Generating WordClouds in Python Tutorial

Learn how to perform Exploratory Data Analysis for Natural Language Processing using WordCloud in Python.

Contents

Nov 2019 · 18 min read

Duong Vu

Run and edit the code from this tutorial online

Open Workspace

What is WordCloud?

Many times you might have seen a cloud filled with lots of words in different sizes, which represent the frequency or the importance of each word. This is called Tag Cloud or WordCloud. For this tutorial, you will learn how to create a WordCloud of your own in Python and customize it as you see fit. This tool will be quite handy for exploring text data and making your report more lively.

In this tutorial we will use a wine review dataset taking from Wine Enthusiast website to learn:

- How to create a basic wordcloud from one to several text documents
- Adjust color, size and number of text inside your wordcloud
- Mask your wordcloud into any shape of your choice

- Mask your wordcloud into any color pattern of your choice

Prerequisites

You will need to install some packages below:

- **numpy**
- **pandas**
- **matplotlib**
- **pillow**
- **wordcloud**

The `numpy` library is one of the most popular and helpful libraries that is used for handling multi-dimensional arrays and matrices. It is also used in combination with `Pandas` library to perform data analysis.

The Python `os` module is a built-in library, so you don't have to install it. To read more about handling files with os module, **this DataCamp tutorial** will be helpful.

For visualization, `matplotlib` is a basic library that enables many other libraries to run and plot on its base including **seaborn** or `wordcloud` that you will use in this tutorial. The `pillow` library is a package that enables image reading. Its tutorial can be found **here**. Pillow is a wrapper for PIL - Python Imaging Library. You will need this library to read in image as the mask for the wordcloud.

`wordcloud` can be a little tricky to install. If you only need it for plotting a basic wordcloud, then `pip install wordcloud` or `conda install -c conda-forge wordcloud` would be sufficient. However, the latest version with the ability to mask the cloud into any shape of your choice requires a different method of installation as below:

```
git clone https://github.com/amueller/word_cloud.git
cd word_cloud
pip install .
```

POWERED BY DATACAMP WORKSPACE COPY CODE

Dataset:

This tutorial uses the **wine review dataset** from **Kaggle**. This collection is a great dataset for learning with no missing values (which will take time to handle) and a lot of text (wine reviews), categorical, and numerical data.

Now let's get started!

First thing first, you load all the necessary libraries:

```
# Start with loading all necessary libraries
import numpy as np
import pandas as pd
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import matplotlib.pyplot as plt
% matplotlib inline
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
c:\intelpython3\lib\site-packages\matplotlib\__init__.py:
import warnings
warnings.filterwarnings("ignore")
```

POWERED BY DATACAMP WORKSPACE COPY CODE

If you have more than 10 libraries, organize them by sections (such as basic libs, visualization, models, etc.) using comments in the code will make your code clean and easy to follow.
Now, using pandas `read_csv` to load in the dataframe. Notice the use of `index_col=0` meaning we don't read in row name (index) as a separated column.

```
# Load in the dataframe
df = pd.read_csv("data/winemag-data-130k-v2.csv", index_col=0)
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
# Looking at first 5 rows of the dataset
df.head()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	winery
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicosia
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Quinta dos Avidagos

2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainstorm
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	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	Paul Gregutt	@paulgwine	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks

You can printout some basic information about the dataset using `print()` combined with `.format()` to have a nice printout.

```
atures in this dataset. \n".format(df.shape[0],df.shape[1]))

dataset such as {}... \n".format(len(df.variety.unique()),
                                 ", ".join(df.variety.unique()[0:5])))

ne in this dataset such as {}... \n".format(len(df.country.unique()),
                                            ", ".join(df.country.unique()[0:5])))
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
here are 129971 observations and 13 features in this dataset.

here are 708 types of wine in this dataset such as White Blend, Portuguese Red, Pir

here are 44 countries producing wine in this dataset such as Italy, Portugal, US, S
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
df[["country", "description","points"]].head()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

	country	description	points
0	Italy	Aromas include tropical fruit, broom, brimston...	87
1	Portugal	This is ripe and fruity, a wine that is smooth...	87
2	US	Tart and snappy, the flavors of lime flesh and...	87
3	US	Pineapple rind, lemon pith and orange blossom ...	87
4	US	Much like the regular bottling from 2012, this...	87

To make comparisons between groups of a feature, you can use `groupby()` and compute summary statistics.

