Exploring Linear Models: Variable Selection Criteria, Ridge/LASSO regressions, and Significance Probabilities

Dylan Cicero

2/17/2019

Introduction

In this analysis, I explore a dataset on wine charactaristics and quality for 1599 samples of wine from the north of Portugal. The dataset includes eleven measured physicochemical charactaristics for each sample of wine (i.e. *fixed acidity*, *pH*, *residual sugar*, *alcohol content*, etc) and a more subjective "*quality*" score assigned by expert judges. Here, my premise is that the various physicochemical charactaristics of wine can be used to predict wine quality.

The data is first segmented into a training set and a test set. After the training data is transformed to better approximate normality, various model selection methods are trialed- each strategically seeks to include a set of variables that accurately predict wine quality while minimimizing "noise". (Every additional parameter introduces inherent randomness into the model, so that if a parameter presents only similar information as other included parameters, it might bias the model away from the true relationship). The methods are then trialed on the test set and evaluated. Amid the analysis, a simulation experiment is conducted, which considers the likelihood of false statistical significance for model parameters.

Part 1

Read in data and segment into training set and test set

```r
x <- read.csv("https://tinyurl.com/tjruom9", sep=";")
colnames(x)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

```r
dim(x)
```

```
## [1] 1599   12
```

```r
set.seed(1)
test.which <- sample(1:nrow(x), floor(nrow(x)/10))
test <- x[test.which, ]
dim(test)
```

```
## [1] 159  12
```

```r
train <- x[-test.which, ]
dim(train)
```

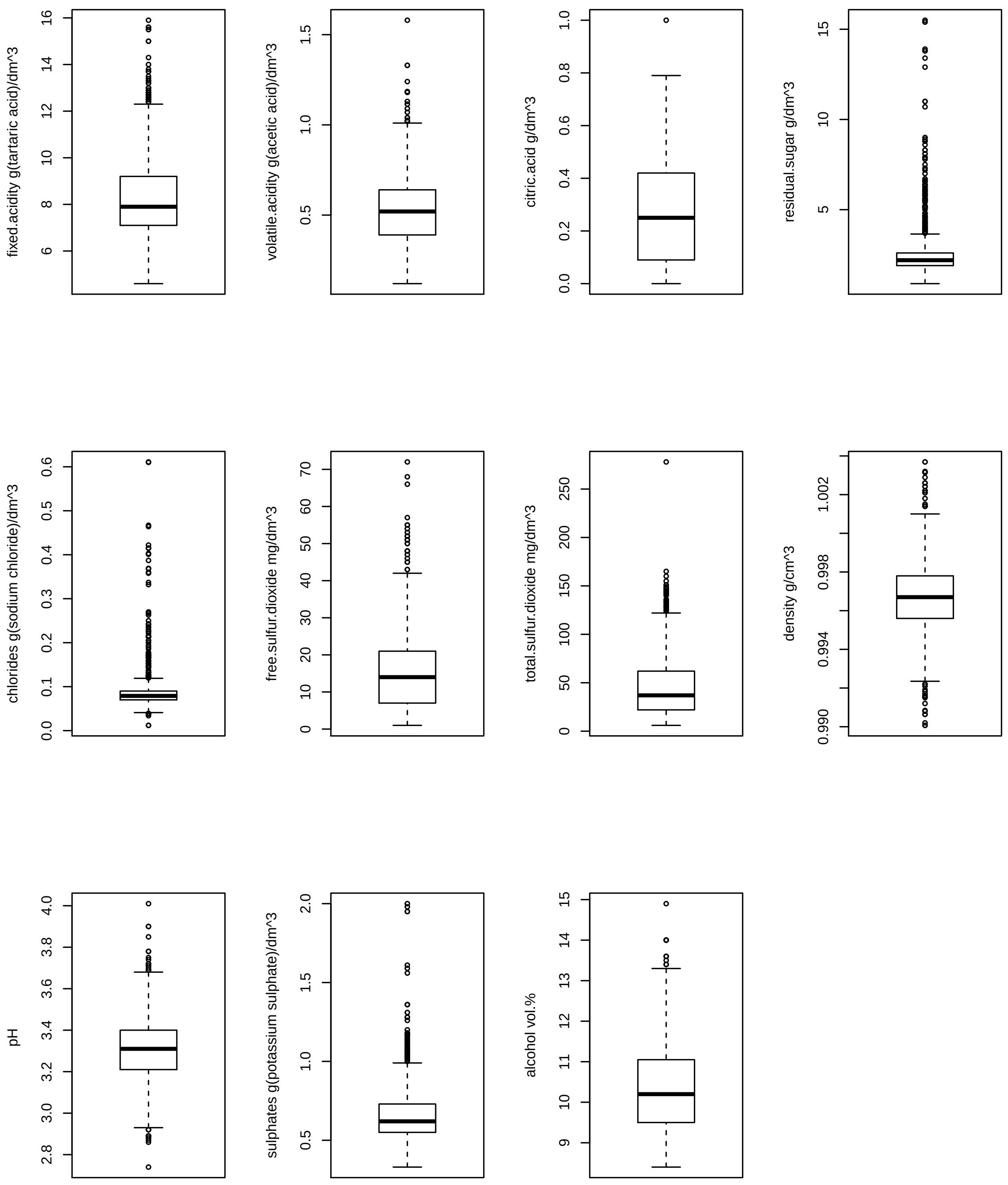
```
## [1] 1440   12
```

Draw a boxplot for each variable in the dataset. If it seems skewed, find a transformation that results in a more normal distribution. Create a dataframe that includes any transformed variables in place of the original ones.

```r
# ****************************************************************************
# PART 1

# take a look at boxplots without any transformations
par(mfrow = c(3,4), oma = c(0,0,2,0))
for (i in 1:11) {
  append = ""
  if (i == 1) {
    append = "g(tartaric acid)/dm^3"
  }
  if (i == 2) {
    append = "g(acetic acid)/dm^3"
  }
  if (i == 3 | i == 4) {
    append = "g/dm^3"
  }
  if (i == 5) {
    append = "g(sodium chloride)/dm^3"
  }
  if (i == 6 | i == 7) {
    append = "mg/dm^3"
  }
  if (i == 8) {
    append = "g/cm^3"
  }
  if (i == 10) {
    append = "g(potassium sulphate)/dm^3"
  }
  if (i == 11) {
    append = "vol.%"
  }
  boxplot(train[ ,i], ylab = paste(as.character(colnames(train)[i]), append))
}
mtext("Boxplots of 11 Variables Before Transformation", outer = TRUE, cex = 1.5)
```

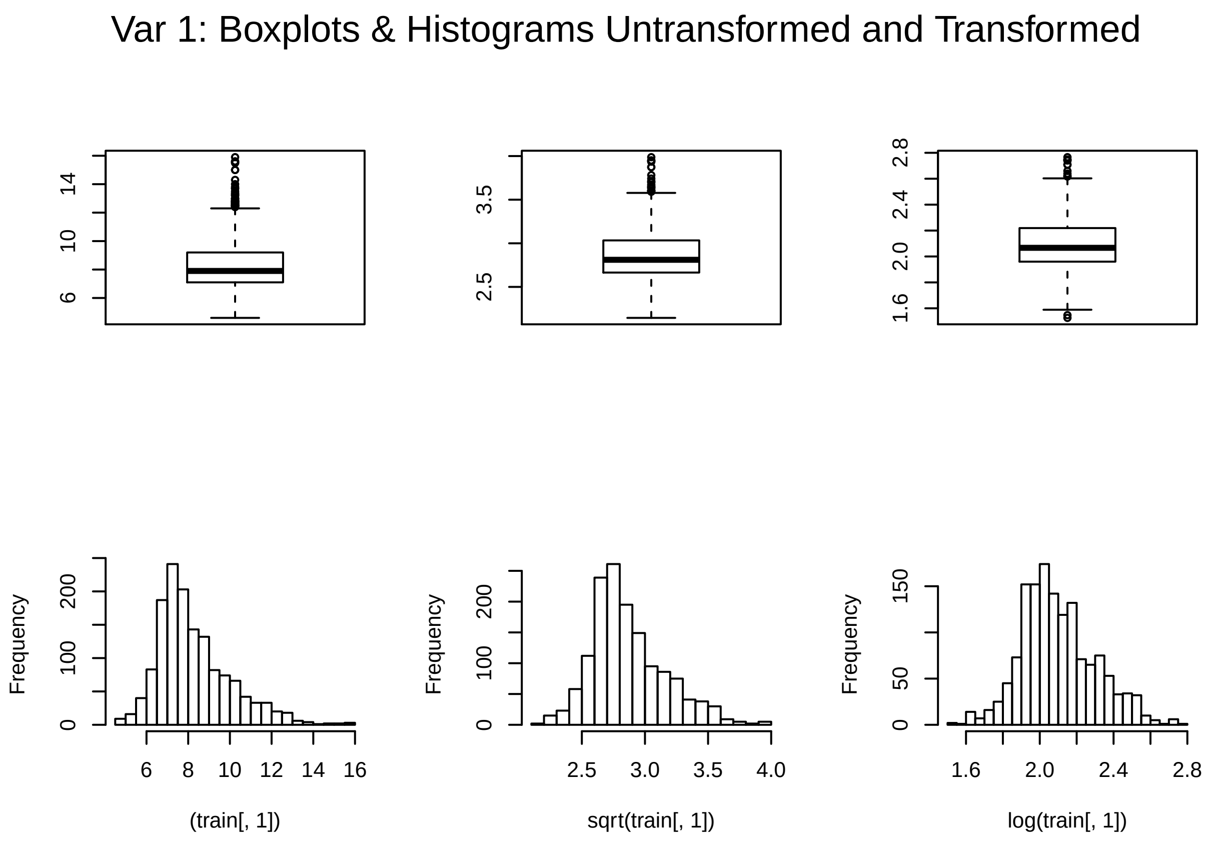
Boxplots of 11 Variables Before Transformation



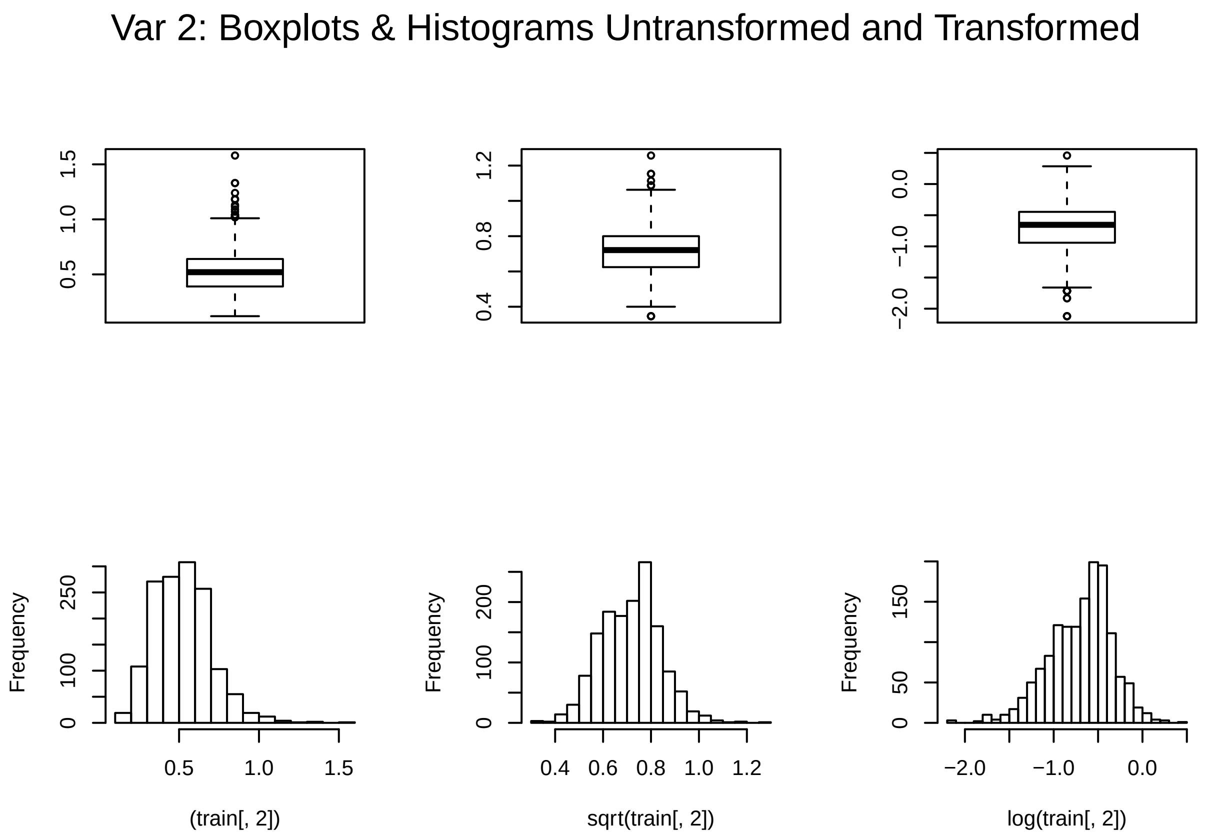
Most variables display some positive skewness…

Go through each variable, comparing boxplots/histograms for the original data and transformations sensibly applied to right-skewed data. For each, select the transformation that outputs the most normall distributed data. This is important because a lot of techniques to follow are based on gaussian data… the more normal the data, the more legitimate the statistical methods.

