Wine Quality Analysis

Roko Mijic

24 May 2016

Visualizing the data

Start with a dataset about white wine quality:

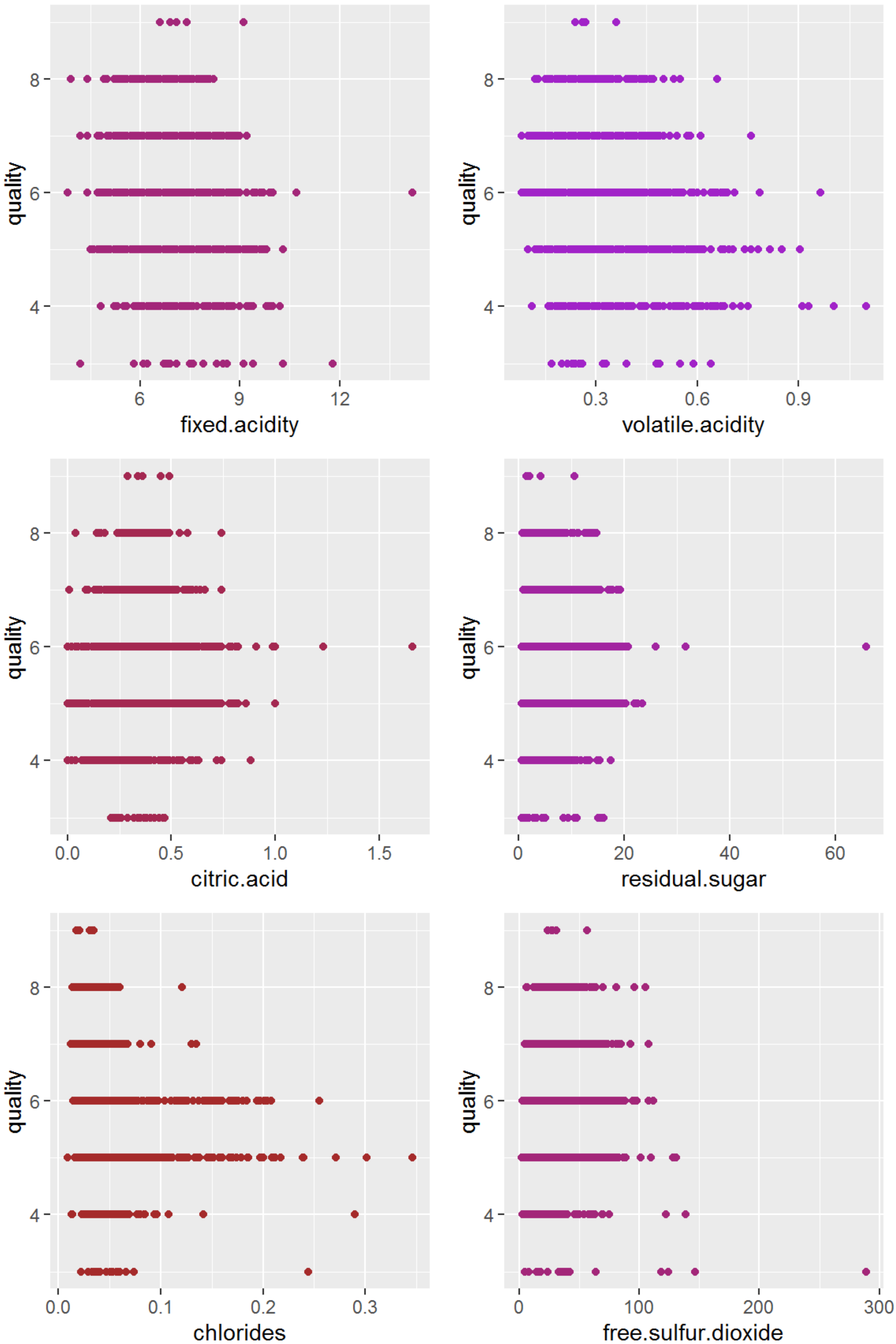
```
df_white[1:4,]
```

