Wines Around the World

YZ Analytics

December 11, 2019

Table of Contents

Introduction

The first traces of wine were found in Georgia in 6000BCE[1]. In **Figure 1** we show a map of more than 10 thousand wineries operational today. For more than 8000 years,

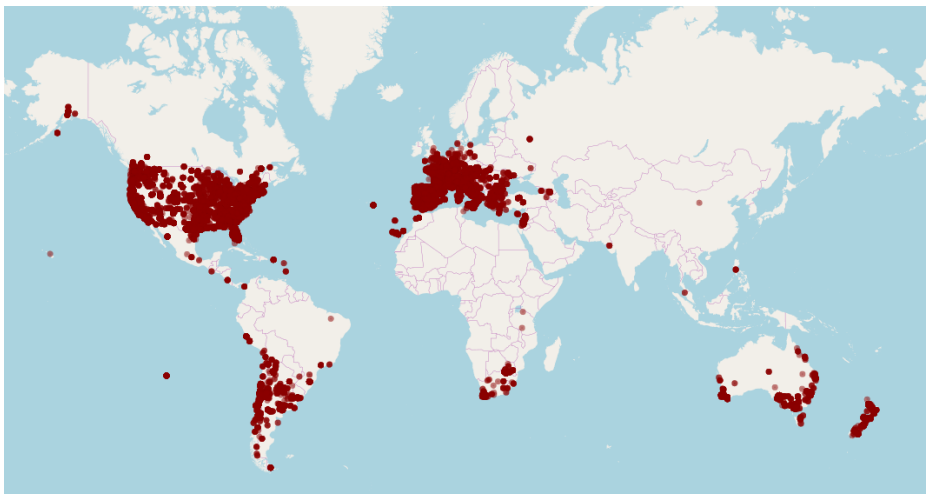


Figure 1: Map of more than 10000 wineries present in our dataset.

wine has been a relevant part of civilization, and yet, many people still do not think twice when they say "I'm not a wine person". Even then, there are many people that are also able to comfortably say: "This juicy, fruit-forward wine delectates the palate."

This report and the accompanying web application aim for two things: 1) to showcase an advanced usage of visualization techniques in the context of a product for wine exploration, and 2) to expand and develop statistical techniques for a seamless incorporation into our product. Therefore, our main goal is to provide an immer-

[1]Watson (2010)

sive experience which leverages technology and builds upon the statistical/technical knowledge of a bachelor-level student, all within the context of wine.

0.1 Data Provenance

In this project we will be working with Kaggle's wine review data, found here[2]. There are close to 130,000 wine reviews from Wine Magazine's website[3]. A small glimpse of what the data looks like is available below for convenience:

```
Observations: 129,971
Variables: 14
$ X1                    <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12...
$ country               <chr> "Italy", "Portugal", "US", "US", "US", "...
$ description           <chr> "Aromas include tropical fruit, broom, b...
$ designation           <chr> "Vulkà Bianco", "Avidagos", NA, "Reserve...
$ points                <dbl> 87, 87, 87, 87, 87, 87, 87, 87, 87, 87, ...
$ price                 <dbl> NA, 15, 14, 13, 65, 15, 16, 24, 12, 27, ...
$ province              <chr> "Sicily & Sardinia", "Douro", "Oregon", ...
$ region_1              <chr> "Etna", NA, "Willamette Valley", "Lake M...
$ region_2              <chr> NA, NA, "Willamette Valley", NA, "Willam...
$ taster_name           <chr> "Kerin O'Keefe", "Roger Voss", "Paul Gre...
$ taster_twitter_handle <chr> "@kerinokeefe", "@vossroger", "@paulgwin...
$ title                 <chr> "Nicosia 2013 Vulkà Bianco  (Etna)", "Qu...
$ variety               <chr> "White Blend", "Portuguese Red", "Pinot ...
$ winery                <chr> "Nicosia", "Quinta dos Avidagos", "Rains...
```

Next we proceed with a discussion of the variables that are available in our dataset.

[2]Thoutt (2017)
[3]("Wine enthusiast reviews," 2019)

0.1.1 Variables:

Our dataset contains almost 130K observations, one for each wine review, and 17 variables.

Below, a description of the seventeen variables is included:

- id - The unique obervation identifier for reviews in our dataset
- country - The country that the wine is from
- description - A few sentences from a sommelier describing the wine's taste, smell, look, feel, etc
- 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 (i.e. Rutherford inside the Napa Valley), but this value can sometimes be blank
- variety - The type of grapes used to make the wine (ie Pinot Noir)
- winery - The winery that made the wine
- title - The title of the wine review, which often contains the vintage if you're interested in extracting that feature
- taster_name - name of the taster
- taster_twitter_handle - Twitter handle of the taster

—— below to be created —

- address - A combination of the winery and the country
- latitude - to be geocoded latitude
- longitude - to be geocoded longitude

0.1.2 An Assessment of Data Quality

Frequently undergraduate analyses are done with carefully curated data or fail to consider the quality of the datasets used until the conclusion. We decided to assess the quality of our data through research of wine production as this was decisive in creating our product.

In **Figure 2** we show a map of the most relevant wine producing regions in the world. It is clearly evident that our dataset does not represent the global production of wine because the presence of observations originating from either Russia and China is lacking. Apart from the representativeness of our dataset, we consider the amount

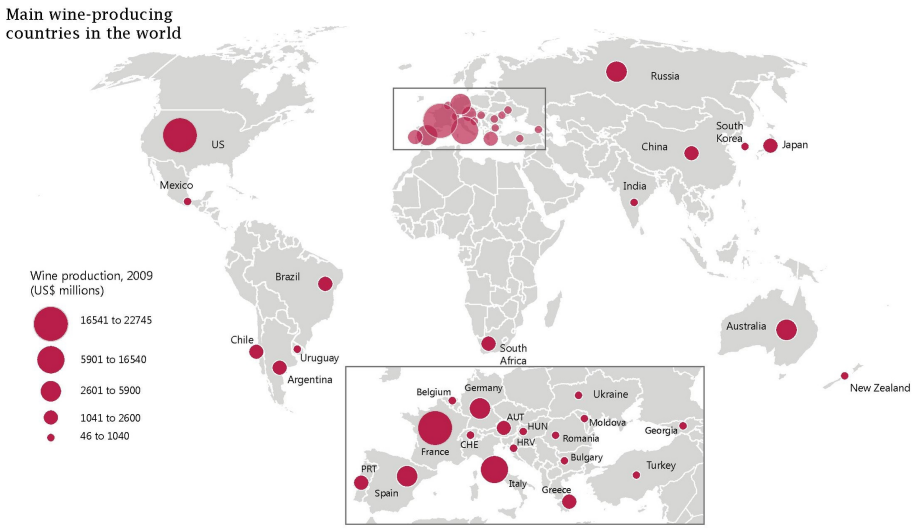


Figure 2: Map from Wikipedia with most relevant wine regions. Note the difference between the figure generated from our dataset (Figure 1) and this map.

of information present, or the usefulness of our data. Most of the information we have

is categorical, so we will have to generate some quantitative information to open up the number of possible analyzes or visualizations that are possible with our data.

Chapter 1 Exploratory Data Analysis

```
library(tidyverse)
library(GGally)
```

1.1 Countries

It will be important to understand the countries that are represented in our dataset in order to be able to know what types of mapping capabilities we have to have to create a good experience.

```
path <- "../data/winemag-data-130k-v2.csv"
Wine <- read_csv(path,
                 col_types = cols(
                     X1 = col_double(),
                     country = col_character(),
                     description = col_character(),
                     designation = col_character(),
                     points = col_double(),
                     price = col_double(),
                     province = col_character(),
                     region_1 = col_character(),
```

