

WHICH GRAPE JUICE IS WORTH THE SQUEEZE?

Jovany Candelario
Jessica Fernandez
Rawaf Rawaf
Brice Wilson
Jeremy Brzozowski

TEAM FIVE LOOP

Research Questions

1. Which countries are responsible for producing wine ?
2. Is wine rating correlated with wine price?
3. Do certain years have a higher rating relative to other years?
4. Do certain countries have more variety vs other countries?
5. What is the difference between an expensive winery vs a least expensive winery

Resources

- KAGGLE.COM - WINE REVIEWS

<https://www.kaggle.com/zynicide/wine-reviews>

The data was scraped from **WineEnthusiast** during the week of June 15th, 2017.

- OpenCage Geocoding API

<https://opencagedata.com/api#intro>

Forward geocoding (text to lat/long) via open data sources to get geographic data for wineries.


- Open Weather API

<https://openweathermap.org/api>

Allows you to input a city and receive live weather updates

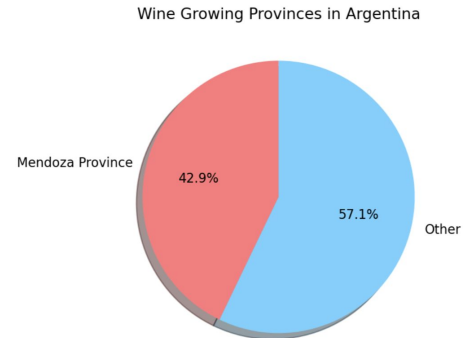
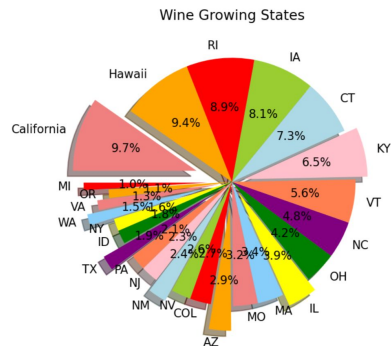
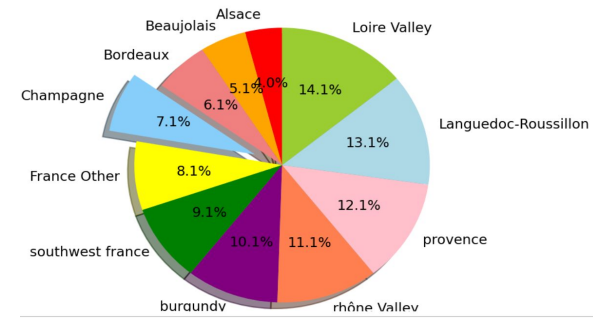
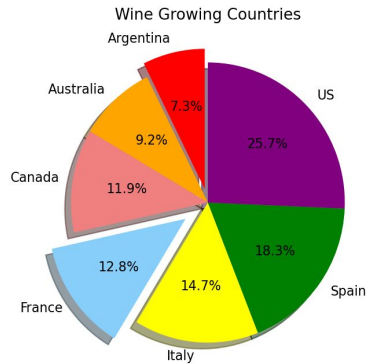
Wine Reviews

130k wine reviews with variety, location, winery, price, and description

 zackthoutt • updated 3 years ago (Version 4)

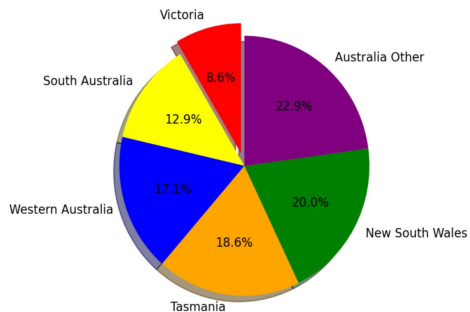
country	description	designation	points	price	province	region_1	region_2	title	variety	winery
US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian
US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks
Spain	Blackberry and raspberry aromas show a typical...	Ars In Vitro	87	15.0	Northern Spain	Navarra	NaN	Tandem 2011 Ars In Vitro Tempranillo-Merlot (N...	Tempranillo-Merlot	Tandem
Italy	Here's a bright, informal red that opens with ...	Belsito	87	16.0	Sicily & Sardinia	Vittoria	NaN	Terre di Giurfo 2013 Belsito Frappato (Vittoria)	Frappato	Terre di Giurfo
France	This has great depth of flavor with its fresh ...	Les Natures	87	27.0	Alsace	Alsace	NaN	Jean-Baptiste Adam 2012 Les Natures Pinot Gris...	Pinot Gris	Jean-Baptiste Adam
...
France	The granite soil of the Brand Grand Cru vineya...	Brand Grand Cru	90	57.0	Alsace	Alsace	NaN	Cave de Turckheim 2010 Brand Grand Cru Pinot G...	Pinot Gris	Cave de Turckheim
Italy	Blackberry, cassis, grilled herb and toasted a...	Sàgana Tenuta San Giacomo	90	40.0	Sicily & Sardinia	Sicilia	NaN	Cusumano 2012 Sàgana Tenuta San Giacomo Nero d...	Nero d'Avola	Cusumano
France	While it's rich, this beautiful dry wine also ...	Seppi Landmann Vallée Noble	90	28.0	Alsace	Alsace	NaN	Domaine Rieffé-Landmann 2013 Seppi Landmann Va...	Pinot Gris	Domaine Rieffé-Landmann

Countries that Grow Wine

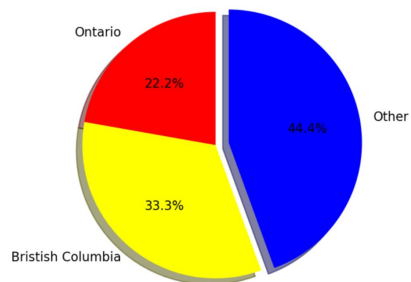


Countries that Grow Wine

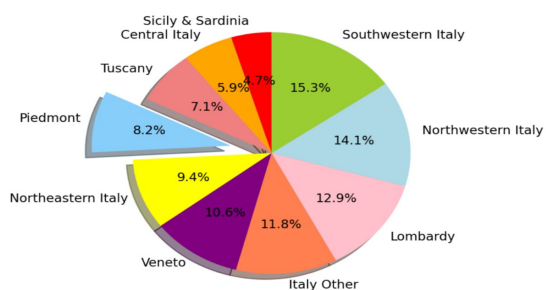
Wine Growing Provinces in Australia



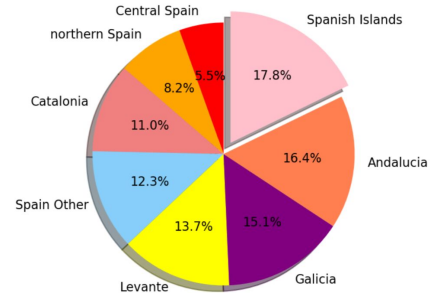
Wine Growing Provinces in Canada



Wine Growing Provinces in Italy



Wine Growing Provinces in Spain



Overall Relationship Between Countries - Jeremy



Overall wine relationship between countries:



Overall Wine Quality and Relationship Between Selected Countries.



