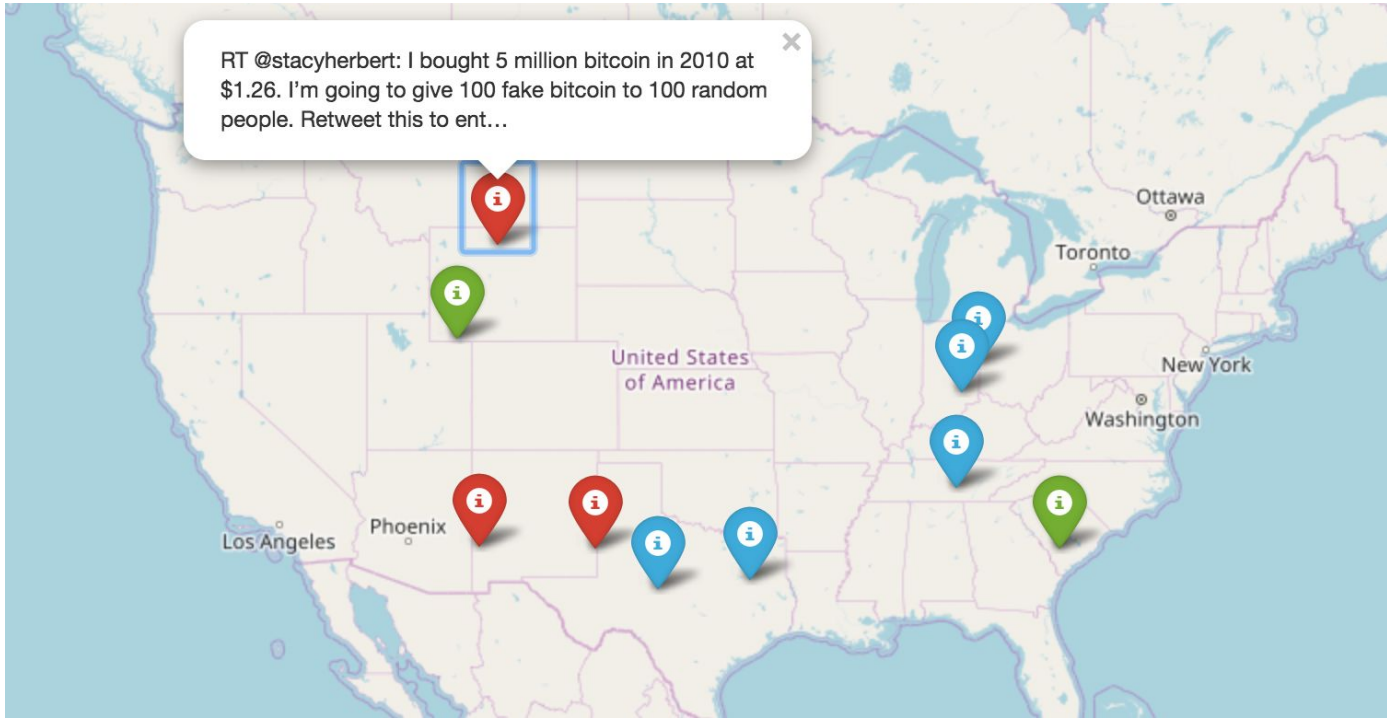
Realtime Sentiment Analysis
Using Tweepy



Shruti Shambhavi (92854)

Overview

- Code Overview
- Jupyter Notebooks
- Tweepy
  - Authentication
  - Basics Features
  - Tweet Extraction
    - Streaming
    - api.search API
- Sentiment Analysis
  - TextBlob
- Plots
  - Folium Maps
- References

Jupyter Notebooks 📌

- an IDE that works on the browser
- supports live code, visualizations and text
- Benefits:
  - Easy to demonstrate code
  - Performs well for testing code snippets
  - Good support for visualizations
  - Ideal for data scientists and scientists
  - Efficient for books, tutorial, code walkthroughs
  - Browser support: easier to install javascript based python libraries

Jupyter Notebooks

Support for both documentation and live code, that can be compiled directly in-browser

