Red Wines - upper

Katie, Rita, and Chang

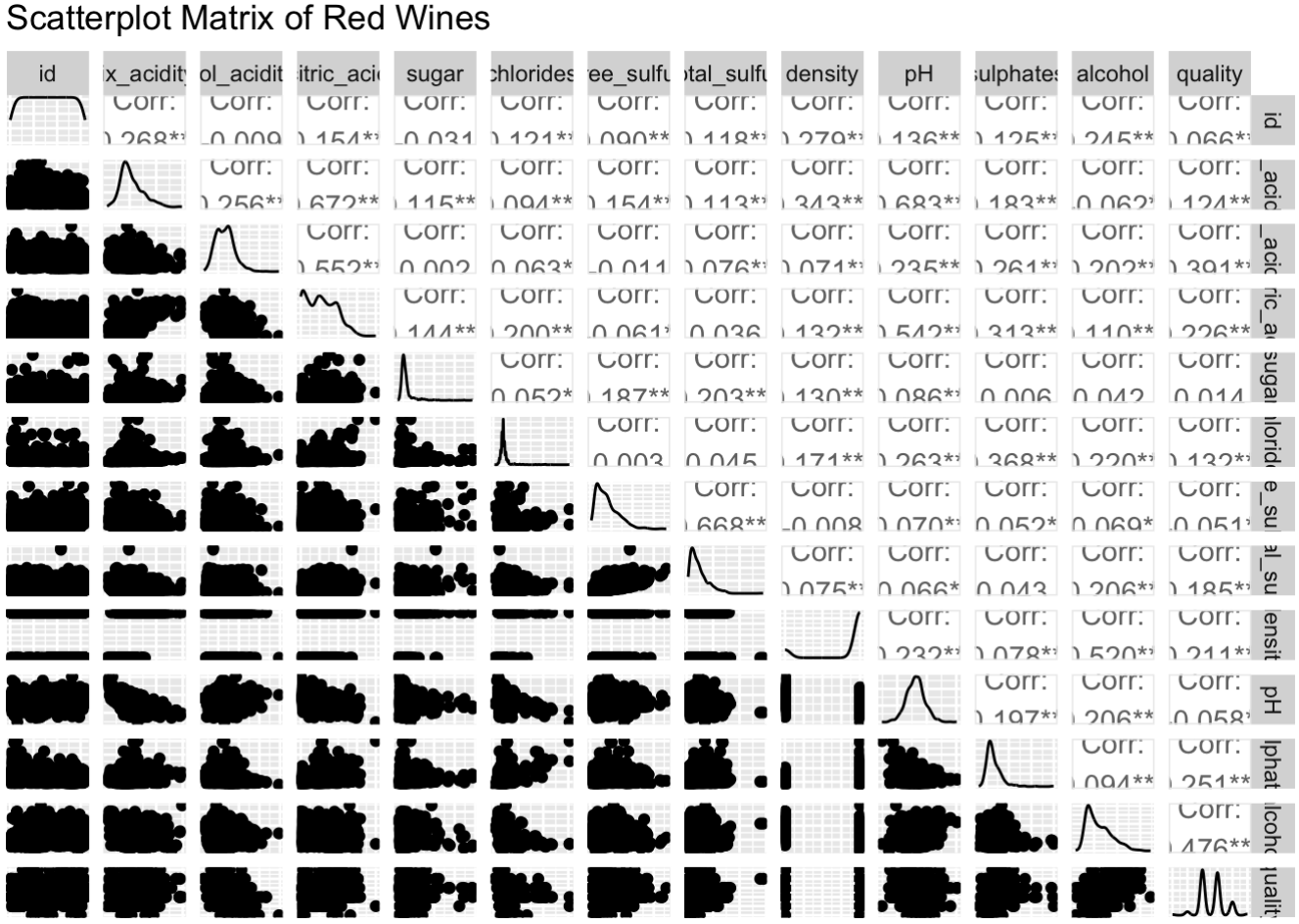
2023-11-01

Scatterplot Matrix

```
library("GGally")
ggpairs(red, axisLabels = "none",
        title = "Scatterplot Matrix of Red Wines")
```

```
# corr codes
```

Scatterplot Matrix



Create Binary Dependent Variable

```
red$highquality = factor((red$quality >= 6))
red$highquality <- as.integer(as.logical(red$highquality))
```

Create Test and Training Data

```
library("caTools")
set.seed = 100
split = sample.split(red$highquality, SplitRatio = 0.6)
train = subset(red, split == TRUE)
test = subset(red, split == FALSE)
print(dim(train)); print(dim(test))
```

```
## [1] 959  14
```

```
## [1] 640  14
```

