Advanced Business Analytics using R Project

QrlhmvVeroaev

835: 3645=

```r
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.5.2
```

```r
library(dplyr)
```

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

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

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

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

```r
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v readr   1.1.1
## v tibble  1.4.2     v purrr   0.2.5
## v tidyr   0.8.1     v stringr 1.3.1
## v ggplot2 3.0.0     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------- tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```
library(ggplot2)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.5.2
```

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tidyverse)
library(gains)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(mefa4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## Loading required package: pbapply
```

```
## Warning: package 'pbapply' was built under R version 3.5.2
```

```
## mefa4 0.3-5   2018-03-24
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(glmnet)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loaded glmnet 2.0-16
```

```
library(corrplot)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(arm)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: lme4
```

```
##
## arm (Version 1.10-1, built: 2018-4-12)
```

```
## Working directory is /Users/mihirraikar/Downloads
```

```
##
## Attaching package: 'arm'
```

```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

```
library(GGally)
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:arm':
##
##     logit
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(ggcorrplot)
```

```
wine <- fread("winequality-white.csv")
```

```
sum(duplicated(wine))
```

```
## [1] 0
```

```
dim(wine)
```

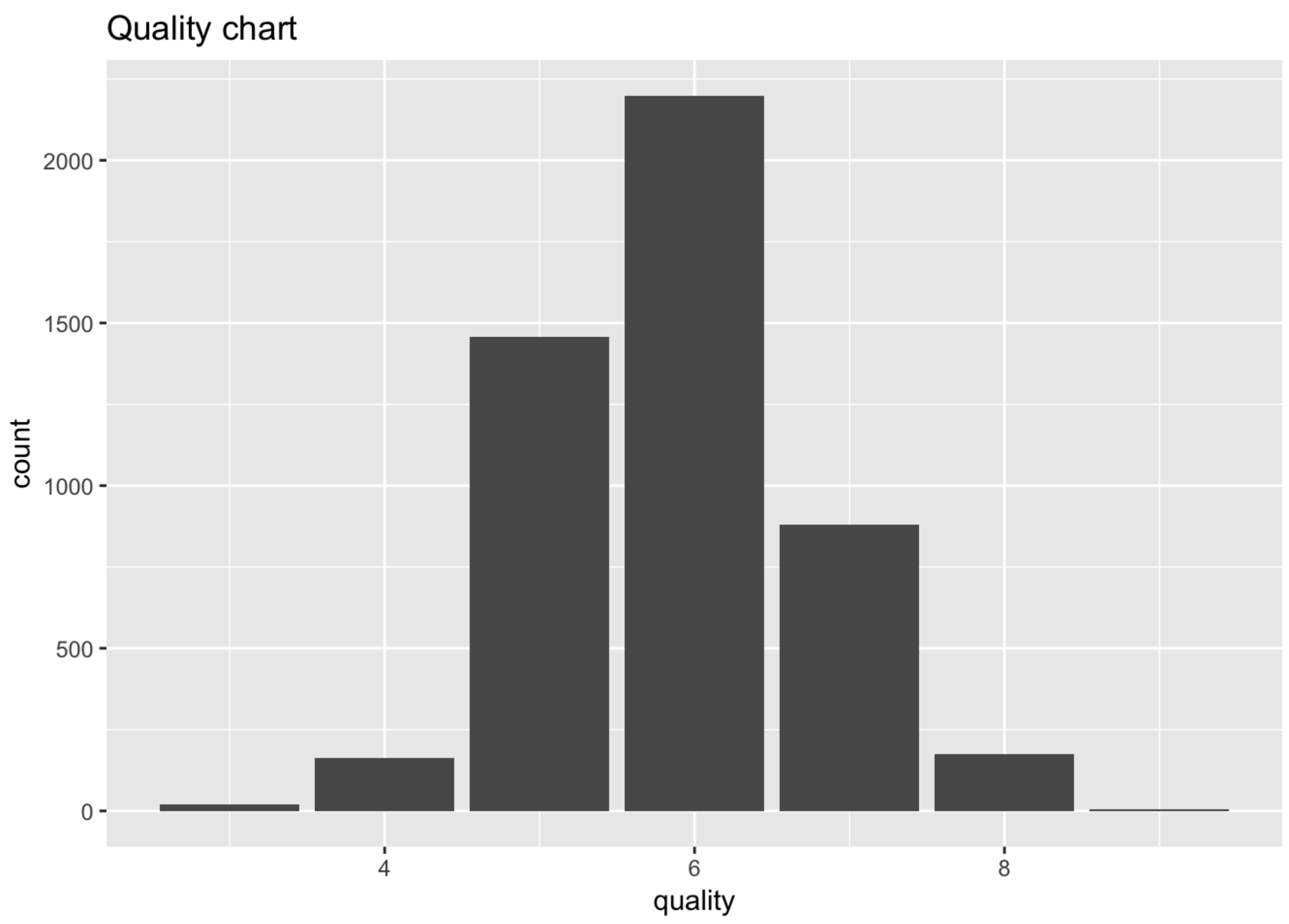
```
## [1] 4898   13
```

```
summary(wine)
```

```
##       Obs       fixed_acidity    volatile_acidity  citric_acid    
##  Min.   :   1   Min.   : 3.800   Min.   :0.0800   Min.   :0.0000  
##  1st Qu.:1225   1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700  
##  Median :2450   Median : 6.800   Median :0.2600   Median :0.3200  
##  Mean   :2450   Mean   : 6.855   Mean   :0.2782   Mean   :0.3342  
##  3rd Qu.:3674   3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900  
##  Max.   :4898   Max.   :14.200   Max.   :1.1000   Max.   :1.6600  
##  residual_sugar     chlorides       free_sulfur_dioxide
##  Min.   : 0.600   Min.   :0.00900   Min.   :  2.00     
##  1st Qu.: 1.700   1st Qu.:0.03600   1st Qu.: 23.00     
##  Median : 5.200   Median :0.04300   Median : 34.00     
##  Mean   : 6.391   Mean   :0.04577   Mean   : 35.31     
##  3rd Qu.: 9.900   3rd Qu.:0.05000   3rd Qu.: 46.00     
##  Max.   :65.800   Max.   :0.34600   Max.   :289.00     
##  total_sulfur_dioxide    density             pH          sulphates     
##  Min.   :  9.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200  
##  1st Qu.:108.0        1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100  
##  Median :134.0        Median :0.9937   Median :3.180   Median :0.4700  
##  Mean   :138.4        Mean   :0.9940   Mean   :3.188   Mean   :0.4898  
##  3rd Qu.:167.0        3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500  
##  Max.   :440.0        Max.   :1.0390   Max.   :3.820   Max.   :1.0800  
##     alcohol          quality     
##  Min.   : 8.00   Min.   :3.000  
##  1st Qu.: 9.50   1st Qu.:5.000  
##  Median :10.40   Median :6.000  
##  Mean   :10.51   Mean   :5.878  
##  3rd Qu.:11.40   3rd Qu.:6.000  
##  Max.   :14.20   Max.   :9.000
```

Our Dependent Variable is Quality, hence we will take a look at the distribution of quality

```
quality_plot <- ggplot(aes(quality), data = wine) + geom_bar() + ggtitle ("Quality ch
art")
quality_plot
```

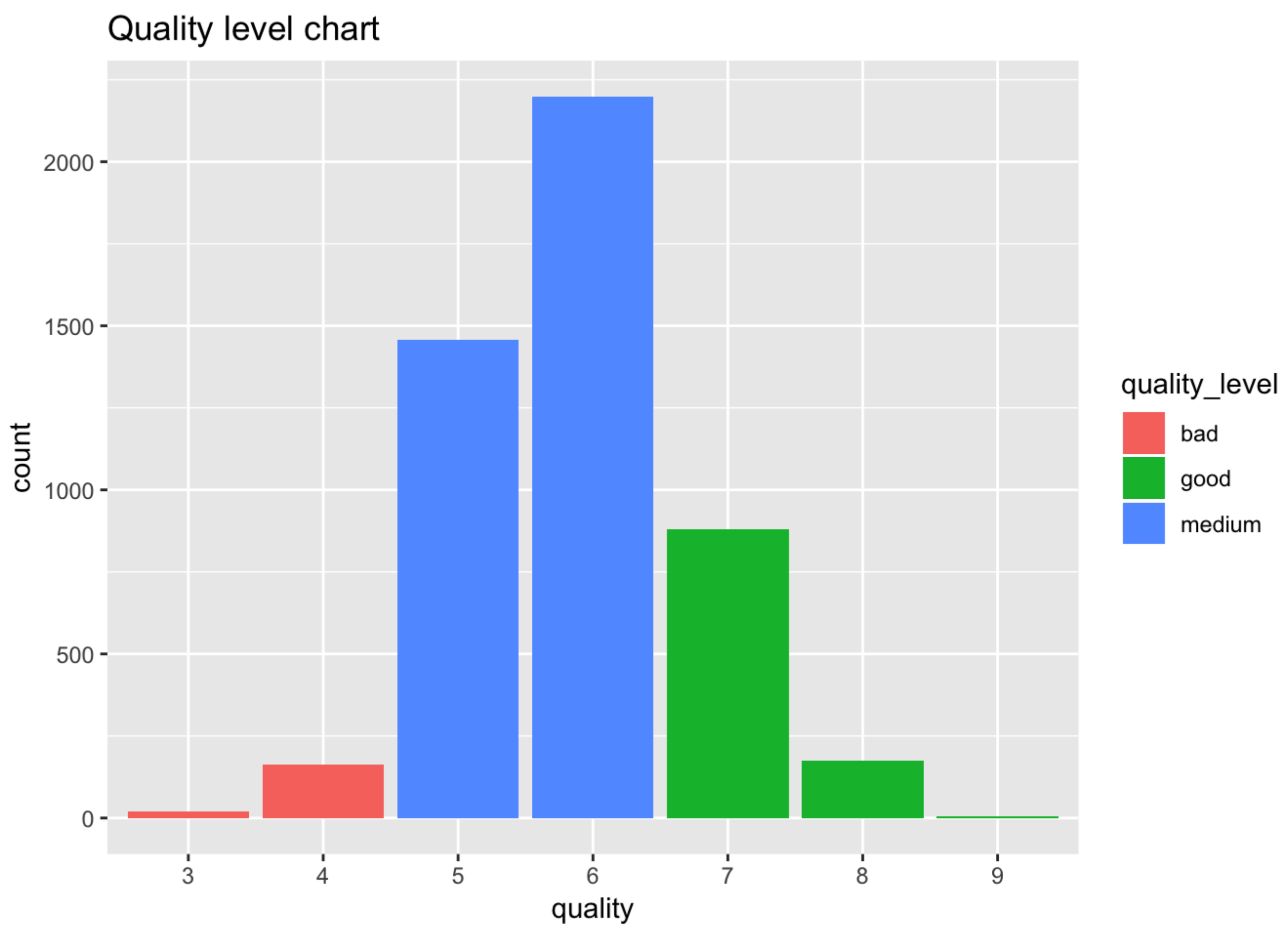


The range of Quality is from 3 to 9. We can bucket the data into good, medium and bad with reference to Quality

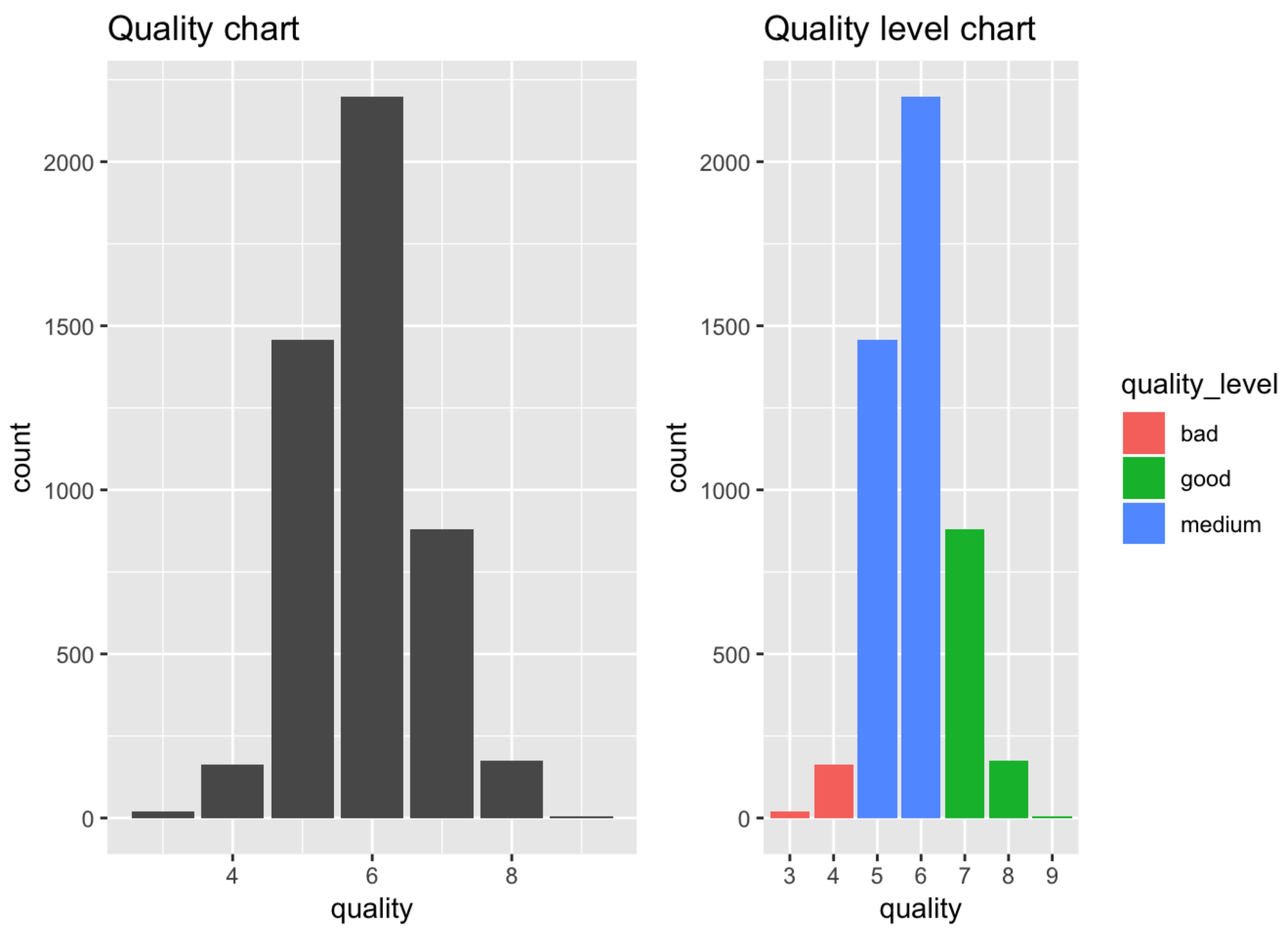
```
wine$quality_level <- ifelse(wine$quality >= 7, "good", NA)
wine$quality_level <- ifelse(wine$quality <= 6, "medium", wine$quality_level)
wine$quality_level <- ifelse(wine$quality <= 4, "bad", wine$quality_level)
wine$quality_level <- as.factor(wine$quality_level)
wine$quality <- as.factor(wine$quality)
```

Plot and Visualize Quality level

```
qualitylevel_plot <- ggplot(aes(quality,fill=quality_level),data=wine) + geom_bar() +
ggtitle ("Quality level chart")
qualitylevel_plot
```



```
qualitylevelcount_plot<-qplot(wine$quality_level) + xlab("quality level") + ggtitle(
"count of quality level")
grid.arrange(quality_plot,qualitylevel_plot,ncol=2)
```

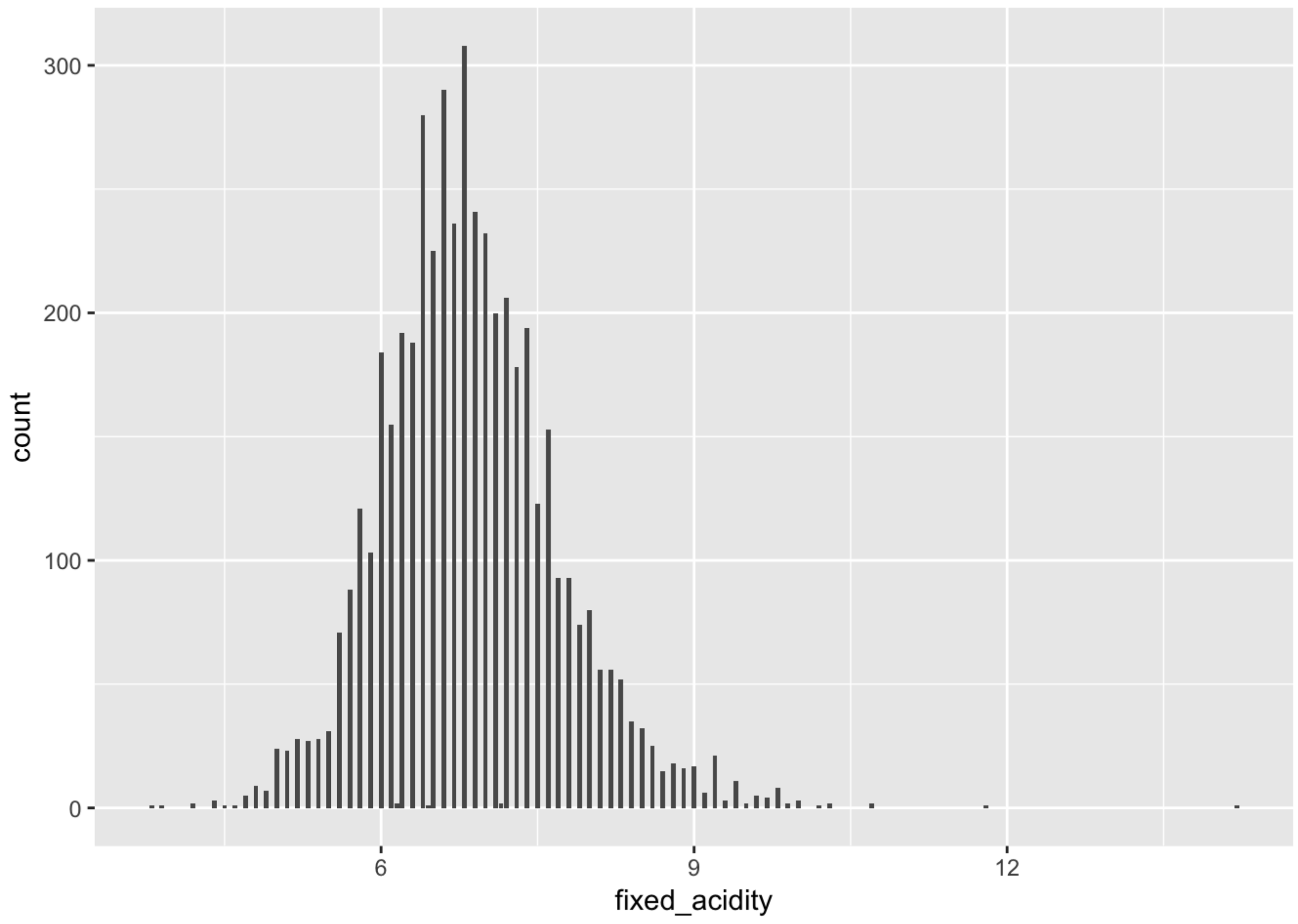


We will now explore various the variables

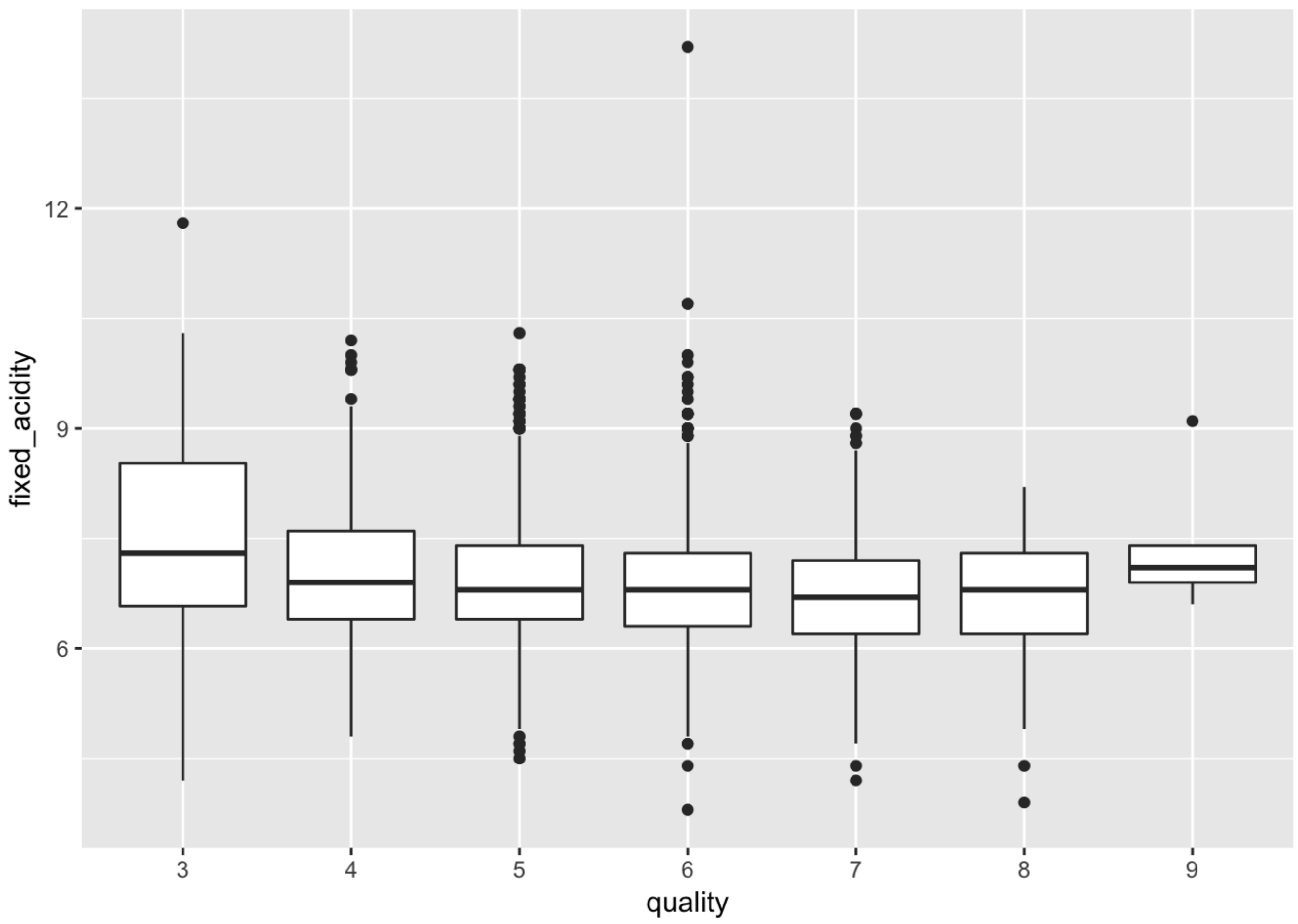
```
summary(wine$fixed_acidity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   3.800   6.300   6.800   6.855   7.300  14.200
```

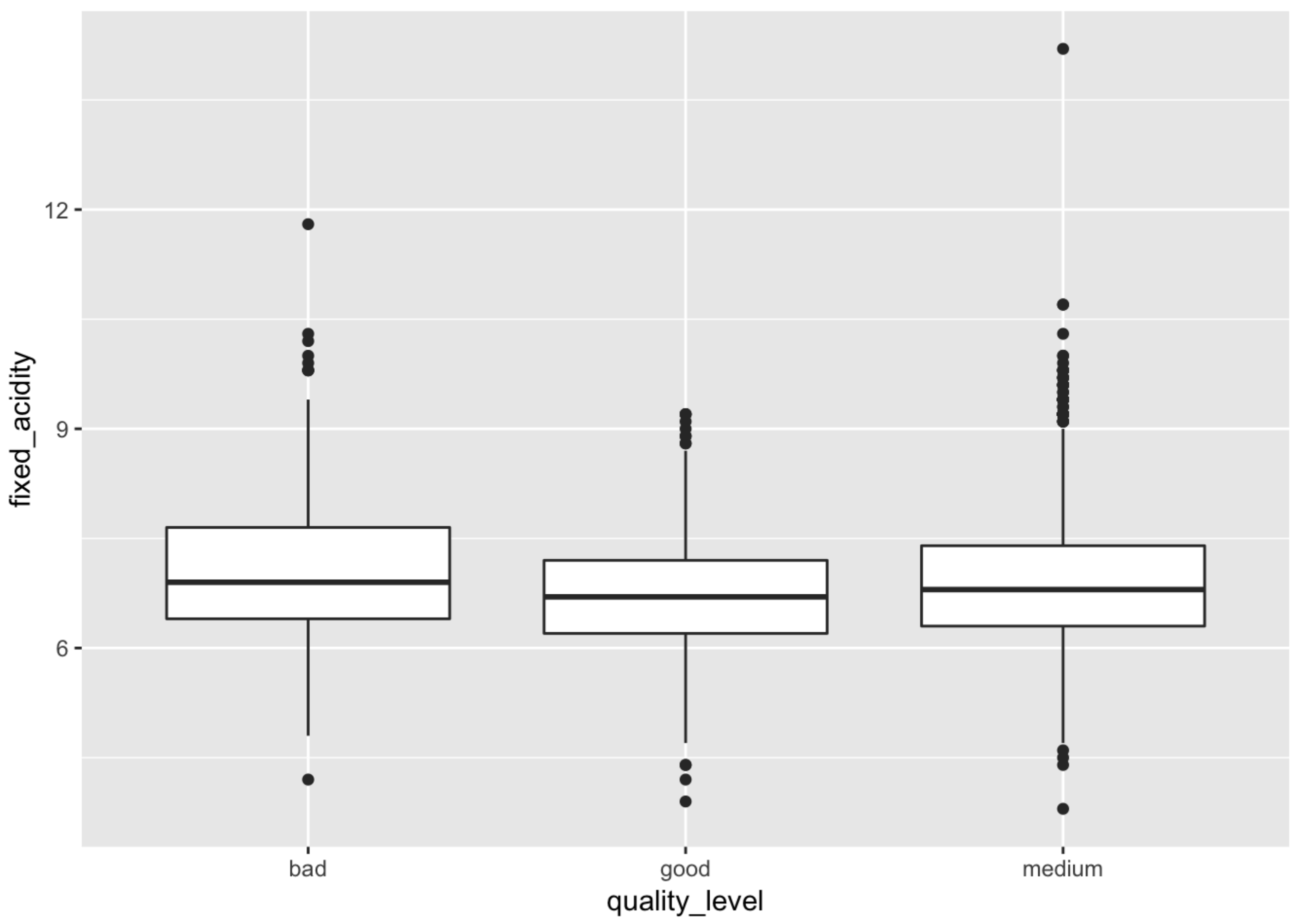
```
ggplot(aes(fixed_acidity), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=fixed_acidity), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=fixed_acidity), data = wine) + geom_boxplot()
```

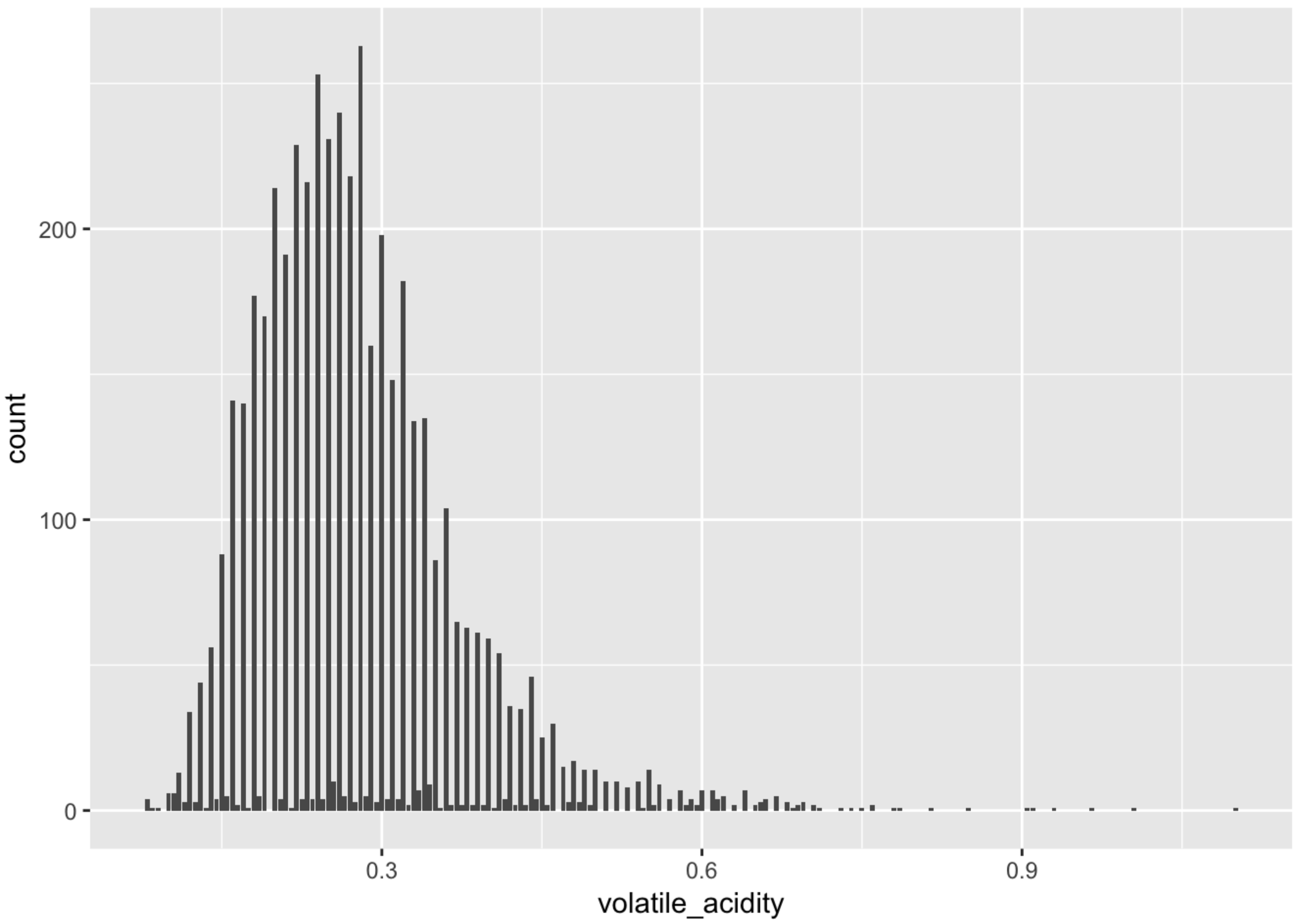


The data is positively skewed and also has some outliers The average fixed acidity tends to decrease with increase in quality but increases with quality=9 The observations of fixed acidity at it has in general a negative relationship with quality level