Add locations to the map

We start by creating an instance of `folium.Map` object, specifying `centre` of the map as [latitude, longiture], followed by `zoom`

```
folium.Marker([45.3288, -121.6625],
              popup='Mt. Hood Meadows',
              icon=folium.Icon(color='green')).add_to(m)
first param : list of [latitude, longitude]
second param: @popup tool_tip name (set to the tweet text)
third param : icon color
```

```
m = folium.Map(location=[38, -102],
               tiles='Mapbox Bright',
               zoom_start=4)

folium.Marker([45.3288, -121.6625],
              popup='Hello there from Oregon !',
              icon=folium.Icon(color='green')).add_to(m)
folium.Marker([47.3288, -115.6625],
              popup='Its too cold out here !!',
              icon=folium.Icon(color='blue')).add_to(m)
folium.Marker([40.3288, -90.6625],
              popup='Chicago, here I come :)',
              icon=folium.Icon(color='red')).add_to(m)
m
```

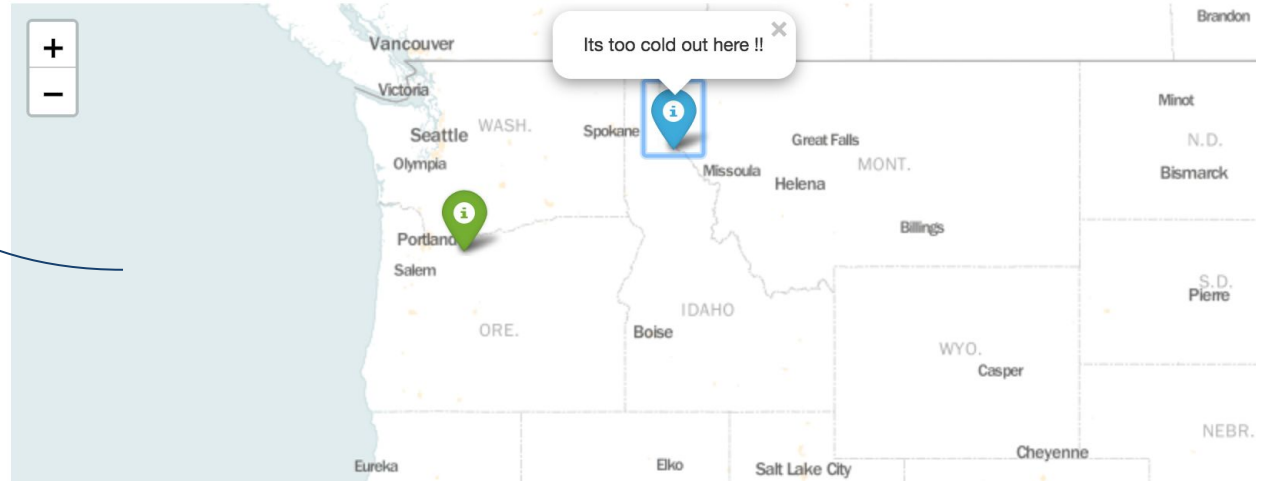
Jupyter Notebooks

In browser visualization

```
]: m = folium.Map(location=[38, -102],
                  tiles='Mapbox Bright',
                  zoom_start=4)

folium.Marker([45.3288, -121.6625],
              popup='Hello there from Oregon !',
              icon=folium.Icon(color='green')).add_to(m)
folium.Marker([47.3288, -115.6625],
              popup='Its too cold out here !!',
              icon=folium.Icon(color='blue')).add_to(m)
folium.Marker([40.3288, -90.6625],
              popup='Chicago, here I come :)',
              icon=folium.Icon(color='red')).add_to(m)
m
```

]:



Tweepy

An easy-to-use Python library for accessing the Twitter API.