Compilation of winemag data includes:

- 1) Countries: 7
- 2) Wineries: 10,030
- 3) Time Period: 2001 - 2016 (15 years)
- 4) Avg Winery Rating: 88.81 (scale of 100)
- 5) Number of Varieties: Tens of thousands

```
# Calculate the total number of unique countries
number_of_countries = clean_data_pd["country"].nunique()
print(f'The total number of countries reviewed: {number_of_countries:,}')
```

The total number of countries reviewed: 7

```
# Calculate the total number of unique wineries
number_of_wineries = clean_data_pd["winery"].nunique()
print(f'The total number of unique wineries reviewed: {number_of_wineries:,}')
```

The total number of unique wineries reviewed: 10,030

```
# Calculate the average winery rating over all years
avg_winery_rating = clean_data_pd["points"].mean()
print(f'The average winery rating of reviewed wineries (Yrs 2001 - 2016): {avg_winery_rating:.2f}')
```

The average winery rating of reviewed wineries (Yrs 2001 - 2016): 88.81

```
# How many varieties of wine does each country produce
variety_df = clean_data_pd[["country", "variety", "counter"]]
variety = variety_df.groupby('country')
variety_by_country_df = variety['variety'].count()
variety_by_country_df.sort_values(ascending=False, inplace=True)
variety_by_country_df.to_frame()
```

US	36558
Italy	11973
France	11782
Spain	5249
Argentina	2821
Australia	1616
Canada	176

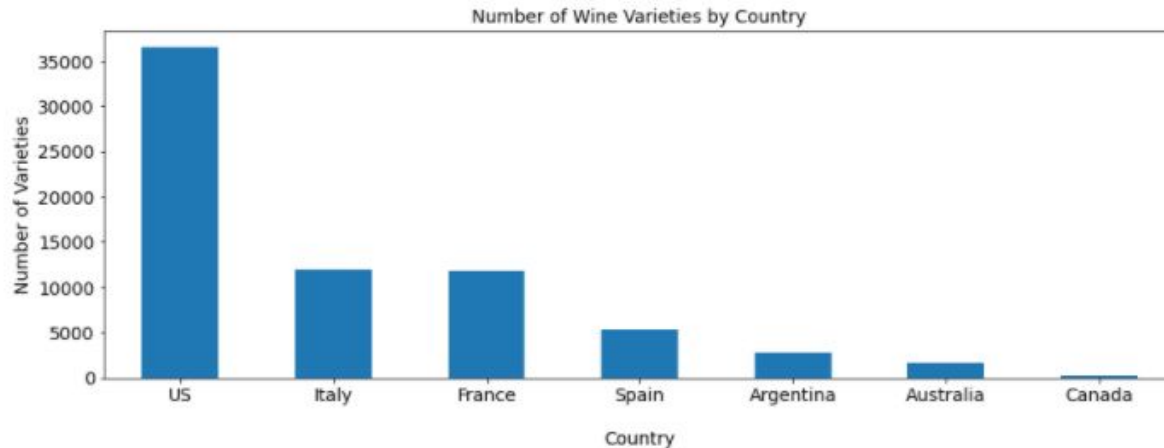
Data analysis examines:

- 1) Quality of wine by country (score based)
- 2) Diversity of wine by country
- 3) Most popular wine varieties by country
- 4) Wine quality by country over time (score based)

Number of Wine Varieties by Country



```
variety_by_country_df.plot(kind="bar", figsize=(15,5), fontsize=14, rot=0)
plt.title("Number of Wine Varieties by Country", fontsize=14)
plt.xlabel("\nCountry", fontsize=14)
plt.ylabel("Number of Varieties", fontsize=14)
|
```



US	36558
Italy	11973
France	11782
Spain	5249
Argentina	2821
Australia	1816
Canada	176

Most Popular Wines by Country



```
# Create a dataframe with a count of varieties by country
pop_variety_df = clean_data_pd[["country", "variety", "counter"]]
variety_df = pop_variety_df.groupby(['country', 'variety'], as_index=False)
country_variety_df = variety_df['counter'].sum()
country_variety_df.set_index('country', inplace=True)
country_variety_df.sort_values(by="country", ascending=False, inplace=True)
country_variety_df.reset_index(inplace=True)

# Create a blank dataframe used to store the most popular varieties by country
popular_variety_by_country_df = pd.DataFrame(columns=['country', 'variety', 'counter'])

j=0
country = ''
variety = ''
counter = 0

# Loop through existing variety by country df and extract the most
# popular wine variety by country. Write that value to a new df
for i in range(len(country_variety_df)-1):

    k = i+1

    if country_variety_df.loc[k,'country'] == country_variety_df.loc[i,'country']:
        if country_variety_df.loc[i,'counter'] > counter:
            country = country_variety_df.loc[i,'country']
            counter = country_variety_df.loc[i,'counter']
            variety = country_variety_df.loc[i,'variety']

    if country_variety_df.loc[k,'country'] != country_variety_df.loc[i,'country']:
        j+=1
        counter = 0

    popular_variety_by_country_df.loc[j, 'country'] = country
    popular_variety_by_country_df.loc[j, 'variety'] = variety
    popular_variety_by_country_df.loc[j, 'counter'] = counter

# show new df that includes most popular wine variety by country
popular_variety_by_country_df
```

To get most popular varieties by country:

1. Create a new dataframe using groupby and get the total number for each variety.
2. Create a blank dataframe.
3. Loop through groupby dataframe dataframe and find the most popular variety.
4. Insert the most popular variety into the new dataframe.

	country	variety	counter
0	US	Pinot Noir	7505
1	Spain	Tempranillo	1215
2	Italy	Red Blend	2404
3	France	Chardonnay	1598
4	Canada	Riesling	43
5	Australia	Shiraz	458
6	Argentina	Malbec	1133

Wine Rating by Country vs Overall Wine Rating (2001–2016)



```
# Calculate the average winery rating by country for all years
country = clean_data_pd.groupby('country')
avg_country_rating_df = country['points'].mean()

# Create 2 new columns and populate with the avg wine rating and the avg classification
avg_country_rating_df = avg_country_rating_df.to_frame()
avg_country_rating_df['avg_rating'] = avg_winery_rating
avg_country_rating_df['wine_rating'] = ""

def calc_wine_rating(row):
    if row['points'] < row['avg_rating']:
        return "Below Average Wine"
    elif row['points'] == row['avg_rating']:
        return "Average Wine"
    else:
        return "Above Average Wine"

avg_country_rating_df['wine_rating'] = avg_country_rating_df.apply(calc_wine_rating, axis=1)
avg_country_rating_df.sort_values(by='points', ascending = False, inplace = True)

avg_country_rating_df.rename(columns={'points':'Country Rating', 'avg_rating':'Overall Rating', \
                                     'wine_rating':'Wine Rating'}, inplace=True)
avg_country_rating_df
```

	Country Rating	Overall Rating	Wine Rating
country			
Canada	89.363636	88.806341	Above Average Wine
France	89.077831	88.806341	Above Average Wine
US	89.017671	88.806341	Above Average Wine
Australia	89.003713	88.806341	Above Average Wine
Italy	88.839055	88.806341	Above Average Wine
Spain	87.465994	88.806341	Below Average Wine
Argentina	87.141085	88.806341	Below Average Wine

Country Rating Compared to Overall Rating:

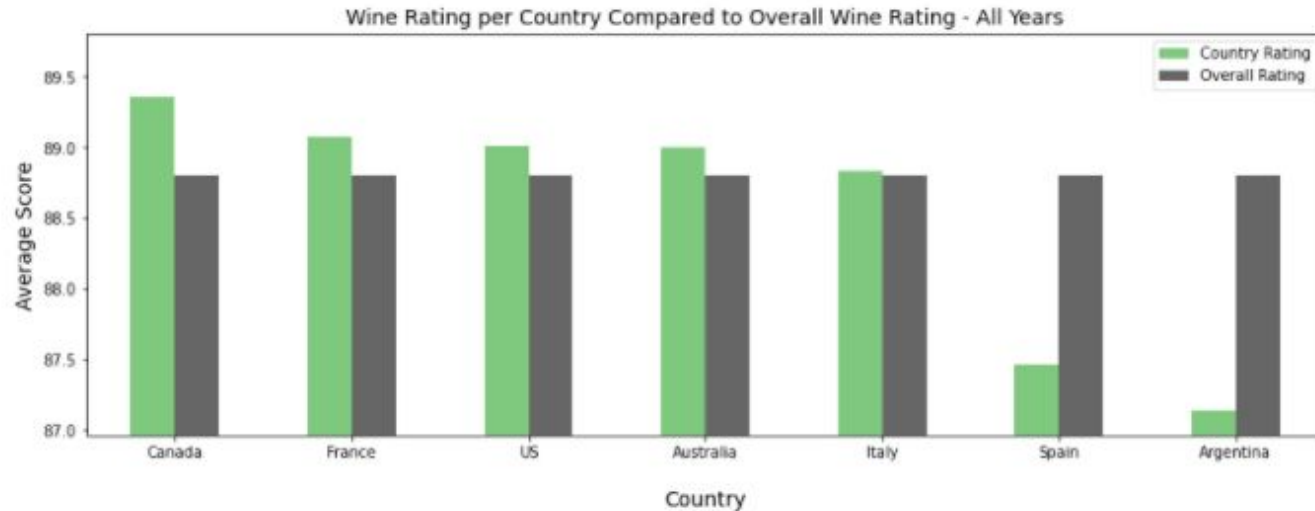
- 1) Each countries average wine rating is compared to the overall wine rating for all countries in the group.
- 2) Based on the score comparison each country is tagged as above or below average.

