Predicting Wine Prices with Machine Learning

Evan Williams

May 8th, 2019

```
# Load libraries
library(tidyverse)
library(tidytext)
library(RCurl)
library(caret)
library(knitr)
library(kableExtra)
library(rpart)

# Pull the data from my GitHub and load it into our environment
x <- getURL("https://raw.githubusercontent.com/EvanUp/Wine-Review-Data/master/Data/winem
ag-data-130k-v2.csv")
wine_reviews <- read.csv(text=x, stringsAsFactors=FALSE, na.strings = "")
rm(x)
```

Introduction

How much is a bottle of wine worth? What price should a vendor charge for a bottle given its description, area of origin, rater, and point rating? In this report, we will generate and assess several recommendation system algorithms for wine prices using wine-reviews (%22https://www.kaggle.com/zynicide/wine-reviews%22) dataset on Kaggle. This dataset is composed of reviews scraped from WineEnthusiast (%22https://www.winemag.com/?s=&drink_type=wine%22). We will first explore the Dataset and then generate and compare several recommendation systems (fixed effects and a Random Forest) in an effort to predict price. We will evaluate these systems using Root Mean Square Error (RMSE).

Dataset

This dataset was extracted from Kaggle and it includes reviews for over 110,000 different wines published by Wine Enthusiast Magazine between 1999 and 2017 with a rating of 80 or higher (out of 100). Wines that receive a rating below 80 are not reviewed. The full reviewing process is described here (%22https://www.winemag.com/2010/04/09/you-asked-how-is-a-wines-score-determined/%22).

To allow the data to be downloaded without a kaggle account (and by extension this code), I've uploaded the raw data to GitHub. Here is a summary of the

Now we'll examine the dataset: There are 129,971 rows and 14 columns

```
dim(wine_reviews)
```

```
## [1] 129971     14
```

The dataset contains the country of origin, a description written by the reviewer, the vineyard designation within the winery, the number of points on a scale of 1-100 (that only includes reviews of 80 or more), the price in USD, the province or state of origin, the wine growing area, a more specific wine growing area, the name of taster, the taster's twitter handle, the title of the review, the grape varietal, and the winery.

```
glimpse(wine_reviews)
```

```
## Observations: 129,971
## Variables: 14
## $ X                     <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, …
## $ country               <chr> "Italy", "Portugal", "US", "US", "US", "Sp…
## $ description           <chr> "Aromas include tropical fruit, broom, bri…
## $ designation           <chr> "Vulkà Bianco", "Avidagos", NA, "Reserve L…
## $ points                <int> 87, 87, 87, 87, 87, 87, 87, 87, 87, 87, 87…
## $ price                 <dbl> NA, 15, 14, 13, 65, 15, 16, 24, 12, 27, 19…
## $ province              <chr> "Sicily & Sardinia", "Douro", "Oregon", "M…
## $ region_1              <chr> "Etna", NA, "Willamette Valley", "Lake Mic…
## $ region_2              <chr> NA, NA, "Willamette Valley", NA, "Willamet…
## $ taster_name           <chr> "Kerin O'Keefe", "Roger Voss", "Paul Gregu…
## $ taster_twitter_handle <chr> "@kerinokeefe", "@vossroger", "@paulgwine …
## $ title                 <chr> "Nicosia 2013 Vulkà Bianco  (Etna)", "Quin…
## $ variety               <chr> "White Blend", "Portuguese Red", "Pinot Gr…
## $ winery                <chr> "Nicosia", "Quinta dos Avidagos", "Rainsto…
```

Lets look at the number of distinct values for a few of these indicators

```
# blanks are counted as 1, so we subtract 1 from each unique total
cat("Number of wine tasters:", (n_distinct(wine_reviews$taster_name)-1))
```

```
## Number of wine tasters: 19
```

```
cat("Number of countries in the dataset:", (n_distinct(wine_reviews$country)-1))
```

```
## Number of countries in the dataset: 43
```

```
cat("Number of varietals:", (n_distinct(wine_reviews$variety)-1))
```

```
## Number of varietals: 707
```

Data Exploration and Cleaning

In this section, we'll explore the data and clean it.

First, let's take a look at the prices, as this is the variable we are interested in predicting. We can see that the cheapest wine in the dataset is a cheap 4USD bottle, whereas the most expensive costs a whopping 3,300USD. But before we look at the data, lets answer the most important question that this dataset can answer: what are the highest rated wines for 30.00USD or less?

P.S. - If you'd like longer version of the following tables with more detail, you can download them from my GitHub (%22https://github.com/EvanUp/Wine-Review-Data/tree/master/Data%22).

```
summary(wine_reviews$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    4.00   17.00   25.00   35.36   42.00 3300.00    8996
```

Here are the 10 highest rated wines for 10$ or less!

```
wine_reviews %>% filter(price <=10) %>%  select(title, price, points) %>% arrange(desc(p
oints)) %>% slice(1:10) %>% rename(Wine=`title`, Price_USD =`price`, Points = `points`)
 %>%
  kable("html", align = "c", caption = "Best Wines under 10$") %>%
  kable_styling(bootstrap_options =
                  c("striped", "condensed", "bordered"),
                  full_width = FALSE)
```

Best Wines under 10$

Wine	Price_USD	Points
Chateau Ste. Michelle 2011 Riesling (Columbia Valley (WA))	9	91
Quinta do Portal 2007 Mural Reserva Red (Douro)	10	91
José Maria da Fonseca 2007 Moscatel de Setúbal	10	91
Snoqualmie 2006 Winemaker's Select Riesling (Columbia Valley (WA))	8	91
Barnard Griffin 2012 Riesling (Columbia Valley (WA))	10	91
Herdade dos Machados 2012 Toutalga Red (Alentejano)	7	91
Lujon 2015 Riesling (Willamette Valley)	10	91
Lujon 2015 Riesling (Willamette Valley)	10	91
Chateau Ste. Michelle 2010 Dry Riesling (Columbia Valley (WA))	9	91
Vilalva 2013 Reserva Red (Douro)	10	91

Here are the 10 highest rated wines 20$ or less!

```
wine_reviews %>% filter(price <=20) %>%
arrange(desc(points)) %>% select(title, price, points) %>% slice(1:10) %>% rename(Wine=`
title`, Price_USD =`price`, Points = `points`) %>%
  kable("html", align = "c", caption = "Best Wines under 20$") %>%
  kable_styling(bootstrap_options =
                  c("striped", "condensed", "bordered"),
                  full_width = FALSE)
```

Best Wines under 20$

Wine	Price_USD	Points
Rulo 2007 Syrah (Columbia Valley (WA))	20	96
Dunham 2010 Lewis Estate Vineyard Riesling (Columbia Valley (WA))	20	95
Alain Brumont 2010 Château Bouscassé Red (Madiran)	20	95
Januik 2012 Bacchus Vineyard Riesling (Columbia Valley (WA))	20	95
Poet’s Leap 2009 Riesling (Columbia Valley (WA))	20	95
Stottle 2011 Lucille Late Harvest Viognier (Yakima Valley)	18	94
Brian Carter Cellars 2009 Opulento Dessert Wine Touriga-Souzao-Tinto Cão Port (Yakima Valley)	20	94
Château Vignelaure 2016 Rosé (Coteaux d’Aix-en-Provence)	20	94
Osborne NV Pedro Ximenez 1827 Sweet Sherry Sherry (Jerez)	14	94
Domaine Marcel Deiss 2011 White (Alsace)	20	94

And finally the 10 highest rated wines for 30$ or less!

```
wine_reviews %>% filter(price <=30) %>%
  arrange(desc(points)) %>% select(title, price, points) %>% slice(1:10) %>% rename(Wine
=`title`, Price_USD =`price`, Points = `points`) %>%
  kable("html", align = "c", caption = "Best Wines under 30$") %>%
  kable_styling(bootstrap_options =
                  c("striped", "condensed", "bordered"),
                  full_width = FALSE)
```

