A Taste of French Wine

<Yun-Chung LIU>

Abstract

In this study, I briefly described the number of products and price level for French wine in the dataset extracted from wine enthusiast website. Then, I used the Aroma Wheel paradigm to dig out the taste of French wine from different provinces and presented a aroma-based recommendation system. Finally, I compared two classification methods, Naïve Bayes and K-Nearest Neighbor. The result suggests that Naïve Bayesian classifier can can predict the origin of French wine more accurately.

Motivation

Have you ever wondered how the wines from the old world taste like? Bordeaux, Chardonnay, Pinot Noir and other words that make you merry? In this study, I used the data from Wine Enthusiast to help you have a glimpse on the amazing world of French Wine. You will ”see” how these wine “taste” and “find” the wine for you! Come check it out!

Dataset

In this project, I used the wine review dataset downloaded from Kaggle (https://www.kaggle.com/zynicide/wine-reviews). The dataset was built by Zackthoutt using wine review data from Wine Enthusiast website (https://www.winemag.com/). The dataset contains more than 150,000 wine items with country of origin, region, province, designation, description, price, points given by tasters.

Research Questions

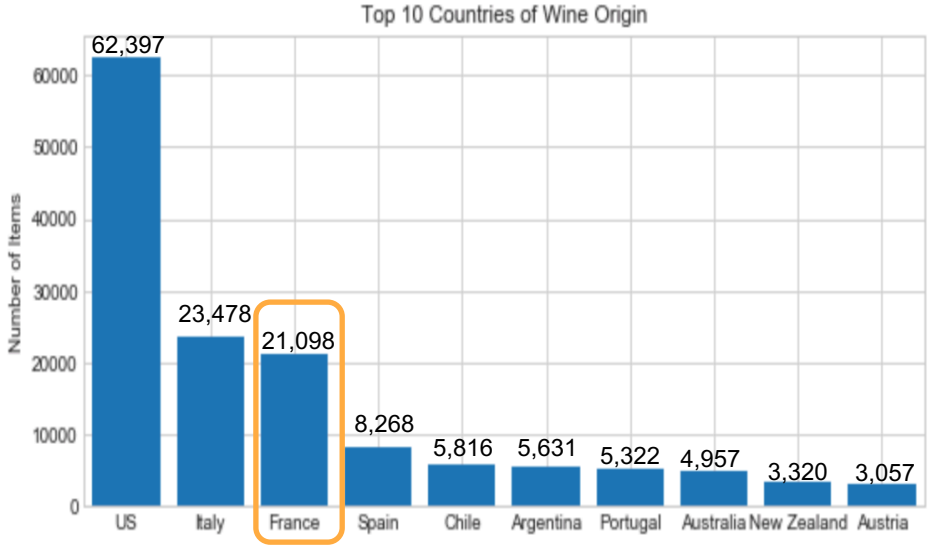
1. What are some characteristics in terms of number of items and price of French wine on the market?

2. What are some common words used to describe French wine? Can we use descriptions to distinguish province of wine origin with similar aroma?

3. Can we use descriptions of wine to predict the province of wine origin? Which classification method, Naïve Bayes or K-Nearest Neighbors, performs better?

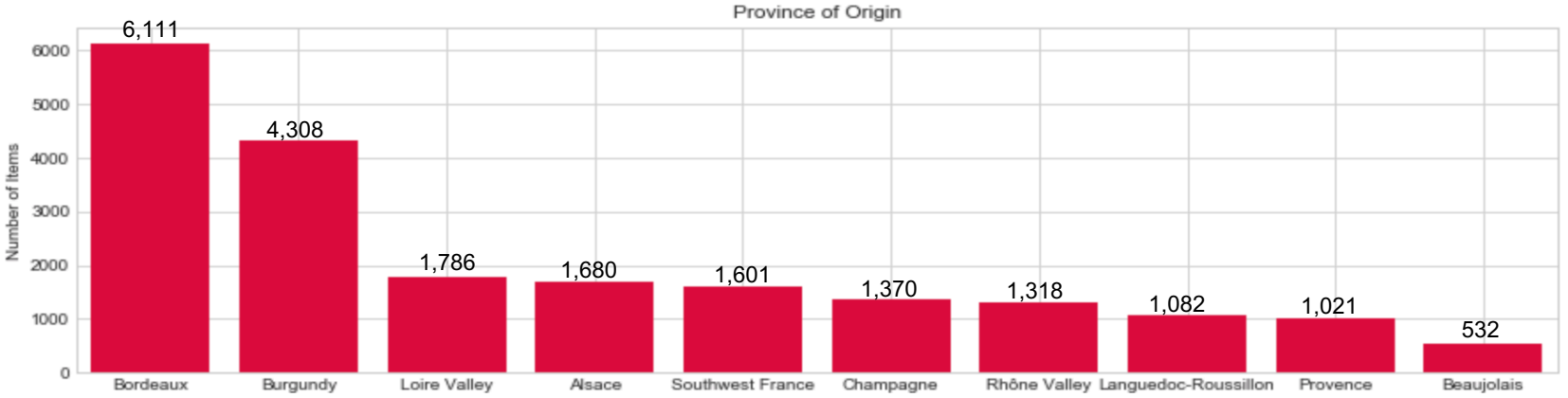
French Wine 101

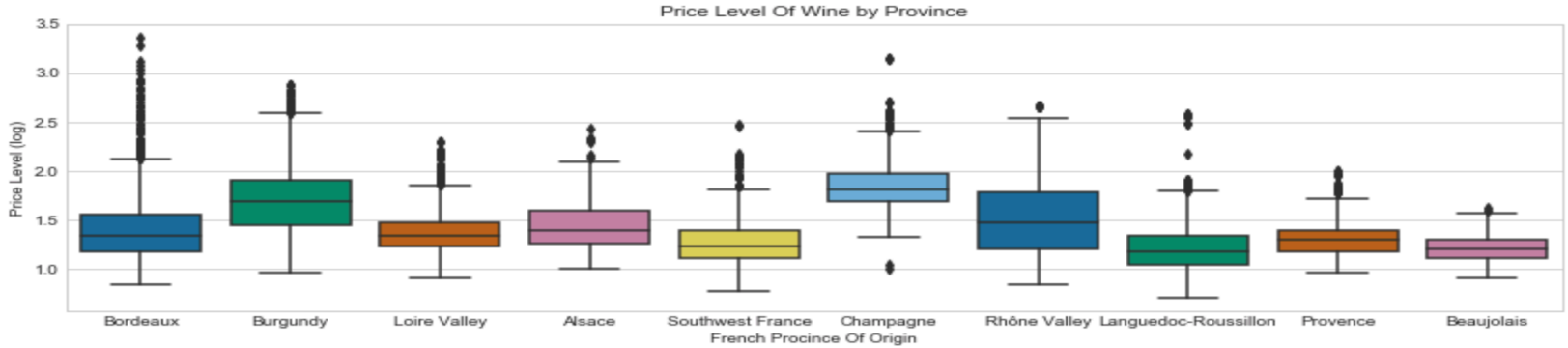
French wine represents 14.0% of items in the Wine Enthusiast platform, ranked the 3rd after the US and Italy. The average price of French wine is $45.6 (3rd among other countries) and the standard deviation is $69.7(1st among other countries), which indicates that price of French wine can vary.