With the wine dataset, you can group by country and look at either the summary statistics for all countries' points and price or select the most popular and expensive ones.

```
# Groupby by country
country = df.groupby("country")

# Summary statistic of all countries
country.describe().head()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

	points								price							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
country																
Argentina	3800.0	86.710263	3.179627	80.0	84.00	87.0	89.00	97.0	3756.0	24.510117	23.430122	4.0	12.00	17.0	25.00	230.0
Armenia	2.0	87.500000	0.707107	87.0	87.25	87.5	87.75	88.0	2.0	14.500000	0.707107	14.0	14.25	14.5	14.75	15.0
Australia	2329.0	88.580507	2.989900	80.0	87.00	89.0	91.00	100.0	2294.0	35.437663	49.049458	5.0	15.00	21.0	38.00	850.0
Austria	3345.0	90.101345	2.499799	82.0	88.00	90.0	92.00	98.0	2799.0	30.762772	27.224797	7.0	18.00	25.0	36.50	1100.0
Bosnia and Herzegovina	2.0	86.500000	2.121320	85.0	85.75	86.5	87.25	88.0	2.0	12.500000	0.707107	12.0	12.25	12.5	12.75	13.0

This selects the top 5 highest average points among all 44 countries:

```
country.mean().sort_values(by="points",ascending=False).head()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

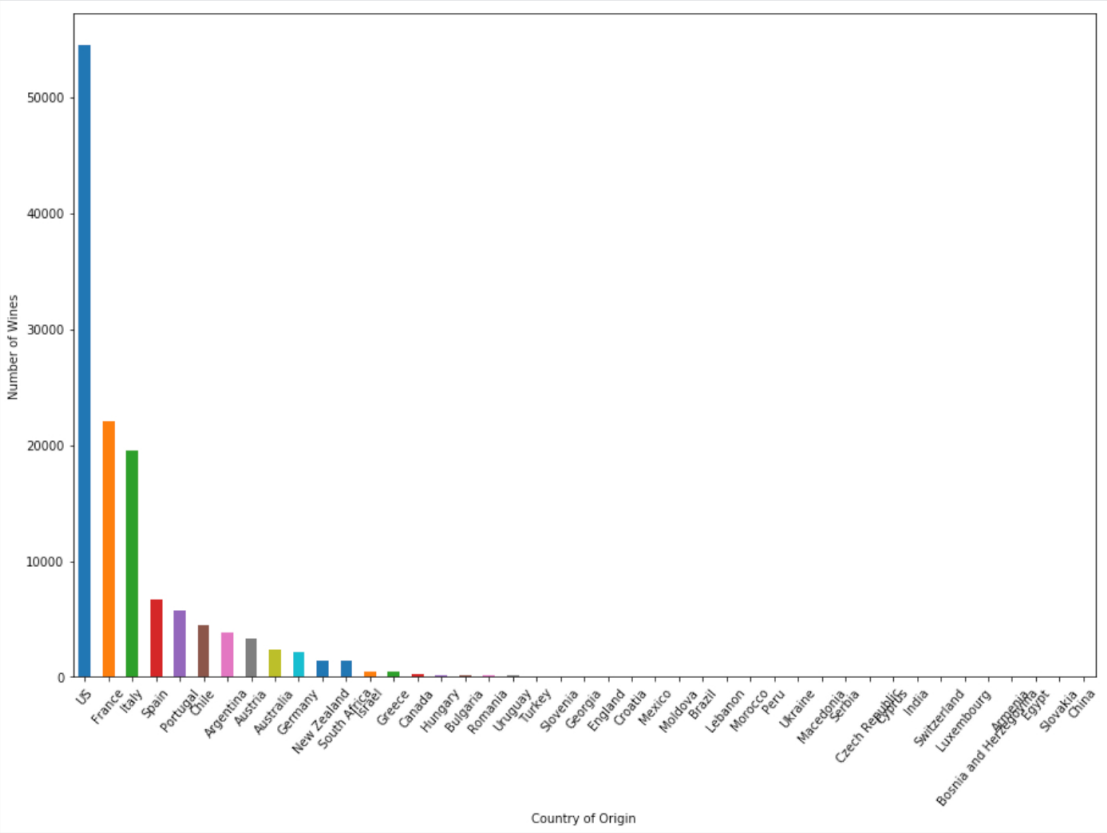
	points	price
country		
England	91.581081	51.681159
India	90.222222	13.333333
Austria	90.101345	30.762772
Germany	89.851732	42.257547
Canada	89.369650	35.712598

