EDA and Visualization on the Wine Data

In this notebook, we will create basic data analysis to understand some characteristics about the different wines based on the regions and different wineries.

Import Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy.stats import kurtosis, skew
from scipy import stats
```

Functions def

In [2]:
```
def resumetable(df):
    print(f"Dataset Shape: {df.shape}")
    summary = pd.DataFrame(df.dtypes,columns=['dtypes'])
    summary = summary.reset_index()
    summary['Name'] = summary['index']
    summary = summary[['Name','dtypes']]
    summary['Missing'] = df.isnull().sum().values
    summary['Uniques'] = df.nunique().values
    summary['First Value'] = df.loc[0].values
    summary['Second Value'] = df.loc[1].values
    summary['Third Value'] = df.loc[2].values

    for name in summary['Name'].value_counts().index:
        summary.loc[summary['Name'] == name, 'Entropy'] = round(stats.entropy(df[name].value_counts(normalize=True), base=2),2)

    return summary

def CalcOutliers(df_num):
    '''

    Set a numerical value and it will calculate the upper, lower and total number of outliers
    It will print a lot of statistics of the numerical feature that you set on input

    '''
    # calculating mean and std of the array
    data_mean, data_std = np.mean(df_num), np.std(df_num)

    # seting the cut line to both higher and lower values
    # You can change this value
    cut = data_std * 3

    #Calculating the higher and lower cut values
    lower, upper = data_mean - cut, data_mean + cut

    # creating an array of lower, higher and total outlier values
    outliers_lower = [x for x in df_num if x < lower]
    outliers_higher = [x for x in df_num if x > upper]
    outliers_total = [x for x in df_num if x < lower or x > upper]
```

```
    # array without outlier values
    outliers_removed = [x for x in df_num if x > lower and x < upper]

    print('Identified lowest outliers: %d' % len(outliers_lower)) # printing total number of values in lower cut of outliers
    print('Identified upper outliers: %d' % len(outliers_higher)) # printing total number of values in higher cut of outliers
    print('Identified outliers: %d' % len(outliers_total)) # printing total number of values outliers of both sides
    print('Non-outlier observations: %d' % len(outliers_removed)) # printing total number of non outlier values
    print("Total percentual of Outliers: ", round((len(outliers_total) / len(outliers_removed) )*100, 4)) # Percentual of outliers in points

    return
```

In [3]:
```
# Importing dataset in variable df_wine1
df_wine1 = pd.read_csv('../input/winemag-data-130k-v2.csv', index_col=0)
```

Read data

In [4]:
```
resumetable(df_wine1)
```

```
Dataset Shape: (129971, 13)
```

Out[4]:

	Name	dtypes	Missing	Uniques	First Value	Second Value	Third Value	Entropy
0	country	object	63	43	Italy	Portugal	US	2.77
1	description	object	0	119955	Aromas include tropical fruit, broom, brimston...	This is ripe and fruity, a wine that is smooth...	Tart and snappy, the flavors of lime flesh and...	16.83
2	designation	object	37465	37979	Vulkà Bianco	Avidagos	NaN	13.83
3	points	int64	0	21	87	87	87	3.64
4	price	float64	8996	390	NaN	15	14	5.77
5	province	object	63	425	Sicily & Sardinia	Douro	Oregon	5.05
6	region_1	object	21247	1229	Etna	NaN	Willamette Valley	7.89
7	region_2	object	79460	17	NaN	NaN	Willamette Valley	3.28
8	taster_name	object	26244	19	Kerin O'Keefe	Roger Voss	Paul Gregutt	3.39
9	taster_twitter_handle	object	31213	15	@kerinokeefe	@vossroger	@paulgwine	3.23
10	title	object	0	118840	Nicosia 2013 Vulkà Bianco (Etna)	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Rainstorm 2013 Pinot Gris (Willamette Valley)	16.81
11	variety	object	1	707	White Blend	Portuguese Red	Pinot Gris	5.67
12	winery	object	0	16757	Nicosia	Quinta dos Avidagos	Rainstorm	12.98

looking the distribuition of Points and Prices

In [5]:
```
# The function describe is focused on numerical features
# in this case are points and price
print("Statistics of numerical data: ")
print(df_wine1.describe())
```

```
Statistics of numerical data:
              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
```

Very interesting distribuition of Points and Price.

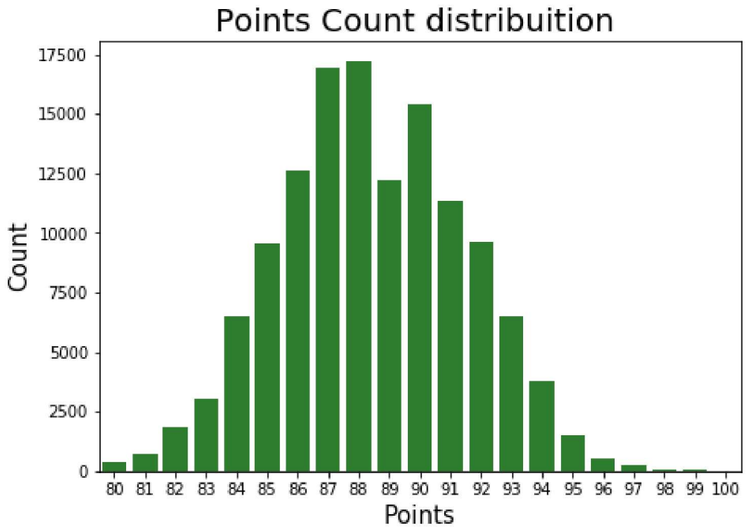
- We can see that the values of points are distributed between 80 and 100
- The price have a high difference between the values and a high standard deviation
- In prices, the IQR (Interquantil Range) is 25. The max value is 3300;

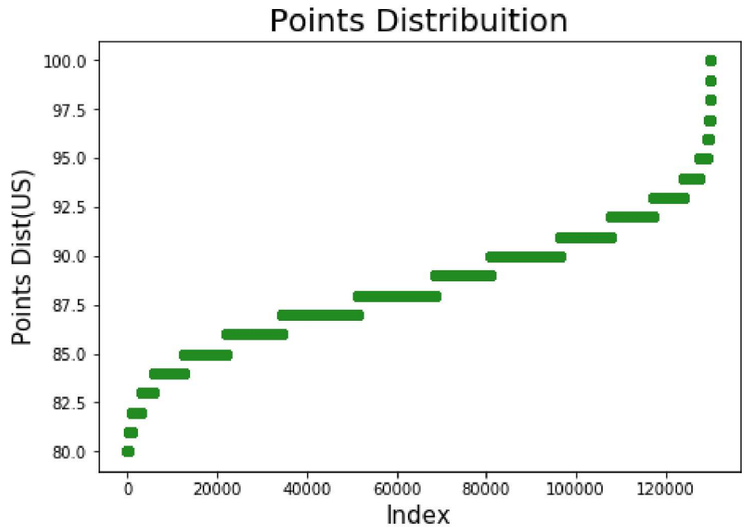
We will explore it further later, lets start.

Points Distribution

- I will start exploring points and after I will see the same on price
- Let's see the points distributions and quantiles

In [6]:
```
# define the size of figures that I will build
plt.figure(figsize=(16,5))

plt.subplot(1,2,1) # this will create a grid of 1 row and 2 columns; this is the first graphic
g = sns.countplot(x='points', data=df_wine1, color='forestgreen') # seting the seaborn countplot to known the points distribuition
g.set_title("Points Count distribuition ", fontsize=20) # seting title and size of font
g.set_xlabel("Points", fontsize=15) # seting xlabel and size of font
g.set_ylabel("Count", fontsize=15) # seting ylabel and size of font

plt.subplot(1,2,2)  # this will set the second graphic of our grid
plt.scatter(range(df_wine1.shape[0]), np.sort(df_wine1.points.values), color='forestgreen') # creating a cumulative distribution
plt.xlabel('Index', fontsize=15)  # seting xlabel and size of font
plt.ylabel('Points Dist(US)', fontsize=15)  # seting ylabel and size of font
plt.title("Points Distribuition", fontsize=20) # seting title and size of font

plt.show() #rendering the graphs
```





We can clearly see the distribution of the data. It appears very similar to a normal distribution.

Only a small number of wines have fewer than 82 points or more than 95 points.

As points are one of the most important metrics in this dataset, I will explore it further. I will look for more important information, such as quantiles, and perhaps try to categorize wines based on it.

Set the points into categories.

- I will create a new feature containing the range of the points with a rank to this pontuations

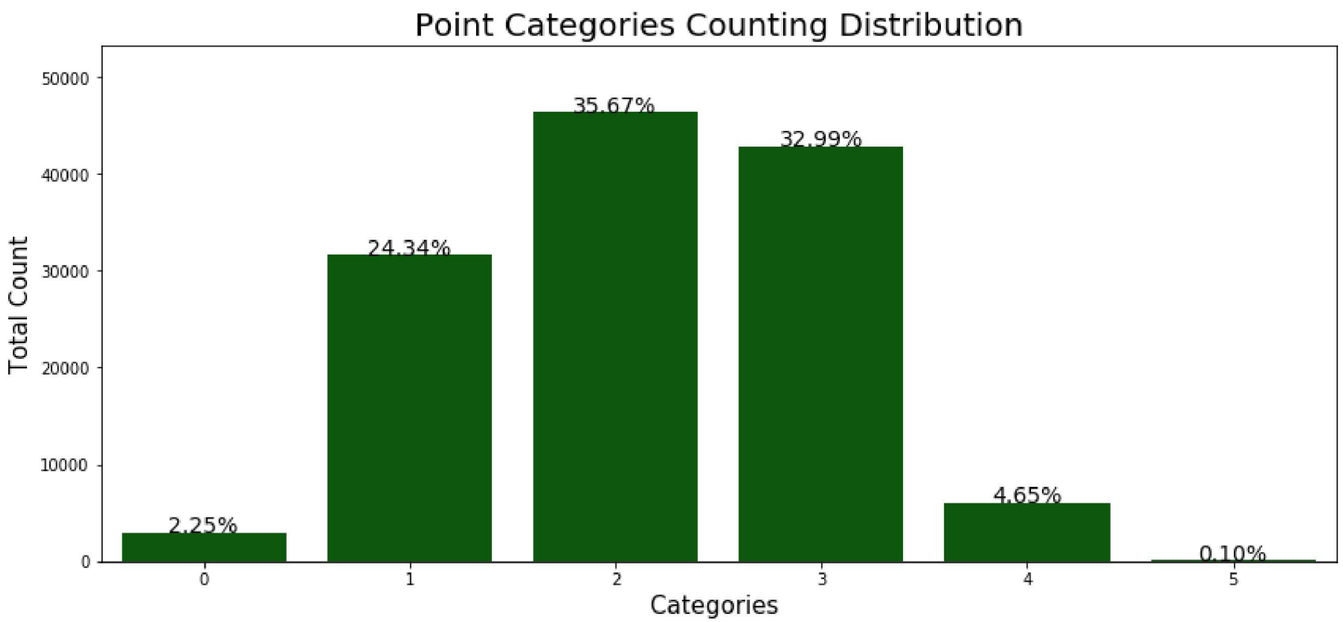
In [7]:
```
def cat_points(points):
    if points in list(range(80,83)):
        return 0
    elif points in list(range(83,87)):
        return 1
    elif points in list(range(87,90)):
        return 2
    elif points in list(range(90,94)):
        return 3
    elif points in list(range(94,98)):
        return 4
    else:
        return 5

df_wine1["rating_cat"] = df_wine1["points"].apply(cat_points)
```

Ploting Rating categories

- Let's see the distribution after the transformation

In [8]:
```
total = len(df_wine1)
plt.figure(figsize=(14,6))
```

```
g = sns.countplot(x='rating_cat', color='darkgreen',
                  data=df_wine1)
g.set_title("Point Categories Counting Distribution", fontsize=20)
g.set_xlabel("Categories ", fontsize=15)
g.set_ylabel("Total Count", fontsize=15)

sizes=[]

for p in g.patches:
    height = p.get_height()
    sizes.append(height)
    g.text(p.get_x()+p.get_width()/2.,
           height + 3,
           '{:1.2f}%'.format((height/total)*100),
           ha="center", fontsize=14)

g.set_ylim(0, max(sizes) * 1.15)

plt.show()
```



Now we can have a clearly understand of our data in a more clean way. It could be interesting to compare each other regions.

Detecting Outlier Points

In [9]:
```
CalcOutliers(df_wine1['points'])
```

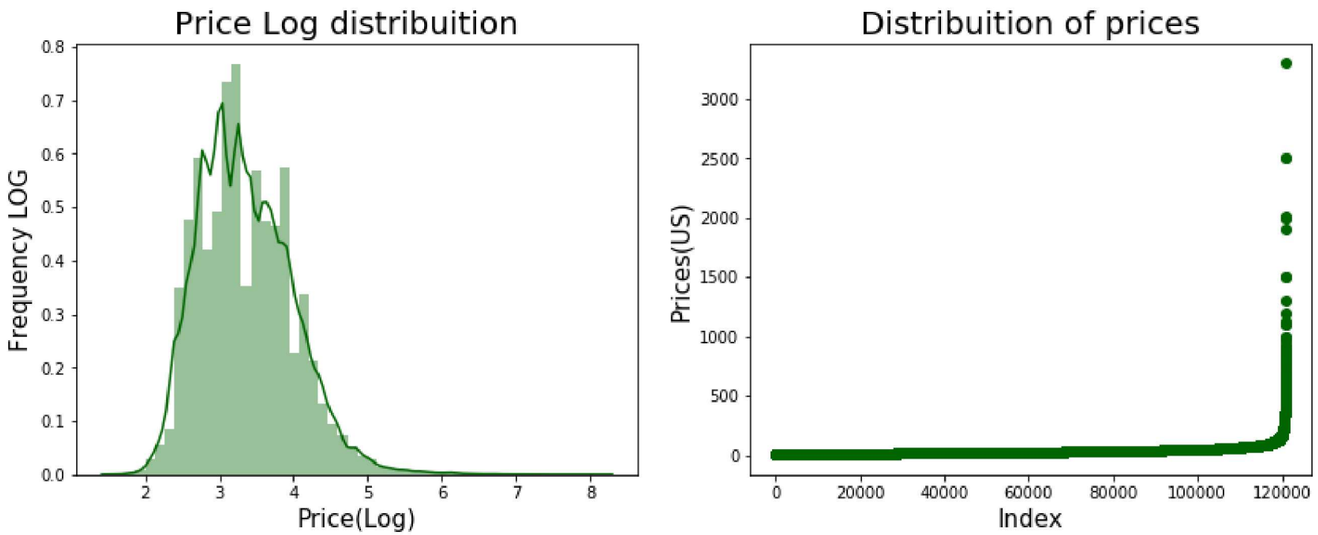
```
Identified lowest outliers: 0
Identified upper outliers: 129
Identified outliers: 129
Non-outlier observations: 129842
Total percentual of Outliers:  0.0994
```

We can see that all outliers of this feature is in upper values.

Also, we have less than 0,01% of outlier values in points.... Just 129 wines have more than 98 points.

Price Distribution

In [10]:
```
plt.figure(figsize=(14,5))

