winemag-data-analysis

Kru123

2/9/2019

Wine Reviews Data Analysis

Introduction/overview/executive summary

The data set is from the Wine Reviews project on Kaggle (https://www.kaggle.com/zynicide/wine-reviews). The data is protected under https://creativecommons.org/licenses/by-nc-sa/4.0/. User zackthoutt scraped the data.

Columns in the data are:

(descriptions are from the original data posting on Kaggle)
* country - The country that the wine is from
* description
* designation - The vineyard within the winery where the grapes that made the wine are from
* points - The number of points WineEnthusiast rated the wine on a scale of 1-100 (though they say they only post reviews for wines that score >=80)
* price - The cost for a bottle of wine
* province - The province or state that the wine is from
* region_1 - The wine growing area in a province or state (ie Napa)
* region_2 - Sometimes there are more specific regions specified within a wine growing area (ie Rutherford inside the Napa Valley), but this value can sometimes be blank
* taster_name
* taster_twitter_handle
* title - The title of the wine review, which often contains the vintage if you're interested in extracting that feature
* variety - The type of grapes used to make the wine (ie Pinot Noir)
* winery - The winery that made the wine

The goal of my project was to build a model to predict the quality of wines based on the price, a description from a sommelier, the vintage and color of the wine.

The key steps in the project included data cleaning, exploration and visualization of the data. I also used 16 different models in the Caret package on a small data subset to determine the best regression model to use for building the final prediction model.

Methods/analysis

section that explains the process and techniques used, such as data cleaning, data exploration and visualization, any insights gained, and your modeling approach

Libraries used:

```
library(caret)
library(dplyr)
library(tidyverse)
```

Download the data:

```
dl <- tempfile()
download.file("http://kaffeinekard.net/data/winemag-data-130k-v2.csv", dl)
```

Load the data:

```
dat <- read.csv(dl, stringsAsFactors = FALSE)
rm(dl)
```

Look at a summary of the data:

```
summary(dat)
```

```
##        X            country          description        designation
##  Min.   :     0   Length:129971      Length:129971      Length:129971
##  1st Qu.: 32492   Class :character   Class :character   Class :character
##  Median : 64985   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 64985
##  3rd Qu.: 97478
##  Max.   :129970
##
##      points           price           province           region_1
##  Min.   : 80.00   Min.   :   4.00   Length:129971      Length:129971
##  1st Qu.: 86.00   1st Qu.:  17.00   Class :character   Class :character
##  Median : 88.00   Median :  25.00   Mode  :character   Mode  :character
##  Mean   : 88.45   Mean   :  35.36
##  3rd Qu.: 91.00   3rd Qu.:  42.00
##  Max.   :100.00   Max.   :3300.00
##                   NA's   :8996
##    region_2         taster_name        taster_twitter_handle
##  Length:129971      Length:129971      Length:129971
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##     title              variety             winery
##  Length:129971      Length:129971      Length:129971
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
```

We see that price has NA values that we want to remove because we want to predict points based on price.

```
dat_no_na <- dat[complete.cases(dat),]
# how much did we remove
(nrow(dat) - nrow(dat_no_na))/nrow(dat)*100
```

```
## [1] 6.921544
```

We want to use the description in the analysis. To do this I am looking at the number of words in the description as a measure of the description. Add the variable wordcount to the data set.

```
dat_no_na <- dat_no_na %>%
  mutate(wordcount = sapply(strsplit(description, " "), length))
```

Extract the year out of the title column since the vintage can be a predictor for the quality.

```
dat_no_na <- dat_no_na %>%
  extract(title, c("year"), regex = "([0-9]{4})", remove = F)
dat_no_na$year <- as.integer(dat_no_na$year)
```

I want to look only at wines from the 1900's and 200's and also some titles contain dates that are not the wine vintage. We can remove the records.

```
# filter older than current year and younger than 1900 because obvious date extract issues
dat_no_na <- dat_no_na %>%
  filter(year>1900 & year <2019)
dat_no_na <- dat_no_na[complete.cases(dat_no_na),]
```

To make sure we look only at wines with enough reviews we are limiting the data set to only wines with more than 100 reviews.

```
dat_no_na <- dat_no_na %>%
  group_by(variety) %>%
  filter(n() > 100) %>%
  ungroup(variety)
```

Some countries do not have many wine reviews and to have enough variation we filter the data to include only records from countries with more than 1000 reviews.

```
dat_country <- dat_no_na %>%
  group_by(country) %>%
  filter(n()>=1000) %>%
  ungroup(country)
```

Because there could be variation in quality based on where the wine originates and the color of the wine we are going to add the continent and the color of the wine.

```
# Use the country to determine continent

americas <- c("Argentina","Brazil","Canada","Chile","Mexico","Peru",
              "Uruguay","US")
africas <- c("Morocco","South Africa")
australias <- c("Australia","New Zealand")
europes <- c("Armenia","Austria","Bulgaria","Croatio","Cyprus","Czech Republic",
             "England","France","Georgia","Germany","Greece","Hungary","Italy",
             "Luxembourg","Macedonia","Moldova","Portugal","Romania",
             "Serbia","Slovakia","Spain","Switzerland","Turkey","Ukraine")
asias <- c("India","Israel","Lebanon")

dat_country <- dat_country %>%
  mutate(continent = ifelse(country %in% americas,"Ame",
                            ifelse(country %in% africas,"Afr",
                                   ifelse(country %in% australias, "Aus",
                                          ifelse(country %in% europes, "Eur",
                                                 ifelse(country %in% asias, "Asi","Unk"))))))
# do we have any unclassified countries?
dat_country[dat_country$continent=="Unk",13]
```

```
## # A tibble: 0 x 1
## # ... with 1 variable: year <int>
```

```
dat_country$continent <- as.factor(dat_country$continent)

# Use the variety to classify the wines into red or white.

reds <- c("Aglianico","Barbera","Blaufränkisch","Bonarda","Bordeaux-style Red Blend",
          "Cabernet Franc","Cabernet Sauvignon","Cabernet Sauvignon-Merlot",
          "Cabernet Sauvignon-Syrah","Carmenère","Champagne Blend",
          "Corvina, Rondinella, Molinara","Dolcetto","G-S-M","Gamay",
          "Garnacha","Grenache","Malbec","Mencía","Meritage","Merlot",
          "Monastrell","Montepulciano","Mourvèdre","Nebbiolo","Nerello Mascalese",
          "Nero d'Avola","Petit Verdot","Petite Sirah","Pinot Nero","Pinot Noir",
          "Pinotage","Port","Portuguese Red","Primitivo","Red Blend",
          "Rhône-style Red Blend","Sangiovese","Sangiovese Grosso","Shiraz",
          "Syrah","Tannat","Tempranillo","Tempranillo Blend","Tinta de Toro",
          "Touriga Nacional","Zinfandel","Zweigelt")
whites <- c("Albariño","Alvarinho","Bordeaux-style White Blend","Chardonnay",
            "Chenin Blanc","Fiano","Friulano","Garganega","Gewürztraminer",
            "Glera","Greco","Grenache Blanc","Grillo","Grüner Veltliner",
            "Melon","Moscato","Muscat","Pinot Bianco","Pinot Blanc",
            "Pinot Grigio","Pinot Gris","Portuguese White",
            "Rhône-style White Blend","Riesling","Rosé","Roussanne",
            "Sauvignon","Sauvignon Blanc","Sémillon","Sparkling Blend",
            "Torrontés","Turbiana","Verdejo","Verdicchio","Vermentino",
            "Vernaccia","Viognier","Viura","White Blend")

dat_country <- dat_country %>%
  mutate(color = ifelse(variety %in% reds,"Red",
                        ifelse(variety %in% whites, "White","Unk")))

# did we miss any?
dat_country[dat_country$color=="Unk",14]
```

```
## # A tibble: 0 x 1
## # ... with 1 variable: variety <chr>
```

```
dat_country$color <- as.factor(dat_country$color)
```