```r
# 1
par(mfrow = c(2,3), oma = c(0,0,2,0))
boxplot((train[ ,1]))
boxplot(sqrt(train[ ,1]))
boxplot(log(train[ ,1])) # wins
hist((train[ ,1]), breaks = 20, main = NULL)
hist(sqrt(train[ ,1]), breaks = 20, main = NULL)
hist(log(train[ ,1]), breaks = 20, main = NULL) # wins
mtext("Var 1: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



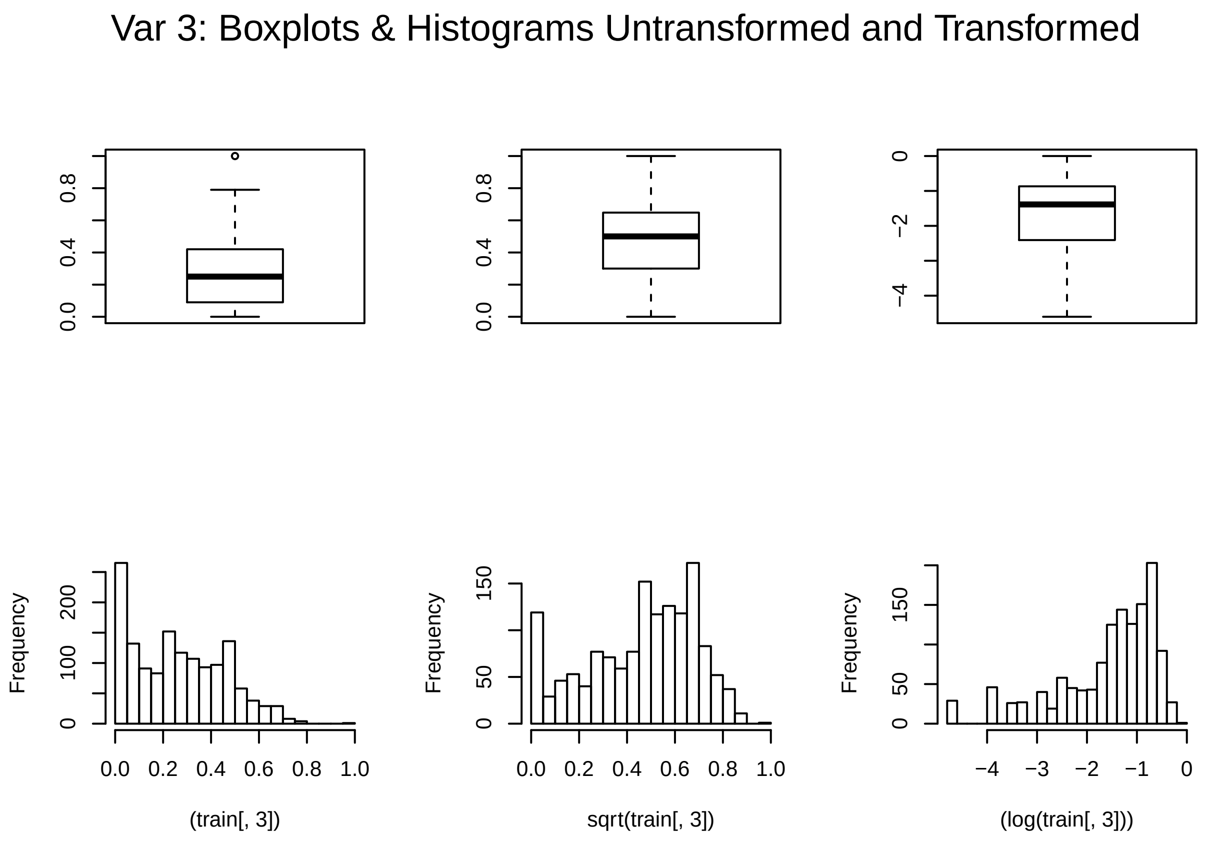
```r
# 2
par(mfrow = c(2,3), oma = c(0,0,2,0))
boxplot((train[ ,2]))
boxplot(sqrt(train[ ,2]))
boxplot(log(train[ ,2])) # wins
hist((train[ ,2]), breaks = 20, main = NULL)
hist(sqrt(train[ ,2]), breaks = 20, main = NULL)
hist(log(train[ ,2]), breaks = 20, main = NULL) # wins
mtext("Var 2: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



```
# 3
par(mfrow = c(2,3), oma = c(0,0,2,0))
boxplot((train[ ,3]))
boxplot(sqrt(train[ ,3])) # wins
boxplot(log(train[ ,3]))
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out =
## z$out[z$group == : Outlier (-Inf) in boxplot 1 is not drawn
```

```
hist((train[ ,3]), breaks = 20, main = NULL)
hist(sqrt(train[ ,3]), breaks = 20, main = NULL) # wins
hist((log(train[ ,3])), breaks = 20, main = NULL)
mtext("Var 3: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



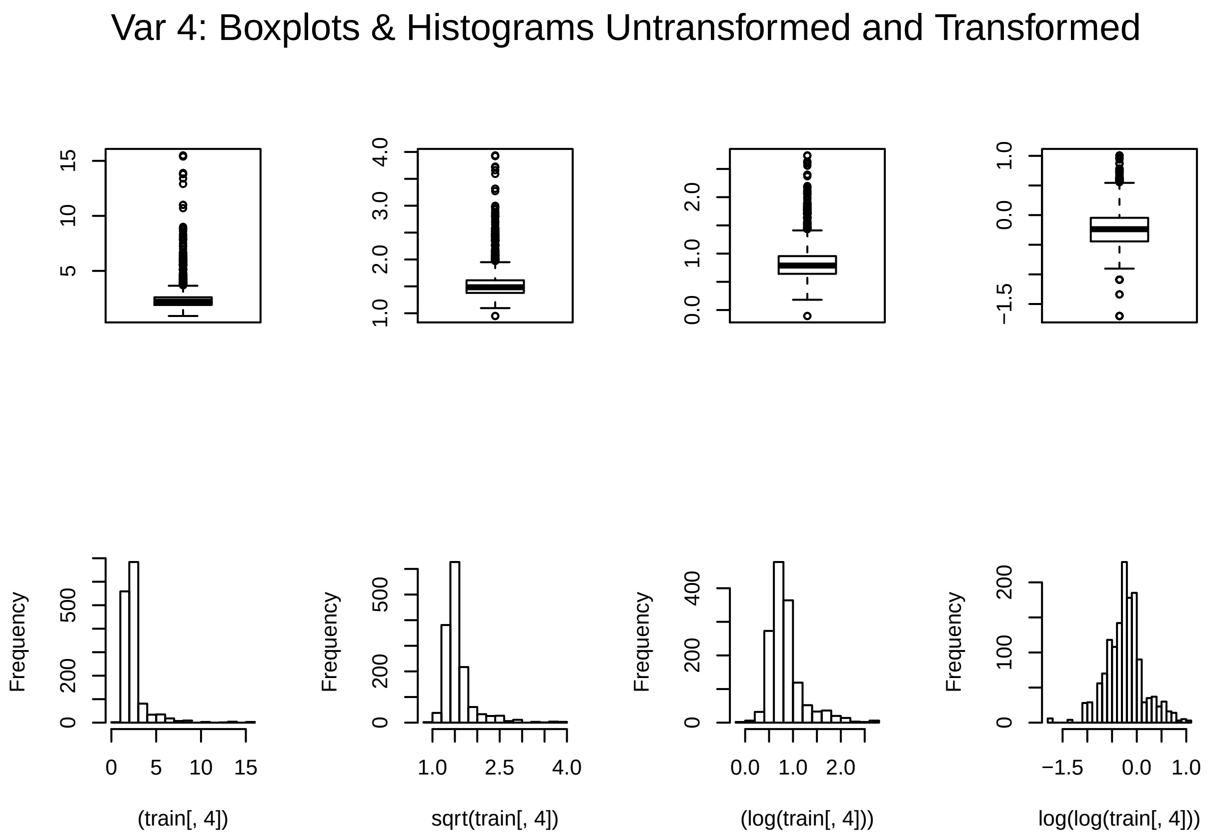
```r
# 4
par(mfrow = c(2,4), oma = c(0,0,2,0))
boxplot((train[ ,4]))
boxplot(sqrt(train[ ,4]))
boxplot((log(train[ ,4])))
boxplot(log(log(train[ ,4]))) # wins
```

```
## Warning in log(log(train[, 4])): NaNs produced
```

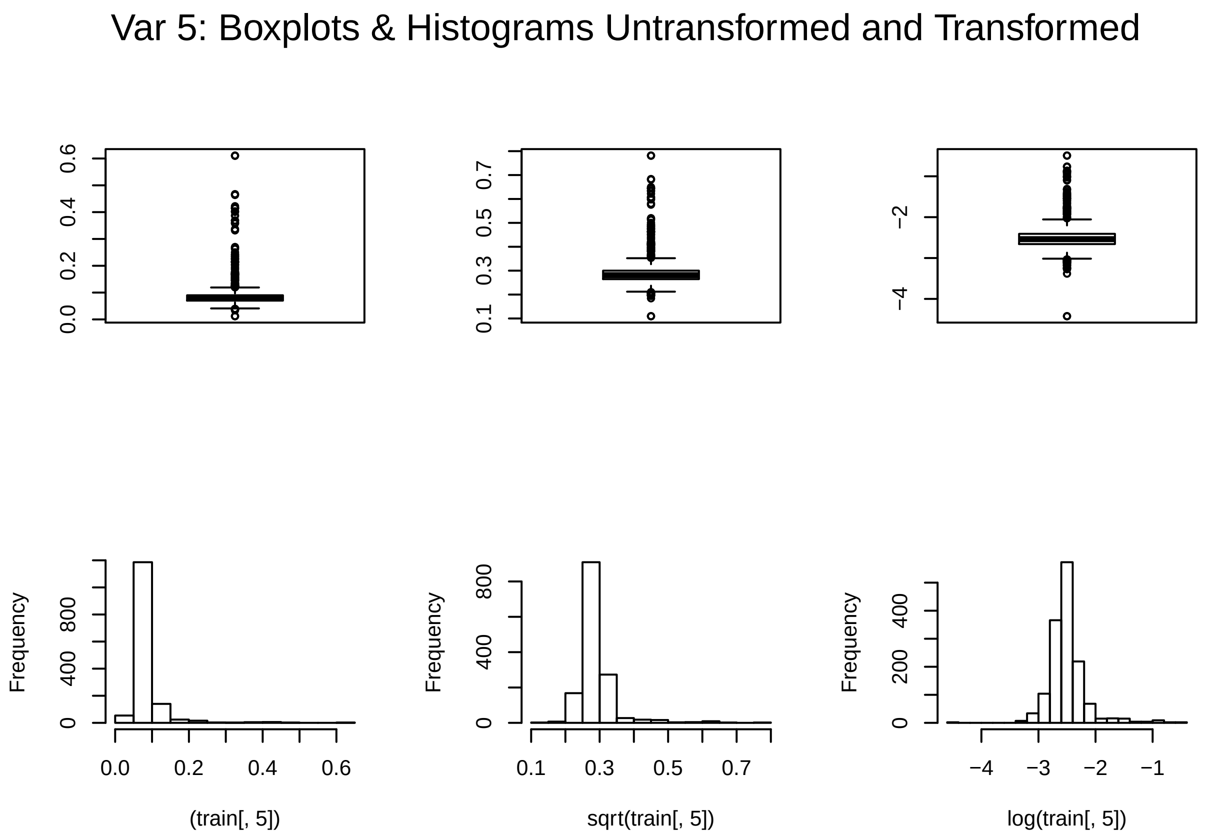
```r
hist((train[ ,4]), breaks = 20, main = NULL)
hist(sqrt(train[ ,4]), breaks = 20, main = NULL)
hist((log(train[ ,4])), breaks = 20, main = NULL)
hist(log(log(train[ ,4])), breaks = 20, main = NULL) # wins
```

```
## Warning in log(log(train[, 4])): NaNs produced
```

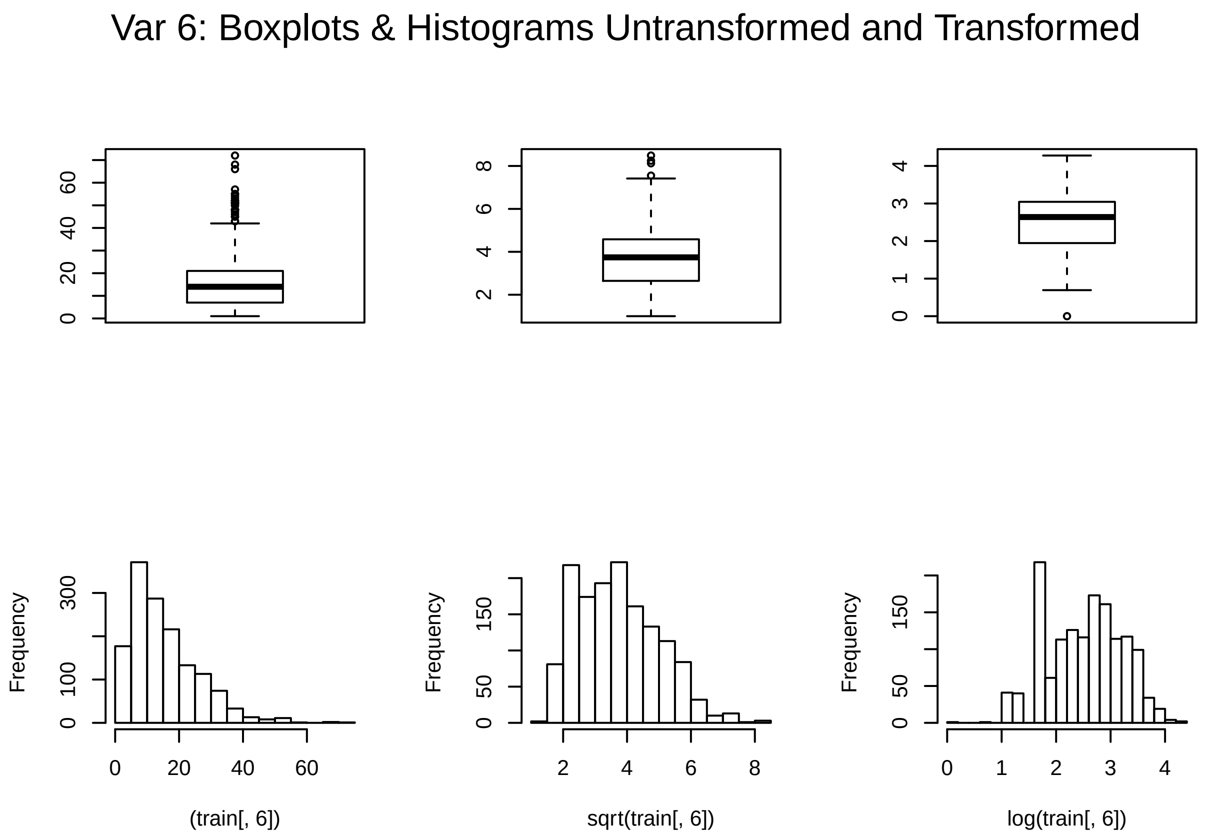
```r
mtext("Var 4: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



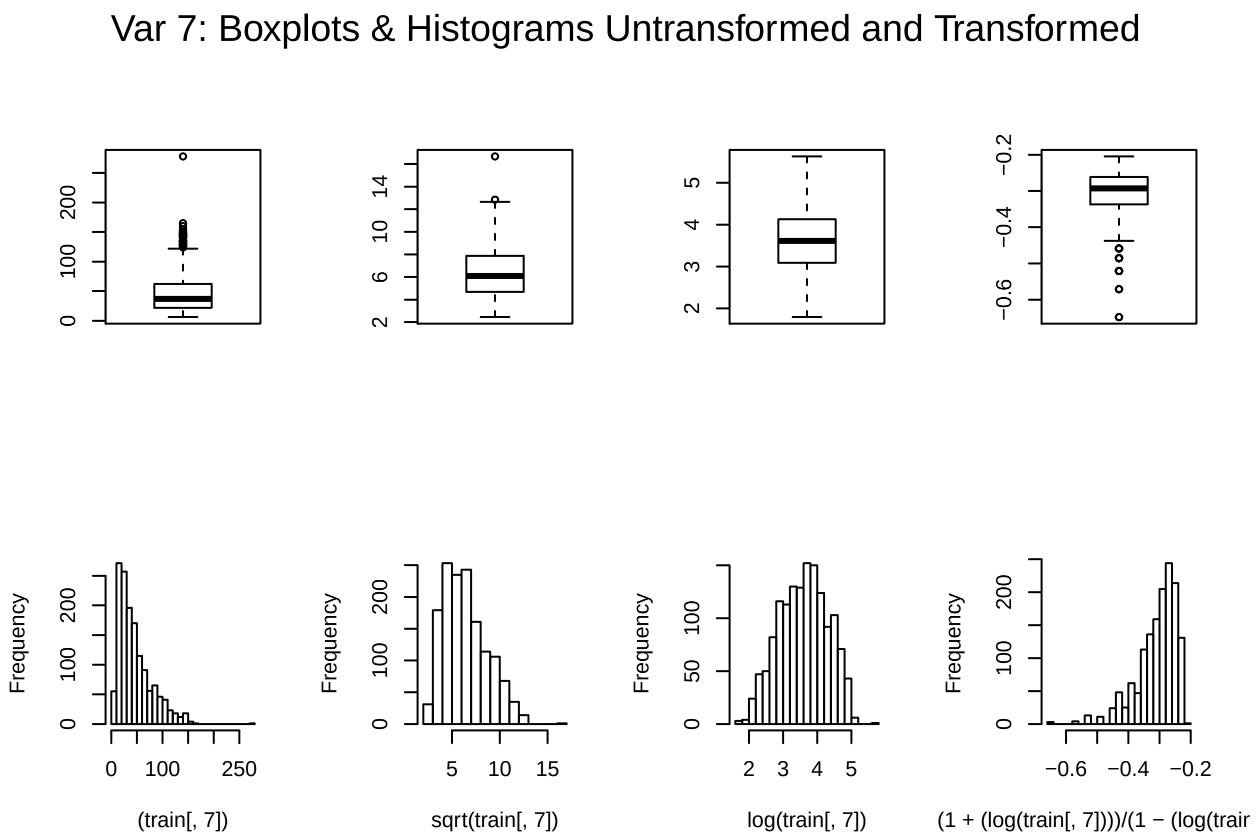
```r
# 5
par(mfrow = c(2,3), oma = c(0,0,2,0))
boxplot((train[ ,5]))
boxplot(sqrt(train[ ,5]))
boxplot(log(train[ ,5])) # wins
hist((train[ ,5]), breaks = 20, main = NULL)
hist(sqrt(train[ ,5]), breaks = 20, main = NULL)
hist(log(train[ ,5]), breaks = 20, main = NULL) # wins
mtext("Var 5: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