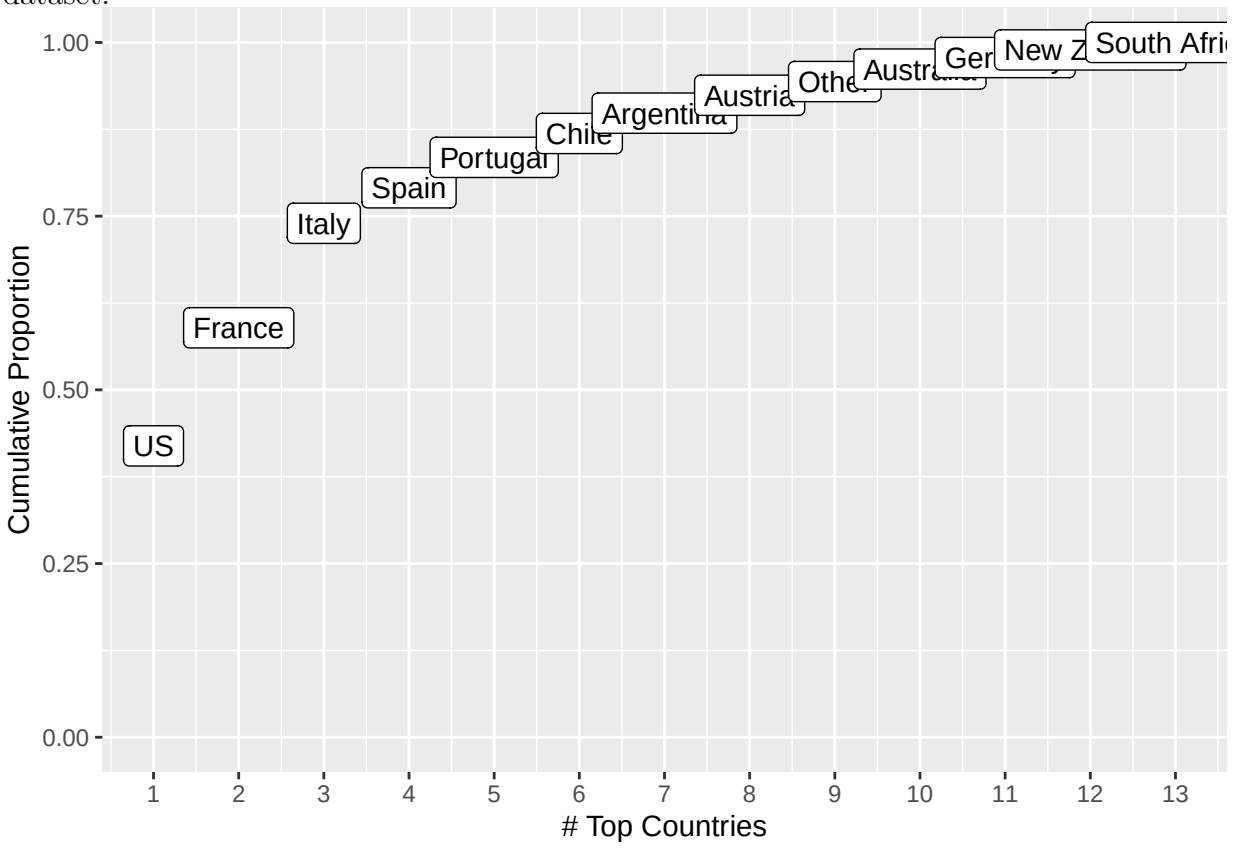
```
                    region_2 = col_character(),
                    taster_name = col_character(),
                    taster_twitter_handle = col_character(),
                    title = col_character(),
                    variety = col_character(),
                    winery = col_character()),
                progress = FALSE
                ) %>%
    rename(id = X1)
```

```
top_countries_tbl <- Wine %>%
    mutate(country = fct_explicit_na(country)) %>%
    mutate(country = fct_lump(country, 12)) %>%
    count(country, sort = TRUE) %>%
    mutate(prop = n / sum(n))


top_countries_tbl
```

```
   # A tibble: 13 x 3
      country          n   prop
      <fct>        <int>  <dbl>
    1 US           54504 0.419
    2 France       22093 0.170
    3 Italy        19540 0.150
    4 Spain         6645 0.0511
    5 Portugal      5691 0.0438
    6 Chile         4472 0.0344
```

```
 7 Argentina     3800 0.0292
 8 Austria       3345 0.0257
 9 Other         2567 0.0198
10 Australia     2329 0.0179
11 Germany       2165 0.0167
12 New Zealand   1419 0.0109
13 South Africa  1401 0.0108
```

The top 13 categories, including the lumped-together category of "Other" consist of those categories which have a count consisting of more than 1% of the observations in the dataset.



Note that most of the observations, in fact, more than 90% of the observations are contained in the 8 most represented countries and 80% on the top 4, and 60% on the

top 2 (USA and France).

It looks like it will be possible to create an interactive map. Now we need to geolocate the wineries. Worst case scenario we have the countries and their representation in the dataset.

Another interesting fact is that since 40% of the observations come from the USA, then perhaps it will be possible to get historical information to add quantitative predictors to our dataset, but it is not crucial since our focus is in the presentation of the data.

1.2 Wineries

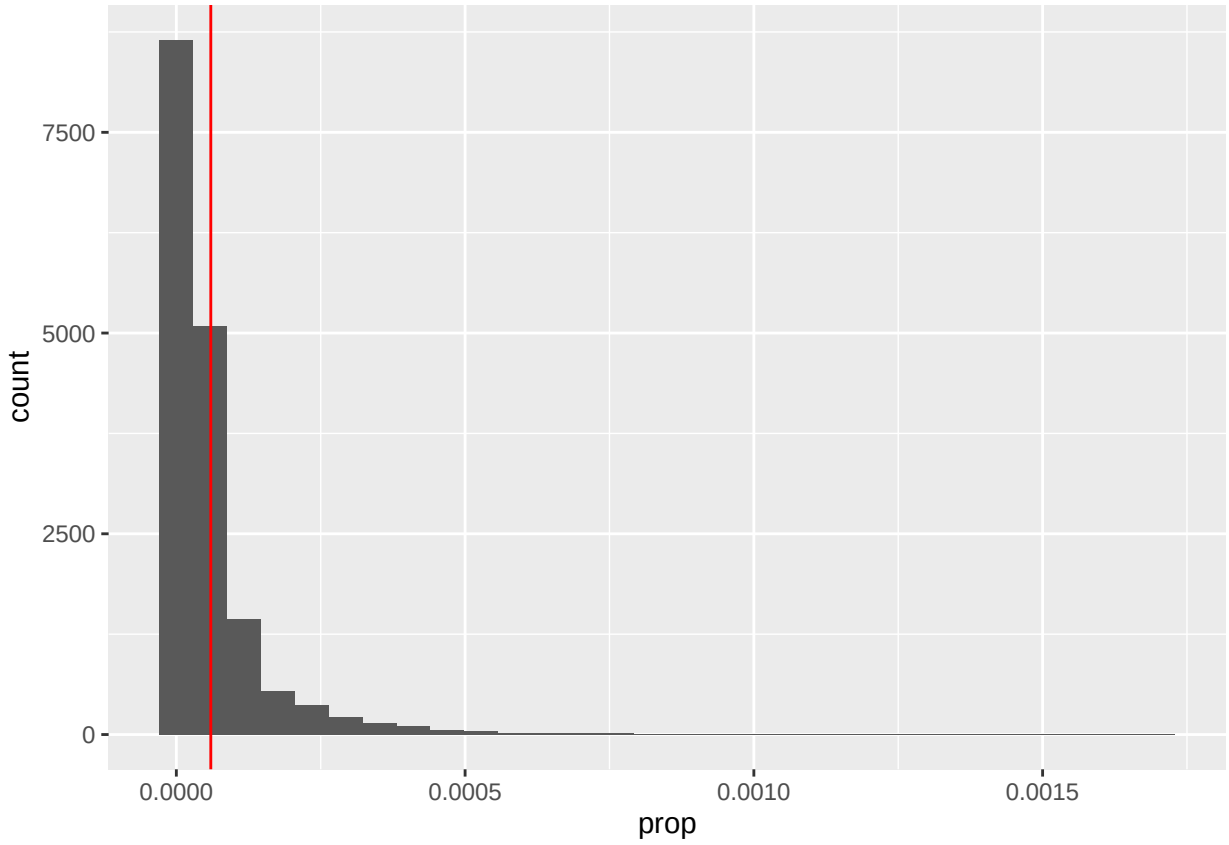
Is there a similar concentration for the wineries? Turns out no. Lumping won't work because there are just so many wineries and there aren't any ones that particularly dominate.

```
top_wineries_tbl <- Wine %>%
    count(winery, sort = TRUE) %>%
    mutate(prop = n / sum(n)) %>%
    mutate(prop_cumulative = cumsum(prop))


mean_prop <- mean(top_wineries_tbl$prop)
mean_prop
```

```
[1] 5.967655e-05
```

```
top_wineries_tbl %>%
    ggplot(aes(x = prop)) +
```

```
    geom_histogram(bins = 30) +
    geom_vline(xintercept = mean_prop, color = "red")
```



The table below summarizes the information for the proportions of each winery.

```
summary(top_wineries_tbl)
```

```
   winery                n                 prop
Length:16757       Min.   :  1.000   Min.   :7.694e-06
Class :character   1st Qu.:  1.000   1st Qu.:7.694e-06
Mode  :character   Median :  3.000   Median :2.308e-05
                   Mean   :  7.756   Mean   :5.968e-05
                   3rd Qu.:  8.000   3rd Qu.:6.155e-05
                   Max.   :222.000   Max.   :1.708e-03
```

```
 prop_cumulative
 Min.   :0.001708
 1st Qu.:0.721630
 Median :0.892214
 Mean   :0.807783
 3rd Qu.:0.967770
 Max.   :1.000000
```

From the summary we note that half of the wineries contain more than 90% of the observations, and 25% of the wineries contain more than 70% of the observations. However there are 16757 wineries which means that 25% of the observations is around 4000 wineries.

