

Bruce Campbell Final Project

Fri Jul 29 08:12:55 2016

```
rm(list = ls())
set.seed(7)

setwd("C:/st-617/")

## use read.csv2 since data fields are delimited by semicolons (;)

cls = c(rep("numeric", 11), "integer")

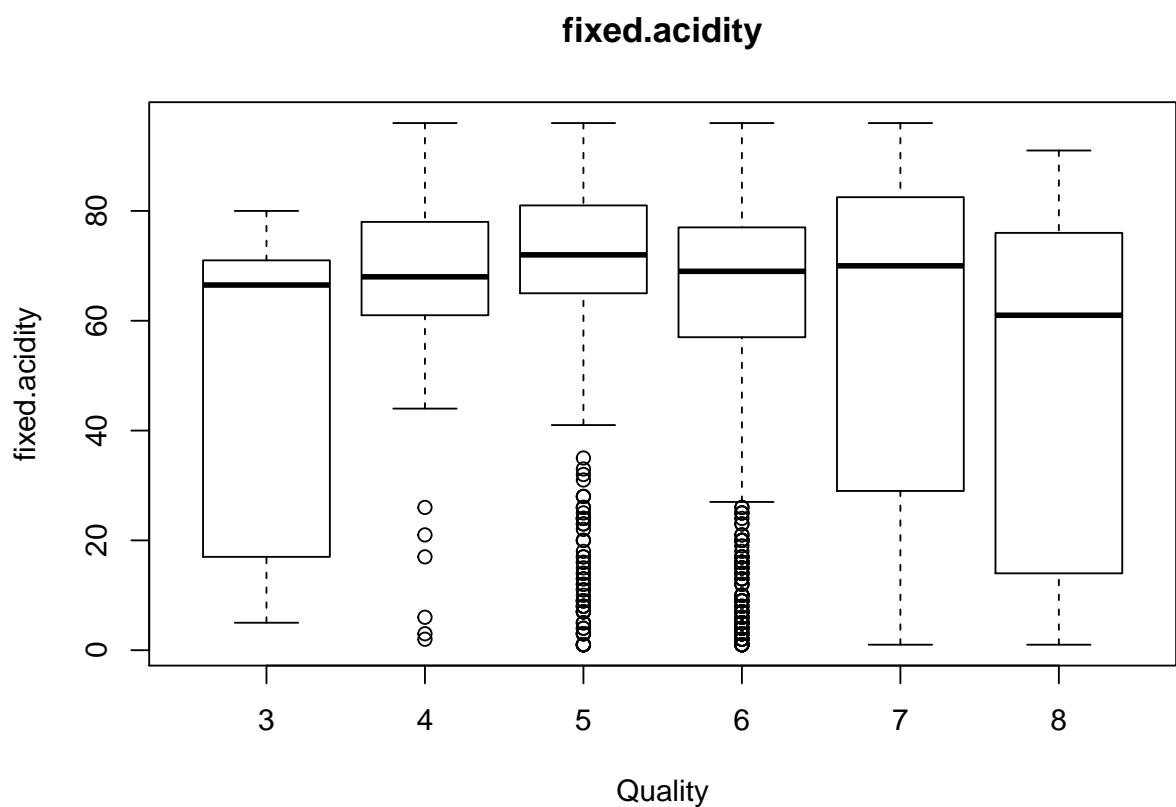
wine.data <- read.csv2("FinalProject/winequality-red.csv")

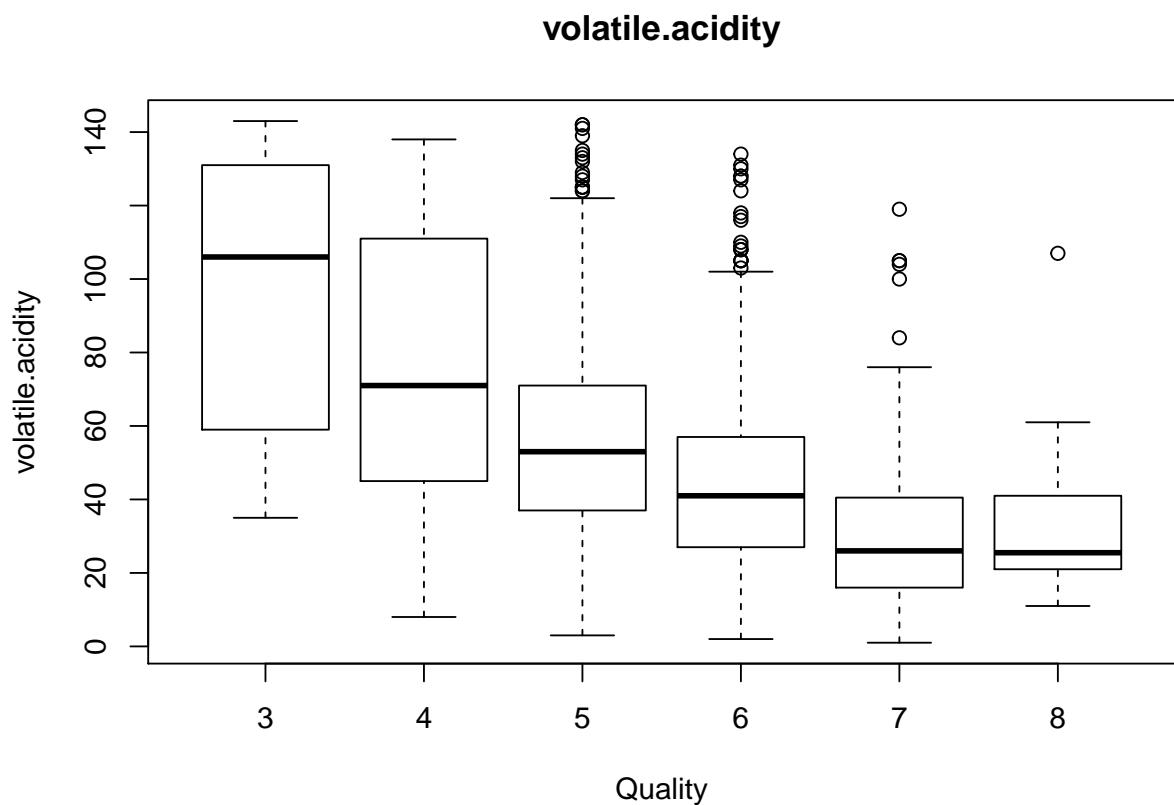
# Specifying the data type via colClasses did not work - so we sapply
# as.numeric to convert the factor data to numeric. We may want to convert
# some of the variables to factors later - but for visualization of the raw
# features we should use numeric type.
wine.data[, c(1:12)] <- sapply(wine.data[, c(1:12)], as.numeric)

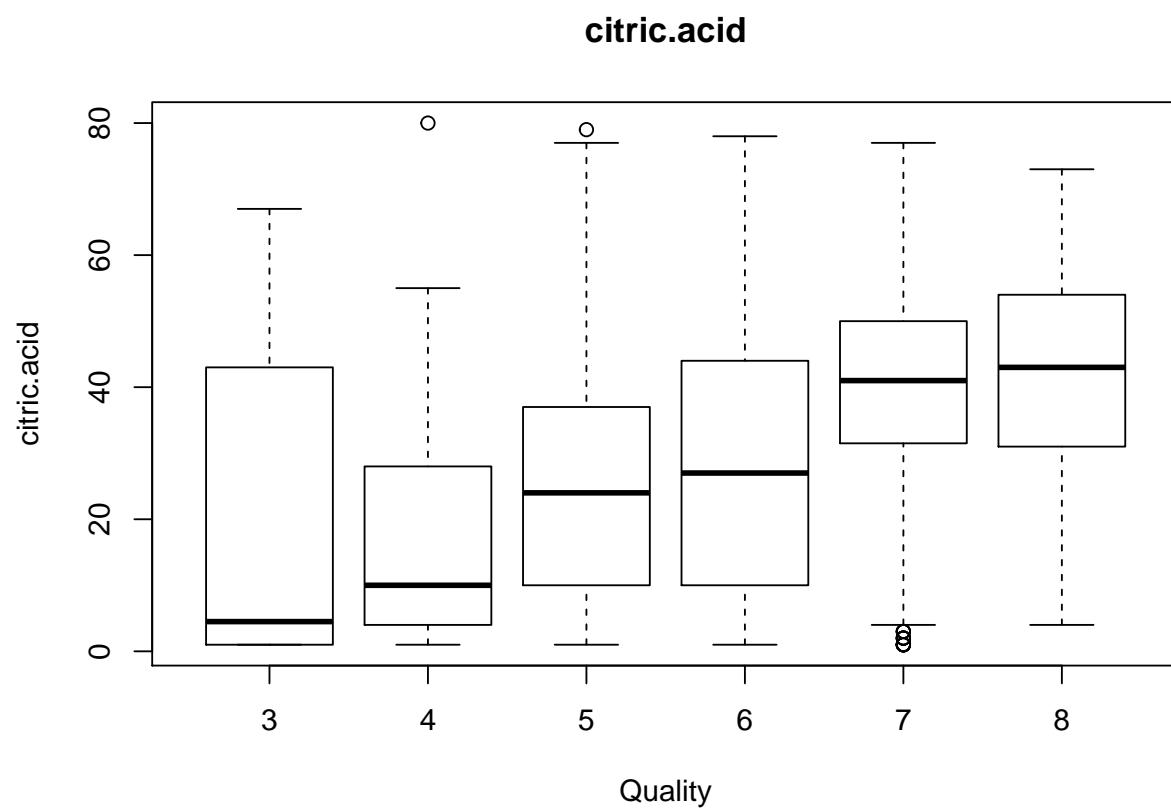
DF <- wine.data
```

Box plots of features by quality

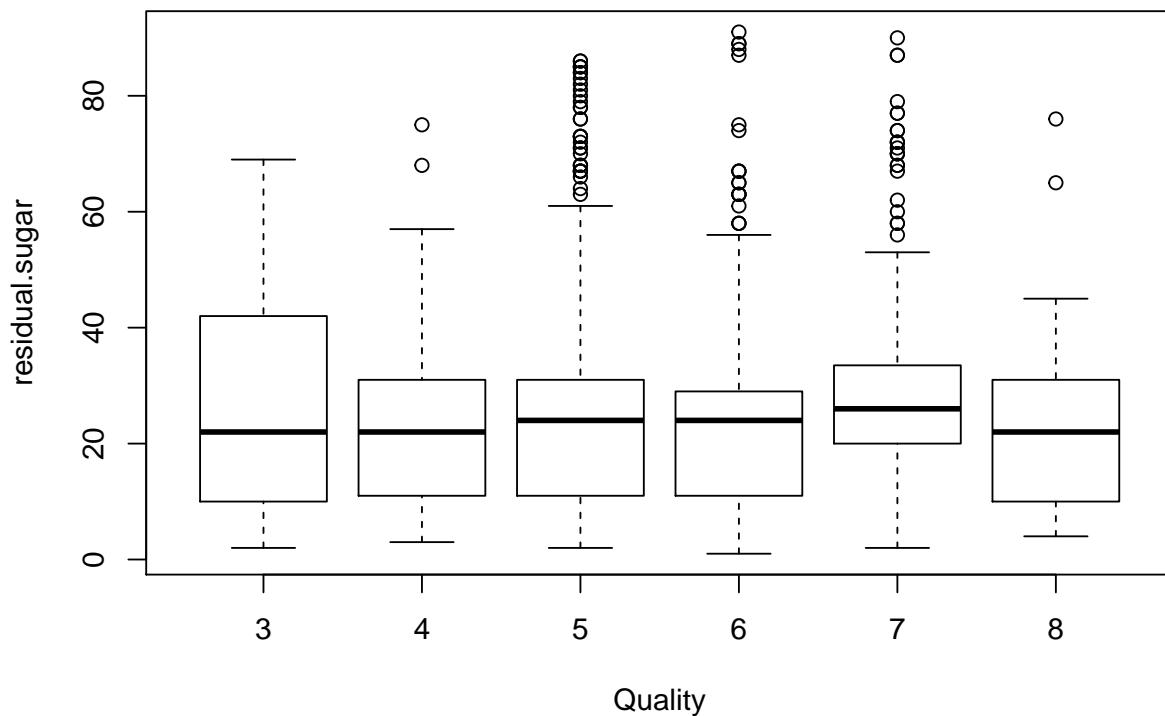
```
for (i in 1:(ncol(wine.data) - 1)) {
  feature = wine.data[, i]
  featureName = names(wine.data)[i]
  boxplot(feature ~ DF$quality, xlab = "Quality", ylab = featureName, main = featureName)
}
```