At this point I was wondering if the specific flavors used by a sommelier in describing the wine could be a predictor. I used the flavors that are used to describe red and white wines from the website winefolly (https://winefolly.com/review/identifying-flavors-in-wine/) to create a list of words that can be used to give a score to the description based on the number of words matching the list. This score was added as column winewords.

```
sommelier_words <- c("blackberry","black currant","marionberry","black plum","blueberry",
                     "black cherry","black raspberry","acai","jam","prune","fig",
                     "black raisins","cranberry","pomegranate","red currant",
                     "bing cherry","strawberry","cherry","raspberry","red plum",
                     "goji berry","dragon fruit","candied cherries","candied berries",
                     "apricot","peach","white peach","nectarine","apple","pear",
                     "red grapefruit","orange","pink grapefruit",
                     "passion fruit","lemon","lime","pineapple")
```

```
wcount <- sapply(sommelier_words, function(sw){
  str_count(dat_country$description,sw)
})

dat_country <- dat_country %>%
  mutate(winewords = rowSums(wcount))
```

The taster column is sparsely populated with only 18 unique tasters and over 22,000 blank values and thus not much help.

```
# taster is sparsely populated
dat_country %>%
  summarise(n_user = n_distinct(taster_name),
            n_wines = n_distinct(title))
```

```
## # A tibble: 1 x 2
##   n_user n_wines
##    <int>   <int>
## 1     18   99243
```

Other columns like designation, winery and province have great variance with only one or two wines in many cases. We will not use these columns.

Create the cleaned data set and remove data sets and other objects that are not needed. At this point I will also change the order of the resulting columns to group the numerical and categorical variables together and have the results column on the right.

```
wines <- dat_country %>%
  select(-X,-country,-description,-designation,-province,-region_1,-region_2,
         -taster_name,-taster_twitter_handle,-title,-variety,-winery)

rm(dat,dat_country,dat_no_na,africas,americas,asias,
   australias,europes,reds,whites, sommelier_words,wcount)

#reorder columns
wines <- wines[c("price","year","wordcount","winewords","continent","color","points")]
```

Data exploration

Now that we have the final data set we can do data exploration and visualization.

```
#dimensions of dataset
dim(wines)
```

```
## [1] 107919      7
```

```
# type for each attribute
sapply(wines,class)
```

```
##     price      year wordcount winewords continent     color    points
## "numeric" "integer" "integer" "numeric"  "factor"  "factor" "integer"
```

Have a look at the data.

```
head(wines)
```

```
## # A tibble: 6 x 7
##   price  year wordcount winewords continent color points
```

```
##   <dbl> <int>     <int>     <dbl> <fct>     <fct>  <int>
## 1    15  2011        39         0 Eur       Red       87
## 2    14  2013        28         3 Ame       White     87
## 3    13  2013        33         3 Ame       White     87
## 4    65  2012        41         0 Ame       Red       87
## 5    24  2012        21         0 Eur       White     87
## 6    12  2013        24         1 Eur       White     87
```

And look at the summary statistics for each column.

```
summary(wines)
```

```
##      price              year        wordcount        winewords
##  Min.   :   4.00   Min.   :1904   Min.   :  3.00   Min.   :0.000
##  1st Qu.:  17.00   1st Qu.:2009   1st Qu.: 33.00   1st Qu.:0.000
##  Median :  26.00   Median :2011   Median : 40.00   Median :1.000
##  Mean   :  36.09   Mean   :2011   Mean   : 40.69   Mean   :1.255
##  3rd Qu.:  44.00   3rd Qu.:2013   3rd Qu.: 47.00   3rd Qu.:2.000
##  Max.   :3300.00   Max.   :2017   Max.   :135.00   Max.   :9.000
##  continent     color           points
##  Afr: 1234   Red  :69277   Min.   : 80.0
##  Ame:59654   White:38642   1st Qu.: 86.0
##  Aus: 3404                 Median : 88.0
##  Eur:43627                 Mean   : 88.5
##                            3rd Qu.: 91.0
##                            Max.   :100.0
```

Levels of the categorical variables.

```
levels(wines$continent)
```

```
## [1] "Afr" "Ame" "Aus" "Eur"
```

```
levels(wines$color)
```

```
## [1] "Red"   "White"
```

Frequency tables for the categorical variables.

```
percentage <- prop.table(table(wines$color))*100
cbind(freq=table(wines$color), percentage=percentage)
```

```
##        freq percentage
## Red   69277   64.19352
## White 38642   35.80648
```

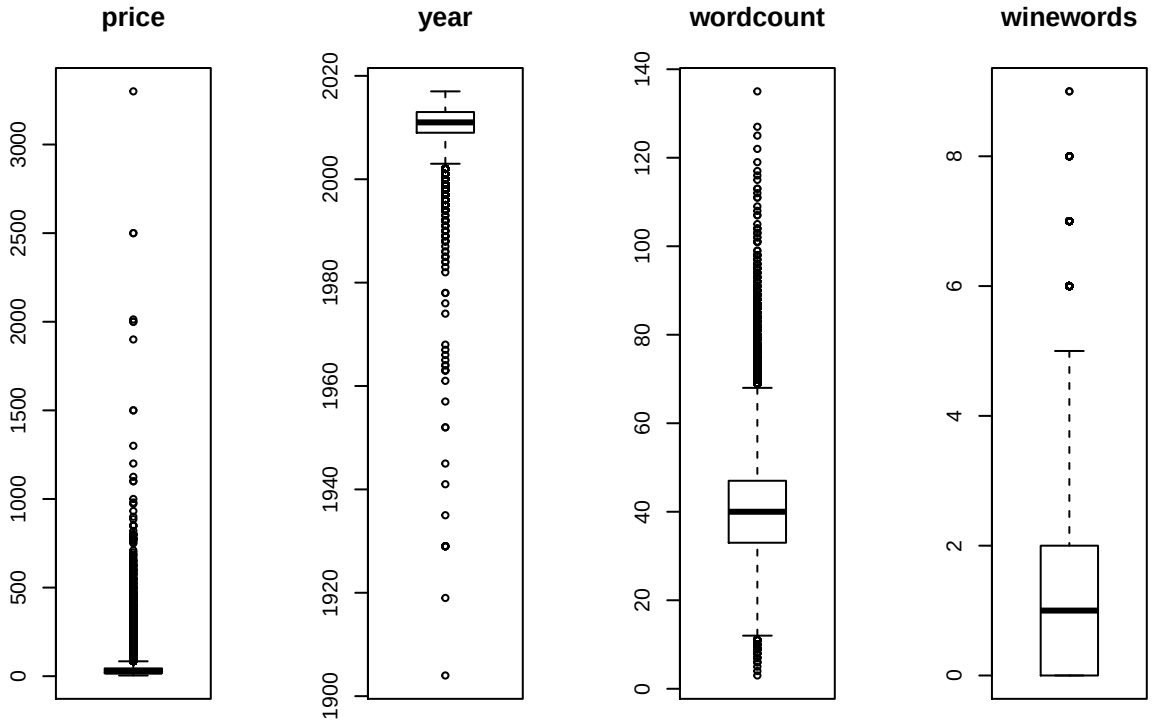
```
percentage <- prop.table(table(wines$continent))*100
cbind(freq=table(wines$continent), percentage=percentage)
```

```
##      freq percentage
## Afr  1234   1.143450
## Ame 59654  55.276643
## Aus  3404   3.154218
## Eur 43627  40.425690
```