g1 = plt.subplot(121)
g1 = sns.distplot(np.log(df_wine1['price'].dropna() + 1),
                  color='darkgreen')
g1.set_title("Price Log distribuition  ", fontsize=20)
g1.set_xlabel("Price(Log)", fontsize=15)
g1.set_ylabel("Frequency LOG", fontsize=15)

plt.subplot(122)
plt.scatter(range(df_wine1.shape[0]), np.sort(df_wine1.price.values),
            color='darkgreen')
plt.xlabel('Index', fontsize=15)
plt.ylabel('Prices(US)', fontsize=15)
plt.title("Distribuition of prices", fontsize=20)


plt.show()
```



The Price Log give us an impression that the data could normaly distributed, but like in points, we need to test.

Outliers in Prices

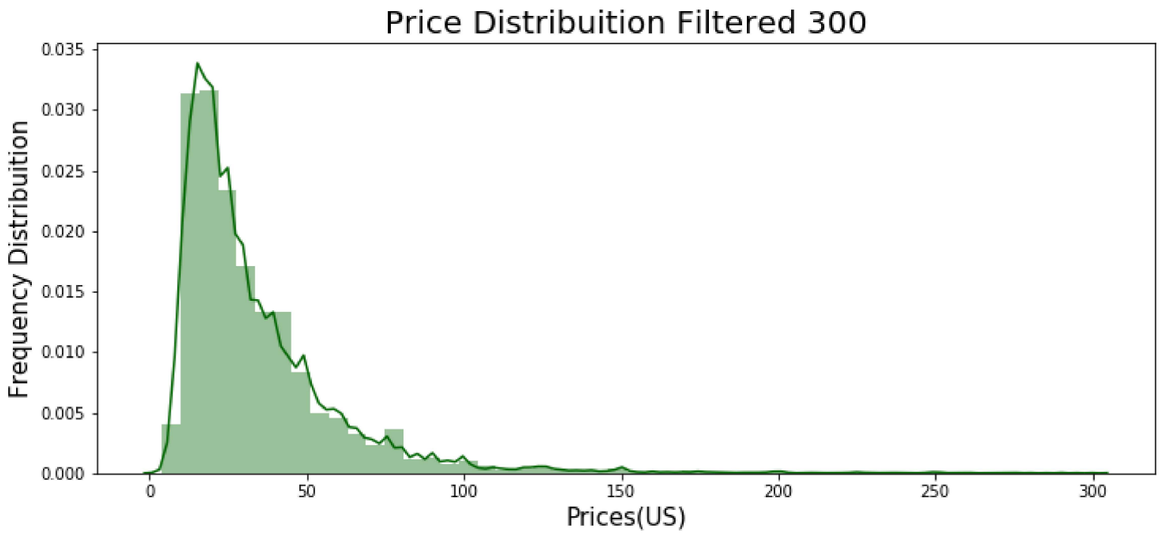
In [11]:
```
CalcOutliers(df_wine1['price'])
```

```
Identified lowest outliers: 0
Identified upper outliers: 1177
Identified outliers: 1177
Non-outlier observations: 119798
Total percentual of Outliers:  0.9825
```

Now we can clearly see that we have less than 1% of outliers, that is 1177 prices that is out of our range

Let's see a filtered distribution of Prices

In [12]:
```
plt.figure(figsize=(12,5))

g = sns.distplot(df_wine1[df_wine1['price'] < 300]['price'], color='darkgreen')
g.set_title("Price Distribuition Filtered 300", fontsize=20)
g.set_xlabel("Prices(US)", fontsize=15)
g.set_ylabel("Frequency Distribuition", fontsize=15)


plt.show()
```



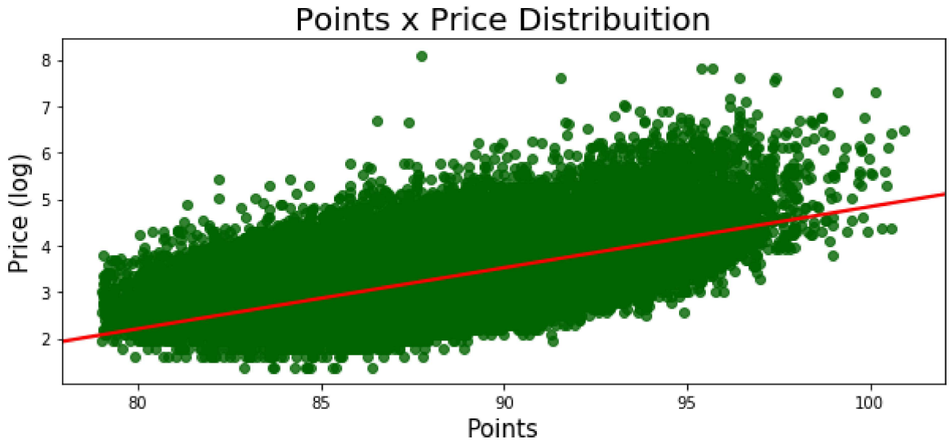
It confirms what we saw in the above exploration. The greastest part of all analyzed wines have values under $ 100;

In [13]:
```
# Let's get tehe price_log to better work with this feature
df_wine1['price_log'] = np.log(df_wine1['price'])
```

```
/opt/conda/lib/python3.6/site-packages/pandas/core/series.py:853: RuntimeWarning: invalid value encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Crossing prices and Points

In [14]:
```
plt.figure(figsize=(10,4))

g = sns.regplot(x='points', y='price_log',
                data=df_wine1, line_kws={'color':'red'},
                x_jitter=True, fit_reg=True, color='darkgreen')
g.set_title("Points x Price Distribuition", fontsize=20)
g.set_xlabel("Points", fontsize= 15)
g.set_ylabel("Price (log)", fontsize= 15)

plt.show()
```



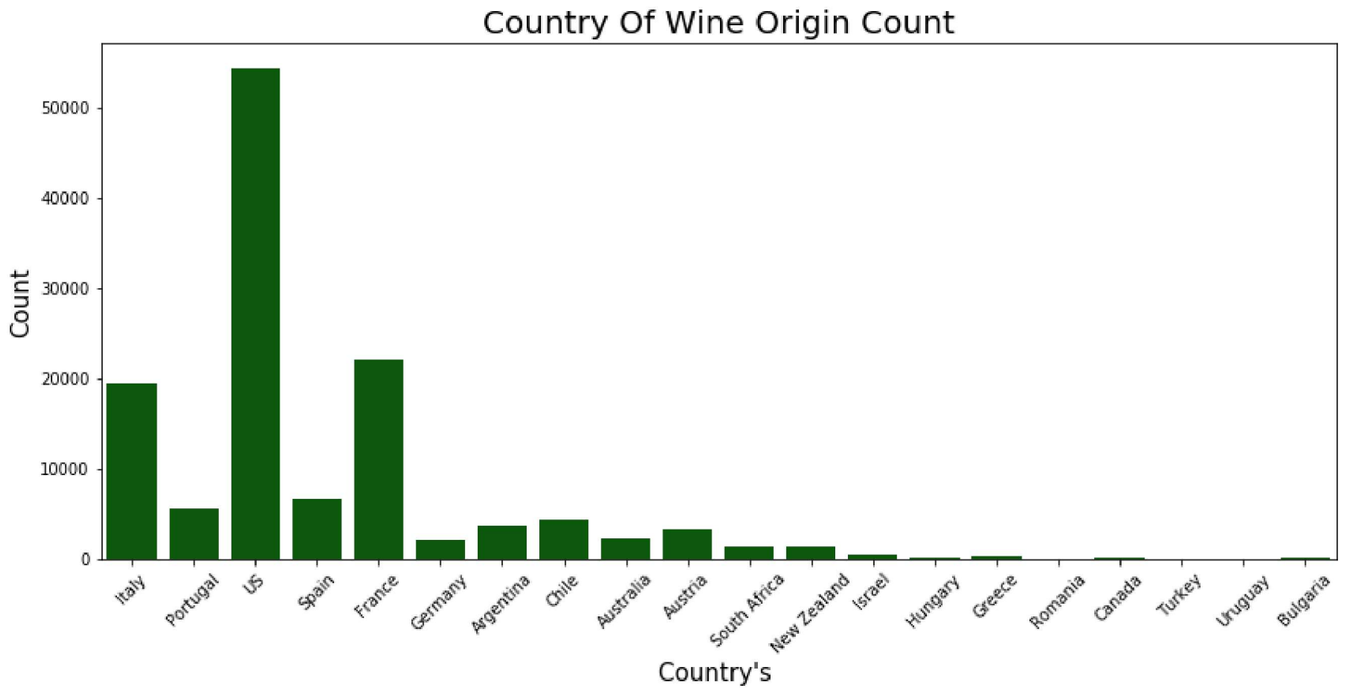
Very meaningful scatter plot.

- The highest prices isn't of the wine with highest pontuation.
- The most expensive wine have ponctuation between 87 and 90

Maybe it would be interesting to build an recommender system to find cheapest wines with the same quality.

Country Feature

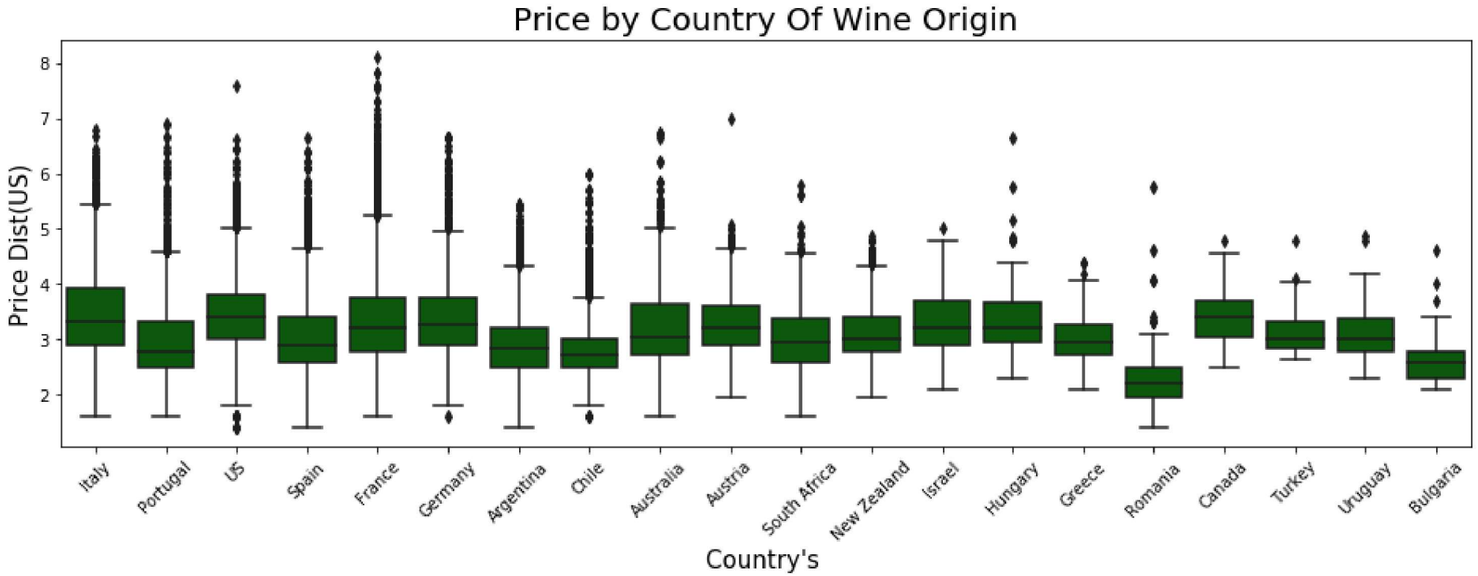
In [15]:
```
plt.figure(figsize=(14,6))

country = df_wine1.country.value_counts()[:20]

g = sns.countplot(x='country',
                  data=df_wine1[df_wine1.country.isin(country.index.values)],
                  color='darkgreen')
g.set_title("Country Of Wine Origin Count", fontsize=20)
g.set_xlabel("Country's ", fontsize=15)
g.set_ylabel("Count", fontsize=15)
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.show()
```

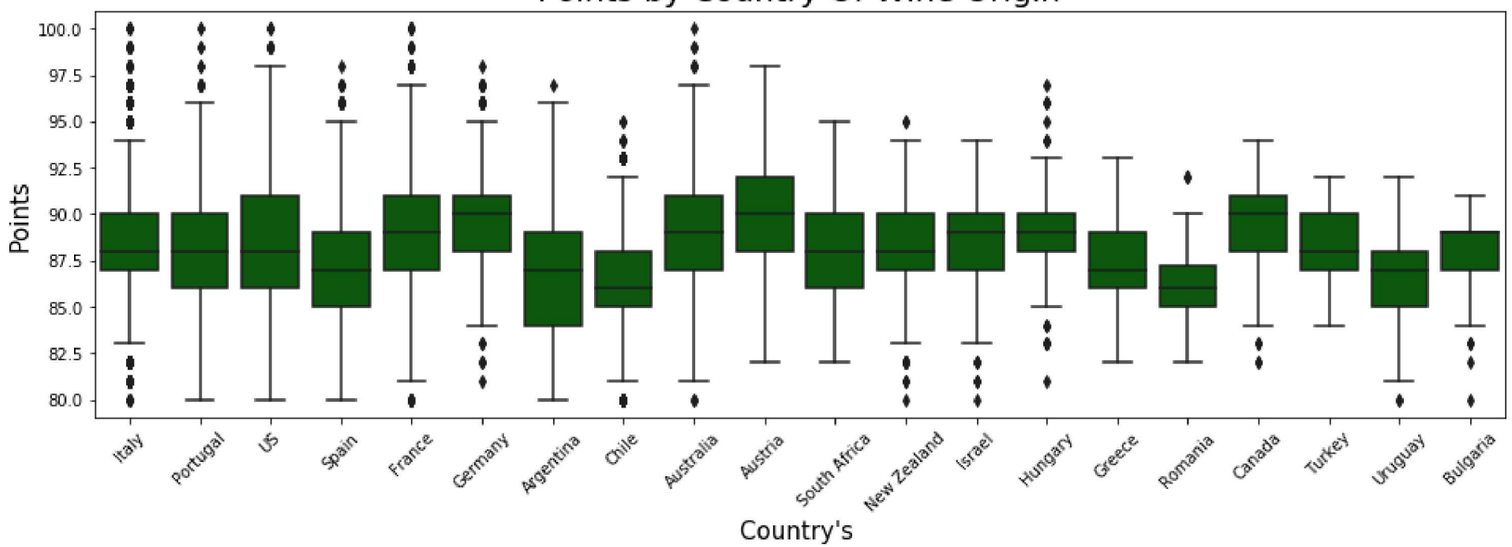


I was expecting to see Italy, Chile or Argentina as the biggest wine productor.