French Wine by Province

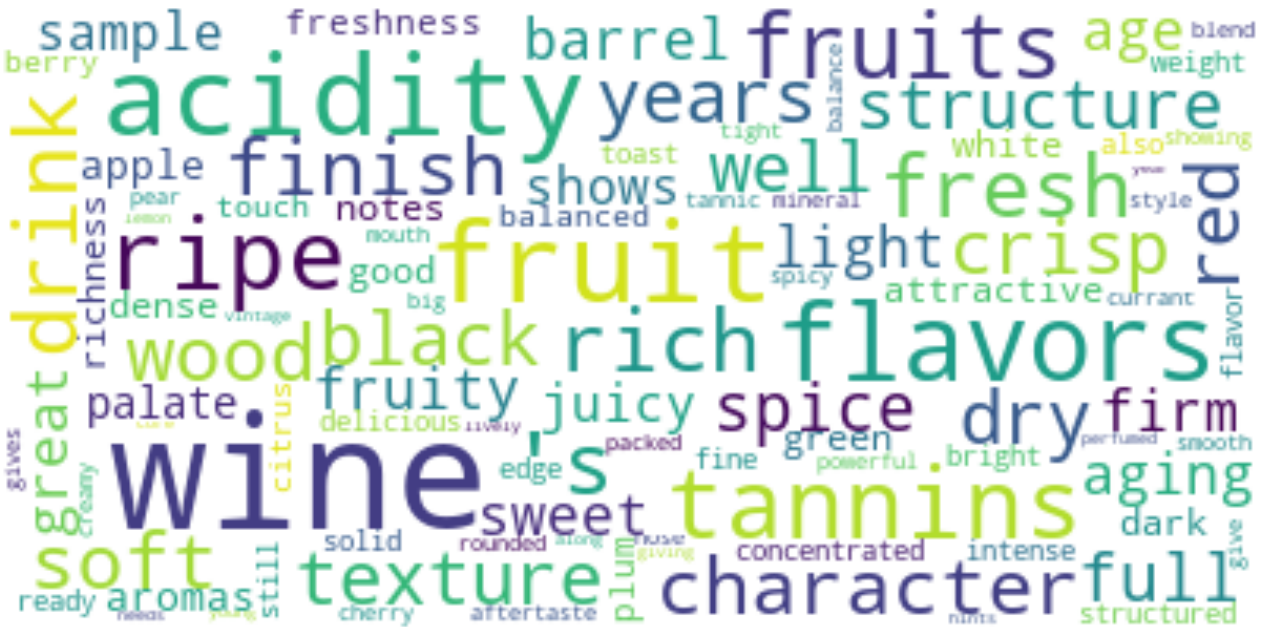
Most French wine comes from Bordeaux, representing 29.0% of French wine products. Burgundy wine ranked the second(20.4%). In terms of price, Champagne(avg. $93.4) and Burgundy(avg. $70.6) stand out. The average price of Bordeaux is not too high($42.6), but it has relative large dispersion.





A taste of French Wine

How does French wine taste like? Using all the description for French wine, several terms appear repetitively (word size represents the frequency of appearance in the corpus), such as fruit, fresh, tannins, dry, firm, wood, soft..etc. However, some of them are irrelevant to the taste (e.g. wine, structure).



The Aroma Wheel

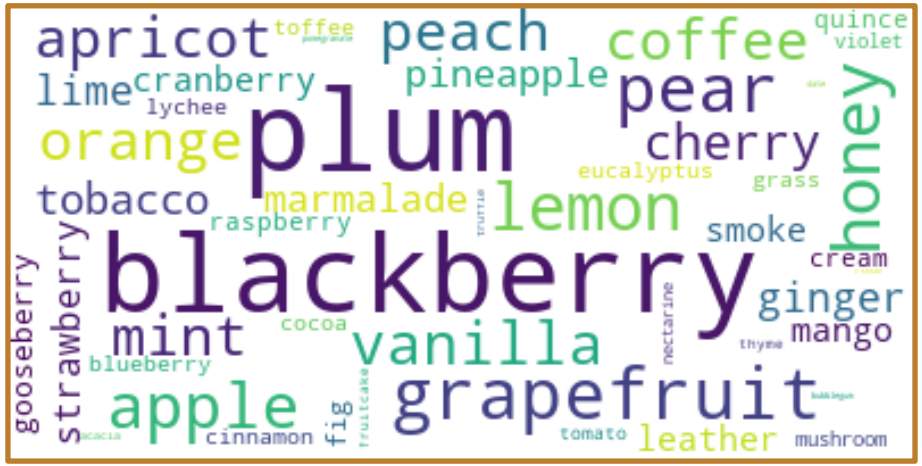
To filter out words that are related to the taste of wine, I used a modified version of the wine aroma wheel paradigm(https://winefolly.com/). In this paradigm, Noble et al. listed a wheel of words related to the wine aroma (related to the underlying chemical substances) to represent the taste of wine. The words are categorized in 3 tiers(see example below). In the project, I use the 3rd tier terms(the most detailed ones).

Table 1. The Aroma Wheel

Principal term	2nd tier term	3rd tier term
Caramelized	Carmel	Honey
Wood	Phenolic	Vanilla
	Burned	Smoke
Fruity	Citrus	Grapefruit
		Lemon
	Tropical Fruit	Pineapple
		Mango

The Aromas of Bordeaux and Burgundy Wine

Using the Aroma Wheel Paradigm, we can see the similarity and difference between Bordeaux and Burgundy wine. In Bordeaux wine description, plum (word count: 429), blackberry(word count: 419) and grapefruit(word count: 127) appear the most often. For Burgundy, people consider it to have the aroma of apple(word count: 397), plum(word count: 367) and pear(word count: 295) more often.



The Bordeaux Aroma

The Burgundy Aroma

The Aroma Distance

To quantify the similarity between different origin of wine, I first calculated the prrportion of every aroma word used to describe the wine (e.g. x_1 = wordcount of Aroma 1 / total Aroma word count of Province A).The Aroma Distance is the sum of the square root of squared difference between all Aroma words in the Aroma Wheel between Province A and B (see table 2).

Table 2. The Aroma Distance Methodology

Aroma Word	Province A	Province B	Distance
Aroma 1	x_1	y_1	$\sqrt{(x_1 - y_1)^2}$
Aroma 2	x_2	y_2	$\sqrt{(x_2 - y_2)^2}$
Aroma 3	x_3	y_3	$\sqrt{(x_3 - y_3)^2}$
… Aroma n	… x_n	… y_n	… $\sqrt{(x_n - yn)^2}$
The Aroma Distance between Province A and Province B ($0 \leq \text{distance} \leq 2$)			$\sum_{i=1}^{n} \sqrt{(x_n - yn)^2}$

The Aroma Distance Based Wine Recommendation

Using the aroma distance, we can recommend people the wine with the shortest Aroma Distance. For example, a person who likes Bordeaux a lot may be tempted to try a wine from Southeast France. A Burgundy fan can try a wine from Loire Valley. He or she may like it a lot because of the similar aroma.

Aroma Distance from Bordeaux wine

Province of Origin	Distance
Southwest France	0.43
Burgundy	0.79
Languedoc-Roussillon	0.92
Loire Valley	0.91
Rhône Valley	0.99
Provence	1.05
Alsace	1.05
Champagne	1.07
Beaujolais	1.37

Aroma Distance from Burgundy wine

Province of Origin	Distance
Loire Valley	0.59
Champagne	0.63
Provence	0.70
Southwest France	0.71
Alsace	0.74
Bordeaux	0.79
Languedoc-Roussillon	0.84
Rhône Valley	0.92
Beaujolais	1.27

The Aroma Distance Matrix

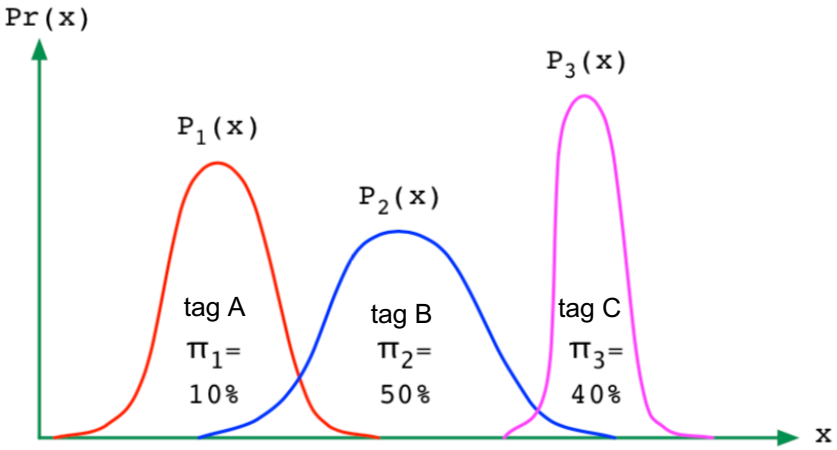
This matrix below summarizes the Aroma Distance between province of wine origins. Pink indicates short distance (0.44 – 0.91); purple represents medium distance(0.92-1.07) and blue implies long distance(1.08-1.63).

