# Regular Expressions

CS5154/6054

Yizong Cheng

8/25/2022

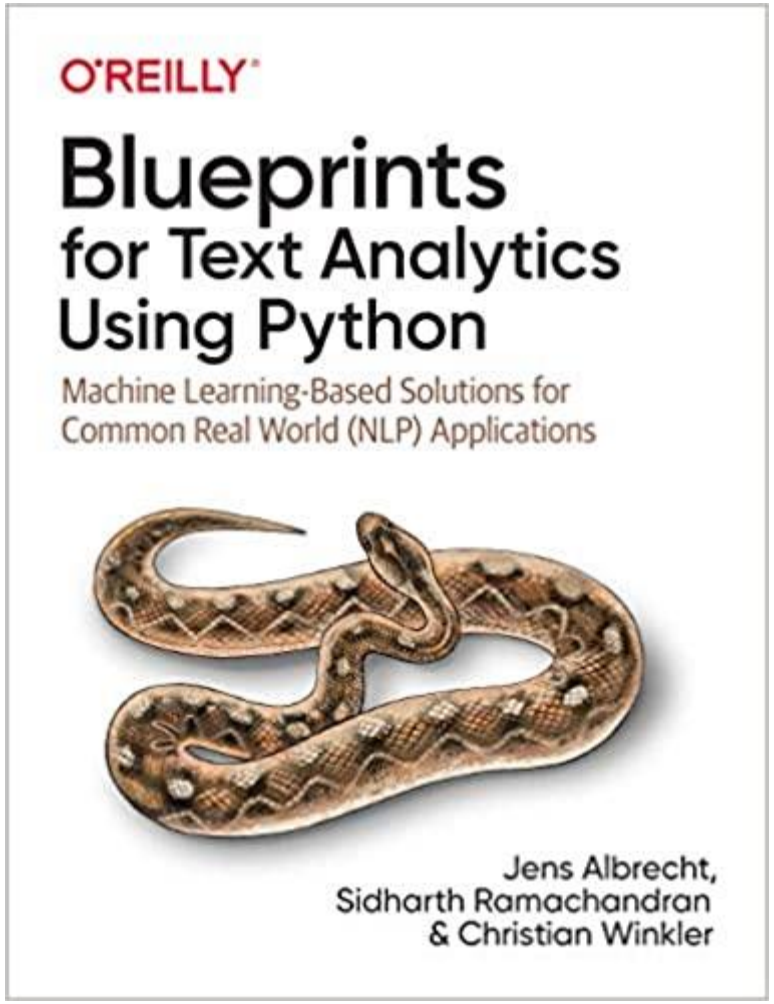
```
f = open("bible.txt", "r")
docs = f.readlines()
f.close()
invertedIndex = {}
for i in range(len(docs)):
    for s in docs[i].split():
        if invertedIndex.get(s) == None:
            invertedIndex.update({s : {i}})
        else:
            invertedIndex.get(s).add(i)
```

```
import re

f = open("bible.txt", "r")
docs = f.readlines()
f.close()
invertedIndex = {}
for i in range(len(docs)):
    for s in re.split('\s', docs[i]):
        if invertedIndex.get(s) == None:
            invertedIndex.update({s : {i}})
        else:
            invertedIndex.get(s).add(i)
```

```
import re

f = open("bible.txt", "r")
docs = f.readlines()
f.close()
invertedIndex = {}
for i in range(len(docs)):
    for s in re.findall('\w+', docs[i]):
        if invertedIndex.get(s) == None:
            invertedIndex.update({s : {i}})
        else:
            invertedIndex.get(s).add(i)
```


O'REILLY
Blueprints
for Text Analytics
Using Python
Machine Learning-Based Solutions for
Common Real World (NLP) Applications
Jens Albrecht,
Sidharth Ramachandran
& Christian Winkler

# Chapter 4. Preparing Textual Data for Statistics and Machine Learning

## REGULAR EXPRESSIONS

Regular expressions are an essential tool for text data preparation. They can be used not only for tokenization and data cleaning but also for the identification and treatment of email addresses, salutations, program code, and more.