Visualize the data

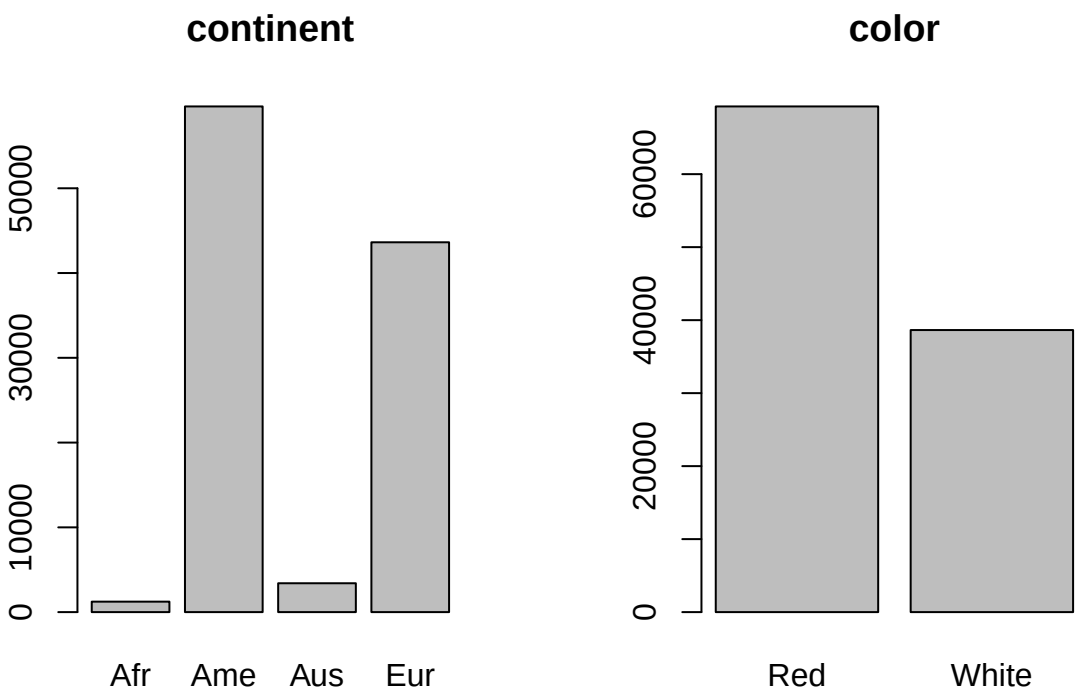
Univariate plots for numerical variables:

```
# univariate plots
x_num<-wines[,1:4]
x_fac<-wines[,5:6]
y<-wines[,7]

#numerical input ditribution

par(mfrow=c(1,4))
for (i in 1:4) {
  boxplot(x_num[,i], main=names(wines)[i])
}
```



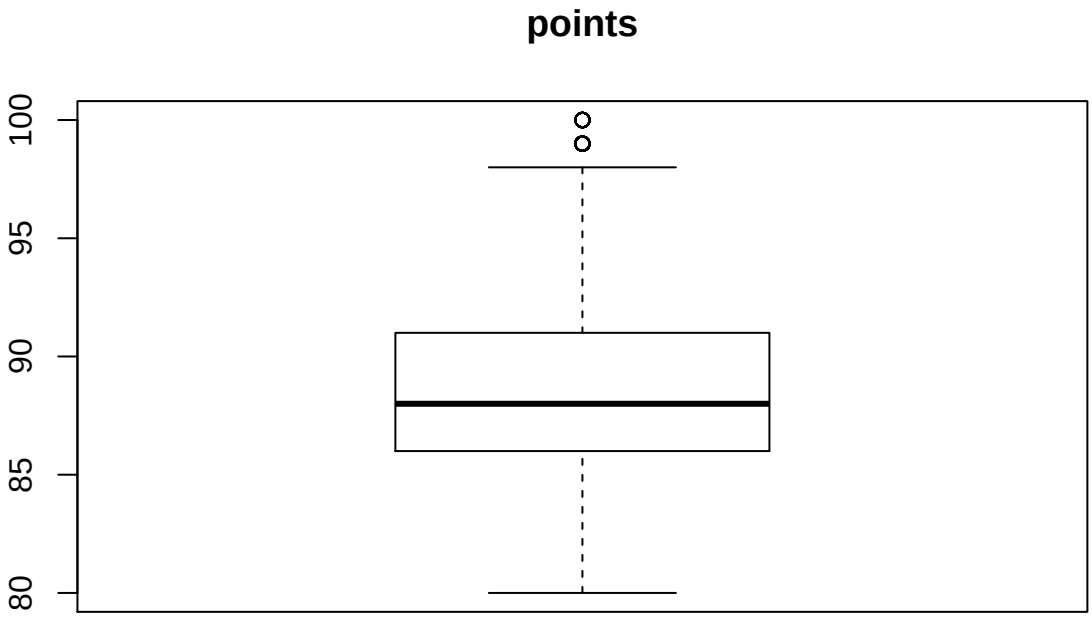
Univariate plots for the factor inputs:

```
#factor input distribution
par(mfrow=c(1,2))
for (i in 1:2) {
  plot(x_fac[,i], main=names(wines)[i+4])
}
```



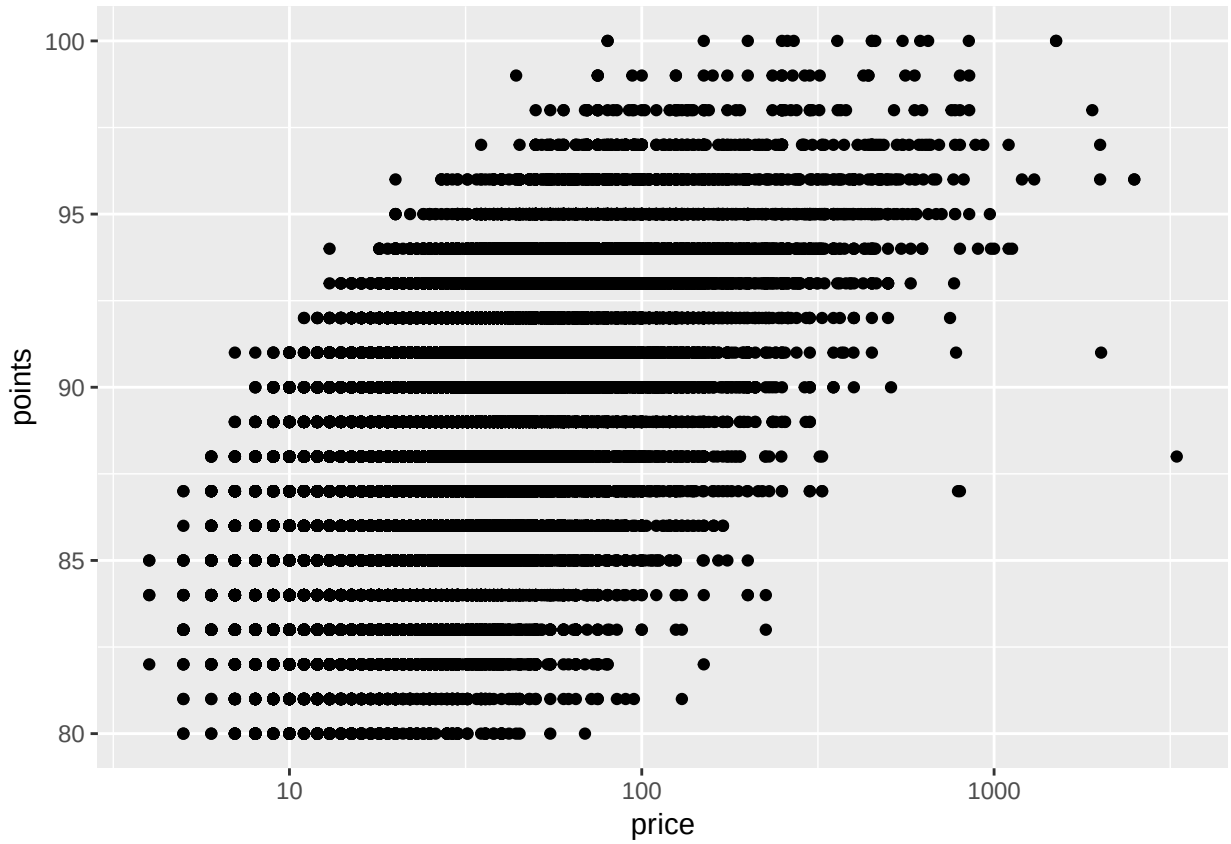
Distribution of the output variable or points:

```
# output distribution
boxplot(y,main=names(wines)[7])
```