Descriptive Statistics

```
library("Rmisc")
```

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

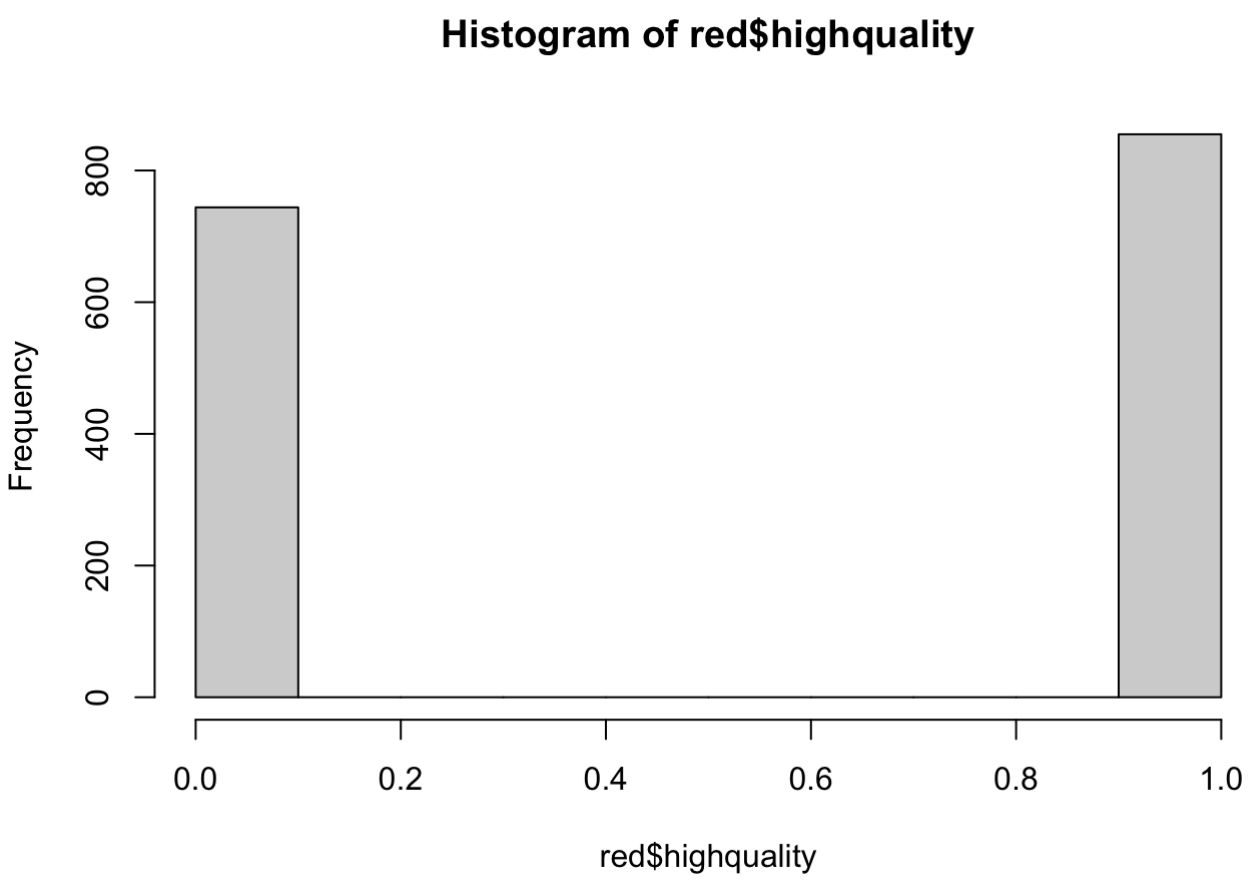
```
## Loading required package: plyr
```

```
sum = summary(red)
sum
```

```
##        id          fix_acidity     vol_acidity      citric_acid
##  Min.   :   1.0   Min.   : 4.60   Min.   :0.1200   Min.   :0.000
##  1st Qu.: 400.5   1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090
##  Median : 800.0   Median : 7.90   Median :0.5200   Median :0.260
##  Mean   : 800.0   Mean   : 8.32   Mean   :0.5284   Mean   :0.271
##  3rd Qu.:1199.5   3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420
##  Max.   :1599.0   Max.   :15.90   Max.   :1.5800   Max.   :1.000
##      sugar          chlorides        free_sulfur     total_sulfur
##  Min.   : 0.900   Min.   :0.01000   Min.   : 1.00   Min.   :  6.00
##  1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00   1st Qu.: 22.00
##  Median : 2.200   Median :0.08000   Median :14.00   Median : 38.00
##  Mean   : 2.539   Mean   :0.08787   Mean   :15.87   Mean   : 46.47
##  3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:21.00   3rd Qu.: 62.00
##  Max.   :15.500   Max.   :0.61000   Max.   :72.00   Max.   :289.00
##     density             pH          sulphates         alcohol
##  Min.   :0.9900   Min.   :2.740   Min.   :0.3300   Min.   : 8.40
##  1st Qu.:1.0000   1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50
##  Median :1.0000   Median :3.310   Median :0.6200   Median :10.20
##  Mean   :0.9985   Mean   :3.311   Mean   :0.6581   Mean   :10.42
##  3rd Qu.:1.0000   3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10
##  Max.   :1.0000   Max.   :4.010   Max.   :2.0000   Max.   :14.90
##     quality       highquality
##  Min.   :3.000   Min.   :0.0000
##  1st Qu.:5.000   1st Qu.:0.0000
##  Median :6.000   Median :1.0000
##  Mean   :5.636   Mean   :0.5347
##  3rd Qu.:6.000   3rd Qu.:1.0000
##  Max.   :8.000   Max.   :1.0000
```

Plot high quality vs low quality distribution

```
hist (red$highquality)
```



Random Forest

```
library("randomForest")
```

```
## randomForest 4.7-1.1
```

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

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

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

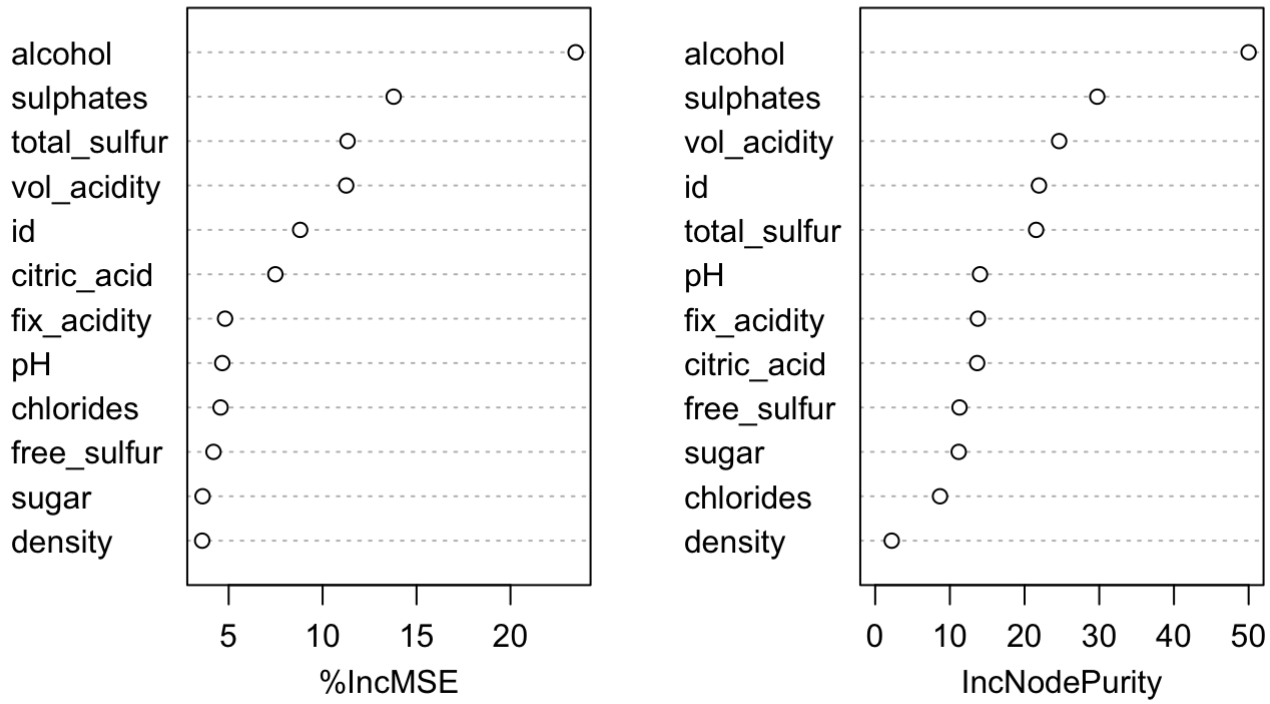
```
library("caret")
library("e1071")
library("rpart")

rf <- randomForest(highquality ~ . - quality, data = train, mtry = 4, importance = TRUE,
ntree = 50, na.action = na.omit)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
print(rf)
```

```
##
## Call:
##  randomForest(formula = highquality ~ . - quality, data = train,      mtry = 4, impor
tance = TRUE, ntree = 50, na.action = na.omit)
##                Type of random forest: regression
##                      Number of trees: 50
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 0.1566168
##                     % Var explained: 37.05
```

```
varImpPlot(rf)
```




```
# predictions on test set
set.seed(100)
predictTest = predict(rf, newdata = test, type = "response")

# confusion matrix on test set
table(test$highquality, predictTest >= 0.5)
```

```
##
##     FALSE TRUE
##   0   231   67
##   1    65  277
```

```
520/nrow(test)
```

```
## [1] 0.8125
```

```
# the model is accurate 81.3 percent of the time
```

