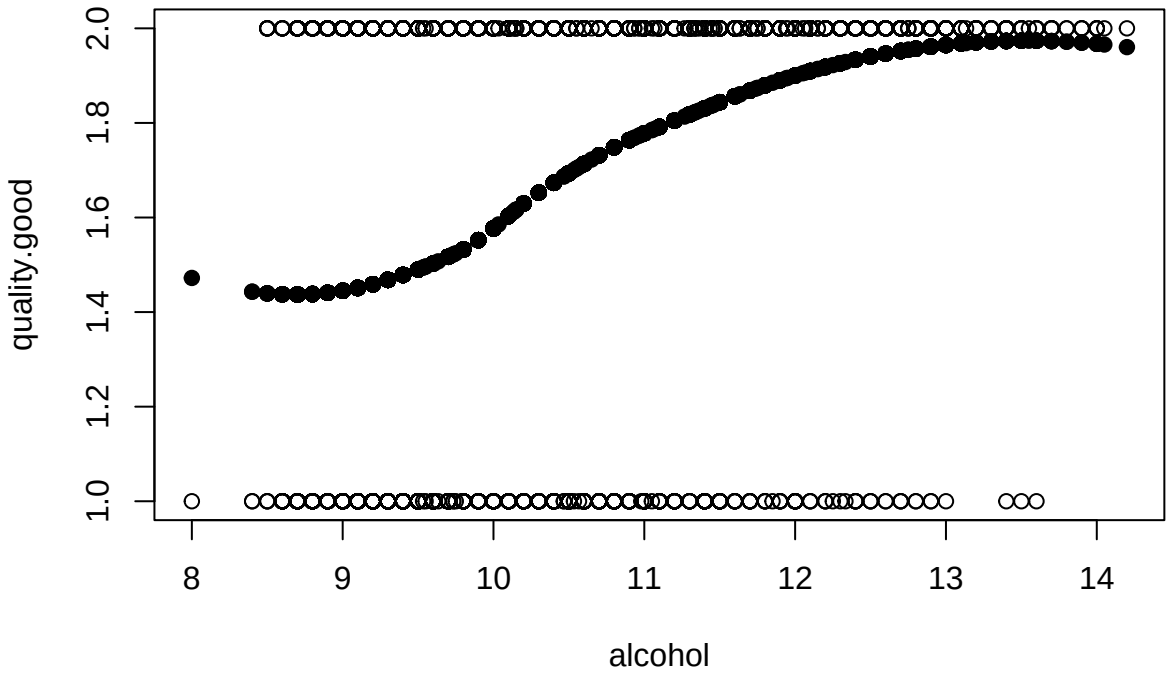

```
plot(train.white$pH, train.white$quality.good, ylab = 'quality.good', xlab = 'pH')
points(train.white$pH, loess(as.numeric(train.white$quality.good)~train.white$pH)$fitted, pch=19)
```




```
plot(train.white$sulphates, train.white$quality.good, ylab = 'quality.good', xlab = 'sulphates')
points(train.white$sulphates, loess(as.numeric(train.white$quality.good)~train.white$sulphates)$fitted,
```




```
plot(train.white$alcohol, train.white$quality.good, ylab = 'quality.good', xlab = 'alcohol')
points(train.white$alcohol, loess(as.numeric(train.white$quality.good)~train.white$alcohol)$fitted, pch=
```

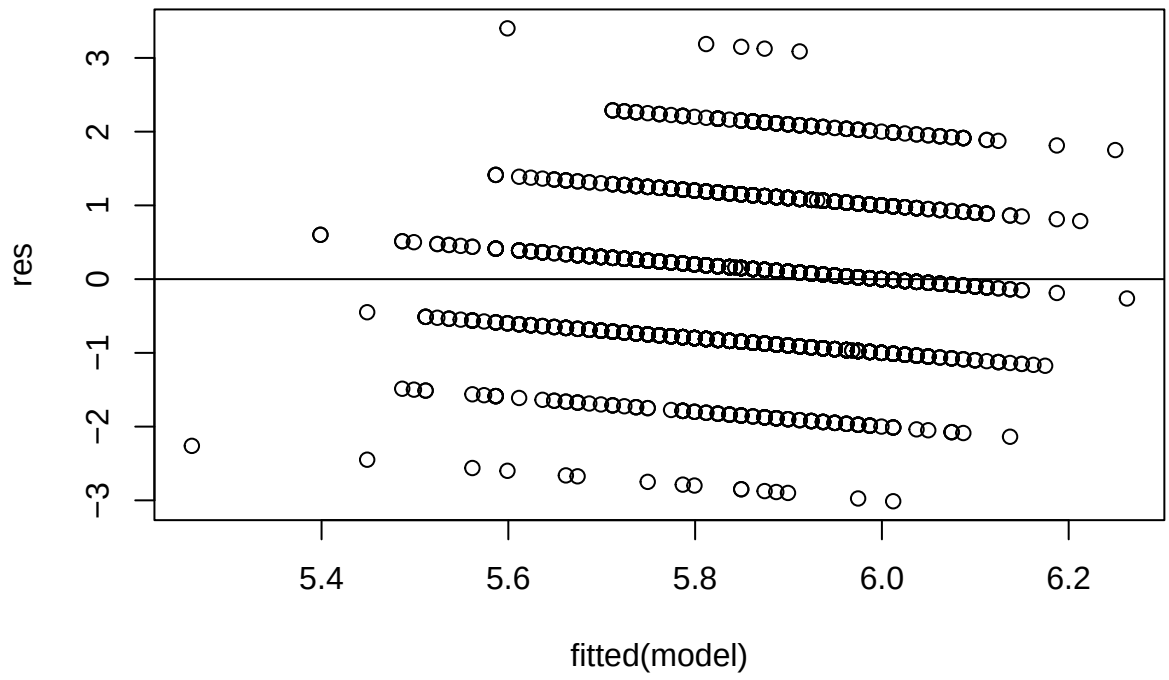


Residual Plots

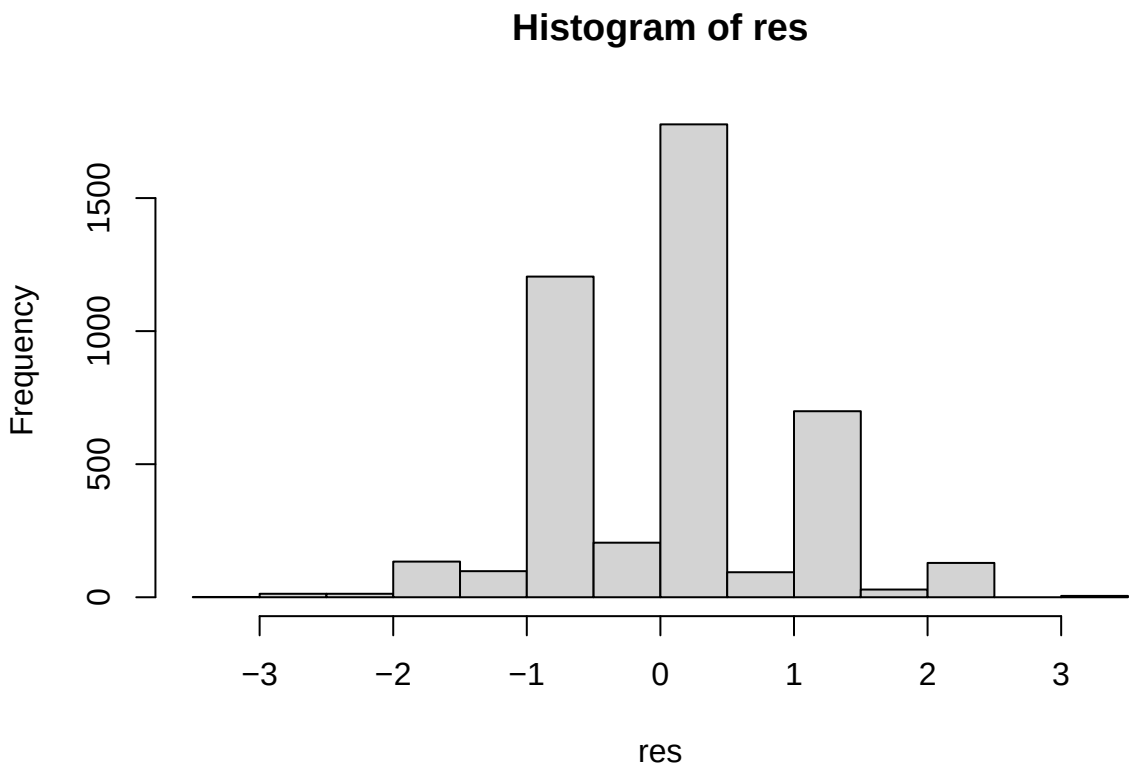
MLR model Assumptions and tests: - Linearity; test by creating a scatterplot of residuals vs predictor variable and observing no structure or pattern - Independent Observations; test by looking at Scatterplot of residuals vs the predictor variable and observing that they don't show pattern – Error terms Centered around zero - Normality of error terms; test by looking at a histogram of the residuals and see if there is a bell curve - Equal variances of error terms; test by observing Scatterplot of residuals vs fitted (or predictor variables) don't show funneling - X is nonstochastic and rank(x) » n; we know this for sure already because the x are not random and 4402 » 11

fixed acidity

```
model <- lm(quality~(fixed.acidity), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```



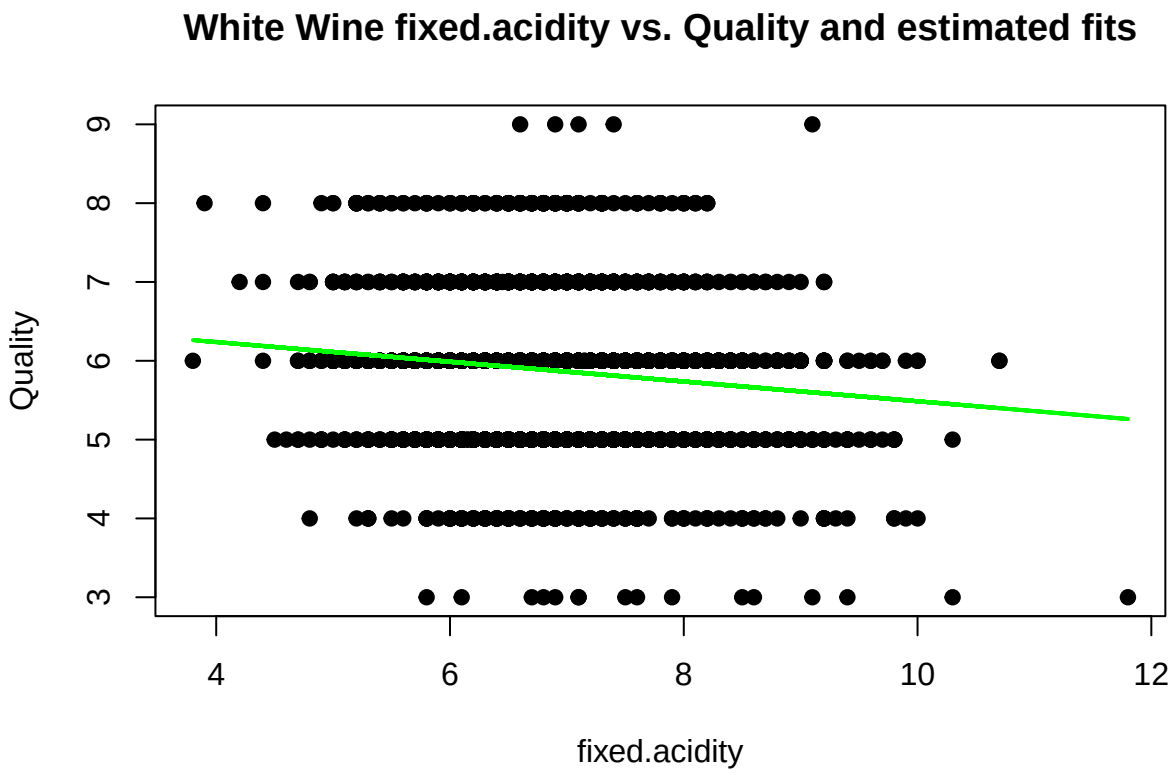

```
# produce histogram of residuals
hist(res)
```




```
#durbinWatsonTest(model)
```

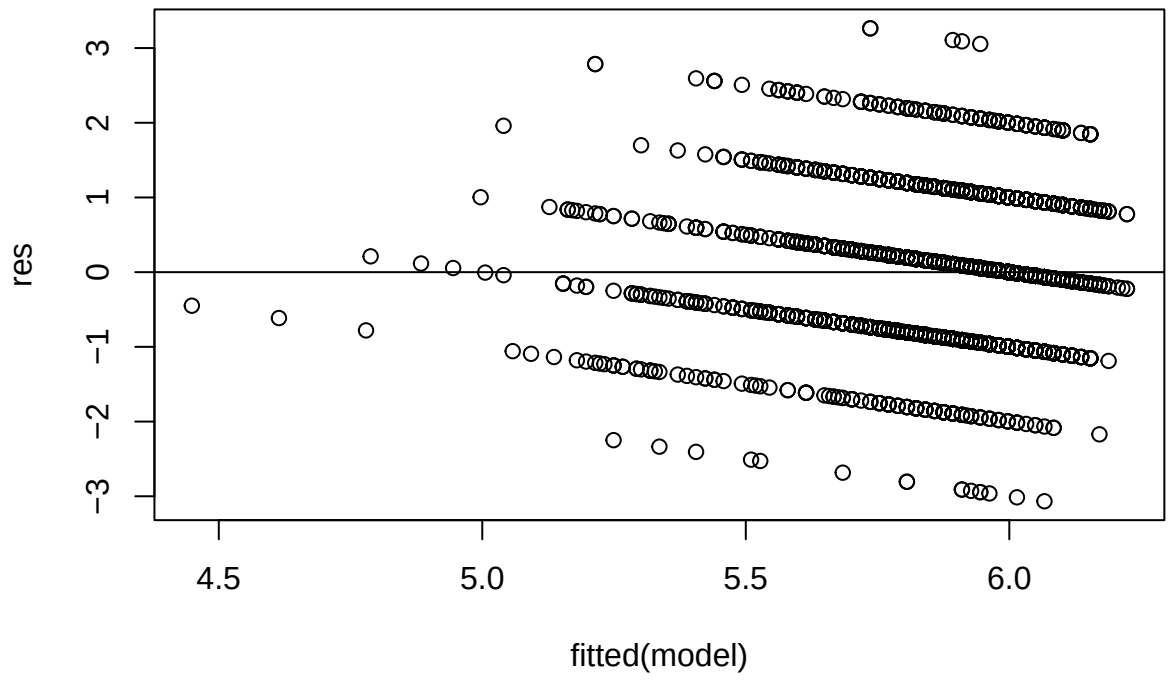
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$fixed.acidity,train.white$quality,pch=19,xlab='fixed.acidity',ylab='Quality',main='Whi
lines(train.white$fixed.acidity,model$fitted.values,col='green',lwd=2)
```

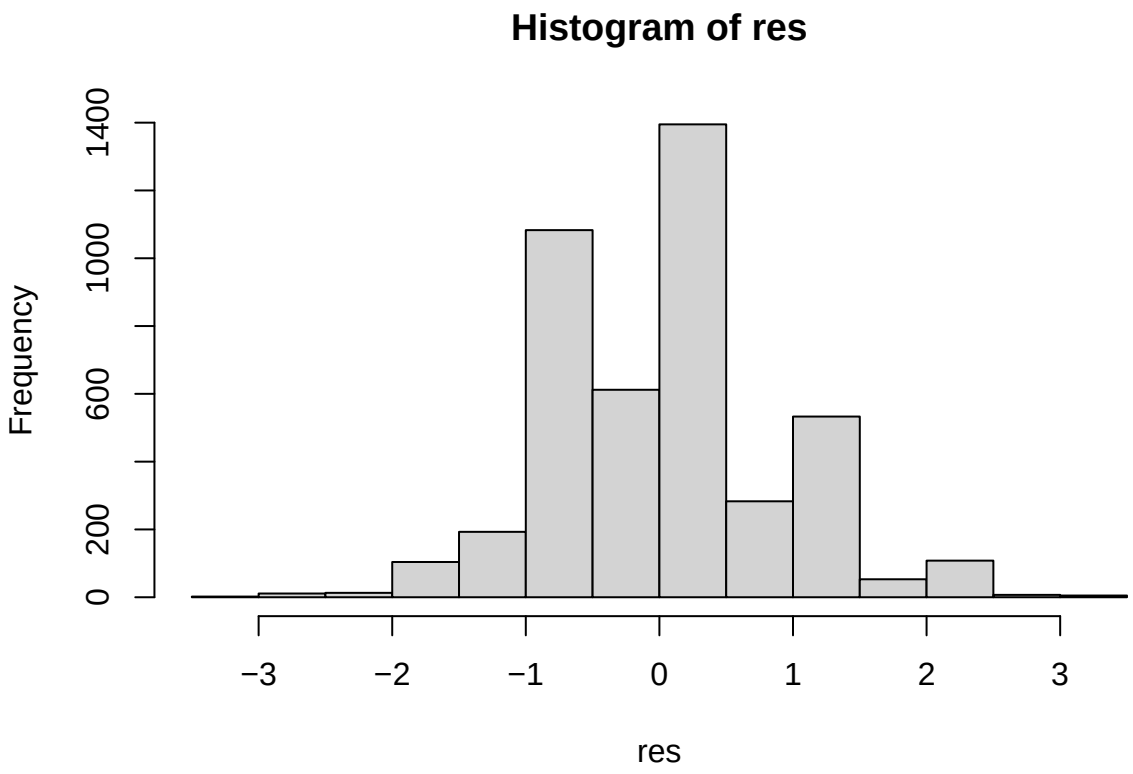


volatile.acidity before transformation

```
model <- lm(quality~(volatile.acidity), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

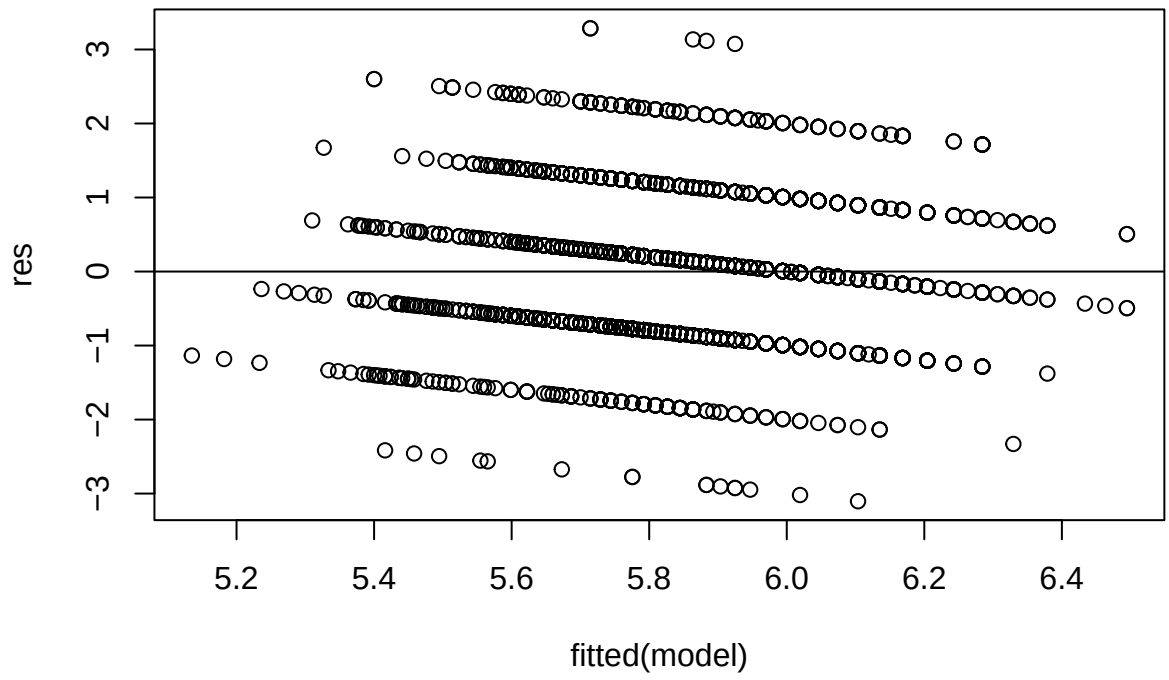



```
# produce histogram of residuals
hist(res)
```

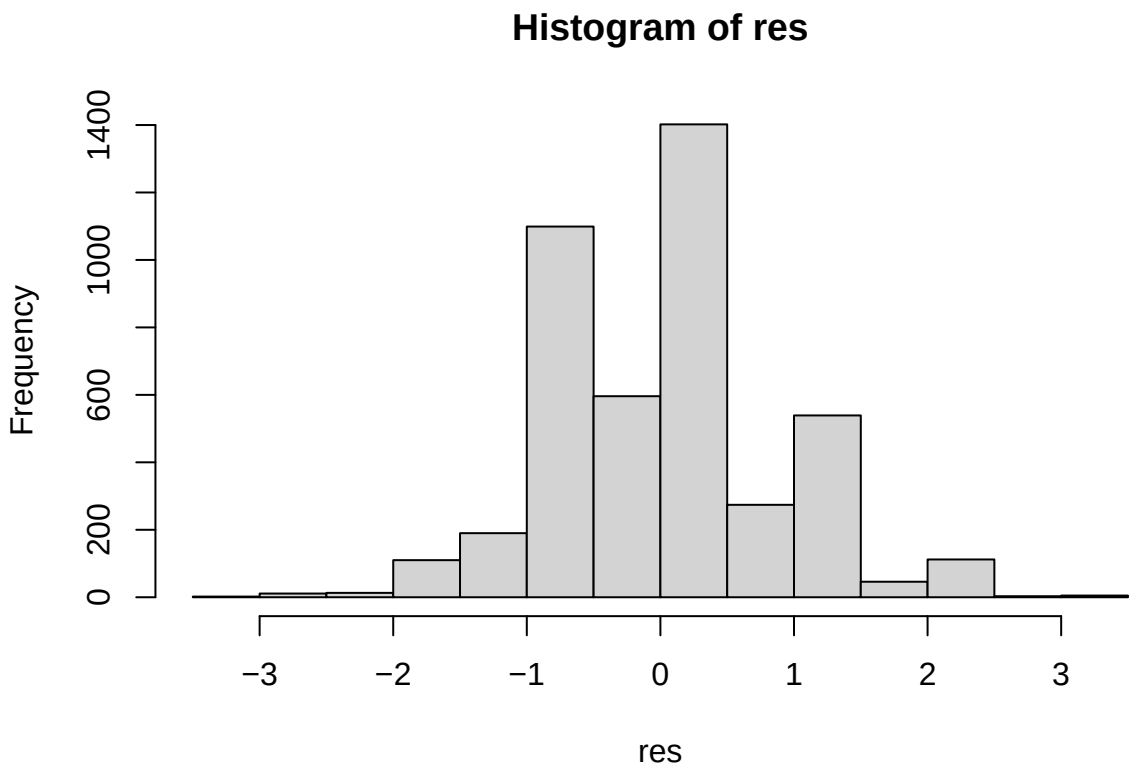


volatile.acidity after transformation

```
model <- lm(quality~log(volatile.acidity), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

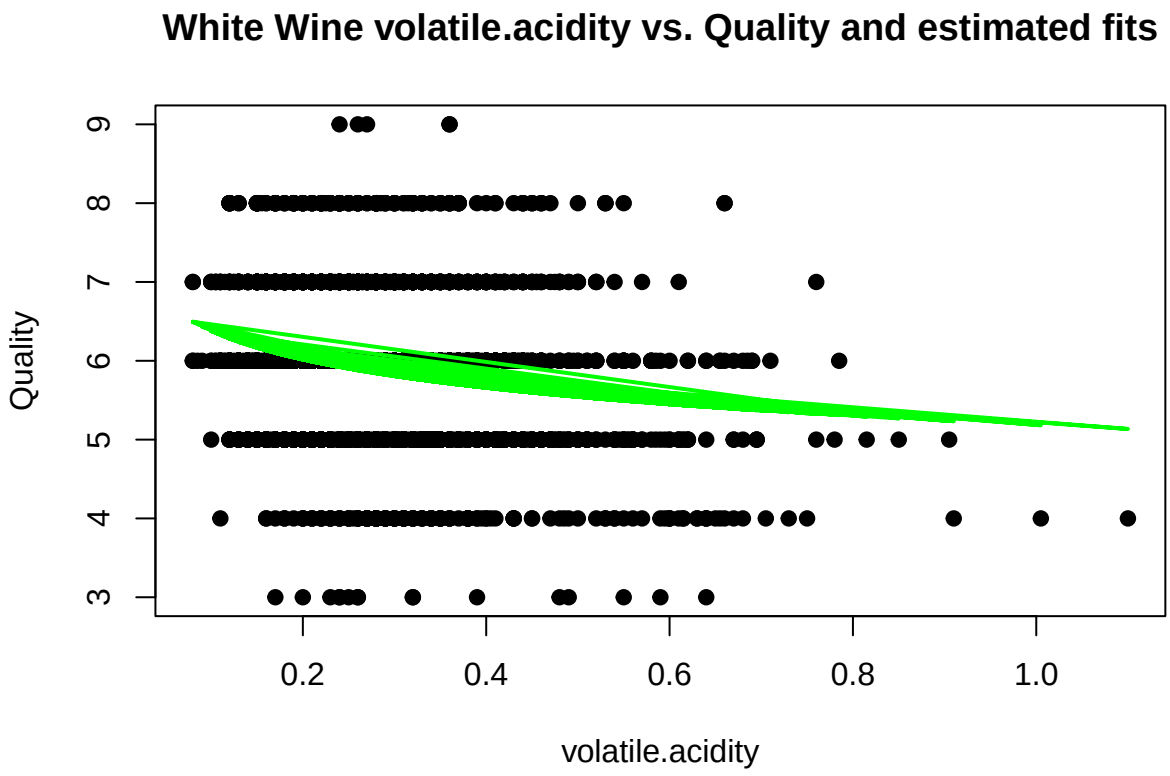



```
# produce histogram of residuals
hist(res)
```