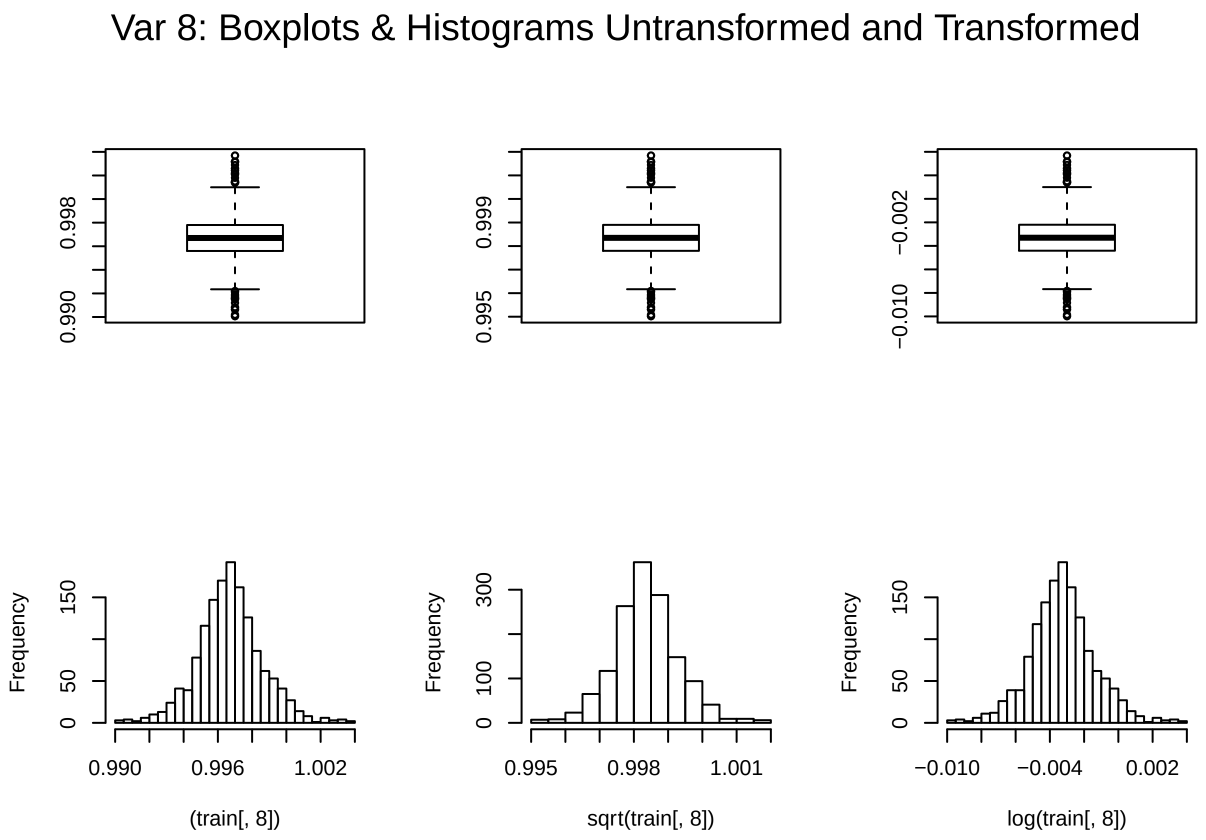
```r
# 6
par(mfrow = c(2,3), oma = c(0,0,2,0))
boxplot((train[ ,6]))
boxplot(sqrt(train[ ,6]))
boxplot(log(train[ ,6])) # wins
hist((train[ ,6]), breaks = 20, main = NULL)
hist(sqrt(train[ ,6]), breaks = 20, main = NULL)
hist(log(train[ ,6]), breaks = 20, main = NULL) # wins
mtext("Var 6: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



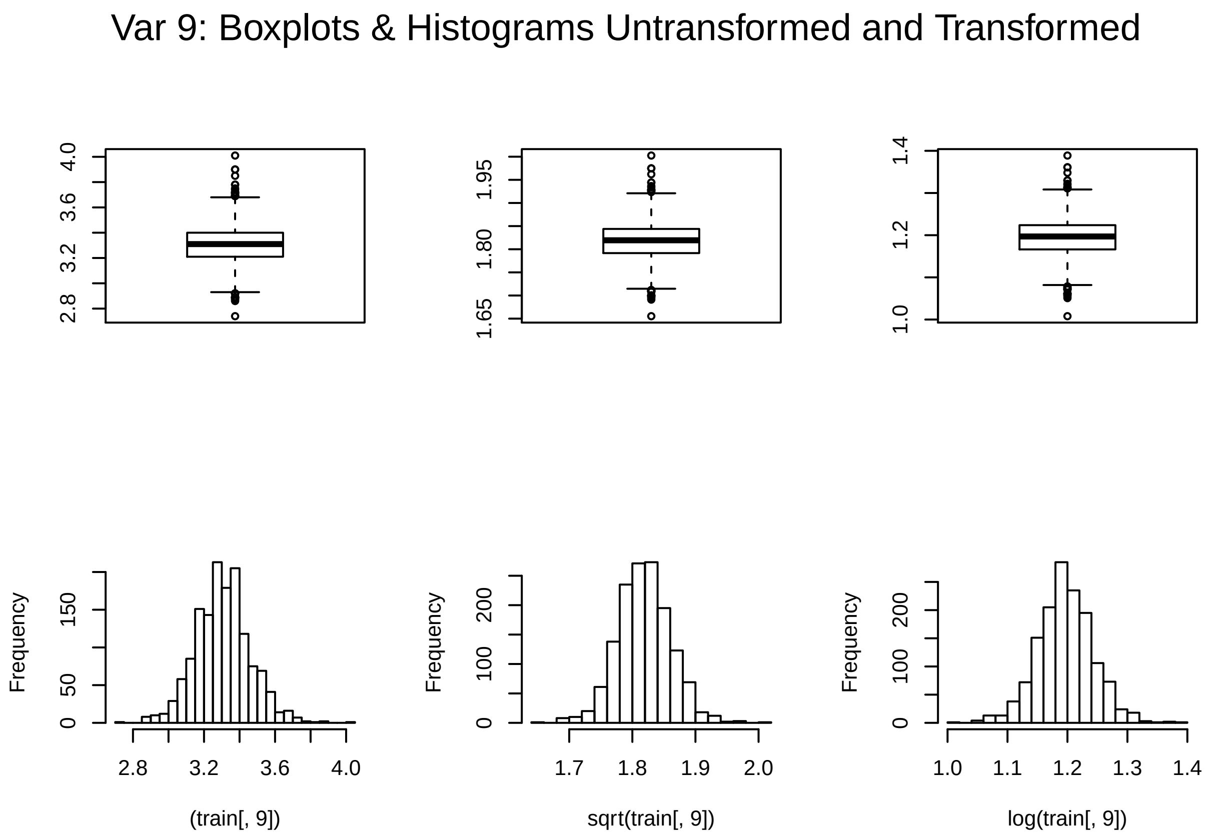
```
# 7
par(mfrow = c(2,4), oma = c(0,0,2,0))
boxplot((train[ ,7]))
boxplot(sqrt(train[ ,7]))
boxplot(log(train[ ,7])) # wins
boxplot(log(1 + (log(train[ ,7])))/(1 - (log(train[ ,7])))/2)
hist((train[ ,7]), breaks = 20, main = NULL)
hist(sqrt(train[ ,7]), breaks = 20, main = NULL)
hist(log(train[ ,7]), breaks = 20, main = NULL) # wins
hist(log(1 + (log(train[ ,7])))/(1 - (log(train[ ,7])))/2, breaks = 20, main = NULL)
mtext("Var 7: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



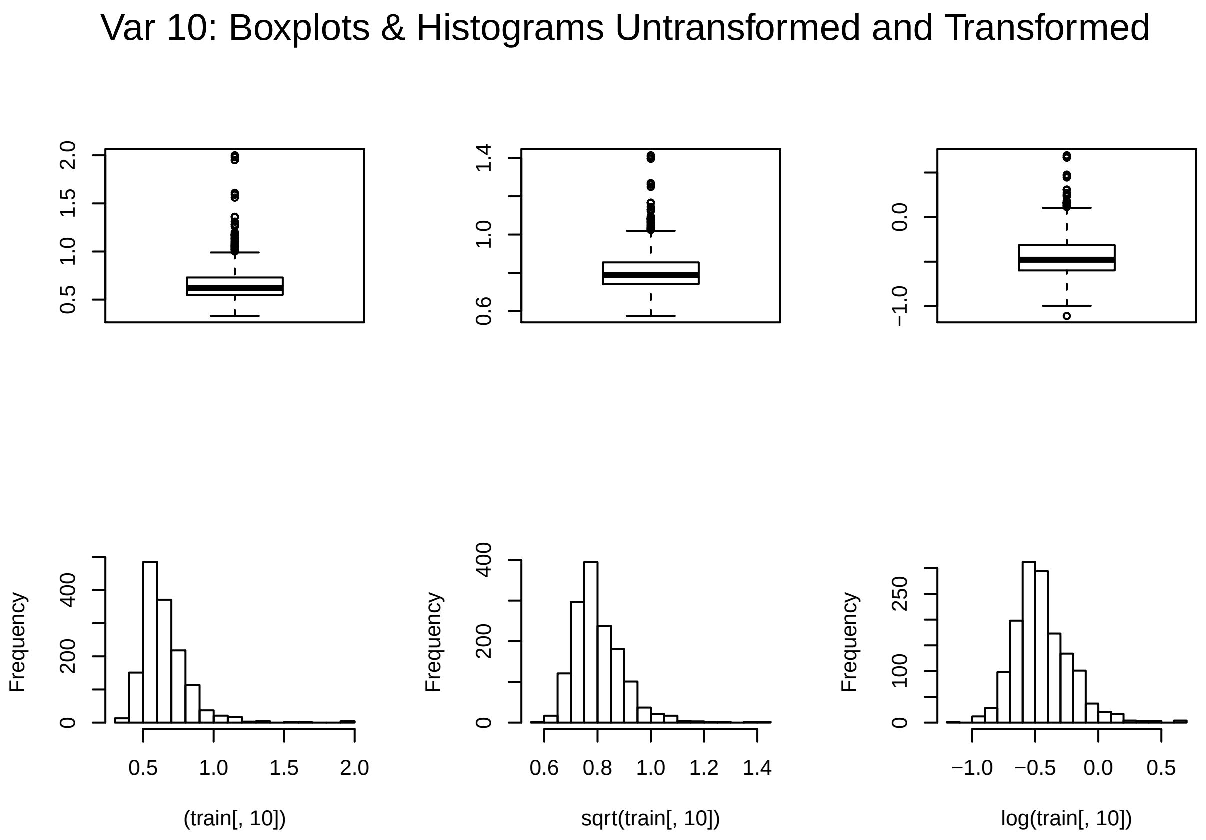
```r
# 8
par(mfrow = c(2,3), oma = c(0,0,2,0))
boxplot((train[ ,8])) # wins
boxplot(sqrt(train[ ,8]))
boxplot(log(train[ ,8]))
hist((train[ ,8]), breaks = 20, main = NULL) # wins
hist(sqrt(train[ ,8]), breaks = 20, main = NULL)
hist(log(train[ ,8]), breaks = 20, main = NULL)
mtext("Var 8: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



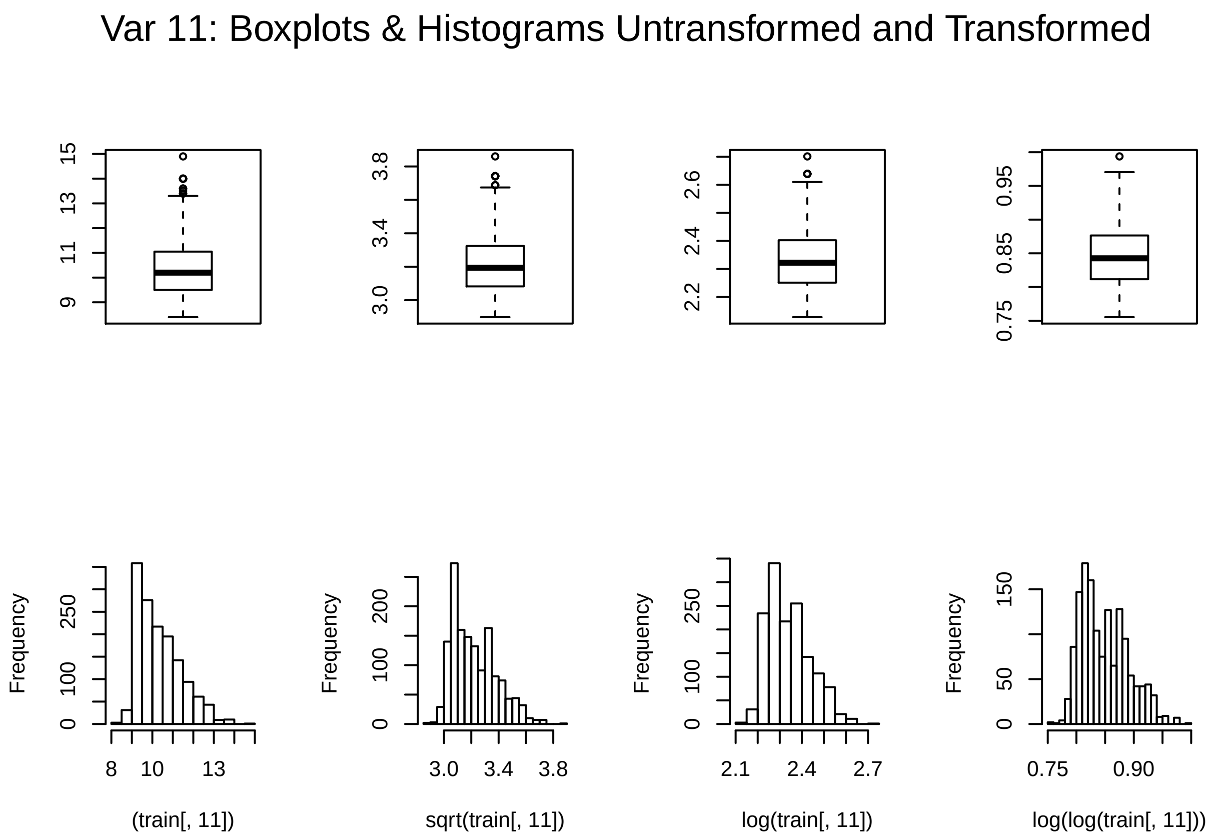
```r
# 9
par(mfrow = c(2,3), oma = c(0,0,2,0))
boxplot((train[ ,9]))
boxplot(sqrt(train[ ,9]))
boxplot(log(train[ ,9])) # wins
hist((train[ ,9]), breaks = 20, main = NULL)
hist(sqrt(train[ ,9]), breaks = 20, main = NULL)
hist(log(train[ ,9]), breaks = 20, main = NULL) # wins
mtext("Var 9: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



```r
# 10
par(mfrow = c(2,3), oma = c(0,0,2,0))
boxplot((train[ ,10]))
boxplot(sqrt(train[ ,10]))
boxplot(log(train[ ,10])) # wins
hist((train[ ,10]), breaks = 20, main = NULL)
hist(sqrt(train[ ,10]), breaks = 20, main = NULL)
hist(log(train[ ,10]), breaks = 20, main = NULL) # wins
mtext("Var 10: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



