# Lecture 19: Plotting

CS 51P

November 18, 2019

# Last Week: Dictionaries

- a data structure that stores key:value pairs

```
d = {'apple': .99, 'banana': .19, 'cantelope': 2.99}
```

- restriction:
  - key must be immutable

# Last Week: Dictionary Operations

**adding to a dictionary**

- a_dict[key] = value
- a_dict.update(*b_dict*)

**removing from a dictionary**

- del (a_dict[*key*])
- a_dict.pop(*key*)
  - returns a_dict[key]

**other**

- len(a_dict)
- a_dict.keys()
  - returns list
- a_dict.values()
  - returns list
- a_dict.items()
  - returns list of tuples
- b_dict = a_dict.copy()
  - shallow copy!

# Last Week: Lists vs. dictionaries

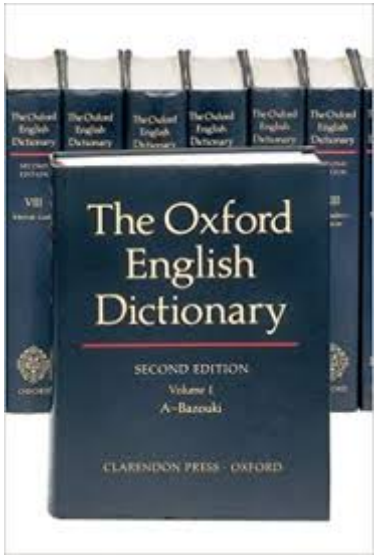
- Both data structures.
- Why would you choose one over the other?
  - a data structure is something that holds a collection of data and that supports certain operations for working with that data

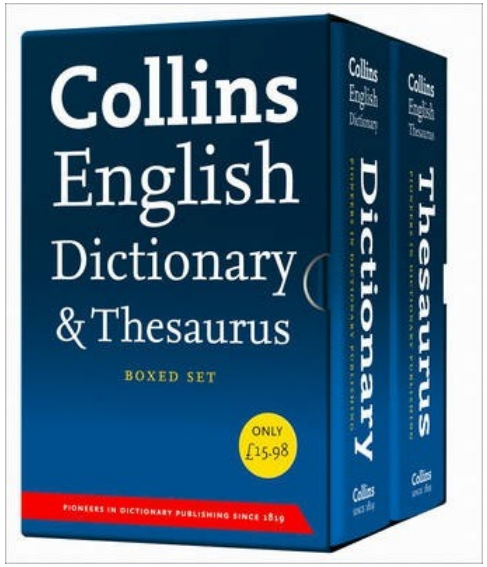
- Lists: sequential access
- Dictionaries: fast lookup

```
def mystery(my_dict):
    d = {}
    for i in my_dict.keys():
        if my_dict[i] in d:
            d[my_dict[i]].append(i)
        else:
            d[my_dict[i]] = [i]
    return d

def main():
    d = {"a":1, "b":2, "c":1, "d":0, "e":2}
    print(mystery(d))

main()
```

