Prediction of the Quality of Red and White Wines

Aggarwal, Nandan nsa52@cornell.edu

Choe, Ben bmc243@cornell.edu

Harris, Jalen jah668@cornell.edu

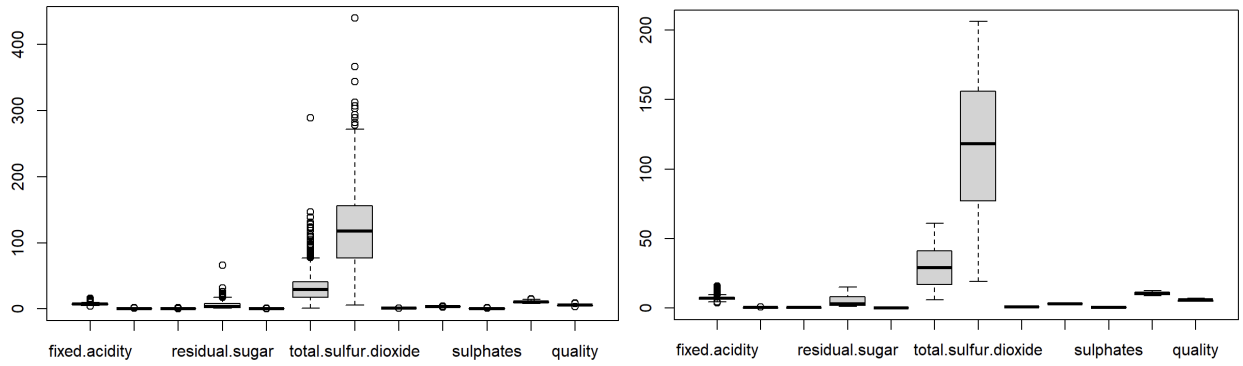
Nirmal, Srinivasan ns757@cornell.edu

December 6, 2022

1 Introduction

People first began to drink wine over five thousand years ago. Since then, viticulture and viniculture have been experimented with and studied vigorously in order to produce thousands of wine varieties and production methods that create wines of different tastes and quality. This study will explore how different factors of wine affect its quality. The original data sets for this study consist of thousands of individual wines, with each wine having the following variables measured: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulfates, and alcohol. Additionally, each wine has also been given a score for the wine's quality ranging from 0-10. The data was collected in two separate sets, one for red wine and one for white wine, and no other colors were used in this study. For the models we developed, we used the quality score as our dependent variable, and we explored how the other measured characteristics affect this quality score to ultimately be able to predict how a wine will be scored based on its measurable traits.

Due to the nature of the data, the data processing and model selection techniques used in this study are those of regression rather than classification. The idea of using classification methods to cluster the wines into groups based on their quality score (i.e., eleven clusters,



(a) Prior to capping

(b) After capping

Figure 1: Boxplots for each variable in the wine dataset at different stages of outlier removal

with each cluster having a different quality score of zero through ten) was explored, but the clusters would not be independent from each other. A wine classified as a two would be more similar to the wines classified as a three than the wines classified as a four, five, and so forth. Thus, it is unnecessary to turn the quantitative quality variable into classifiable categories. Instead, regression analysis will be able to predict quality more accurately.

The following topics will be explored in the remainder of the study: data processing techniques and how the data was cleaned/structured, the model selection process and how certain regression techniques and parameters were chosen, and lastly the data analysis which includes the validation set methods and our ultimate prediction model.

2 Data Processing

The original data came in two sets: one for red wines and one for white wines. The objective of this study was to predict the quality of wines, so the two datasets were combined to create one dataset with red and white wines. The color of a wine has no bearing on its quality and it is instead seen as a method to categorize the wines. Because of this, the color variable was eliminated from our combined data set. After compiling the data and deleting the color variable, the dataset needed to be checked for outliers and then trimmed or adjusted as necessary.

Figure 1a shows a series of box plots which elucidate the distribution of each variable in the combined data set before outliers were adjusted. Visually, the data needed to be adjusted for outliers as there is evidence of outliers in the box plots. To solve this, the capping method was applied to the dataset. Capping is one of the most widely used and robust methods in the data science industry for outlier treatment. In this method, the values which lie above the capping value (in this case 5th percentile and 95th percentile are considered the lower and upper capping values) are replaced with the capped values. In other words, instead of removing the outliers (another method for outlier treatment) altogether, which might lead to loss of information, they are converted and brought into the range or limit of our data. The box plots in figure 1b reflect the data after the capping method was applied. All of the points that lay outside the 5th-95th percentile have been adjusted, and this is reflected in box plots as the outlying points are no longer present. The rest of the analysis in this study was applied to this capped dataset.

3 Model Selection

Due to the nature of our response variable (quality) being a rating 0-10, we decided that linear regression techniques would be the most effective way in predicting quality. An important part of any sort of regression is the model selection and choosing which parameters to include. Best subset selection is widely regarded as the safest selection method because it looks at all p models that contain exactly one predictor, all models that contain two predictors, and so forth. Best subset selection is impractical in some cases with a large amount of parameters because it is computationally expensive. However, with our data set only containing 11 parameters, best subset selection was easily run using the 'leaps' library and the 'regsubsets' function. In an attempt to find the ideal number of predictors using the different criterion, the following results were obtained shown in figure 2. The maximum adjusted R-squared and minimum adjusted C_p both suggest that 10 variables should be selected while the minimum

adjusted BIC (Bayesian information criterion) suggests that only 9 variables be selected. This aligns with the math behind BIC because it penalizes each additional variable more than adjusted R-squared or C_p, so it makes sense that the BIC suggests one fewer parameter.

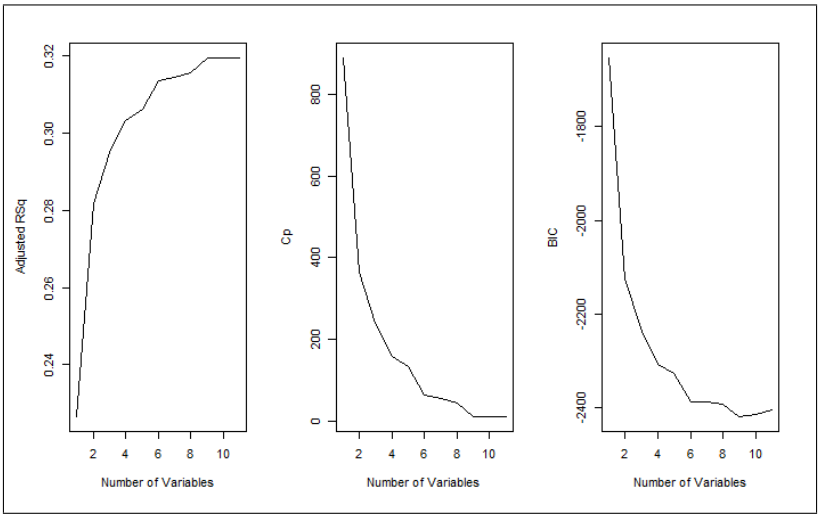


Figure 2: Plots of the Adjusted R^2 (left), C_p (middle), and BIC (right) vs. the number of variables to be selected in the model.

Because the three criteria suggest different model sizes, various cross validation methods will be used to determine training and test sets and how many parameters to include in the model. First, the validation set approach was used to calculate the test error rates of the best models for each amount of parameters. The output below shows us that, by using the validation set approach, the test error rate is lowest with nine variables because [9] 0.3616509 is the lowest value among all of the validation errors as shown in figure 2.

```
##  [1] 0.4148673 0.3828393 0.3746997 0.3705519 0.3691098 0.3655426 0.3650684
##  [8] 0.3639048 0.3616509 0.3616630 0.3618294
```

Figure 3: Test errors of best subset selection used in tandem with the validation set approach

Furthermore, a 10-fold cross validation was also implemented to confirm these results. For this analysis, best subset selection was used with each of the k training sets. After implementing the k-fold cross validation, the resulting validation errors are shown in figure 4. Just as in the validation set approach method, the model with nine parameters yielded the

smallest test mean squared error (MSE) as well. Both cross validation methods had lowest validation errors with a nine parameter model, and the BIC criterion was lowest at nine parameters. This is significant evidence to choose the best subset selection model with nine parameters as the optimal model for this study. Running this nine parameter best subset model yields the coefficients shown in figure 5.

```
##         1         2         3         4         5         6         7         8
## 0.4136115 0.3841631 0.3771888 0.3727125 0.3712452 0.3674260 0.3677570 0.3667030
##         9        10        11
## 0.3648552 0.3649498 0.3649542
```

Figure 4: Test errors of best subset selection used in tandem with 10-fold cross validataion

```
##          (Intercept)        fixed.acidity     volatile.acidity
##         70.003827333          0.069246882         -1.153656143
##       residual.sugar  free.sulfur.dioxide total.sulfur.dioxide
##          0.045698051          0.005878583         -0.002447426
##              density                   pH            sulphates
##        -69.311834099          0.520147565          0.874475435
##              alcohol
##          0.227325902
```

Figure 5: Coefficients for the 9 parameter linear regression model found from applying the best subset selection algorithm

Here, we can see that the two variables excluded from the original 11 parameters were citric acid and chlorides. This makes sense because if one were to run an ordinary least squares (OLS) regression on quality with all the variables in the data set, the only two estimators that are not statistically significant at the 1% level are citric acid and chlorides. The R output in figure 6 below shows the values of a general OLS regression and the p values can be seen on the right side column.

The only two parameters with p values greater than 0.01 are citric acid and chlorides, so it makes intuitive sense that our best subset model eliminated those two explanatory variables.

