Wine Ratings Project - Natural Language Processing

Code

Jas Beausejour

March 19, 2019

- Executive Summary
- Introduction
- Creating the Data Sets
- Methods and Analysis
    - Exploratory Data Analysis and Visualization of non-NLP variables
        - Country
        - Points
        - Price
        - Province
        - Taster
        - Vintage
        - Variety
        - Winery
    - NLP on the Description Variable
    - Modelling
        - Country Effect
        - Taster Effect
        - Vintage Effect
        - Winery Effect
        - Variety Effect
        - Province Effect
        - Machine Learning Algorithm
- Results
- Conclusion

Note 1: This report was created in the context of the HarvardX Professional Certificate in Data Science program available on edX. It was created solely for the purpose of this course by Jas Beausejour.

Note 2: Running this code can easily take 3-5 minutes. The actual run time is indicated at the end of the report.

Note 3: This code requires internet connection.

Executive Summary

In this report, we will show that when asked to predict the score given to a wine bottle by a connoiseur (on 100), a simple guess of **88.42** would yield a Root Mean Square Error of around 3.06. Given the range of scores given in the dataset is 80-100, a RMSE of 3.06 is already pretty good. However, we show that we can improve this performance by a combination of the techniques used in the MovieLens project and machine learning.

We will show that with the knowledge of the country of origin, the identity of the connoisseur, the vintage of the bottle, the winery that produced the wine, the variety of grape, and the province of origin, one would be able to reduce one's RMSE by about 23%.

We then show that the layering of a simple linear regression model that takes into account the previous predictions, the price of the bottle and a few characteristics of the description given by the connoisseur (obtained through Natural Language Processing) can further improve RMSE by roughly 12%.

In that linear regression, we show that the most important variables are price, the results from the predictions using the categorical variables discussed previously and finally a list of words that are commonly associated with great scores that are present in the descriptions: **age, vineyard, complex, vintage, firm, dense, chocolate, power, delicious, powerful, time, richness, fine, currant, mineral, intense, layer.**

We believe the performance of the model could be further improved by digging deeper into natural language processing of the descriptions as well as experimenting with other machine learning algorithms, which we decided not to do to based on computing constraints.

Introduction

In our daily life, we are fans of wine. From the old world to the new world, from the darkest of reds to the crispest of whites, we love tasting everything we can get our hands on. Even though we have our personal preferences for dark, full-bodied reds like Tempranillos from Rioja or Amarone della Valpolicella, we are always eager to try new wines.

Wine, however, can be seen as a bet that one makes that one will like the bottle one just bought for $20. It is one of the very few products that has very little to tell us visually as long as it is in the bottle. And we make that bet every time we visit friends or buy a bottle for ourselves.

Enter the points systems. World reknowned connoisseurs make a living off tasting wine and attributing it a rating from 0 to 100. Customers, in turn, look at these ratings to help them discover new bottles that they might enjoy. These ratings are often accompanied by detailed tasting notes written by the same connoisseurs.

The motivation of this report is to use a data set that includes many wine tasting notes from various connoisseurs and use machine learning algorithm to try and predict what ratings would be assigned based on the tasting notes and a few other variables. This is important: points by and large can determine the value of a bottle. What's more, tasting notes can sometimes come out before the rating comes out, allowing one to get one's hands on an underpriced bottle of wine.

The main goal of this report will be to create a machine learning model that minimizes the **Root Mean Square Error** of our predictions. The **RMSE** can be computed as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{c,t,v,w,va,p}\left(\hat{y}_{c,t,v,w,va,p} - y_{c,t,v,w,va,p}\right)^2}$$

We define $y_{c,t,v,w,va,p}$ as the true rating for country c, by taster t, of vintage v, from grape variety va produced in province p, and denote our prediction with $\hat{y}_{c,t,v,w,va,p}$. N is the number of predictions we make in a dataset.

Let us create a formula that automatically calculates the **RMSE**.

Hide

```
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
  }
```

In this report, we will be using a dataset created by *zackthoutt* which can easily be downloaded from **Kaggle** at this link (https://www.kaggle.com/zynicide/wine-reviews). To ensure that this code runs on a standalone basis, I include code that downloads the dataset into the user's temporary files from my Github repository (https://github.com/jasbeausejour/EDX_Updated_Movie_Project).

This report will contain four main sections.

First, we will use the **Creating the Datasets** section as an opportunity to set up the various datasets that we will be using throughout this report. There will be 3 main datasets:

1. **rating**: This contains all rows from the two sets below. This is the only set that will be created in this section. The other two will be created when we start our modelling exercise in the **Methods and Analysis** section.
2. **test_set**: This will be our final test set. We will not reference to it until we are ready to test our model and report our final RMSE, on which we will be graded. This will represent 10% of the data.
3. **train_set**: This is the bulk of the data, on which we will be performing our analysis. It represents about 90% of the data. We will act as thought this was this only data

available, and train our models on this dataset.

Second, in the **Methods and Analysis** section, we will describe the process we use to create our rating system, but we will first do a bit of exploratory data analysis to better understand the data set. We will then turn our attention to the modelling of the problem, which we will explain in detail in the hopes that the reader can easily follow our thoughts.

Third, we will use the model created in the previous section to make predictions on the **testing_set** set. We will then proceed to calculate our final **RMSE**. This is the **Results** section.

Fourth, our report will end on a **Conclusion**, where we suggest paths forward to further improve this algorithm.

Creating the Data Sets

First off, we download the full dataset from our personal Github repository (https://github.com/jasbeausejour/EDX_Updated_Movie_Project).

Hide

```
# Now, I create a temporary file
dl <- tempfile()
# Here, we download the dataset in its raw format from my GitHub repository
download.file("https://raw.githubusercontent.com/jasbeausejour/Wine_Updated_Pro
ject/master/Data/winemag-data-130k-v2.csv",dl)
```

```
trying URL 'https://raw.githubusercontent.com/jasbeausejour/Wine_Updated_Projec
t/master/Data/winemag-data-130k-v2.csv'
Content type 'text/plain; charset=utf-8' length 52908702 bytes (50.5 MB)
downloaded 50.5 MB
```

We can now read the file into the R environment.

Hide

```
ratings <- read.csv(dl,
                      sep = ",",
                      fill = TRUE)
```

Although we will need a training and a testing set, we wait until all our manipulations have occurred before creating them.

Methods and Analysis

Exploratory Data Analysis and Visualization of non-NLP variables

Before diving into the machine learning exercises, it is important to understand the data we are using. In this section, we will dive into the details of the data that is available to us.

Let us now define our dataset.

Hide

```
dim(ratings)
```

```
[1] 129971     14
```

We have 120,971 observations with 14 variables. Each row represents a review that has been given by one of 19 tasters in the dataset. Let's list these variables.

Variables	Description
X	Row Identifier
country	Country of origin of the wine
description	The full text of the review written by the taster
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 the 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	Name of the taster

Variables	Description
taster_twitter_handle	Twitter Handle of the taster
title	The title of the wine review
variety	The type of grapes used to make the wine (ie Pinot Noir)
winery	The winery that made the wine

Let's now explore some of these variables.

Country

Let us now turn our attention to which countries are represented.

We can compute how many different countries are represented.

Hide

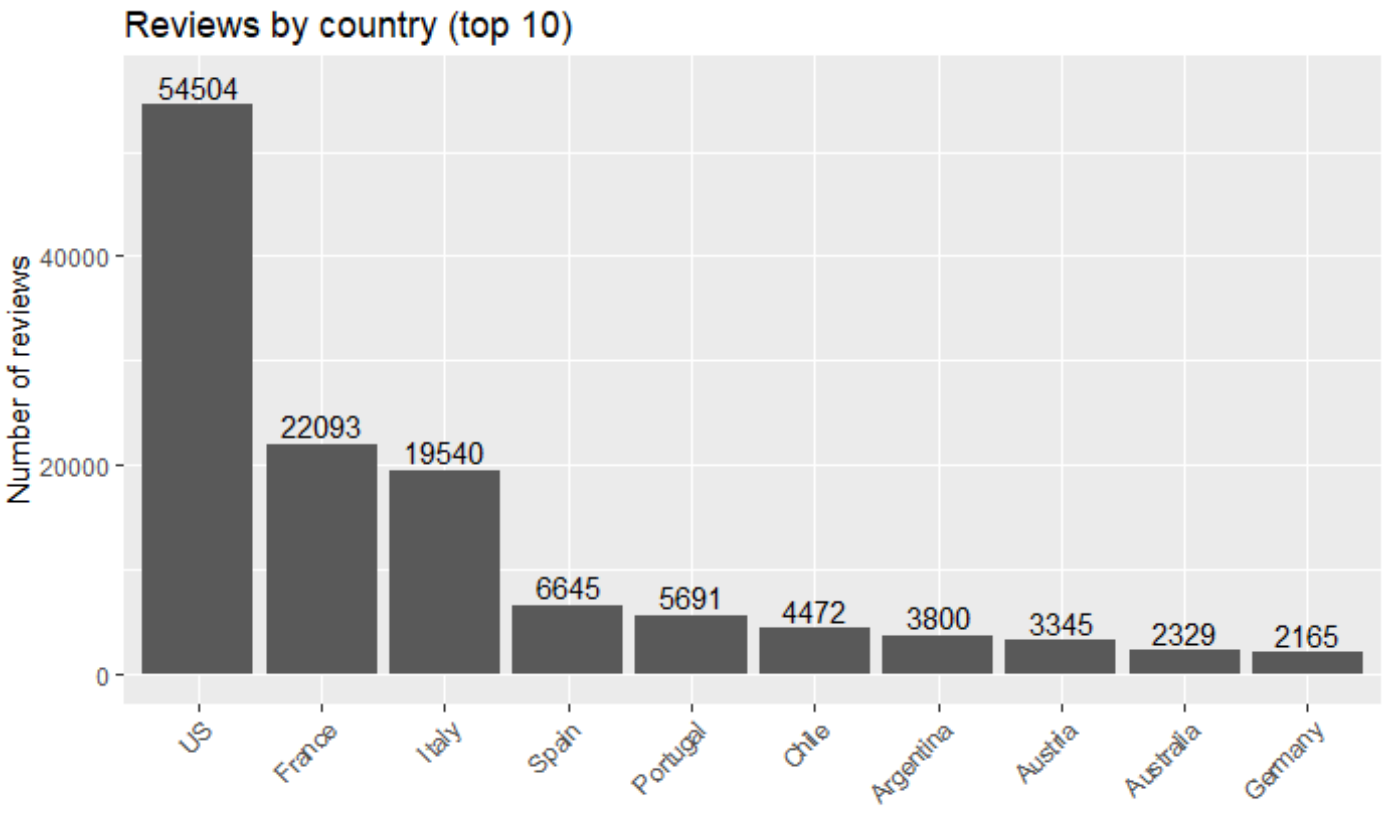
```
length(unique(ratings$country))
```

```
[1] 44
```

We can make a bar plot showing the number of reviews by country like this (for the top 10).

Hide

```
ratings %>% group_by(country) %>% summarize(n=n()) %>% top_n(10,wt=n) %>%
  ggplot(aes(x=reorder(country,-n),y=n)) +
  geom_bar(stat="identity")+
  geom_text(aes(label=n),nudge_y = 1800)+
  labs(title="Reviews by country (top 10)", y="Number of reviews", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

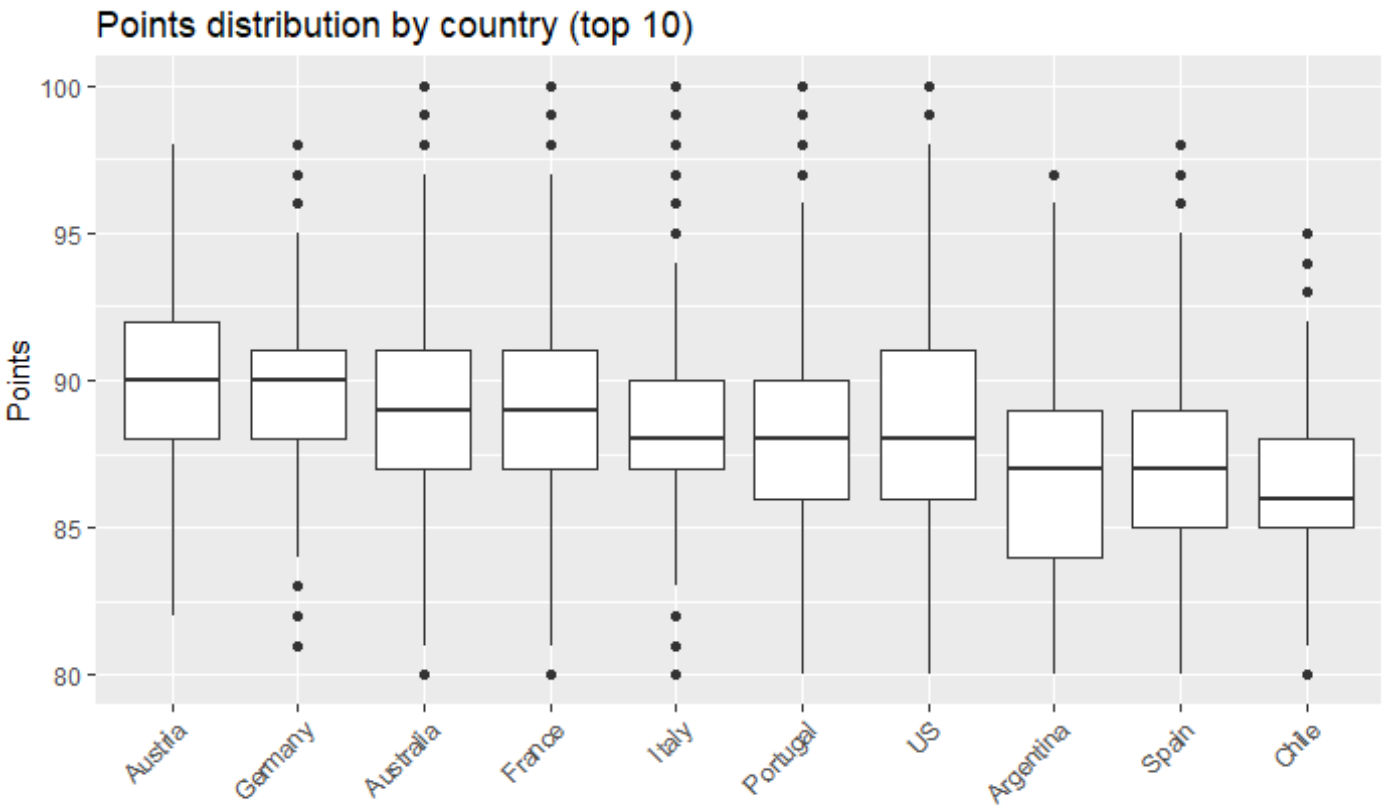


We see that this is a very US-centric dataset, with about 42% of the reviews.

We can also have a quick look at the distribution of scores by country. We notice that Austria and Germany have very high median scores and that Spain and Chile take the bottom two spots in the top 10 countries with most reviews. Additionally, in the top 10, we notice that only 5 countries have a review of 100 points: Australia, France, Italy, Portugal and the US.

Hide

```
top_10_countries <- ratings %>% group_by(country) %>% summarize(n=n()) %>% top_
n(10,wt=n) %>% .$country
ratings %>% filter(country %in% top_10_countries) %>%
  ggplot(aes(x=reorder(country,-points,FUN = median), y=points)) +
  geom_boxplot()+
  labs(title="Points distribution by country (top 10)", y="Points", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Points

As we are primarly interested about the points given to each wine, let us now examine the distribution of points.