Best Wines under 30$

Wine	Price_USD	Points
Domaines Schlumberger 2014 Saering Grand Cru Riesling (Alsace)	29	96
Isole e Olena 2010 Chianti Classico	27	96
Château Filhot 2014 Sauternes	28	96
Donkey & Goat 2012 Grenache Noir Grenache (El Dorado)	27	96
Eichinger 2014 Gaisberg Reserve Riesling (Kamptal)	28	96
Efeste 2009 Lola Chardonnay (Columbia Valley (WA))	30	96
Rulo 2007 Syrah (Columbia Valley (WA))	20	96
The Eyrie Vineyards 2014 Estate Chardonnay (Dundee Hills)	27	96
Sineann 2015 TFL Pinot Noir (Willamette Valley)	30	96
Alvear NV Solera 1927 Pedro Ximénez (Montilla-Moriles)	30	96

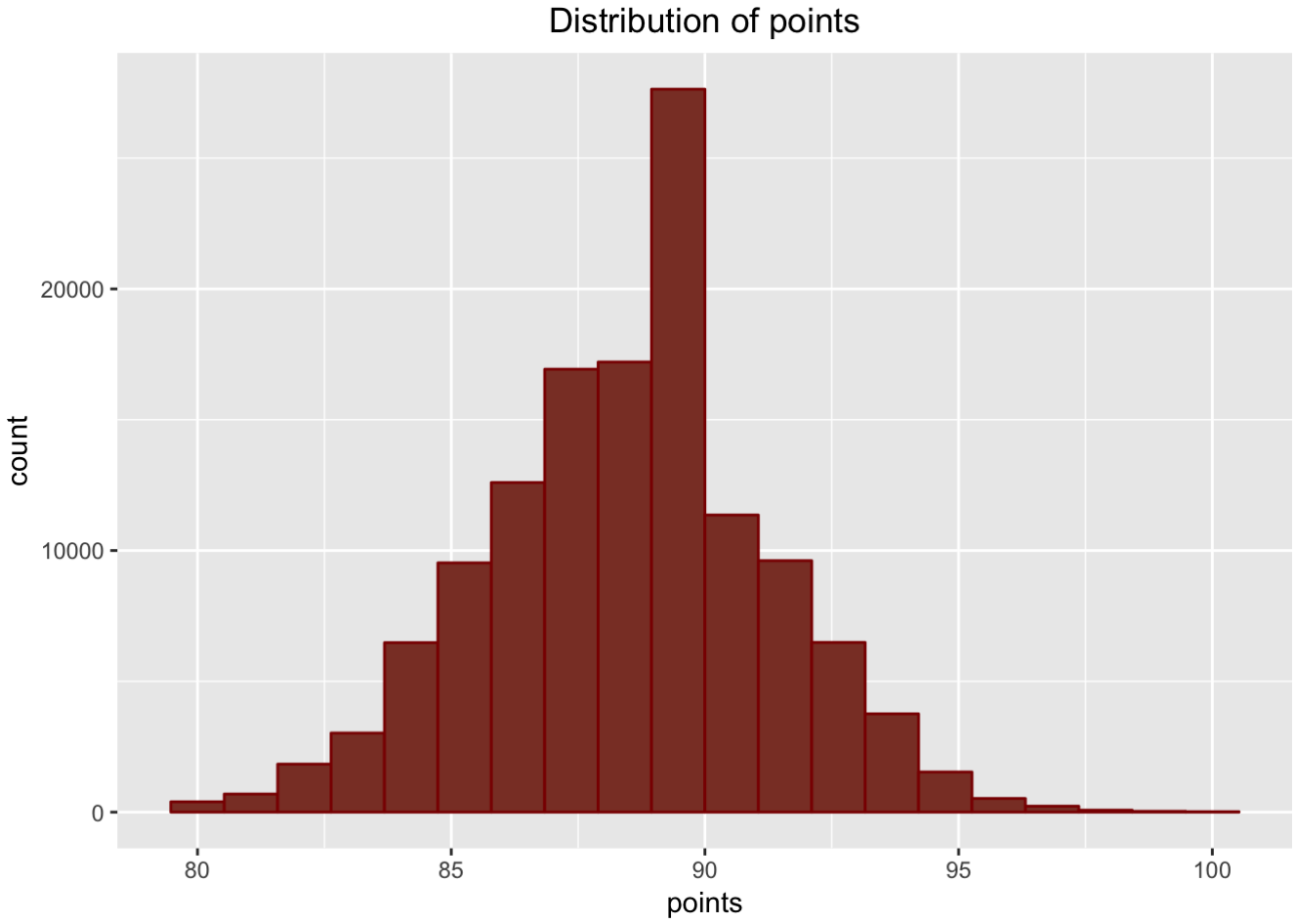
Now that the most important question is out of the way, lets look more in depth at the distributions of points and price as well as their relationship.

Points

Points are the ratings that Sommeliers gave the wines. We know that the range is between 80 and 100, but lets look at the distribution of point scores.

We can see that the majority of the ratings fall in the upper 80s (around 89). The data looks like a more or less normal distribution.

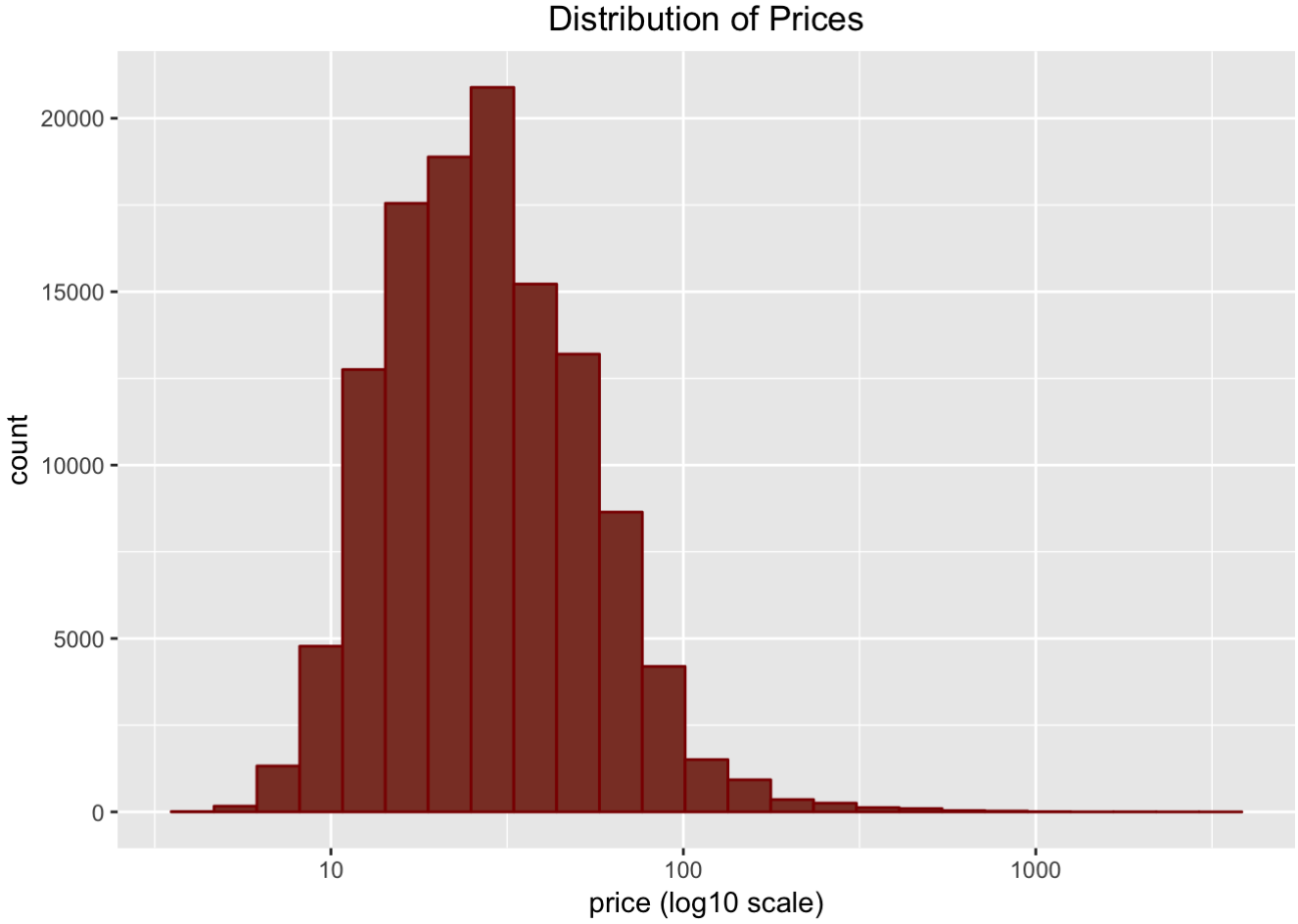
```
wine_reviews %>%
  ggplot(aes(points)) +
  geom_histogram(bins=20, fill="coral4", color="red4") +
  labs(title="Distribution of points")+
  theme(plot.title = element_text(hjust = 0.5))
```