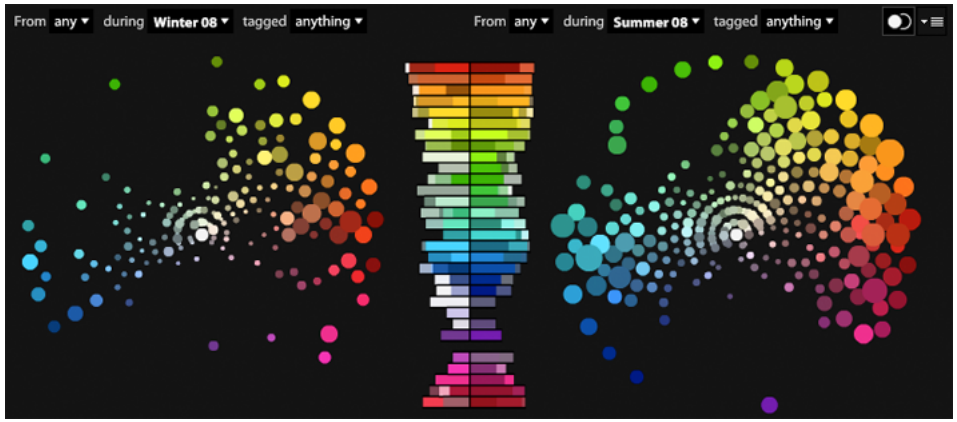
# Exercise

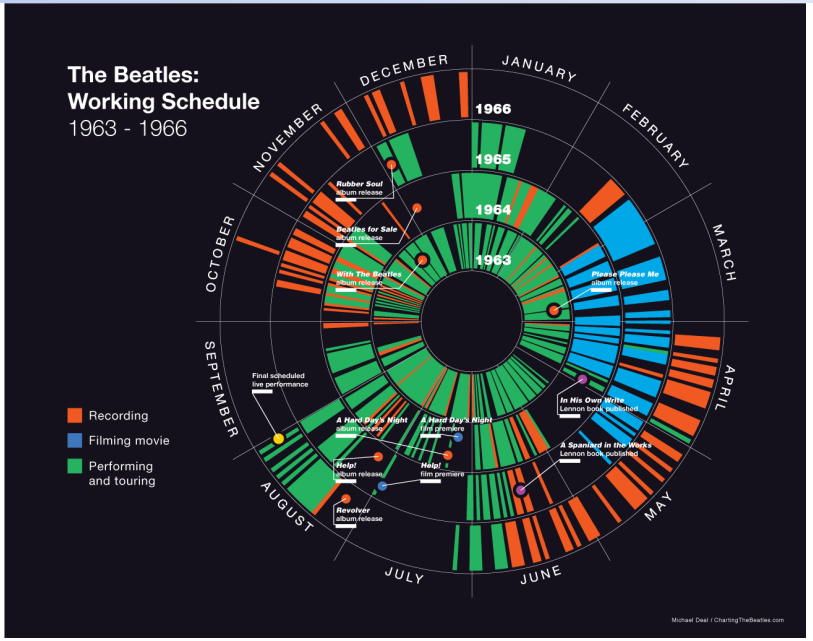
- The function merge(d1,d2) merges two dictionaries and returns a single dictionary containing all the key:value pairs from both input dictionaries. If both dictionaries contain the same key, the new dictionary should contain the smaller of the two values.