You can plot the number of wines by country using the plot method of Pandas DataFrame and Matplotlib. If you are not familiar with Matplotlib, I suggest taking a quick look at **this tutorial**.

```
plt.figure(figsize=(15,10))
country.size().sort_values(ascending=False).plot.bar()
plt.xticks(rotation=50)
plt.xlabel("Country of Origin")
```

```
plt.ylabel("Number of Wines")
plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE



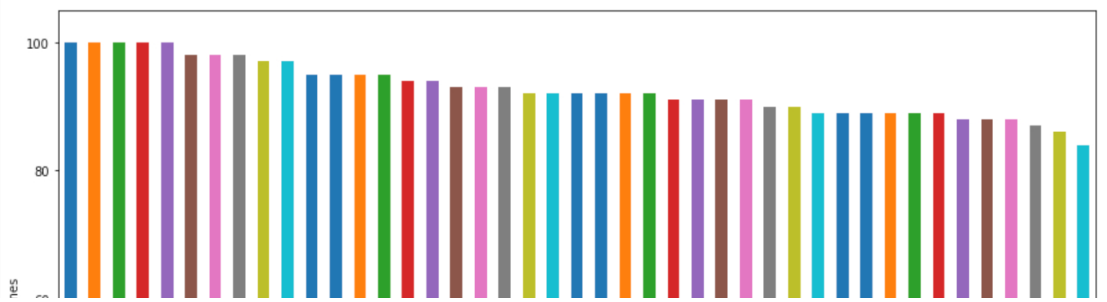
Among 44 countries producing wine, US has more than 50,000 types of wine in the wine review dataset, twice as much as the next one in the rank: France - the country famous for its wine. Italy also produces a lot of quality wine, having nearly 20,000 wines open to review.

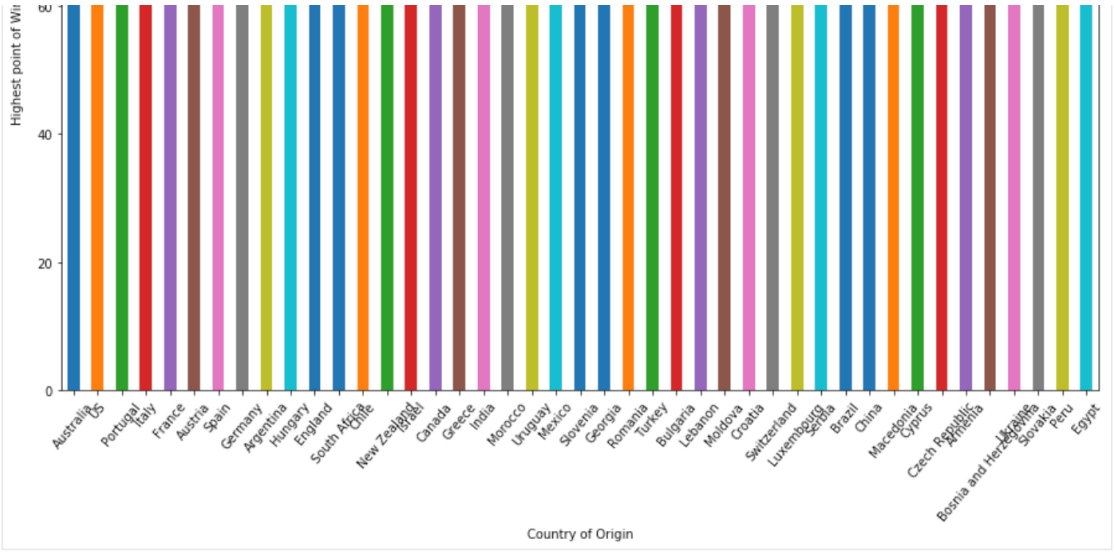
Does quantity over quality?

Let's now take a look at the plot of all 44 countries by its highest rated wine, using the same plotting technique as above:

```
plt.figure(figsize=(15,10))
country.max().sort_values(by="points",ascending=False)["points"].plot.bar()
plt.xticks(rotation=50)
plt.xlabel("Country of Origin")
plt.ylabel("Highest point of Wines")
plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE





Australia, US, Portugal, Italy, and France all have 100 points wine. If you notice, Portugal ranks 5th and Australia ranks 9th in the number of wines produces in the dataset, and both countries have less than 8000 types of wine.
That's a little bit of data exploration to get to know the dataset that you are using today. Now you will start to dive into the main course of the meal: **WordCloud**.