```
# 11
par(mfrow = c(2,4), oma = c(0,0,2,0))
boxplot((train[ ,11]))
boxplot(sqrt(train[ ,11]))
boxplot(log(train[ ,11])) # wins
boxplot(log(log(train[ ,11])))
hist((train[ ,11]), breaks = 20, main = NULL)
hist(sqrt(train[ ,11]), breaks = 20, main = NULL)
hist(log(train[ ,11]), breaks = 20, main = NULL) # wins
hist(log(log(train[ ,11])), breaks = 20, main = NULL)
mtext("Var 11: Boxplots & Histograms Untransformed and Transformed", outer = TRUE, cex = 1.2)
```



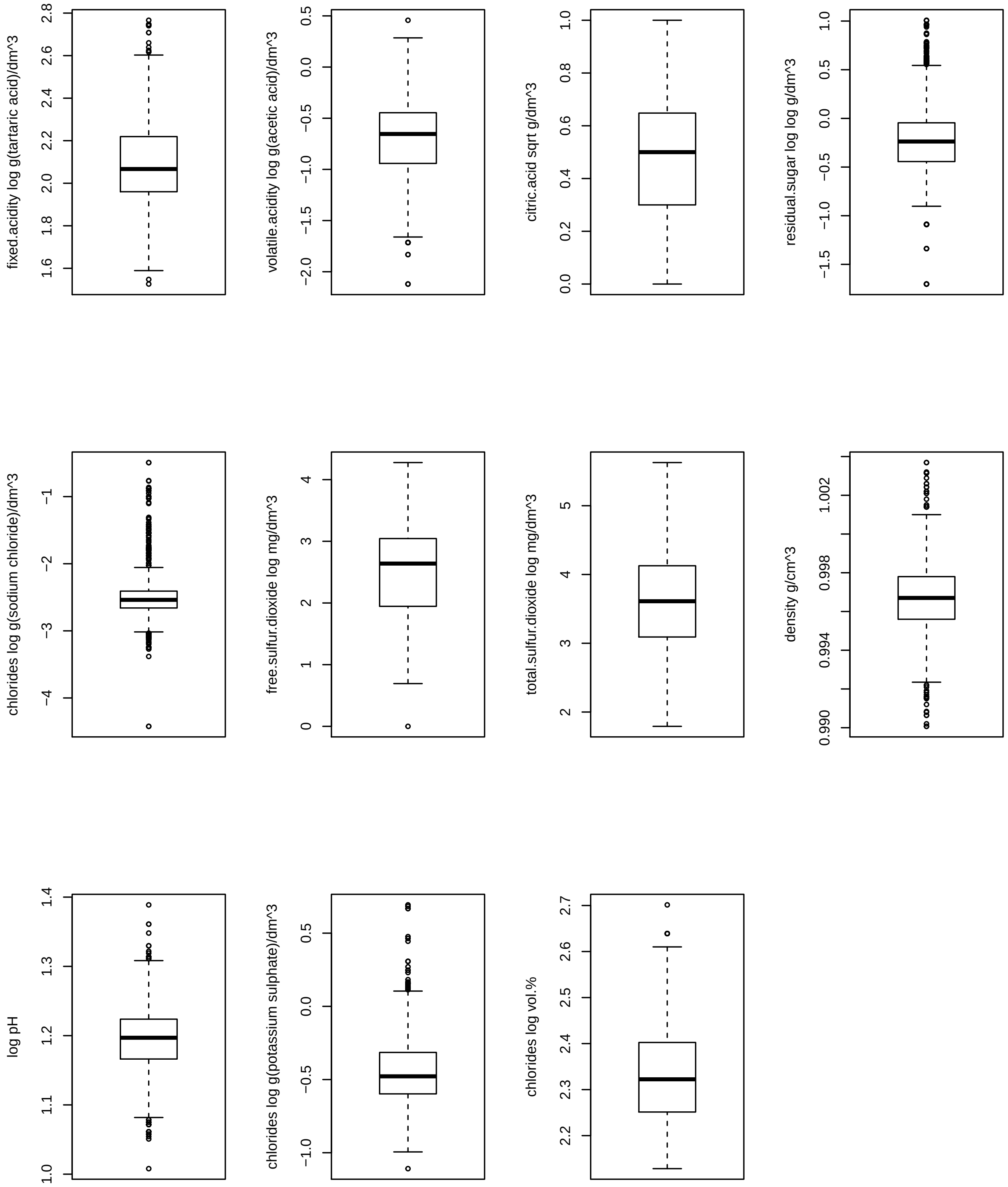
Revisit boxplots with transformations applied

```r
# Revisit boxplots with transformations applied
par(mfrow = c(3,4), oma = c(0,0,2,0))
boxplot(log(train[ ,1]), ylab = paste(as.character(colnames(train)[1]), "log g(tartaric acid)/dm^3"))
boxplot(log(train[ ,2]), ylab = paste(as.character(colnames(train)[2]), "log g(acetic acid)/dm^3"))
boxplot(sqrt(train[ ,3]), ylab = paste(as.character(colnames(train)[3]), "sqrt g/dm^3"))
boxplot(log(log(train[ ,4])), ylab = paste(as.character(colnames(train)[4]), "log log g/dm^3"))
```

```
## Warning in log(log(train[, 4])): NaNs produced
```

```r
boxplot(log(train[ ,5]), ylab = paste(as.character(colnames(train)[5]), "log g(sodium chloride)/dm^3"))
boxplot(log(train[ ,6]), ylab = paste(as.character(colnames(train)[6]), "log mg/dm^3"))
boxplot(log(train[ ,7]), ylab = paste(as.character(colnames(train)[7]), "log mg/dm^3"))
boxplot((train[ ,8]), ylab = paste(as.character(colnames(train)[8]), "g/cm^3"))
boxplot(log(train[ ,9]), ylab = paste('log', as.character(colnames(train)[9])))
boxplot(log(train[ ,10]), ylab = paste(as.character(colnames(train)[5]), "log g(potassium sulphate)/dm^
boxplot(log(train[ ,11]), ylab = paste(as.character(colnames(train)[5]), "log vol.%"))
mtext("Boxplots of 11 Variables After Transformation", outer = TRUE, cex = 1.5)
```

Boxplots of 11 Variables After Transformation



Create a new dataframe that includes any transformed variables in place of the original ones

```
y = train
y[ ,1] = log(y[ ,1])
y[ ,2] = log(y[ ,2])
y[ ,3] = sqrt(y[ ,3])
y[ ,4] = log(log(y[ ,4]))
```

```
## Warning in log(log(y[, 4])): NaNs produced
```

```
y[ ,5] = log(y[ ,5])
y[ ,6] = log(y[ ,6])
y[ ,7] = log(y[ ,7])
y[ ,9] = log(y[ ,9])
y[ ,10] = log(y[ ,10])
y[ ,11] = log(y[ ,11])



# Create a vector of variable names from dataset
y_names = colnames(y)
y_names
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

Part 2

Use Mallow's Cp with forward step-wise selection to select a model. Repeat for AIC and then for BIC. These statistics seek to balance misfit and complexity in linear regression.

Mallow's Cp

Determine first variable to include in the model based on Mallow's Cp

```
library(olsrr)
```

```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers
```

```
full_model <- lm(quality ~ ., data = y)
cp = c()
for (i in 1:(dim(y)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,i])
  mallow <- ols_mallows_cp(model, full_model)
  cp <- c(cp, mallow)
}
cp
```

```
##  [1] 739.7921 444.8528 674.0011 764.6292 695.6737 761.7697 702.7206
##  [8] 703.8886 760.2227 560.6870 285.2442
```

```
min(cp)
```

```
## [1] 285.2442
```

```
all_selected <- c()
index <- which(cp == min(cp))
index
```

```
## [1] 11
```

```
all_selected <- c(all_selected,index)
all_selected
```

```
## [1] 11
```

```
selected = y_names[index]
selected
```

```
## [1] "alcohol"
```

```
y_new <- y[ ,!y_names %in% selected]
dim(y_new)
```

```
## [1] 1440   11
```

```
names = colnames(y_new)
names
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "quality"
```

Mallow's Cp statistic nearest to the number of predictors in the model is for y[,11]: alcohol (285.24). Alcohol will be first predictor added to model based off selection criterion.

Determine second variable to include in the model based on Mallow's Cp

```
cp = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,i])
  mallow <- ols_mallows_cp(model, full_model)
  cp <- c(cp, mallow)
}
cp
min(cp)
index <- which(cp == min(cp))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Mallow's Cp statistic nearest to the number of predictors in the model is for y[,2]: volatile acidity (113.32). Volatile acidity will be second predictor added to model based off selection criterion.

Determine third variable to include in the model based on Mallow's Cp

```
# Determine third variable to include in the model based on Mallow's Cp
cp = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,i])
  mallow <- ols_mallows_cp(model, full_model)
  cp <- c(cp, mallow)
}
min(cp)
cp
index <- which(cp == min(cp))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Mallow's Cp statistic nearest to the number of predictors in the model is for y[,1]: fixed acidity (102.42). Fixed acidity will be third predictor added to model based off selection criterion.

Determine fourth variable to include in the model based on Mallow's Cp

```
cp = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,i])
  mallow <- ols_mallows_cp(model, full_model)
  cp <- c(cp, mallow)
}
min(cp)
cp
index <- which(cp == min(cp))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected
# Mallow's Cp statistic nearest to the number of predictors in the model is for  y[ ,9]: pH (101.08)
# pH will be fourth predictor added to model based off selection criterion.
y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Mallow's Cp statistic nearest to the number of predictors in the model is for y[,9]: pH (101.08). pH will be fourth predictor added to model based off selection criterion.

Determine fifth variable to include in the model based on Mallow's Cp

```
cp = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,9] + y[ ,i])
  mallow <- ols_mallows_cp(model, full_model)
  cp <- c(cp, mallow)
}
min(cp)
cp
index <- which(cp == min(cp))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Mallow's Cp statistic nearest to the number of predictors in the model is for y[,10]: sulphates (100.12). Sulphates will be fifth predictor added to model based off selection criterion.

Determine sixth variable to include in the model based on Mallow's Cp

```
cp = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,9] + y[ ,10] + y[ ,i])
  mallow <- ols_mallows_cp(model, full_model)
  cp <- c(cp, mallow)
}
min(cp)
cp
index <- which(cp == min(cp))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Mallow's Cp statistic nearest to the number of predictors in the model is for y[,7]: total sulfer dioxide (25.83). Total sulfer dioxide will be sixth predictor added to model based off selection criterion.

Determine seventh variable to include in the model based on Mallow's Cp

```
cp = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,9] + y[ ,10] + y[ ,7] + y[ ,i])
  mallow <- ols_mallows_cp(model, full_model)
  cp <- c(cp, mallow)
}
min(cp)
cp
index <- which(cp == min(cp))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Mallow's Cp statistic nearest to the number of predictors in the model is for y[,8]: density (17.58). Density will be seventh predictor added to model based off selection criterion.

Determine eighth variable to include in the model based on Mallow's Cp

```
cp = c()
```

```
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,9] + y[ ,10] + y[ ,7] +y[ ,8] + y[ ,i])
  mallow <- ols_mallows_cp(model, full_model)
  cp <- c(cp, mallow)
}
min(cp)
```

Minimum Mallow's Cp statistic is less favorable than in the prior round. No additional variables will be included in the model based of selection criterion.

Model based on Mallow's Cp:

```
model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,9] + y[ ,10] + y[ ,7] +y[ ,8])
summary(model)
```

```
##
## Call:
## lm(formula = y[, 12] ~ y[, 11] + y[, 2] + y[, 10] + y[, 9] +
##     y[, 10] + y[, 7] + y[, 8])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6810 -0.3620 -0.0509  0.4698  1.9940
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.91632   11.70080   0.933  0.35100
## y[, 11]       3.04886    0.21974  13.875  < 2e-16 ***
## y[, 2]       -0.50198    0.05403  -9.290  < 2e-16 ***
## y[, 10]       0.68032    0.08494   8.009 2.37e-15 ***
## y[, 9]       -1.10410    0.41500  -2.660  0.00789 **
## y[, 7]       -0.07937    0.02549  -3.114  0.00188 **
## y[, 8]      -10.90088   11.33599  -0.962  0.33640
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6568 on 1433 degrees of freedom
## Multiple R-squared:  0.336,  Adjusted R-squared:  0.3333
## F-statistic: 120.9 on 6 and 1433 DF,  p-value: < 2.2e-16
```

AIC

Determine first variable to include in the model based on AIC

```
aic = c()
for (i in 1:(dim(y)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,i])
  stat <- ols_aic(model)
  aic <- c(aic, stat)
}
aic
```

```
##  [1] 3446.433 3236.672 3402.219 3458.668 3416.934 3460.905 3421.687
##  [8] 3422.473 3459.891 3322.732 3108.976
```

```
min(aic)
```

```
## [1] 3108.976
```

```
index <- which(aic == min(aic))
index
```

```
## [1] 11
```

```
selected = y_names[index]
selected
```

```
## [1] "alcohol"
```

```
all_selected = c()
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected
```

```
## [1] 11
```

```
y_new <- y[ ,!y_names %in% selected]
dim(y_new)
```

```
## [1] 1440   11
```

```
names = colnames(y_new)
names
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "quality"
```

Minimum AIC statistic for y[,11]: alcohol (3108.98). Alcohol will be first predictor added to model based off selection criterion.

Determine second variable to include in the model based on AIC

```
aic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,i])
  stat <- ols_aic(model)
  aic <- c(aic, stat)
}
aic
```

```
min(aic)
index <- which(aic == min(aic))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum AIC statistic for y[,2]: volatile acidity (2957.58). Volatile acidity will be second predictor added to model based off selection criterion.

Determine third variable to include in the model based on AIC

```
aic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,i])
  stat <- ols_aic(model)
  aic <- c(aic, stat)
}
aic
min(aic)
index <- which(aic == min(aic))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum AIC statistic for y[,1]: fixed acidity (2947.53). Fixed acidity will be third predictor added to model based off selection criterion.