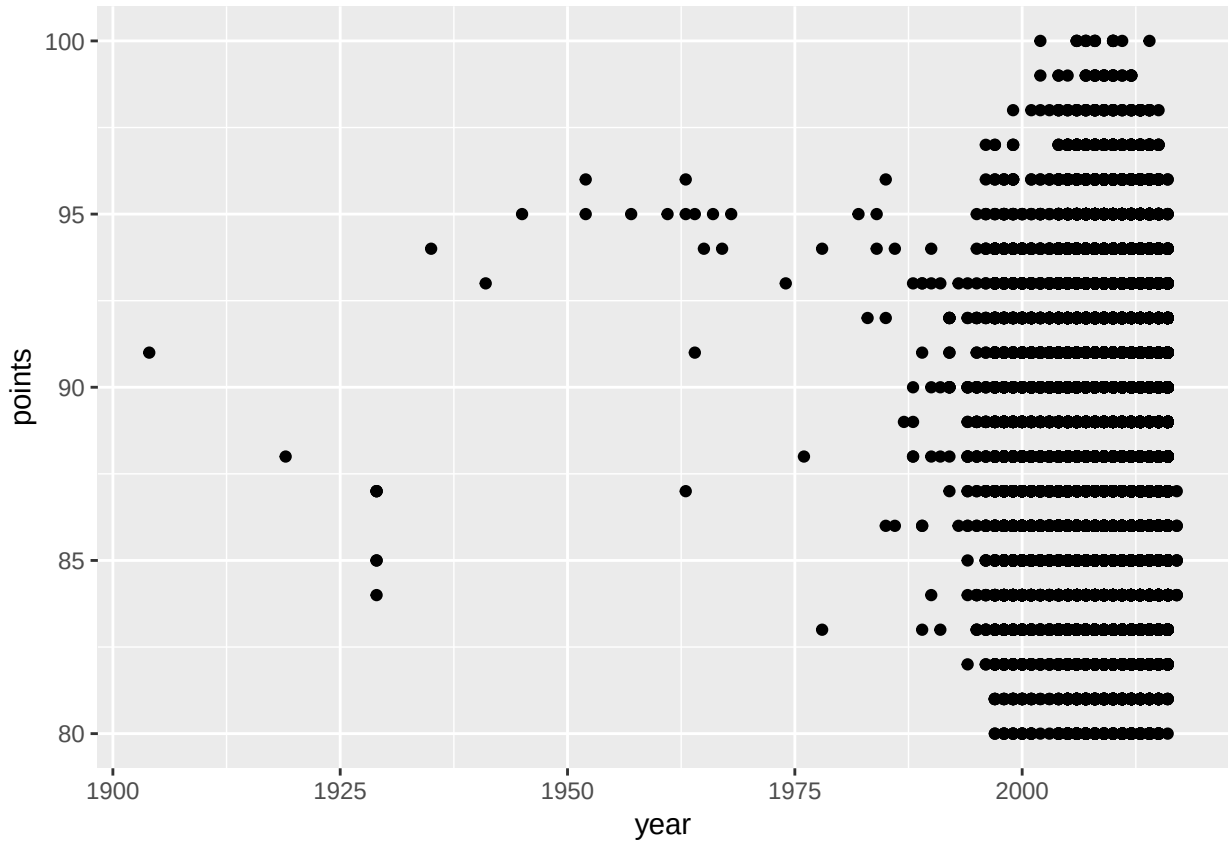


Multivariate plots:

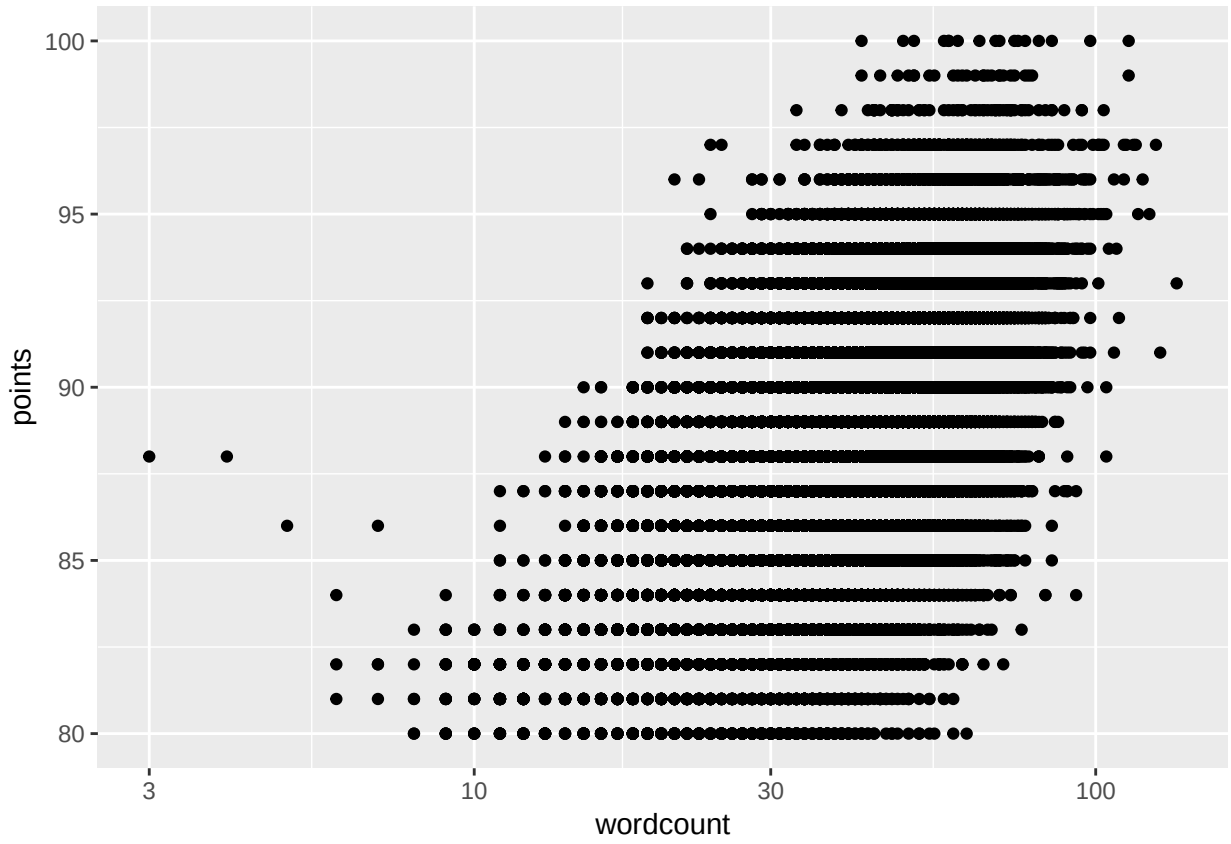
```
# price vs points
wines %>%
  ggplot(aes(x=price,y=points)) +
  geom_point() +
  scale_x_log10()
```