Set up a Basic WordCloud

WordCloud is a technique to show which words are the most frequent among the given text. The first thing you may want to do before using any functions is check out the docstring of the function, and see all required and optional arguments. To do so, type `?function` and run it to get all information.

```
?WordCloud
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
[1;31mInit signature: [0m  [0mWordCloud [0m[1;33m( [0m[0mfont_path [0m[1;33m= [0m
[1;31mDocstring: [0m
Word cloud object for generating and drawing.

Parameters
----------
font_path : string
    Font path to the font that will be used (OTF or TTF).
    Defaults to DroidSansMono path on a Linux machine. If you are on
    another OS or don't have this font; you need to adjust this path.

width : int (default=400)
    Width of the canvas.

height : int (default=200)
    Height of the canvas.

prefer_horizontal : float (default=0.90)
    The ratio of times to try horizontal fitting as opposed to vertical.
    If prefer_horizontal < 1, the algorithm will try rotating the word
    if it doesn't fit. (There is currently no built-in way to get only
    vertical words.)
```

```
mask : nd-array or None (default=None)
    If not None, gives a binary mask on where to draw words. If mask is not
    None, width and height will be ignored, and the shape of mask will be
    used instead. All white (#FF or #FFFFFF) entries will be considered
    "masked out" while other entries will be free to draw on. [This
    changed in the most recent version!]

contour_width: float (default=0)
    If mask is not None and contour_width > 0, draw the mask contour.

contour_color: color value (default="black")
    Mask contour color.

scale : float (default=1)
    Scaling between computation and drawing. For large word-cloud images,
    using scale instead of larger canvas size is significantly faster, but
    might lead to a coarser fit for the words.

min_font_size : int (default=4)
    Smallest font size to use. Will stop when there is no more room in this
    size.

font_step : int (default=1)
    Step size for the font. font_step > 1 might speed up computation but
    give a worse fit.

max_words : number (default=200)
    The maximum number of words.

stopwords : set of strings or None
    The words that will be eliminated. If None, the build-in STOPWORDS
    list will be used.

background_color : color value (default="black")
    Background color for the word cloud image.

max_font_size : int or None (default=None)
    Maximum font size for the largest word. If None, the height of the image is
    used.

mode : string (default="RGB")
    Transparent background will be generated when mode is "RGBA" and
    background_color is None.

relative_scaling : float (default=.5)
    Importance of relative word frequencies for font-size.  With
    relative_scaling=0, only word-ranks are considered.  With
    relative_scaling=1, a word that is twice as frequent will have twice
    the size.  If you want to consider the word frequencies and not only
    their rank, relative_scaling around .5 often looks good.

    .. versionchanged: 2.0
        Default is now 0.5.

color_func : callable, default=None
    Callable with parameters word, font_size, position, orientation,
    font_path, random_state that returns a PIL color for each word.
    Overwrites "colormap".
    See colormap for specifying a matplotlib colormap instead.
```

```
regexp : string or None (optional)
    Regular expression to split the input text into tokens in process_text.
    If None is specified, ``r"\w[\w']+"`` is used.

collocations : bool, default=True
    Whether to include collocations (bigrams) of two words.

    .. versionadded: 2.0

colormap : string or matplotlib colormap, default="viridis"
    Matplotlib colormap to randomly draw colors from for each word.
    Ignored if "color_func" is specified.

    .. versionadded: 2.0

normalize_plurals : bool, default=True
    Whether to remove trailing 's' from words. If True and a word
    appears with and without a trailing 's', the one with trailing 's'
    is removed and its counts are added to the version without
    trailing 's' -- unless the word ends with 's'.

Attributes
----------
``words_`` : dict of string to float
    Word tokens with associated frequency.

    .. versionchanged: 2.0
        ``words_`` is now a dictionary

``layout_`` : list of tuples (string, int, (int, int), int, color))
    Encodes the fitted word cloud. Encodes for each word the string, font
    size, position, orientation, and color.

Notes
-----
Larger canvases will make the code significantly slower. If you need a
large word cloud, try a lower canvas size, and set the scale parameter.

The algorithm might give more weight to the ranking of the words
then their actual frequencies, depending on the ``max_font_size`` and the
scaling heuristic.
[1;31mFile: [0m           c:\intelpython3\lib\site-packages\wordcloud\wordcloud.p
[1;31mType: [0m           type
```

POWERED BY DATACAMP WORKSPACE COPY CODE

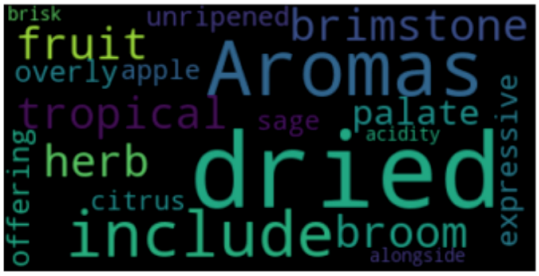
You can see that the only required argument for a WordCloud object is the **text**, while all others are optional.

So let's start with a simple example: using the first observation description as the input for the wordcloud. The three steps are:

- Extract the review (text document)
- Create and generate a wordcloud image
- Display the cloud using matplotlib

```
# Start with one review:
text = df.description[0]

# Create and generate a word cloud image:
wordcloud = WordCloud().generate(text)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE



Great! You can see that the first review mentioned a lot about dried flavors and the aromas of the wine.

Now, change some optional arguments of the WordCloud like `max_font_size`, `max_word`, and `background_color`.

```
# lower max_font_size, change the maximum number of word and lighten the backgrou
wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

Ugh, it seems like `max_font_size` here might not be a good idea. It makes it more difficult to see the differences between word frequencies. However, brightening the background makes the cloud easier to read.

If you want to save the image, WordCloud provides a function `to_file`

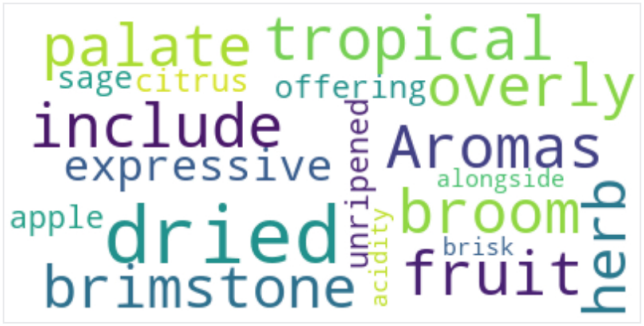
```
# Save the image in the img folder:
wordcloud.to_file("img/first_review.png")
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
<wordcloud.wordcloud.WordCloud at 0x16f1d704978>
```

POWERED BY DATACAMP WORKSPACE COPY CODE

The result will look like this when you load them in:



You've probably noticed the argument `interpolation="bilinear"` in the `plt.imshow()`. This is to make the displayed image appear more smoothly. For more information about the choice, **here** is a helpful link to explore more about this choice.