Determine fourth variable to include in the model based on AIC

```
aic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,i])
  stat <- ols_aic(model)
  aic <- c(aic, stat)
}
aic
min(aic)
index <- which(aic == min(aic))
index
selected = names[index]
```

```
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum AIC statistic for y[,6]: free sulphur dioxide (2943.53). Free sulphur dioxide will be fourth predictor added to model based off selection criterion.

Determine fifth variable to include in the model based on AIC

```
aic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,6] + y[ ,i])
  stat <- ols_aic(model)
  aic <- c(aic, stat)
}
aic
min(aic)
index <- which(aic == min(aic))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum AIC statistic for y[,10]: sulphates (2940.26). Sulphates will be fifth predictor added to model based off selection criterion.

Determine sixth variable to include in the model based on AIC

```
aic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,6] + y[ ,10] + y[ ,i])
  stat <- ols_aic(model)
  aic <- c(aic, stat)
}
aic
min(aic)
index <- which(aic == min(aic))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected
```

```
y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum AIC statistic for y[,8]: density (2879.40). Density will be sixth predictor added to model based off selection criterion.

Determine seventh variable to include in the model based on AIC

```
aic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,6] + y[ ,10] + y[ ,8] + y[ ,i])
  stat <- ols_aic(model)
  aic <- c(aic, stat)
}
aic
min(aic)
index <- which(aic == min(aic))
index
selected = names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum AIC statistic for y[,9]: pH (2875.65). pH will be seventh predictor added to model based off selection criterion.

Determine eighth variable to include in the model based on AIC

```
aic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,6] + y[ ,10] + y[ ,8] + y[ ,i])
  stat <- ols_aic(model)
  aic <- c(aic, stat)
}
aic
min(aic)
```

Minimum AIC statistic is less favorable than in the prior round. No additional variables will be included in the model based of selection criterion.

Model based on AIC:

```
model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,6] + y[ ,10] + y[ ,8])
summary(model)
```

```
##
## Call:
```

```
## lm(formula = y[, 12] ~ y[, 11] + y[, 2] + y[, 1] + y[, 6] + y[,
##     10] + y[, 8])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.79078 -0.37008 -0.06849  0.47811  2.01176
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  42.180234  16.292877   2.589 0.009726 **
## y[, 11]       2.870665   0.226123  12.695  < 2e-16 ***
## y[, 2]       -0.478310   0.055908  -8.555  < 2e-16 ***
## y[, 1]        0.484531   0.138622   3.495 0.000488 ***
## y[, 6]       -0.009068   0.026489  -0.342 0.732161
## y[, 10]       0.687134   0.085093   8.075 1.42e-15 ***
## y[, 8]      -44.436957  16.236239  -2.737 0.006279 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6577 on 1433 degrees of freedom
## Multiple R-squared:  0.3342, Adjusted R-squared:  0.3315
## F-statistic: 119.9 on 6 and 1433 DF,  p-value: < 2.2e-16
```

BIC

Determine first variable to include in the model based on BIC

```r
bic = c()
for (i in 1:(dim(y)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,i])
  stat <- ols_sbic(model, full_model)
  bic <- c(bic, stat)
}
bic
```

```
##  [1] -641.6914 -850.9187 -685.7960 -623.8226 -671.1174 -627.2545 -666.3767
##  [8] -665.5924 -628.2660 -765.0817 -978.2743
```

```r
min(bic)
```

```
## [1] -978.2743
```

```r
index <- which(bic == min(bic))
index
```

```
## [1] 11
```

```r
selected = y_names[index]
selected
```

```
## [1] "alcohol"
```

```r
all_selected = c()
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected
```

```
## [1] 11
```

```r
y_new <- y[ ,!names %in% selected]
dim(y_new)
```

```
## [1] 1440   12
```

```r
names = colnames(y_new)
names
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

Minimum BIC statistic for y[,11]: alcohol (-978.27). Alcohol will be first predictor added to model based off selection criterion.

Determine second variable to include in the model based on BIC

```r
bic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,i])
  stat <- ols_sbic(model, full_model)
  bic <- c(bic, stat)
}
bic
```

```
min(bic)
index <- which(bic == min(bic))
index
selected = y_names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum BIC statistic for y[,2]: volatile acidity (-1129.39). Volatile acidity will be first predictor added to model based off selection criterion.

Determine third variable to include in the model based on BIC

```
bic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,i])
  stat <- ols_sbic(model, full_model)
  bic <- c(bic, stat)
}
bic
min(bic)
index <- which(bic == min(bic))
index
selected = y_names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum BIC statistic for y[,10]: sulphates (-1190.84). Sulphates will be third predictor added to model based off selection criterion.

Determine fourth variable to include in the model based on BIC

```
bic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,i])
  stat <- ols_sbic(model, full_model)
  bic <- c(bic, stat)
}
bic
min(bic)
index <- which(bic == min(bic))
index
selected = y_names[index]
```

```
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum BIC statistic for y[,5]: chlorides (-1201.82). Chlorides will be fourth predictor added to model based off selection criterion.

Determine fifth variable to include in the model based on BIC

```
bic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,5] + y[ ,i])
  stat <- ols_sbic(model, full_model)
  bic <- c(bic, stat)
}
bic
min(bic)
index <- which(bic == min(bic))
index
selected = y_names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum BIC statistic for y[,7]: total sulfur dioxide (-1210.56). Total sulfur dioxide will be fifth predictor added to model based off selection criterion.

Determine sixth variable to include in the model based on BIC

```
bic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,5] + y[ ,7] + y[ ,i])
  stat <- ols_sbic(model, full_model)
  bic <- c(bic, stat)
}
bic
min(bic)
index <- which(bic == min(bic))
index
selected = y_names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected
```

```
y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum BIC statistic for y[,1]: fixed acidity (-1215.05).Fixed acidity will be sixth predictor added to model based off selection criterion.

Determine seventh variable to include in the model based on BIC

```
bic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,5] + y[ ,7] + y[ ,1] + y[ ,i])
  stat <- ols_sbic(model, full_model)
  bic <- c(bic, stat)
}
bic
min(bic)
index <- which(bic == min(bic))
index
selected = y_names[index]
selected
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected

y_new <- y_new[ ,!names %in% selected]
dim(y_new)
names = colnames(y_new)
names
```

Minimum BIC statistic for y[,6]: free sulfur dioxide (-1219.99). Free sulfur dioxide will be seventh predictor added to model based off selection criterion.

Determine eighth variable to include in the model based on BIC

```
bic = c()
for (i in 1:(dim(y_new)[2]-1)) {
  model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,5] + y[ ,7] + y[ ,1] + y[ ,6] + y[ ,i])
  stat <- ols_sbic(model, full_model)
  bic <- c(bic, stat)
}
bic
min(bic)
```

Minimum BIC statistic is equal to that of the prior round. No additional variables will be included in the model based of selection criterion.

Model based on BIC:

```
model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,5] + y[ ,7] + y[ ,1] + y[ ,6])
summary(model)
```

```
##
## Call:
```

```
## lm(formula = y[, 12] ~ y[, 11] + y[, 2] + y[, 10] + y[, 5] +
##     y[, 7] + y[, 1] + y[, 6])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7679 -0.3557 -0.0557  0.4612  2.0277
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.80538    0.55586  -3.248 0.001189 **
## y[, 11]      2.79896    0.19961  14.022  < 2e-16 ***
## y[, 2]      -0.45535    0.05499  -8.280 2.79e-16 ***
## y[, 10]      0.76061    0.08639   8.804  < 2e-16 ***
## y[, 5]      -0.24243    0.06045  -4.011 6.37e-05 ***
## y[, 7]      -0.15822    0.04112  -3.848 0.000124 ***
## y[, 1]       0.27725    0.09464   2.929 0.003450 **
## y[, 6]       0.11131    0.04214   2.641 0.008349 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6524 on 1432 degrees of freedom
## Multiple R-squared:  0.3453, Adjusted R-squared:  0.3421
## F-statistic: 107.9 on 7 and 1432 DF,  p-value: < 2.2e-16
```

Part 3

Perform iterative fitting by looking at residual plots

Determine first variable to include by looking at scatterplots of variables against the response

```
all_selected = c()
par(mfrow = c(3,4), oma = c(0,0,2,0))
for (i in 1:11) {
  plot(y$quality, y[ ,i])
}
mtext("Simple Scatterplots of Quality Against 11 Predictors", outer = TRUE, cex = 1.2)

selected <- y_names[11]
selected
```

```
## [1] "alcohol"
```

```
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected
```

```
## [1] 11
```

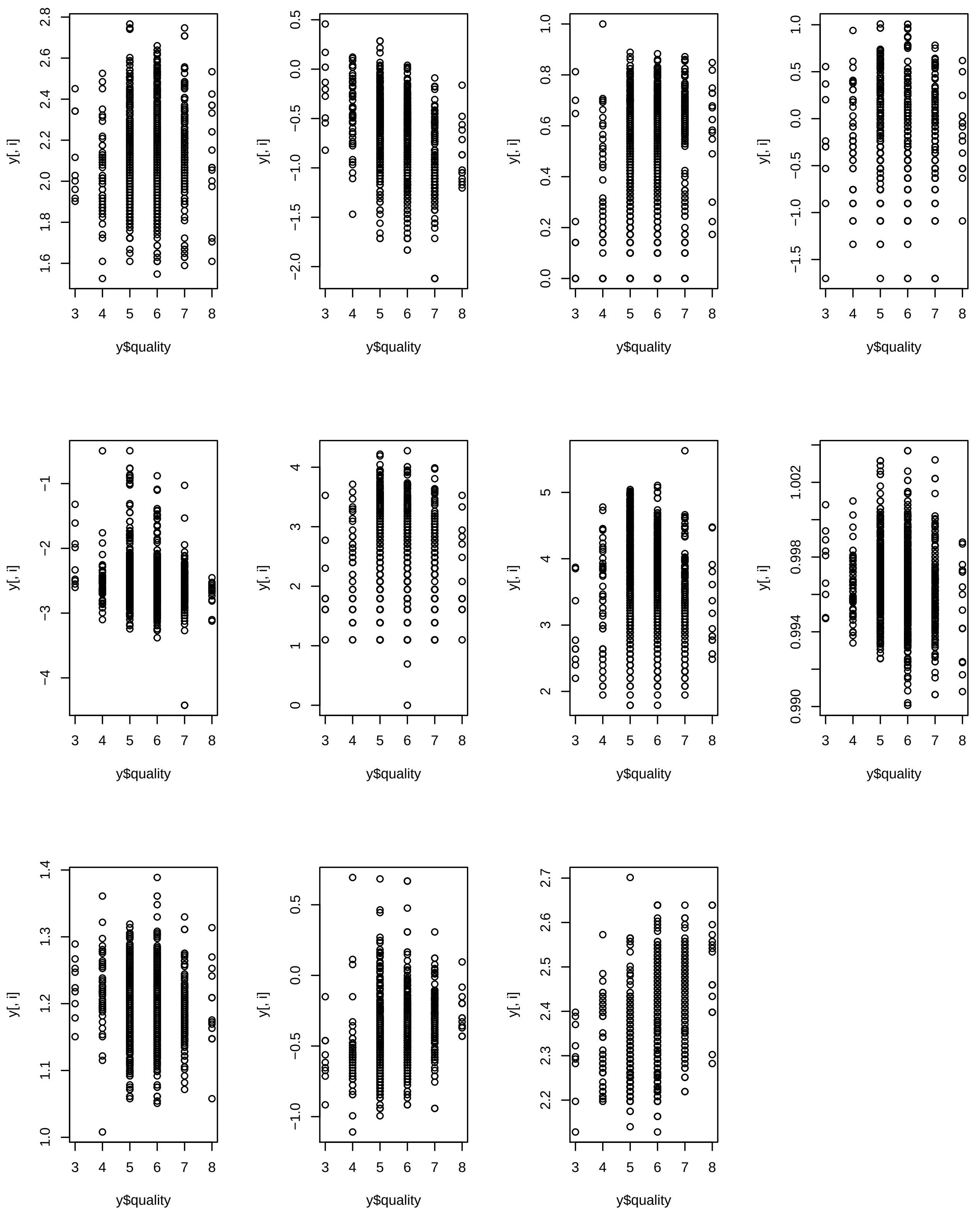
```
y_new <- y[ ,!y_names %in% selected]
dim(y_new)
```

```
## [1] 1440   11
```

```
names = colnames(y_new)
names
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "quality"
```

Simple Scatterplots of Quality Against 11 Predictors



Plot 11 (alcohol) seems to be the strongest predictor of quality based on scatterplots. (Trend most apparent). Trend appears to be linear.

Determine second variable to include by looking at scatterplots of variables against the residual (after including alcohol in the model)

```
par(mfrow = c(3,4), oma = c(0,0,2,0))
model <- lm(y$quality ~ y[ ,11])
for (i in 1:10) {
 plot(model$residuals, y_new[ ,i])
}
mtext("Simple Scatterplots of Residuals Against 10 Predictors", outer = TRUE, cex = 1.2)

selected <- names[2]
selected
```

```
## [1] "volatile.acidity"
```

```
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected
```

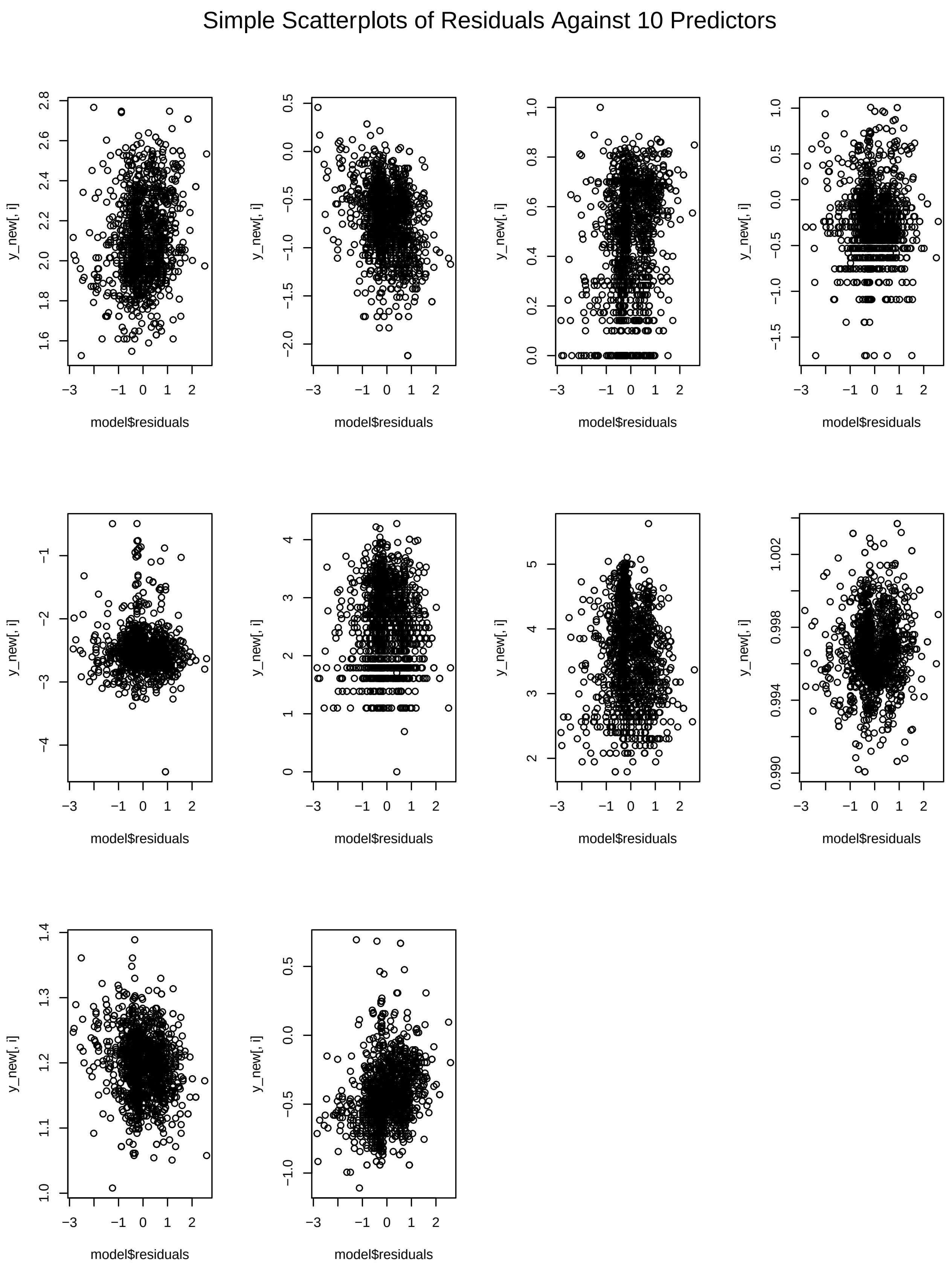
```
## [1] 11  2
```

```
y_new <- y_new[ ,!names %in% selected]
dim(y_new)
```

```
## [1] 1440   10
```

```
names = colnames(y_new)
names
```

```
##  [1] "fixed.acidity"        "citric.acid"          "residual.sugar"
##  [4] "chlorides"            "free.sulfur.dioxide"  "total.sulfur.dioxide"
##  [7] "density"              "pH"                   "sulphates"
## [10] "quality"
```



Plot 2 (volatile acidity) seems to be the strongest predictor of quality based on scatterplots. Trend appears to be linear.