Python has the standard library `re` for regular expressions and the newer, backward-compatible library regex that offers support for POSIX character classes and some more flexibility.

A good overview about the available meta-characters like `^` as well as character classes like `\w` is available at W3Schools. There is also a number of interactive websites to develop and test regular expressions, e.g., *https://regex101.com* (make sure to set the flavor to Python).

In many packages, you will find the precompiled regular expressions like this:

```
RE_BRACKET = re.compile('\[[^\[\]]*\]')
text = RE_BRACKET.sub(' ', text)
```

File Edit View History Bookmarks Tools Help

Python RegEx

https://www.w3schools.com/python/python_regex.asp

Tutorials ▾ References ▾ Exercises ▾ Videos Website Paid Courses Log in

HTML CSS JAVASCRIPT SQL PYTHON PHP BOOTSTRAP HOW TO

Python Tutorial

- Python HOME
- Python Intro
- Python Get Started
- Python Syntax
- Python Comments
- Python Variables
- Python Data Types
- Python Numbers
- Python Casting
- Python Strings
- Python Booleans
- Python Operators
- Python Lists
- Python Tuples
- Python Sets
- Python Dictionaries

# Python RegEx

❮ Previous Next ❯

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern.

# RegEx Functions

The `re` module offers a set of functions that allows us to search a string for a match:

| Function | Description |
|---|---|
| findall | Returns a list containing all matches |
| search | Returns a Match object if there is a match anywhere in the string |
| split | Returns a list where the string has been split at each match |
| sub | Replaces one or many matches with a string |

# Metacharacters

Metacharacters are characters with a special meaning:

| Character | Description | Example | Try it |
|---|---|---|---|
| [] | A set of characters | "[a-m]" | Try it » |
| \ | Signals a special sequence (can also be used to escape special characters) | "\d" | Try it » |
| . | Any character (except newline character) | "he..o" | Try it » |
| ^ | Starts with | "^hello" | Try it » |
| $ | Ends with | "world$" | Try it » |
| * | Zero or more occurrences | "aix*" | Try it » |
| + | One or more occurrences | "aix+" | Try it » |
| {} | Exactly the specified number of occurrences | "al{2}" | Try it » |
| \| | Either or | "falls\|stays" | Try it » |
| () | Capture and group | | |

# Special Sequences

A special sequence is a `\` followed by one of the characters in the list below, and has a special meaning:

| Character | Description | Example | Try it |
|---|---|---|---|
| \A | Returns a match if the specified characters are at the beginning of the string | "\AThe" | Try it » |
| \b | Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\bain"<br>r"ain\b" | Try it »<br>Try it » |
| \B | Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\Bain"<br>r"ain\B" | Try it »<br>Try it » |

| | | | |
|---|---|---|---|
| \d | Returns a match where the string contains digits (numbers from 0-9) | "\d" | Try it » |
| \D | Returns a match where the string DOES NOT contain digits | "\D" | Try it » |
| \s | Returns a match where the string contains a white space character | "\s" | Try it » |
| \S | Returns a match where the string DOES NOT contain a white space character | "\S" | Try it » |
| \w | Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character) | "\w" | Try it » |
| \W | Returns a match where the string DOES NOT contain any word characters | "\W" | Try it » |
| \Z | Returns a match if the specified characters are at the end of the string | "Spain\Z" | Try it » |

