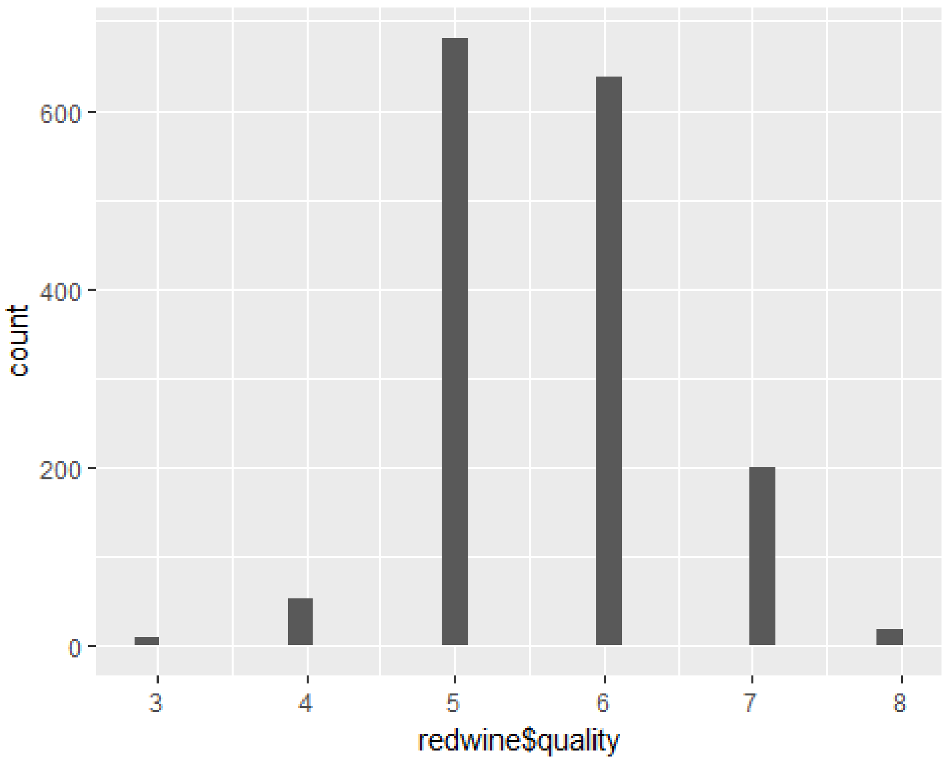
Machine Learning Project

Prediction of Wine quality Analysis

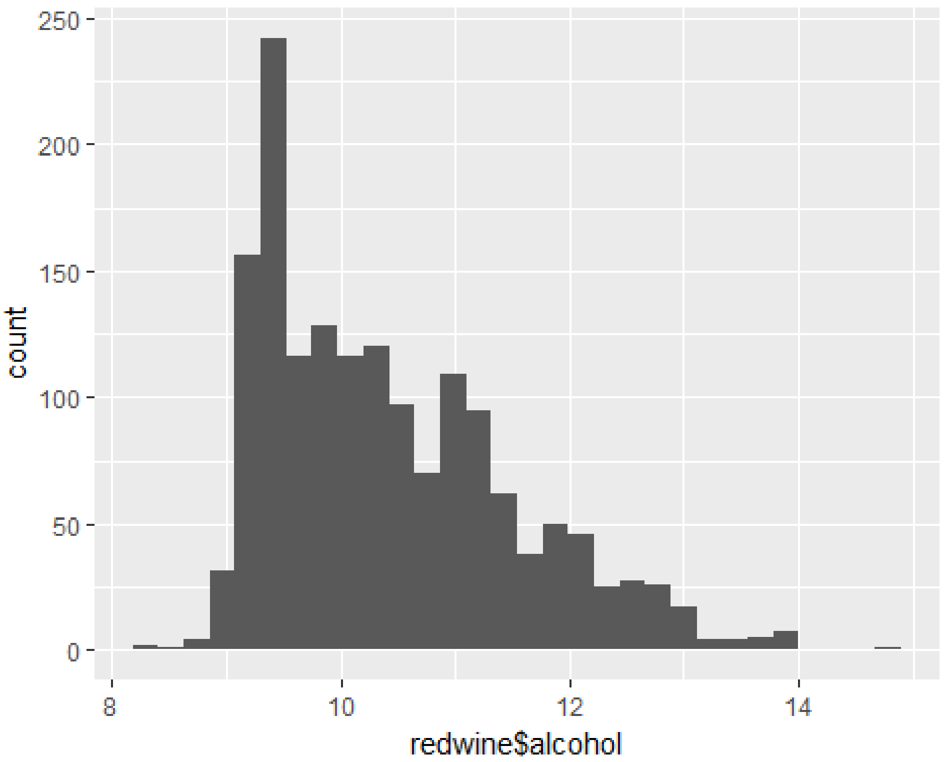
```
library(ggplot2)
library (gridExtra)

## Warning: package 'gridExtra' was built under R version 3.4.4

library(corrplot)

## Warning: package 'corrplot' was built under R version 3.4.4

## corrplot 0.84 loaded

#load the wine quality dataset
redwine <- read.csv("C:/Users/LENOVO/Desktop/dataset/redwine.csv")
View(redwine)
redwine<-data.frame(redwine)
str(redwine)

## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58
0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069
0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36
3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57
0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...

summary(redwine)

##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00
##  Median :0.07900   Median :14.00       Median : 38.00
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00
##     density             pH          sulphates         alcohol
##  Min.   :0.9901   Min.   :2.740   Min.   :0.3300   Min.   : 8.40
##  1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50
##  Median :0.9968   Median :3.310   Median :0.6200   Median :10.20
##  Mean   :0.9967   Mean   :3.311   Mean   :0.6581   Mean   :10.42
##  3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10
##  Max.   :1.0037   Max.   :4.010   Max.   :2.0000   Max.   :14.90
##     quality
##  Min.   :3.000
##  1st Qu.:5.000
##  Median :6.000
##  Mean   :5.636
##  3rd Qu.:6.000
##  Max.   :8.000
```

```
#histogram analysis
#plot shows that quality 5 is more
qplot(redwine$quality, geom="histogram")
```

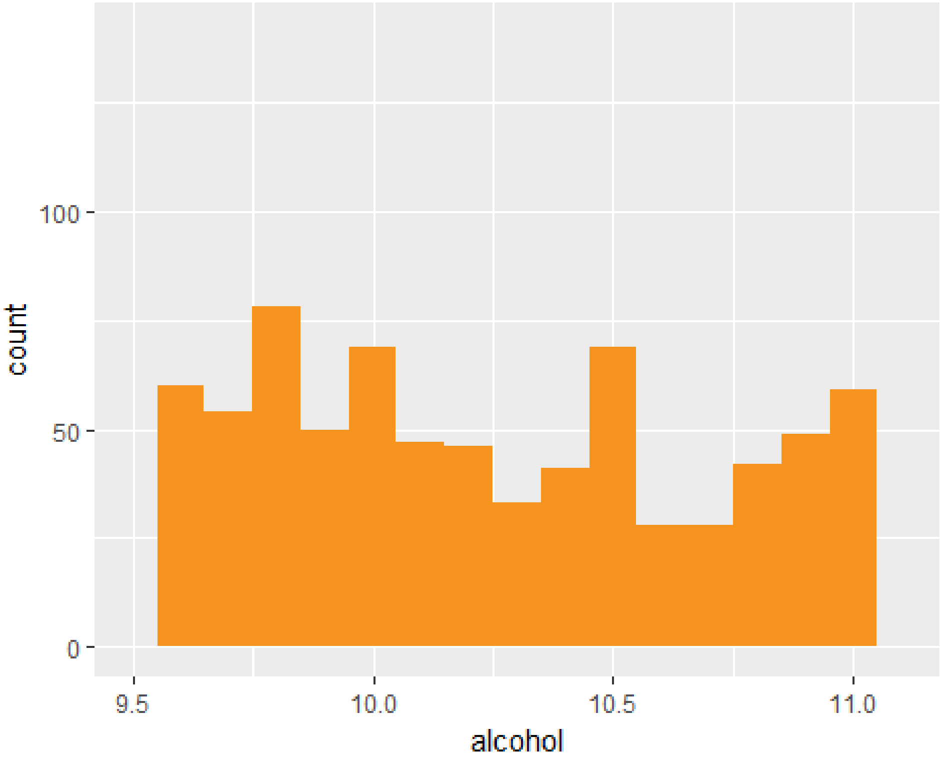
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



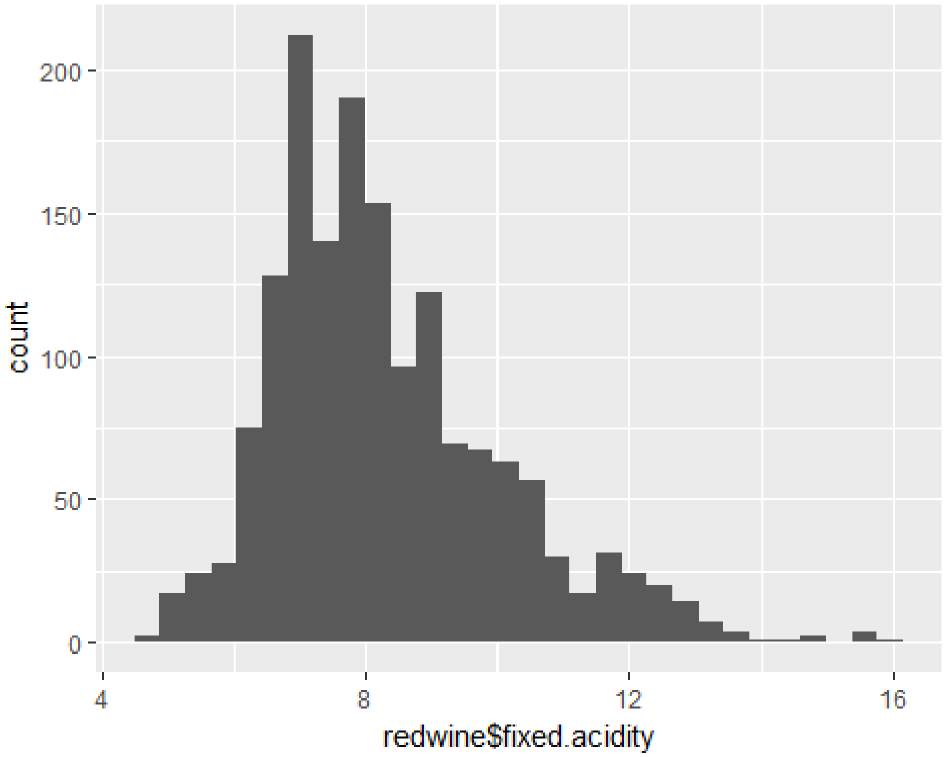
```
#plot shows that alcohol
summary(redwine$alcohol)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    8.40    9.50   10.20   10.42   11.10   14.90

#"alcohol" is right-skewed distributed with some outliers located at right
side. The most frequent values are between 9.4-9.6.
qplot(redwine$alcohol, geom="histogram")

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



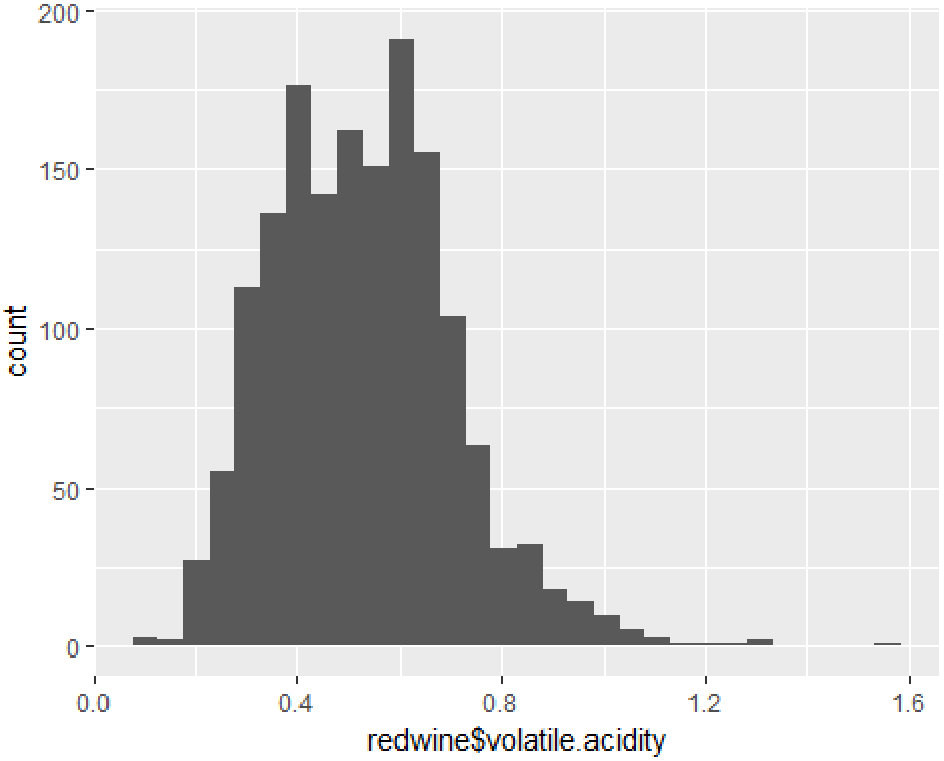
```
#Now the IQR is 1.4, 3rd quantile is not very far from max value, and data
gathers more in center.
qplot(alcohol, data=redwine, fill=I('#F79420'), binwidth =
.1)+xlim(9.50,11.10 )