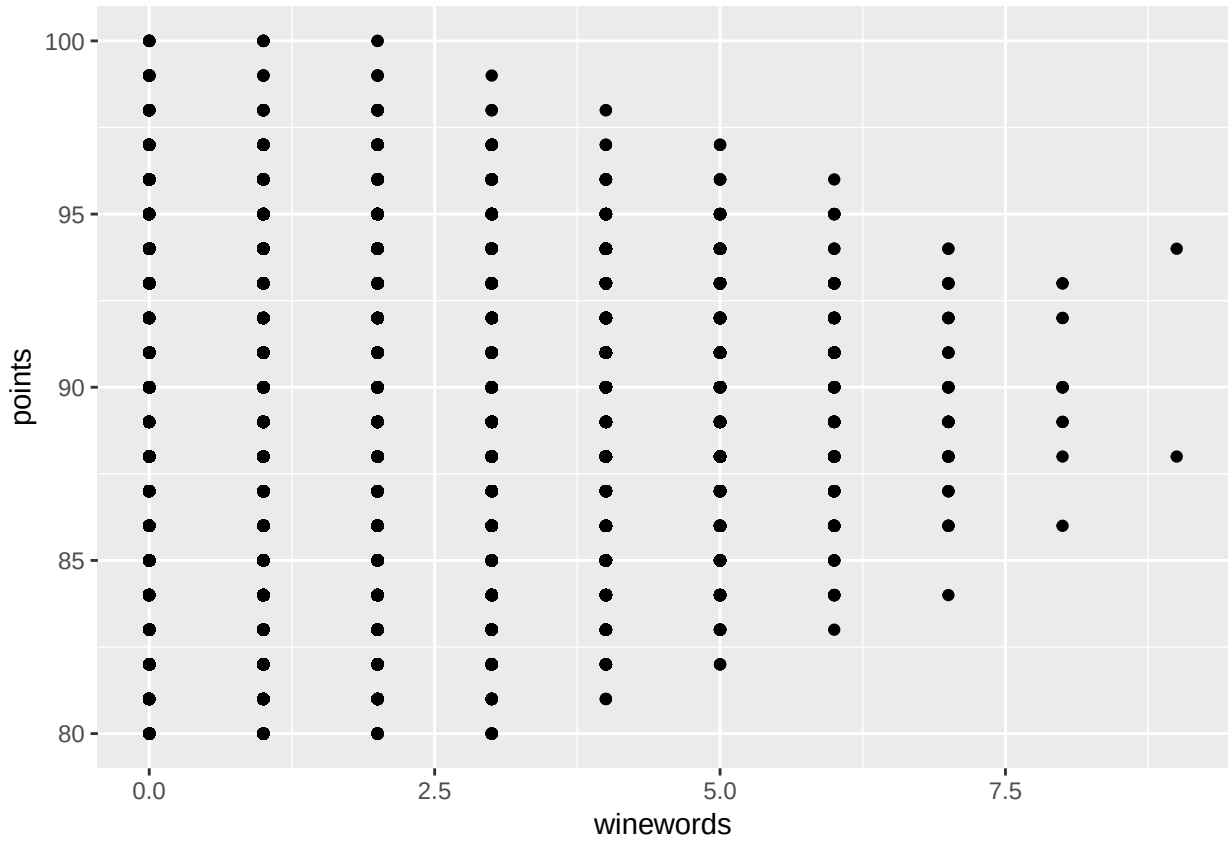

```
# year vs points
wines %>%
  ggplot(aes(x=year,y=points)) +
  geom_point()
```



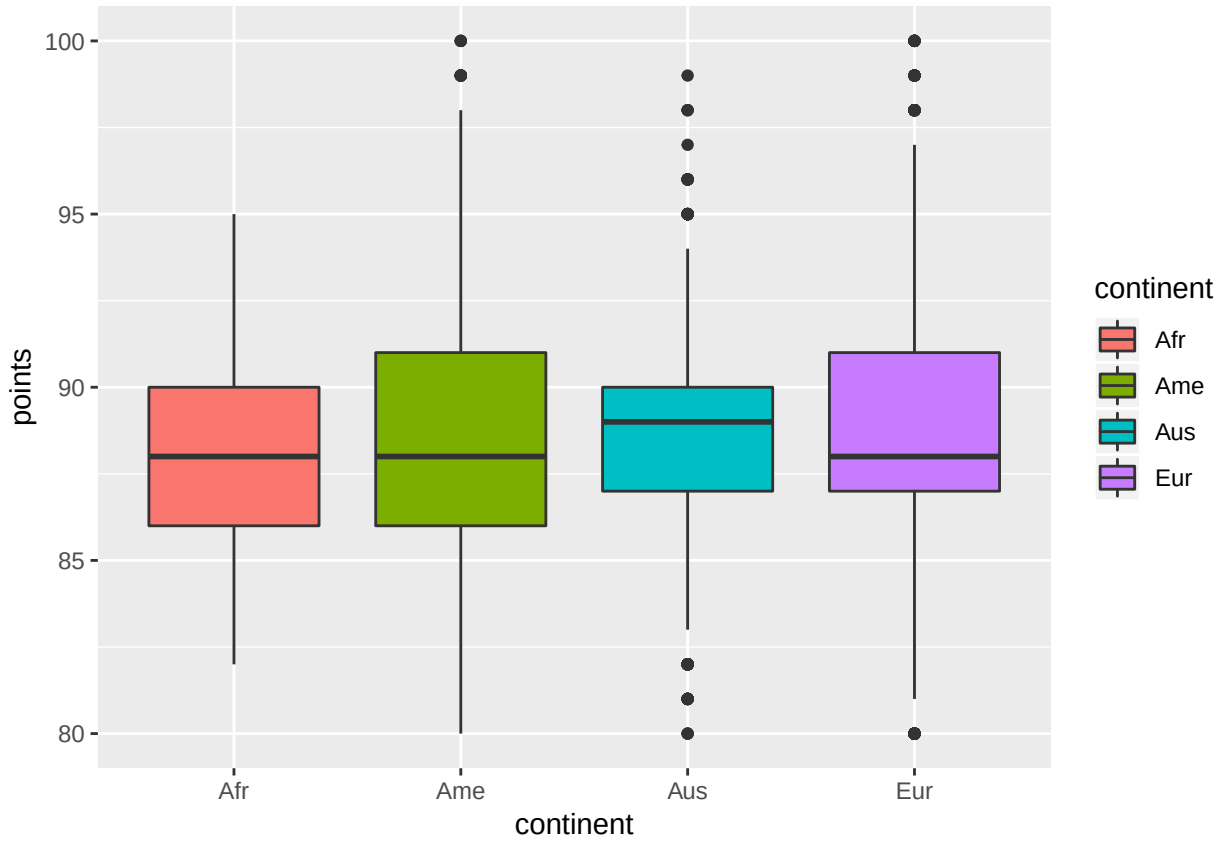

```
# wordcount vs points
wines %>%
  ggplot(aes(x=wordcount,y=points)) +
  geom_point() +
  scale_x_log10()
```