Let's first look at a few statistics:

Hide

```
ratings %>% select(points) %>%
  summarize(Min=min(points),
            Max=max(points),
            Average=round(mean(points),2),
            Median=round(median(points),2),
            "Standard Deviation"=round(sd(points),2)) %>%
  kable(caption = "Points Statistics", align = rep("c",5)) %>%
  kable_styling(full_width = FALSE)
```

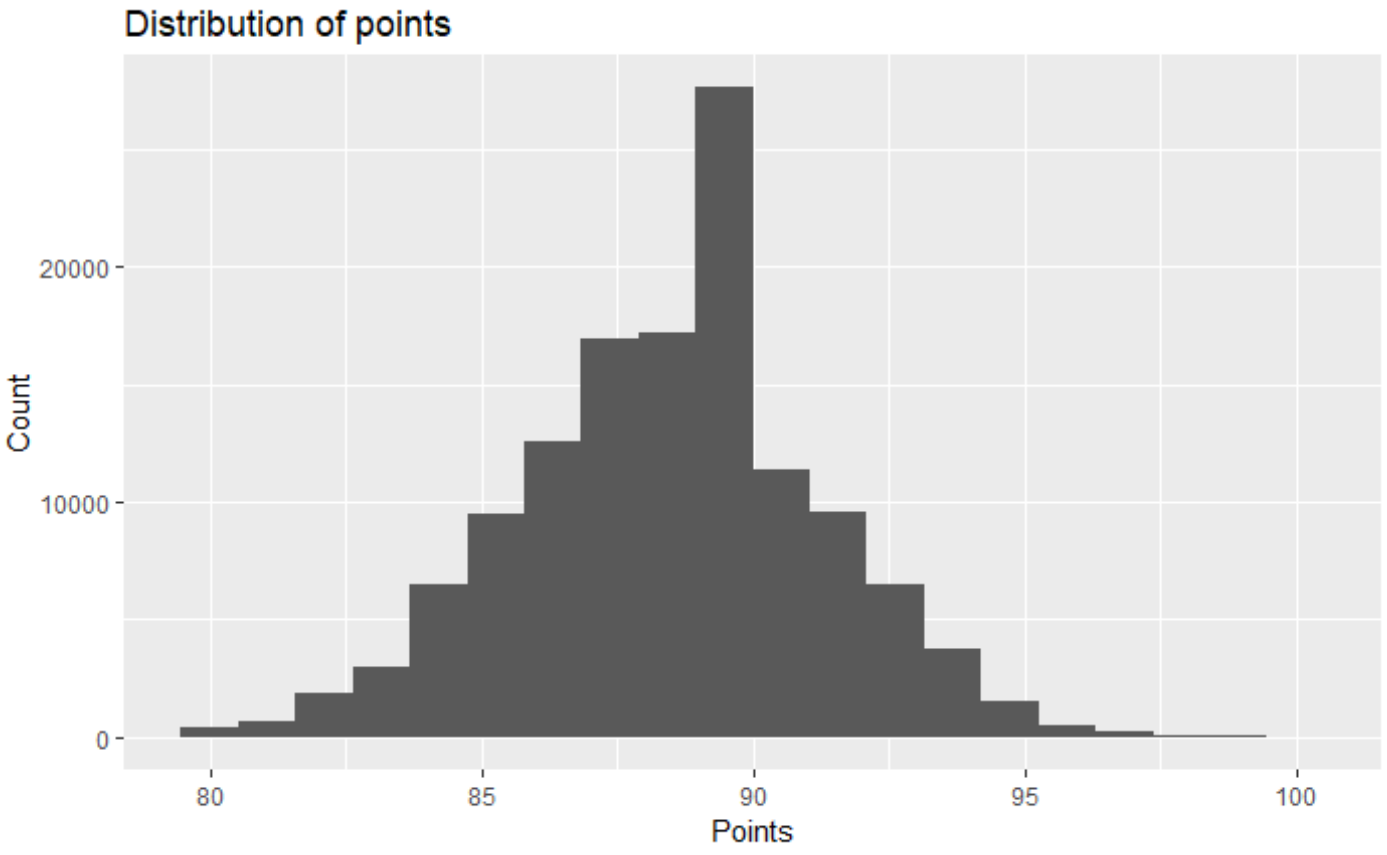
Points Statistics

Min	Max	Average	Median	Standard Deviation
80	100	88.45	88	3.04

We can get a better understanding of the distribution by looking at a histogram.

Hide

```
ratings %>% select(points) %>%
  ggplot(aes(points)) +
  geom_histogram(bins = 20)+
  labs(title="Distribution of points", x="Points", y="Count")
```



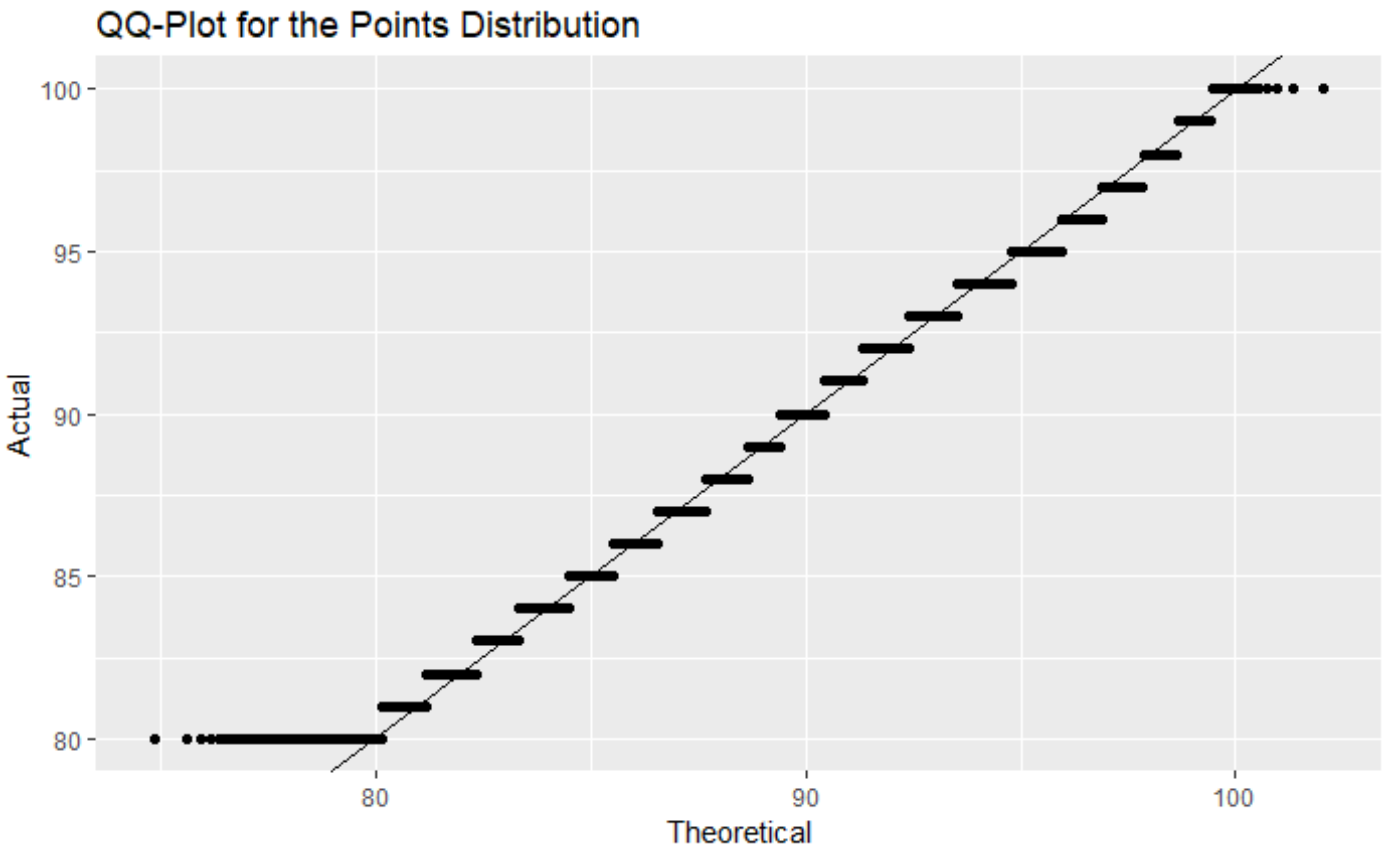
Hide

```
NA
```

We can check if the data is normally distributed with this code:

Hide

```
ratings %>% ggplot(aes(sample=points)) +
  geom_qq(dparams=summarize(ratings, mean=mean(points), sd=sd(points))) +
  geom_abline() + labs(title="QQ-Plot for the Points Distribution", y="Actual",
x="Theoretical")
```



We see from this plot that, although the data starts at 80 points and ends at 100 points, the normal distribution is a relatively good estimate between those two end points.

Price

Let us now look at the price variable, which is the only numerical variable we have at our disposal, thus far, to build our model. Let's look at a distribution.

Hide

```
ratings %>% filter(price != "") %>%
  summarize(Min=min(price),
            Max=max(price),
            Average=round(mean(price),2),
            Median=round(median(price),2),
            "Standard Deviation"=round(sd(price),2)) %>%
  kable(caption = "Price Statistics", align = rep("c",5)) %>%
  kable_styling(full_width = FALSE)
```

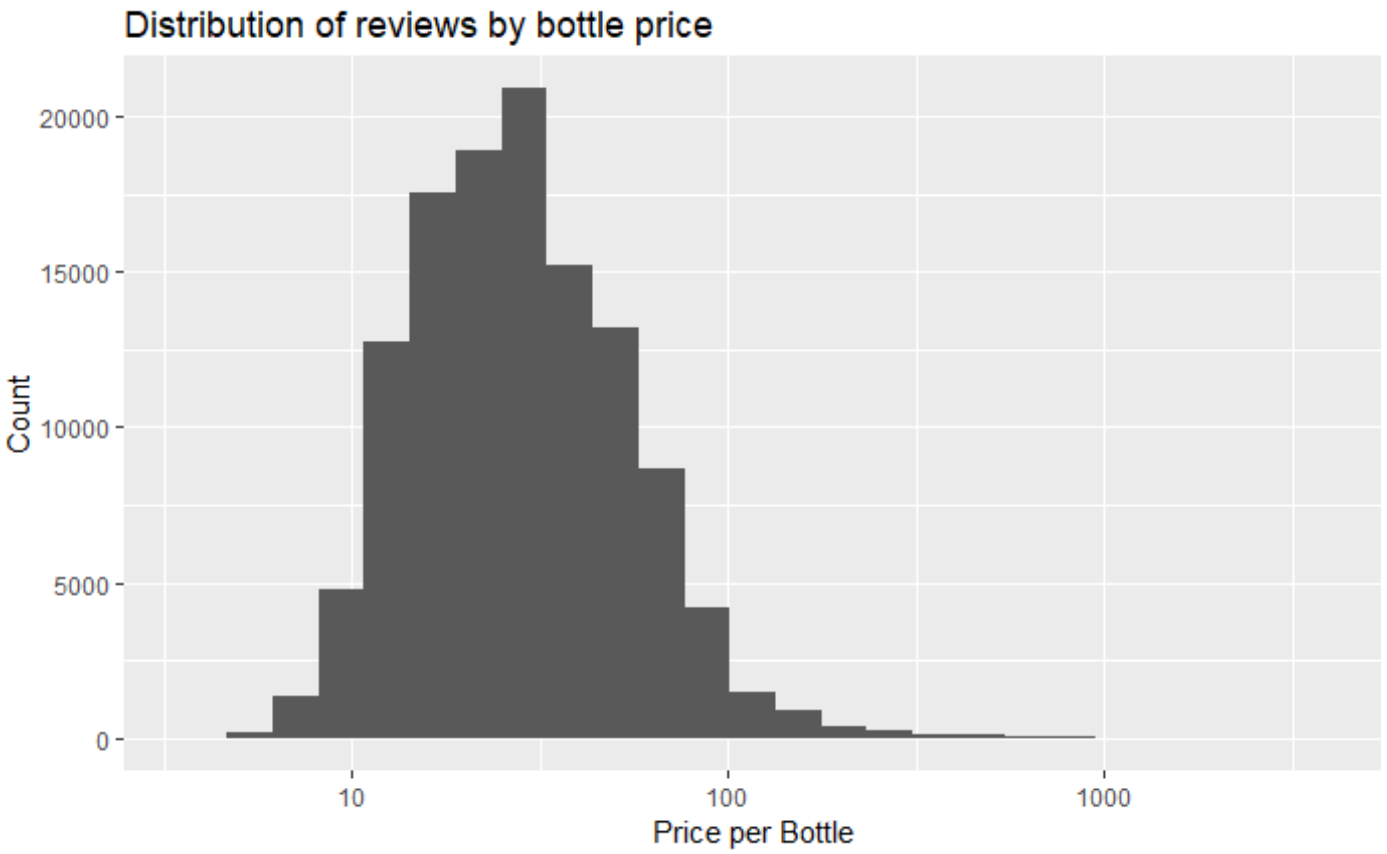
Price Statistics

Min	Max	Average	Median	Standard Deviation
4	3300	35.36	25	41.02

We can build a histogram to get a better view.