Random Forest Model

```
# Logit
randomforestmodlogit <- glm(highquality ~ alcohol + sulphates + total_sulfur + vol_acidi
ty, data = red, family = "binomial"(link = "logit"))
summary(randomforestmodlogit)
```

```
##
## Call:
## glm(formula = highquality ~ alcohol + sulphates + total_sulfur +
##     vol_acidity, family = binomial(link = "logit"), data = red)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1638  -0.8675   0.3076   0.8629   2.3262
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.588813   0.795118 -10.802  < 2e-16 ***
## alcohol       0.927362   0.069268  13.388  < 2e-16 ***
## sulphates     2.059047   0.365976   5.626 1.84e-08 ***
## total_sulfur -0.011976   0.001924  -6.225 4.83e-10 ***
## vol_acidity  -3.083277   0.364832  -8.451  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1684.2  on 1594  degrees of freedom
## AIC: 1694.2
##
## Number of Fisher Scoring iterations: 4
```

```
# Cloglog
randomforestmodcloglog <- glm(highquality ~ alcohol + sulphates + total_sulfur + vol_aci
dity, data = red, family = "binomial"(link = "cloglog"))
summary(randomforestmodcloglog)
```

```
##
## Call:
## glm(formula = highquality ~ alcohol + sulphates + total_sulfur +
##     vol_acidity, family = binomial(link = "cloglog"), data = red)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5006  -0.9020   0.2185   0.9295   2.0506
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.958517   0.478252 -10.368  < 2e-16 ***
## alcohol       0.505807   0.038543  13.123  < 2e-16 ***
## sulphates     1.324184   0.221318   5.983 2.19e-09 ***
## total_sulfur -0.009109   0.001364  -6.679 2.41e-11 ***
## vol_acidity  -2.022997   0.238813  -8.471  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209  on 1598  degrees of freedom
## Residual deviance: 1701  on 1594  degrees of freedom
## AIC: 1711
##
## Number of Fisher Scoring iterations: 7
```

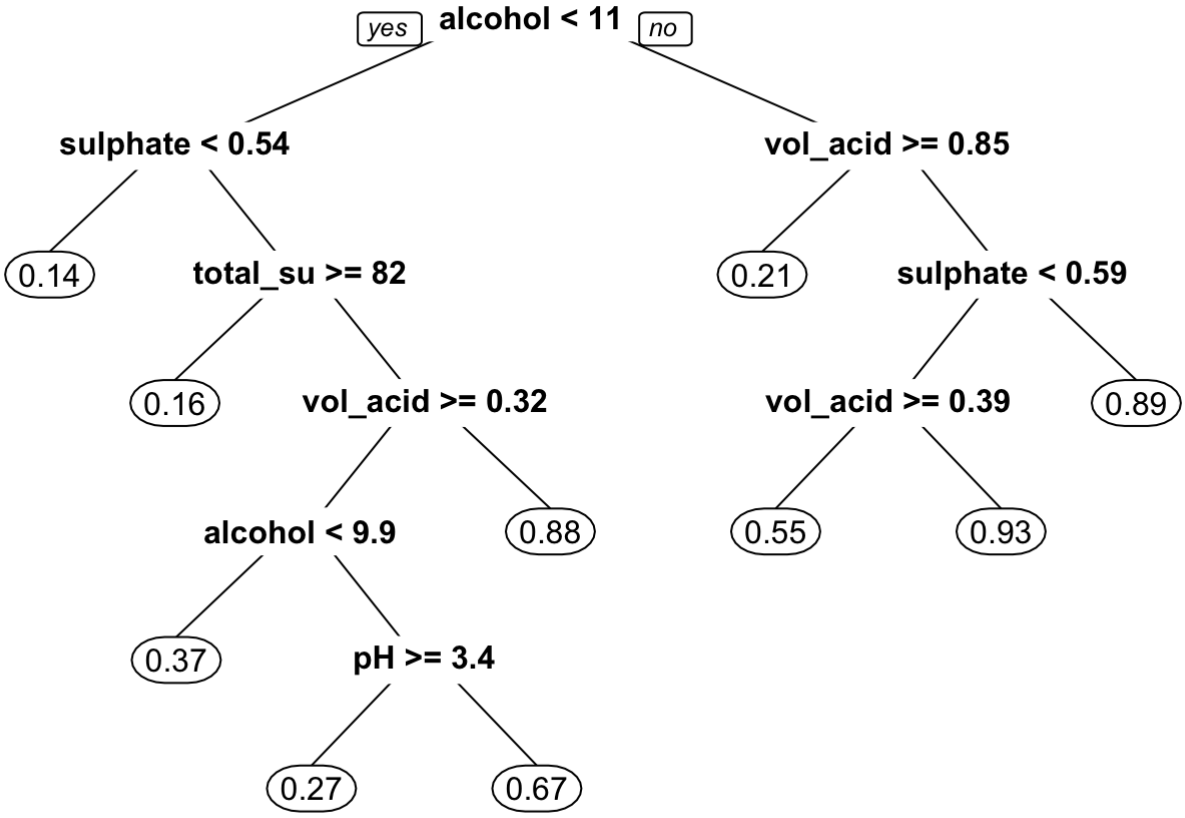
```
# The logit model performed better with a lower AIC value
```