Take a look in the distribuition of this top 20 countrys by price and rating

In [16]:
```
plt.figure(figsize=(16,12))

plt.subplot(2,1,1)
g = sns.boxplot(x='country', y='price_log',
                 data=df_wine1.loc[(df_wine1.country.isin(country.index.values))],
                color='darkgreen')
g.set_title("Price by Country Of Wine Origin", fontsize=20)
g.set_xlabel("Country's ", fontsize=15)
g.set_ylabel("Price Dist(US)", fontsize=15)
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.subplot(2,1,2)
g1 = sns.boxplot(x='country', y='points',
                  data=df_wine1[df_wine1.country.isin(country.index.values)],
                color='darkgreen')
g1.set_title("Points by Country Of Wine Origin", fontsize=20)
g1.set_xlabel("Country's ", fontsize=15)
g1.set_ylabel("Points", fontsize=15)
g1.set_xticklabels(g1.get_xticklabels(),rotation=45)

plt.subplots_adjust(hspace = 0.6,top = 0.9)

plt.show()
```

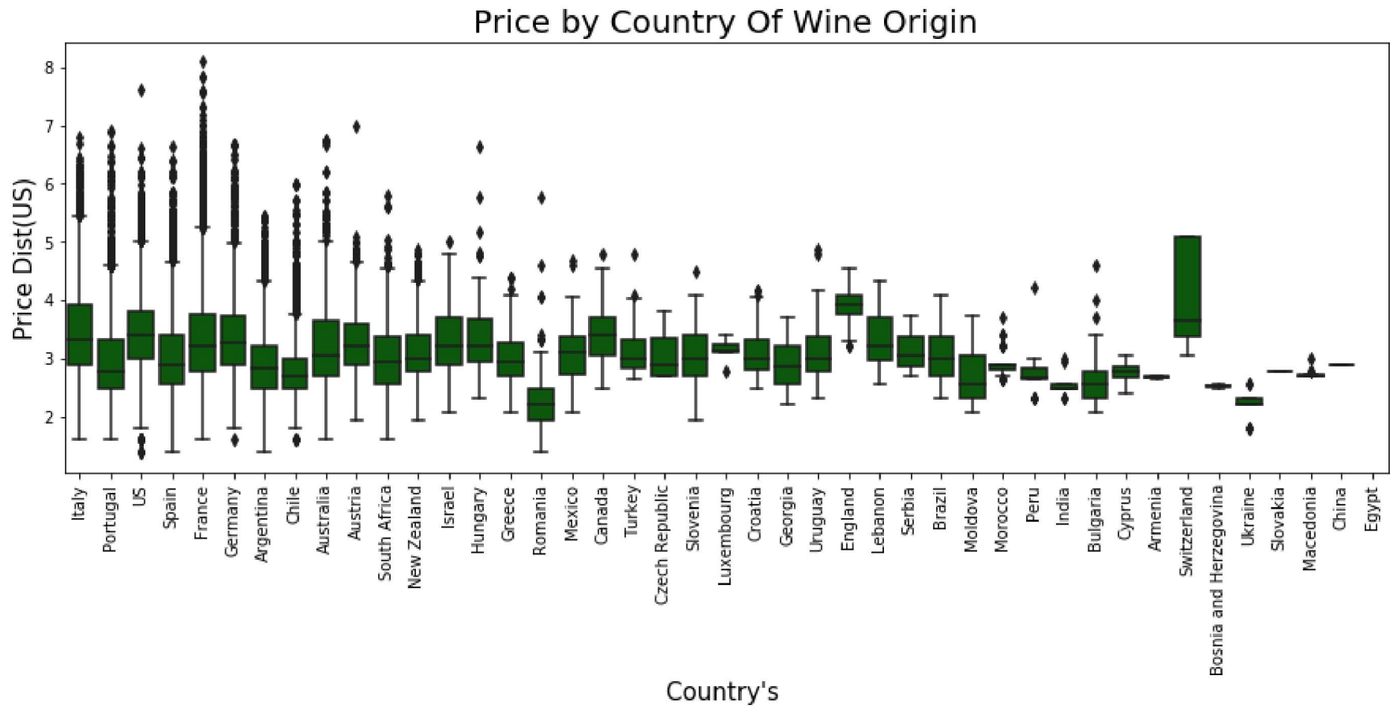




Taking a look on values lowest than 500

In [17]:
```
plt.figure(figsize=(15,5))
g = sns.boxplot(x='country', y='price_log', color='darkgreen',
                data=df_wine1)
g.set_title("Price by Country Of Wine Origin", fontsize=20)
g.set_xlabel("Country's ", fontsize=15)
g.set_ylabel("Price Dist(US)", fontsize=15)
g.set_xticklabels(g.get_xticklabels(),rotation=90)
```

```
plt.show()
```

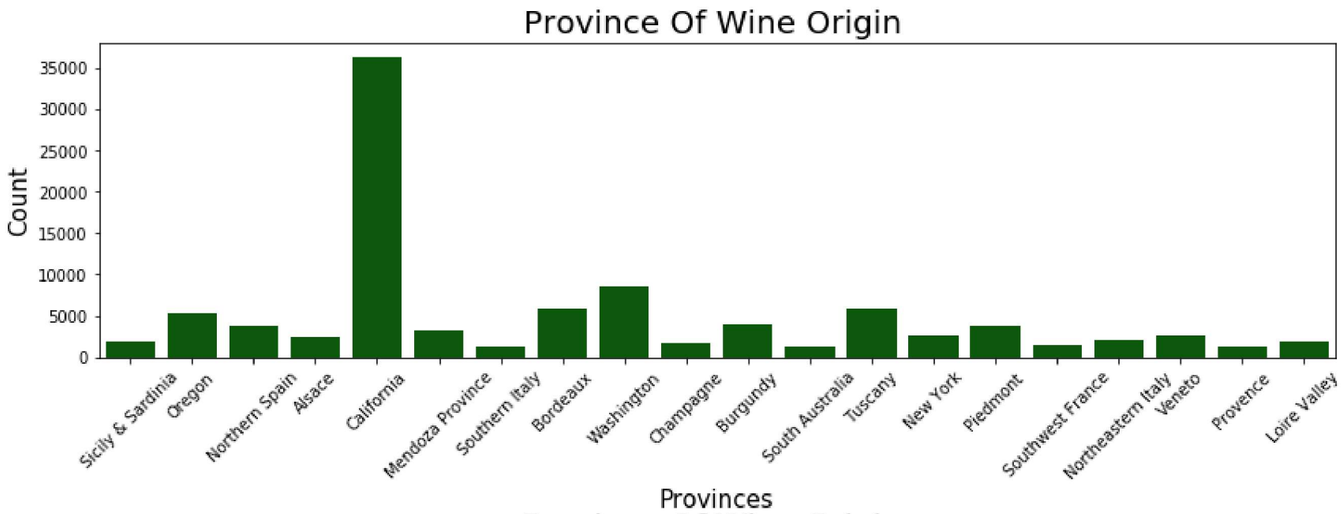


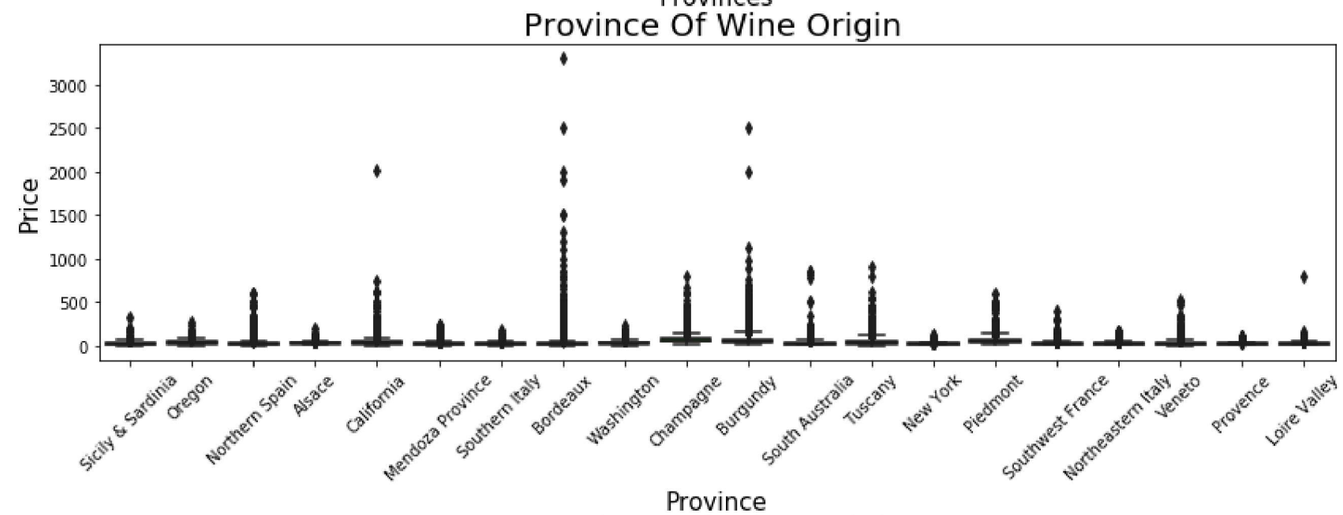
It's very interesting that all wines have quartiles in a values lower than 100

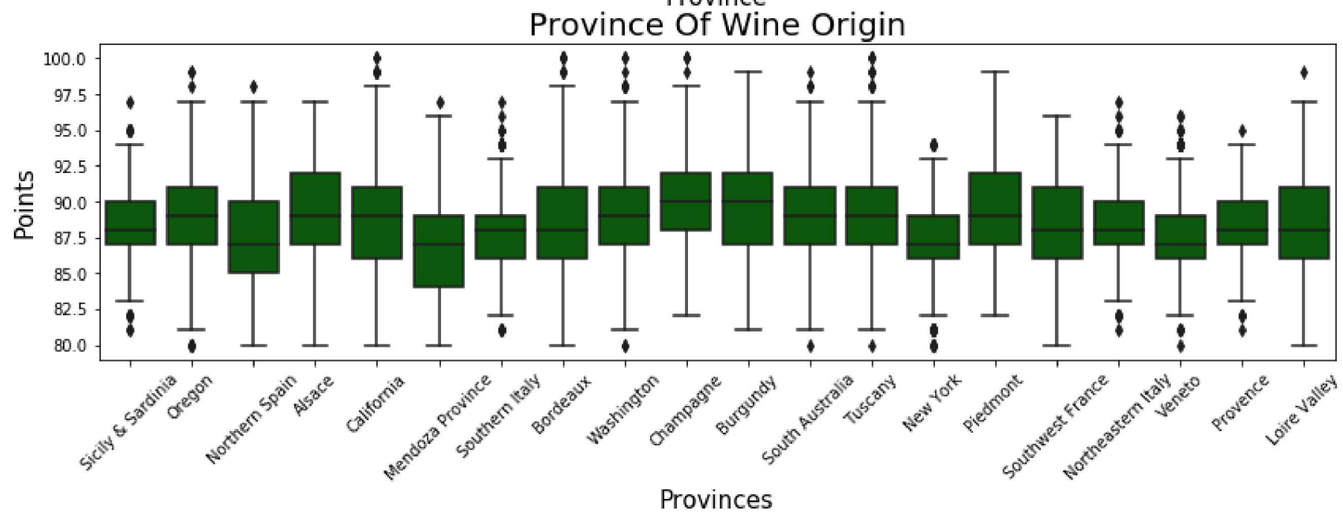
Province Exploration

In [18]:
```
plt.figure(figsize=(14,15))

provinces = df_wine1['province'].value_counts()[:20]

plt.subplot(3,1,1)
g = sns.countplot(x='province',
                  data=df_wine1.loc[(df_wine1.province.isin(provinces.index.values))],
                  color='darkgreen')
g.set_title("Province Of Wine Origin ", fontsize=20)
g.set_xlabel("Provinces", fontsize=15)
g.set_ylabel("Count", fontsize=15)
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.subplot(3,1,2)
g1 = sns.boxplot(y='price', x='province',
                  data=df_wine1.loc[(df_wine1.province.isin(provinces.index.values))],
                  color='darkgreen')
g1.set_title("Province Of Wine Origin ", fontsize=20)
g1.set_xlabel("Province", fontsize=15)
g1.set_ylabel("Price", fontsize=15)
g1.set_xticklabels(g1.get_xticklabels(),rotation=45)
```

```
plt.subplot(3,1,3)
g2 = sns.boxplot(y='points', x='province',
                 data=df_wine1.loc[(df_wine1.province.isin(provinces.index.values))],
                 color='darkgreen')
g2.set_title("Province Of Wine Origin", fontsize=20)
g2.set_xlabel("Provinces", fontsize=15)
g2.set_ylabel("Points", fontsize=15)
g2.set_xticklabels(g2.get_xticklabels(),rotation=45)

plt.subplots_adjust(hspace = 0.6,top = 0.9)

plt.show()
```

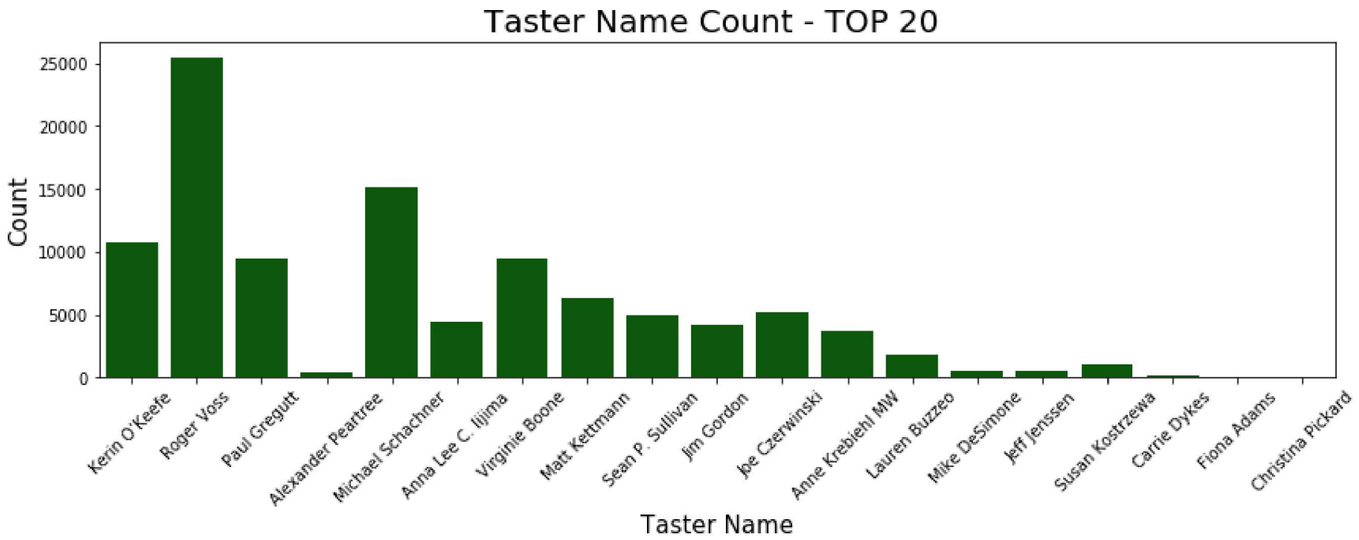

Province Of Wine Origin
Count
0
5000
10000
15000
20000
25000
30000
35000
Sicily & Sardinia
Oregon
Northern Spain
Alsace
California
Mendoza Province
Southern Italy
Bordeaux
Washington
Champagne
Burgundy
South Australia
Tuscany
New York
Piedmont
Southwest France
Northeastern Italy
Veneto
Provence
Loire Valley
Provinces