Hide

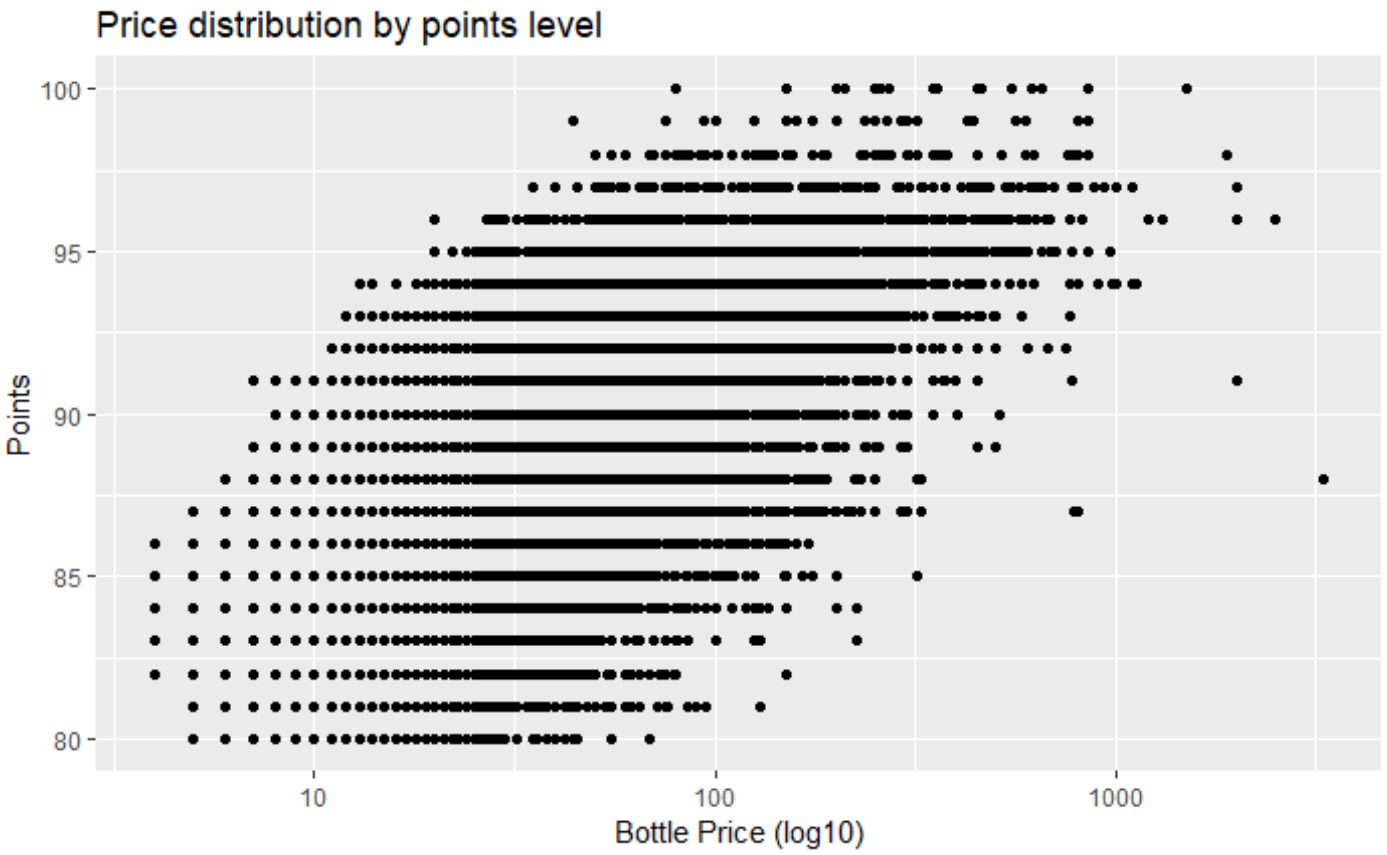
```
ratings %>% filter(price != "") %>%
  ggplot(aes(price)) +
  geom_histogram(bins=25)+
  scale_x_log10()+
  labs(title="Distribution of reviews by bottle price", y="Count", x="Price per
Bottle")
```



Now, the question on everyone's lip is whether there is a correlation between price and quality (points). Let's look at a scatterplot.

Hide

```
ratings %>% filter(price !="") %>%
  ggplot(aes(x=price,y=points))+
  geom_point()+
  scale_x_log10()+
  labs(title="Price distribution by points level", y="Points", x="Bottle Price
 (log10)")
```

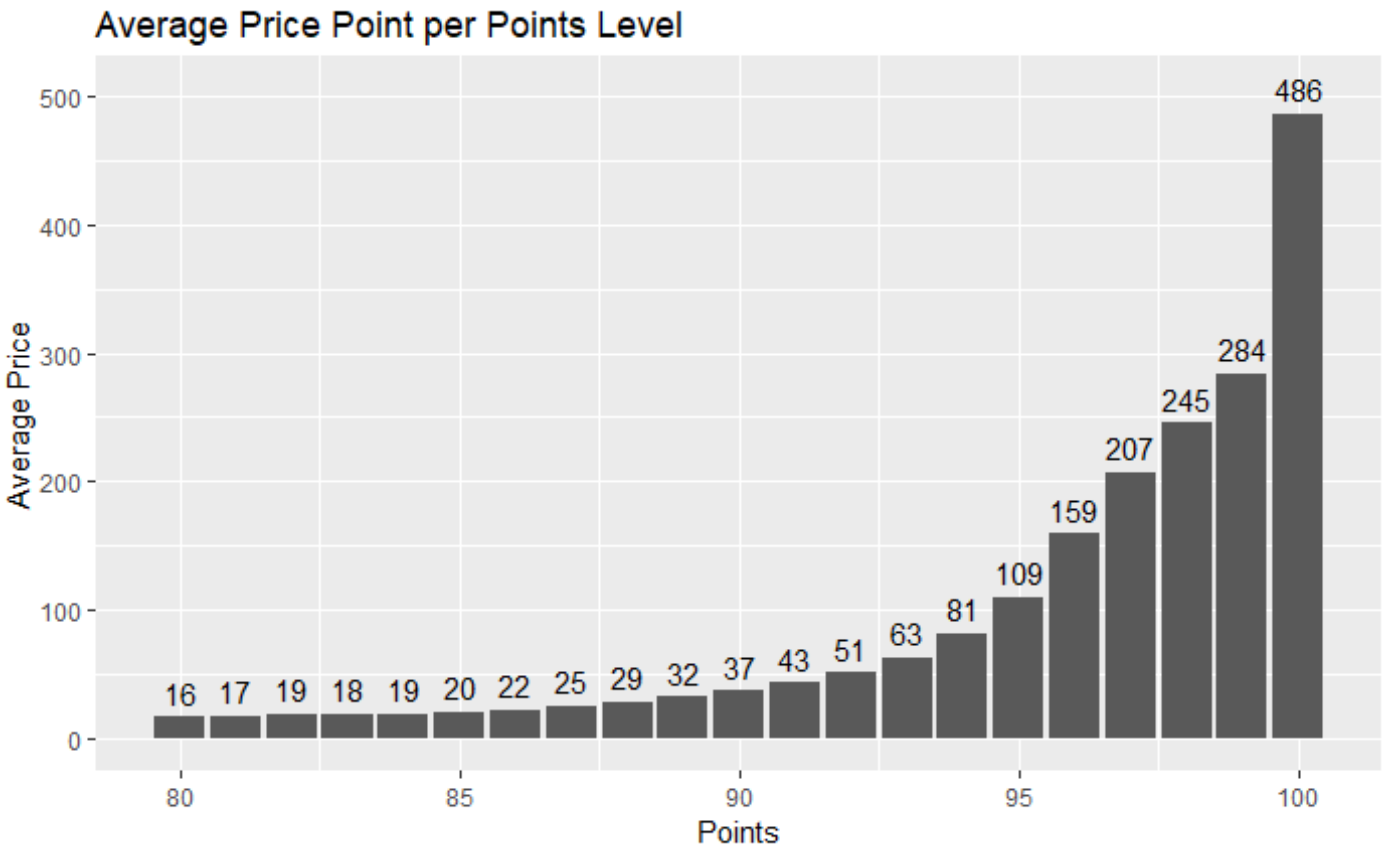


We can see from this graph that there does seem to be a slight correlation, with most of the very low scores in the low-end of the price range. We also notice how, as we go up in points, we tend to shift to the right ot the price range.

To better understand this phenomenon, let's look at the average price at each point level.

Hide

```
ratings %>% filter(price != "") %>% group_by(points) %>%
  summarise(`Average Price`=mean(price)) %>%
  ggplot(aes(x=points, y=`Average Price`)) +
  geom_bar(stat = "identity")+
  geom_text(aes(label=round(`Average Price`)),nudge_y = 20)+
  labs(title="Average Price Point per Points Level", x="Points")
```



Here, we clearly see that bottles over and above 90 points tend to be much more expensive. On average, bottles with 95+ points cost more than $100. Only very expensive bottles receive 100 points.

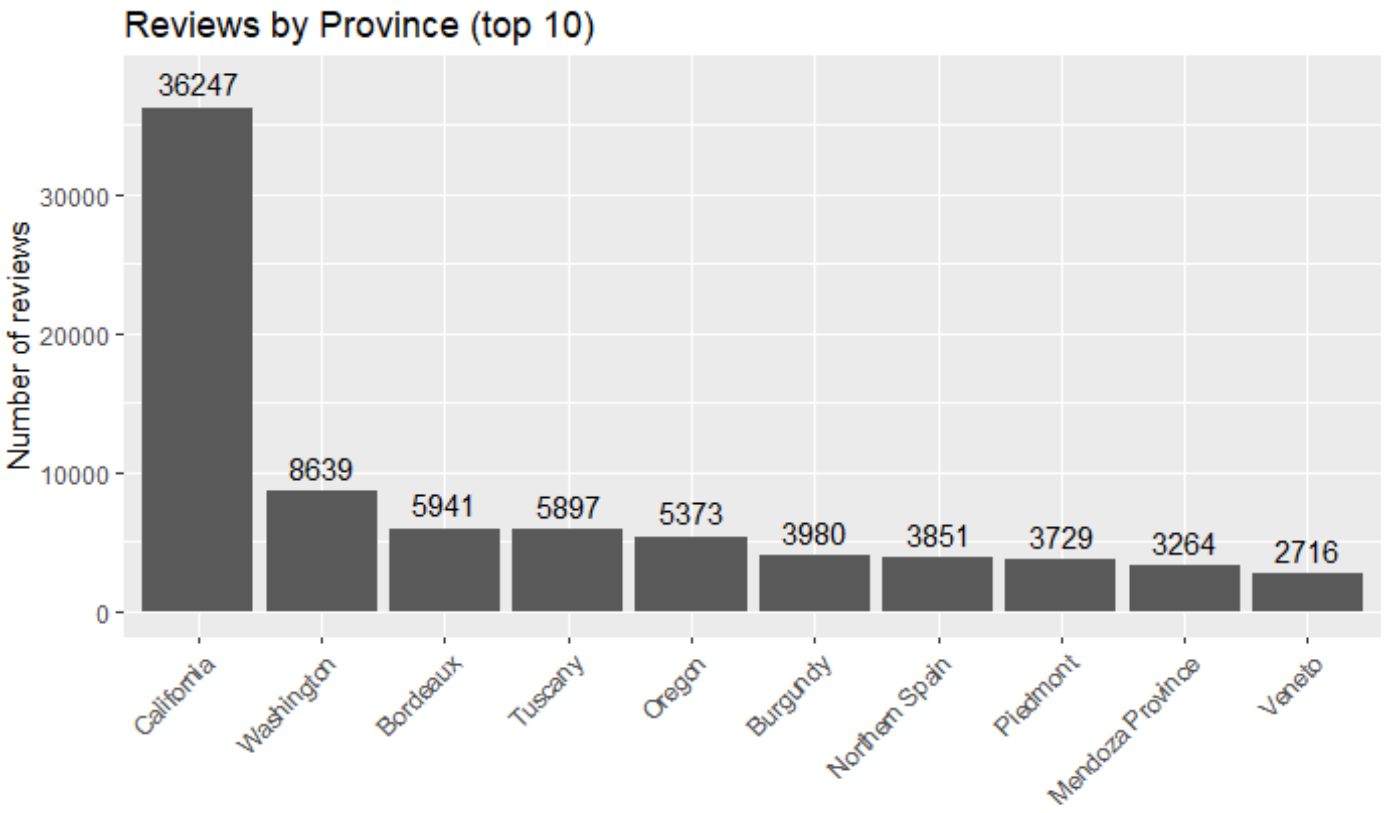
Province

Most of the time, however, knowing the country and price of a wine isn't sufficient to determine what to expect. For instance, a CÃ´te-du-Rhone from France can be expected to taste very different from a Bordeaux or a Burgundy, and therefore affect the ratings..

Let us see which *provinces* are the most prevalent in our data.