For the final step in our model selection progress, the integrity of a linear regression needed to be verified. A key assumption of an OLS regression that is commonly broken is the idea of none of the explanatory variables having *perfect* multicollinearity. To test if any of

```
Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.044e+02  1.410e+01   7.401 1.52e-13 ***
fixed.acidity         8.507e-02  1.576e-02   5.396 7.05e-08 ***
volatile.acidity     -1.492e+00  8.135e-02 -18.345  < 2e-16 ***
citric.acid          -6.262e-02  7.972e-02  -0.786   0.4322
residual.sugar        6.244e-02  5.934e-03  10.522  < 2e-16 ***
chlorides            -7.573e-01  3.344e-01  -2.264   0.0236 *
free.sulfur.dioxide   4.937e-03  7.662e-04   6.443 1.25e-10 ***
total.sulfur.dioxide -1.403e-03  3.237e-04  -4.333 1.49e-05 ***
density              -1.039e+02  1.434e+01  -7.248 4.71e-13 ***
pH                    4.988e-01  9.058e-02   5.506 3.81e-08 ***
sulphates             7.217e-01  7.624e-02   9.466  < 2e-16 ***
alcohol               2.227e-01  1.807e-02  12.320  < 2e-16 ***
color                 3.613e-01  5.675e-02   6.367 2.06e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 6: R output for a general least squares regression performed on the wine dataset

the variables in the data set suffered from this, a plot (figure 7) was generated to visualize the relationships each explanatory variable has with each other. In figure 7, each dot represents a regression run on the two crossed variables. The color and size of the dot correspond with the correlation of the regression, with larger and more opaque dots being closer to one or negative one (perfect correlation). One can see that the correlation between free sulfur dioxide and total sulfur dioxide, as well as the one with alcohol and density, are particularly strong. Although this does hinder the regression slightly by increasing the standard errors of the multicollinear parameters, they do not suffer from perfect multicollinearity and can stay in the model due to their significant effect on quality.

4 Exploring Alternative Models

To implement and check additional machine learning models, a Ridge and LASSO analysis (figures 8a and 8b) were also run. LASSO and Ridge regression continue to assume a parametric linear equation, but differ by applying a penalty. LASSO adds a L_1 (absolute value) penalty while Ridge applies a L_2-squared penalty. LASSO can create sparse models as it can shrink unimportant parameters to be close to zero. Given that we have 11 variables, this shrinkage process can create a model that is more interpretable. Both methods utilize a tuning parameter, λ, which controls the strength of the penalty. The ridge regression minimized

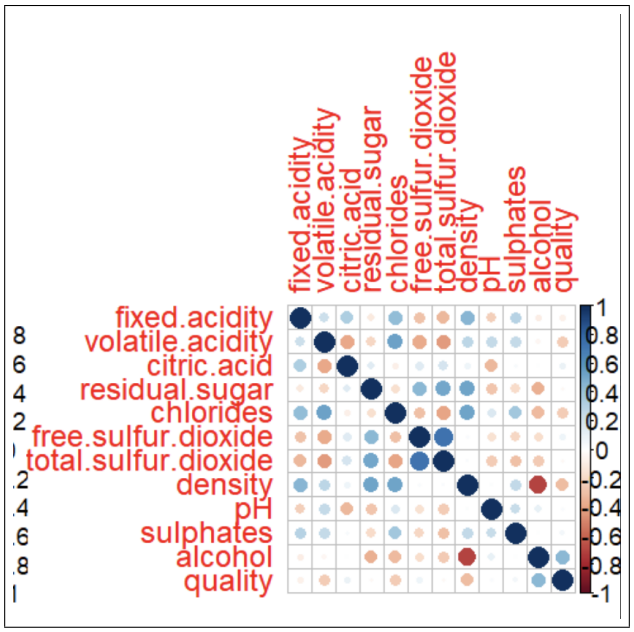


Figure 7: Correlation plot to visualize the relationships between the different variables in the dataset

the MSE to a value of 0.3605988 and produced the coefficients above. The LASSO on the other hand, produced a MSE of 0.3631217 and produced the following non-zero coefficients. This shows us that the following coefficients were reduced to zero: citric acid, fixed.acidity, and density. When compared to our best subset selection, there is some discrepancy as the best subset model eliminated citric acid and chloride while the LASSO reduced citric acid, fixed.acidity, and density to zero. However, LASSO and Ridge methods are most useful when best subset selection is computationally infeasible or to reduce overfitting, so for this data, the best model to predict wine quality is still the regression on page 5.

(Intercept)	fixed.acidity	volatile.acidity
40.934526532	0.043956261	-1.134589970
citric.acid	residual.sugar	chlorides
-0.062270550	0.030537484	-1.173505845
free.sulfur.dioxide	total.sulfur.dioxide	density
0.005148968	-0.002015011	-39.330410551
pH	sulphates	alcohol
0.345771173	0.785362246	0.243205550

(a) Ridge-Regression Model

(Intercept)	volatile.acidity	residual.sugar
2.474306957	-1.189634457	0.014669290
chlorides	free.sulfur.dioxide	total.sulfur.dioxide
-0.754664550	0.003832357	-0.001368393
pH	sulphates	alcohol
0.093997115	0.634427372	0.297747749

(b) LASSO model

Figure 8: Model coefficient estimates for two different shrinkage methods

5 Conclusion

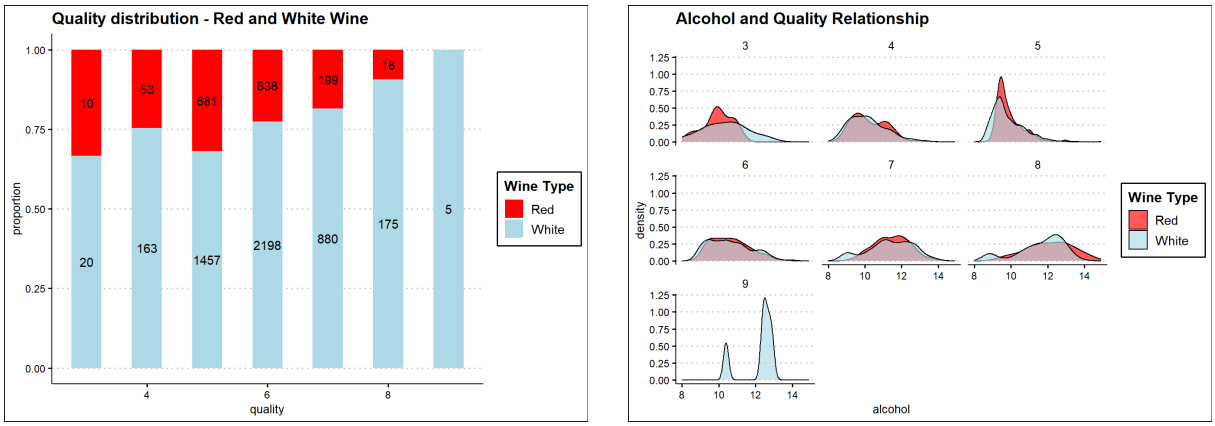
After combining the two data sets and cleaning it for outliers, various subset selection methods were experimented with to come up with the final model. Additionally, cross-validation methods were used to evaluate the models on different sets of training and test data. The final model that most accurately predicts wine quality is a **nine-parameter linear regression model** with parameters determined via best subset. The values of the coefficients are as follows:

$$\begin{aligned}\text{quality} = {} & 70.0038 + 0.0692(\text{fixed.acidity}) - 1.1537(\text{volatile.acidity}) + 0.0457(\text{residual.sugar}) \\ & + 0.0059(\text{free.sulfur.dioxide}) - 0.0024(\text{total.sulfur.dioxide}) - 69.3118(\text{density}) \\ & + 0.5201(\text{pH}) + 0.8745(\text{sulphates}) + 0.2273(\text{alcohol})\end{aligned}$$

and the CV error from a k-fold cross validation of this model is 0.3648. It is important to note that this CV error is actually slightly greater than the test MSEs of the ridge regression and the LASSO (0.3606 and 0.3631 respectively). However, the best subset model is still a more accurate estimator of wine quality because both the ridge and the LASSO are designed to reduce overfitting, and in this case, that is unnecessary. The rationale behind this is that the measurable characteristics of all wines fall within a reasonable range of each other. The model will not have to be extrapolated to wines that have outlier values of every trait simply because no wines exist as such. Thus, a model that is slightly overfit to the current dataset will still be acceptable even if the test MSE was marginally smaller than a non-overfitted model. After cleansing the data and experimenting with subset selections and k-fold cross-validations, the ultimate regression designed to predict the quality of wine is the nine-parameter best subset model described in the equation above.

6 Appendix

Some graphs made using ggplot are presented, depicting the various inherent relationships in the wine dataset. The amount of alcohol in the wine was the most positively correlated variable among others with quality in the wine dataset. Figure 10 shows the proportional imbalance between red and white wine data across the wine dataset, and there's not even a single observation for red wine under quality - '9'.



(a) Quality distribution of the red and white wines (b) Alcohol and quality relationships

Figure 9: Relationships between alcohol type and quality

Figure 10 shows evidence of Alcohol content being negatively correlated with density.

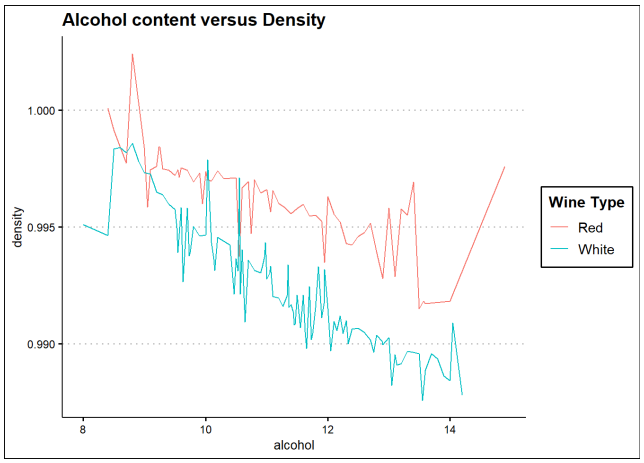


Figure 10: Relationship between density vs alcohol content (%)

MSE 5740 Final Project Code

Data Processing

Load the data

```
wine_data = read.csv("winequality-all.csv", header = T, na.strings = "?", stringsAsFactors = T)
wine_data$color <- factor(wine_data$color, levels=c(1, 0), labels=c('R', 'W'))

drop <- c("color")
wine_data = wine_data[,!(names(wine_data) %in% drop)]
```

Capping

```
library(scales)
wine_data$fixed.acidity <- squish(wine_data$fixed.acidity, quantile(wine_data$fixed_acidity, c(.
05, .95)))

wd_capped <- wine_data

varlist <- names(wine_data)
# varlist <- varlist[-length(varlist)]

for (i in varlist) {
    var <- eval(parse(text = paste0("wine_data$", i)))
    var <- squish(var, quantile(var, c(.05, .95)))
}


wd_capped$fixed.acidity <- squish(wd_capped$fixed.acidity, quantile(wd_capped$fixed_acidity, c(.
05, .95)))
wd_capped$volatile.acidity <- squish(wd_capped$volatile.acidity, quantile(wd_capped$volatile.aci
dity, c(.05, .95)))
wd_capped$citric.acid<- squish(wd_capped$citric.acid, quantile(wd_capped$citric.acid, c(.05, .95
)))
wd_capped$residual.sugar <- squish(wd_capped$residual.sugar, quantile(wd_capped$residual.sugar,
 c(.05, .95)))
wd_capped$chlorides <- squish(wd_capped$chlorides, quantile(wd_capped$chlorides, c(.05, .95)))
wd_capped$free.sulfur.dioxide <- squish(wd_capped$free.sulfur.dioxide, quantile(wd_capped$free.s
ulfur.dioxide, c(.05, .95)))
wd_capped$total.sulfur.dioxide <- squish(wd_capped$total.sulfur.dioxide, quantile(wd_capped$tota
l.sulfur.dioxide, c(.05, .95)))
wd_capped$density <- squish(wd_capped$density, quantile(wd_capped$density, c(.05, .95)))
wd_capped$pH <- squish(wd_capped$pH, quantile(wd_capped$pH, c(.05, .95)))
wd_capped$sulphates <- squish(wd_capped$sulphates, quantile(wd_capped$sulphates, c(.05, .95)))
wd_capped$alcohol <- squish(wd_capped$alcohol, quantile(wd_capped$alcohol, c(.05, .95)))
wd_capped$quality <- squish(wd_capped$quality, quantile(wd_capped$quality, c(.05, .95)))