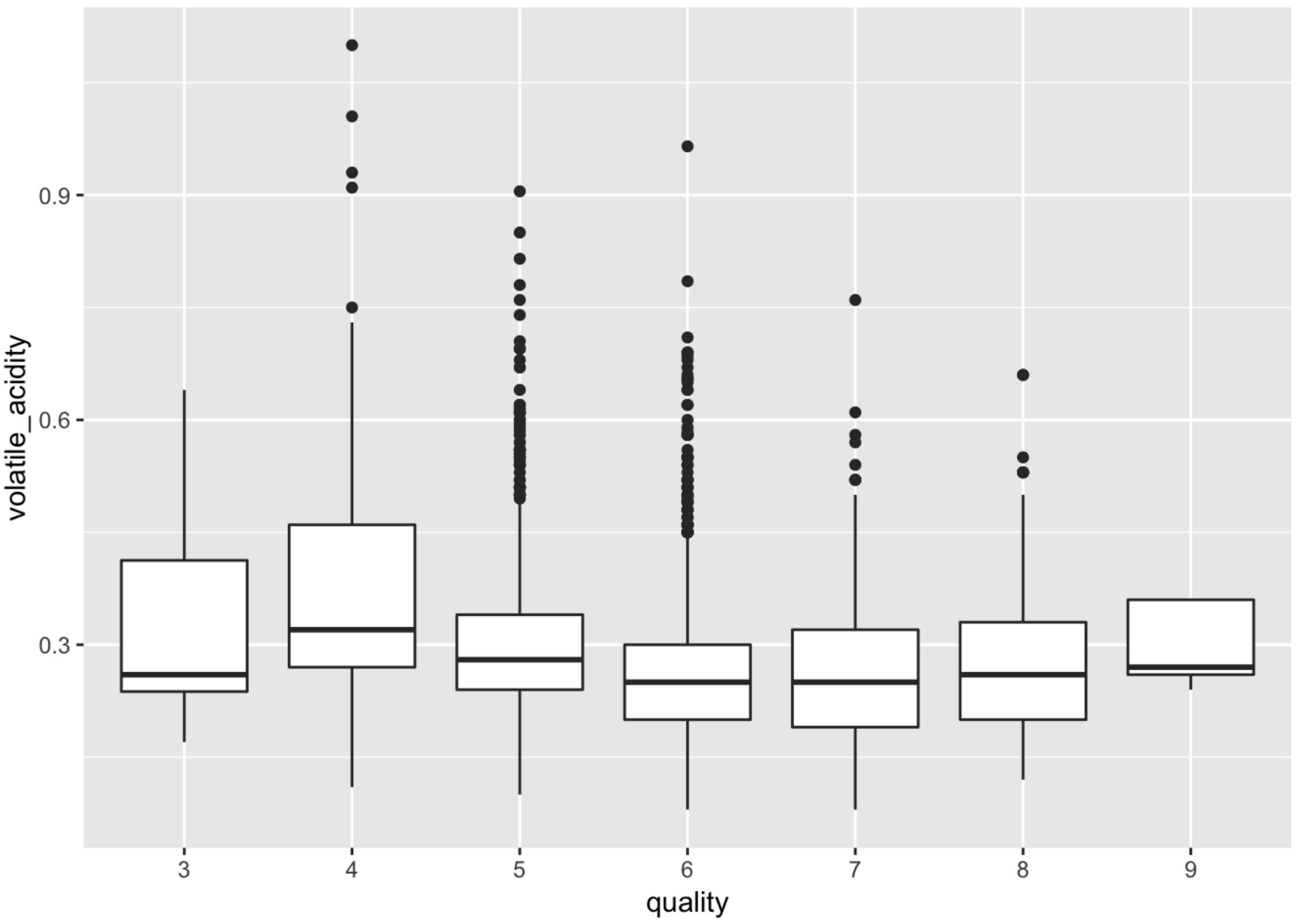
```
summary(wine$volatile_acidity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0800  0.2100  0.2600  0.2782  0.3200  1.1000
```

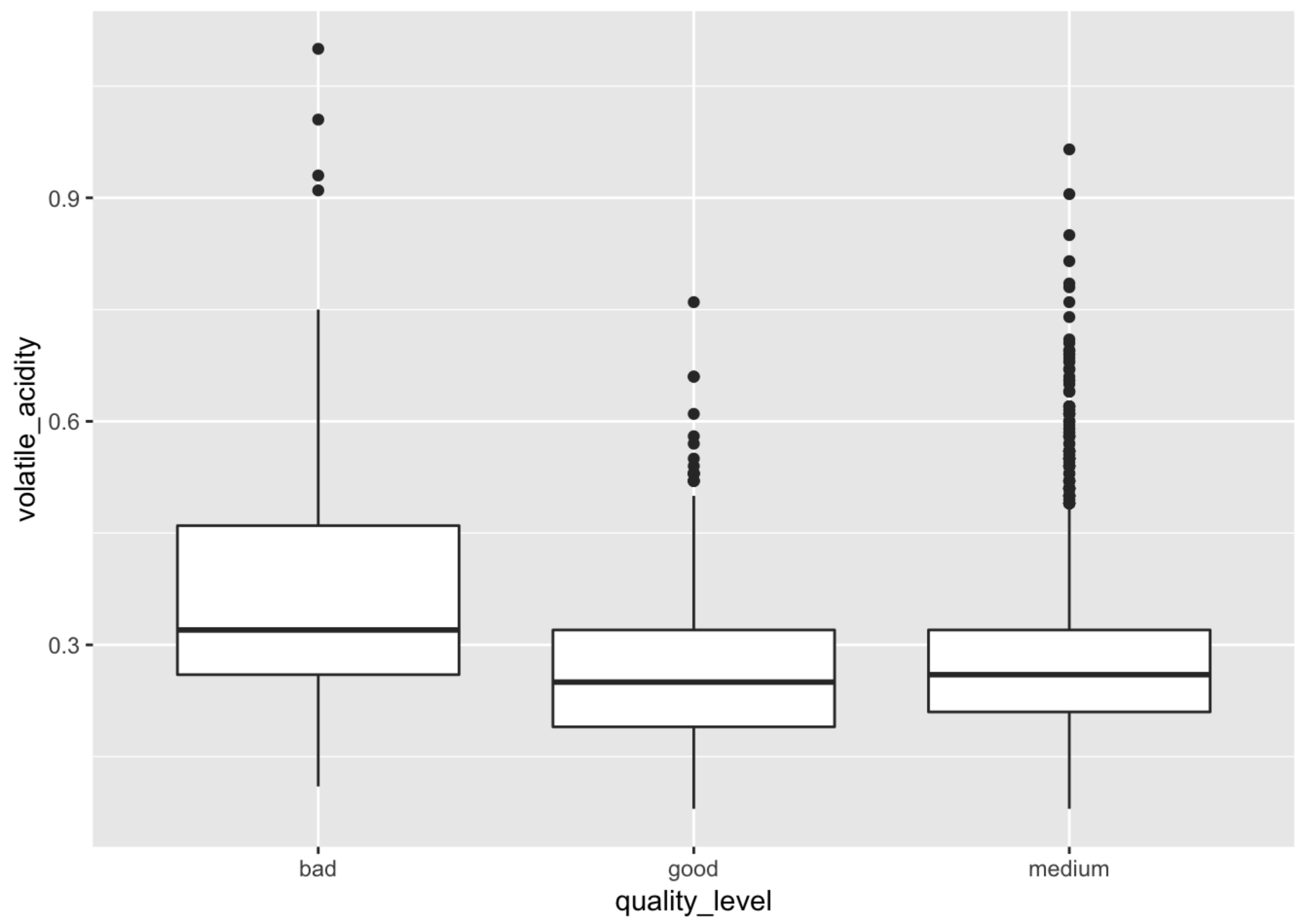
```
ggplot(aes(volatile_acidity), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=volatile_acidity), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=volatile_acidity), data = wine) + geom_boxplot()
```

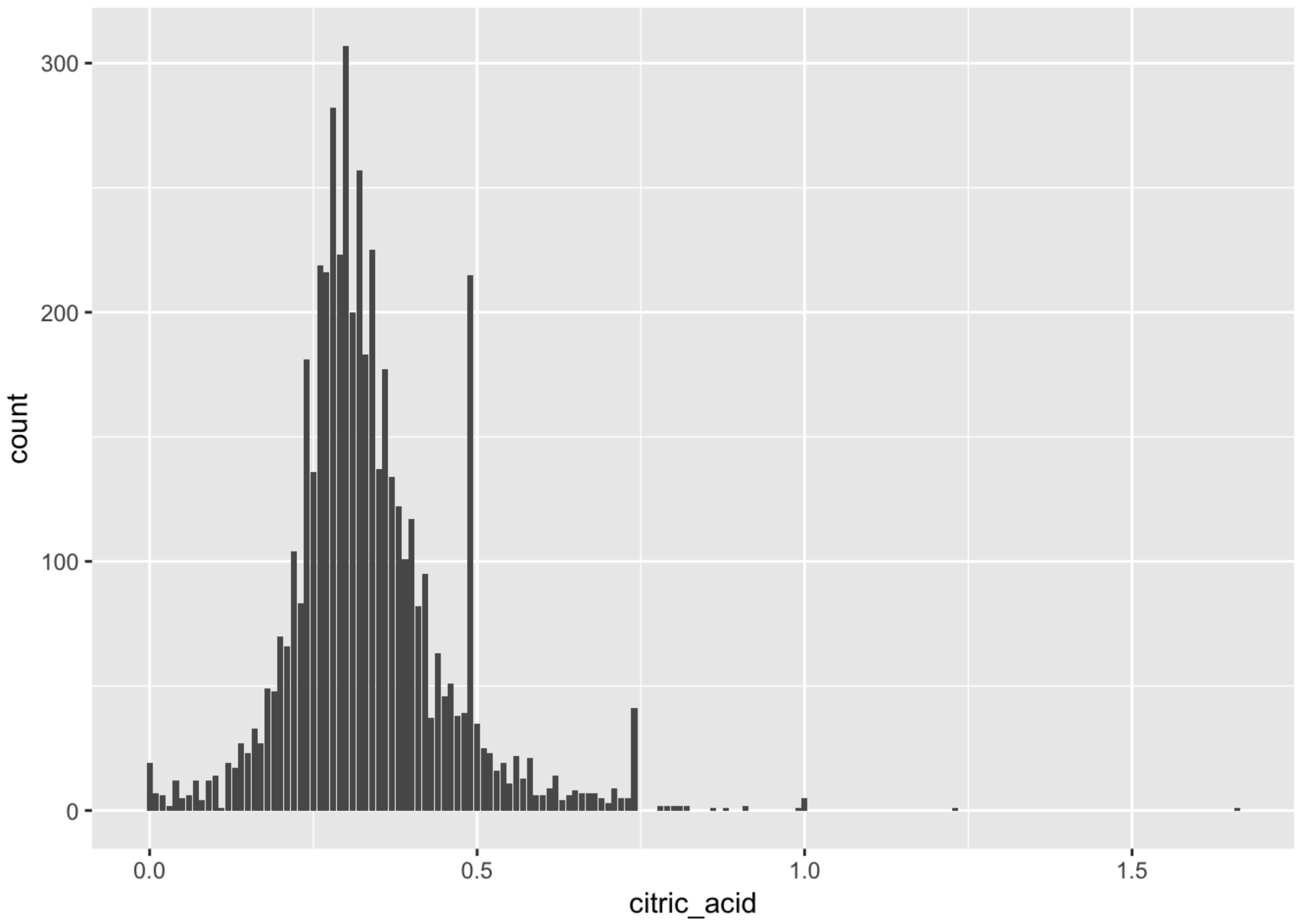


Similar to fixed acidity, volatile acidity is also positive skewed with outliers but the range is very small The relationship with quality is unclear as there is no trend The relationship of volatile acidity with quality level is negative

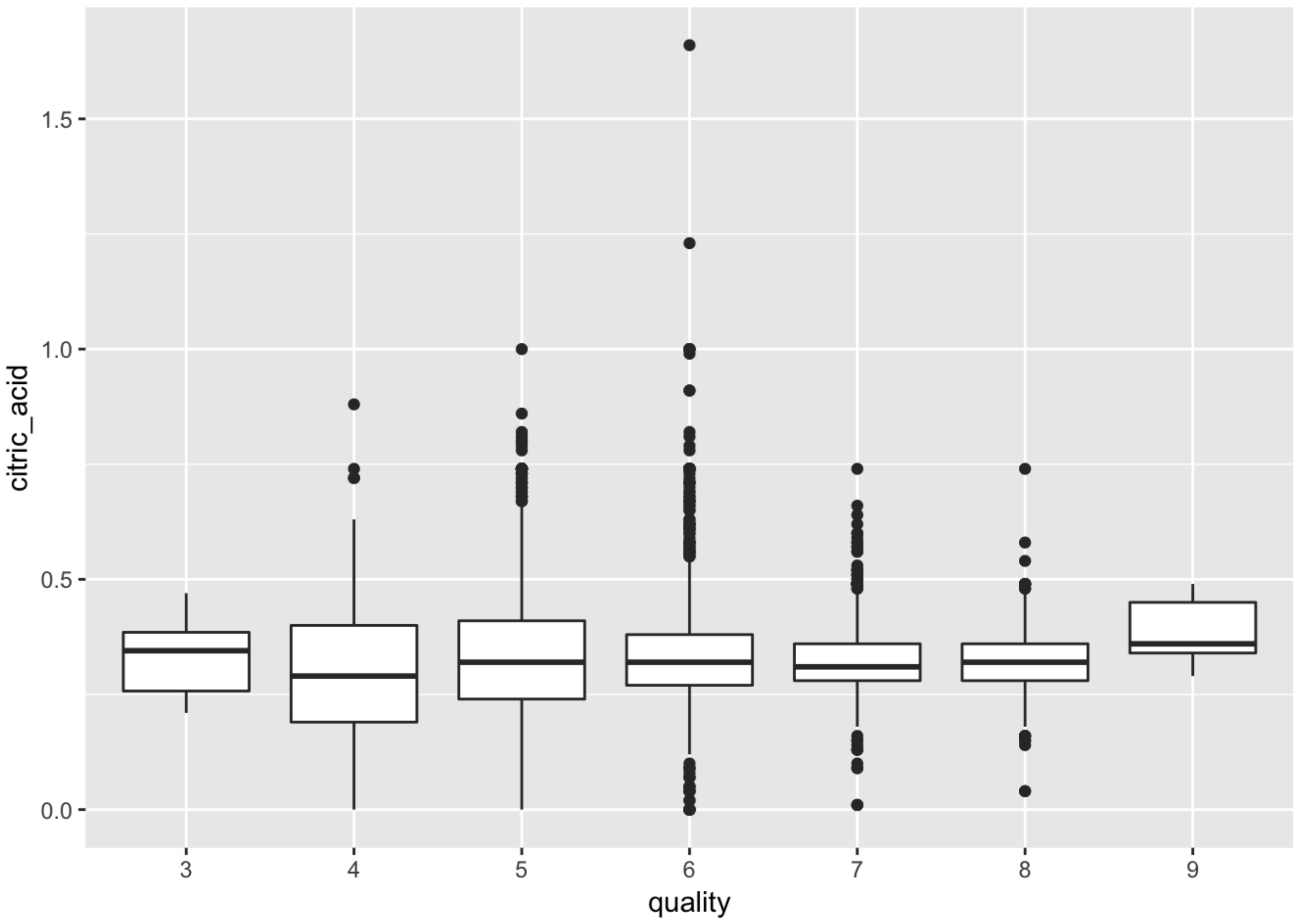
```
summary(wine$citric_acid)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.0000  0.2700  0.3200  0.3342  0.3900  1.6600
```

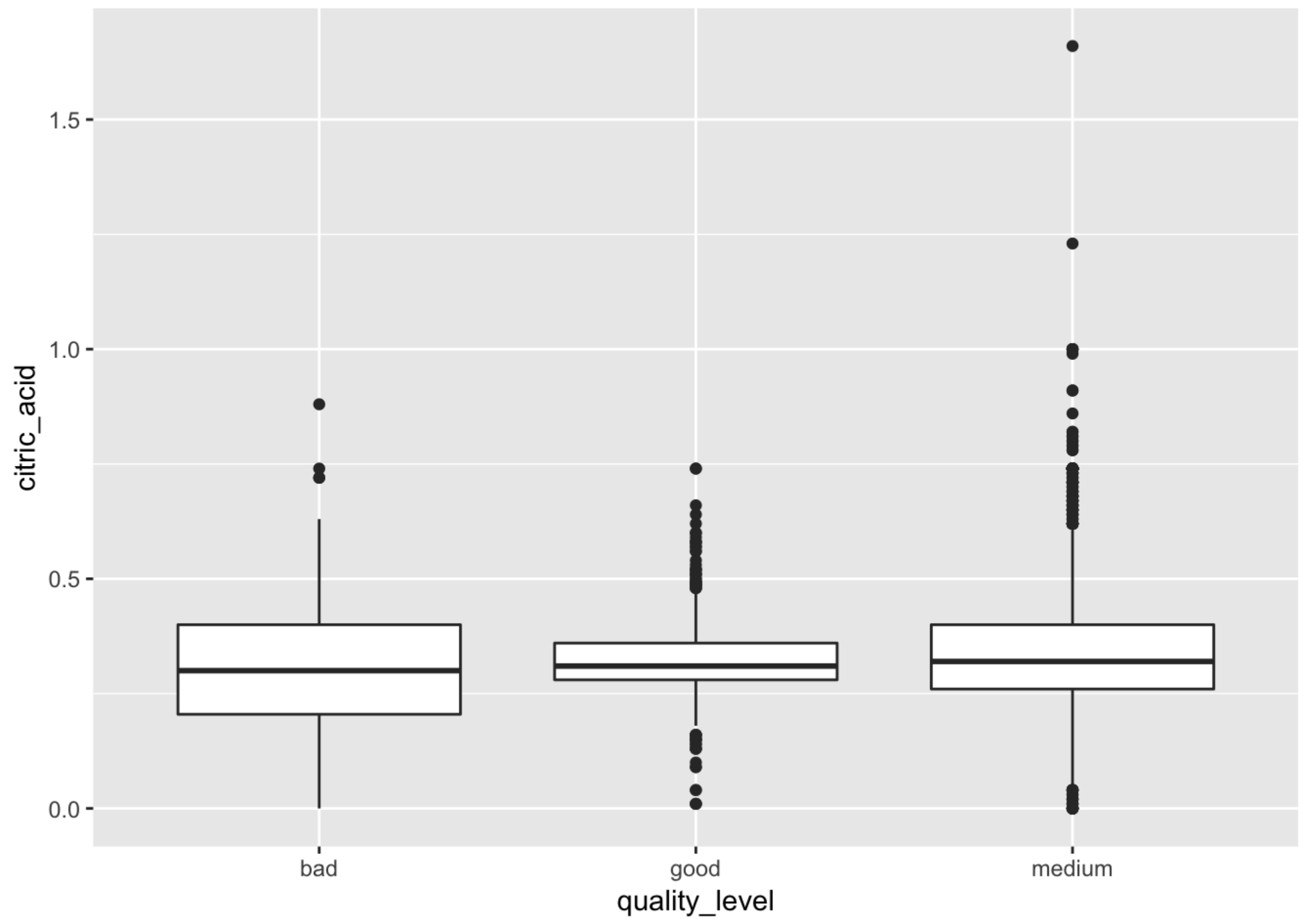
```
ggplot(aes(citric_acid), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=citric_acid), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=citric_acid), data = wine) + geom_boxplot()
```

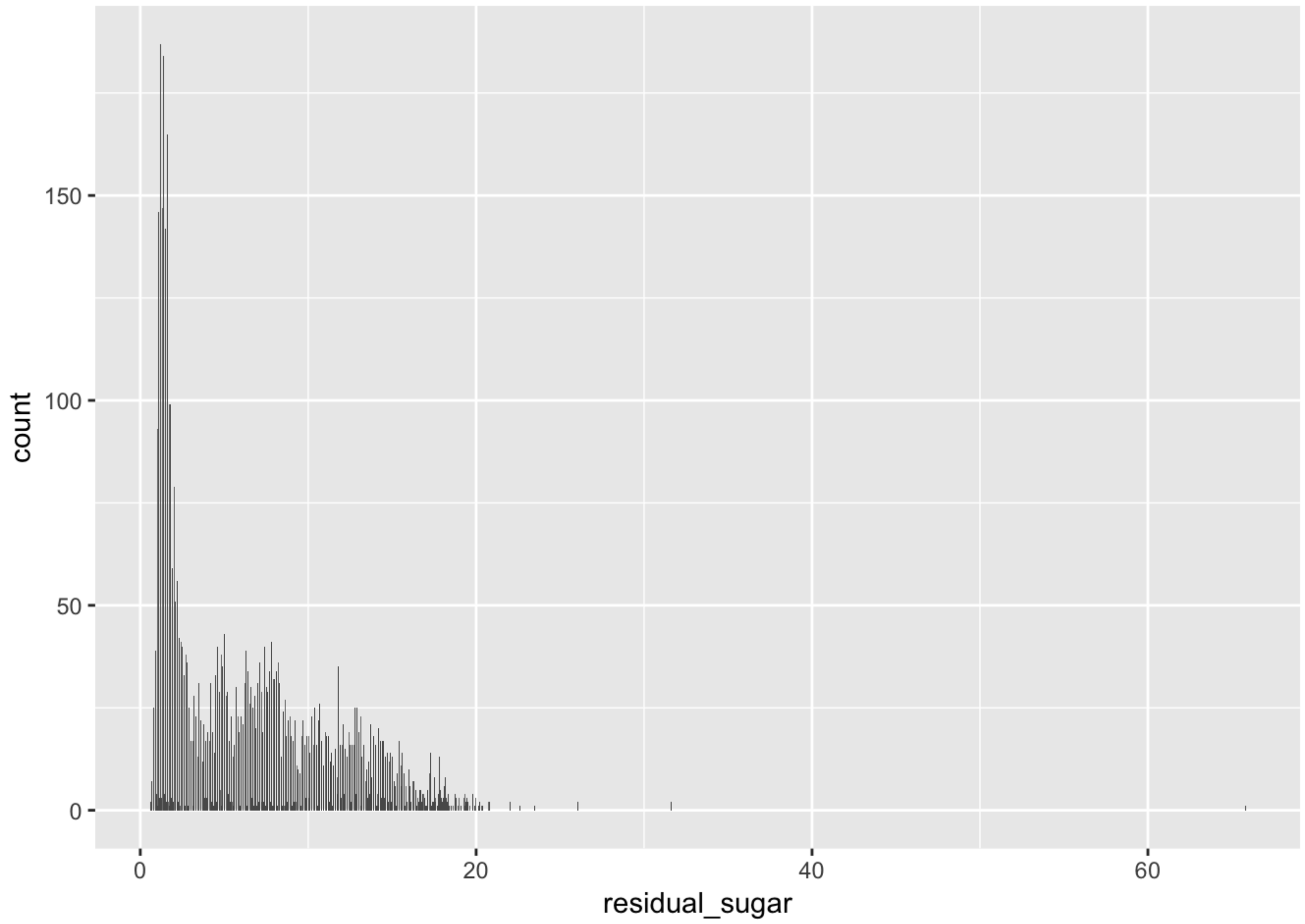


The distribution of citric acid is similar to normal but has 2 unusual peaks and a few outliers. The peaks are causing the data to not have a general trend

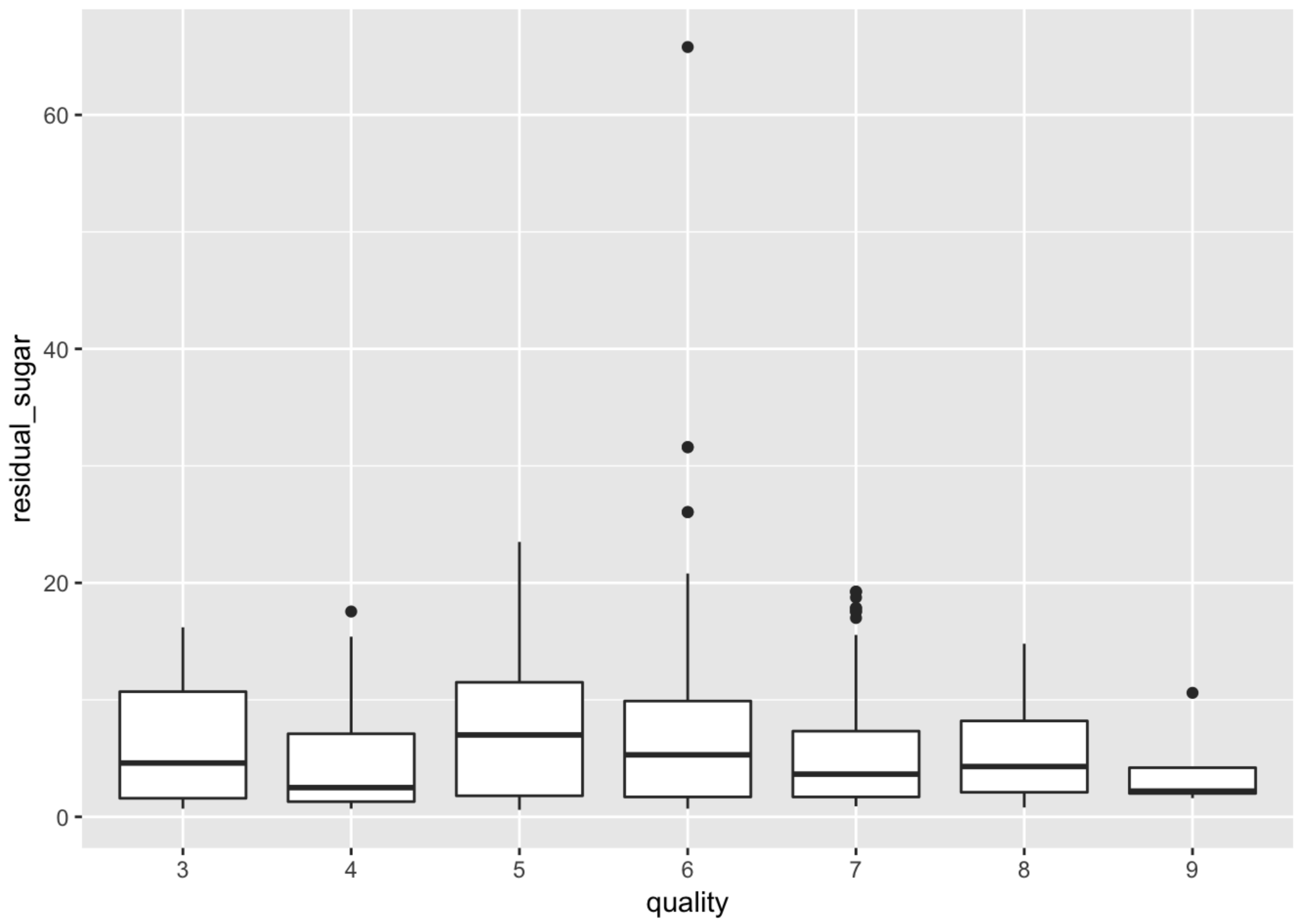
```
summary(wine$residual_sugar)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.600   1.700   5.200   6.391   9.900  65.800
```

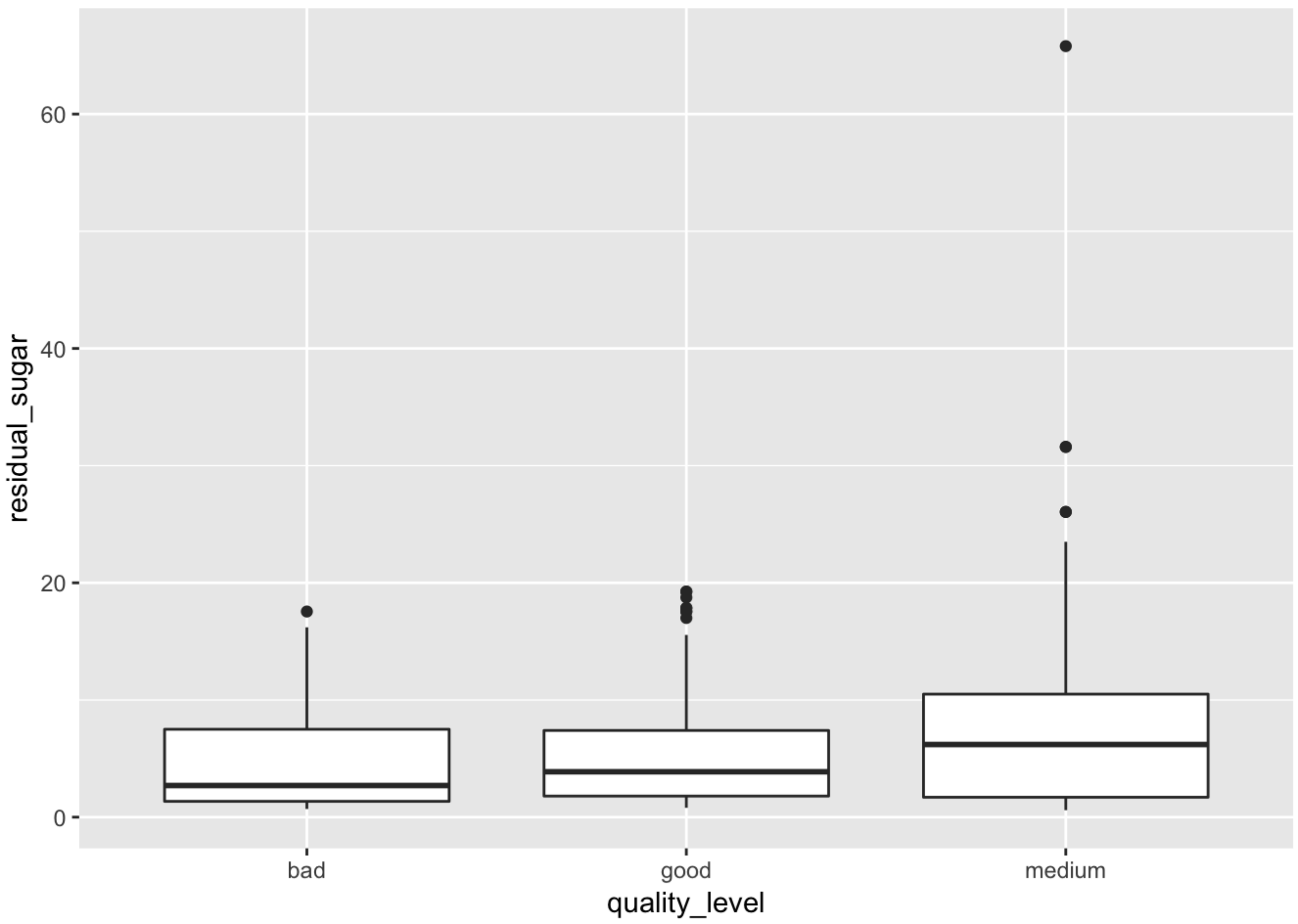
```
ggplot(aes(residual_sugar), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=residual_sugar), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=residual_sugar), data = wine) + geom_boxplot()
```

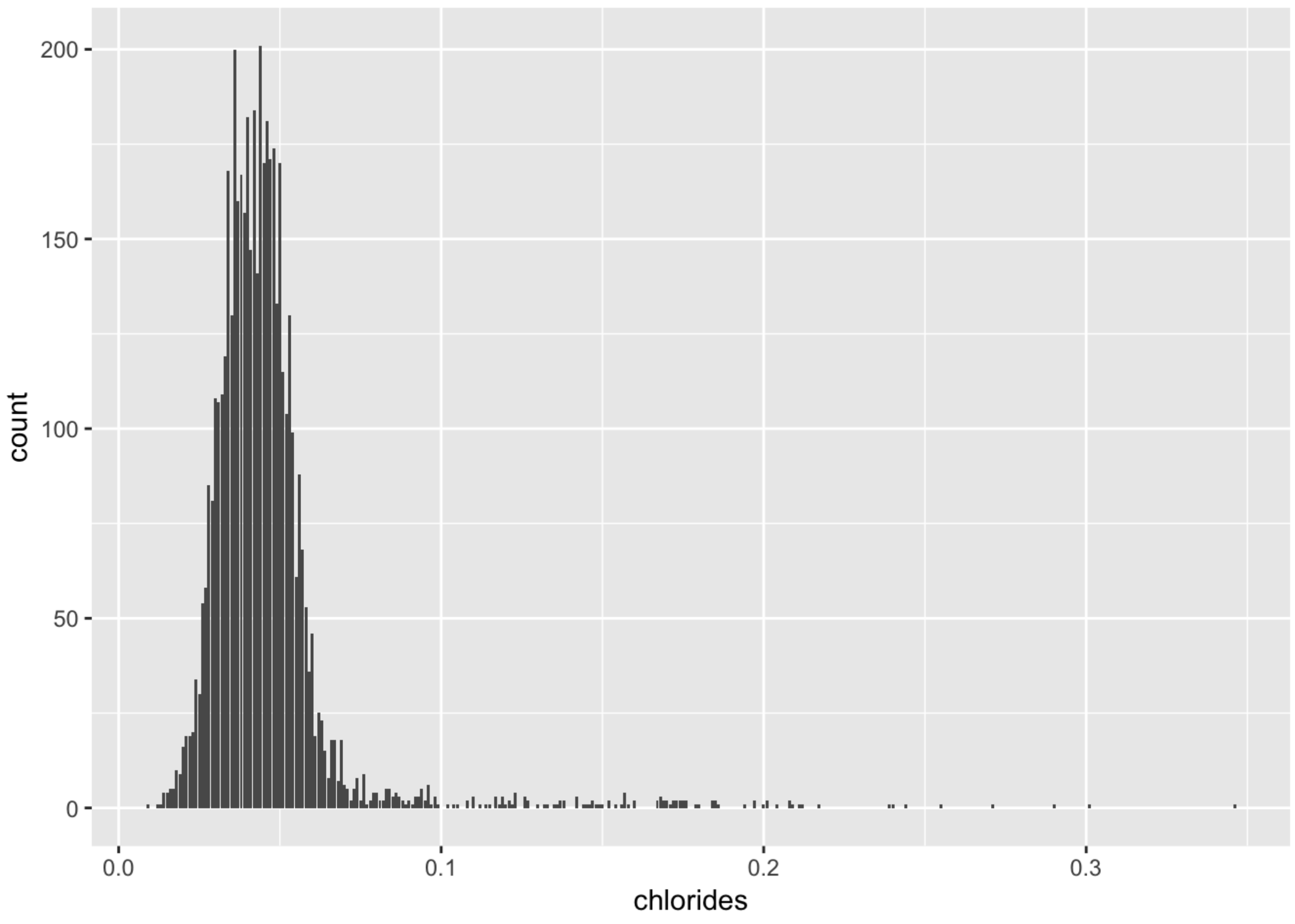


The data is positively skewed. It looks like most of the wines have very low residual sugars It looks like residual sugar is low in bad wines but comaritively high in medium and again comparatively lower in good. So generally, residual sugar in wine is good, but really good wines dont have it as much as average wines