Price

We can see that about 99% of wines that were rated were $150 or less- additionally we see farther down a wide confidence interval for wines over that price. We also see there are 8996 wines with missing prices.

```
wine_reviews %>% ggplot(aes(x=price)) +
  geom_histogram(bins=25, fill="coral4", color="red4") +
  scale_x_log10() +
  labs(x = "price (log10 scale)", title= "Distribution of Prices")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
cat("Percentage of wines over 100USD: ",mean(wine_reviews$price>150, na.rm=TRUE)*100,"%"
, sep="")
```

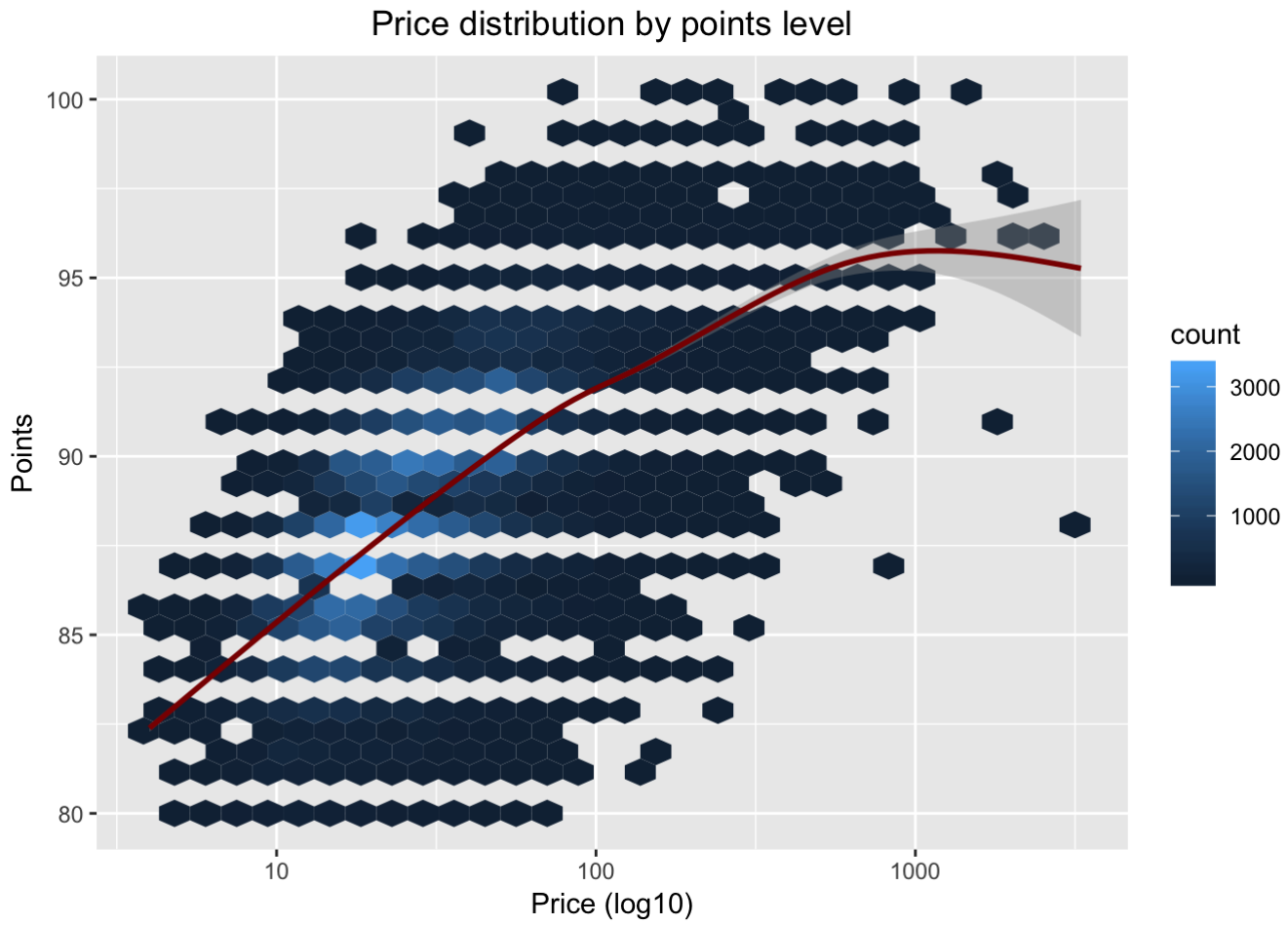
```
## Percentage of wines over 100USD: 1.017566%
```

```
cat("Number of wines with no price data",  sum(is.na(wine_reviews$price)))
```

```
## Number of wines with no price data 8996
```

When we plot points against price, there does seem to be a relationship trend, but that trend has higher variability when it reaches bottles worth over 150USD (only about 1% of the data). To improve our algorithm's accuracy, we can trim price at 150USD. If a vendor thinks their wine is worth over 150USD, they'd benefit from consulting sommeliers directly. We can also drop wines with no price data as we're interested in the relationship between price and quality.

```
wine_reviews %>% filter(price != is.na(price)) %>%
  ggplot(aes(x=price,y=points))+
  geom_hex()+
  geom_smooth(color="red4", span=.2)+
  scale_x_log10()+
  labs(title="Price distribution by points level", y="Points", x="Price (log10)")+
  theme(plot.title = element_text(hjust = 0.5))
```