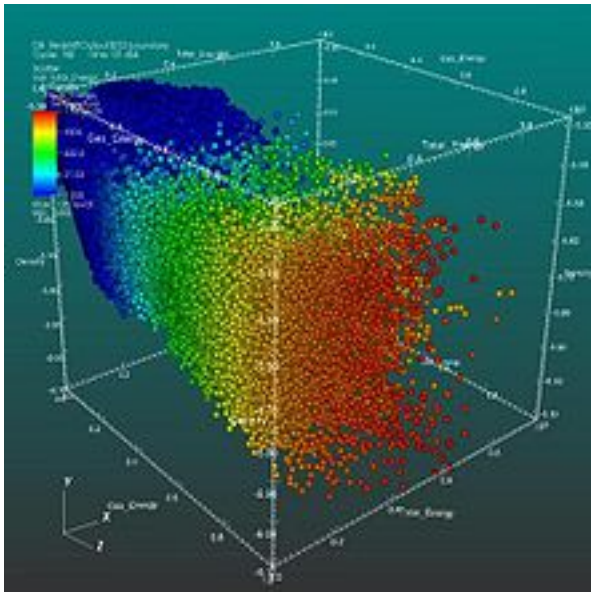


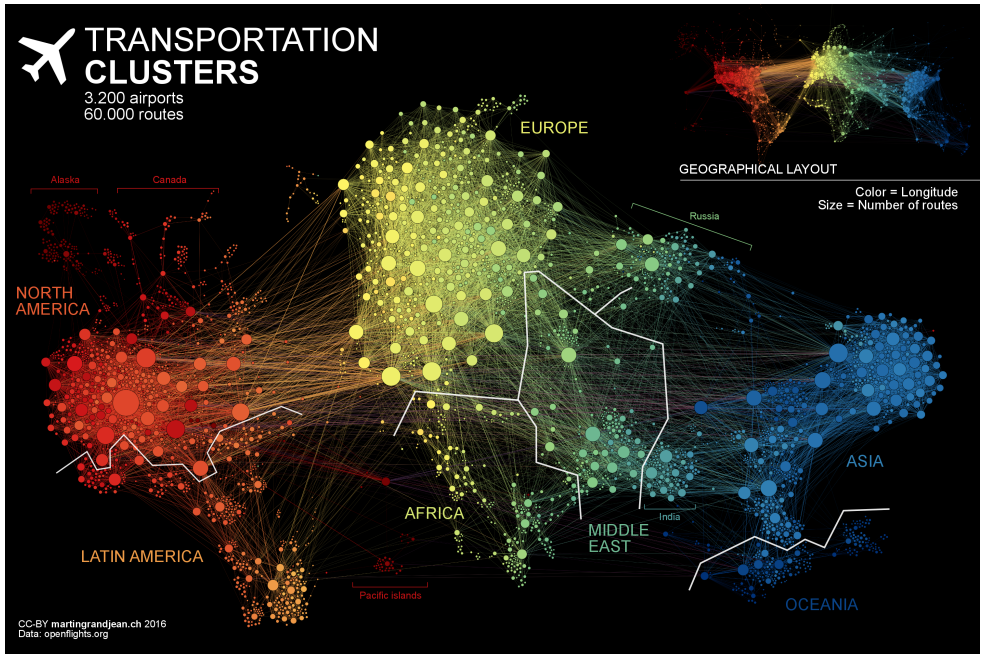



# Data Visualization



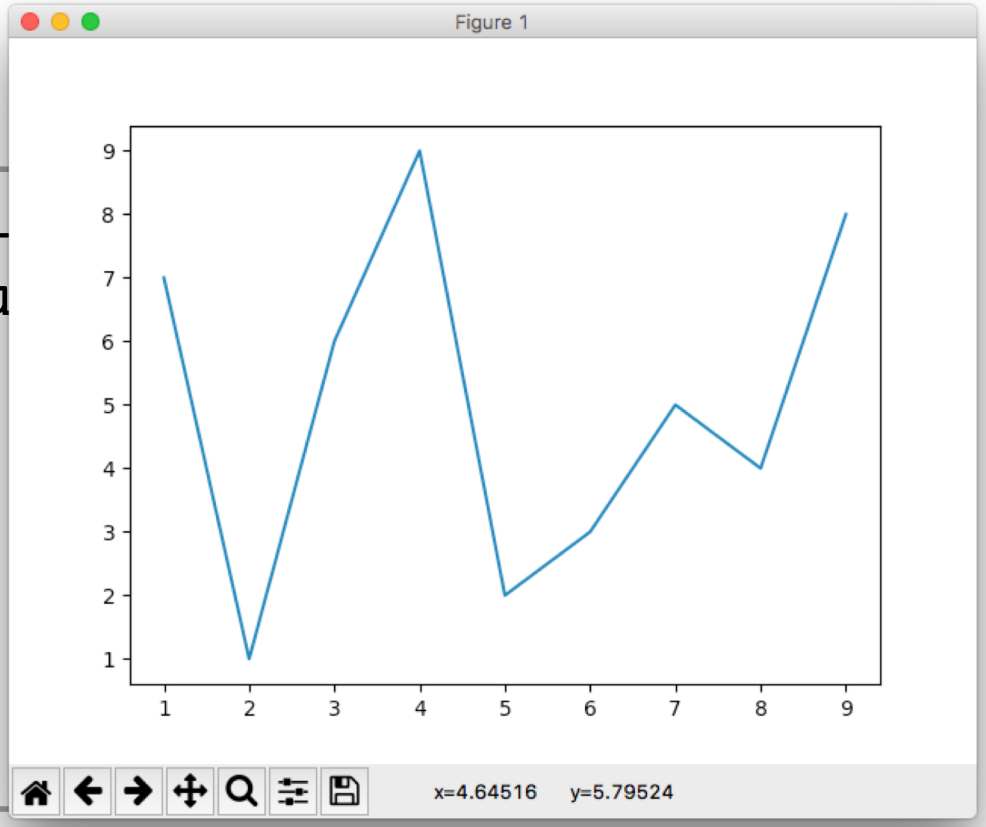





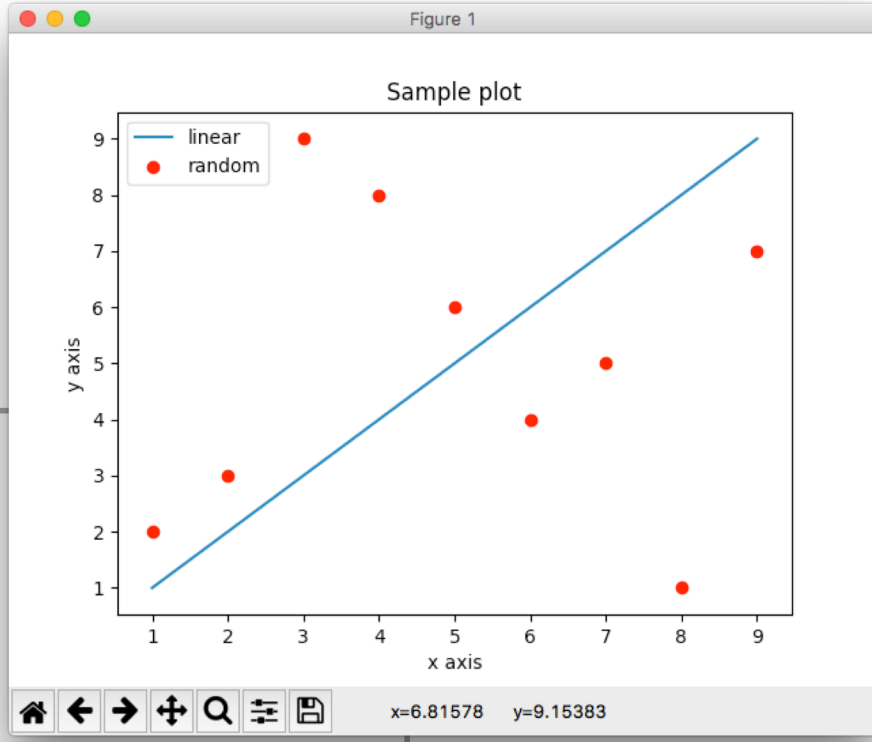
# Data Visualization

- matplotlib
  - https://matplotlib.org/index.html
  - based on Matlab

```
import matplotlib.pypl
from random import shu

x = list(range(1,10))
y = x.copy()
shuffle(y)
plt.plot(x, y)
plt.show()
```