Hide

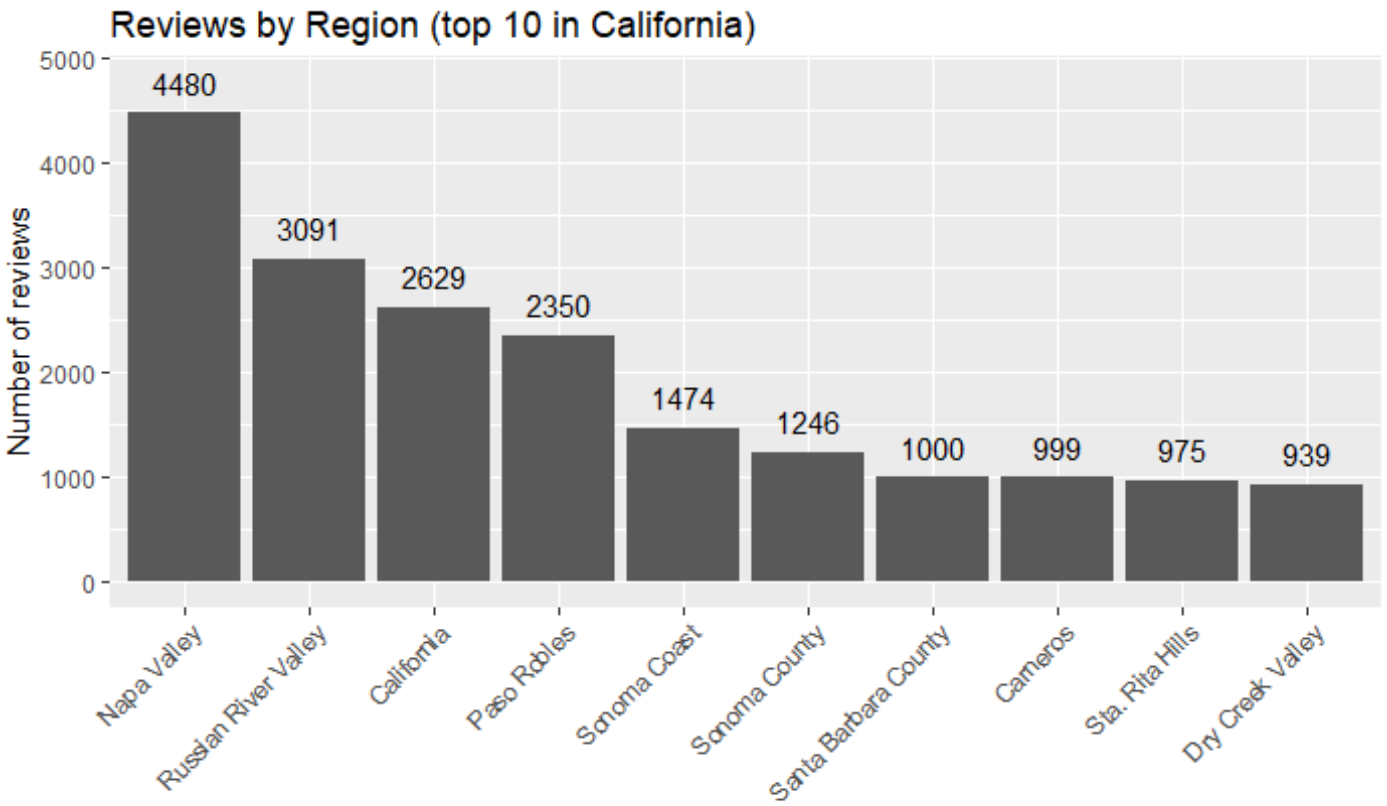
```
ratings %>% group_by(province) %>% summarize(n=n()) %>% top_n(10,wt=n) %>%
  ggplot(aes(x=reorder(province,-n),y=n)) +
  geom_bar(stat="identity")+
  geom_text(aes(label=n),nudge_y = 1800)+
  labs(title="Reviews by Province (top 10)", y="Number of reviews", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Again, we notice how skewed our data is towards the US, with California in particular. Within California, we see Napa, Russian River Valley, Paso Robles and Sonoma have a very strong showing. Weirdly enough, “California” appear again in this variable.

Hide

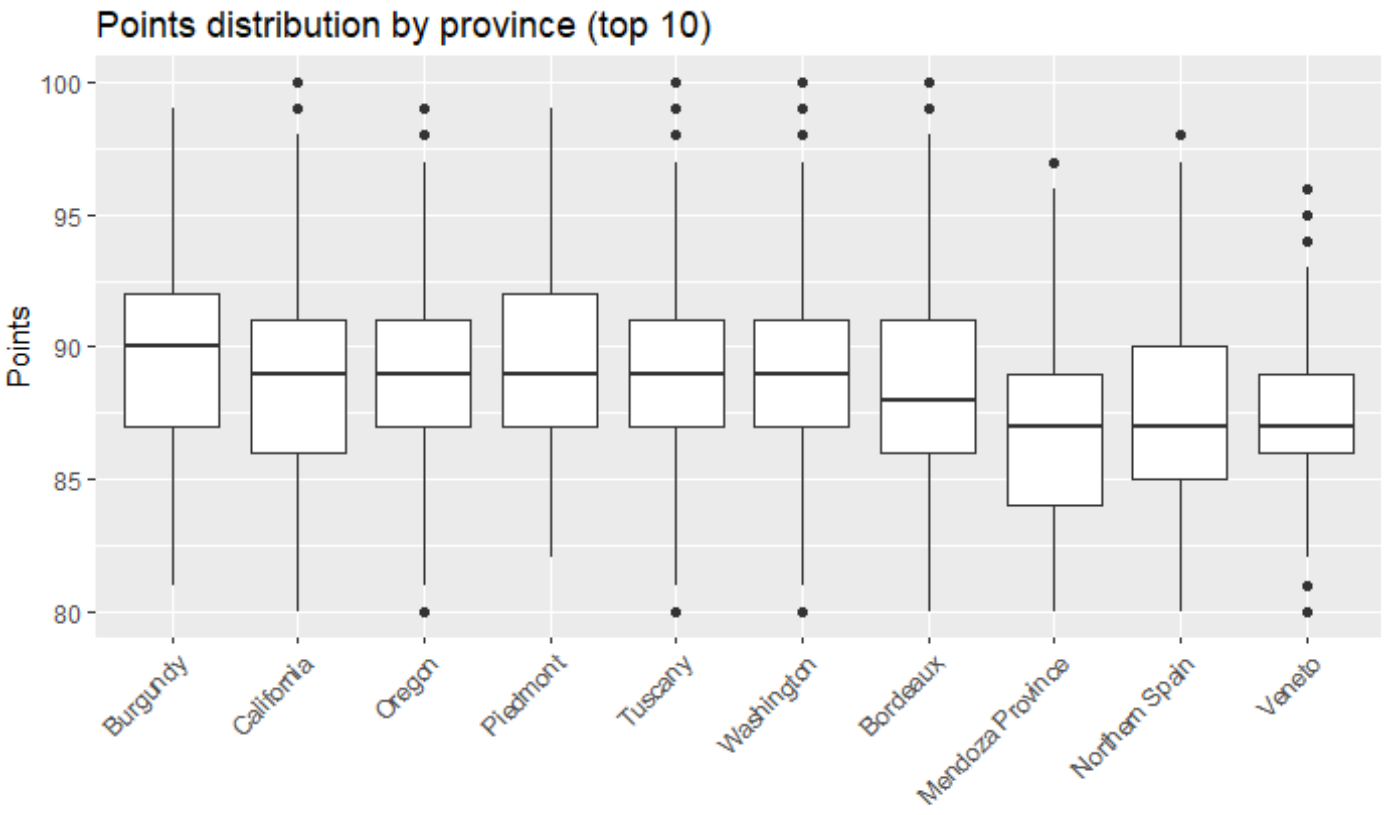
```
ratings %>% filter(province=="California") %>%
  group_by(region_1) %>% summarize(n=n()) %>% top_n(10,wt=n) %>%
  ggplot(aes(x=reorder(region_1,-n),y=n)) +
  geom_bar(stat="identity")+
  geom_text(aes(label=n),nudge_y = 300)+
  labs(title="Reviews by Region (top 10 in California)", y="Number of reviews",
x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Back to the whole dataset, let's examine the score distribution by province for the top 10.

Hide

```
top_10_province <- ratings %>% group_by(province) %>% summarize(n=n()) %>% top_
n(10,wt=n) %>% .$province
ratings %>% filter(province %in% top_10_province) %>%
  ggplot(aes(x=reorder(province,-points,FUN = median), y=points)) +
  geom_boxplot()+
  labs(title="Points distribution by province (top 10)", y="Points", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Taster

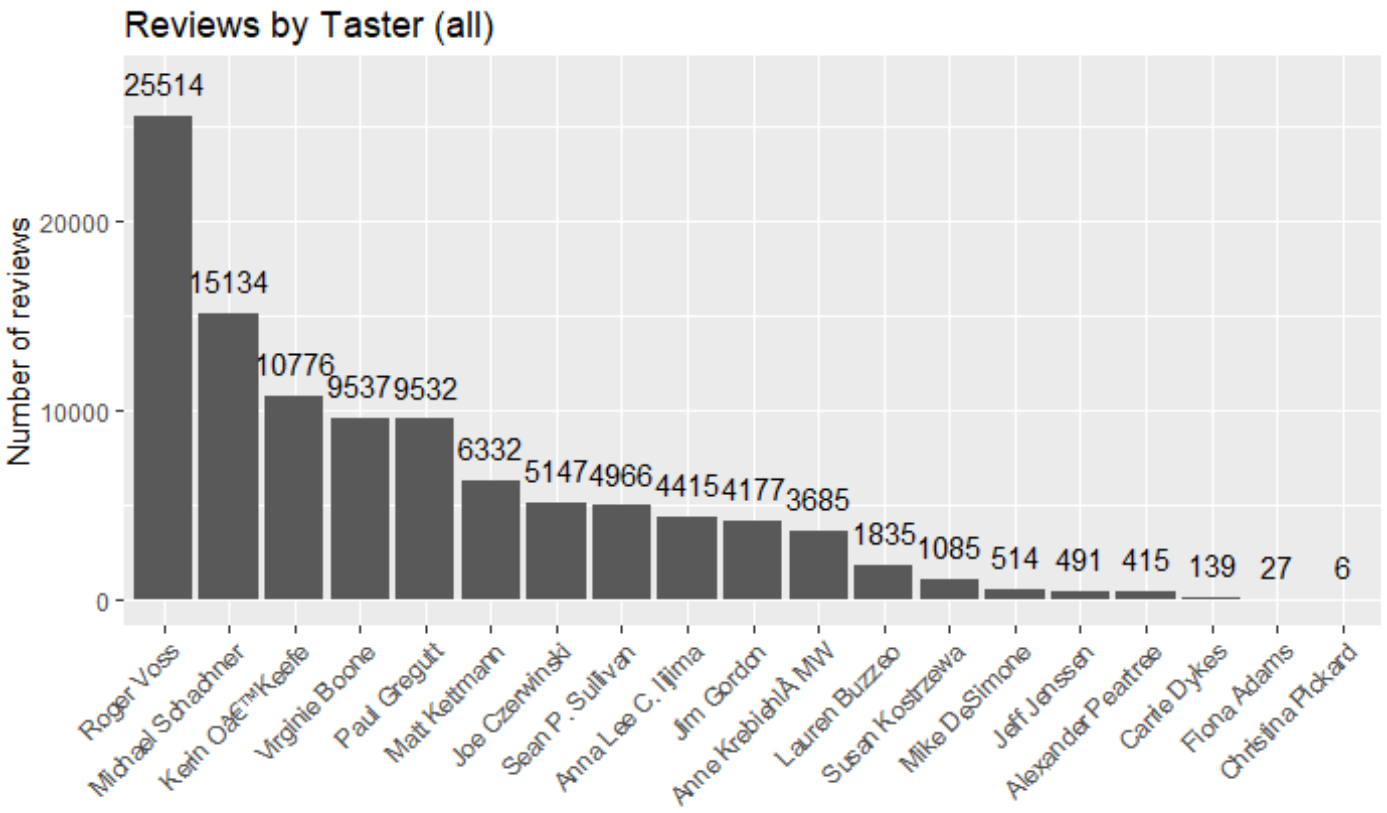
The next variable we will examine is who exactly wrote the review. This information is held in the **taster_name** variable.

There are 20 unique values in this variable, one of which is the absence of the rater's name. We will exclude those observations in the following analysis.

Let us see who was most prolific.

Hide

```
ratings %>% filter(taster_name != "") %>% group_by(taster_name) %>% summarize(n
=n()) %>%
  ggplot(aes(x=reorder(taster_name,-n),y=n)) +
  geom_bar(stat="identity")+
  geom_text(aes(label=n),nudge_y = 1800)+
  labs(title="Reviews by Taster (all)", y="Number of reviews", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

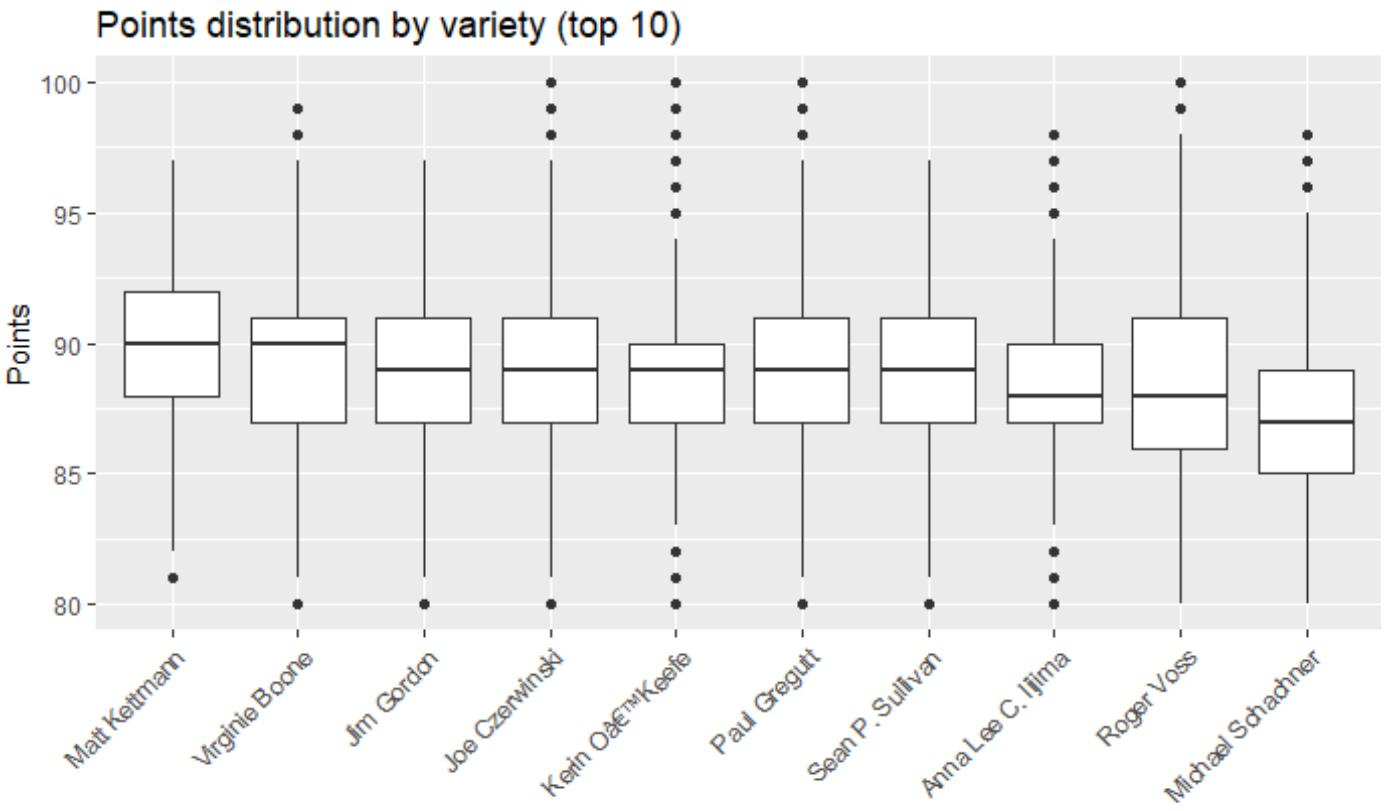


We can appreciate that Roger Voss, Michael Schadner and Kerin O'Keefe have a very strong contribution to the dataset. Especially Mr. Voss… that is a lot of wine.