## Warning: Removed 677 rows containing non-finite values (stat_bin).

## Warning: Removed 1 rows containing missing values (geom_bar).
```



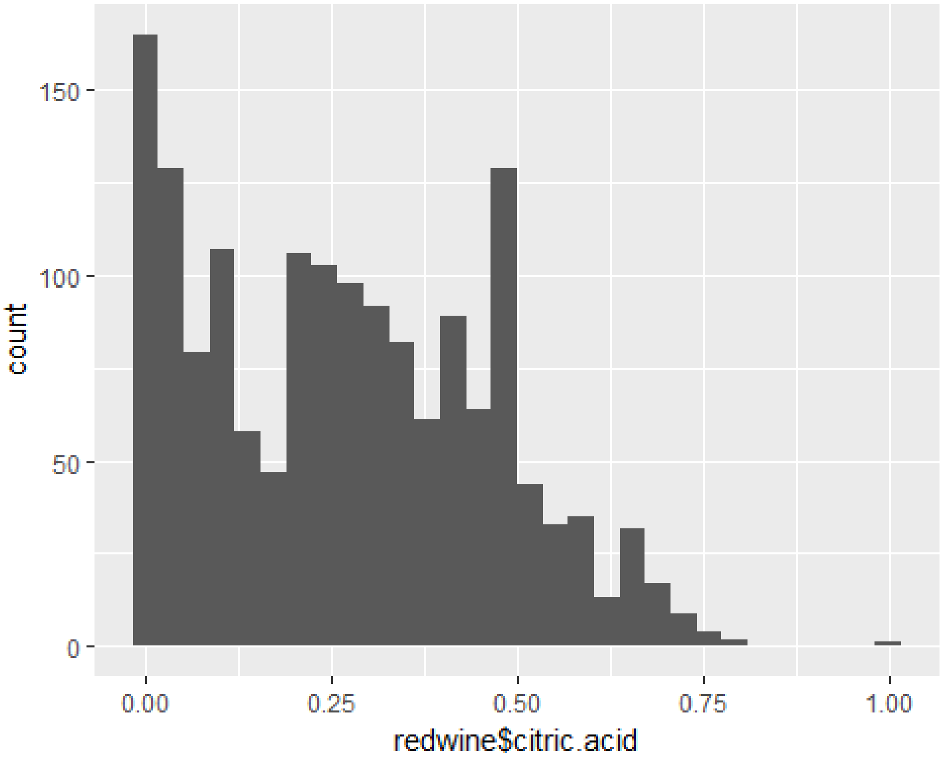
```
#We can see majority fixed acidity gathering in middle part
qplot(redwine$fixed.acidity, geom="histogram")

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



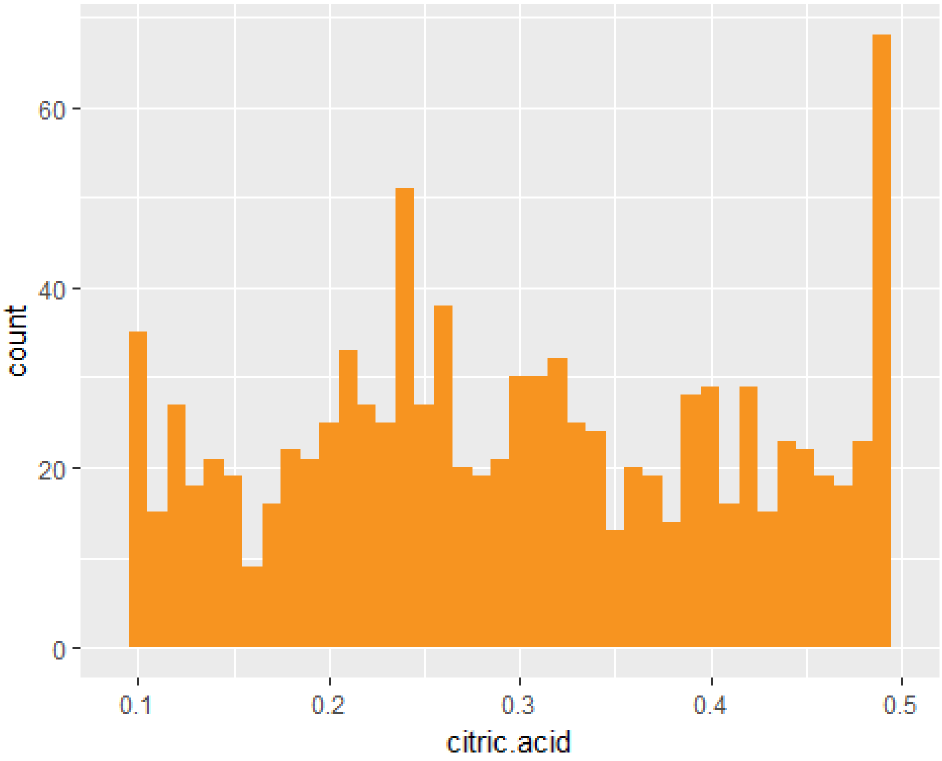
```
# The most frequent values are between 0.4-0.6.right skewed
qplot(redwine$volatile.acidity, geom="histogram")

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



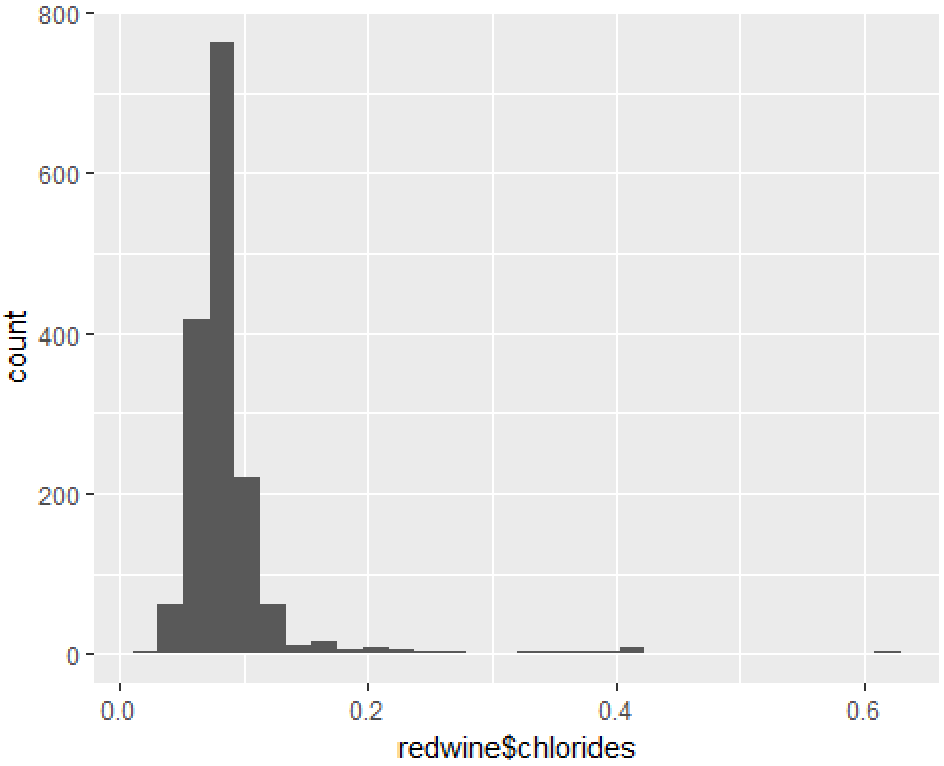
```
qplot(redwine$citric.acid, geom="histogram")

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



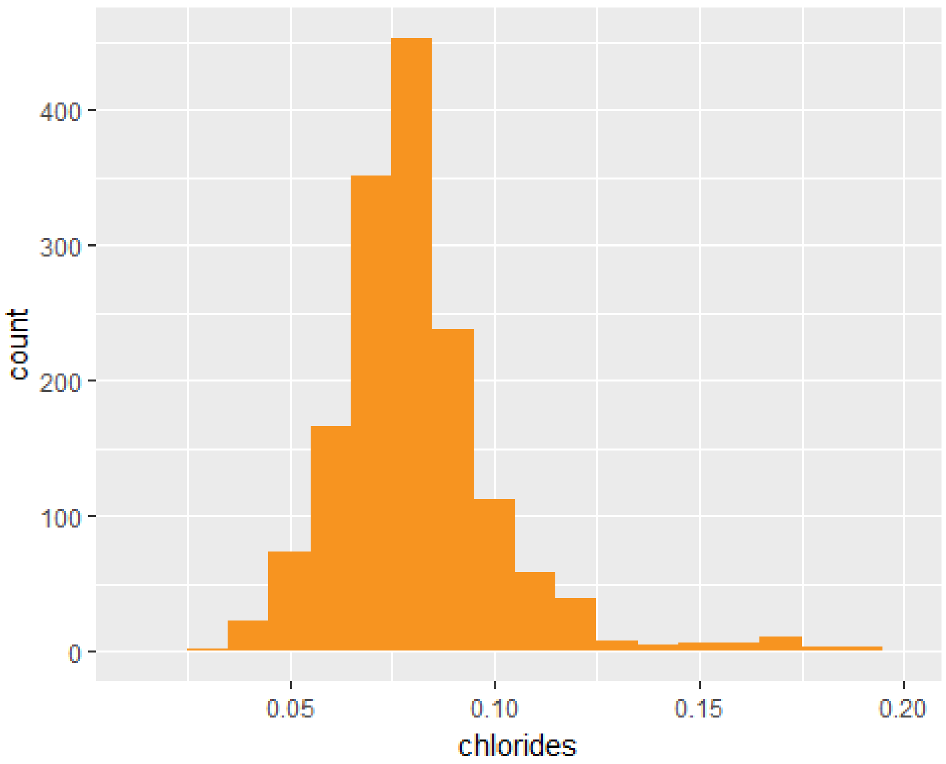
```
#summary
summary(redwine$citric.acid)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.090   0.260   0.271   0.420   1.000

#Ater removing outliers.The most common citric.acid values are around 0.49
qplot(citric.acid, data=redwine, fill=I('orange'), binwidth =
.01)+xlim(0.090,.5)

## Warning: Removed 563 rows containing non-finite values (stat_bin).
```