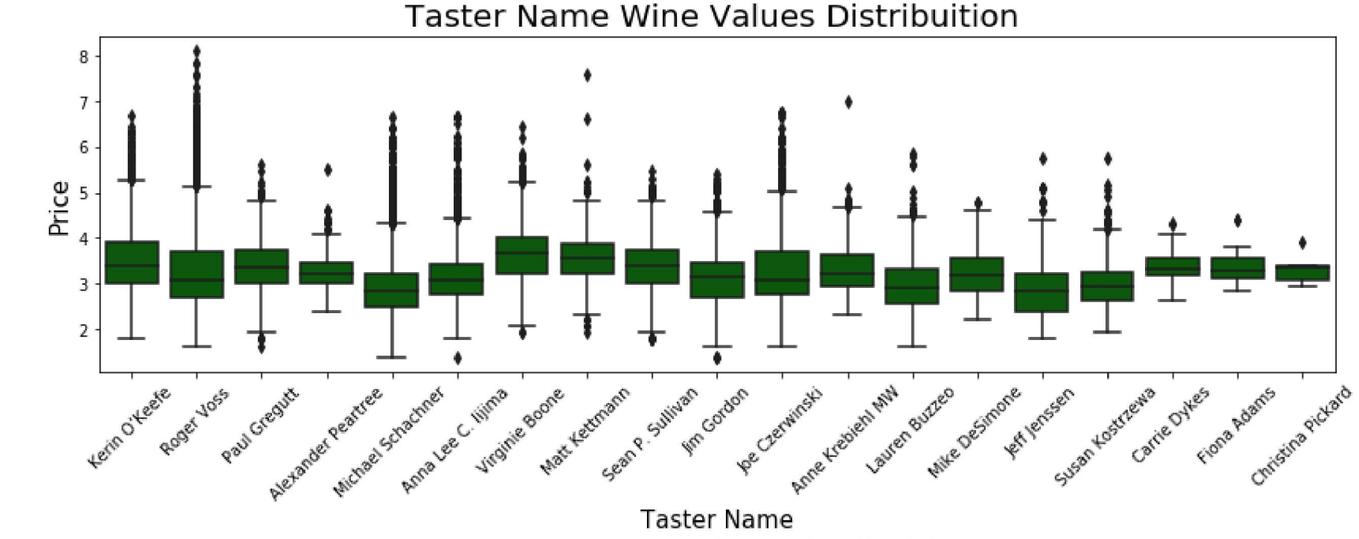

Province Of Wine Origin
Price
0
500
1000
1500
2000
2500
3000
Sicily & Sardinia
Oregon
Northern Spain
Alsace
California
Mendoza Province
Southern Italy
Bordeaux
Washington
Champagne
Burgundy
South Australia
Tuscany
New York
Piedmont
Southwest France
Northeastern Italy
Veneto
Provence
Loire Valley
Province

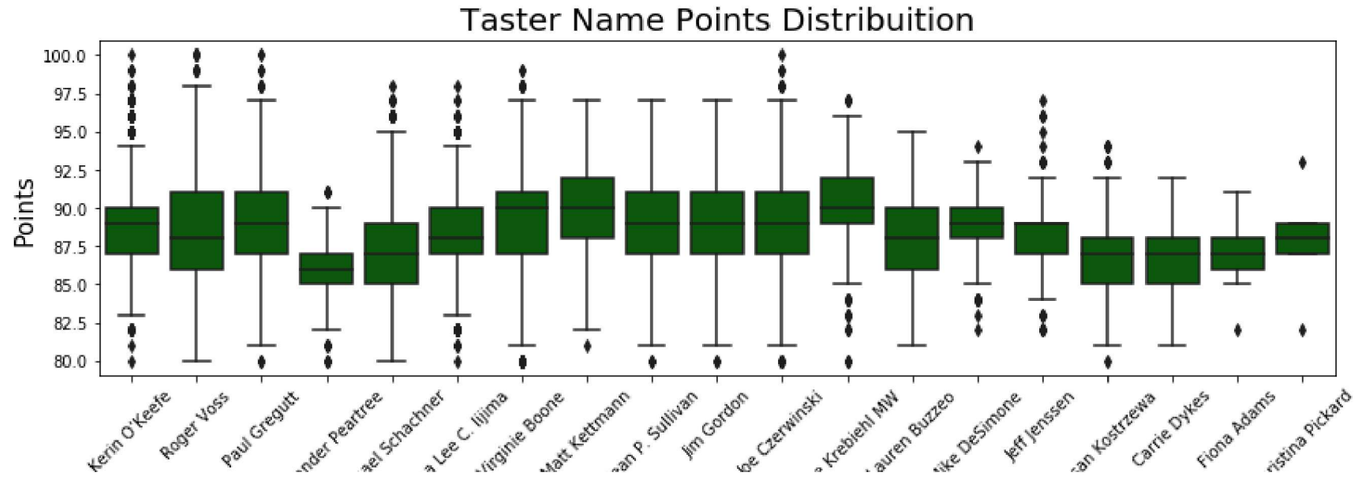

Province Of Wine Origin
Points
80.0
82.5
85.0
87.5
90.0
92.5
95.0
97.5
100.0
Sicily & Sardinia
Oregon
Northern Spain
Alsace
California
Mendoza Province
Southern Italy
Bordeaux
Washington
Champagne
Burgundy
South Australia
Tuscany
New York
Piedmont
Southwest France
Northeastern Italy
Veneto
Provence
Loire Valley
Provinces

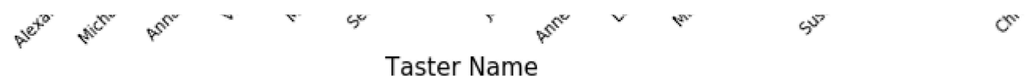
Taster Feature

In [19]:
```
plt.figure(figsize=(14,16))

provinces = df_wine1['province'].value_counts()[:20]

plt.subplot(3,1,1)
g = sns.countplot(x='taster_name', data=df_wine1, color='darkgreen')
g.set_title("Taster Name Count - TOP 20 ", fontsize=20)
g.set_xlabel("Taster Name", fontsize=15)
g.set_ylabel("Count", fontsize=15)
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.subplot(3,1,2)
g1 = sns.boxplot(y='price_log', x='taster_name', data=df_wine1,
                 color='darkgreen')
g1.set_title("Taster Name Wine Values Distribuition ", fontsize=20)
g1.set_xlabel("Taster Name", fontsize=15)
g1.set_ylabel("Price", fontsize=15)
g1.set_xticklabels(g1.get_xticklabels(),rotation=45)

plt.subplot(3,1,3)
g2 = sns.boxplot(y='points', x='taster_name',
                  data=df_wine1, color='darkgreen')
g2.set_title("Taster Name Points Distribuition", fontsize=20)
g2.set_xlabel("Taster Name", fontsize=15)
g2.set_ylabel("Points", fontsize=15)
g2.set_xticklabels(g2.get_xticklabels(),rotation=45)

plt.subplots_adjust(hspace = 0.6,top = 0.9)

plt.show()
```


Taster Name Count - TOP 20
Count
25000
20000
15000
10000
5000
0
Kerin O'Keefe
Roger Voss
Paul Gregutt
Alexander Peartree
Michael Schachner
Anna Lee C. Iijima
Virginie Boone
Matt Kettmann
Sean P. Sullivan
Jim Gordon
Joe Czerwinski
Anne Krebiehl MW
Lauren Buzzeo
Mike DeSimone
Jeff Jenssen
Susan Kostrzewa
Carrie Dykes
Fiona Adams
Christina Pickard
Taster Name

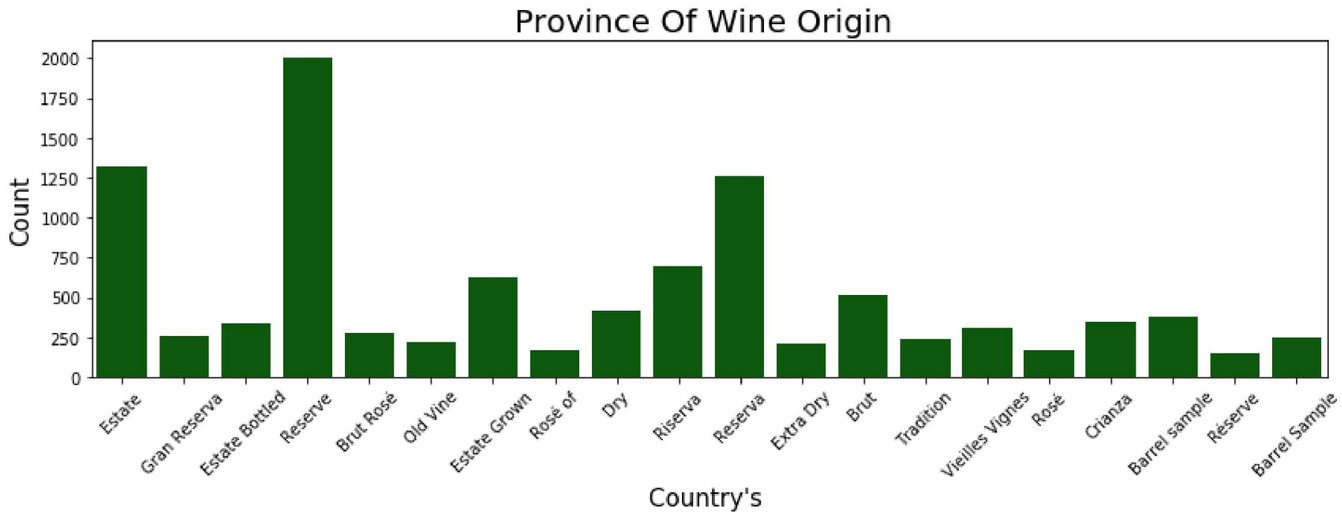

Taster Name Wine Values Distribuition
Price
8
7
6
5
4
3
2
Kerin O'Keefe
Roger Voss
Paul Gregutt
Alexander Peartree
Michael Schachner
Anna Lee C. Iijima
Virginie Boone
Matt Kettmann
Sean P. Sullivan
Jim Gordon
Joe Czerwinski
Anne Krebiehl MW
Lauren Buzzeo
Mike DeSimone
Jeff Jenssen
Susan Kostrzewa
Carrie Dykes
Fiona Adams
Taster Name

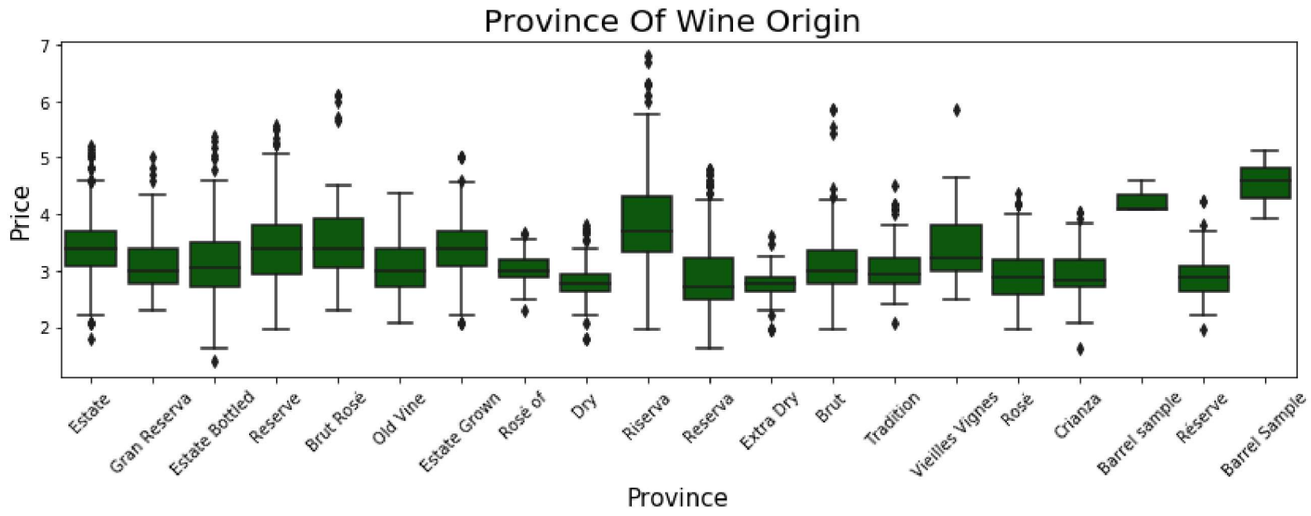

Taster Name Points Distribuition
Points
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
Kerin O'Keefe
Roger Voss
Paul Gregutt
Jim Gordon
Carrie Dykes
Fiona Adams

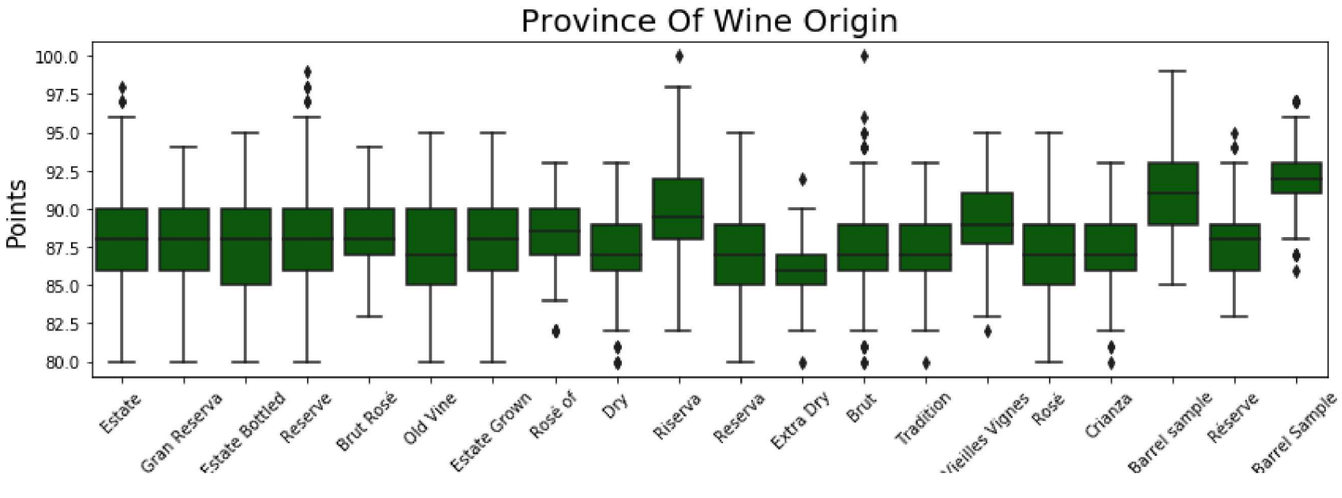


PROVINVCE FEATURE

In [20]:
```
plt.figure(figsize=(14,16))

designation = df_wine1.designation.value_counts()[:20]

plt.subplot(3,1,1)
g = sns.countplot(x='designation',
                  data=df_wine1.loc[(df_wine1.designation.isin(designation.index.values))],
                  color='darkgreen')
g.set_title("Province Of Wine Origin ", fontsize=20)
g.set_xlabel("Country's ", fontsize=15)
g.set_ylabel("Count", fontsize=15)
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.subplot(3,1,2)
g1 = sns.boxplot(y='price_log', x='designation',
                  data=df_wine1.loc[(df_wine1.designation.isin(designation.index.values))],
                 color='darkgreen')
g1.set_title("Province Of Wine Origin ", fontsize=20)
g1.set_xlabel("Province", fontsize=15)
g1.set_ylabel("Price", fontsize=15)
g1.set_xticklabels(g1.get_xticklabels(),rotation=45)

plt.subplot(3,1,3)
g2 = sns.boxplot(y='points', x='designation',
                  data=df_wine1.loc[(df_wine1.designation.isin(designation.index.values))],
                 color='darkgreen')
g2.set_title("Province Of Wine Origin", fontsize=20)
g2.set_xlabel("Provinces", fontsize=15)
g2.set_ylabel("Points", fontsize=15)
g2.set_xticklabels(g2.get_xticklabels(),rotation=45)

plt.subplots_adjust(hspace = 0.6,top = 0.9)

plt.show()
```