Let us look now at a distribution of score by taster. We will only use the top 10 at this stage.

Hide

```
top_10_taster <- ratings %>% filter(taster_name!="") %>% group_by(taster_name)
 %>% summarize(n=n()) %>% top_n(10,wt=n) %>% .$taster_name
ratings %>% filter(taster_name %in% top_10_taster) %>%
  ggplot(aes(x=reorder(taster_name,-points,FUN = median), y=points)) +
  geom_boxplot()+
  labs(title="Points distribution by variety (top 10)", y="Points", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



From this graph, we can see that some tasters tend to grade wines higher (like Matt Kettmann), and others more hasrshly (like Roger Voss and Michael Schachner).

Vintage

From the title of the review, we are often able to extract the vintage of the wine that was tasted.

For instance, wine 353 has this title: Rochioli 2014 South River Chardonnay (Russian River Valley). It is a wine from the vintage 2004.

Let us try to do this systematically. This analysis will not be perfect, but wil give us a very good idea nevertheless.

Hide

```
year_pattern <- "\\d\\d\\d\\d"
ratings <- ratings %>% mutate(Vintage = as.numeric(str_extract(title,year_patte
rn)))
# Remove anything that could be an error. We will assume that no wine in the da
taset was made before 1910.
ratings$Vintage <-  ifelse(ratings$Vintage>2018,"",ratings$Vintage)
ratings$Vintage <-  ifelse(ratings$Vintage<1910,"",ratings$Vintage)
# Create decades to make visualization easier
ratings <- ratings %>%
  mutate(Decade =
           ifelse(ratings$Vintage %in% 1900:1909, "1900s",
           ifelse(ratings$Vintage %in% 1910:1919, "1910s",
           ifelse(ratings$Vintage %in% 1920:1929, "1920s",
           ifelse(ratings$Vintage %in% 1930:1939, "1930s",
           ifelse(ratings$Vintage %in% 1940:1949, "1940s",
           ifelse(ratings$Vintage %in% 1950:1959, "1950s",
           ifelse(ratings$Vintage %in% 1960:1969, "1960s",
           ifelse(ratings$Vintage %in% 1970:1979, "1970s",
           ifelse(ratings$Vintage %in% 1980:1989, "1980s",
           ifelse(ratings$Vintage %in% 1990:1999, "1990s",
           ifelse(ratings$Vintage %in% 2000:2009, "2000s",
           ifelse(ratings$Vintage %in% 2010:2019, "2010s",
                  "")))))))))))))
```

The above code added two variables to our dataset: Vintage and Decade.

Let's now look at this last variable:

Hide

```
ratings %>% filter(Decade != "") %>% group_by(Decade) %>%
  summarise(Reviews = n(), `Average Score`=round(mean(points),2)) %>%
  kable(align = rep("c",3)) %>%
  kable_styling(full_width = FALSE)
```

Decade	Reviews	Average Score
1910s	9	89.89
1920s	8	88.12
1930s	2	93.50

Decade	Reviews	Average Score
1940s	3	93.67
1950s	3	95.33
1960s	12	94.17
1970s	5	90.80
1980s	28	90.86
1990s	1661	87.99
2000s	37896	88.24
2010s	85566	88.60

Although earlier decades seem to display higher scores, this is likely due to the fact that there are only a few reviews for these years, of very good bottles.

For curiosity, let's look at the average score for the region of Bordeaux for the years 2005+. For anyone with wine knowledge, the results won't be surprising: 2009, 2008 and 2010 are very high on the list. I must say that 2015 being second to last is unexpected however.

Hide

```
ratings %>% filter(Vintage>2005 & province=="Bordeaux") %>%
  group_by(Vintage) %>%
  summarize("Average Score"=round(mean(points),2), Reviews = n()) %>%
  arrange(-`Average Score`) %>%
  kable(align = rep("c",3)) %>%
  kable_styling(full_width = F)
```

Vintage	Average Score	Reviews
2009	90.18	473
2008	89.75	287
2010	89.40	733

Vintage	Average Score	Reviews
2006	89.14	272
2007	88.94	179
2014	88.73	966
2012	88.67	824
2011	88.17	726
2013	87.85	506
2015	87.26	525
2016	86.64	218

Variety

Let us now turn our attention to which grape varietals are represented.

We can compute how many different grape varietals are represented.

Hide

```
length(unique(ratings$variety))
```

```
[1] 708
```

To the untrained eye, 708 may seem like a lot, but let us remember that there are approximately 10,000 grape varietals in the world. Of course, not all of them are frequently found in wine. Let’s look at the top 12 varieties.

Hide

```
ratings %>%
  group_by(variety) %>%
  summarize(Reviews=n()) %>%
  top_n(12,Reviews) %>%
  arrange(desc(Reviews)) %>%
  select(Variety=variety,Reviews=Reviews) %>%
  kable(align = rep("c",2)) %>%
  kable_styling(full_width = FALSE)
```

Variety	Reviews
Pinot Noir	13272
Chardonnay	11753
Cabernet Sauvignon	9472
Red Blend	8946
Bordeaux-style Red Blend	6915
Riesling	5189
Sauvignon Blanc	4967
Syrah	4142
RosÃ©	3564
Merlot	3102
Nebbiolo	2804
Zinfandel	2714

We notice that the most common variety are Pinot Noir, Chardonnay and Cabernet Sauvignon. This is not surprising as those grapes are very common in the US. We then see “Red Blend” and “Bordeaux-style Red Blend”. The former is a bit dissapointing as Red Blend could litteraly mean anything. Let’s see where those wines come from:

Hide

```
ratings %>% filter(variety=="Red Blend") %>%
  group_by(country) %>%
  summarize(Reviews =n()) %>%
  select(Country=country,Reviews=Reviews) %>%
  top_n(5,Reviews) %>%
  arrange(desc(Reviews)) %>%
  kable(align = rep("c",2)) %>%
  kable_styling(full_width = FALSE)
```

Country	Reviews
Italy	3624
US	2972
Spain	818
Chile	409
France	304

We are a bit surprised that Italy would have so many “Red Blends” since there are many *appelations* that would have made the data more specific. However, we are not surprised to see the US high on this list given the flexibility of winemaking regulations in the country.

For wines of the old world, France for instance, we expect to see specific blends. For instance, all wines in Bordeaux would have some concentration of Cabernet Sauvignon, Cabernet Franc and Merlot, and perhaps a bit of Petit Verdot. Let’s see.

Hide

```
ratings %>% filter(province=="Bordeaux") %>%
  group_by(variety) %>%
  summarize(Reviews=n()) %>%
  select(Variety=variety,Reviews) %>%
  top_n(5,Reviews) %>%
  arrange(desc(Reviews)) %>%
  kable(align = rep("c",2)) %>%
  kable_styling(full_width = FALSE)
```

Variety	Reviews

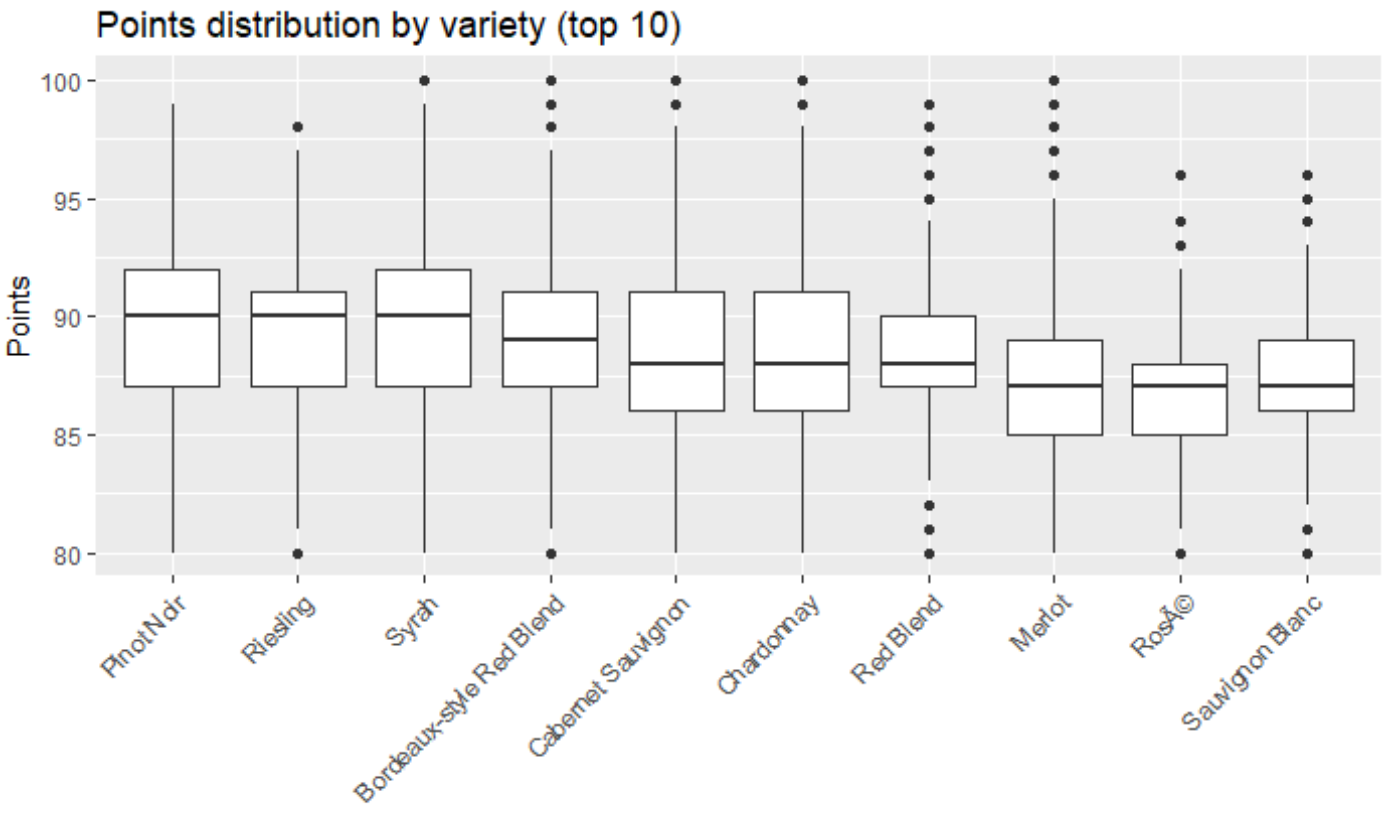
Variety	Reviews
Bordeaux-style Red Blend	4617
Bordeaux-style White Blend	951
RosÃ©	191
Merlot	72
Sauvignon Blanc	68

Our intuition is confirmed.

Let us now look at the top 10 varieties and their score distribution.

Hide

```
top_10_varieties <- ratings %>% group_by(variety) %>% summarize(n=n()) %>% top_
n(10,wt=n) %>% .$variety
ratings %>% filter(variety %in% top_10_varieties) %>%
  ggplot(aes(x=reorder(variety,-points,FUN = median), y=points)) +
  geom_boxplot()+
  labs(title="Points distribution by variety (top 10)", y="Points", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

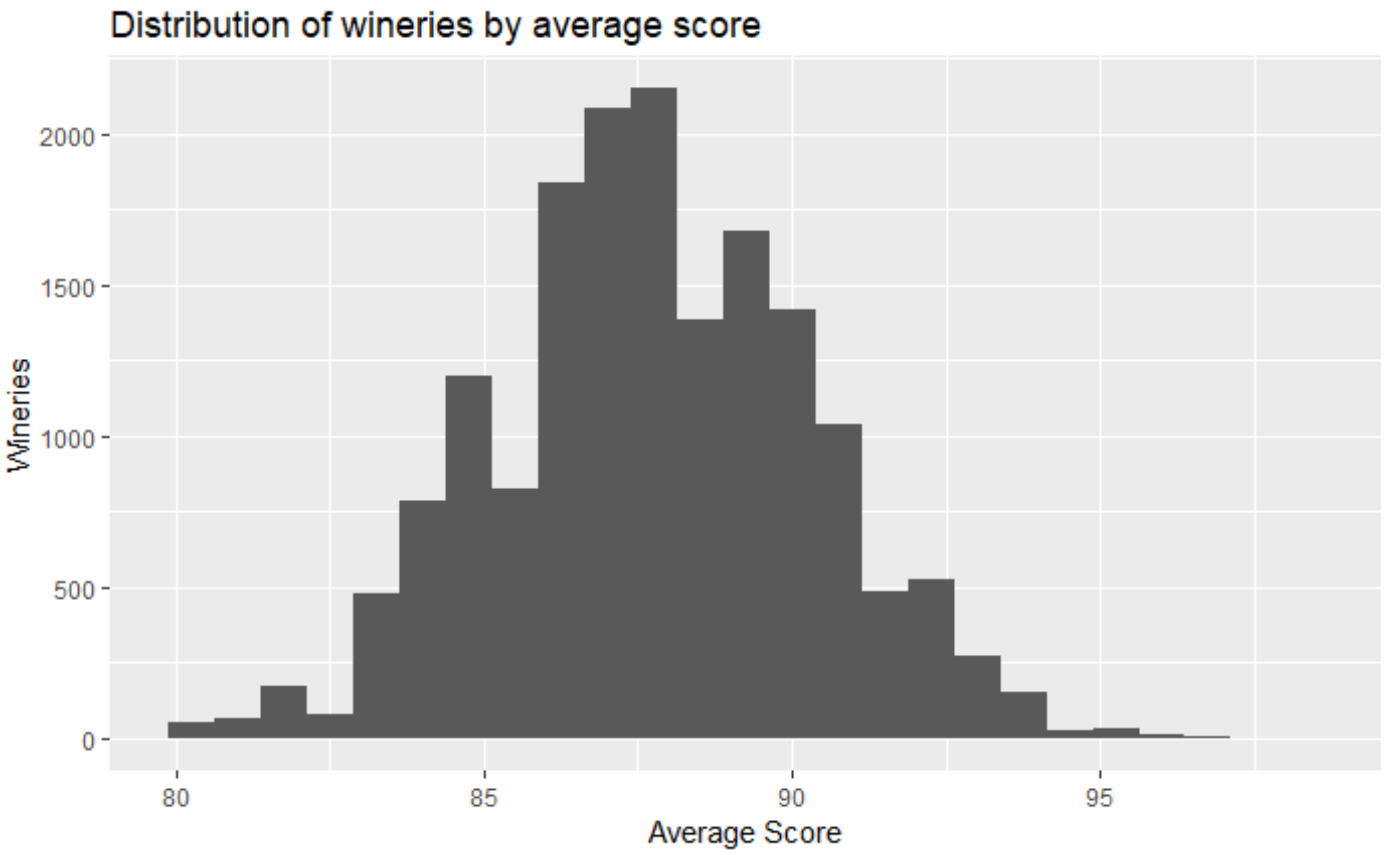


Winery

We know that some wineries have a very strong reputation and tend to systematically attract more points. Let us look at the distribution of the average score by winery.