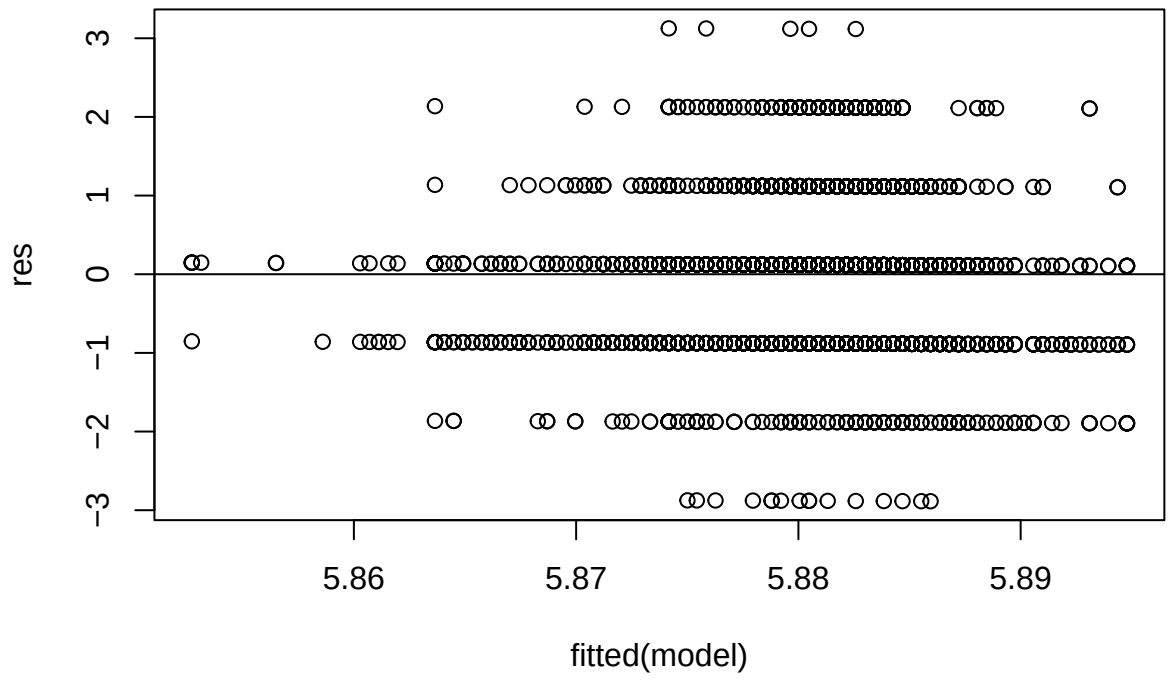
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$volatile.acidity,train.white$quality,pch=19,xlab='volatile.acidity',ylab='Quality',mai
lines(train.white$volatile.acidity,model$fitted.values,col='green',lwd=2)
```

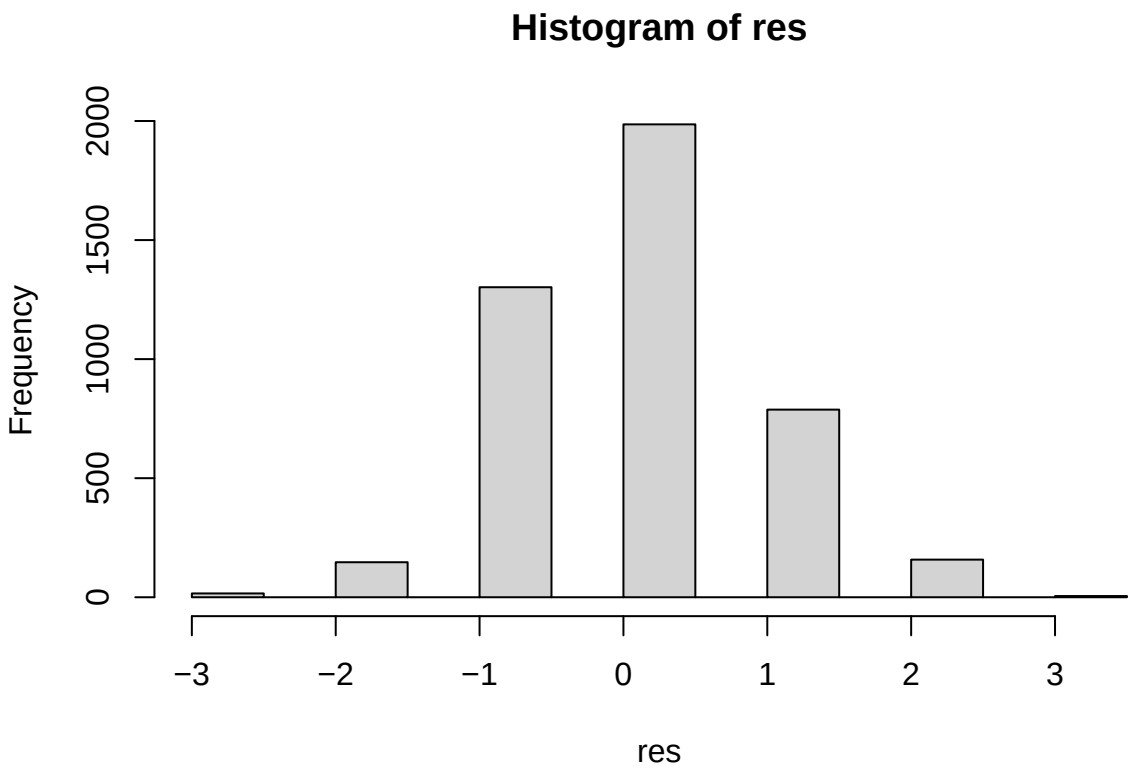


Citric Acid

```
model <- lm(quality~citric.acid, data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

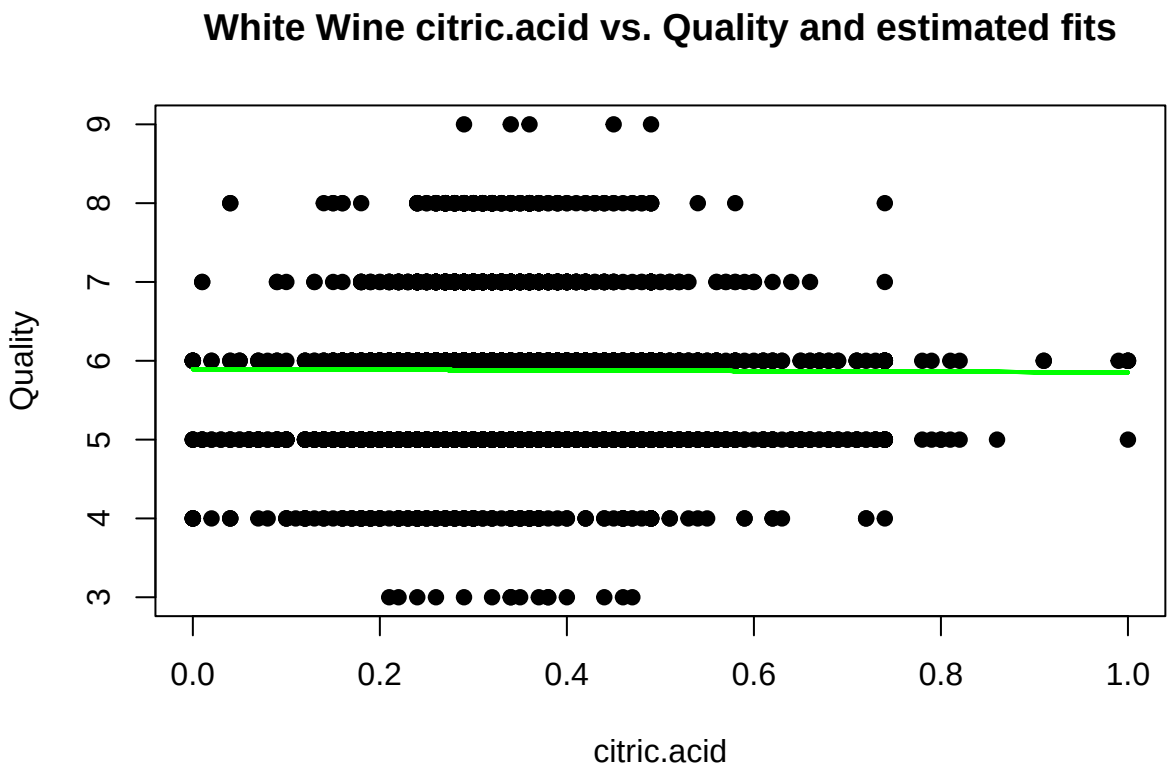



```
# produce histogram of residuals
hist(res)
```



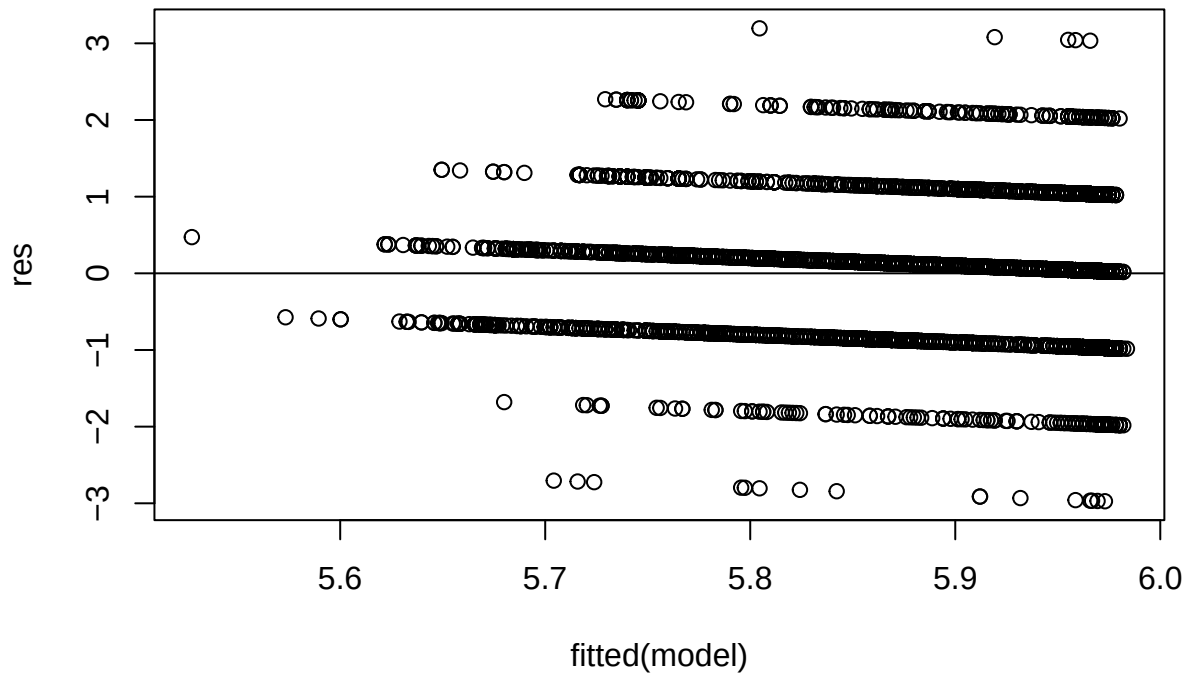
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$citric.acid,train.white$quality,pch=19,xlab='citric.acid',ylab='Quality',main='White W
lines(train.white$citric.acid,model$fitted.values,col='green',lwd=2)
```

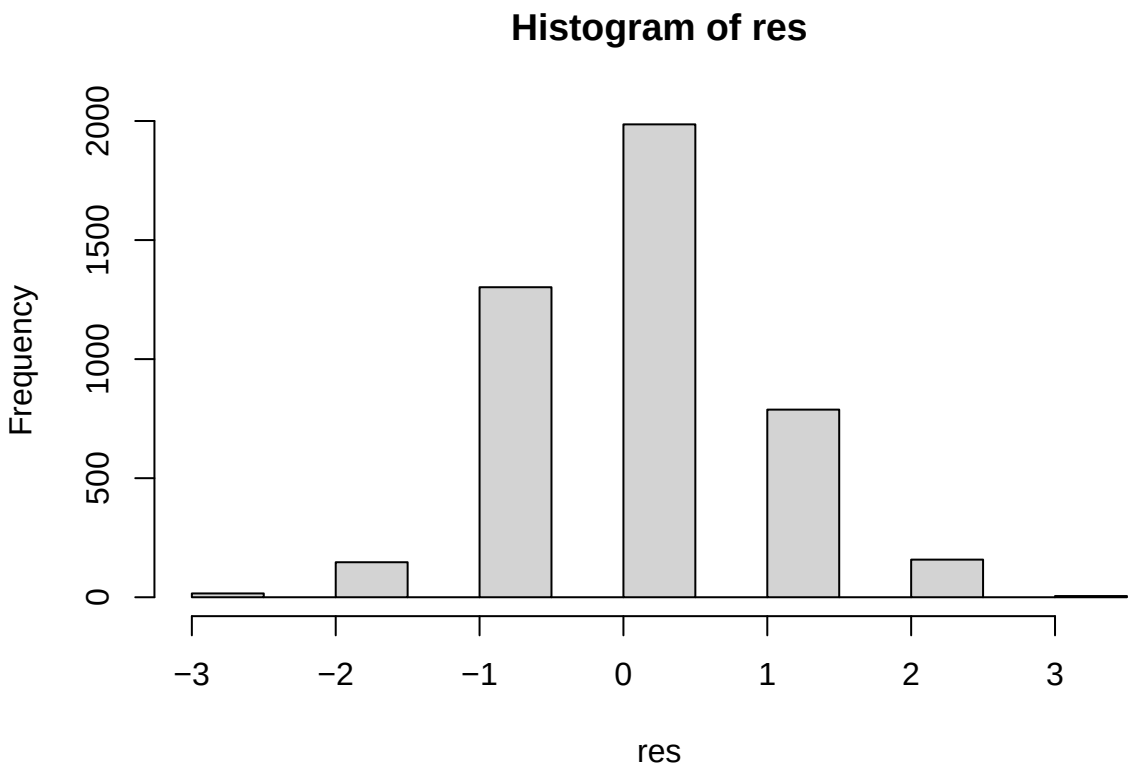


residual.sugar before transformation

```
model <- lm(quality~(residual.sugar), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

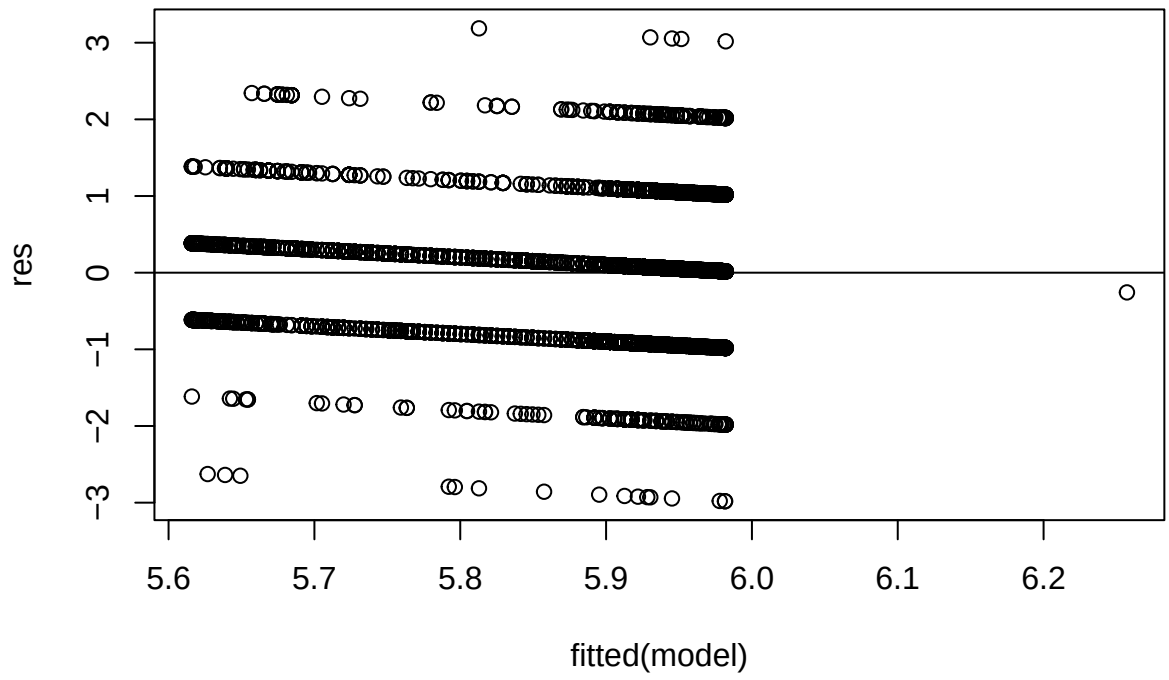



```
# produce histogram of residuals
hist(res)
```

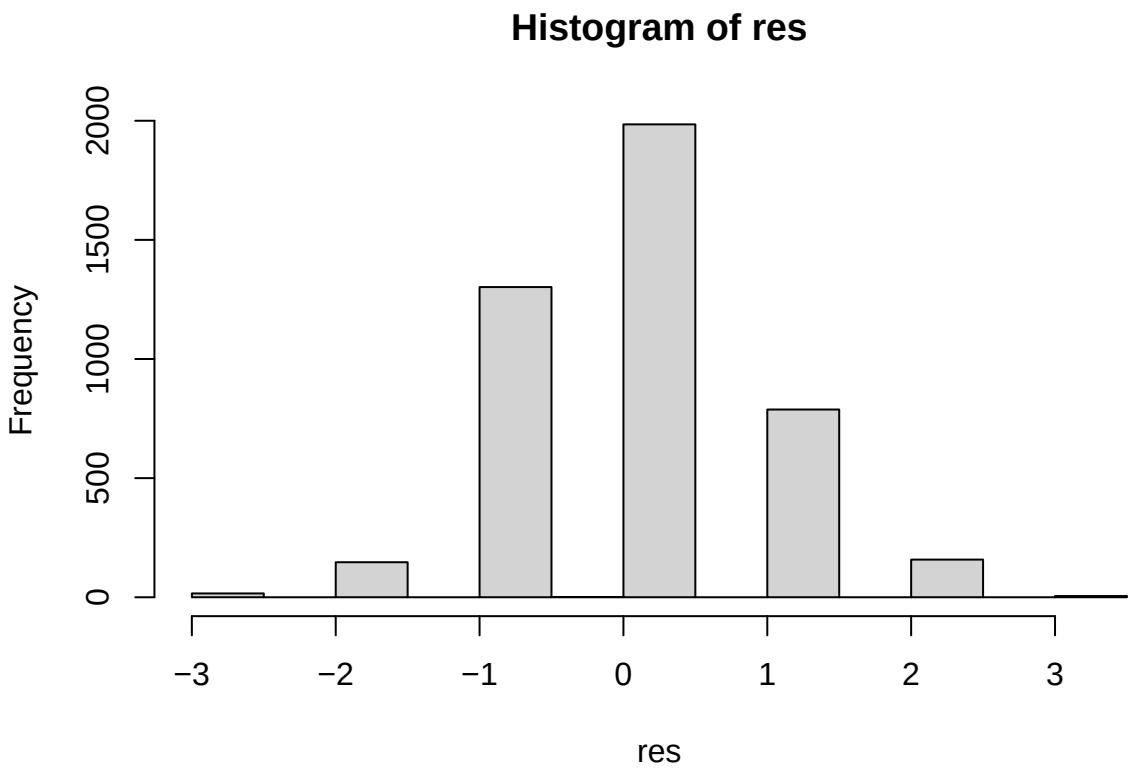


residual.sugar after transformation

```
model <- lm(quality~poly(residual.sugar, 3), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

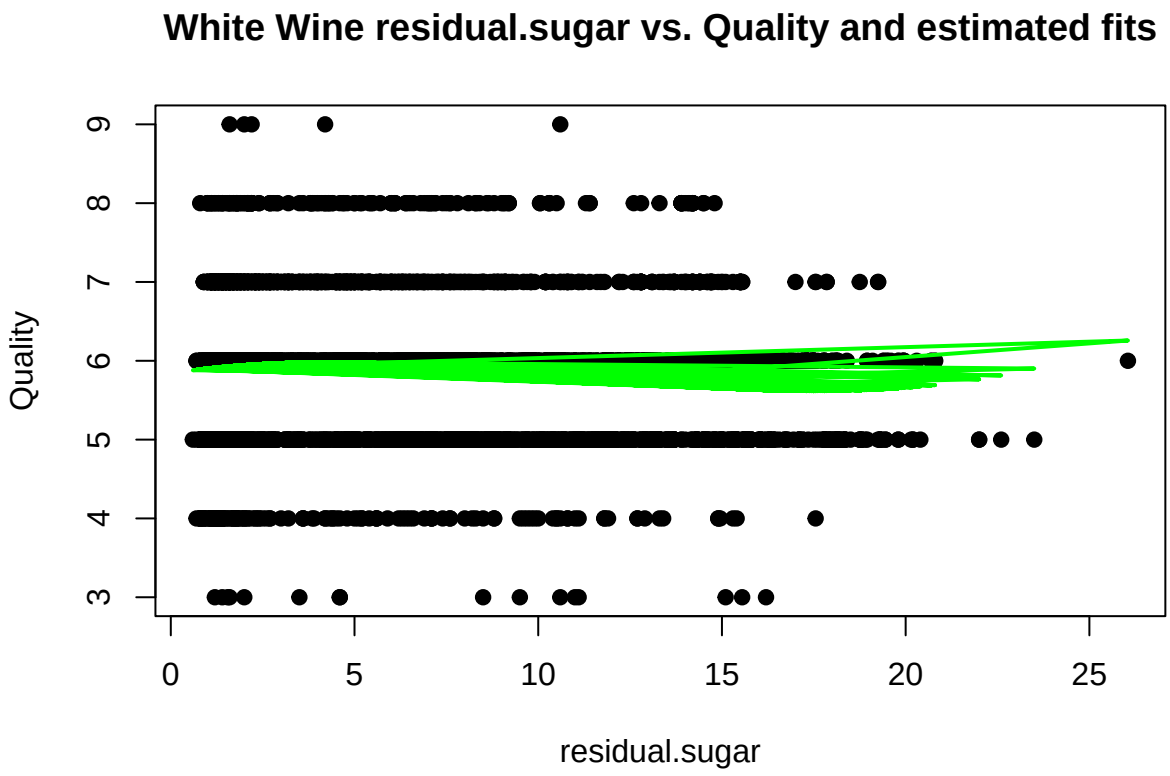



```
# produce histogram of residuals
hist(res)
```



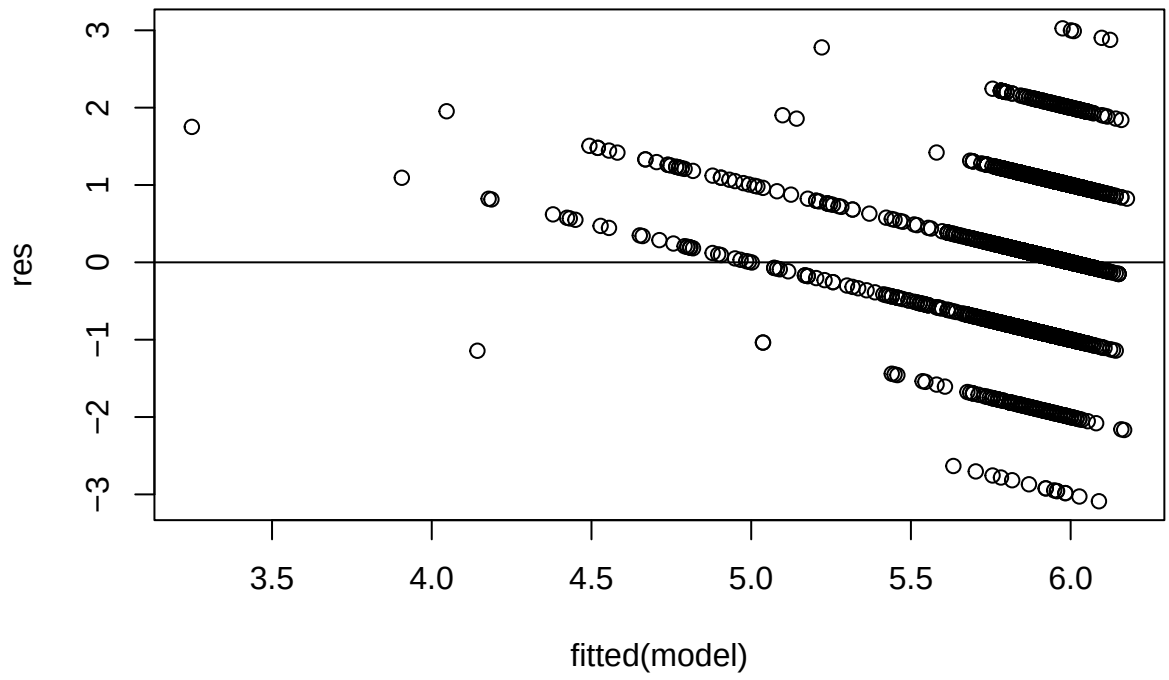
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$residual.sugar,train.white$quality,pch=19,xlab='residual.sugar',ylab='Quality',main='Wh
lines(train.white$residual.sugar,model$fitted.values,col='green',lwd=2)
```