Province Of Wine Origin
Count
2000
1750
1500
1250
1000
750
500
250
0
Estate
Gran Reserva
Estate Bottled
Reserve
Brut Rosé
Old Vine
Estate Grown
Rosé of
Dry
Riserva
Reserva
Extra Dry
Brut
Tradition
Vieilles Vignes
Rosé
Crianza
Barrel sample
Réserve
Barrel Sample
Country's

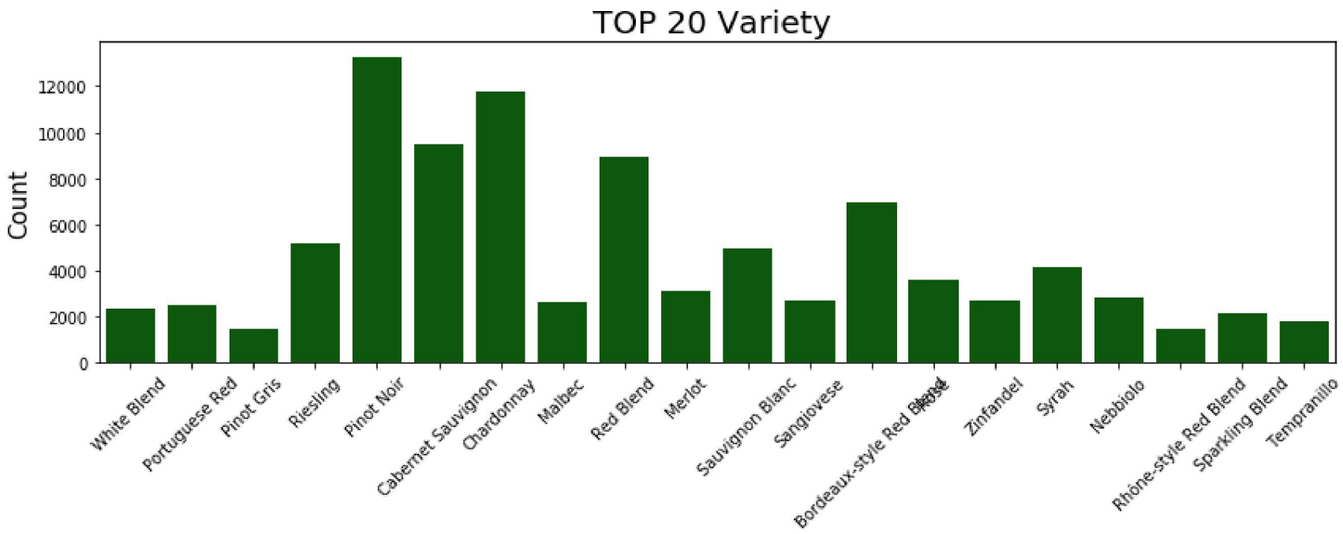

Province Of Wine Origin
Price
7
6
5
4
3
2
Estate
Gran Reserva
Estate Bottled
Reserve
Brut Rosé
Old Vine
Estate Grown
Rosé of
Dry
Riserva
Reserva
Extra Dry
Brut
Tradition
Vieilles Vignes
Rosé
Crianza
Barrel sample
Réserve
Barrel Sample
Province

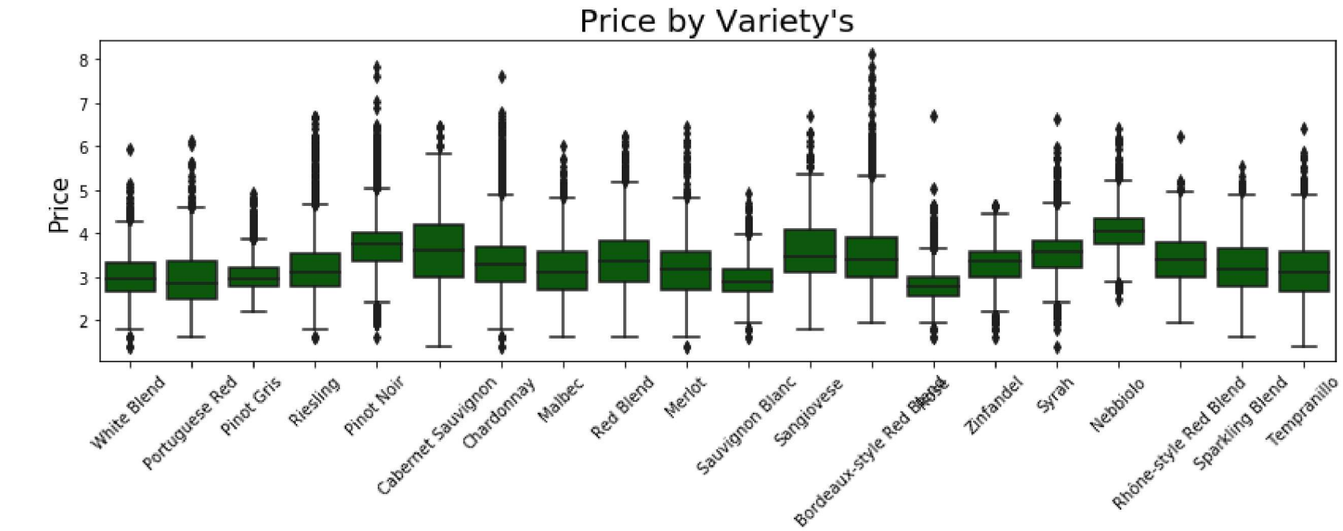

Province Of Wine Origin
Points
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
Estate
Gran Reserva
Estate Bottled
Reserve
Brut Rosé
Old Vine
Estate Grown
Rosé of
Dry
Riserva
Reserva
Extra Dry
Brut
Tradition
Vieilles Vignes
Rosé
Crianza
Barrel sample
Réserve
Barrel Sample

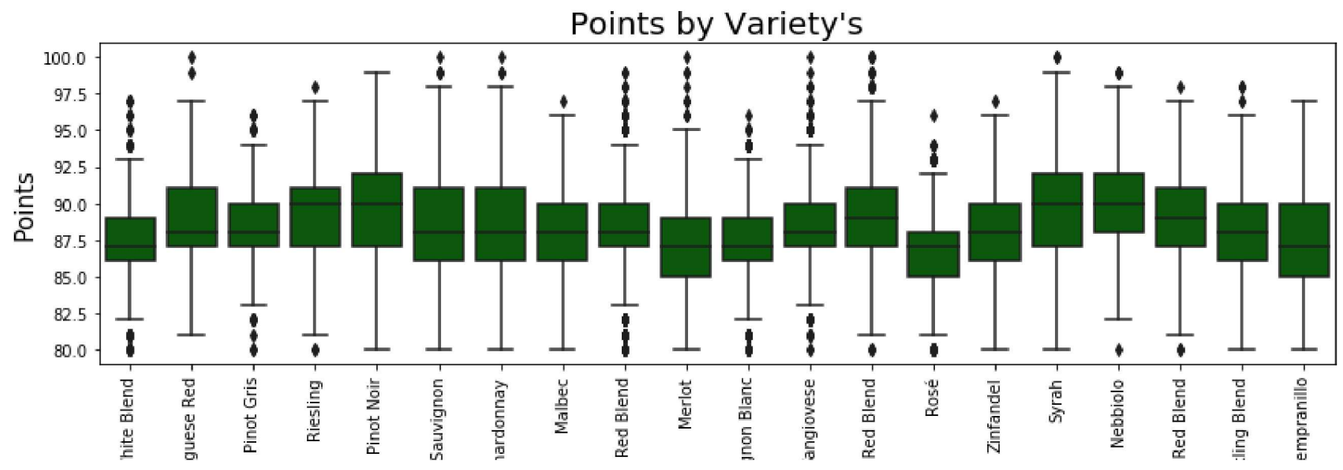
Provinces

VARIETY FEATURE

In [21]:
```
plt.figure(figsize=(14,16))

variety = df_wine1.variety.value_counts()[:20]

plt.subplot(3,1,1)
g = sns.countplot(x='variety',
                  data=df_wine1.loc[(df_wine1.variety.isin(variety.index.values))],
                  color='darkgreen')
g.set_title("TOP 20 Variety ", fontsize=20)
g.set_xlabel(" ", fontsize=15)
g.set_ylabel("Count", fontsize=15)
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.subplot(3,1,2)
g1 = sns.boxplot(y='price_log', x='variety',
                  data=df_wine1.loc[(df_wine1.variety.isin(variety.index.values))],
                 color='darkgreen')
g1.set_title("Price by Variety's", fontsize=20)
g1.set_xlabel("", fontsize=15)
g1.set_ylabel("Price", fontsize=15)
g1.set_xticklabels(g1.get_xticklabels(),rotation=45)

plt.subplot(3,1,3)
g2 = sns.boxplot(y='points', x='variety',
                  data=df_wine1.loc[(df_wine1.variety.isin(variety.index.values))],
                 color='darkgreen')
g2.set_title("Points by Variety's", fontsize=20)
g2.set_xlabel("Variety's", fontsize=15)
g2.set_ylabel("Points", fontsize=15)
g2.set_xticklabels(g2.get_xticklabels(),rotation=90)

plt.subplots_adjust(hspace = 0.7,top = 0.9)

plt.show()
```


TOP 20 Variety
Count
0
2000
4000
6000
8000
10000
12000
White Blend
Portuguese Red
Pinot Gris
Riesling
Pinot Noir
Cabernet Sauvignon
Chardonnay
Malbec
Red Blend
Merlot
Sauvignon Blanc
Sangiovese
Bordeaux-style Red Blend
Zinfandel
Syrah
Nebbiolo
Rhône-style Red Blend
Sparkling Blend
Tempranillo


Price by Variety's
Price
2
3
4
5
6
7
8
White Blend
Portuguese Red
Pinot Gris
Riesling
Pinot Noir
Cabernet Sauvignon
Chardonnay
Malbec
Red Blend
Merlot
Sauvignon Blanc
Sangiovese
Red Blend
Rosé
Zinfandel
Syrah
Nebbiolo
Red Blend
Sparkling Blend
Tempranillo


Points by Variety's
Points
80.0
82.5
85.0
87.5
90.0
92.5
95.0
97.5
100.0
hite Blend
guese Red
Pinot Gris
Riesling
Pinot Noir
Sauvignon
hardonnay
Malbec
Red Blend
Merlot
gnon Blanc
angiovese
Red Blend
Rosé
Zinfandel
Syrah
Nebbiolo
Red Blend
ling Blend
empranillo

Let's take a look at Winery Distribuitions

Generate a word cloud image

mask = np.array(Image.open("img/france.png")) wordcloud_fra = WordCloud(stopwords=stopwords, background_color="white", mode="RGBA", max_words=5200, mask=mask).generate(fra)

create coloring from image

image_colors = ImageColorGenerator(mask) plt.figure(figsize=[7,7]) plt.imshow(wordcloud_fra.recolor(color_func=image_colors), interpolation="bilinear") plt.axis("off")