If we need to geolocate the wineries and we run into trouble, then perhaps only doing half will suffice for our visualization.

1.3 Variety

Variety of a wine refers to the type of grape that is used - for example, among white grape wines, there are varieties such as Sauvignon Blanc, Chardonnay, and Riesling. Among red grape wines, some varieties include Merlot, Cabernet Sauvignon, and Pinot Noir.

```
top_varieties_tbl <- Wine %>%
    count(variety, sort = TRUE) %>%
    mutate(prop = n / sum(n)) %>%
    mutate(prop_cumulative = cumsum(prop))


summary(top_varieties_tbl)
```

```
  variety                n                  prop
 Length:708         Min.   :    1.00   Min.   :7.690e-06
 Class :character   1st Qu.:    2.00   1st Qu.:1.539e-05
 Mode  :character   Median :    6.00   Median :4.616e-05
                    Mean   :  183.57   Mean   :1.412e-03
                    3rd Qu.:   28.25   3rd Qu.:2.174e-04
                    Max.   :13272.00   Max.   :1.021e-01
 prop_cumulative
 Min.   :0.1021
 1st Qu.:0.9749
 Median :0.9937
 Mean   :0.9654
 3rd Qu.:0.9984
 Max.   :1.0000
```

```
top_varieties_tbl %>%
  arrange(desc(prop)) %>%
  head()
```

```
 # A tibble: 6 x 4
   variety                    n   prop prop_cumulative
   <chr>                  <int>  <dbl>           <dbl>
 1 Pinot Noir             13272 0.102           0.102
 2 Chardonnay             11753 0.0904          0.193
 3 Cabernet Sauvignon      9472 0.0729          0.265
```

```
4 Red Blend                  8946 0.0688          0.334
5 Bordeaux-style Red Blend   6915 0.0532          0.387
6 Riesling                   5189 0.0399          0.427
```

We can see from the summary that with among wine variety, 25% of the varieties include more than 97% of of the wines and half of the varieties account for more than 99% of the observed wines. Additionally, Pinot Noir accounts for more than 10% of the wines, followed by Chardonnay with 9.0%, Cabernet Sauvignon with 7.3%, and Red Blend with 6.9%.

1.4 Designation

Designation is a tricky variable to work with. It refers to a label placed on the wine by the winemaker in regulation with rules of the country, although not every country has the same rules. For example, the designation of "Reserve" wine generally means the wine has been set aside to age for a longer time than other wines generally would, and it often implies a higher quality. While "Reserva" refers to reserve wines in Spain, and "Riserva" to those in Italy, the two countries have different rules about how long the wine must be aged for in order to receive their respective designations. Other countries, like the U.S., don't have any rules in general. Given this general lack of universality of designation, this variable likely will not mean much in our project, but we can still look at its characteristics.

```
top_designation_tbl <- Wine %>%
    count(designation, sort = TRUE) %>%
    mutate(prop = n / sum(n)) %>%
    mutate(prop_cumulative = cumsum(prop))
```

```
summary(top_designation_tbl)
```

```
    designation                n                   prop
    Length:37980       Min.    :    1.00   Min.   :7.690e-06
    Class :character   1st Qu.:    1.00   1st Qu.:7.690e-06
    Mode  :character   Median :    1.00   Median :7.690e-06
                       Mean   :    3.42   Mean   :2.633e-05
                       3rd Qu.:    2.00   3rd Qu.:1.539e-05
                       Max.   :37465.00   Max.   :2.883e-01
    prop_cumulative
    Min.   :0.2883
    1st Qu.:0.7374
    Median :0.8539
    Mean   :0.8205
    3rd Qu.:0.9269
    Max.   :1.0000
```

```
top_designation_tbl %>%
  arrange(desc(prop)) %>%
  head()
```

```
   # A tibble: 6 x 4
     designation      n    prop prop_cumulative
     <chr>        <int>   <dbl>           <dbl>
   1 <NA>         37465 0.288             0.288
   2 Reserve       2009 0.0155            0.304
```

```
3 Estate         1322 0.0102            0.314
4 Reserva        1259 0.00969           0.324
5 Riserva         698 0.00537           0.329
6 Estate Grown    621 0.00478           0.334
```

While 28.8% of the wines do not have a designation, 25% of the designations contain more than 73% of the wines. We see that of the most common 5 designations, three of them are related to reserve wines but in different languages, while the other two refer to estate wines - wines in which the grapes are grown and the wine is made in the same location.

1.5 Taster

The tasters are Wine Enthusiast Magazine wine reviewers.

```
top_taster_tbl <- Wine %>%
    mutate(taster_name = fct_explicit_na(taster_name)) %>%
    mutate(taster_name = fct_lump(taster_name, 15)) %>%
    count(taster_name, sort = TRUE) %>%
    mutate(prop = n / sum(n))


top_taster_tbl
```

```
# A tibble: 16 x 3
   taster_name            n    prop
   <fct>              <int>   <dbl>
 1 (Missing)          26244 0.202
 2 Roger Voss         25514 0.196
```

```
 3 Michael Schachner  15134 0.116
 4 Kerin O'Keefe      10776 0.0829
 5 Virginie Boone      9537 0.0734
 6 Paul Gregutt        9532 0.0733
 7 Matt Kettmann       6332 0.0487
 8 Joe Czerwinski      5147 0.0396
 9 Sean P. Sullivan    4966 0.0382
10 Anna Lee C. Iijima  4415 0.0340
11 Jim Gordon          4177 0.0321
12 Anne Krebiehl MW    3685 0.0284
13 Lauren Buzzeo       1835 0.0141
14 Susan Kostrzewa     1085 0.00835
15 Other               1078 0.00829
16 Mike DeSimone        514 0.00395
```

While 20% of the wines do not have tasters listed, 19.6% of the wines were tasted by Roger Voss, followed by 11.6% which were tasted by Michael Schachner. A potentially interesting side project could be to try and differentiate the wine descriptions between tasters, or to search for patterns in each taster's preferred wines.

We can speculate if any of the tasters are biased for more positive or negative reviews by looking at mean points per taster:

```
Wine %>%
  group_by(taster_name) %>%
  summarize(meanpoints = mean(points)) %>%
  arrange(desc(meanpoints))
```

```
 # A tibble: 20 x 2
```

	taster_name	meanpoints
	<chr>	<dbl>
1	Anne Krebiehl MW	90.6
2	Matt Kettmann	90.0
3	Virginie Boone	89.2
4	Mike DeSimone	89.1
5	Paul Gregutt	89.1
6	Kerin O'Keefe	88.9
7	Sean P. Sullivan	88.8
8	Roger Voss	88.7
9	Jim Gordon	88.6
10	Joe Czerwinski	88.5
11	Anna Lee C. Iijima	88.4
12	Jeff Jenssen	88.3
13	Christina Pickard	87.8
14	<NA>	87.8
15	Lauren Buzzeo	87.7
16	Michael Schachner	86.9
17	Fiona Adams	86.9
18	Susan Kostrzewa	86.6
19	Carrie Dykes	86.4
20	Alexander Peartree	85.9

The mean points per taster range between 85.9 and 90.6. Although there are likely many factors underlying these differences in points between reviewers, if I were a wine maker, I would want Anne Krebiehl MW or Matt Kettmann reviewing my wine, not Alexander Peartree.