# We compare the dimensions of the dataset before and after outlier removal

dim(wine_data)
```

```
## [1] 6497   12
```

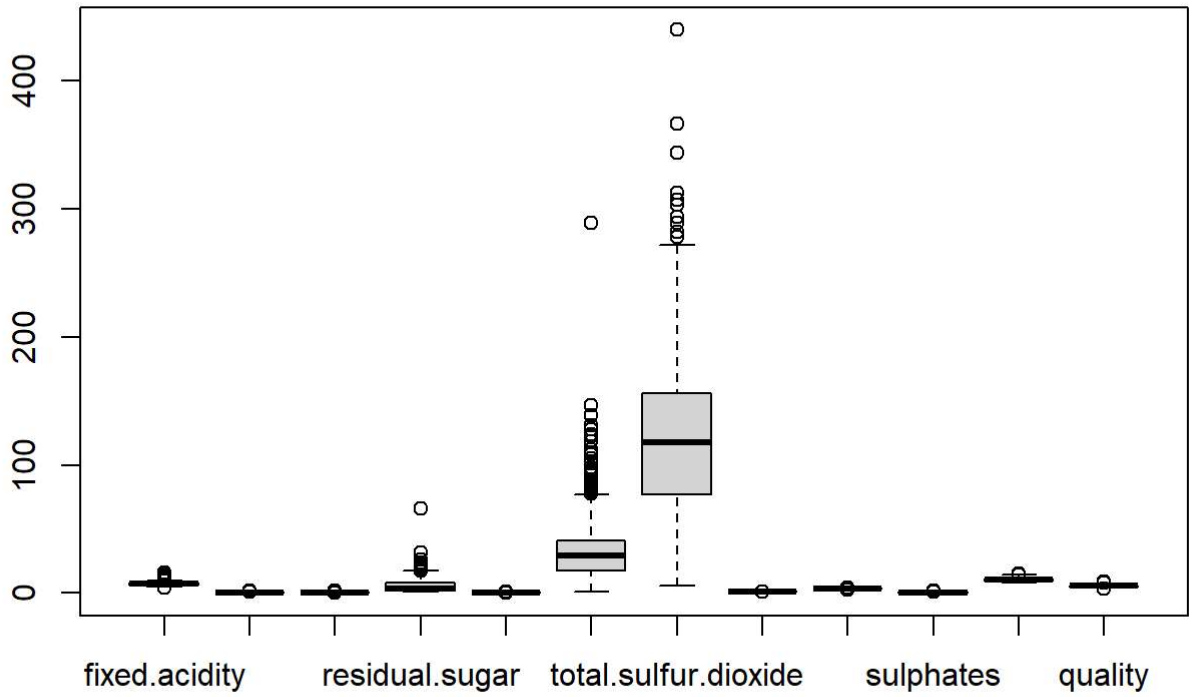
```
dim(wd_capped)
```

```
## [1] 6497   12
```

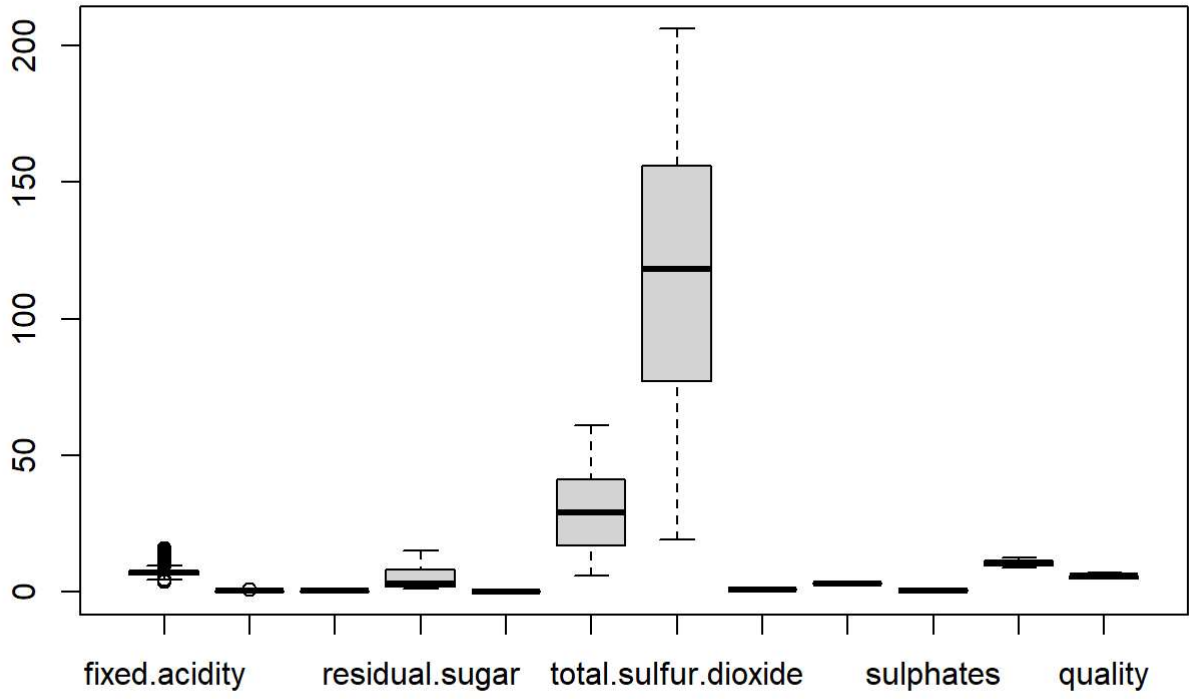
```
dim(wine_data)[1] - dim(wd_capped)[1]
```

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

```
boxplot(wine_data)
```

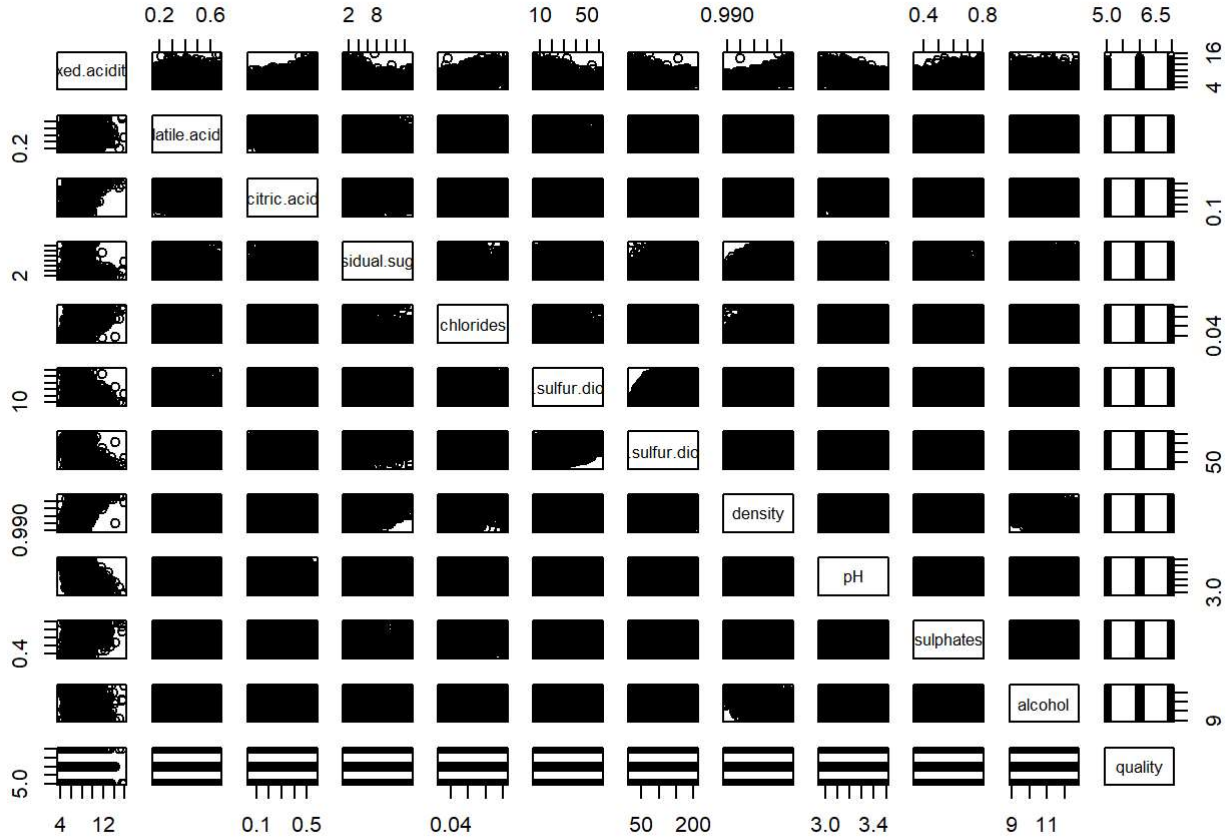



```
boxplot(wd_capped)
```




```
wine_data <- wd_capped
```

```
pairs(wine_data)
```



Summary of the data

```
summary(wine_data)
```

```
##  fixed.acidity    volatile.acidity  citric.acid    residual.sugar
##  Min.   : 3.800   Min.   :0.1600   Min.   :0.050   Min.   : 1.200
##  1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.250   1st Qu.: 1.800
##  Median : 7.000   Median :0.2900   Median :0.310   Median : 3.000
##  Mean   : 7.215   Mean   :0.3345   Mean   :0.315   Mean   : 5.334
##  3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.390   3rd Qu.: 8.100
##  Max.   :15.900   Max.   :0.6700   Max.   :0.560   Max.   :15.000
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.02800   Min.   : 6.00       Min.   : 19          Min.   :0.9899
##  1st Qu.:0.03800   1st Qu.:17.00       1st Qu.: 77          1st Qu.:0.9923
##  Median :0.04700   Median :29.00       Median :118          Median :0.9949
##  Mean   :0.05327   Mean   :30.03       Mean   :115          Mean   :0.9947
##  3rd Qu.:0.06500   3rd Qu.:41.00       3rd Qu.:156          3rd Qu.:0.9970
##  Max.   :0.10200   Max.   :61.00       Max.   :206          Max.   :0.9994
##        pH          sulphates         alcohol         quality
##  Min.   :2.970   Min.   :0.3500   Min.   : 9.00   Min.   :5.00
##  1st Qu.:3.110   1st Qu.:0.4300   1st Qu.: 9.50   1st Qu.:5.00
##  Median :3.210   Median :0.5100   Median :10.30   Median :6.00
##  Mean   :3.217   Mean   :0.5257   Mean   :10.48   Mean   :5.83
##  3rd Qu.:3.320   3rd Qu.:0.6000   3rd Qu.:11.30   3rd Qu.:6.00
##  Max.   :3.500   Max.   :0.7900   Max.   :12.70   Max.   :7.00
```

Removal of outliers (Interquartile range)

To begin, we must first identify the outliers in a dataset; typically, two methods are available. 1. z-scores 2. interquartile range

Description of methods

z-score method

The z-score indicates the number of standard deviations a given value deviates from the mean. A z-score is calculated using the following formula:

$$z = (X - \mu)/\sigma$$

where:

X is a single raw data value

μ is the population mean

σ is the population standard deviation

If an observation's z-score is less than -3 or larger than 3, it's considered an outlier.

We implement this method below:

```
z_scores <- as.data.frame(sapply(wine_data, function(wine_data) (abs(wine_data-mean(wine_data))/
sd(wine_data))))
no_outliers <- z_scores[!rowSums(z_scores>3), ]
# head(no_outliers)
dim(wine_data)
```