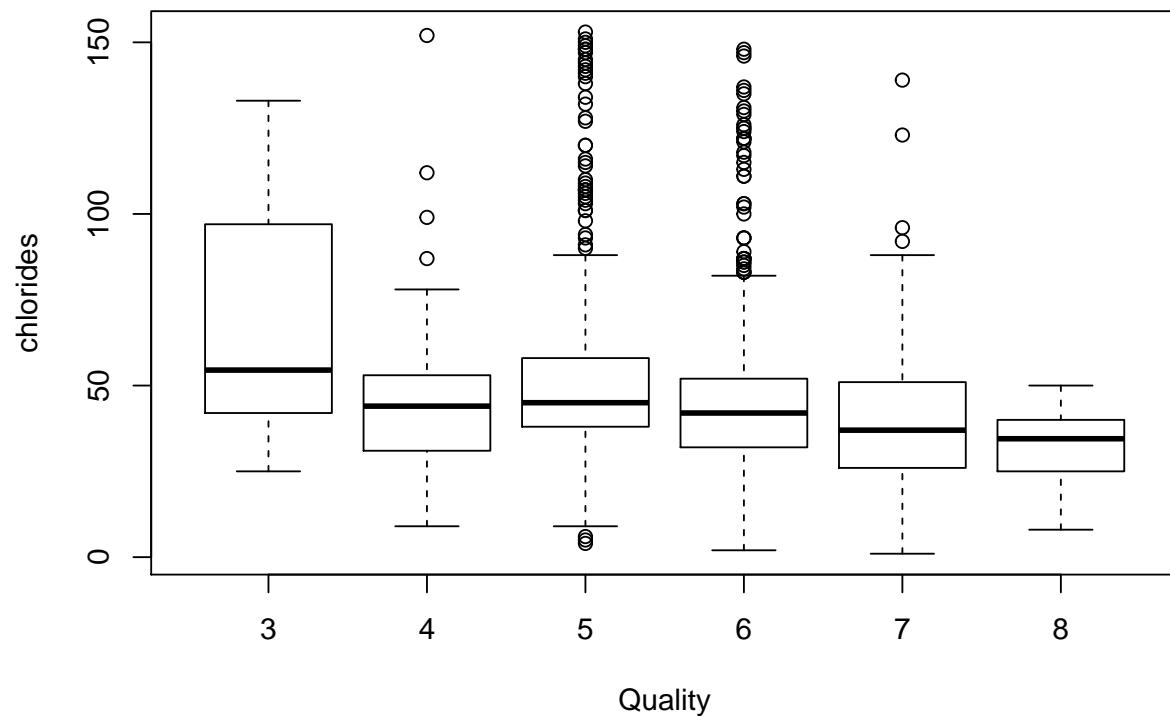




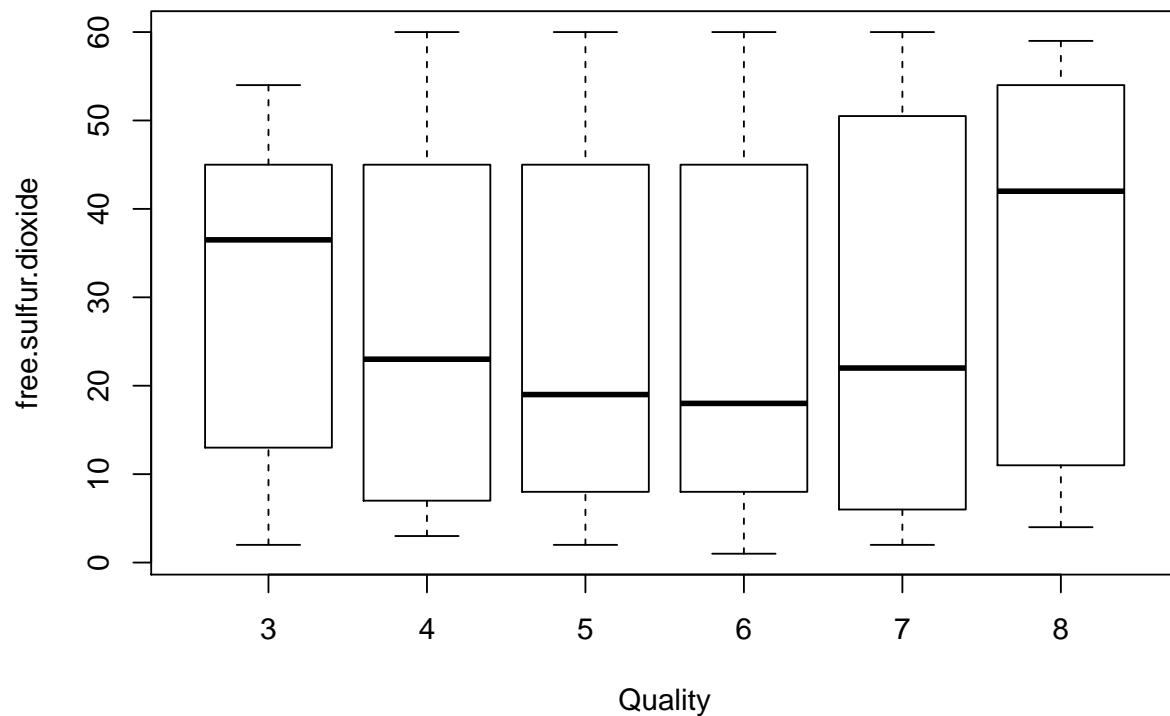
residual.sugar



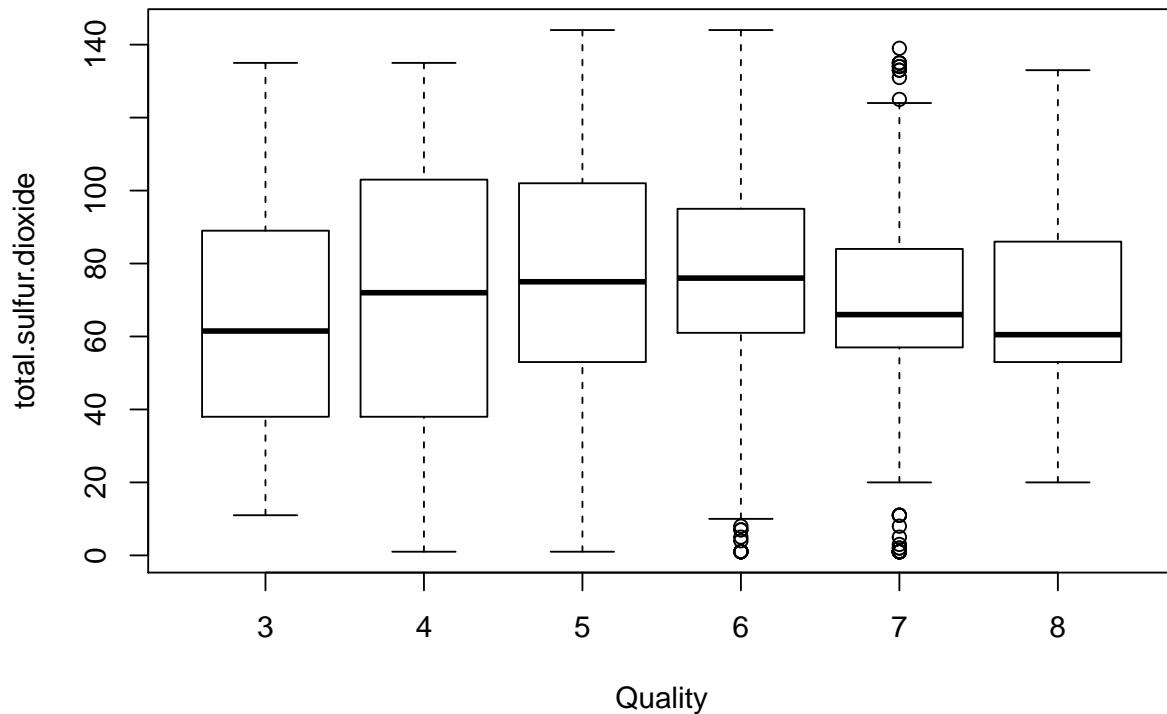
chlorides

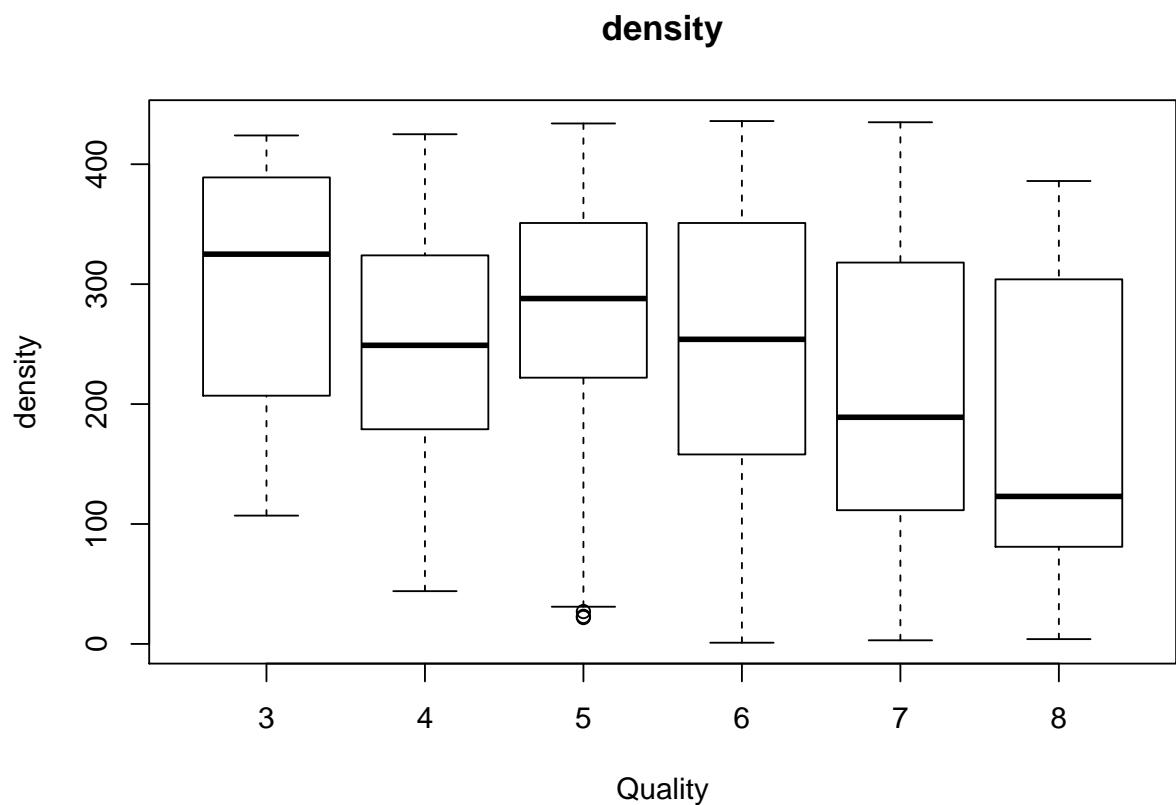


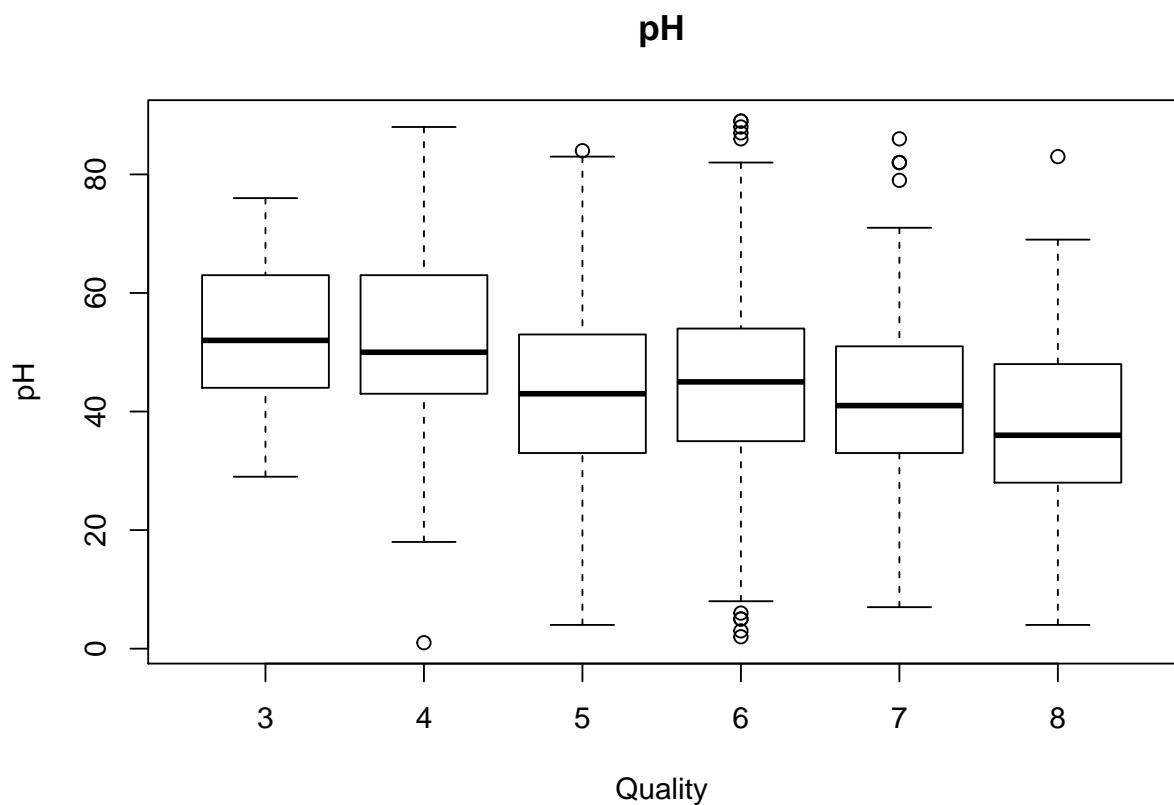
free.sulfur.dioxide



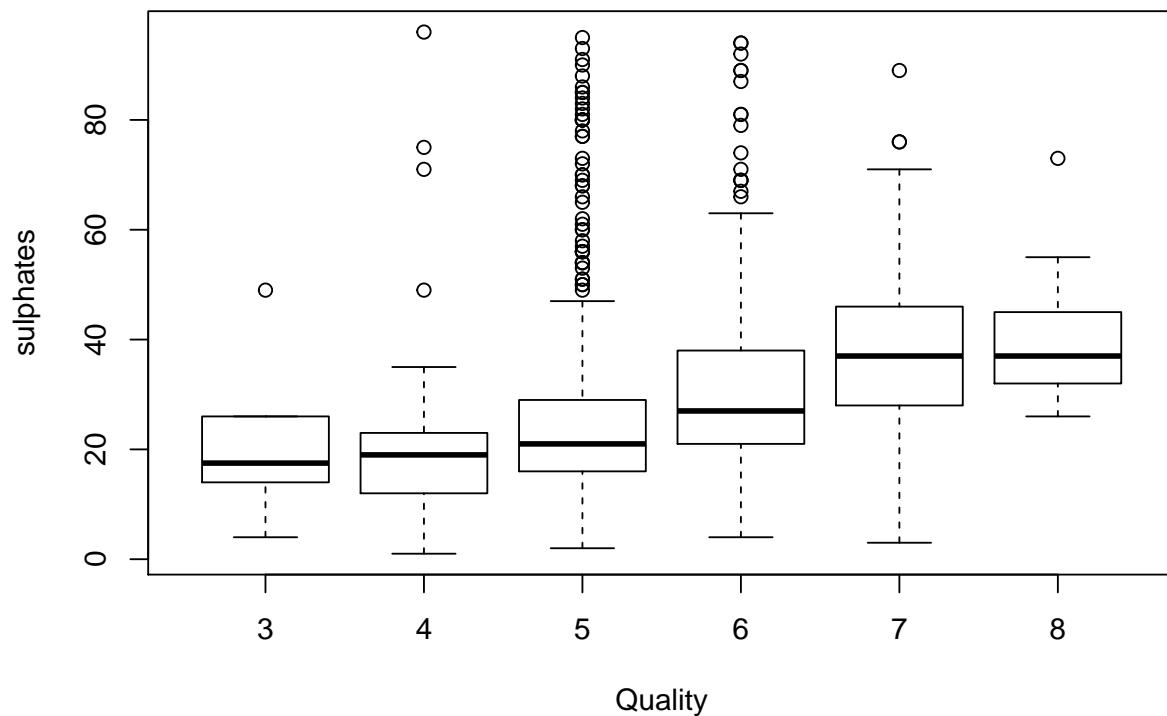
total.sulfur.dioxide

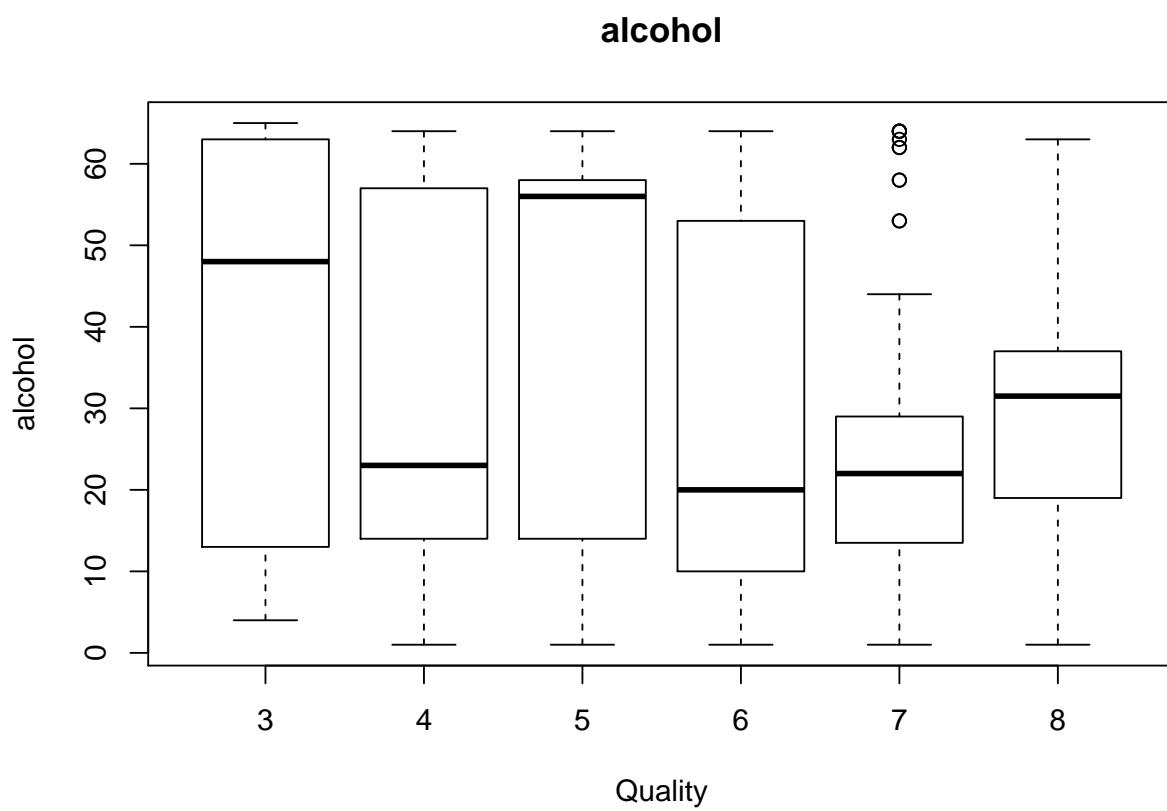






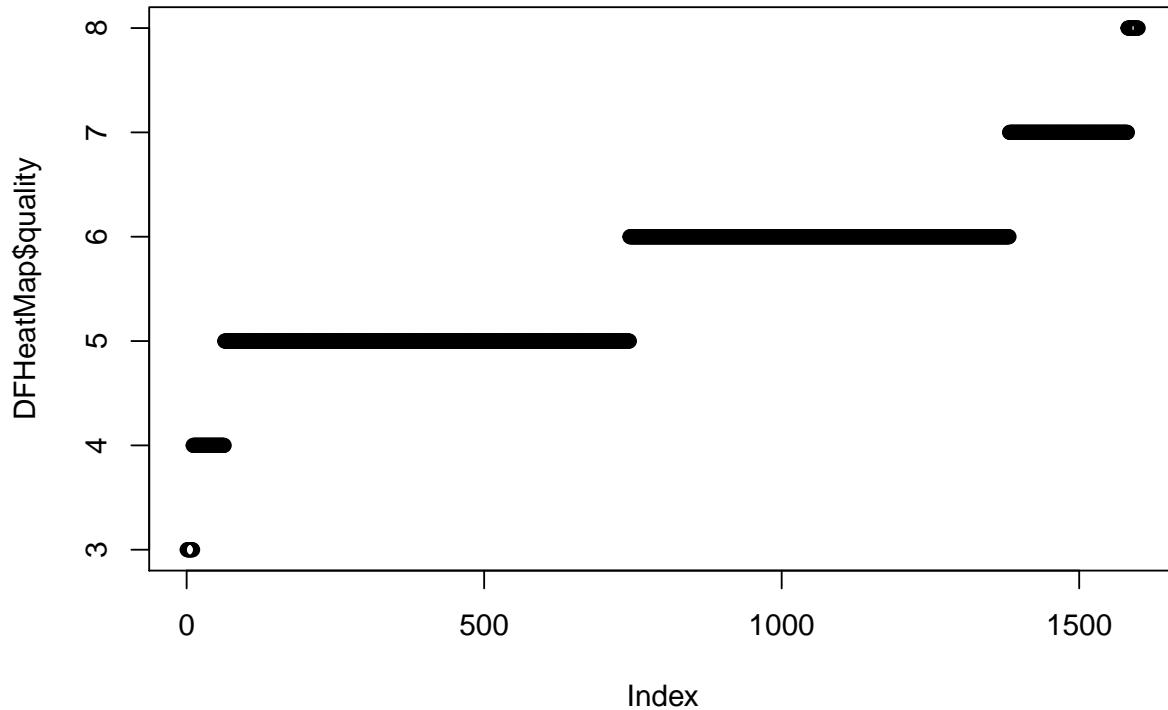
sulphates



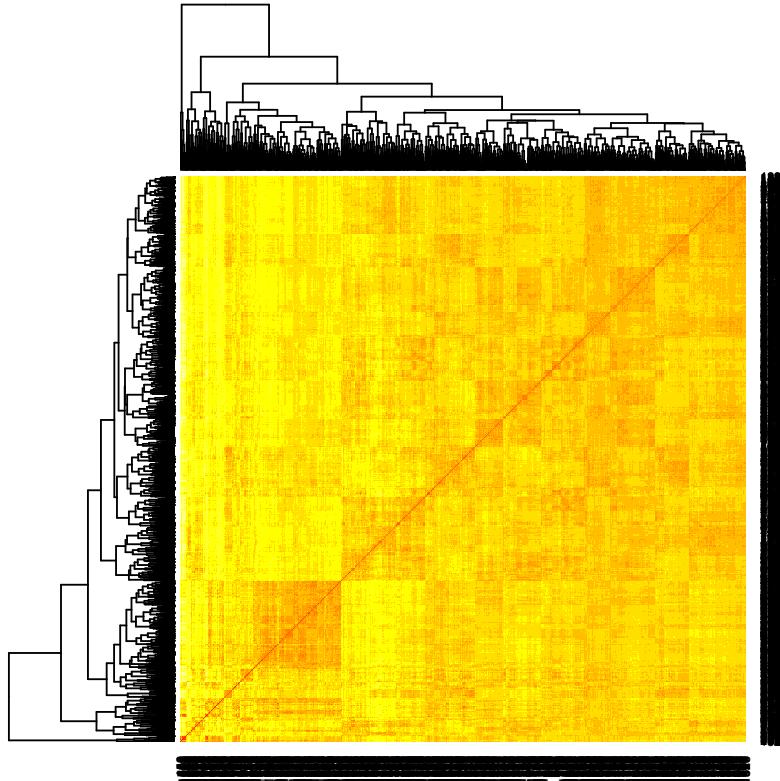


Heat Map - sorting by quality

```
DFHeatMap <- DF[order(DF$quality), ]  
plot(DFHeatMap$quality)
```



```
DFHeatMap$quality <- NULL  
dist.mat <- as.matrix(dist(scale(DFHeatMap, center = TRUE, scale = TRUE), method = "euclidian"))  