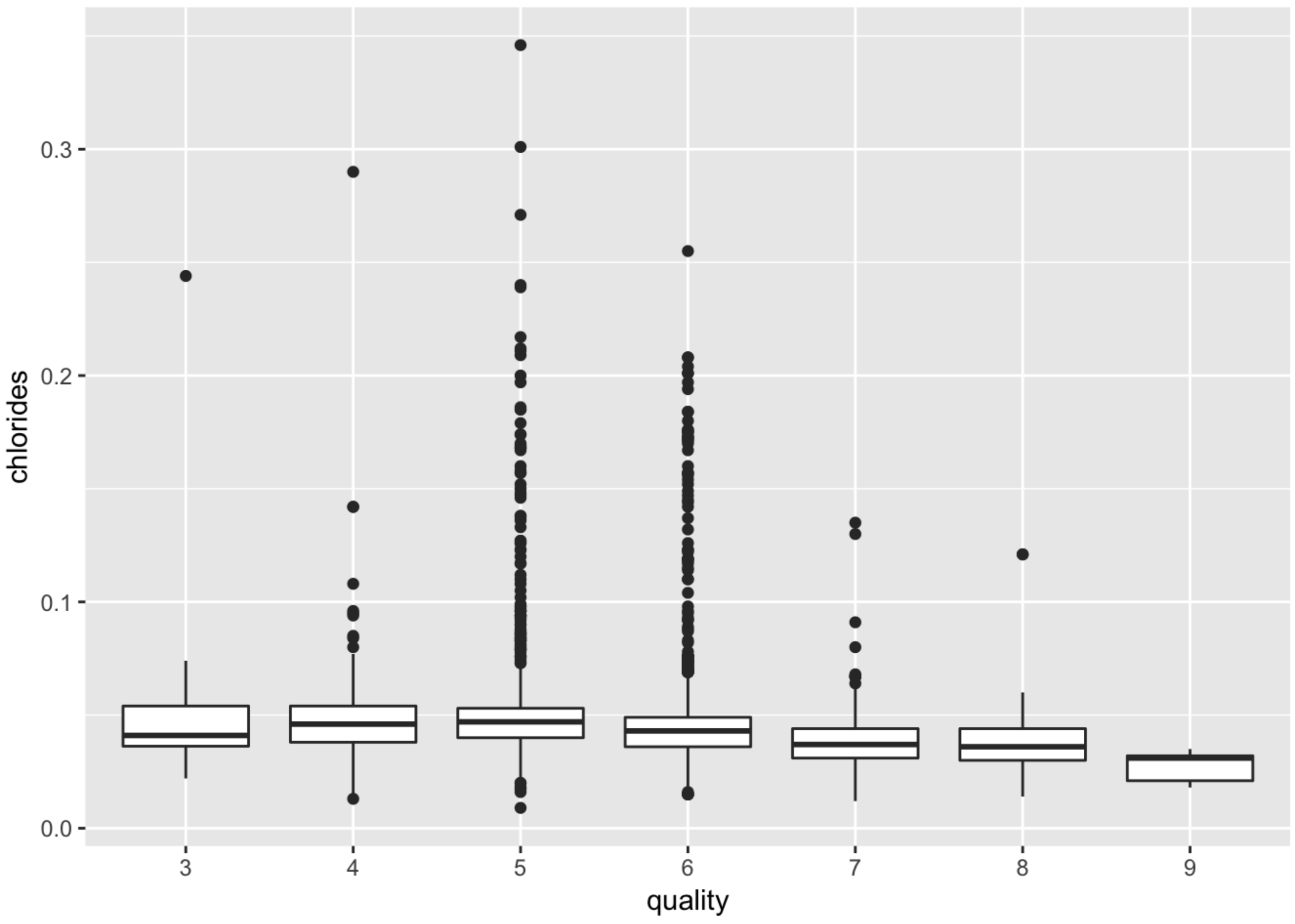
```
summary(wine$chlorides)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## 0.00900 0.03600 0.04300 0.04577 0.05000 0.34600
```

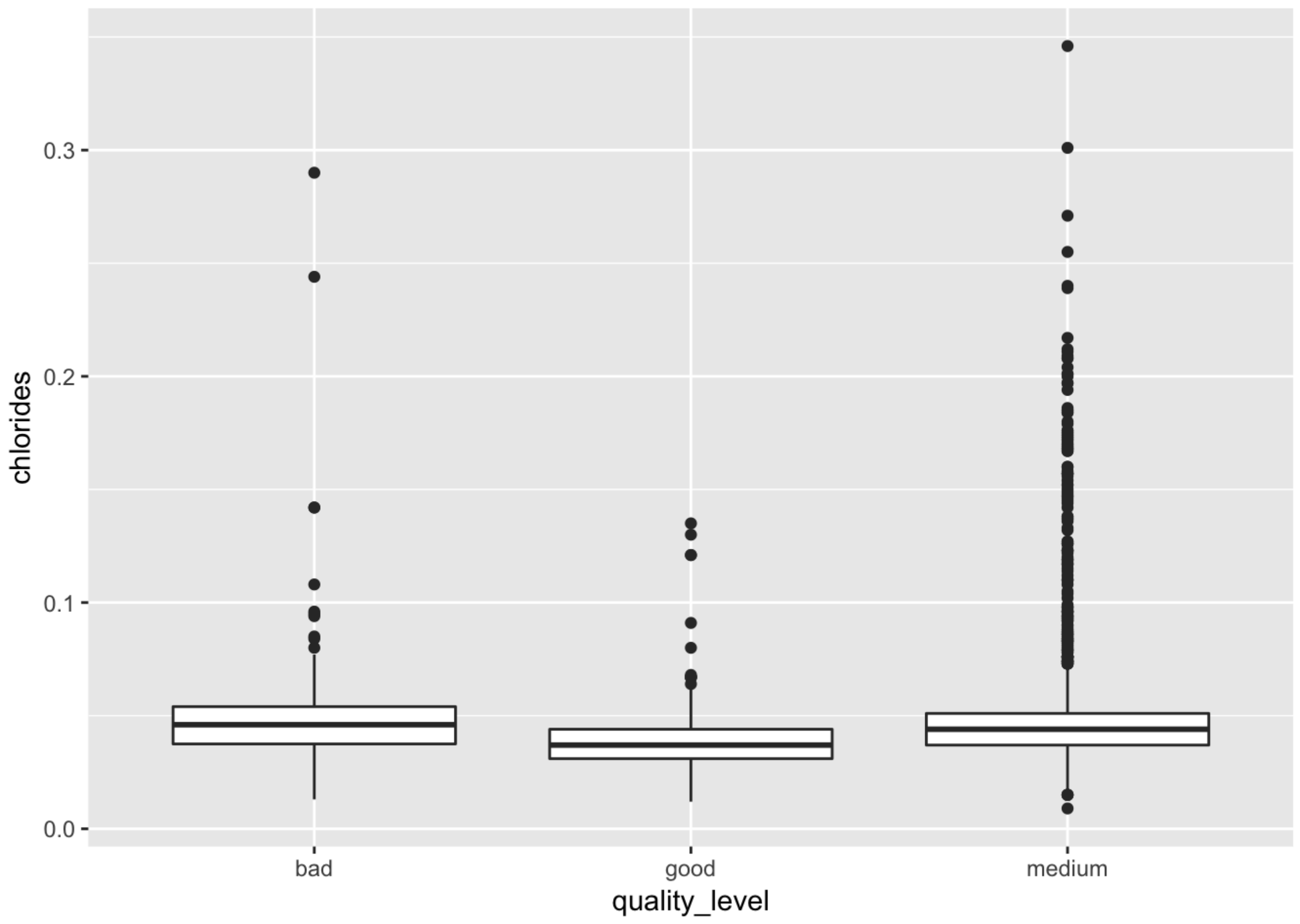
```
ggplot(aes(chlorides), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=chlorides), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=chlorides), data = wine) + geom_boxplot()
```

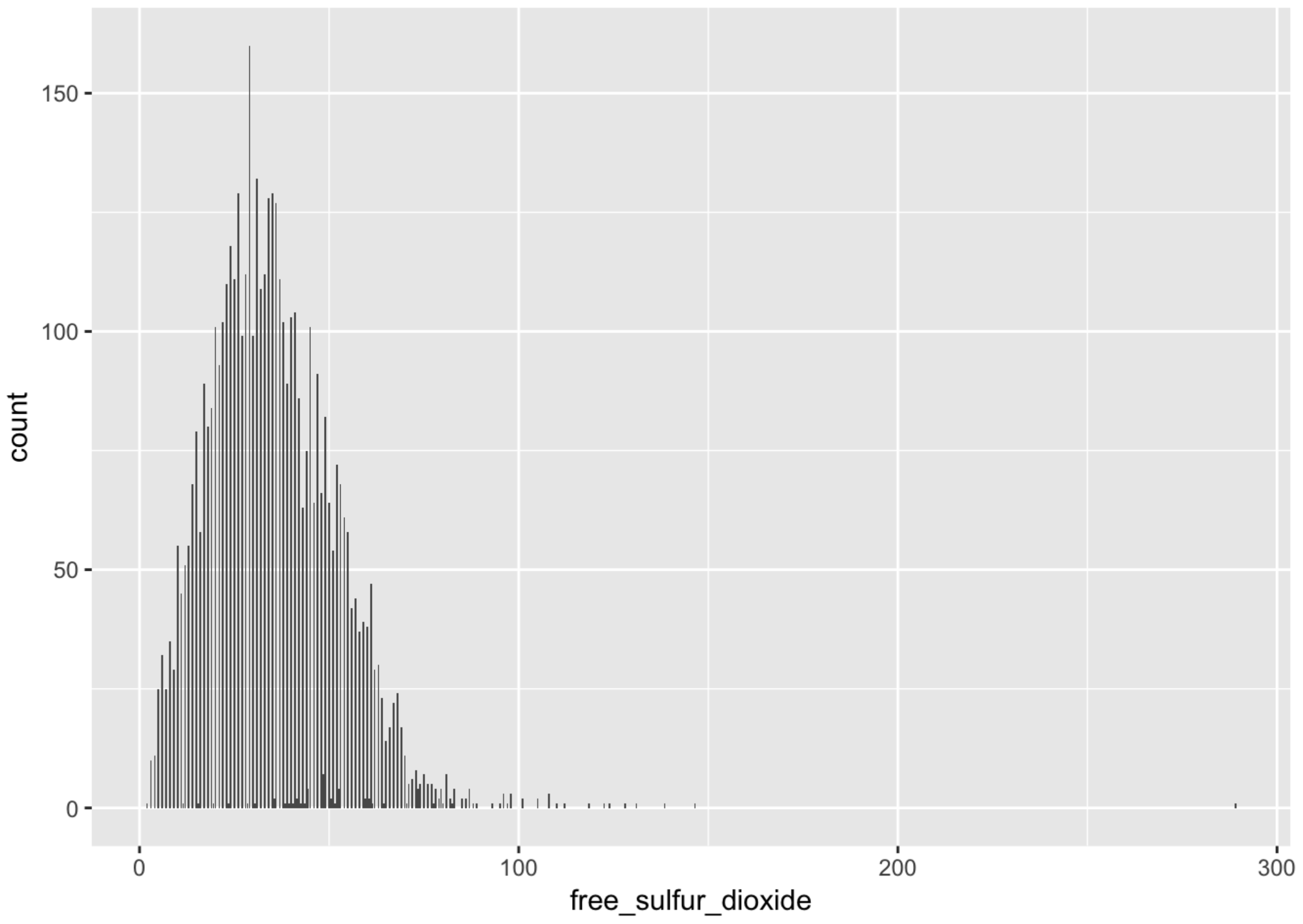


Chlorides are also positively skewed. The max value is lot higher than mean. Chlorides also have a negative relationship with quality level but not very significant

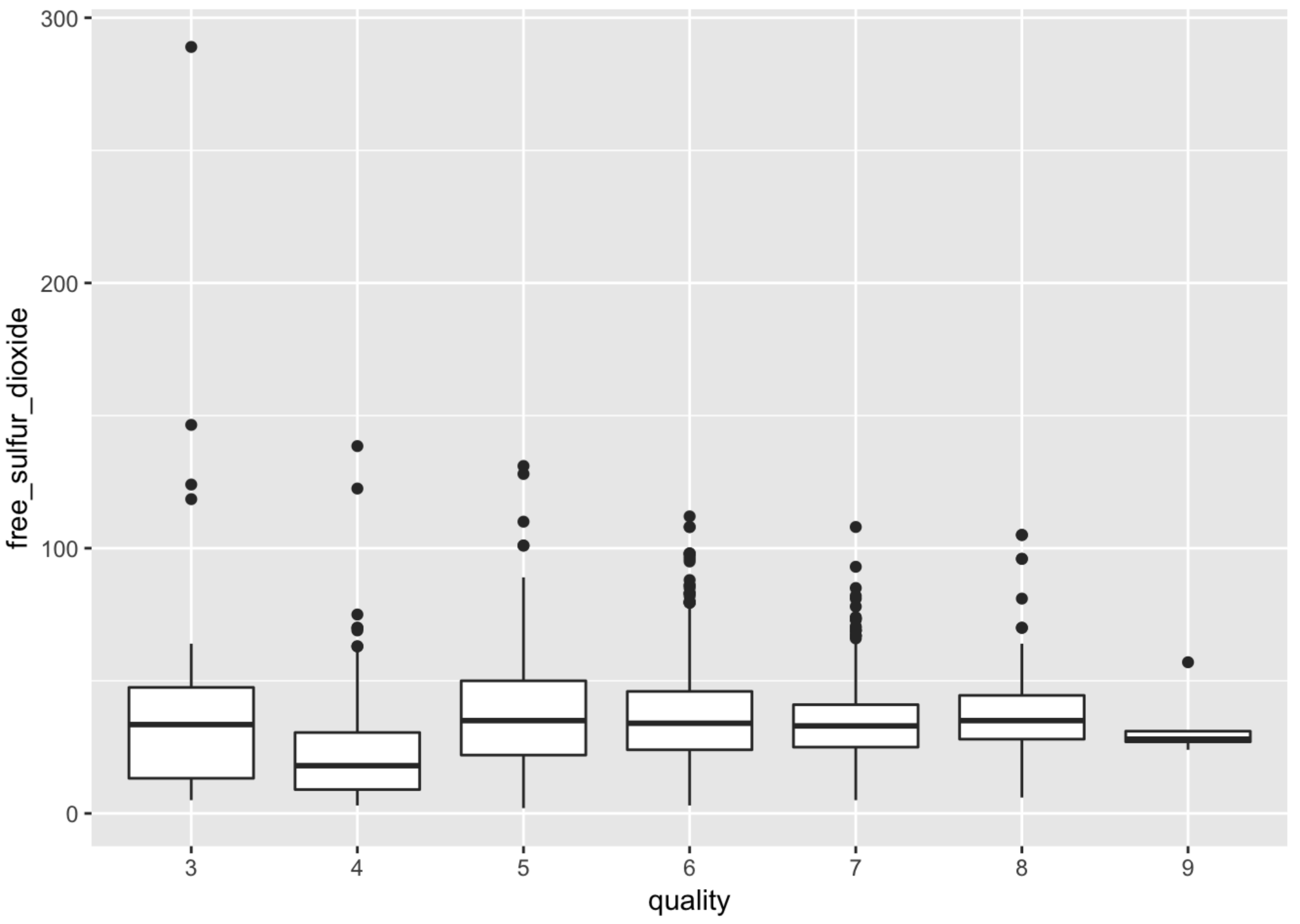
```
summary(wine$free_sulfur_dioxide)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   23.00   34.00   35.31   46.00  289.00
```

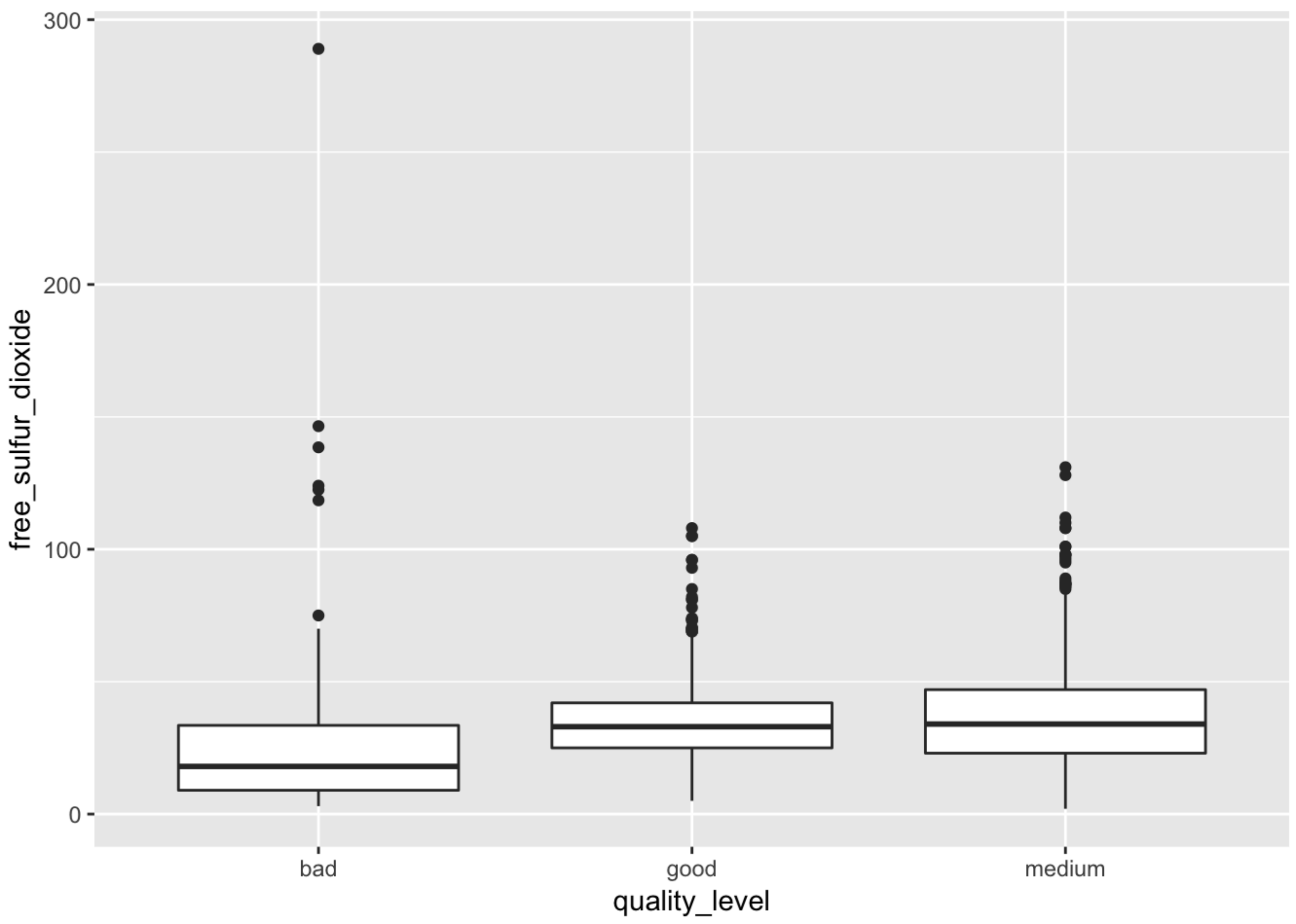
```
ggplot(aes(free_sulfur_dioxide), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=free_sulfur_dioxide), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=free_sulfur_dioxide), data = wine) + geom_boxplot()
```

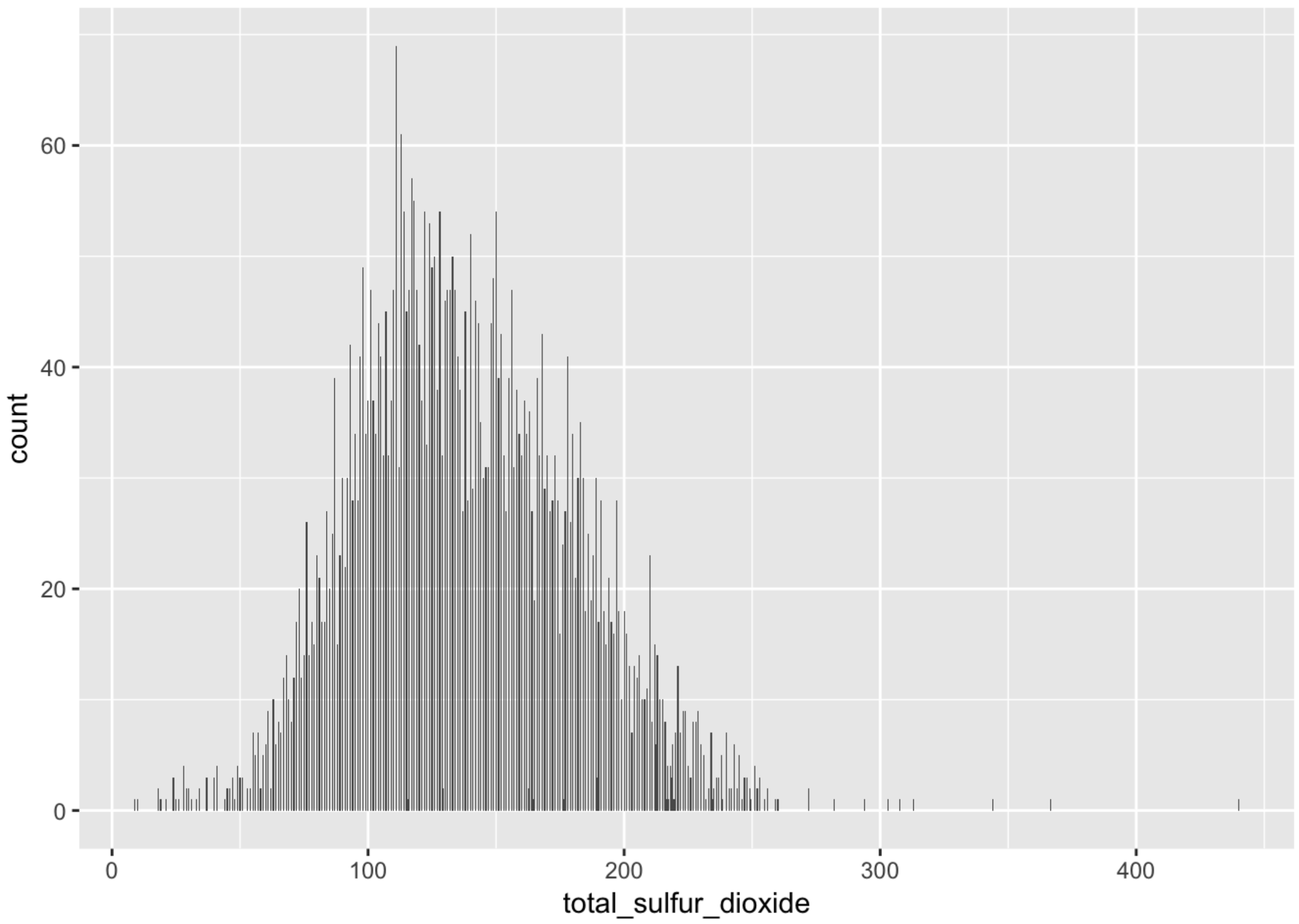


There is a huge outlier(289) whereas the rest of the data is below 150. Apart from the outlier also, the data is a little positive skewed. There is a positive relationship with quality but is not that significant with good quality

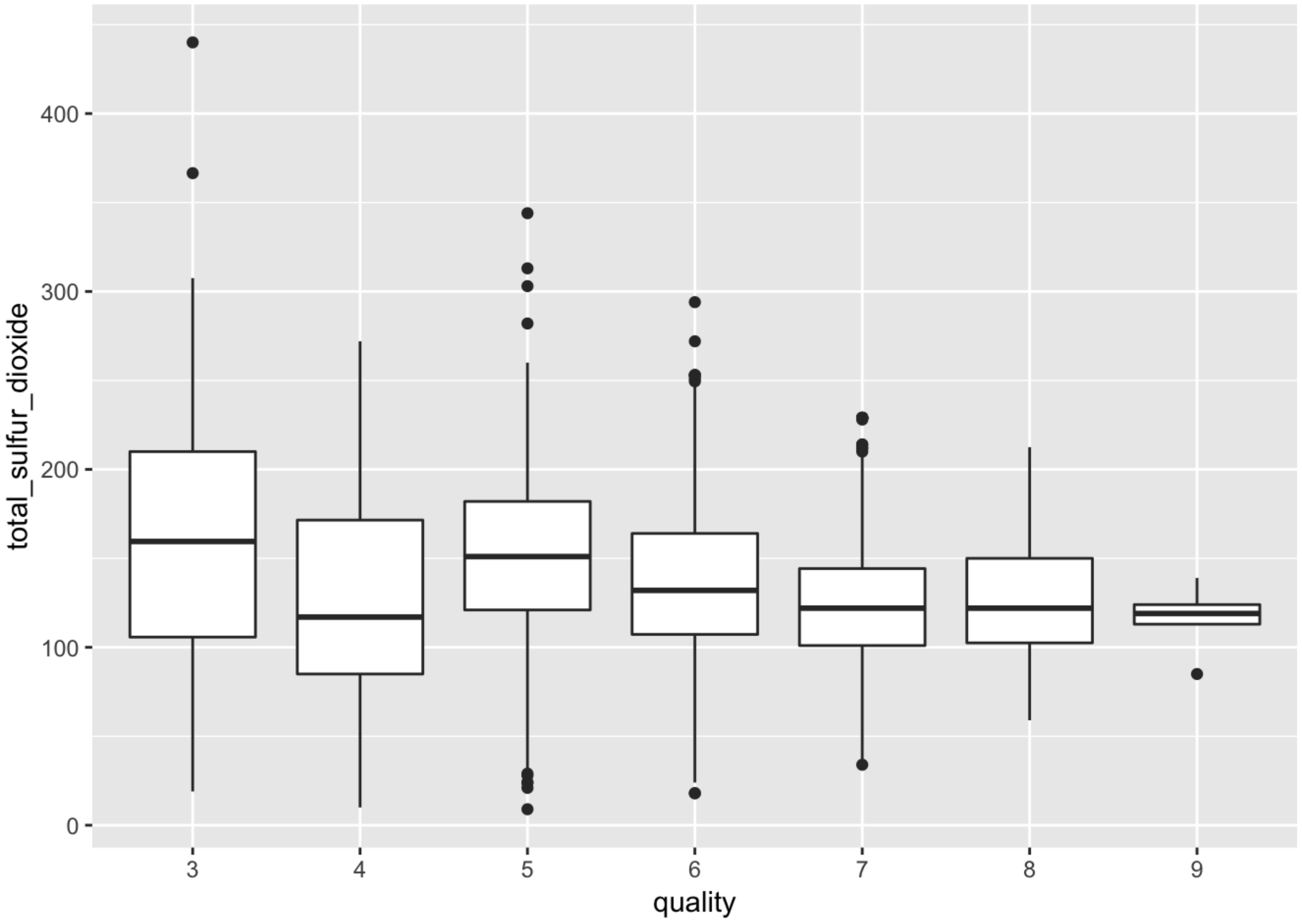
```
summary(wine$total_sulfur_dioxide)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     9.0   108.0   134.0   138.4   167.0   440.0
```

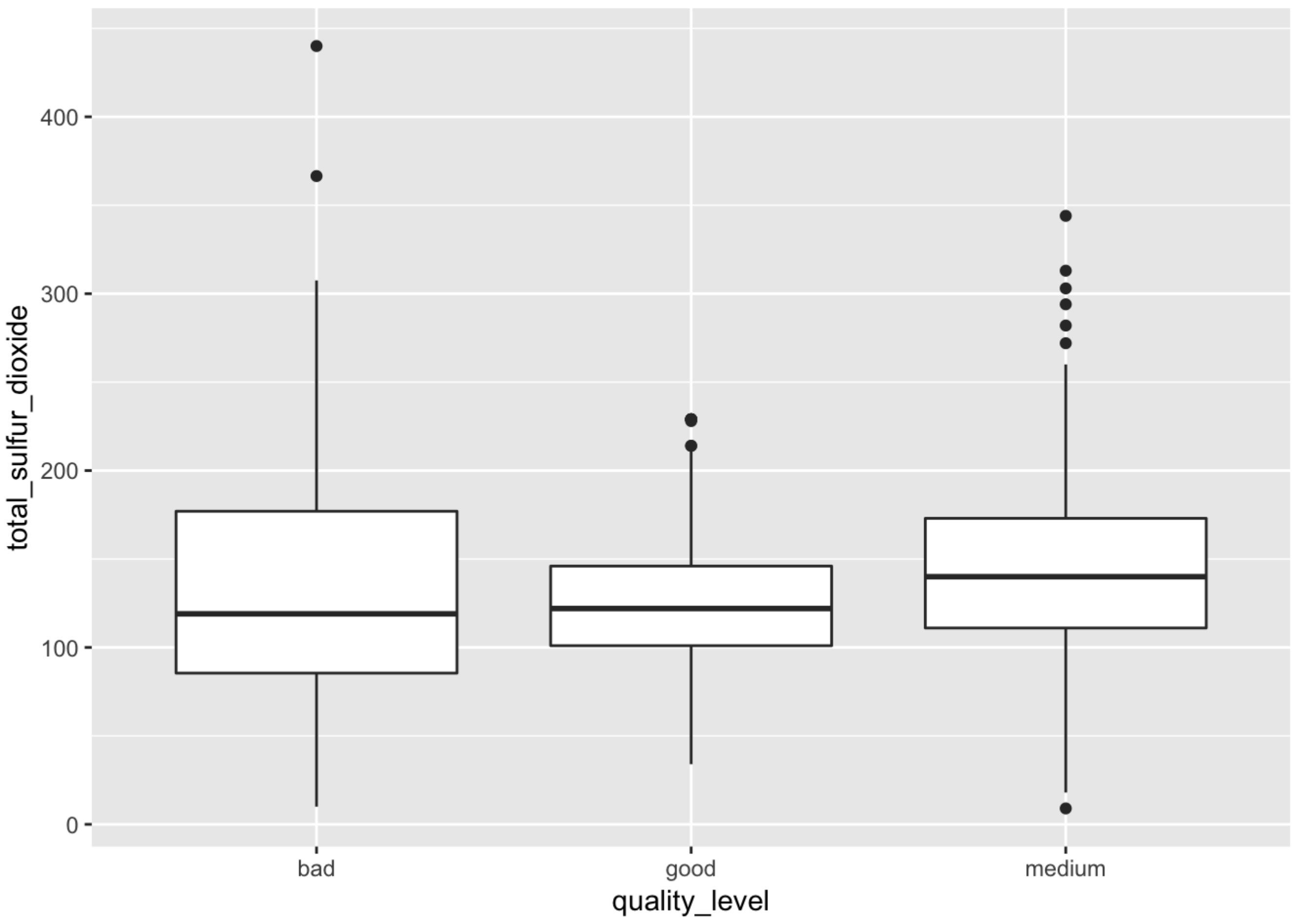
```
ggplot(aes(total_sulfur_dioxide), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=total_sulfur_dioxide), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=total_sulfur_dioxide), data = wine) + geom_boxplot()
```