Wine Rating by Country vs Overall Wine Rating (2001-2016)



```
#Set upper and lower y-limits based on data set
lower_lim = avg_country_rating_df['Country Rating'].min() - (avg_country_rating_df['Country Rating'].min() * .002)
upper_lim = avg_country_rating_df['Country Rating'].max() + (avg_country_rating_df['Country Rating'].max() * .005)
# colormap = {avg_country_rating_df['points']: 'blue', avg_country_rating_df['avg_rating']: 'yellow'}

avg_country_rating_df.plot(kind="bar", figsize=(15,5), use_index=True, ylim=(lower_lim, upper_lim), colormap='Accent', rot=0)
plt.title("Wine Rating per Country Compared to Overall Wine Rating - All Years", fontsize=14)
plt.ylabel("Average Score", fontsize=14)
plt.xlabel("\nCountry", fontsize=14)
plt.legend(loc='best')
```



Wine Rating by Country Over Time



```
# Calculate the average wine rating by country for each year
country_year = clean_data_pd.groupby(['country', 'Wine Year'], as_index=False)
overall = clean_data_pd.groupby('Wine Year', as_index=False)
avg_country_year_rating_df = country_year['points'].mean()

# Calculate the average wine rating overall by year
avg_wine_rating_overall_df = overall['points'].mean()

# Create new columns for avg rating for all wines per year and overall wine rating
# avg_country_year_rating_df['avg_rating'] = ""
avg_country_year_rating_df['wine_rating'] = ""

# merge avg rating into the country wine rating df
avg_country_year_rating_df = avg_country_year_rating_df.merge(avg_wine_rating_overall_df, \
                                                             how="left", on="Wine Year", suffixes=('_country', '_overall'))

def calc_wine_year_rating(row):
    if row['points_country'] < row['points_overall']:
        return "Below Average Wine"
    elif row['points_country'] == row['points_overall']:
        return "Average Wine"
    else:
        return "Above Average Wine"

avg_country_year_rating_df['wine_rating'] = avg_country_year_rating_df.apply(calc_wine_year_rating, axis=1)

avg_country_year_rating_df = avg_country_year_rating_df[avg_country_year_rating_df['Wine Year'] != '2017']
avg_country_year_rating_df.head()
```

- 1) Calculate the wine rating for each country by year.
- 2) Compare each country to the average for that year.
- 3) Create the dataframe and then plot the results.

	country	Wine Year	points_country	wine_rating	points_overall
0	Argentina	2000	83.000000	Below Average Wine	87.919881
1	Argentina	2001	82.000000	Below Average Wine	88.461827
2	Argentina	2002	88.000000	Below Average Wine	88.442105
3	Argentina	2003	86.875000	Below Average Wine	88.582090
4	Argentina	2004	88.214286	Below Average Wine	89.083102