heatmap(dist.mat)
```



K-means clustering

```

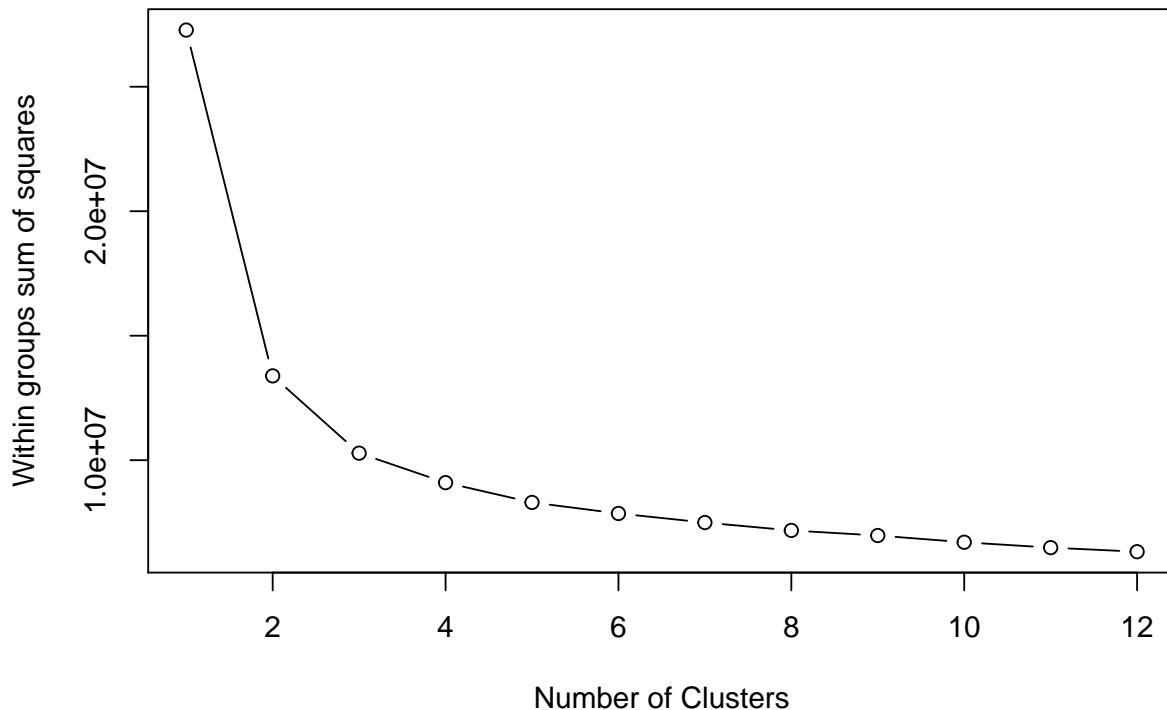
DFOrdered <- DF[order(DF$quality), ]

# Determine number of clusters. Here we calculate the
wss <- (nrow(DFOrdered) - 1) * sum(apply(DFOrdered, 2, var))
for (i in 2:12) {
  wss[i] <- sum(kmeans(DFOrdered, centers = i)$withinss)
}

plot(1:12, wss, type = "b", xlab = "Number of Clusters", ylab = "Within groups sum of squares")
title("Sum of square distance to cluster centroid by cluster number \n We seek a kink in this plot to id

```

Sum of square distance to cluster centrod by cluster number
We seek a kink in this plot to identify the correct number of clusters
k in [3,8] seems like a good choice.