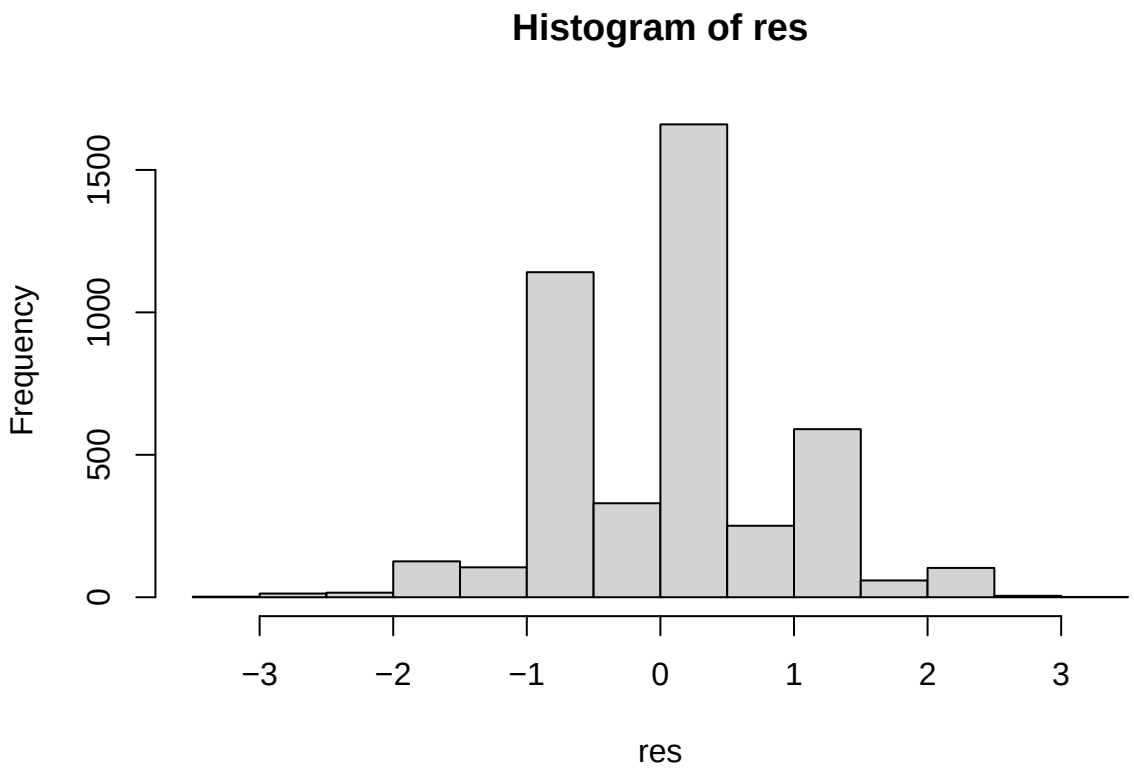


chlorides before transformation

```
model <- lm(quality~(chlorides), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

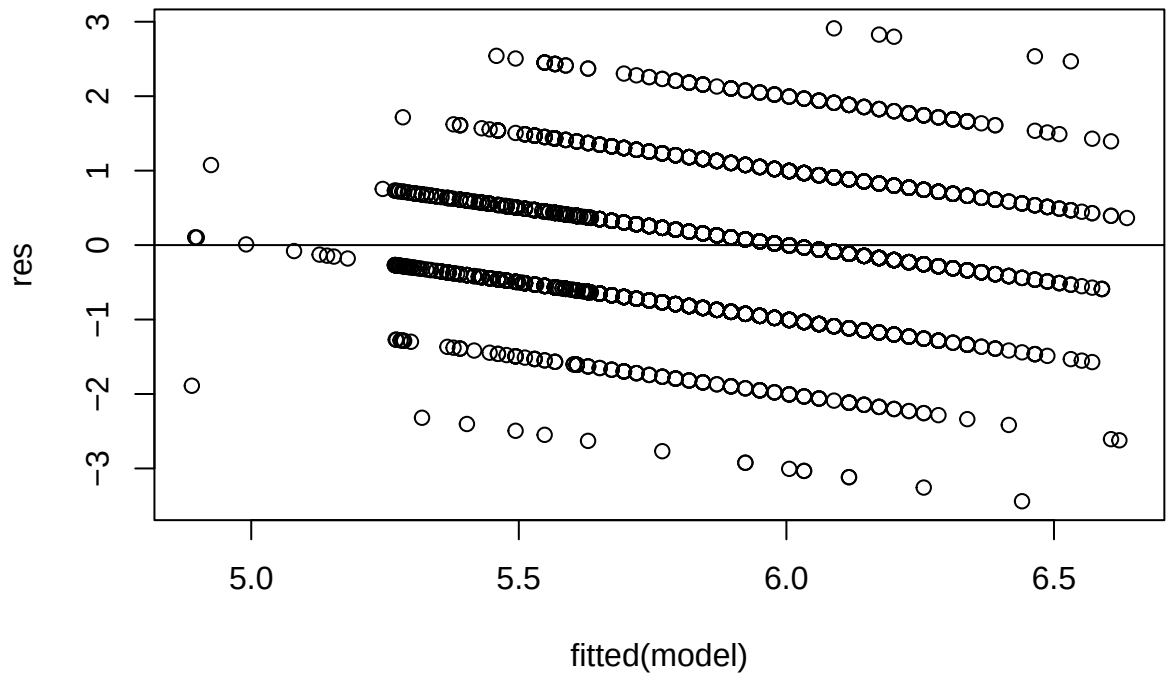



```
# produce histogram of residuals
hist(res)
```

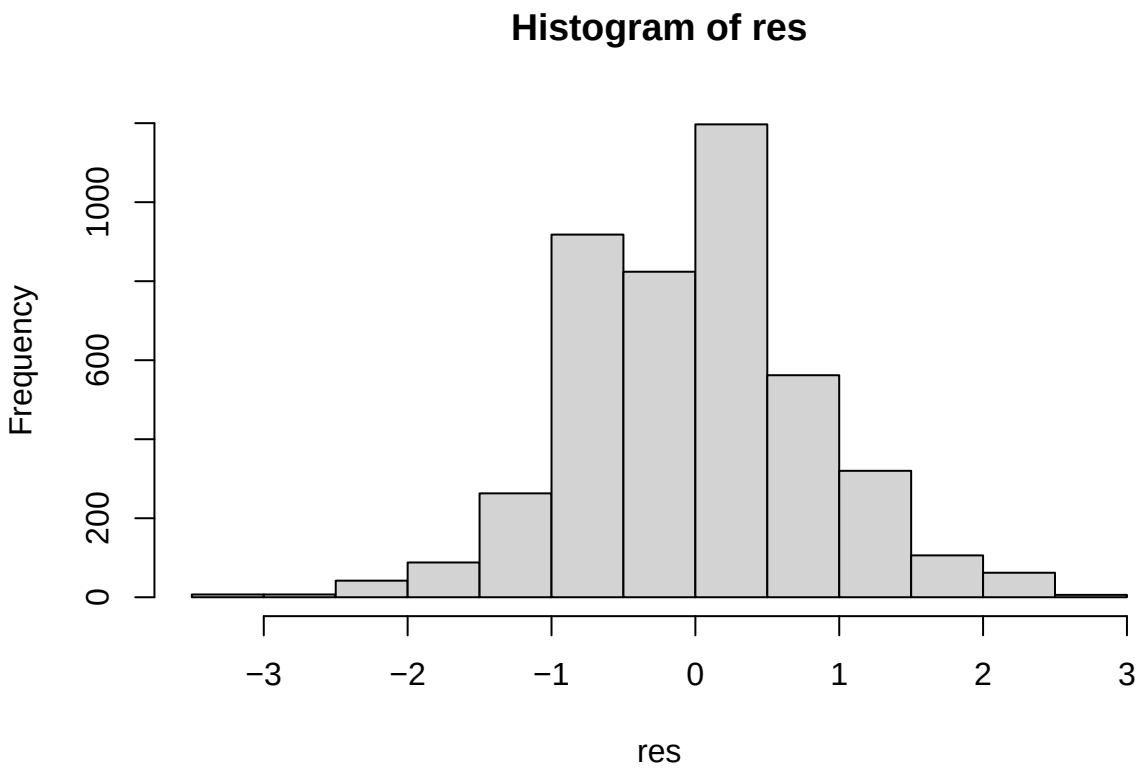


chlorides after transformation

```
model <- lm(quality~poly(chlorides,6), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

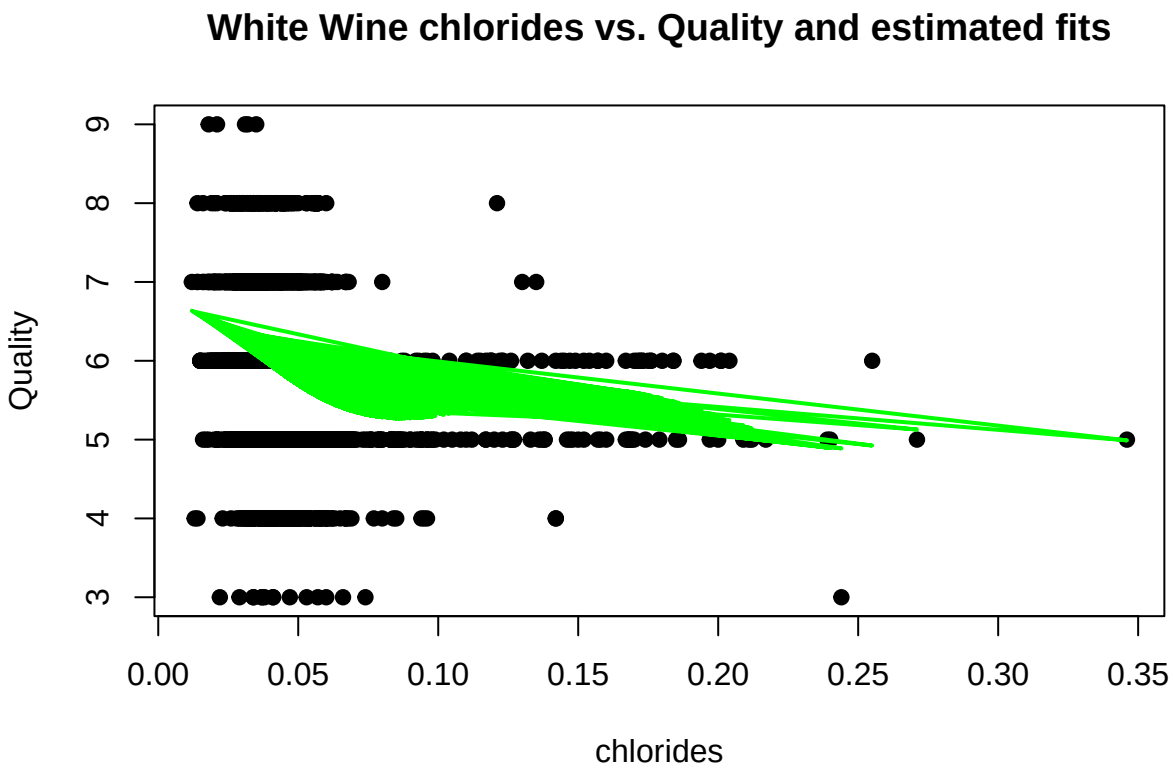



```
# produce histogram of residuals
hist(res)
```



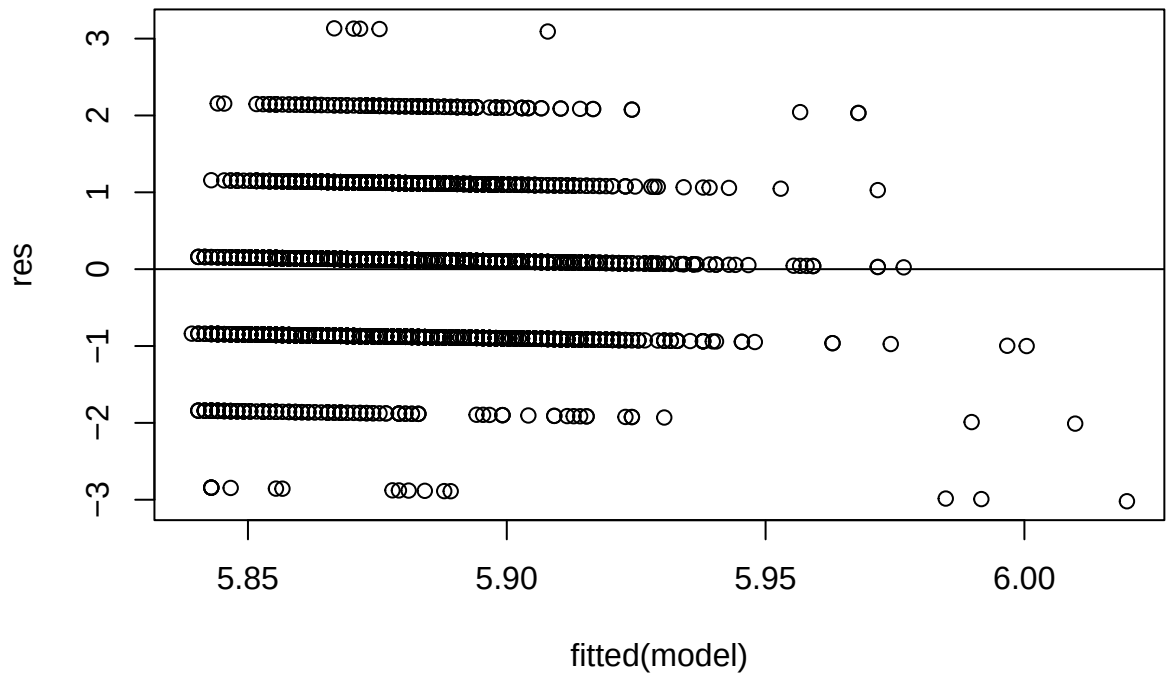
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$chlorides,train.white$quality,pch=19,xlab='chlorides',ylab='Quality',main='White Wine
lines(train.white$chlorides,model$fitted.values,col='green',lwd=2)
```

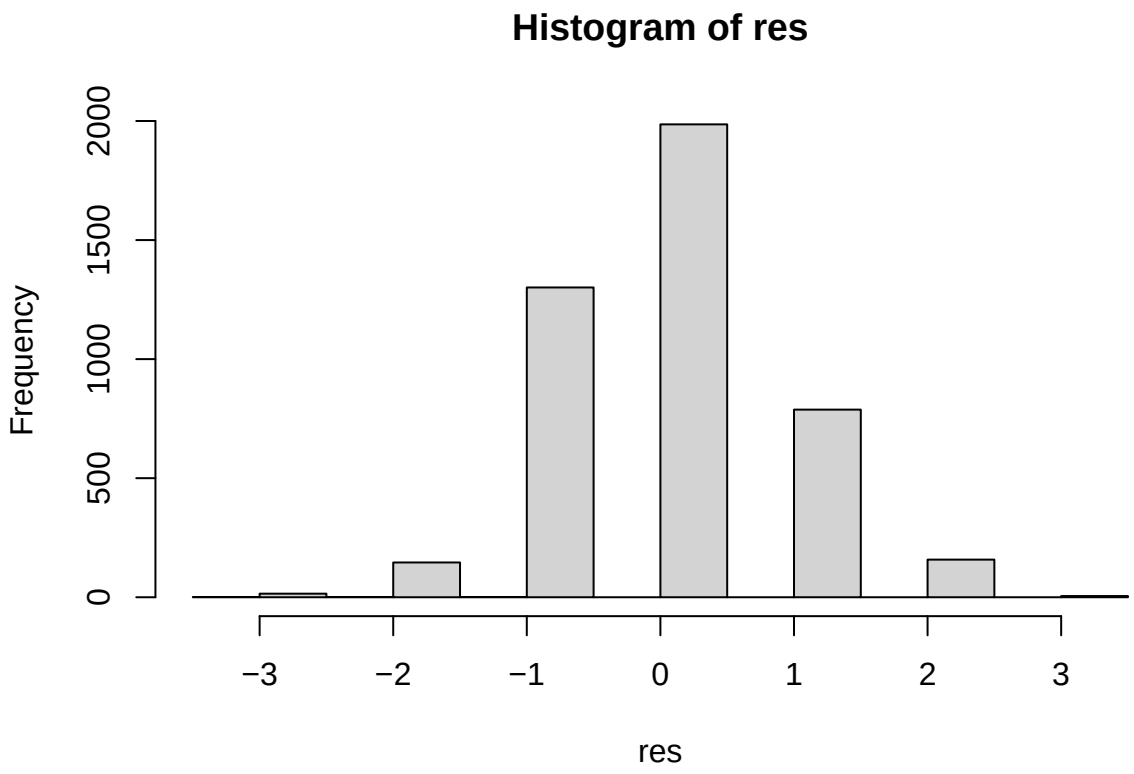


free.sulfur.dioxide before transformation

```
model <- lm(quality~free.sulfur.dioxide, data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

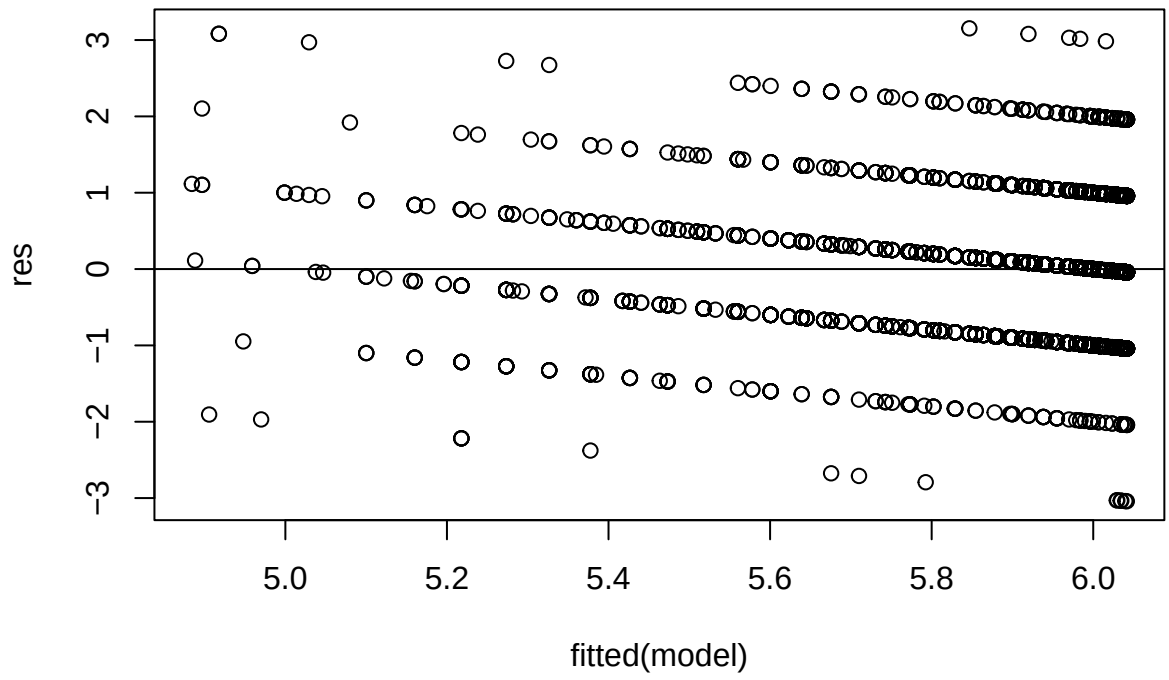



```
# produce histogram of residuals
hist(res)
```

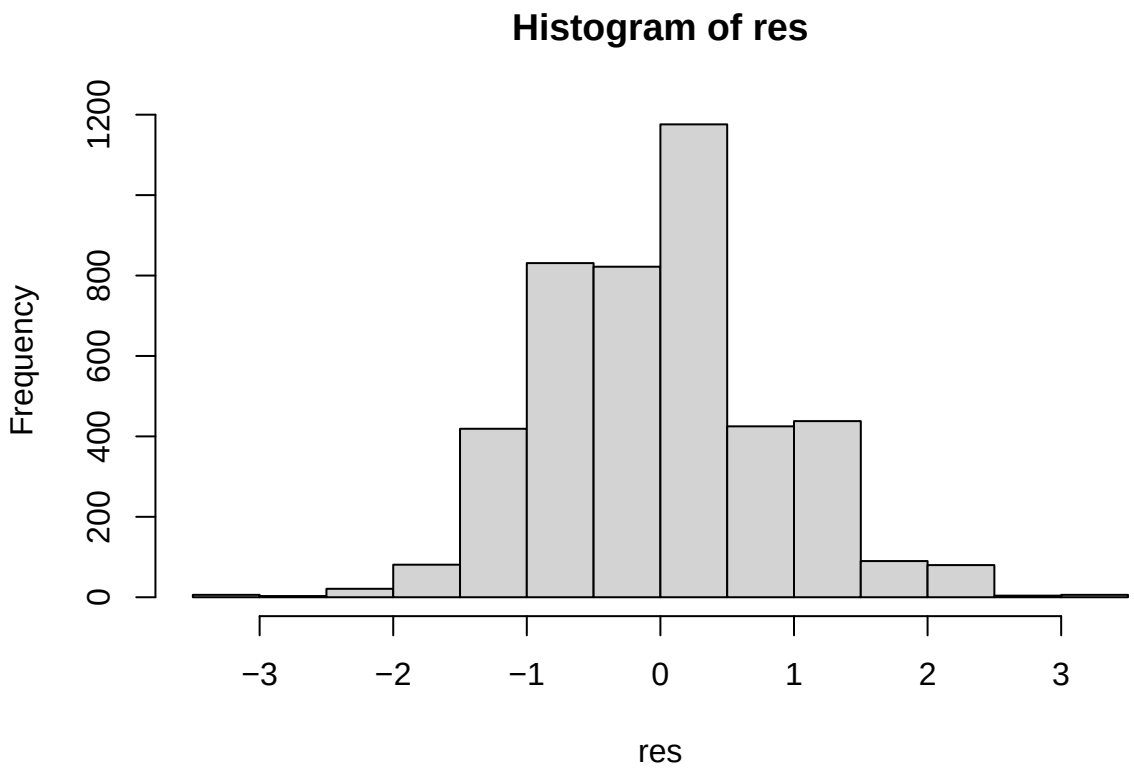


free.sulfur.dioxide after transformation

```
model <- lm(quality~poly(free.sulfur.dioxide, 3), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

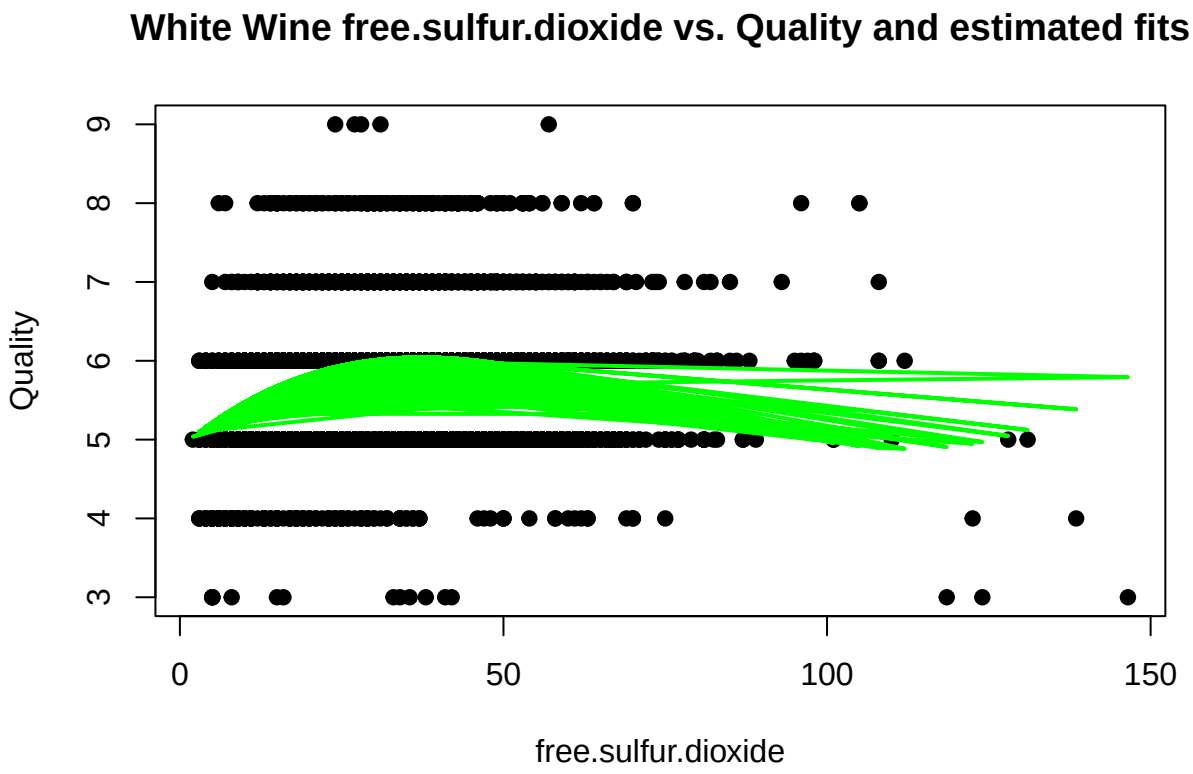



```
# produce histogram of residuals
hist(res)
```



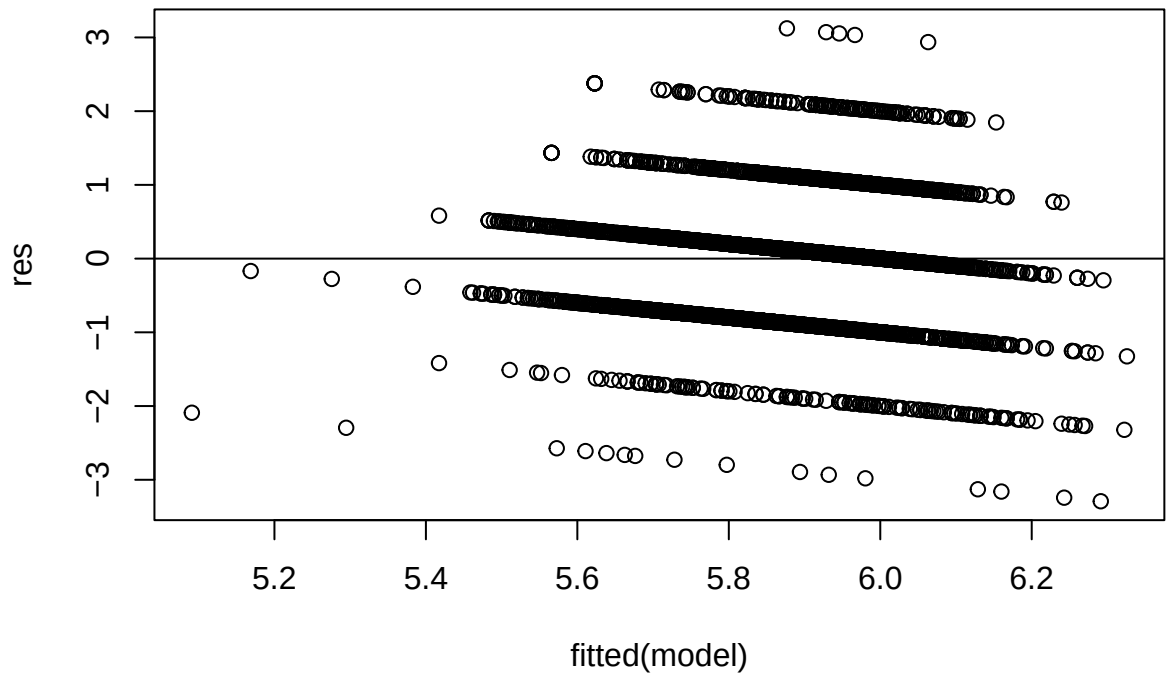
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$free.sulfur.dioxide,train.white$quality,pch=19,xlab='free.sulfur.dioxide',ylab='Quality
lines(train.white$free.sulfur.dioxide,model$fitted.values,col='green',lwd=2)
```