# Sets

A set is a set of characters inside a pair of square brackets `[]` with a special meaning:

| Set | Description | Try it |
|---|---|---|
| [arn] | Returns a match where one of the specified characters ( `a` , `r` , or `n` ) are present | Try it » |
| [a-n] | Returns a match for any lower case character, alphabetically between `a` and `n` | Try it » |
| [^arn] | Returns a match for any character EXCEPT `a` , `r` , and `n` | Try it » |
| [0123] | Returns a match where any of the specified digits ( `0` , `1` , `2` , or `3` ) are present | Try it » |
| [0-9] | Returns a match for any digit between `0` and `9` | Try it » |
| [0-5][0-9] | Returns a match for any two-digit numbers from `00` and `59` | Try it » |
| [a-zA-Z] | Returns a match for any character alphabetically between `a` and `z` , lower case OR upper case | Try it » |
| [+] | In sets, `+` , `*` , `.` , `\|` , `()` , `$` ,`{}` has no special meaning, so `[+]` means: return a match for any `+` character in the string | Try it » |

# The findall() Function

The `findall()` function returns a list containing all matches.

## Example

Print a list of all matches:

```
import re

txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
```

Try it Yourself »

The list contains the matches in the order they are found.

If no matches are found, an empty list is returned:

# The search() Function

The `search()` function searches the string for a match, and returns a Match object if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

## Example

Search for the first white-space character in the string:

```
import re

txt = "The rain in Spain"
x = re.search("\s", txt)

print("The first white-space character is located in position:",
x.start())
```

Try it Yourself »

If no matches are found, the value `None` is returned:

# The split() Function

The `split()` function returns a list where the string has been split at each match:

## Example

Split at each white-space character:

```
import re

txt = "The rain in Spain"
x = re.split("\s", txt)
print(x)
```

Try it Yourself »

You can control the number of occurrences by specifying the `maxsplit` parameter:

# The sub() Function

The `sub()` function replaces the matches with the text of your choice:

## Example

Replace every white-space character with the number 9:

```
import re

txt = "The rain in Spain"
x = re.sub("\s", "9", txt)
print(x)
```

Try it Yourself »

```
def clean(text):
    # convert html escapes like &amp; to characters.
    text = html.unescape(text)
    # tags like <tab>
    text = re.sub(r'<[^<>]*>', ' ', text)
    # markdown URLs like [Some text](https://....)
    text = re.sub(r'\[([^\[\]]*)\]\([^\(\)]*\)', r'\1', text)
    # text or code in brackets like [0]
    text = re.sub(r'\[[^\[\]]*\]', ' ', text)
    # standalone sequences of specials, matches &# but not #cool
    text = re.sub(r'(?:^|\s)[&#<>{}\[\]+|\\:-]{1,}(?:\s|$)', ' ', text)
    # standalone sequences of hyphens like --- or ==
    text = re.sub(r'(?:^|\s)[\-=\+]{2,}(?:\s|$)', ' ', text)
    # sequences of white spaces
    text = re.sub(r'\s+', ' ', text)
    return text.strip()
```

File Edit View History Bookmarks Tools Help

wordcloud.WordCloud — wordclou

https://amueller.github.io/word_cloud/generated/wordcloud.WordCloud.html



# wordcloud.WordCloud

*class* `wordcloud.WordCloud`(*font_path=None, width=400, height=200, margin=2, ranks_only=None, prefer_horizontal=0.9, mask=None, scale=1, color_func=None, max_words=200, min_font_size=4, stopwords=None, random_state=None, background_color='black', max_font_size=None, font_step=1, mode='RGB', relative_scaling='auto', regexp=None, collocations=True, colormap=None, normalize_plurals=True, contour_width=0, contour_color='black', repeat=False, include_numbers=False, min_word_length=0, collocation_threshold=30*) [source]

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.