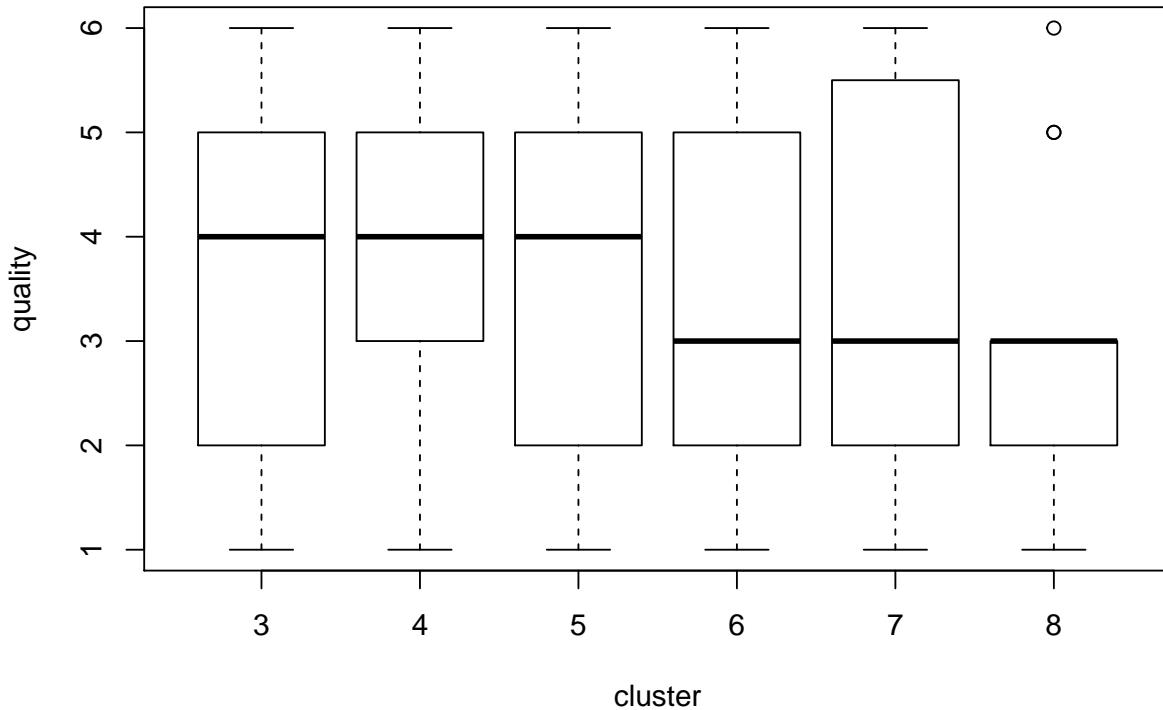


```
# Run k-means with k=6
km.out = kmeans(DF, 6, nstart = 20)

cluster_kmean.label <- km.out$cluster

boxplot(cluster_kmean.label ~ DF$quality, xlab = "cluster", ylab = "quality",
        main = "Quality by Cluster number for k=6 in K-means")
```

Quality by Cluster number for k=6 in K-means



```
TB <- table(cluster_kmean.label, DF$quality)
library(pander)
pander(TB, caption = "Cluster matchup by quality for kmeans k=6. This is NOT a confusion matrix. The c
```

Table 1: Cluster matchup by quality for kmeans k=6. This is NOT a confusion matrix. The cluster numbers are not identified with quality.

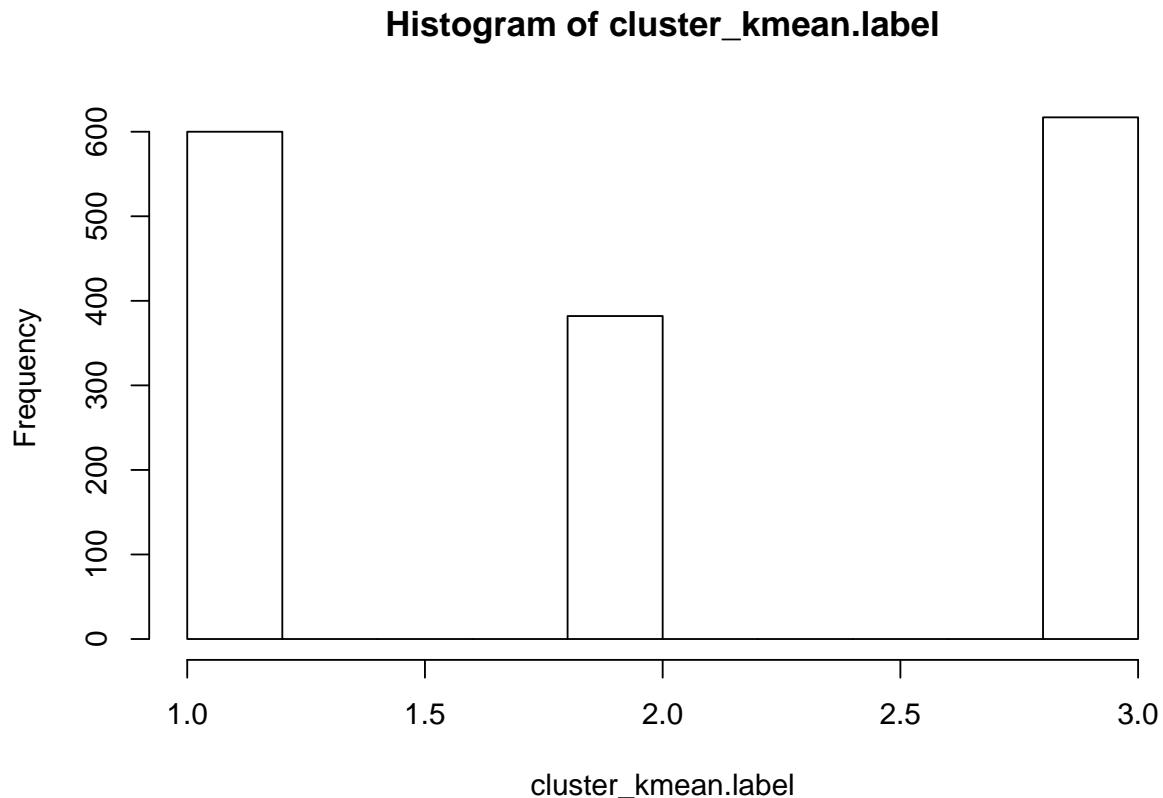
	3	4	5	6	7	8
1	7	156	131	14	1	
3	5	66	110	40	5	
0	6	37	90	50	9	
2	10	201	97	17	0	
2	15	129	64	28	2	
2	10	92	146	50	1	

```
# Run k-means with k=3
km.out = kmeans(DF, 3, nstart = 20)

cluster_kmean.label <- km.out$cluster

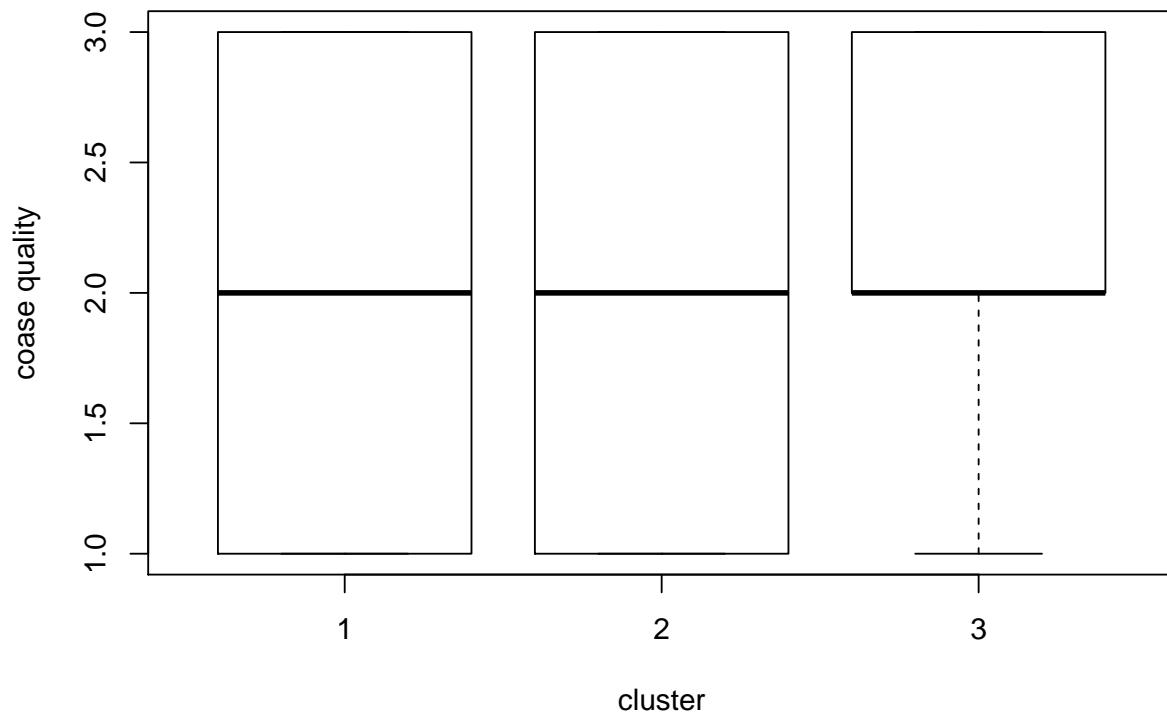
quality_cat <- (ifelse(wine.data$quality < 5, 1, ifelse(wine.data$quality >
6, 3, 2)))
```

```
hist(cluster_kmean.label)
```

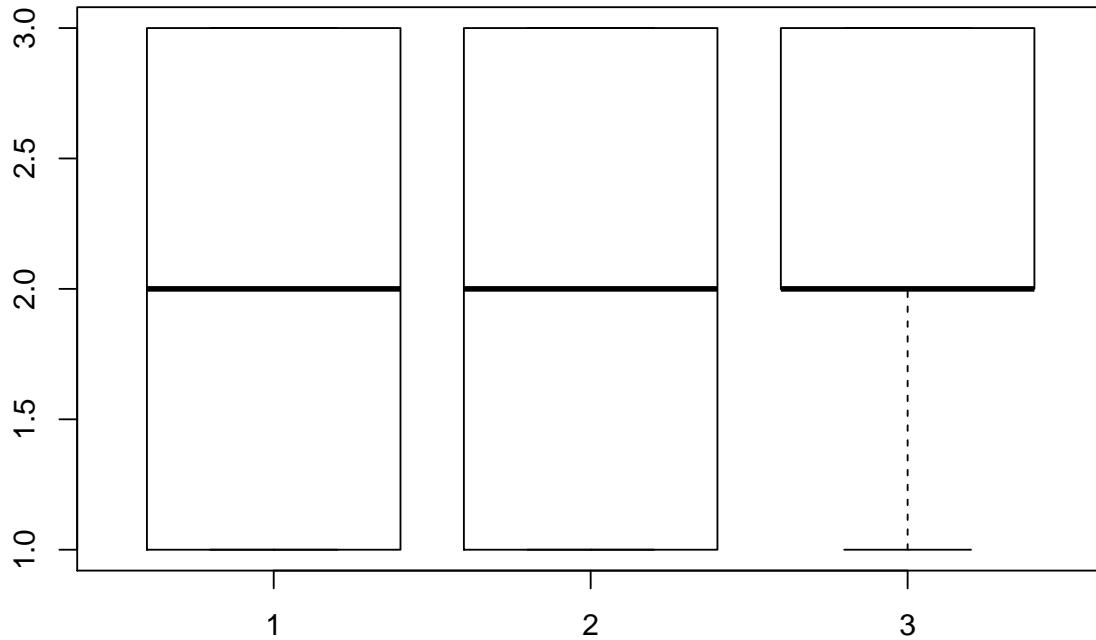


```
boxplot(cluster_kmean.label ~ quality_cat, xlab = "cluster", ylab = "coase quality",
        main = "Quality by Cluster number for k=3 in K-means")
```

Quality by Cluster number for k=3 in K-means



```
boxplot(cluster_kmean.label ~ quality_cat)
```



```
TB <- table(cluster_kmean.label, DF$quality)
library(pander)
pander(TB, caption = "Cluster matchup by quality for kmeans k=3. This is NOT a confusion matrix. The c
```

Table 2: Cluster matchup by quality for kmeans k=3. This is NOT a confusion matrix. The cluster numbers are not identified with quality.