Cart

```
library("caret")
library("e1071")
library("rpart")
library("rpart.plot")

cartmodel = rpart(highquality ~ . - quality, data = train)
print(cartmodel)
```

```
## n= 959
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 959 238.579800 0.5349322
##    2) alcohol< 10.525 593 138.381100 0.3709949
##      4) sulphates< 0.535 145  17.958620 0.1448276 *
##      5) sulphates>=0.535 448 110.604900 0.4441964
##       10) total_sulfur>=82 75  10.080000 0.1600000 *
##       11) total_sulfur< 82 373  93.249330 0.5013405
##         22) vol_acidity>=0.315 339  84.289090 0.4631268
##           44) alcohol< 9.85 193  45.139900 0.3730570 *
##           45) alcohol>=9.85 146  35.513700 0.5821918
##             90) pH>=3.425 33   6.545455 0.2727273 *
##             91) pH< 3.425 113  24.884960 0.6725664 *
##         23) vol_acidity< 0.315 34   3.529412 0.8823529 *
##    3) alcohol>=10.525 366  58.439890 0.8005464
##      6) vol_acidity>=0.845 19   3.157895 0.2105263 *
##      7) vol_acidity< 0.845 347  48.305480 0.8328530
##       14) sulphates< 0.585 87  19.333330 0.6666667
##         28) vol_acidity>=0.385 60  14.850000 0.5500000 *
##         29) vol_acidity< 0.385 27   1.851852 0.9259259 *
##       15) sulphates>=0.585 260  25.765380 0.8884615 *
```

```
prp(cartmodel)
```




```
# predictions on test set
set.seed(100)
predictTest = predict(cartmodel, newdata = test, type = "matrix")

# confusion matrix on test set
table(test$highquality, predictTest >= 0.5)
```

```
##
##     FALSE TRUE
##   0   223   75
##   1    93  249
```

Cart Model

```
# Logit
cartmodlogit <- glm(highquality ~ alcohol + sulphates + total_sulfur + fix_acidity, data
= red, family = "binomial"(link = "logit"))
summary(cartmodlogit)
```

```
##
## Call:
## glm(formula = highquality ~ alcohol + sulphates + total_sulfur +
##     fix_acidity, family = binomial(link = "logit"), data = red)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3737  -0.9154   0.3562   0.8762   2.0206
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -12.172146   0.828396 -14.694  < 2e-16 ***
## alcohol        0.989178   0.068276  14.488  < 2e-16 ***
## sulphates      2.587844   0.370028   6.994 2.68e-12 ***
## total_sulfur  -0.011171   0.001895  -5.895 3.75e-09 ***
## fix_acidity    0.109461   0.035511   3.082  0.00205 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1752.5  on 1594  degrees of freedom
## AIC: 1762.5
##
## Number of Fisher Scoring iterations: 4
```

```
# Cloglog
cartmodcloglog <- glm(highquality ~ alcohol + sulphates + total_sulfur + fix_acidity, da
ta = red, family = "binomial"(link = "cloglog"))
summary(cartmodcloglog)
```

```
##
## Call:
## glm(formula = highquality ~ alcohol + sulphates + total_sulfur +
##     fix_acidity, family = binomial(link = "cloglog"), data = red)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7058  -0.9408   0.3075   0.9490   1.9387
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.835907   0.481268 -14.204  < 2e-16 ***
## alcohol       0.542953   0.037720  14.394  < 2e-16 ***
## sulphates     1.639060   0.217233   7.545 4.52e-14 ***
## total_sulfur -0.009315   0.001384  -6.732 1.67e-11 ***
## fix_acidity   0.027351   0.021284   1.285    0.199
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1777.5  on 1594  degrees of freedom
## AIC: 1787.5
##
## Number of Fisher Scoring iterations: 18
```

```
# The logit model performed better with the lower AIC value
```

Compare best logit model with AIC

```
library("AICcmodavg")
```

```
##
## Attaching package: 'AICcmodavg'
```

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

```
models <- list(randomforestmodlogit, cartmodlogit)
mod.names <- c('RandomForest', 'Cart')
aictab(cand.set = models, modnames = mod.names)
```

```
##
## Model selection based on AICc:
##
##              K    AICc Delta_AICc AICcWt Cum.Wt      LL
## RandomForest 5 1694.21       0.00      1      1 -842.09
## Cart         5 1762.56      68.35      0      1 -876.26
```

```
# The random forest logit model performed the best
```

Compare best model with BIC

```
library("flexmix")
BIC(randomforestmodlogit)
```

```
## [1] 1721.058
```

```
BIC(randomforestmodcloglog)
```

```
## [1] 1737.845
```

```
BIC(cartmodlogit)
```

```
## [1] 1789.404
```

```
BIC(cartmodcloglog)
```

```
## [1] 1814.418
```

```
# The random forest logit model performed the best
```

Confusion matrix for random forest logit model

```
confusionred = predict(randomforestmodlogit, newdata = red, type = "response")

# confusion matrix on test set
table(red$highquality, confusionred >= 0.5)
```

```
##
##      FALSE TRUE
##   0   548  196
##   1   216  639
```