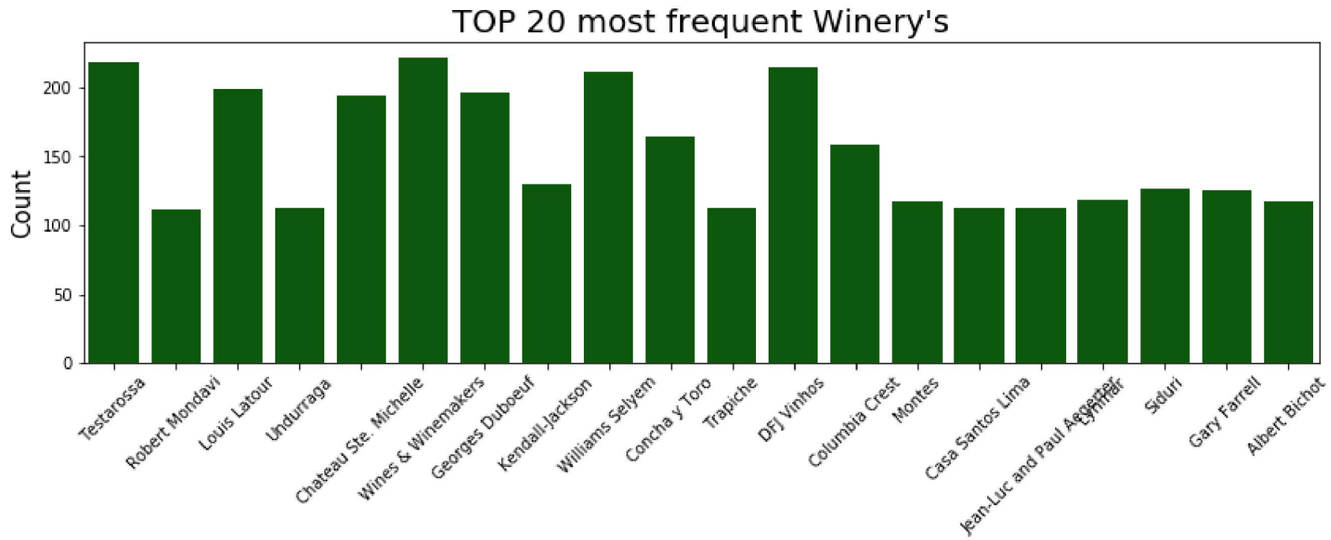
store to file

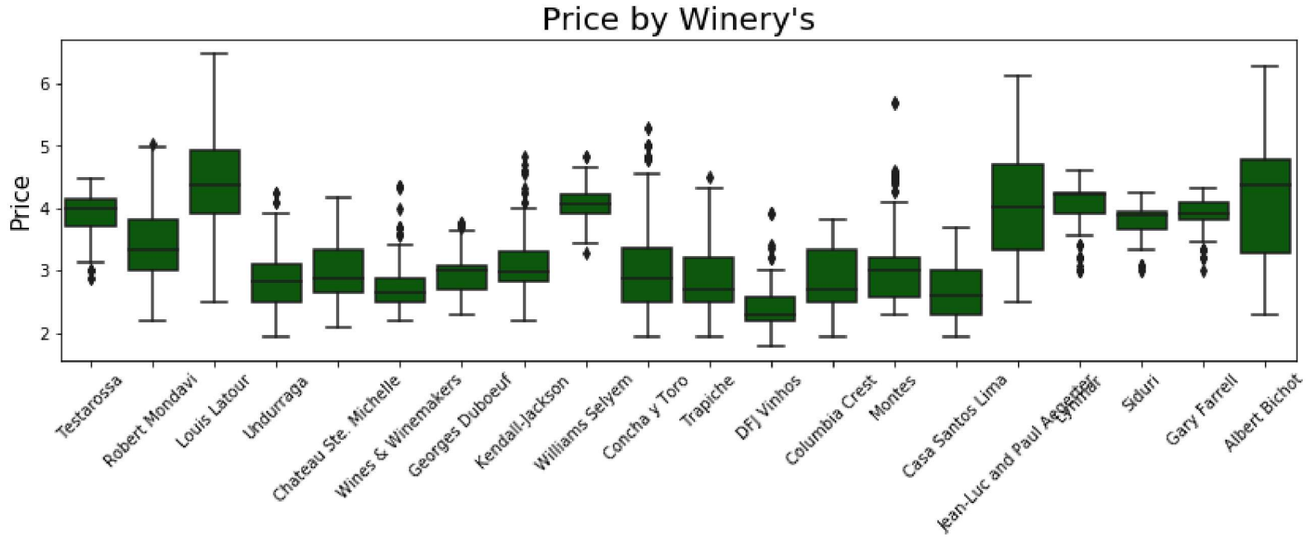
plt.savefig("img/fra_wine.png", format="png")

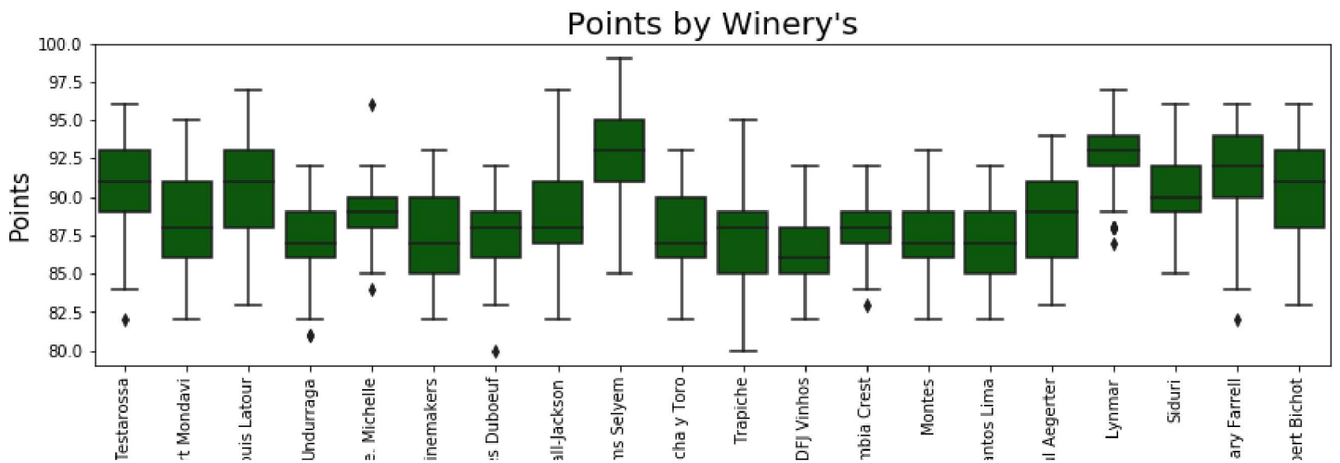
plt.show()

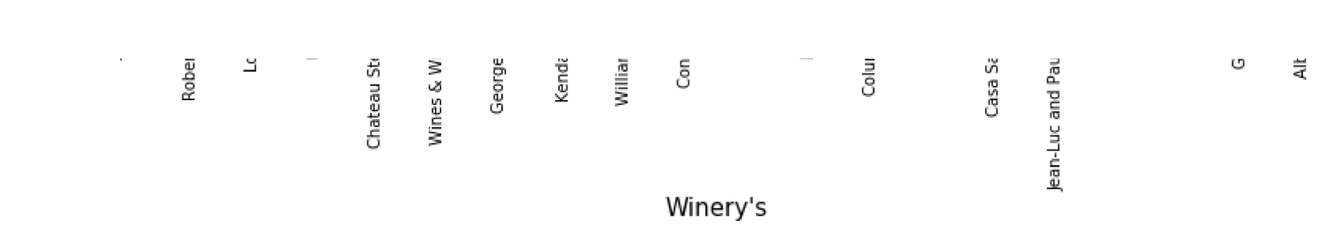
In [22]:
```
plt.figure(figsize=(14,16))

winery = df_wine1.winery.value_counts()[:20]

plt.subplot(3,1,1)
g = sns.countplot(x='winery',
                  data=df_wine1.loc[(df_wine1.winery.isin(winery.index.values))],
                  color='darkgreen')
g.set_title("TOP 20 most frequent Winery's", fontsize=20)
g.set_xlabel(" ", fontsize=15)
g.set_ylabel("Count", fontsize=15)
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.subplot(3,1,2)
g1 = sns.boxplot(y='price_log', x='winery',
                  data=df_wine1.loc[(df_wine1.winery.isin(winery.index.values))],
                 color='darkgreen')
g1.set_title("Price by Winery's", fontsize=20)
g1.set_xlabel("", fontsize=15)
g1.set_ylabel("Price", fontsize=15)
g1.set_xticklabels(g1.get_xticklabels(),rotation=45)

plt.subplot(3,1,3)
g2 = sns.boxplot(y='points', x='winery',
                  data=df_wine1.loc[(df_wine1.winery.isin(winery.index.values))],
                 color='darkgreen')
g2.set_title("Points by Winery's", fontsize=20)
g2.set_xlabel("Winery's", fontsize=15)
```

```
g2.set_ylabel("Points", fontsize=15)
g2.set_xticklabels(g2.get_xticklabels(),rotation=90)

plt.subplots_adjust(hspace = 0.7,top = 0.9)

plt.show()
```


TOP 20 most frequent Winery's
Count
0
50
100
150
200
Testarossa
Robert Mondavi
Louis Latour
Undurraga
Chateau Ste. Michelle
Wines & Winemakers
Georges Duboeuf
Kendall-Jackson
Williams Selyem
Concha y Toro
Trapiche
DFJ Vinhos
Columbia Crest
Montes
Casa Santos Lima
Jean-Luc and Paul Aegerter
Lynmar
Siduri
Gary Farrell
Albert Bichot


Price by Winery's
Price
2
3
4
5
6
Testarossa
Robert Mondavi
Louis Latour
Undurraga
Chateau Ste. Michelle
Wines & Winemakers
Georges Duboeuf
Kendall-Jackson
Williams Selyem
Concha y Toro
Trapiche
DFJ Vinhos
Columbia Crest
Montes
Casa Santos Lima
Jean-Luc and Paul Aegerter
Lynmar
Siduri
Gary Farrell
Albert Bichot


Points by Winery's
Points
80.0
82.5
85.0
87.5
90.0
92.5
95.0
97.5
100.0
Testarossa
Lynmar
Siduri

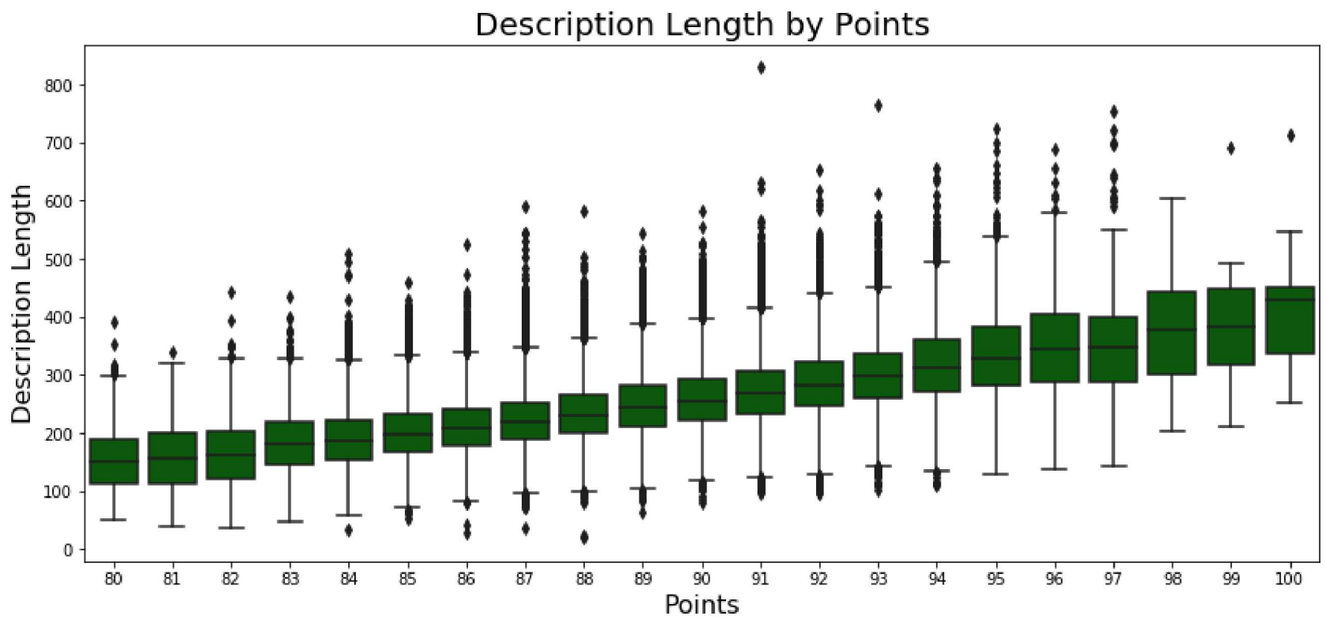


We can see that some winery's have +- 200 label in his portfolio's.

Also, we can verify that Willians Selyem have highest points distribuition

Better understanding the lenght of reviews

In [23]:
```
df_wine1 = df_wine1.assign(desc_length = df_wine1['description'].apply(len))

plt.figure(figsize=(14,6))
g = sns.boxplot(x='points', y='desc_length', data=df_wine1,
                color='darkgreen')
g.set_title('Description Length by Points', fontsize=20)
g.set_ylabel('Description Length', fontsize = 16) # Y label
g.set_xlabel('Points', fontsize = 16) # X label
plt.show()
```

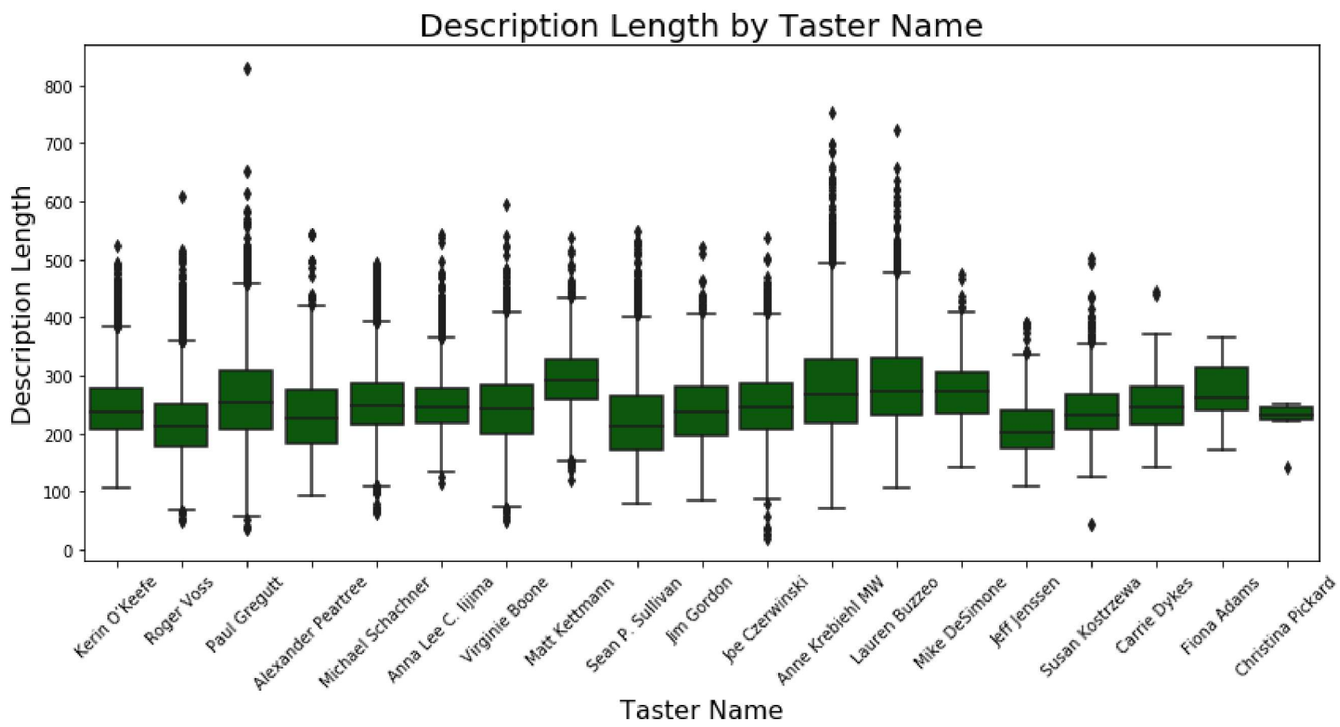


We can see that the wine's with highest points also have the biggest descriptions length.

Who are the sommeliers with biggest descriptions length?