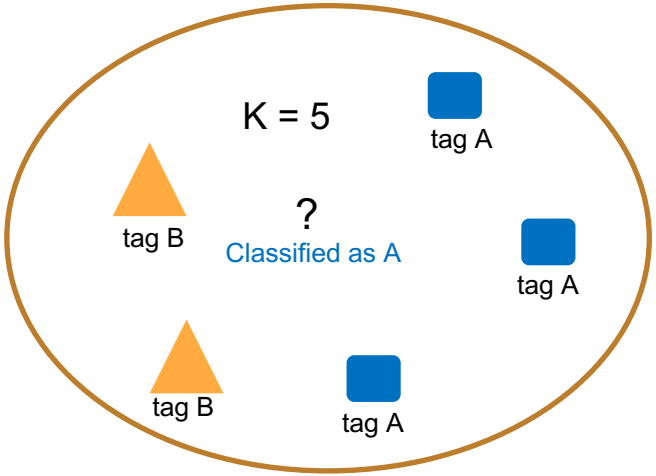
	Champagne	Bordeaux	Burgundy	Rhône Valley	Languedoc-Roussillon	Alsace	Southwest France	Loire Valley	Provence	Beaujolais
Champagne	-	1.07	0.63	1.24	1.20	0.64	1.00	0.44	0.96	1.63
Bordeaux	1.07	-	0.80	0.99	0.92	1.05	0.43	0.91	1.05	1.37
Burgundy	0.63	0.80	-	0.92	0.84	0.74	0.71	0.59	0.70	1.27
Rhône Valley	1.24	0.99	0.92	-	0.69	1.25	0.99	1.15	1.06	1.19
Languedoc-Roussillon	1.20	0.92	0.84	0.69	-	1.30	0.95	1.14	0.85	1.07
Alsace	0.64	1.05	0.74	1.25	1.30	-	0.97	0.54	1.11	1.78
Southwest France	1.00	0.43	0.71	0.99	0.95	0.97	-	0.83	0.95	1.36
Loire Valley	0.44	0.91	0.59	1.15	1.14	0.54	0.83	-	0.99	1.61
Provence	0.96	1.05	0.70	1.06	0.85	1.11	0.95	0.99	-	1.29
Beaujolais	1.63	1.37	1.27	1.19	1.07	1.78	1.36	1.61	1.29	-

Classifying Province of Wine Origin

Lastly, I compared the performance of two classification methods. Since words not included in the aroma wheel could affect the classification result (e.g. “sunshine”, “texture”), I included all the words in the description for the classification tasks.

Naïve Bayes Classification	K-Nearest Neighbors
Classify wine origin based on Bayes theory considering 1) probability of wine origin in the dataset 2) probability of word(features) describing wine from different province.	Calculate the distance between descriptions (the more words appeared in both description A and B, the less the distance is). Classify a new description based on the wine origin of the majority of k-nearest neighbors.





Classifying wine origin based on description

The result suggests that Naïve Bayes classifier performs better (accuracy rate: 70.1%) than K-Nearest Neighbors (accuracy rate: 21.3%).

	Naïve Bayes Classification	K-Nearest Neighbors
Preprocessing	Use the Natural Language Toolkit(NLTK) tokenizer to generate word list of wine descriptions, excluding meaningless and repetitive words, and punctuation.	
Train dataset	14,768 randomly selected wine description.	-
Train Accuracy	78.0%	-
Test dataset	6,329 randomly selected wine description.	500 randomly selected wine description * 10 provinces = 5,000 samples.
Test Accuracy	70.1%	20.6%

Discussion

Comparing the 2 classification methods, I found that Naïve Bayes classifier demonstrated far higher accuracy rate than K-nearest neighbors(KNN). This can be due to the fact that wine descriptions has higher sparsity. There are not many words on a single wine description. Thus, it works better when comparing a piece of wine description to the whole set of wine tag and description data(Naïve Bayes) rather than a single piece of wine description (KNN). Also, sample size matters, due to the nature of KNN, the sample size is limited to the wine origin with the lease amount of wine product.

Conclusions

In this study, I first revealed the status of French wine in terms of price and number of products on the market for consumer's reference. Then, I presented the taste of wine from different origins using word cloud. Based on the proportions of aroma words used to describe wine from different province, I developped an Aroma Distance based recommendation system for people to try to wine with aroma closer to their favorite wine origins. Finally, I compared 2 classification methods, Naive Bayes and K-Nearest neighbors. The result suggests that Naive Bayes classifier performs better on the task(accuracy rate higher than 70%).

This study demonstrates a possible recommendation paradigm based on natural language processing. It can not only be applied to wine recommendation system, but also it also demostrate a more effective methods for classification for few tags and large amount of features.

Limitations

The wine dataset used in this study came from a single US Based website. Some characteristics and products can be under-represented. Also, due to the limitation of time and resources, this study remains an exploratory analysis and demonstration of methodology. For example, I only considered one word aroma in the aroma wheel. Further effort is needed to provide information with business impact or insights on machine learning methodologies.

Future Work

With record of consumer buying behavior and website log, more sophisticated recommendation system can be formed. Thus, the effectiveness of the recommendation system can be measured and fine-tuned continuously.

Acknowledgements

First of all, I would like to dedicate this work to Christophe PALLIER, who inspired me to the world of Python. In this series of endeavor, I would like to first of all give many thanks to Fang-Yu, CHENG, gave me advice along the way of my data science-related endeavors. Also, Pei-Chun CHIANG gave me the idea of using the Aroma Wheel paradigm. Finaly, many thanks to Li-Tsu YANG for reviewing the materials.

References

Noble, A.C., Arnold, R.A., Buechsenstein, J., Leach, E.J., Schmidt, J. O. and Stern, P.M.(1987). Modification of a Standardized System of Wine Aroma Terminology. *American Journal of Enology and Viticulture*, 32(2), p.143-146.

Final_Project_Yun-Chung_LIU

December 18, 2018

0.1 A Data Science Journey of Wine

Hvae you ever wonder how a wine taste like? Have you evry got lost in the wide variety of wine to choose from? With data science. Let's walk through the "taste" of wine with data science methods. Come find your wine of choice without even one sip!

In this notebook. I will do some exploratory analysis on the wine dataset extracted from the website https://www.winemag.com in June. 2017.

First of all, let's import the necessary libraies that are likely to be used in the project.

```
In [1]: # import libraries
        import pandas as pd
        import numpy as np
        import sklearn
        import matplotlib.pyplot as plt
        import os
        import nltk
        import string
        import seaborn as sns
        import random
        from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
        from collections import Counter
```

```
In [2]: #Check what is in the directory
        print(os.listdir())
```

```
['.DS_Store', 'Images', 'wine-reviews', 'French_Wine_Review.png', 'Final Project_Yun-Chung LIU_
```

```
In [3]: Wine_Review = pd.read_csv('./wine-reviews/winemag-data_first150k.csv', sep =',')
```

The dataset contains more than 150 thousand rows of wine review data, including columns like **countries of origins**, **descriptions** of the wine, **designation**, review **points**, **price**, **province** and so on.

```
In [4]: Wine_Review.dropna()
        print(Wine_Review.shape)
```

```
(150930, 11)
```

```
In [5]: print(Wine_Review.head())
```

```
   Unnamed: 0 country                                        description  \
0           0      US  This tremendous 100% varietal wine hails from ...
1           1   Spain  Ripe aromas of fig, blackberry and cassis are ...
2           2      US  Mac Watson honors the memory of a wine once ma...
3           3      US  This spent 20 months in 30% new French oak, an...
4           4  France  This is the top wine from La Bégude, named aft...

                          designation  points  price        province  \
0                   Martha's Vineyard      96  235.0      California
1  Carodorum Selección Especial Reserva      96  110.0  Northern Spain
2         Special Selected Late Harvest      96   90.0      California
3                               Reserve      96   65.0          Oregon
4                            La Brûlade      95   66.0        Provence

            region_1           region_2             variety  \
0        Napa Valley               Napa  Cabernet Sauvignon
1               Toro                NaN       Tinta de Toro
2    Knights Valley             Sonoma     Sauvignon Blanc
3  Willamette Valley  Willamette Valley          Pinot Noir
4             Bandol                NaN  Provence red blend

                   winery
0                   Heitz
1  Bodega Carmen Rodríguez
2                Macauley
3                   Ponzi
4    Domaine de la Bégude
```