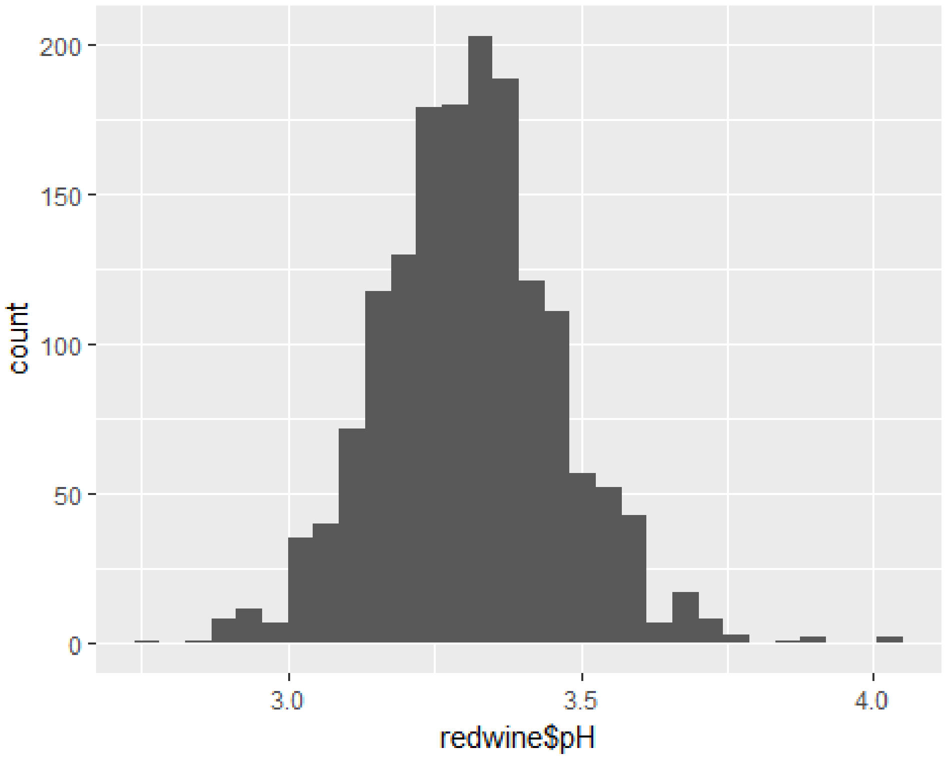
```
qplot(redwine$chlorides, geom="histogram")

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



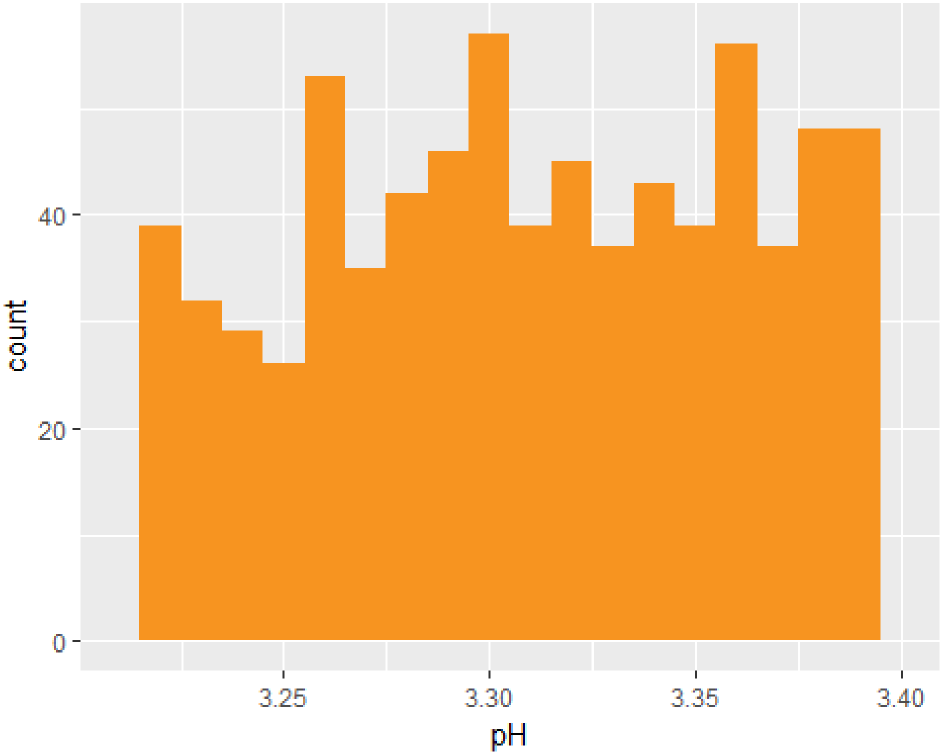
```
qplot(chlorides, data=redwine, fill=I('orange'), binwidth = .01)+ xlim(.012,
.2)

## Warning: Removed 41 rows containing non-finite values (stat_bin).

## Warning: Removed 1 rows containing missing values (geom_bar).
```



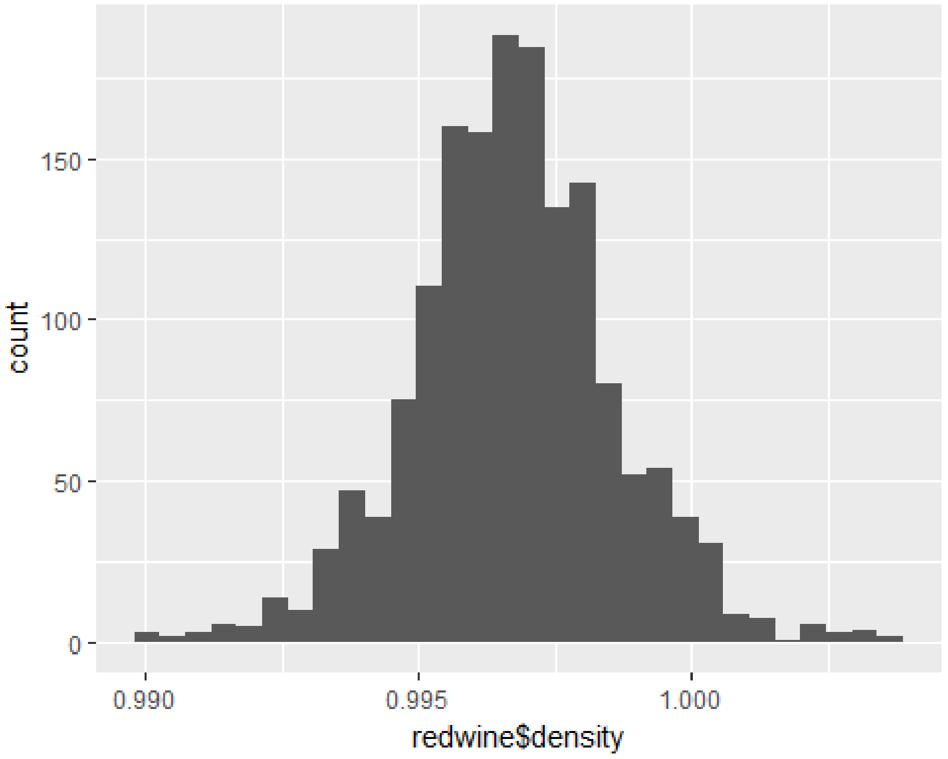
```
qplot(redwine$pH, geom="histogram")

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



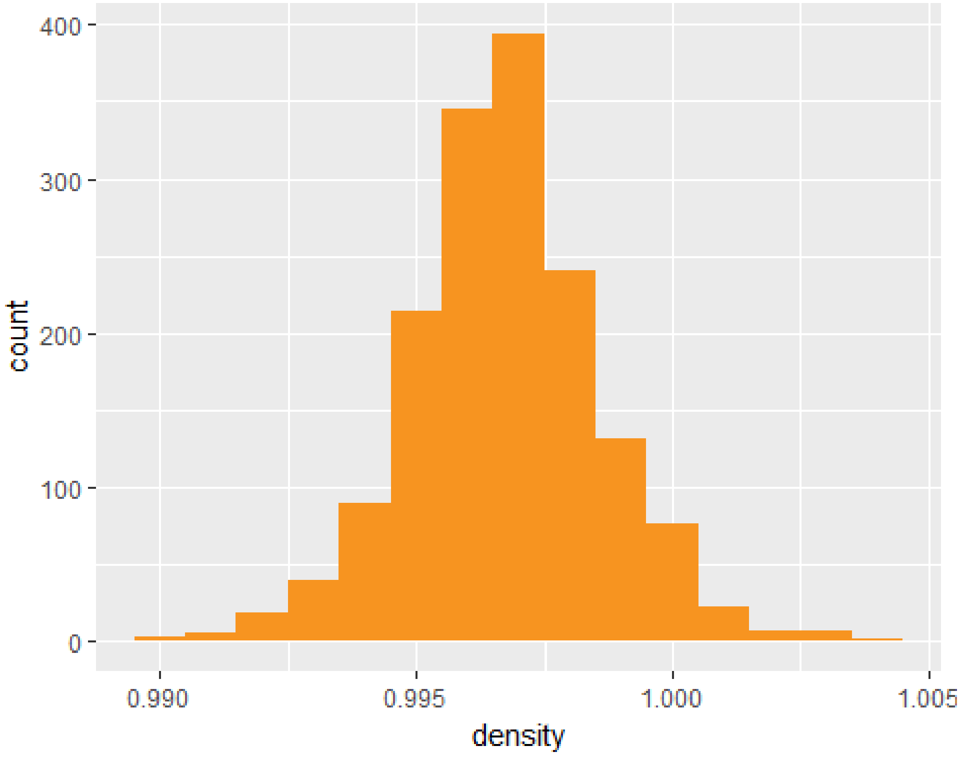
```
qplot(pH, data=redwine, fill=I('orange'), binwidth = .01)+xlim(3.210, 3.400)

## Warning: Removed 775 rows containing non-finite values (stat_bin).

## Warning: Removed 1 rows containing missing values (geom_bar).
```