Predictions for random forest logit model

```
pred_test <- predict(randomforestmodlogit, test, type = "response")

pred_test
```

```
##          1          4          6          8          9         13         14
## 0.21686414 0.52789064 0.22573317 0.35364079 0.35226650 0.27778694 0.69723806
##         16         21         24         31         37         38         40
## 0.13867752 0.45581006 0.25095181 0.23338984 0.65305951 0.59357547 0.60541307
##         43         44         48         49         50         60         61
## 0.78970651 0.75462869 0.51951414 0.43019840 0.27520137 0.37446288 0.35540196
##         64         65         66         69         72         76         78
## 0.22177359 0.48498736 0.48498736 0.76772166 0.11856745 0.67706290 0.41857071
##         81         87         91         92         94         96         98
## 0.38434763 0.81269343 0.12488112 0.81269343 0.43771981 0.81609562 0.36239110
##         99        100        103        106        109        114        116
## 0.22403417 0.24182616 0.24182616 0.17168420 0.49637573 0.56940622 0.61534547
##        121        125        126        127        129        130        132
## 0.03125237 0.17454268 0.19469456 0.15237332 0.73731458 0.47573192 0.88460563
##        134        135        137        138        142        147        152
## 0.46022375 0.13831261 0.26189520 0.36049701 0.26189520 0.18741376 0.86018228
##        155        156        158        160        161        166        167
## 0.44034290 0.43822117 0.43822117 0.15834014 0.15424192 0.16875609 0.22409934
##        173        174        179        181        183        185        189
## 0.42272086 0.69738295 0.21924830 0.27931853 0.16145478 0.19399307 0.13003255
##        190        196        202        207        213        214        215
## 0.13218057 0.11752977 0.32718865 0.85218494 0.67118625 0.29462991 0.30759231
##        217        220        221        223        226        228        230
## 0.56377995 0.12986042 0.40137970 0.36725609 0.56522890 0.26752645 0.64457164
##        231        233        234        237        238        241        247
## 0.77766942 0.47994817 0.64457164 0.19063712 0.20849103 0.39273703 0.19535300
##        252        254        255        257        258        260        267
## 0.44429231 0.11931143 0.44429231 0.50388022 0.18145879 0.80917975 0.19291723
##        268        272        274        278        282        284        288
## 0.97152743 0.87483178 0.22637701 0.87483178 0.86571544 0.71864536 0.63094835
##        289        292        293        294        297        299        300
## 0.69463023 0.85547233 0.50253385 0.67070411 0.14998798 0.42029514 0.33509483
##        301        303        307        310        312        314        319
## 0.59572880 0.41165107 0.20435986 0.51474055 0.16092124 0.13631050 0.68916140
##        322        325        326        327        329        331        332
## 0.22451563 0.32170068 0.32170068 0.85787373 0.66866920 0.94054119 0.94054119
##        338        340        341        342        344        345        346
## 0.56472573 0.88983001 0.74512435 0.89763321 0.66228068 0.68982583 0.34930907
##        347        348        351        352        354        357        359
## 0.84745884 0.94069892 0.53418258 0.28077747 0.92386355 0.81776585 0.76935573
##        361        364        369        372        373        377        387
## 0.17383851 0.74339678 0.37745429 0.55754674 0.87743922 0.87515687 0.31578174
##        390        391        392        394        400        401        404
## 0.59089106 0.80025092 0.62763464 0.09608587 0.16572036 0.11072664 0.52614838
##        411        413        415        417        419        421        424
## 0.29774721 0.13582974 0.16976587 0.86998145 0.77020269 0.79052736 0.91967155
##        426        430        431        433        434        435        437
## 0.69125350 0.32784541 0.91967155 0.96051533 0.45313982 0.62568563 0.31324574
##        439        444        445        450        451        453        454
## 0.62568563 0.86779419 0.93413308 0.74289381 0.74289381 0.45914078 0.88497153
##        458        461        462        465        466        472        473
## 0.22920186 0.82420951 0.29995555 0.55427468 0.74550780 0.76084070 0.68760803
```

```
##         475         480         485         486         492         494         496
## 0.69215031 0.43048988 0.97215159 0.35050201 0.97829349 0.64444930 0.84287574
##         499         504         505         509         510         512         514
## 0.84287574 0.92902168 0.92990496 0.40270830 0.87805407 0.40270830 0.87663195
##         515         518         519         520         522         524         525
## 0.87663195 0.17161954 0.93281411 0.69166380 0.35179506 0.15801090 0.23574293
##         527         529         531         540         541         542         543
## 0.69166380 0.28289255 0.85077943 0.79067776 0.33434428 0.81658345 0.34240899
##         544         546         547         550         551         554         557
## 0.60467189 0.16061952 0.44918649 0.36606459 0.37808419 0.26817772 0.82873036
##         561         565         566         567         568         569         570
## 0.74635299 0.92935383 0.74635299 0.20667837 0.20667837 0.76484762 0.86080487
##         571         572         573         574         579         586         588
## 0.87692510 0.86080487 0.77494478 0.28595807 0.26273769 0.41223315 0.09780108
##         590         594         596         601         602         614         615
## 0.83511782 0.25462429 0.12498036 0.18854205 0.28353983 0.61614952 0.52341231
##         619         621         623         624         629         631         632
## 0.66590198 0.16493964 0.30620045 0.92396940 0.31752796 0.31752796 0.60526652
##         634         642         644         645         649         653         656
## 0.32737388 0.24690166 0.24690166 0.32321697 0.79606704 0.99329501 0.27966086
##         657         659         660         666         668         670         677
## 0.40958829 0.53465925 0.51004428 0.30272887 0.51847411 0.51847411 0.49457701
##         678         680         681         682         683         684         690
## 0.22140871 0.50931169 0.26144196 0.53604318 0.37713648 0.61274001 0.71405217
##         692         694         696         697         700         701         707
## 0.08792767 0.17710261 0.92968944 0.33118031 0.78652604 0.09555064 0.27290235
##         710         711         712         716         717         721         726
## 0.66657651 0.09703634 0.16497241 0.27155586 0.30467669 0.31642036 0.49939352
##         729         734         735         736         737         742         746
## 0.43341582 0.31147678 0.29287694 0.11882456 0.11882456 0.12964712 0.45423361
##         750         751         754         755         756         757         761
## 0.45423361 0.24423781 0.24423781 0.54228709 0.24061283 0.33804274 0.18944864
##         762         763         767         769         770         773         774
## 0.21832154 0.35211810 0.18768624 0.19045961 0.23550990 0.09786165 0.58787221
##         776         777         782         784         787         789         790
## 0.16206695 0.32070205 0.49407054 0.49407054 0.36376503 0.37973849 0.09437875
##         791         794         795         796         797         799         802
## 0.42161900 0.63971010 0.95029362 0.45940264 0.41405130 0.66715805 0.42345371
##         807         808         809         815         818         820         821
## 0.97273626 0.97678970 0.48396705 0.84554487 0.93891093 0.21151380 0.32105800
##         822         831         833         836         838         839         842
## 0.98401461 0.60268443 0.60619390 0.16263773 0.73172430 0.90163251 0.46356092
##         843         849         861         864         866         867         869
## 0.65608836 0.38077515 0.16845476 0.19469921 0.18470531 0.85828874 0.81336010
##         870         874         875         877         879         882         883
## 0.59381839 0.89396556 0.91690605 0.74967371 0.27035098 0.66426561 0.94676764
##         884         886         887         888         889         890         894
## 0.17504411 0.44881228 0.39943319 0.87920584 0.60509698 0.07413691 0.18472125
##         906         907         913         917         918         919         920
## 0.15063496 0.61655717 0.92326068 0.47643991 0.57254719 0.73857291 0.84865159
##         922         926         934         935         939         941         944
## 0.73857291 0.85851776 0.37796683 0.61571145 0.96015192 0.95924086 0.39208642
```