IR2A - Notepad

File Edit Format View Help

```
# tokenize and count words
# use WordCloud to show words with top frequencies
# Usage: python IR2A.py bible.txt

import re
import sys
from collections import Counter
from wordcloud import WordCloud
from matplotlib import pyplot as plt

f = open(sys.argv[1], 'r')

counter = Counter()
for t in f:
    counter.update(re.findall('\w+', t))

wc = WordCloud()
wc.generate_from_frequencies(counter)
plt.imshow(wc)
plt.axis("off")
plt.show()
```

File Edit View History Bookmarks Tools Help

wordcloud.WordCloud — wordclou

https://amueller.github.io/word_cloud/generated/wordcloud.WordCloud.html



**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. To create a word cloud with a single color, use `color_func=lambda *args, **kwargs: "white"`. The single color can also be specified using RGB code. For example `color_func=lambda *args, **kwargs: (255,0,0)` sets color to red.

**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. Ignored if using generate_from_frequencies.

**collocations** : *bool, default=True*

Whether to include collocations (bigrams) of two words. Ignored if using generate_from_frequencies.

**colormap** : *string or matplotlib colormap, default="viridis"*

Matplotlib colormap to randomly draw colors from for each word. Ignored if "color_func" is specified.

File Edit View History Bookmarks Tools Help

Choosing Colormaps in Matplo

https://matplotlib.org/stable/tutorials/colors/colormaps.html

matplotlib

Plot types | Examples | Tutorials | Reference | User guide | Develop | Release notes

Search the docs ...



# Choosing Colormaps in Matplotlib

Matplotlib has a number of built-in colormaps accessible via `matplotlib.cm.get_cmap`. There are also external libraries that have many extra colormaps, which can be viewed in the Third-party colormaps section of the Matplotlib documentation. Here we briefly discuss how to choose between the many options. For help on creating your own colormaps, see Creating Colormaps in Matplotlib.

## Overview

The idea behind choosing a good colormap is to find a good representation in 3D colorspace for your data set. The best colormap for any given data set depends on many things including:

- Whether representing form or metric data ([Ware])
- Your knowledge of the data set (*e.g.*, is there a critical value from which the other values deviate?)