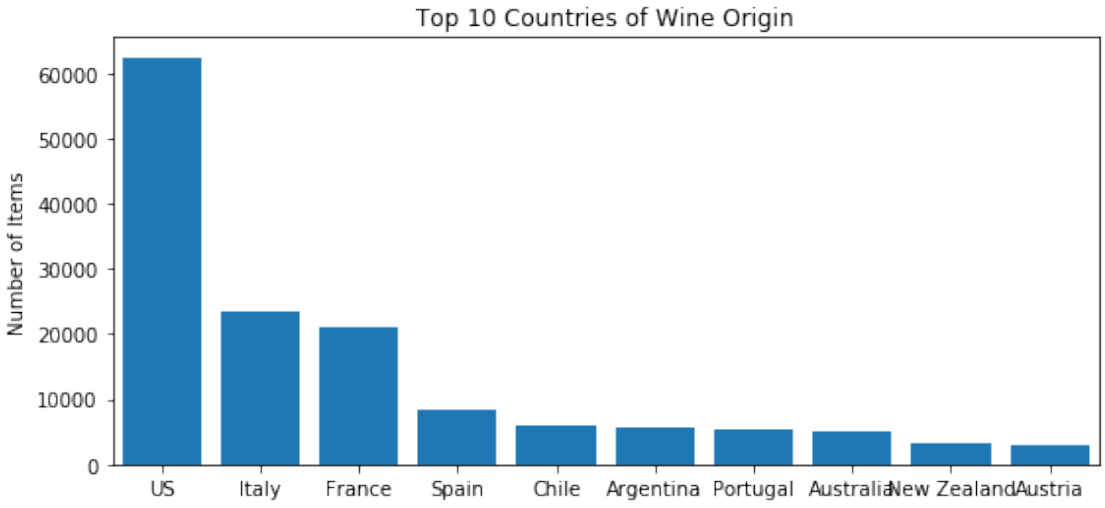
0.2 Descriptive statistics: Wine Items by Country

```
In [7]: #Top countries of origins
        Countries_Description = Wine_Review[['description','country']].groupby('country').coun
        Countries_Description = Countries_Description.sort_values(by=['description'], ascendin
        Countries_Description.columns = [['country', 'number']]
        print(Countries_Description[:10])
```

```
        country number
44           US  62397
22        Italy  23478
15       France  21098
40        Spain   8268
8         Chile   5816
1     Argentina   5631
33     Portugal   5322
2     Australia   4957
```

```
32  New Zealand   3320
3       Austria   3057
```

In [8]:
```
#draw a histogram for the top 10 countries of origins


top_10_countries_of_origins = Countries_Description[:10]

Countries = top_10_countries_of_origins.iloc[:,0].tolist()


y_pos = np.arange(len(Countries))

No_wines = top_10_countries_of_origins.iloc[:,1].tolist()
plt.figure(figsize=(9, 4))

plt.xlim([-0.5, 9.5])
plt.bar(y_pos, No_wines, align='center', alpha=1)


plt.xticks(y_pos, Countries)
plt.ylabel('Number of Items')
plt.title('Top 10 Countries of Wine Origin')

plt.show()
```



In [9]:
```
#Top countries of origins with the highest price
Countries_Price = Wine_Review[['country','price']].groupby('country').mean().reset_inde
Countries_Price = Countries_Price.sort_values(by=['price'], ascending = False)
print(Countries_Price[:10])
```

```
       country      price
45   US-France  50.000000
14     England  47.500000
15      France  45.619885
19     Hungary  44.204348
26  Luxembourg  40.666667
17     Germany  39.011078
22       Italy  37.547913
7       Canada  34.628866
44          US  33.653808
21      Israel  31.304918
```

In [10]:
```
#Top countries of origins with the highest price deviation
Countries_Price_deviation = Wine_Review[['country','price']].groupby('country').std()
Countries_Price_deviation = Countries_Price_deviation.sort_values(by=['price'], ascend
print(Countries_Price_deviation[:10])
```

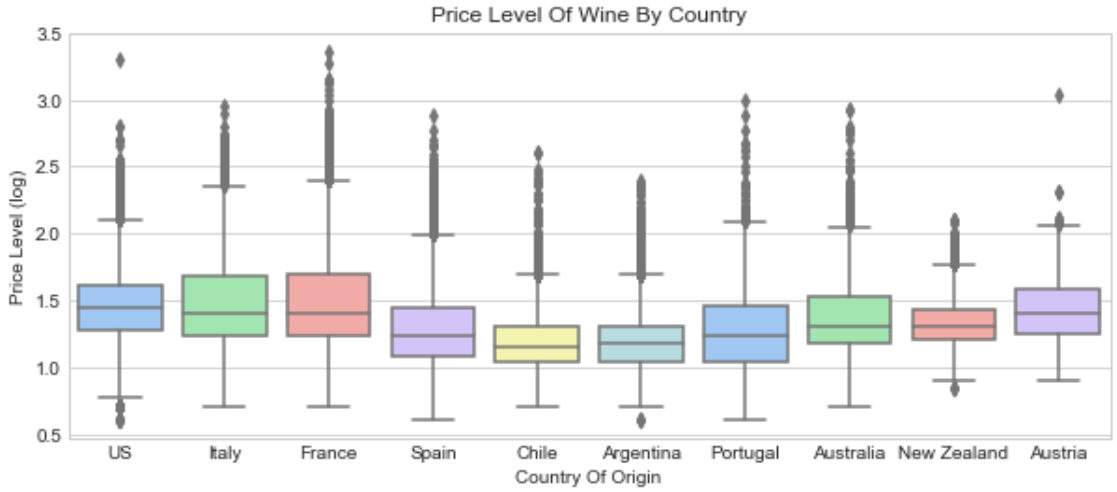
```
      country      price
15     France  69.697060
19    Hungary  66.264502
17    Germany  56.857128
2   Australia  39.008512
22      Italy  37.067869
33   Portugal  35.242873
40      Spain  33.861666
34    Romania  28.845571
3     Austria  28.540861
44         US  24.891343
```

In [11]:
```
# Draw a boxplot of Country of Wine Origin and Price.



Wine_Review_top_10 = Wine_Review.loc[(Wine_Review['country'] == 'US') | (Wine_Review[
                                     (Wine_Review['country'] == 'France') | (Wine_Revi
                                     (Wine_Review['country'] == 'Chile') | (Wine_Revie
                                     (Wine_Review['country'] == 'Argentina') | (Wine_
                                     (Wine_Review['country'] == 'Austria') | (Wine_Rev

Wine_Review_top_10 = Wine_Review_top_10[['country', 'price']]
Wine_Review_top_10.columns = ['Country Of Origin', 'Price']

# The price deviates a lot from the median, so I used log10 to normalize it.
Wine_Review_top_10['Price Level (log)'] = np.log10(Wine_Review_top_10['Price'])
```

```
sns.set_style("whitegrid")
```

```
fig, ax = plt.subplots(figsize=(10,4))
boxplot = sns.boxplot(x='Country Of Origin', y='Price Level (log)', data = Wine_Revie
boxplot.set_title('Price Level Of Wine By Country')
```

Out[11]: Text(0.5,1,'Price Level Of Wine By Country')



0.3 Vive la France!

After exploratory analysis of wine in the world. We come to the old world, France, and check some characteristic of French wine.