```
##         949         952         956         960         962         964         965
## 0.96739607 0.96007673 0.81941210 0.43498225 0.34752279 0.88720837 0.82503678
##         972         973         974         975         976         977         979
## 0.91601469 0.89840541 0.80832015 0.93910091 0.28346202 0.28346202 0.78896295
##         982         983         988         990         991         994         997
## 0.26394962 0.93219598 0.31207909 0.80108051 0.40407434 0.32169795 0.91773179
##        1002        1004        1006        1016        1017        1022        1024
## 0.78374144 0.96113831 0.96113831 0.89906458 0.95824366 0.91151720 0.94561683
##        1025        1026        1030        1031        1034        1036        1037
## 0.57387540 0.31368976 0.57387540 0.77233191 0.54729817 0.77472663 0.94495889
##        1038        1043        1044        1049        1050        1052        1053
## 0.08467347 0.80300719 0.82020006 0.80953886 0.78021636 0.67804325 0.91104211
##        1055        1058        1059        1062        1063        1065        1070
## 0.14570490 0.20603080 0.81405117 0.95253002 0.87540468 0.75516090 0.54212500
##        1071        1072        1073        1074        1079        1080        1083
## 0.88863261 0.08796075 0.36308322 0.54325401 0.82775448 0.40453879 0.36030691
##        1084        1085        1086        1092        1094        1095        1098
## 0.89315798 0.36030691 0.28421079 0.87464611 0.95181067 0.49531036 0.34983074
##        1100        1103        1106        1108        1109        1110        1113
## 0.34983074 0.74464695 0.89775611 0.94262703 0.20906211 0.67961164 0.83718598
##        1114        1115        1117        1118        1119        1122        1125
## 0.64019617 0.96048872 0.67849027 0.67849027 0.96598962 0.88827487 0.80352421
##        1127        1129        1136        1138        1139        1142        1150
## 0.98230418 0.42390723 0.90991385 0.85680402 0.20660676 0.79255354 0.91677782
##        1154        1156        1158        1160        1164        1165        1169
## 0.78814787 0.32861855 0.91555543 0.75007048 0.34911691 0.34911691 0.89257541
##        1174        1176        1180        1183        1184        1186        1187
## 0.40354568 0.71702966 0.83725702 0.73870157 0.18325571 0.83216767 0.75844840
##        1189        1190        1193        1194        1204        1211        1213
## 0.39228204 0.12264121 0.96428659 0.39866755 0.12664335 0.53707859 0.53707859
##        1217        1220        1221        1224        1226        1228        1230
## 0.25697732 0.80645578 0.89545422 0.93813689 0.23830590 0.43944184 0.35954537
##        1231        1239        1242        1247        1250        1251        1260
## 0.93764299 0.21802710 0.44890191 0.23445159 0.61830476 0.61830476 0.69403671
##        1261        1264        1269        1272        1273        1274        1275
## 0.44018286 0.18918454 0.37253912 0.78782193 0.71983985 0.19873507 0.66138806
##        1278        1284        1286        1290        1292        1293        1297
## 0.35315581 0.35403270 0.64986391 0.52796774 0.59190236 0.93827703 0.15149263
##        1298        1300        1302        1304        1306        1308        1311
## 0.87066317 0.06683777 0.61081590 0.81440601 0.20683713 0.58216493 0.20683713
##        1312        1313        1316        1317        1319        1320        1321
## 0.93088901 0.11783860 0.20771741 0.79987457 0.20771741 0.37260225 0.27840724
##        1323        1326        1328        1329        1331        1332        1333
## 0.86432689 0.65726913 0.65726913 0.27847288 0.21628945 0.06947426 0.44771727
##        1336        1337        1339        1340        1342        1346        1349
## 0.89912713 0.33063230 0.33063230 0.57850387 0.57850387 0.56699331 0.23966391
##        1351        1357        1358        1360        1361        1362        1369
## 0.29613263 0.57441114 0.68815611 0.55865303 0.63469844 0.20131694 0.15039302
##        1371        1372        1376        1379        1380        1381        1386
## 0.30081941 0.91160867 0.22071490 0.47985390 0.62870483 0.62870483 0.06251845
##        1388        1389        1390        1392        1394        1395        1396
## 0.32286340 0.45091508 0.21224158 0.60775822 0.38905380 0.15440151 0.33954980
```

```
##       1397       1400       1401       1402       1404       1406       1407
## 0.34921494 0.69724047 0.10211360 0.10211360 0.85527482 0.92222761 0.92066898
##       1409       1410       1414       1415       1418       1420       1423
## 0.96956379 0.73685309 0.45197689 0.67996830 0.93856044 0.11430274 0.80043906
##       1424       1426       1427       1430       1432       1435       1436
## 0.72836169 0.66331164 0.93501778 0.92040632 0.37563680 0.18849461 0.18849461
##       1437       1440       1442       1443       1447       1449       1450
## 0.16556629 0.47741729 0.09511377 0.43582198 0.43582198 0.37160792 0.87394441
##       1454       1456       1457       1461       1466       1468       1474
## 0.17120492 0.45960983 0.37601586 0.59735007 0.33900665 0.30916495 0.58301194
##       1475       1477       1478       1479       1481       1483       1485
## 0.19663663 0.19663663 0.96511629 0.28090696 0.49670748 0.55985828 0.47494678
##       1488       1493       1494       1495       1497       1498       1503
## 0.66158978 0.79984815 0.14135590 0.69740812 0.14135590 0.71090938 0.28363547
##       1504       1508       1509       1510       1511       1512       1514
## 0.81885400 0.85817699 0.83087548 0.91307103 0.57620576 0.36079233 0.46974028
##       1515       1516       1517       1522       1523       1525       1533
## 0.12840124 0.13049129 0.82595385 0.25612152 0.82595385 0.60658881 0.52363402
##       1534       1538       1540       1543       1546       1547       1548
## 0.23235943 0.50712482 0.56882617 0.36046967 0.45450879 0.51435731 0.82069018
##       1549       1552       1555       1557       1558       1559       1560
## 0.60297285 0.24069329 0.48014133 0.23011305 0.48014133 0.12503776 0.14727854
##       1562       1566       1571       1572       1573       1574       1575
## 0.14727854 0.65453799 0.96429949 0.82810158 0.10892466 0.88033989 0.51978506
##       1577       1579       1580       1581       1583       1587       1588
## 0.85326672 0.72404537 0.73417847 0.90818659 0.71768308 0.88684408 0.65976944
##       1593       1596       1599
## 0.75118218 0.74190964 0.81940411
```