So now you'll combine all wine reviews into one big text and create a big fat cloud to see which characteristics are most common in these wines.

```
text = " ".join(review for review in df.description)
print ("There are {} words in the combination of all review.".format(len(text)))
```

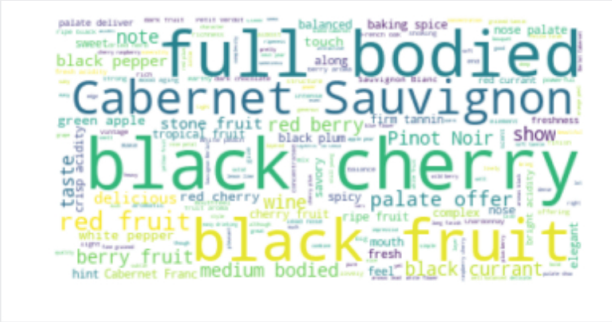
POWERED BY DATACAMP WORKSPACE COPY CODE

```
There are 31661073 words in the combination of all review.
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
# Create stopword list:
stopwords = set(STOPWORDS)
stopwords.update(["drink", "now", "wine", "flavor", "flavors"])

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(tex

# Display the generated image:
# the matplotlib way:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE



Ohhh, it seems like black cherry and full-bodied are the most mentioned characteristics, and Cabernet Sauvignon is the most popular of them all. This aligns with the fact that Cabernet Sauvignon "is one of the world's most widely recognized red wine grape varieties. It is grown in nearly every major wine producing country among a diverse spectrum of climates from Canada's Okanagan Valley to Lebanon's Beqaa Valley".[1]

Now, let's pour these words into a cup of wine!
Seriously,
Even a bottle of wine if you wish!

In order to create a shape for your wordcloud, first, you need to find a PNG file to become the mask. Below is a nice one that is available on the internet:

Not all mask images have the same format resulting in different outcomes, hence making the WordCloud function not working properly. To make sure that your mask works, let's take a look at it in the numpy array form:

```
wine_mask = np.array(Image.open("img/wine_mask.png"))
wine_mask
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

POWERED BY DATACAMP WORKSPACE COPY CODE

The way the masking functions works is that it requires all white part of the mask should be 255 not 0 (integer type). This value represents the "intensity" of the pixel. Values of 255 are pure white, whereas values of 1 are black. Here, you can use the provided function below to transform your mask if your mask has the same format as above. Notice if you have a mask that the background is not 0, but 1 or 2, adjust the function to match your mask.

First, you use the `transform_format()` function to swap number 0 to 255.

```
def transform_format(val):
    if val == 0:
        return 255
    else:
        return val
```

POWERED BY DATACAMP WORKSPACE COPY CODE

Then, create a new mask with the same shape as the mask you have in hand and apply the function `transform_format()` to each value in each row of the previous mask.

```
# Transform your mask into a new one that will work with the function:
transformed_wine_mask = np.ndarray((wine_mask.shape[0],wine_mask.shape[1]), np.in
```

```
for i in range(len(wine_mask)):
    transformed_wine_mask[i] = list(map(transform_format, wine_mask[i]))
```

POWERED BY DATACAMP WORKSPACE COPY CODE

Now, you have a new mask in the correct form. Printout the transformed mask is the best way to check if the function works fine.

```
# Check the expected result of your mask
transformed_wine_mask
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
array([[255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       ...,
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255]])
```

