# Week12_Word_Clouds

April 18, 2018

## 1 Python Word Clouds

Word clouds are a visual representation of the terms contained in one document or a collection of documents, referred to as a corpus. Word clouds can be based upon term frequency counts, which is the default approach used in most software. Alternatively, word clouds can be based upon the term weight, such as TFIDF weights, which provide a better reflection of terms that are considered more important in identifying topic collections. That is, in identifying the topics that reflect the document contents.

In both cases tokenization, stemming and removing stop words is important. Word clouds should reflect the important terms and not the terms that are most common and contain little information.

In Python, one of the most popular packages for constructing word clouds is wordcloud authored by Andreas Mueller. In its basic form, it is very easy to use, but in more sophisticated applications, it requires a better understanding of other Python graphic packages.

#### 1.0.1 Installing WordCloud

The wordcloud package is not included in the basic Anaconda package. However, it's easily installed using the CONDA command:

conda install -c conda-forge wordcloud

This command will install wordcloud and its supporting package. In total, the packages installed are:

```
package                    |            build
---------------------------|-----------------
openssl-1.0.2o             |                0         3.3 MB  conda-forge
ca-certificates-2018.1.18  |                0         141 KB  conda-forge
wordcloud-1.4.1            |          py36_0          156 KB  conda-forge
certifi-2018.1.18          |          py36_0          143 KB  conda-forge
------------------------------------------------------------
                                       Total:         3.7 MB
```

These versions are as of early 2018, and of course they may change with package updates.

#### 1.0.2 Import Packages

The import statements used in this example include wordcloud as well as other used in text analytics. The four statements at the end are the new imports used to produce word clouds.

```
In [1]: import pandas as pd
        import numpy  as np
        import string
        from time import time
        from nltk import pos_tag
        from nltk.tokenize import word_tokenize
        from nltk.stem.snowball import SnowballStemmer
        from nltk.stem import WordNetLemmatizer
        from nltk.corpus import wordnet as wn
        from nltk.corpus import stopwords
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.feature_extraction.text import TfidfTransformer
        from sklearn.decomposition import LatentDirichletAllocation
        from sklearn.decomposition import TruncatedSVD
        from sklearn.decomposition import NMF

        import matplotlib.pyplot as plt
        from wordcloud import WordCloud, STOPWORDS
        from PIL import Image
        import random
```

### 1.0.3 Wine Reviews - The Data

The description of words clouds will use the reviews of California Cabernet. These data consist of 13,575 short reviews.

```
In [2]: # Increase column width to let pandy read large text columns
        pd.set_option('max_colwidth', 32000)
        # Read N=13,575 California Cabernet Savignon Reviews
        file_path = '/Users/Home/Desktop/python/Excel/'
        df = pd.read_excel(file_path+"CaliforniaCabernet.xlsx")

        # Setup program constants and reviews
        n_reviews  = len(df['description'])
        s_words    = 'english'
        ngram = (1,2)
        reviews = df['description']
```

### 1.0.4 A simple word cloud

The individual reviews are in the list reviews. Displaying a word cloud for any of these reviews is done by passing the string representing that review to the wordcloud class. In this example a word cloud is generated for the first review contained in reviews[0].

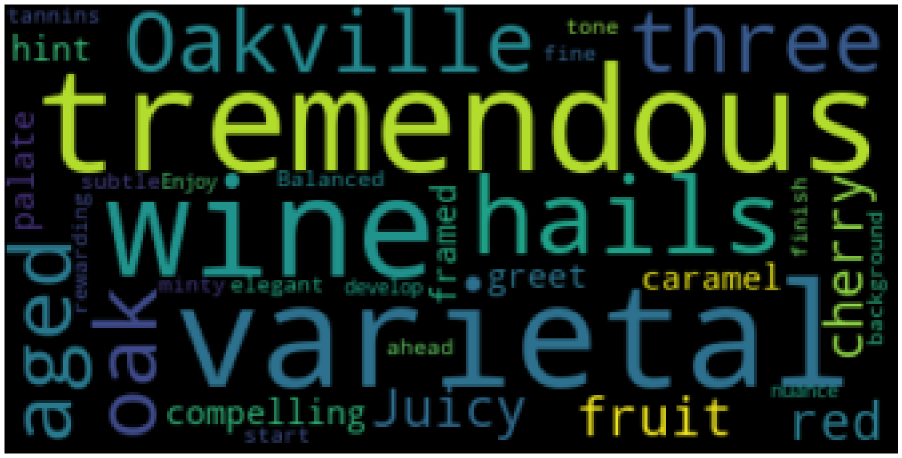
First, the stopwords supplied by this packages are stored in the array sw. Next the term "years" is added to this list, just to illustrate how you can remove words that do not belong. Next a wordcloud object wc is generated defining the default parameters for the cloud.

```
In [3]: sw = set(STOPWORDS)
        sw.add("years")
        wc = WordCloud(stopwords=sw)
        wc.generate(reviews[0])

Out[3]: <wordcloud.wordcloud.WordCloud at 0x1a192c9b00>
```