```
## [1] 6497   12
```

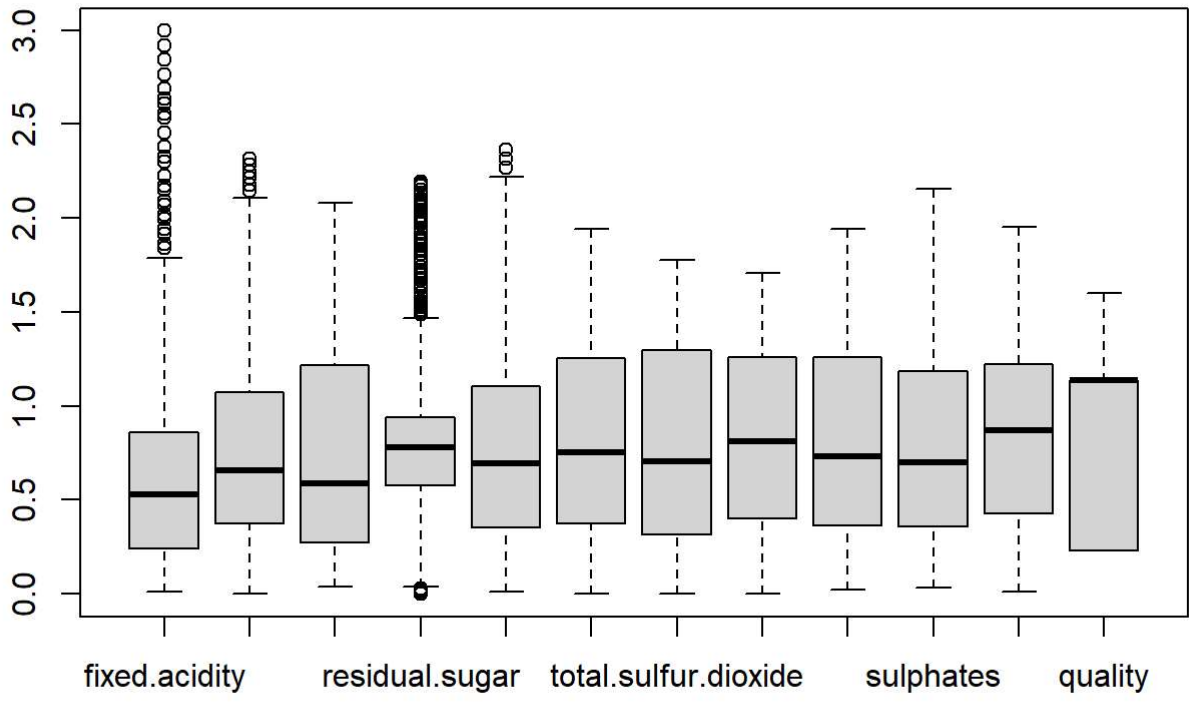
```
dim(no_outliers)
```

```
## [1] 6369   12
```

```
dim(wine_data)[1] - dim(no_outliers)[1]
```

```
## [1] 128
```

```
boxplot(no_outliers)
```



With the z-score method we see that we have 508 less observations in the dataset

Interquartile range method

In a dataset, it is the difference between the 75th percentile (Q3) and the 25th percentile (Q1).

The interquartile range (IQR) is a measurement of the spread of values in the middle 50%.

If an observation is 1.5 times the interquartile range more than the third quartile (Q3) or 1.5 times the interquartile range less than the first quartile (Q1), it is considered an outlier (Q1).

```
varlist <- names(wine_data)
# varlist <- varlist[-length(varlist)]

for (i in varlist) {
    var <- eval(parse(text = paste0("wine_data$", i)))
    Q1 <- quantile(var, 0.25)
    Q3 <- quantile(var, 0.75)
    iqr <- IQR(var)
    no_outliers <- subset(wine_data, var > (Q1 - 1.5*iqr) & var < (Q3 + 1.5*iqr) )

}

# We compare the dimensions of the dataset before and after outlier removal
dim(wine_data)
```

```
## [1] 6497   12
```

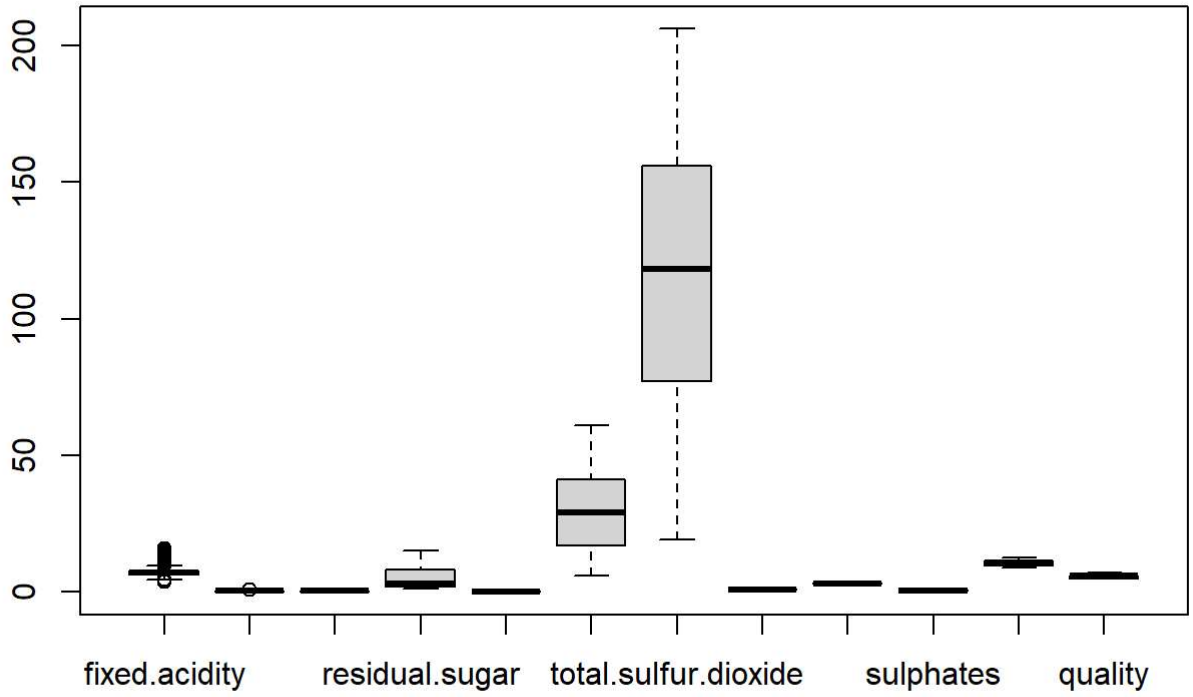
```
dim(no_outliers)
```

```
## [1] 6497   12
```

```
dim(wine_data)[1] - dim(no_outliers)[1]
```

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

```
boxplot(no_outliers)
```



With the z-score method we see that we have 228 less observations in the dataset

We should probably figure out which method we want to use at some point.

Linear model selection and regularization

Subset selection methods

Best Subset Selection

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.2.2
```

```
# Perform best subset selection using the regsubsets() function included
# in the leaps library
regfit.full <- regsubsets(quality ~ ., wine_data, nvmax = 11)

# Summary of the full best subset selection model, the models with 1-11 variables
# are shown below
# shows the best set of variables for each model size
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(quality ~ ., wine_data, nvmax = 11)
## 11 Variables  (and intercept)
##                       Forced in Forced out
## fixed.acidity             FALSE      FALSE
## volatile.acidity          FALSE      FALSE
## citric.acid               FALSE      FALSE
## residual.sugar            FALSE      FALSE
## chlorides                 FALSE      FALSE
## free.sulfur.dioxide       FALSE      FALSE
## total.sulfur.dioxide      FALSE      FALSE
## density                   FALSE      FALSE
## pH                        FALSE      FALSE
## sulphates                 FALSE      FALSE
## alcohol                   FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: exhaustive
##           fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1  ( 1 )  " "           " "              " "         " "            " "
## 2  ( 1 )  " "           "*"              " "         " "            " "
## 3  ( 1 )  " "           "*"              " "         " "            " "
## 4  ( 1 )  " "           "*"              " "         "*"            " "
## 5  ( 1 )  " "           "*"              " "         "*"            " "
## 6  ( 1 )  " "           "*"              " "         "*"            " "
## 7  ( 1 )  " "           "*"              " "         "*"            " "
## 8  ( 1 )  " "           "*"              " "         "*"            " "
## 9  ( 1 )  "*"           "*"              " "         "*"            " "
## 10  ( 1 ) "*"           "*"              "*"         "*"            " "
## 11  ( 1 ) "*"           "*"              "*"         "*"            "*"
##           free.sulfur.dioxide total.sulfur.dioxide density pH  sulphates
## 1  ( 1 )  " "                 " "                 " "     " " " "
## 2  ( 1 )  " "                 " "                 " "     " " " "
## 3  ( 1 )  " "                 " "                 " "     " " "*"
## 4  ( 1 )  " "                 " "                 " "     " " "*"
## 5  ( 1 )  " "                 "*"                 " "     " " "*"
## 6  ( 1 )  "*"                 "*"                 " "     " " "*"
## 7  ( 1 )  "*"                 "*"                 " "     "*" "*"
## 8  ( 1 )  "*"                 "*"                 "*"     "*" "*"
## 9  ( 1 )  "*"                 "*"                 "*"     "*" "*"
## 10  ( 1 ) "*"                 "*"                 "*"     "*" "*"
## 11  ( 1 ) "*"                 "*"                 "*"     "*" "*"
##           alcohol
## 1  ( 1 )  "*"
## 2  ( 1 )  "*"
## 3  ( 1 )  "*"
## 4  ( 1 )  "*"
## 5  ( 1 )  "*"
## 6  ( 1 )  "*"
## 7  ( 1 )  "*"
## 8  ( 1 )  "*"
## 9  ( 1 )  "*"
```

```
## 10  ( 1 ) "*"
## 11  ( 1 ) "*"
```

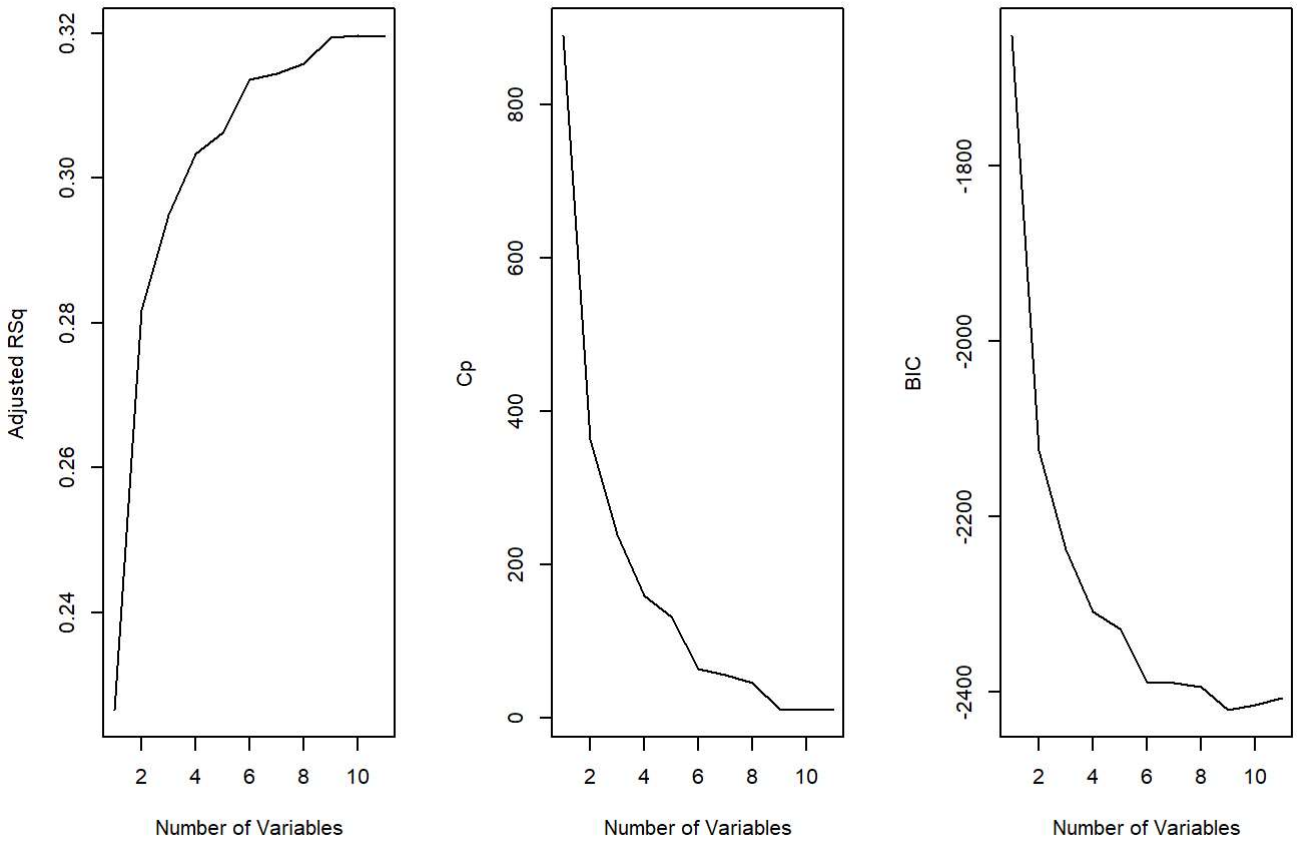
We now check to the different fitting criterion to see which model is best

```
reg.summary = summary(regfit.full)
```

```
par(mfrow = c(1, 3))
plot(reg.summary$adjr2, xlab = "Number of Variables" ,
ylab = "Adjusted RSq" , type = "l")

plot (reg.summary$cp, xlab = "Number of Variables" ,
ylab = "Cp" , type = "l")

plot (reg.summary$bic, xlab = "Number of Variables" ,
ylab = "BIC" , type = "l")
```