3	4	5	6	7	8
3	24	298	228	45	2
2	11	84	181	94	10
5	18	299	229	60	6

```
# Run k-means with k=4
km.out = kmeans(DF, 4, nstart = 20)

cluster_kmean.label <- km.out$cluster

TB <- table(cluster_kmean.label, DF$quality)
library(pander)
pander(TB, caption = "Cluster matchup by quality for kmeans k=4. This is NOT a confusion matrix. The c
```

Table 3: Cluster matchup by quality for kmeans k=4. This is NOT a confusion matrix. The cluster numbers are not identified with quality.

3	4	5	6	7	8
5	7	187	170	44	3
2	20	252	160	37	3
2	10	55	139	74	10
1	16	187	169	44	2

We split into training and test sets for now. Notice that there are very few points in the extremes of quality. There are arguments for and against splitting the extremes. The features for those sample points could help or hurt a classifier in the middle ranges of quality. It's standard to split the data into training and test sets, but we'd like to reserve the right to revisit how we handle the classes with low counts at the end of the analysis.

```
# train <- sample(nrow(DF), floor(nrow(DF)* 2/3))

# write.csv(train,file = 'train.sample.csv',row.names = FALSE ,quote =
# FALSE)

# Check that it works
train <- as.integer(unlist(read.csv("train.sample.csv", header = TRUE)))

DFTrain <- DF[train, ]

DFTest <- DF[-train, ]

TDTrain <- table(DFTrain$quality)

TDTTest <- table(DFTest$quality)

ratio.class <- TDTrain/TDTTest

library(pander)
pander(TDTrain, caption = "Distribution of quality for training sample")
```

Table 4: Distribution of quality for training sample

3	4	5	6	7	8
8	35	453	415	142	13

```
pander(TDTTest, caption = "Distribution of quality for test sample")
```

Table 5: Distribution of quality for test sample

3	4	5	6	7	8
2	18	228	223	57	5

LDA

```
library(MASS)
lda.fit = lda(quality ~ ., data = DFTrain)
lda.fit

## Call:
## lda(quality ~ ., data = DFTrain)
##
## Prior probabilities of groups:
##      3       4       5       6       7       8 
## 0.00750469 0.03283302 0.42495310 0.38930582 0.13320826 0.01219512
##
## Group means:
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 3     48.00000          96.62500     21.50000    30.00000 79.62500
## 4     59.57143         79.25714     16.45714    25.34286 41.28571
## 5     68.14349         55.30905     25.81898    24.79249 49.98896
## 6     59.32289         42.92289     29.17349    23.46988 45.08434
## 7     57.04225         30.69014     38.30282    29.35211 39.23239
## 8     45.84615         35.23077     42.07692    27.69231 33.00000
##   free.sulfur.dioxide total.sulfur.dioxide density      pH sulphates
## 3       31.25000          65.25000 302.3750 52.00000 19.62500
## 4       26.11429          72.88571 240.5714 52.57143 19.00000
## 5       24.68653          73.55188 276.2009 43.81015 24.44592
## 6       24.82892          77.34217 243.9494 44.47952 30.72289
## 7       25.52113          64.88732 213.9296 41.69014 36.61972
## 8       34.61538          64.76923 179.9231 40.38462 38.53846
##   alcohol
## 3 33.00000
## 4 34.94286
## 5 40.54967
## 6 27.27229
## 7 22.29577
## 8 26.53846
##
## Coefficients of linear discriminants:
##           LD1        LD2        LD3        LD4
## fixed.acidity -0.0065111404  0.024436828 -0.009532841 -0.0003469340
## volatile.acidity -0.0185341165 -0.029053556 -0.014879177 -0.0074534741
## citric.acid 0.0038691850 -0.018829195 -0.034792127  0.0112188995
## residual.sugar 0.0163789540 -0.005783305 -0.032080721  0.0085494090
## chlorides -0.0116658109 -0.004324411  0.011580919  0.0417800645
## free.sulfur.dioxide -0.0010528827 -0.006144894  0.001009564 -0.0067156178
## total.sulfur.dioxide 0.0007387804  0.002609760  0.014328658 -0.0087314508
## density -0.0048510299  0.003823320  0.004054947 -0.0001108575
## pH -0.0065916941 -0.029058850  0.007104401  0.0132065980
## sulphates 0.0377371174 -0.002838857 -0.004609353 -0.0058031310
## alcohol -0.0156181508  0.009124249 -0.025642449 -0.0091778291
##           LD5
## fixed.acidity -0.010717349
## volatile.acidity 0.004507974
## citric.acid 0.012455231
```

```

## residual.sugar      -0.019135658
## chlorides          0.004926826
## free.sulfur.dioxide 0.030014077
## total.sulfur.dioxide 0.011513051
## density            -0.002456265
## pH                 -0.014776405
## sulphates          0.012575906
## alcohol             0.020559970
##
## Proportion of trace:
##    LD1     LD2     LD3     LD4     LD5
## 0.7608 0.1151 0.0848 0.0295 0.0098

```

```

lda.pred = predict(lda.fit, DFTest)
lda.class = lda.pred$class
TD <- table(lda.class, DFTest$quality)
pander(TD)

```

	3	4	5	6	7	8
3	0	1	4	0	1	0
4	0	0	2	4	0	0
5	2	9	163	90	6	0
6	0	7	54	119	31	4
7	0	1	5	10	19	1
8	0	0	0	0	0	0

```
ACC_lda = (sum(diag(as.array(TD))))/length(DFTest$quality)
```

The accuracy 0.564728 is not great - but we see that there is some good overall agreement in the 5-6 and 6-7 range. We also note that NO low quality wines were classified as high quality and that no high quality wines were classified as low quality. We might consider three level factor grouping quality {3,4}, {5,6}, {7,8}

```

DFCoarse <- DF
DFCoarse$quality_cat <- ifelse(wine.data$quality < 5, "LOW", ifelse(wine.data$quality >
6, "HIGH", "MEDIUM"))
DFCoarse$quality <- NULL
DFCoarseTrain <- DFCoarse[train, ]
DFCoarseTest <- DFCoarse[-train, ]

lda.fit = lda(quality_cat ~ ., data = DFCoarseTrain)
lda.fit

```

```

## Call:
## lda(quality_cat ~ ., data = DFCoarseTrain)
##
## Prior probabilities of groups:
##      HIGH      LOW      MEDIUM
## 0.14540338 0.04033771 0.81425891
##
## Group means:
##      fixed.acidity volatile.acidity citric.acid residual.sugar chlorides

```

```

## HIGH      56.10323      31.07097      38.61935      29.21290  38.70968
## LOW       57.41860      82.48837      17.39535      26.20930  48.41860
## MEDIUM    63.92627      49.38710      27.42281      24.16014  47.64401
##          free.sulfur.dioxide total.sulfur.dioxide density      pH
## HIGH      26.28387      64.87742 211.0774 41.58065
## LOW       27.06977      71.46512 252.0698 52.46512
## MEDIUM    24.75461      75.36406 260.7811 44.13018
##          sulphates      alcohol
## HIGH     36.78065 22.65161
## LOW      19.11628 34.58140
## MEDIUM   27.44700 34.20161
##
## Coefficients of linear discriminants:
##                               LD1          LD2
## fixed.acidity      -8.161248e-05 0.018938074
## volatile.acidity   1.614776e-02 -0.031574594
## citric.acid        -1.199469e-02 -0.026433335
## residual.sugar     -2.393399e-02 -0.017199071
## chlorides           1.194487e-02 0.003651235
## free.sulfur.dioxide 2.524600e-03 -0.002775764
## total.sulfur.dioxide 3.816477e-03 0.007441980
## density             5.038371e-03 0.004701649
## pH                  1.073514e-02 -0.025406808
## sulphates           -3.420472e-02 -0.004130855
## alcohol              5.812238e-03 0.002369115
##
## Proportion of trace:
##      LD1      LD2
## 0.8181 0.1819

```

```

lda.pred = predict(lda.fit, DFCoarseTest)
lda.class = lda.pred$class
TD <- table(lda.class, DFCoarseTest$quality)
pander(TD)

```

	HIGH	LOW	MEDIUM
HIGH	16	1	10
LOW	0	1	8
MEDIUM	46	18	433

```
ACC_lda_coarse = (sum(diag(as.array(TD))))/length(DFTest$quality)
```

We see the accuracy 0.8442777 has improved by coalescing the quality levels.

Fit a two class SVM classification model

```
# We do a logistic classification for comparison
```

```
DFCoarse <- DF
```

```

DFCoarse$quality_cat <- ifelse(wine.data$quality <= 5, 0, 1)
DFCoarse$quality <- NULL
DFCoarseTrain <- DFCoarse[train, ]
DFCoarseTest <- DFCoarse[-train, ]

glm.fit <- glm(quality_cat ~ ., data = DFCoarseTrain, family = binomial)
summary(glm.fit)

##
## Call:
## glm(formula = quality_cat ~ ., family = binomial, data = DFCoarseTrain)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.5317  -0.8364   0.4104   0.8525   2.4000
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)              2.7113203  0.5486222  4.942 7.73e-07 ***
## fixed.acidity          -0.0113550  0.0030596 -3.711 0.000206 ***
## volatile.acidity        -0.0216674  0.0034508 -6.279 3.41e-10 ***
## citric.acid            -0.0045903  0.0055387 -0.829 0.407231
## residual.sugar          0.0097113  0.0049915  1.946 0.051704 .
## chlorides               -0.0108621  0.0036092 -3.010 0.002616 **
## free.sulfur.dioxide    0.0003851  0.0037779  0.102 0.918807
## total.sulfur.dioxide   0.0049662  0.0023648  2.100 0.035724 *
## density                 -0.0042597  0.0008244 -5.167 2.38e-07 ***
## pH                      -0.0036695  0.0059213 -0.620 0.535446
## sulphates               0.0424062  0.0056815  7.464 8.40e-14 ***
## alcohol                 -0.0223319  0.0033278 -6.711 1.94e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1472.6  on 1065  degrees of freedom
## Residual deviance: 1154.6  on 1054  degrees of freedom
## AIC: 1178.6
##
## Number of Fisher Scoring iterations: 4

glm.probs = predict(glm.fit, DFCoarseTest, type = "response")

glm.pred = rep(0, nrow(DFCoarseTest))
glm.pred[glm.probs > 0.5] = 1
TB <- table(glm.pred, DFCoarseTest$quality_cat)
pander(TB, caption = "Logistic Regression")

```

Table 8: Logistic Regression

	0	1
0	176	94

	0	1
1	72	191

```

ACC_glm_binomial = (TB[1] + TB[4])/length(DFCoarseTest$quality_cat)
Specificity = TB[1]/sum(DFCoarseTest$quality_cat == 0)
Sensitivity = TB[4]/sum(DFCoarseTest$quality_cat == 1)

##### We require a categorical response for the e1071 package so the data frame
##### for the 2 class SVM is regenerated here

DFCoarse <- DF
DFCoarse$quality_cat <- as.factor(ifelse(wine.data$quality <= 5, "LOW", "HIGH"))
DFCoarse$quality <- NULL
DFCoarseTrain <- DFCoarse[train, ]
DFCoarseTest <- DFCoarse[-train, ]

library(e1071)

tune.svm = tune(svm, quality_cat ~ ., data = DFCoarseTrain, kernel = "linear",
                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))

svm.fit <- tune.svm$best.model

svm.pred <- predict(svm.fit, DFCoarseTest)
TB <- table(svm.pred, DFCoarseTest$quality_cat)
pander(TB)