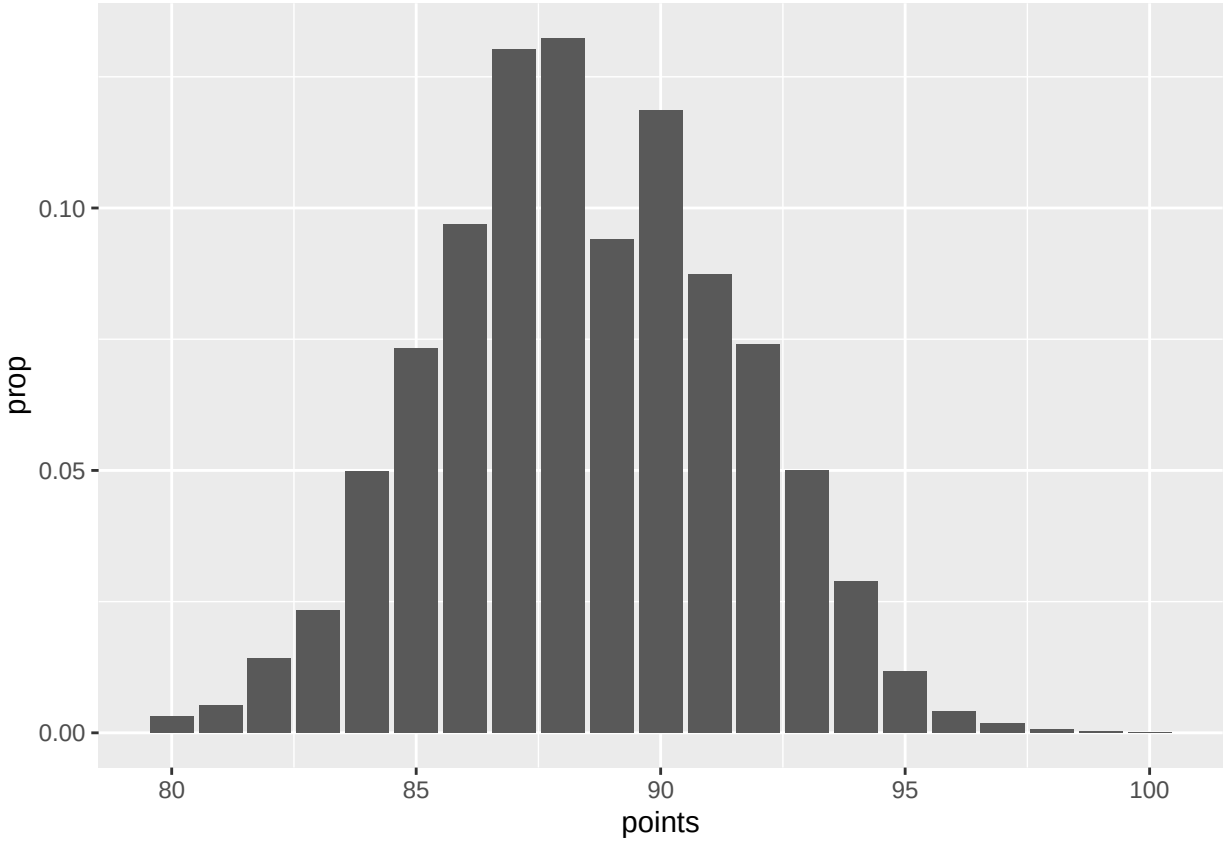
1.6 Points

Points is the variable we will be trying to predict.

```
points_tbl <- Wine %>%
    count(points, sort = TRUE) %>%
    mutate(prop = n / sum(n))

summary(points_tbl)
```

```
     points          n              prop
 Min.   : 80   Min.   :   19   Min.   :0.0001462
 1st Qu.: 85   1st Qu.:  523   1st Qu.:0.0040240
 Median : 90   Median : 3758   Median :0.0289141
 Mean   : 90   Mean   : 6189   Mean   :0.0476191
 3rd Qu.: 95   3rd Qu.:11359   3rd Qu.:0.0873964
 Max.   :100   Max.   :17207   Max.   :0.1323911
```

```
points_tbl %>%
  ggplot(aes(x = points, y = prop)) +
  geom_bar(stat = "identity")
```



The points look fairly normally distributed - there might be a slight right skew due to very few wines being rated over 95 points. The points range from 80-100 with both mean and median of 90 points.

1.7 Price

```
price_tbl <- Wine %>%
    count(price, sort = TRUE) %>%
    mutate(prop = n / sum(n)) %>%
    mutate(prop_cumulative = cumsum(prop))

summary(price_tbl)
```

```
     price              n                prop           prop_cumulative
 Min.   :   4.0   Min.   :   1.0   Min.   :7.690e-06   Min.   :0.06922
 1st Qu.: 101.2   1st Qu.:   1.0   1st Qu.:7.690e-06   1st Qu.:0.98640
 Median : 203.5   Median :   4.0   Median :3.078e-05   Median :0.99761
 Mean   : 293.9   Mean   : 332.4   Mean   :2.558e-03   Mean   :0.95011
 3rd Qu.: 369.8   3rd Qu.:  47.0   3rd Qu.:3.616e-04   3rd Qu.:0.99925
 Max.   :3300.0   Max.   :8996.0   Max.   :6.922e-02   Max.   :1.00000
 NA's   :1
```

We can see from the table that the price for wine ranges between 4 and 3,300 USD. More than 98% of the wines are under 101.20 USD, and more than 99.7% of the wines are less than 203.5 USD.

1.8 Description

Here is an example of the description.

```
Wine %>% pull(description) %>% pluck(1)
```

[1] "Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity."

This is one example. We will want to extract features from the description in order to incorporate this information into any model we do.

1.9 Province and Regions

Province and regions are related to country for self-explanatory reasons. Region is a smaller area of a province. Let's just explore province.

```r
top_province_tbl <- Wine %>%
    count(province, sort = TRUE) %>%
    mutate(prop = n / sum(n)) %>%
    mutate(prop_cumulative = cumsum(prop))

summary(top_province_tbl)
```

```
      province                 n                  prop          
    Length:426         Min.   :    1.00   Min.   :7.690e-06  
    Class :character   1st Qu.:    3.00   1st Qu.:2.308e-05  
    Mode  :character   Median :   12.00   Median :9.233e-05  
                       Mean   :  305.10   Mean   :2.347e-03  
                       3rd Qu.:   53.75   3rd Qu.:4.135e-04  
                       Max.   :36247.00   Max.   :2.789e-01  
    prop_cumulative  
    Min.   :0.2789   
    1st Qu.:0.9733   
    Median :0.9935   
    Mean   :0.9593   
    3rd Qu.:0.9987   
    Max.   :1.0000   
```

```r
top_province_tbl %>%
  arrange(desc(prop)) %>%
  head()
```

```
# A tibble: 6 x 4
  province       n   prop prop_cumulative
  <chr>      <int>  <dbl>           <dbl>
1 California 36247 0.279           0.279
2 Washington  8639 0.0665          0.345
3 Bordeaux    5941 0.0457          0.391
4 Tuscany     5897 0.0454          0.436
5 Oregon      5373 0.0413          0.478
6 Burgundy    3980 0.0306          0.508
```

Unsurprisingly, provinces in the U.S., France, and Italy dominate the top provinces list. A whopping 28% of our wines come from California alone.