Determine third variable to include by looking at scatterplots of variables against the residual (after including selected variables in the model)

```
par(mfrow = c(3,4), oma = c(0,0,2,0))
model <- lm(y$quality ~ y[ ,11] + y[ ,2])
for (i in 1:9) {
  plot(model$residuals, y_new[ ,i])
}
mtext("Simple Scatterplots of Residuals Against 9 Predictors", outer = TRUE, cex = 1.2)


selected <- names[9]
selected
```

```
## [1] "sulphates"
```

```
all_selected <- c(all_selected, which(y_names %in% selected))
all_selected
```

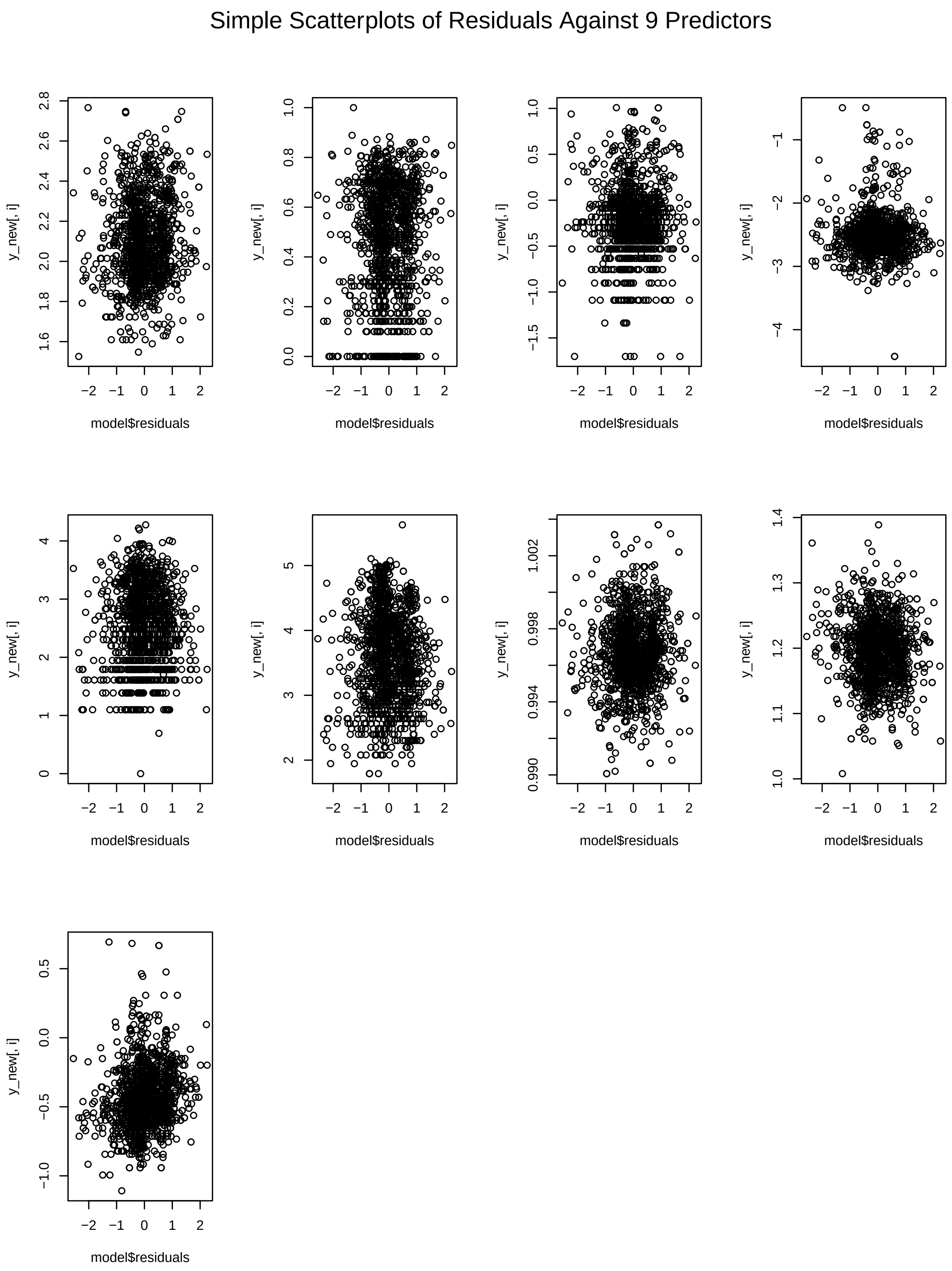
```
## [1] 11  2 10
```

```
y_new <- y_new[ ,!names %in% selected]
dim(y_new)
```

```
## [1] 1440    9
```

```
names = colnames(y_new)
names
```

```
## [1] "fixed.acidity"        "citric.acid"          "residual.sugar"
## [4] "chlorides"            "free.sulfur.dioxide"  "total.sulfur.dioxide"
## [7] "density"              "pH"                   "quality"
```



Plot 9 (sulphates) seems to be the strongest predictor of quality based on scatterplots. Trend appears to be linear.

Determine fourth variable to include by looking at scatterplots of variables against the residual (after including selected variables in the model). This time, I plot each residual within a "null lineup". I give myself 20 seconds to see if any variable stands out against the null field. (Code displayed below, but plots are excluded).

```
library("BBmisc")

par(mfrow = c(3,3), oma = c(0,0,2,0))
model <- lm(y$quality ~ y[ ,11] + y[ ,2] + y[ ,10])
for (i in 1:8) {
  answer <- sample(1:9,1)
  for (j in 1:9) {
    if (j==answer) {
      var <- normalize(y_new[ ,i], method = "standardize")
      plot(model$residuals, var)
    }
    else {
      fakes <- rnorm(length(y_new[ ,1]))
      var <- normalize(fakes, method = "standardize")
      plot(model$residuals,var)
    }
  }
  Sys.sleep(10)
  print(answer)
  Sys.sleep(10)
}
selected <- names[7]
selected
```

No variables seem to stand out against the null lineup. Variable selection accomplished.

Model selected via iterative fitting:

```
model <- lm(y$quality ~ y[ ,11] + y[ ,2] + y[ ,10])
summary(model)
```

```
##
## Call:
## lm(formula = y$quality ~ y[, 11] + y[, 2] + y[, 10])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.73738 -0.36997 -0.05877  0.48287  2.13387
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.84891    0.42748  -4.325 1.63e-05 ***
## y[, 11]      3.15592    0.18286  17.258  < 2e-16 ***
## y[, 2]      -0.54838    0.05253 -10.439  < 2e-16 ***
## y[, 10]      0.65630    0.08147   8.056 1.64e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.66 on 1436 degrees of freedom
## Multiple R-squared:  0.3281, Adjusted R-squared:  0.3267
## F-statistic: 233.7 on 3 and 1436 DF,  p-value: < 2.2e-16
```

Part 4

Trial ridge and lasso regression, regularization techniques that discount parameters based on the size of their coefficients.

First, standardize transformed variables so that they don't depend on scale

```
library("BBmisc")
```

```
##
## Attaching package: 'BBmisc'

## The following object is masked from 'package:base':
##
##     isFALSE
```

```
library("glmnet")
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16
```

```
r = y
for (i in 1:11) {
  var <- normalize(y[ ,i], method = "standardize")
  r[ ,i] <- var
}
```

(Double log transformation created two NaN values, which must be dropped before ridge regression)

```
# (Double log transformation created two NaN values, which must be dropped before ridge regression)
which(is.na(y[ ,4]) %in% TRUE)
```

```
## [1] 915 916
```

```
r <- rbind(r[1:914, ], r[917:1440, ])
dim(r)
```

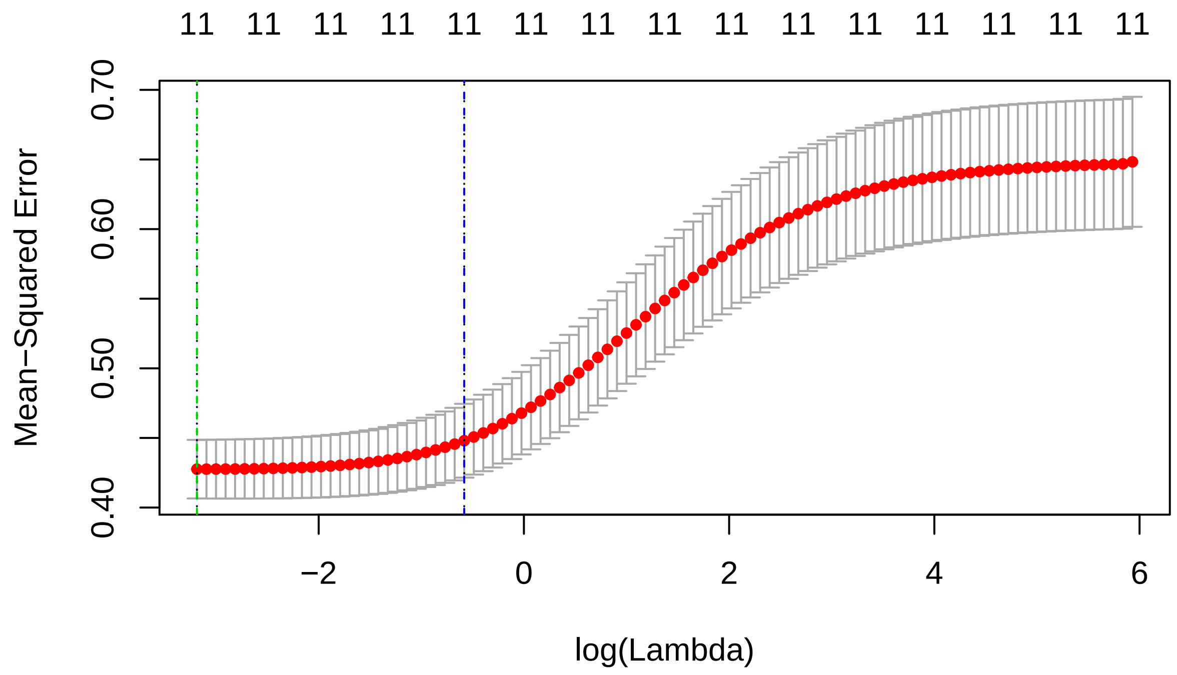
```
## [1] 1438   12
```

Ridge Regression

```
x <- as.matrix(r[,1:11])
y1 <- as.matrix(r[,12])

# Run ridge regression
CV.ridge <- cv.glmnet(x, y1, alpha=0)
par(mfrow = c(1,1), oma = c(0,0,2,0))
plot(CV.ridge) # CV test error vs log lambda
abline(v=log(CV.ridge$lambda.min), col=3, lty=2)
abline(v=log(CV.ridge$lambda.1se), col=4, lty=2)
mtext("Optimal Coefficients for Lambda", outer = TRUE, cex = 1.2)
```

Optimal Coefficients for Lambda

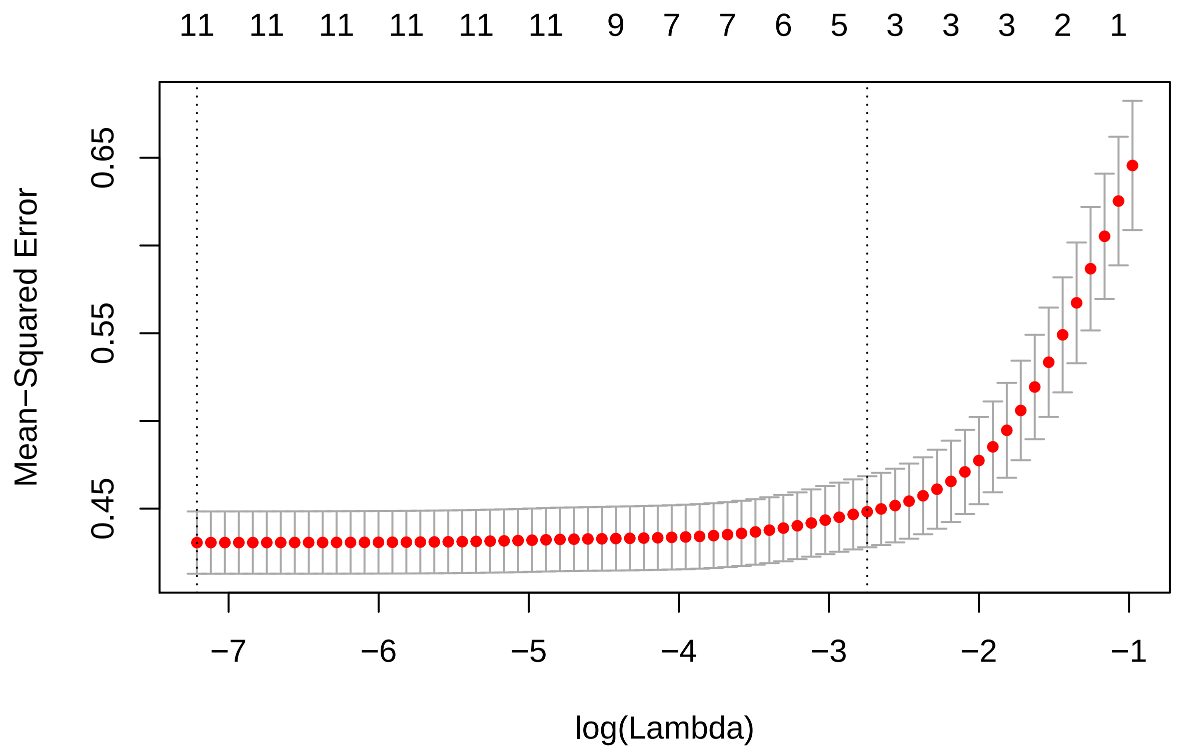


```
ridge <- coef(CV.ridge, s=CV.ridge$lambda.1se)
ridge
```