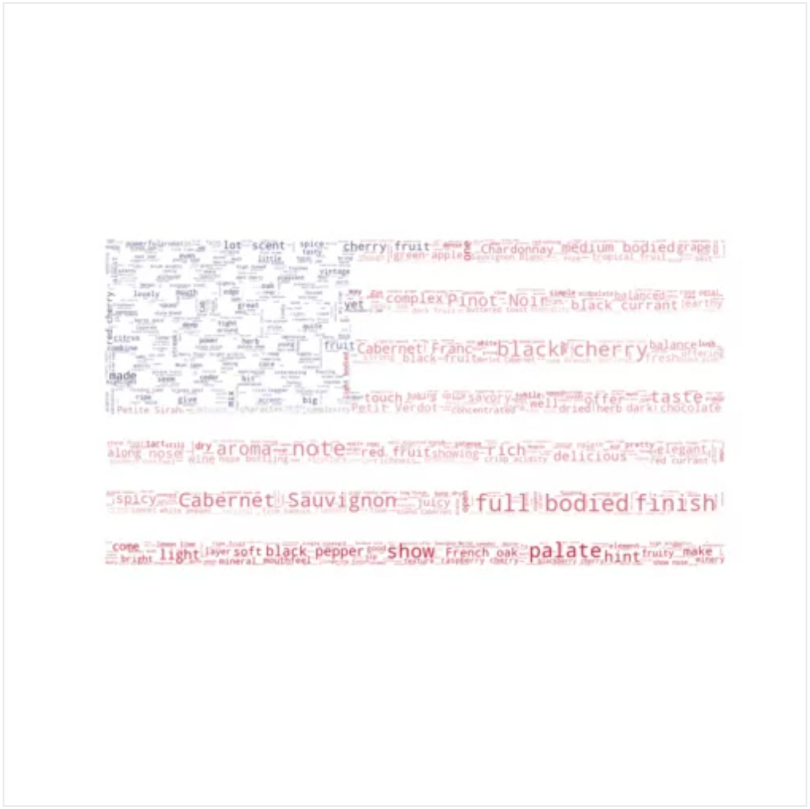
POWERED BY DATACAMP WORKSPACE COPY CODE

Okay! With the right mask, you can start making the wordcloud with your selected shape. Notice in the `WordCloud` function, there is a `mask` argument that takes in the transformed mask that you created above. The `contour_width` and `contour_color` are, as their name, arguments to adjust the outline characteristics of the cloud. The wine bottle you have here is a red wine bottle, so firebrick seems like a good choice for contour color. For more choice of color, you can take a look at this **color code table**

```
# Create a word cloud image
wc = WordCloud(background_color="white", max_words=1000, mask=transformed_wine_ma
               stopwords=stopwords, contour_width=3, contour_color='firebrick')

# Generate a wordcloud
wc.generate(text)

# store to file
wc.to_file("img/wine.png")

# show
plt.figure(figsize=[20,10])
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

Voila! You created a wordcloud in the shape of a wine bottle! It seems like wine descriptions most often mention about black cherry, fruit flavors and full-bodied characteristics of the wine. Now let's take a closer look at the reviews for each country and plot the wordcloud using each country flag. For you easy to imagine, this is an example that you will create soon:



Creating Wordcloud Following a Color Pattern

You can combine all the reviews of five countries that have the most wines. To find those countries, you can either look at the plot *country vs number* of wine above or use the group that you got above to find the number of observations for each country (each group) and `sort_values()` with argument `ascending=False` to sort descending.

```
country.size().sort_values(ascending=False).head()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
country
US          54504
France      22093
Italy       19540
Spain        6645
Portugal     5691
dtype: int64
```

POWERED BY DATACAMP WORKSPACE COPY CODE

So now you have 5 top countries: US, France, Italy, Spain, and Portugal. You can change the number of countries by putting your choice number insider `head()` like below

```
country.size().sort_values(ascending=False).head(10)
```

POWERED BY DATACAMP WORKSPACE COPY CODE

```
country
US           54504
France       22093
Italy        19540
Spain         6645
Portugal      5691
Chile         4472
Argentina     3800
Austria       3345
Australia     2329
Germany       2165
dtype: int64
```

POWERED BY DATACAMP WORKSPACE COPY CODE

For now, 5 countries should be enough.

To get all review for each country, you can concatenate all of the reviews using the `" ".join(list)` syntax, which joins all elements in a list separating them by whitespace.

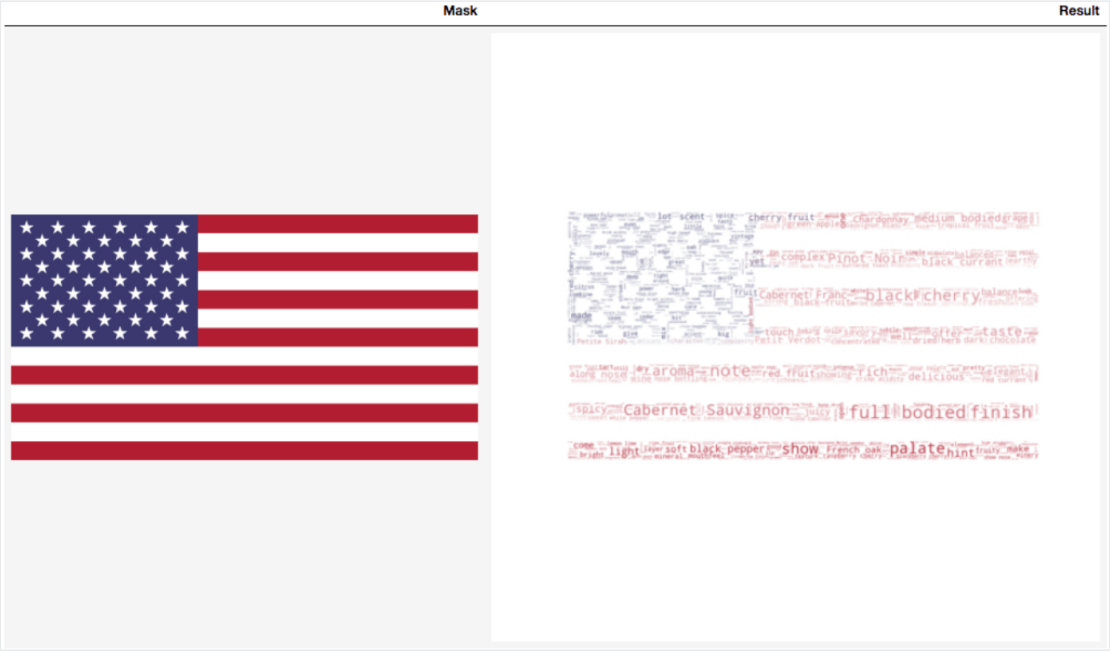
```
# Join all reviews of each country:
usa = " ".join(review for review in df[df["country"]=="US"].description)
fra = " ".join(review for review in df[df["country"]=="France"].description)
ita = " ".join(review for review in df[df["country"]=="Italy"].description)
spa = " ".join(review for review in df[df["country"]=="Spain"].description)
por = " ".join(review for review in df[df["country"]=="Portugal"].description)
```

POWERED BY DATACAMP WORKSPACE COPY CODE

Then, creating the wordcloud as above. You can combine the two steps of creating and generate into one as below. The color mapping is done right before you plot the cloud using the **ImageColorGenerator** function from WordCloud library.