Wine Rating by Country Over Time

```
# Set country as the index
# avg_country_year_rating_df.set_index('country', inplace=True)

## Get US wine rating by year
ca_wine_ratings_df = avg_country_year_rating_df.loc["Canada"]
fr_wine_ratings_df = avg_country_year_rating_df.loc["France"]
us_wine_ratings_df = avg_country_year_rating_df.loc["US"]
au_wine_ratings_df = avg_country_year_rating_df.loc["Australia"]
it_wine_ratings_df = avg_country_year_rating_df.loc["Italy"]
sp_wine_ratings_df = avg_country_year_rating_df.loc["Spain"]
ag_wine_ratings_df = avg_country_year_rating_df.loc["Argentina"]

us_wine_ratings_df

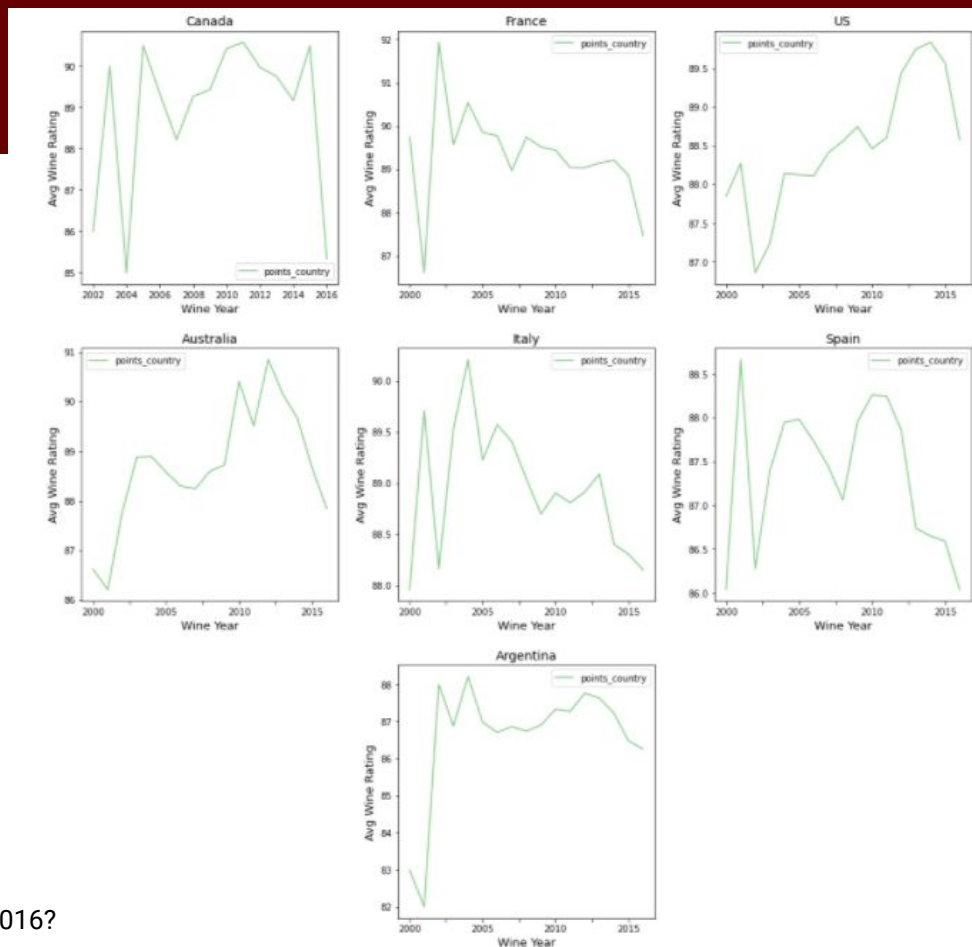
# Create subplots for each country to show rating by year
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15,15))
ca_wine_ratings_df.plot(ax=axes[0,0], x='Wine Year', y='points_country', colormap='Accent')
fr_wine_ratings_df.plot(ax=axes[0,1], x='Wine Year', y='points_country', colormap='Accent')
us_wine_ratings_df.plot(ax=axes[0,2], x='Wine Year', y='points_country', colormap='Accent')
au_wine_ratings_df.plot(ax=axes[1,0], x='Wine Year', y='points_country', colormap='Accent')
it_wine_ratings_df.plot(ax=axes[1,1], x='Wine Year', y='points_country', colormap='Accent')
sp_wine_ratings_df.plot(ax=axes[1,2], x='Wine Year', y='points_country', colormap='Accent')
ag_wine_ratings_df.plot(ax=axes[2,1], x='Wine Year', y='points_country', colormap='Accent')

## Add xlabel for each graph
for x in range(3):
    for y in range(3):
        axes[x,y].set_xlabel("Wine Year", fontsize=13)
        axes[x,y].set_ylabel("Avg Wine Rating", fontsize=13)

## Add titles for each graph
axes[0,0].set_title("Canada", fontsize=14)
axes[0,1].set_title("France", fontsize=14)
axes[0,2].set_title("US", fontsize=14)
axes[1,0].set_title("Australia", fontsize=14)
axes[1,1].set_title("Italy", fontsize=14)
axes[1,2].set_title("Spain", fontsize=14)
axes[2,1].set_title("Argentina", fontsize=14)

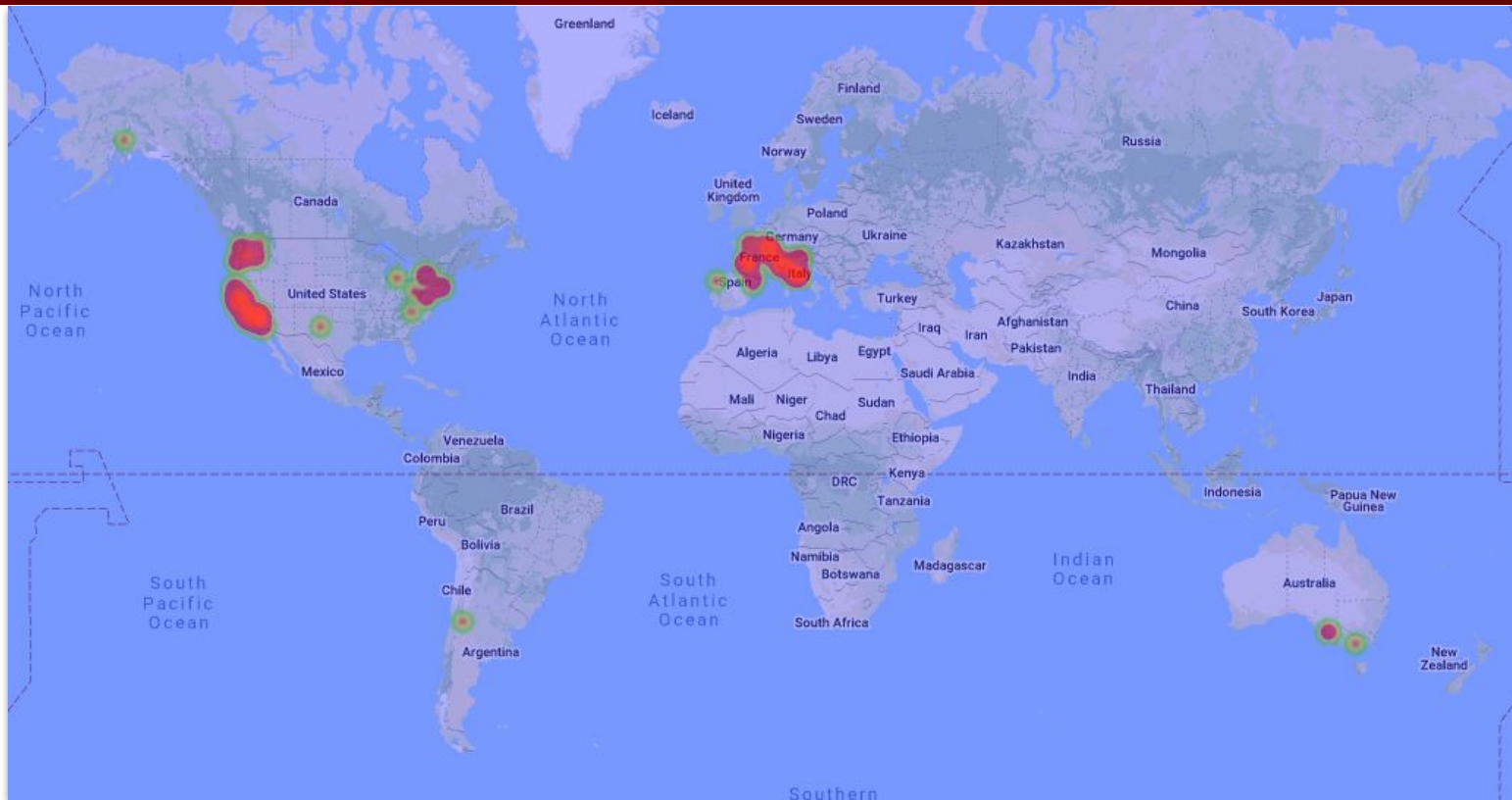
fig.tight_layout(pad=2.0)

fig.delaxes(axes[2][0])
fig.delaxes(axes[2][2])
```



- 1) Plot results show the average wine score for each country over time.
- 2) Trend shows a drop in wine score / quality for all countries in 2015 / 2016?

Mapping Wines by Region



Constructing the API Query

```
1 # Hit the OpenCageData API for each winery.
2 # Set up a Dictionary to hold reponse info for:
3 town=[]
4 lat=[]
5 lng=[]
6
7
8 print("Beginning search for nearby wineries... \n-----")
9
10 # Loop through the DataFrame to append a Lat, Long, and town for each winery
11 for index, row in map_sample_pd.iterrows():
12
13     # Query OpenCage API with the Winery Name, Region 1, Province, and Country
14     query = f'{row["winery"]},{row["region_1"]},{row["province"]},{row["country"]}'
15     place = urllib.parse.quote(query)
16
17     base_url = (f'https://api.opencagedata.com/geocode/v1/json?q={place}&key={api_key}')
18
```

Using Conditionals to Parse JSON Responses

```
18
19
20 try:
21     print(f"Searching for winery: {query}")
22     response = requests.get(base_url).json()
23
24     # Selecting responses with a medium to high confidence score: >4 or
25     num_responses = len(response['results'])
26     print(f'num of responses {num_responses}')
27     for x in range(num_responses):
28         print(f'response{x}')
29         confidence = response['results'][x]['confidence']
30         if confidence >4:
31             print(f'confidence:{confidence}')
32             town.append(response['results'][x]['components']['town'])
33             lat.append(response['results'][x]['geometry']['lat'])
34             lng.append(response['results'][x]['geometry']['lng'])
35             print(f'---> {query} appended')
36             print(" ")
37             break
38         elif x == (num_responses - 1):
39             print(f'Not confident: {confidence} - Skipping {query}...')
40             print(" ")
41             town.append(None)
42             lat.append(None)
43             lng.append(None)
44         else:
45             print(f'Not confident: {confidence} - Next result...')
46             print(" ")
47     except:
48         print(f"X--X No wineries found. Skipping {query}...")
49         print(" ")
50         town.append(None)
51         lat.append(None)
52         lng.append(None)
```

Beginning search for nearby wineries...

Searching for winery: Château Tour des Gendres,Bergerac Sec,Southwest France,France

num of responses 4

response0

Not confident: 1 - Next result...

response1

Not confident: 2 - Next result...

response2

Not confident: 2 - Next result...

response3

confidence:10

X--X No wineries found. Skipping Château Tour des Gendres,Bergerac Sec,Southwest France,France...