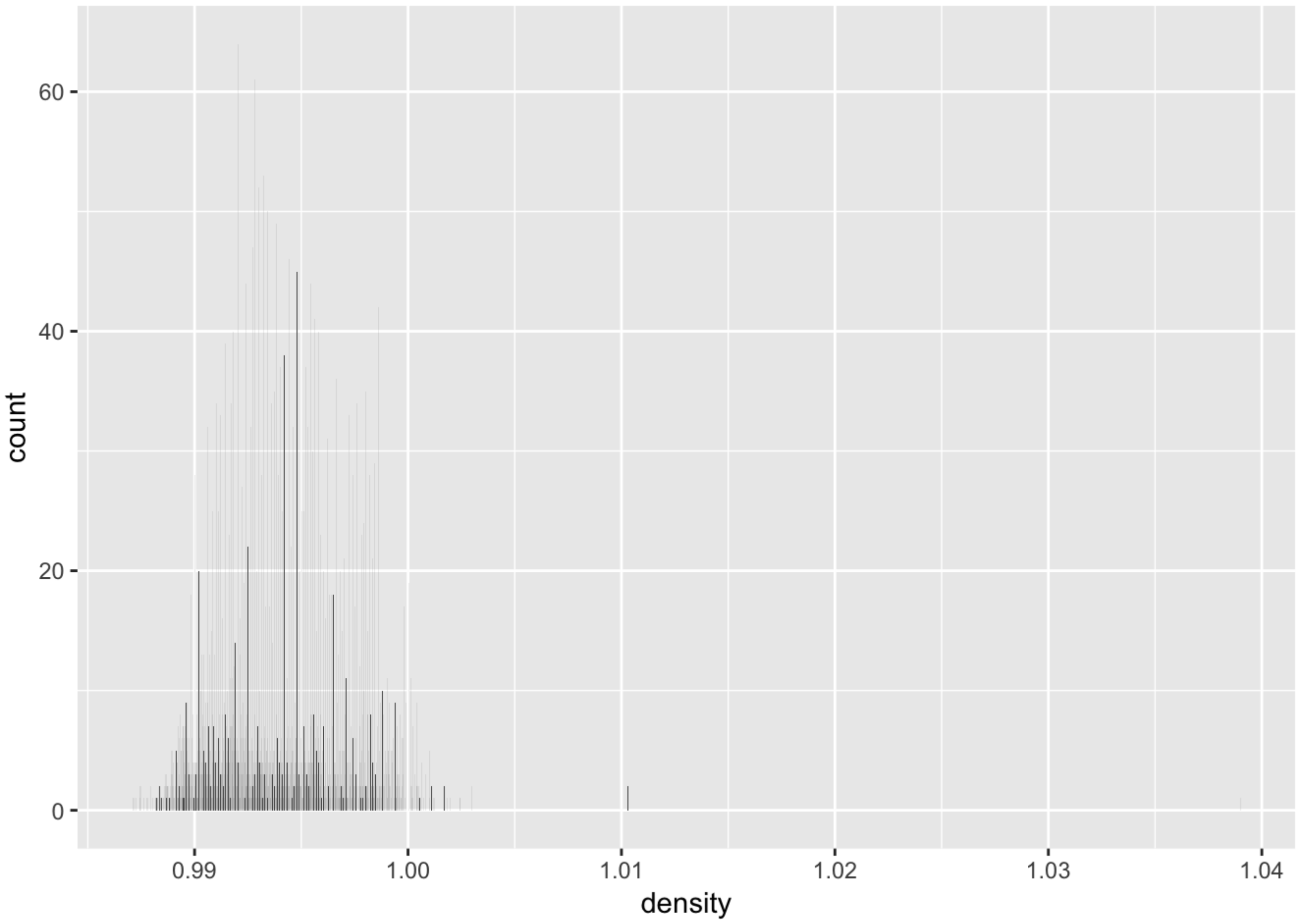


There are some peaks but the distribution is similar to normal Similar to residual sugar, the bad wines have low total sulphur dioxide, medium has high but good have lower than medium

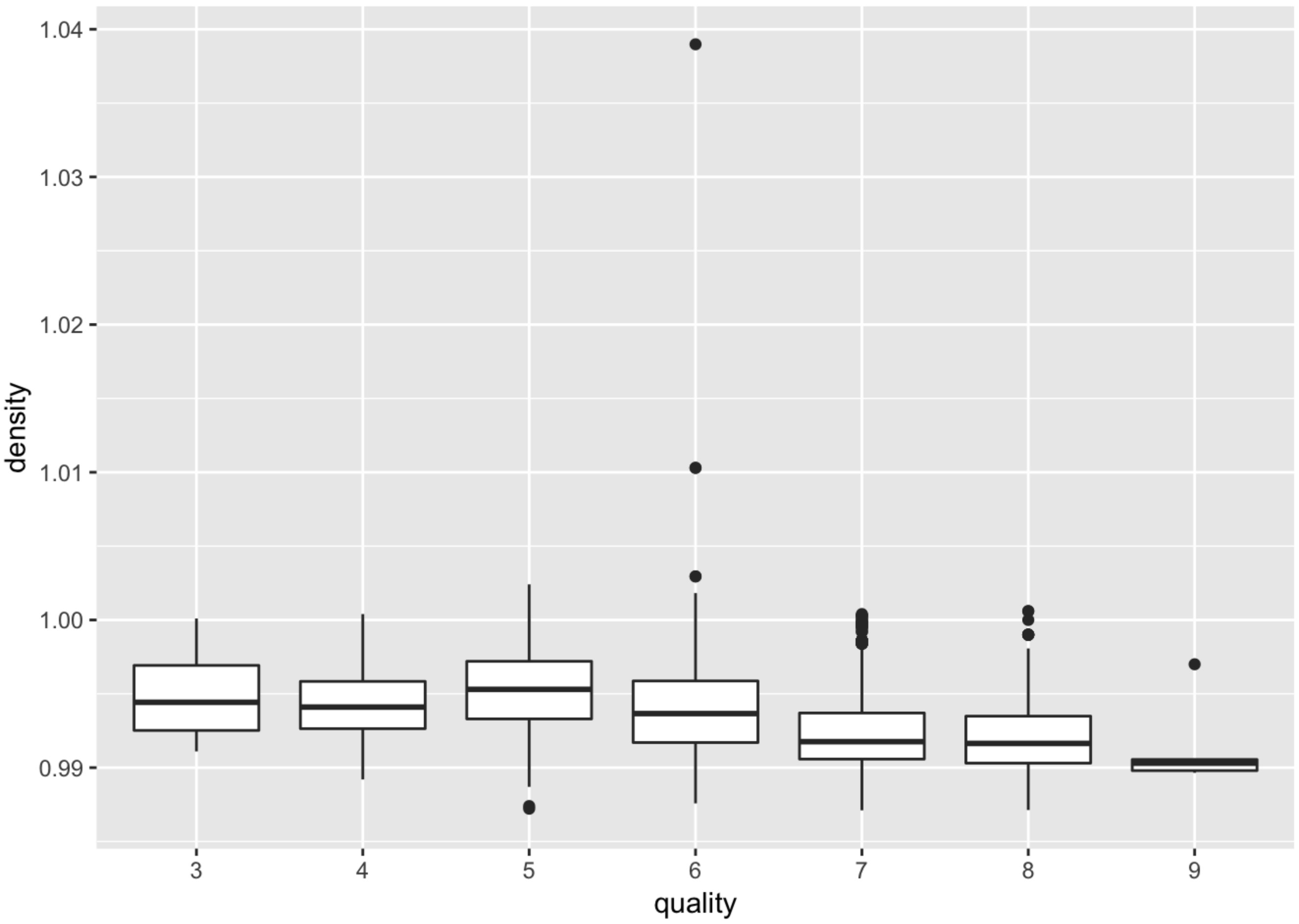
```
summary(wine$density)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9871  0.9917  0.9937  0.9940  0.9961  1.0390
```

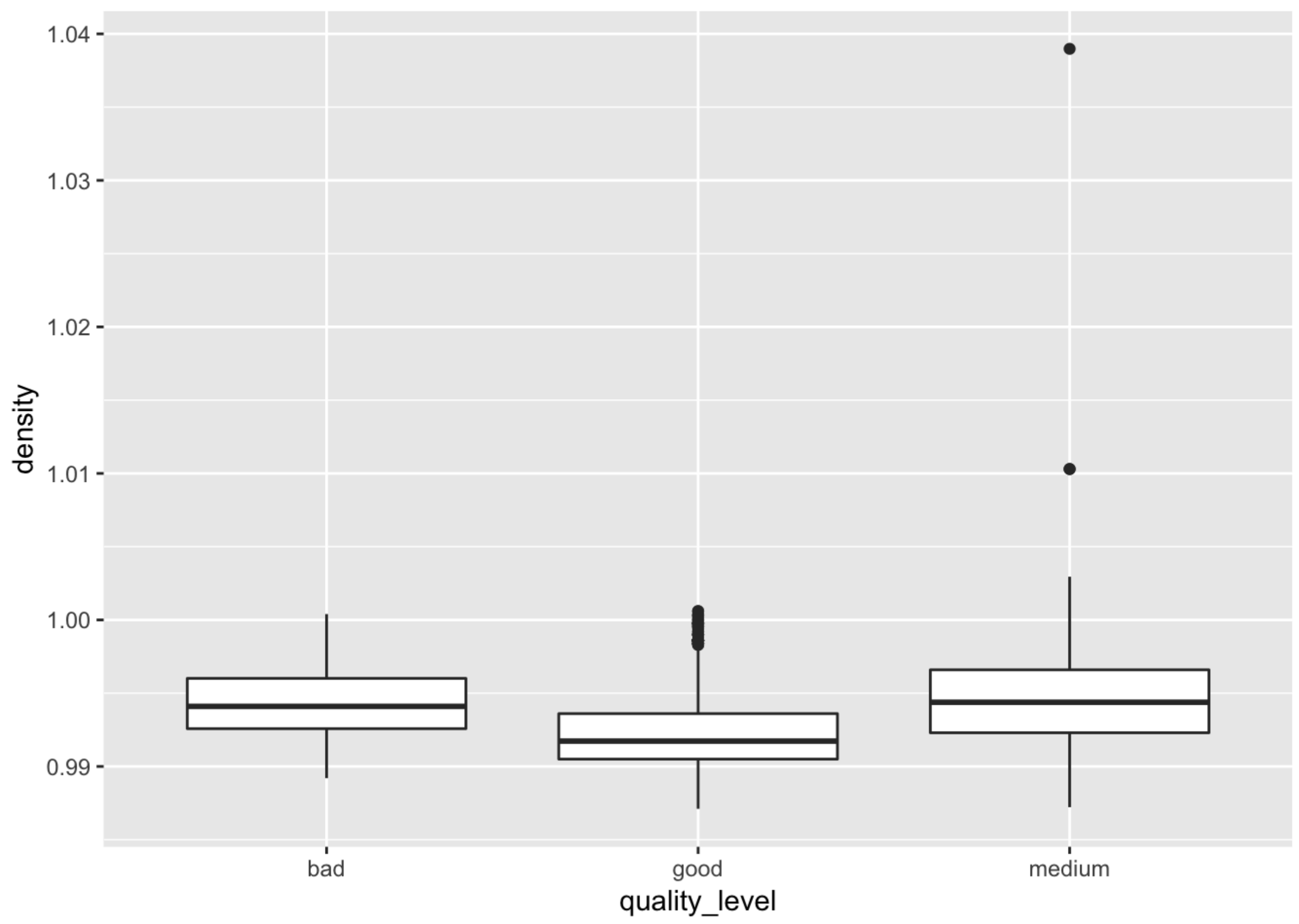
```
ggplot(aes(density), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=density), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=density), data = wine) + geom_boxplot()
```

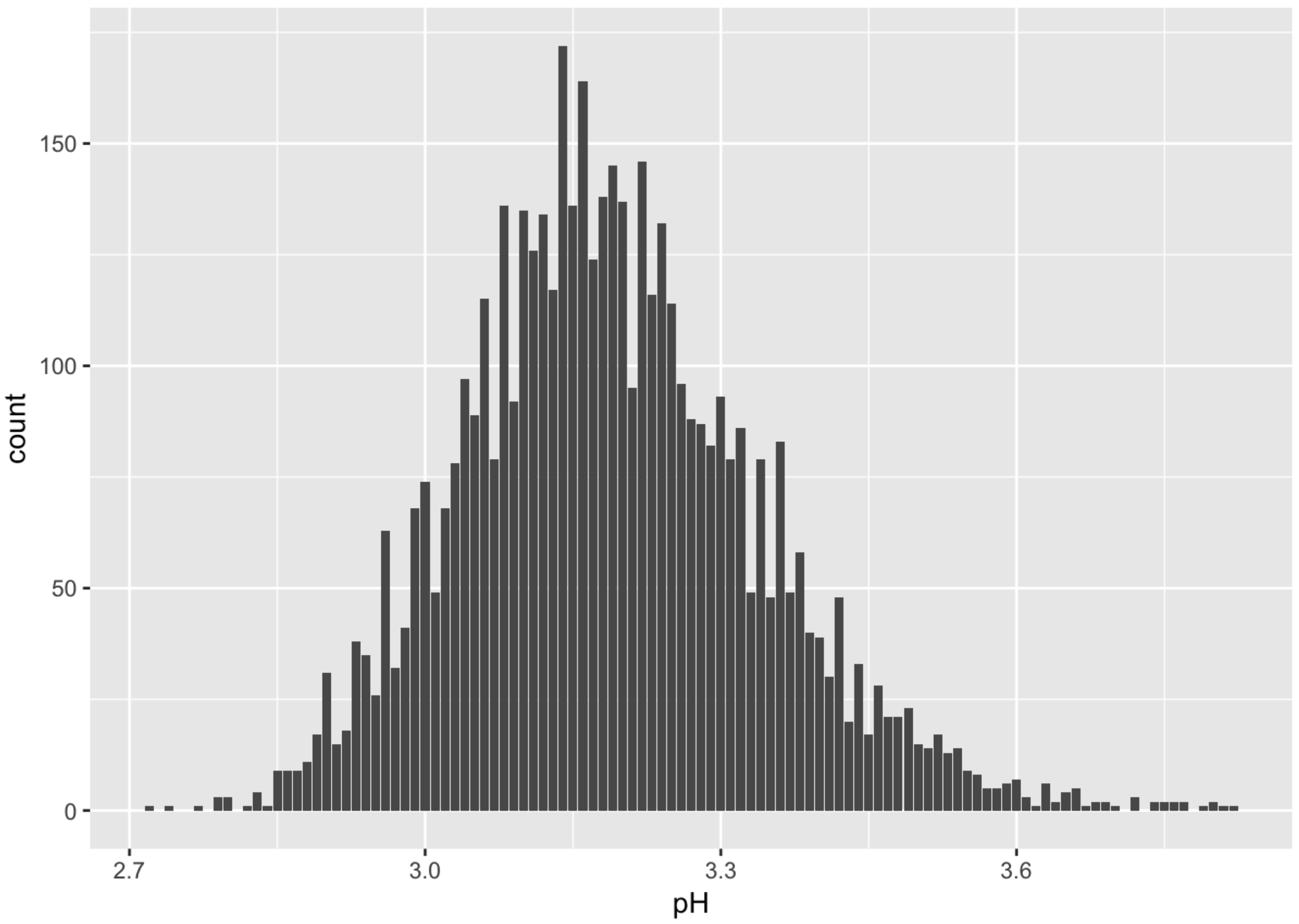


The range is very low and has a couple of outliers There is a negative relationship between density and quality level

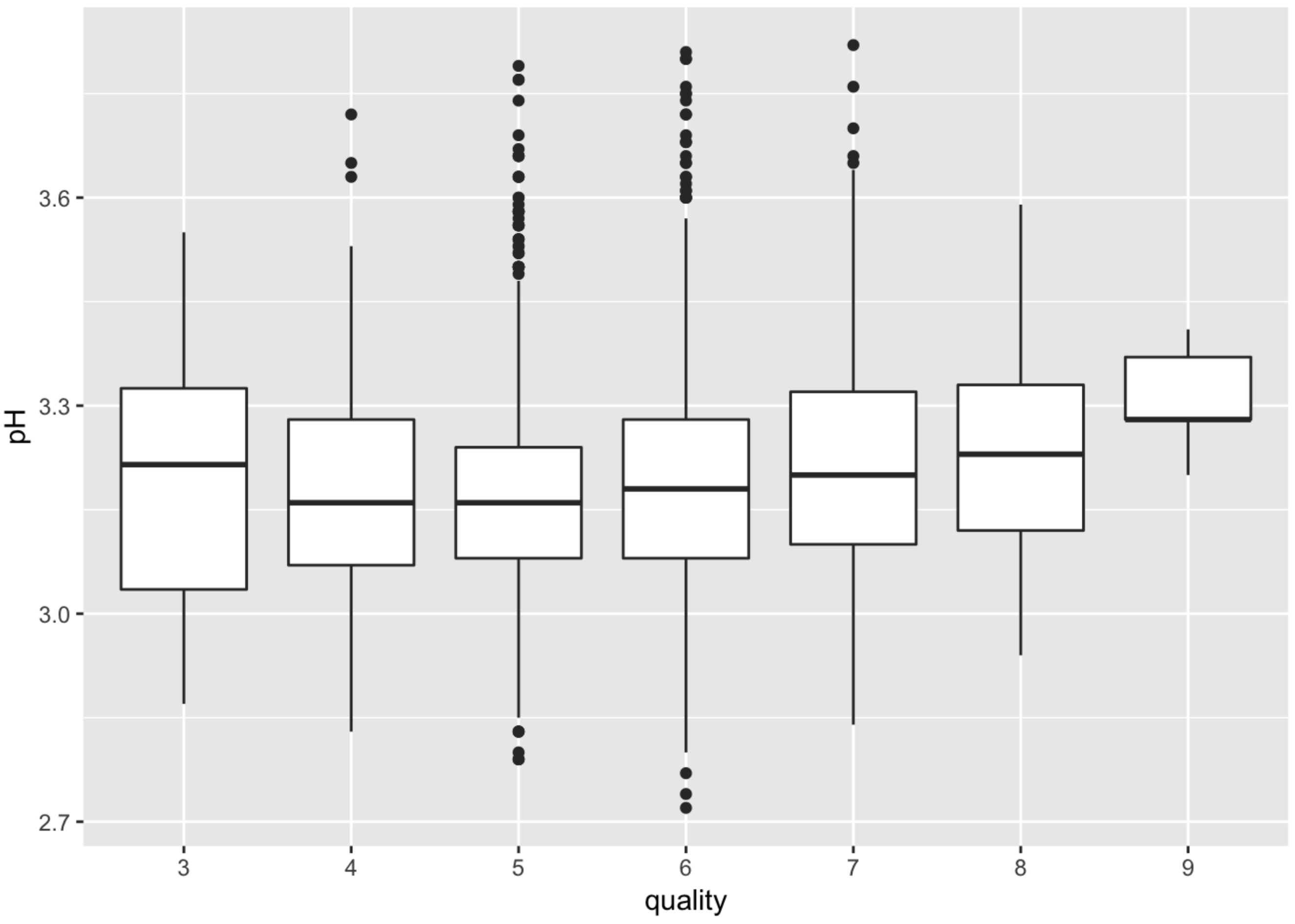
```
summary(wine$volatile_acidity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0800  0.2100  0.2600  0.2782  0.3200  1.1000
```

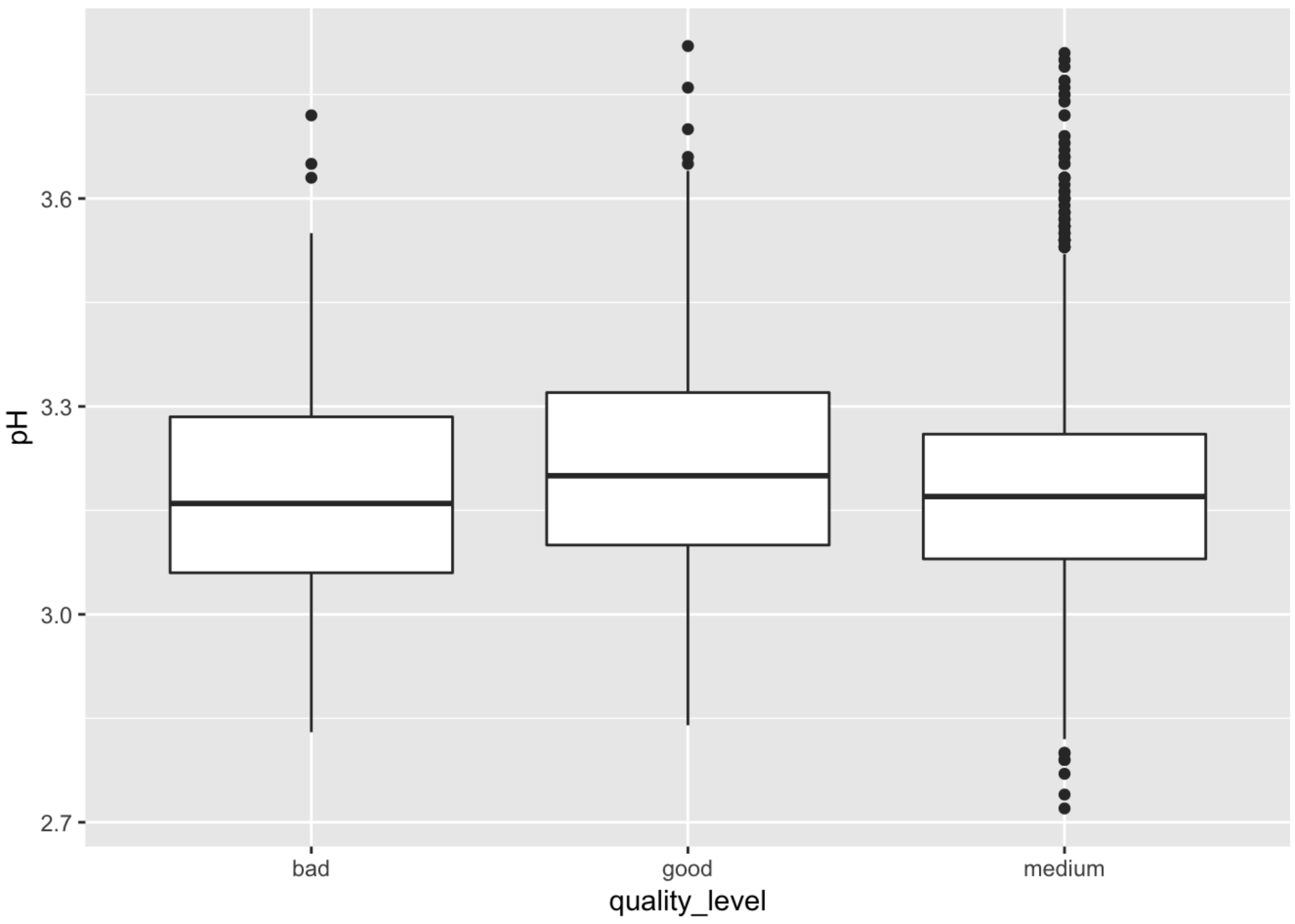
```
ggplot(aes(pH), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=pH), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=pH), data = wine) + geom_boxplot()
```

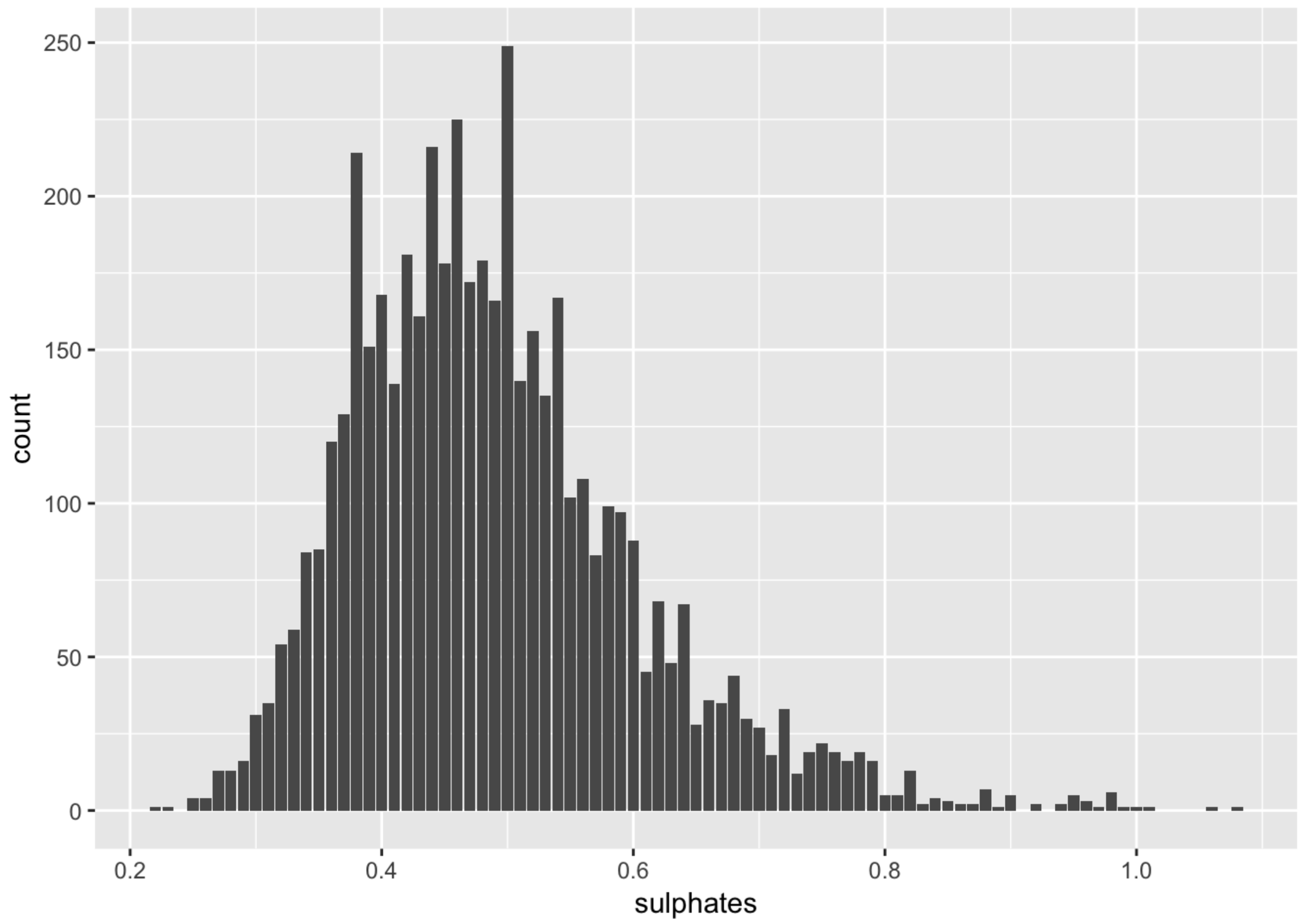


pH has a distribution very close to normal. there are peaks but no significant outliers There is a clear positive relationship between pH and Quality/Quality level.

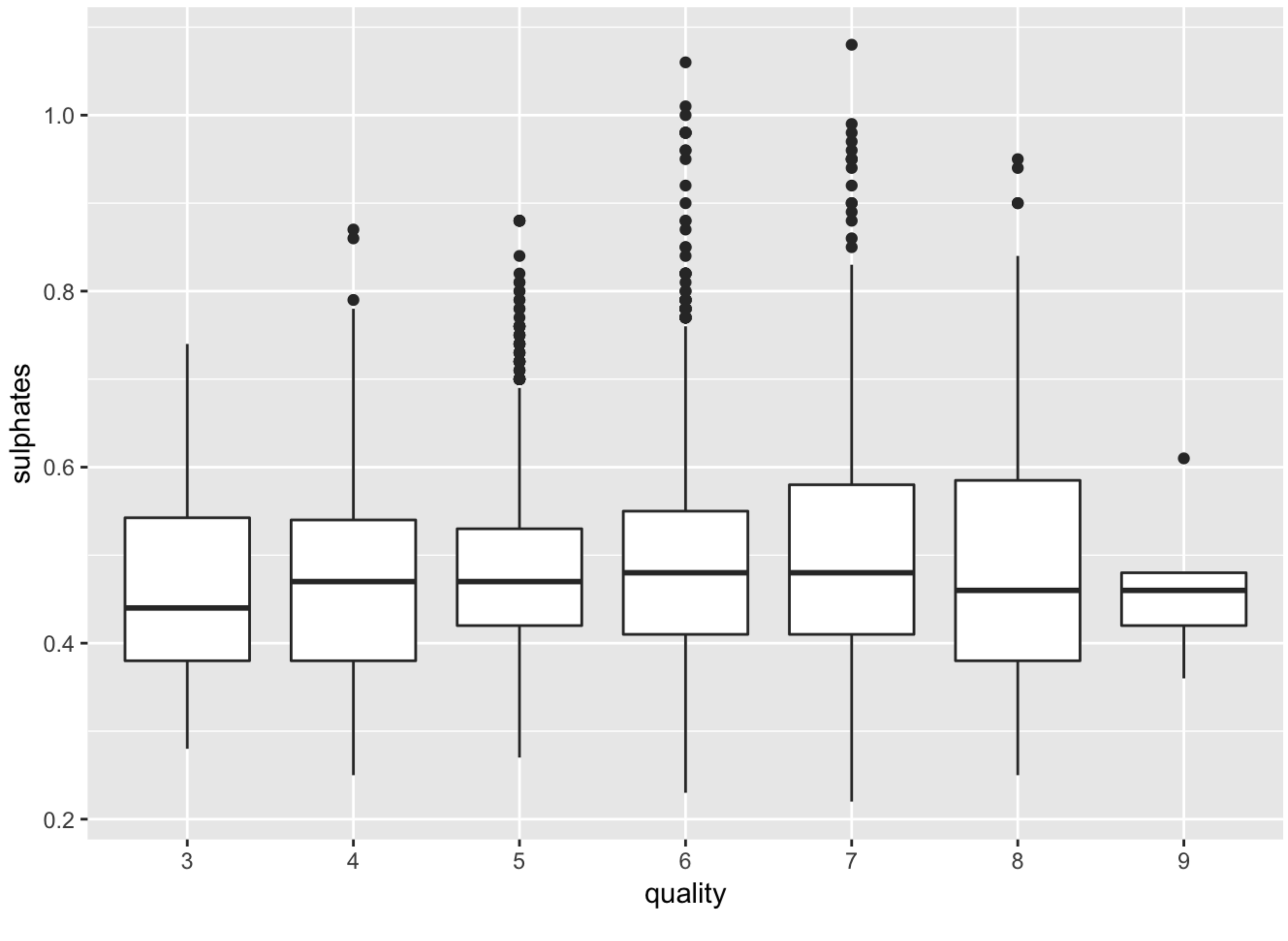
```
summary(wine$sulphates)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.2200  0.4100  0.4700  0.4898  0.5500  1.0800
```

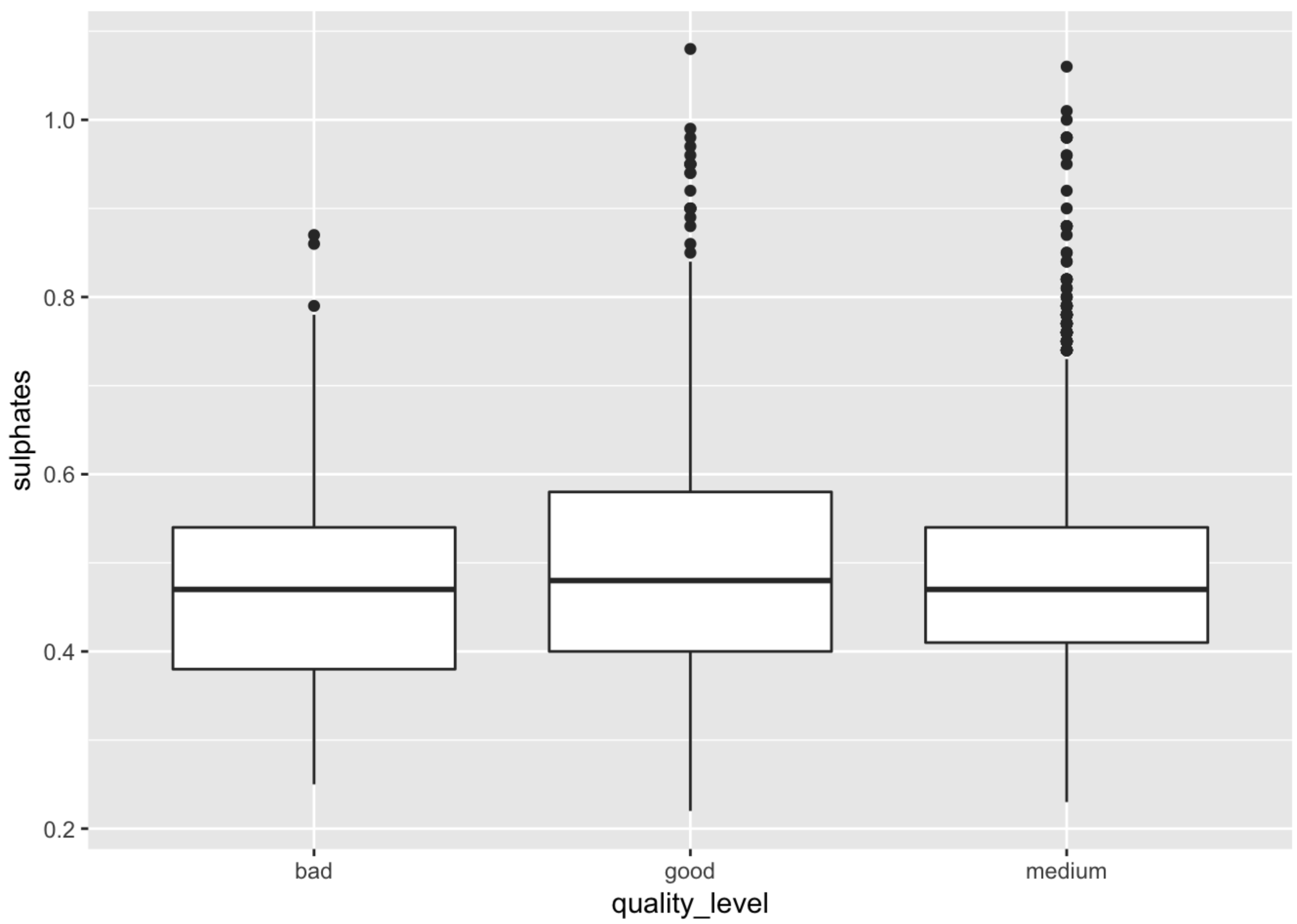
```
ggplot(aes(sulphates), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=sulphates), data = wine) + geom_boxplot()
```



```
ggplot(aes(x=quality_level, y=sulphates), data = wine) + geom_boxplot()
```