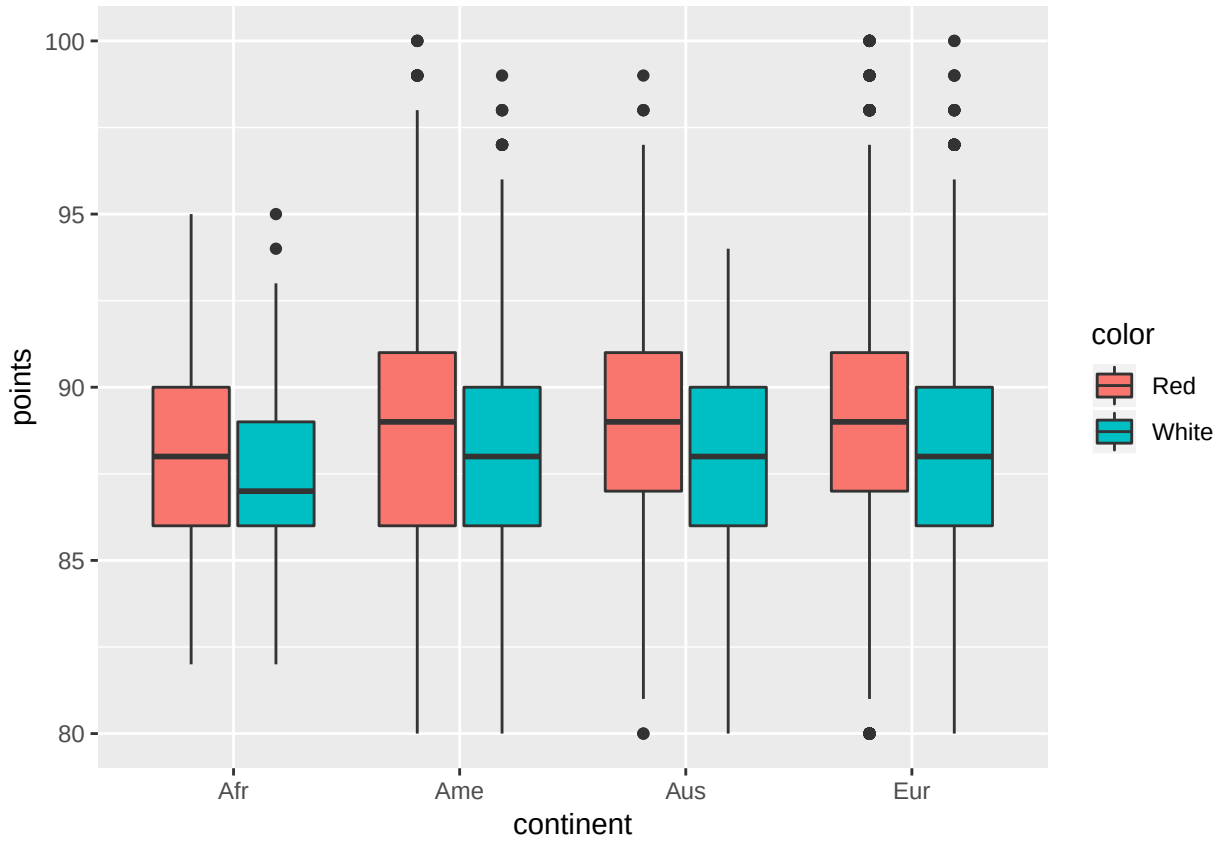

```
# winewords vs points
wines %>%
  ggplot(aes(x=winewords,y=points)) +
  geom_point()
```



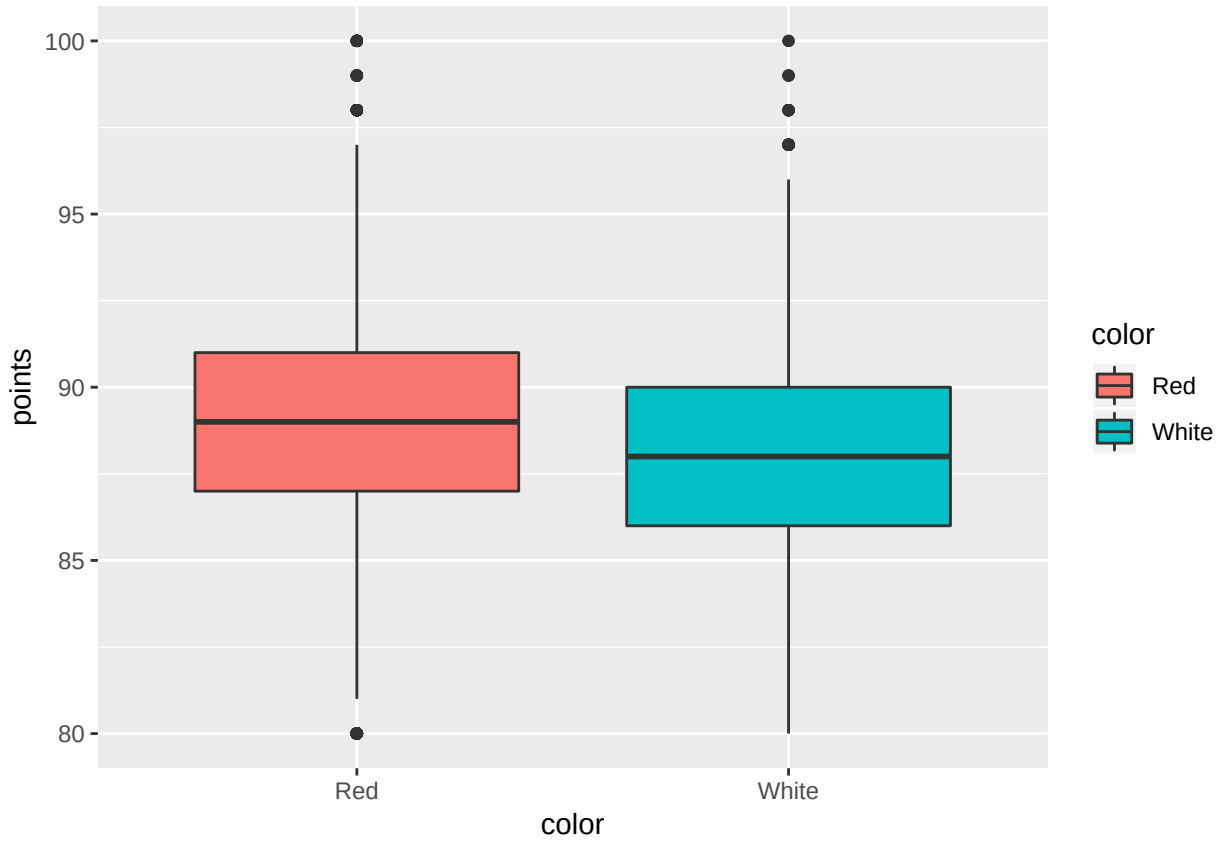

```
# continent vs points
wines %>%
  ggplot(aes(x=continent,y=points,fill=continent)) +
  geom_boxplot()
```



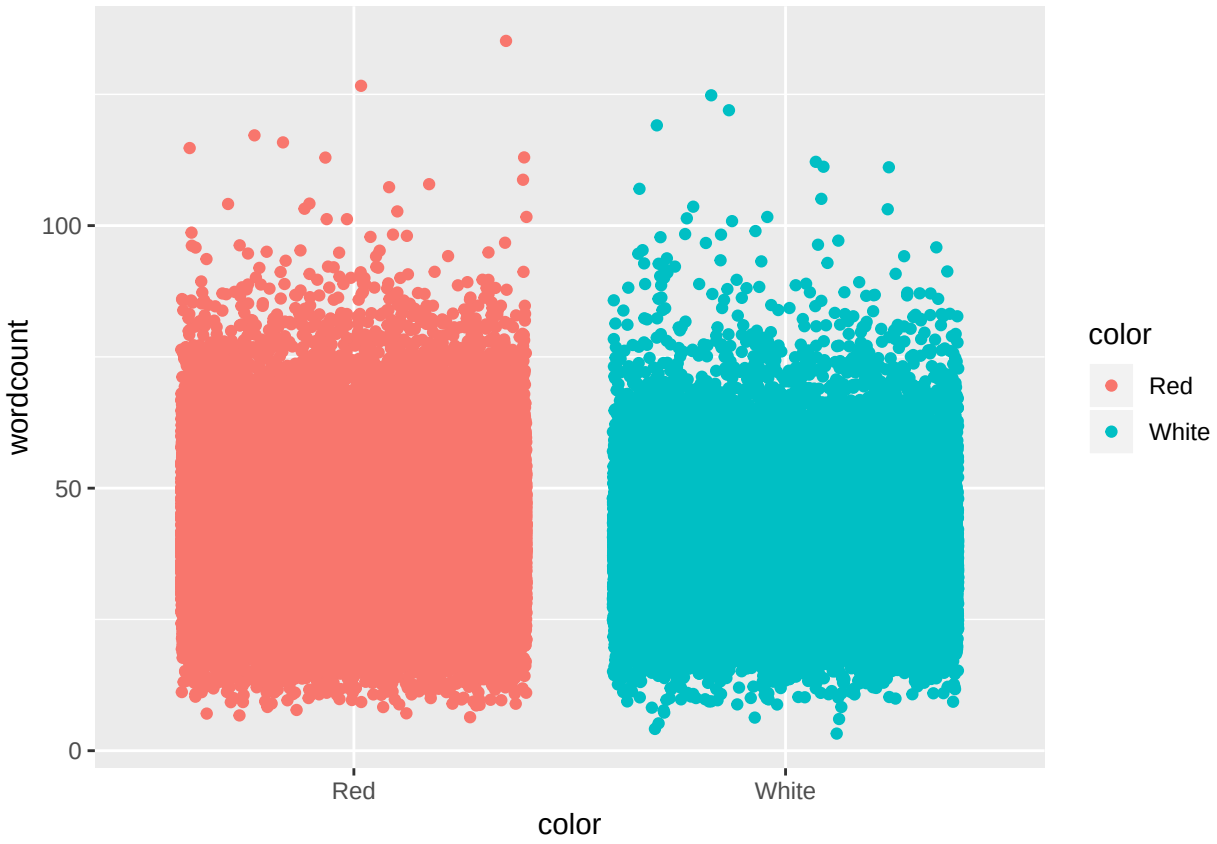

```
# continent vs points for each color
wines %>%
  ggplot(aes(x=continent,y=points,fill = color)) +
  geom_boxplot()
```




```
# color vs points
wines %>%
  ggplot(aes(x=color,y=points,fill=color)) +
  geom_boxplot()
```




```
# color vs wordcount for fun
wines %>%
  ggplot(aes(x=color,y=wordcount, col=color)) +
  geom_jitter()
```