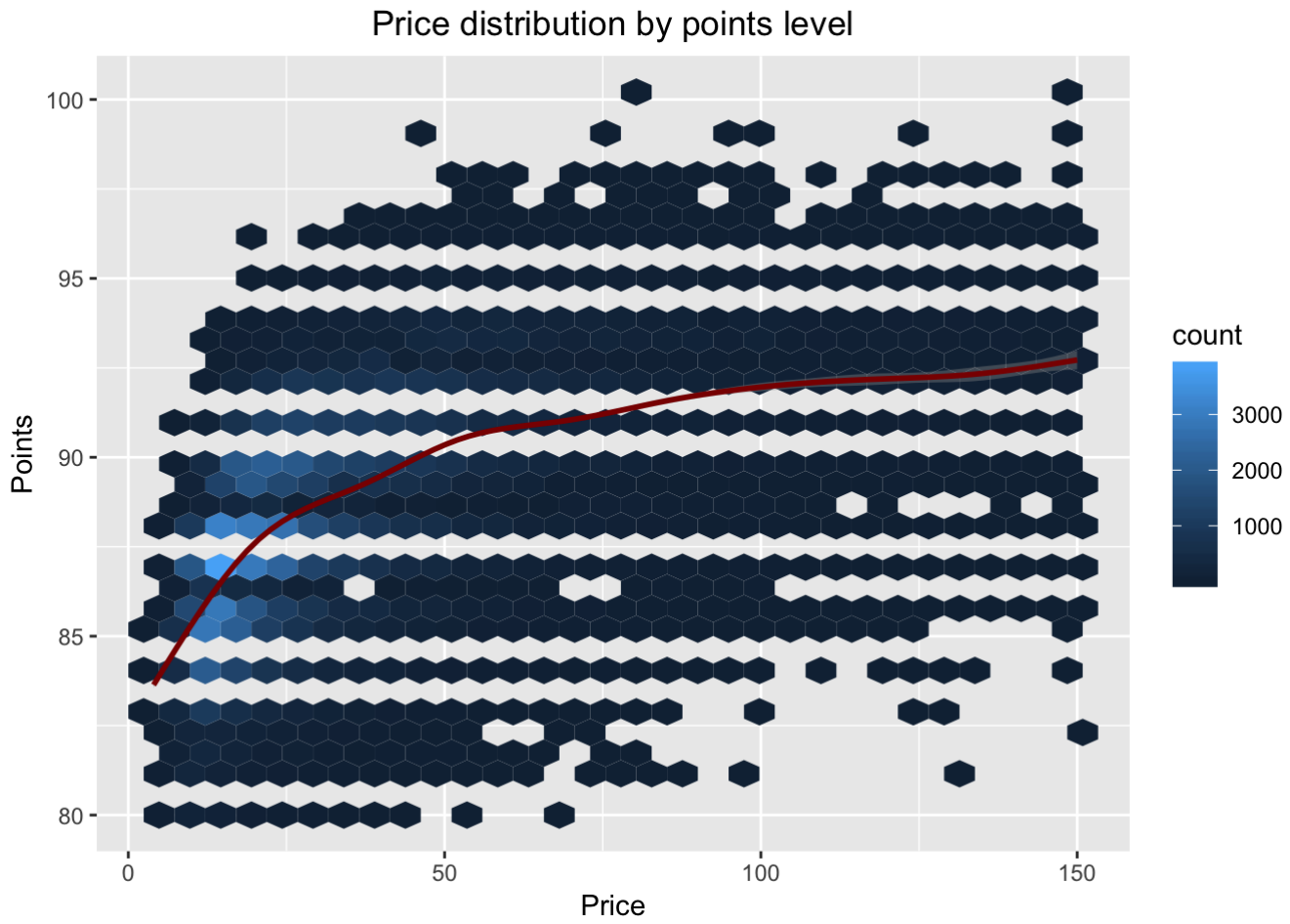


We'll trim the data and drop all wines over 200USD.

```
# Filter silently drops NA's
wr_150 <- wine_reviews %>% filter(price <= 150)
```

When we trim price, we can see that there's a rapid increase in points up until about 30$ per bottle, but after that, points increase more gradually.

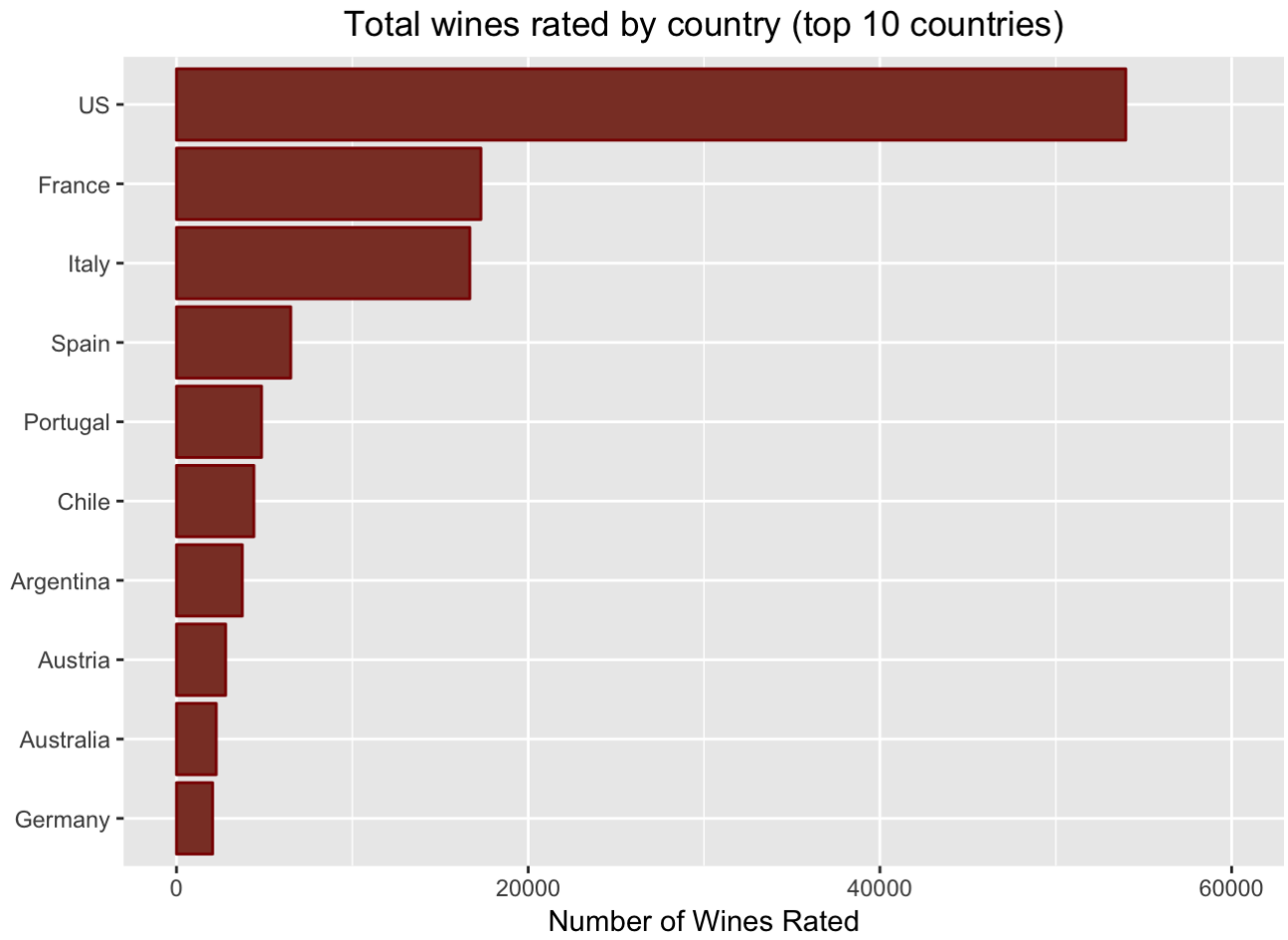
```
wr_150 %>%
  ggplot(aes(x=price,y=points))+
  geom_hex()+
  geom_smooth(color="red4", span=.2)+
  labs(title="Price distribution by points level", y="Points", x="Price")+
  theme(plot.title = element_text(hjust = 0.5))
```



Country

Lets examine the top 10 countries of origin of the wines reviewed. We can see that the majority of the wine reviews were of US wines (with over 50,000 wines reviewed). France is the country with the second highest number of wines reviewed (over 20,000 French wines were reviewed, but we dropped many of them when we trimmed and filtered price.)