Hide

```
ratings %>% group_by(winery) %>% summarize(Avg=mean(points)) %>%
  ggplot(aes(Avg)) +
  geom_histogram(bins=25)+
  labs(title="Distribution of wineries by average score", y="Wineries", x="Aver
age Score")
```



Of course, it is possible that some of the extreme cases here are wineries that are only seldom reviewed. The average number of review per winery is:

Hide

```
ratings %>% group_by(winery) %>% summarise(n=n()) %>% ungroup() %>% summarise(A
verage=mean(n)) %>% .$Average
```

```
[1] 7.756221
```

At this point, it is worth looking at the wineries with an average score above 95 points, with at least 10 reviews. That's an impressive showing, and we wouldn't hesitate to get our hands on a bottle from Tenuta dell'Ornellaia if we could buy one. Here is their website (http://www.ornellaia.com/en/), for anyone interested.

Hide

```
ratings %>% group_by(winery) %>%
  summarise(Avg=mean(points),Reviews=n(), Price = round(mean(price),2)) %>%
  filter(Avg>95 & Reviews>=8) %>% arrange(-Avg) %>%
  select(Winery=winery, `Average Score`=Avg, Reviews = Reviews, `Average Bottle
Price`=Price) %>%
  kable(align = rep("c",3)) %>%
  kable_styling(full_width = F)
```

Winery	Average Score	Reviews	Average Bottle Price
Tenuta dell'Ornellaia	96.70000	10	286.50
ChÃ¢teau LÃ©oville Barton	95.66667	9	NA
Wayfarer	95.33333	18	111.67
Mascarello Giuseppe e Figlio	95.20000	10	174.30
Horsepower	95.09091	11	116.82

NLP on the Description Variable

In the following section, we will dive into the content of the **description** variable, which contains the text of the review written by the tasters. We will use tecniques of text mining and natural language processing inspired by the tutorial posted by Debbie Liske at this link (https://www.datacamp.com/community/tutorials/R-nlp-machine-learning).

We now create a quick function to get rid of English contractions

Hide

```
fix.contractions <- function(doc) {
  # "won't" is a special case as it does not expand to "wo not"
  doc <- gsub("won't", "will not", doc)
  doc <- gsub("can't", "can not", doc)
  doc <- gsub("n't", " not", doc)
  doc <- gsub("'ll", " will", doc)
  doc <- gsub("'re", " are", doc)
  doc <- gsub("'ve", " have", doc)
  doc <- gsub("'m", " am", doc)
  doc <- gsub("'d", " would", doc)
  # 's could be 'is' or could be possessive: it has no expansion
  doc <- gsub("'s", "", doc)
  return(doc)
}
```

Let's apply it, and convert everything to lower case.

Hide

```
ratings$description <- sapply(ratings$description,fix.contractions)
ratings$description <- tolower(ratings$description)
```

Let us now create a data frame in a tidy format, where each word has a row. We use the **udpipe** package to also lemmatize each word (e.g., aromas = aroma) and get its Part of Speech (e.g., adjective, noun, etc.). I anti-join the dataset *stop_words* to get rid of overly common words like "where", "has", "yet", etc. Finally, I only keep words with 3 characters or more, since most small words do not reveal that much meaning.

For the sake of speeding things up in the future, I save a copy of this data frame, which I will upload onto my Github repository (https://github.com/jasbeausejour/EDX_Updated_Movie_Project).

Because of how long the above step takes, I've decided to save a copy of that data frame to my Github repository (https://github.com/jasbeausejour/EDX_Updated_Movie_Project). I've deactivated the above code and instead am downloading the file here.

Hide

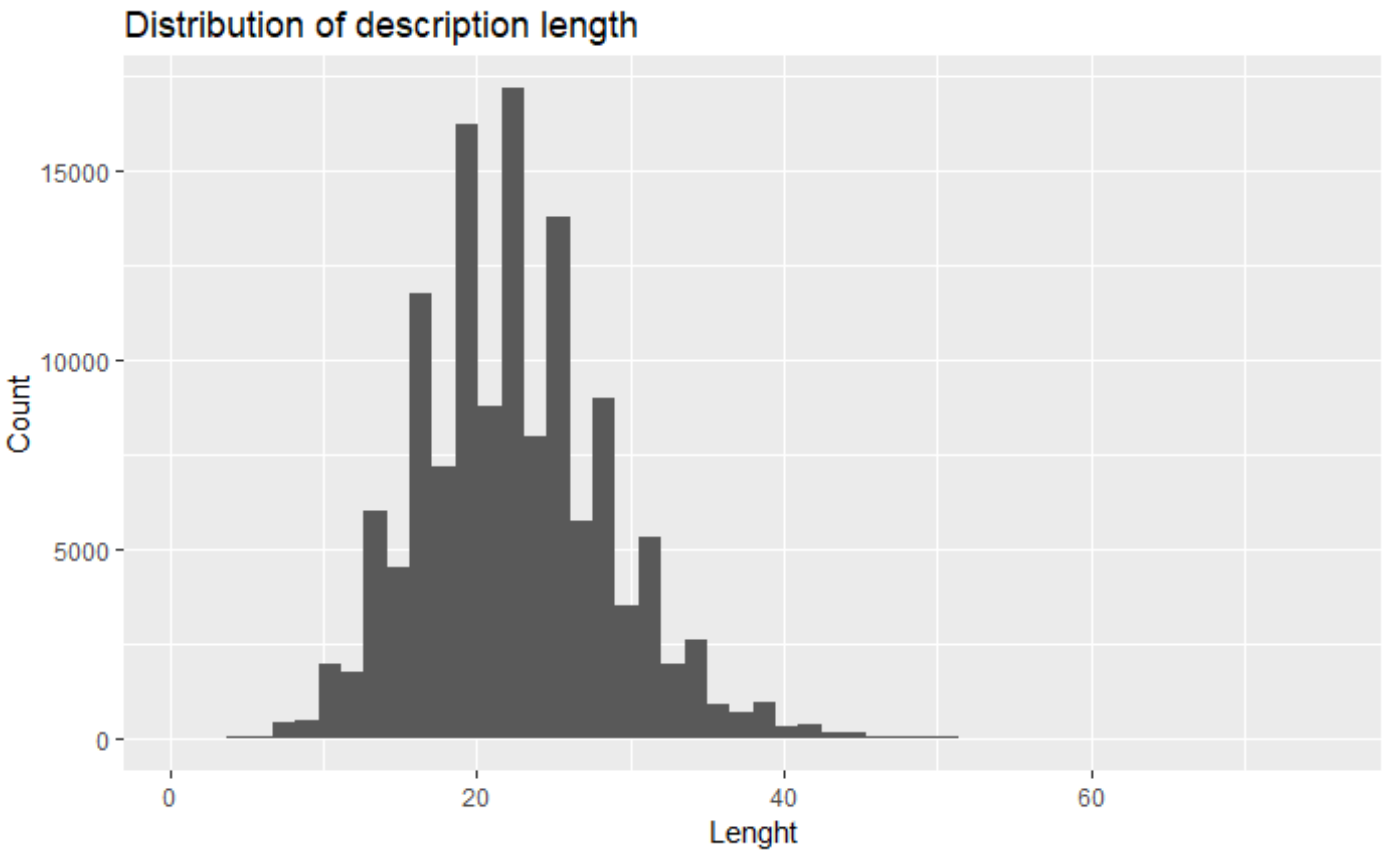
```
githubURL <- url("https://github.com/jasbeausejour/Wine_Updated_Project/blob/ma
ster/Data/words_in_description.rda?raw=true")
print(load(githubURL))
```

```
[1] "words_in_descriptions"
```

The first thing we can look at is the lenght of the descriptions.

Hide

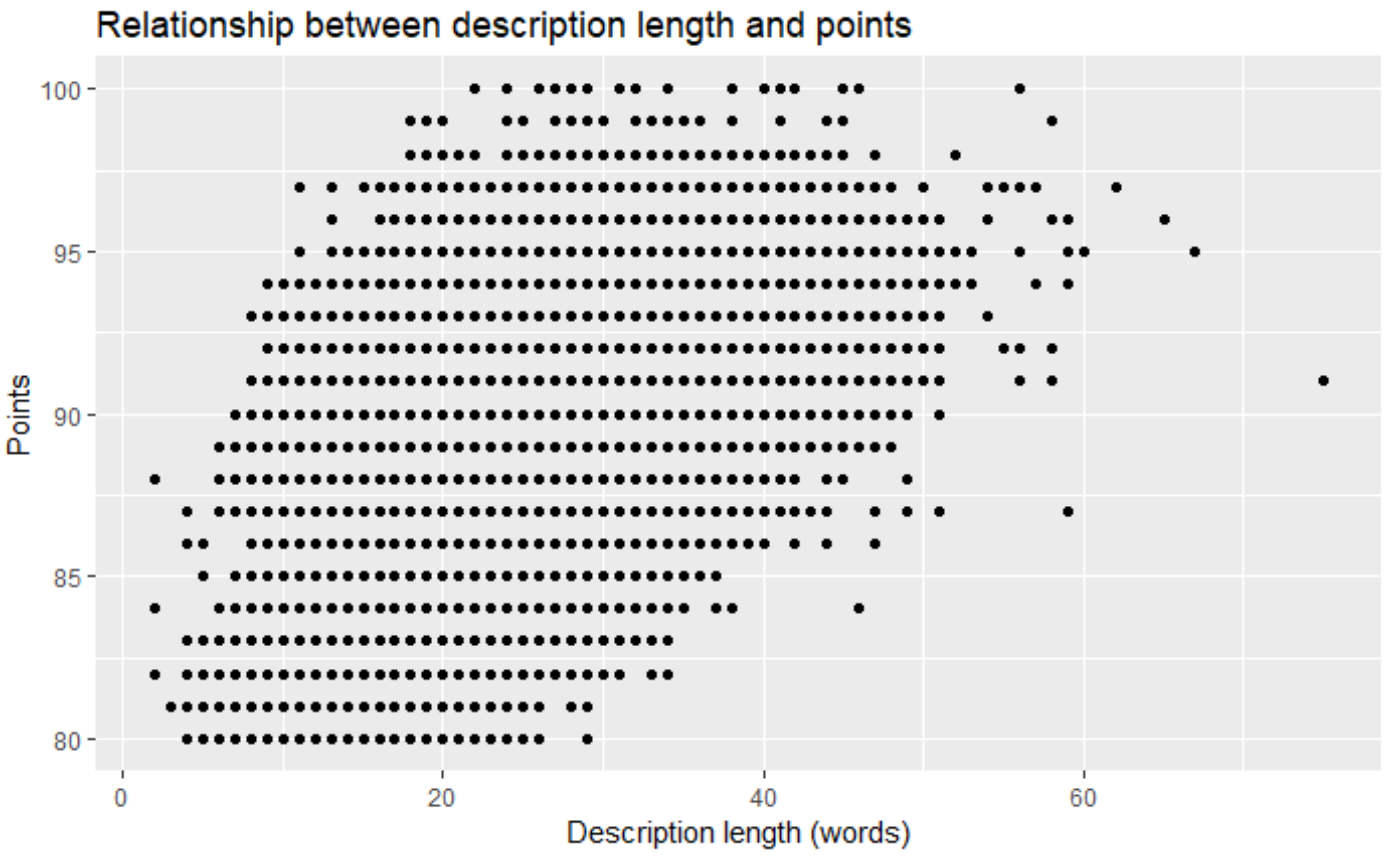
```
length_of_decription <- words_in_descriptions %>% group_by(doc_id) %>% summaris
e(Length=n()) %>% mutate(X=as.integer(doc_id)) %>% select(X,Length)
length_of_decription %>% ggplot(aes(x=Length))+
  geom_histogram(bins=50)+
  labs(title="Distribution of description length",x="Lenght",y="Count")
```



Let us see if there seems to be a relationship between the length of the descriptions and the points.

Hide

```
# Adding the length of description as a variable in ratings
ratings <- ratings %>% left_join(length_of_decription,by = "X")
ratings %>% ggplot(aes(x=Length,y=points))+
  geom_point()+
  labs(title="Relationship between description length and points", x="Descripti
on length (words)", y="Points")
```



Next, let's look at the most popular root words being used.

Hide

```
words_in_descriptions %>% group_by(lemma) %>%
  summarise(Count=n()) %>%
  arrange(desc(Count)) %>%
  top_n(10,Count) %>%
  select(`Root Word`=lemma,Count) %>%
  kable(align = c("c","c")) %>%
  kable_styling(full_width = F)
```

Root Word	Count
wine	83095
flavor	69925
fruit	63864
aroma	40670
finish	39181

Root Word	Count
palate	38537
acidity	35000
cherry	33573
drink	33181
tannin	32971

Of course, the word wine and flavor appear quite a lot. After that, most of the common words are very important descriptors of wine. Apart from **aroma**, however, we notice that tasters mostly focus on the mouth elements of the wine, while the visual aspects and the nose are being left out slightly.

We can also take a look at what nouns and adjectives are often used together.