Correlation between predictors and points:

```
# price and points
cor(log(wines$price),wines$points)
```

```
## [1] 0.6150412
```

```
# year and points
cor(wines$year,wines$points)
```

```
## [1] 0.06477017
```

```
# wordcount and points
cor(wines$wordcount,wines$points)
```

```
## [1] 0.5380795
```

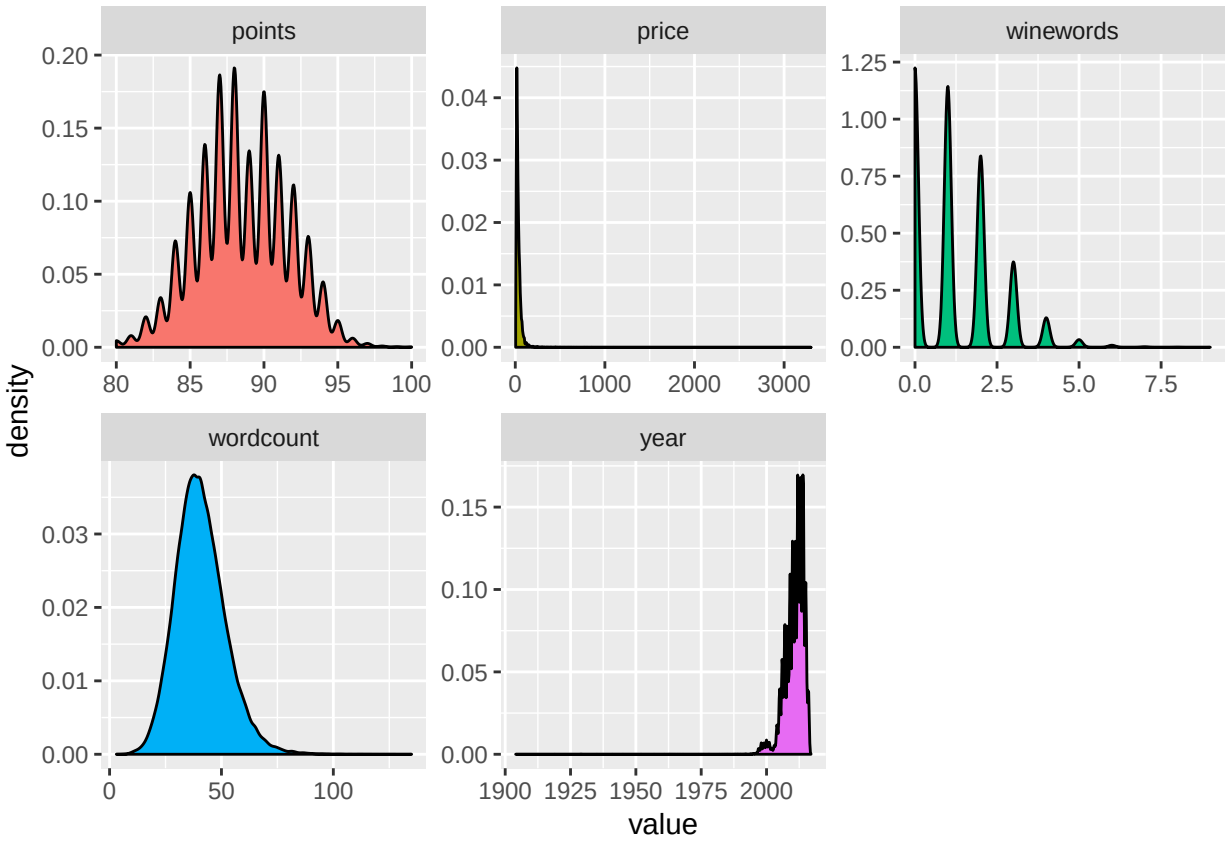
```
# winewords and points
cor(wines$winewords,wines$points)
```

```
## [1] 0.08772956
```

We see that there are positive correlations between prica and points and also worcount and points.

Density plots of the variables:

```
wines %>%
  select_if(is.numeric) %>%
  gather(metric, value) %>%
  ggplot(aes(value, fill = metric)) +
  geom_density(show.legend = FALSE) +
  facet_wrap( ~ metric, scales = "free")
```



Results

Set up the train and test set with the test set (wine-test) having a size of 10 percent of the data.

```
#set up test and train
set.seed(123)
test_index <- createDataPartition(y = wines$points, times = 1, p=0.1, list = FALSE)

wine_train <- wines[-test_index,]
temp <- wines[test_index,]

## Make sure variety and year in test set is in train set as well
wine_test <- temp %>%
  semi_join(wine_train, by = "continent") %>%
  semi_join(wine_train, by = "price") %>%
  semi_join(wine_train, by = "year") %>%
  semi_join(wine_train, by = "color")

# add rows removed from test set back to train set
removed <- anti_join(temp, wine_test)
```

```
## Joining, by = c("price", "year", "wordcount", "winewords", "continent", "color", "points")
```

```
wine_train <- rbind(wine_train, removed)

rm(temp, removed)
```

```
#summarize dataset
summary(wine_train)
```

```
##      price              year        wordcount        winewords
##  Min.   :   4.00   Min.   :1904   Min.   :  3.00   Min.   :0.000
##  1st Qu.:  17.00   1st Qu.:2009   1st Qu.: 33.00   1st Qu.:0.000
##  Median :  26.00   Median :2011   Median : 40.00   Median :1.000
##  Mean   :  36.13   Mean   :2011   Mean   : 40.69   Mean   :1.252
##  3rd Qu.:  44.00   3rd Qu.:2013   3rd Qu.: 47.00   3rd Qu.:2.000
##  Max.   :3300.00   Max.   :2017   Max.   :135.00   Max.   :9.000
##  continent     color           points
##  Afr: 1127   Red  :62343   Min.   : 80.0
##  Ame:53731   White:34801   1st Qu.: 86.0
##  Aus: 3035                 Median : 88.0
##  Eur:39251                 Mean   : 88.5
##                            3rd Qu.: 91.0
##                            Max.   :100.0
```

To reduce the time to run all 16 models create wines_mini, a small subset with 100 records of wine_train so that we can run all the models to find the best model before we run it on the large train set.

```
# small test to test models and get timing
set.seed(123)
wines_mini <- sample_n(wine_train,1000)
```

Build all the models on wines_mini to see if they complete and determine the best model to use for the final build.

models <- c("lm","svmLinear", "gamLoess", "knn", "kknn", "gam", "rf", "ranger", "Rborist", "avNNet", "mlp", "monmlp", "gbm", "svmRadial", "svmRadialCost", "svmRadialSigma")

fits <- lapply(models, function(model){ print(model) fit <- train(points ~ ., method = model, data = wines_mini)

})

names(fits) <- models

Write the resulting RMSE's to a recordset and select the one with the lowest RMSE for the final model.