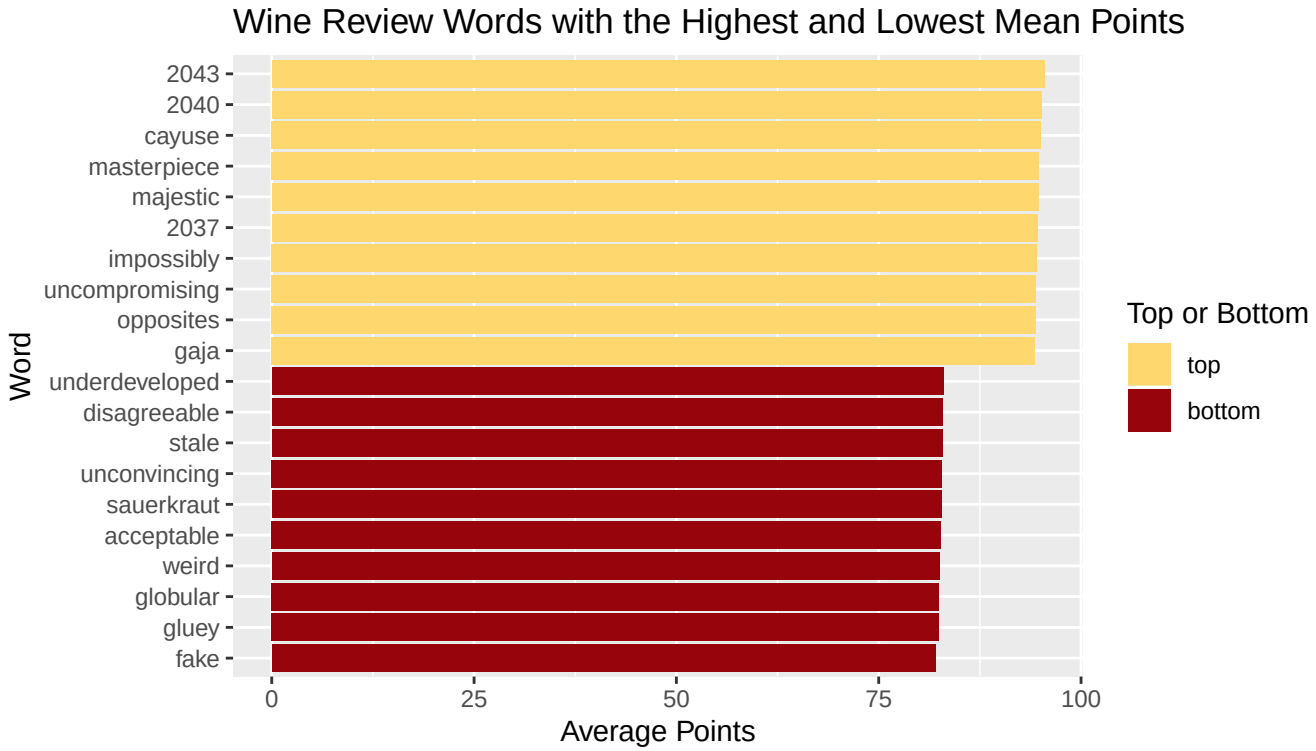
Chapter 2 Text Analysis and Prediction

2.1 Extracting Features from Text

A key part of this project is learning how to extract features from text. In the case of our data with wine reviews, the largest body of text we have is from the description variable. First we'll load in our data.

First, we can look at some exploratory plots. For example, following code from Kaggle user nnnnick (2018)[1], we can look at the words in the wine description with the highest and lowest mean scores:

[1] ("Predicting wine ratings using lightgbm + text2vec," 2018)



For the words with the top mean points, we can see several words that are actually years in the future - for example, 2043 and 2040. This is because many wine reviews of good wines will included something like "This wine will age well to 2040" or something of the sort. Some words in the top 10 with the highest mean point values may not be familiar to most people. "Cayuse," for example, is the name of vineyards in Washington state that are famous for tasty wines. Simialrly, "Gaja" is the name of a famous Italian producer of wine.

Looking at the words with the lowest mean points, nobody wants to drink a wine that's described as "gluey" or "fake." How often do these words show up though, and how can we use them in a predictive model? To find out, we need to create a document-term matrix, which shows the the frequency of terms that occur in a collection of documents.

2.1.1 Creating a Document-Term Matrix (DTM)

A DTM is a matrix in which the rows correspond to documents in a corpus (in our case, each wine description constitutes a document) and each column corresponds to terms, or words. This code to create a DTM for our wine dataset is based on this blog post[2].

```
#Create DTM
dtm <- CreateDtm(Wine$description,
                 doc_names = Wine$id,
                 ngram_window = c(1, 1),
                 lower = TRUE,
                 remove_punctuation = TRUE,
                 remove_numbers = TRUE,
                 stem_lemma_function = wordStem)
```

The resulting DTM is huge - about 43 megabytes with close to 3 billion elements. We need to create some functions that will allow us to use the DTM:

```
#Create functions
get.token.occurrences<- function(dtm, token)
  dtm[, token] %>% as.data.frame() %>% rename(count=".") %>%
  mutate(token=row.names(.)) %>% arrange(-count)


get.total.freq<- function(dtm, token) dtm[, token] %>% sum


get.doc.freq<- function(dtm, token)
```

[2] Ho (2018)

```
  dtm[, token] %>% as.data.frame() %>% rename(count=".") %>%
  filter(count>0) %>% pull(count) %>% length
```

Now we can see how many wines are actually described as "fake":

```
dtm %>% get.doc.freq(wordStem("fake"))
```

```
[1] 13
```

Which 13 wines?

```
fakewines <- dtm %>% get.token.occurrences(wordStem("fake")) %>% head(13)
Wine$title[c(as.numeric(fakewines$token))]
```

```
 [1] "Robert Stemmler 2005 Nugent Vineyard Pinot Noir (Russian River Valley)"
 [2] "Funky Llama 2011 Merlot (Mendoza)"
 [3] "Pierre Chardigny 2015 Vieilles Vignes  (Saint-Véran)"
 [4] "Mellisoni 2016 Estate Pinot Grigio (Lake Chelan)"
 [5] "Pradorey 2016 Tempranillo-Merlot Fermentado en Barrica Rosado (Ribera del Duero
 [6] "Skylite 2005 Skylite Vineyard Merlot (Walla Walla Valley (WA))"
 [7] "Black Stallion 2014 Cabernet Sauvignon (Napa Valley)"
 [8] "Adega Cooperativa Ponte de Barca 2013 Ela Rosé (Vinho Verde)"
 [9] "St. Julian 2013 Reserve Pinot Grigio (Lake Michigan Shore)"
[10] "Cannonball 2010 Cabernet Sauvignon (California)"
[11] "Finca Patagonia 2015 Expedicion Pinot Noir (Maule Valley)"
[12] "Loken Cellars NV Reserve Lot 14 Rosé (California)"
[13] "St. Andrews Estate 2000 Ceravolo Chardonnay (Adelaide Hills)"
```

Let's look at the description of the fourth wine:

[1] “Scattershot aromas of generic berry and cinnamon smell forced and fake. This has a tannic scrubbing mouthfeel and artificial flavors of chocolate and cheap oak. A green note and burn on the finish do nothing to help this along.”

Yikes. This seems like a bad wine.

From our document-term matrix, we could create a list of the top words by frequency and use those in our predictive model. However, if we were basing our top words by term frequency, we would be including words that are used so often that they probably don't have much meaning. Therefore, a better way to rank our words would be term frequency-inverse document frequency, which will be explained in the next section.

2.1.2 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistic that evaluates how important a word is in a document or corpus. It is calculated through dividing the term frequency of how often a word appears in a document by its inverse document frequency, which is the inverse of the proportion of how many documents in a corpus have a certain term. Therefore, high frequency terms but with little importance such as “the” or “and” will have low TF-IDF values, and so TF-IDF can be used as a weighting measure in ranking.

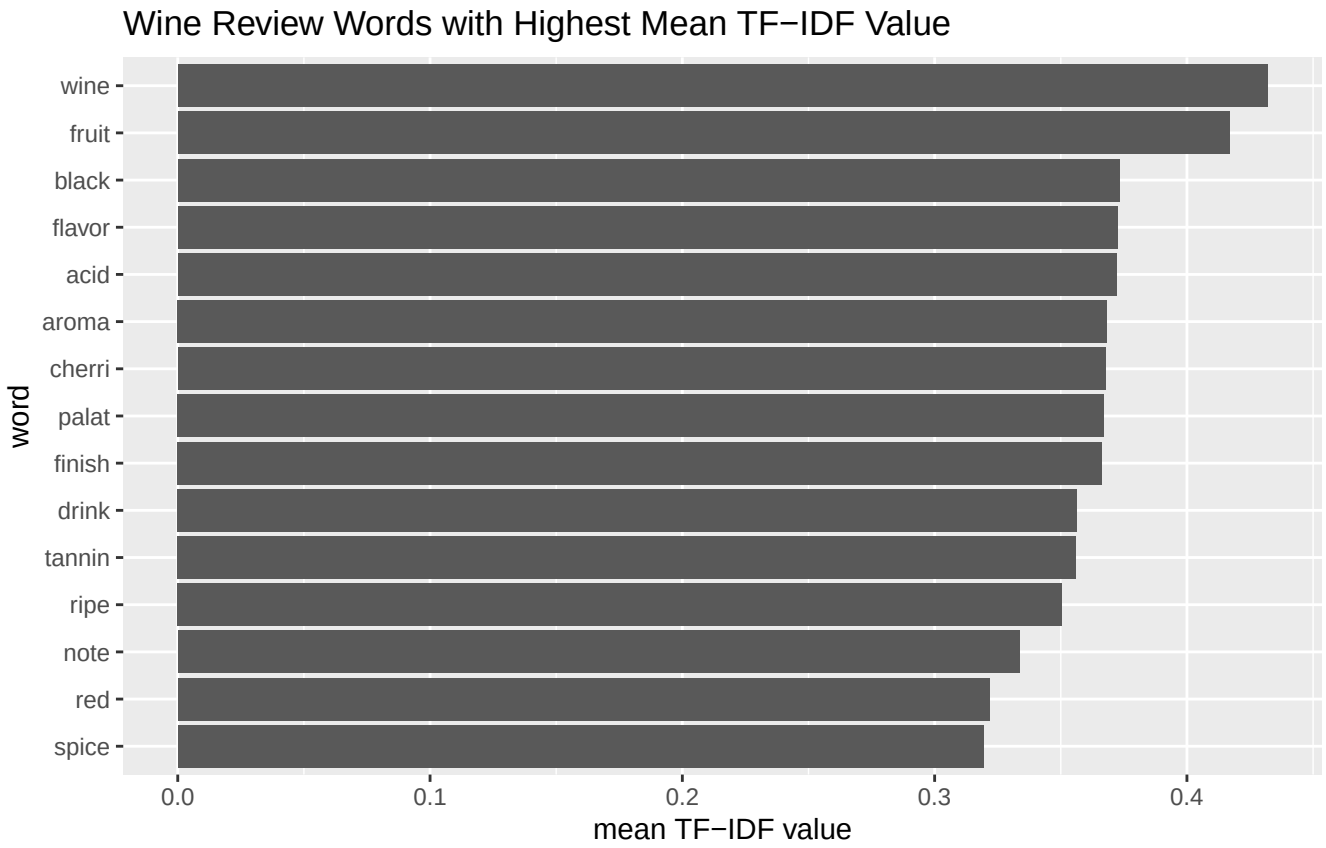
TF-IDF will be useful in our prediction model becuase we can include the words with the higest mean TF-IDF values in our model. Let's see what these words with the highest mean TF-IDF values are. The following code has been modified from the one of the cran.R vignettes of the `textmineR` package[3].