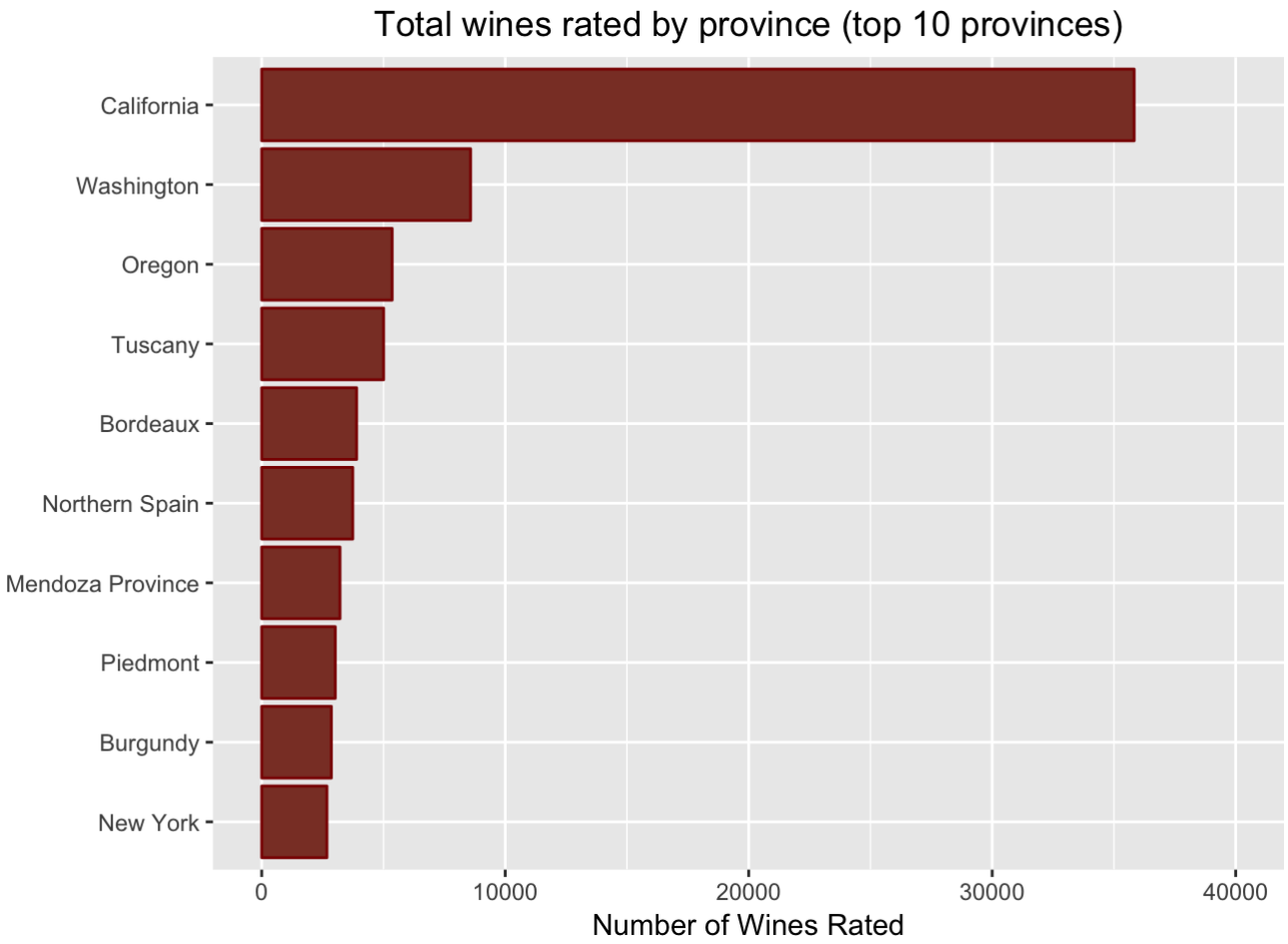
```
wr_150 %>% group_by(country)%>% summarize(count = n()) %>% arrange(desc(count)) %>% slic
e(1:10) %>% ggplot(aes(x =reorder(country, count), y =  count )) +
  geom_bar(stat='identity',colour="red4", fill = "coral4") +
  theme(plot.title = element_text(hjust = 0.5))+
  labs(x = '', y = "Number of Wines Rated", title = 'Total wines rated by country (top 1
0 countries)') +
  ylim(0, 60000)+
  coord_flip()
```



Province

We can drill down further and look at the province (or state) of origin for the wines. We know the data are dominated by US wines, so we should expect to see California as number one. However, we also see a decent number of wines from Tuscany, Bourdeaux, Northern Spain, and a few other foreign provinces.

```
wr_150 %>% group_by(province)%>% summarize(count = n()) %>% arrange(desc(count)) %>% sli
ce(1:10) %>% ggplot(aes(x =reorder(province, count), y =  count )) +
  geom_bar(stat='identity',colour="red4", fill = "coral4") +
  theme(plot.title = element_text(hjust = 0.5))+
  labs(x = '', y = "Number of Wines Rated", title = 'Total wines rated by province (top
 10 provinces)') +
  ylim(0, 40000)+
  coord_flip()
```



Additionally, we can see that there are 59 missing provinces. We can drop these as we'll use province in our algorithm- anyone wanting to know what a wine is worth would surely know the province or state that it was made in.

```
cat("Data with missing province:", sum(is.na(wr_150$province)))
```

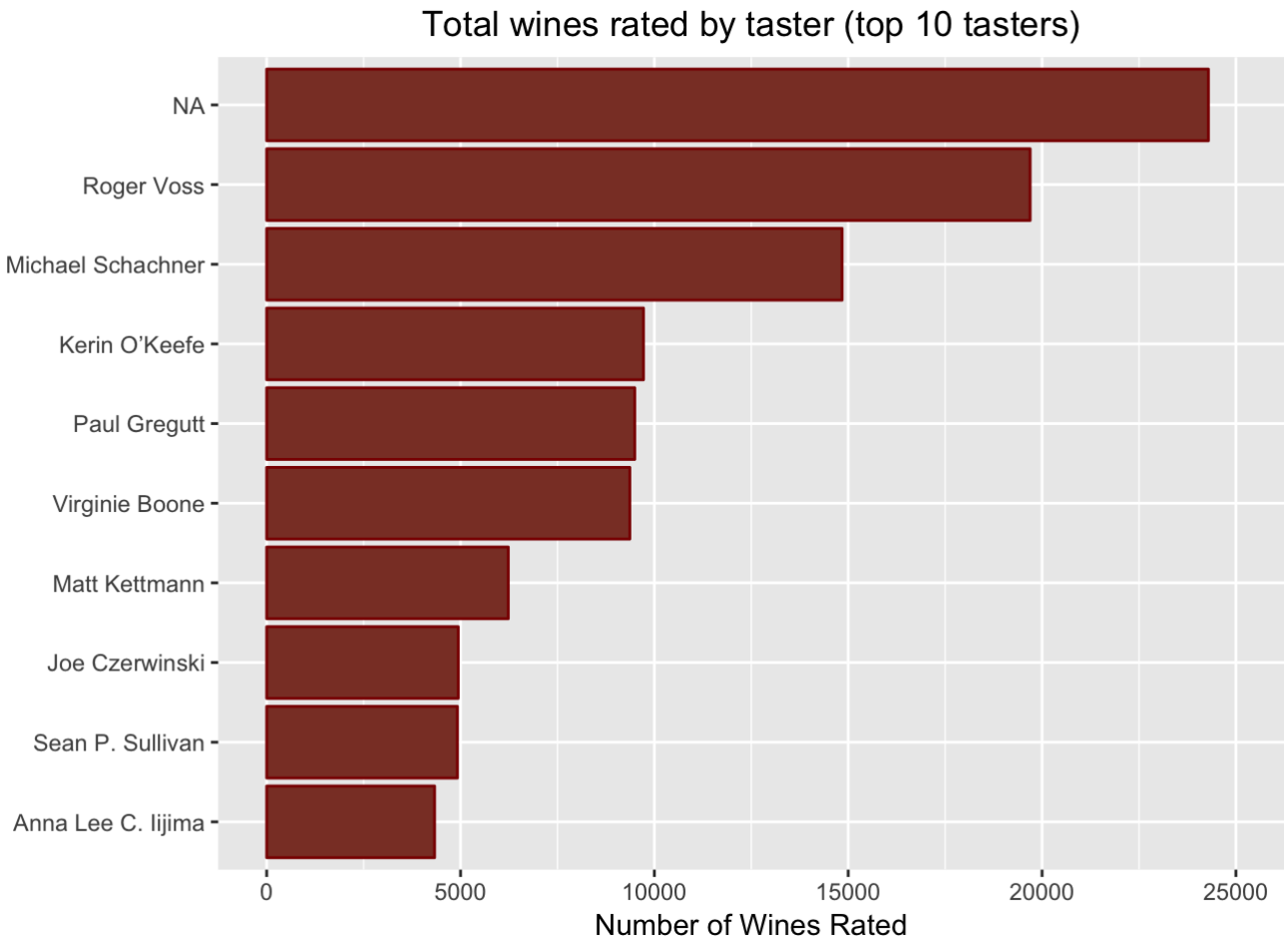
```
## Data with missing province: 59
```

```
wr_150_p <- wr_150 %>% drop_na(province)
```