- Python library
- Easy Authentication
- Rich list of features that are available through Twitter UI
- Multiple filters for searching and extracting tweets at once
- Additional feature to create Streams for collecting *real time* data on tweets

- Getting started
- Basic Features
  - Update status
  - Find friends
  - Follow / Unfollow
  - Timeline
- Search (at once)
- Search (through streams)

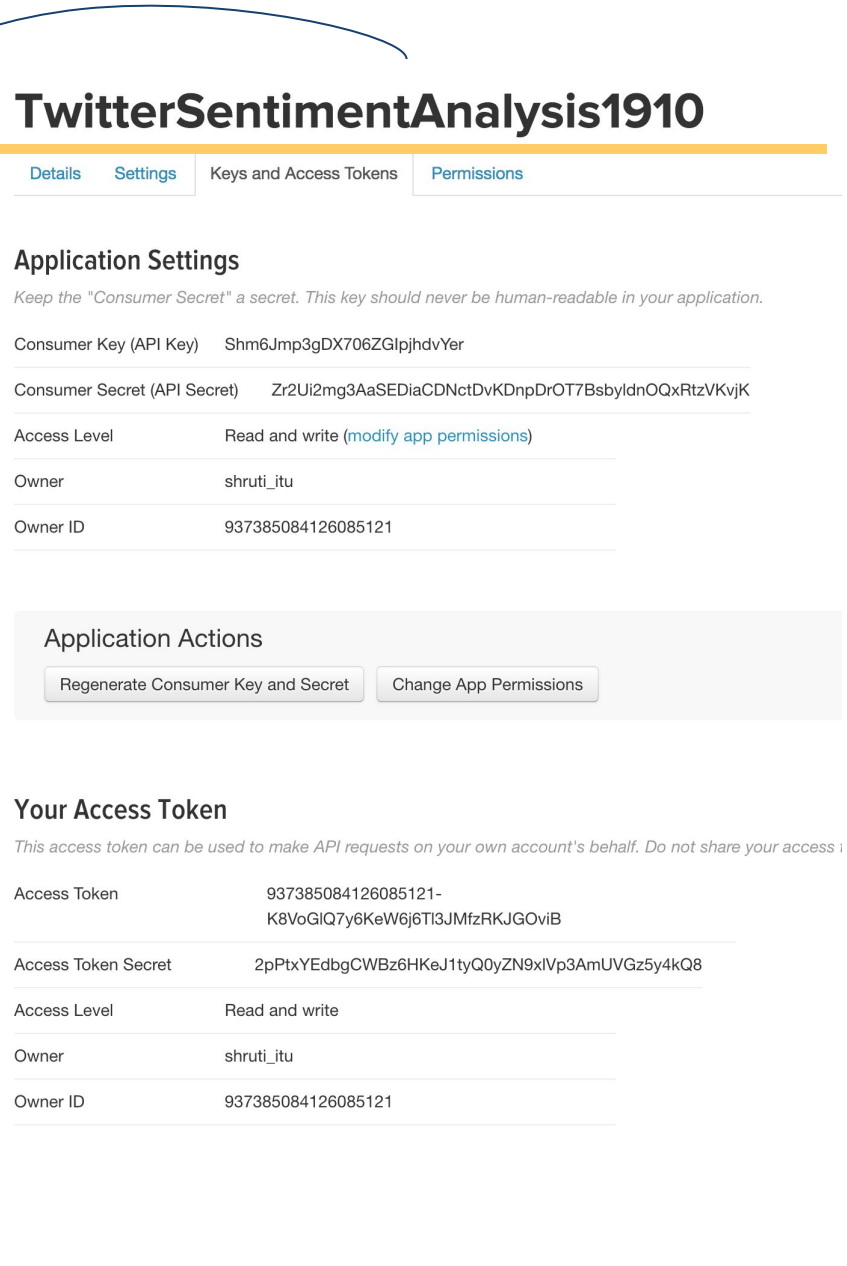
Tweepy: Authentication

- ➔ Create account with Twitter Developers
- ➔ create an app
- ➔ follow the boilerplate code to get access

```
consumer_key = 'Shm6Jmp3gDX706ZGIpjhdvYer'
consumer_secret = 'Zr2Ui2mg3AaSEDiaCDNctDvKDnpDrOT7BsbyldnOQxRtzVKvjK'
access_token = '937385084126085121-K8VoGlQ7y6KeW6j6Tl3JMfzRKJGOviB'
access_token_secret = '2pPtxYEdbgCWBz6HKeJ1tyQ0yZN9xlVp3AmUVGz5y4kQ8'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)
```