Searching for winery: Edmeades,Mendocino Ridge,California,US

num of responses 2

response0

Not confident: 4 - Next result...

response1

confidence:7

---> Edmeades,Mendocino Ridge,California,US appended

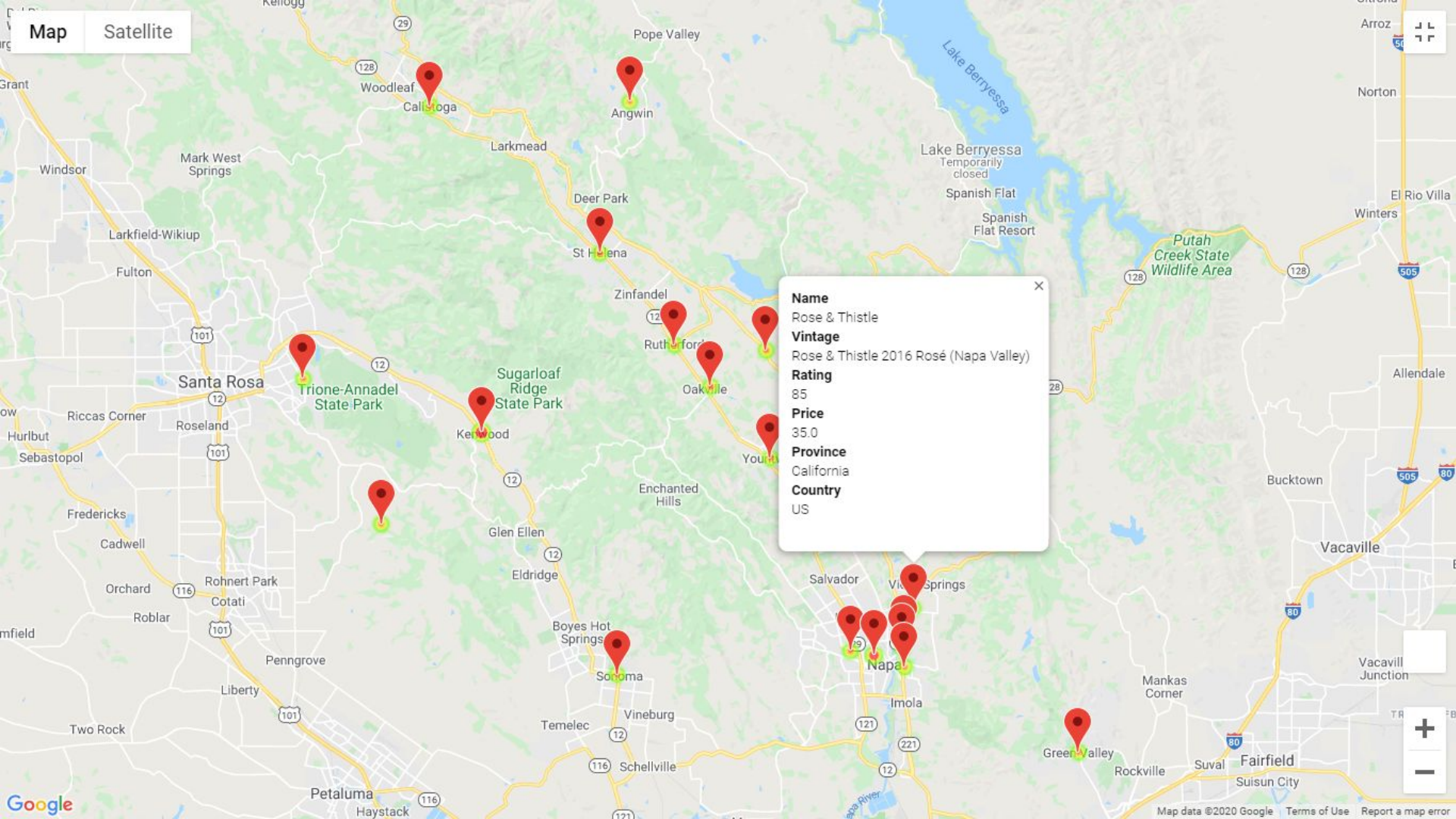
Searching for winery: Cave B,Columbia Valley (WA),Washington,US

num of responses 2

response0

Not confident: 4 - Next result...

Map Satellite



Name
Rose & Thistle

Vintage
Rose & Thistle 2016 Rosé (Napa Valley)

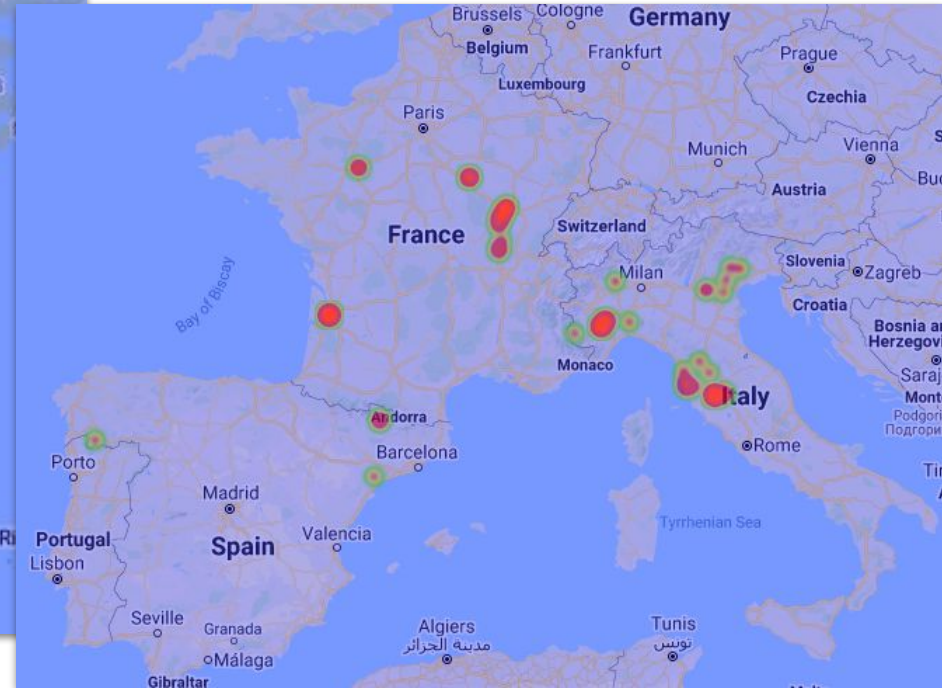
Rating
85

Price
35.0

Province
California

Country
US

Creating Heatmap Layers: Wine Rating & Price



Wine Reviews

	country	description	designation	points	price	province	region_1	region_2	title	variety	winery
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore		St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks

```
wine_data_pd.describe().drop(['Unnamed: 0'], axis=1)
```

	points	price
count	129971.000000	120975.000000
mean	88.447138	35.363389
std	3.039730	41.022218
min	80.000000	4.000000
25%	86.000000	17.000000
50%	88.000000	25.000000
75%	91.000000	42.000000
max	100.000000	3300.000000



```
clean_data_pd.describe().drop(['Unnamed: 0'], axis=1)
```

	points	price
count	70175.000000	70175.000000
mean	88.806341	39.559017
std	3.061324	35.729358
min	80.000000	4.000000
25%	87.000000	20.000000
50%	89.000000	30.000000
75%	91.000000	49.000000
max	100.000000	2013.000000



**VERIFYING CLEANING DATA
DOES NOT MISREPRESENT ORIGINAL FINDINGS**

	country	description	designation	points	price	province	region_1	region_2	title	variety	winery
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks



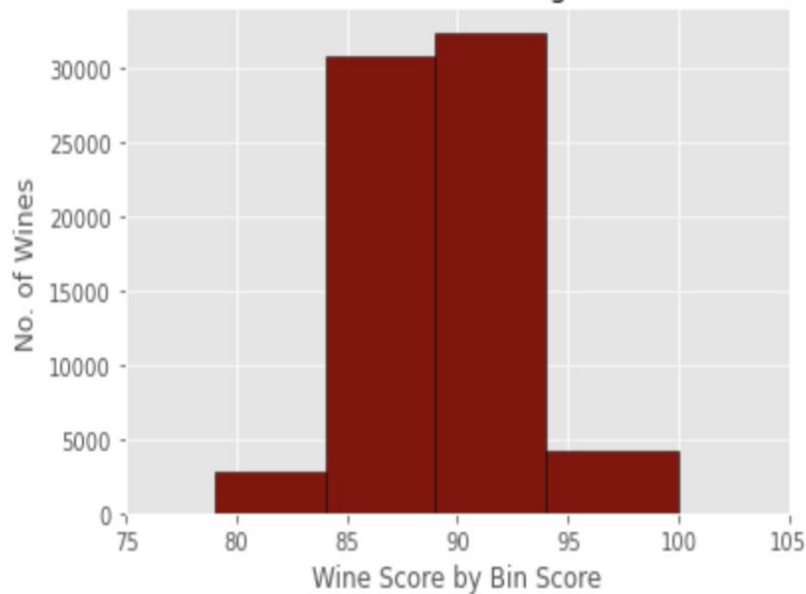
Wine Scores

```
bins = [0,74,79,84,89,94,100]
group_names = ["50-74 Not recommended",
               "75-79 Mediocre: a drinkable wine that may have minor flaws",
               "80-84 Good: a solid, well-made wine",
               "85-89 Very good: a wine with special qualities",
               "90-94 Outstanding: a wine of superior character and style",
               "95-100 Classic: a great wine"]
```

```
: clean_data_pd["Wine Score"] = pd.cut(clean_data_pd["points"],bins,
                                       labels=group_names, include_lowest=True)
wine_score_bins = clean_data_pd.groupby("Wine Score").count()
wine_score_bins[["variety"]]
```