In [12]:
```
#Filter French wine from the data set.
French_Wine_Review = Wine_Review.loc[Wine_Review['country'] == "France"]
print(French_Wine_Review.shape)
#List unique values in the df['name'] column
```

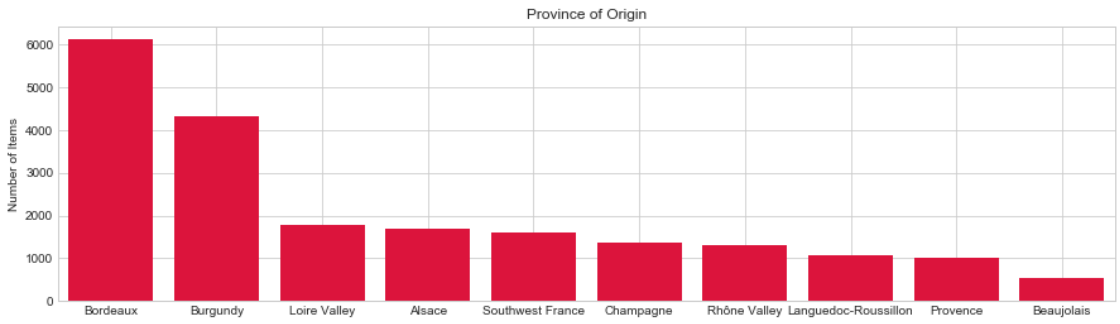
```
(21098, 11)
```

In [13]:
```
#province as origins
French_Wine_Province = French_Wine_Review[['province','description']].groupby('provin
French_Wine_Province = French_Wine_Province.sort_values(by=['description'], ascending
print(French_Wine_Province)
```

```
                province  description
2              Bordeaux         6111
3              Burgundy         4308
7          Loire Valley         1786
0                Alsace         1680
10     Southwest France         1601
4             Champagne         1370
9          Rhône Valley         1318
6   Languedoc-Roussillon         1082
8              Provence         1021
1            Beaujolais          532
5          France Other          289
```

In [14]: *#draw a histogram for the 10 province of origins*

```
Province_Of_Origins = French_Wine_Province[:10]

Provinces = Province_Of_Origins.iloc[:,0].tolist()


y_pos = np.arange(len(Provinces))

No_wines = Province_Of_Origins.iloc[:,1].tolist()
plt.figure(figsize=(15, 4))

plt.xlim([-0.5, 9.5])
plt.bar(y_pos, No_wines, align='center', color = 'crimson', alpha=1)


plt.xticks(y_pos, Provinces)
plt.ylabel('Number of Items')
plt.title('Province of Origin')

plt.show()
```



In [15]:
```
#French province of origins with the highest price
French_Province_Price = French_Wine_Review[['province','price']].groupby('province').m
French_Province_Price = French_Province_Price.sort_values(by=['price'], ascending = Fa
print(French_Province_Price[:10])
```

```
                province      price
4              Champagne  93.412305
3               Burgundy  70.602633
9           Rhône Valley  49.832656
2               Bordeaux  42.601956
0                 Alsace  31.876380
7           Loire Valley  27.071891
8               Provence  23.442029
10      Southwest France  22.879363
6   Languedoc-Roussillon  22.015640
1              Beaujolais  17.267327
```

In [16]:
```
#French province of origins with the highest price deviation
French_Province_Price_deviation = French_Wine_Review[['province','price']].groupby('pr
French_Province_Price_deviation = French_Province_Price_deviation.sort_values(by=['pri
print(French_Province_Price_deviation[:10])
```

```
                province       price
4              Champagne  102.510714
2               Bordeaux   96.293838
3               Burgundy   78.317744
9           Rhône Valley   61.401223
6   Languedoc-Roussillon   33.541045
0                 Alsace   22.149001
10      Southwest France   20.836563
7           Loire Valley   20.538205
8               Provence   13.140126
5           France Other    9.235507
```

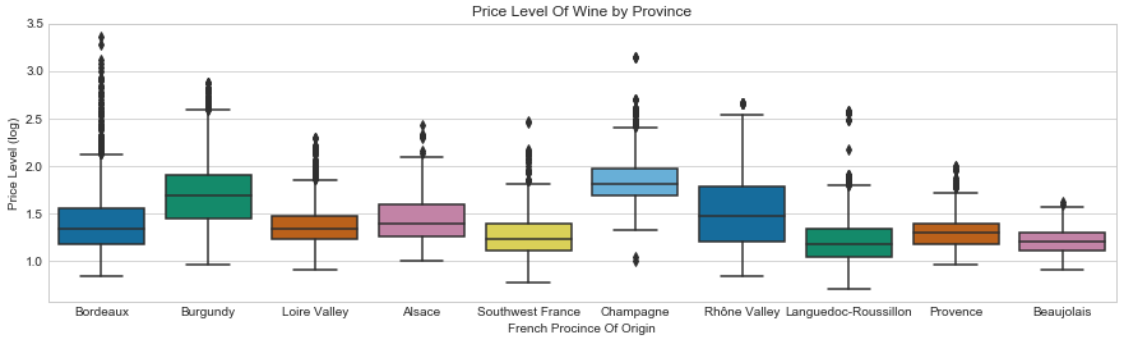
In [17]:
```
# Draw a boxplot of Province of Wine Origin and Price.


French_Wine_Province = French_Wine_Review.loc[(French_Wine_Review['province'] == 'Bord
                                              (French_Wine_Review['province'] == 'Loire Valley
                                              (French_Wine_Review['province'] == 'Southwest Fra
                                              (French_Wine_Review['province'] == 'Rhône Valley
                                              (French_Wine_Review['province'] == 'Provence') |

French_Wine_Province = French_Wine_Province[['province', 'price']]
French_Wine_Province.columns = ['French Procince Of Origin', 'Price']

#to overcome the issue of large price dispersion, I used the log of the price instead
```

```
French_Wine_Province['Price Level (log)'] = np.log10(French_Wine_Province['Price'])
```

```
sns.set_style("whitegrid")
```

```
fig, ax = plt.subplots(figsize=(15,4))
boxplot = sns.boxplot(x='French Procince Of Origin', y='Price Level (log)', data = Fre
boxplot.set_title('Price Level Of Wine by Province')
```

Out[17]: Text(0.5,1,'Price Level Of Wine by Province')



0.4 French Wine Wordcloud

In [18]:
```
#download nltk punkt package to tokenize English reviews with puctuation removed
nltk.download("punkt")
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/halfmoonliu/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[18]: True

In [19]:
```
#download nltk stopwords package
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/halfmoonliu/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[19]: True

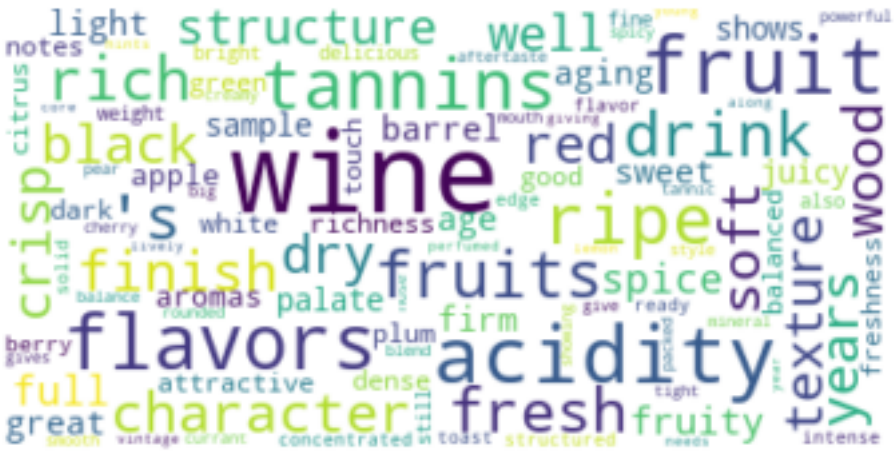
```
In [20]: useless_words = nltk.corpus.stopwords.words("english") + list(string.punctuation)

In [21]: French_Wine_Review_only = French_Wine_Review.loc[French_Wine_Review['country'] == "Fra
         print(type(French_Wine_Review_only))
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
In [22]: All_French_Review_words = list()

         for i in range(len(French_Wine_Review_only)):
             word_list = nltk.word_tokenize(French_Wine_Review_only.iloc[i]['description'])
             for j in range(len(word_list)):
                 lower_word = word_list[j].lower()
                 if lower_word not in useless_words:
                     All_French_Review_words.append(lower_word)


         print(len(All_French_Review_words))
```

```
460555
```

```
In [23]: # Create and generate a word cloud image:
         word_could_dict=Counter(All_French_Review_words)
         wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").gener
         plt.figure()
         plt.imshow(wordcloud, interpolation="bilinear")

         plt.axis("off")#turn off grids

         plt.show()
         wordcloud.to_file("French_Wine_Review.png")
```



```
Out[23]: <wordcloud.wordcloud.WordCloud at 0x1a1b9ca8d0>

In [24]: #Below is an example for a single wine description

         Review_Example = French_Wine_Review['description'].iloc[[0,1]]

In [25]: Review_Example = "".join(Review_Example)

In [26]: print(Review_Example)

This is the top wine from La Bégude, named after the highest point in the vineyard at 1200 feet


In [27]: #The tokenized word list of the discription
         Review_Example = nltk.word_tokenize(Review_Example)
         print(Review_Example)

['This', 'is', 'the', 'top', 'wine', 'from', 'La', 'Bégude', ',', 'named', 'after', 'the', 'hig
```