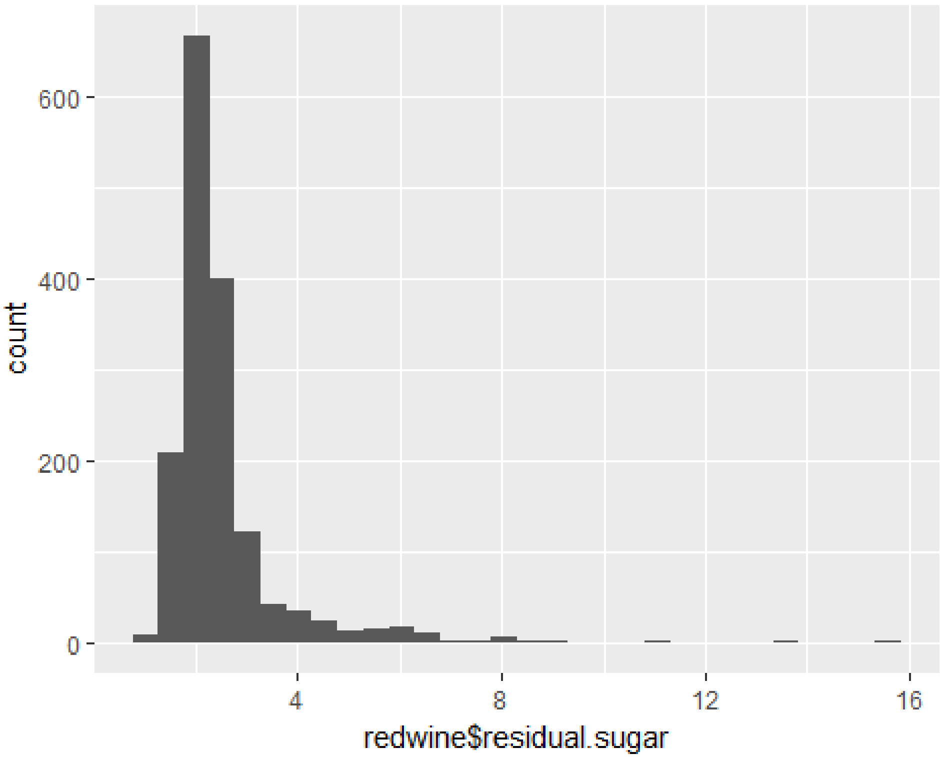
```
qplot(redwine$density, geom="histogram")

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



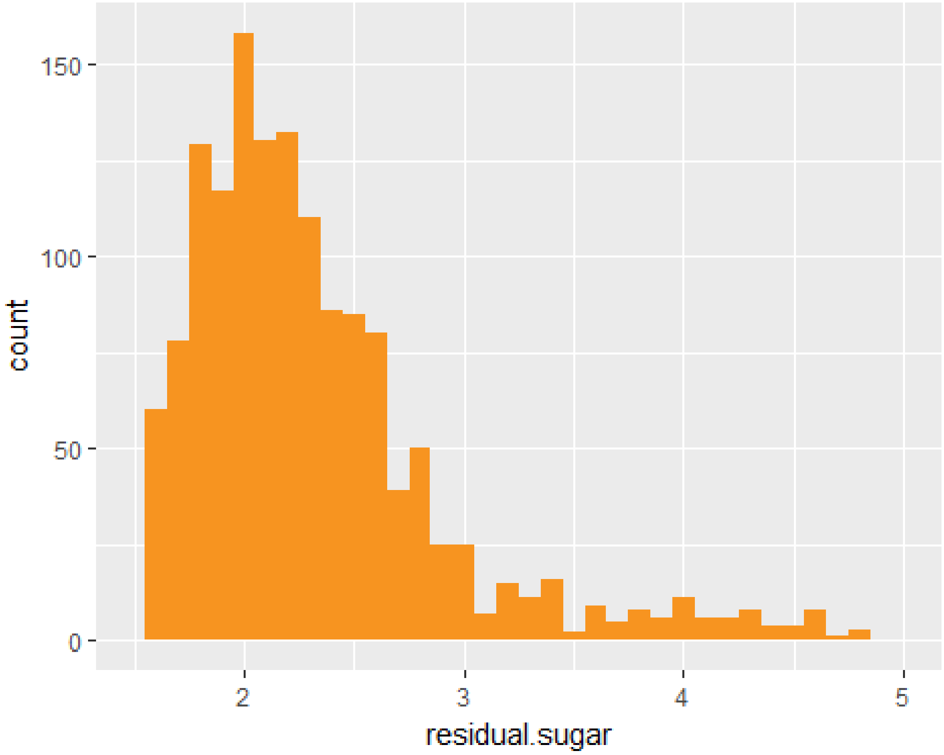
```r
qplot(density, data=redwine, fill=I('orange'), binwidth = .001)
```



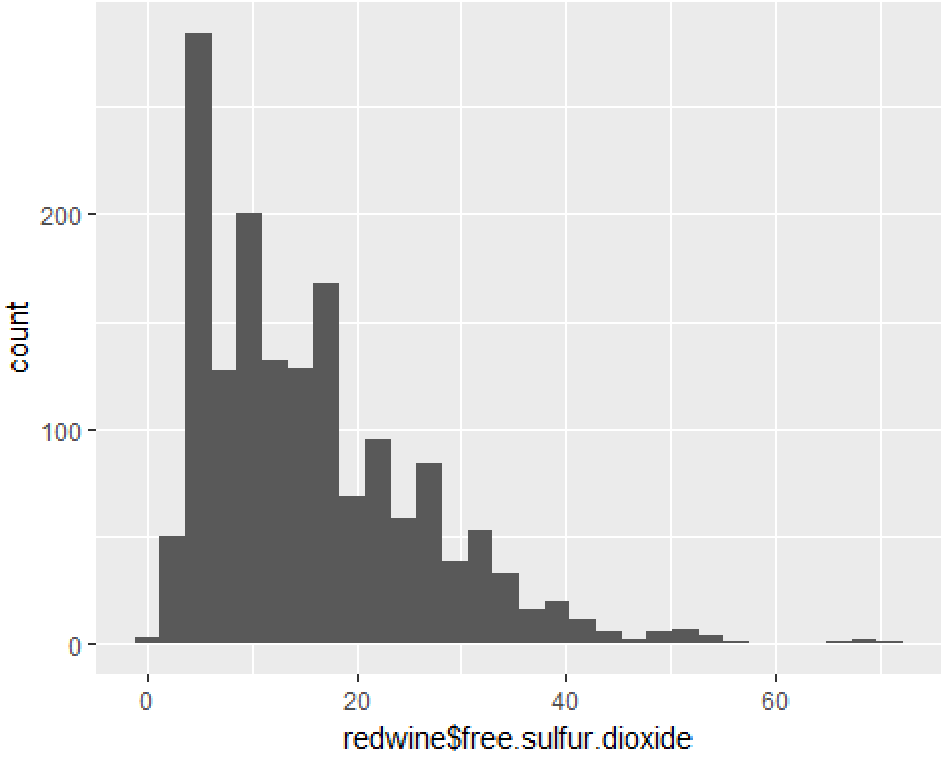
```
#The residual sugar content is high between 1.9 to 2.6. Residual sugar has a
positively skewed distribution; even after eliminating the outliers
distribution will remain skewed
qplot(redwine$residual.sugar, geom="histogram")

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



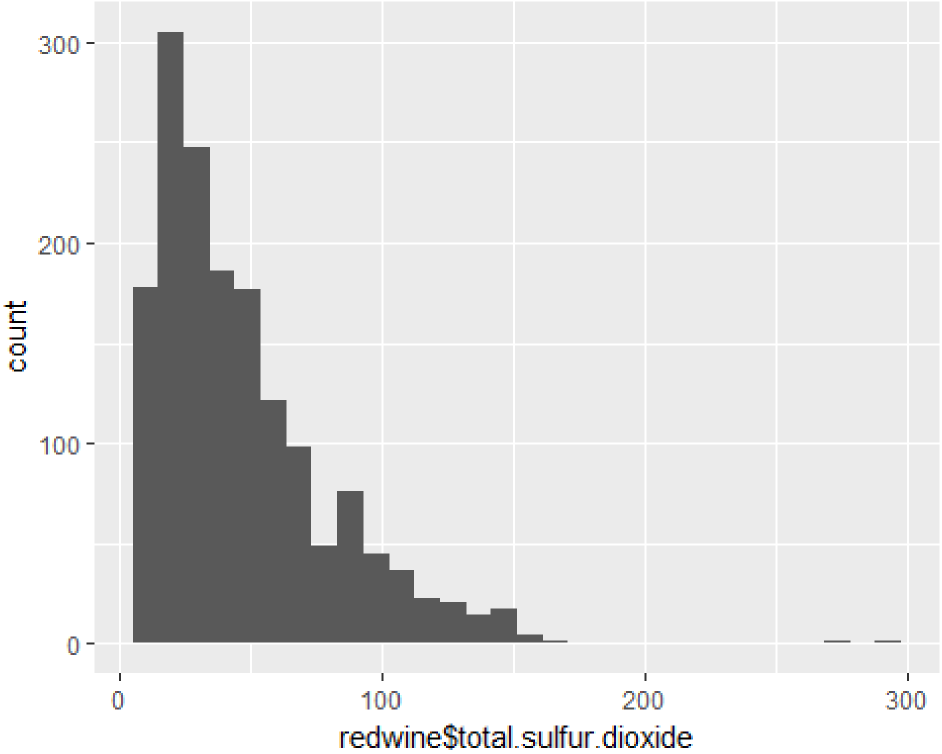
```
qplot(residual.sugar, data=redwine, fill=I('orange'), binwidth =
.1)+xlim(1.5,5)

## Warning: Removed 134 rows containing non-finite values (stat_bin).
```



```
qplot(redwine$free.sulfur.dioxide, geom="histogram")

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



```
qplot(redwine$total.sulfur.dioxide, geom="histogram")

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



```
# Overall distribution

#A distribution analysis of the key paramaters pH,Total and Free
sulfur.dioxide and Alcohol is explored, as these are the key factors

cor(redwine)

##                      fixed.acidity volatile.acidity citric.acid
## fixed.acidity           1.00000000     -0.256130895  0.67170343
## volatile.acidity       -0.25613089      1.000000000 -0.55249568
## citric.acid             0.67170343     -0.552495685  1.00000000
## residual.sugar          0.11477672      0.001917882  0.14357716
## chlorides               0.09370519      0.061297772  0.20382291
## free.sulfur.dioxide    -0.15379419     -0.010503827 -0.06097813
## total.sulfur.dioxide   -0.11318144      0.076470005  0.03553302
## density                 0.66804729      0.022026232  0.36494718
## pH                     -0.68297819      0.234937294 -0.54190414
## sulphates               0.18300566     -0.260986685  0.31277004
## alcohol                -0.06166827     -0.202288027  0.10990325
## quality                 0.12405165     -0.390557780  0.22637251
##                      residual.sugar    chlorides free.sulfur.dioxide
## fixed.acidity           0.114776724  0.093705186        -0.153794193
## volatile.acidity        0.001917882  0.061297772        -0.010503827
## citric.acid             0.143577162  0.203822914        -0.060978129
## residual.sugar          1.000000000  0.055609535         0.187048995
## chlorides               0.055609535  1.000000000         0.005562147
## free.sulfur.dioxide     0.187048995  0.005562147         1.000000000
## total.sulfur.dioxide    0.203027882  0.047400468         0.667666450
## density                 0.355283371  0.200632327        -0.021945831
## pH                     -0.085652422 -0.265026131         0.070377499
## sulphates               0.005527121  0.371260481         0.051657572
## alcohol                 0.042075437 -0.221140545        -0.069408354
## quality                 0.013731637 -0.128906560        -0.050656057
##                      total.sulfur.dioxide     density          pH
## fixed.acidity                 -0.11318144  0.66804729 -0.68297819
## volatile.acidity               0.07647000  0.02202623  0.23493729
## citric.acid                    0.03553302  0.36494718 -0.54190414
## residual.sugar                 0.20302788  0.35528337 -0.08565242
## chlorides                      0.04740047  0.20063233 -0.26502613
## free.sulfur.dioxide            0.66766645 -0.02194583  0.07037750
## total.sulfur.dioxide           1.00000000  0.07126948 -0.06649456
## density                        0.07126948  1.00000000 -0.34169933
## pH                            -0.06649456 -0.34169933  1.00000000
## sulphates                      0.04294684  0.14850641 -0.19664760
## alcohol                       -0.20565394 -0.49617977  0.20563251
## quality                       -0.18510029 -0.17491923 -0.05773139
##                         sulphates     alcohol     quality
## fixed.acidity         0.183005664 -0.06166827  0.12405165
## volatile.acidity     -0.260986685 -0.20228803 -0.39055778
## citric.acid           0.312770044  0.10990325  0.22637251
## residual.sugar        0.005527121  0.04207544  0.01373164
## chlorides             0.371260481 -0.22114054 -0.12890656
## free.sulfur.dioxide   0.051657572 -0.06940835 -0.05065606
## total.sulfur.dioxide  0.042946836 -0.20565394 -0.18510029
## density               0.148506412 -0.49617977 -0.17491923
## pH                   -0.196647602  0.20563251 -0.05773139
## sulphates             1.000000000  0.09359475  0.25139708
## alcohol               0.093594750  1.00000000  0.47616632
## quality               0.251397079  0.47616632  1.00000000
```

```
attach(redwine)
cor(quality,alcohol)
```

```
## [1] 0.4761663
```

```
cor(quality,pH)
```