[3]Jones (2018)

```
tf_mat <- TermDocFreq(dtm = dtm)
head(tf_mat[ order(tf_mat$term_freq, decreasing = TRUE) , ], 10)
```

```
          term term_freq doc_freq       idf
wine      wine     83107    66140 0.6755376
flavor flavor     70968    65697 0.6822581
fruit   fruit     63935    55692 0.8474748
aroma   aroma     41052    40492 1.1662069
finish finish     40466    40083 1.1763590
acid     acid     39812    38586 1.2144218
palat   palat     38636    37796 1.2351081
drink   drink     33970    33244 1.3634370
cherri cherri     33590    31328 1.4227991
tannin tannin     32981    31960 1.4028262
```

```
tfidf_mat <- t(dtm[ , tf_mat$term ]) * tf_mat$idf #calculating TF-IDF
tfidf <- t(tfidf_mat)
tfidf_means <- colMeans(tfidf) #calculating mean values
tfidf_means <- as.data.frame(tfidf_means)
tfidf_means$word <- rownames(tfidf_means)
#top 200 with highest mean TF-IDF values
top200 <- tfidf_means %>% arrange(desc(tfidf_means)) %>% top_n(200, tfidf_means)
```



"Wine" and "fruit" have especially high mean TF-IDF values, followed by "black," "flavor," "acid," and "aroma."

Now let's see how we can use the DTM in a data frame for prediction.

```
# Match dtm column names to the words with top 200 mean tf-idf values
dtm_small <- dtm[, colnames(dtm) %in% top200$word]
ncol(dtm_small)
```

```
[1] 200
```

We're left with a document-term matrix with the 200 words with the highest mean tf-idf value.

2.2 Prediction

Let's look at predicting wines. We are looking to build a model that can be implemented for an average user of our web app to input values and text and receive an output of points.

Because we have a lot of missing data and a mixture of numerical and categorical data, methods like random forest are difficult to implement. Let's try gradient boosting, which in R can include categorical variables of up to 1024 categories (unlike randomForest, which only allows up to 53 categories per categorical variable).

First, we need to clean our data. We will remove variables that either 1) are factor variables with more than 1024 categories, or 2) are variables that are not necessarily relevant to an average person looking to explore wine. An example of variables in the latter category are `taster_name` and `taster_twitter_handle`.

```
dtm_df <- as.data.frame(as.matrix(dtm_small))
dtm_df$id <- c(0:129970)
Wine_dtm <- left_join(Wine, dtm_df, by = "id")
Wine_dtm <- Wine_dtm %>%
  select(-id, -description, -designation, -region_1, -region_2, -taster_name,
         -taster_twitter_handle, -title, -winery) %>%
  mutate(country = as.factor(country),
         province = as.factor(province),
         variety = as.factor(variety))
```

Now we will split our data into training and test sets.

```
# Test/Train split
set.seed(1)
smp_size <- floor(0.8 * nrow(Wine_dtm))
train_ind <- sample(seq_len(nrow(Wine_dtm)), size = smp_size)
train <- Wine_dtm[train_ind, ]
test <- Wine_dtm[-train_ind, ]
```

Now we can apply a gradient boosting algorithm to create our prediction model. First, let's try a boosting model with 5 trees. This is not a lot of trees, so we'd expect that the model wouldn't do so well with prediction on our test set.

```
set.seed(1)
boost_wine5 <- gbm(points ~ .,
                   data = train,
                   distribution = "gaussian",
                   n.trees = 5,
                   interaction.depth = 4)

boost_estimate5 <- predict(boost_wine5, # predict on test set
                           newdata = test,
                           n.trees = 5,
                           na.action = NULL)
mse_boost5 <- mean((boost_estimate5 - test$points)^2)
sqrt_mse5 <- sqrt(mse_boost5); sqrt_mse5 # calculate MSE
```

```
[1] 2.659352
```

With this model, the prediction is off by an average of 2.6593525 points. That's not

great. Let's compare this square root of the MSE to that of a model where we use 500 trees.

```
set.seed(1)
boost_wine <- gbm(points ~ .,
                  data = train,
                  distribution = "gaussian",
                  n.trees = 500,
                  interaction.depth = 4)

# Save model to .rds file
saveRDS(boost_wine, "boost_wine500.rds")
```

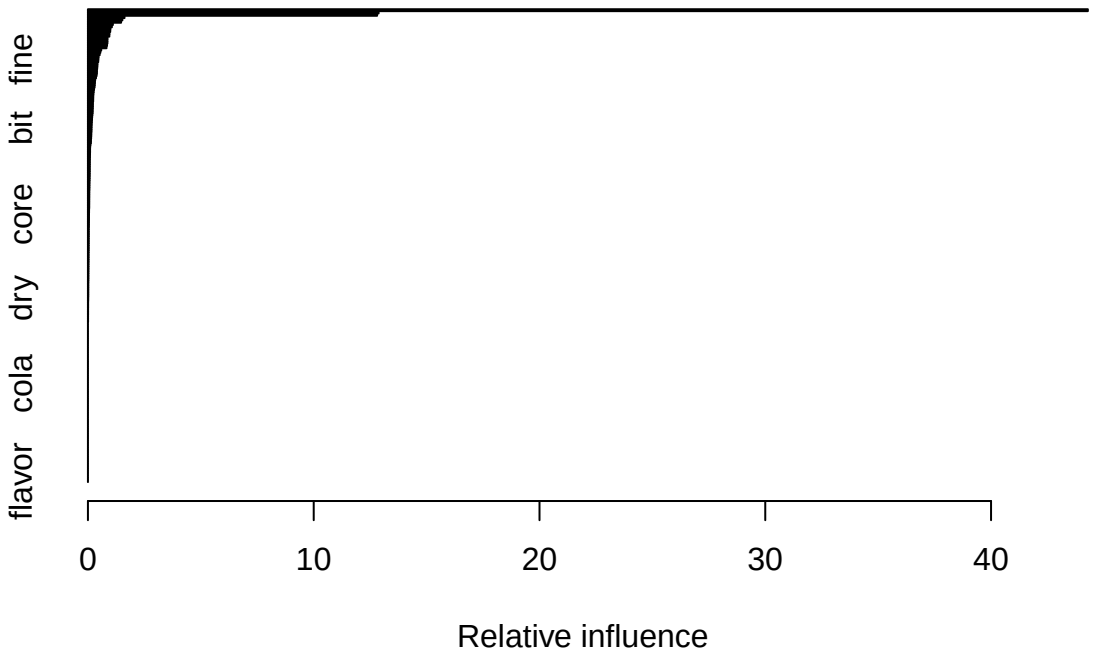
Running the above chunk takes too long, so we've loaded the model in the chunk below. We can check to see how well this model with 500 trees does with prediction on the test set:

```
# load model
boost_wine <- readr::read_rds("~/git/Stat495-F19-GroupD/FinalProject/index/data/boost_wi
boost_estimate <- predict(boost_wine,
                          newdata = test,
                          n.trees = 500,
                          na.action = NULL)
mse_boost <- mean((boost_estimate - test$points)^2)
sqrt_mse <- sqrt(mse_boost); sqrt_mse
```

```
[1] 1.834457
```

The square root of the mean squared error is 1.834457 - much smaller than that of the model with only 5 trees. Let's look at the most important features in this more accurate model.

```
top_n(summary(boost_wine), 10, rel.inf)
```




```
         var    rel.inf
1      price 44.2872441
2    variety 12.8744278
3   province 12.8074735
4       rich  1.6195324
5    complex  1.5243999
6      simpl  1.4679213
7       long  1.1043904
8     delici  1.0599025
9      black  0.9946763
10 concentr  0.9875296
```

Unsurprisingly, the variable with the most relative influence is price, followed by variety, then province. Stemmed words that are the most important are "rich", "complex," "simpl."