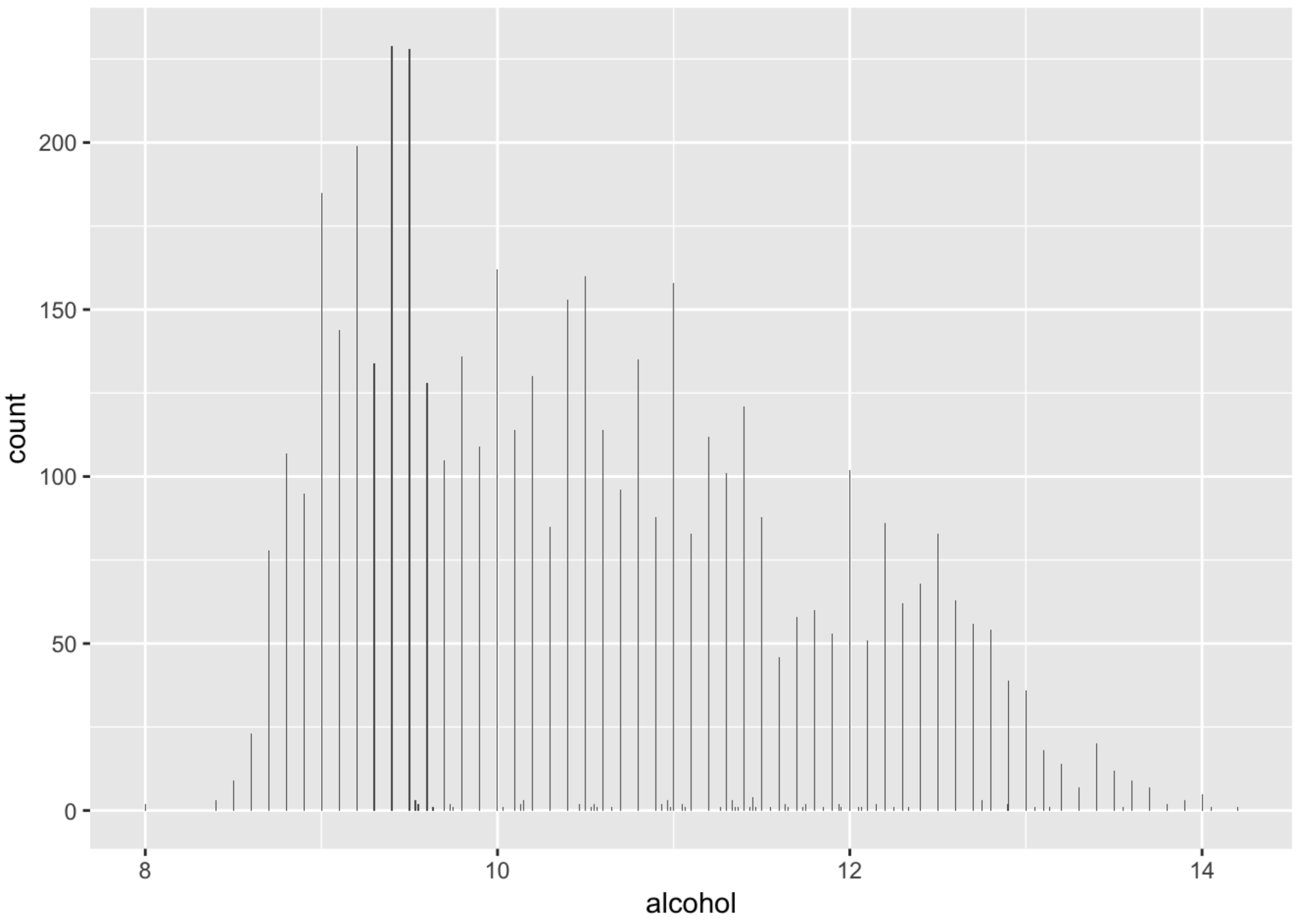


The data is positive skewed and a few high values Clear positive relation with quality

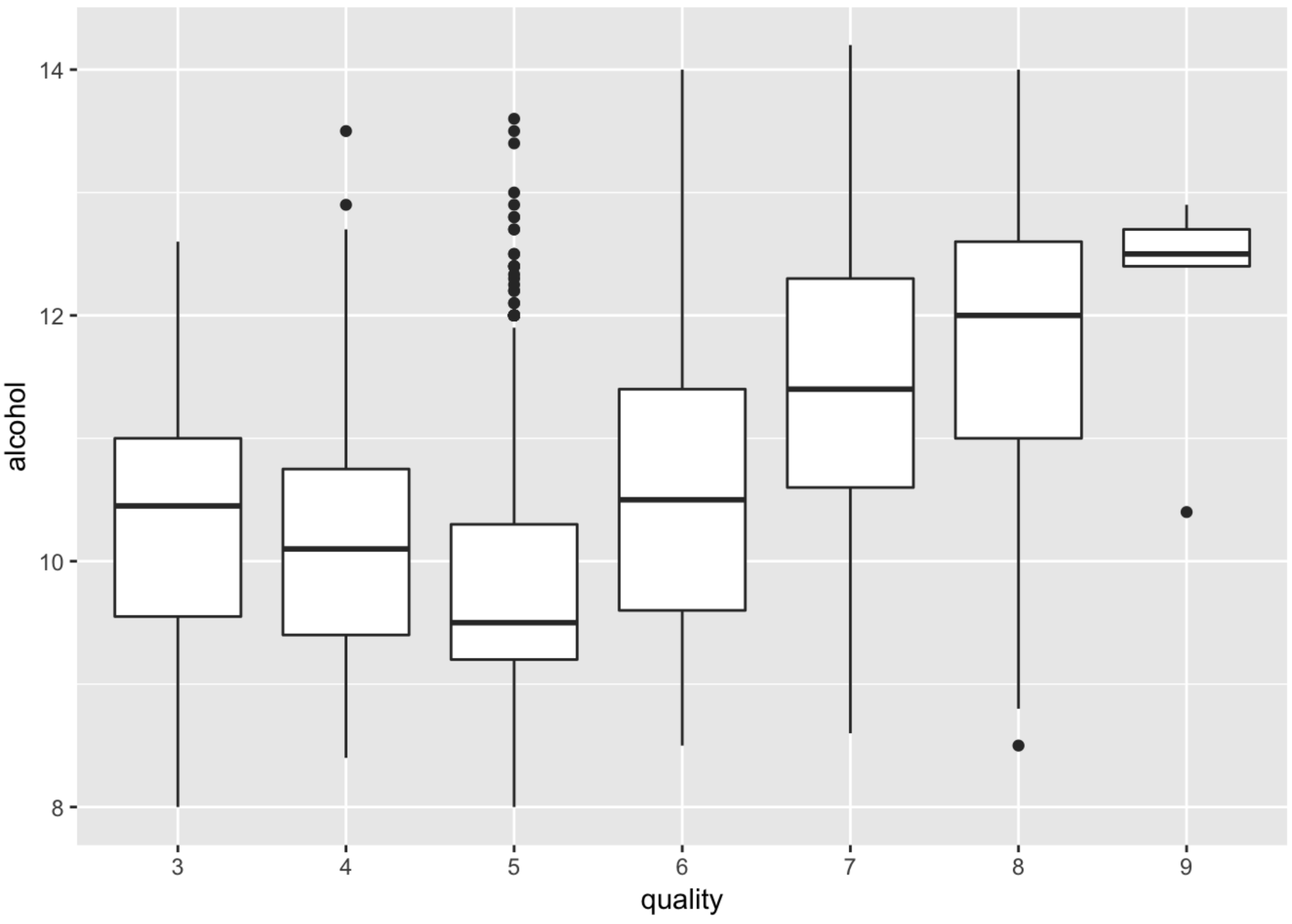
```
summary(wine$alcohol)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.00    9.50   10.40   10.51   11.40   14.20
```

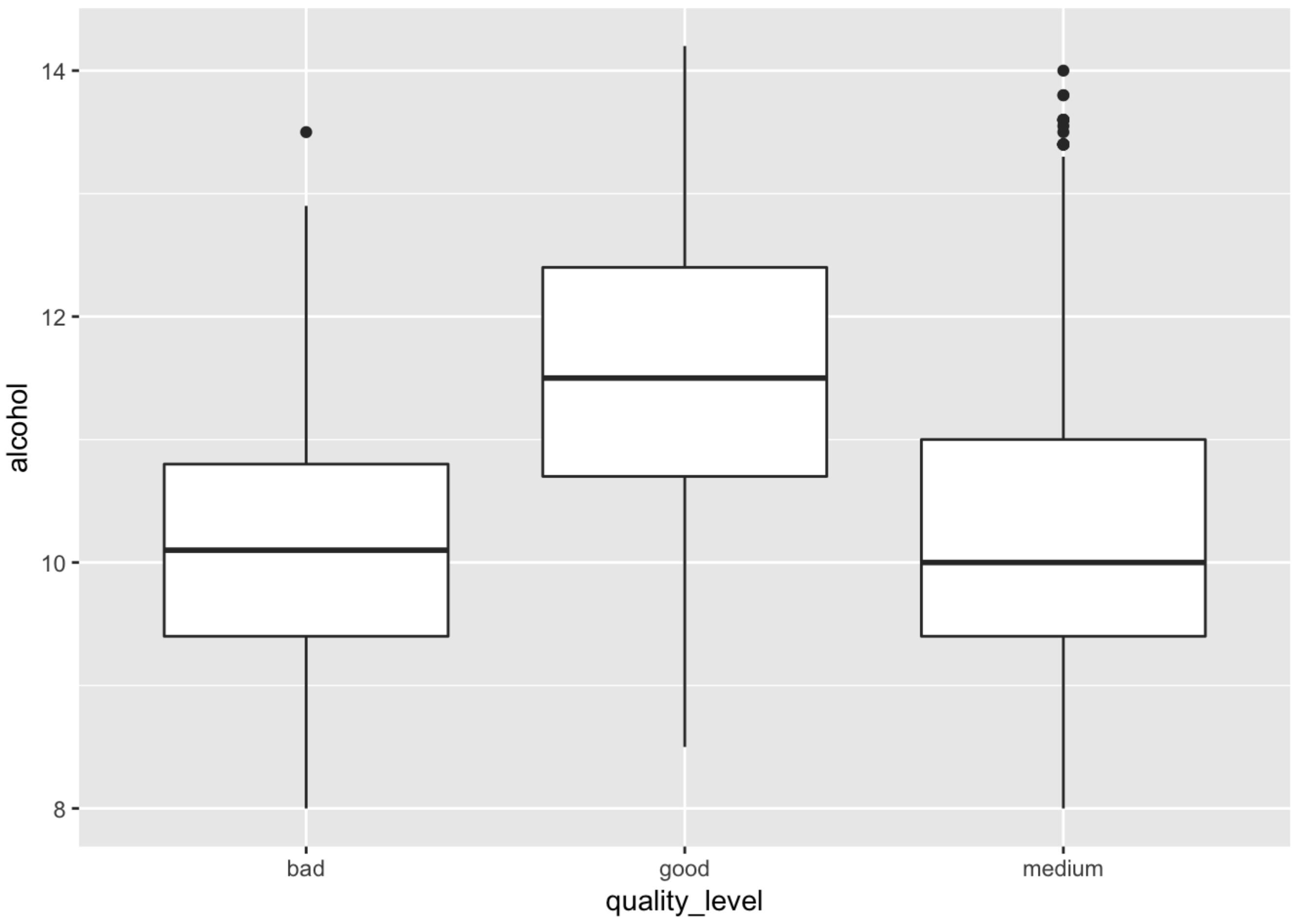
```
ggplot(aes(alcohol), data = wine) + geom_bar()
```



```
ggplot(aes(x=quality, y=alcohol), data = wine) + geom_boxplot()
```



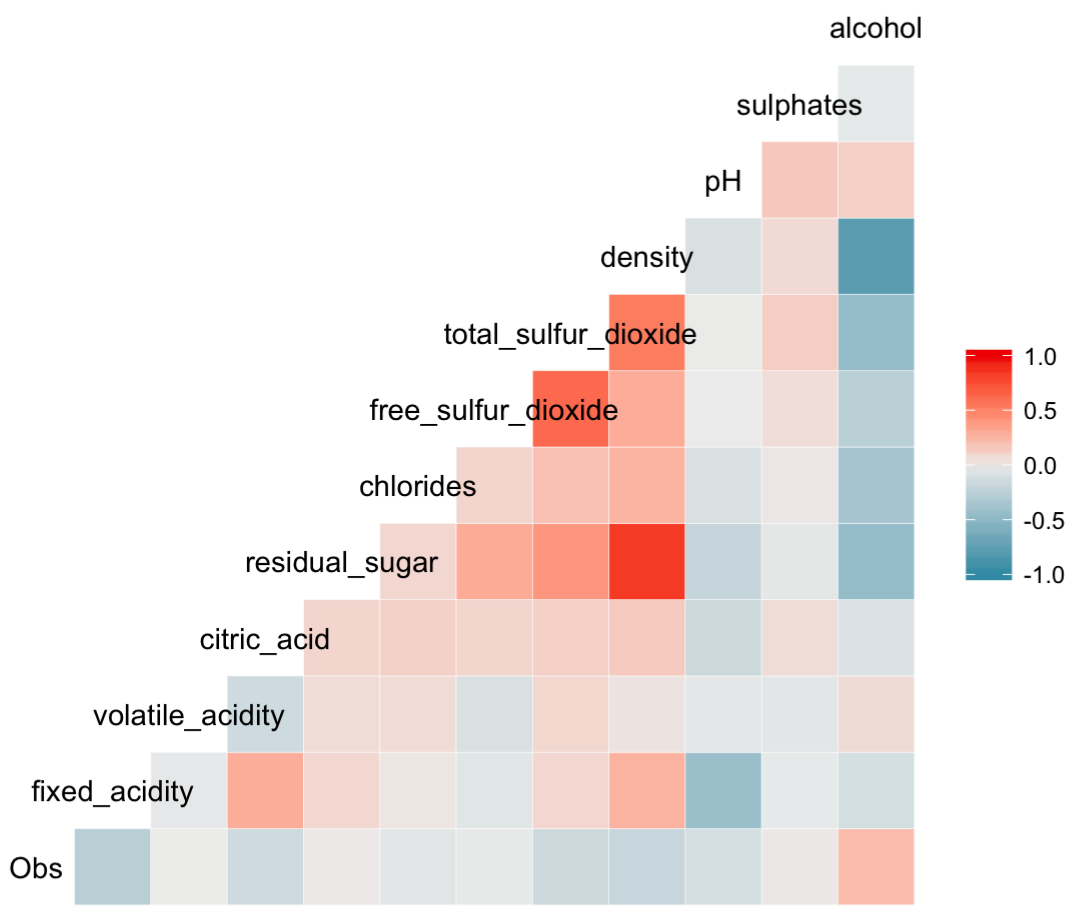
```
ggplot(aes(x=quality_level, y=alcohol), data = wine) + geom_boxplot()
```



There are a number of peaks, looks like no outliers There seems to be a positive relationship with quality butthe trend is not completely clear. Good wines have very high alchohol but medium wines have lower alchohol than bad wines which is surprising

```
ggcorr(wine)
```

```
## Warning in ggcorr(wine): data in column(s) 'quality', 'quality_level' are
## not numeric and were ignored
```



Density and Residual sugar has really high correlation (0.84) Density and alchohol also has a high (negative) correlation (-0.78) Density also as a high correlation with total sulfur dioxide(0.53) Free sulfur dioxide and total sulfur dioxide are also highly correlated(0.62) Alchohol is negatively correlated with total sulfur dioxide and residual sugar(both -45) Alchohol and quality has a correlation of 44 pH and fixed acidity are also negatively correlated (-0.43)

```
set.seed(123)
samp <- sample(nrow(wine), 0.6 * nrow(wine))
train <- wine[samp, ]
test <- wine[-samp, ]
```

```
svmlinear1 <- train(quality ~., data = train,
                   method = "svmLinear",
                   trControl=trainControl(method = "repeatedcv",
                                          number = 10, repeats = 10),
                   preProcess = c("center", "scale"),
                   tuneLength = 10)

svmlinear1$bestTune
```

```
##   C
## 1 1
```

```
summary(svmlinear1)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
predsvmLinear1 <- predict(svmlinear1, test)
confusionMatrix(table(predsvmLinear1, test$quality))
```

```
## Confusion Matrix and Statistics
##
##
## predsvmLinear1   3   4   5   6   7   8   9
##              3   0   1   0   0   0   0   0
##              4   6  72   0   0   0   0   0
##              5   0   0 304 139   0   0   0
##              6   0   0 284 743   0   0   0
##              7   0   0   0   0 337  73   1
##              8   0   0   0   0   0   0   0
##              9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.7429
##                  95% CI : (0.7229, 0.7621)
##     No Information Rate : 0.45
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6097
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                       Class: 3 Class: 4 Class: 5 Class: 6 Class: 7
## Sensitivity          0.0000000  0.98630   0.5170   0.8424   1.0000
## Specificity          0.9994882  0.99682   0.8987   0.7365   0.9544
## Pos Pred Value       0.0000000  0.92308   0.6862   0.7235   0.8200
## Neg Pred Value       0.9969372  0.99947   0.8128   0.8510   1.0000
## Prevalence           0.0030612  0.03724   0.3000   0.4500   0.1719
## Detection Rate       0.0000000  0.03673   0.1551   0.3791   0.1719
## Detection Prevalence 0.0005102  0.03980   0.2260   0.5240   0.2097
## Balanced Accuracy    0.4997441  0.99156   0.7078   0.7895   0.9772
##                      Class: 8  Class: 9
## Sensitivity           0.00000 0.0000000
## Specificity           1.00000 1.0000000
## Pos Pred Value            NaN       NaN
## Neg Pred Value        0.96276 0.9994898
## Prevalence            0.03724 0.0005102
## Detection Rate        0.00000 0.0000000
## Detection Prevalence  0.00000 0.0000000
## Balanced Accuracy     0.50000 0.5000000
```

We have run the SVM model on Quality variable. For the best model, we found C=1. The accuracy is 74.29%

```
svmlinear2 <- train(quality_level ~., data = train,
              method = "svmLinear",
              trControl=trainControl(method = "repeatedcv",
                                     number = 10, repeats = 10),
              preProcess = c("center", "scale"),
              tuneLength = 10)

svmlinear2$bestTune
```

```
##   C
## 1 1
```

```
summary(svmlinear2)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
predsvmLinear2 <- predict(svmlinear2, test)
confusionMatrix(table(predsvmLinear2, test$quality_level))
```

```
## Confusion Matrix and Statistics
##
##
## predsvmLinear2  bad good medium
##         bad      79    0      0
##         good      0  411      0
##         medium    0    0   1470
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9981, 1)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: bad Class: good Class: medium
## Sensitivity             1.00000      1.0000          1.00
## Specificity             1.00000      1.0000          1.00
## Pos Pred Value          1.00000      1.0000          1.00
## Neg Pred Value          1.00000      1.0000          1.00
## Prevalence              0.04031      0.2097          0.75
## Detection Rate          0.04031      0.2097          0.75
## Detection Prevalence    0.04031      0.2097          0.75
## Balanced Accuracy       1.00000      1.0000          1.00
```

We have run the SVM model on Quality Level variable. For the best model, we found C=1. The accuracy is 100%

```
nb1 <- naiveBayes(quality ~ ., data = train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##           3           4           5           6           7           8
## 0.004765146 0.030633084 0.295779442 0.447923758 0.184819605 0.034717495
##           9
## 0.001361470
##
## Conditional probabilities:
##    Obs
## Y       [,1]      [,2]
##   3 2099.929 1376.8218
##   4 2225.933 1358.8583
##   5 2348.731 1389.2656
##   6 2572.020 1443.3965
##   7 2435.808 1420.9248
##   8 2308.284 1301.7364
##   9 1021.500  391.8865
##
##    fixed_acidity
## Y       [,1]      [,2]
##   3 7.085714 1.3283055
##   4 7.224444 1.1396147
##   5 6.952589 0.8623638
##   6 6.853305 0.8474624
##   7 6.744015 0.7628095
##   8 6.683333 0.7880397
##   9 7.625000 1.0045729
##
##    volatile_acidity
## Y        [,1]       [,2]
##   3 0.3425000 0.14183888
##   4 0.3950000 0.17425604
##   5 0.3030667 0.10381436
##   6 0.2630471 0.09024842
##   7 0.2637385 0.09060641
##   8 0.2681863 0.10808135
##   9 0.2825000 0.05315073
##
##    citric_acid
## Y        [,1]       [,2]
##   3 0.3142857 0.08327104
##   4 0.3047778 0.16604607
##   5 0.3403452 0.14220570
##   6 0.3371657 0.12126998
##   7 0.3270350 0.07921812
##   8 0.3280392 0.07226244
##   9 0.4100000 0.07164728
##
##    residual_sugar
## Y       [,1]     [,2]
##   3 7.207143 5.383338
##    4 4.403889 3.824766
##    5 7.318297 5.359412
##    6 6.571429 5.264301
##    7 5.126611 4.296436
##    8 5.754902 4.428405
##    9 4.750000 4.024508
## 
##     chlorides
## Y          [,1]        [,2]
##    3 0.05907143 0.054862716
##    4 0.05053333 0.030955278
##    5 0.05108516 0.025828611
##    6 0.04520289 0.020006136
##    7 0.03804788 0.010193040
##    8 0.03855882 0.012628409
##    9 0.02900000 0.007527727
## 
##     free_sulfur_dioxide
## Y        [,1]     [,2]
##    3 68.17857 78.83688
##    4 21.83889 17.50246
##    5 36.11795 17.88182
##    6 35.50684 15.31668
##    7 34.41344 13.58294
##    8 37.03922 17.56793
##    9 35.75000 14.26826
## 
##     total_sulfur_dioxide
## Y        [,1]      [,2]
##    3 200.0714 113.36715
##    4 122.9444  50.84088
##    5 149.4689  43.69120
##    6 137.5794  41.25911
##    7 124.9429  33.02970
##    8 126.9314  33.33679
##    9 123.7500  11.11680
## 
##     density
## Y         [,1]        [,2]
##    3 0.9950629 0.002737872
##    4 0.9942519 0.002343708
##    5 0.9952456 0.002533501
##    6 0.9940534 0.003078789
##    7 0.9924701 0.002811543
##    8 0.9924160 0.002822665
##    9 0.9919125 0.003405969
## 
##     pH
## Y        [,1]       [,2]
##    3 3.206429 0.22165834
##    4 3.164778 0.16100267
##    5 3.166605 0.13844631
##    6 3.187302 0.15220639
##    7 3.211142 0.15931757
##    8 3.213431 0.14482830
##    9 3.282500 0.06946222
## 
##     sulphates
## Y         [,1]       [,2]
##   3 0.5092857 0.11505613
##   4 0.4713333 0.13418250
##   5 0.4817722 0.09809165
##   6 0.4905243 0.11392860
##   7 0.5075322 0.13382875
##   8 0.4920588 0.14982061
##   9 0.4300000 0.05291503
## 
##    alcohol
## Y         [,1]     [,2]
##   3 10.300000 1.179309
##   4 10.142222 1.015495
##   5  9.812712 0.845418
##   6 10.552685 1.122599
##   7 11.336108 1.263374
##   8 11.511765 1.284584
##   9 12.125000 1.161536
## 
##    quality_level
## Y   bad good medium
##   3   1    0      0
##   4   1    0      0
##   5   0    0      1
##   6   0    0      1
##   7   0    1      0
##   8   0    1      0
##   9   0    1      0
```

```
# probabilities
pred.prob <- predict(nb1, newdata = test, type = "raw")
# class membership
pred.class1 <- predict(nb1, newdata = test)
confusionMatrix(pred.class1, test$quality)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   2   4   1   0   0   0
##          4   6  64   2   1   0   0   0
##          5   0   7 341 234  10   3   0
##          6   0   0 241 646   0   1   0
##          7   0   0   0   0 324  67   1
##          8   0   0   0   0   3   1   0
##          9   0   0   0   0   0   1   0
##
## Overall Statistics
##
##                Accuracy : 0.702
##                  95% CI : (0.6812, 0.7222)
##     No Information Rate : 0.45
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5547
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7
## Sensitivity          0.000000  0.87671   0.5799   0.7324   0.9614
## Specificity          0.996418  0.99523   0.8149   0.7755   0.9581
## Pos Pred Value       0.000000  0.87671   0.5731   0.7275   0.8265
## Neg Pred Value       0.996928  0.99523   0.8190   0.7799   0.9917
## Prevalence           0.003061  0.03724   0.3000   0.4500   0.1719
## Detection Rate       0.000000  0.03265   0.1740   0.3296   0.1653
## Detection Prevalence 0.003571  0.03724   0.3036   0.4531   0.2000
## Balanced Accuracy    0.498209  0.93597   0.6974   0.7540   0.9598
##                       Class: 8  Class: 9
## Sensitivity          0.0136986 0.0000000
## Specificity          0.9984102 0.9994895
## Pos Pred Value       0.2500000 0.0000000
## Neg Pred Value       0.9631902 0.9994895
## Prevalence           0.0372449 0.0005102
## Detection Rate       0.0005102 0.0000000
## Detection Prevalence 0.0020408 0.0005102
## Balanced Accuracy    0.5060544 0.4997448
```

We have run the Naive Bayes model on Quality variable. The accuracy is 70.2%

```
nb2 <- naiveBayes(quality_level ~ ., data = train)
nb2
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        bad       good     medium
## 0.03539823 0.22089857 0.74370320
##
## Conditional probabilities:
##         Obs
## Y            [,1]     [,2]
##   bad    2208.971 1355.226
##   good   2407.049 1402.733
##   medium 2483.215 1425.990
##
##         fixed_acidity
## Y            [,1]      [,2]
##   bad    7.205769 1.1606683
##   good   6.739908 0.7703979
##   medium 6.892792 0.8546064
##
##         volatile_acidity
## Y             [,1]       [,2]
##   bad    0.3879327 0.17059094
##   good   0.2645532 0.09330084
##   medium 0.2789634 0.09783234
##
##         citric_acid
## Y             [,1]       [,2]
##   bad    0.3060577 0.15719272
##   good   0.3277042 0.07828682
##   medium 0.3384302 0.12997836
##
##         residual_sugar
## Y            [,1]     [,2]
##   bad    4.781250 4.149998
##   good   5.223035 4.315678
##   medium 6.868467 5.313702
##
##         chlorides
## Y              [,1]       [,2]
##   bad    0.05168269 0.03487767
##   good   0.03807242 0.01060977
##   medium 0.04754233 0.02268080
##
##         free_sulfur_dioxide
## Y            [,1]     [,2]
##   bad    28.07692 36.07952
##   good   34.83436 14.29274
##   medium 35.74989 16.38377
##
##         total_sulfur_dioxide
## Y            [,1]     [,2]
##   bad    133.3269 67.49278
##   good   125.2481 32.96710
##   medium 142.3080 42.63243
##
##         density
## Y              [,1]        [,2]
##   bad    0.9943611 0.002402041
##   good   0.9924581 0.002812382
##   medium 0.9945276 0.002932400
##
##         pH
## Y             [,1]      [,2]
##   bad    3.170385 0.1697166
##   good   3.211941 0.1566950
##   medium 3.179071 0.1472051
##
##         sulphates
## Y              [,1]      [,2]
##   bad    0.4764423 0.1319011
##   good   0.5046225 0.1362286
##   medium 0.4870435 0.1079706
##
##         alcohol
## Y             [,1]     [,2]
##   bad    10.16346 1.034179
##   good   11.36858 1.267324
##   medium 10.25839 1.083546
##
##         quality
## Y                  3           4           5           6           7
##   bad    0.134615385 0.865384615 0.000000000 0.000000000 0.000000000
##   good   0.000000000 0.000000000 0.000000000 0.000000000 0.836671803
##   medium 0.000000000 0.000000000 0.397711670 0.602288330 0.000000000
##         quality
## Y                  8           9
##   bad    0.000000000 0.000000000
##   good   0.157164869 0.006163328
##   medium 0.000000000 0.000000000
```

```
# probabilities
pred.prob <- predict(nb2, newdata = test, type = "raw")
# class membership
pred.class2 <- predict(nb2, newdata = test)
confusionMatrix(pred.class2, test$quality_level)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  bad good medium
##     bad      72    1     16
##     good      2  393      2
##     medium    5   17   1452
## 
## Overall Statistics
## 
##                Accuracy : 0.9781          
##                  95% CI : (0.9706, 0.9841)
##     No Information Rate : 0.75            
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.944           
##  Mcnemar's Test P-Value : 0.0004531       
## 
## Statistics by Class:
## 
##                      Class: bad Class: good Class: medium
## Sensitivity             0.91139      0.9562        0.9878
## Specificity             0.99096      0.9974        0.9551
## Pos Pred Value          0.80899      0.9899        0.9851
## Neg Pred Value          0.99626      0.9885        0.9630
## Prevalence              0.04031      0.2097        0.7500
## Detection Rate          0.03673      0.2005        0.7408
## Detection Prevalence    0.04541      0.2026        0.7520
## Balanced Accuracy       0.95118      0.9768        0.9714
```