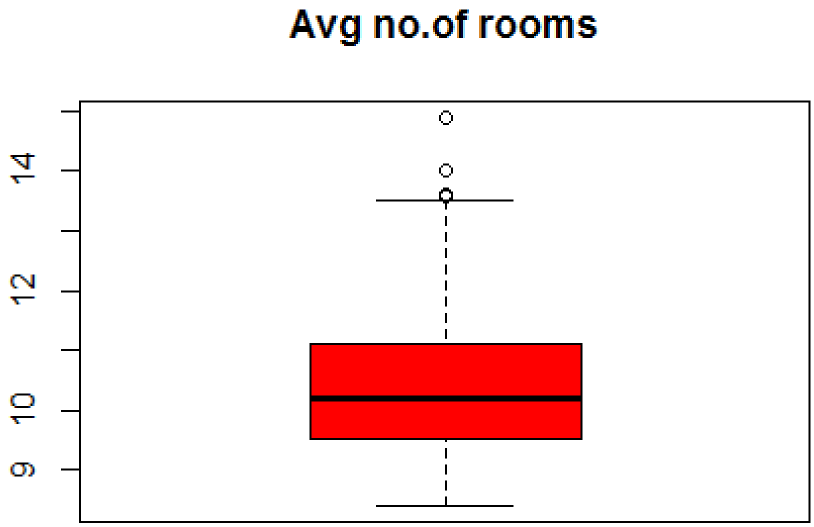
```
## [1] -0.05773139
```

```
cor(quality,sulphates)
```

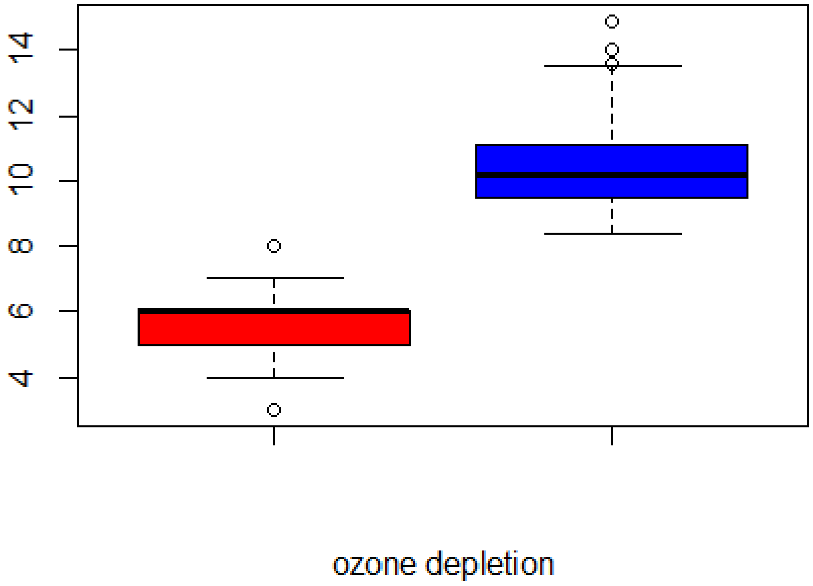
```
## [1] 0.2513971
```

```
cor(volatile.acidity,quality)
```

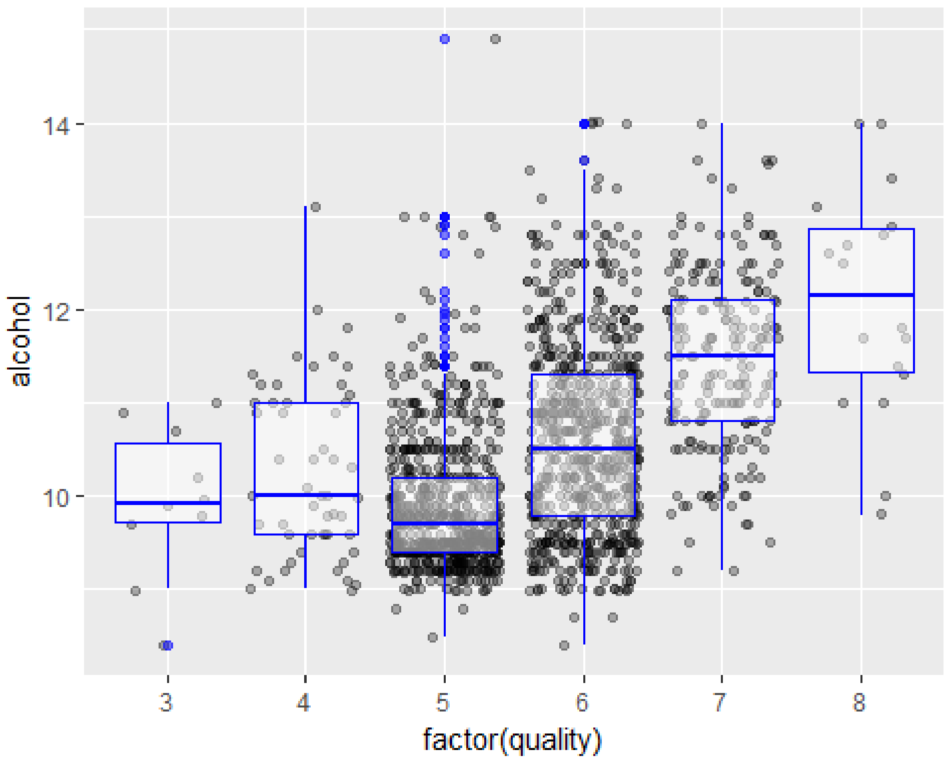
```
## [1] -0.3905578

cor(quality,citric.acid)

## [1] 0.2263725

#correlation is high for alcohol and quality





#boxplot analysis
boxplot(redwine$alcohol,main="Avg no.of rooms",col="red")
```



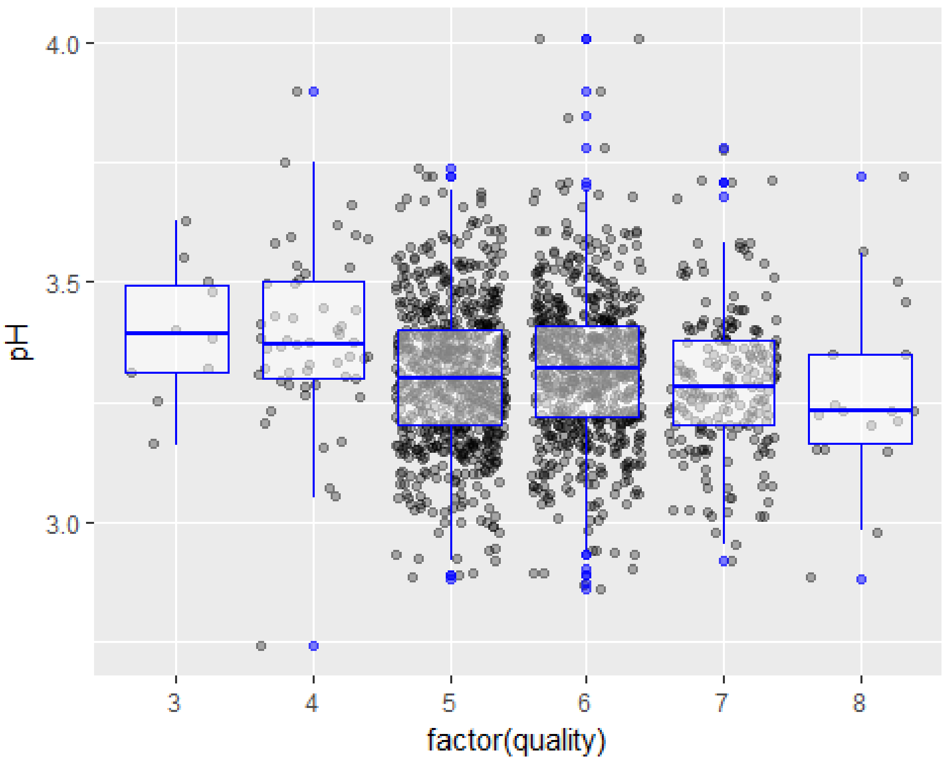
```
boxplot(quality,alcohol,col = c("red","blue"),xlab="ozone depletion")
```



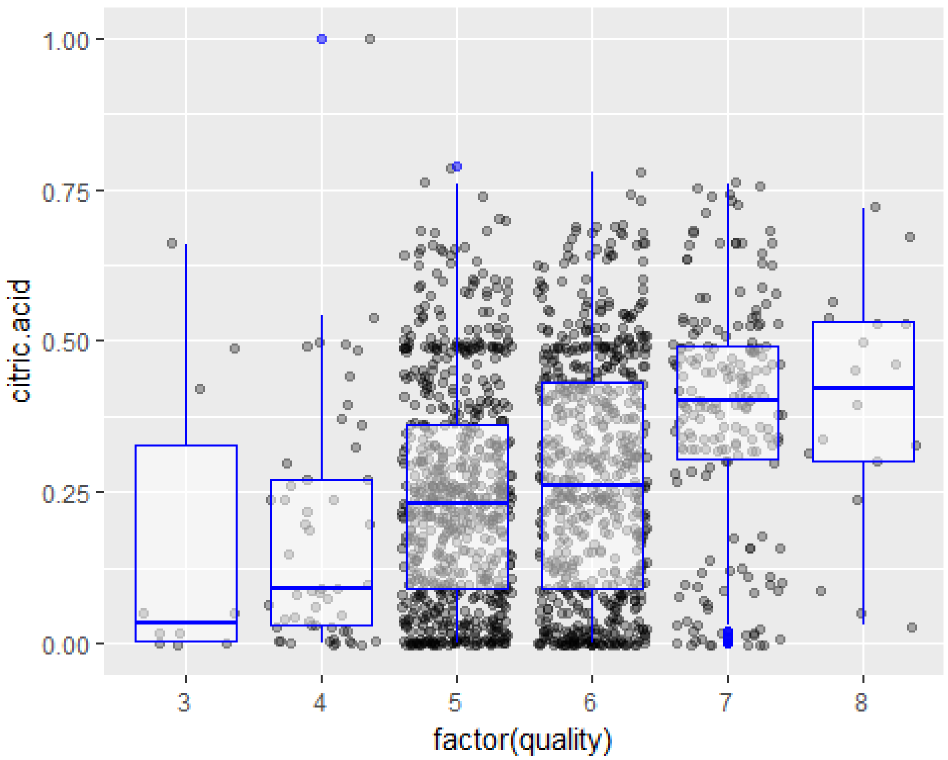
```
ggplot(aes(factor(quality),
           alcohol),
       data = redwine) +
  geom_jitter( alpha = .3)  +
  geom_boxplot( alpha = .5,color = 'blue')
```