	variety
Wine Score	
50-74 Not recommended	0
75-79 Mediocre: a drinkable wine that may have minor flaws	0
80-84 Good: a solid, well-made wine	5642
85-89 Very good: a wine with special qualities	34565
90-94 Outstanding: a wine of superior character and style	28416
95-100 Classic: a great wine	1552

Wine Scores Histogram



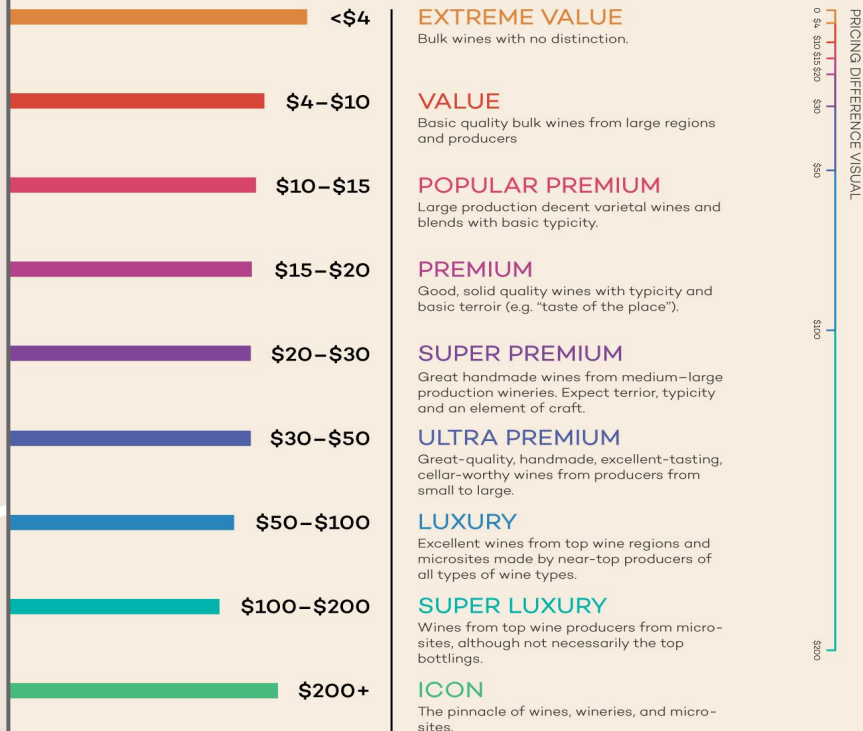
Wine Price

```
price_bins = [0,4,10,15,20,30,50,100,200,3000]
price_group_names = ["EXTREME VALUE <$4", "VALUE $4-$10",
    "POPULAR PREMIUM $10-$15", "PREMIUM $15-$20",
    "SUPER PREMIUM $20-$30", "ULTRA PREMIUM $30-$50",
    "LUXURY $50-$100", "SUPER LUXURY $100-$200", "ICON $200 +"]
clean_data_pd["Wine Price Segments"] = pd.cut(clean_data_pd["price"], price_bins,
    labels=price_group_names, include_lowest=True)
group = clean_data_pd.groupby('Wine Price Segments')
wine_price_bins = clean_data_pd.groupby("Wine Price Segments").count()
wine_price_bins[["variety"]]
```

	variety
Wine Price Segments	
EXTREME VALUE <\$4	5
VALUE 4-10	2048
POPULAR PREMIUM 10-15	8072
PREMIUM 15-20	10332
SUPER PREMIUM 20-30	15389
ULTRA PREMIUM 30-50	19448
LUXURY 50-100	12407
SUPER LUXURY 100-200	2102
ICON \$200 +	372

WINE PRICING SEGMENTS

(What You Get For What You Spend)



TYPICITY: A wine tasting "varietally" correct (e.g. a Cabernet Franc wine that is indicative of the Cabernet Franc variety).

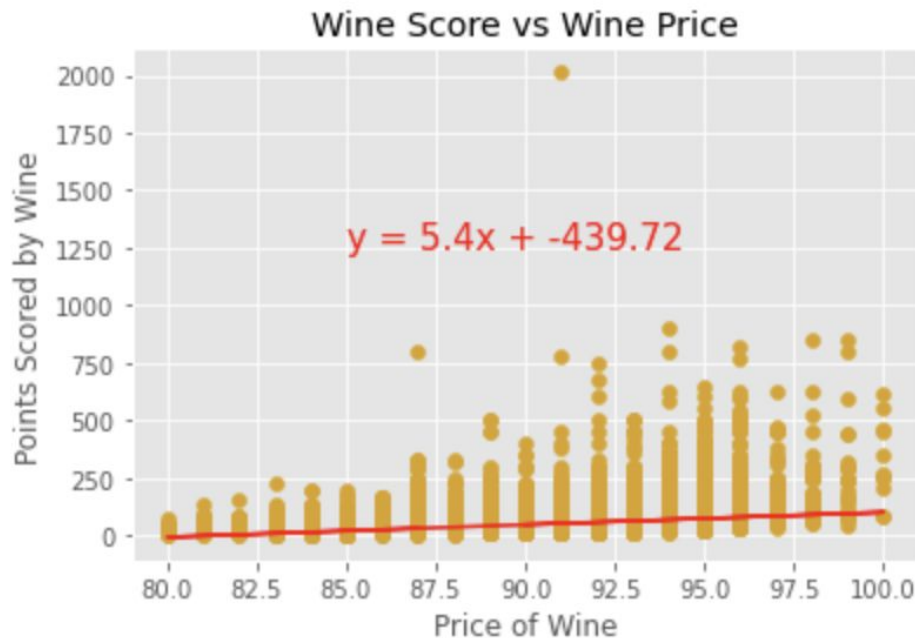
TERROIR: A wine having flavors (and aromas) that are indicative of the location where it was grown.

This model is based on the Table 5 from "Analyzing the US retail wine market using price and consumer segmentation models," AWBR (2005), with inflation accounted for (2005-2016) and consideration taken from price observations from online retailers (klwines.com, wine.com, totalwine.com and winelibrary.com). We are not economists, this is purely for observation and not official reference.