0.5 The Aroma Wheel Paradigm

To filter out words irrelavant to the taste, or aroma of wine, I used a modified version (https://winefolly.com/) of the aroma wheel (Noble et al., 1987) to analyze the "taste" of French wine.

```
In [28]: #Create a list of words describing the aroma of wine.

         Aroma_Wheel = [
         #flower
         'iris', 'peony', 'elderflower', 'acacia', 'lilac', 'jasmine', 'honeysuckle', 'violet'
         'potpourri', 'hibiscus',
         #citrus
         'lime', 'lemon', 'grapefruit', 'orange', 'marmalade',
         #tree fruit
         'quince', 'apple', 'pear', 'nectarine', 'peach', 'apricot', 'persimmon',
         #tropical fruit
         'pineapple','mango', 'guava', 'passion fruit', 'lychee', 'bubblegum',
         #Red Fruit
         'cranberry', 'red plum', 'pomegranate', 'sour cherry', 'strawberry', 'cherry', 'raspbe
         #Black Fruit
         'boysenberry', 'black currant', 'black cherry', 'plum', 'blackberry', 'blueberry', 'ol
         #Dried Fruit
         'raisin', 'fig', 'date', 'fruitcake',
         #noble rot
         'beeswax', 'ginger', 'honey',
```

```
#Spice
'white pepper', 'red pepper', 'black pepper', 'cinnamon', 'anise'
#5-Spice
'fennel', 'eucalyptus', 'mint', 'thyme',
#Vegetable
'grass', 'tomato leaf', 'gooseberry', 'bell pepper', 'jalapeño', 'bitter almond','toma
'black tea',
#earth
'clay pot', 'slate', 'wet gravel', 'potting soil', 'red beet', 'volcanic rocks', 'petr
#Secondary Aromas
#microbial
'butter', 'cream', 'sourdough', 'lager', 'truffle', 'mushroom',
#Tertiary Aromas
'oak aging', 'vanilla', 'coconut', 'baking spices', 'cigar box', 'smoke', 'dill',
#General Aging
'dried fruit', 'nutty flavors', 'tobacco', 'coffee', 'cocoa', 'leather',
#Faults & Other
#cork taint
'musty cardboard', 'wet Dog'
#Sulfides & Mercaptans
'cured meat', 'boiled eggs', 'burnt rubber', 'lit match', 'garlic', 'onion', 'cat pee
#brettanomyces
'black cardamon','band-aid', 'sweaty leather saddle', 'horse manure',
#madeirized
'toffee', 'stewed fruit',
#volatile acidity
'vinegar'
]
```

0.6 The Bordeau Taste

First, let's use the most common wine in the data set, Bordeau, and see the words used in the description.

In [29]:
```
#Select description of wine coming from Bordeaux.
Bordeaux_Review = French_Wine_Review.loc[French_Wine_Review['province'] == "Bordeaux"]
print(Bordeaux_Review.shape)
```

```
(6111, 1)
```

In [30]:
```
#Create a list of lowercase words describing Bordeaux wine

All_Bordeaux_words = list()

for i in range(len(Bordeaux_Review)):
    word_list = nltk.word_tokenize(Bordeaux_Review.iloc[i]['description'])
    for j in range(len(word_list)):
        lower_word = word_list[j].lower()
```

```
                All_Bordeaux_words.append(lower_word)


        print(len(All_Bordeaux_words))
```

```
242448
```

In [31]:
```
        #Filter out stopwords and punctuation.
        filtered_Bordeaux_words = [word for word in All_Bordeaux_words if not word in useless
        print(len((filtered_Bordeaux_words)))
```

```
123336
```

In [32]:
```
        from collections import Counter

        Bordeaux_word_counter = Counter(filtered_Bordeaux_words)
```

In [33]:
```
        #Count the most common words describing Bordeaux.
        Bordeaux_most_common_words = Bordeaux_word_counter.most_common()[:20]
        print(Bordeaux_most_common_words)
```

```
[('wine', 6611), ('tannins', 3235), ('fruit', 3047), ('acidity', 2248), ('ripe', 2177), ('barre
```

In [34]:
```
        #Filtering out the Aroma words in Bordeaux wine description.
        Bordeaux_Aroma_Wheel_Words = [word for word in filtered_Bordeaux_words if word in Aro
        print(len(Bordeaux_Aroma_Wheel_Words))
```

```
2555
```

In [35]:
```
        #Create a word counter for Aroma words describing Bordeaux wine.
        Bordeaux_Aroma_Wheel_Counter = Counter(Bordeaux_Aroma_Wheel_Words)
```

In [36]:
```
        Bordeaux_most_common_aroma = Bordeaux_Aroma_Wheel_Counter.most_common()[:20]
        print(Bordeaux_most_common_aroma)
```

```
[('plum', 429), ('blackberry', 419), ('grapefruit', 127), ('pear', 121), ('lemon', 111), ('hon
```

In [37]:
```
        #Use Word Cloud to present the results. The larger the word, the more frequent it is


        wordcloud_bordeaux_Aroma = WordCloud(max_font_size=50, max_words=100, background_colo
```

```
plt.figure()
plt.imshow(wordcloud_bordeaux_Aroma, interpolation="bilinear")

plt.axis("off")
plt.show()
wordcloud_bordeaux_Aroma.to_file("Bordeaux_Wine_Aroma.png")
```


```
Out[37]: <wordcloud.wordcloud.WordCloud at 0x1a28778be0>
```

0.7 The Burgundy taste

Then, I applied the same methods to present the tsate of Burgundy wine.

```
In [38]: #Select description of wine coming from Burgundy.
         Burgundy_Review = French_Wine_Review.loc[French_Wine_Review['province'] == "Burgundy"]
         print(Burgundy_Review.shape)
```

```
(4308, 1)
```

```
In [39]: #Create a list of lowercase words describing Burgundy wine
         All_Burgundy_words = list()

         for i in range(len(Burgundy_Review)):
             word_list = nltk.word_tokenize(Burgundy_Review.iloc[i]['description'])
             for j in range(len(word_list)):
                 lower_word = word_list[j].lower()
                 All_Burgundy_words.append(lower_word)


         print(len(All_Burgundy_words))
```

```
185593
```

```
In [40]: #Filter out stopwords and punctuation.
         filtered_Burgundy_words = [word for word in All_Burgundy_words if not word in useless_
         print(len((filtered_Burgundy_words)))
```

```
93232
```

```
In [41]: #Count the most common words describing Burgundy.
         Burgundy_word_counter = Counter(filtered_Burgundy_words)
         Burgundy_most_common_words = Burgundy_word_counter.most_common()[:20]
         print(Burgundy_most_common_words)
```

```
[('wine', 4469), ('acidity', 2221), ('fruit', 1894), ('flavors', 1737), ('ripe', 1594), ('fruit
```

```
In [42]: #Filtering out the Aroma words in Burgundy wine description.
         Burgundy_Aroma_Wheel_Words = [word for word in filtered_Burgundy_words if word in Aro
```

```
In [43]: #Create a word counter for Aroma words describing Burgundy wine.
         Burgundy_Aroma_Wheel_Counter = Counter(Burgundy_Aroma_Wheel_Words)
```