```
# Generate a word cloud image
mask = np.array(Image.open("img/us.png"))
wordcloud_usa = WordCloud(stopwords=stopwords, background_color="white", mode="RG

# create coloring from image
image_colors = ImageColorGenerator(mask)
plt.figure(figsize=[7,7])
plt.imshow(wordcloud_usa.recolor(color_func=image_colors), interpolation="bilinea
plt.axis("off")

# store to file
plt.savefig("img/us_wine.png", format="png")

plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE



Looks good! Now let's repeat with a review from France.

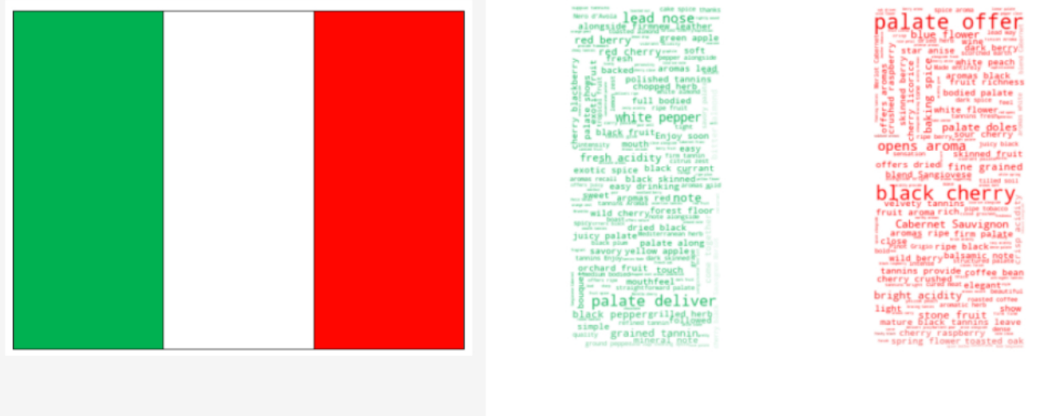
```
# Generate a word cloud image
mask = np.array(Image.open("img/france.png"))
wordcloud_fra = WordCloud(stopwords=stopwords, background_color="white", mode="RG

# create coloring from image
image_colors = ImageColorGenerator(mask)
plt.figure(figsize=[7,7])
plt.imshow(wordcloud_fra.recolor(color_func=image_colors), interpolation="bilinea
plt.axis("off")

# store to file
plt.savefig("img/fra_wine.png", format="png")

#plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

Please note that you should save the image after plotting to have the wordcloud with the desired color pattern.


```
# Generate a word cloud image
mask = np.array(Image.open("img/italy.png"))
wordcloud_ita = WordCloud(stopwords=stopwords, background_color="white", max_word

# create coloring from image
image_colors = ImageColorGenerator(mask)
plt.figure(figsize=[7,7])
plt.imshow(wordcloud_ita.recolor(color_func=image_colors), interpolation="bilinea
plt.axis("off")

# store to file
plt.savefig("img/ita_wine.png", format="png")

#plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE



Following Italy is Spain:

```
# Generate a word cloud image
mask = np.array(Image.open("img/spain.png"))
wordcloud_spa = WordCloud(stopwords=stopwords, background_color="white", max_word

# create coloring from image
image_colors = ImageColorGenerator(mask)
plt.figure(figsize=[7,7])
plt.imshow(wordcloud_spa.recolor(color_func=image_colors), interpolation="bilinea
plt.axis("off")

# store to file
plt.savefig("img/spa_wine.png", format="png")
#plt.show()
```