Find the ideal number of predictors using the different criterion

```
# Find the number of predictors that corresponds to the maximum adjusted Rsq val
which.max(reg.summary$adjr2)
```

```
## [1] 10
```

```
# Find the number of predictors that corresponds to the minimum adjusted Cp val
which.min(reg.summary$cp)
```

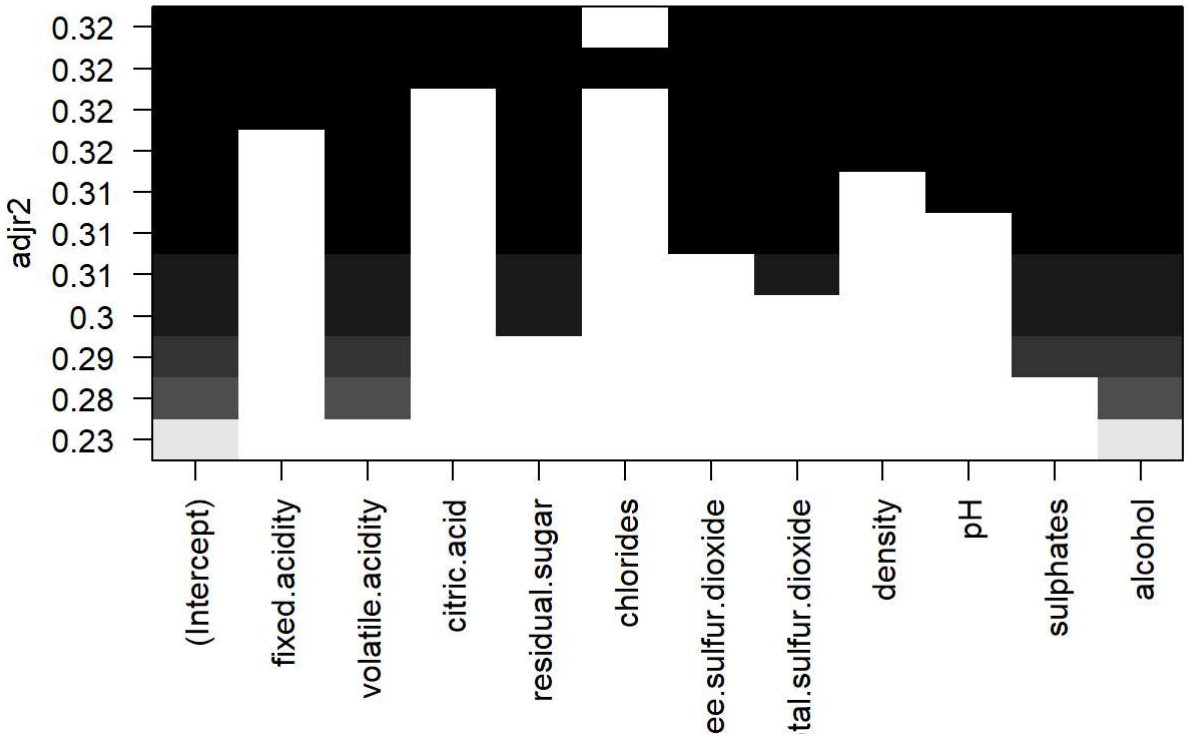
```
## [1] 10
```

```
# Find the number of predictors that corresponds to the minimum adjusted BIC val
which.min(reg.summary$bic)
```

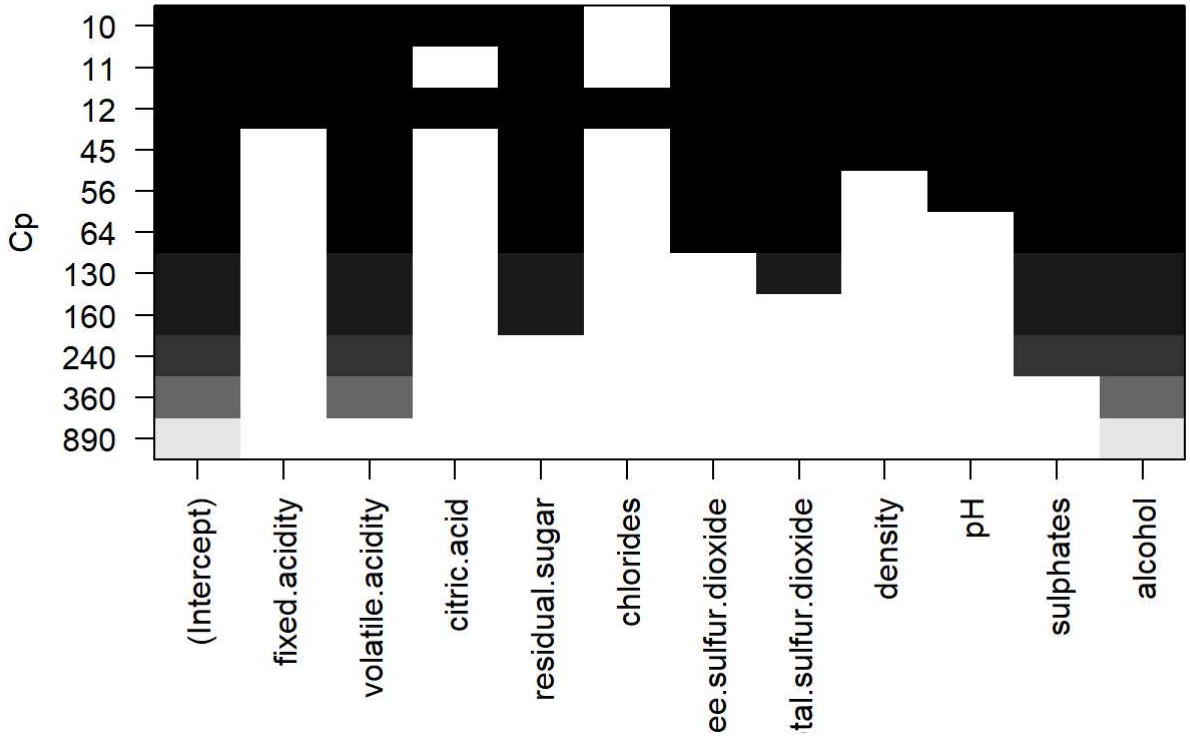
```
## [1] 9
```

Plot which models indicate the smallest statistic

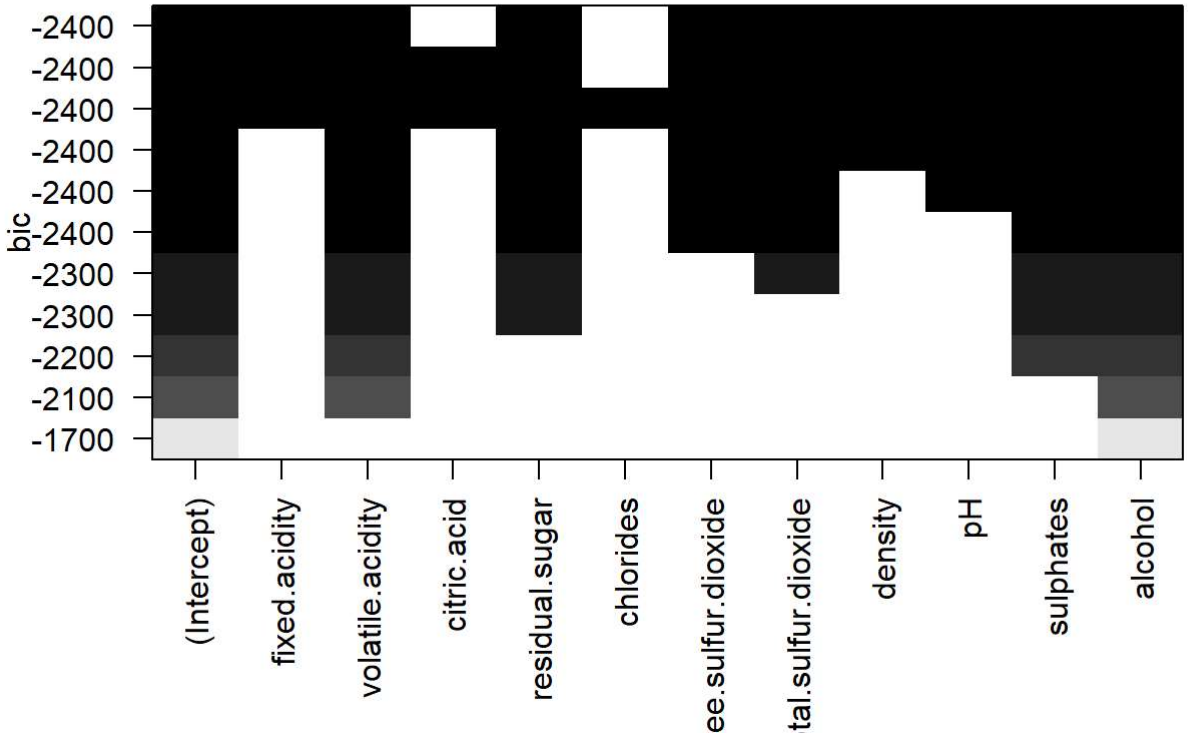
```
plot(regfit.full, scale = "adjr2")
```




```
plot(regfit.full, scale = "Cp")
```




```
plot(regfit.full, scale = "bic")
```



Choosing among models using the validation-set approach and cross-validation

Cross validation

```
set.seed(1)
train <- sample(c(TRUE, FALSE), nrow(wine_data), replace = TRUE)
test <- (!train)
```

```
regfit.best <- regsubsets(quality ~ ., data = wine_data[train, ], nvmax = 11)
test.mat <- model.matrix(quality ~ ., data = wine_data[test, ])
```

```
val.errors <- rep(0, 11)


for (i in 1:11) {
    coefi <- coef(regfit.best, id = i)
    pred <- test.mat[, names(coefi)] %*% coefi
    val.errors[i] <- mean((wine_data$quality[test] - pred)^2)

}
```

```
val.errors
```

```
##  [1] 0.4148673 0.3828393 0.3746997 0.3705519 0.3691098 0.3655426 0.3650684
##  [8] 0.3639048 0.3616509 0.3616630 0.3618294
```

```
which.min(val.errors)
```

```
## [1] 9
```