### 1.0.5 Display the Cloud

The word cloud object wc can be displayed using the matplotlib.pyplot function.

```
In [4]: # Display the word cloud.
        plt.imshow(wc, interpolation="bilinear")
        plt.axis("off")
        plt.figure()
        plt.show()
```



```
<matplotlib.figure.Figure at 0x1a19a42f28>
```

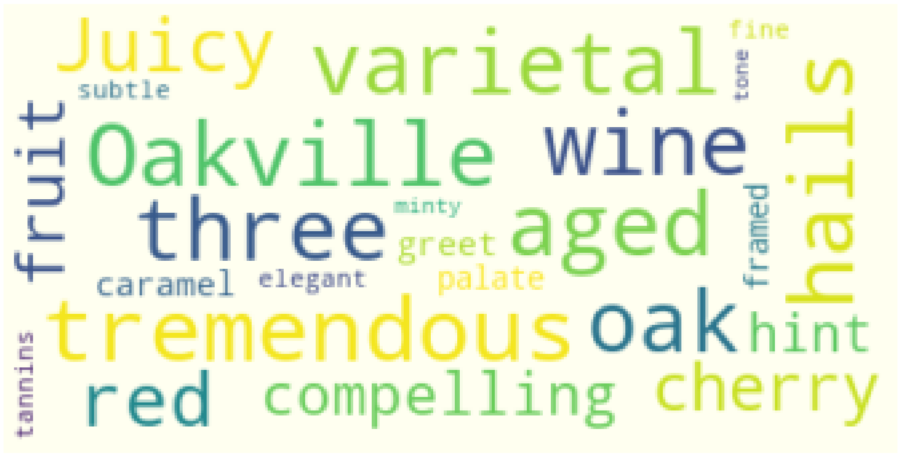
### 1.0.6 Improvements to the Basic Cloud

This is the default display for your word cloud. By default the background is black, and some of the words have dark colors, making them hard to see.

A more complete wordcloud object is descibed in the following code.

```
In [5]: wc = WordCloud(background_color="ivory", max_words=200, stopwords=sw, \
                       max_font_size=40,  min_font_size=10, prefer_horizontal=0.7, \
                       relative_scaling=0.5, width=400, height=200,  \
                       margin=10, random_state=341)

        wc.generate(reviews[0])
```

```
# Display the word cloud.
plt.imshow(wc, interpolation="bilinear")
plt.axis("off")
plt.figure()
plt.show()
```



```
<matplotlib.figure.Figure at 0x1a192c3cc0>
```

The main diffence is the background color. It was changed from black to ivory. White is also a good choice for this display. Most of the parameters given in the WordCloud( ) function are intuitive, but a few are not.

The pixel values for the width and height of the cloud are the default values. These can be changed to enlarge the image.

The default margin is margin=2, but a value of 10 moves the words off the cloud boundaries by little, which makes them easier to read.

The prefer_horizontal parameter specifies that vertical words are allowed. A value of 1 requires all words to be listed horizontally. By default prefer_horizonal=0.95 which allows some words to appear vertically, such as 'fruit' and 'hails'. In this example the default was changed to 0.7, allowing more words to appear vertically.