Wine Score^{v.} Wine Price

- Very little to no relationship based on R-squared value
- 21% R-squared value
- There is some correlation

The r-squared is : 0.2138253489337102



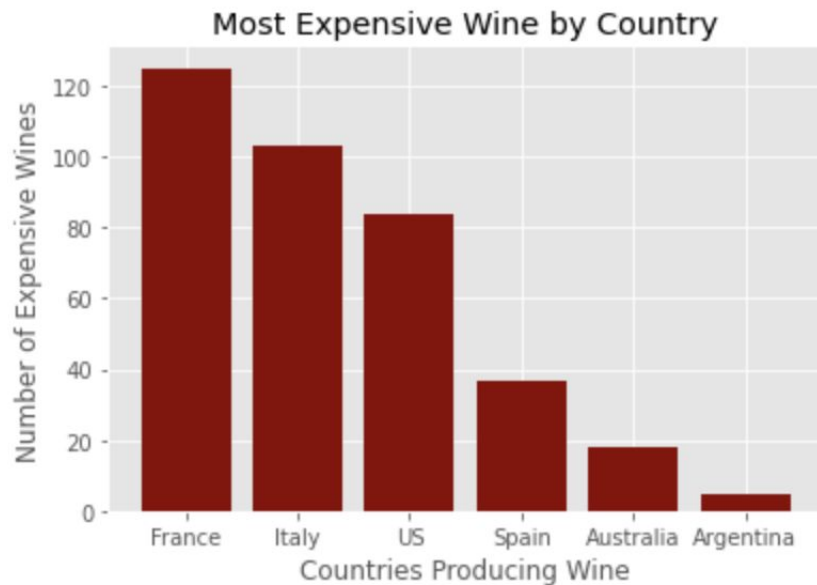
```
correlation = sts.pearsonr(x_values,y_values)
print(f"The correlation between price and points is {round(correlation[0],2)}")
```

The correlation between both factors is 0.46

****NOTICE**** 100 point perfect-scoring-wines at a fairly low price point.

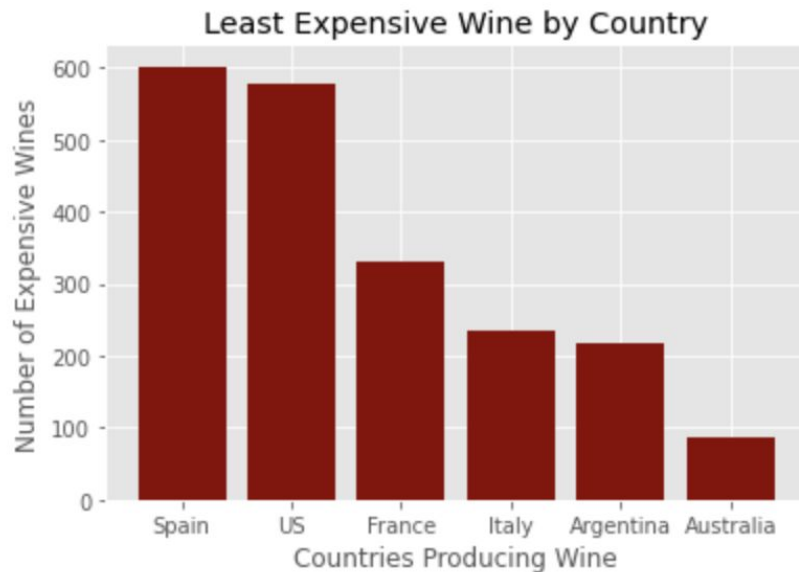
Most Expensive Wines by Country

	country	count
0	France	125
1	Italy	103
2	US	84
3	Spain	37
4	Australia	18
5	Argentina	5



Least Expensive Wines by Country

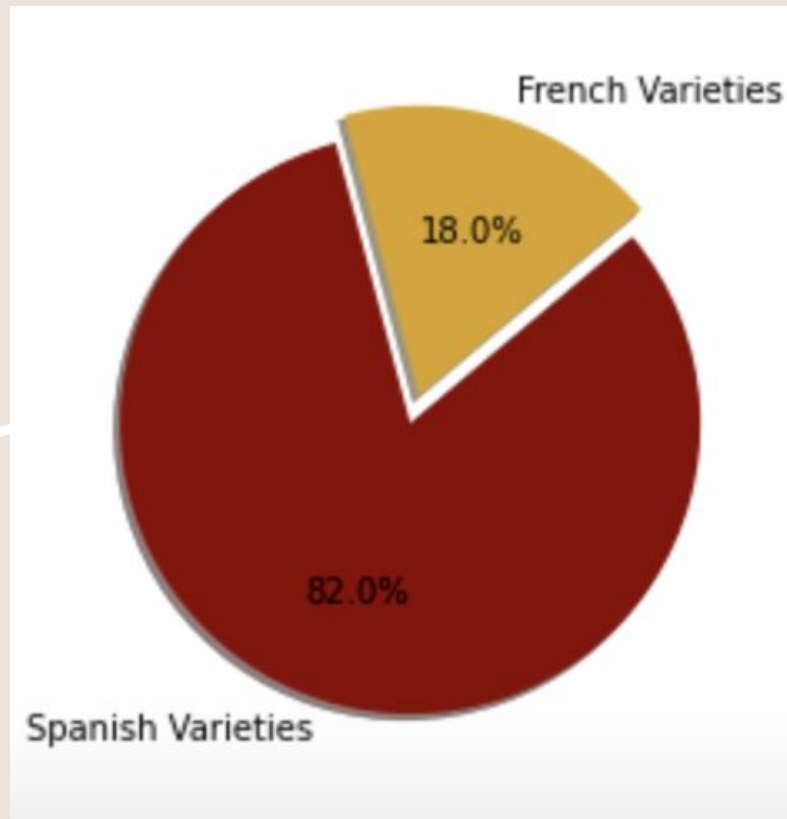
	country	count
0	Spain	602
1	US	578
2	France	330
3	Italy	234
4	Argentina	218
5	Australia	86



Most V. Least Expensive Varieties

	\$200+	ICON The pinnacle of wines, wineries, and micro-sites.
	\$4-\$10	VALUE Basic quality bulk wines from large regions and producers

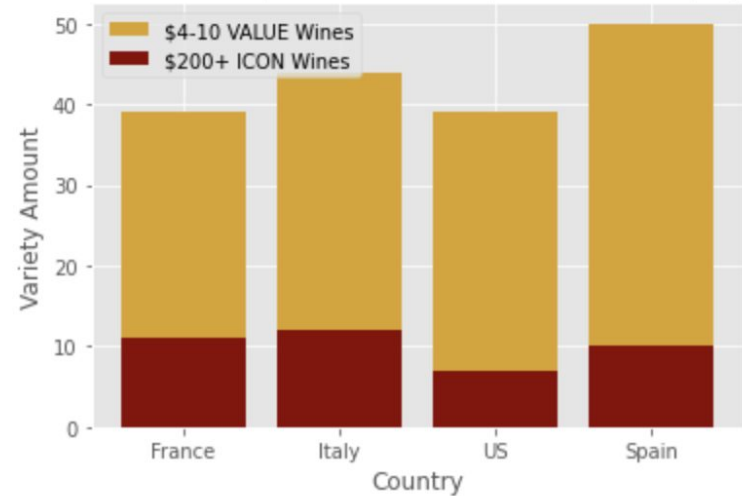
- There are far more varieties in Spain
- We can expect more varieties from larger regions and producers that are producing bulk wines



Varieties Pattern

	Most Expensive	Least Expensive
France	11	39
Italy	12	44
US	7	39
Spain	10	50

Number of Varieties (Most v Least Expensive Wines) by Country



```
expensive_french_wine = most_expensive_wine.loc[most_expensive_wine["country"]=="France"]
expensive_italian_wine = most_expensive_wine.loc[most_expensive_wine["country"]=="Italy"]
expensive_us_wine = most_expensive_wine.loc[most_expensive_wine["country"]=="US"]
expensive_spanish_wine = most_expensive_wine.loc[most_expensive_wine["country"]=="Spain"]
```

```
least_expensive_spanish_wine = least_expensive_wine.loc[least_expensive_wine["country"]=="Spain"]
least_expensive_us_wine = least_expensive_wine.loc[least_expensive_wine["country"]=="US"]
least_expensive_french_wine = least_expensive_wine.loc[least_expensive_wine["country"]=="France"]
least_expensive_italian_wine = least_expensive_wine.loc[least_expensive_wine["country"]=="Italy"]
```

Temperature vs Price

```
In [262]: #Making a for loop to find the the latitude and longitude of each city in which the winery is located
for location in location:
    url = (f'https://api.opencagedata.com/geocode/v1/json?q={location[0]}%2C%20{location[1]}&key={api_key}')
    response = requests.get(url)
    cordinate_data = response.json()
    print(cordinate_data)
    #Appending the lat and long to their list
    lat.append(cordinate_data['results'][0]['geometry']['lat'])
    lng.append(cordinate_data['results'][0]['geometry']['lng'])
```

Open Cage

First, I cleaned the data so every row would not contain any NaN. By doing this it eliminated a lot of data, but allowed me to have a more precise location in region_2.