We find that the best model is on that contains 10 variables. Therefore we choose to use this model in our analysis.

```
predict.regsubsets <- function(object, newdata, id , ...) {
form <- as.formula(object$call[[2]])
mat <- model.matrix(form, newdata )
coefi <- coef(object, id = id)
xvars <- names (coefi)
mat[, xvars ] %*% coefi
}
```

```
regfit.best <- regsubsets(quality ~ ., data = wine_data, nvmax = 11)
coef(regfit.best, 10)
```

```
##          (Intercept)        fixed.acidity     volatile.acidity
##         68.960350471          0.074094737         -1.204312818
##          citric.acid       residual.sugar  free.sulfur.dioxide
##         -0.132337321          0.045462743          0.005868213
## total.sulfur.dioxide              density                   pH
##         -0.002382853        -68.228268139          0.504158979
##            sulphates              alcohol
##          0.883924623          0.230180918
```

Now try to choose amongst the models of different sizes using cross-validation (k-fold) 10-folds

Must perform best subset selection with each of the k training sets.

```
k <- 5
n <- nrow(wine_data)
set.seed(1)
folds <- sample(rep(1:k, length = n))
cv.errors <- matrix(NA, k, 11, dimnames = list(NULL, paste(1:11)))
```

```
for (j in 1:k) {
    best.fit <- regsubsets(quality ~ ., data = wine_data[folds != j, ], nvmax = 11)

    for (i in 1:11) {
        pred <- predict(best.fit, wine_data[folds == j, ], id = i)
        cv.errors[j, i] <- mean((wine_data$quality[folds == j] - pred)^2)
    }
}
```

```
mean.cv.errors <- apply(cv.errors, 2, mean)
mean.cv.errors
```

```
##         1         2         3         4         5         6         7         8
## 0.4136115 0.3841631 0.3771888 0.3727125 0.3712452 0.3674260 0.3677570 0.3667030
##         9        10        11
## 0.3648552 0.3649498 0.3649542
```

```
min(mean.cv.errors)
```

```
## [1] 0.3648552
```

We see that cross validation selects a 9 variable model.

We now perform best subset selection on the full data set in order to obtain this model

```
reg.best <- regsubsets(quality ~ ., data = wine_data, nvmax = 11)
coef(reg.best, 9)
```

```
##          (Intercept)        fixed.acidity     volatile.acidity
##         70.003827333          0.069246882         -1.153656143
##       residual.sugar  free.sulfur.dioxide total.sulfur.dioxide
##          0.045698051          0.005878583         -0.002447426
##              density                   pH            sulphates
##        -69.311834099          0.520147565          0.874475435
##              alcohol
##          0.227325902
```

Ridge-regression and the lasso

Ridge-regression

```
set.seed(1)
x <- model.matrix(quality ~ ., wine_data)[, -1]
y <- wine_data$quality
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

```
library(glmnet)
```

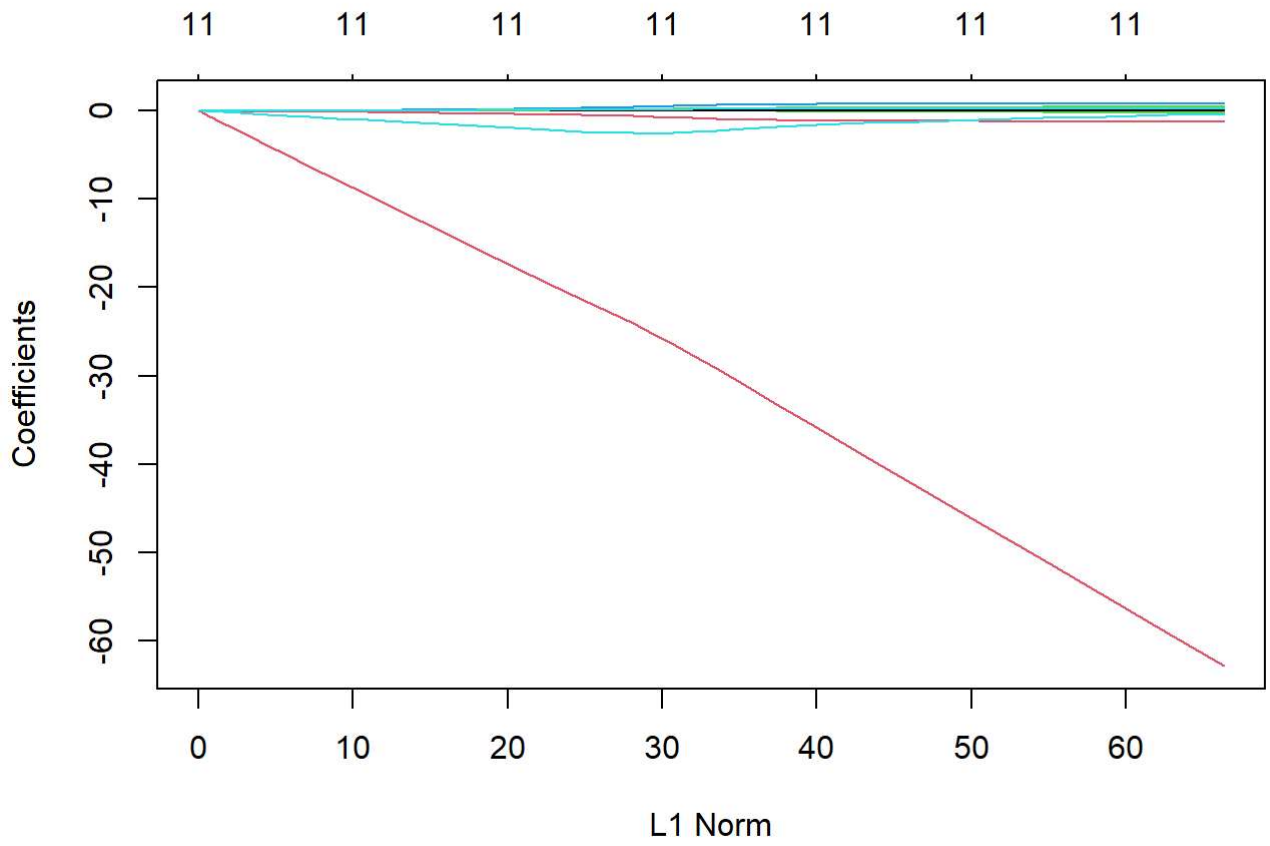
```
## Warning: package 'glmnet' was built under R version 4.2.2
```

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

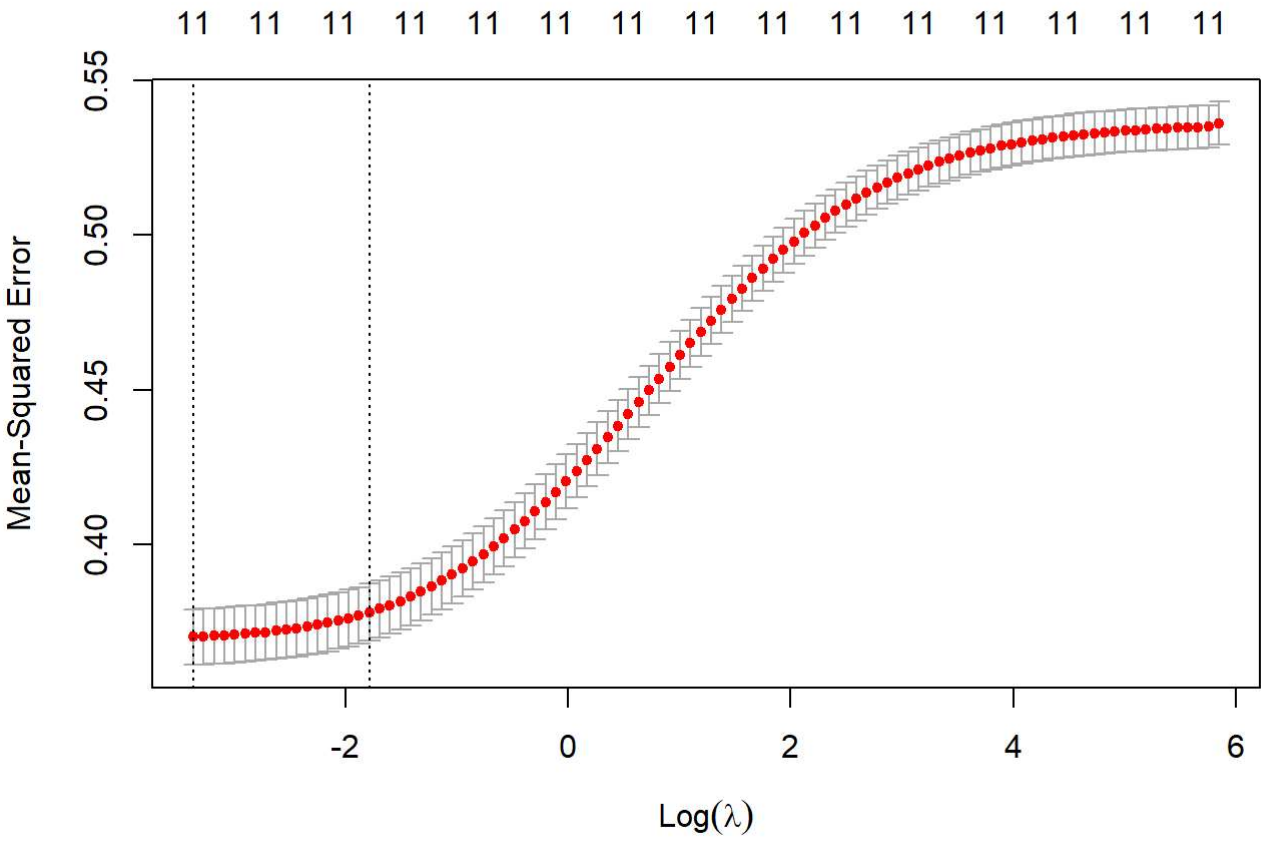
```
## Loaded glmnet 4.1-6
```

```
grid <- 10^seq(10, -2, length = 100)
```

```
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0, lambda = grid)
plot(ridge.mod)
```



```
set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
lasso.pred <- predict(ridge.mod, s = bestlam, newx = x[test, ])
mean((lasso.pred - y.test)^2)
```

```
## [1] 0.3605988
```

```
out <- glmnet(x, y, alpha = 0, lambda = grid)
ridge.coef <- predict(out, type = "coefficients", s = bestlam)[1:12, ]
ridge.coef
```

```
##          (Intercept)        fixed.acidity     volatile.acidity
##         40.934526532          0.043956261         -1.134589970
##          citric.acid       residual.sugar            chlorides
##         -0.062270550          0.030537484         -1.173505845
##  free.sulfur.dioxide total.sulfur.dioxide              density
##          0.005148968         -0.002015011        -39.330410551
##                   pH            sulphates              alcohol
##          0.345771173          0.785362246          0.243205550
```