RMSE results of the different models:

```
rmse_results %>% knitr::kable()
```

method	RMSE
lm	2.334127
svmLinear	2.457581
gamLoess	210.555892
knn	2.400092
knn	2.322125
knn	2.283098
kknn	2.845184
kknn	2.841667
kknn	2.841667
gam	2.150196
gam	2.164980
rf	2.170066
rf	2.195239

method	RMSE
rf	2.231265
ranger	2.187967
ranger	2.342305
ranger	2.214771
ranger	2.161846
ranger	2.259709
ranger	2.175336
Rborist	2.222455
Rborist	2.247935
Rborist	2.293573
avNNet	87.605941
avNNet	87.605941
avNNet	87.605949
avNNet	87.605941
avNNet	87.605941
avNNet	87.605946
avNNet	87.605941
avNNet	87.605941
avNNet	87.605945
mlp	51.038800
mlp	13.660301
mlp	3.538297
monmlp	2.157610
monmlp	2.182061
monmlp	2.263069
gbm	2.193605
gbm	2.137137
gbm	2.134497
gbm	2.133590
gbm	2.135757
gbm	2.154720
gbm	2.137419
gbm	2.150886
gbm	2.173749
svmRadial	2.278774
svmRadial	2.270679
svmRadial	2.288804
svmRadialCost	2.287768
svmRadialCost	2.275083
svmRadialCost	2.286627
svmRadialSigma	2.185397
svmRadialSigma	2.181550
svmRadialSigma	2.187476
svmRadialSigma	2.304355
svmRadialSigma	2.297231
svmRadialSigma	2.321122
svmRadialSigma	2.406423
svmRadialSigma	2.396338
svmRadialSigma	2.426826

Model with the lowest RMSE:

```
#model with lowest RMSE
rmse_results[which.min(rmse_results$RMSE),]
```

```
## # A tibble: 1 x 2
##   method  RMSE
##   <chr>  <dbl>
## 1 gbm     2.13
```

The model with the lowest RMSE is gbm. Train this model on the wine_train set and use it to predict the points for the wine_test set.

fit <- train(points ~ ., method = "gbm", data = wine_train)

```
# model summary
fit
```

```
## Stochastic Gradient Boosting
##
## 97144 samples
##     6 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 97144, 97144, 97144, 97144, 97144, 97144, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##   1                   50      2.200191  0.5056568  1.756873
##   1                  100      2.114632  0.5314479  1.681932
##   1                  150      2.087474  0.5392619  1.656776
##   2                   50      2.116163  0.5324491  1.682299
##   2                  100      2.067513  0.5468069  1.636686
##   2                  150      2.059213  0.5493798  1.627505
##   3                   50      2.086141  0.5418923  1.655217
##   3                  100      2.056365  0.5507410  1.625124
##   3                  150      2.051747  0.5525284  1.620429
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
# Loss function to calculate RMSE
myRMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Predict the points in the test set
y_hat <- predict(fit,wine_test)
#round values to points
y_hat <- ifelse(y_hat >100,100,ifelse(y_hat<80,80,round(y_hat)))

# Use the confusion matrix to look at accuracy
confusionMatrix(data = as.factor(y_hat),reference = as.factor(wine_test$points))$overall["Accuracy"]
```

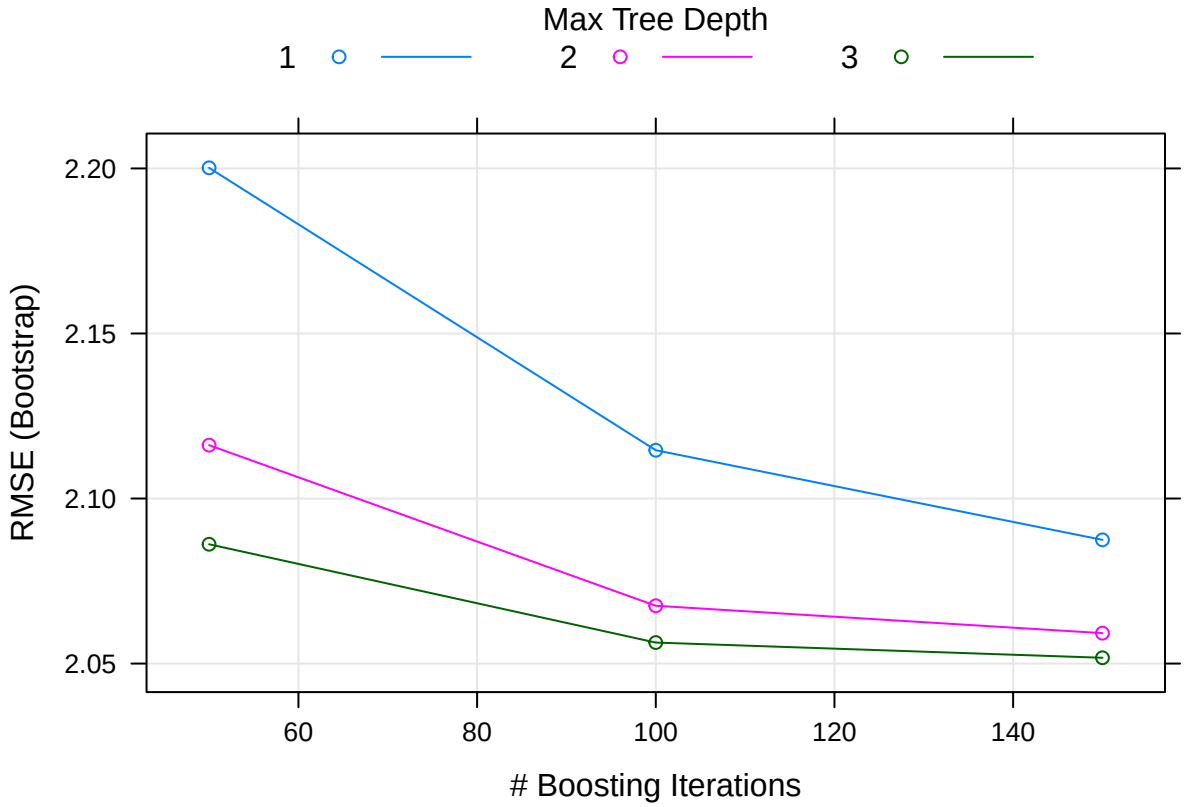
```
##  Accuracy
## 0.1969374
```

```
# Use the loss function to calculate RMSE
myRMSE(wine_test$points,y_hat)
```

```
## [1] 2.068323
```

Plot model:

```
par(mfrow = c(2,2))
plot(fit)
```



Conclusion

The RMSE is quite low and prediction accuracy is not very high. There is a positive correlation between price and points and also between wordcount and points. Both of these correlations are weak to moderate and as such not very strong predictors points.

Better results could possibly be obtained by parameter tuning of the models and also perhaps by using some of the predictors that were excluded. Using a different list to calculate the winewords score would also alter the values of this predictor and add or remove bias that could have an affect on the accuracy of the predictions.