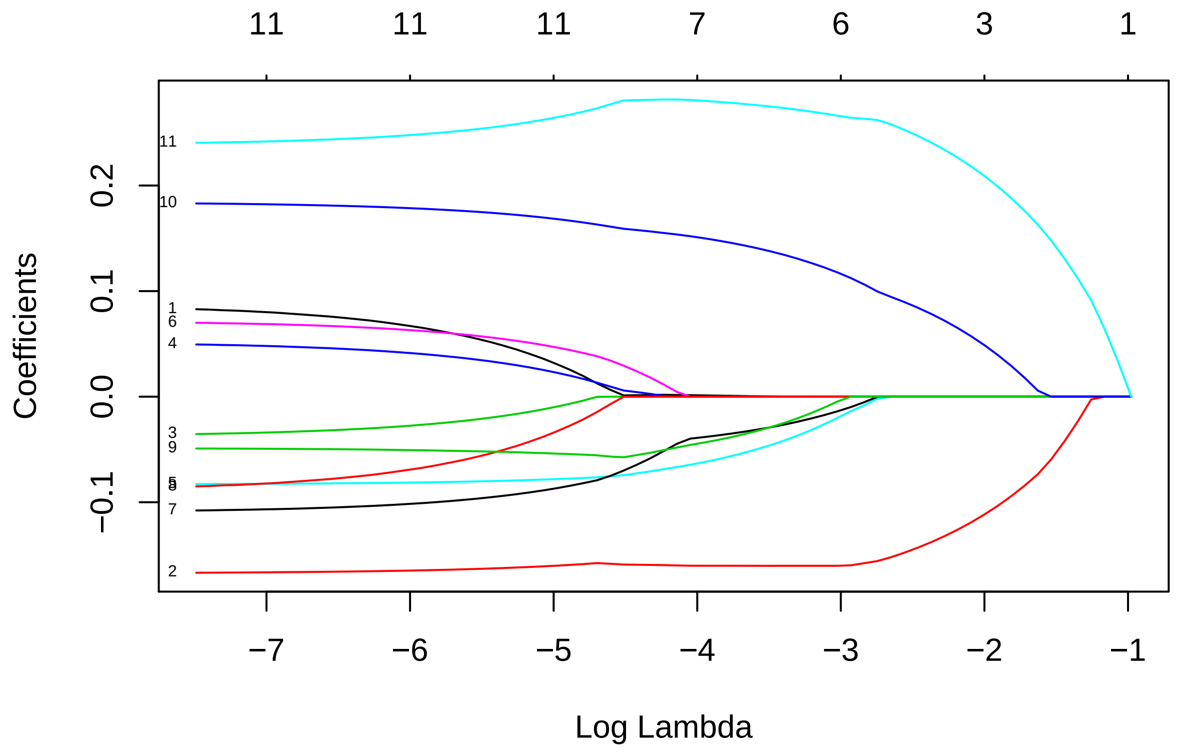
```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept)          5.62138368
## fixed.acidity        0.03689490
## volatile.acidity    -0.11388891
## citric.acid          0.02900575
## residual.sugar       0.02484447
## chlorides           -0.05919115
## free.sulfur.dioxide  0.01164506
## total.sulfur.dioxide -0.05485247
## density             -0.05796271
## pH                  -0.01884314
## sulphates            0.11574079
## alcohol              0.16446936
```

LASSO Regression

```
CV.lasso <- cv.glmnet(x, y1, alpha=1)
plot(CV.lasso) # CV test error vs lambda
```



```
plot(CV.lasso$glmnet.fit, xvar="lambda", label=TRUE)
mtext("Optimal Coefficients for Lambda", outer = TRUE, cex = 1.2)
```



```
lasso <- coef(CV.lasso, s=CV.lasso$lambda.1se)
lasso
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)           5.6214558629
## fixed.acidity         .
## volatile.acidity     -0.1556960207
## citric.acid           .
## residual.sugar        .
## chlorides            -0.0019801483
## free.sulfur.dioxide   .
## total.sulfur.dioxide -0.0001741696
## density               .
## pH                    .
## sulphates             0.0996447726
## alcohol               0.2620921199
```

Part 5

A break from model selection: exploring significance probabilities. I perform the following simulation 100,000 times: Permute the response (wine quality) values at random in order to break any relationship with the explanatory variables, and record the 11 coefficients' significance probabilities when all 11 of the explanatory variables are included in a multiple linear model. For each variable, calculate the proportion of simulation trials in which its significance probability was less than .05.

Create an empty dataframe to store significance probabilities

```
column= rep(NA, dim(y)[1])
significance_probs <- data.frame(column, column, column, column, column, column, column, column, column
dim(significance_probs)
```

```
## [1] 1440   11
```

```
colnames(significance_probs) <- colnames(y)[1:11]
z <- y[1:11]
```

Run simulation 100,000 times

```
for (i in 1:100000) {
  # progress indicator
  if(i%%1000==0) {
    print(paste("----", i))
  }

  # set random permutation
  random_quality <- sample(train$quality, size = length(train$quality), replace = FALSE)
  y$quality <- random_quality

  # Regress the permuted quality variable against all 11 supplied predictors
  model <- lm(y$quality ~ ., data = z)
  significance_probs[i,] <- summary(model)$coefficients[ ,4][2:12]
}
```

```
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
```

```r
head(significance_probs)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1     0.2445312        0.6507026  0.92621609     0.56310521 0.71381107
## 2     0.6296004        0.2355628  0.09015678     0.06686352 0.22438766
## 3     0.7702769        0.2708342  0.44658162     0.85113248 0.57891489
## 4     0.5263636        0.2554974  0.57772808     0.16689406 0.08952525
## 5     0.8881933        0.5142830  0.49751909     0.35442465 0.35296084
## 6     0.8323941        0.8921019  0.58242106     0.53090581 0.60309991
##   free.sulfur.dioxide total.sulfur.dioxide   density        pH sulphates
## 1           0.4370981            0.5149207 0.6977580 0.1298366 0.8915223
## 2           0.3015407            0.5044814 0.5292112 0.9652537 0.2015194
## 3           0.3427612            0.5110948 0.6559146 0.9822668 0.2297952
## 4           0.1951309            0.4683864 0.2421925 0.2267764 0.2771097
## 5           0.6350654            0.5564602 0.9948413 0.1866608 0.8778612
## 6           0.4078011            0.9444602 0.5808354 0.6083197 0.6698558
##     alcohol
## 1 0.8259354
## 2 0.3883613
## 3 0.8058061
## 4 0.2793435
## 5 0.5780918
## 6 0.4989751
```

For each variable, calculate the proportion of simulation trials in which its significance probability was less than .05

```r
library("expss")

proportions <- c()
for (i in 1:11) {
  count <- count_if(lt(.05), significance_probs[ ,i])
  proportion <- count/100000
  proportions <- c(proportions, proportion)
}

proportions <- data.frame(proportions,  row.names = colnames(significance_probs))
proportions
```

```
##                      proportions
## fixed.acidity            0.04997
## volatile.acidity         0.04934
## citric.acid              0.05107
## residual.sugar           0.05107
## chlorides                0.04969
## free.sulfur.dioxide      0.05111
## total.sulfur.dioxide     0.05036
## density                  0.04977
## pH                       0.04999
## sulphates                0.04937
## alcohol                  0.04974
```

Proportions are (as expected!) all very close to .05 false positive rate.

Part 6

Calculate the proportion of the simulation trials in which at least one significance probability was smaller than the Bonferroni threshold of (.05/11).

Create a vector of threshold values

```
thresholds <- c()
threshold <- 0
i <- 1
while (threshold <= .05) {
  threshold <- (.05/11)*i
  thresholds <- c(thresholds,threshold)
  i <- i+1
}
thresholds
```

```
##  [1] 0.004545455 0.009090909 0.013636364 0.018181818 0.022727273
##  [6] 0.027272727 0.031818182 0.036363636 0.040909091 0.045454545
## [11] 0.050000000
```

Iterate through the thresholds, calculating the proportion of total observations that have at least one variable which are significant according to that threshold.

```
false_positives_by_threshold <- data.frame(thresholds, proportion_false_pos = NA)
false_positives_by_threshold
```

```
##     thresholds proportion_false_pos
## 1  0.004545455                   NA
## 2  0.009090909                   NA
## 3  0.013636364                   NA
## 4  0.018181818                   NA
## 5  0.022727273                   NA
## 6  0.027272727                   NA
## 7  0.031818182                   NA
## 8  0.036363636                   NA
## 9  0.040909091                   NA
## 10 0.045454545                   NA
## 11 0.050000000                   NA
```

```
for (i in 1:(length(thresholds))) {
  total <- 0
  for (j in 1:100000){
    # progress indicator
    if(j%%1000==0) {
      print(paste("----", j))
    }

    count <- count_row_if(lt(thresholds[i]), significance_probs[j,])
    if (count > 0) {
      total = total + 1
    }
  }
  proportion <- total/100000
  false_positives_by_threshold[i,2] <- proportion
}
```

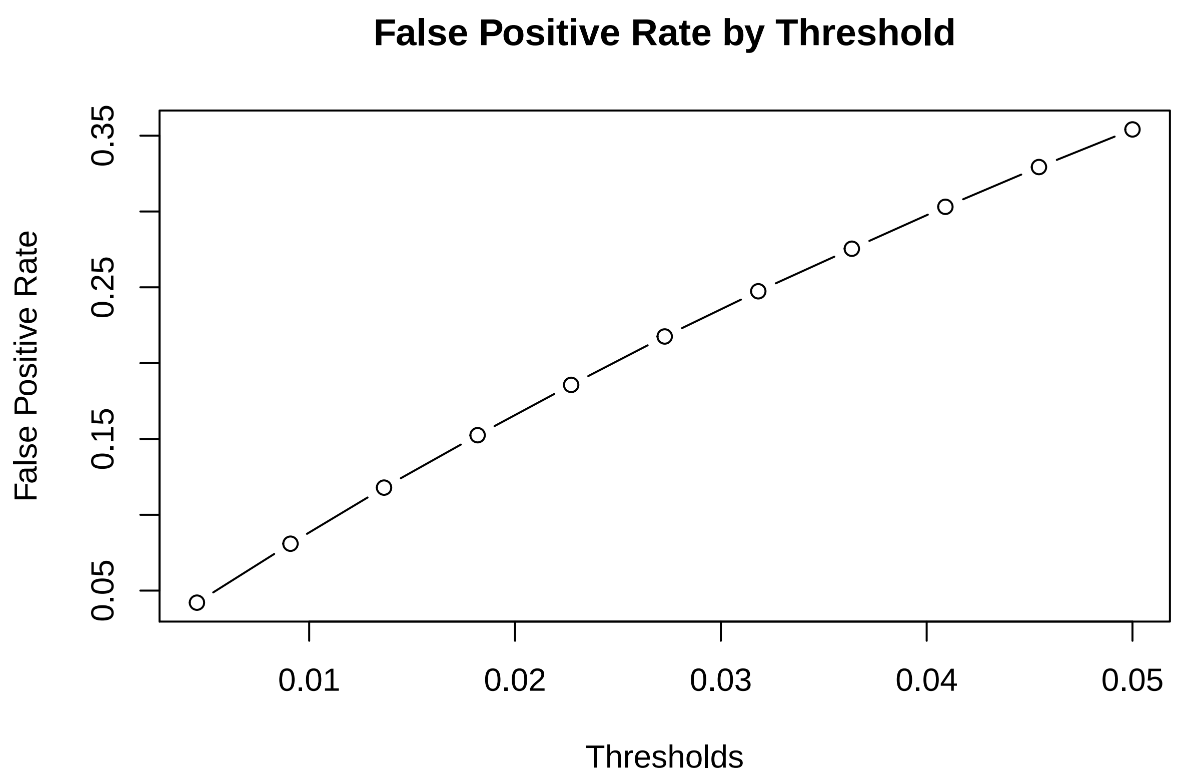
```
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
## [1] "---- 1000"
## [1] "---- 2000"
## [1] "---- 3000"
## [1] "---- 4000"
## [1] "---- 5000"
## [1] "---- 6000"
## [1] "---- 7000"
## [1] "---- 8000"
## [1] "---- 9000"
## [1] "---- 10000"
## [1] "---- 11000"
## [1] "---- 12000"
## [1] "---- 13000"
## [1] "---- 14000"
## [1] "---- 15000"
## [1] "---- 16000"
## [1] "---- 17000"
## [1] "---- 18000"
## [1] "---- 19000"
## [1] "---- 20000"
## [1] "---- 21000"
## [1] "---- 22000"
## [1] "---- 23000"
## [1] "---- 24000"
## [1] "---- 25000"
## [1] "---- 26000"
## [1] "---- 27000"
## [1] "---- 28000"
## [1] "---- 29000"
## [1] "---- 30000"
## [1] "---- 31000"
## [1] "---- 32000"
## [1] "---- 33000"
## [1] "---- 34000"
## [1] "---- 35000"
## [1] "---- 36000"
## [1] "---- 37000"
## [1] "---- 38000"
## [1] "---- 39000"
## [1] "---- 40000"
## [1] "---- 41000"
## [1] "---- 42000"
## [1] "---- 43000"
## [1] "---- 44000"
## [1] "---- 45000"
## [1] "---- 46000"
## [1] "---- 47000"
## [1] "---- 48000"
## [1] "---- 49000"
## [1] "---- 50000"
## [1] "---- 51000"
## [1] "---- 52000"
## [1] "---- 53000"
## [1] "---- 54000"
## [1] "---- 55000"
## [1] "---- 56000"
## [1] "---- 57000"
## [1] "---- 58000"
## [1] "---- 59000"
## [1] "---- 60000"
## [1] "---- 61000"
## [1] "---- 62000"
## [1] "---- 63000"
## [1] "---- 64000"
## [1] "---- 65000"
## [1] "---- 66000"
## [1] "---- 67000"
## [1] "---- 68000"
## [1] "---- 69000"
## [1] "---- 70000"
## [1] "---- 71000"
## [1] "---- 72000"
## [1] "---- 73000"
## [1] "---- 74000"
## [1] "---- 75000"
## [1] "---- 76000"
## [1] "---- 77000"
## [1] "---- 78000"
## [1] "---- 79000"
## [1] "---- 80000"
## [1] "---- 81000"
## [1] "---- 82000"
## [1] "---- 83000"
## [1] "---- 84000"
## [1] "---- 85000"
## [1] "---- 86000"
## [1] "---- 87000"
## [1] "---- 88000"
## [1] "---- 89000"
## [1] "---- 90000"
## [1] "---- 91000"
## [1] "---- 92000"
## [1] "---- 93000"
## [1] "---- 94000"
## [1] "---- 95000"
## [1] "---- 96000"
## [1] "---- 97000"
## [1] "---- 98000"
## [1] "---- 99000"
## [1] "---- 100000"
```

```
false_positives_by_threshold
```

```
##     thresholds proportion_false_pos
## 1  0.004545455              0.04206
## 2  0.009090909              0.08092
## 3  0.013636364              0.11792
## 4  0.018181818              0.15249
## 5  0.022727273              0.18565
## 6  0.027272727              0.21758
## 7  0.031818182              0.24739
## 8  0.036363636              0.27543
## 9  0.040909091              0.30308
## 10 0.045454545              0.32929
## 11 0.050000000              0.35410
```