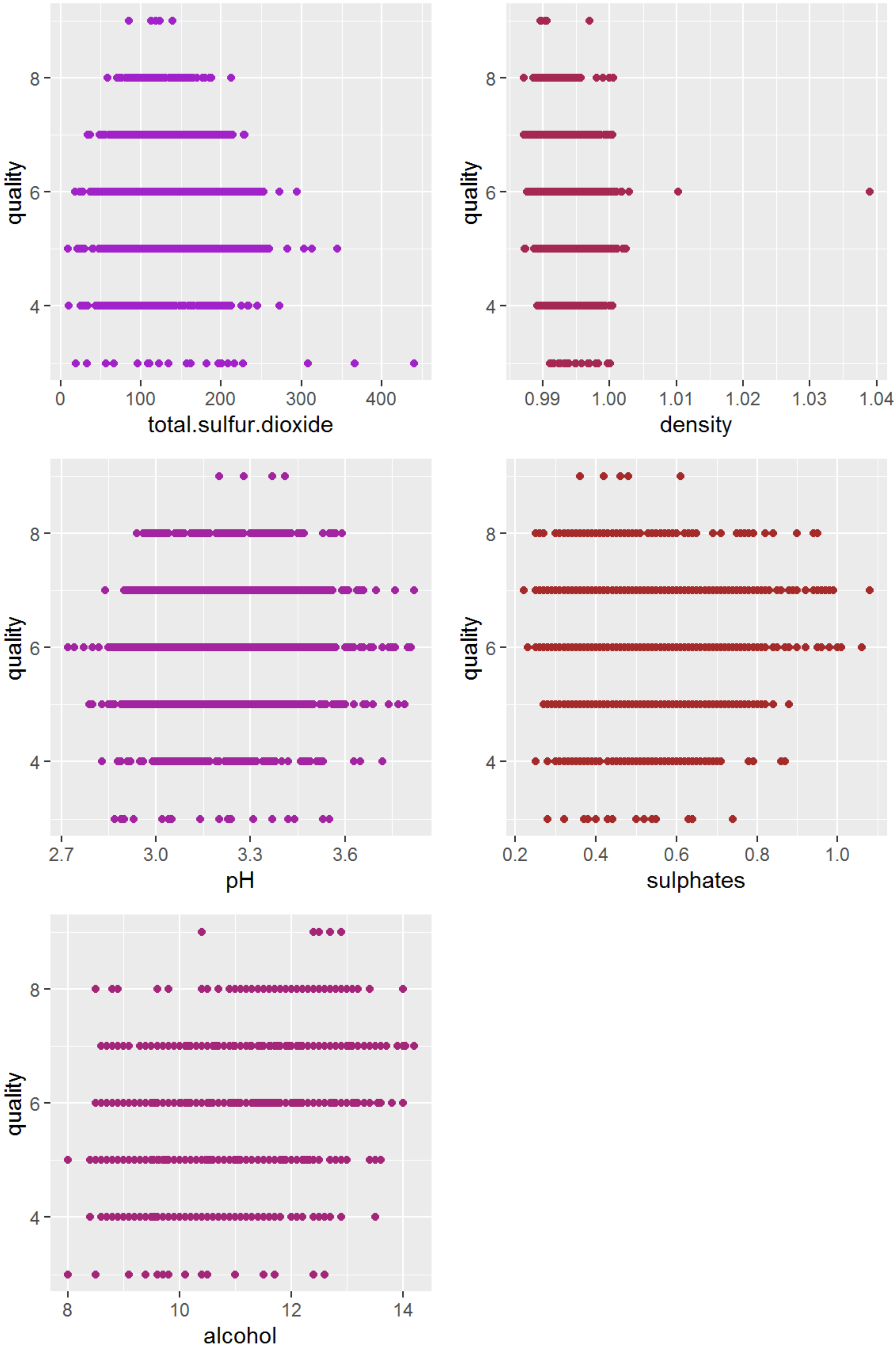
```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4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

Plot graphs of the individual variables versus wine quality:

```
numvars = ncol(df_white)-1
mypal <- colorRampPalette( c( "brown", "purple" ) )( numvars )

for(i in c(1:numvars)   )
{
  print(ggplot( data = df_white, aes(x = df_white[[i]], y=df_white$quality) ) + geom_po
int(color = mypal[[ (4*i)%%(numvars-1) + 1  ]] ) + labs(x = colnames(df_white)[[i]], y
= "quality" ))
}
```


quality
8
6
4
6
9
12
fixed.acidity
quality
8
6
4
0.3
0.6
0.9
volatile.acidity
quality
8
6
4
0.0
0.5
1.0
1.5
citric.acid
quality
8
6
4
0
20
40
60
residual.sugar
quality
8
6
4
0.0
0.1
0.2
0.3
chlorides
quality
8
6
4
0
100
200
300
free.sulfur.dioxide



Linear model training

Use caret to train a linear model to predict wine quality. First create training and test sets:

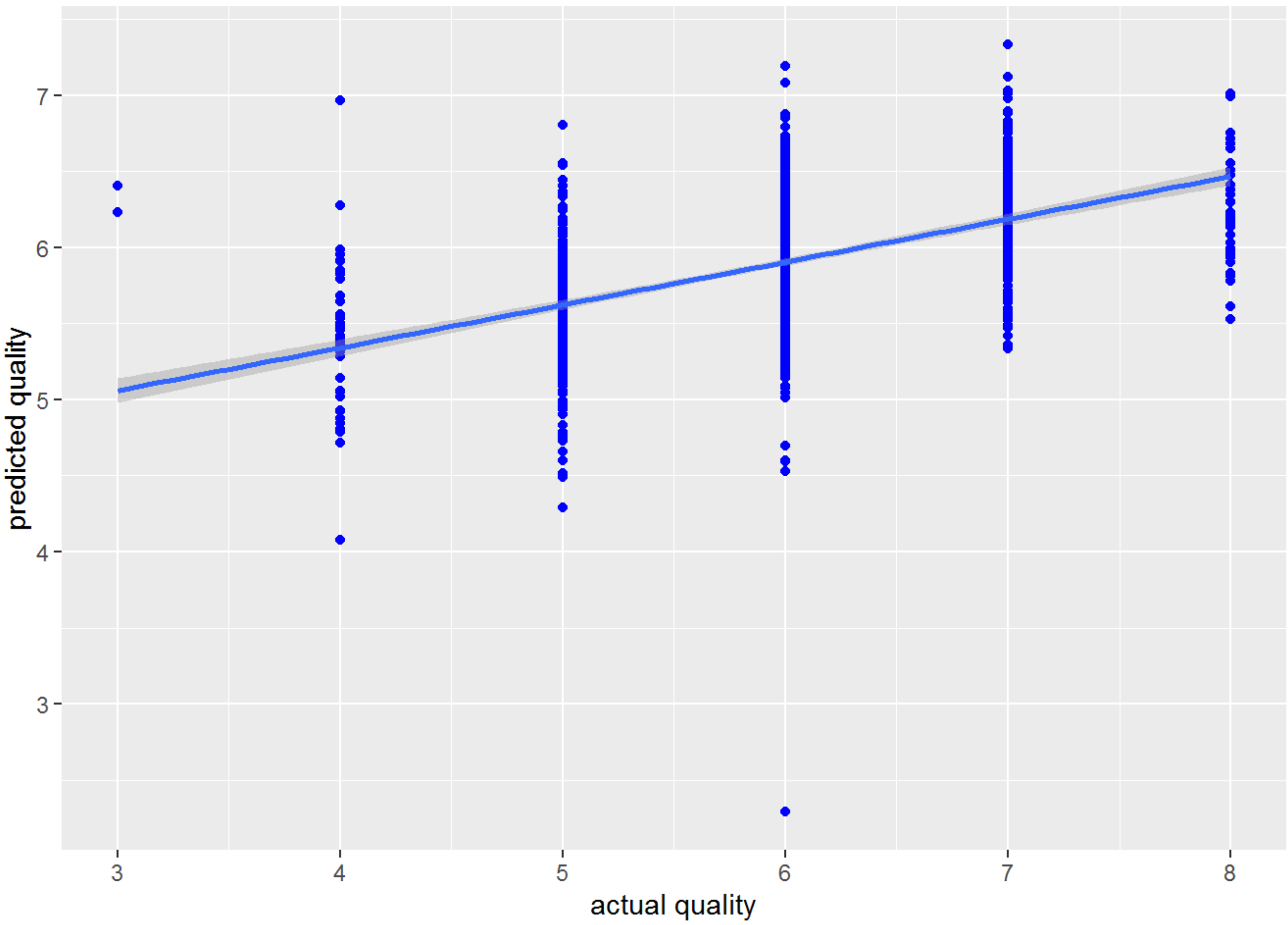
```
inTraining <- createDataPartition(df_white$quality, p = .75, list = FALSE)
training <- df_white[ inTraining,]
testing  <- df_white[-inTraining,]
```

Then use caret to train an optimal linear model:

```
fitControl <- trainControl(method = "cv", number = 3)
glmFit1 = train(quality ~ ., data = training, method = "glmnet",  trControl = fitContro
l)
predicted_linear_quality = predict(glmFit1, dplyr::select(testing,-quality) )
df_linear_fit = data.frame(testing$quality, predicted_linear_quality )
```

Plot the linear fit - predicted vs actual:

```
ggplot( data = df_linear_fit, aes(x = testing.quality, y=predicted_linear_quality) ) +
geom_point(color = "blue" ) + labs(x = "actual quality", y = "predicted quality" ) + ge
om_smooth(method = "lm")
```



Finally find the RMSE for the linear model:

```
RMSE(testing$quality, df_linear_fit$predicted_linear_quality )
```

```
## [1] 0.7454905
```

Examine the coefficients of the best model:

```
coef(glmFit1$finalModel, s = glmFit1$bestTune$lambda )
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)           1.824889e+02
## fixed.acidity         7.651159e-02
## volatile.acidity     -1.863768e+00
## citric.acid           7.305169e-02
## residual.sugar        9.144309e-02
## chlorides            -1.107119e-01
## free.sulfur.dioxide   3.786642e-03
## total.sulfur.dioxide -3.890006e-04
## density              -1.828907e+02
## pH                    7.884834e-01
## sulphates             8.466415e-01
## alcohol               1.491766e-01
```

Nonlinear modelling K-neasest neighbours

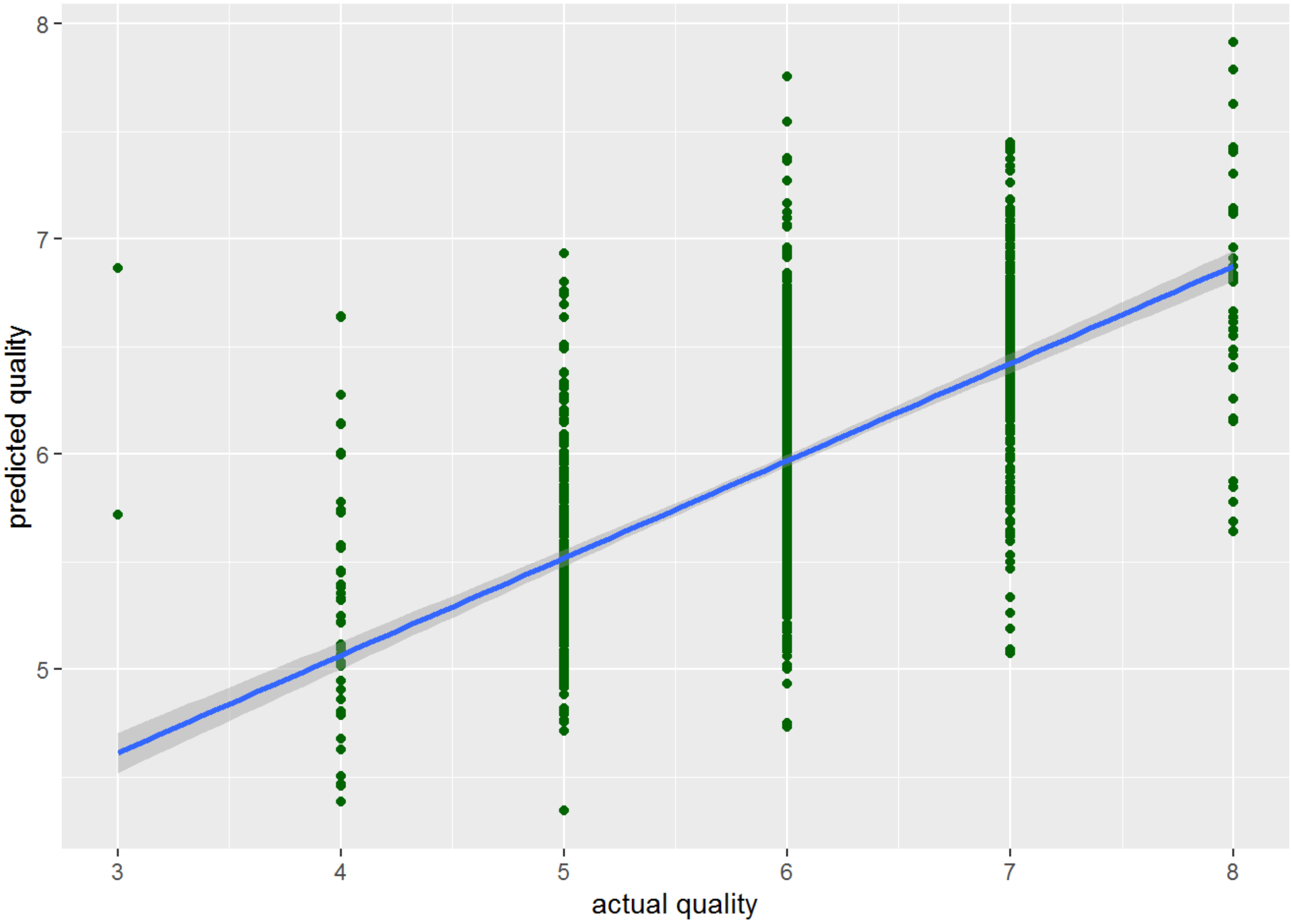
Next, we will use K-Nearest Neighbors to try to predict the wine quality.