File Edit View History Bookmarks Tools Help

Choosing Colormaps in Matplo

https://matplotlib.org/stable/tutorials/colors/colormaps.html 150%

matplotlib

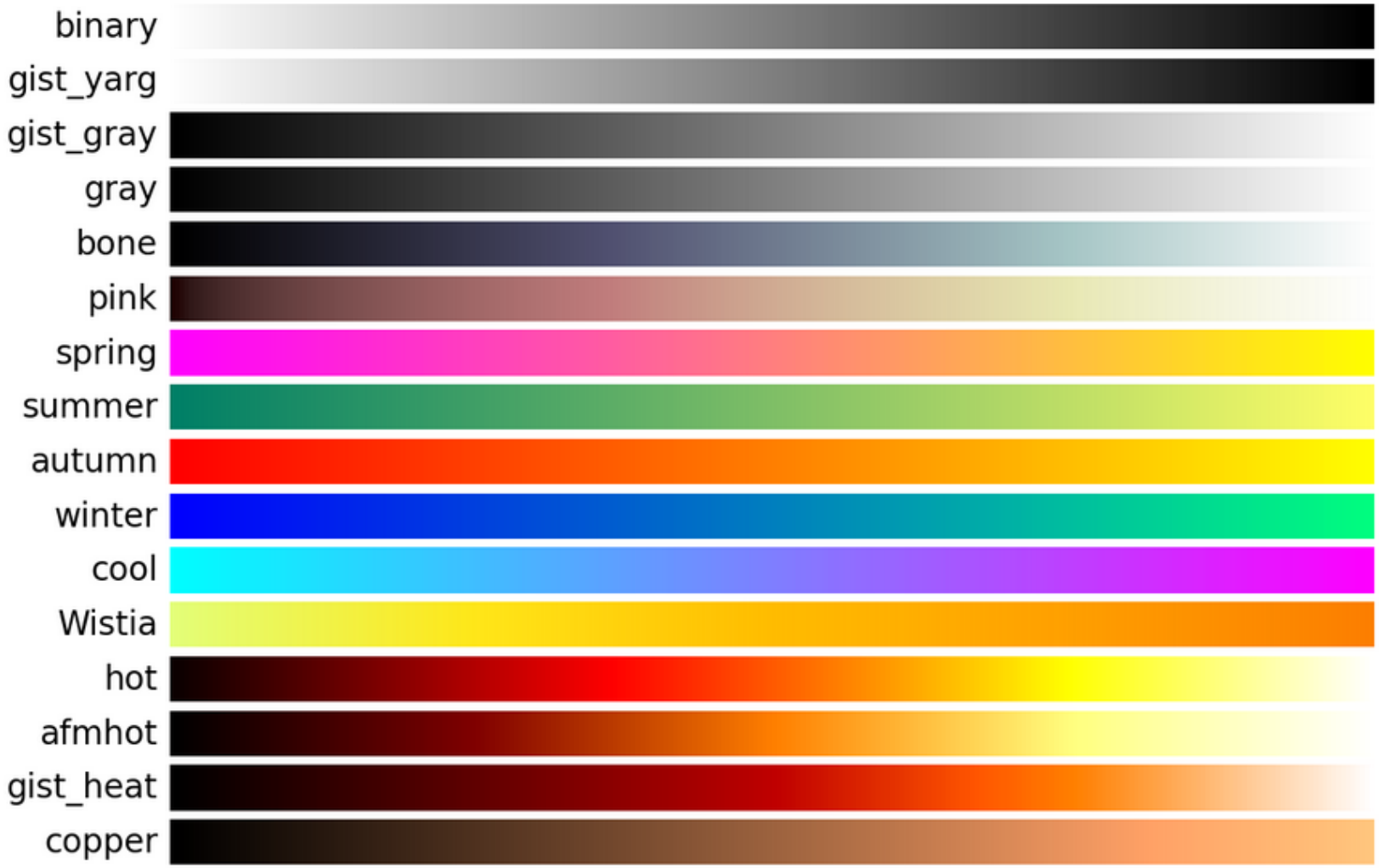
```
plot_color_gradients('Perceptually Uniform Sequential',
                     ['viridis', 'plasma', 'inferno', 'magma', 'cividis'])
```