Taster

Finally, before we dive into the text analysis, lets take a look at the tasters. The majority of wines were rated by anonymous tasters, but Roger Voss rated nearly 20,000 wines under the price of 150$. Roger Voss often rates the most expensive French wines, so his real total is much higher. I have a new personal hero.

```
wr_150_p %>% group_by(taster_name)%>% summarize(count = n()) %>% arrange(desc(count)) %
>% slice(1:10) %>% ggplot(aes(x =reorder(taster_name, count), y =  count )) +
  geom_bar(stat='identity',colour="red4", fill = "coral4") +
  theme(plot.title = element_text(hjust = 0.5))+
  labs(x = '', y = "Number of Wines Rated", title = 'Total wines rated by taster (top 10
tasters)') +
  ylim(0, 25000)+
  coord_flip()
```



Description

Given that descriptions are written descriptions of wine, it's strange that there would be duplicate descriptions- but we see that there are over 9,000 repeated wine descriptions in our dataset. If we want to use text analysis, we'll need to drop these duplicates.

```
cat("Number of Duplicated Descriptions:", nrow(wr_150_p) - n_distinct(wr_150_p$descripti
on))
```

```
## Number of Duplicated Descriptions: 9365
```

We'll create our final, clean dataset by removing these duplicates.

```
wr_clean <- wr_150_p %>%
  mutate(dups = duplicated(description)) %>%
  filter(dups==FALSE) %>%
  select(-dups)
```

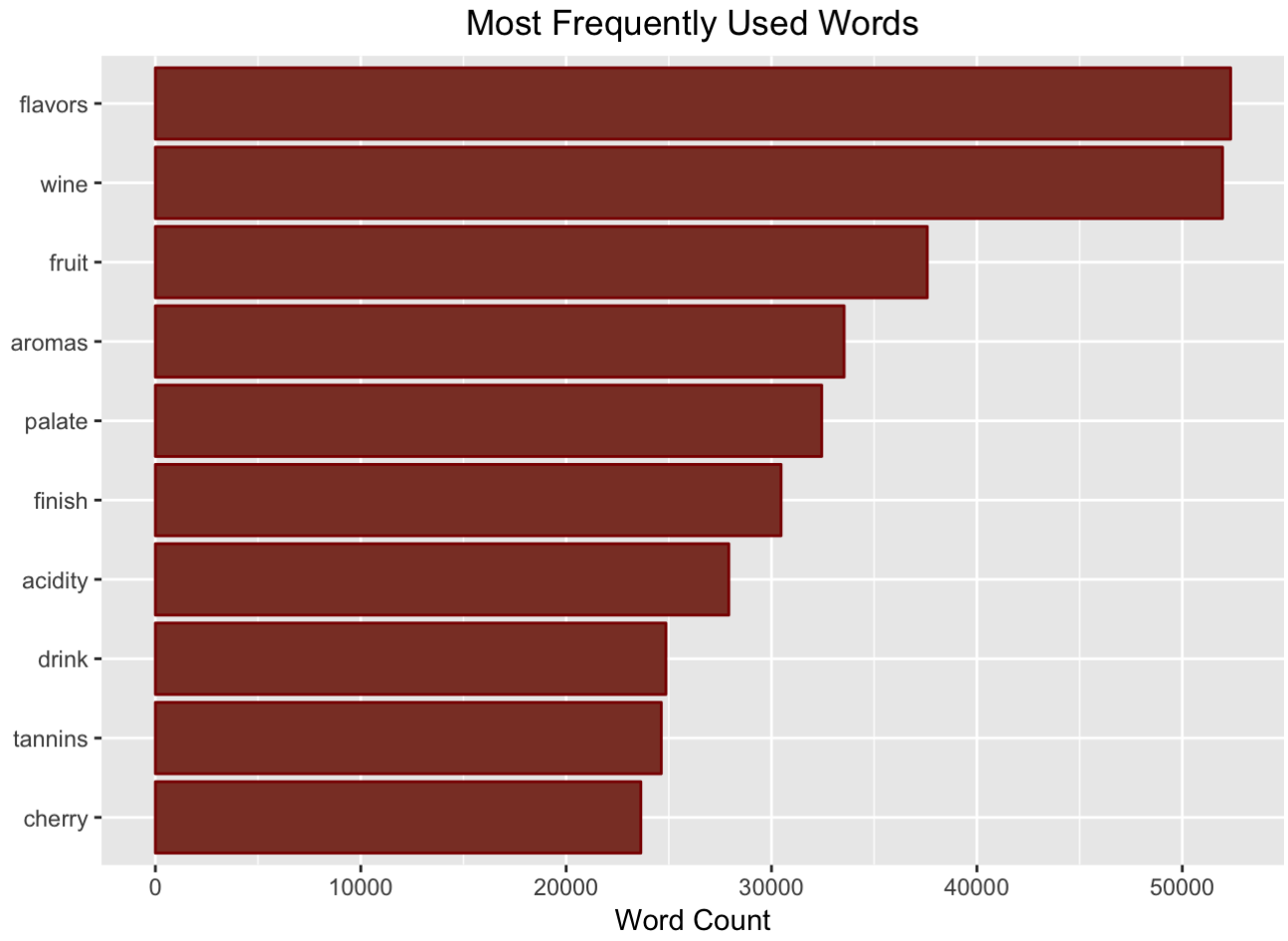
Natural Language Processing and Cleaning

In this section, we will examine the most-used words in descsriptions of wine and we'll create a variable for length of descriptions. We'll also do a small amount of cleaning.

```
wine_descriptions <- wr_clean %>%
  mutate(description = tolower(description)) %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  distinct()
```

Unsurprisingly, the most-used words are the words we associate with wine: flavors, wine, fruit, aromas, palate, finish, etc.

```
wine_descriptions %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_bar(stat='identity',colour="red4", fill = "coral4") +
  theme(plot.title = element_text(hjust = 0.5))+
  labs(x = '', y = "Word Count", title = 'Most Frequently Used Words') +
  coord_flip()
```