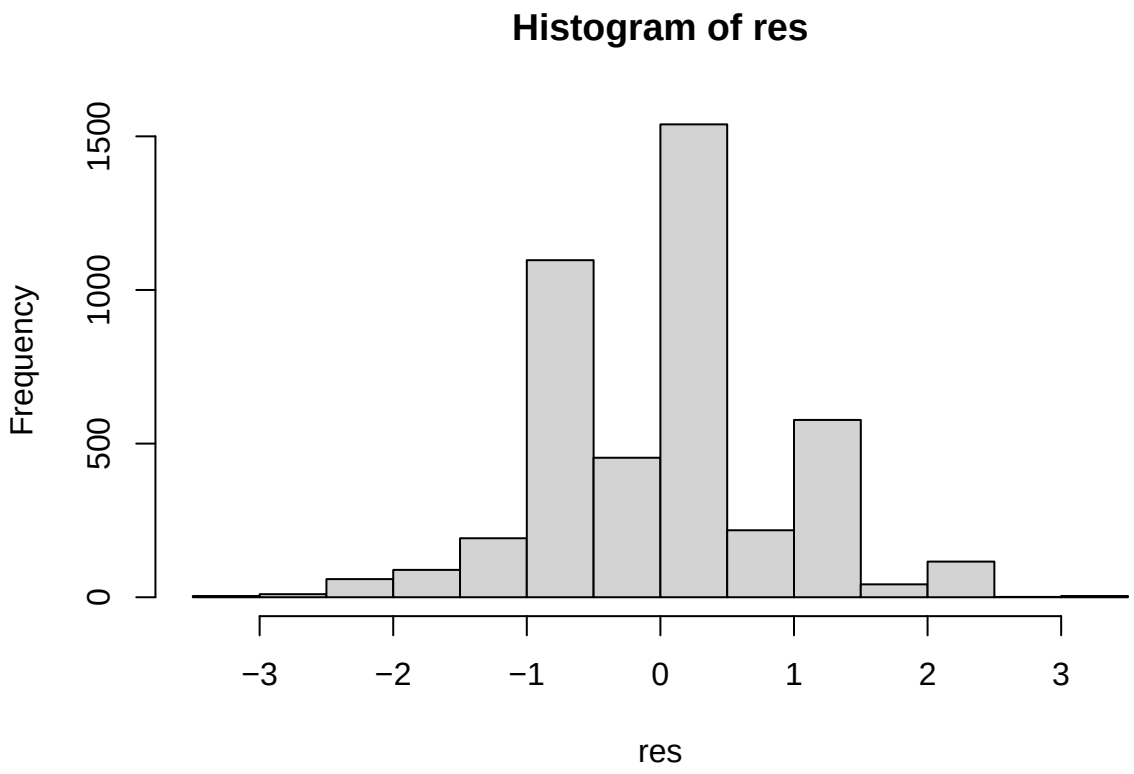


Total Sulfur Dioxide

```
model <- lm(quality~(total.sulfur.dioxide), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

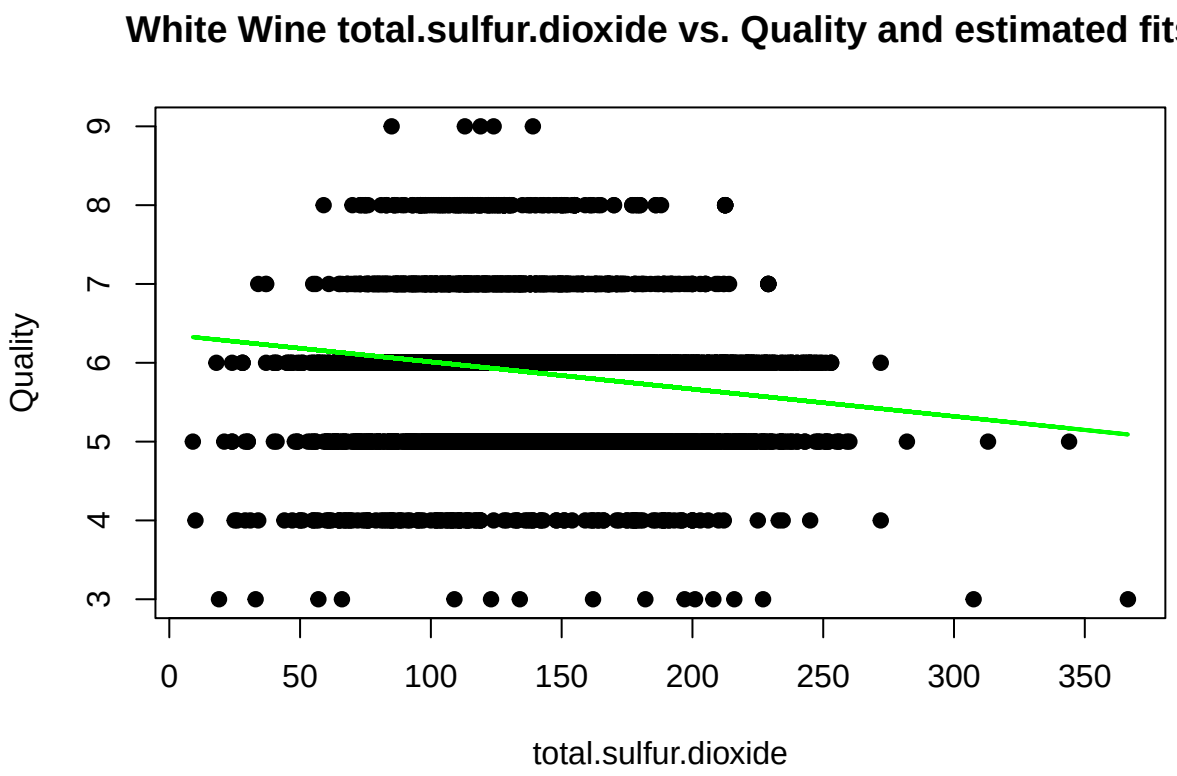



```
# produce histogram of residuals
hist(res)
```



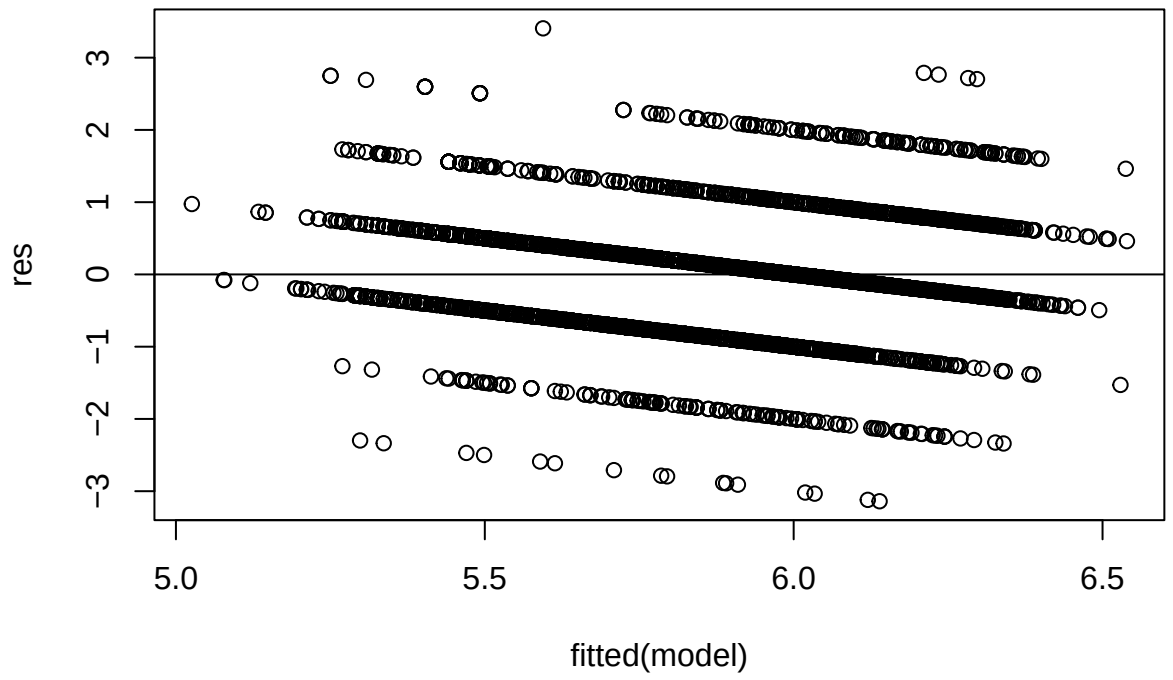
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$total.sulfur.dioxide,train.white$quality,pch=19,xlab='total.sulfur.dioxide',ylab='Quali
lines(train.white$total.sulfur.dioxide,model$fitted.values,col='green',lwd=2)
```

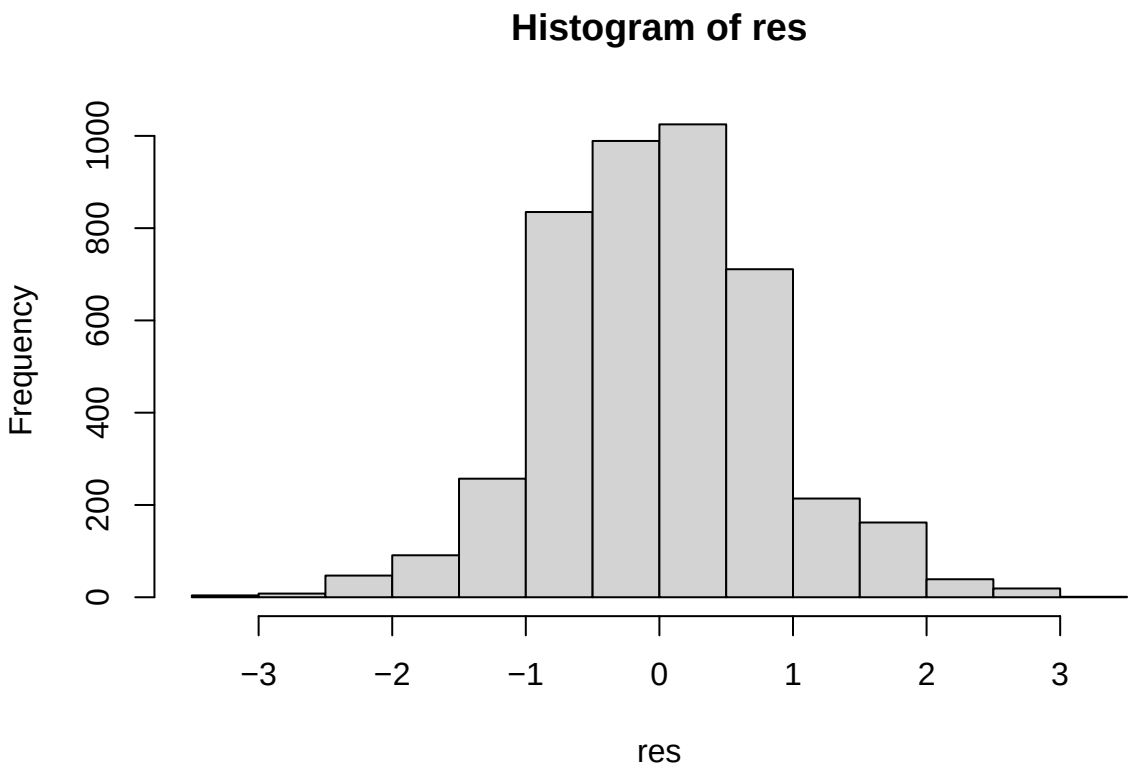


Density; displaying more pattern than I am comfortable with, but no transformation I tried would fix it. It gets dropped for most of the modelling anyways

```
model <- lm(quality~density, data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

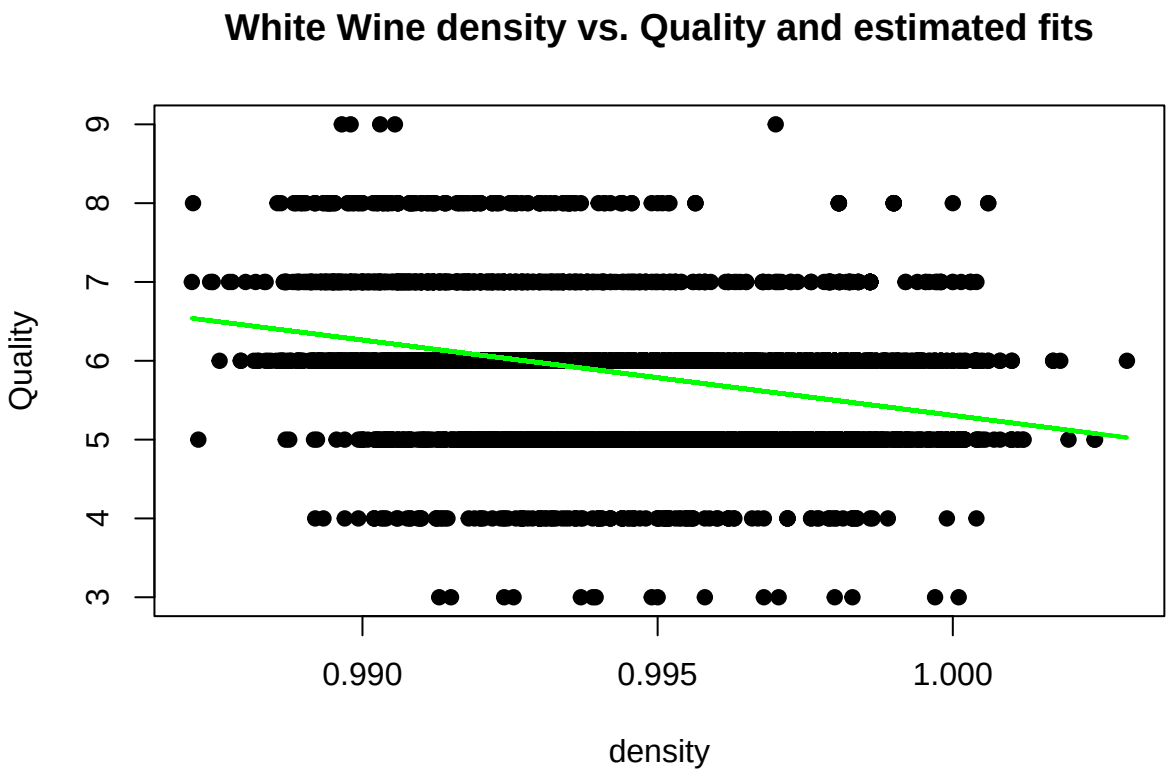



```
# produce histogram of residuals
hist(res)
```



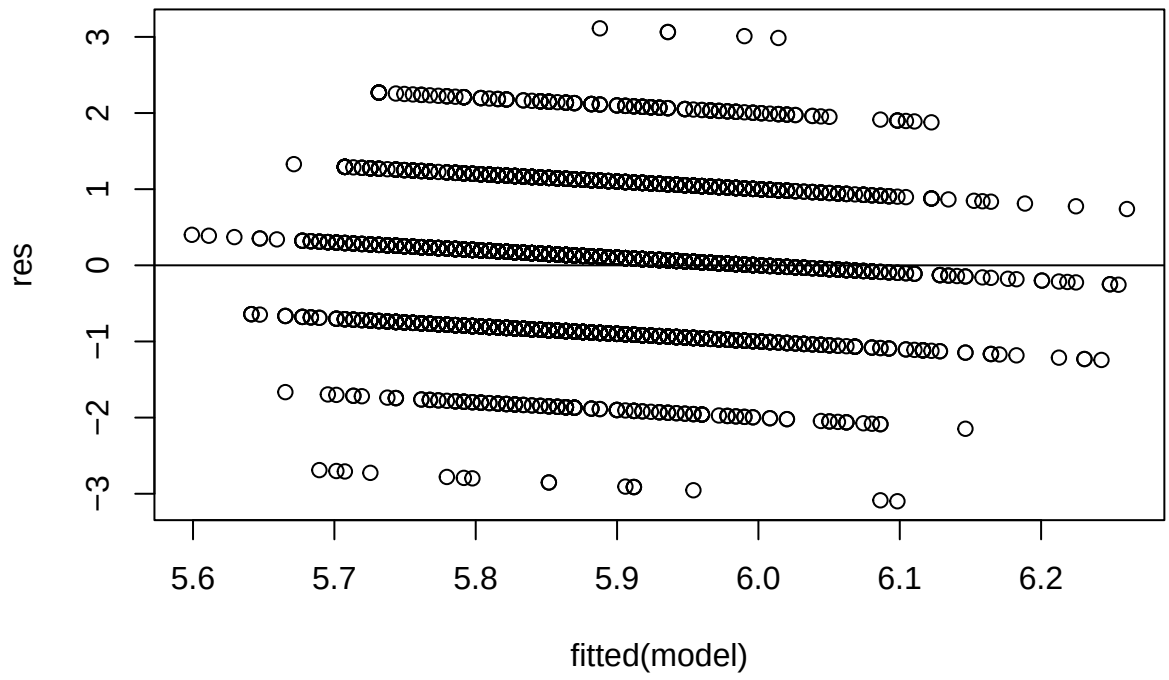
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$density,train.white$quality,pch=19,xlab='density',ylab='Quality',main='White Wine densi
lines(train.white$density,model$fitted.values,col='green',lwd=2)
```

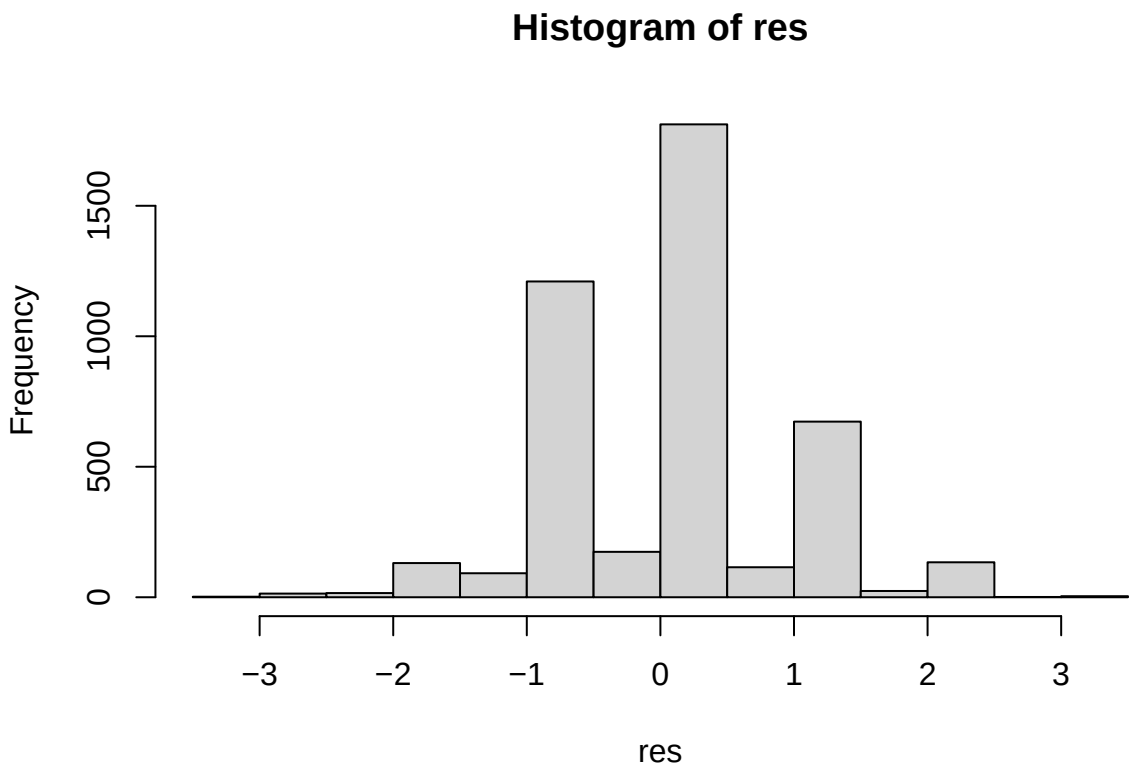


pH before transformation

```
model <- lm(quality~(pH), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

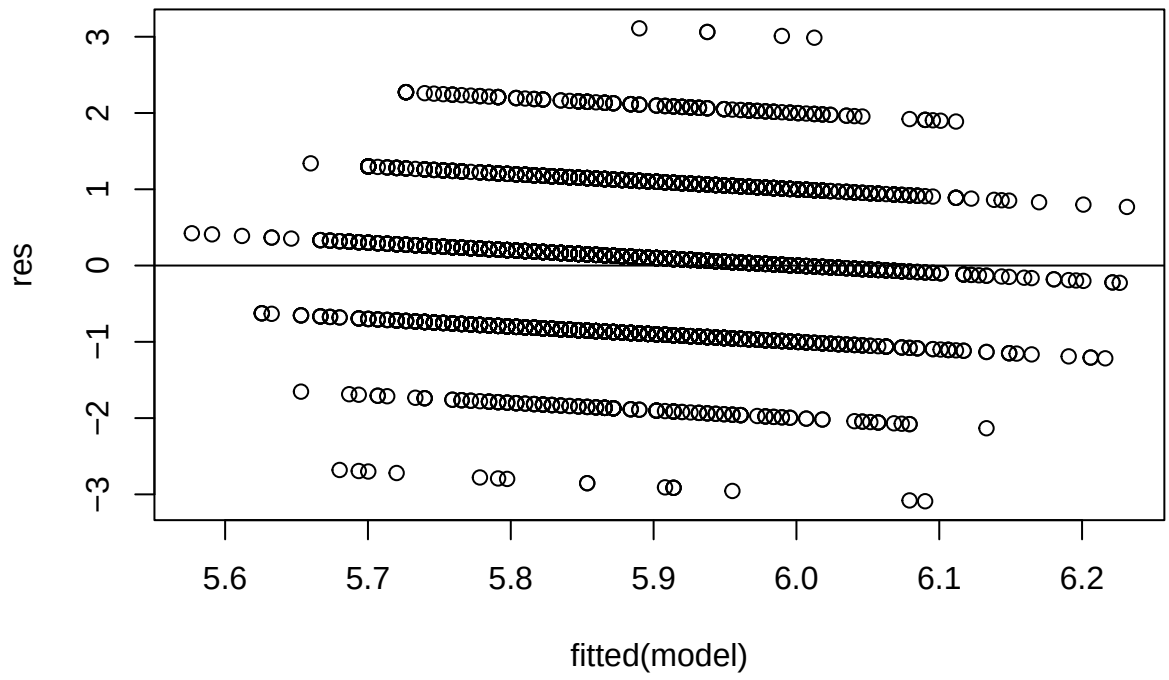



```
# produce histogram of residuals
hist(res)
```

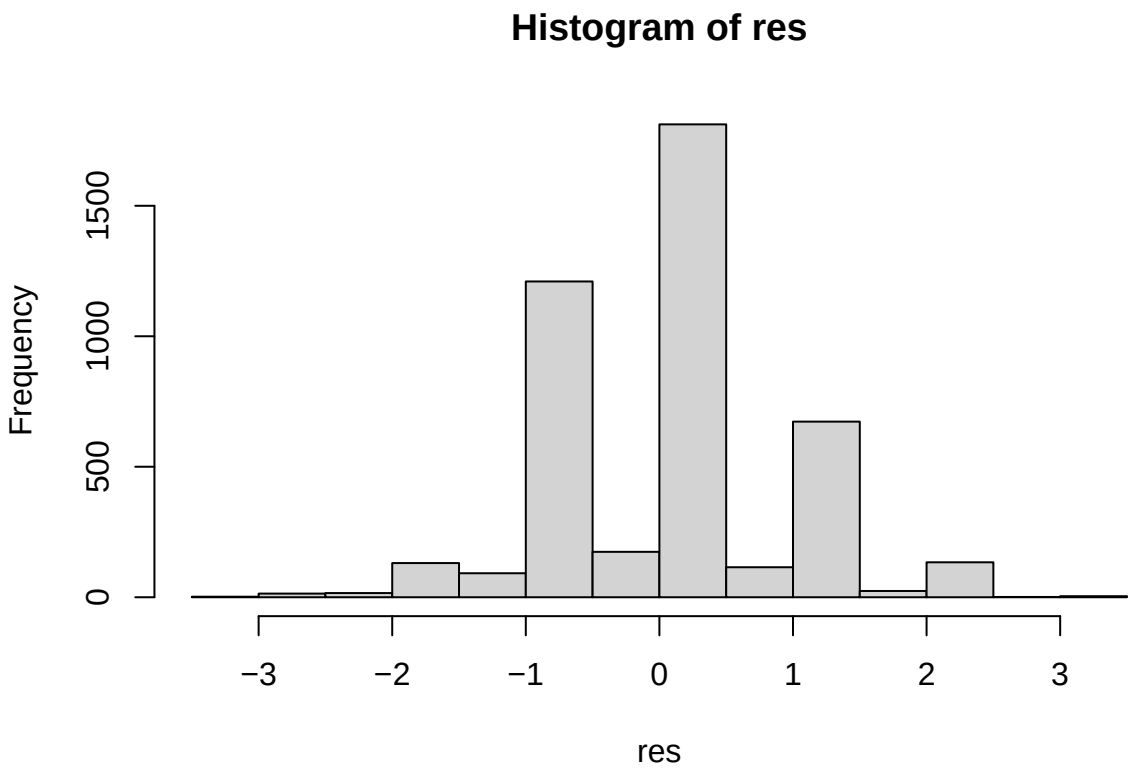


pH after transformation

```
model <- lm(quality~log(pH), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

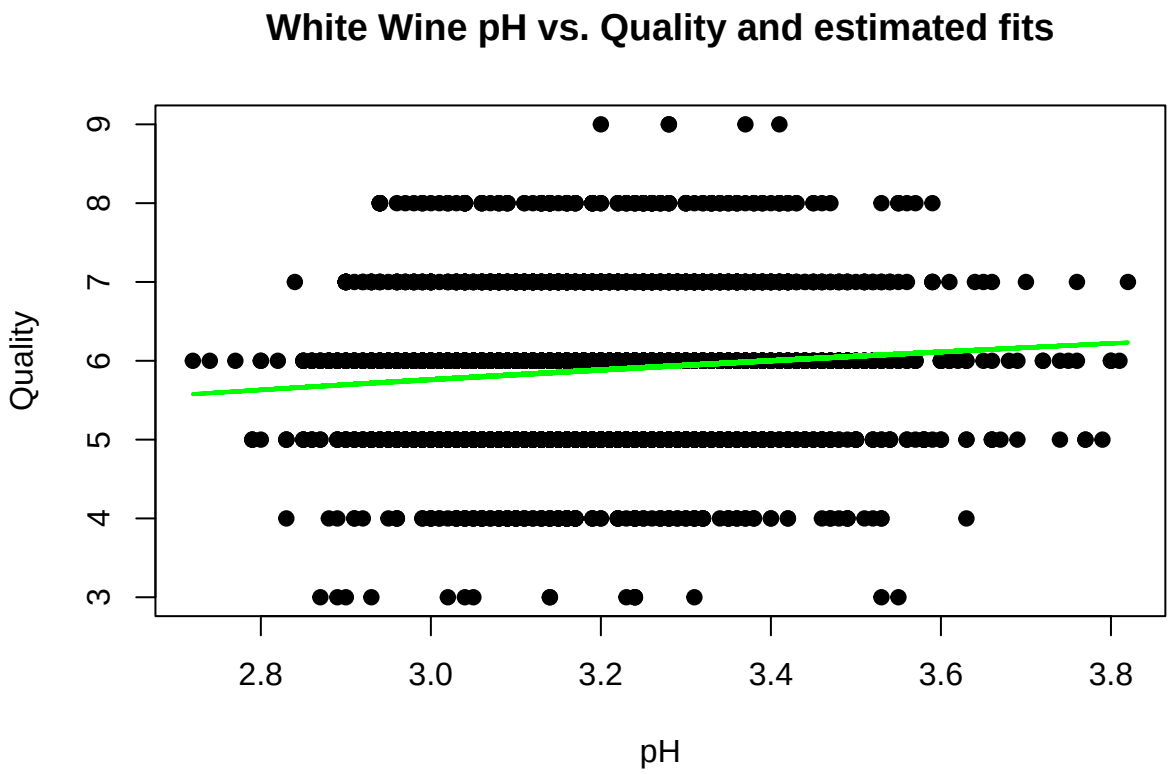



```
# produce histogram of residuals
hist(res)
```