Model Diagnostics

```
accuracy = (548+639)/(548+196+216+639)
accuracy
```

```
## [1] 0.742339
```

```
sensitivity = 639/(639+196)
sensitivity
```

```
## [1] 0.7652695
```

```
specificity = 548/(548+216)
specificity
```

```
## [1] 0.7172775
```

AUC and ROC

```
library("pROC")
```

```
## Type 'citation("pROC")' for a citation.
```

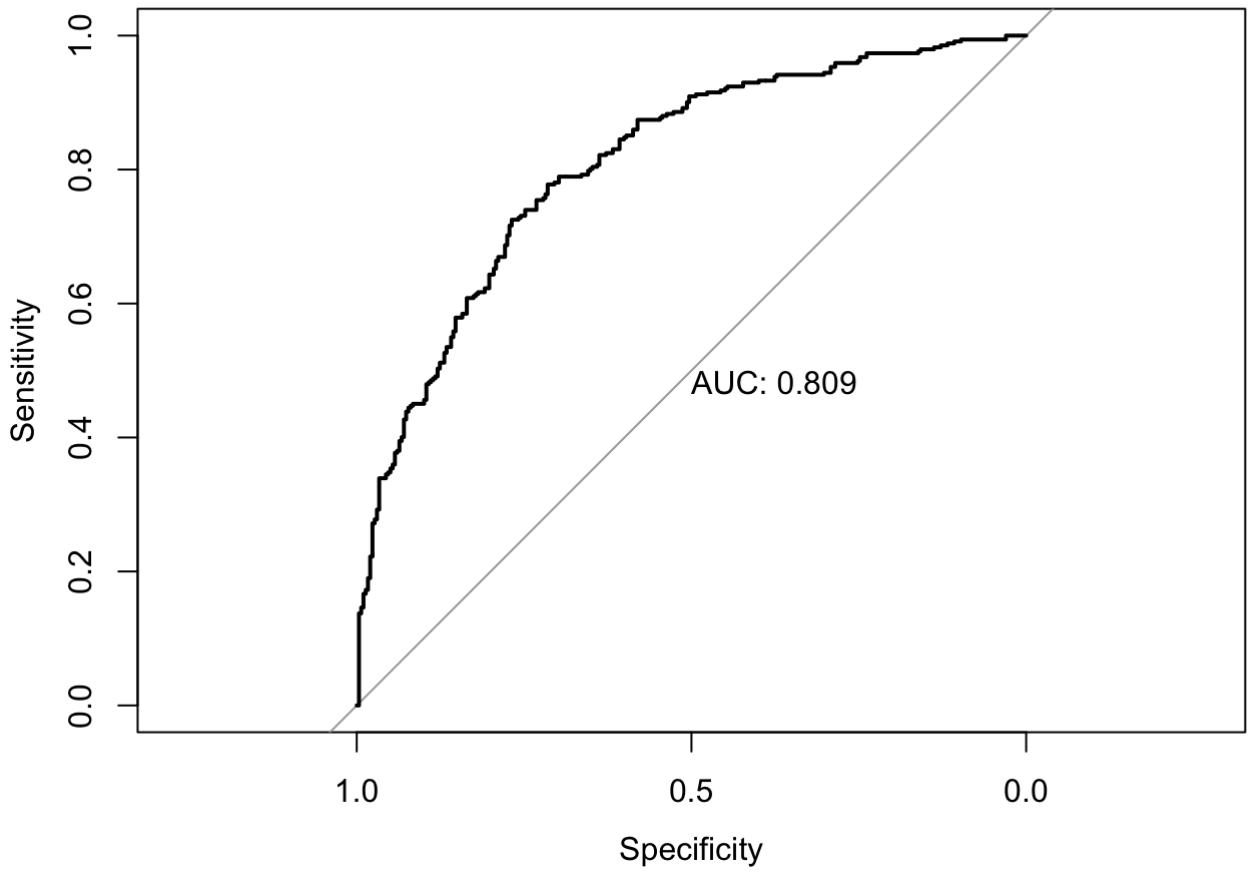
```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
test_prob = predict(randomforestmodlogit, test, type = "response")

test_roc = roc(test$highquality ~ test_prob, plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
as.numeric(test_roc$auc)
```

```
## [1] 0.8093822
```