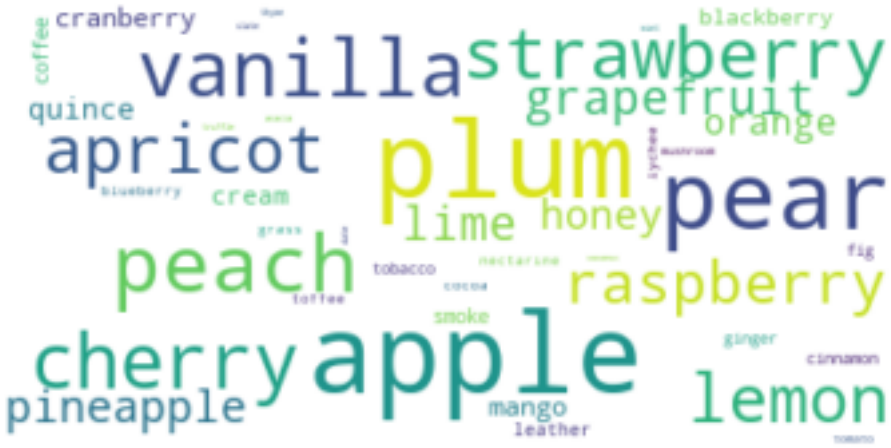
```
In [44]: Burgundy_most_common_aroma = Burgundy_Aroma_Wheel_Counter.most_common()[:20]
         print(Burgundy_most_common_aroma)
```

```
[('apple', 397), ('plum', 367), ('pear', 295), ('peach', 201), ('vanilla', 192), ('cherry', 18
```

```
In [45]: #Use Word Cloud to present the results. The larger the word, the more frequent it is
```

```
         wordcloud_Burgundy_Aroma = WordCloud(max_font_size=50, max_words=100, background_colo
```

```
         plt.figure()
         plt.imshow(wordcloud_Burgundy_Aroma, interpolation="bilinear")

         plt.axis("off")
         plt.show()
         wordcloud_Burgundy_Aroma.to_file("Burgundy_Wine_Aroma.png")
```



Out[45]: <wordcloud.wordcloud.WordCloud at 0x1a29c32b38>

0.8 The Aroma-Distance Based Recommendation.

Here, I tried to use the persentage of aroma words used to describe the wine from a given province as parameters to determine the "distance of taste", or Aroma-Distance between provinces of origin.

In [46]:
```
French_Province = ['Champagne', 'Bordeaux', 'Burgundy', 'Rhône Valley', 'Languedoc-Rou
                   'Southwest France', 'Loire Valley', 'Provence', 'Beaujolais']
```

In [47]:
```
# First, I created a dictionary of province-list pairs, with all the aroma words used
# to describe the wine coming from the province.

Province_Dictionary = {}
for province_index in range(len(French_Province)):
    df = French_Wine_Review.loc[French_Wine_Review['province'] == French_Province[prov
    list_temp = list()

    for i in range(len(df)):
        token_list_list = nltk.word_tokenize(df.iloc[i]['description'])
        for j in range(len(token_list_list)):
            lower_word = token_list_list[j].lower()
            list_temp.append(lower_word)

    filtered_list_temp = [word for word in list_temp if not word in useless_words]
    Aroma_Wheel_list_temp = [word for word in filtered_list_temp if word in Aroma_Whee
    Province_Dictionary[French_Province[province_index]] = Aroma_Wheel_list_temp
```

In [48]:
```
#print the first 10 aroma words in the list describing Champagne.
print(Province_Dictionary['Champagne'][:10])
```

```
['raspberry', 'orange', 'apple', 'apple', 'apple', 'peach', 'apple', 'apple', 'pear', 'honey']
```

In [49]:
```
#Create a dictionary for all provinces for aroma word counts


Aroma_template = {} #Create a template dictionary for all aromas in the aroma wheel.
for aroma in Aroma_Wheel:
    Aroma_template[aroma] =0


Province_Aroma_Count = {}

for province in French_Province:
    Aroma_Dic = Aroma_template.copy()

    #Count the times a given aroma appeared in all reviews wine originated from a pro
    for review_word_aroma in Province_Dictionary[province]:
        Aroma_Dic[review_word_aroma] +=1



    Province_Aroma_Count[province] = Aroma_Dic


#Check the results with previous results using the counter method.
print(Province_Aroma_Count['Bordeaux']['plum'])
```

```
429
```

In [50]:
```
#Create the aroma score, percentage of aroma used to describe the wine.
#i.e. word count of an aroma / sum of all wordcounts in the aroma wheel for every pro
#Thus, the sum of all scores should be 1.0

Province_Aroma_Score = {}

for province in French_Province:
    Dic_Temp = {}
    for aroma in Aroma_Wheel:
        Dic_Temp[aroma] = Province_Aroma_Count[province][aroma] / len(Province_Dictio

    Province_Aroma_Score[province] = Dic_Temp


Champagne_total = 0
for aroma in Province_Aroma_Score['Champagne']:
    Champagne_total += Province_Aroma_Score['Champagne'][aroma]
print(Champagne_total)
```

```
1.0
```

In [51]:
```
# Calculating the distance between Bordeau and other province.

Bordeaux_Neighbor_list = ['Champagne', 'Burgundy', 'Rhône Valley', 'Languedoc-Roussil
                          'Southwest France', 'Loire Valley', 'Provence', 'Beaujolais']

Distance_From_Bordeaux = {}

for neighbor in Bordeaux_Neighbor_list:
    neighbor_distance = 0
    for aroma in Aroma_Wheel:
        distance_aroma = 0
        distance_aroma += ((Province_Aroma_Score['Bordeaux'][aroma]-Province_Aroma_Sc
        neighbor_distance += distance_aroma
    Distance_From_Bordeaux[neighbor] = neighbor_distance

print(Distance_From_Bordeaux)
```

```
{'Champagne': 1.0719933303333786, 'Burgundy': 0.7944740720819754, 'Rhône Valley': 0.98680522752
```

In [52]:
```
# Calculating the distance between Burgundy and other province.

Burgundy_Neighbor_list = ['Champagne', 'Bordeaux', 'Rhône Valley', 'Languedoc-Roussil
                          'Southwest France', 'Loire Valley', 'Provence', 'Beaujolais']

Distance_From_Burgundy = {}

for neighbor in Burgundy_Neighbor_list:
    neighbor_distance = 0
    for aroma in Aroma_Wheel:
        distance_aroma = 0
        distance_aroma += ((Province_Aroma_Score['Burgundy'][aroma]-Province_Aroma_Sc
        neighbor_distance += distance_aroma
    Distance_From_Burgundy[neighbor] = neighbor_distance

print(Distance_From_Burgundy)
```

```
{'Champagne': 0.6282821940711516, 'Bordeaux': 0.7944740720819754, 'Rhône Valley': 0.9190934441
```

In [53]:
```
#Create a grand aroma distance table for all wine.

French_Province = ['Champagne', 'Bordeaux', 'Burgundy', 'Rhône Valley', 'Languedoc-Rou
                   'Southwest France', 'Loire Valley', 'Provence', 'Beaujolais']

Aroma_Distance_Matrix = dict()
```

```
        for province in French_Province:
            Aroma_Distance_Matrix[province] = dict()
            Neighbor_list = French_Province.copy()
            Neighbor_list.remove(province)
            for neighbor in Neighbor_list:
                neighbor_distance = 0
                for aroma in Aroma_Wheel:
                    distance_aroma = 0
                    distance_aroma += ((Province_Aroma_Score[province][aroma]-Province_Aroma_S
                    neighbor_distance += distance_aroma
                    Aroma_Distance_Matrix[province][neighbor] = neighbor_distance

        #print only the Burgundy line of Aroma Distance Matrix
        print(Aroma_Distance_Matrix['Burgundy'])
```

```
{'Champagne': 0.6282821940711516, 'Bordeaux': 0.7944740720819754, 'Rhône Valley': 0.919093444
```

```
In [54]: x = ['Champagne', 'Bordeaux', 'Burgundy', 'Rhône Valley', 'Languedoc-Roussillon', 'Als
                        'Southwest France', 'Loire Valley', 'Provence', 'Beaujolais']
         y = x.copy()
         y.remove('Champagne')
         print(y)
```

```
['Bordeaux', 'Burgundy', 'Rhône Valley', 'Languedoc-Roussillon', 'Alsace', 'Southwest France',
```