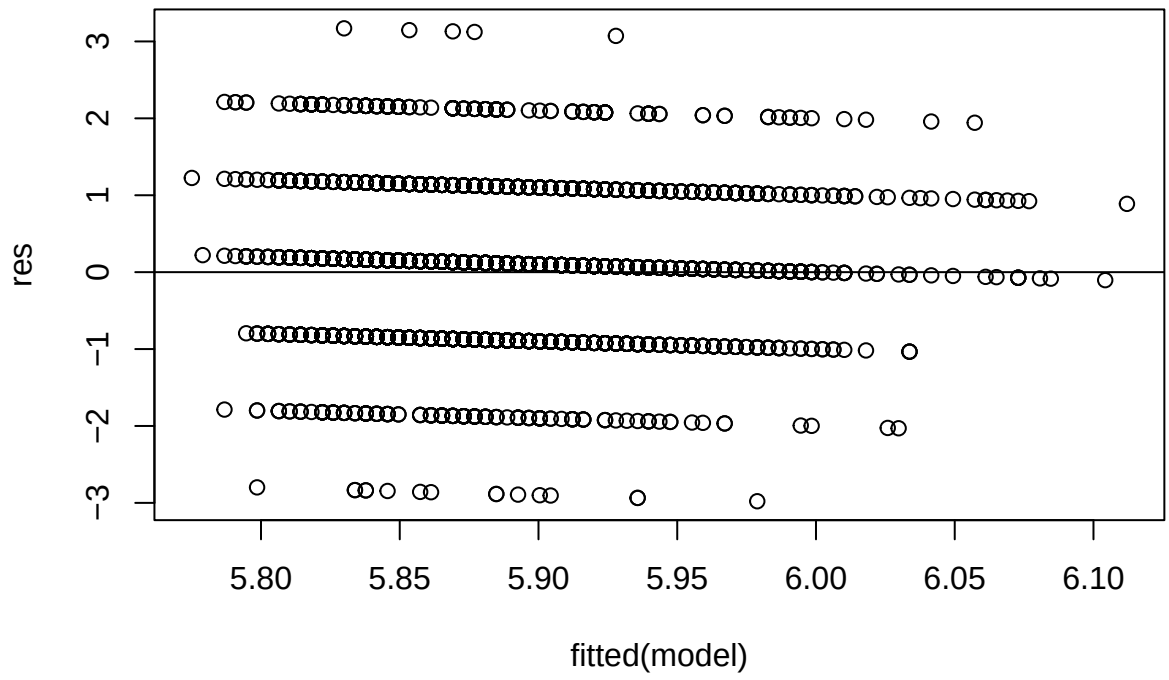
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$pH,train.white$quality,pch=19,xlab='pH',ylab='Quality',main='White Wine pH vs. Quality
lines(train.white$pH,model$fitted.values,col='green',lwd=2)
```

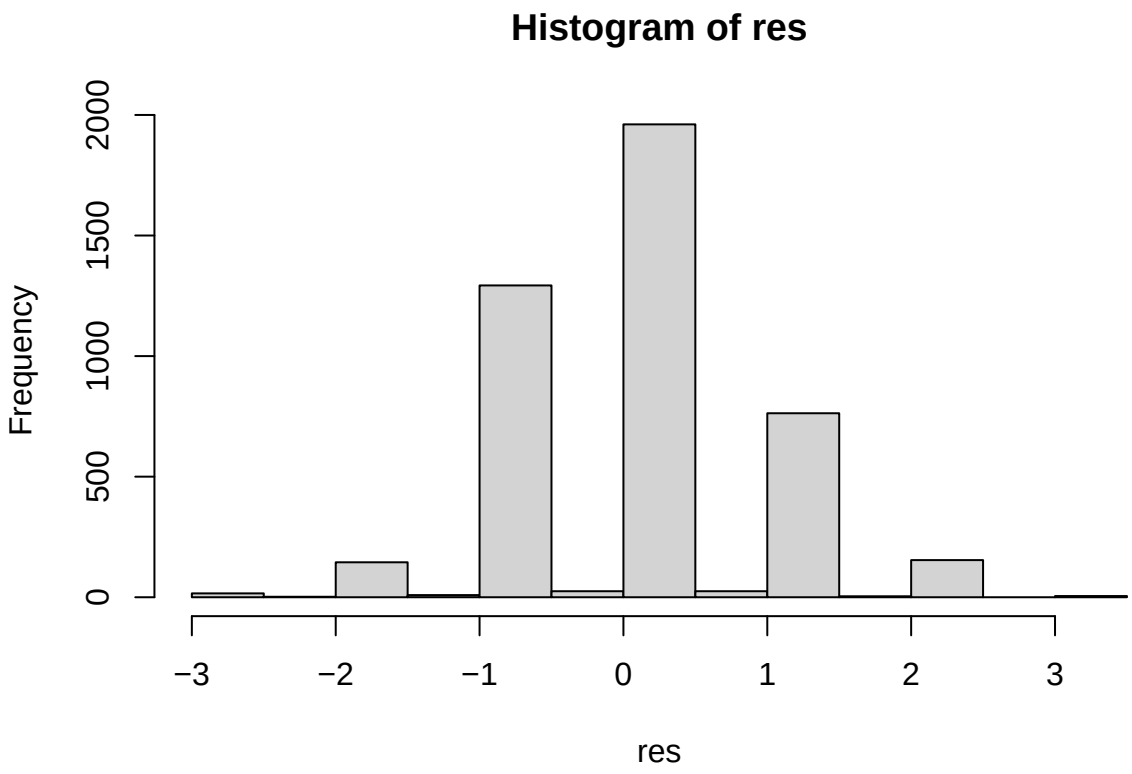


Sulphates before Transformation

```
model <- lm(quality~(sulphates), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

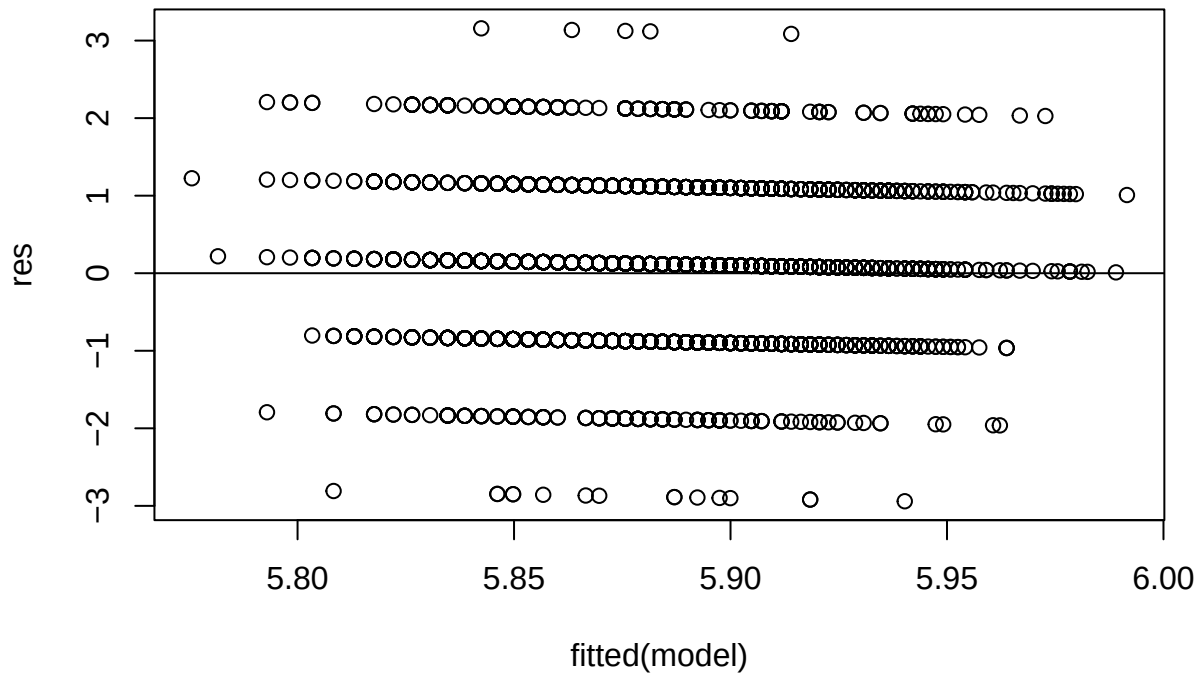



```
# produce histogram of residuals
hist(res)
```

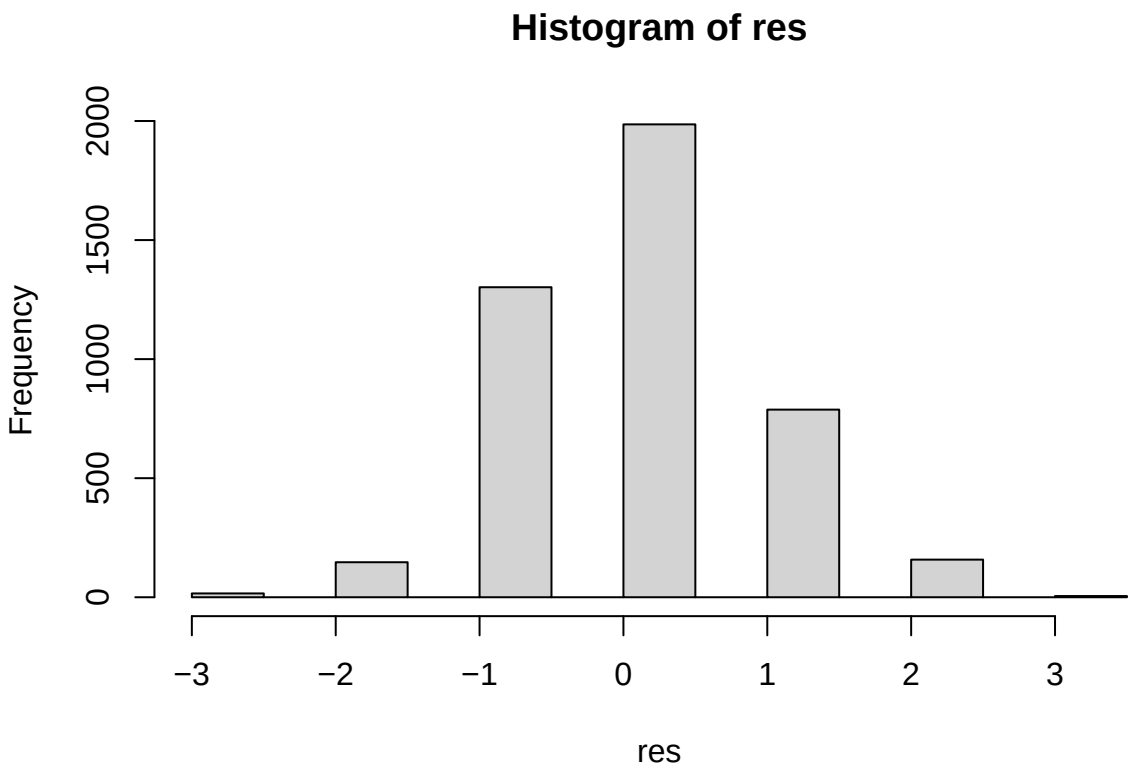


Sulphates after Transformation

```
model <- lm(quality~log(sulphates), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

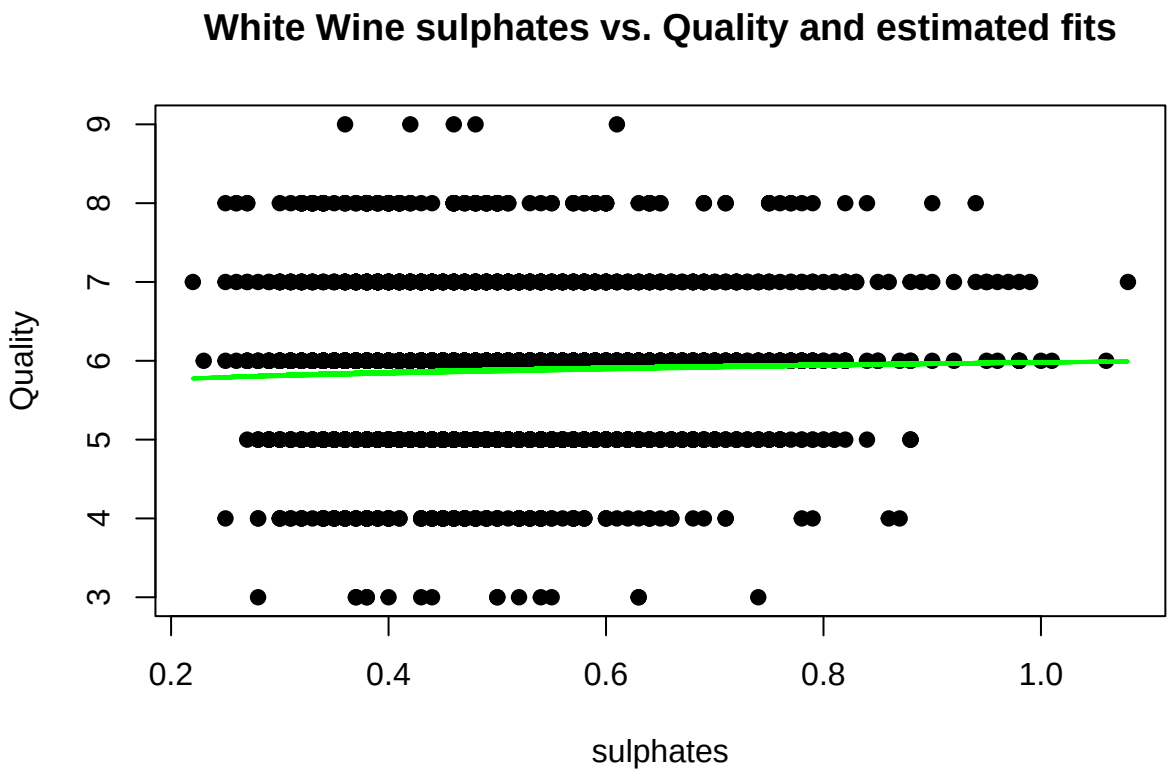



```
# produce histogram of residuals
hist(res)
```



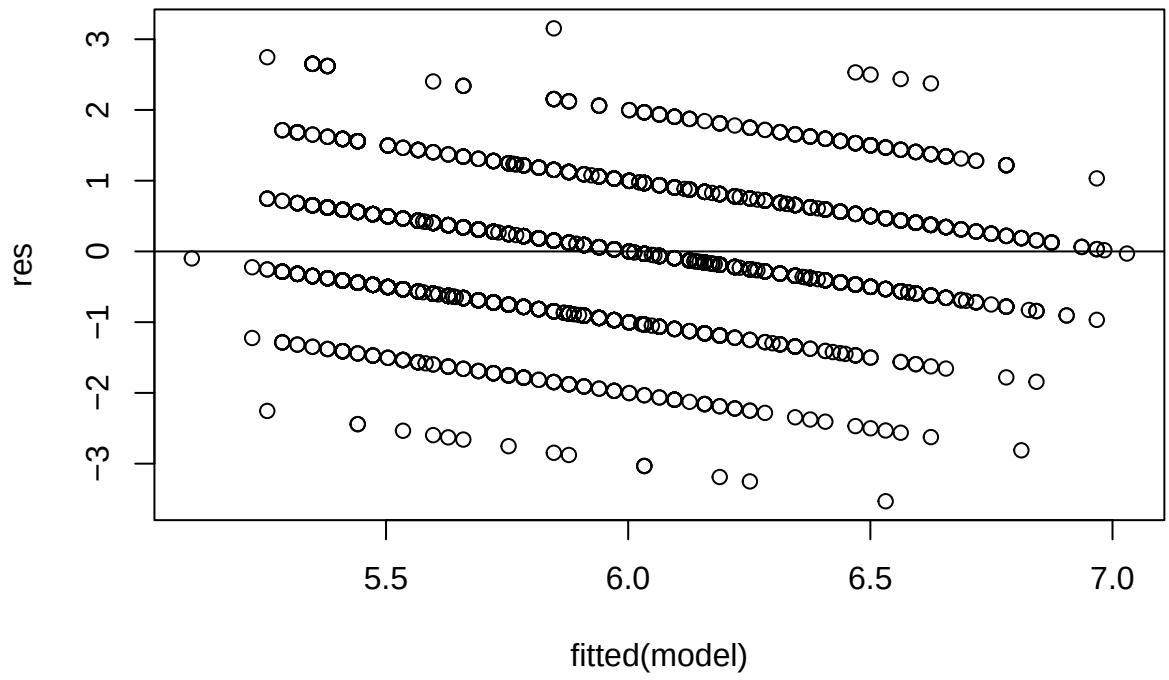
Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$sulphates,train.white$quality,pch=19,xlab='sulphates',ylab='Quality',main='White Wine s
lines(train.white$sulphates,model$fitted.values,col='green',lwd=2)
```

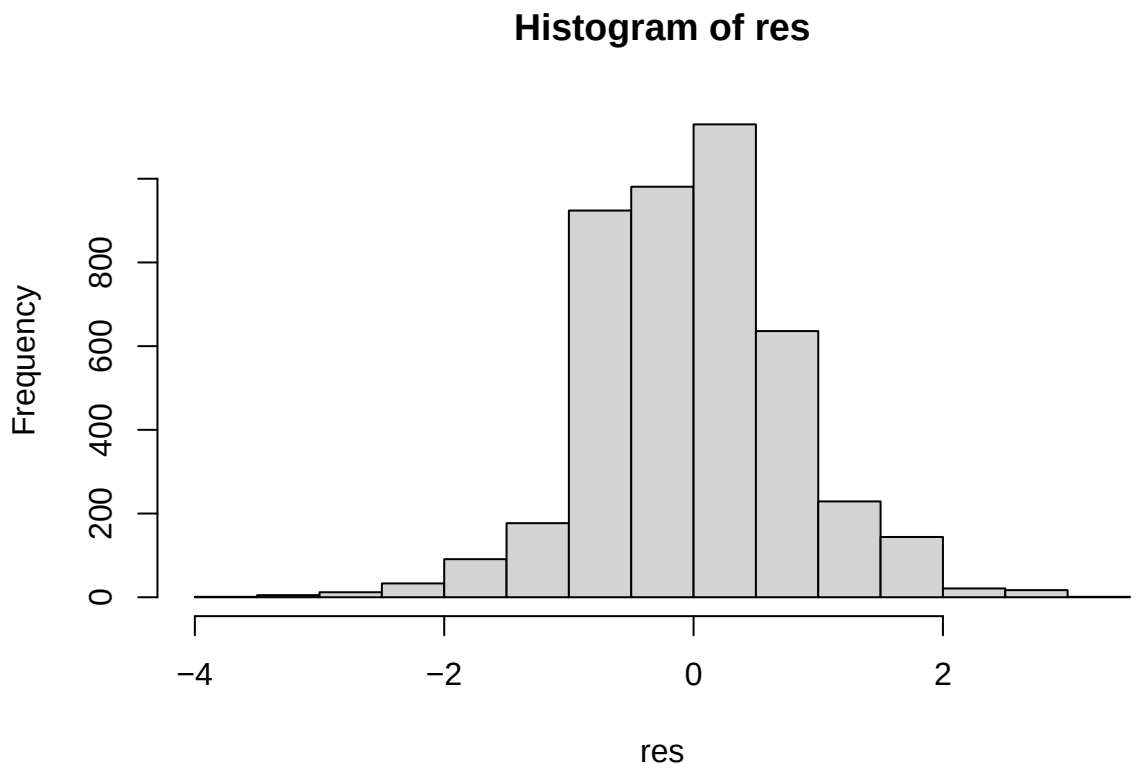


Alcohol before transformation

```
model <- lm(quality~alcohol, data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

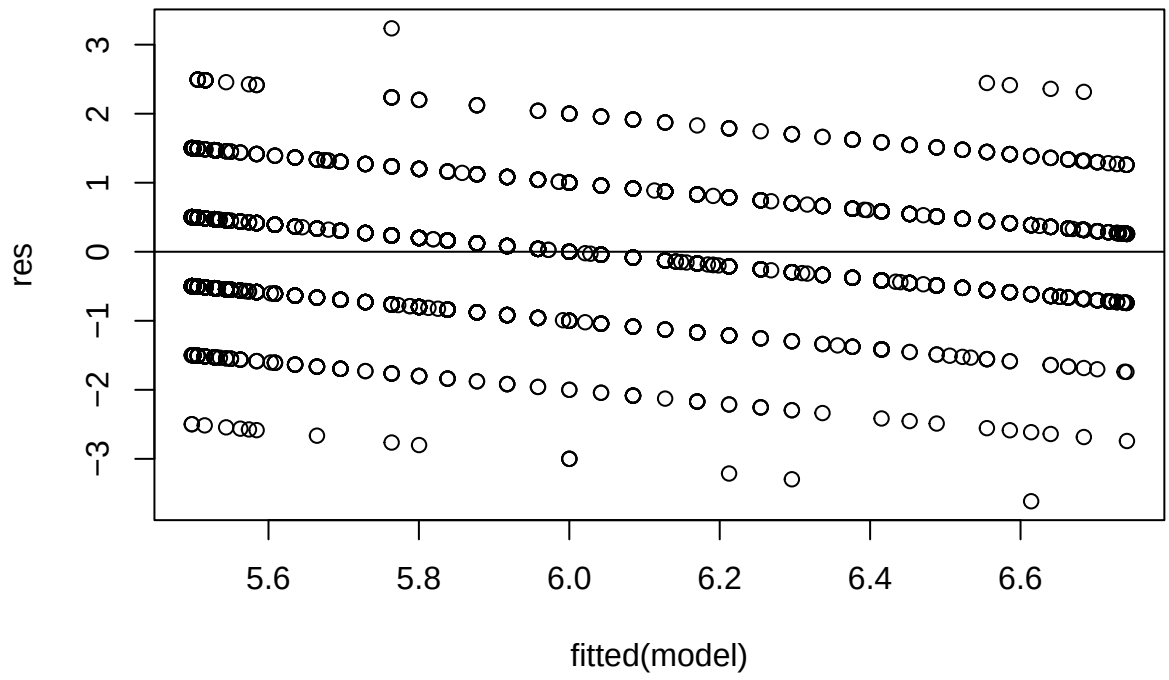



```
# produce histogram of residuals
hist(res)
```

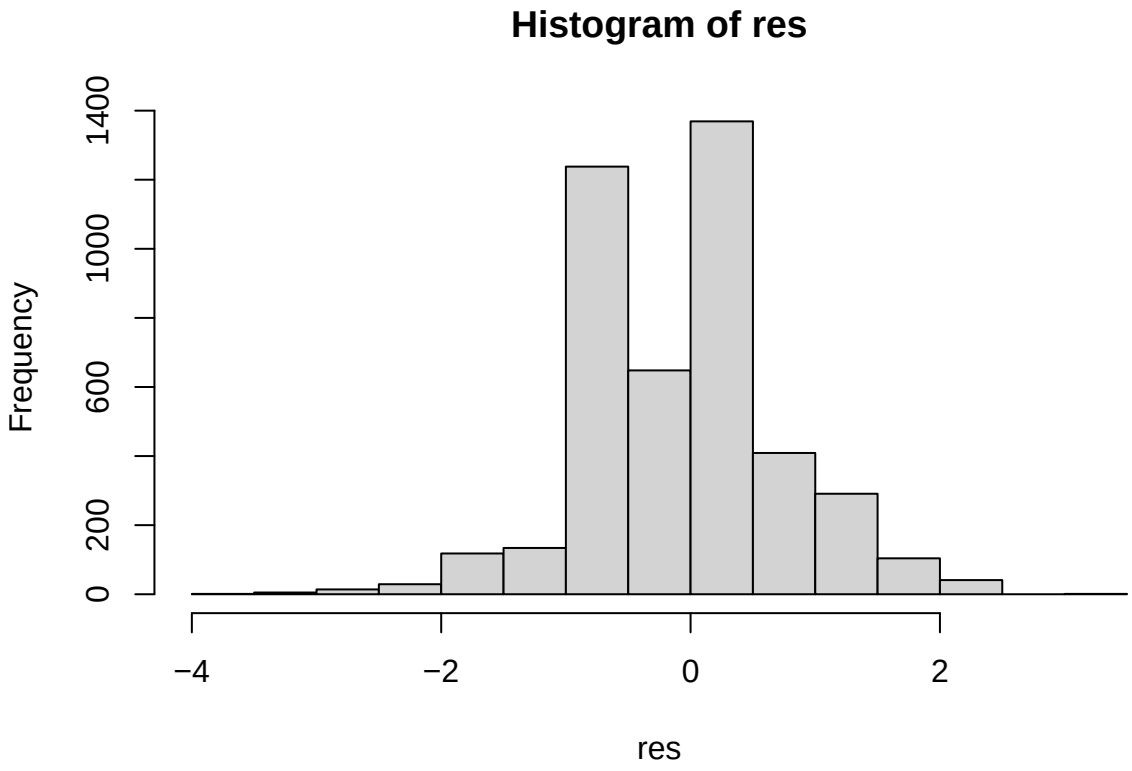


Alcohol After transformation

```
model <- lm(quality~poly(alcohol,3), data=train.white)

#get list of residuals
res <- resid(model)

#produce residual vs. fitted plot
plot(fitted(model), res)

#add a horizontal line at 0
abline(0,0)
```

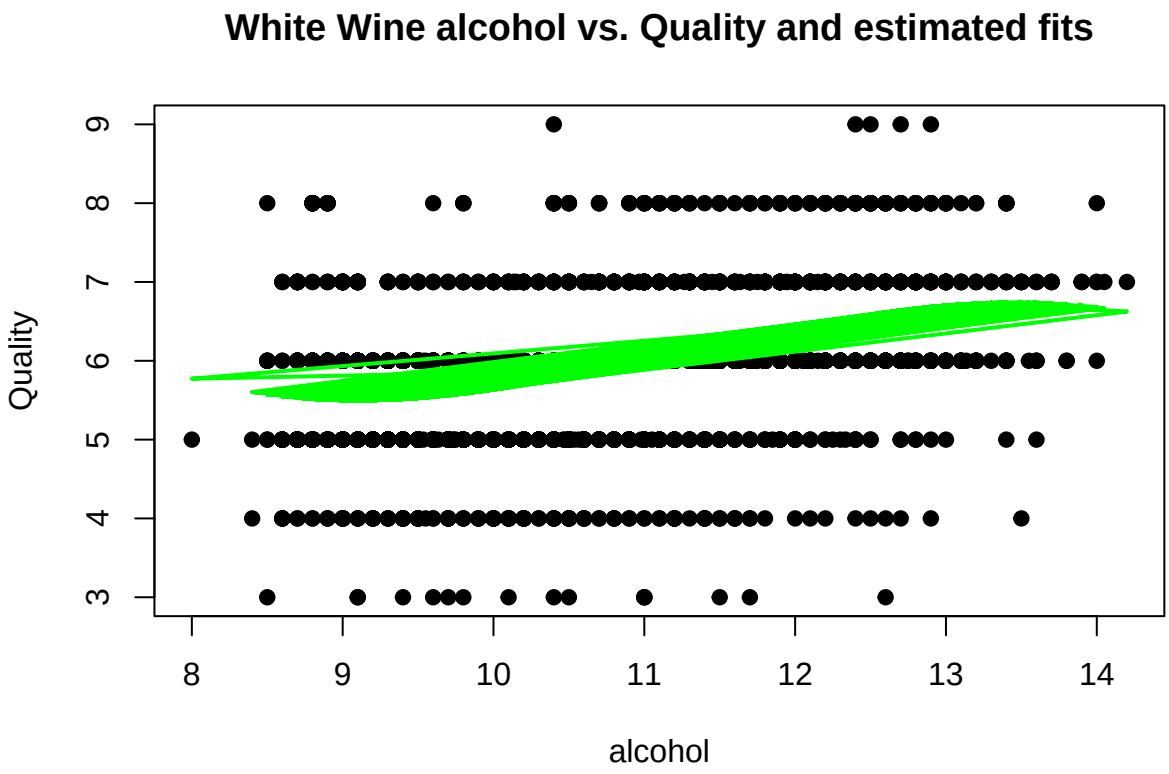



```
# produce histogram of residuals
hist(res)
```



Checking a graph of the model plotted against the scatterplot of the original data seems to confirm a pretty good fit.