```
ridge.coef[ridge.coef != 0]
```

```
##          (Intercept)        fixed.acidity     volatile.acidity
##         40.934526532          0.043956261         -1.134589970
##          citric.acid       residual.sugar            chlorides
##         -0.062270550          0.030537484         -1.173505845
##  free.sulfur.dioxide total.sulfur.dioxide              density
##          0.005148968         -0.002015011        -39.330410551
##                   pH            sulphates              alcohol
##          0.345771173          0.785362246          0.243205550
```

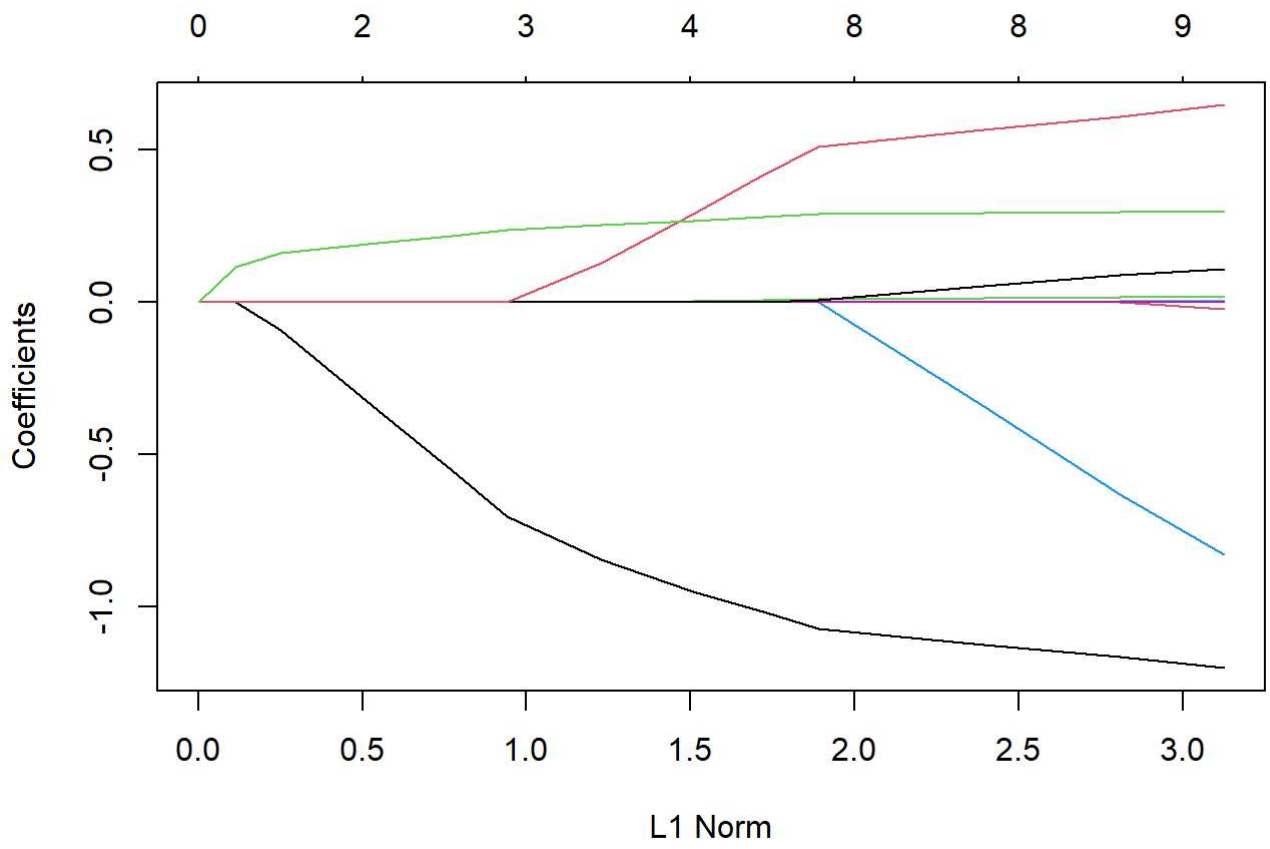
The lasso

```
set.seed(1)
x <- model.matrix(quality ~ ., wine_data)[, -1]
y <- wine_data$quality
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

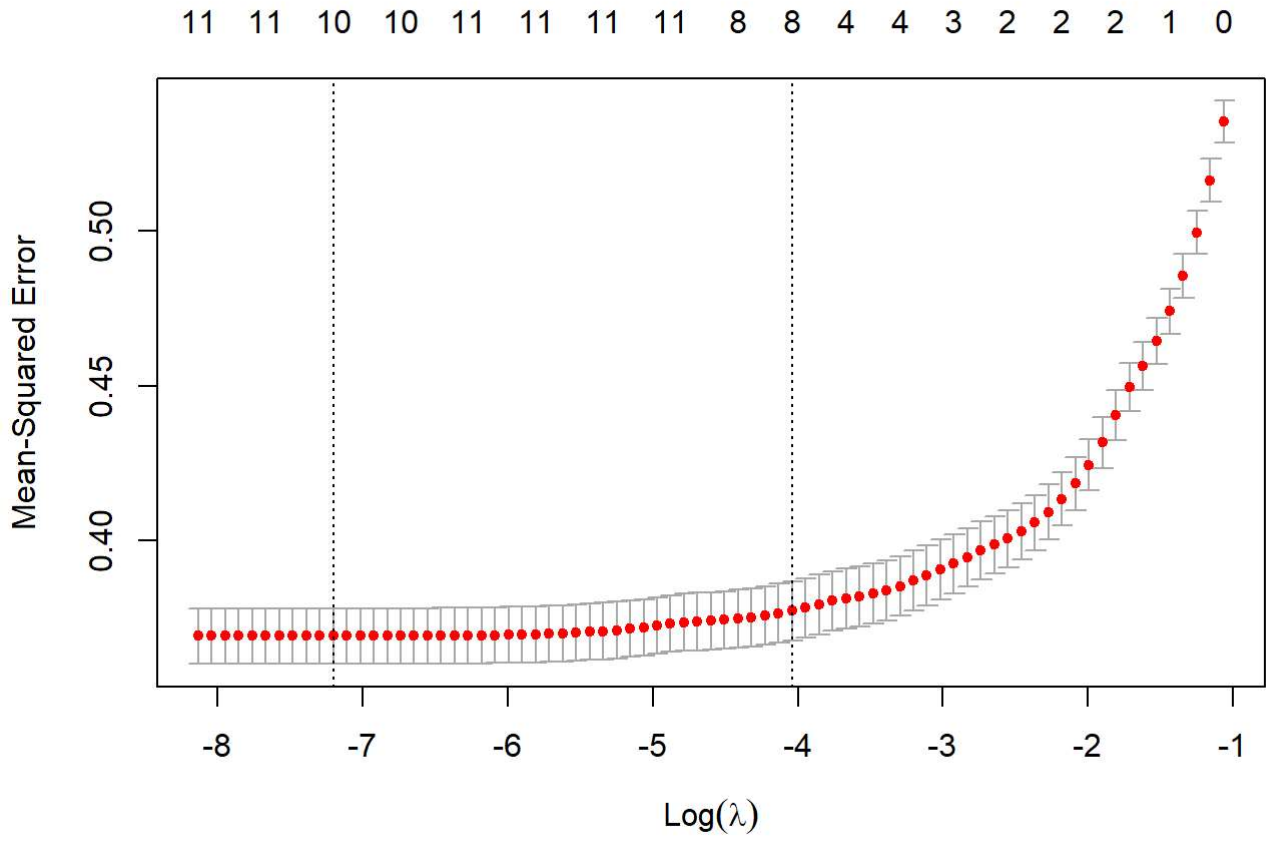
```
library(glmnet)
grid <- 10^seq(10, -2, length = 100)
```

```
lasso.mod <- glmnet(x[train, ], y[train], alpha = 1, lambda = grid)
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 1)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test, ])
mean((lasso.pred - y.test)^2)
```

```
## [1] 0.3631217
```

```
out <- glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:12, ]
lasso.coef
```

```
##          (Intercept)        fixed.acidity     volatile.acidity
##          2.474306957          0.000000000         -1.189634457
##          citric.acid       residual.sugar            chlorides
##          0.000000000          0.014669290         -0.754664550
##  free.sulfur.dioxide total.sulfur.dioxide              density
##          0.003832357         -0.001368393          0.000000000
##                   pH            sulphates              alcohol
##          0.093997115          0.634427372          0.297747749
```

```
lasso.coef[lasso.coef != 0]
```

```
##          (Intercept)     volatile.acidity       residual.sugar
##          2.474306957         -1.189634457          0.014669290
##            chlorides  free.sulfur.dioxide total.sulfur.dioxide
##         -0.754664550          0.003832357         -0.001368393
##                   pH            sulphates              alcohol
##          0.093997115          0.634427372          0.297747749
```

```
lasso.coef[lasso.coef == 0]
```

```
## fixed.acidity   citric.acid       density
##             0             0             0
```

Princpal components regression

```
set.seed(1)
x <- model.matrix(quality ~ ., wine_data)[, -1]
y <- wine_data$quality
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.2.2
```

```
##
## Attaching package: 'pls'
```

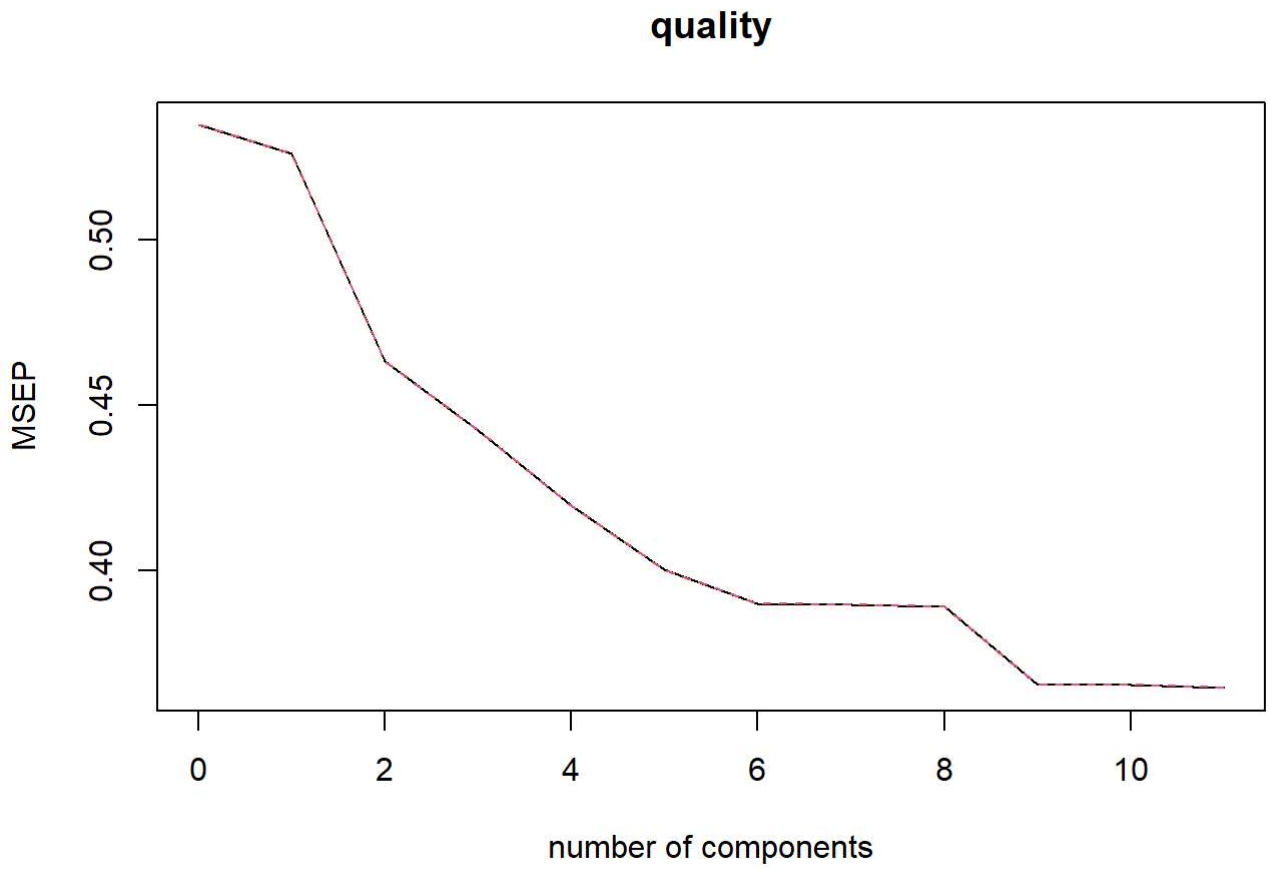
```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(2)
pcr.fit <- pcr(quality ~ ., data = wine_data, scale = TRUE, validation = "CV")
```