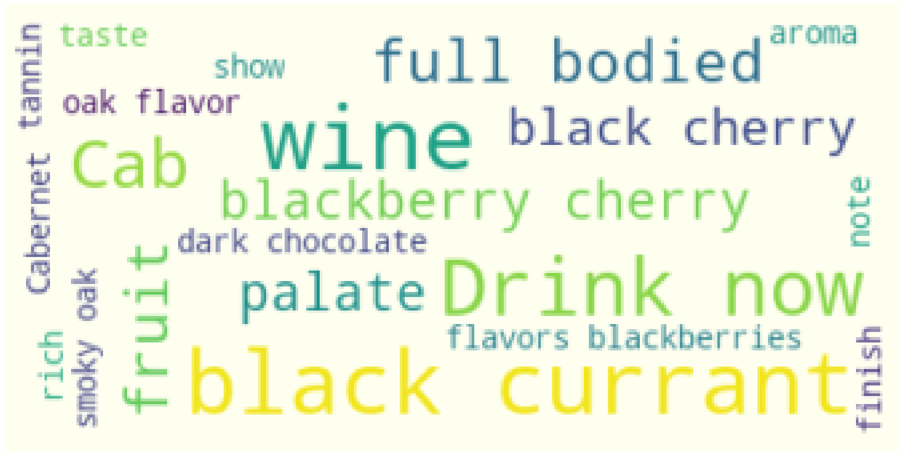
The parameter relative_scaling can have one of three values: 0, 0.5 or 1. A value relative_scaling=0 indicates that word size should be based upon word frequency. Words that appear more often are larger. A value relative_scaling=1 bases word size on their rank, or order, in the list of topic words, and a value of 0.5 mixes this and uses a little of both to display the word size. By default relative_scaling=0.5.

### 1.0.7 A Word Cloud for the Corpus

The following code takes all of the reviews and puts them into a long string s. A word cloud is build for the corpus using the original word cloud object wc.

```
In [6]: s = ""
        for i in range(len(reviews)):
            s += reviews[i]
        wc.generate(s)
        print("The cabernet corpus has", len(reviews)," reviews containing", len(s), " character

        # Display the word cloud.
        plt.imshow(wc, interpolation="bilinear")
        plt.axis("off")
        plt.figure()
        plt.show()
```

```
The cabernet corpus has 13135  reviews containing 3263373  characters.
```



```
<matplotlib.figure.Figure at 0x1a19a48fd0>
```

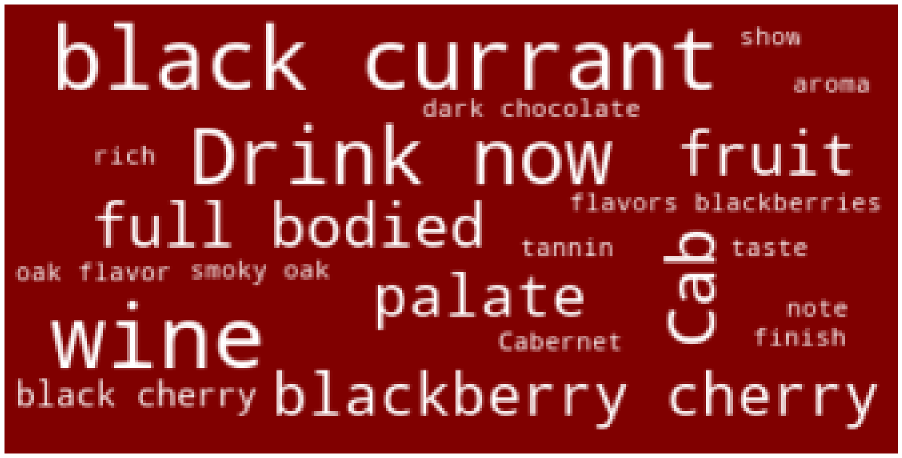
### 1.0.8 Shades of Grey

Regardless of the background color, there are some words that are either too dark or too light to be seen. If this is a problem, an easy solution is to display the words in shades of grey on a dark background.