```
plot(train.white$alcohol,train.white$quality,pch=19,xlab='alcohol',ylab='Quality',main='White Wine alcol
lines(train.white$alcohol,model$fitted.values,col='green',lwd=2)
```



Looking at the full model

```
mlr.full <- lm(quality~ . - quality.good - quality.factor, data=train.white)

sm.full <- summary(mlr.full)
sm.full
```

```
##
## Call:
## lm(formula = quality ~ . - quality.good - quality.factor, data = train.white)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5684 -0.5007 -0.0407  0.4558  3.1295
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.124e+02  2.493e+01   8.519  < 2e-16 ***
## fixed.acidity         1.054e-01  2.492e-02   4.229 2.40e-05 ***
## volatile.acidity     -1.875e+00  1.202e-01 -15.597  < 2e-16 ***
## citric.acid           9.152e-02  1.017e-01   0.900  0.36803
## residual.sugar        9.885e-02  9.326e-03  10.599  < 2e-16 ***
## chlorides            -3.544e-01  5.927e-01  -0.598  0.54991
## free.sulfur.dioxide   4.328e-03  9.035e-04   4.790 1.72e-06 ***
## total.sulfur.dioxide  1.513e-04  4.042e-04   0.374  0.70829
## density              -2.131e+02  2.527e+01  -8.434  < 2e-16 ***
```

```
## pH                    8.625e-01  1.202e-01   7.174 8.51e-13 ***
## sulphates             6.919e-01  1.074e-01   6.444 1.29e-10 ***
## alcohol               1.150e-01  3.145e-02   3.655  0.00026 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7483 on 4390 degrees of freedom
## Multiple R-squared:  0.2847, Adjusted R-squared:  0.2829
## F-statistic: 158.8 on 11 and 4390 DF,  p-value: < 2.2e-16
```

Looking at the full model minus density - the Adjusted R-squared score goes down from 0.2765 to 0.2646, but it doesn't matter because if the previous model violated an assumption it is null.

```
mlr.full <- lm(quality~ . - quality.good - quality.factor - density, data=train.white)

sm.full <- summary(mlr.full)
sm.full
```

```
##
## Call:
## lm(formula = quality ~ . - quality.good - quality.factor - density,
##     data = train.white)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3915 -0.5054 -0.0295  0.4595  3.1975
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.1277520  0.3706479   5.741 1.01e-08 ***
## fixed.acidity        -0.0580090  0.0158019  -3.671 0.000244 ***
## volatile.acidity     -1.9610776  0.1207334 -16.243  < 2e-16 ***
## citric.acid           0.0404357  0.1022838   0.395 0.692620
## residual.sugar        0.0236376  0.0027534   8.585  < 2e-16 ***
## chlorides            -1.2402908  0.5879095  -2.110 0.034944 *
## free.sulfur.dioxide   0.0055541  0.0008988   6.179 7.02e-10 ***
## total.sulfur.dioxide -0.0007557  0.0003928  -1.924 0.054432 .
## pH                    0.1585934  0.0872250   1.818 0.069100 .
## sulphates             0.4046256  0.1026327   3.942 8.19e-05 ***
## alcohol               0.3607022  0.0119493  30.186  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7543 on 4391 degrees of freedom
## Multiple R-squared:  0.2731, Adjusted R-squared:  0.2714
## F-statistic: 164.9 on 10 and 4391 DF,  p-value: < 2.2e-16
```

Looking at the model after dropping density and transforming variables, we see the R2 adju went up from 0.2646 to 0.3095

```
model.tranformed.full <- lm(quality~
              fixed.acidity+
              log(volatile.acidity)+
```

```
              (citric.acid)+
              poly(residual.sugar, 3)+
              poly(chlorides,6)+
              poly(free.sulfur.dioxide, 3)+
              (total.sulfur.dioxide)+
              log(pH)+
              log(sulphates)+
              poly(alcohol,3),
            data=train.white)


sm.trans.full <- summary(model.tranformed.full)
sm.trans.full
```

```
##
## Call:
## lm(formula = quality ~ fixed.acidity + log(volatile.acidity) +
##     (citric.acid) + poly(residual.sugar, 3) + poly(chlorides,
##     6) + poly(free.sulfur.dioxide, 3) + (total.sulfur.dioxide) +
##     log(pH) + log(sulphates) + poly(alcohol, 3), data = train.white)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.04408 -0.52210  0.00247  0.44495  3.14102
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   5.0916368  0.3848190  13.231  < 2e-16 ***
## fixed.acidity                -0.0411725  0.0156106  -2.637  0.00838 **
## log(volatile.acidity)        -0.5974268  0.0359205 -16.632  < 2e-16 ***
## citric.acid                  -0.0474506  0.1011436  -0.469  0.63899
## poly(residual.sugar, 3)1      5.8315588  0.9263285   6.295 3.37e-10 ***
## poly(residual.sugar, 3)2     -3.2786743  0.7581834  -4.324 1.56e-05 ***
## poly(residual.sugar, 3)3     -0.7134502  0.7496360  -0.952  0.34129
## poly(chlorides, 6)1          -2.2150795  0.8208878  -2.698  0.00699 **
## poly(chlorides, 6)2           2.3679767  0.8053721   2.940  0.00330 **
## poly(chlorides, 6)3          -2.1674354  0.7776513  -2.787  0.00534 **
## poly(chlorides, 6)4           0.5677864  0.7473835   0.760  0.44748
## poly(chlorides, 6)5           1.6948297  0.7413922   2.286  0.02230 *
## poly(chlorides, 6)6          -0.3549163  0.7372759  -0.481  0.63026
## poly(free.sulfur.dioxide, 3)1  6.2677692  0.9810873   6.389 1.85e-10 ***
## poly(free.sulfur.dioxide, 3)2 -9.4301951  0.7480408 -12.607  < 2e-16 ***
## poly(free.sulfur.dioxide, 3)3  4.4131172  0.7441755   5.930 3.26e-09 ***
## total.sulfur.dioxide         -0.0010598  0.0003964  -2.674  0.00753 **
## log(pH)                       0.5324168  0.2775738   1.918  0.05516 .
## log(sulphates)                0.2509981  0.0513822   4.885 1.07e-06 ***
## poly(alcohol, 3)1            26.4104963  1.0342546  25.536  < 2e-16 ***
## poly(alcohol, 3)2             4.4172298  0.8051376   5.486 4.34e-08 ***
## poly(alcohol, 3)3            -3.2525357  0.7582105  -4.290 1.83e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7306 on 4380 degrees of freedom
```

```
## Multiple R-squared:  0.3196, Adjusted R-squared:  0.3163
## F-statistic: 97.97 on 21 and 4380 DF,  p-value: < 2.2e-16
```

(b) Identify the statistical learning methods you will use to address the overall goal of the analysis. At least one method for each data set analysis must be selected from those we covered in Chapters 6 through 12. Provide formal representation of the methods (such as a mathematical expression for a model or a description of how the method works/is fit) and identify any important components in your representation. Your methods must depend on some type of tuning parameter. Identify the tuning parameter for your methods.

Regression Models

Multiple Linear Regression Model with Best Subsets

```
regfit.full <- regsubsets(quality ~
                            fixed.acidity+
              log(volatile.acidity)+
              (citric.acid)+
              poly(residual.sugar, 3)+
              poly(chlorides,6)+
              poly(free.sulfur.dioxide, 3)+
              (total.sulfur.dioxide)+
              log(pH)+
              log(sulphates)+
              poly(alcohol,3),
              data = train.white, nvmax = 21)
reg.summary <- summary(regfit.full)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(quality ~ fixed.acidity + log(volatile.acidity) +
##     (citric.acid) + poly(residual.sugar, 3) + poly(chlorides,
##     6) + poly(free.sulfur.dioxide, 3) + (total.sulfur.dioxide) +
##     log(pH) + log(sulphates) + poly(alcohol, 3), data = train.white,
##     nvmax = 21)
## 21 Variables  (and intercept)
##                               Forced in Forced out
## fixed.acidity                     FALSE      FALSE
## log(volatile.acidity)             FALSE      FALSE
## citric.acid                       FALSE      FALSE
## poly(residual.sugar, 3)1          FALSE      FALSE
## poly(residual.sugar, 3)2          FALSE      FALSE
## poly(residual.sugar, 3)3          FALSE      FALSE
## poly(chlorides, 6)1               FALSE      FALSE
## poly(chlorides, 6)2               FALSE      FALSE
## poly(chlorides, 6)3               FALSE      FALSE
## poly(chlorides, 6)4               FALSE      FALSE
## poly(chlorides, 6)5               FALSE      FALSE
## poly(chlorides, 6)6               FALSE      FALSE
## poly(free.sulfur.dioxide, 3)1     FALSE      FALSE
```

```
## poly(free.sulfur.dioxide, 3)2     FALSE      FALSE
## poly(free.sulfur.dioxide, 3)3     FALSE      FALSE
## total.sulfur.dioxide              FALSE      FALSE
## log(pH)                           FALSE      FALSE
## log(sulphates)                    FALSE      FALSE
## poly(alcohol, 3)1                 FALSE      FALSE
## poly(alcohol, 3)2                 FALSE      FALSE
## poly(alcohol, 3)3                 FALSE      FALSE
## 1 subsets of each size up to 21
## Selection Algorithm: exhaustive
##           fixed.acidity log(volatile.acidity) citric.acid
## 1  ( 1 )  " "           " "                   " "
## 2  ( 1 )  " "           "*"                   " "
## 3  ( 1 )  " "           "*"                   " "
## 4  ( 1 )  " "           "*"                   " "
## 5  ( 1 )  " "           "*"                   " "
## 6  ( 1 )  " "           "*"                   " "
## 7  ( 1 )  "*"           "*"                   " "
## 8  ( 1 )  "*"           "*"                   " "
## 9  ( 1 )  "*"           "*"                   " "
## 10  ( 1 ) "*"           "*"                   " "
## 11  ( 1 ) "*"           "*"                   " "
## 12  ( 1 ) "*"           "*"                   " "
## 13  ( 1 ) "*"           "*"                   " "
## 14  ( 1 ) "*"           "*"                   " "
## 15  ( 1 ) "*"           "*"                   " "
## 16  ( 1 ) "*"           "*"                   " "
## 17  ( 1 ) "*"           "*"                   " "
## 18  ( 1 ) "*"           "*"                   " "
## 19  ( 1 ) "*"           "*"                   " "
## 20  ( 1 ) "*"           "*"                   " "
## 21  ( 1 ) "*"           "*"                   "*"
##           poly(residual.sugar, 3)1 poly(residual.sugar, 3)2
## 1  ( 1 )  " "                      " "
## 2  ( 1 )  " "                      " "
## 3  ( 1 )  " "                      " "
## 4  ( 1 )  " "                      " "
## 5  ( 1 )  "*"                      " "
## 6  ( 1 )  "*"                      " "
## 7  ( 1 )  "*"                      " "
## 8  ( 1 )  "*"                      " "
## 9  ( 1 )  "*"                      " "
## 10  ( 1 ) "*"                      " "
## 11  ( 1 ) "*"                      "*"
## 12  ( 1 ) "*"                      "*"
## 13  ( 1 ) "*"                      "*"
## 14  ( 1 ) "*"                      "*"
## 15  ( 1 ) "*"                      "*"
## 16  ( 1 ) "*"                      "*"
## 17  ( 1 ) "*"                      "*"
## 18  ( 1 ) "*"                      "*"
## 19  ( 1 ) "*"                      "*"
## 20  ( 1 ) "*"                      "*"
## 21  ( 1 ) "*"                      "*"
```

```
##           poly(residual.sugar, 3)3 poly(chlorides, 6)1 poly(chlorides, 6)2
## 1  ( 1 )  " "                      " "                 " "
## 2  ( 1 )  " "                      " "                 " "
## 3  ( 1 )  " "                      " "                 " "
## 4  ( 1 )  " "                      " "                 " "
## 5  ( 1 )  " "                      " "                 " "
## 6  ( 1 )  " "                      " "                 " "
## 7  ( 1 )  " "                      " "                 " "
## 8  ( 1 )  " "                      " "                 " "
## 9  ( 1 )  " "                      " "                 " "
## 10  ( 1 ) " "                      " "                 " "
## 11  ( 1 ) " "                      " "                 " "
## 12  ( 1 ) " "                      " "                 " "
## 13  ( 1 ) " "                      " "                 " "
## 14  ( 1 ) " "                      "*"                 "*"
## 15  ( 1 ) " "                      "*"                 "*"
## 16  ( 1 ) " "                      "*"                 "*"
## 17  ( 1 ) " "                      "*"                 "*"
## 18  ( 1 ) "*"                      "*"                 "*"
## 19  ( 1 ) "*"                      "*"                 "*"
## 20  ( 1 ) "*"                      "*"                 "*"
## 21  ( 1 ) "*"                      "*"                 "*"
##           poly(chlorides, 6)3 poly(chlorides, 6)4 poly(chlorides, 6)5
## 1  ( 1 )  " "                 " "                 " "
## 2  ( 1 )  " "                 " "                 " "
## 3  ( 1 )  " "                 " "                 " "
## 4  ( 1 )  " "                 " "                 " "
## 5  ( 1 )  " "                 " "                 " "
## 6  ( 1 )  " "                 " "                 " "
## 7  ( 1 )  " "                 " "                 " "
## 8  ( 1 )  " "                 " "                 " "
## 9  ( 1 )  " "                 " "                 " "
## 10  ( 1 ) " "                 " "                 " "
## 11  ( 1 ) " "                 " "                 " "
## 12  ( 1 ) " "                 " "                 " "
## 13  ( 1 ) " "                 " "                 "*"
## 14  ( 1 ) " "                 " "                 " "
## 15  ( 1 ) "*"                 " "                 " "
## 16  ( 1 ) "*"                 " "                 "*"
## 17  ( 1 ) "*"                 " "                 "*"
## 18  ( 1 ) "*"                 " "                 "*"
## 19  ( 1 ) "*"                 "*"                 "*"
## 20  ( 1 ) "*"                 "*"                 "*"
## 21  ( 1 ) "*"                 "*"                 "*"
##           poly(chlorides, 6)6 poly(free.sulfur.dioxide, 3)1
## 1  ( 1 )  " "                 " "
## 2  ( 1 )  " "                 " "
## 3  ( 1 )  " "                 " "
## 4  ( 1 )  " "                 "*"
## 5  ( 1 )  " "                 "*"
## 6  ( 1 )  " "                 "*"
## 7  ( 1 )  " "                 "*"
## 8  ( 1 )  " "                 "*"
## 9  ( 1 )  " "                 "*"
```

```
## 10  ( 1 ) " "                 "*"
## 11  ( 1 ) " "                 "*"
## 12  ( 1 ) " "                 "*"
## 13  ( 1 ) " "                 "*"
## 14  ( 1 ) " "                 "*"
## 15  ( 1 ) " "                 "*"
## 16  ( 1 ) " "                 "*"
## 17  ( 1 ) " "                 "*"
## 18  ( 1 ) " "                 "*"
## 19  ( 1 ) " "                 "*"
## 20  ( 1 ) "*"                 "*"
## 21  ( 1 ) "*"                 "*"
##           poly(free.sulfur.dioxide, 3)2 poly(free.sulfur.dioxide, 3)3
## 1  ( 1 )  " "                           " "
## 2  ( 1 )  " "                           " "
## 3  ( 1 )  "*"                           " "
## 4  ( 1 )  "*"                           " "
## 5  ( 1 )  "*"                           " "
## 6  ( 1 )  "*"                           "*"
## 7  ( 1 )  "*"                           "*"
## 8  ( 1 )  "*"                           "*"
## 9  ( 1 )  "*"                           "*"
## 10  ( 1 ) "*"                           "*"
## 11  ( 1 ) "*"                           "*"
## 12  ( 1 ) "*"                           "*"
## 13  ( 1 ) "*"                           "*"
## 14  ( 1 ) "*"                           "*"
## 15  ( 1 ) "*"                           "*"
## 16  ( 1 ) "*"                           "*"
## 17  ( 1 ) "*"                           "*"
## 18  ( 1 ) "*"                           "*"
## 19  ( 1 ) "*"                           "*"
## 20  ( 1 ) "*"                           "*"
## 21  ( 1 ) "*"                           "*"
##           total.sulfur.dioxide log(pH) log(sulphates) poly(alcohol, 3)1
## 1  ( 1 )  " "                  " "     " "             "*"
## 2  ( 1 )  " "                  " "     " "             "*"
## 3  ( 1 )  " "                  " "     " "             "*"
## 4  ( 1 )  " "                  " "     " "             "*"
## 5  ( 1 )  " "                  " "     " "             "*"
## 6  ( 1 )  " "                  " "     " "             "*"
## 7  ( 1 )  " "                  " "     " "             "*"
## 8  ( 1 )  " "                  " "     " "             "*"
## 9  ( 1 )  " "                  " "     "*"             "*"
## 10  ( 1 ) " "                  " "     "*"             "*"
## 11  ( 1 ) " "                  " "     "*"             "*"
## 12  ( 1 ) "*"                  " "     "*"             "*"
## 13  ( 1 ) "*"                  " "     "*"             "*"
## 14  ( 1 ) "*"                  " "     "*"             "*"
## 15  ( 1 ) "*"                  " "     "*"             "*"
## 16  ( 1 ) "*"                  " "     "*"             "*"
## 17  ( 1 ) "*"                  "*"     "*"             "*"
## 18  ( 1 ) "*"                  "*"     "*"             "*"
## 19  ( 1 ) "*"                  "*"     "*"             "*"
```

```
## 20  ( 1 ) "*"                  "*"     "*"            "*"
## 21  ( 1 ) "*"                  "*"     "*"            "*"
##           poly(alcohol, 3)2 poly(alcohol, 3)3
## 1  ( 1 )  " "                " "
## 2  ( 1 )  " "                " "
## 3  ( 1 )  " "                " "
## 4  ( 1 )  " "                " "
## 5  ( 1 )  " "                " "
## 6  ( 1 )  " "                " "
## 7  ( 1 )  " "                " "
## 8  ( 1 )  "*"                " "
## 9  ( 1 )  "*"                " "
## 10  ( 1 ) "*"                "*"
## 11  ( 1 ) "*"                "*"
## 12  ( 1 ) "*"                "*"
## 13  ( 1 ) "*"                "*"
## 14  ( 1 ) "*"                "*"
## 15  ( 1 ) "*"                "*"
## 16  ( 1 ) "*"                "*"
## 17  ( 1 ) "*"                "*"
## 18  ( 1 ) "*"                "*"
## 19  ( 1 ) "*"                "*"
## 20  ( 1 ) "*"                "*"
## 21  ( 1 ) "*"                "*"
```

```
# the best subset model obtained according to the Cp criterion has 18 variables
which.min(reg.summary$cp)
```

```
## [1] 17
```

```
# coefficient table of estimates
coef(regfit.full, 18)
```

```
##                   (Intercept)                 fixed.acidity
##                    5.10917443                   -0.04341065
##         log(volatile.acidity)      poly(residual.sugar, 3)1
##                   -0.59394819                    5.76995362
##      poly(residual.sugar, 3)2      poly(residual.sugar, 3)3
##                   -3.28419879                   -0.68129039
##           poly(chlorides, 6)1           poly(chlorides, 6)2
##                   -2.21626987                    2.29891387
##           poly(chlorides, 6)3           poly(chlorides, 6)5
##                   -2.08417065                    1.67093933
## poly(free.sulfur.dioxide, 3)1 poly(free.sulfur.dioxide, 3)2
##                    6.28689484                   -9.43485235
## poly(free.sulfur.dioxide, 3)3          total.sulfur.dioxide
##                    4.43957889                   -0.00110285
##                       log(pH)                log(sulphates)
##                    0.52454468                    0.24874154
##             poly(alcohol, 3)1             poly(alcohol, 3)2
##                   26.47308963                    4.41366952
##             poly(alcohol, 3)3
##                   -3.23493548
```

```
# the best subset model obtained according to the BIC criterion has 12 variables
which.min(reg.summary$bic)
```

```
## [1] 12
```

```
# coefficient table of estimates
coef(regfit.full, 12)
```

```
##                   (Intercept)                 fixed.acidity
##                   5.823067853                  -0.056290986
##         log(volatile.acidity)      poly(residual.sugar, 3)1
##                  -0.603105545                   5.520555419
##      poly(residual.sugar, 3)2 poly(free.sulfur.dioxide, 3)1
##                  -3.410160408                   6.703036411
## poly(free.sulfur.dioxide, 3)2 poly(free.sulfur.dioxide, 3)3
##                  -9.433183607                   4.305030605
##          total.sulfur.dioxide                log(sulphates)
##                  -0.001294004                   0.254167451
##             poly(alcohol, 3)1             poly(alcohol, 3)2
##                  28.314547196                   4.051049059
##             poly(alcohol, 3)3
##                  -3.336932953
```

```
# the best subset model obtained according to the adjusted R2 criterion has 20 variables
which.max(reg.summary$adjr2)
```

```
## [1] 17
```

```
# coefficient table of estimates
coef(regfit.full, 20)
```

```
##                   (Intercept)                 fixed.acidity
##                   5.089489378                  -0.043006107
##         log(volatile.acidity)      poly(residual.sugar, 3)1
##                  -0.594858560                   5.817420069
##      poly(residual.sugar, 3)2      poly(residual.sugar, 3)3
##                  -3.285826911                  -0.702563475
##           poly(chlorides, 6)1           poly(chlorides, 6)2
##                  -2.261229975                   2.356440499
##           poly(chlorides, 6)3           poly(chlorides, 6)4
##                  -2.132128651                   0.559313252
##           poly(chlorides, 6)5           poly(chlorides, 6)6
##                   1.666809638                  -0.342839831
## poly(free.sulfur.dioxide, 3)1 poly(free.sulfur.dioxide, 3)2
##                   6.259612673                  -9.426049329
## poly(free.sulfur.dioxide, 3)3          total.sulfur.dioxide
##                   4.440965691                  -0.001073667
##                       log(pH)                log(sulphates)
##                   0.534933826                   0.249234480
##             poly(alcohol, 3)1             poly(alcohol, 3)2
##                  26.388965267                   4.388632861
##             poly(alcohol, 3)3
##                  -3.211471284
```

```r
# the best subset model obtained according to the Cp criterion

mlr.white.cp <- lm(quality~
                   fixed.acidity+
                   log(volatile.acidity)+
                   poly(residual.sugar, 2)+
                   poly(chlorides, 6) -
                     chlorides^4 +
                   poly(free.sulfur.dioxide, 3)+
                   total.sulfur.dioxide+
                     log(pH)+
                   log(sulphates)+
                   poly(alcohol,3), data = train.white)


# the best subset model obtained according to the BIC criterion
mlr.white.bic <- lm(quality~
                   fixed.acidity+
                   log(volatile.acidity)+
                   poly(residual.sugar, 2)+
                   poly(free.sulfur.dioxide, 3)+
                   total.sulfur.dioxide+
                   log(sulphates)+
                   poly(alcohol,3), data = train.white)


# the best subset model obtained according to the R2 adjusted criterion
mlr.white.adjr2 <- lm(quality~
                   fixed.acidity+
                     citric.acid+
                   log(volatile.acidity)+
                   poly(residual.sugar, 3)+
                   poly(chlorides, 6) -
                     chlorides^4 +
                   poly(free.sulfur.dioxide, 3)+
                   total.sulfur.dioxide+
                     log(pH)+
                   log(sulphates)+
                   poly(alcohol,3), data = train.white)


sm.cp <- summary(mlr.white.cp)
sm.bic <- summary(mlr.white.bic)
sm.adjr2 <- summary(mlr.white.adjr2)

# full original model adjusted r2 score
sm.full$adj.r.squared
```

```
## [1] 0.271407
```

```r
# transformed model adjusted r2 score
sm.trans.full$adj.r.squared
```

```
## [1] 0.3163354
```

```
# adjusted r2 scores for the best subset models obtained according to the Cp, BIC, and R2 adjusted crit
# cp
sm.cp$adj.r.squared
```

```
## [1] 0.3164759
```

```
# bic
sm.bic$adj.r.squared
```

```
## [1] 0.3132618
```

```
# adj r squared
sm.adjr2$adj.r.squared
```

```
## [1] 0.3163354
```

Train & Test MSE for MLR model fitted on full original, Train & Test MSE for MLR model fitted on regsubset. Not much change!