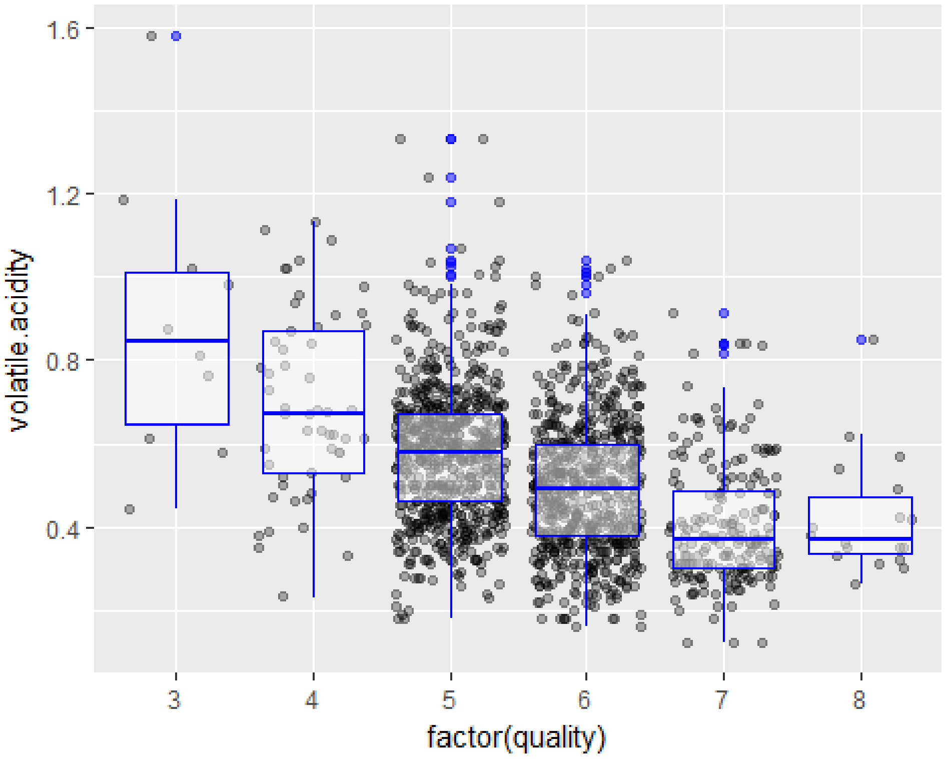
```
ggplot(aes(factor(quality),
           pH),
       data = redwine) +
  geom_jitter( alpha = .3)  +
  geom_boxplot( alpha = .5,color = 'blue')
```



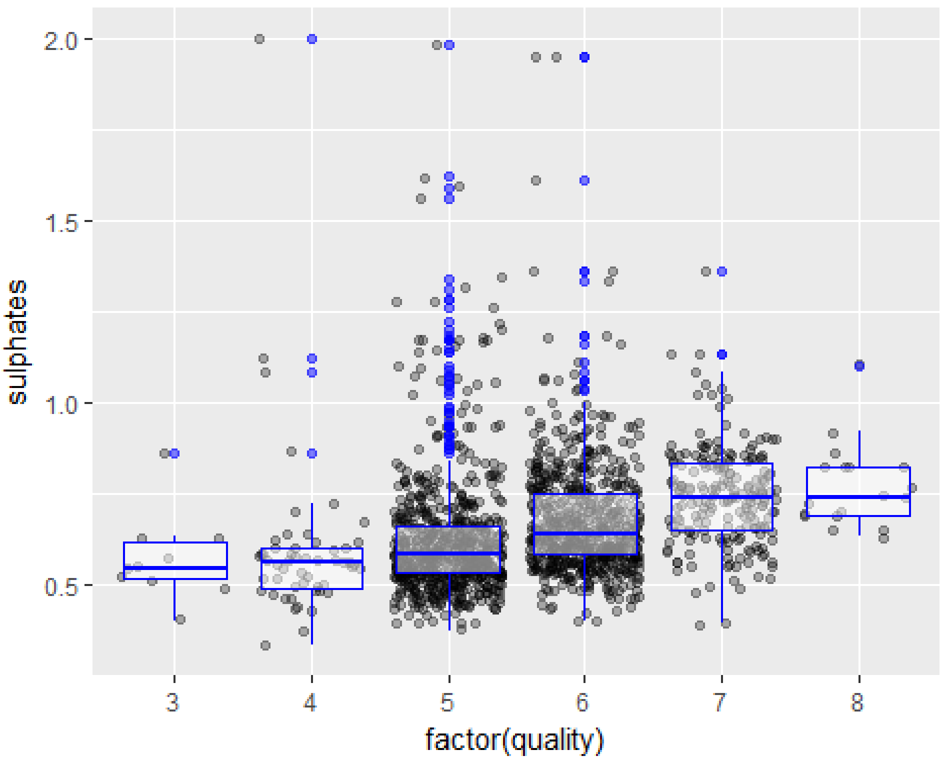
```
ggplot(aes(factor(quality),
           citric.acid),
       data = redwine) +
  geom_jitter( alpha = .3)  +
  geom_boxplot( alpha = .5,color = 'blue')
```



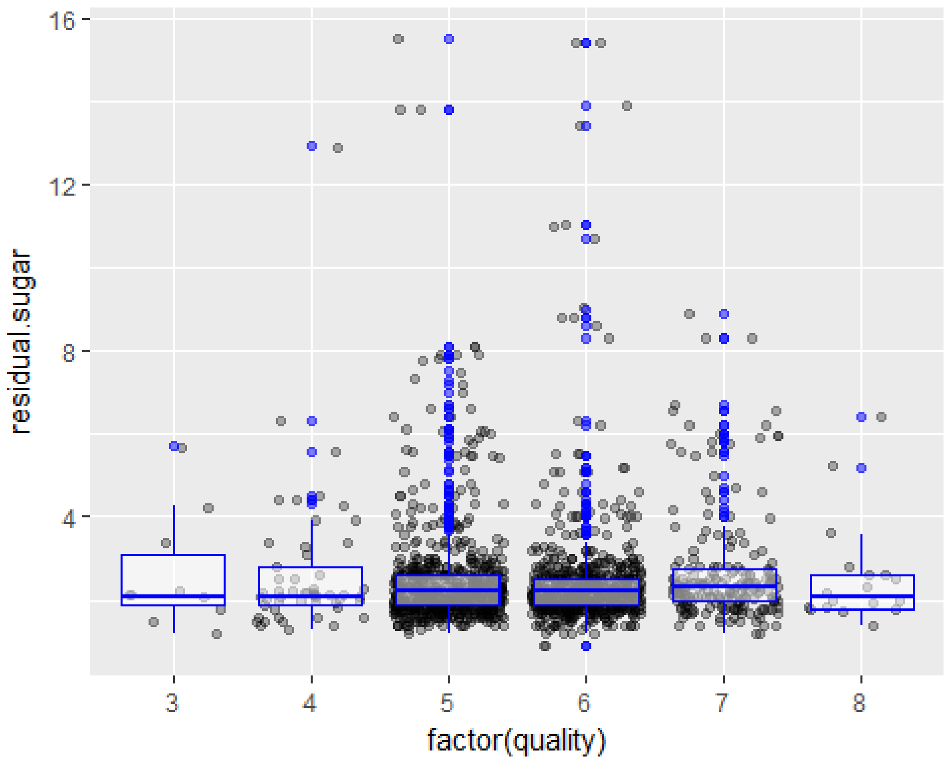
```
ggplot(aes(factor(quality),
           volatile.acidity),
       data = redwine) +
  geom_jitter( alpha = .3)  +
  geom_boxplot( alpha = .5,color = 'blue')
```