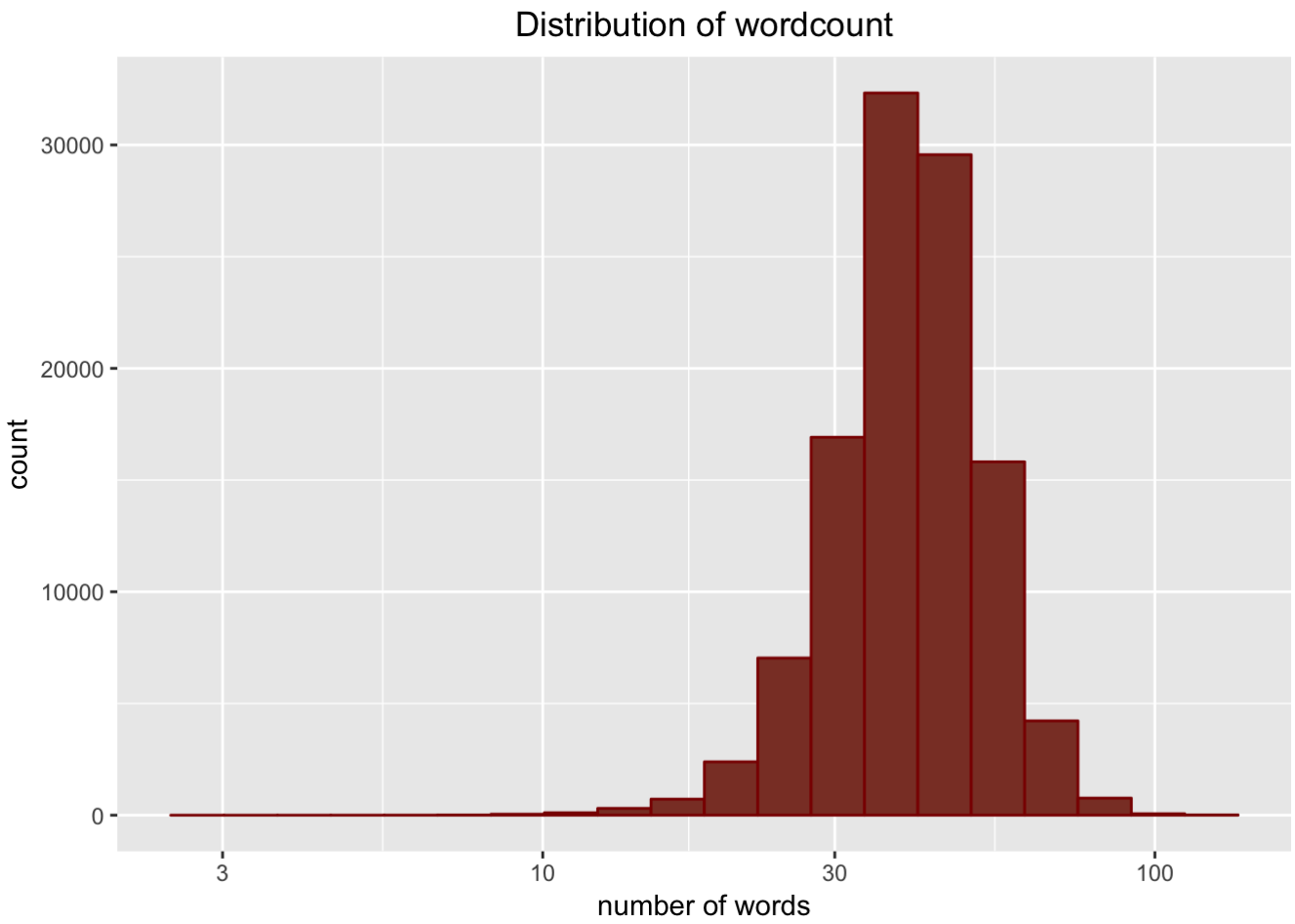


Now lets look at wordcount per review- perhaps there's a relationship between the length of a review and the quality of a wine.

```
wine_des_total <- wr_clean %>%
  mutate(description = tolower(description)) %>%
  unnest_tokens(word, description) %>%
  group_by(X, price) %>%
  summarize(num_words = n()) %>%
  arrange(desc(num_words))
```

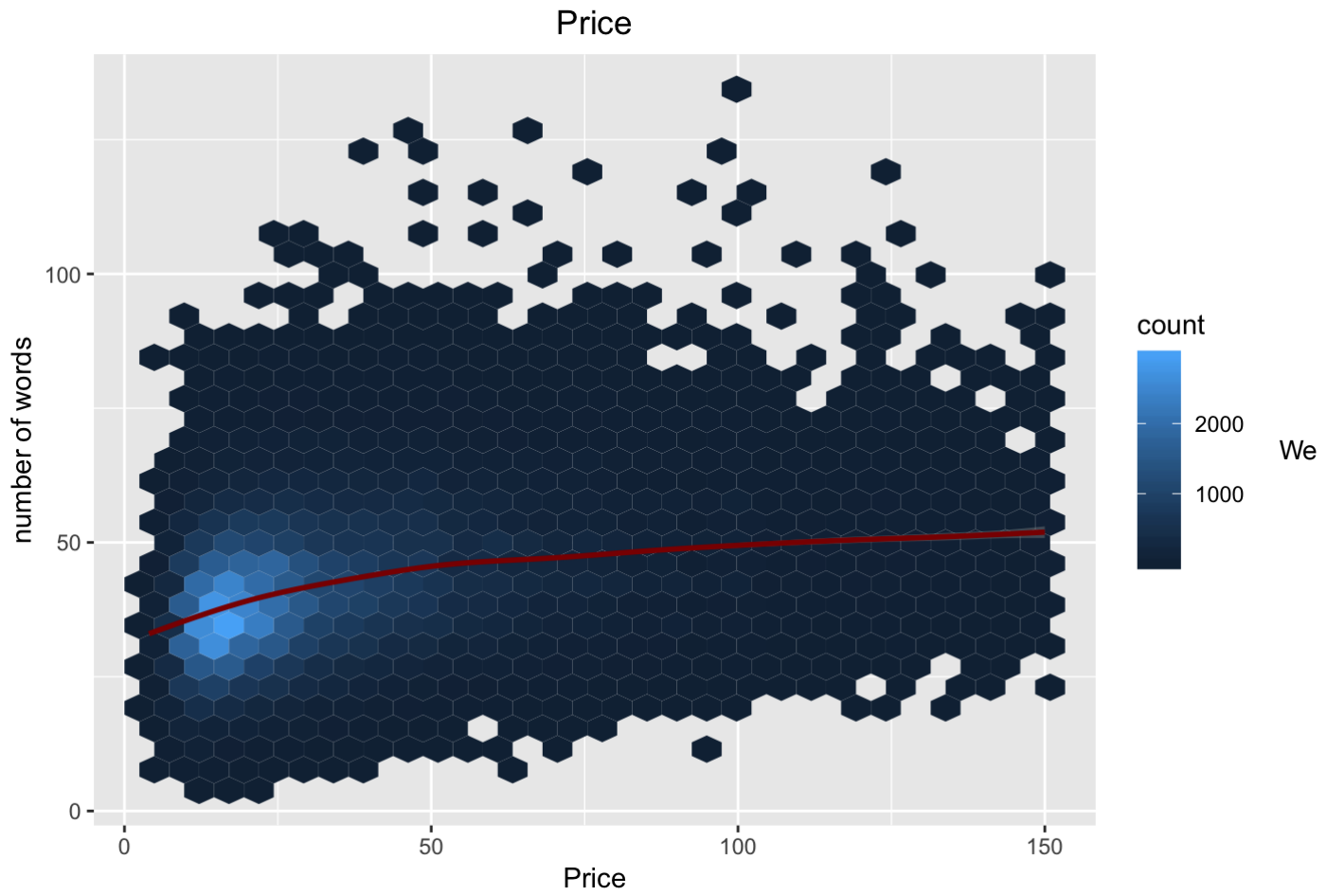
Like price, there seems to be somewhat of a normal distribution of wordcounts.

```
wine_des_total %>% ggplot(aes(x=num_words)) +
  geom_histogram(bins=20, fill="coral4", color="red4") +
  scale_x_log10() +
  labs(x = "number of words", y= "count", title= "Distribution of wordcount")+
  theme(plot.title = element_text(hjust = 0.5))
```



When we look at price against wordcount, we see somewhat of a relationship. More words seem to generally indicate better reviews, but we see a wider range around low prices.

```
wine_des_total %>%
  ggplot(aes(x=price,y=num_words))+
  geom_hex()+
  geom_smooth(color="red4", span=.2)+
  labs(title="Price", y="number of words", x="Price")+
  theme(plot.title = element_text(hjust = 0.5))
```



can drop descriptions over 100 words and below 10 words as these are clearly outliers and we can merge this into our dataframe. This will drop 68 observations. Finally, we'll convert characters to factors.

```
wine_des_total1 <- wine_des_total %>% select(-price) %>%
  filter(num_words>=10 & num_words<=100)

wr_clean_num <- wr_clean %>% left_join(wine_des_total1, by="X") %>%
  drop_na(num_words) %>%
  mutate(province = factor(province),
         variety = factor(variety),
         taster_name = factor(taster_name),
         winery = factor(winery))
```

Methods

To create a price recommendation system, we'll implement the model-based approach that we learned in the machine learning course on Edx. We'll first consider a model where we simply recommend the same price for all wines (by taking average price), and then we'll add controls for province effects, points effects, taster effects, and wordcount effects in order to lower the RMSE.