```
# Train & Test MSE for MLR model fitted on full original
mseTrainfull <- mean((train.white$quality-mlr.full$fitted)^2)
yhat <- predict(mlr.full,newdata=test.white)
mseTestfull <- mean((test.white$quality-yhat)^2)

mseTrainfull
```

```
## [1] 0.567478
```

```
mseTestfull
```

```
## [1] 0.5694374
```

```
# Train & Test MSE for MLR model fitted on regsubset with Cp as the criterion
mlr.white.best <- mlr.white.cp


mseTrainbest <- mean((train.white$quality-mlr.white.best$fitted)^2)
ybesthat <- predict(mlr.white.best,newdata=test.white)
mseTestbest <- mean((test.white$quality-ybesthat)^2)

mseTrainbest
```

```
## [1] 0.5312841
```

```
mseTestbest
```

```
## [1] 0.5457865
```

```
# Train & Test MSE for MLR model fitted on regsubset with BIC as the criterion
mlr.white.best <- mlr.white.bic


mseTrainbest <- mean((train.white$quality-mlr.white.best$fitted)^2)
ybesthat <- predict(mlr.white.best,newdata=test.white)
mseTestbest <- mean((test.white$quality-ybesthat)^2)

mseTrainbest
```

```
## [1] 0.534635
```

```
mseTestbest
```

```
## [1] 0.5526389
```

```
# Train & Test MSE for MLR model fitted on regsubset with adjusted R2 score as the criterion
mlr.white.best <- mlr.white.adjr2


mseTrainbest <- mean((train.white$quality-mlr.white.best$fitted)^2)
ybesthat <- predict(mlr.white.best,newdata=test.white)
mseTestbest <- mean((test.white$quality-ybesthat)^2)

mseTrainbest
```

```
## [1] 0.5311507
```

```
mseTestbest
```

```
## [1] 0.5448495
```

Train & Test MSE for MLR model fitted on regsubset with Cp as the criterion had the best Test MSE score of 0.4487168 Note that Having a lower test error score than a train error score, which happened with the MLR regressions, suggests overfitting.

Summary of MLR model fitted on regsubset with Cp as the criterion

```
sm.cp
```

```
##
## Call:
## lm(formula = quality ~ fixed.acidity + log(volatile.acidity) +
##     poly(residual.sugar, 2) + poly(chlorides, 6) - chlorides^4 +
##     poly(free.sulfur.dioxide, 3) + total.sulfur.dioxide + log(pH) +
##     log(sulphates) + poly(alcohol, 3), data = train.white)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.03500 -0.52085  0.00408  0.44756  3.14368
##
```

```
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    5.0829245  0.3846886  13.213  < 2e-16 ***
## fixed.acidity                 -0.0420485  0.0150773  -2.789  0.00531 **
## log(volatile.acidity)         -0.5966540  0.0354456 -16.833  < 2e-16 ***
## poly(residual.sugar, 2)1       5.7548266  0.9233335   6.233 5.02e-10 ***
## poly(residual.sugar, 2)2      -3.3004214  0.7577924  -4.355 1.36e-05 ***
## poly(chlorides, 6)1           -2.2824701  0.8145736  -2.802  0.00510 **
## poly(chlorides, 6)2            2.3397750  0.8047176   2.908  0.00366 **
## poly(chlorides, 6)3           -2.1147067  0.7736986  -2.733  0.00630 **
## poly(chlorides, 6)4            0.5305710  0.7464595   0.711  0.47726
## poly(chlorides, 6)5            1.6729876  0.7388772   2.264  0.02361 *
## poly(chlorides, 6)6           -0.3459866  0.7367430  -0.470  0.63865
## poly(free.sulfur.dioxide, 3)1  6.3082758  0.9794587   6.441 1.32e-10 ***
## poly(free.sulfur.dioxide, 3)2 -9.4115789  0.7477525 -12.586  < 2e-16 ***
## poly(free.sulfur.dioxide, 3)3  4.3990018  0.7403769   5.942 3.04e-09 ***
## total.sulfur.dioxide          -0.0010974  0.0003944  -2.782  0.00542 **
## log(pH)                        0.5363028  0.2774896   1.933  0.05334 .
## log(sulphates)                 0.2502057  0.0512288   4.884 1.08e-06 ***
## poly(alcohol, 3)1             26.2592484  1.0238273  25.648  < 2e-16 ***
## poly(alcohol, 3)2              4.4167708  0.8021833   5.506 3.88e-08 ***
## poly(alcohol, 3)3             -3.2138886  0.7530592  -4.268 2.02e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7306 on 4382 degrees of freedom
## Multiple R-squared:  0.3194, Adjusted R-squared:  0.3165
## F-statistic: 108.2 on 19 and 4382 DF,  p-value: < 2.2e-16
```

Shrinkage Methods (LASSO vs Ridge)

LASSO

```
# format an x and y for the lasso model - since assumptions are the same as MLR, use the variable set u

#x.shrinkage <- model.matrix(quality ~ . - quality.good - quality.factor, train.white)[, -1] #throws ou


x.shrinkage <- model.matrix(quality ~
                              fixed.acidity+
              log(volatile.acidity)+
              (citric.acid)+
              poly(residual.sugar, 3)+
              poly(chlorides,6)+
              poly(free.sulfur.dioxide, 3)+
              (total.sulfur.dioxide)+
              log(pH)+
              log(sulphates)+
              poly(alcohol,3), train.white)[, -1] #throws out first column because its all 1's in the d
```

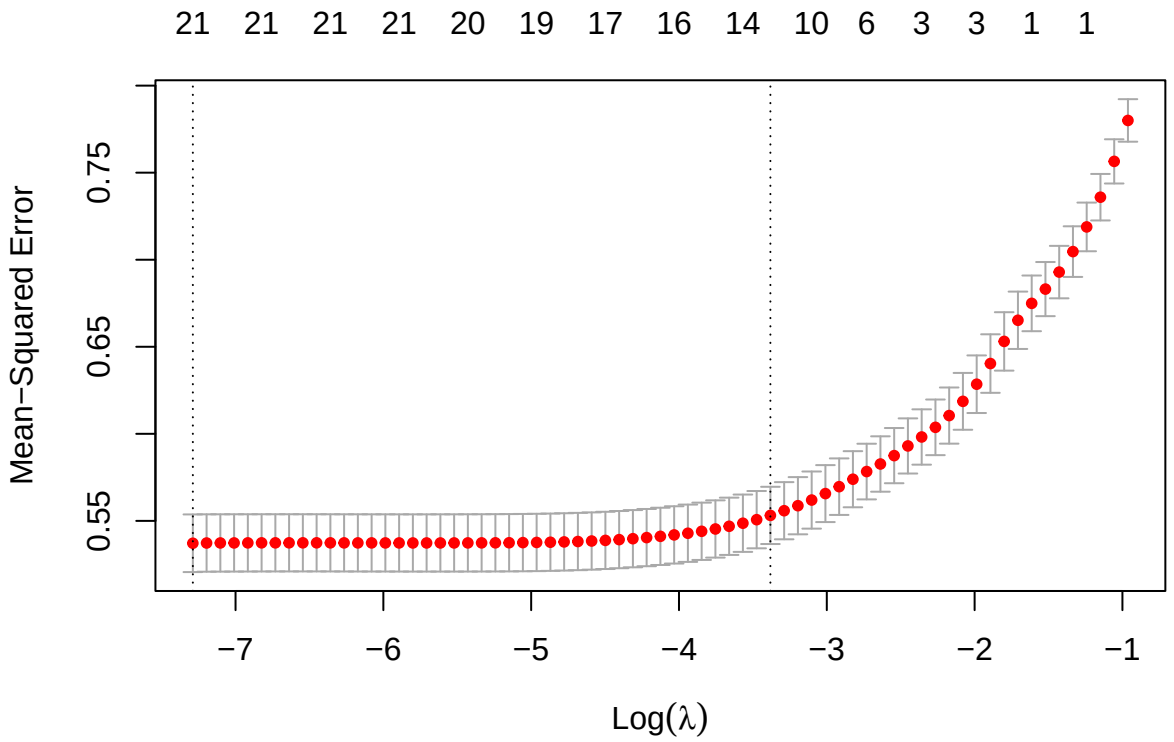
```
head(x.shrinkage) # removed intercept term
```

```
##   fixed.acidity log(volatile.acidity) citric.acid poly(residual.sugar, 3)1
## 1           7.0             -1.309333        0.36              0.043333320
## 2           6.3             -1.203973        0.34             -0.014334972
## 3           8.1             -1.272966        0.40              0.001667224
## 4           7.2             -1.469676        0.32              0.006498076
## 5           8.1             -1.272966        0.40              0.001667224
## 6           6.2             -1.139434        0.16              0.001969152
##   poly(residual.sugar, 3)2 poly(residual.sugar, 3)3 poly(chlorides, 6)1
## 1              0.077011796              0.086088135       -0.0005084884
## 2              0.009515244             -0.004554433        0.0023257833
## 3             -0.016248926              0.012677623        0.0030343512
## 4             -0.017310777              0.004261138        0.0087028947
## 5             -0.016248926              0.012677623        0.0030343512
## 6             -0.016406542              0.012247538       -0.0005084884
##   poly(chlorides, 6)2 poly(chlorides, 6)3 poly(chlorides, 6)4
## 1       -0.003380850         0.004661714       -0.0057538436
## 2       -0.006993993         0.006817867       -0.0053686952
## 3       -0.007852522         0.007259219       -0.0050895944
## 4       -0.014076265         0.009463523       -0.0006174362
## 5       -0.007852522         0.007259219       -0.0050895944
## 6       -0.003380850         0.004661714       -0.0057538436
##   poly(chlorides, 6)5 poly(chlorides, 6)6 poly(free.sulfur.dioxide, 3)1
## 1        0.005398142        -0.001735165                   0.008827215
## 2        0.002170835         0.002820961                  -0.019259416
## 3        0.001158154         0.003992596                  -0.004763090
## 4       -0.008905032         0.012389240                   0.010639256
## 5        0.001158154         0.003992596                  -0.004763090
## 6        0.005398142        -0.001735165                  -0.004763090
##   poly(free.sulfur.dioxide, 3)2 poly(free.sulfur.dioxide, 3)3
## 1                   -0.009461457                  0.0019624146
## 2                    0.013980636                 -0.0088566038
## 3                   -0.005335280                  0.0101473564
## 4                   -0.008989238                 -0.0002134634
## 5                   -0.005335280                  0.0101473564
## 6                   -0.005335280                  0.0101473564
##   total.sulfur.dioxide  log(pH) log(sulphates) poly(alcohol, 3)1
## 1                  170 1.098612     -0.7985077      -0.021018396
## 2                  132 1.193922     -0.7133499      -0.012415140
## 3                   97 1.181727     -0.8209806      -0.005040921
## 4                  186 1.160021     -0.9162907      -0.007498994
## 5                   97 1.181727     -0.8209806      -0.005040921
## 6                  136 1.156881     -0.7550226      -0.011186104
##   poly(alcohol, 3)2 poly(alcohol, 3)3
## 1       0.023728563      -0.022502428
## 2       0.001225340       0.009943601
## 3      -0.010520375       0.013306120
## 4      -0.007378753       0.014073387
## 5      -0.010520375       0.013306120
## 6      -0.001215789       0.011789670
```

```
y.shrinkage <- train.white$quality


grid <- 10^seq(10,-2,length=100) # grid of lambda values goes from 10^10 down to 10^-2 = 0.01


lasso.mod <- glmnet(x.shrinkage, y.shrinkage, alpha = 1, lambda = grid)    # alpha = 1 corresponds to la

dim(coef(lasso.mod)) # 22 coefficients by 100 lambda values
```

```
## [1]  22 100
```

```
# We can use cross-validation to choose the tuning parameter.
# By default this performs 10-fold cross-validation
set.seed(1)
cv.lasso.mod <- cv.glmnet(x.shrinkage, y.shrinkage, alpha = 1)  # alpha = 1 corresponds to lasso



plot(cv.lasso.mod)
```




```
cv.lasso.mod$lambda.min # lambda = 0.002073759
```

```
## [1] 0.0006830689
```

```
cv.lasso.mod
```

```
##
## Call:  cv.glmnet(x = x.shrinkage, y = y.shrinkage, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.00068    69  0.5371 0.01656      21
## 1se 0.03400    27  0.5531 0.01643      13
```

```
lassop <- predict(lasso.mod, s=cv.lasso.mod$lambda.min, type="coefficients" )
lassop
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                          s1
## (Intercept)                     5.2801125142
## fixed.acidity                  -0.0419867028
## log(volatile.acidity)          -0.5711562116
## citric.acid                     .
## poly(residual.sugar, 3)1        4.3889957980
## poly(residual.sugar, 3)2       -2.3419779236
## poly(residual.sugar, 3)3        .
## poly(chlorides, 6)1            -1.8942632597
## poly(chlorides, 6)2             1.7163418871
## poly(chlorides, 6)3            -1.4094267337
## poly(chlorides, 6)4             .
## poly(chlorides, 6)5             0.9998751934
## poly(chlorides, 6)6             .
## poly(free.sulfur.dioxide, 3)1   5.1113007090
## poly(free.sulfur.dioxide, 3)2 -8.8694906765
## poly(free.sulfur.dioxide, 3)3   3.7352717909
## total.sulfur.dioxide           -0.0004537082
## log(pH)                         0.2753431331
## log(sulphates)                  0.1823613462
## poly(alcohol, 3)1              26.1670905705
## poly(alcohol, 3)2               3.6454722363
## poly(alcohol, 3)3              -2.5847857155
```

```
# train mse
lasso.pred.y <- predict(lasso.mod, x.shrinkage, s=cv.lasso.mod$lambda.min, type = "class")
train.mse.lasso <- mean((train.white$quality-lasso.pred.y)^2)
train.mse.lasso
```

```
## [1] 0.5333032
```

```
# test mse
#x.shrinkage.test <- model.matrix(quality ~ . - quality.good - quality.factor, test.white)[, -1]

x.shrinkage.test <- model.matrix(quality ~
                                   fixed.acidity+
              log(volatile.acidity)+
```

```
              (citric.acid)+
              poly(residual.sugar, 3)+
              poly(chlorides,6)+
              poly(free.sulfur.dioxide, 3)+
              (total.sulfur.dioxide)+
              log(pH)+
              log(sulphates)+
              poly(alcohol,3), test.white)[, -1]

lasso.pred.y <- predict(lasso.mod, x.shrinkage.test, s=cv.lasso.mod$lambda.min, type = "class")
test.mse.lasso <- mean((test.white$quality-lasso.pred.y)^2)
test.mse.lasso
```

```
## [1] 1.184434
```

Lasso Train MSE = 0.5442927 Lasso Test MSE = 1.097064

Ridge

```
# format an x and y for the ridge model
#x.shrinkage <- model.matrix(quality ~ . - quality.good - quality.factor, train.white)[, -1] #throws ou

head(x.shrinkage) # removed intercept term
```

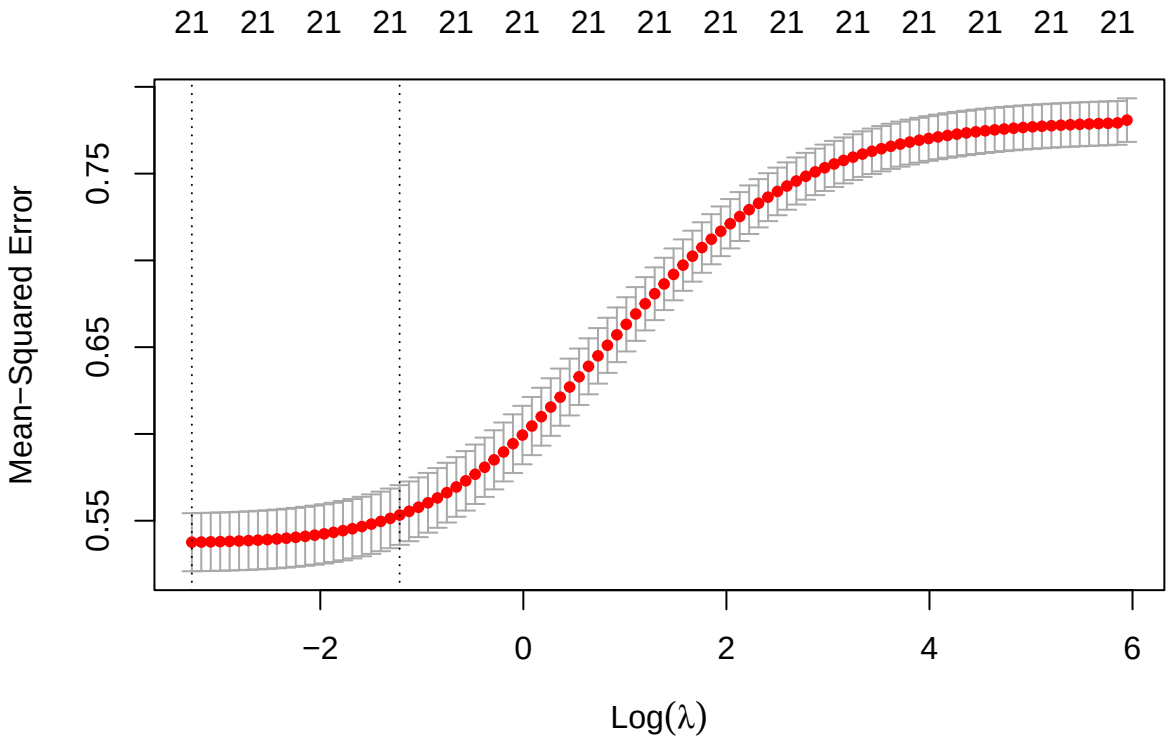
```
##   fixed.acidity log(volatile.acidity) citric.acid poly(residual.sugar, 3)1
## 1           7.0             -1.309333        0.36              0.043333320
## 2           6.3             -1.203973        0.34             -0.014334972
## 3           8.1             -1.272966        0.40              0.001667224
## 4           7.2             -1.469676        0.32              0.006498076
## 5           8.1             -1.272966        0.40              0.001667224
## 6           6.2             -1.139434        0.16              0.001969152
##   poly(residual.sugar, 3)2 poly(residual.sugar, 3)3 poly(chlorides, 6)1
## 1              0.077011796              0.086088135       -0.0005084884
## 2              0.009515244             -0.004554433        0.0023257833
## 3             -0.016248926              0.012677623        0.0030343512
## 4             -0.017310777              0.004261138        0.0087028947
## 5             -0.016248926              0.012677623        0.0030343512
## 6             -0.016406542              0.012247538       -0.0005084884
##   poly(chlorides, 6)2 poly(chlorides, 6)3 poly(chlorides, 6)4
## 1       -0.003380850         0.004661714       -0.0057538436
## 2       -0.006993993         0.006817867       -0.0053686952
## 3       -0.007852522         0.007259219       -0.0050895944
## 4       -0.014076265         0.009463523       -0.0006174362
## 5       -0.007852522         0.007259219       -0.0050895944
## 6       -0.003380850         0.004661714       -0.0057538436
##   poly(chlorides, 6)5 poly(chlorides, 6)6 poly(free.sulfur.dioxide, 3)1
## 1         0.005398142        -0.001735165                   0.008827215
## 2         0.002170835         0.002820961                  -0.019259416
## 3         0.001158154         0.003992596                  -0.004763090
## 4        -0.008905032         0.012389240                   0.010639256
## 5         0.001158154         0.003992596                  -0.004763090
```

```
## 6         0.005398142        -0.001735165                  -0.004763090
##   poly(free.sulfur.dioxide, 3)2 poly(free.sulfur.dioxide, 3)3
## 1                   -0.009461457                  0.0019624146
## 2                    0.013980636                 -0.0088566038
## 3                   -0.005335280                  0.0101473564
## 4                   -0.008989238                 -0.0002134634
## 5                   -0.005335280                  0.0101473564
## 6                   -0.005335280                  0.0101473564
##   total.sulfur.dioxide  log(pH) log(sulphates) poly(alcohol, 3)1
## 1                  170 1.098612     -0.7985077      -0.021018396
## 2                  132 1.193922     -0.7133499      -0.012415140
## 3                   97 1.181727     -0.8209806      -0.005040921
## 4                  186 1.160021     -0.9162907      -0.007498994
## 5                   97 1.181727     -0.8209806      -0.005040921
## 6                  136 1.156881     -0.7550226      -0.011186104
##   poly(alcohol, 3)2 poly(alcohol, 3)3
## 1       0.023728563      -0.022502428
## 2       0.001225340       0.009943601
## 3      -0.010520375       0.013306120
## 4      -0.007378753       0.014073387
## 5      -0.010520375       0.013306120
## 6      -0.001215789       0.011789670
```

```
y.shrinkage <- train.white$quality


grid <- 10^seq(10,-2,length=100) # grid of lambda values goes from 10^10 down to 10^-2 = 0.01


ridge.mod <- glmnet(x.shrinkage, y.shrinkage, alpha = 0, lambda = grid) # alpha = 0 corresponds to ridg


dim(coef(ridge.mod)) # 22 coefficients by 100 lambda values
```

```
## [1]  22 100
```

```
# We can use cross-validation to choose the tuning parameter.
# By default this performs 10-fold cross-validation
set.seed(1)
cv.ridge.mod <- cv.glmnet(x.shrinkage, y.shrinkage, alpha = 0)  # alpha = 0 corresponds to ridge


plot(cv.ridge.mod) # needs to be tuned more, should be between two negative values
```




```
cv.ridge.mod$lambda.min # lambda = 0.0379651
```

```
## [1] 0.03818909
```

```
cv.ridge.mod
```

```
##
## Call:  cv.glmnet(x = x.shrinkage, y = y.shrinkage, alpha = 0)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.03819   100  0.5376 0.01673      21
## 1se 0.29568    78  0.5533 0.01720      21
```

```
ridgep <- predict(ridge.mod, s=cv.ridge.mod$lambda.min, type="coefficients" )
ridgep
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                5.119964359
## fixed.acidity             -0.041992866
## log(volatile.acidity)     -0.563322950
## citric.acid               -0.023677439
```

```
## poly(residual.sugar, 3)1       4.935300967
## poly(residual.sugar, 3)2      -3.111397812
## poly(residual.sugar, 3)3      -0.508874326
## poly(chlorides, 6)1           -2.765021745
## poly(chlorides, 6)2            2.692418125
## poly(chlorides, 6)3           -2.289723719
## poly(chlorides, 6)4            0.636149258
## poly(chlorides, 6)5            1.599532258
## poly(chlorides, 6)6           -0.401623233
## poly(free.sulfur.dioxide, 3)1  6.030403059
## poly(free.sulfur.dioxide, 3)2 -9.173201951
## poly(free.sulfur.dioxide, 3)3  4.381866460
## total.sulfur.dioxide          -0.001081351
## log(pH)                        0.538816344
## log(sulphates)                 0.236547815
## poly(alcohol, 3)1             24.417031895
## poly(alcohol, 3)2              4.324575877
## poly(alcohol, 3)3             -3.135477653
```

```
# train mse
ridge.pred.y <- predict(ridge.mod, x.shrinkage, s=cv.ridge.mod$lambda.min, type = "class")
train.mse.ridge <- mean((train.white$quality-ridge.pred.y)^2)
train.mse.ridge
```

```
## [1] 0.5317972
```

```
# test mse
#x.shrinkage.test <- model.matrix(quality ~ . - quality.good - quality.factor, test.white)[, -1]
ridge.pred.y <- predict(ridge.mod, x.shrinkage.test, s=cv.ridge.mod$lambda.min, type = "class")
test.mse.ridge <- mean((test.white$quality-ridge.pred.y)^2)
test.mse.ridge
```

```
## [1] 1.17227
```

Ridge Train MSE = 0.5425599 Ridge Test MSE = 1.116491

Ridge Test MSE (1.116491) > Lasso Test MSE (1.097064), so Lasso Model was the better of the two shrinkage methods, with a lambda tuned to 0.002073759

The optimized MLR Test MSE was 0.4487168, so of the two regression models, the MLR was the best.

Classification

SVM

```
# Tuning kernel and cost
set.seed(100)
rg <- list(kernel=c("linear","polynomial","radial","sigmoid"), cost=seq(1,3,.5))
rg
```

```
## $kernel
## [1] "linear"     "polynomial" "radial"     "sigmoid"
##
## $cost
## [1] 1.0 1.5 2.0 2.5 3.0
```

```
svmt <- tune(svm,quality.good ~ . - quality - quality.factor - density, data=train.white, ranges = rg)
svmt # radial permorms best, with a cost of 3
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  kernel cost
##  radial    3
##
## - best performance: 0.2128649
```

radial permorms best, with a cost of 3

```
svm.tuned <- svm(quality.good ~ . - quality - quality.factor- density, data= train.white,kernel="radial
svm.tuned
```

```
##
## Call:
## svm(formula = quality.good ~ . - quality - quality.factor - density,
##     data = train.white, kernel = "radial", cost = 3)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  3
##
## Number of Support Vectors:  2328
```

```
#plot(svm.tuned,College,Expend ~ S.F.Ratio)
#plot(svm.tuned,College,F.Undergrad ~ Outstate)
yhat = predict(svm.tuned)
table(yhat,train.white$quality.good)
```

```
##
## yhat    0    1
##    0  989  266
##    1  476 2671
```

```
mean(yhat != train.white$quality.good) # Train error rate
```

```
## [1] 0.1685597
```

```
test.yhat <- predict(svm.tuned, test.white)
table(test.yhat,test.white$quality.good)
```

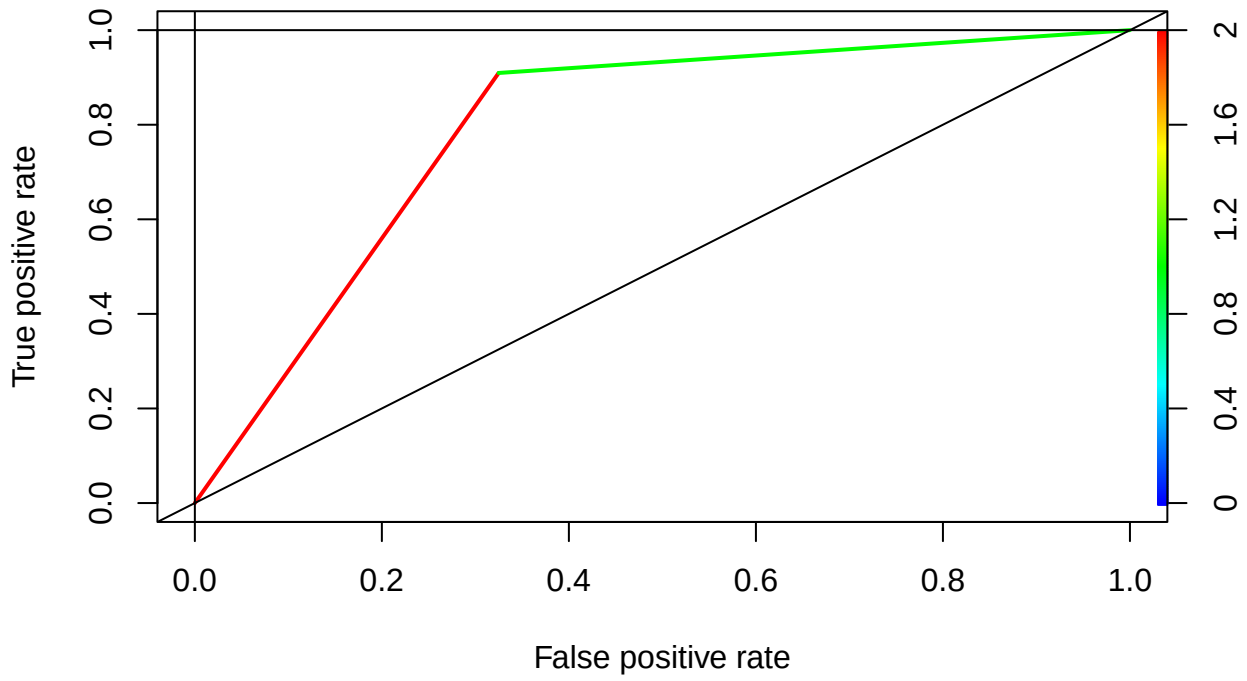
```
##
## test.yhat   0   1
##         0 116  27
##         1  58 289
```