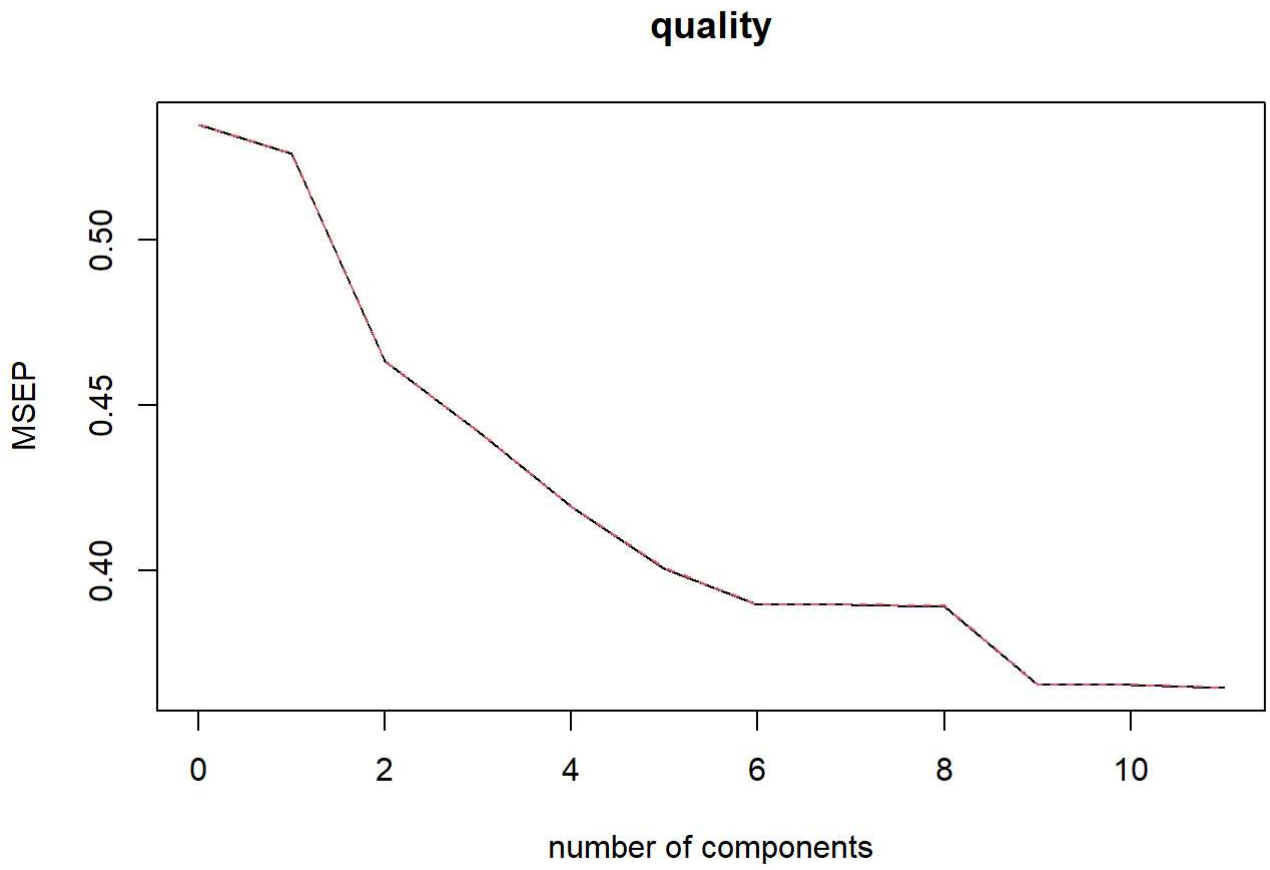
```
summary(pcr.fit)
```

```
## Data:    X dimension: 6497 11
##  Y dimension: 6497 1
## Fit method: svdpc
## Number of components considered: 11
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.7312   0.7252   0.6807   0.6651   0.6478   0.6327   0.6246
## adjCV       0.7312   0.7252   0.6807   0.6651   0.6477   0.6327   0.6245
##        7 comps  8 comps  9 comps  10 comps  11 comps
## CV      0.6243   0.6238   0.6045    0.6047    0.6038
## adjCV   0.6242   0.6238   0.6045    0.6046    0.6038
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         30.154    53.23    67.55    76.24    81.72    87.12    91.63    94.88
## quality    1.628    13.39    17.34    21.61    25.23    27.18    27.28    27.41
##          9 comps  10 comps  11 comps
## X          97.81     99.71    100.00
## quality    31.86     31.86     32.07
```

```
validationplot(pcr.fit, val.type = "MSEP")
```



```
set.seed(1)
pcr.fit <- pcr(quality ~ ., data = wine_data, scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```




```
pcr.pred <- predict(pcr.fit, x[test, ], ncomp = 5)
mean((pcr.pred - y.test)^2)
```

```
## [1] 0.3942515
```

```
pcr.fit <- pcr(y ~ x, scale = TRUE, ncomp = 5)
summary(pcr.fit)
```

```
## Data:    X dimension: 6497 11
##  Y dimension: 6497 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps
## X   30.154    53.23    67.55    76.24    81.72
## y    1.628    13.39    17.34    21.61    25.23
```

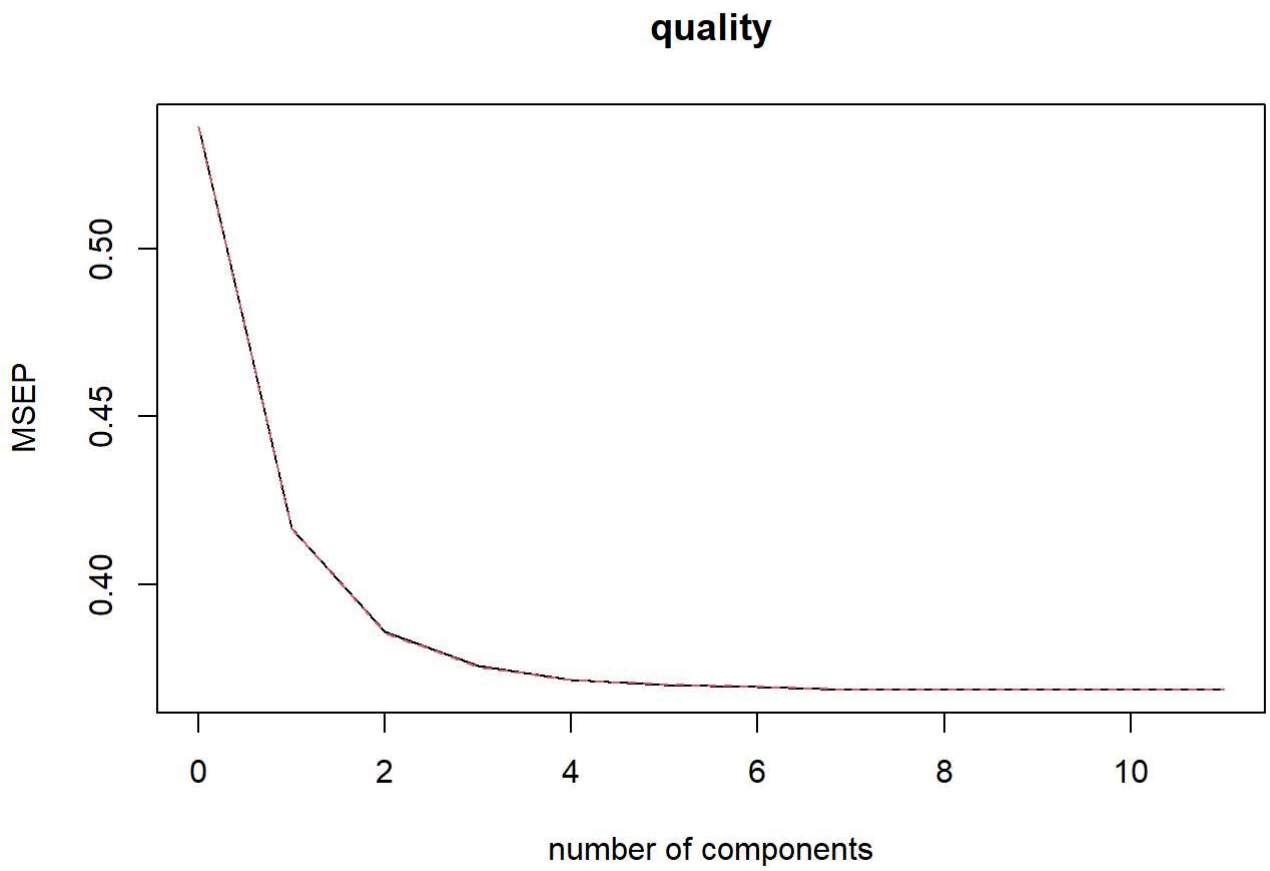
Partial least squares

```
set.seed(1)
x <- model.matrix(quality ~ ., wine_data)[, -1]
y <- wine_data$quality
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

```
pls.fit <- plsr(quality ~ ., data = wine_data, subset = train, scale = TRUE, validation = "CV")
summary(pls.fit)
```

```
## Data:    X dimension: 3248 11
##  Y dimension: 3248 1
## Fit method: kernelpls
## Number of components considered: 11
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.7323   0.6456   0.6211   0.6130   0.6096   0.6083   0.6080
## adjCV       0.7323   0.6456   0.6211   0.6129   0.6095   0.6082   0.6079
##        7 comps  8 comps  9 comps  10 comps  11 comps
## CV      0.6072   0.6072   0.6072    0.6072    0.6072
## adjCV   0.6071   0.6071   0.6071    0.6071    0.6071
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X          22.67    38.37    58.38    70.97    78.71    84.36    84.86    88.20
## quality    22.46    28.36    30.26    31.07    31.37    31.44    31.66    31.66
##          9 comps  10 comps  11 comps
## X          91.91     96.16    100.00
## quality    31.66     31.66     31.66
```

```
validationplot(pls.fit, val.type = "MSEP")
```




```
pls.pred <- predict(pls.fit, x[test, ], ncomp = 1)
mean((pls.pred - y.test)^2)
```

```
## [1] 0.4068395
```

```
pls.fit <- plsr(quality ~ ., data = wine_data, scale = TRUE, ncomp = 1)
summary(pls.fit)
```

```
## Data:    X dimension: 6497 11
##  Y dimension: 6497 1
## Fit method: kernelpls
## Number of components considered: 1
## TRAINING: % variance explained
##          1 comps
## X          22.53
## quality    23.15
```