0.9 Wine Description Classification Methods Comparison.

Finally, I will compare 2 classification methods, Naive Bayes and K nearest Neighbors to classify wine descriptions.

```
In [55]: from nltk.classify import NaiveBayesClassifier
```

```
In [56]: French_Wine_Province_Description = French_Wine_Review[['province','description']]
         print(French_Wine_Review.shape)
```

```
(21098, 11)
```

```
In [57]: print(French_Wine_Province_Description.shape)
         print(French_Wine_Province_Description.head(n=5))
```

```
(21098, 2)
            province                                        description
4           Provence  This is the top wine from La Bégude, named aft...
13  Southwest France  This wine is in peak condition. The tannins an...
18  Southwest France  Coming from a seven-acre vineyard named after ...
33      France Other  Pale in color, this is nutty in character, wit...
36      Rhône Valley  Gingery spice notes accent fresh pear and melo...
```

0.10 Naïve Baiyes

In Naïve Bayes classification, every item of wine is classified based on 1.) the amount of items wine origin in the dataset(e.g. Bordeaux should be of the highest probability because there more Bordeaux items than any other province) and 2) the probability of a word appeared in the descrption of wine from a given province (e.g. apple is more likely to appear in the description of Burgundy wine than wine from Bordeaux). In Naïve Bayes classification paradigm, features are assumed indepedent.

In [58]:
```
#Choose 70% of data in the dataset as training data.

Random_list = random.sample(range(len(French_Wine_Province_Description)), len(French_

train_size = 0.7
```

In [59]:
```
#province as origins
test = French_Wine_Province_Description[['province','description']].groupby('province
test = test.sort_values(by=['description'], ascending = False)
print(test)
```

```
              province  description
2              Bordeaux         6111
3              Burgundy         4308
7          Loire Valley         1786
0                Alsace         1680
10     Southwest France         1601
4             Champagne         1370
9          Rhône Valley         1318
6   Languedoc-Roussillon         1082
8              Provence         1021
1            Beaujolais          532
5          France Other          289
```

In [60]:
```
#build a bag-of-word method to represent word appearing in the description of wine,
#excluding stop words and punctuations.
def build_bag_of_words_features_filtered(words):
    work_bag_dict = {}
    for word in words:
        if word not in useless_words:

            lower_word = word.lower()
            work_bag_dict[lower_word] = 1
    return work_bag_dict
```

In [61]:
```
#Create train_data list consisting of (word-bag, province) tuples.
train_data = list()
for sample in range(int(train_size*len(French_Wine_Province_Description))):
    description = French_Wine_Province_Description.iloc[sample][1]
```

```
        description = nltk.word_tokenize(description)
        description_dic = build_bag_of_words_features_filtered(description)
        tag = French_Wine_Province_Description.iloc[sample][0]

        train_data.append((description_dic, tag))
```

In [62]:
```
#an Example of tokenized wine description and tag of wine origin.
print(len(train_data))
```

```
14768
```

In [63]:
```
#Create a Naïve Bayes classifier.
Province_classifier = NaiveBayesClassifier.train(train_data)
```

In [64]:
```
# the train data prediction accuracy.
nltk.classify.util.accuracy(Province_classifier, train_data)*100
```

Out[64]:
```
78.00650054171182
```

In [65]:
```
#Get test data set.
test_size = 1-train_size
test_data = list()
for sample in range(int(test_size*len(French_Wine_Province_Description))):
    description = French_Wine_Province_Description.iloc[-sample][1]
    description = nltk.word_tokenize(description)
    description_dic = build_bag_of_words_features_filtered(description)
    tag = French_Wine_Province_Description.iloc[-sample][0]

    test_data.append((description_dic, tag))
print(len(test_data))
```

```
6329
```

In [66]:
```
nltk.classify.util.accuracy(Province_classifier, test_data)*100
```

Out[66]:
```
70.09006162110919
```

0.11 K-Nearest Neighbors

Using k-Nearest Neighbor classification, I first calculated the distance between description of wine(i.e. if more words appeared in both description A and B then A and C, than wine A and B are closer in distance). Then, the algorithm identifies the k nearest neighbors and assign the tag of the majority of "neighbors" to the item being classified.

In [73]:
```
#Create a list of (description word-bag, province of origin) tuple for French wine.
French_Wine_Descr_tag = list()

for sample in range(len(French_Wine_Province_Description)):
```

```
            description = French_Wine_Province_Description.iloc[sample][1]
            description = nltk.word_tokenize(description)
            description_dic = build_bag_of_words_features_filtered(description)
            tag = French_Wine_Province_Description.iloc[-sample][0]

            French_Wine_Descr_tag.append((description_dic, tag))
        print(French_Wine_Descr_tag[0])
```

```
({'this': 1, 'top': 1, 'wine': 1, 'la': 1, 'bégude': 1, 'named': 1, 'highest': 1, 'point': 1,
```

In [74]:
```
#Create a dictionary where the keys are provinces of origin and the values are word b
#wine descriptions used to describe the wine coming from the province.

Province_Dictionary_tag = {}

French_Province_KNN  = French_Province.copy()


for province in French_Province_KNN:
    Province_Dictionary_tag[province] = list()


for tag in range(len(French_Wine_Descr_tag)):
    try:
        Province_Dictionary_tag[French_Wine_Descr_tag[tag][1]].append(French_Wine_Des
    except KeyError:
        pass

    # an example of Dictionary of word bags of wine from Champagne.
print(Province_Dictionary_tag['Bordeaux'][1])
```

```
{'lightly': 1, 'structured': 1, 'balanced': 1, 'ripe': 1, 'wine': 1, 'it': 1, 'profited': 1, '
```

In [69]:
```
#Choose equally large sample for every province of wine orgins.
Sample_list_Province = dict()

for province in French_Province_KNN:
    #choose 500 because there are a bit more than 500 Beayjolais products.
    random_list = random.sample(range(len(Province_Dictionary_tag[province])), 500)
    Sample_list_Province[province] = random_list
print(len(Sample_list_Province['Bordeaux']))
```

```
500
```

In [70]:
```
# Create Train Dataset
sample_data_KNN = list()
```

```
        for province in French_Province_KNN:
            for sample in Sample_list_Province[province]:
                sample_data_KNN.append((Province_Dictionary_tag[province][sample], province))

        print(len(sample_data_KNN))
```

```
5000
```

```
In [71]: k = 10
```

```
        example = sample_data_KNN[0]

        data = sample_data_KNN

        def KNN_Result(word_bag_tag, data):

            distance_list = list()
            for neighbor_index in range(len(data)):
                distance = 0
                for word in word_bag_tag[0]:
                    #adddistance if a word in the description does not appear in nrighbor's d
                    if word not in data[neighbor_index][0]:
                        distance +=1
                distance_list.append((neighbor_index, distance))



            def takeSecond(elem):
                return elem[1]

            distance_list.sort(key = takeSecond)


            nearest_neighbors_tuple = distance_list[:k]

            neighbor_dict = dict()



            for neighbor_tuple in nearest_neighbors_tuple:
                if data[neighbor_tuple[0]][1] not in neighbor_dict:
                    neighbor_dict[data[neighbor_tuple[0]][1]] = 1
```

```
                else:
                    neighbor_dict[data[neighbor_tuple[0]][1]] += 1
            return max(neighbor_dict)

        print(KNN_Result(example, data))
```

```
Rhône Valley
```

```
In [72]: # KNN Accuracy

        hit = 0
        miss = 0

        for i in sample_data_KNN:
            if KNN_Result(i, sample_data_KNN) == i[1]:
                hit +=1
            else:
                miss +=1
        print(hit, miss, hit/(hit+miss))
```

```
1031 3969 0.2062
```