We have run the Naive Bayes model on Quality Level variable. The accuracy is 97.81%

```
set.seed(123)
ridgeReg1 <- train(quality~., train, method = 'glmnet',
                  tuneGrid = expand.grid(alpha = 0,
                                         lambda = seq(0.0001, 0.5, length = 5)),
                  trControl = trainControl(method = "repeatedcv",
                                           number = 10, repeats = 10,
                                           verboseIter = TRUE))
```

```
## + Fold01.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep01: alpha=0, lambda=0.5
## + Fold02.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold02.Rep01: alpha=0, lambda=0.5
## + Fold03.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep01: alpha=0, lambda=0.5
## + Fold04.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep01: alpha=0, lambda=0.5
## + Fold05.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep01: alpha=0, lambda=0.5
## + Fold06.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep01: alpha=0, lambda=0.5
## + Fold07.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep01: alpha=0, lambda=0.5
## + Fold08.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep01: alpha=0, lambda=0.5
## + Fold09.Rep01: alpha=0, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep01: alpha=0, lambda=0.5
## + Fold10.Rep01: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep01: alpha=0, lambda=0.5
## + Fold01.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep02: alpha=0, lambda=0.5
## + Fold02.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep02: alpha=0, lambda=0.5
## + Fold03.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep02: alpha=0, lambda=0.5
## + Fold04.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep02: alpha=0, lambda=0.5
## + Fold05.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold05.Rep02: alpha=0, lambda=0.5
## + Fold06.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep02: alpha=0, lambda=0.5
## + Fold07.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep02: alpha=0, lambda=0.5
## + Fold08.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep02: alpha=0, lambda=0.5
## + Fold09.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep02: alpha=0, lambda=0.5
## + Fold10.Rep02: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep02: alpha=0, lambda=0.5
## + Fold01.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep03: alpha=0, lambda=0.5
## + Fold02.Rep03: alpha=0, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep03: alpha=0, lambda=0.5
## + Fold03.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep03: alpha=0, lambda=0.5
## + Fold04.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep03: alpha=0, lambda=0.5
## + Fold05.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep03: alpha=0, lambda=0.5
## + Fold06.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep03: alpha=0, lambda=0.5
## + Fold07.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep03: alpha=0, lambda=0.5
## + Fold08.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold08.Rep03: alpha=0, lambda=0.5
## + Fold09.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep03: alpha=0, lambda=0.5
## + Fold10.Rep03: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep03: alpha=0, lambda=0.5
## + Fold01.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep04: alpha=0, lambda=0.5
## + Fold02.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep04: alpha=0, lambda=0.5
## + Fold03.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep04: alpha=0, lambda=0.5
## + Fold04.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep04: alpha=0, lambda=0.5
## + Fold05.Rep04: alpha=0, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep04: alpha=0, lambda=0.5
## + Fold06.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep04: alpha=0, lambda=0.5
## + Fold07.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep04: alpha=0, lambda=0.5
## + Fold08.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep04: alpha=0, lambda=0.5
## + Fold09.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep04: alpha=0, lambda=0.5
## + Fold10.Rep04: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep04: alpha=0, lambda=0.5
## + Fold01.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold01.Rep05: alpha=0, lambda=0.5
## + Fold02.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep05: alpha=0, lambda=0.5
## + Fold03.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep05: alpha=0, lambda=0.5
## + Fold04.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep05: alpha=0, lambda=0.5
## + Fold05.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep05: alpha=0, lambda=0.5
## + Fold06.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep05: alpha=0, lambda=0.5
## + Fold07.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep05: alpha=0, lambda=0.5
## + Fold08.Rep05: alpha=0, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep05: alpha=0, lambda=0.5
## + Fold09.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep05: alpha=0, lambda=0.5
## + Fold10.Rep05: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep05: alpha=0, lambda=0.5
## + Fold01.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep06: alpha=0, lambda=0.5
## + Fold02.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep06: alpha=0, lambda=0.5
## + Fold03.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep06: alpha=0, lambda=0.5
## + Fold04.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold04.Rep06: alpha=0, lambda=0.5
## + Fold05.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep06: alpha=0, lambda=0.5
## + Fold06.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep06: alpha=0, lambda=0.5
## + Fold07.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep06: alpha=0, lambda=0.5
## + Fold08.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep06: alpha=0, lambda=0.5
## + Fold09.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep06: alpha=0, lambda=0.5
## + Fold10.Rep06: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep06: alpha=0, lambda=0.5
## + Fold01.Rep07: alpha=0, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep07: alpha=0, lambda=0.5
## + Fold02.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep07: alpha=0, lambda=0.5
## + Fold03.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep07: alpha=0, lambda=0.5
## + Fold04.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep07: alpha=0, lambda=0.5
## + Fold05.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep07: alpha=0, lambda=0.5
## + Fold06.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep07: alpha=0, lambda=0.5
## + Fold07.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold07.Rep07: alpha=0, lambda=0.5
## + Fold08.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep07: alpha=0, lambda=0.5
## + Fold09.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep07: alpha=0, lambda=0.5
## + Fold10.Rep07: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep07: alpha=0, lambda=0.5
## + Fold01.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep08: alpha=0, lambda=0.5
## + Fold02.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep08: alpha=0, lambda=0.5
## + Fold03.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep08: alpha=0, lambda=0.5
## + Fold04.Rep08: alpha=0, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep08: alpha=0, lambda=0.5
## + Fold05.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep08: alpha=0, lambda=0.5
## + Fold06.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep08: alpha=0, lambda=0.5
## + Fold07.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep08: alpha=0, lambda=0.5
## + Fold08.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep08: alpha=0, lambda=0.5
## + Fold09.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep08: alpha=0, lambda=0.5
## + Fold10.Rep08: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold10.Rep08: alpha=0, lambda=0.5
## + Fold01.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep09: alpha=0, lambda=0.5
## + Fold02.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep09: alpha=0, lambda=0.5
## + Fold03.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep09: alpha=0, lambda=0.5
## + Fold04.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep09: alpha=0, lambda=0.5
## + Fold05.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep09: alpha=0, lambda=0.5
## + Fold06.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep09: alpha=0, lambda=0.5
## + Fold07.Rep09: alpha=0, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep09: alpha=0, lambda=0.5
## + Fold08.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep09: alpha=0, lambda=0.5
## + Fold09.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep09: alpha=0, lambda=0.5
## + Fold10.Rep09: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep09: alpha=0, lambda=0.5
## + Fold01.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep10: alpha=0, lambda=0.5
## + Fold02.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep10: alpha=0, lambda=0.5
## + Fold03.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold03.Rep10: alpha=0, lambda=0.5
## + Fold04.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep10: alpha=0, lambda=0.5
## + Fold05.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep10: alpha=0, lambda=0.5
## + Fold06.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep10: alpha=0, lambda=0.5
## + Fold07.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep10: alpha=0, lambda=0.5
## + Fold08.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep10: alpha=0, lambda=0.5
## + Fold09.Rep10: alpha=0, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep10: alpha=0, lambda=0.5
## + Fold10.Rep10: alpha=0, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

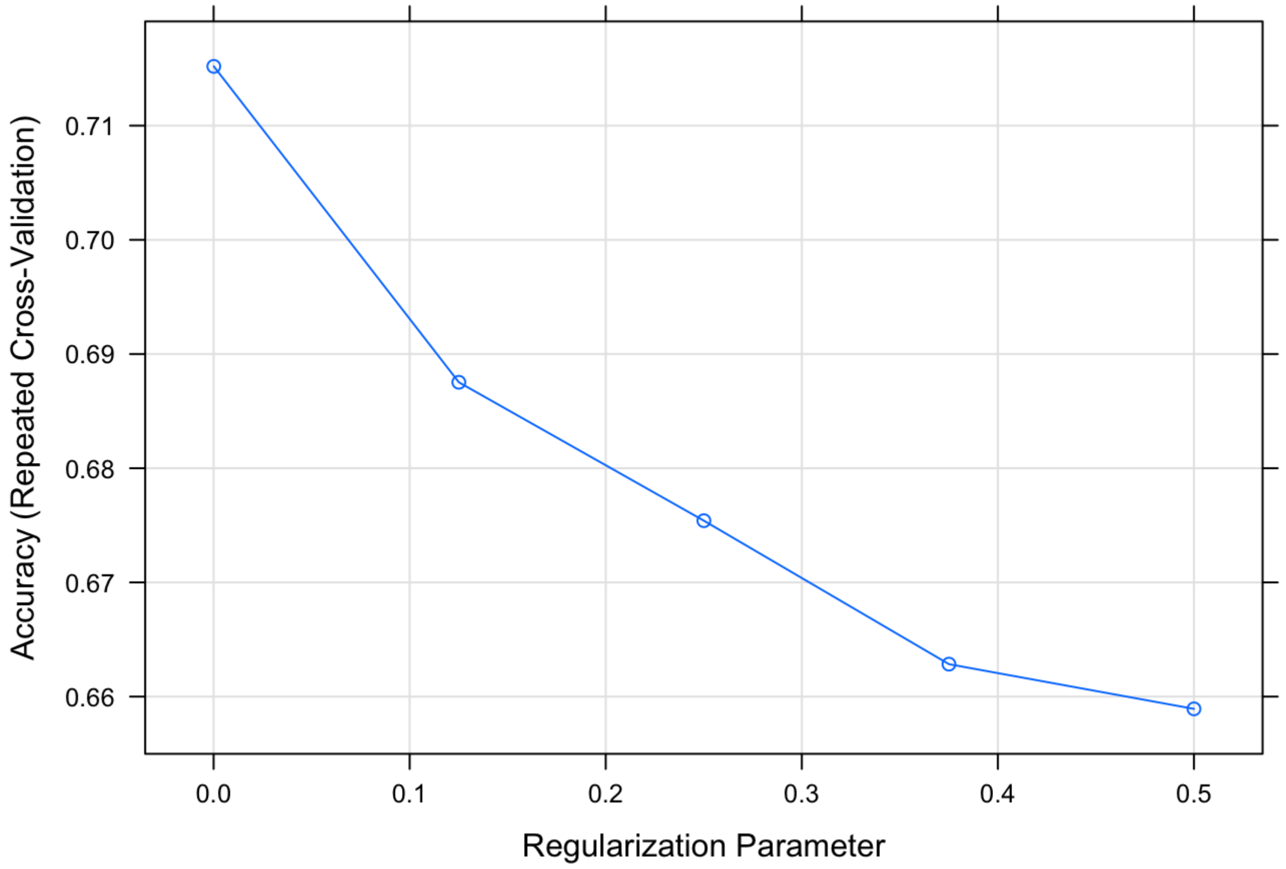
```
## - Fold10.Rep10: alpha=0, lambda=0.5
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0, lambda = 1e-04 on full training set
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

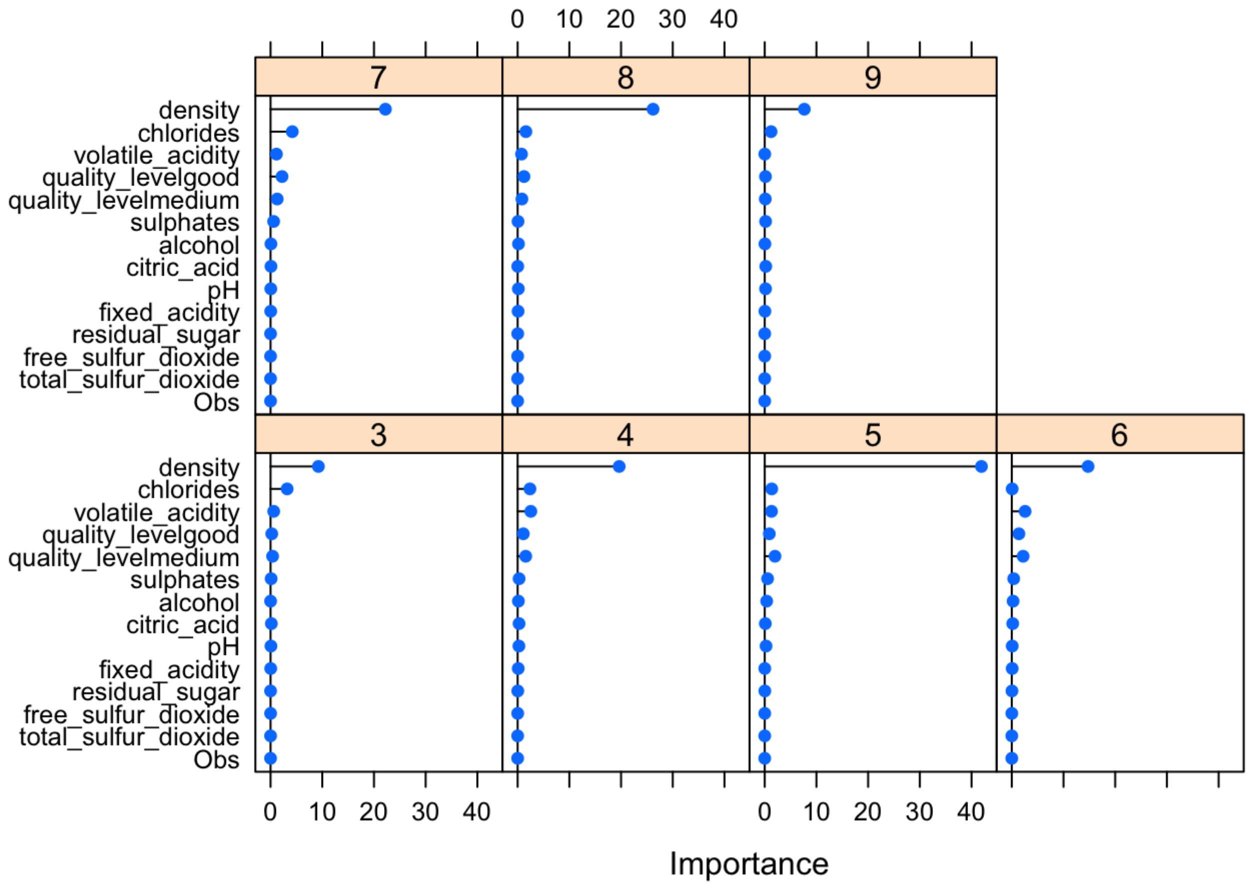
```
# print results
print(ridgeReg1)
```

```
## glmnet
##
## 2938 samples
##   13 predictor
##    7 classes: '3', '4', '5', '6', '7', '8', '9'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 2645, 2645, 2645, 2644, 2644, 2644, ...
## Resampling results across tuning parameters:
##
##   lambda    Accuracy   Kappa
##   0.000100  0.7151884  0.5642168
##   0.125075  0.6875257  0.5115899
##   0.250050  0.6754056  0.4874629
##   0.375025  0.6628432  0.4629472
##   0.500000  0.6589265  0.4533713
##
## Tuning parameter 'alpha' was held constant at a value of 0
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0 and lambda = 1e-04.
```

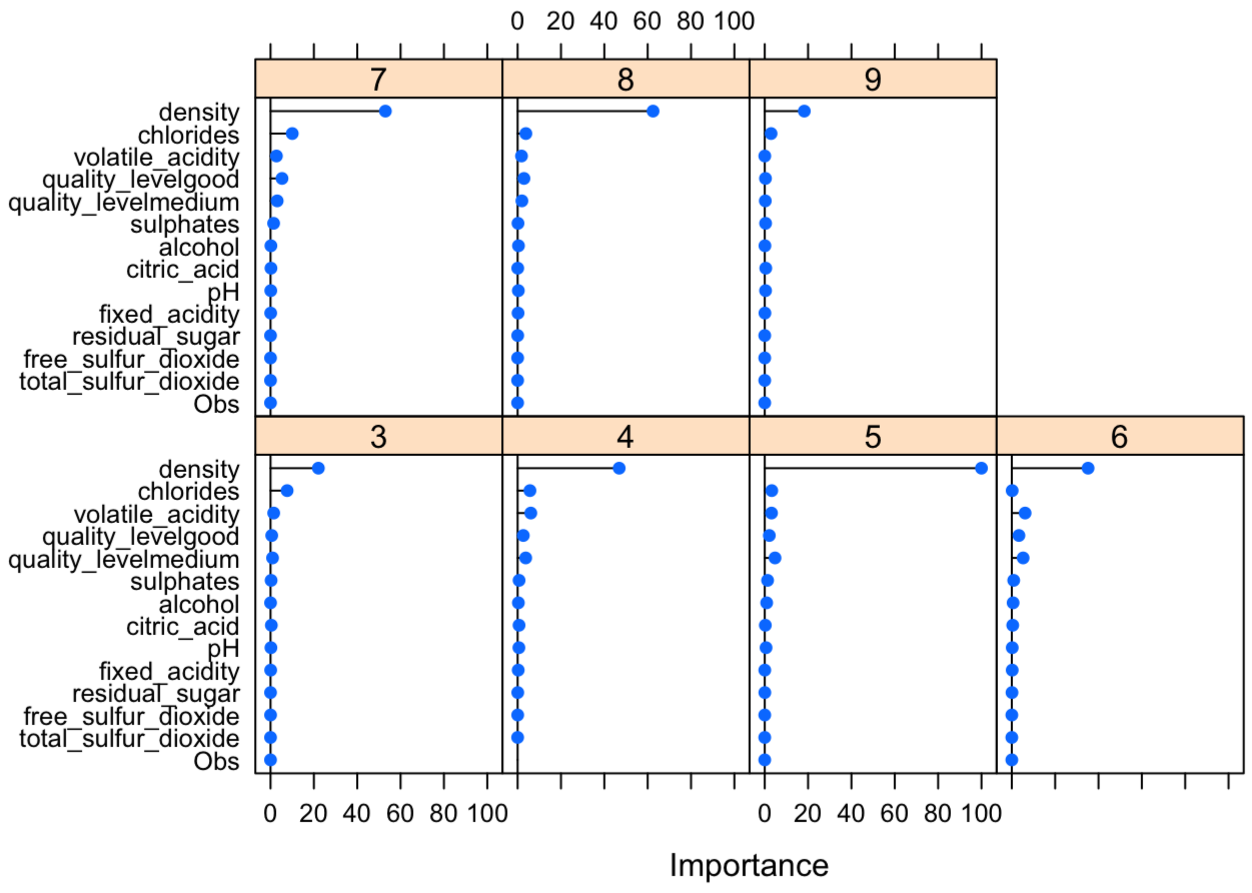
```
# plot results
plot(ridgeReg1)
```



```
plot(varImp(ridgeReg1, scale = FALSE))
```



```
plot(varImp(ridgeReg1, scale = TRUE))
```



```
PredictRidge1 <- predict(ridgeReg1, test)
confusionMatrix(PredictRidge1, test$quality)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   4  46   0   0   0   0   0
##          5   0  15 278 132   0   0   0
##          6   2  12 310 750   0   0   0
##          7   0   0   0   0 337  73   1
##          8   0   0   0   0   0   0   0
##          9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.7199
##                  95% CI : (0.6994, 0.7397)
##     No Information Rate : 0.45
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.57
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.63014   0.4728   0.8503   1.0000  0.00000
## Specificity          1.000000  0.99788   0.8929   0.6994   0.9544  1.00000
## Pos Pred Value            NaN  0.92000   0.6541   0.6983   0.8200      NaN
## Neg Pred Value       0.996939  0.98586   0.7980   0.8510   1.0000  0.96276
## Prevalence           0.003061  0.03724   0.3000   0.4500   0.1719  0.03724
## Detection Rate       0.000000  0.02347   0.1418   0.3827   0.1719  0.00000
## Detection Prevalence 0.000000  0.02551   0.2168   0.5480   0.2097  0.00000
## Balanced Accuracy    0.500000  0.81401   0.6828   0.7749   0.9772  0.50000
##                       Class: 9
## Sensitivity          0.0000000
## Specificity          1.0000000
## Pos Pred Value             NaN
## Neg Pred Value       0.9994898
## Prevalence           0.0005102
## Detection Rate       0.0000000
## Detection Prevalence 0.0000000
## Balanced Accuracy    0.5000000
```

We have run the Ridge model on Quality variable. The accuracy is 71.99%

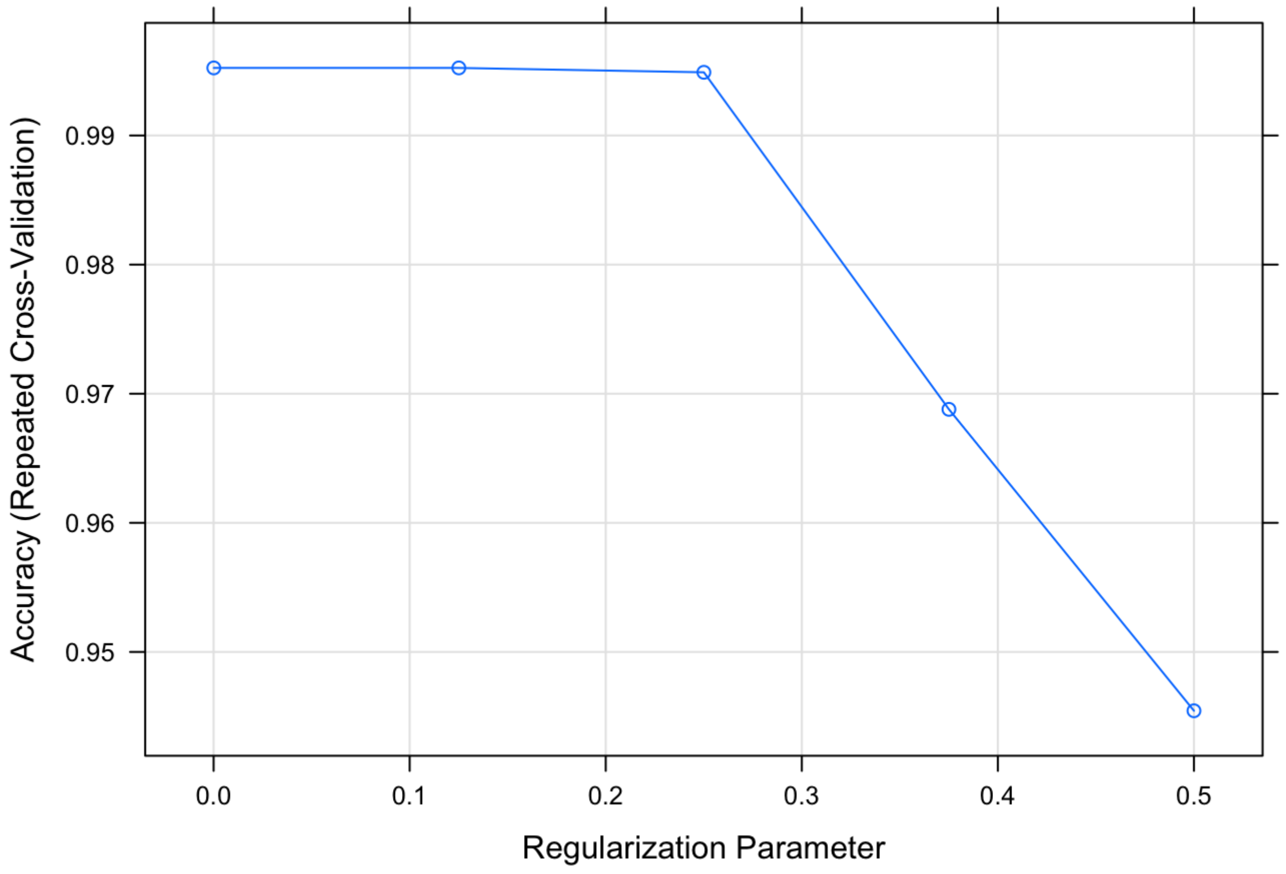
```
set.seed(123)
ridgeReg2 <- train(quality_level~., train, method = 'glmnet',
                  tuneGrid = expand.grid(alpha = 0,
                                         lambda = seq(0.0001, 0.5, length = 5)),
                  trControl = trainControl(method = "repeatedcv",
                                           number = 10, repeats = 10,
                                           verboseIter = TRUE))
```

```
## + Fold01.Rep01: alpha=0, lambda=0.5
## - Fold01.Rep01: alpha=0, lambda=0.5
## + Fold02.Rep01: alpha=0, lambda=0.5
## - Fold02.Rep01: alpha=0, lambda=0.5
## + Fold03.Rep01: alpha=0, lambda=0.5
## - Fold03.Rep01: alpha=0, lambda=0.5
## + Fold04.Rep01: alpha=0, lambda=0.5
## - Fold04.Rep01: alpha=0, lambda=0.5
## + Fold05.Rep01: alpha=0, lambda=0.5
## - Fold05.Rep01: alpha=0, lambda=0.5
## + Fold06.Rep01: alpha=0, lambda=0.5
## - Fold06.Rep01: alpha=0, lambda=0.5
## + Fold07.Rep01: alpha=0, lambda=0.5
## - Fold07.Rep01: alpha=0, lambda=0.5
## + Fold08.Rep01: alpha=0, lambda=0.5
## - Fold08.Rep01: alpha=0, lambda=0.5
## + Fold09.Rep01: alpha=0, lambda=0.5
## - Fold09.Rep01: alpha=0, lambda=0.5
## + Fold10.Rep01: alpha=0, lambda=0.5
## - Fold10.Rep01: alpha=0, lambda=0.5
## + Fold01.Rep02: alpha=0, lambda=0.5
## - Fold01.Rep02: alpha=0, lambda=0.5
## + Fold02.Rep02: alpha=0, lambda=0.5
## - Fold02.Rep02: alpha=0, lambda=0.5
## + Fold03.Rep02: alpha=0, lambda=0.5
## - Fold03.Rep02: alpha=0, lambda=0.5
## + Fold04.Rep02: alpha=0, lambda=0.5
## - Fold04.Rep02: alpha=0, lambda=0.5
## + Fold05.Rep02: alpha=0, lambda=0.5
## - Fold05.Rep02: alpha=0, lambda=0.5
## + Fold06.Rep02: alpha=0, lambda=0.5
## - Fold06.Rep02: alpha=0, lambda=0.5
## + Fold07.Rep02: alpha=0, lambda=0.5
## - Fold07.Rep02: alpha=0, lambda=0.5
## + Fold08.Rep02: alpha=0, lambda=0.5
## - Fold08.Rep02: alpha=0, lambda=0.5
## + Fold09.Rep02: alpha=0, lambda=0.5
## - Fold09.Rep02: alpha=0, lambda=0.5
## + Fold10.Rep02: alpha=0, lambda=0.5
## - Fold10.Rep02: alpha=0, lambda=0.5
## + Fold01.Rep03: alpha=0, lambda=0.5
## - Fold01.Rep03: alpha=0, lambda=0.5
## + Fold02.Rep03: alpha=0, lambda=0.5
## - Fold02.Rep03: alpha=0, lambda=0.5
## + Fold03.Rep03: alpha=0, lambda=0.5
## - Fold03.Rep03: alpha=0, lambda=0.5
## + Fold04.Rep03: alpha=0, lambda=0.5
## - Fold04.Rep03: alpha=0, lambda=0.5
## + Fold05.Rep03: alpha=0, lambda=0.5
## - Fold05.Rep03: alpha=0, lambda=0.5
## + Fold06.Rep03: alpha=0, lambda=0.5
## - Fold06.Rep03: alpha=0, lambda=0.5
## + Fold07.Rep03: alpha=0, lambda=0.5
## - Fold07.Rep03: alpha=0, lambda=0.5
## + Fold08.Rep03: alpha=0, lambda=0.5
## - Fold08.Rep03: alpha=0, lambda=0.5
## + Fold09.Rep03: alpha=0, lambda=0.5
## - Fold09.Rep03: alpha=0, lambda=0.5
## + Fold10.Rep03: alpha=0, lambda=0.5
## - Fold10.Rep03: alpha=0, lambda=0.5
## + Fold01.Rep04: alpha=0, lambda=0.5
## - Fold01.Rep04: alpha=0, lambda=0.5
## + Fold02.Rep04: alpha=0, lambda=0.5
## - Fold02.Rep04: alpha=0, lambda=0.5
## + Fold03.Rep04: alpha=0, lambda=0.5
## - Fold03.Rep04: alpha=0, lambda=0.5
## + Fold04.Rep04: alpha=0, lambda=0.5
## - Fold04.Rep04: alpha=0, lambda=0.5
## + Fold05.Rep04: alpha=0, lambda=0.5
## - Fold05.Rep04: alpha=0, lambda=0.5
## + Fold06.Rep04: alpha=0, lambda=0.5
## - Fold06.Rep04: alpha=0, lambda=0.5
## + Fold07.Rep04: alpha=0, lambda=0.5
## - Fold07.Rep04: alpha=0, lambda=0.5
## + Fold08.Rep04: alpha=0, lambda=0.5
## - Fold08.Rep04: alpha=0, lambda=0.5
## + Fold09.Rep04: alpha=0, lambda=0.5
## - Fold09.Rep04: alpha=0, lambda=0.5
## + Fold10.Rep04: alpha=0, lambda=0.5
## - Fold10.Rep04: alpha=0, lambda=0.5
## + Fold01.Rep05: alpha=0, lambda=0.5
## - Fold01.Rep05: alpha=0, lambda=0.5
## + Fold02.Rep05: alpha=0, lambda=0.5
## - Fold02.Rep05: alpha=0, lambda=0.5
## + Fold03.Rep05: alpha=0, lambda=0.5
## - Fold03.Rep05: alpha=0, lambda=0.5
## + Fold04.Rep05: alpha=0, lambda=0.5
## - Fold04.Rep05: alpha=0, lambda=0.5
## + Fold05.Rep05: alpha=0, lambda=0.5
## - Fold05.Rep05: alpha=0, lambda=0.5
## + Fold06.Rep05: alpha=0, lambda=0.5
## - Fold06.Rep05: alpha=0, lambda=0.5
## + Fold07.Rep05: alpha=0, lambda=0.5
## - Fold07.Rep05: alpha=0, lambda=0.5
## + Fold08.Rep05: alpha=0, lambda=0.5
## - Fold08.Rep05: alpha=0, lambda=0.5
## + Fold09.Rep05: alpha=0, lambda=0.5
## - Fold09.Rep05: alpha=0, lambda=0.5
## + Fold10.Rep05: alpha=0, lambda=0.5
## - Fold10.Rep05: alpha=0, lambda=0.5
## + Fold01.Rep06: alpha=0, lambda=0.5
## - Fold01.Rep06: alpha=0, lambda=0.5
## + Fold02.Rep06: alpha=0, lambda=0.5
## - Fold02.Rep06: alpha=0, lambda=0.5
## + Fold03.Rep06: alpha=0, lambda=0.5
## - Fold03.Rep06: alpha=0, lambda=0.5
## + Fold04.Rep06: alpha=0, lambda=0.5
## - Fold04.Rep06: alpha=0, lambda=0.5
## + Fold05.Rep06: alpha=0, lambda=0.5
## - Fold05.Rep06: alpha=0, lambda=0.5
## + Fold06.Rep06: alpha=0, lambda=0.5
## - Fold06.Rep06: alpha=0, lambda=0.5
## + Fold07.Rep06: alpha=0, lambda=0.5
## - Fold07.Rep06: alpha=0, lambda=0.5
## + Fold08.Rep06: alpha=0, lambda=0.5
## - Fold08.Rep06: alpha=0, lambda=0.5
## + Fold09.Rep06: alpha=0, lambda=0.5
## - Fold09.Rep06: alpha=0, lambda=0.5
## + Fold10.Rep06: alpha=0, lambda=0.5
## - Fold10.Rep06: alpha=0, lambda=0.5
## + Fold01.Rep07: alpha=0, lambda=0.5
## - Fold01.Rep07: alpha=0, lambda=0.5
## + Fold02.Rep07: alpha=0, lambda=0.5
## - Fold02.Rep07: alpha=0, lambda=0.5
## + Fold03.Rep07: alpha=0, lambda=0.5
## - Fold03.Rep07: alpha=0, lambda=0.5
## + Fold04.Rep07: alpha=0, lambda=0.5
## - Fold04.Rep07: alpha=0, lambda=0.5
## + Fold05.Rep07: alpha=0, lambda=0.5
## - Fold05.Rep07: alpha=0, lambda=0.5
## + Fold06.Rep07: alpha=0, lambda=0.5
## - Fold06.Rep07: alpha=0, lambda=0.5
## + Fold07.Rep07: alpha=0, lambda=0.5
## - Fold07.Rep07: alpha=0, lambda=0.5
## + Fold08.Rep07: alpha=0, lambda=0.5
## - Fold08.Rep07: alpha=0, lambda=0.5
## + Fold09.Rep07: alpha=0, lambda=0.5
## - Fold09.Rep07: alpha=0, lambda=0.5
## + Fold10.Rep07: alpha=0, lambda=0.5
## - Fold10.Rep07: alpha=0, lambda=0.5
## + Fold01.Rep08: alpha=0, lambda=0.5
## - Fold01.Rep08: alpha=0, lambda=0.5
## + Fold02.Rep08: alpha=0, lambda=0.5
## - Fold02.Rep08: alpha=0, lambda=0.5
## + Fold03.Rep08: alpha=0, lambda=0.5
## - Fold03.Rep08: alpha=0, lambda=0.5
## + Fold04.Rep08: alpha=0, lambda=0.5
## - Fold04.Rep08: alpha=0, lambda=0.5
## + Fold05.Rep08: alpha=0, lambda=0.5
## - Fold05.Rep08: alpha=0, lambda=0.5
## + Fold06.Rep08: alpha=0, lambda=0.5
## - Fold06.Rep08: alpha=0, lambda=0.5
## + Fold07.Rep08: alpha=0, lambda=0.5
## - Fold07.Rep08: alpha=0, lambda=0.5
## + Fold08.Rep08: alpha=0, lambda=0.5
## - Fold08.Rep08: alpha=0, lambda=0.5
## + Fold09.Rep08: alpha=0, lambda=0.5
## - Fold09.Rep08: alpha=0, lambda=0.5
## + Fold10.Rep08: alpha=0, lambda=0.5
## - Fold10.Rep08: alpha=0, lambda=0.5
## + Fold01.Rep09: alpha=0, lambda=0.5
## - Fold01.Rep09: alpha=0, lambda=0.5
## + Fold02.Rep09: alpha=0, lambda=0.5
## - Fold02.Rep09: alpha=0, lambda=0.5
## + Fold03.Rep09: alpha=0, lambda=0.5
## - Fold03.Rep09: alpha=0, lambda=0.5
## + Fold04.Rep09: alpha=0, lambda=0.5
## - Fold04.Rep09: alpha=0, lambda=0.5
## + Fold05.Rep09: alpha=0, lambda=0.5
## - Fold05.Rep09: alpha=0, lambda=0.5
## + Fold06.Rep09: alpha=0, lambda=0.5
## - Fold06.Rep09: alpha=0, lambda=0.5
## + Fold07.Rep09: alpha=0, lambda=0.5
## - Fold07.Rep09: alpha=0, lambda=0.5
## + Fold08.Rep09: alpha=0, lambda=0.5
## - Fold08.Rep09: alpha=0, lambda=0.5
## + Fold09.Rep09: alpha=0, lambda=0.5
## - Fold09.Rep09: alpha=0, lambda=0.5
## + Fold10.Rep09: alpha=0, lambda=0.5
## - Fold10.Rep09: alpha=0, lambda=0.5
## + Fold01.Rep10: alpha=0, lambda=0.5
## - Fold01.Rep10: alpha=0, lambda=0.5
## + Fold02.Rep10: alpha=0, lambda=0.5
## - Fold02.Rep10: alpha=0, lambda=0.5
## + Fold03.Rep10: alpha=0, lambda=0.5
## - Fold03.Rep10: alpha=0, lambda=0.5
## + Fold04.Rep10: alpha=0, lambda=0.5
## - Fold04.Rep10: alpha=0, lambda=0.5
## + Fold05.Rep10: alpha=0, lambda=0.5
## - Fold05.Rep10: alpha=0, lambda=0.5
## + Fold06.Rep10: alpha=0, lambda=0.5
## - Fold06.Rep10: alpha=0, lambda=0.5
## + Fold07.Rep10: alpha=0, lambda=0.5
## - Fold07.Rep10: alpha=0, lambda=0.5
## + Fold08.Rep10: alpha=0, lambda=0.5
## - Fold08.Rep10: alpha=0, lambda=0.5
## + Fold09.Rep10: alpha=0, lambda=0.5
## - Fold09.Rep10: alpha=0, lambda=0.5
## + Fold10.Rep10: alpha=0, lambda=0.5
## - Fold10.Rep10: alpha=0, lambda=0.5
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0, lambda = 0.125 on full training set
```