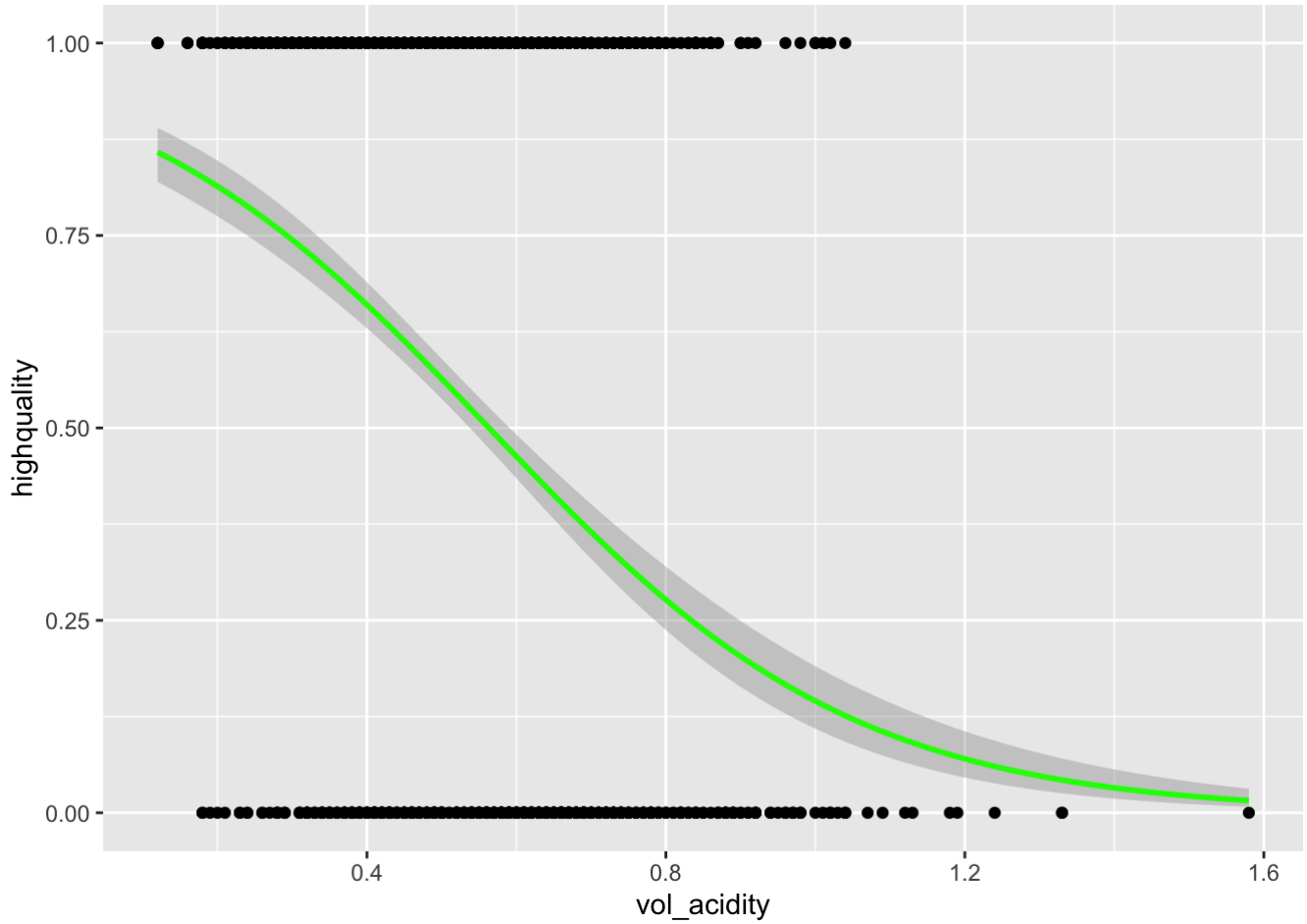
AUC and ROC with just one variable

```
library("ggplot2")

simple <- glm(highquality ~ vol_acidity, data = red, family = "binomial"(link = "logi
t"))
summary(simple)
```

```
##
## Call:
## glm(formula = highquality ~ vol_acidity, family = binomial(link = "logit"),
##     data = red)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8697  -1.1148   0.7156   1.0375   2.0349
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.2874     0.1838   12.45   <2e-16 ***
## vol_acidity  -4.0607     0.3334  -12.18   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 2033.4  on 1597  degrees of freedom
## AIC: 2037.4
##
## Number of Fisher Scoring iterations: 4
```

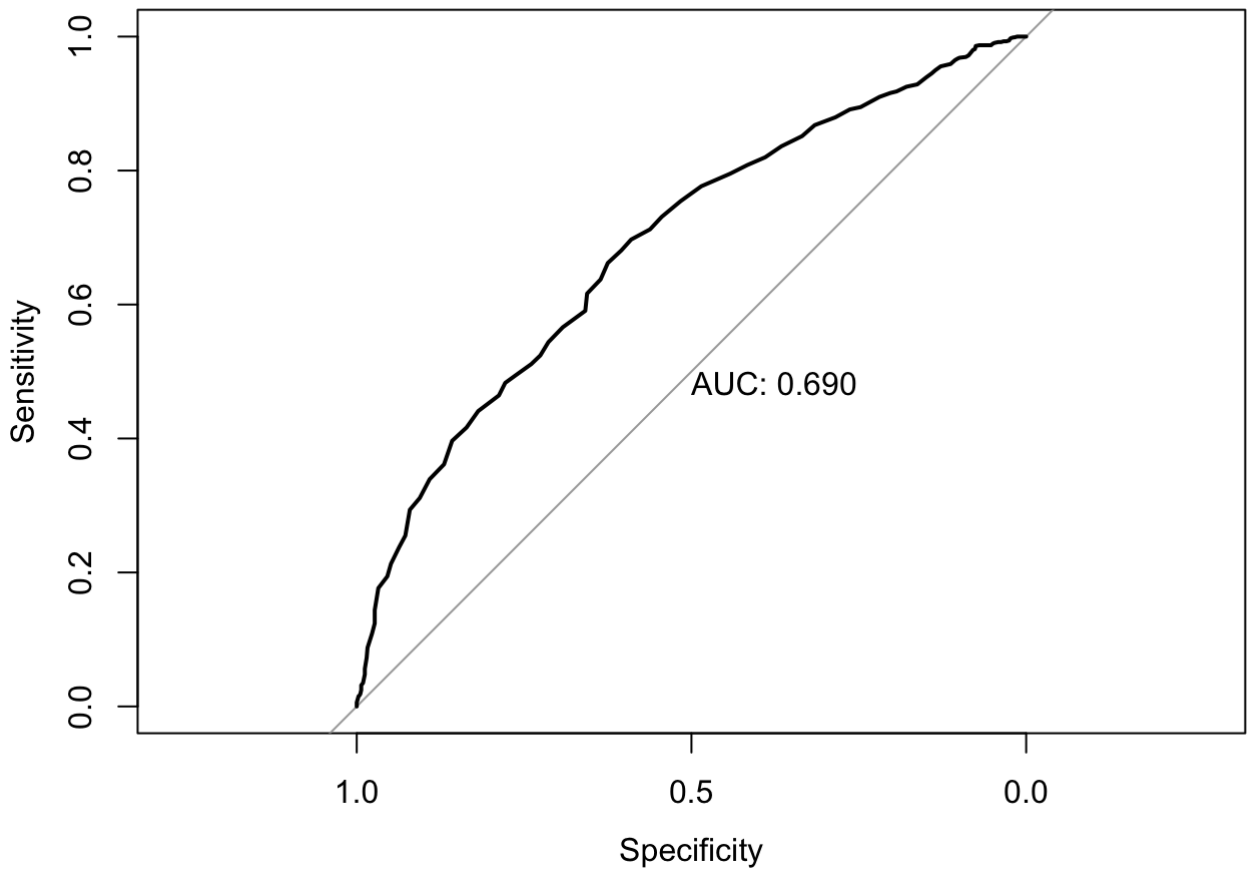
```
ggplot(red, aes(x = vol_acidity, y = highquality)) +geom_point()+stat_smooth(method="gl
m", color="green", se=TRUE, method.args = list(family=binomial))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
test_prop1 = predict(simple, red, type = "response")

test_roc1 = roc(red$highquality ~ test_prop1, plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```




```
as.numeric(test_roc1$auc)
```

```
## [1] 0.6900011
```