```

	HIGH	LOW
HIGH	215	92
LOW	70	156

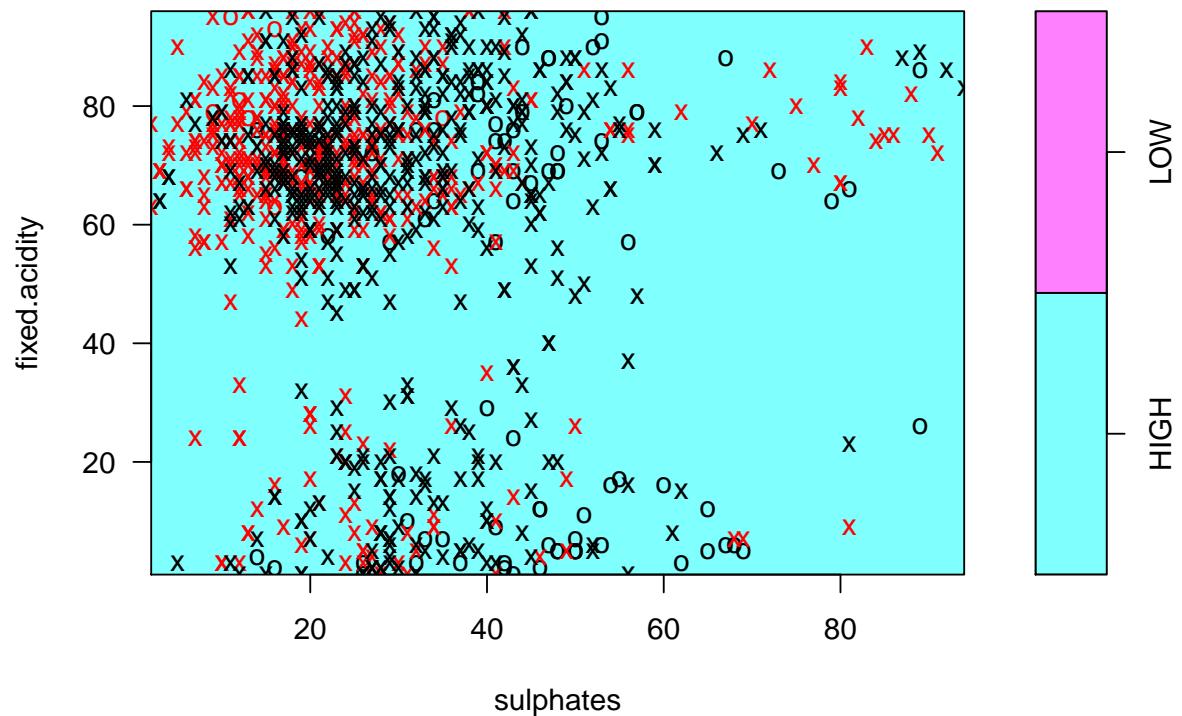
```

ACC_Linear_SVM = (TB[1] + TB[4])/length(DFCoarseTest$quality_cat)

plot(svm.fit, DFCoarseTrain, fixed.acidity ~ sulphates)

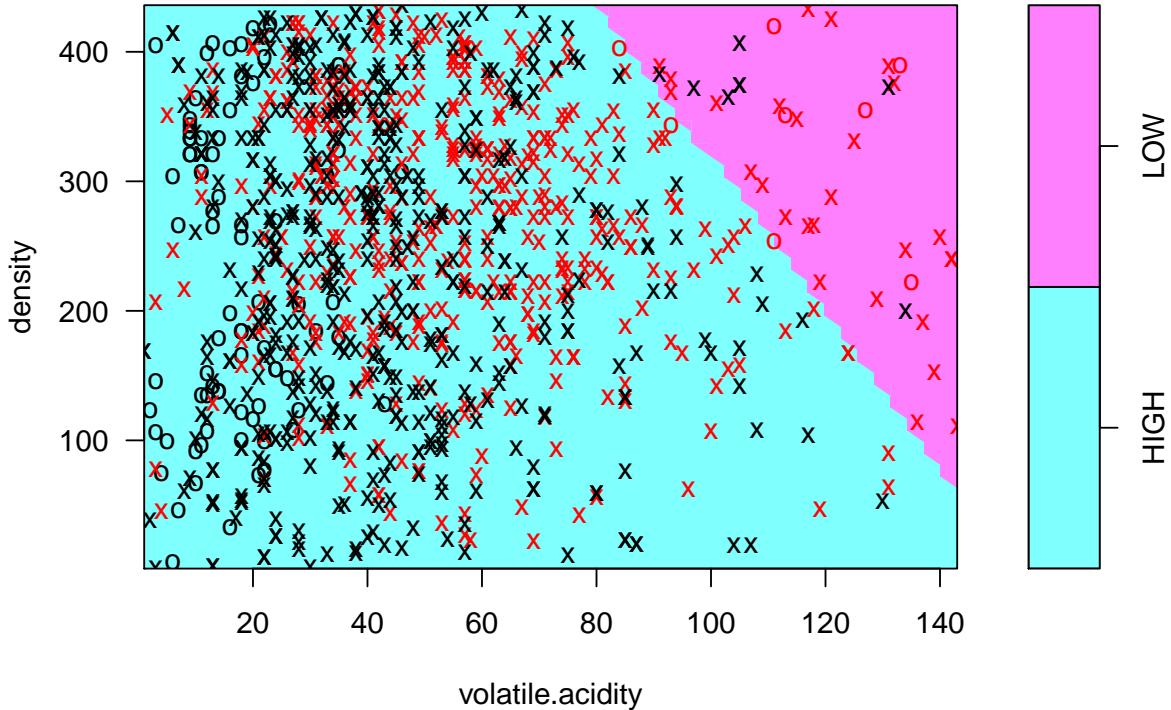
```

SVM classification plot



```
plot(svm.fit, DFCoarseTrain, density ~ volatile.acidity)
```

SVM classification plot



We see that we need to use a non linear kernel. We train a non linear SVM below using the built in cross validation method tune to optimize the cost and gamma hyper-parameters

```
gamma_default <- 1/(ncol(DFCoarseTrain) - 1)

gamma_list <- ((1:10)/5) * gamma_default

tune.svm = tune(svm, quality_cat ~ ., data = DFCoarseTrain, kernel = "radial",
                 ranges = list(gamma = gamma_list, cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))

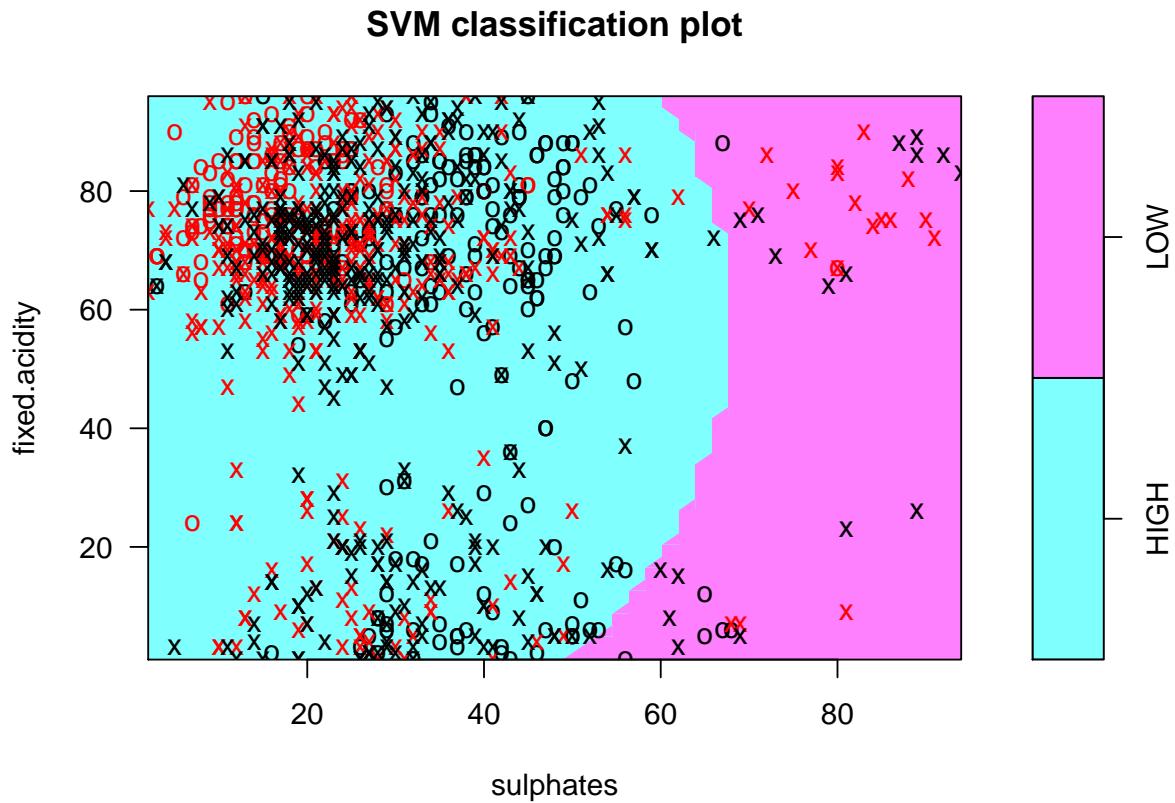
svm.fit <- tune.svm$best.model

svm.pred <- predict(svm.fit, DFCoarseTest)
TB <- table(svm.pred, DFCoarseTest$quality_cat)
pander(TB)
```

	HIGH	LOW
HIGH	208	57
LOW	77	191

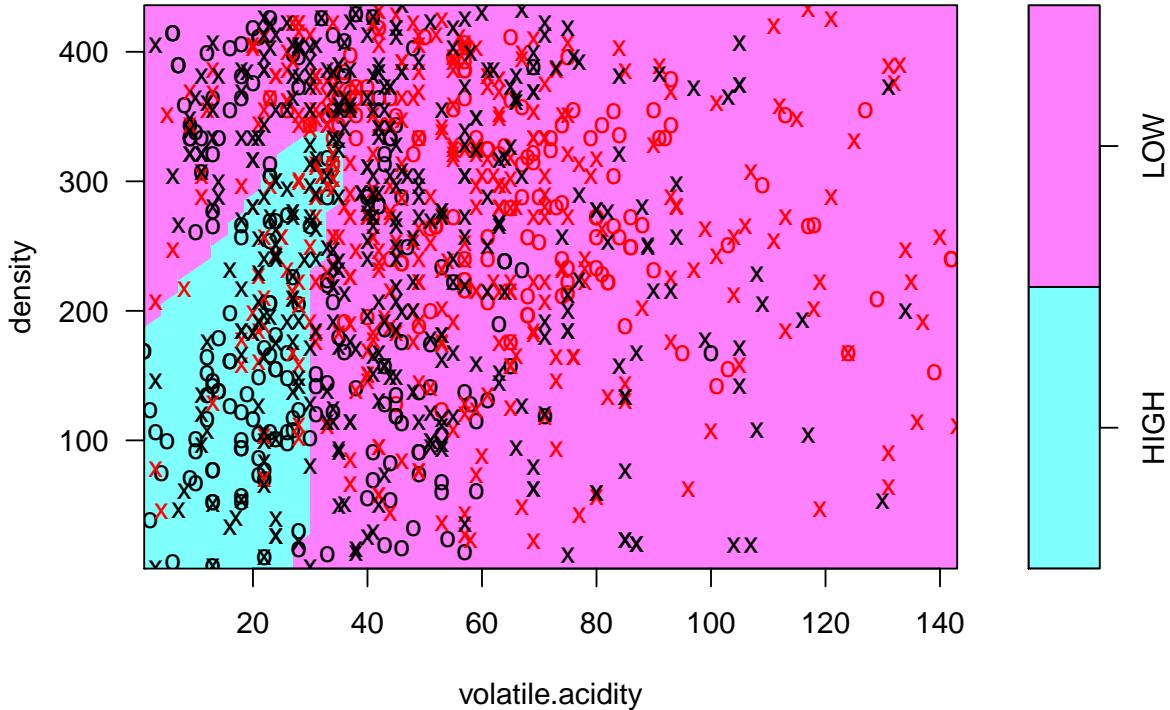
```
ACC_Radial_SVM = (TB[1] + TB[4])/length(DFCoarseTest$quality_cat)
```

```
# Beware - these plots may have a bug for the non linear sum!
plot(svm.fit, DFCoarseTrain, fixed.acidity ~ sulphates)
```



```
plot(svm.fit, DFCoarseTrain, density ~ volatile.acidity)
```