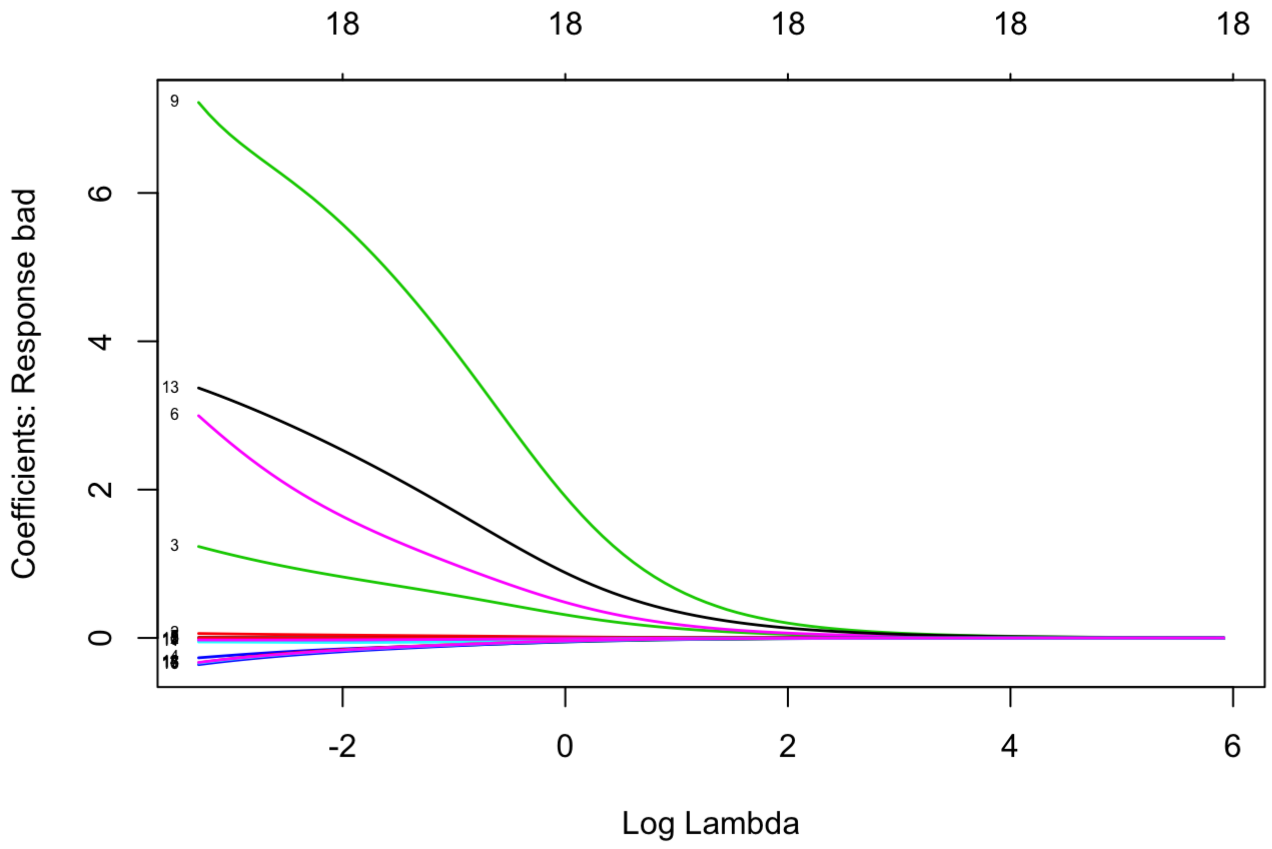
```
# print results
print(ridgeReg2)
```

```
## glmnet
##
## 2938 samples
##   13 predictor
##    3 classes: 'bad', 'good', 'medium'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 2645, 2645, 2644, 2644, 2645, 2643, ...
## Resampling results across tuning parameters:
##
##   lambda    Accuracy   Kappa
##   0.000100  0.9952337  0.9878288
##   0.125075  0.9952337  0.9878288
##   0.250050  0.9948934  0.9869549
##   0.375025  0.9687897  0.9167402
##   0.500000  0.9454428  0.8490061
##
## Tuning parameter 'alpha' was held constant at a value of 0
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0 and lambda = 0.125075.
```

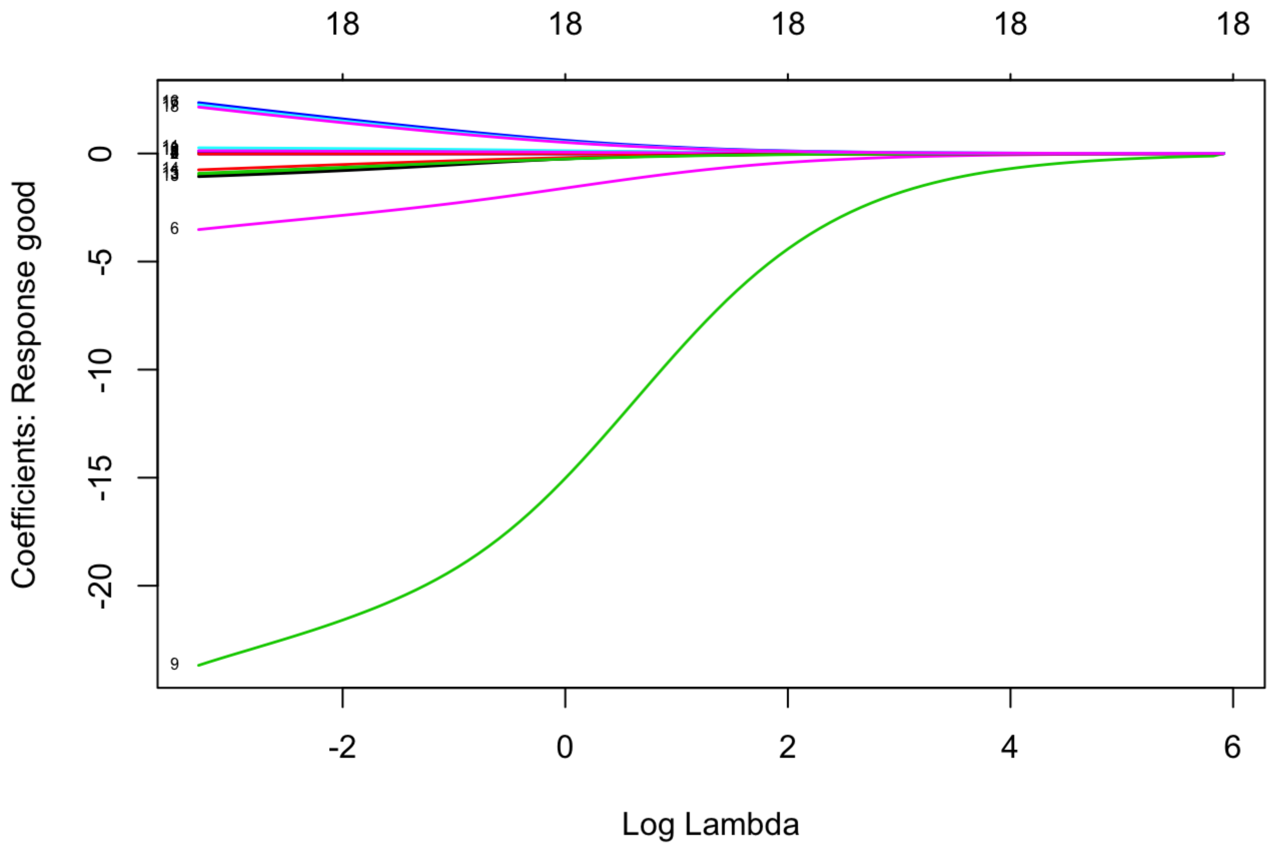
```
# plot results
plot(ridgeReg2)
```



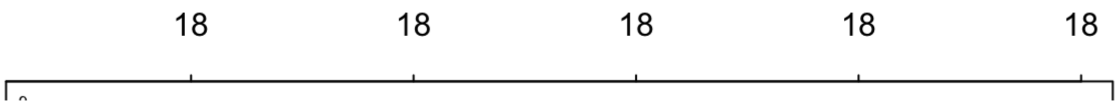
```
plot(ridgeReg2$finalModel, xvar = 'lambda', lwd =1.4, label = TRUE)
```



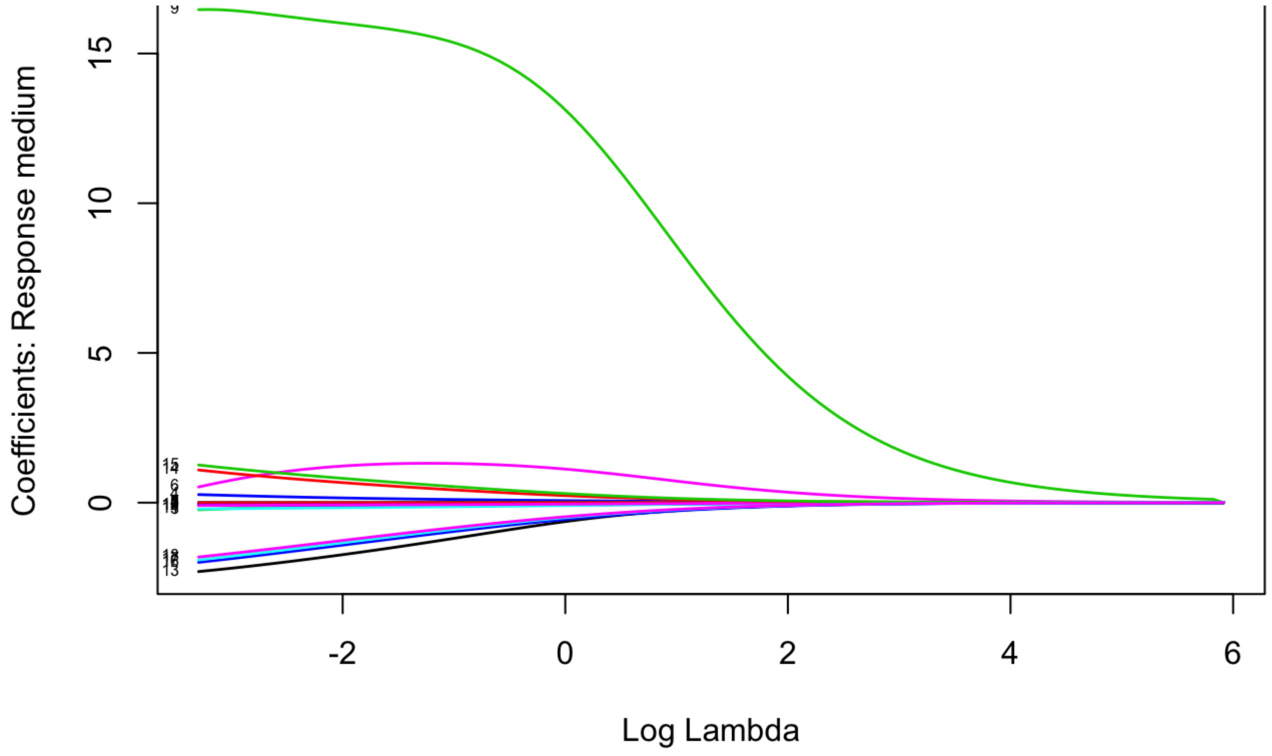
18
18
18
18
18
Coefficients: Response bad
Log Lambda
-2
0
2
4
6
0
2
4
6
9
13
6
3



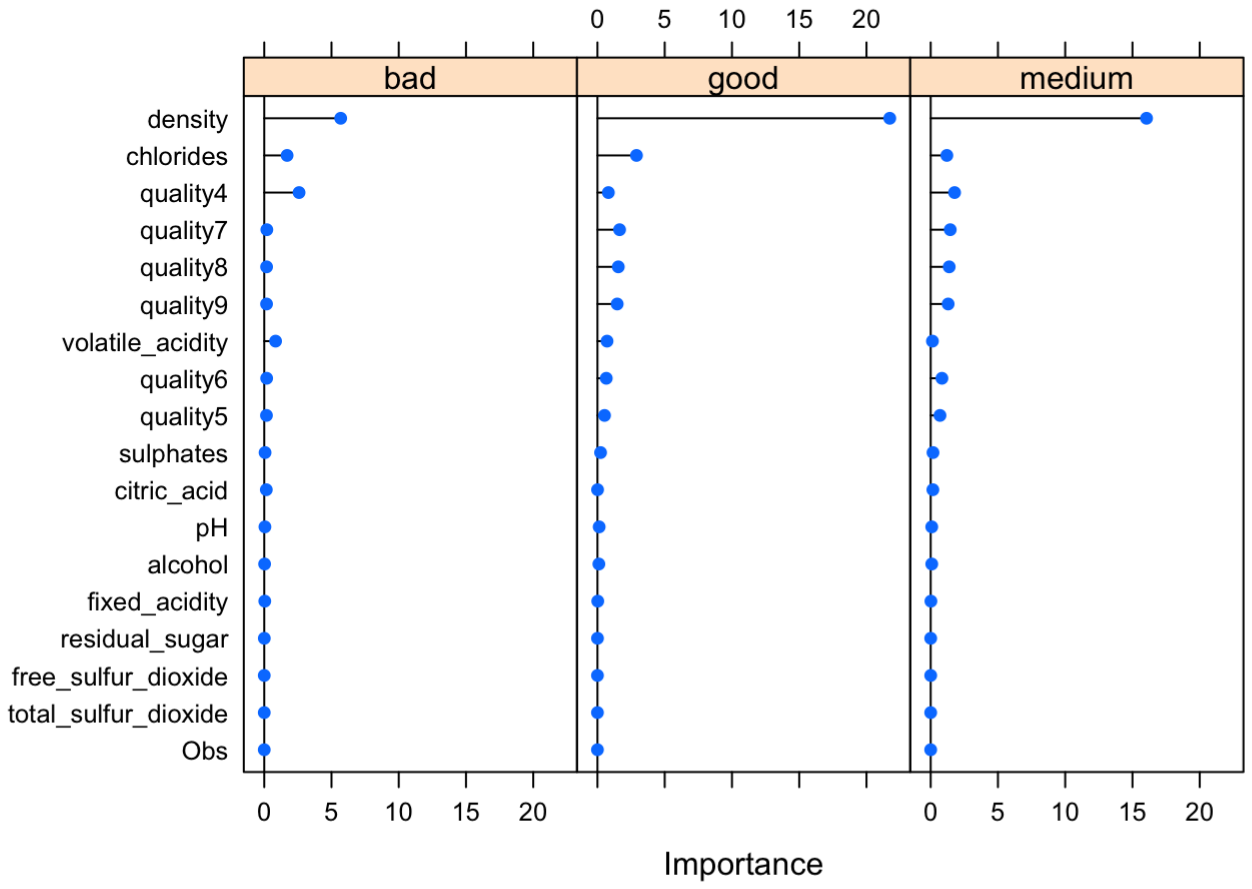
18
18
18
18
18
Coefficients: Response good
Log Lambda
-2
0
2
4
6
0
-5
-10
-15
-20
16
18
6
9



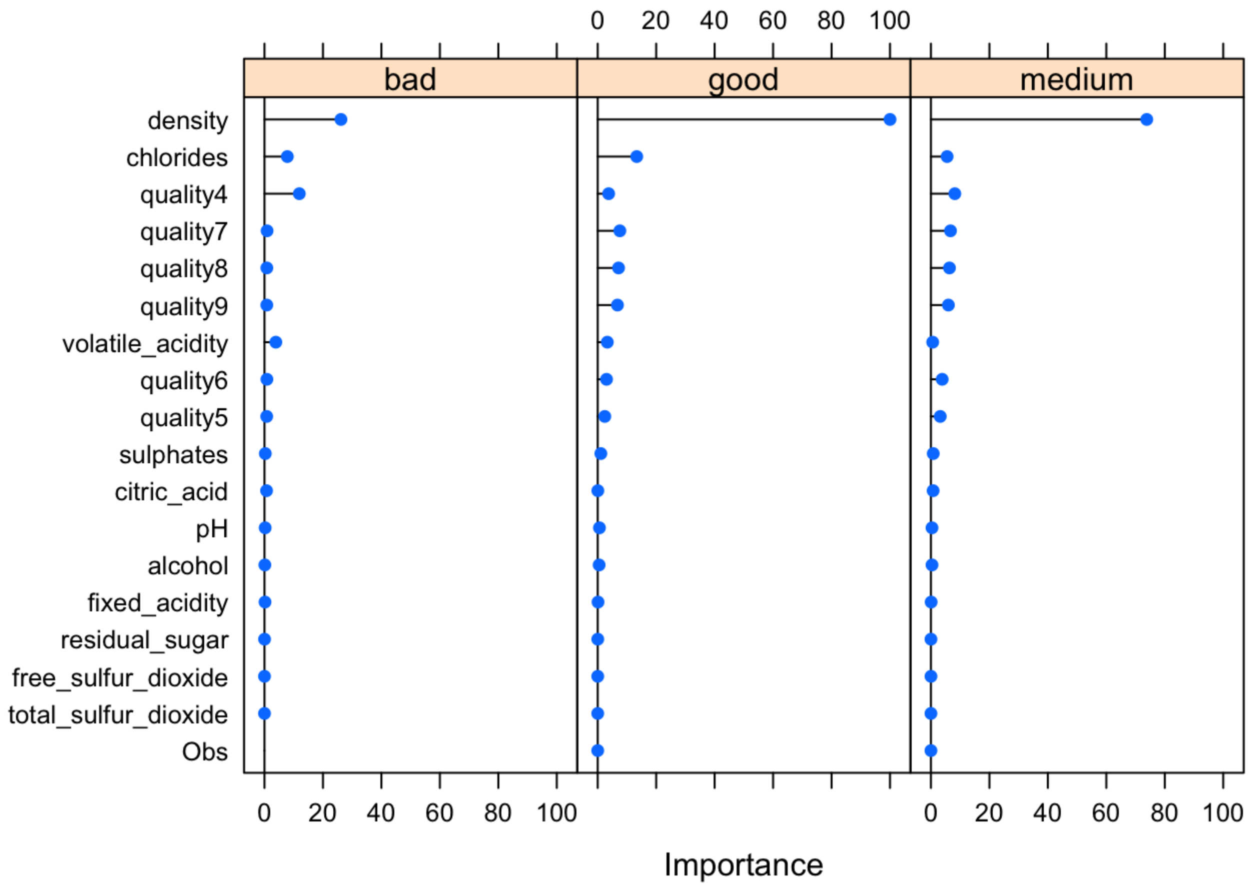
18
18
18
18
18



```
plot(varImp(ridgeReg2, scale = FALSE))
```



```
plot(varImp(ridgeReg2, scale = TRUE))
```



```
PredictRidge2 <- predict(ridgeReg2, test)
confusionMatrix(PredictRidge2, test$quality_level)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  bad good medium
##     bad      73    0      0
##     good      0  411      0
##     medium    6    0   1470
## 
## Overall Statistics
## 
##                Accuracy : 0.9969
##                  95% CI : (0.9933, 0.9989)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9921
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: bad Class: good Class: medium
## Sensitivity             0.92405      1.0000        1.0000
## Specificity             1.00000      1.0000        0.9878
## Pos Pred Value          1.00000      1.0000        0.9959
## Neg Pred Value          0.99682      1.0000        1.0000
## Prevalence              0.04031      0.2097        0.7500
## Detection Rate          0.03724      0.2097        0.7500
## Detection Prevalence    0.03724      0.2097        0.7531
## Balanced Accuracy       0.96203      1.0000        0.9939
```

We have run the Ridge model on Quality Level variable. The accuracy is 99.69%

```
set.seed(123)
lassoReg1 <- train(quality~., train, method = 'glmnet',
                  tuneGrid = expand.grid(alpha = 1,
                                         lambda = seq(0.0001, 0.5, length = 5)),
                  trControl = trainControl(method = "repeatedcv",
                                           number = 10, repeats = 5,
                                           verboseIter = TRUE))
```

```
## + Fold01.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep1: alpha=1, lambda=0.5
## + Fold02.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold02.Rep1: alpha=1, lambda=0.5
## + Fold03.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep1: alpha=1, lambda=0.5
## + Fold04.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep1: alpha=1, lambda=0.5
## + Fold05.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep1: alpha=1, lambda=0.5
## + Fold06.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep1: alpha=1, lambda=0.5
## + Fold07.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep1: alpha=1, lambda=0.5
## + Fold08.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep1: alpha=1, lambda=0.5
## + Fold09.Rep1: alpha=1, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep1: alpha=1, lambda=0.5
## + Fold10.Rep1: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep1: alpha=1, lambda=0.5
## + Fold01.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep2: alpha=1, lambda=0.5
## + Fold02.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep2: alpha=1, lambda=0.5
## + Fold03.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep2: alpha=1, lambda=0.5
## + Fold04.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep2: alpha=1, lambda=0.5
## + Fold05.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold05.Rep2: alpha=1, lambda=0.5
## + Fold06.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep2: alpha=1, lambda=0.5
## + Fold07.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep2: alpha=1, lambda=0.5
## + Fold08.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep2: alpha=1, lambda=0.5
## + Fold09.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep2: alpha=1, lambda=0.5
## + Fold10.Rep2: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep2: alpha=1, lambda=0.5
## + Fold01.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep3: alpha=1, lambda=0.5
## + Fold02.Rep3: alpha=1, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep3: alpha=1, lambda=0.5
## + Fold03.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep3: alpha=1, lambda=0.5
## + Fold04.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep3: alpha=1, lambda=0.5
## + Fold05.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep3: alpha=1, lambda=0.5
## + Fold06.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep3: alpha=1, lambda=0.5
## + Fold07.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep3: alpha=1, lambda=0.5
## + Fold08.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold08.Rep3: alpha=1, lambda=0.5
## + Fold09.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep3: alpha=1, lambda=0.5
## + Fold10.Rep3: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep3: alpha=1, lambda=0.5
## + Fold01.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold01.Rep4: alpha=1, lambda=0.5
## + Fold02.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep4: alpha=1, lambda=0.5
## + Fold03.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep4: alpha=1, lambda=0.5
## + Fold04.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep4: alpha=1, lambda=0.5
## + Fold05.Rep4: alpha=1, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep4: alpha=1, lambda=0.5
## + Fold06.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep4: alpha=1, lambda=0.5
## + Fold07.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep4: alpha=1, lambda=0.5
## + Fold08.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep4: alpha=1, lambda=0.5
## + Fold09.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold09.Rep4: alpha=1, lambda=0.5
## + Fold10.Rep4: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold10.Rep4: alpha=1, lambda=0.5
## + Fold01.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
## - Fold01.Rep5: alpha=1, lambda=0.5
## + Fold02.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold02.Rep5: alpha=1, lambda=0.5
## + Fold03.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold03.Rep5: alpha=1, lambda=0.5
## + Fold04.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold04.Rep5: alpha=1, lambda=0.5
## + Fold05.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold05.Rep5: alpha=1, lambda=0.5
## + Fold06.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold06.Rep5: alpha=1, lambda=0.5
## + Fold07.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold07.Rep5: alpha=1, lambda=0.5
## + Fold08.Rep5: alpha=1, lambda=0.5
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

```
## - Fold08.Rep5: alpha=1, lambda=0.5
## + Fold09.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

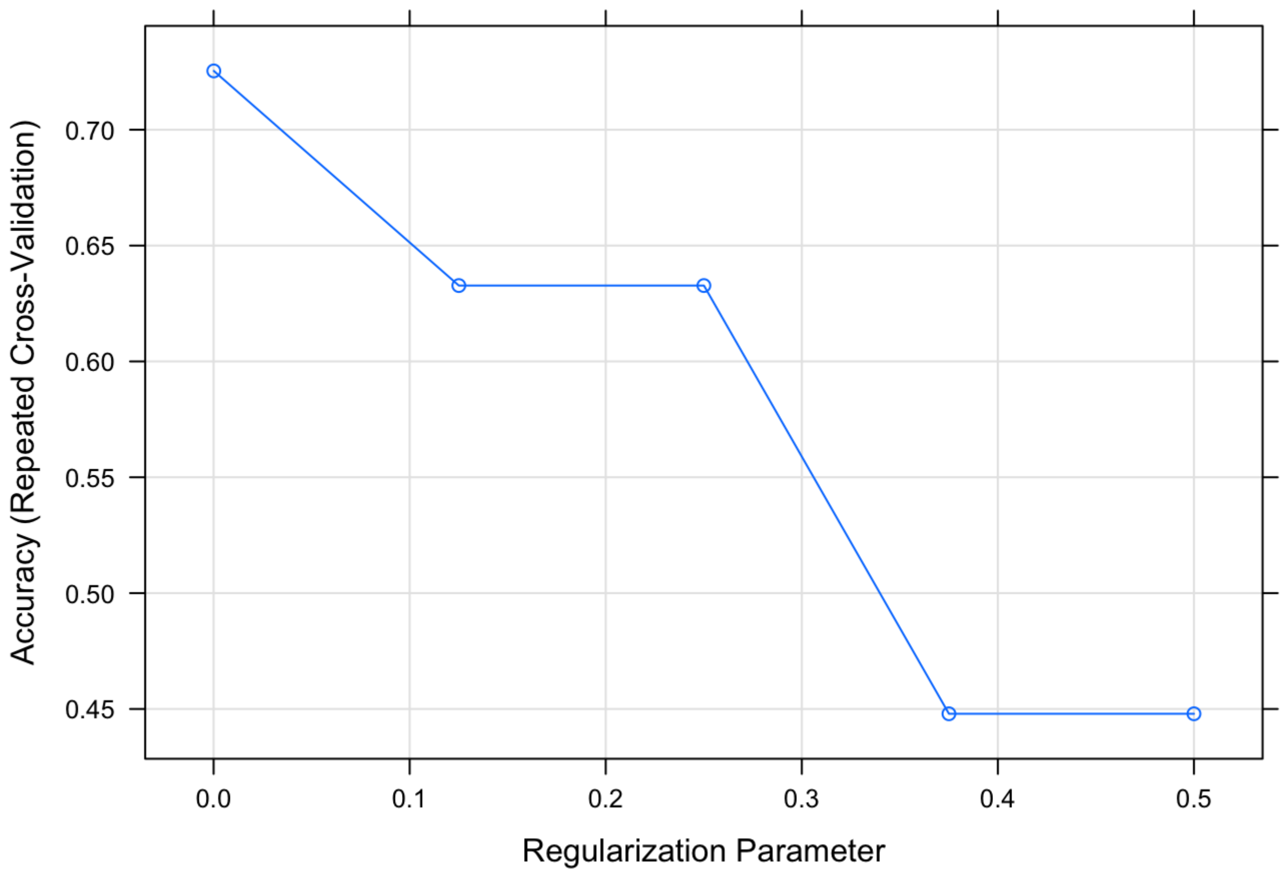
```
## - Fold09.Rep5: alpha=1, lambda=0.5
## + Fold10.Rep5: alpha=1, lambda=0.5
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

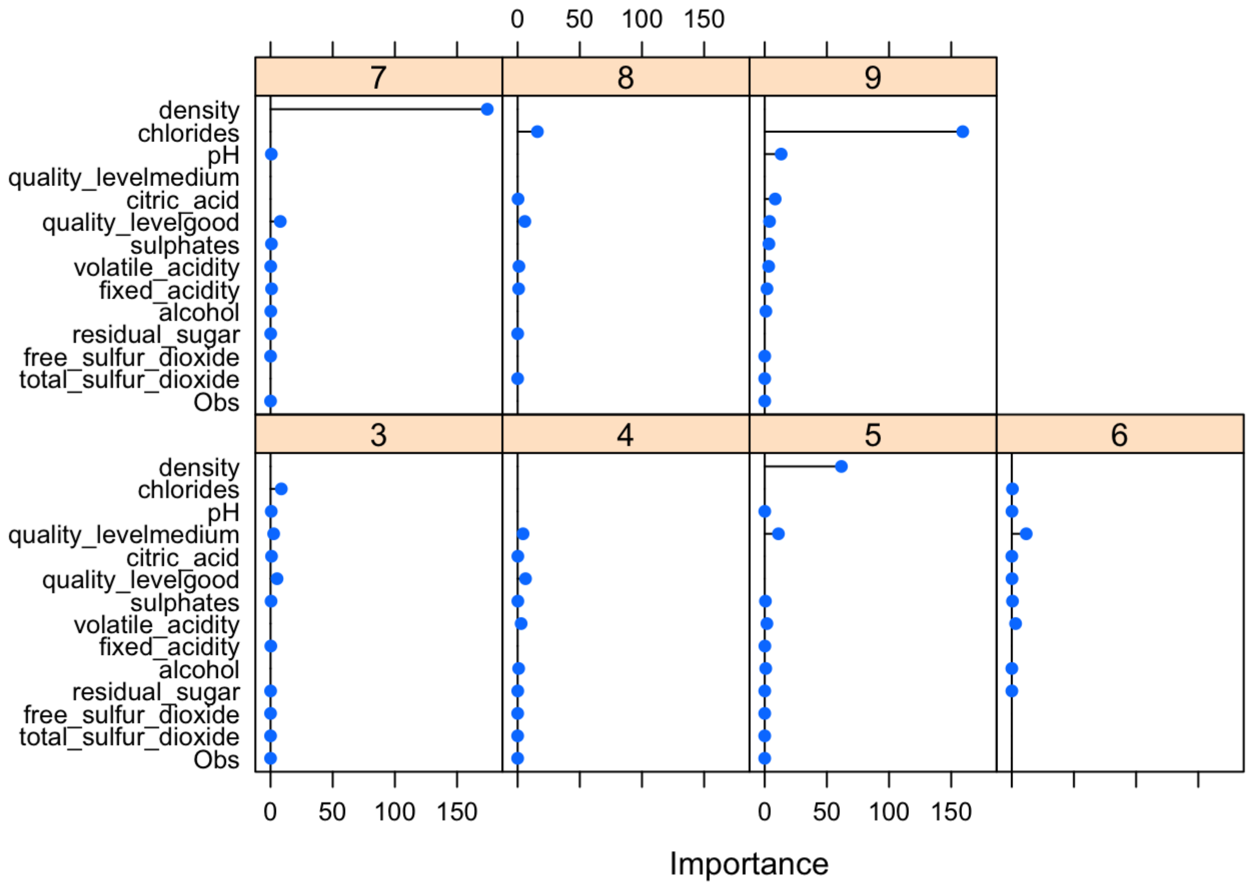
```
## - Fold10.Rep5: alpha=1, lambda=0.5
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 1, lambda = 1e-04 on full training set
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
## one multinomial or binomial class has fewer than 8 observations; dangerous
## ground
```

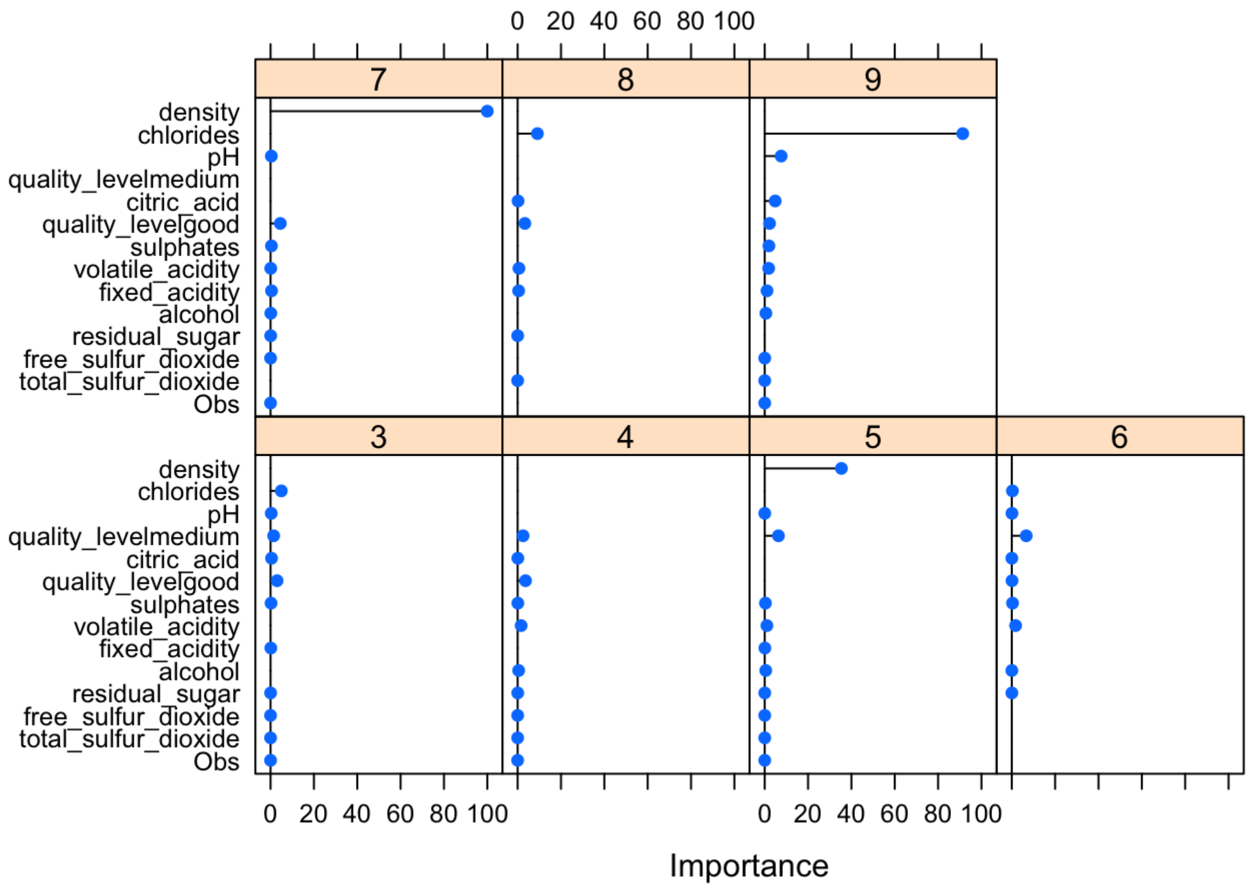
```
# plot results
plot(lassoReg1)
```