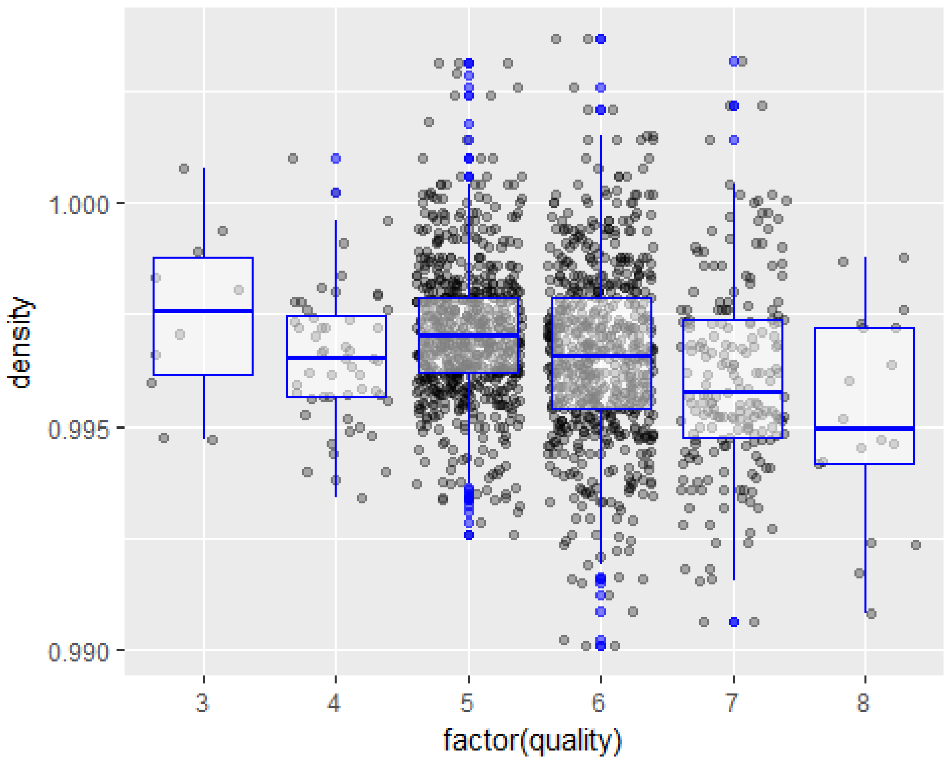
```
ggplot(aes(factor(quality),
           sulphates),
       data = redwine) +
  geom_jitter( alpha = .3)  +
  geom_boxplot( alpha = .5,color = 'blue')
```



```
ggplot(aes(factor(quality),
           residual.sugar),
       data = redwine) +
  geom_jitter( alpha = .3)  +
  geom_boxplot( alpha = .5,color = 'blue')
```



```
ggplot(aes(factor(quality),
           density),
       data = redwine) +
  geom_jitter( alpha = .3)  +
  geom_boxplot( alpha = .5,color = 'blue')
```



```
#linear regression
r<-lm(formula = quality ~alcohol, data = redwine)
r

##
## Call:
## lm(formula = quality ~ alcohol, data = redwine)
##
## Coefficients:
## (Intercept)      alcohol
##      1.8750       0.3608

summary(r)

##
## Call:
## lm(formula = quality ~ alcohol, data = redwine)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8442 -0.4112 -0.1690  0.5166  2.5888
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.87497    0.17471   10.73   <2e-16 ***
## alcohol      0.36084    0.01668   21.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7104 on 1597 degrees of freedom
## Multiple R-squared:  0.2267, Adjusted R-squared:  0.2263
## F-statistic: 468.3 on 1 and 1597 DF,  p-value: < 2.2e-16
```

```
anova(r)
```

```
## Analysis of Variance Table
##
## Response: quality
##             Df Sum Sq Mean Sq F value    Pr(>F)
## alcohol      1 236.29 236.295  468.27 < 2.2e-16 ***
## Residuals 1597 805.87   0.505
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Create Training and Test data -
set.seed(100)  # setting seed to reproduce results of random sampling
model <- sample(1:nrow(redwine), 0.8*nrow(redwine))  # row indices for
training data
trainingData <- redwine[model, ]  # model training data
testData  <- redwine[-model, ]   # test data
lmMod <- lm(quality ~ alcohol, data=trainingData)  # build the model
Pred <- predict(lmMod, testData)
Pred
```

```
##        3       11       13       15       20       21       23       35
## 5.400634 5.183037 5.436900 5.183037 5.183037 5.255569 5.291835 5.183037
##       37       38       43       51       57       59       64       66
## 5.763296 5.364367 5.654497 5.183037 5.328101 5.473166 5.400634 5.799562
##       70       72       82       87       89       91       92       96
## 5.654497 5.291835 5.255569 5.436900 5.291835 5.291835 5.436900 6.524885
##      100      112      114      124      133      135      138      143
## 5.110504 5.400634 5.364367 5.291835 6.561152 5.436900 5.183037 6.923813
##      144      149      151      152      157      164      173      177
## 5.255569 5.545698 5.581965 5.255569 5.654497 5.255569 5.183037 5.291835
##      187      188      194      199      215      216      219      220
## 5.183037 5.364367 5.364367 6.561152 5.400634 5.654497 5.364367 5.291835
##      222      224      233      237      238      241      244      248
## 5.255569 5.473166 5.328101 5.110504 5.183037 5.219303 5.183037 5.219303
##      253      257      258      260      266      272      275      279
## 5.654497 5.183037 5.183037 5.654497 5.545698 5.509432 5.255569 6.416087
##      281      285      287      292      293      297      311      327
## 5.219303 5.400634 5.581965 5.654497 5.581965 5.436900 5.219303 6.017159
##      329      332      335      339      343      349      361      362
## 5.364367 6.379821 6.089691 6.017159 5.400634 5.473166 5.255569 5.436900
##      364      372      374      375      381      388      395      422
## 5.545698 5.001706 5.183037 5.763296 5.255569 5.328101 5.436900 6.307288
##      428      440      443      447      448      457      458      462
## 5.690763 5.219303 5.908360 5.654497 5.908360 5.581965 5.328101 5.219303
##      464      468      469      474      489      503      505      513 
## 5.110504 6.923813 5.183037 5.690763 5.763296 6.198490 5.763296 5.364367 
##      519      531      533      534      535      543      550      558 
## 6.089691 5.581965 5.581965 6.488619 5.473166 5.436900 5.255569 5.872094 
##      559      563      564      571      577      589      593      594 
## 6.089691 5.183037 5.509432 5.835828 5.473166 6.923813 5.364367 5.110504 
##      597      600      602      608      610      611      612      617 
## 5.219303 5.219303 5.110504 5.654497 6.452353 5.183037 5.255569 5.364367 
##      627      630      640      647      658      661      665      667 
## 5.146770 5.219303 5.581965 5.509432 5.654497 5.328101 5.545698 5.291835 
##      673      674      675      687      689      694      695      703 
## 5.291835 5.291835 5.219303 5.146770 5.255569 5.255569 5.255569 5.255569 
##      708      710      713      717      718      727      728      738 
## 5.654497 5.473166 5.255569 5.255569 5.473166 5.872094 5.291835 5.219303 
##      747      749      750      756      761      762      763      768 
## 5.110504 5.328101 5.291835 5.727029 5.364367 5.328101 5.473166 5.291835 
##      773      777      780      782      783      786      791      796 
## 5.255569 5.581965 5.400634 5.400634 5.473166 5.291835 5.364367 5.545698 
##      807      811      814      821      822      824      825      826 
## 6.379821 5.654497 5.980893 5.364367 6.923813 5.400634 5.581965 5.727029 
##      835      838      844      853      855      856      872      878 
## 5.255569 6.089691 5.255569 5.364367 5.799562 6.089691 5.690763 6.125957 
##      879      880      885      891      897      899      901      911 
## 5.473166 5.291835 5.473166 5.509432 6.379821 6.379821 6.125957 6.633684 
##      919      924      936      940      942      946      948      952 
## 5.835828 5.509432 6.125957 5.980893 6.379821 5.872094 6.343555 6.343555 
##      959      964      974      976      977      986      992     1003 
## 5.944626 5.908360 5.727029 5.255569 5.255569 5.944626 5.255569 6.089691 
##     1007     1020     1027     1028     1030     1031     1033     1041 
## 6.089691 5.618231 6.416087 6.053425 5.654497 6.017159 5.328101 5.545698 
##     1043     1051     1055     1065     1068     1069     1070     1071 
## 5.872094 5.618231 5.146770 6.125957 5.799562 5.799562 5.763296 5.872094 
##     1074     1081     1085     1092     1094     1103     1107     1124 
## 5.727029 6.125957 5.291835 5.980893 6.198490 5.908360 6.307288 5.908360 
##     1143     1144     1149     1151     1154     1160     1161     1164 
## 5.980893 5.473166 6.053425 6.488619 5.872094 5.763296 5.872094 5.473166 
##     1175     1178     1183     1185     1204     1205     1206     1208 
## 5.291835 6.452353 5.654497 5.799562 5.255569 5.835828 5.835828 5.473166 
##     1218     1222     1248     1259     1268     1275     1279     1280 
## 6.198490 6.017159 5.654497 5.944626 5.980893 5.908360 5.291835 6.017159 
##     1281     1284     1294     1298     1299     1301     1316     1317 
## 5.618231 5.545698 5.364367 6.234756 6.271022 6.379821 5.400634 6.379821 
##     1319     1331     1339     1346     1347     1352     1353     1354 
## 5.400634 5.400634 5.291835 5.545698 5.980893 6.125957 5.581965 5.581965 
##     1355     1368     1374     1376     1378     1380     1383     1385 
## 5.473166 5.654497 5.219303 5.436900 6.271022 5.763296 5.436900 5.291835 
##     1386     1391     1393     1404     1407     1413     1416     1418 
## 5.183037 6.379821 5.400634 5.473166 5.799562 5.799562 5.255569 6.234756 
##     1439     1440     1443     1446     1448     1449     1457     1460 
## 5.581965 5.860005 5.400634 5.328101 5.364367 5.364367 5.672630 6.162224 
##     1462     1470     1472     1485     1486     1494     1499     1501 
## 5.473166 5.364367 6.416087 5.799562 5.364367 5.364367 5.763296 5.328101 
##     1511     1516     1524     1525     1537     1542     1544     1551 
## 5.364367 5.201170 5.618231 5.473166 5.581965 5.799562 5.618231 5.291835 
##     1552     1561     1562     1575     1581     1582     1590     1596 
## 5.255569 5.436900 5.436900 5.654497 6.162224 5.944626 5.183037 5.908360
```

```
summary (lmMod)
```

```
## 
## Call:
## lm(formula = quality ~ alcohol, data = trainingData)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -2.8358 -0.4006 -0.1622  0.5208  2.5994 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  1.84655    0.19950   9.256   <2e-16 ***
## alcohol      0.36266    0.01903  19.059   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7209 on 1277 degrees of freedom
## Multiple R-squared:  0.2215, Adjusted R-squared:  0.2208 
## F-statistic: 363.2 on 1 and 1277 DF,  p-value: < 2.2e-16
```

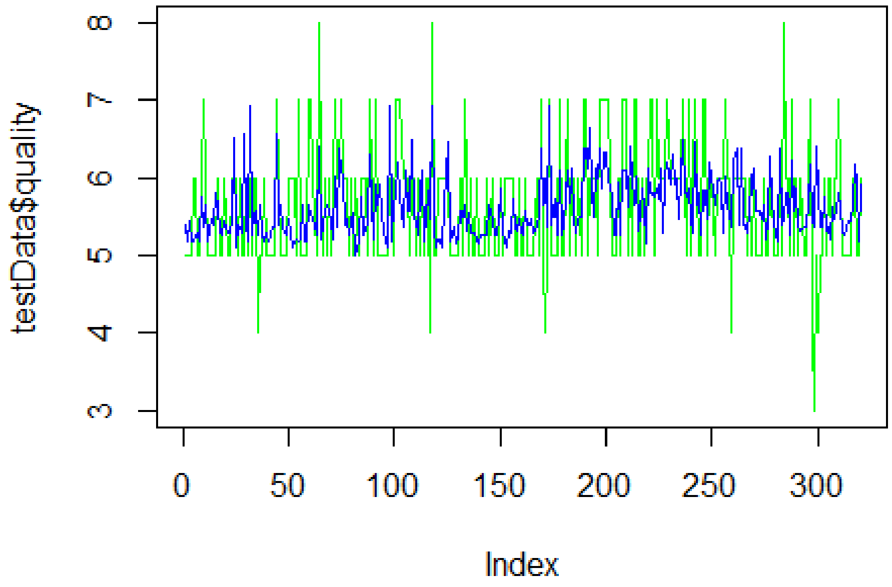
```
actuals_preds <- data.frame(cbind(actuals=testData$quality, predicteds=Pred))
correlation_accuracy <- cor(actuals_preds)
correlation_accuracy
```

```
##            actuals predicteds
## actuals    1.000000   0.502429
## predicteds 0.502429   1.000000
```

```
head(actuals_preds)
```

```
##    actuals predicteds
## 3        5   5.400634
## 11       5   5.183037
## 13       5   5.436900
## 15       5   5.183037
## 20       6   5.183037
## 21       6   5.255569
```