In [24]:
```
plt.figure(figsize=(14,6))

g = sns.boxplot(x='taster_name', y='desc_length',
                data=df_wine1, color='darkgreen')
```

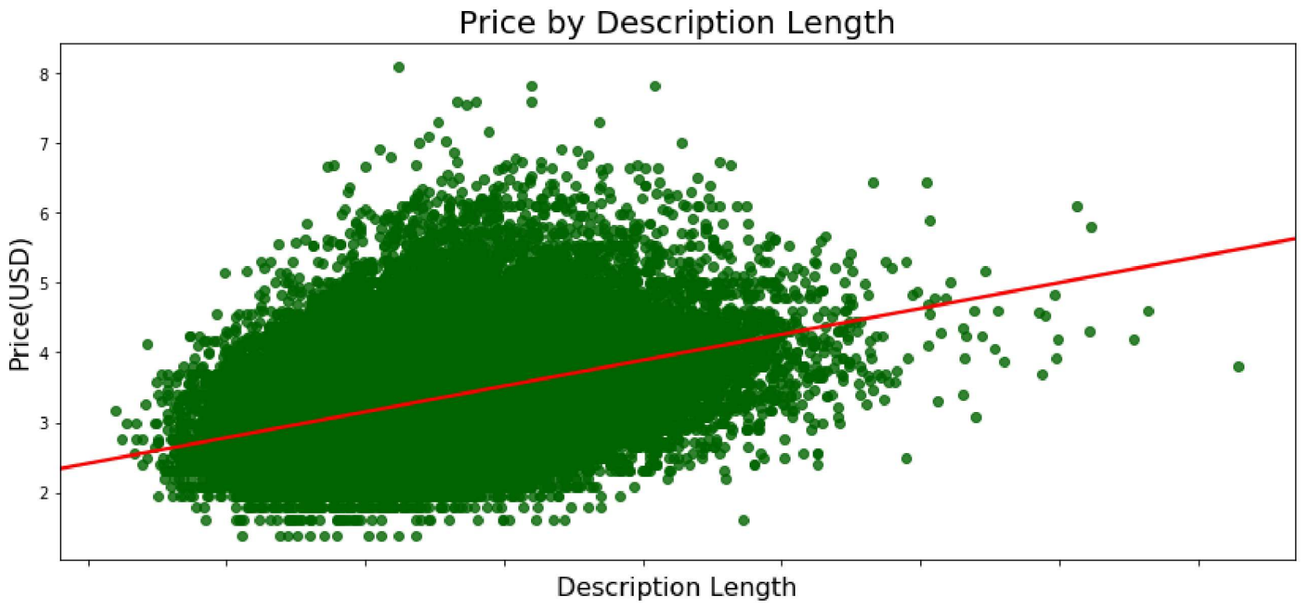
```
g.set_title('Description Length by Taster Name', fontsize=20)
g.set_ylabel('Description Length', fontsize = 16) # Y Label
g.set_xlabel('Taster Name', fontsize = 16) # X Label
g.set_xticklabels(g.get_xticklabels(),rotation=45)
plt.show()
```



we can see a difference description length per taster name

Scatter plot of the description length and the price

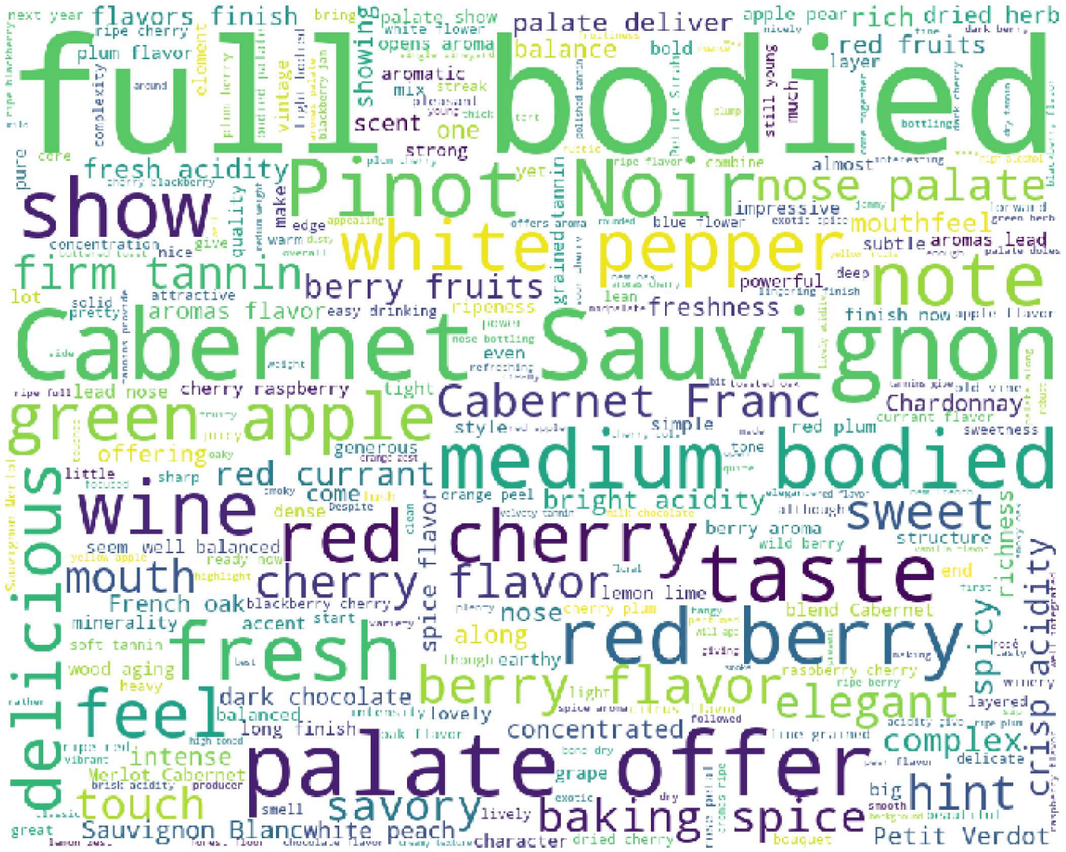
In [25]:
```
plt.figure(figsize=(14,6))

g = sns.regplot(x='desc_length', y='price_log', line_kws={'color':'red'},
                data=df_wine1, fit_reg=True, color='darkgreen', )
g.set_title('Price by Description Length', fontsize=20)
g.set_ylabel('Price(USD)', fontsize = 16)
g.set_xlabel('Description Length', fontsize = 16)
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.show()
```



WORDCLOUDS OF DESCRIPTIONS

In [26]:
```
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)

newStopWords = ['fruit', "Drink", "black", 'wine', 'drink']

stopwords.update(newStopWords)

wordcloud = WordCloud(
    background_color='white',
    stopwords=stopwords,
    max_words=300,
    max_font_size=200,
    width=1000, height=800,
    random_state=42,
).generate(" ".join(df_wine1['description'].astype(str)))

print(wordcloud)
fig = plt.figure(figsize = (12,14))
plt.imshow(wordcloud)
plt.title("WORD CLOUD - DESCRIPTION",fontsize=25)
plt.axis('off')
plt.show()
```

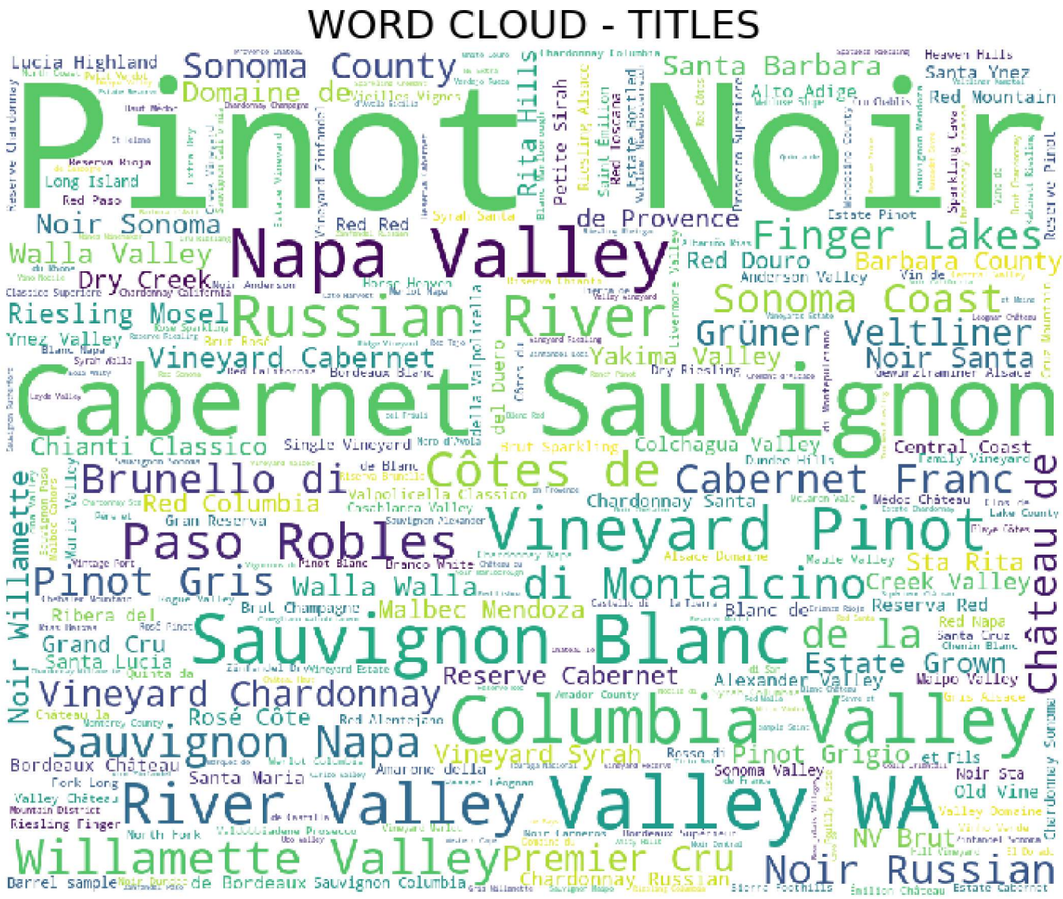
```
<wordcloud.wordcloud.WordCloud object at 0x7d667c583b70>
```



WORDCLOUD OF WINE TITLES

In [27]:
```
wordcloud = WordCloud(
    background_color='white',
    stopwords=stopwords,
    max_words=300,
    max_font_size=200,
    width=1000, height=800,
    random_state=42,
).generate(" ".join(df_wine1['title'].astype(str)))

print(wordcloud)
fig = plt.figure(figsize = (12,14))
plt.imshow(wordcloud)
plt.title("WORD CLOUD - TITLES",fontsize=25)
plt.axis('off')
plt.show()
```

```
<wordcloud.wordcloud.WordCloud object at 0x7d667c7f2390>
```

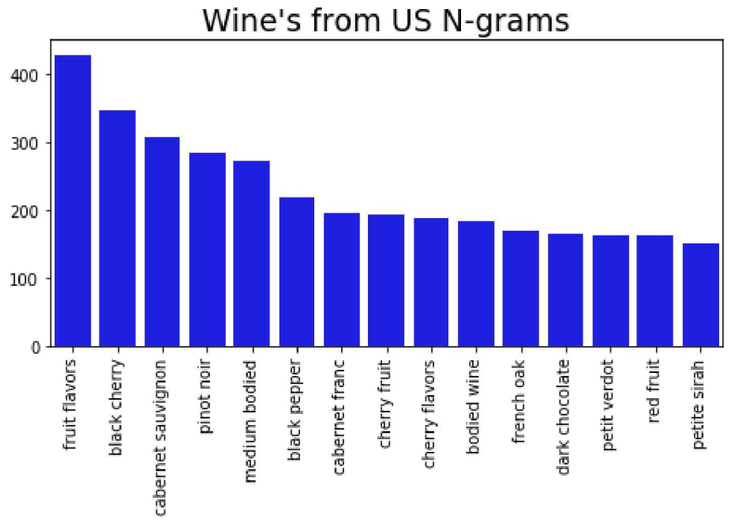


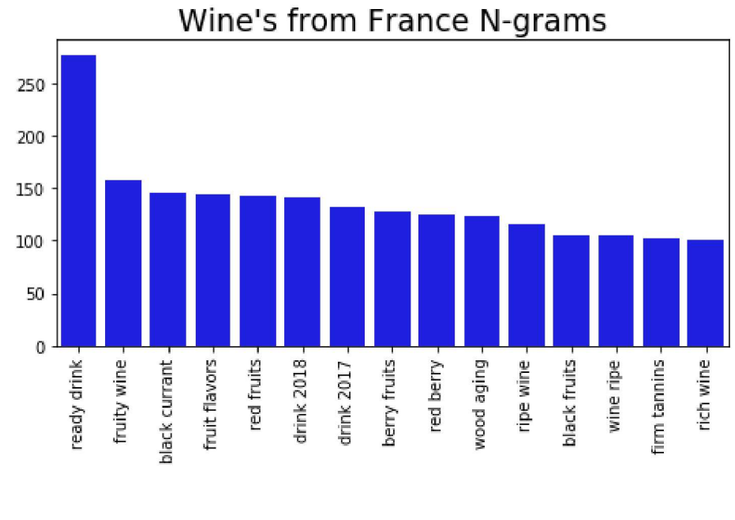
TFIDF Vectorizer in the Wine Review

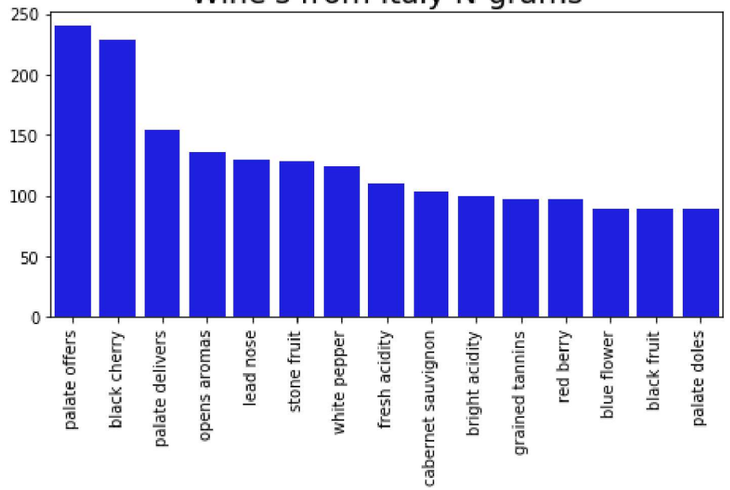
Description N-gram

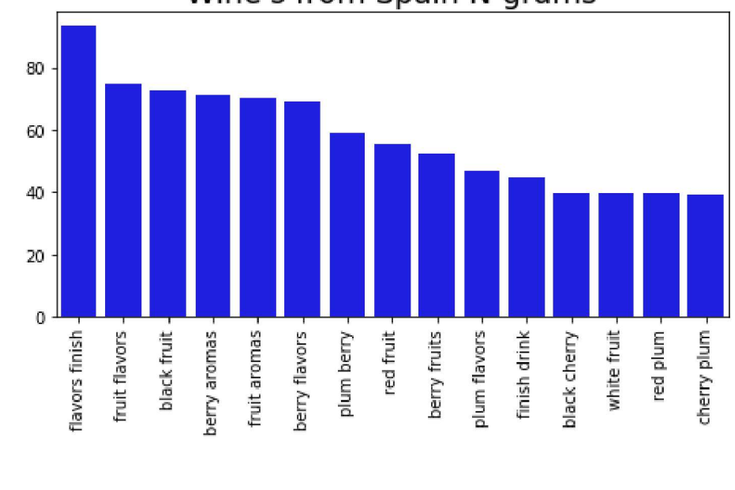
In [28]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import matplotlib.gridspec as gridspec # to do the grid of plots

grid = gridspec.GridSpec(5, 2)
plt.figure(figsize=(16,7*4))

for n, cat in enumerate(country.index[:10]):

    ax = plt.subplot(grid[n])
    # print(f'PRINCIPAL WORDS CATEGORY: {cat}')
    # vectorizer = CountVectorizer(ngram_range = (3,3))
    # X1 = vectorizer.fit_transform(df_train[df_train['host_cat'] == cat]['answer'])
    # print(cat)
    # Applying TFIDF
    vectorizer = TfidfVectorizer(ngram_range = (2, 3), min_df=5,
                                 stop_words='english',
                                 max_df=.5)
```

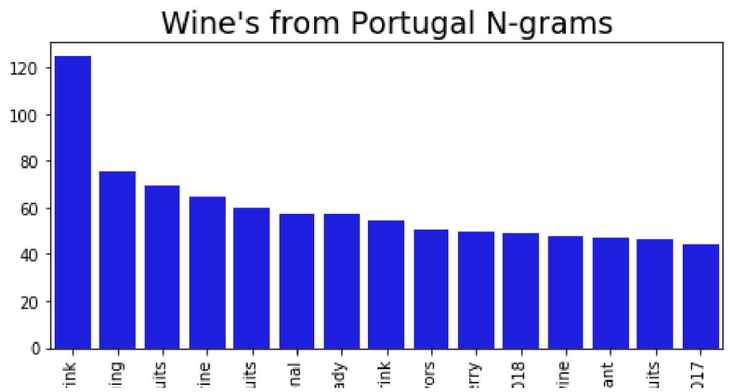
```
    X2 = vectorizer.fit_transform(df_wine1.loc[(df_wine1.country == cat)]['description'])
    features = (vectorizer.get_feature_names())
    scores = (X2.toarray())