Let's see how this model predicts the points of a wine that is not in the data set.

For example, we can predict the points of a Portuguese Red Touriga Nacional wine that is $35 from Dão with the description: "This is a solidly structured wine that has big tannins in place. That will change as the wine ages further, bringing the rich black fruits forward and reveling in the perfumed acidity of the wine. Drink from 2021."

Below, we have created a function called `estimatepoints` that uses inputs of country, price, description, province, and variety and returns a points estimate based on our model.

```
estimatepoints <- function(country, price, description, province, variety) {
  newwine <- data.frame(id = 1, country, price, description, province,
                        variety) %>%
    mutate(description = as.character(description))

  # Create DTM for new wine
  dtm_newwine <- CreateDtm(newwine$description,
                doc_names = newwine$id,
                ngram_window = c(1, 1),
                lower = TRUE,
                remove_punctuation = TRUE,
                remove_numbers = TRUE,
                stem_lemma_function = wordStem)
  # words in top 200
```

```
  dtm_newwine2 <- dtm_newwine[, colnames(dtm_newwine) %in% top200$word]
  dtm_newwine_df <- as.data.frame(as.matrix(t(dtm_newwine2)))
  # find the top 200 words not in new wine but in dtm
    otherwords <- top200 %>%
    filter(!word %in% dtm_newwine_df)
  # fill the words not in new wine to df with 0
  dtm_newwine_df[otherwords$word] <- 0

  newwine <- cbind(newwine, dtm_newwine_df) # fill in rest of data

  # Estimate points of new wine
  test_estimate <- predict(boost_wine,
                           newdata = newwine,
                           n.trees = 100,
                           na.action = NULL)

  return(test_estimate)
}

# Save function & top200 data frame for use in Shiny app
save(top200, estimatepoints, file="estimatepoints_function.Rda")
```

So now to predict the number of points our Portuguese wine would receive, we can just plug in the input values.

```
estimatepoints(country = "Portugal",
               price = 35,
               description = "This is a solidly structured wine that
               has big tannins in place. That will change as the wine
               ages further, bringing the rich black fruits forward
               and reveling in the perfumed acidity of the wine.
               Drink from 2021.",
               province = "Dão",
               variety = "Touriga Nacional, Portuguese Red")
```

```
[1] 88.83818
```

Our model predicts this wine would receive about 89 points. Not bad.

Model with just description

The model we are using to predict points uses variables like price, province, and variety, which all have much higher relative influence compared to just the words in the description. What if we removed these variables with higher relative influence and looked just at how well words in the description can predict points?

We will select just the points variable and the word variables and set up a training and test set to create the same gradient boosting model with just the word variables.

```
# leaves just the DTM of words
Wine_justwords <- Wine_dtm %>%
  select(-country, -price, -province, -variety)


# Train/Test split
set.seed(1)
```

```
smp_size2 <- floor(0.8 * nrow(Wine_justwords))
train_ind2 <- sample(seq_len(nrow(Wine_justwords)), size = smp_size2)
train2 <- Wine_justwords[train_ind2, ]
test2 <- Wine_justwords[-train_ind2, ]
```

Now we can fit the model. Again, this algorithm takes a while to run, so the code to create the model is shown in the first code chunk below, but we will just load the model object into the next code chunk for analysis.

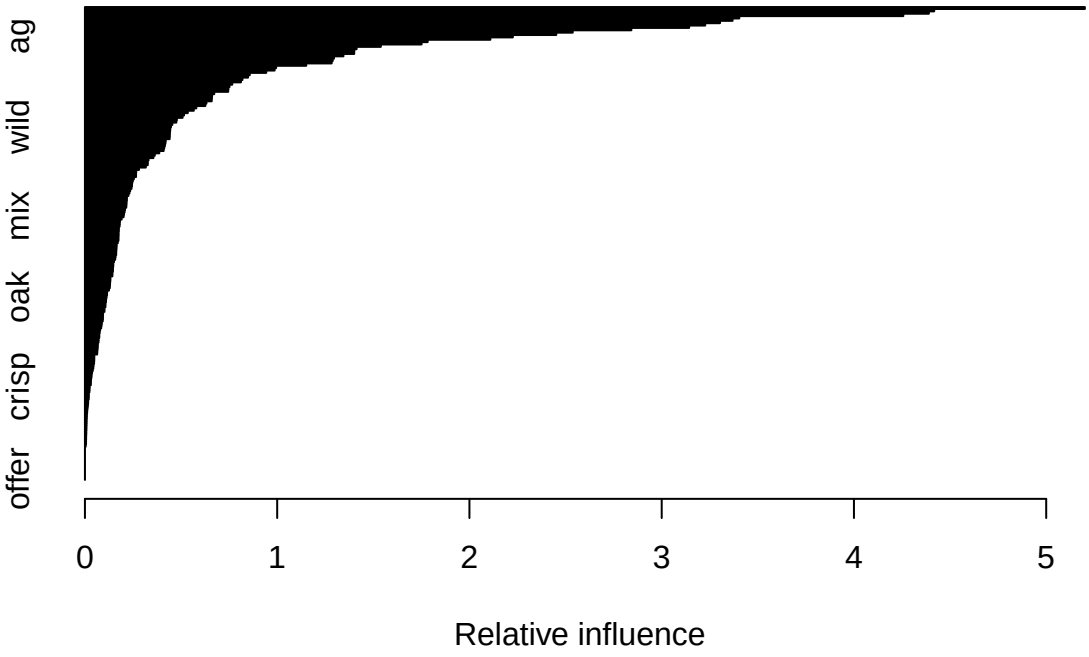
```
set.seed(1)
boost_wine_words <- gbm(points ~ .,
                    data = train2,
                    distribution = "gaussian",
                    n.trees = 500,
                    interaction.depth = 4)

# Save model to .rds file
saveRDS(boost_wine_words, "boost_wine_words.rds")
```

```
# load model
boost_wine_words <- readr::read_rds("~/git/Stat495-F19-GroupD/FinalProject/index/d
boost_estimate_words <- predict(boost_wine_words,
                            newdata = test2,
                            n.trees = 500,
                            na.action = NULL)
mse_boost_words <- mean((boost_estimate_words - test2$points)^2)
sqrt_mse_words <- sqrt(mse_boost_words); sqrt_mse_words
```

```
[1] 2.177103
```

The model with just words performs a little worse than our model with price, variety, and province included with a mean prediction error of 2.1771026 points. We can look at which words have the most relative influence in this model:

```
top_n(summary(boost_wine_words), 10, rel.inf)
```




```
        var  rel.inf
1      rich 5.201027
2  vineyard 4.417346
3   complex 4.389741
4     simpl 4.256652
5  concentr 3.403552
6     black 3.368652
7      year 3.303258
8      long 3.226283
```

```
9     power 3.142659
10     eleg 2.842253
```

The word "rich" has the highest relative influence, followed by words like "vineyard," "complex," the stemmed "simpl," and the stemmed "concentr."

However, because the model with price, province, and variety included has a lower MSE, and because these variables would not be difficult to find for an average wine drinker, we will use the model with 500 trees with price, province, and variety included to build our prediction engine.

Chapter 3 Geocoding and Visualization

3.1 Why Geocode?

Geocoding locations is generally a good idea because it allows for spatial analysis and spatial visualizations. In our case, geocoding the wine reviews will allow us to create interactive maps that visualize the wineries and allows users to explore our dataset.

3.2 Using ggmap

The most convenient approach to perform geocoding in R is to use the `ggmap` package. However, the rather recent change in Google's API requires setting up a project and generating a key in the Google Cloud Project. While we are billed for every observation that we geocode, we have credits available each month. To find out more about the API usage and billing refer to the official website. To learn more about the `ggmap` package check out the project's Github repository.