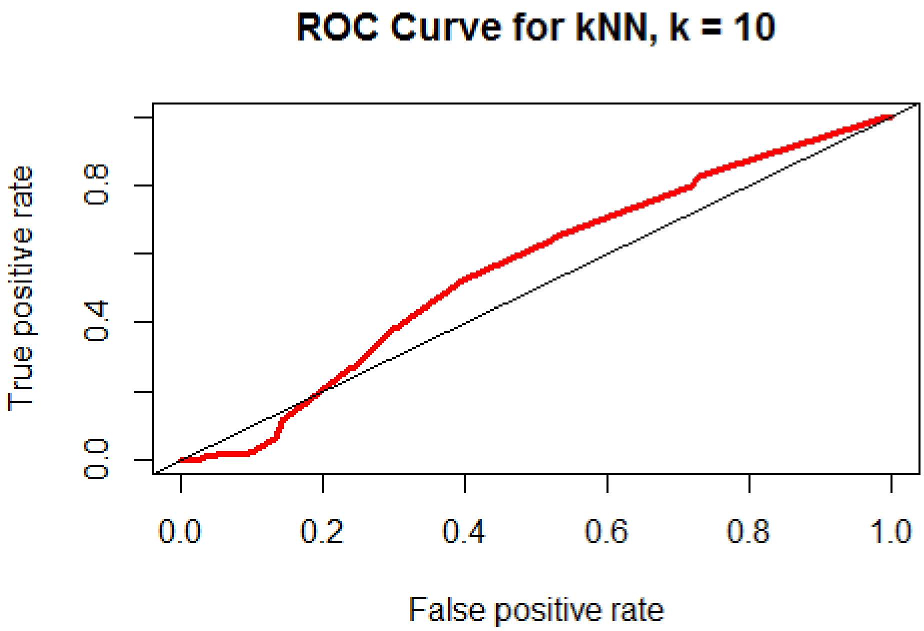
```
plot(testData$quality,type='l',lty=1.8,col="green")
lines(Pred,type='l',col="blue")
```



```
summary(r)

##
## Call:
## lm(formula = quality ~ alcohol, data = redwine)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8442 -0.4112 -0.1690  0.5166  2.5888
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.87497    0.17471   10.73   <2e-16 ***
## alcohol      0.36084    0.01668   21.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7104 on 1597 degrees of freedom
## Multiple R-squared:  0.2267, Adjusted R-squared:  0.2263
## F-statistic: 468.3 on 1 and 1597 DF,  p-value: < 2.2e-16
```

```
anova(r)

## Analysis of Variance Table
##
## Response: quality
##             Df Sum Sq Mean Sq F value    Pr(>F)
## alcohol      1 236.29 236.295  468.27 < 2.2e-16 ***
## Residuals 1597 805.87   0.505
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#rmse value is 0.7 the model is not good so we go for classification

#knn
# converting quality into a binary factor
for (i in 1:nrow(redwine)) {
  if (redwine$quality[i] > 5)
    redwine$label[i] <- 1
  else
    redwine$label[i] <- 0
}
redwine$label <- factor(redwine$label, levels = c(0, 1), labels = c("bad",
"good"))
# removing the quality and type variable
redwine$quality <- NULL
redwine$type<- NULL



# using a subset of 1000 obs for the training set
test_indices <- sample(1:nrow(redwine), 1000)
test <- redwine[test_indices,]
train <- redwine[-test_indices,]

set.seed(10)
library(class)
# using 20 nearest neighbors
knn_pred <- knn(train = train[,-12],
                test = test[,-12],
                cl = train$label,
                k = 35, prob = TRUE)
View(redwine)
# confusion matrix
knn_conf <- table(pred = knn_pred, true = test$label)
knn_conf

##       true
## pred   bad good
##   bad  228  117
##   good 245  410

# Creating the ROC curve for knn
# library(dplyr) is loaded
knn_prob <- attr(knn_pred, "prob")
knn_prob <- 2 * ifelse(knn_pred == "-1", 1-knn_prob, knn_prob) - 1
```

```
# confusion matrix
 knn_conf <- table(pred = knn_pred, true = test$label)
 knn_conf
      true
pred   bad good
  bad    0    0
  good   0    0
 # Plotting the KNN ROC curve
 > plot(knn_roc_perf, col = 2, lwd = 3,
 +      main = "ROC Curve for kNN, k = 10")
 > abline(0,1)
 > # Area under the knn curve
 > knn_auc_perf <- performance(knn_roc_pred, measure = "auc")
 > knn_AUC <- knn_auc_perf@y.values[[1]]
 > knn_AUC
 [1] 0.55741
```

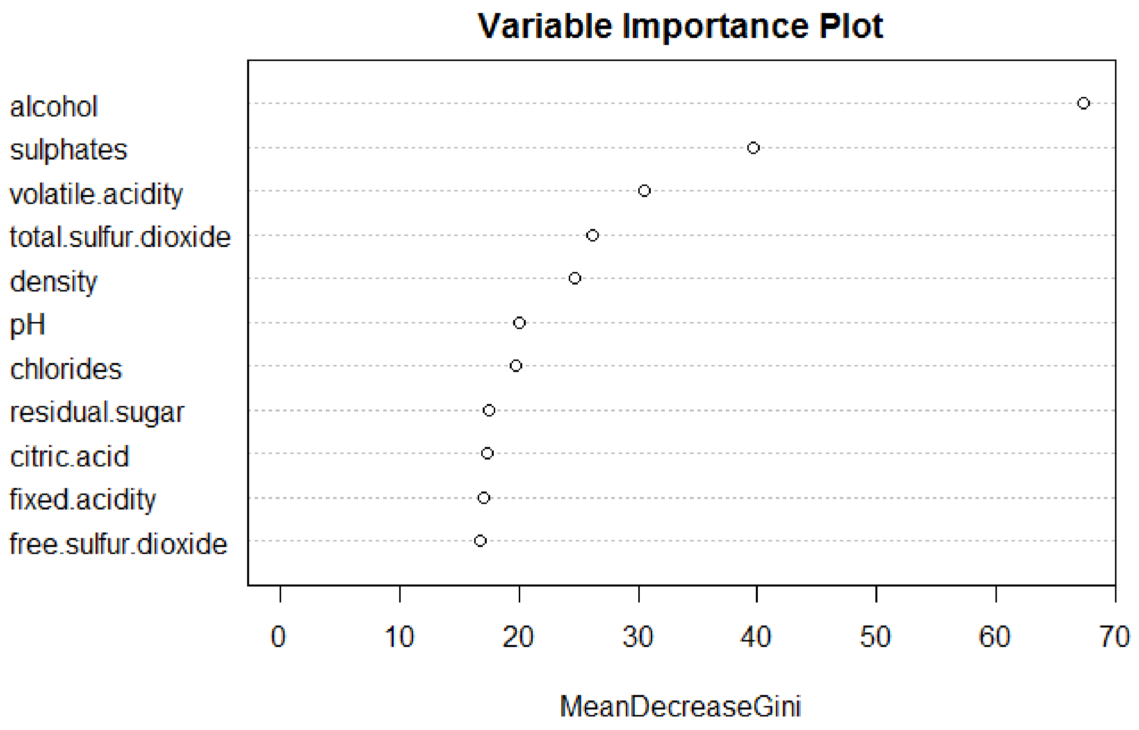


```
 >
```

```
#random forest

library(randomForest)

library(class)

rf <- randomForest(formula = label ~ .,

                   data = train,

                   mtry = 8)
```

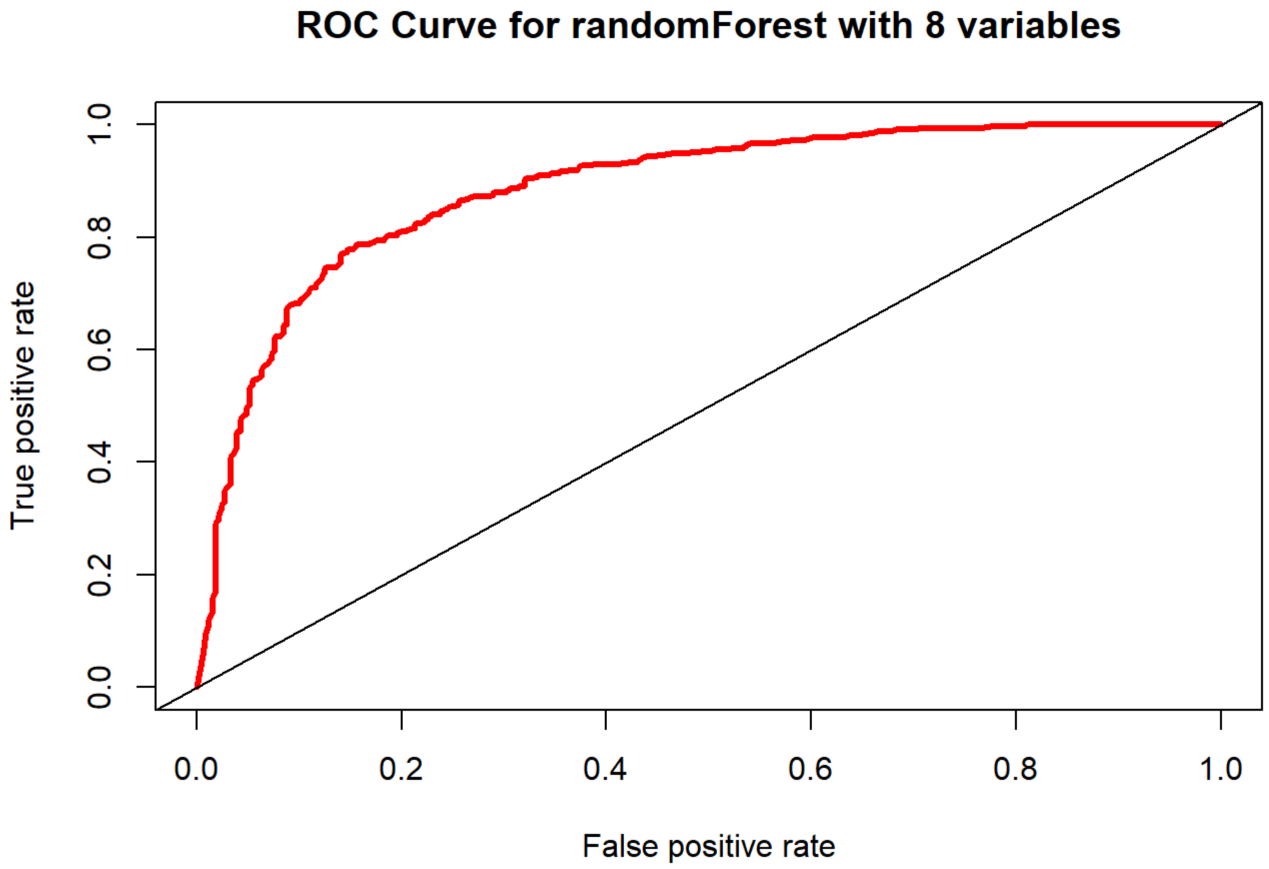
```
print(rf)

Call:
 randomForest(formula = label ~ ., data = train, mtry = 8)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 8

        OOB estimate of  error rate: 24.87%
Confusion matrix:
     bad good class.error
bad  194   78   0.2867647
good  71  256   0.2171254
```

```
varImpPlot(rf, main = "Variable Importance Plot")
```



```
rf_pred <- predict(rf, test, type = "class")
```

```
# predicting on the test set
rf_pred2 <- predict(rf, test, type = "class")
# Building the ROC Curve
> rf_pred2 <- as.data.frame(predict(rf, test, type = 'prob'))
> rf_pred_probs2 <- rf_pred2[,2]
> rf_roc_pred2 <- prediction(rf_pred_probs2, test$label)
> rf_perf2 <- performance(rf_roc_pred2,
+                         measure = "tpr",
                       x.measure = "fpr")
> # Plotting the curve
> plot(rf_perf2, col = 2, lwd = 8,
+      main = "ROC Curve for randomForest with 8 variables")
rf_AUC <- rf_perf2@y.values[[1]]
> rf_AUC
[1] 0.8519437
```



The model shows that the classification method Random forest is good when compare to all other model.