The API open cage allows you to do inputs by place, but I noticed that by running it by only place, it occasionally returned incorrect coordinates. To eliminate this issue I used region_2 and the country code and it generally returned correct latitudes and longitudes.

Temperature vs Price

```
In [264]: Setting city_list as a list
city_list = []

#Finding the city for the corresponding coordinates
for coordinates in coordinates:
    city = citipy.nearest_city(coordinates[0], coordinates[1]).city_name
    city_list.append(city)

#Finding the length of the list of cities
len(city_list)
```

CitiPy

After receiving the latitude and longitude of the corresponding region_2, I then find the nearest city for that area. Although the city is not the exact point of the winery, the city is in the same general area of the winery.

Temperature vs Price

```
In [266]: #Setting temp and hum and a list
temp = []
hum = []

#Setting city to its own DataFrame
city = best_us_wineries['City']

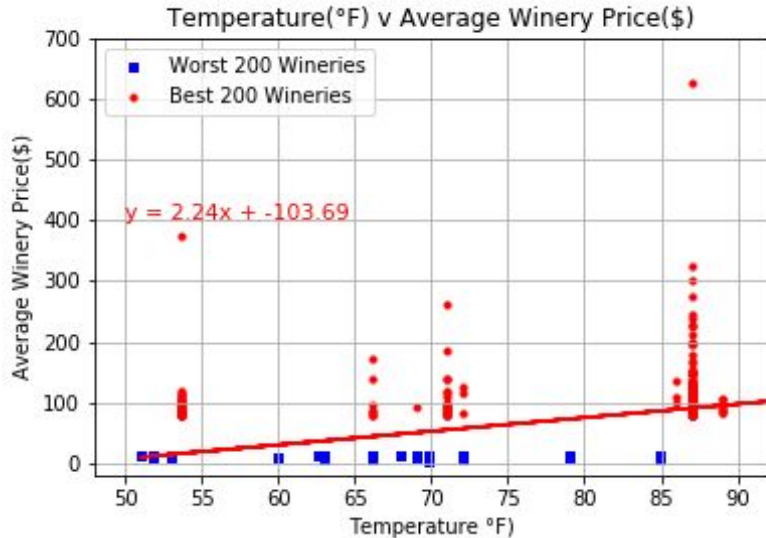
#Finding the temperature for the corresponding city
for city in city:
    query_url = url + 'appid=' + weather_api_key + '&q=' + city + '&units=imperial'
    weather_response = requests.get(query_url)
    weather_data = weather_response.json()

    temp.append(weather_data['main']['temp_max'])
```

Open Weather API

After getting the cities from citipy I then enter the city into the openweather api and obtain the max temperature for the city. Following this I append all the information in a data frame for the best 200 wineries vs the worst 200 wineries.

Temperature vs Price



Linear Regression

I then plot the best 200 wineries v the worst 200 wineries. At first you see a scattered distribution along all temperatures for both data sets, which leads me to believe that there is very little relationship between average winery price and temperature.

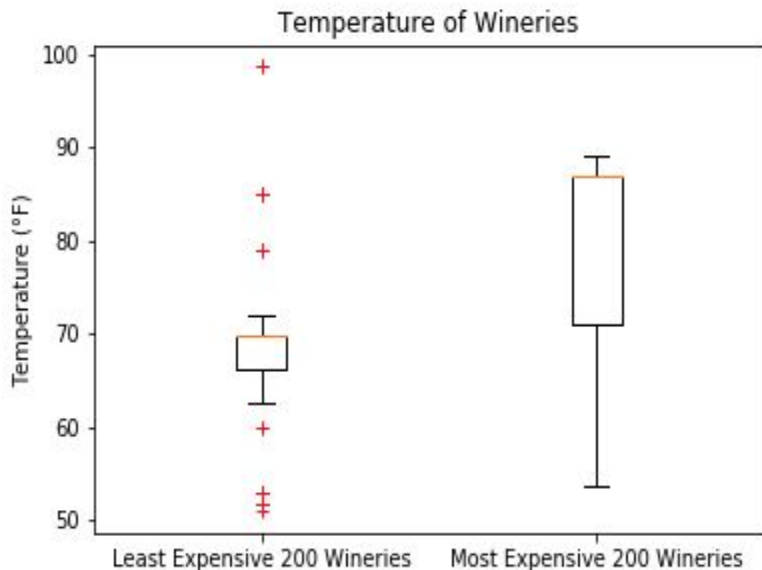
After taking the linear regression, that became more clear with a relatively flat linear regression line and an R^2 value that is equal to 0.15164039280952818.

This led me to reject my thought that temperature can predict average winery price. While this was unsuccessful, it is evident that most of the Best 200 Wineries tended to fall in much higher temperatures relative to the Worst 200 Wineries

Temperature vs Type of Winery

Histogram

This ultimately led me to look further into the distributions of the temperatures for both least and most expensive wineries. After creating and analyzing the histogram, it seems that almost 75% of most expensive wineries were in areas that had a higher temperature than the least expensive wineries.



Temperature vs Type of Winery

$$H_o = \bar{x}_1 \neq \bar{x}_2$$

$$H_a = \bar{x}_1 = \bar{x}_2$$

```
#Finding the p_value and printing
```

```
p_value = ttest_ind(best_temp, temp)
```

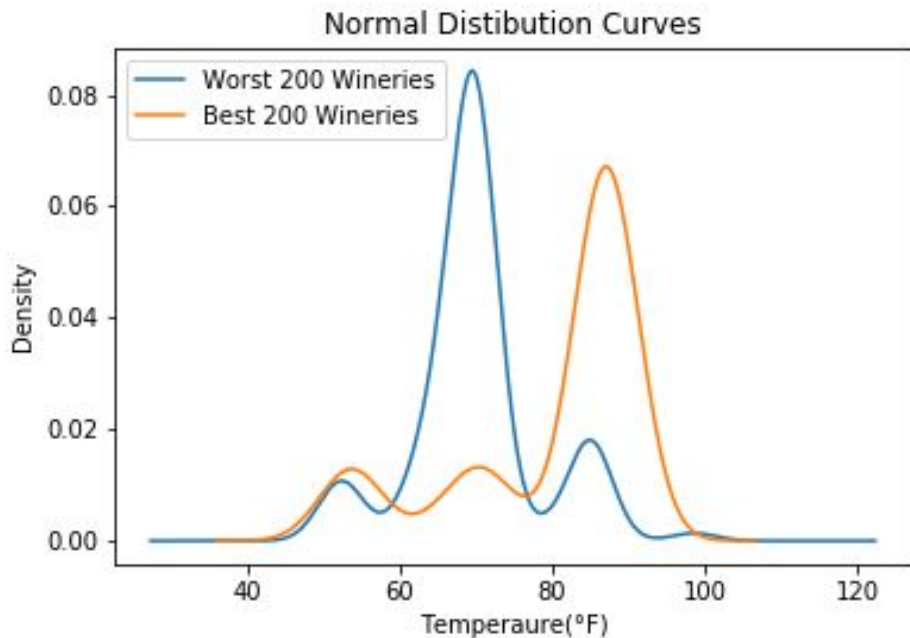
```
(f'The pvalue for Best 200 Winneries and Worst 200 Wineries is {p_value[1]}')
```

```
'The pvalue for Best 200 Winneries and Worst 200 Wineries is 9.94354677351303e-20'
```

T-Test

To confirm my suspicion, I decided to take a t-test to affirm that both averages are significantly different. The worst wineries had an average temperature of 69.9 °F while the best wineries had an average temperature of 79.9 °F. After running the t-test I found that the p-value is 9.9×10^{-20} , meaning that we fail to reject H_o .

Temperature vs Type of Winery



If one wants to look at their normal distribution, it becomes even more evident that the worst wineries have the peak of their distribution to the left of the best wineries.

Questions?

...