    # Getting top ranking features
    sums = X2.sum(axis = 0)
    data1 = []

    for col, term in enumerate(features):
        data1.append( (term, sums[0,col] ))

    ranking = pd.DataFrame(data1, columns = ['term','rank'])
    words = (ranking.sort_values('rank', ascending = False))[:15]

    sns.barplot(x='term', y='rank', data=words, ax=ax,
                color='blue', orient='v')
    ax.set_title(f"Wine's from {cat} N-grams", fontsize=19)
    ax.set_xticklabels(ax.get_xticklabels(),rotation=90)
    ax.set_ylabel(' ')
    ax.set_xlabel(" ")

plt.subplots_adjust(top = 0.95, hspace=.9, wspace=.1)

plt.show()
```

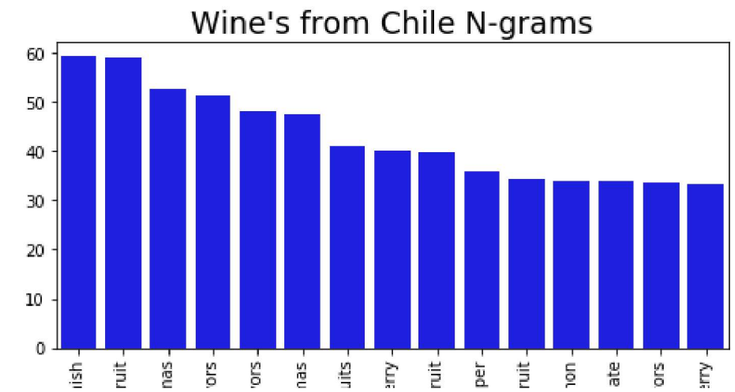

Wine's from US N-grams
0
100
200
300
400
fruit flavors
black cherry
cabernet sauvignon
pinot noir
medium bodied
black pepper
cabernet franc
cherry fruit
cherry flavors
bodied wine
french oak
dark chocolate
petit verdot
red fruit
petite sirah

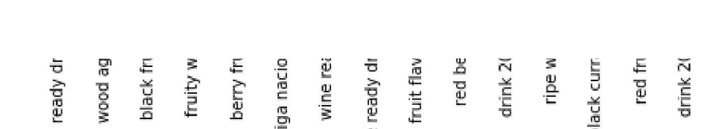

Wine's from France N-grams
0
50
100
150
200
250
ready drink
fruity wine
black currant
fruit flavors
red fruits
drink 2018
drink 2017
berry fruits
red berry
wood aging
ripe wine
black fruits
wine ripe
firm tannins
rich wine

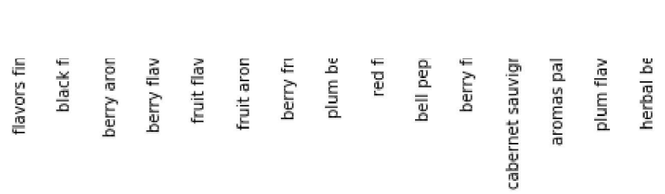

Wine's from Italy N-grams
0
50
100
150
200
250
palate offers
black cherry
palate delivers
opens aromas
lead nose
stone fruit
white pepper
fresh acidity
cabernet sauvignon
bright acidity
grained tannins
red berry
blue flower
black fruit
palate doles

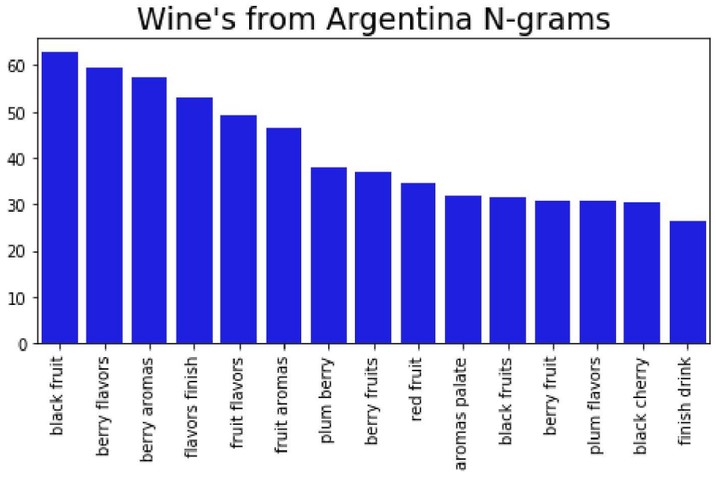

Wine's from Spain N-grams
0
20
40
60
80
flavors finish
fruit flavors
black fruit
berry aromas
fruit aromas
berry flavors
plum berry
red fruit
berry fruits
plum flavors
finish drink
black cherry
white fruit
red plum
cherry plum

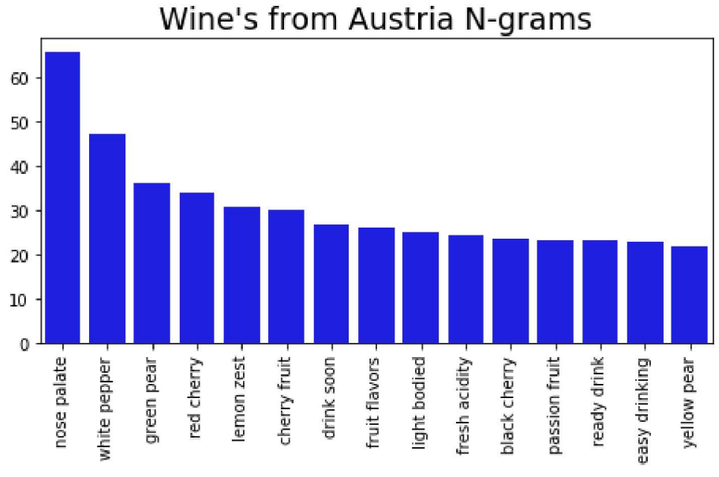

Wine's from Portugal N-grams
0
20
40
60
80
100
120

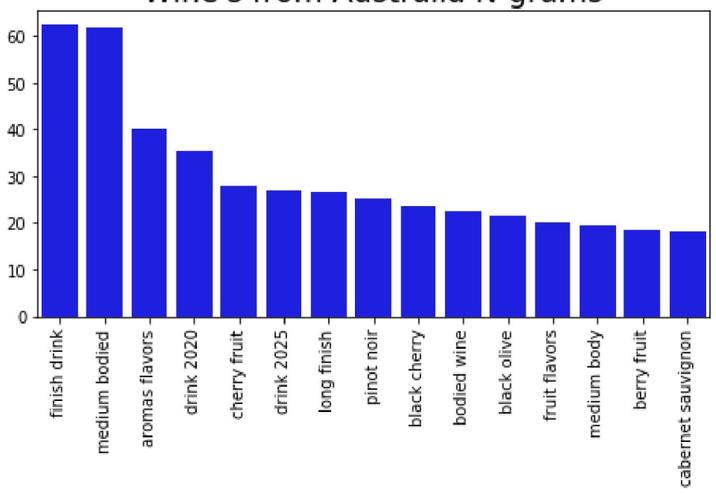

Wine's from Chile N-grams
0
10
20
30
40
50
60

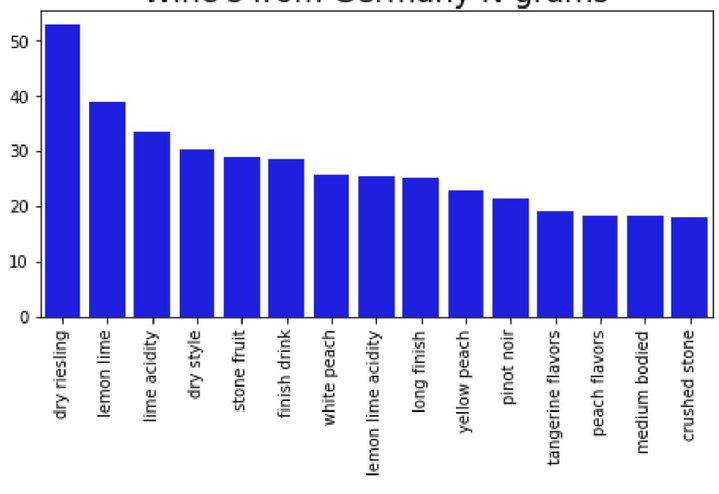












It seems a very meaningful information about the wine quality and characteristics.

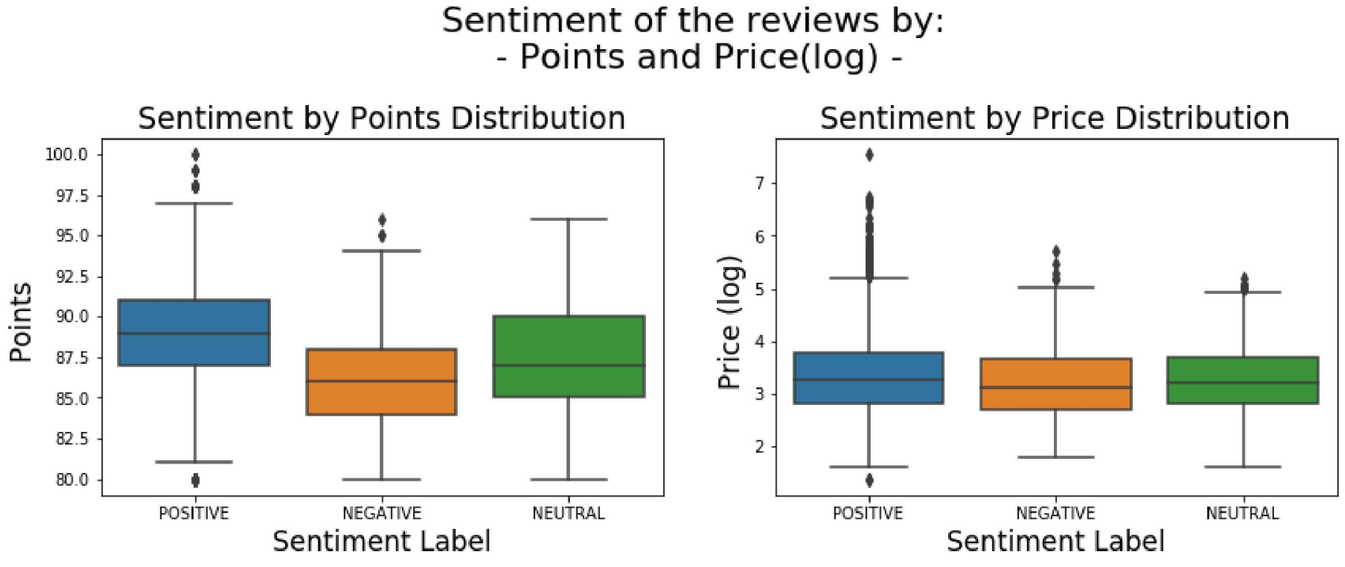
Sentiment Analysis

In [29]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

SIA = SentimentIntensityAnalyzer()

# Applying Model, Variable Creation
sentiment = df_wine1.sample(15000).copy()
sentiment['polarity_score']=sentiment.description.apply(lambda x:SIA.polarity_scores(x)['compound'])
sentiment['neutral_score']=sentiment.description.apply(lambda x:SIA.polarity_scores(x)['neu'])
sentiment['negative_score']=sentiment.description.apply(lambda x:SIA.polarity_scores(x)['neg'])
sentiment['positive_score']=sentiment.description.apply(lambda x:SIA.polarity_scores(x)['pos'])

sentiment['sentiment']= np.nan
sentiment.loc[sentiment.polarity_score>0,'sentiment']='POSITIVE'
sentiment.loc[sentiment.polarity_score==0,'sentiment']='NEUTRAL'
sentiment.loc[sentiment.polarity_score<0,'sentiment']='NEGATIVE'
```

```
/opt/conda/lib/python3.6/site-packages/nltk/twitter/__init__.py:20: UserWarning: The twython library has not been installed. Some functionality from the twitter package will not be available.
  warnings.warn("The twython library has not been installed. "
```

Ploting the sentiment labels

In [30]:
```
plt.figure(figsize=(14,5))

plt.suptitle('Sentiment of the reviews by: \n- Points and Price(log) -', size=22)

plt.subplot(121)
ax = sns.boxplot(x='sentiment', y='points', data=sentiment)
ax.set_title("Sentiment by Points Distribution", fontsize=19)
ax.set_ylabel("Points ", fontsize=17)
ax.set_xlabel("Sentiment Label", fontsize=17)

plt.subplot(122)
ax1= sns.boxplot(x='sentiment', y='price_log', data=sentiment)
ax1.set_title("Sentiment by Price Distribution", fontsize=19)
ax1.set_ylabel("Price (log) ", fontsize=17)
ax1.set_xlabel("Sentiment Label", fontsize=17)

plt.subplots_adjust(top = 0.75, wspace=.2)
plt.show()
```


Sentiment of the reviews by:
- Points and Price(log) -
Sentiment by Points Distribution
Points
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
POSITIVE
NEGATIVE
NEUTRAL
Sentiment Label
Sentiment by Price Distribution
Price (log)
7
6
5
4
3
2
POSITIVE
NEGATIVE
NEUTRAL
Sentiment Label