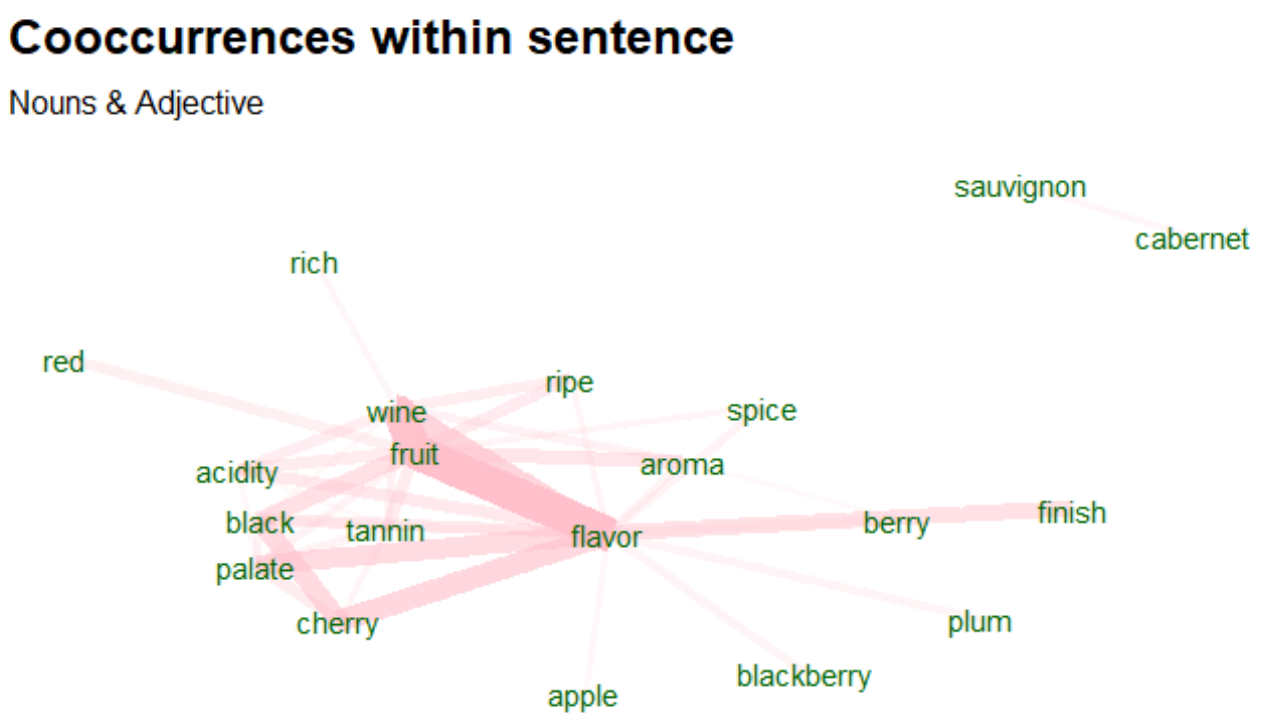
Hide

```
cooc <- cooccurrence(x = subset(words_in_descriptions, upos %in% c("NOUN", "AD
J")),
                     term = "lemma",
                     group = c("doc_id", "paragraph_id", "sentence_id"))
kable(cooc[1:5,],align = rep("c",3)) %>%
  kable_styling(full_width = F)
```

term1	term2	cooc
flavor	fruit	17681
fruit	wine	16809
flavor	wine	13269
cherry	flavor	12945
black	cherry	12212

Hide

```
wordnetwork <- head(cooc, 40)
wordnetwork <- graph_from_data_frame(wordnetwork)
ggraph(wordnetwork, layout = "fr") +
  geom_edge_link(aes(width = cooc, edge_alpha = cooc), edge_colour = "pink") +
  geom_node_text(aes(label = name), col = "darkgreen", size = 4) +
  theme_graph(base_family = "Arial") +
  theme(legend.position = "none") +
  labs(title = "Cooccurrences within sentence", subtitle = "Nouns & Adjective")
```



From the above, we can clearly see that it is very common for a wine to have cherry flavor or/and black fruit flavor. Other words that are often associated with flavor are: spice, finish (as in the wine's finish), palate, ripe, acidity and tannins.

Let's see which words are within the top 50 most common for bottles scoring 95+, but not within the top 50 for bottles scoring lower. These words may turn out to be very good predictors of a bottle's score.

Hide

```
words_in_descriptions <- words_in_descriptions %>% mutate(X=as.integer(doc_id))
%>%
  left_join(ratings, by = "X") %>% select(X,word,lemma,upos,points,price)
top_50_good <- words_in_descriptions %>% filter(points>=95) %>% group_by(lemma)
%>%
  summarise(Count=n()) %>%
  arrange(desc(Count)) %>%
  top_n(50,Count)
top_50_lessthangood <- words_in_descriptions %>% filter(points<95) %>% group_by
(lemma) %>%
  summarise(Count=n()) %>%
  arrange(desc(Count)) %>%
  top_n(50,Count)
good_predictor_words <- top_50_good %>% anti_join(top_50_lessthangood, by="lemm
a") %>% .$lemma
print(good_predictor_words)
```

```
 [1] "age"       "vineyard"  "complex"   "vintage"   "firm"
 [6] "dense"     "chocolate" "power"     "delicious" "powerful"
[11] "time"      "richness"  "fine"      "currant"   "mineral"
[16] "intense"   "layer"
```

Let us now create some more descriptors that relate to each descriptions, and add those to our **ratings** dataset. This is called “feature engineering”.

We add: lenght of descrition in words, the lexical diversity (distinct words), lexical density (diversity as % of total words), repetition (Length/diversity), number of large words with 8 or more characters, number of adjectives, number of nouns, number of verbs and, most importantly, number of words that are in the list of words typically associated with very good bottles as defined above.

Hide

```
descriptors <- words_in_descriptions %>% group_by(X) %>%
  summarise(Length = n(),
            lexical_diversity = n_distinct(word),
            lexical_density = lexical_diversity/Length,
            repetition = Length/lexical_diversity,
            large_word_count = sum(ifelse((nchar(word) > 7), 1, 0)),
            adjectives_count = sum(ifelse(upos == "ADJ",1,0)),
            noun_count = sum(ifelse(upos == "NOUN",1,0)),
            verb_count = sum(ifelse(upos == "VERB",1,0)),
            good_words = sum(ifelse(lemma %in% good_predictor_words,1,0)))
```

We now add these descriptors to the **ratings** dataframe.

Hide

```
#Adding the descriptors
ratings <- ratings %>% left_join(descriptors,by = "X")
#Cleaning the environment
rm(descriptors, words_in_descriptions)
#Selecting only the variables which we intend to use in our analysis and creati
ng a ratings set for machine learning purposes
ratings_ML <- ratings %>%
  select(X,
         country,
         price,
         province,
         taster_name,
         variety,
         winery,
         Vintage,
         Lenght=Length.x,
         lexical_diversity,
         lexical_density,
         repetition,
         large_word_count,
         adjectives_count,
         noun_count,
         verb_count,
         good_words,
         points)
#Removing bottles for which price is not available
NAs <- which(is.na(ratings_ML$price))
ratings_ML <- ratings_ML[-NAs,]
```

Modelling

We are now ready to use our data for Machine Learning purposes. We will begin by splitting our **ratings_ML** dataset into a **train_set** and a **test_set**.

Hide

```
# Test set will be 10% of Ratings_ML data
set.seed(1)
test_index <- createDataPartition(y = ratings_ML$points, times = 1, p = 0.10, l
ist = FALSE)
train_set <- ratings_ML[-test_index,]
temp_set <- ratings_ML[test_index,]
# Make sure country, province, taster_name, variety, winery and vintage are als
o in the training set
test_set <- temp_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 = "Vintage")
# Add rows removed from test set set back into train set
removed_rows <- anti_join(temp_set, test_set)
train_set <- rbind(train_set, removed_rows)
rm(removed_rows, temp_set, test_index)
```

Great, now, let's give ourselves a benchmark. What if we predicted the average rating for each wine?

Hide

```
average_rating <- mean(train_set$points)
naiveRMSE <- RMSE(test_set$points,average_rating)
rmse_results <- bind_rows(data_frame(Method = "Just the average", RMSE = naiveR
MSE))
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	3.045522

Here, we get a RMSE of about 3.0455. Not great. We are on average 3 full points away from the correct rating.

To improve on this, we will construct a model with the following structure. We do not want to train our ML algorithms just yet because the computation would take too long.

$$Y_{c,t,v,w,va,p} = \mu + \hat{b}_c + \hat{b}_t + \hat{b}_v + \hat{b}_w + \hat{b}_{va} + \hat{b}_p + \varepsilon_{c,t,v,w,va,p}$$

In this model, μ is the average rating in the training set.

The various $\hat{b}$ are respectively the country, taster, vintage, winery, variety and province effect. We compute these sequentially such that each effect takes into consideration all the ones before. This is to speed up the process (as opposed to running a very large regression).

Finally, $\varepsilon_{c,t,v,w,va,p}$ is the residual when using only this linear model.

We will be referring to this model as **our linear model*.

Once that model is optimized, we will try to minimize $\varepsilon_{c,t,v,w,va,p}$ by using machine learning algorithms such that:

$$Y_{c,t,v,w,va,p} = \hat{Y}_{c,t,v,w,va,p} + \hat{Z}_{c,t,v,w,va,p} + \varepsilon_{z,c,t,v,w,va,p}$$

where $\varepsilon_{z,c,t,v,w,va,p}$ is what we are striving to minimize and $Z_{c,t,v,w,va,p}$ is the contribution of our machine learning algorithms and $\hat{Y}_{c,t,v,w,va,p}$ is the contribution from our linear model described before..

Country Effect

Some countries, one average, rate higher than others. We would call this the **country effect**. For each wine, the country effect is calculated as the average of $Y_{c,t,v,w,va,p} - \hat{\mu}$ for each country c.

We calculate it using this piece of code:

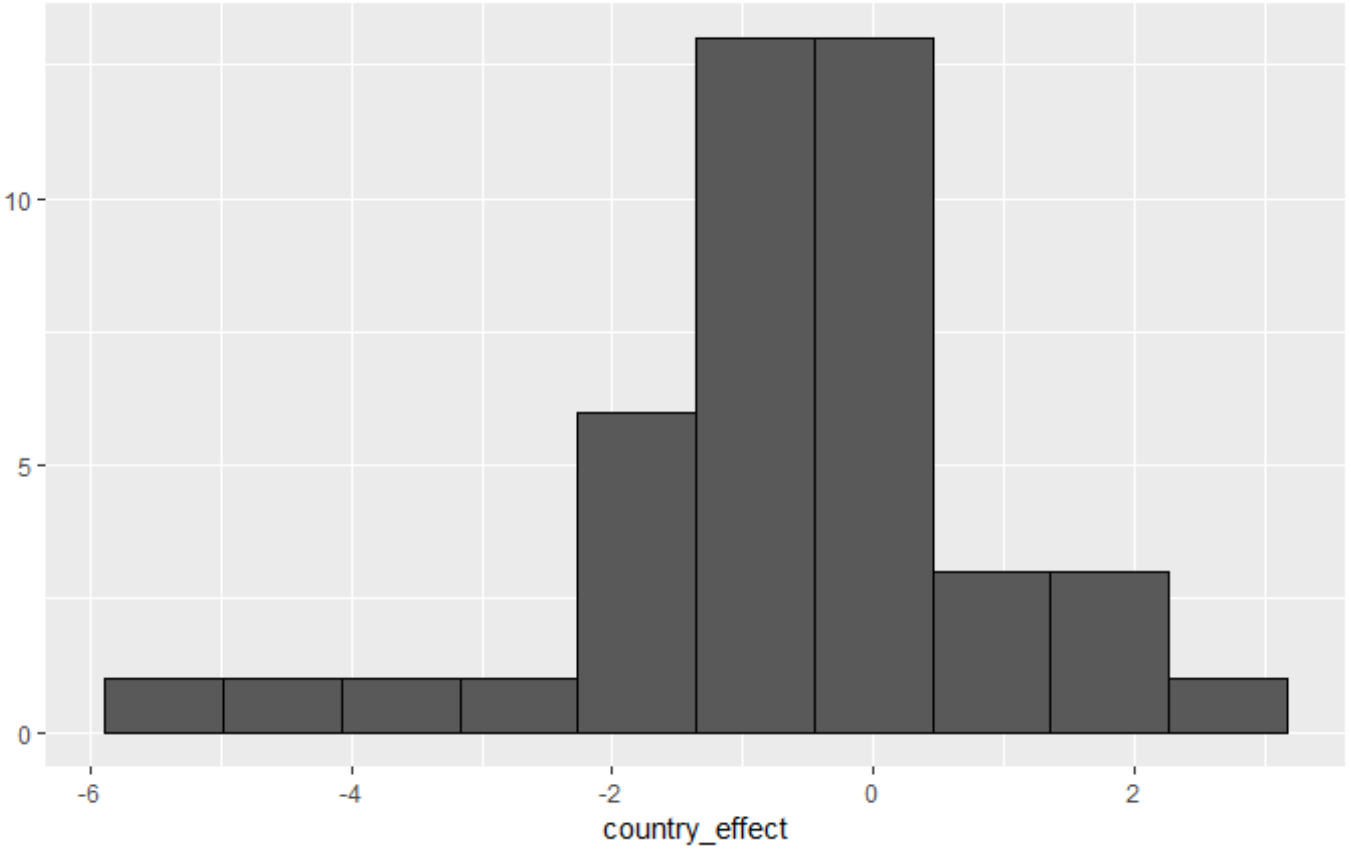
Hide

```
country_avgs <- train_set %>%
  group_by(country) %>%
  summarize(country_effect = mean(points - average_rating))
```

We can see that these estimates vary substantially from -6 to +3. Given our average rating is about 88, a country effect of 3 brings the score close to 100.

Hide

```
country_avgs %>% qplot(country_effect, geom ="histogram", bins = 10, data = .,
 color = I("black"))
```



We can now try to estimate each observation of the test set using this country effect. To be clear, this is the model we are trying, where μ is the average rating, b_c is the movie_effect and $\varepsilon_{c,t,v,w,va,p}$ is the error term:

$$Y_{c,t,v,w,va,p} = \mu + \hat{b}_c + \varepsilon_{c,t,v,w,va,p}$$

Hide

```
predicted_ratings <- average_rating + test_set %>%
  left_join(country_avgs, by='country') %>%
  .$country_effect
country_effect_RMSE <- RMSE(predicted_ratings, test_set$points)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "With Country Effect",
                                     RMSE=country_effect_RMSE))
kable(rmse_results,align=rep("c",3), caption = "Metrics calculated on test set
 only") %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Metrics calculated on test set only

Method	RMSE
Just the average	3.045522
With Country Effect	2.973631

This is already an improvement. We could dive into regularization of this effect, but we decide not to here for the sake of expediency. Note that we have tried it on the side and it doesn't help much.