POWERED BY DATACAMP WORKSPACE COPY CODE

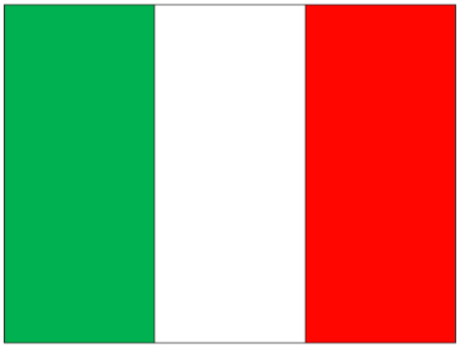
Lastly, Portugal:

```
# Generate a word cloud image
mask = np.array(Image.open("img/portugal.png"))
wordcloud_por = WordCloud(stopwords=stopwords, background_color="white", max_word

# create coloring from image
image_colors = ImageColorGenerator(mask)
plt.figure(figsize=[7,7])
plt.imshow(wordcloud_por.recolor(color_func=image_colors), interpolation="bilinea
plt.axis("off")

# store to file
plt.savefig("img/por_wine.png", format="png")
#plt.show()
```

The end result is in the below table to compare between the mask and the wordcloud. Which one is your favorite?

Mask	Result
	


give fine
finish
yellow fruit balanced
ready




lead nose
red cherry
white pepper
black fruit
fresh acidity
palate deliver
palate offer
black cherry
Cabernet Sauvignon
bright acidity


palate
aroma
nose
note
feel
sweet
finish
fresh
black fruit
tannic
blend
herbal
solid
ripe
berry
vanilla
spice


fruit
ready
acidity
ripe
black fruit
structure
tannin
structured
balanced
attractive
wood aging
perfumed
give

Congratulations!

You made it! You have learned several ways to draw a WordCloud that would be helpful for visualization of any text analysis. You also learn how to mask the cloud into any shape, using any color of your choice. If you want to practice your skills, consider the DataCamp's project: **The Hottest Topics in Machine Learning**

If you'd like to get in touch with me, you can drop me an e-mail at dqvu.ubc@gmail.com or connect with me via **LinkedIn**.

If you are interested in learning more about Natural Language Processing, take our **Natural Language Processing Fundamentals in Python** course.

TOPICS

Python | Data Analysis | Data Visualization | AI

Python Courses

,724

lysis with online course face and

art Course

Intermediate Python

Beginner | 4 hours | 876,357

Level up your data science skills by creating visualizations using Matplotlib and manipulating DataFrames with pandas.

See Details → | Start Course

Python Data Science Toolbox (Part 2)

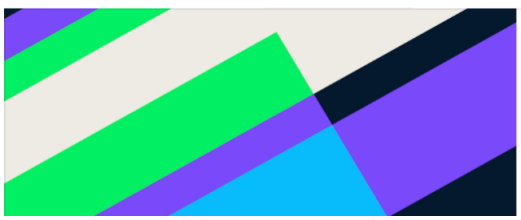
Beginner | 4 hours | 224,182

Continue to build your modern Data Science skills by learning about iterators and list comprehensions.

See Details → | Start Course

See all courses →

Related



Working with Dates and Times in Python Cheat Sheet

Working with dates and times is essential when manipulating data in Python. Learn the basics of working with datetime data in this cheat sheet.

DataCamp Team •

Plotly Express Cheat Sheet

Plotly is one of the most widely used data visualization packages in Python. Learn more about it in this cheat sheet.

DataCamp Team • 0 min

Python pandas tutorial: The ultimate guide for beginners

Are you ready to begin your pandas journey? Here's a step-by-step guide on how to get started. [Updated November 2022]

Vidhi Chugh • 15 min

Python Iterators and Generators Tutorial

Explore the difference between Python Iterators and Generators and learn which are the best to use in various situations.

Kurtis Pykes • 10 min

See More →

Learn Python
Learn R
Learn SQL
Learn Power BI
Learn Tableau
Assessments
Career Tracks
Skill Tracks
Courses
Data Science Roadmap

Python Courses
R Courses
SQL Courses
Power BI Courses
Tableau Courses
Spreadsheet Courses
Data Analysis Courses
Data Visualization Courses
Machine Learning Courses
Data Engineering Courses

Get Started
Templates
Integrations
Documentation

CERTIFICATION

Certifications
Data Scientist
Data Analyst
Hire Data Professionals

RESOURCES

Resource Center
Upcoming Events
Blog
Tutorials
Open Source
RDocumentation
Course Editor
Book a Demo with DataCamp for Business

PLANS

Pricing
For Business
For Classrooms
Discounts, Promos & Sales
DataCamp Donates

SUPPORT

Help Center
Become an Instructor
Become an Affiliate

ABOUT

About Us
Learner Stories
Careers
Press
Leadership
Contact Us

clear as data