SVM classification plot



For the SVM trained with a kernel using the radial basis similarity function the accuracy has gone from 0.69606 to 0.7485929.

```
gamma_default <- 1/(ncol(DFCoarseTrain) - 1)

gamma_list <- ((1:10)/5) * gamma_default

tune.svm = tune(svm, quality_cat ~ ., data = DFCoarseTrain, kernel = "polynomial",
  ranges = list(gamma = gamma_list, cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
    degree = c(2, 3, 4)))

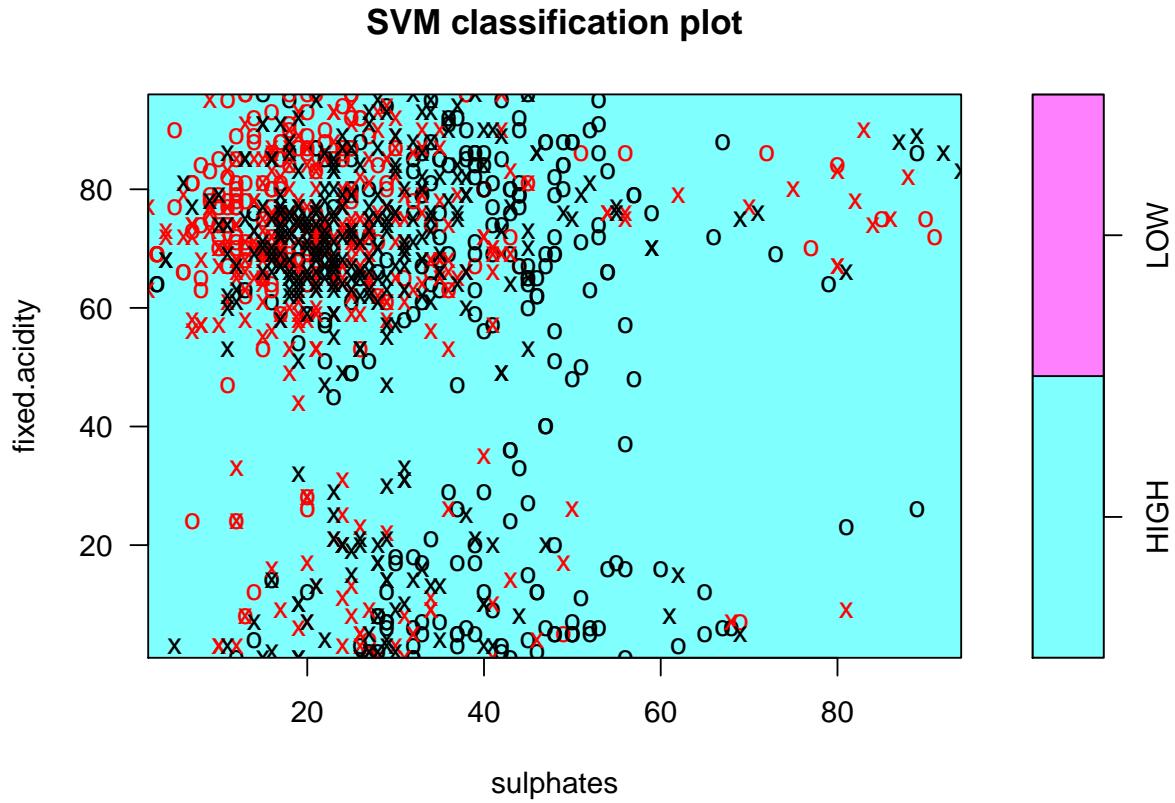
svm.fit <- tune.svm$best.model

svm.pred <- predict(svm.fit, DFCoarseTest)
TB <- table(svm.pred, DFCoarseTest$quality_cat)
pander(TB)
```

	HIGH	LOW
HIGH	203	79
LOW	82	169

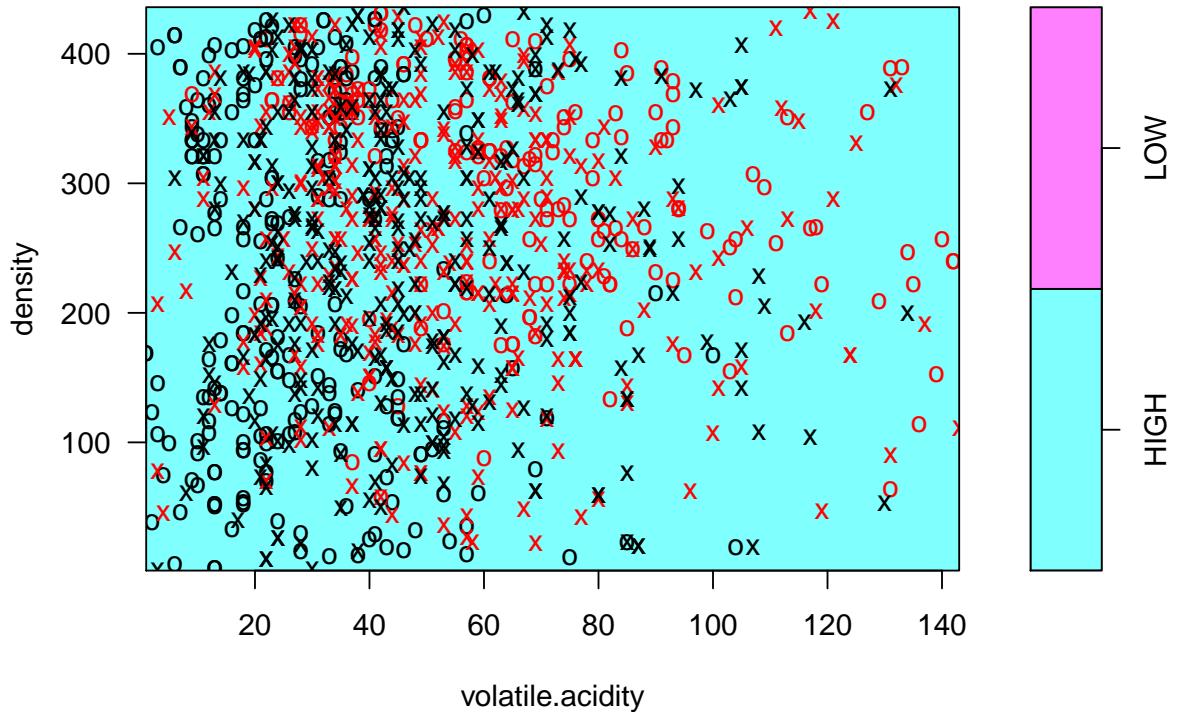
```
ACC_Polynomial_SVM = (TB[1] + TB[4])/length(DFCoarseTest$quality_cat)
# Beware - these plots may have a bug for the non linear sum!
```

```
plot(svm.fit, DFCoarseTrain, fixed.acidity ~ sulphates)
```



```
plot(svm.fit, DFCoarseTrain, density ~ volatile.acidity)
```

SVM classification plot



The polynomial svm tuned to use the best degree,gamma, and cost yielded an accuracy of 0.6979362
Now we try the nu-svm

```
gamma_default <- 1/(ncol(DFCoarseTrain) - 1)

gamma_list <- ((1:10)/5) * gamma_default

tune.svm = tune(svm, quality_cat ~ ., data = DFCoarseTrain, kernel = "radial",
                 type = "nu-classification", ranges = list(gamma = gamma_list, cost = c(0.001,
                 0.01, 0.1, 1, 5, 10, 100)))

svm.fit <- tune.svm$best.model

svm.pred <- predict(svm.fit, DFCoarseTest)
TB <- table(svm.pred, DFCoarseTest$quality_cat)
pander(TB)
```

	HIGH	LOW
HIGH	209	57
LOW	76	191

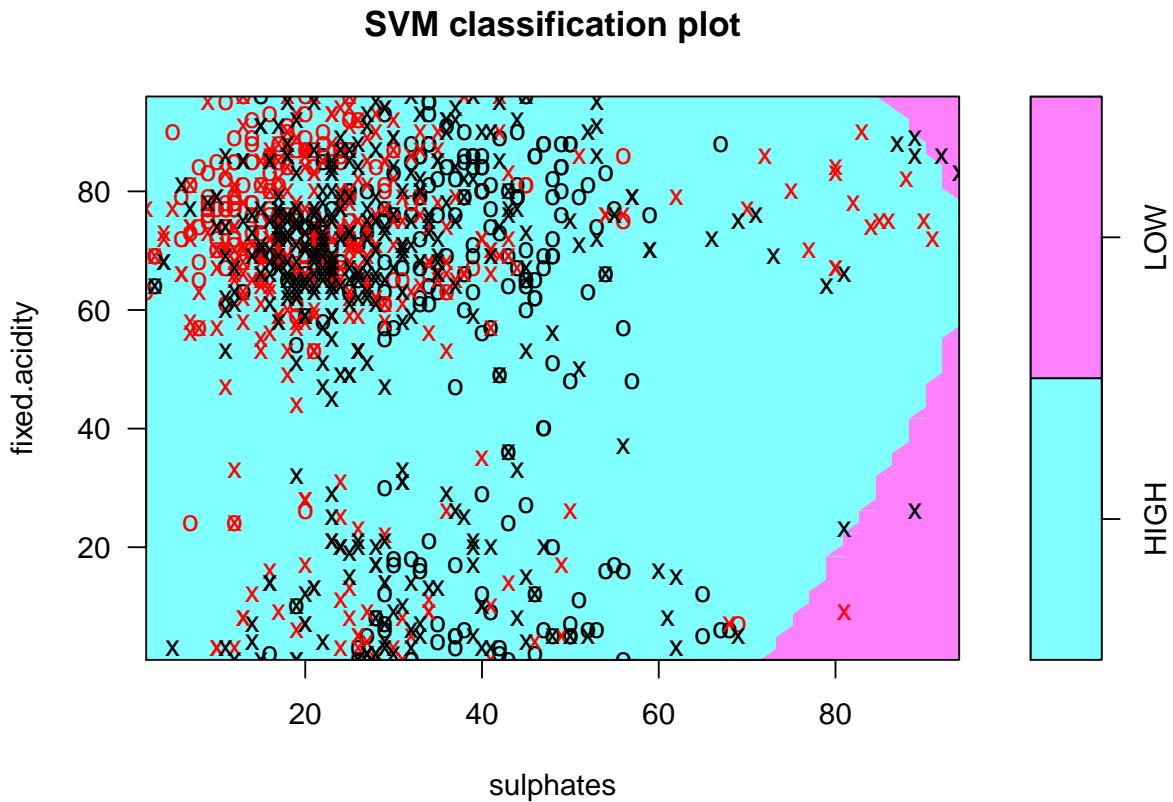
```

ACC_Radial_nuSVM = (TB[1] + TB[4])/length(DFCoarseTest$quality_cat)

# Beware - these plots may have a bug for the non linear sum!

plot(svm.fit, DFCoarseTrain, fixed.acidity ~ sulphates)

```

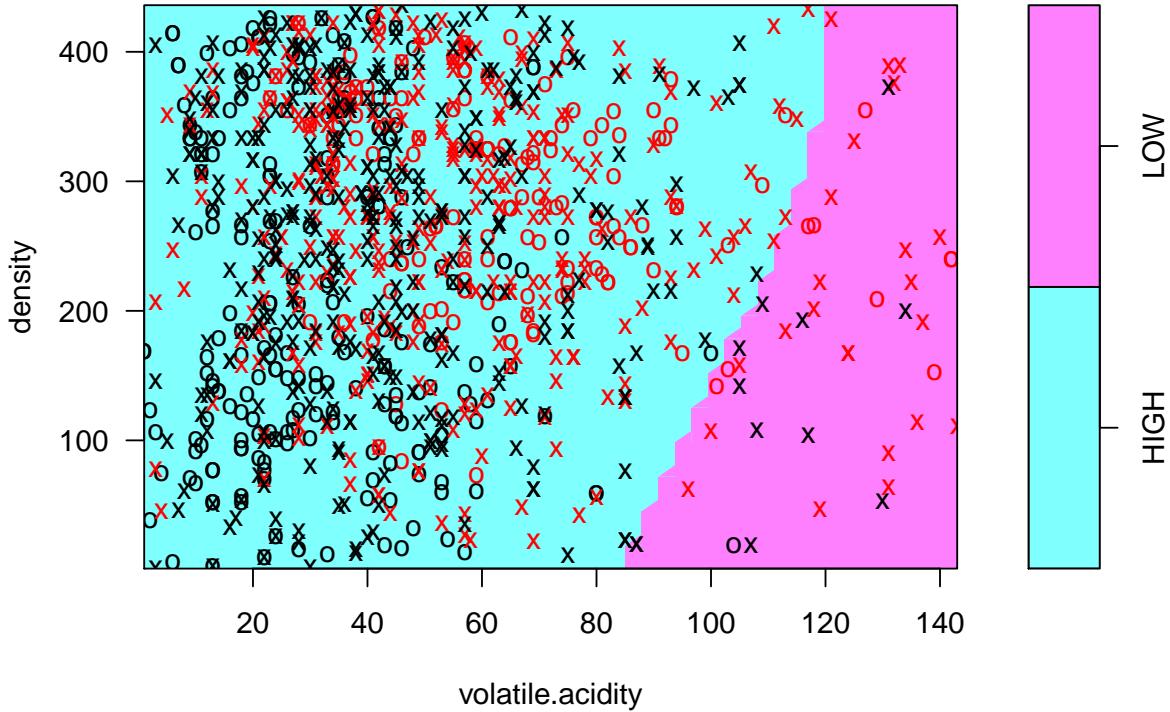


```

plot(svm.fit, DFCoarseTrain, density ~ volatile.acidity)