File Edit View History Bookmarks Tools Help

Choosing Colormaps in Matplo

https://matplotlib.org/stable/tutorials/colors/colormaps.html 150%

matplotlib



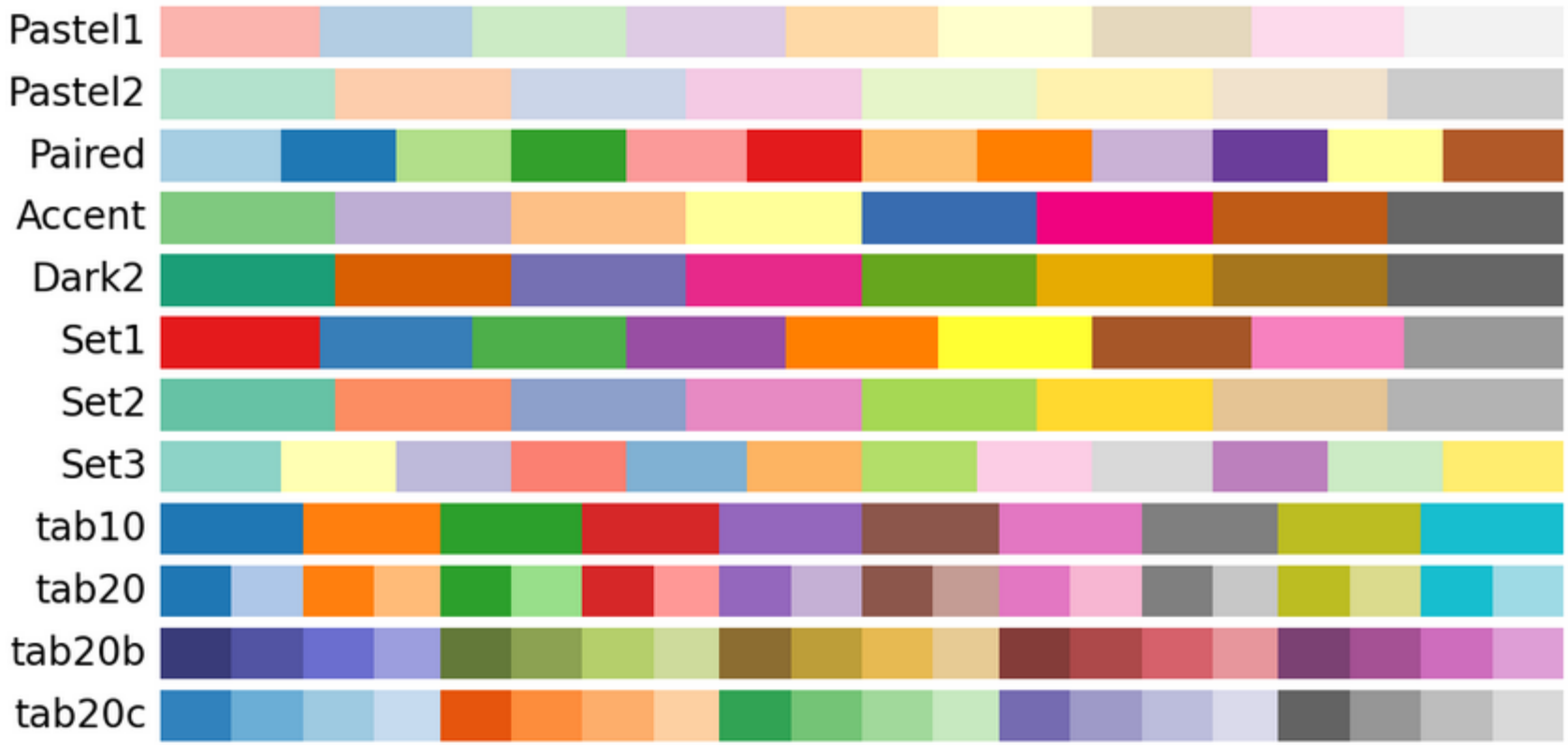
File Edit View History Bookmarks Tools Help