```
mean(test.yhat != test.white$quality.good) # Test error rate
```

```
## [1] 0.1734694
```

SVM Train error rate = 0.1678782 SVM Test error rate = 0.1938776

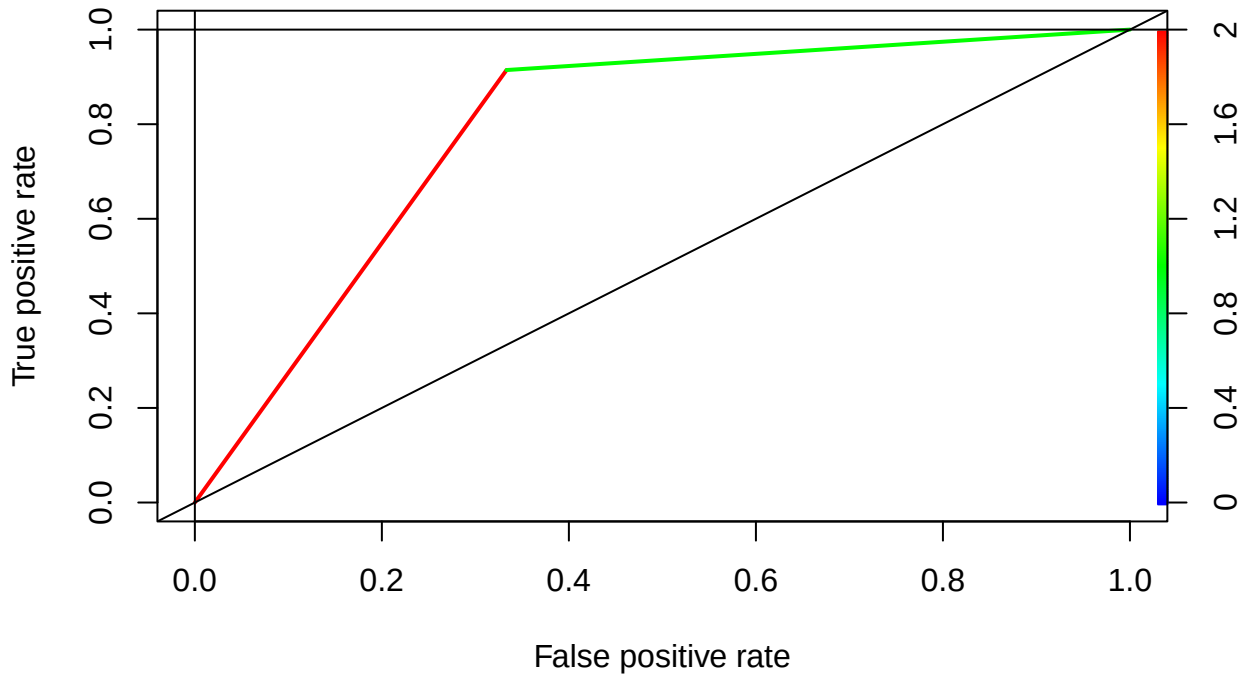
```
# ROC curve - Train
pred <- prediction(as.numeric(yhat=='1'),as.numeric(train.white$quality.good=='1'))
perf <- performance(pred,"tpr","fpr")
plot(perf,colorize=T,lwd=2)
abline(a=0,b=1)
abline(h=1)
abline(v=0)
```



```
performance(pred,measure="auc")@y.values
```

```
## [[1]]
## [1] 0.7922584
```

```
# ROC curve - Test
pred <- prediction(as.numeric(test.yhat=='1'),as.numeric(test.white$quality.good=='1'))
perf <- performance(pred,"tpr","fpr")
plot(perf,colorize=T,lwd=2)
abline(a=0,b=1)
abline(h=1)
abline(v=0)
```




```
performance(pred,measure="auc")@y.values
```

```
## [[1]]
## [1] 0.7906118
```

Bayes Classifiers

```
# 10-fold cross validation
set.seed(1200)
```

```
k <- 10
n <- dim(train.white)[1] # train.white is my complete cases data
ids <- sample(1:n,n)

i1 <- (0*(n/k)+1):(1*(n/k))
i2 <- ((n/k)+1):(2*(n/k))
i3 <- (2*(n/k)+1):(3*(n/k))
i4 <- (3*(n/k)+1):(4*(n/k))
i5 <- (4*(n/k)+1):(5*(n/k))
i6 <- (5*(n/k)+1):(6*(n/k))
i7 <- (6*(n/k)+1):(7*(n/k))
i8 <- (7*(n/k)+1):(8*(n/k))
i9 <- (8*(n/k)+1):(9*(n/k))
i10 <- (9*(n/k)+1):(10*(n/k))

v1 <- ids[i1]  # individuals selected for first validation data set
t1 <- ids[-i1] # individuals in the first training data set
t2 <- ids[-i2]
v2 <- ids[i2]
t3 <- ids[-i3]
v3 <- ids[i3]
t4 <- ids[-i4]
v4 <- ids[i4]
v5 <- ids[i5]
t5 <- ids[-i5]
t6 <- ids[-i6]
v6 <- ids[i6]
t7 <- ids[-i7]
v7 <- ids[i7]
t8 <- ids[-i8]
v8 <- ids[i8]
t9 <- ids[-i9]
v9 <- ids[i9]
t10 <- ids[-i10]
v10 <- ids[i10]

train1 <- train.white[t1,] # first training data set
val1 <- train.white[v1,] # first validation/test data set

train2 <- train.white[t2,] # second training data set
val2 <- train.white[v2,] # second validation/test data set

train3 <- train.white[t3,] # third training data set
val3 <- train.white[v3,] # third validation/test data set

train4 <- train.white[t4,] # fourth training data set
val4 <- train.white[v4,] # fourth validation/test data set


train5 <- train.white[t5,] # fifth training data set
val5 <- train.white[v5,] # fifth validation/test data set

train6 <- train.white[t6,] # sixth training data set
```

```
val6 <- train.white[v6,] # sixth validation/test data set

train7 <- train.white[t7,] # seventh training data set
val7 <- train.white[v7,] # seventh validation/test data set

train8 <- train.white[t8,] # eighth training data set
val8 <- train.white[v8,] # eighth validation/test data set

train9 <- train.white[t9,] # ninth training data set
val9 <- train.white[v9,] # ninth validation/test data set

train10 <- train.white[t10,] # tenth training data set
val10 <- train.white[v10,] # tenth validation/test data set


nrow(val1)
```

```
## [1] 440
```

```
#val1
```

Graphs observing largest associations between all predictors and response variable commented out for sake of not making a 200 page pdf after knitting

```
# # 1st fold/training set
# plot(train1$quality.good, train1$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train1')
# plot(train1$quality.good, train1$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train1'
# plot(train1$quality.good, train1$citric.acid, xlab = 'quality', ylab = 'citric.acid train1')
# plot(train1$quality.good, train1$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train1')
# plot(train1$quality.good, train1$chlorides, xlab = 'quality', ylab = 'chlorides train1')
# plot(train1$quality.good, train1$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train1$quality.good, train1$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train1$quality.good, train1$density, xlab = 'quality', ylab = 'density train1')
# plot(train1$quality.good, train1$pH, xlab = 'quality', ylab = 'pH train1')
# plot(train1$quality.good, train1$sulphates, xlab = 'quality', ylab = 'sulphates train1')
# plot(train1$quality.good, train1$alcohol, xlab = 'quality', ylab = 'alcohol train1')
#
# # 2nd fold/training set
# plot(train2$quality.good, train2$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train2')
# plot(train2$quality.good, train2$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train2'
# plot(train2$quality.good, train1$citric.acid, xlab = 'quality', ylab = 'citric.acid train2')
# plot(train2$quality.good, train1$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train2')
# plot(train2$quality.good, train1$chlorides, xlab = 'quality', ylab = 'chlorides train2')
# plot(train2$quality.good, train1$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train2$quality.good, train1$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train2$quality.good, train1$density, xlab = 'quality', ylab = 'density train2')
# plot(train2$quality.good, train1$pH, xlab = 'quality', ylab = 'pH train2')
# plot(train2$quality.good, train1$sulphates, xlab = 'quality', ylab = 'sulphates train2')
# plot(train2$quality.good, train1$alcohol, xlab = 'quality', ylab = 'alcohol train2')
#
# # 3rd fold/training set
# plot(train3$quality.good, train3$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train3')
# plot(train3$quality.good, train3$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train3'
```

```
# plot(train3$quality.good, train3$citric.acid, xlab = 'quality', ylab = 'citric.acid train3')
# plot(train3$quality.good, train3$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train3')
# plot(train3$quality.good, train3$chlorides, xlab = 'quality', ylab = 'chlorides train3')
# plot(train3$quality.good, train3$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train3$quality.good, train3$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train3$quality.good, train3$density, xlab = 'quality', ylab = 'density train3')
# plot(train3$quality.good, train3$pH, xlab = 'quality', ylab = 'pH train3')
# plot(train3$quality.good, train3$sulphates, xlab = 'quality', ylab = 'sulphates train3')
# plot(train3$quality.good, train3$alcohol, xlab = 'quality', ylab = 'alcohol train3')
#
# # 4th fold/training set
# plot(train4$quality.good, train4$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train4')
# plot(train4$quality.good, train4$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train4'
# plot(train4$quality.good, train4$citric.acid, xlab = 'quality', ylab = 'citric.acid train4')
# plot(train4$quality.good, train4$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train4')
# plot(train4$quality.good, train4$chlorides, xlab = 'quality', ylab = 'chlorides train4')
# plot(train4$quality.good, train4$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train4$quality.good, train4$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train4$quality.good, train4$density, xlab = 'quality', ylab = 'density train4')
# plot(train4$quality.good, train4$pH, xlab = 'quality', ylab = 'pH train4')
# plot(train4$quality.good, train4$sulphates, xlab = 'quality', ylab = 'sulphates train4')
# plot(train4$quality.good, train4$alcohol, xlab = 'quality', ylab = 'alcohol train4')
#
# # 5th fold/training set
# plot(train5$quality.good, train5$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train5')
# plot(train5$quality.good, train5$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train5'
# plot(train5$quality.good, train5$citric.acid, xlab = 'quality', ylab = 'citric.acid train5')
# plot(train5$quality.good, train5$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train5')
# plot(train5$quality.good, train5$chlorides, xlab = 'quality', ylab = 'chlorides train5')
# plot(train5$quality.good, train5$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train5$quality.good, train5$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train5$quality.good, train5$density, xlab = 'quality', ylab = 'density train5')
# plot(train5$quality.good, train5$pH, xlab = 'quality', ylab = 'pH train5')
# plot(train5$quality.good, train5$sulphates, xlab = 'quality', ylab = 'sulphates train5')
# plot(train5$quality.good, train5$alcohol, xlab = 'quality', ylab = 'alcohol train5')
#
# # 6th fold/training set
# plot(train6$quality.good, train6$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train6')
# plot(train6$quality.good, train6$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train6'
# plot(train6$quality.good, train6$citric.acid, xlab = 'quality', ylab = 'citric.acid train6')
# plot(train6$quality.good, train6$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train6')
# plot(train6$quality.good, train6$chlorides, xlab = 'quality', ylab = 'chlorides train6')
# plot(train6$quality.good, train6$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train6$quality.good, train6$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train6$quality.good, train6$density, xlab = 'quality', ylab = 'density train6')
# plot(train6$quality.good, train6$pH, xlab = 'quality', ylab = 'pH train6')
# plot(train6$quality.good, train6$sulphates, xlab = 'quality', ylab = 'sulphates train6')
# plot(train6$quality.good, train6$alcohol, xlab = 'quality', ylab = 'alcohol train6')
#
# # 7th fold/training set
# plot(train7$quality.good, train7$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train7')
# plot(train7$quality.good, train7$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train7'
# plot(train7$quality.good, train7$citric.acid, xlab = 'quality', ylab = 'citric.acid train7')
```

```
# plot(train7$quality.good, train7$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train7')
# plot(train7$quality.good, train7$chlorides, xlab = 'quality', ylab = 'chlorides train7')
# plot(train7$quality.good, train7$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train7$quality.good, train7$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train7$quality.good, train7$density, xlab = 'quality', ylab = 'density train7')
# plot(train7$quality.good, train7$pH, xlab = 'quality', ylab = 'pH train7')
# plot(train7$quality.good, train7$sulphates, xlab = 'quality', ylab = 'sulphates train7')
# plot(train7$quality.good, train7$alcohol, xlab = 'quality', ylab = 'alcohol train7')
#
# # 8th fold/training set
# plot(train8$quality.good, train8$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train8')
# plot(train8$quality.good, train8$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train8'
# plot(train8$quality.good, train8$citric.acid, xlab = 'quality', ylab = 'citric.acid train8')
# plot(train8$quality.good, train8$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train8')
# plot(train8$quality.good, train8$chlorides, xlab = 'quality', ylab = 'chlorides train8')
# plot(train8$quality.good, train8$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train8$quality.good, train8$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train8$quality.good, train8$density, xlab = 'quality', ylab = 'density train8')
# plot(train8$quality.good, train8$pH, xlab = 'quality', ylab = 'pH train8')
# plot(train8$quality.good, train8$sulphates, xlab = 'quality', ylab = 'sulphates train8')
# plot(train8$quality.good, train8$alcohol, xlab = 'quality', ylab = 'alcohol train8')
#
# # 9th fold/training set
# plot(train9$quality.good, train9$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train9')
# plot(train9$quality.good, train9$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train9'
# plot(train9$quality.good, train9$citric.acid, xlab = 'quality', ylab = 'citric.acid train9')
# plot(train9$quality.good, train9$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train9')
# plot(train9$quality.good, train9$chlorides, xlab = 'quality', ylab = 'chlorides train9')
# plot(train9$quality.good, train9$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide t
# plot(train9$quality.good, train9$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxide
# plot(train9$quality.good, train9$density, xlab = 'quality', ylab = 'density train9')
# plot(train9$quality.good, train9$pH, xlab = 'quality', ylab = 'pH train9')
# plot(train9$quality.good, train9$sulphates, xlab = 'quality', ylab = 'sulphates train9')
# plot(train9$quality.good, train9$alcohol, xlab = 'quality', ylab = 'alcohol train9')
#
# # 10th fold/training set
# plot(train10$quality.good, train10$fixed.acidity, xlab = 'quality', ylab = 'fixed.acidity train10')
# plot(train10$quality.good, train10$volatile.acidity, xlab = 'quality', ylab = 'volatile.acidity train
# plot(train10$quality.good, train10$citric.acid, xlab = 'quality', ylab = 'citric.acid train10')
# plot(train10$quality.good, train10$residual.sugar, xlab = 'quality', ylab = 'residual.sugar train10')
# plot(train10$quality.good, train10$chlorides, xlab = 'quality', ylab = 'chlorides train10')
# plot(train10$quality.good, train10$free.sulfur.dioxide, xlab = 'quality', ylab = 'free.sulfur.dioxide
# plot(train10$quality.good, train10$total.sulfur.dioxide, xlab = 'quality', ylab = 'total.sulfur.dioxi
# plot(train10$quality.good, train10$density, xlab = 'quality', ylab = 'density train10')
# plot(train10$quality.good, train10$pH, xlab = 'quality', ylab = 'pH train10')
# plot(train10$quality.good, train10$sulphates, xlab = 'quality', ylab = 'sulphates train10')
# plot(train10$quality.good, train10$alcohol, xlab = 'quality', ylab = 'alcohol train10')
```

```
# error rate of lda model for 1st train/val set
lda.fit1 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train1)
lda.yhat1 <- predict(lda.fit1,val1)
lda.error.rate1 <- sum(lda.yhat1$class!=val1$quality.good)/dim(val1)[1]
#lda.error.rate1
```

```
# error rate of lda model for 2nd train/val set
lda.fit2 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train2)
lda.yhat2 <- predict(lda.fit2,val2)
lda.error.rate2 <- sum(lda.yhat2$class!=val2$quality.good)/dim(val2)[1]
#lda.error.rate2

# error rate of lda model for 3rd train/val set
lda.fit3 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train3)
lda.yhat3 <- predict(lda.fit3,val3)
lda.error.rate3 <- sum(lda.yhat3$class!=val3$quality.good)/dim(val3)[1]
#lda.error.rate3


# error rate of lda model for 4th train/val set
lda.fit4 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train4)
lda.yhat4 <- predict(lda.fit4,val4)
lda.error.rate4 <- sum(lda.yhat4$class!=val4$quality.good)/dim(val4)[1]
#lda.error.rate4


# error rate of lda model for 5th train/val set
lda.fit5 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train5)
lda.yhat5 <- predict(lda.fit5,val5)
lda.error.rate5 <- sum(lda.yhat5$class!=val5$quality.good)/dim(val5)[1]
#lda.error.rate5


# error rate of lda model for 6th train/val set
lda.fit6 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train6)
lda.yhat6 <- predict(lda.fit6,val6)
lda.error.rate6 <- sum(lda.yhat6$class!=val6$quality.good)/dim(val6)[1]
#lda.error.rate6

# error rate of lda model for 7th train/val set
lda.fit7 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train7)
lda.yhat7 <- predict(lda.fit7,val7)
lda.error.rate7 <- sum(lda.yhat7$class!=val7$quality.good)/dim(val7)[1]
#lda.error.rate7


# error rate of lda model for 8th train/val set
lda.fit8 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train8)
lda.yhat8 <- predict(lda.fit8,val8)
lda.error.rate8 <- sum(lda.yhat8$class!=val8$quality.good)/dim(val8)[1]
#lda.error.rate8


# error rate of lda model for 9th train/val set
lda.fit9 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train9)
lda.yhat9 <- predict(lda.fit9,val9)
lda.error.rate9 <- sum(lda.yhat9$class!=val9$quality.good)/dim(val9)[1]
#lda.error.rate9
```

```r
# error rate of lda model for 10th train/val set
lda.fit10 <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train10)
lda.yhat10 <- predict(lda.fit10,val10)
lda.error.rate10 <- sum(lda.yhat10$class!=val10$quality.good)/dim(val10)[1]
#lda.error.rate10


# overall lda train error rate
lda.error.rate.overall <- (lda.error.rate1 + lda.error.rate2 + lda.error.rate3 + lda.error.rate4 + lda.
lda.error.rate.overall
```

```
## [1] 0.2536364
```

```r
# test error rate of lda model for full test set
lda.fit.final <- lda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train.white)
lda.yhat.final <- predict(lda.fit.final,test.white)
lda.error.rate.final <- sum(lda.yhat.final$class!=test.white$quality.good)/dim(test.white)[1]
lda.error.rate.final
```

```
## [1] 0.255102
```

```r
# error rate of qda model for 1st train/val set
qda.fit1 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train1)
qda.yhat1 <- predict(qda.fit1,val1)
qda.error.rate1 <- sum(qda.yhat1$class!=val1$quality.good)/dim(val1)[1]
#qda.error.rate1


# error rate of qda model for 2nd train/val set
qda.fit2 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train2)
qda.yhat2 <- predict(qda.fit2,val2)
qda.error.rate2 <- sum(qda.yhat2$class!=val2$quality.good)/dim(val2)[1]
#qda.error.rate2

# error rate of qda model for 3rd train/val set
qda.fit3 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train3)
qda.yhat3 <- predict(qda.fit3,val3)
qda.error.rate3 <- sum(qda.yhat3$class!=val3$quality.good)/dim(val3)[1]
#qda.error.rate3


# error rate of qda model for 4th train/val set
qda.fit4 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train4)
qda.yhat4 <- predict(qda.fit4,val4)
qda.error.rate4 <- sum(qda.yhat4$class!=val4$quality.good)/dim(val4)[1]
#qda.error.rate4


# error rate of qda model for 5th train/val set
qda.fit5 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train5)
qda.yhat5 <- predict(qda.fit5,val5)
```

```
qda.error.rate5 <- sum(qda.yhat5$class!=val5$quality.good)/dim(val5)[1]
#qda.error.rate5


# error rate of qda model for 6th train/val set
qda.fit6 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train6)
qda.yhat6 <- predict(qda.fit6,val6)
qda.error.rate6 <- sum(qda.yhat6$class!=val6$quality.good)/dim(val6)[1]
#qda.error.rate6

# error rate of qda model for 7th train/val set
qda.fit7 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train7)
qda.yhat7 <- predict(qda.fit7,val7)
qda.error.rate7 <- sum(qda.yhat7$class!=val7$quality.good)/dim(val7)[1]
#qda.error.rate7


# error rate of qda model for 8th train/val set
qda.fit8 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train8)
qda.yhat8 <- predict(qda.fit8,val8)
qda.error.rate8 <- sum(qda.yhat8$class!=val8$quality.good)/dim(val8)[1]
#qda.error.rate8


# error rate of qda model for 9th train/val set
qda.fit9 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train9)
qda.yhat9 <- predict(qda.fit9,val9)
qda.error.rate9 <- sum(qda.yhat9$class!=val9$quality.good)/dim(val9)[1]
#qda.error.rate9


# error rate of qda model for 10th train/val set
qda.fit10 <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train10)
qda.yhat10 <- predict(qda.fit10,val10)
qda.error.rate10 <- sum(qda.yhat10$class!=val10$quality.good)/dim(val10)[1]
#qda.error.rate10



# overall qda train error rate
qda.error.rate.overall <- (qda.error.rate1 + qda.error.rate2 + qda.error.rate3 + qda.error.rate4 + qda.
qda.error.rate.overall
```

```
## [1] 0.255
```

```
# test error rate of qda model for full test set
qda.fit.final <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train.white)
qda.yhat.final <- predict(qda.fit.final,test.white)
qda.error.rate.final <- sum(qda.yhat.final$class!=test.white$quality.good)/dim(test.white)[1]
qda.error.rate.final
```

```
## [1] 0.244898
```

Remember to remove density from the Naive Bayes model to address the issue of multicolinearity

```
# error rate of naiveBayes model for 1st train/val set
naiveBayes.fit1 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train1)
naiveBayes.yhat1 <- predict(naiveBayes.fit1,val1)
naiveBayes.error.rate1 <- sum(naiveBayes.yhat1!=val1$quality.good)/dim(val1)[1]
#naiveBayes.error.rate1



# error rate of naiveBayes model for 2nd train/val set
naiveBayes.fit2 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train2)
naiveBayes.yhat2 <- predict(naiveBayes.fit2,val2)
naiveBayes.error.rate2 <- sum(naiveBayes.yhat2!=val2$quality.good)/dim(val2)[1]
#naiveBayes.error.rate2

# error rate of naiveBayes model for 3rd train/val set
naiveBayes.fit3 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train3)
naiveBayes.yhat3 <- predict(naiveBayes.fit3,val3)
naiveBayes.error.rate3 <- sum(naiveBayes.yhat3!=val3$quality.good)/dim(val3)[1]
#naiveBayes.error.rate3


# error rate of naiveBayes model for 4th train/val set
naiveBayes.fit4 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train4)
naiveBayes.yhat4 <- predict(naiveBayes.fit4,val4)
naiveBayes.error.rate4 <- sum(naiveBayes.yhat4!=val4$quality.good)/dim(val4)[1]
#naiveBayes.error.rate4


# error rate of naiveBayes model for 5th train/val set
naiveBayes.fit5 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train5)
naiveBayes.yhat5 <- predict(naiveBayes.fit5,val5)
naiveBayes.error.rate5 <- sum(naiveBayes.yhat5!=val5$quality.good)/dim(val5)[1]
#naiveBayes.error.rate5


# error rate of naiveBayes model for 6th train/val set
naiveBayes.fit6 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train6)
naiveBayes.yhat6 <- predict(naiveBayes.fit6,val6)
naiveBayes.error.rate6 <- sum(naiveBayes.yhat6!=val6$quality.good)/dim(val6)[1]
#naiveBayes.error.rate6

# error rate of naiveBayes model for 7th train/val set
naiveBayes.fit7 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train7)
naiveBayes.yhat7 <- predict(naiveBayes.fit7,val7)
naiveBayes.error.rate7 <- sum(naiveBayes.yhat7!=val7$quality.good)/dim(val7)[1]
#naiveBayes.error.rate7


# error rate of naiveBayes model for 8th train/val set
naiveBayes.fit8 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train8)
naiveBayes.yhat8 <- predict(naiveBayes.fit8,val8)
naiveBayes.error.rate8 <- sum(naiveBayes.yhat8!=val8$quality.good)/dim(val8)[1]
```

```r
#naiveBayes.error.rate8


# error rate of naiveBayes model for 9th train/val set
naiveBayes.fit9 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train9)
naiveBayes.yhat9 <- predict(naiveBayes.fit9,val9)
naiveBayes.error.rate9 <- sum(naiveBayes.yhat9!=val9$quality.good)/dim(val9)[1]
#naiveBayes.error.rate9


# error rate of naiveBayes model for 10th train/val set
naiveBayes.fit10 <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train10)
naiveBayes.yhat10 <- predict(naiveBayes.fit10,val10)
naiveBayes.error.rate10 <- sum(naiveBayes.yhat10!=val10$quality.good)/dim(val10)[1]
#naiveBayes.error.rate10



# overall naiveBayes train error rate
naiveBayes.error.rate.overall <- (naiveBayes.error.rate1 + naiveBayes.error.rate2 + naiveBayes.error.ra
naiveBayes.error.rate.overall
```

```
## [1] 0.2763636
```

```r
# test error rate of naiveBayes model for full test set
naiveBayes.fit.final <- naiveBayes(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity,train.whi
naiveBayes.yhat.final <- predict(naiveBayes.fit.final,test.white)
naiveBayes.error.rate.final <- sum(naiveBayes.yhat.final!=test.white$quality.good)/dim(test.white)[1]
naiveBayes.error.rate.final
```

```
## [1] 0.2755102
```

```r
# qda is best lets do it on the test
# test error rate of qda model for full test set
qda.fit.final <- qda(quality.good~alcohol+total.sulfur.dioxide+volatile.acidity+density,train.white)
qda.yhat.final <- predict(qda.fit.final,test.white)
qda.error.rate.final <- sum(qda.yhat.final$class!=test.white$quality.good)/dim(test.white)[1]
qda.error.rate.final
```

```
## [1] 0.244898
```

The Best of the Bayes Classifier methods was QDA, which had a test error rate of 0.2367347, but the best SVM model had a Test error rate of 0.1938776; so the SVM performed the best of the classification models