```
KNN_fit = train(quality ~ ., data = training, method = "kknn", tuneLength=10)
predicted_KNN_quality = predict(KNN_fit, dplyr::select(testing,-quality) )
df_KNN_fit = data.frame(testing$quality, predicted_KNN_quality )
head(df_KNN_fit)
```

```
##   testing.quality predicted_KNN_quality
## 1               6              5.502320
## 2               5              5.926795
## 3               5              5.511211
## 4               8              6.637029
## 5               6              5.478943
## 6               6              6.394241
```

plot an actual vs predited fit for K-nearest neighbours:

```
ggplot( data = df_KNN_fit, aes(x = testing.quality, y=predicted_KNN_quality) ) + geom_p
oint(color = "Darkgreen" ) + labs(x = "actual quality", y = "predicted quality" ) + geo
m_smooth(method = "lm")
```



Find the RMSE for KNN:

```
RMSE(testing$quality, df_KNN_fit$predicted_KNN_quality)
```

```
## [1] 0.674283
```

Here the KNN algorithm manages to outperform the linear model, RMSE is 0.67136, versus 0.72143 (linear).

Regression trees

Construct a regression tree model to predict white wine quality:

```
regtree_fit = rpart(quality ~ ., data = training)
```

We can print the tree out:

```
print(regtree_fit)
```

```
## n= 3674
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 3674 2927.30500 5.880240
##    2) alcohol< 10.85 2319 1393.37900 5.610608
##      4) volatile.acidity>=0.2475 1328  658.16790 5.390813
##        8) free.sulfur.dioxide< 17.5 181   90.99448 4.994475 *
##        9) free.sulfur.dioxide>=17.5 1147  534.25460 5.453357 *
##      5) volatile.acidity< 0.2475 991  585.08380 5.905146 *
##    3) alcohol>=10.85 1355 1076.79400 6.341697
##      6) free.sulfur.dioxide< 11.5 88   92.86364 5.386364 *
##      7) free.sulfur.dioxide>=11.5 1267  898.03790 6.408051
##       14) alcohol< 11.74167 617  419.13130 6.170178 *
##       15) alcohol>=11.74167 650  410.85540 6.633846 *
```

And we can get predictions for the dataset

```
predicted_regtree_quality = predict(regtree_fit, dplyr::select(testing,-quality) )
df_regtree_fit = data.frame(testing$quality, predicted_regtree_quality )
head(df_regtree_fit)
```

```
##    testing.quality predicted_regtree_quality
## 4                6                  5.905146
## 11               5                  5.386364
## 12               5                  5.905146
## 18               8                  6.633846
## 19               6                  6.170178
## 27               6                  5.905146
```