3.2.1 Reading in the Data

```
data_dir <- "../data"
file_name <- "winemag-data-130k-v2.csv"


path <- file.path(data_dir, file_name)
```

```
Wine <- read_csv(path) %>%
    rename(id = X1) %>%
    mutate(id = id + 1)
```

The `mutate_geocoded` function from `ggmap` would be great if it actually had some error handling. However, it is not robust enough for our needs. Instead, we will create our own function to handle both network and API errors, while ensuring completion of our code.

We will operate in a small sample of observations. This will allow us to test our function and then apply it to the whole dataset.

```
set.seed(2019)


subset <- Wine %>%
    count(winery, country) %>%
    mutate(address = paste0(winery, " ", country)) %>%
    sample_n(20)
```

3.2.2 Setting Up Helper Function

We mentioned that the function we use to geocode our observations has to be robust. We take advantage of the `purrr` adverbs to handle internal messages that slow down the operation of the geocoding function in `ggmap`. Additionally, we prevent network failures from being an issue by allowing failed requests to retry at most once after a short delay. Finally, in order to be able to ensure the completion of our code without errors, we wrap the geocoding functionality with an alternative for when the function fails.

You can see all these components working together in the function below.

```
geocode_robustly <-
    possibly(
        insistently(
            quietly(geocode),
            rate = rate_delay(0.1, max_times = 2)),
        otherwise = list(result = tibble(lon = NA_real_, lat = NA_real_))
    )
```

It is important to make sure that the `otherwise` argument matches the type of output given by the function that is wrapped by the `possibly` adverb.

Finally, we apply our function to the addresses of the observations.

```
locations <- subset %>%
    pull(address) %>%
    map_dfr(~ geocode_robustly(.x)$result)


subset %>% bind_cols(locations)
```

Note that there will be some `NA` values in our location variables because we are largely relying on the integrity of the dataset and the Google Maps search engine.

Confirming that our function is operating as desired, we can now we geocode all the units in our dataset.

3.2.3 Putting It All Together

In the code below we perform a further optimization by only geocoding the set of wineries. This allows us to avoid performing slow network requests on observations that have already been geocoded. The following represents every step taken to geocode our dataset with more than 100K observations. (The code will take some time to run.)

```r
# 1. Add address column to geocode
Wine <- Wine %>%
    mutate(address = paste0(winery, " ", country))

# 2. Get unique addresses
Addresses <- Wine %>%
    count(address) %>%
    select(-n)

# 3. Geocode unique addresses
Locations <- Addresses %>%
    pull(address) %>%
    map_dfr(~ geocode_robustly(.x)$result)

# 4. Bind location info to addresses
Addresses <- Addresses %>%
    bind_cols(Locations)

# 5. Join into original dataset
Geocoded_Wine <- Wine %>%
    left_join(Addresses, by = "address")

# 4. Save geocoded dataset
file_name <- "geocoded.csv.gz"
Geocoded_Wine %>% write_csv(path = file.path(data_dir, file_name))
```

Now we have a geocoded version of the dataset that we can further refine for our analysis.

3.3 Verify

To prove that our geocoding worked we can import our new dataset.

```
Wine <- read_csv("../data/geocoded.csv.gz") %>%
    glimpse()
```

```
Observations: 129,971
Variables: 17
$ id                    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1...
$ country               <chr> "Italy", "Portugal", "US", "US", "US", "...
$ description           <chr> "Aromas include tropical fruit, broom, b...
$ designation           <chr> "Vulkà Bianco", "Avidagos", NA, "Reserve...
$ points                <dbl> 87, 87, 87, 87, 87, 87, 87, 87, 87, 87, ...
$ price                 <dbl> NA, 15, 14, 13, 65, 15, 16, 24, 12, 27, ...
$ province              <chr> "Sicily & Sardinia", "Douro", "Oregon", ...
$ region_1              <chr> "Etna", NA, "Willamette Valley", "Lake M...
$ region_2              <chr> NA, NA, "Willamette Valley", NA, "Willam...
$ taster_name           <chr> "Kerin O'Keefe", "Roger Voss", "Paul Gre...
$ taster_twitter_handle <chr> "@kerinokeefe", "@vossroger", "@paulgwin...
$ title                 <chr> "Nicosia 2013 Vulkà Bianco  (Etna)", "Qu...
$ variety               <chr> "White Blend", "Portuguese Red", "Pinot ...
$ winery                <chr> "Nicosia", "Quinta dos Avidagos", "Rains...
$ address               <chr> "Nicosia Italy", "Quinta dos Avidagos Po...
```

```
$ lon                   <dbl> 14.395278, -7.276971, -95.712891, -85.89...
$ lat                   <dbl> 37.74692, 41.38793, 37.09024, 42.21225, ...
```

3.4 Building Our Shiny App

Our Shiny app can be found here. There are 3 components: first, an interactive map that allows users to view wineries of the world and filter based on variety. Users can see average points or average price of the wines around the world. The second component of the Shiny app is a searchable catalog, where users can search among any of the wines included in the catalog and geolocate them on the map. The third and final component of our Shiny app is a prediction engine, which allows users to input their own wine with country, province, description, price, and variety, and our prediction model described in the previous chapter will produce an estimate of the number of points the inputted wine would receive.

Conclusion

This project set out to make wine easier to understand for the average person. Through this project, we examined close to 130,000 wine reviews to create a complex web application via Shiny that allows users to explore wines and wineries through an interactive map and searchable catalog as well as to enter new wine observations to receive a prediction of how many point values their new wine would receive. Through the map and catalog explorer and prediction engine, anyone can learn more about wine and explore the different varieties around the world.

This project also involved text analysis beyond the scope of our Stat-495 class, researching into word ranking methods such as term frequency-inverse document frequency and learning how to incorporate document-term matrices into our prediction model. Our gradient boosting model showed that the most important features to predicting wine points are price, variety, and province. The most important word for prediction is "rich."

3.5 Limitations and Future Directions

There were several limitations to the work in this project. They will be discussed below, and various suggestions for future directions to address these limitations will be described.

3.5.1 Data Scraping

The Kaggle dataset we used originated from Wine Magazine's website and was scraped in 2017. However, in the most recent years, almost 100,000 more wines have been added to the website. A future direction this project could go into would be to use the python scraper provided by the Kaggle user who uploaded the original dataset[1] and scrape the newer wine reviews so there would be more up-to-date data.

3.5.2 Model Computation Run Time

A key limitation to the prediction aspect of this project was the sheer amount of computation involved. The model created in this project used only 200 of the words with the highest mean TF-IDF values, but models with many more words could have been created and would likely yield lower MSE values and better predictions. However, given how the gradient boosting algorithm is computationally intensive because it requires creating many small trees, we chose to limit the number of variables used in the dataset to run the model.

Future steps that could be made in order to speed up computation time and create a more accurate model could be to look into faster gradient boosting algorithm methods. In R, there are newer packages such as `XGBoost` and `LightGBM` that were developed specifically for decreasing run time of the computationally intensive gradient boosting algorithms, so the functions in these packages could be explored on the data in this project.

[1] Thoutt (2017)

3.5.3 Confirmation Bias

Although our analyses were run on the assumption that there was no bias among tasters who were rating wine points, we do not know how the original data was collected and whether the tasters were blind to various aspects about the wine they were tasting (for example, price or variety). It is possible that the reason the positive correlation between points and price was acually due to confirmation bias among tasters that more expensive wine should taste better. An interesting further direction this project could go in would be to explore whether different tasters had different patterns of rating wines to possibly discern if these tasters were biased.

References

Ho, S. (2018, April). Document-term matrix: Text mining in r and python. Retrieved from https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958

Jones, T. (2018). TextmineR vignette. Retrieved from https://cran.r-project.org/web/packages/textmineR/vignettes/a_start_here.html

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Boston, MA: Cambridge University Press.

Nabi, J. (2018, September). Machine learning - text processing. Retrieved from https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958

Predicting wine ratings using lightgbm + text2vec. (2018). Retrieved from https://www.kaggle.com/nnnnick/predicting-wine-ratings-using-lightgbm-text2vec

Thoutt, Z. (2017). Wine reviews. Retrieved from https://www.kaggle.com/zynicide/wine-reviews

Watson, I. (2010, April). Unearthing georgia's wine heritage. Retrieved from http:

//edition.cnn.com/2010/WORLD/europe/04/20/georgia.wine.heritage/

Wine enthusiast reviews. (2019). Retrieved from https://www.winemag.com/?s=&drink_type=wine&page=0