Draw a scatterplot showing these thresholds as x values and the estimated total false positive rates as y values.

```
par(mfrow = c(1,1))
plot(false_positives_by_threshold, xlab = "Thresholds", ylab = "False Positive Rate", type = "b", main =
```



Return to the actual values and fit the multiple linear model that includes only the variables whose original significance probabilities were below the threshold you just proposed. Bonferroni threshold of .05/11 gives a total false positive rate of .042

(Code below is used to check whether a variable should be included in model based on Bonferroni criteria)

```
true_model <- lm(y$quality ~ y$fixed.acidity, data = y)
summary(true_model)
true_model <- lm(y$quality ~ y$volatile.acidity, data = y) # IN
summary(true_model)
true_model <- lm(y$quality ~ y$citric.acid, data = y)
summary(true_model)
true_model <- lm(y$quality ~ y$residual.sugar, data = y)
summary(true_model)
true_model <- lm(y$quality ~ y$chlorides, data = y)
summary(true_model)
true_model <- lm(y$quality ~ y$free.sulfur.dioxide, data = y)
summary(true_model)
true_model <- lm(y$quality ~ y$total.sulfur.dioxide, data = y)
summary(true_model)
true_model <- lm(y$quality ~ y$density, data = y) # IN
summary(true_model)
true_model <- lm(y$quality ~ y$pH, data = y)
summary(true_model)
true_model <- lm(y$quality ~ y$sulphates, data = y) # IN
summary(true_model)
true_model <- lm(y$quality ~ y$alcohol, data = y) # IN
summary(true_model)
```

Model selected:

```r
true_model <- lm(y$quality ~ y$volatile.acidity + y$density + y$sulphates + y$alcohol)
summary(true_model)
```

```
##
## Call:
## lm(formula = y$quality ~ y$volatile.acidity + y$density + y$sulphates +
##     y$alcohol)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7044 -0.6221  0.2916  0.4004  2.4459
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         40.98558   13.64525   3.004  0.00271 **
## y$volatile.acidity   0.01164    0.06390   0.182  0.85548
## y$density          -35.32962   13.35005  -2.646  0.00822 **
## y$sulphates          0.10398    0.10304   1.009  0.31307
## y$alcohol           -0.04096    0.26037  -0.157  0.87500
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8029 on 1435 degrees of freedom
## Multiple R-squared:  0.00641,    Adjusted R-squared:  0.00364
## F-statistic: 2.314 on 4 and 1435 DF,  p-value: 0.05551
```

PART 7

Apply fits to test data

Transform variables and standardize transformed variables for use in Ridge and LASSO

```
# Tranform variables
y = test
y[ ,1] = log(y[ ,1])
y[ ,2] = log(y[ ,2])
y[ ,3] = sqrt(y[ ,3])
y[ ,4] = log(log(y[ ,4]))
y[ ,5] = log(y[ ,5])
y[ ,6] = log(y[ ,6])
y[ ,7] = log(y[ ,7])
y[ ,9] = log(y[ ,9])
y[ ,10] = log(y[ ,10])
y[ ,11] = log(y[ ,11])

r = y
for (i in 1:11) {
  var <- normalize(y[ ,i], method = "standardize")
  r[ ,i] <- var
}

x <- as.matrix(r[,1:11])
y1 <- as.matrix(r[,12])
```

Apply fits

```
# Sample mean of the response variable
mean(y[ ,12])
```

```
## [1] 5.773585
```

```
sum((y$quality - mean(y[ ,12]))^2)
```

```
## [1] 107.8491
```

```
# Full least-squares regression with untransformed variables
test_var <- as.matrix(test[,1:11])
model <- lm(test[ ,12] ~ test_var)
summary(model)
```

```
##
## Call:
## lm(formula = test[, 12] ~ test_var)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.77770 -0.37090  0.01671  0.37939  1.60345
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -6.363766  52.806893  -0.121   0.9042
## test_varfixed.acidity          0.046555   0.066781   0.697   0.4868
## test_varvolatile.acidity      -1.642554   0.346388  -4.742 4.96e-06 ***
## test_varcitric.acid          -0.542200   0.440100  -1.232   0.2199
## test_varresidual.sugar       -0.028757   0.063146  -0.455   0.6495
## test_varchlorides            -0.794537   1.067597  -0.744   0.4579
## test_varfree.sulfur.dioxide   0.004702   0.006465   0.727   0.4683
## test_vartotal.sulfur.dioxide -0.001359   0.002146  -0.633   0.5277
## test_vardensity              10.162898  53.754971   0.189   0.8503
## test_varpH                   -0.523044   0.508746  -1.028   0.3056
## test_varsulphates             0.595052   0.351696   1.692   0.0928 .
## test_varalcohol               0.390816   0.074005   5.281 4.54e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.605 on 147 degrees of freedom
## Multiple R-squared:  0.501,  Adjusted R-squared:  0.4637
## F-statistic: 13.42 on 11 and 147 DF,  p-value: < 2.2e-16
```

```
sum((y$quality - fitted(model))^2)
```

```
## [1] 53.81239
```

```
# Full least-squares regression with untransformed variables
y_var <- as.matrix(y[,1:11])
model <- lm(y[ ,12] ~y_var)
summary(model)
```

```
##
## Call:
## lm(formula = y[, 12] ~ y_var)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.04478 -0.34238  0.00621  0.38018  1.54241
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -36.70750   58.86631  -0.624 0.533874
## y_varfixed.acidity         0.13654    0.58051   0.235 0.814378
## y_varvolatile.acidity     -0.68974    0.18165  -3.797 0.000214 ***
## y_varcitric.acid          -0.28540    0.34130  -0.836 0.404391
## y_varresidual.sugar       -0.11140    0.16774  -0.664 0.507642
## y_varchlorides            -0.22004    0.15789  -1.394 0.165533
## y_varfree.sulfur.dioxide   0.08743    0.10902   0.802 0.423871
## y_vartotal.sulfur.dioxide -0.09432    0.12276  -0.768 0.443535
## y_vardensity              34.50081   59.62360   0.579 0.563715
## y_varpH                   -1.95062    1.71576  -1.137 0.257435
## y_varsulphates             0.65652    0.26614   2.467 0.014781 *
## y_varalcohol               4.09990    0.77523   5.289 4.38e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6102 on 147 degrees of freedom
## Multiple R-squared:  0.4925, Adjusted R-squared:  0.4545
## F-statistic: 12.97 on 11 and 147 DF,  p-value: < 2.2e-16
```

```
sum((y$quality - fitted(model))^2)
```

```
## [1] 54.73715
```

```
# Least-squares fit with the transformed variables that were "significant" (end of Part 5 above)
model <- lm(y$quality ~ y$volatile.acidity + y$density + y$sulphates + y$alcohol)
summary(model)
```

```
##
## Call:
## lm(formula = y$quality ~ y$volatile.acidity + y$density + y$sulphates +
##     y$alcohol)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.91028 -0.33634  0.01923  0.36970  1.64454
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -12.4485    32.0334  -0.389  0.69810
## y$volatile.acidity    -0.7073     0.1473  -4.801 3.71e-06 ***
## y$density              8.9318    31.4287   0.284  0.77664
## y$sulphates            0.6295     0.2357   2.670  0.00839 **
## y$alcohol              3.8798     0.5505   7.047 5.74e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6091 on 154 degrees of freedom
## Multiple R-squared:  0.4703, Adjusted R-squared:  0.4565
## F-statistic: 34.18 on 4 and 154 DF,  p-value: < 2.2e-16
```

```
sum((y$quality - fitted(model))^2)
```

```
## [1] 57.13174
```

```
# Least-squares fit with the transformed variables selected by Mallow's Cp
model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,9] + y[ ,10] + y[ ,7] +y[ ,8])
summary(model)
```

```
##
## Call:
## lm(formula = y[, 12] ~ y[, 11] + y[, 2] + y[, 10] + y[, 9] +
##     y[, 10] + y[, 7] + y[, 8])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.93501 -0.32531  0.01297  0.33872  1.60789
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.06009   32.52248  -0.186   0.8524
## y[, 11]      4.00102    0.57376   6.973 8.87e-11 ***
## y[, 2]      -0.65767    0.15288  -4.302 3.02e-05 ***
## y[, 10]      0.57878    0.24188   2.393   0.0179 *
## y[, 9]      -1.32466    1.10142  -1.203   0.2310
## y[, 7]      -0.02428    0.07096  -0.342   0.7327
## y[, 8]       3.92864   31.77925   0.124   0.9018
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6099 on 152 degrees of freedom
## Multiple R-squared:  0.4757, Adjusted R-squared:  0.455
## F-statistic: 22.99 on 6 and 152 DF,  p-value: < 2.2e-16
```

```
sum((y$quality - fitted(model))^2)
```

```
## [1] 56.54314
```

```
# Least-squares fit with the transformed variables selected by AIC
model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,1] + y[ ,6] + y[ ,10] + y[ ,8])
summary(model)
```

```
##
## Call:
## lm(formula = y[, 12] ~ y[, 11] + y[, 2] + y[, 1] + y[, 6] + y[,
##     10] + y[, 8])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.91157 -0.33955  0.00426  0.34993  1.66244
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  20.36373   42.15066   0.483  0.62971
## y[, 11]       3.73979    0.56798   6.584 7.00e-10 ***
## y[, 2]       -0.64280    0.15818  -4.064 7.72e-05 ***
## y[, 1]        0.44760    0.36831   1.215  0.22615
## y[, 6]        0.02463    0.06435   0.383  0.70243
## y[, 10]       0.62438    0.23628   2.643  0.00909 **
## y[, 8]      -24.62533   42.03831  -0.586  0.55889
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6099 on 152 degrees of freedom
## Multiple R-squared:  0.4757, Adjusted R-squared:  0.455
## F-statistic: 22.99 on 6 and 152 DF,  p-value: < 2.2e-16
```

```
sum((y$quality - fitted(model))^2)
```

```
## [1] 56.54321
```

```
# Least-squares fit with the transformed variables selected by BIC
model <- lm(y[ ,12] ~ y[ ,11] + y[ ,2] + y[ ,10] + y[ ,5] + y[ ,7] + y[ ,1] + y[ ,6])
summary(model)
```

```
##
## Call:
## lm(formula = y[, 12] ~ y[, 11] + y[, 2] + y[, 10] + y[, 5] +
##     y[, 7] + y[, 1] + y[, 6])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.99353 -0.33915 -0.00093  0.40923  1.57423
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.84932    1.47229  -2.615  0.00984 **
## y[, 11]      3.56632    0.51793   6.886 1.45e-10 ***
## y[, 2]      -0.62963    0.15472  -4.069 7.57e-05 ***
## y[, 10]      0.75589    0.24907   3.035  0.00283 **
## y[, 5]      -0.22213    0.15148  -1.466  0.14463
## y[, 7]      -0.11357    0.11752  -0.966  0.33540
## y[, 1]       0.36255    0.27906   1.299  0.19587
## y[, 6]       0.09312    0.10669   0.873  0.38418
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6064 on 151 degrees of freedom
## Multiple R-squared:  0.4851, Adjusted R-squared:  0.4612
## F-statistic: 20.32 on 7 and 151 DF,  p-value: < 2.2e-16
```

```
sum((y$quality - fitted(model))^2)
```

```
## [1] 55.53341
```

```
# Ridge regression fit with the transformed variables
CV.ridge <- cv.glmnet(x, y1, alpha=0)
ridge <- coef(CV.ridge, s=CV.ridge$lambda.1se)
ridge
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)           5.773584906
## fixed.acidity         0.037072851
## volatile.acidity     -0.126567952
## citric.acid           0.036438969
## residual.sugar       -0.003755240
## chlorides            -0.051000426
## free.sulfur.dioxide   0.005505898
## total.sulfur.dioxide -0.036252855
## density              -0.056705398
## pH                   -0.014867087
## sulphates             0.094295695
## alcohol               0.174581876
```

```
fit <- CV.ridge$glmnet.fit
y_pred <- predict(fit, s=CV.ridge$lambda.1se, newx = x)
dim(y_pred)
```

```
## [1] 159   1
```

```
sum((y_pred - y$quality)^2)
```

```
## [1] 65.84177
```

```
# LASSO regression fit with the transformed variables
CV.lasso <- cv.glmnet(x, y1, alpha=1)
lasso <- coef(CV.lasso, s=CV.lasso$lambda.1se)
lasso
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)           5.7735849
## fixed.acidity         .
## volatile.acidity     -0.1548960
## citric.acid           .
## residual.sugar        .
## chlorides             .
## free.sulfur.dioxide   .
## total.sulfur.dioxide  .
## density               .
## pH                    .
## sulphates             0.0118399
## alcohol               0.2474522
```

```
fit <- CV.lasso$glmnet.fit
y_pred <- predict(fit, s=CV.lasso$lambda.1se, newx = x)
dim(y_pred)
```

```
## [1] 159   1
```

```
sum((y_pred - y$quality)^2)
```

```
## [1] 68.15063
```

```
# Least-squares fit with the transformed variables selected by iterative fitting
model <- lm(y$quality ~ y[ ,11] + y[ ,2] + y[ ,10])
summary(model)
```

```
##
## Call:
## lm(formula = y$quality ~ y[, 11] + y[, 2] + y[, 10])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.91205 -0.33692  0.00915  0.37525  1.63103
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.3501     1.0903  -3.073  0.00251 **
## y[, 11]       3.7976     0.4671   8.130 1.28e-13 ***
## y[, 2]       -0.7084     0.1468  -4.825 3.33e-06 ***
## y[, 10]       0.6370     0.2336   2.727  0.00713 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6073 on 155 degrees of freedom
## Multiple R-squared:   0.47,  Adjusted R-squared:  0.4597
## F-statistic: 45.81 on 3 and 155 DF,  p-value: < 2.2e-16
```

```
sum((y$quality - fitted(model))^2)
```

```
## [1] 57.1617
```

The sum of squared prediction errors for each model above will always decrease as more variables are added to the model. More conservative approaches exclude more variables to decrease the risk of inserting additional noise into the true model.

Part 8

Devise at least one more variation on the approaches listed above.

Apply ridge regression using the variables and terms that came from the iterative fitting process

```r
x <- as.matrix(data.frame(r[,11], r[,2], r[,10]))
y1 <- as.matrix(r[,12])

CV.ridge <- cv.glmnet(x, y1, alpha=0)
ridge <- coef(CV.ridge, s=CV.ridge$lambda.1se)
ridge
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept)  5.7735849
## r...11.      0.2210488
## r...2.      -0.1608769
## r...10.      0.1008876
```

```r
fit <- CV.ridge$glmnet.fit
y_pred <- predict(fit, s=CV.ridge$lambda.1se, newx = x)
dim(y_pred)
```

```
## [1] 159   1
```

```r
sum((y_pred - y$quality)^2)
```

```
## [1] 65.60499
```

Conclusion

Most models trialed above suggest that the three strongest predictors of wine quality are alcohol content, volatile acidity, and sulphates. Additional variables may contribute more noise than unique information into the model.