Choosing Colormaps in Matplo

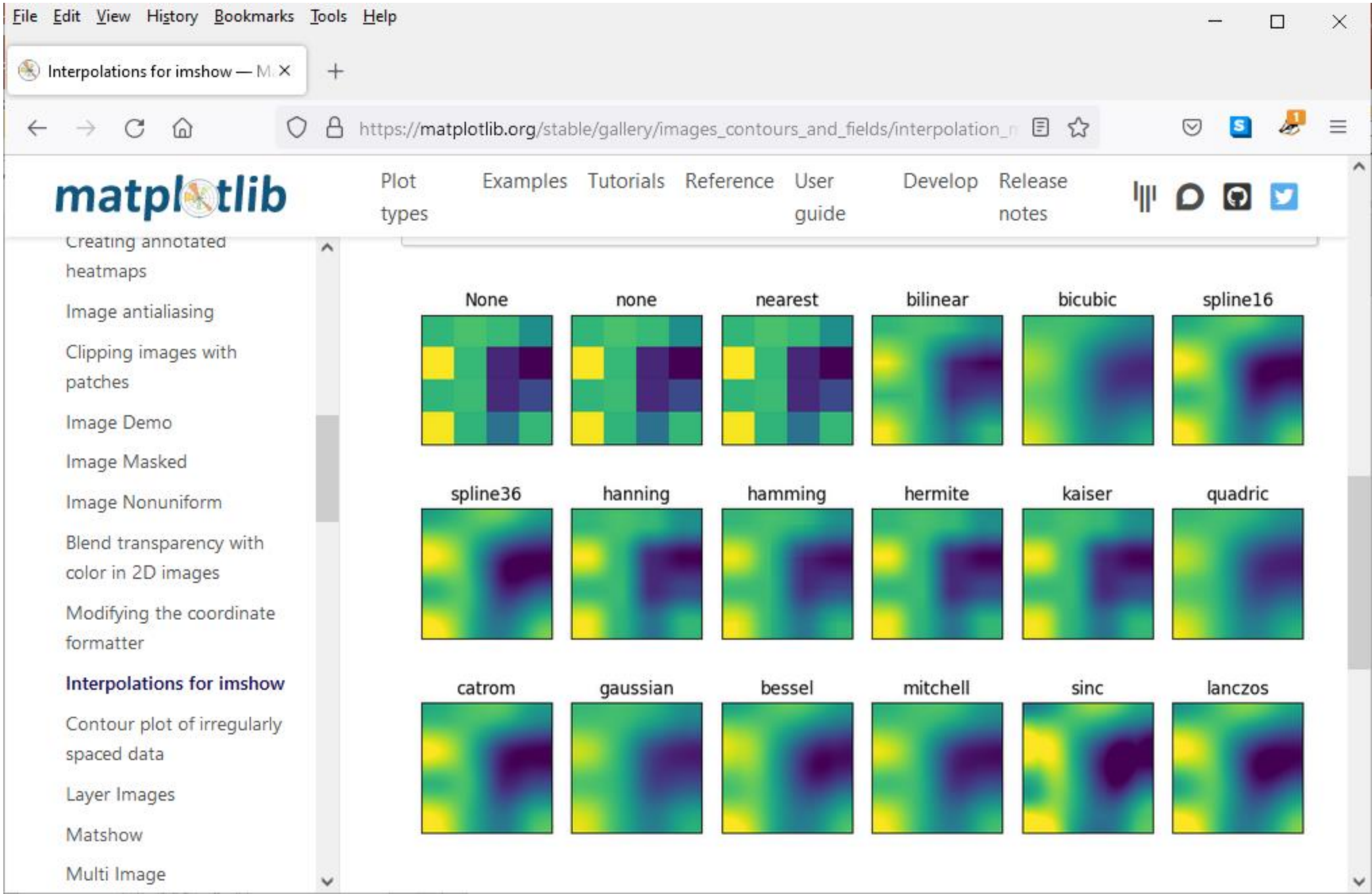
https://matplotlib.org/stable/tutorials/colors/colormaps.html 150%

matplotlib

# Qualitative colormaps



# imshow
## interpolation



# re.findall(), Counter, and WordCloud

IR2A - Notepad

File Edit Format View Help

```
# IR2A.py CS5154/6054 cheng 2022
# read lines from a text file
# tokenize and count words
# use WordCloud to show words with top frequencies
# Usage: python IR2A.py bible.txt

import re
import sys
from collections import Counter
from wordcloud import WordCloud
from matplotlib import pyplot as plt

f = open(sys.argv[1], 'r')

counter = Counter()
for t in f:
    counter.update(re.findall('\w+', t))

wc = WordCloud()
wc.generate_from_frequencies(counter)
plt.imshow(wc)
plt.axis("off")
plt.show()
```


the
and
of
in
And
to
that
shall
LORD
unto
he
his
him
with
which
it
will
all
for
have
said
God
thee
thou
they
them
hath
ye
be
out
by
up
not
thy
my
a
their
Israel
was
but
man
upon
house
David
people
saith
me
when
this
as
is
are
you
from
before
there
great
saying
come
also
on
no
do
made
because
children
land
hand
even
go
let
saying
The
hast
behold
Lord
Jesus
Then
went
things
she
us
son
one
give
shalt
took
father
sons
men
came
thousand
pass
city
every
among
priest
priests
Israel
Jerusalem
Behold
Lord
Christ
Jesus
voice
yet
who
day
heard
thine
knows
sorrow
suffering
therefore
words
called
Jesus
Christ