```
plot(varImp(lassoReg1, scale = FALSE))
```



```
plot(varImp(lassoReg1, scale = TRUE))
```



```
PredictLasso1 <- predict(lassoReg1, test)
confusionMatrix(PredictLasso1, test$quality)
```

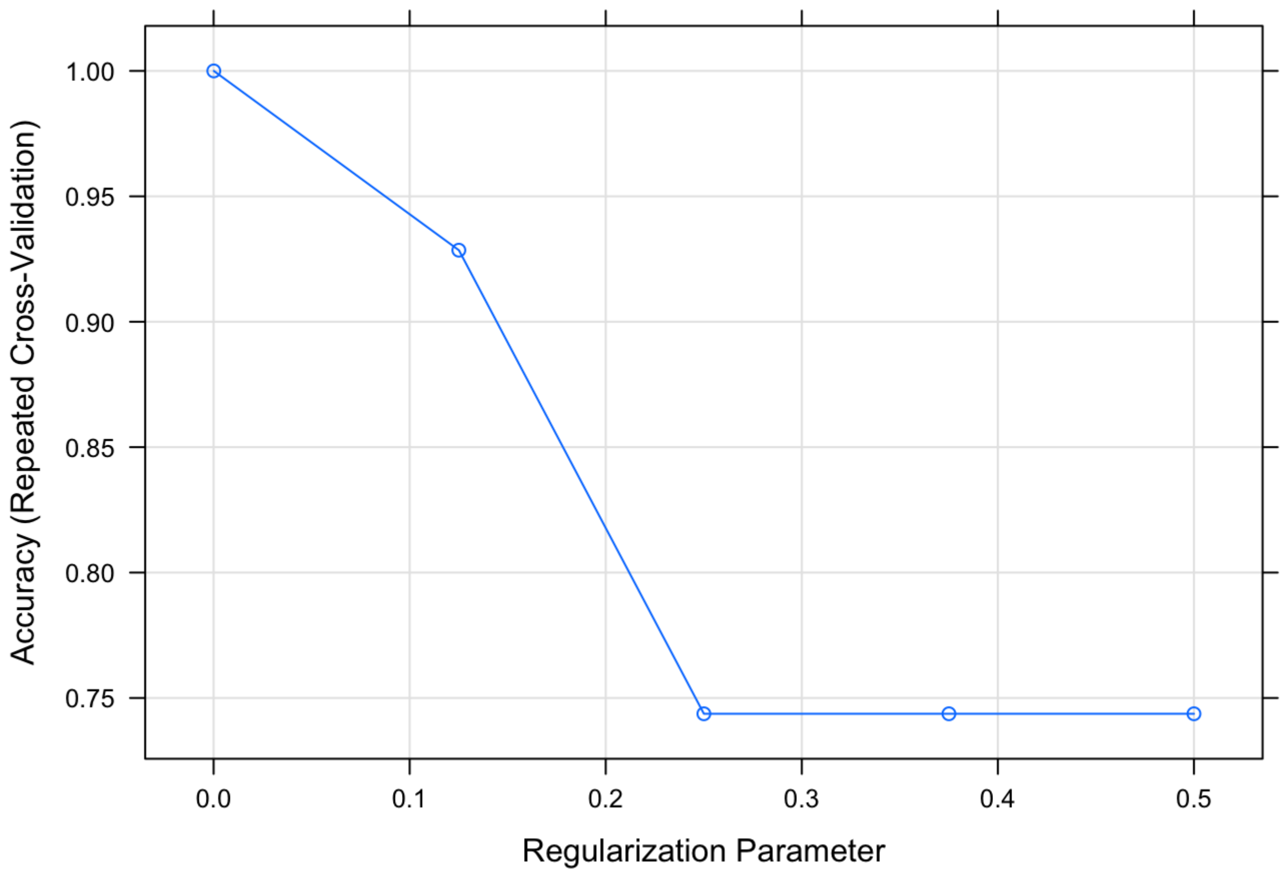
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   2   0   0   0   0   0
##          4   6  71   0   0   0   0   0
##          5   0   0 310 160   0   0   0
##          6   0   0 278 721   0   0   0
##          7   0   0   0   0 337  73   1
##          8   0   0   0   0   0   0   0
##          9   0   0   0   1   0   0   0
## 
## Overall Statistics
## 
##                Accuracy : 0.7342
##                  95% CI : (0.714, 0.7536)
##     No Information Rate : 0.45
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.598
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.97260   0.5272   0.8175   1.0000  0.00000
## Specificity          0.998976  0.99682   0.8834   0.7421   0.9544  1.00000
## Pos Pred Value       0.000000  0.92208   0.6596   0.7217   0.8200      NaN
## Neg Pred Value       0.996936  0.99894   0.8134   0.8325   1.0000  0.96276
## Prevalence           0.003061  0.03724   0.3000   0.4500   0.1719  0.03724
## Detection Rate       0.000000  0.03622   0.1582   0.3679   0.1719  0.00000
## Detection Prevalence 0.001020  0.03929   0.2398   0.5097   0.2097  0.00000
## Balanced Accuracy    0.499488  0.98471   0.7053   0.7798   0.9772  0.50000
##                       Class: 9
## Sensitivity          0.0000000
## Specificity          0.9994895
## Pos Pred Value       0.0000000
## Neg Pred Value       0.9994895
## Prevalence           0.0005102
## Detection Rate       0.0000000
## Detection Prevalence 0.0005102
## Balanced Accuracy    0.4997448
```

We have run the Lasso model on Quality variable. The accuracy is 73.42%

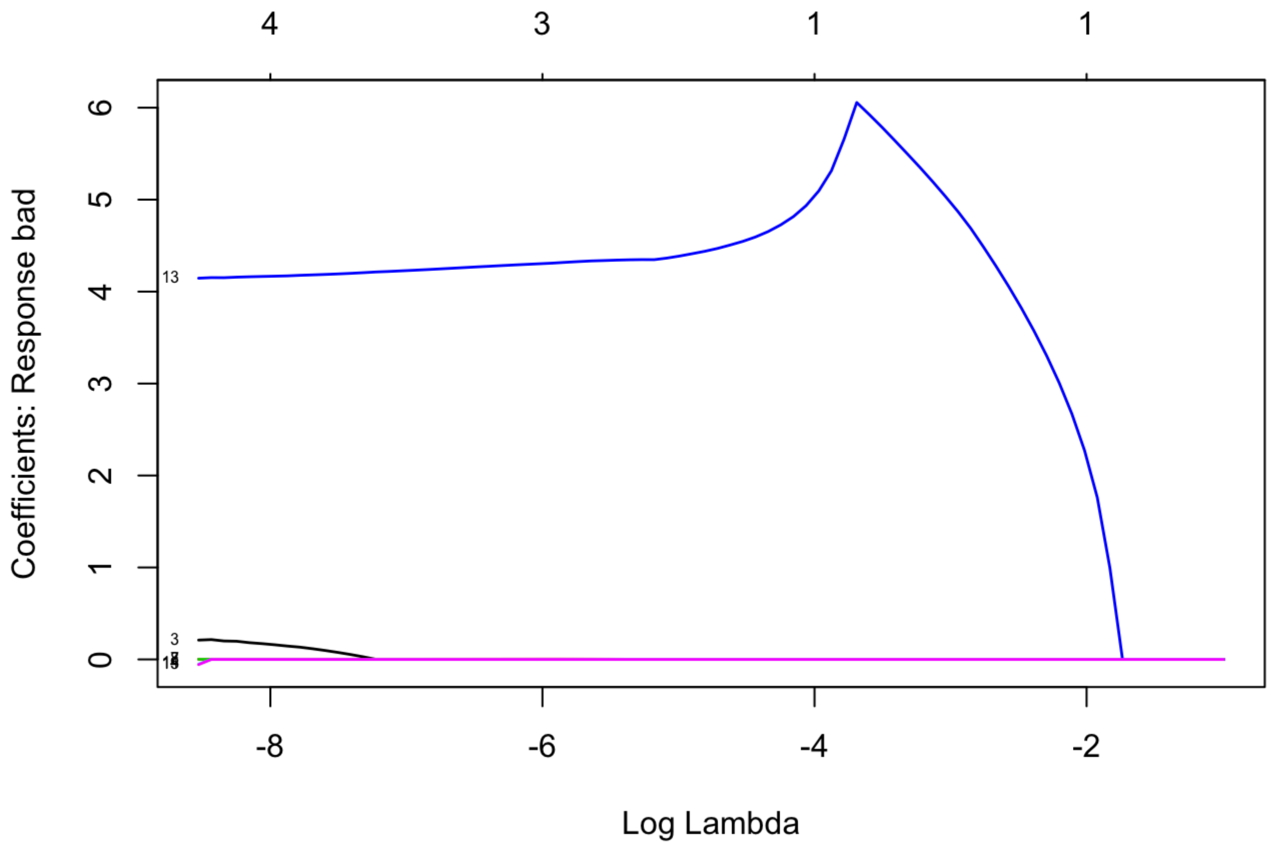
```
lassoReg2 <- train(quality_level~., train, method = 'glmnet',
                  tuneGrid = expand.grid(alpha = 1,
                                         lambda = seq(0.0001, 0.5, length = 5)),
                  trControl = trainControl(method = "repeatedcv",
                                           number = 10, repeats = 5,
                                           verboseIter = TRUE))
```

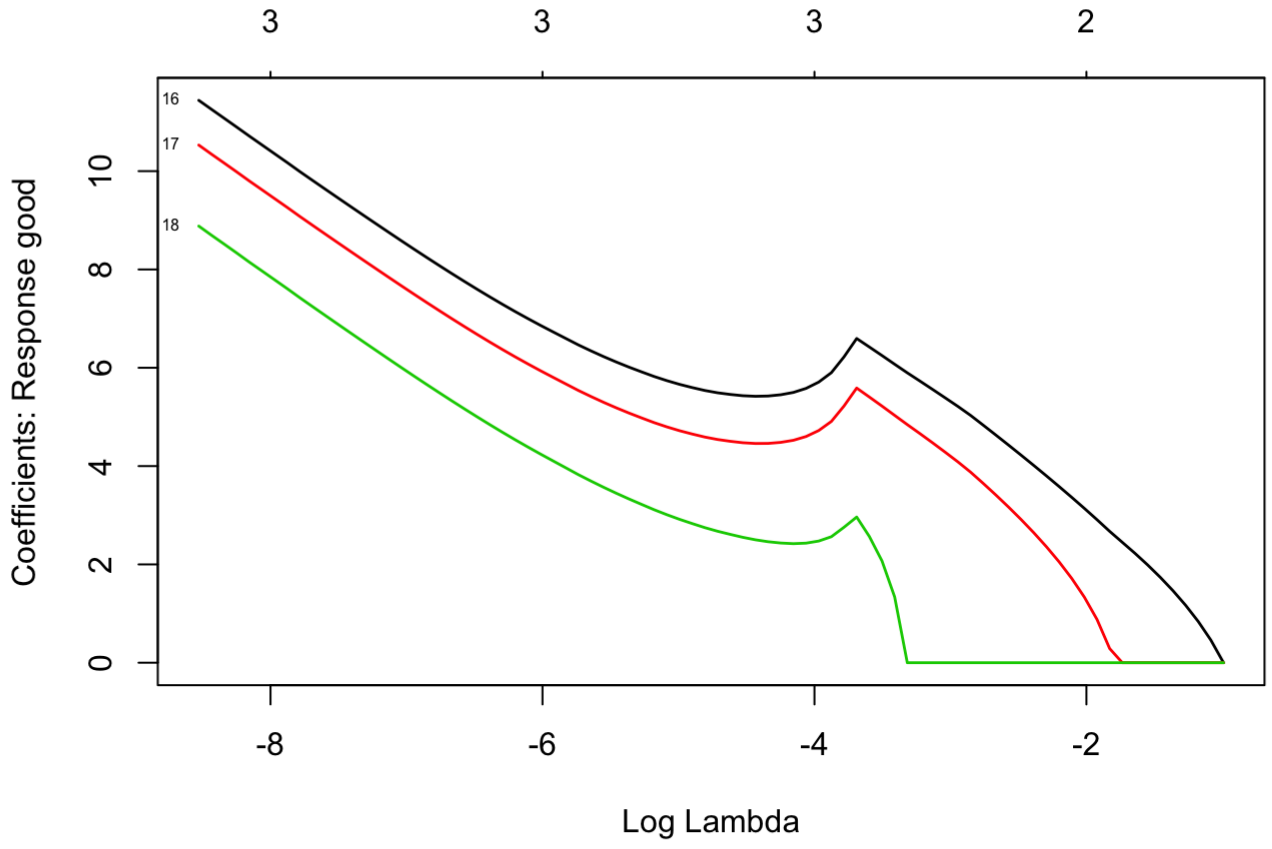
```
## + Fold01.Rep1: alpha=1, lambda=0.5
## - Fold01.Rep1: alpha=1, lambda=0.5
## + Fold02.Rep1: alpha=1, lambda=0.5
## - Fold02.Rep1: alpha=1, lambda=0.5
## + Fold03.Rep1: alpha=1, lambda=0.5
## - Fold03.Rep1: alpha=1, lambda=0.5
## + Fold04.Rep1: alpha=1, lambda=0.5
## - Fold04.Rep1: alpha=1, lambda=0.5
## + Fold05.Rep1: alpha=1, lambda=0.5
## - Fold05.Rep1: alpha=1, lambda=0.5
## + Fold06.Rep1: alpha=1, lambda=0.5
## - Fold06.Rep1: alpha=1, lambda=0.5
## + Fold07.Rep1: alpha=1, lambda=0.5
## - Fold07.Rep1: alpha=1, lambda=0.5
## + Fold08.Rep1: alpha=1, lambda=0.5
## - Fold08.Rep1: alpha=1, lambda=0.5
## + Fold09.Rep1: alpha=1, lambda=0.5
## - Fold09.Rep1: alpha=1, lambda=0.5
## + Fold10.Rep1: alpha=1, lambda=0.5
## - Fold10.Rep1: alpha=1, lambda=0.5
## + Fold01.Rep2: alpha=1, lambda=0.5
## - Fold01.Rep2: alpha=1, lambda=0.5
## + Fold02.Rep2: alpha=1, lambda=0.5
## - Fold02.Rep2: alpha=1, lambda=0.5
## + Fold03.Rep2: alpha=1, lambda=0.5
## - Fold03.Rep2: alpha=1, lambda=0.5
## + Fold04.Rep2: alpha=1, lambda=0.5
## - Fold04.Rep2: alpha=1, lambda=0.5
## + Fold05.Rep2: alpha=1, lambda=0.5
## - Fold05.Rep2: alpha=1, lambda=0.5
## + Fold06.Rep2: alpha=1, lambda=0.5
## - Fold06.Rep2: alpha=1, lambda=0.5
## + Fold07.Rep2: alpha=1, lambda=0.5
## - Fold07.Rep2: alpha=1, lambda=0.5
## + Fold08.Rep2: alpha=1, lambda=0.5
## - Fold08.Rep2: alpha=1, lambda=0.5
## + Fold09.Rep2: alpha=1, lambda=0.5
## - Fold09.Rep2: alpha=1, lambda=0.5
## + Fold10.Rep2: alpha=1, lambda=0.5
## - Fold10.Rep2: alpha=1, lambda=0.5
## + Fold01.Rep3: alpha=1, lambda=0.5
## - Fold01.Rep3: alpha=1, lambda=0.5
## + Fold02.Rep3: alpha=1, lambda=0.5
## - Fold02.Rep3: alpha=1, lambda=0.5
## + Fold03.Rep3: alpha=1, lambda=0.5
## - Fold03.Rep3: alpha=1, lambda=0.5
## + Fold04.Rep3: alpha=1, lambda=0.5
## - Fold04.Rep3: alpha=1, lambda=0.5
## + Fold05.Rep3: alpha=1, lambda=0.5
## - Fold05.Rep3: alpha=1, lambda=0.5
## + Fold06.Rep3: alpha=1, lambda=0.5
## - Fold06.Rep3: alpha=1, lambda=0.5
## + Fold07.Rep3: alpha=1, lambda=0.5
## - Fold07.Rep3: alpha=1, lambda=0.5
## + Fold08.Rep3: alpha=1, lambda=0.5
## - Fold08.Rep3: alpha=1, lambda=0.5
## + Fold09.Rep3: alpha=1, lambda=0.5
## - Fold09.Rep3: alpha=1, lambda=0.5
## + Fold10.Rep3: alpha=1, lambda=0.5
## - Fold10.Rep3: alpha=1, lambda=0.5
## + Fold01.Rep4: alpha=1, lambda=0.5
## - Fold01.Rep4: alpha=1, lambda=0.5
## + Fold02.Rep4: alpha=1, lambda=0.5
## - Fold02.Rep4: alpha=1, lambda=0.5
## + Fold03.Rep4: alpha=1, lambda=0.5
## - Fold03.Rep4: alpha=1, lambda=0.5
## + Fold04.Rep4: alpha=1, lambda=0.5
## - Fold04.Rep4: alpha=1, lambda=0.5
## + Fold05.Rep4: alpha=1, lambda=0.5
## - Fold05.Rep4: alpha=1, lambda=0.5
## + Fold06.Rep4: alpha=1, lambda=0.5
## - Fold06.Rep4: alpha=1, lambda=0.5
## + Fold07.Rep4: alpha=1, lambda=0.5
## - Fold07.Rep4: alpha=1, lambda=0.5
## + Fold08.Rep4: alpha=1, lambda=0.5
## - Fold08.Rep4: alpha=1, lambda=0.5
## + Fold09.Rep4: alpha=1, lambda=0.5
## - Fold09.Rep4: alpha=1, lambda=0.5
## + Fold10.Rep4: alpha=1, lambda=0.5
## - Fold10.Rep4: alpha=1, lambda=0.5
## + Fold01.Rep5: alpha=1, lambda=0.5
## - Fold01.Rep5: alpha=1, lambda=0.5
## + Fold02.Rep5: alpha=1, lambda=0.5
## - Fold02.Rep5: alpha=1, lambda=0.5
## + Fold03.Rep5: alpha=1, lambda=0.5
## - Fold03.Rep5: alpha=1, lambda=0.5
## + Fold04.Rep5: alpha=1, lambda=0.5
## - Fold04.Rep5: alpha=1, lambda=0.5
## + Fold05.Rep5: alpha=1, lambda=0.5
## - Fold05.Rep5: alpha=1, lambda=0.5
## + Fold06.Rep5: alpha=1, lambda=0.5
## - Fold06.Rep5: alpha=1, lambda=0.5
## + Fold07.Rep5: alpha=1, lambda=0.5
## - Fold07.Rep5: alpha=1, lambda=0.5
## + Fold08.Rep5: alpha=1, lambda=0.5
## - Fold08.Rep5: alpha=1, lambda=0.5
## + Fold09.Rep5: alpha=1, lambda=0.5
## - Fold09.Rep5: alpha=1, lambda=0.5
## + Fold10.Rep5: alpha=1, lambda=0.5
## - Fold10.Rep5: alpha=1, lambda=0.5
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 1, lambda = 1e-04 on full training set
```

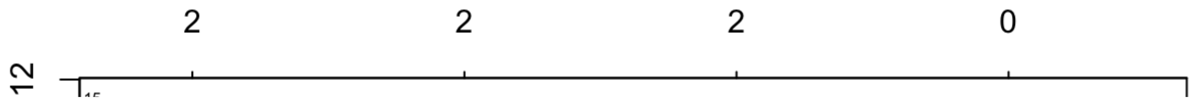
```
# plot results
plot(lassoReg2)
```

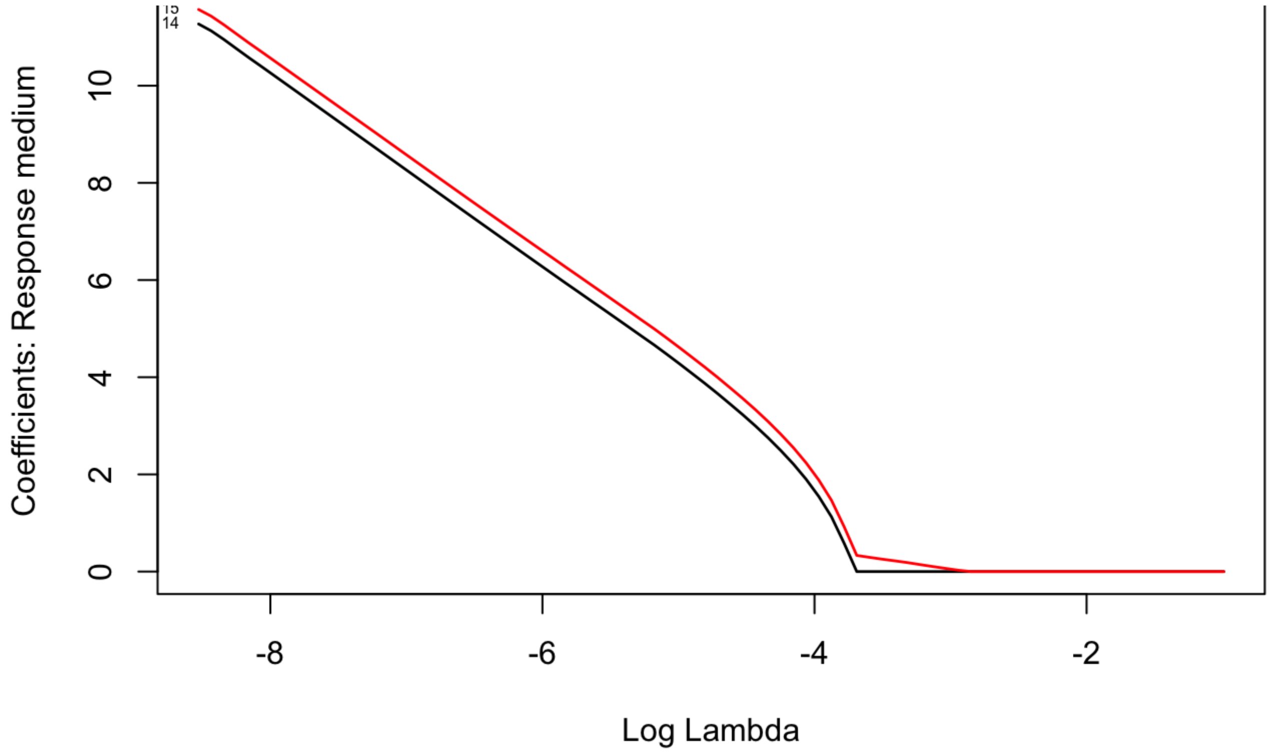


```
plot(lassoReg2$finalModel, xvar = 'lambda', lwd =1.4, label=TRUE)
```

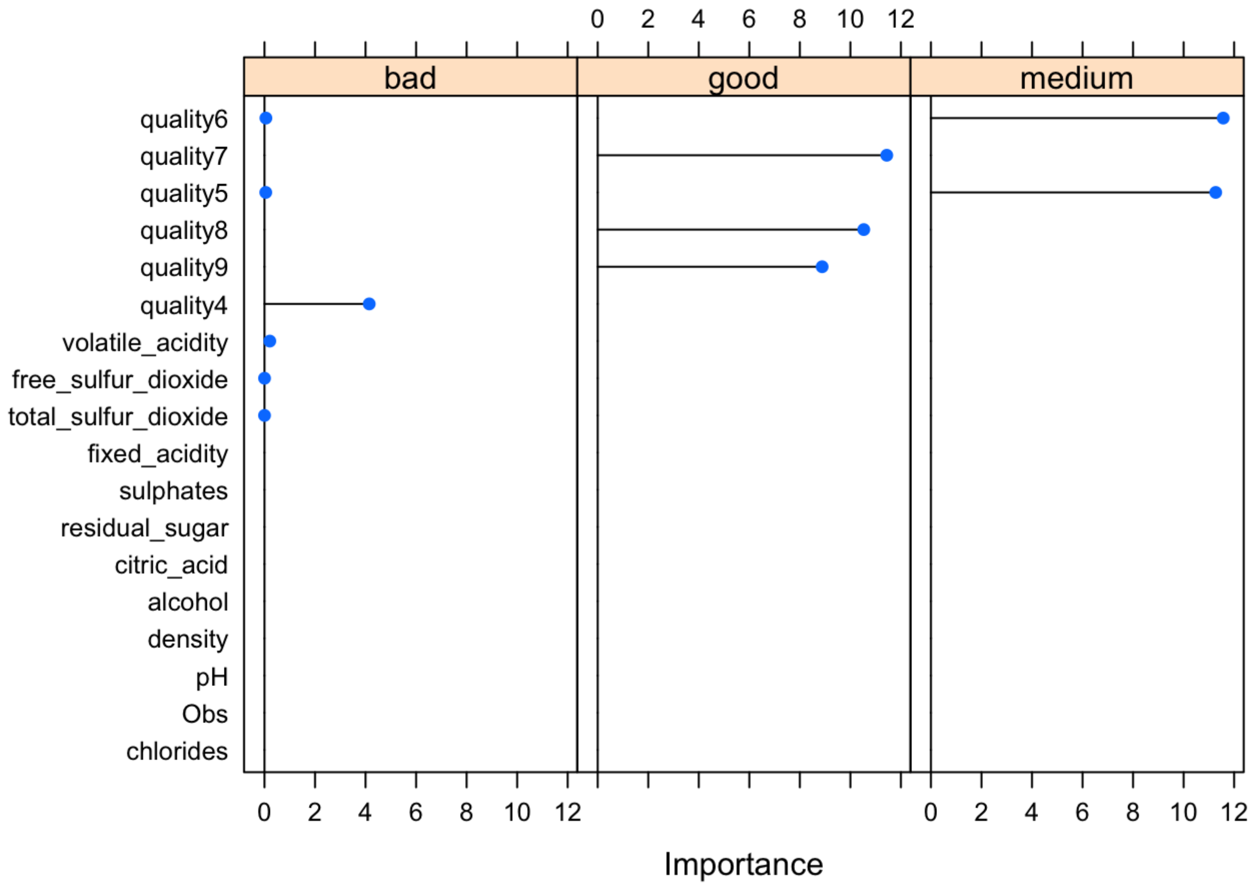

4
3
1
1
Coefficients: Response bad
Log Lambda
13
3
15


3
3
3
2
Coefficients: Response good
Log Lambda
16
17
18

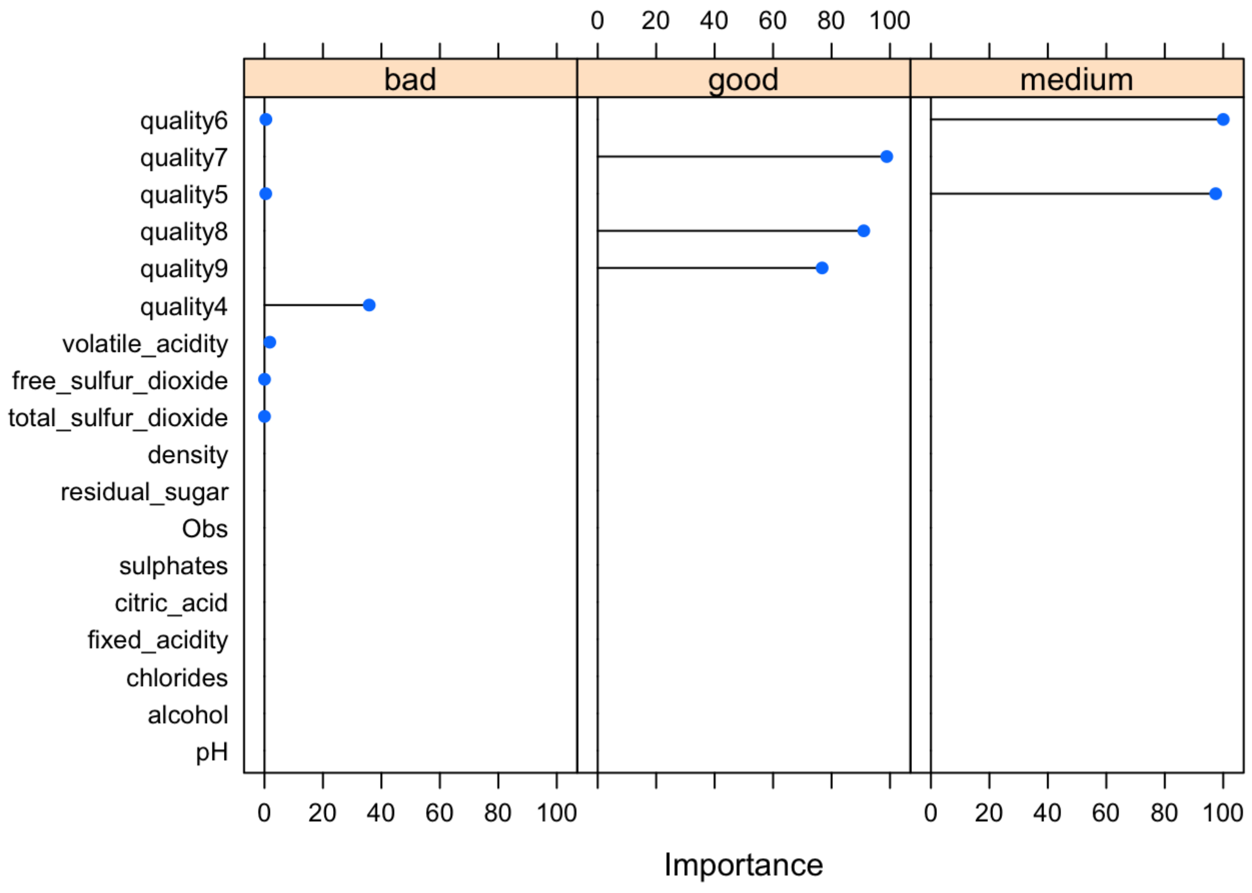

2
2
2
0
12



```
plot(varImp(lassoReg2, scale = FALSE))
```



```
plot(varImp(lassoReg2, scale = TRUE))
```



```
PredictLasso2 <- predict(lassoReg2, test)
confusionMatrix(PredictLasso2, test$quality_level)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  bad good medium
##     bad      79    0      0
##     good      0  411      0
##     medium    0    0   1470
## 
## Overall Statistics
##                                      
##                Accuracy : 1          
##                  95% CI : (0.9981, 1)
##     No Information Rate : 0.75       
##     P-Value [Acc > NIR] : < 2.2e-16  
##                                      
##                   Kappa : 1          
##  Mcnemar's Test P-Value : NA         
## 
## Statistics by Class:
## 
##                      Class: bad Class: good Class: medium
## Sensitivity             1.00000      1.0000          1.00
## Specificity             1.00000      1.0000          1.00
## Pos Pred Value          1.00000      1.0000          1.00
## Neg Pred Value          1.00000      1.0000          1.00
## Prevalence              0.04031      0.2097          0.75
## Detection Rate          0.04031      0.2097          0.75
## Detection Prevalence    0.04031      0.2097          0.75
## Balanced Accuracy       1.00000      1.0000          1.00
```

We have run the Lasso model on Quality Level variable. The accuracy is 100%