Ideally, these effects would be considered using regressions, but unfortunately, attempting to do so would likely crash this computer. Rather, we'll compute an approximation by estimating the overall mean, `mu`. We will then use this to find the province effects, points effects, taster effects, and wordcount effects. Each of these controls is added to our approximation of a regression line, and each subsequent feature will be added to our approximation of a regression line sequentially.

The predicted results for each model wil thus be:
Simple Average: `mu`
Province Effects: `mu + province effects`
Points Effects: `mu + province effects + points_effects`
Taster Effects: `mu + province effects + points_effects + taster effects`
Wordcount Effects: `mu + province effects + points_effects + taster effects + wordcount`

Next, we will run a random forest over the data and compare the results to the fixed effects models. Random forests generate Classification and Regression Trees which stratify data based on if-else rules. The rules divide the dataset into distinct and non-overlapping regions (recursive binary splitting). Random forests generate several trees and obtain a final prediction by averaging or voting.

The models will be evaluated using the root-mean-square error (RMSE). The RMSE is simply the standard deviation of the residuals. The formula is below.

```
RMSE <- function(true_prices, predicted_prices){
    sqrt(mean((true_prices - predicted_prices)^2))
}
```

Results

Before anything else, we'll partition the data into a training set and a test set. The training set contains 90% of the data and the testing set contains 10% of the data.

```
set.seed(123)
test_index <- createDataPartition(y = wr_clean_num$price, times = 1, p = 0.10, list = FA
LSE)
train_set <- wr_clean_num[-test_index,]
temporary_set <- wr_clean_num[test_index,]
# Making sure the variables we care about are in both datasets.
test_set <- temporary_set %>%
      semi_join(train_set, by = "country") %>%
      semi_join(train_set, by = "province") %>%
      semi_join(train_set, by = "taster_name") %>%
      semi_join(train_set, by = "variety") %>%
      semi_join(train_set, by = "winery") %>%
      semi_join(train_set, by = "num_words")
removed_rows <- anti_join(temporary_set, test_set)
train_set <- rbind(train_set, removed_rows)
rm(removed_rows, temporary_set, test_index)
```

First, lets try simply filling all missings with the average and seeing how well that model does at pricing

```
mu <- mean(train_set$price)
naive_rmse <- RMSE(test_set$price, mu)
rmse_results <- bind_rows(tibble(Method = "Just the average", RMSE = naive_rmse))
kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Method	RMSE
Just the average	22.89948

For a producer, overcharging or undercharging by an average 22.90USD wouldn't be ideal as the producer would likely either be unable to sell his or her wine, or sell far below profit. We'll try adding in fixed effects in order to bring down the RMSE. Simply adding province brings our error down to 21.16USD. Lets add a few more effects and see how it changes RMSE.

```
prov_avgs <- train_set %>%
  group_by(province) %>%
  summarize(province_effect = mean(price - mu))

predicted_ratings <- mu + test_set %>%
  left_join(prov_avgs, by="province") %>%
  .$province_effect

province_effect_RMSE <- RMSE(predicted_ratings, test_set$price)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "With Province Effect",
                                     RMSE=province_effect_RMSE))
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Method	RMSE
Just the average	22.89948
With Province Effect	21.16086

Next lets add Points Effects

```
points_avgs <- train_set %>%
  left_join(prov_avgs, by='province') %>%
  group_by(points) %>%
  summarize(points_effect = mean(price - mu - province_effect))

predicted_ratings <- test_set %>%
  left_join(prov_avgs, by="province") %>%
  left_join(points_avgs, by='points') %>%
  mutate(pred = mu + province_effect + points_effect) %>%
  .$pred

points_effect_RMSE <- RMSE(predicted_ratings, test_set$price)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "With Points Effect",
                                     RMSE=points_effect_RMSE))

kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Method	RMSE
Just the average	22.89948
With Province Effect	21.16086
With Points Effect	18.05503

Next lets add Taster Effects

```
taster_avgs <- train_set %>%
  left_join(prov_avgs, by='province') %>%
  left_join(points_avgs, by='points') %>%
  group_by(taster_name) %>%
  summarize(taster_effect = mean(price - mu -province_effect-points_effect))

predicted_ratings <- test_set %>%
  left_join(prov_avgs, by="province") %>%
  left_join(points_avgs, by='points') %>%
  left_join(taster_avgs, by='taster_name') %>%
  mutate(pred = mu + province_effect + points_effect+taster_effect) %>%
  .$pred

taster_effect_RMSE <- RMSE(predicted_ratings, test_set$price)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "With Taster Effect",
                                     RMSE=taster_effect_RMSE))

kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Method	RMSE
Just the average	22.89948
With Province Effect	21.16086
With Points Effect	18.05503
With Taster Effect	17.65994

Next we'll add Wordcount Effects

```
wc_avgs <- train_set %>%
  left_join(prov_avgs, by='province') %>%
  left_join(points_avgs, by='points') %>%
  left_join(taster_avgs, by='taster_name') %>%
  group_by(num_words) %>%
  summarize(wordcount_effect = mean(price - mu -province_effect-points_effect-taster_eff
ect))

predicted_ratings <- test_set %>%
  left_join(prov_avgs, by="province") %>%
  left_join(points_avgs, by='points') %>%
  left_join(taster_avgs, by='taster_name') %>%
  left_join(wc_avgs, by='num_words') %>%
  mutate(pred = mu + province_effect + points_effect + taster_effect + wordcount_effect)
%>%
  .$pred

wordcount_effect_RMSE <- RMSE(predicted_ratings, test_set$price)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "With Wordcount Effect",
                                     RMSE=wordcount_effect_RMSE))

kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Method	RMSE
Just the average	22.89948
With Province Effect	21.16086
With Points Effect	18.05503
With Taster Effect	17.65994
With Wordcount Effect	17.62622

Finally, we'll run a random forest using all of the same features as our wordcount effect model:

```
tree <- rpart(price ~ province+points+taster_name+num_words, data = train_set)
pred <- predict(tree, test_set)
RFRMSE <-RMSE(test_set$price, pred)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "Random Forest",
                                     RMSE=RFRMSE))

kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Method	RMSE
Just the average	22.89948
With Province Effect	21.16086
With Points Effect	18.05503
With Taster Effect	17.65994
With Wordcount Effect	17.62622
Random Forest	18.28926

Conclusion

In conclusion, the fixed Effects model actually outperformed a Random Forest model that was using the same data. That being said, these simple models reduced the RMSE by 23% from the simple average. We now have a model that can generally price a wine within 17.63USD of its correct value.

If this model were to be developed further, we could add more NLP variables, and we could incorporate wineries, grape varieties, and years into the model. Additionally, it would be a good idea to try other machine learning methods to see if we can reduce the RMSE even further.