In the following code, the background color is maroon. A function shades_of_grey allows us to convert the colors used to display individual words to shades of grey.

```
In [7]: # Function used to display words in shades of grey
        def shades_of_grey(word, font_size, position, orientation, random_state=None, \
                           **kwargs):
            return "hsl(0, 0%%, %d%%)" % random.randint(60,1000)
```

```
wc = WordCloud(background_color="maroon", max_words=200, stopwords=sw, \
               max_font_size=40,  min_font_size=10, prefer_horizontal=0.7, \
               relative_scaling=0.5, width=400, height=200,  \
               margin=10, random_state=341)
wc.generate(s)

plt.imshow(wc.recolor(color_func=shades_of_grey, random_state=3), \
                      interpolation="bilinear")
plt.axis("off")
plt.figure()
plt.show()
```



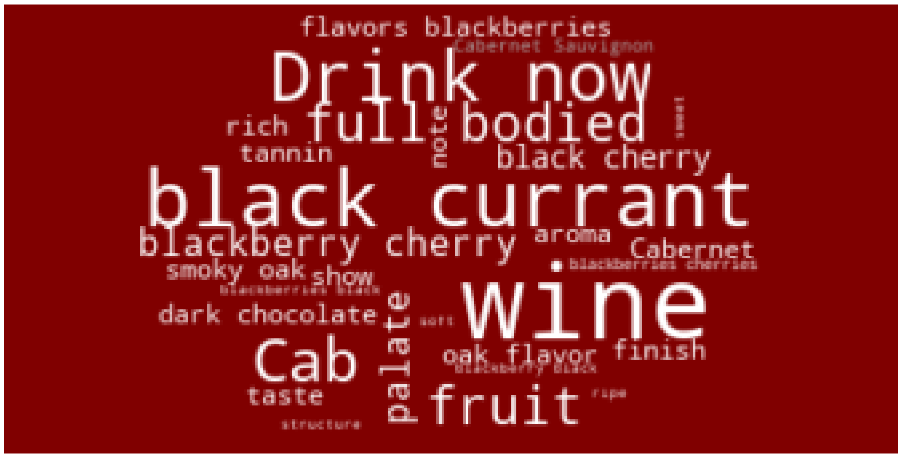
```
<matplotlib.figure.Figure at 0x1a19968668>
```

### 1.0.9 Image Masks

That's better. What if we used an image mask, a mask in the shape of an elipse?

Well we can do that. The following code uses a circle mask to produce the cloud and then displays the mask. Image masks in wordcloud are black and white. The application only writes words in the black area of the mask.

```
In [8]: circle_mask = np.array(Image.open("CircleMask.png"))
        wc = WordCloud(background_color="maroon", max_words=100, stopwords=sw, \
                       mask=circle_mask, width=400, height=200, min_font_size=6, random_state=34
        wc.generate(s)

        plt.imshow(wc.recolor(color_func=shades_of_grey, random_state=3), \
                                        interpolation="bilinear")
        plt.axis("off")
        plt.figure()
        plt.show()

        print("The Image Masked use to display a circular word cloud")
        plt.imshow(circle_mask, cmap=plt.cm.gray, interpolation='bilinear')
        plt.axis("off")
        plt.show()
```

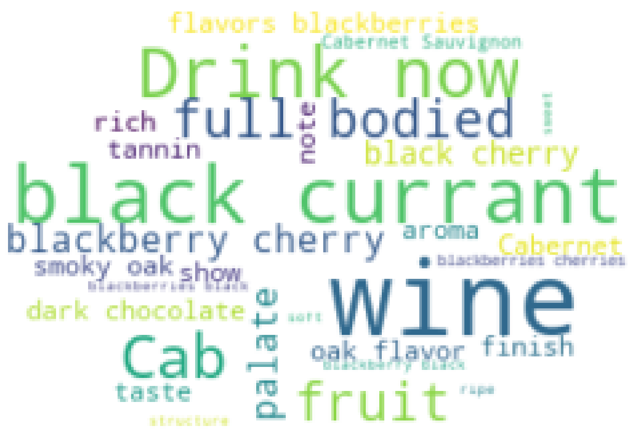


```
<matplotlib.figure.Figure at 0x1a1972a828>

The Image Masked use to display a circular word cloud
```

### 1.0.10 Circular Word Cloud with Colors

The following code uses the same mask with a light background to display a colorful cloud.

```
In [9]: circle_mask = np.array(Image.open("CircleMask.png"))
        wc = WordCloud(background_color="white", max_words=100, stopwords=sw, \
                       mask=circle_mask, width=400, height=200, min_font_size=6, random_state=34
        wc.generate(s)

        plt.imshow(wc, interpolation="bilinear")
        plt.axis("off")
        plt.figure()
        plt.show()
```



```
<matplotlib.figure.Figure at 0x1a1986c9b0>
```