```

SVM classification plot



One Versus All for three level classification. We learned that e1071 now provides direct multiclass classification. We leave the code here for now.

```
# DFCoarse <-DF DFCoarse$quality_cat_low_vs_med_high <-
# as.factor(ifelse(wine.data$quality<5, 'LOW', 'MED_HIGH'))
# DFCoarse$quality_cat_high_vs_low_med <-
# as.factor(ifelse(wine.data$quality>6, 'HIGH', 'LOW_MED'))
# DFCoarse$quality_cat_med_vs_low_high <- as.factor(ifelse((
# wine.data$quality==6 | wine.data$quality==5), 'MED', 'LOW_HIGH'))
# DFCoarse$quality <-NULL DFCoarseTrain <-DFCoarse[train,] DFCoarseTest
# <-DFCoarse[-train,] gamma_default <- 1/(ncol(DFCoarseTrain)-1) gamma_list
# <-((1:10)/5)*gamma_default cost_list=c(0.001 , 0.01, 0.1, 1,5,10,100)
# #Overrides for fast development change back! gamma_list<-c(.17)
# cost_list<-c(.1) #LOW VS REST
# tune.svm.quality_cat_low_vs_med_high=tune(svm,quality_cat_low_vs_med_high~.,
# data=DFCoarseTrain , kernel ='radial',ranges
# =list(gamma=gamma_list, cost=cost_list ),type='nu-classification')
# svm.fit.quality_cat_low_vs_med_high
# <-tune.svm.quality_cat_low_vs_med_high$best.model #MED VS REST
# tune.svm.quality_cat_med_vs_low_high=tune(svm,quality_cat_med_vs_low_high~.,
# data=DFCoarseTrain , kernel ='radial',ranges
# =list(gamma=gamma_list, cost=cost_list ),type='nu-classification')
# svm.fit.quality_cat_med_vs_low_high<-tune.svm.quality_cat_med_vs_low_high$best.model
# #HIGH VS REST
# tune.svm.quality_cat_high_vs_low_med=tune(svm,quality_cat_high_vs_low_med~,
```

```

# data=DFCoarseTrain , kernel ='radial',ranges
# =list(gamma=gamma_list, cost=cost_list ),type='nu-classification')
# sum.fit.quality_cat_high_vs_low_med<-tune.svm.quality_cat_high_vs_low_med$best.model

# sum.pred.quality_cat_low_vs_med_high <-
# predict(svm.fit.quality_cat_low_vs_med_high, DFCoarseTest)
# sum.pred.quality_cat_med_vs_low_high <-
# predict(svm.fit.quality_cat_med_vs_low_high, DFCoarseTest)
# sum.pred.quality_cat_high_vs_low_med <-
# predict(svm.fit.quality_cat_high_vs_low_med, DFCoarseTest)

DFCoarse <- DF
DFCoarse$quality_cat <- factor(ifelse(wine.data$quality < 5, "LOW", ifelse(wine.data$quality > 6, "HIGH", "MED")))
DFCoarse$quality <- NULL
DFCoarseTrain <- DFCoarse[train, ]
DFCoarseTest <- DFCoarse[-train, ]

gamma_default <- 1/(ncol(DFCoarseTrain) - 1)

gamma_list <- ((1:10)/5) * gamma_default
cost_list = c(0.001, 0.01, 0.1, 1, 5, 10, 100)

tune.svm.quality_cat = tune(svm, quality_cat ~ ., data = DFCoarseTrain, kernel = "polynomial",
                             ranges = list(gamma = gamma_list, cost = cost_list))
svm.fit.quality_cat <- tune.svm.quality_cat$best.model

svm.pred.quality_cat <- predict(svm.fit.quality_cat, DFCoarseTest)
TB <- table(svm.pred.quality_cat, DFCoarseTest$quality_cat)
pander(TB)

```

	HIGH	LOW	MED
HIGH	21	2	22
LOW	0	5	13
MED	41	13	416

```
ACC_Multiclass = (sum(diag(TB)))/length(DFCoarseTest$quality_cat)
```

```

method.accuracy <- data.frame(method = c("Multiclass SVM 3 class", "svm rbf nu 2 class",
                                         "lda 3 class", "lda 6 class", "glm 2 class", "linear svm 2 class", "rbf svm 2 class",
                                         "polynomial svm 2 class"), accuracy = c(ACC_Multiclass, ACC_Radial_nuSVM,
                                         ACC_lda_coarse, ACC_lda, ACC_glm_binomial, ACC_Linear_SVM, ACC_Radial_SVM,
                                         ACC_Polynomial_SVM))

method.accuracy <- method.accuracy[order(method.accuracy$accuracy, decreasing = TRUE),
]

pander(method.accuracy, caption = "Accuracy by Method")

```

Table 14: Accuracy by Method

	method	accuracy
3	lda 3 class	0.8443
1	Multiclass SVM 3 class	0.8293
2	svm rbf nu 2 class	0.7505
7	rbf svm 2 class	0.7486
6	linear svm 2 class	0.7317
8	polynomial svm 2 class	0.6979
5	glm 2 class	0.6886
4	lda 6 class	0.5647

Now we train a reduced model choosing the significant variables from the glm

```

DFCoarse <- DF[, c("fixed.acidity", "volatile.acidity", "chlorides", "total.sulfur.dioxide",
  "density", "sulphates", "alcohol")]
DFCoarse$quality_cat <- factor(ifelse(wine.data$quality < 5, "LOW", ifelse(wine.data$quality >
  6, "HIGH", "MED")))
DFCoarse$quality <- NULL
DFCoarseTrain <- DFCoarse[train, ]
DFCoarseTest <- DFCoarse[-train, ]

gamma_default <- 1/(ncol(DFCoarseTrain) - 1)

gamma_list <- ((1:10)/5) * gamma_default
cost_list = c(0.001, 0.01, 0.1, 1, 5, 10, 100)

tune.svm.quality_cat = tune(svm, quality_cat ~ ., data = DFCoarseTrain, kernel = "polynomial",
  ranges = list(gamma = gamma_list, cost = cost_list))
svm.fit.quality_cat <- tune.svm.quality_cat$best.model

svm.pred.quality_cat <- predict(svm.fit.quality_cat, DFCoarseTest)
TB <- table(svm.pred.quality_cat, DFCoarseTest$quality_cat)
pander(TB)

```

	HIGH	LOW	MED
HIGH	15	0	8
LOW	0	2	3
MED	47	18	440

```
ACC_Multiclass_Polynomial_subset = (sum(diag(TB)))/length(DFCoarseTest$quality_cat)
```

```
ACC_Class_HIGH = diag(TB)[1]/sum(DFCoarseTest$quality_cat == "HIGH")
ACC_Class_HIGH
```

```
##      HIGH
## 0.2419355
```

```

ACC_Class_LOW = diag(TB)[2]/sum(DFCoarseTest$quality_cat == "LOW")
ACC_Class_LOW

## LOW
## 0.1

ACC_Class_MED = diag(TB)[3]/sum(DFCoarseTest$quality_cat == "MED")
ACC_Class_MED

## MED
## 0.9756098

pander(data.frame(class = c("LOW", "MED", "HIGH"), accuracy = c(ACC_Class_LOW,
ACC_Class_MED, ACC_Class_HIGH)), caption = "Multiclass Polynomial Subset SVM accuracy by class")

```

Table 16: Multiclass Polynomial Subset SVM accuracy by class

	class	accuracy
LOW	LOW	0.1
MED	MED	0.9756
HIGH	HIGH	0.2419

Looking at the accuracy across the classes we see that a 2 class model may be preferable.

There are ways to reduce the error introduced by class skew such as limiting the number of the over-represented class, or tuning the model to enhance the sensitivity or specificity of some classes. If there a strong desire to have more than two classes - we'd consider applying these methods to the multiple classifier one-versus all approach. We could train the 2 class classifiers for the classes with low representation to favor the underrepresented class and use a voting scheme for the final classification.

It's possible that we could build a generic classifier for the middle levels of quality and a separate classifier for the edge cases of very low and very high as an anomaly detection problem. In this case we would apply both classifiers in succession and use the result of the anomaly detector if there was a positive classification for low or high quality. The anomaly detector itself could be two separate classifiers - one for quality 3 and one for quality 8.

Training with balanced test set

Multiclass Polynomial Subset Balanced Training

```

DFCoarse <- DF[, c("fixed.acidity", "volatile.acidity", "chlorides", "total.sulfur.dioxide",
"density", "sulphates", "alcohol")]

DFCoarse <- data.frame(scale(DF, center = TRUE, scale = TRUE))
DFCoarse$quality_cat <- factor(ifelse(wine.data$quality < 5, "LOW", ifelse(wine.data$quality >

```

```

  6, "HIGH", "MED")))
DFCoarse$quality <- NULL
DFCoarseTrain <- DFCoarse[train, ]
DFM <- DFCoarseTrain[DFCoarseTrain$quality_cat == "MED", ]
DFM <- DFM[1:200, ]
DFL <- DFCoarseTrain[DFCoarseTrain$quality_cat == "LOW", ]
DFH <- DFCoarseTrain[DFCoarseTrain$quality_cat == "HIGH", ]
DFCoarseTrain <- rbind(DFL, DFM, DFH)
DFCoarseTest <- DFCoarse[-train, ]

gamma_default <- 1/(ncol(DFCoarseTrain) - 1)

gamma_list <- ((1:10)/5) * gamma_default
cost_list = c(0.001, 0.01, 0.1, 1, 5, 10, 100)

tune.svm.quality_cat = tune(svm, quality_cat ~ ., data = DFCoarseTrain, kernel = "polynomial",
  ranges = list(gamma = gamma_list, cost = cost_list))
svm.fit.quality_cat <- tune.svm.quality_cat$best.model

svm.pred.quality_cat <- predict(svm.fit.quality_cat, DFCoarseTest)
TB <- table(svm.pred.quality_cat, DFCoarseTest$quality_cat)
pander(TB)

```

	HIGH	LOW	MED
HIGH	41	3	52
LOW	1	3	21
MED	20	14	378

```

ACC_Multiclass_Polynomial_subset_balanced = (sum(diag(TB)))/length(DFCoarseTest$quality_cat)

ACC_Class_HIGH = diag(TB)[1]/sum(DFCoarseTest$quality_cat == "HIGH")
ACC_Class_HIGH

##      HIGH
## 0.6612903

ACC_Class_LOW = diag(TB)[2]/sum(DFCoarseTest$quality_cat == "LOW")
ACC_Class_LOW

##    LOW
## 0.15

ACC_Class_MED = diag(TB)[3]/sum(DFCoarseTest$quality_cat == "MED")
ACC_Class_MED

##      MED
## 0.8381375

```

```
pander(data.frame(class = c("LOW", "MED", "HIGH"), accuracy = c(ACC_Class_LOW,
    ACC_Class_MED, ACC_Class_HIGH)), caption = "Multiclass Polynomial Subset Balanced SVM accuracy by class")
```

Table 18: Multiclass Polynomial Subset Balanced SVM accuracy by class

	class	accuracy
LOW	LOW	0.15
MED	MED	0.8381
HIGH	HIGH	0.6613

Multiclass RBF Subset Balanced Training

```
tune.svm.quality_cat = tune(svm, quality_cat ~ ., data = DFCoarseTrain, kernel = "polynomial",
    ranges = list(gamma = gamma_list, cost = cost_list))
svm.fit.quality_cat <- tune.svm.quality_cat$best.model

svm.pred.quality_cat <- predict(svm.fit.quality_cat, DFCoarseTest)
TB <- table(svm.pred.quality_cat, DFCoarseTest$quality_cat)
pander(TB)
```

	HIGH	LOW	MED
HIGH	41	3	52
LOW	1	3	21
MED	20	14	378

```
ACC_Multiclass_RBF_subset_balanced = (sum(diag(TB)))/length(DFCoarseTest$quality_cat)
```

```
ACC_Class_HIGH = diag(TB)[1]/sum(DFCoarseTest$quality_cat == "HIGH")
ACC_Class_HIGH
```

```
##      HIGH
## 0.6612903
```

```
ACC_Class_LOW = diag(TB)[2]/sum(DFCoarseTest$quality_cat == "LOW")
ACC_Class_LOW
```

```
##   LOW
## 0.15
```

```
ACC_Class_MED = diag(TB)[3]/sum(DFCoarseTest$quality_cat == "MED")
ACC_Class_MED
```

```
##      MED
## 0.8381375
```

```
pander(data.frame(class = c("LOW", "MED", "HIGH"), accuracy = c(ACC_Class_LOW,
ACC_Class_MED, ACC_Class_HIGH)), caption = "Multiclass RBF Subset Balanced SVM accuracy by class")
```

Table 20: Multiclass RBF Subset Balanced SVM accuracy by class

	class	accuracy
LOW	LOW	0.15
MED	MED	0.8381
HIGH	HIGH	0.6613

```
method.accuracy <- data.frame(method = c("Multiclass RBF Subset Balanced SVM",
"Multiclass Polynomial Subset Balanced SVM", "Multiclass Polynomial SVM 3 class",
"svm rbf nu 2 class", "lda 3 class", "lda 6 class", "glm 2 class", "linear svm 2 class",
"rbf svm 2 class", "polynomial svm 2 class"), accuracy = c(ACC_Multiclass_RBF_subset_balanced,
ACC_Multiclass_Polynomial_subset_balanced, ACC_Multiclass, ACC_Radial_nuSVM,
ACC_lda_coarse, ACC_lda, ACC_glm_binomial, ACC_Linear_SVM, ACC_Radial_SVM,
ACC_Polynomial_SVM))

method.accuracy <- method.accuracy[order(method.accuracy$accuracy, decreasing = TRUE),
]

pander(method.accuracy, caption = "Accuracy by Method")
```

Table 21: Accuracy by Method

	method	accuracy
5	lda 3 class	0.8443
3	Multiclass Polynomial SVM 3 class	0.8293
1	Multiclass RBF Subset Balanced SVM	0.7917
2	Multiclass Polynomial Subset Balanced SVM	0.7917
4	svm rbf nu 2 class	0.7505
9	rbf svm 2 class	0.7486
10	polynomial svm 2 class	0.6979
8	linear svm 2 class	0.6961
7	glm 2 class	0.6886
6	lda 6 class	0.5647