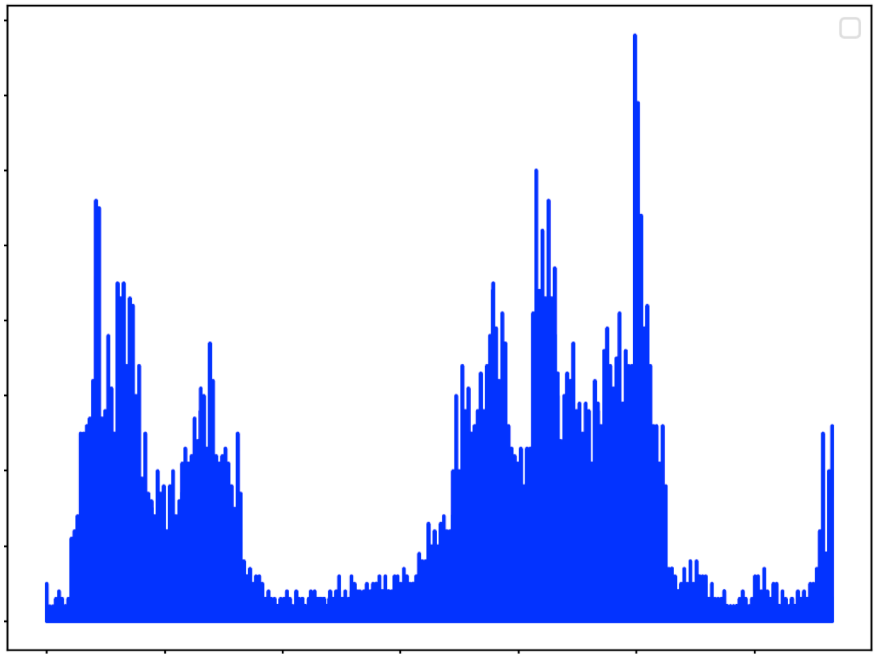
# Customizing



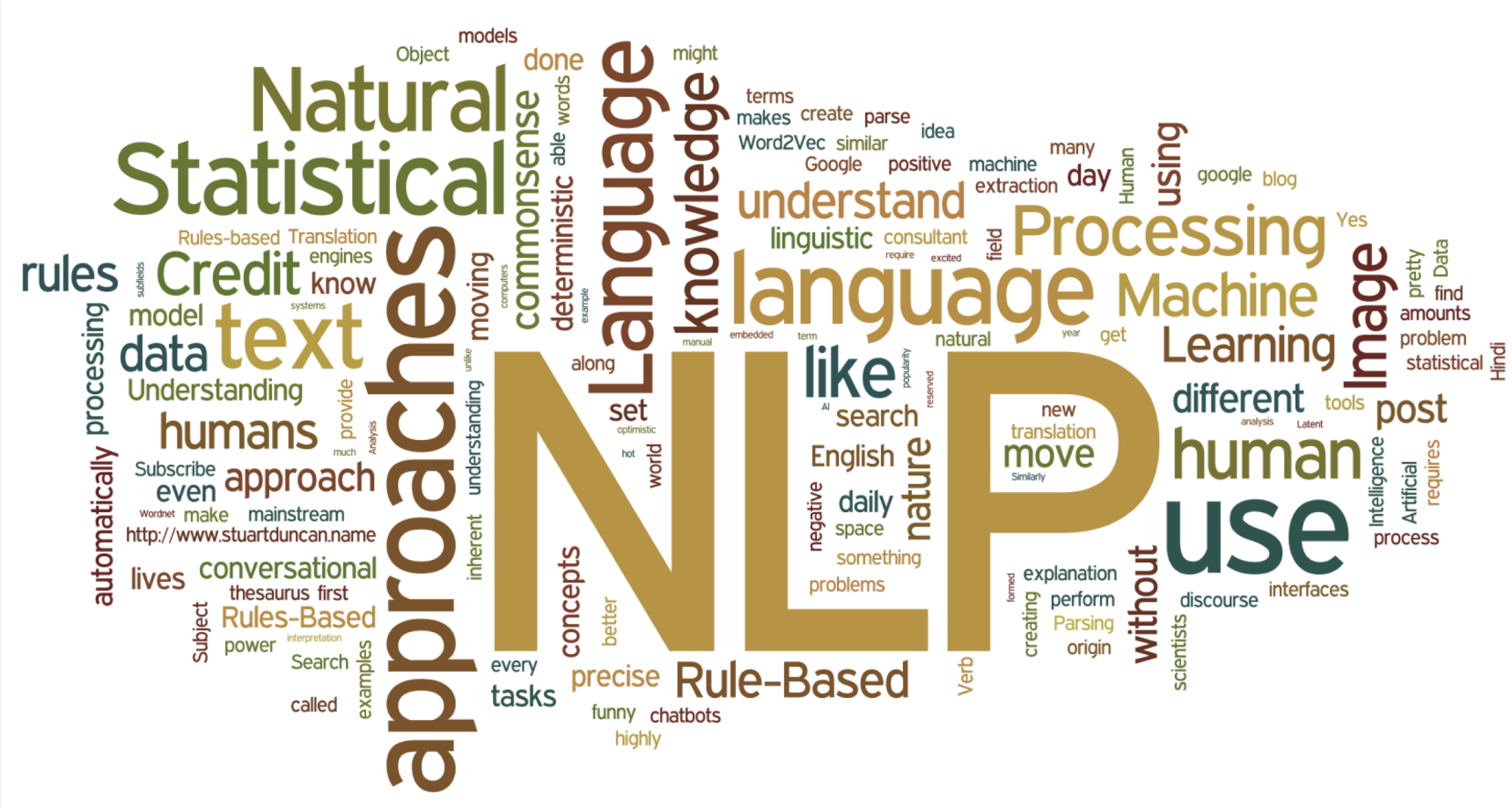

```
import matplotlib.pyplot as plt
from random import shuffle

x = list(range(1,10))
y = x.copy()
shuffle(y)
plt.plot(x, x, label="linear")
p = plt.plot(x, y, 'r.' label="random")
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.title("Sample plot")
plt.legend()
plt.show()
```

# Exercise

- Write a function plot_ppm that takes a filename as input and produces an image histogram (a plot of the number of pixels for each rgb value
  - Hint: r g b values in a ppm file are represented by three numbers (e.g., **250 50 100**). You can represent this pixel as a number **250***(255*255) + **50***(255) + **100**



# Example: Word Count




- Write a function that processes a file and returns a dictionary for handling repeated queries of the form "How many times does the word _____ appear?"