Taster Effect

Now, we turn our attention to tasters. Some of them, as we saw before, tend to rate higher than others. Here we can calculate the **taster effect** which is a taster-specific effect once the **country effect** has been taken into consideration.

To be clear, this is the model we are trying, where μ is the average rating, $\hat{b}_c$ is the country effect, $\hat{b}_t$ is the taster effect and $\varepsilon_{c,t,v,w,va,p}$ is the error term:

$$Y_{c,t,v,w,va,p} = \mu + \hat{b}_c + \hat{b}_t + \varepsilon_{c,t,v,w,va,p}$$

Hide

```
taster_avgs <- train_set %>%
  left_join(country_avgs, by='country') %>%
  group_by(taster_name) %>%
  summarize(taster_effect = mean(points - average_rating - country_effect))
```

Let's create predictions see how it improves our model.

Hide

```
predicted_ratings <- test_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by='taster_name') %>%
  mutate(pred = average_rating + country_effect + taster_effect) %>%
  .$pred
country_and_taster_effect <- RMSE(predicted_ratings, test_set$points)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "Country & Taster Effects",
                                     RMSE=country_and_taster_effect))
kable(rmse_results,align=rep("c",3), caption = "Metrics calculated on test set
 only") %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Metrics calculated on test set only

Method	RMSE
Just the average	3.045522
With Country Effect	2.973631
Country & Taster Effects	2.906345

Vintage Effect

We now turn our attention to the vintage variable and make a similar analysis.

Hide

```
vintage_avgs <- train_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by = "taster_name") %>%
  group_by(Vintage) %>%
  summarize(vintage_effect = mean(points - average_rating - country_effect - ta
ster_effect))
predicted_ratings <- test_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by='taster_name') %>%
  left_join(vintage_avgs, by = 'Vintage') %>%
  mutate(pred = average_rating + country_effect + taster_effect+ vintage_effec
t) %>%
  .$pred
country_taster_vintage_effect_RMSE <- RMSE(predicted_ratings, test_set$points)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "Country & Taster & Vintage Effect
s",
                                     RMSE=country_taster_vintage_effect_RMSE))
kable(rmse_results,align=rep("c",3), caption = "Metrics calculated on test set
 only") %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Metrics calculated on test set only

Method	RMSE
Just the average	3.045522
With Country Effect	2.973631
Country & Taster Effects	2.906345
Country & Taster & Vintage Effects	2.893075

At this point, we have completed our model, and only improved our RMSE by about 5%, which is not great. We are hopeful that the other variables will be helpful.

Winery Effect

We now turn our attention to the winery variable and make a similar analysis.

Hide

```
winery_avgs <- train_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by = "taster_name") %>%
  left_join(vintage_avgs, by = "Vintage") %>%
  group_by(winery) %>%
  summarize(winery_effect = mean(points - average_rating - country_effect - tas
ter_effect - vintage_effect))
predicted_ratings <- test_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by='taster_name') %>%
  left_join(vintage_avgs, by = 'Vintage') %>%
  left_join(winery_avgs,by='winery') %>%
  mutate(pred = average_rating + country_effect + taster_effect+ vintage_effect
+winery_effect) %>%
  .$pred
country_taster_vintage_winery_effect_RMSE <- RMSE(predicted_ratings, test_set$p
oints)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "Country & Taster & Vintage & Wine
ry Effects",
                                     RMSE=country_taster_vintage_winery_effect_
RMSE))
kable(rmse_results,align=rep("c",3), caption = "Metrics calculated on test set
 only") %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Metrics calculated on test set only

Method	RMSE
Just the average	3.045522
With Country Effect	2.973631
Country & Taster Effects	2.906345
Country & Taster & Vintage Effects	2.893075
Country & Taster & Vintage & Winery Effects	2.385231

Variety Effect

We now turn our attention to the variety variable and make a similar analysis.

Hide

```
variety_avgs <- train_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by = "taster_name") %>%
  left_join(vintage_avgs, by = "Vintage") %>%
  left_join(winery_avgs, by = 'winery') %>%
  group_by(variety) %>%
  summarize(variety_effect = mean(points - average_rating - country_effect - ta
ster_effect - vintage_effect - winery_effect))
predicted_ratings <- test_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by='taster_name') %>%
  left_join(vintage_avgs, by = 'Vintage') %>%
  left_join(winery_avgs,by='winery') %>%
  left_join(variety_avgs,by='variety') %>%
  mutate(pred = average_rating + country_effect + taster_effect+ vintage_effect
+winery_effect+variety_effect) %>%
  .$pred
country_taster_vintage_winery_variety_effect_RMSE <- RMSE(predicted_ratings, te
st_set$points)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "Country & Taster & Vintage & Wine
ry & Variety Effects",
                                     RMSE=country_taster_vintage_winery_variety
_effect_RMSE))
kable(rmse_results,align=rep("c",3), caption = "Metrics calculated on test set
 only") %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Metrics calculated on test set only

Method	RMSE
Just the average	3.045522
With Country Effect	2.973631
Country & Taster Effects	2.906345
Country & Taster & Vintage Effects	2.893075

Method	RMSE
Country & Taster & Vintage & Winery Effects	2.385231
Country & Taster & Vintage & Winery & Variety Effects	2.351174

Province Effect

We now turn our attention to the province variable and make a similar analysis.

Hide

```
province_avgs <- train_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by = "taster_name") %>%
  left_join(vintage_avgs, by = "Vintage") %>%
  left_join(winery_avgs, by = 'winery') %>%
  left_join(variety_avgs,by='variety') %>%
  group_by(province) %>%
  summarize(province_effect = mean(points - average_rating - country_effect - t
aster_effect - vintage_effect - winery_effect - variety_effect))
predicted_ratings <- test_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by='taster_name') %>%
  left_join(vintage_avgs, by = 'Vintage') %>%
  left_join(winery_avgs,by='winery') %>%
  left_join(variety_avgs,by='variety') %>%
  left_join(province_avgs,by="province") %>%
  mutate(pred = average_rating + country_effect + taster_effect+ vintage_effect
+winery_effect+variety_effect+province_effect) %>%
  .$pred
country_taster_vintage_winery_variety_province_effect_RMSE <- RMSE(predicted_ra
tings, test_set$points)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "Country & Taster & Vintage & Wine
ry & Variety & Province Effects",
                                     RMSE=country_taster_vintage_winery_variety
_province_effect_RMSE))
kable(rmse_results,align=rep("c",3), caption = "Metrics calculated on test set
 only") %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Metrics calculated on test set only

Method	RMSE
Just the average	3.045522
With Country Effect	2.973631
Country & Taster Effects	2.906345
Country & Taster & Vintage Effects	2.893075
Country & Taster & Vintage & Winery Effects	2.385231
Country & Taster & Vintage & Winery & Variety Effects	2.351174
Country & Taster & Vintage & Winery & Variety & Province Effects	2.348546

Machine Learning Algorithm

Below, we train linear regression to minimize the errors when predicting **points** using the predicted values as well as all remaining variables: the price of the bottle as well as the descriptors of the description obtained from our Natural Language Processing section.

Hide

```
# Create predictions for the training set
predicted_ratings_train <- train_set %>%
  left_join(country_avgs, by='country') %>%
  left_join(taster_avgs, by='taster_name') %>%
  left_join(vintage_avgs, by = 'Vintage') %>%
  mutate(pred = average_rating + country_effect + taster_effect+ vintage_effec
t) %>%
  .$pred
#Add predicted values to both sets
train_set <- train_set %>% mutate(predicted_ratings = predicted_ratings_train,
                                  epsilon = points - predicted_ratings_train)
test_set <- test_set %>% mutate(predicted_ratings=predicted_ratings,
                                epsilon = points - predicted_ratings)
#Train the model
glm_trained <- train(points ~ predicted_ratings+ price+Lenght+lexical_diversity
+lexical_density+repetition+large_word_count+adjectives_count+noun_count+verb_c
ount+good_words, method="glm", data = train_set )
```

Let’s now test whether this improves our results.

Hide

```
glm_predicted_ratings <- predict(glm_trained,test_set)
glm_RMSE <- RMSE(glm_predicted_ratings, test_set$points)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method= "Linear Regression on Epsilon",
                                     RMSE=glm_RMSE))
kable(rmse_results,align=rep("c",3), caption = "Metrics calculated on test set
 only") %>%
  kable_styling(full_width = F) %>%
  column_spec(1,bold=T,border_right = T)
```

Metrics calculated on test set only

Method	RMSE
Just the average	3.045522
With Country Effect	2.973631
Country & Taster Effects	2.906345
Country & Taster & Vintage Effects	2.893075
Country & Taster & Vintage & Winery Effects	2.385231
Country & Taster & Vintage & Winery & Variety Effects	2.351174
Country & Taster & Vintage & Winery & Variety & Province Effects	2.348546
Linear Regression on Epsilon	2.062728

Results

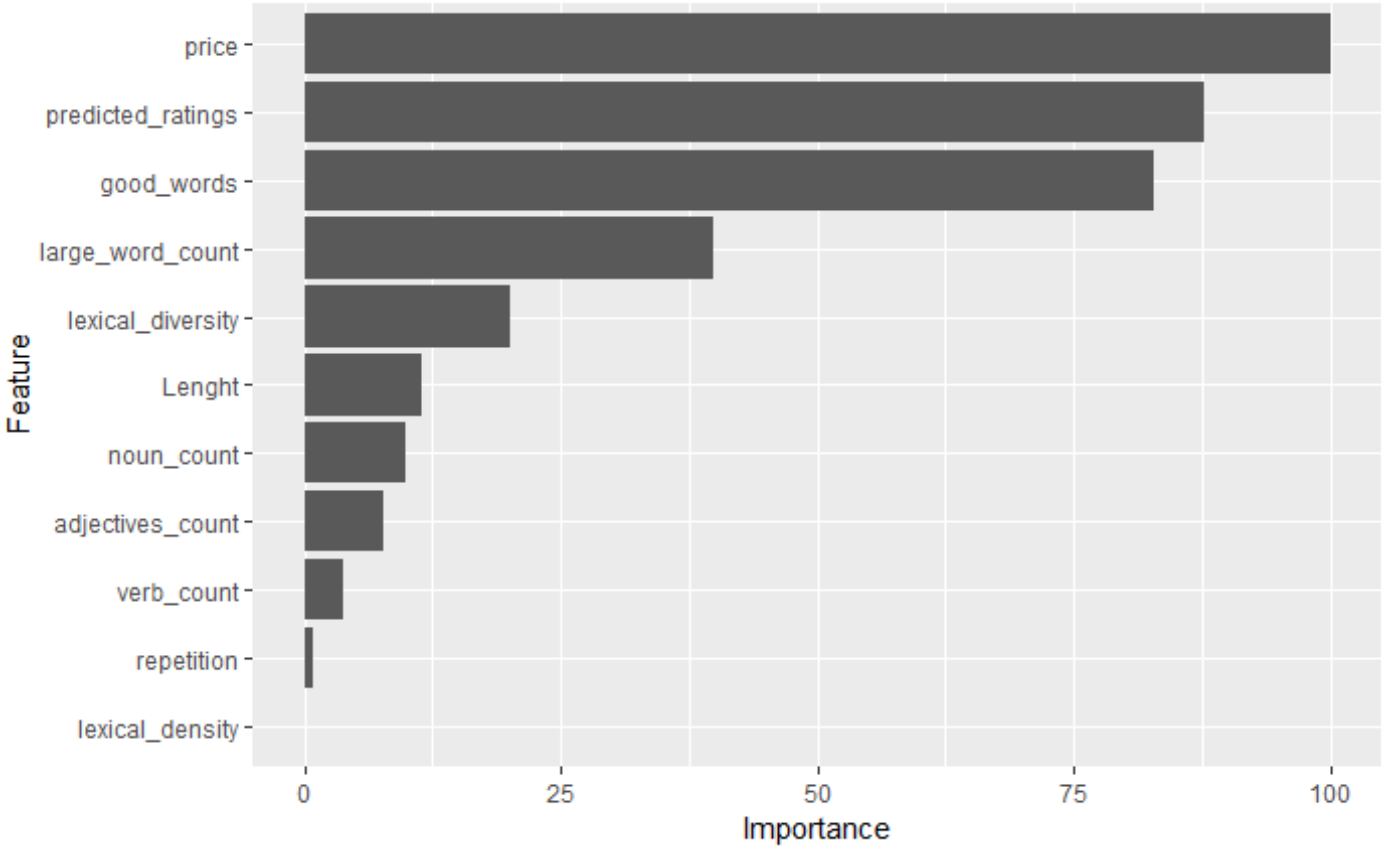
This method further brought down our RMSE to 2.0627281. That is a total improvement of 32.27% over our naive method of forecasting the average.

What's more, we've also uncovered that the inclusion of price and the NLP variables improved the RMSE by 12.17% from the linear model.

The importance of each variable can be illustrated as follows:

Hide

```
varImp(glm_trained) %>% ggplot()
```



Unsurprisingly, price takes the first position, and the second position goes to our linear model predictions. Then, as we would have expected, the presence of words that are commonly associated with high ratings comes in third. The importance drops significantly after that.

These results are encouraging. They mean that with a limited analysis of the descriptions, one can predict the rating that a wine would get within ~2 points, which is one full point better than just predicting the average of 88.42.

Conclusion

We started this paper by stating the motivation for this project: we love wine, and we rely on points systems to help us guide our buying.

By looking at the data, we've noticed that price was certainly a good indicater of how well a bottle would be rated. However, we had no way of establishing causality.

We then dove into a linear model that took into consideration Country, Taster, Vintage, Winery, Variety and Province variables to predict the scores. We managed to improve our RMSE by 22.89%.

Then, we trained a linear regression algorithm on the data and managed to further improve our RMSE by 12.17%, for a total of 32.27%.

The most important variable in the regression was price, followed by our predicted values from our linear model, and finally by the presence of key words that are often associated with good wine.

Here they are again, for your convenience: **age, vineyard, complex, vintage, firm, dense, chocolate, power, delicious, powerful, time, richness, fine, currant, mineral, intense, layer**.

To further improve the accuracy of this model, we would suggest two main paths:

1. Experimenting with different features in the NLP of the descriptions
2. Experimenting with different machine learning algorithms to determine whether some may perform better than linear regression. We have decided against it here for the sake of computing time.

Note: This report took 4.78 minutes to populate.