Tweepy: Basic Features

Tweepy Basics

```
user = api.me()

# update status
api.update_status('Hello, San Francisco !')

# get list of tweets from one's timeline
print "--------------------------------"
print 'My Timeline:'
for tweet in api.user_timeline():
    print tweet.text
print "--------------------------------"

# print my location
print 'location:', user.location

# print the count of my followers
print 'followers:', len(api.friends_ids())
```

```
--------------------------------
My Timeline:
Hello, San Francisco !
Hello, San Jose !
hey, another tweet coming from tweepy
hello, from tweepy !
--------------------------------
location: San Francisco, CA
followers: 58
```

Tweepy: Tweet Extraction (api.search)

Search for tweets on a particular topic

```
tweets = tweepy.Cursor(api.search, q='Bitcoin', lang='en').items(max_count)
```

```
API.search(q[, lang][, locale][, rpp][, page][, since_id][, geocode][, show_user])
```

Returns tweets that match a specified query.

Parameters:

- `q` – the search query string
- `lang` – Restricts tweets to the given language, given by an ISO 639-1 code.
- `locale` – Specify the language of the query you are sending. This is intended for language-specific clients and the default should work in the majority of cases.
- `rpp` – The number of tweets to return per page, up to a max of 100.
- `page` – The page number (starting at 1) to return, up to a max of roughly 1500 results (based on rpp * page.
- `since_id` – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
- `geocode` – Returns tweets by users located within a given radius of the given latitude/longitude. The location is preferentially taking from the Geotagging API, but will fall back to their Twitter profile. The parameter value is specified by "latitide,longitude,radius", where radius units must be specified as either "mi" (miles) or "km" (kilometers). Note that you cannot use the near operator via the API to geocode arbitrary locations; however you can use this geocode parameter to search near geocodes directly.
- `show_user` – When true, prepends ":" to the beginning of the tweet. This is useful for readers that do not display Atom's author field. The default is false. ""

Tweepy: Tweet Extraction (api.search)

```
8
9  tweets = tweepy.Cursor(api.search, q='Bitcoin', lang='en').items(max_count)
10
```

```
1000
Tom Lee: Buy these stocks to ride the bitcoin fever https://t.co/oxau8QBRlf #bitcoin #jack_dorsey https://t.co/wChZsZ
TuCo
RT @ScottBVS: BITCOIN Theft.....North Korea is 'hacking soaring Bitcoin exchanges', say researchers https://t.co/7hew
0hrzN2
@JaydeathZeus @litecoin If you are trying to get started using Bitcoin Diamond you might to know how to claim free… h
ttps://t.co/z4hsaenTfT
RT @BitfuryGeorge: @rogerkver @ErikVoorhees @BTCTN @JihanWu @JiangZhuoer @JimmyWinMedia @CalvinAyre @officialmcafee @
brian_armstrong @gavin…
RT @arampell: My attempt at the grand unified theory of cryptocurrency — where it came from, how it works, and why it
 matters (and why much…
RT @DOLLABILLGATES: Me after i invested $7 into Bitcoin yesterday https://t.co/VrOW401M0b
RT @msg: I tweeted about bitcoin in 2013 for FREE. I will pick 5 random people who retweet this and give one tweet to
 each of you. Merry Ch…
Don’t treat Bitcoin as a typical investment avenue. Savings accounts, IRAs, Stocks, and Bonds are safer bets.
@AdanJSuarez If you are trying to get started using Bitcoin Diamond you might to know how to claim free 10BCD from… h
ttps://t.co/DeUOxJbGF8
RT @TravWeav: Bought 1,500 bitcoin in 2011 for $2.87 each. I will pick 5 random people who retweet this and give one
 to each of you. Merry…
```

Tweepy: Tweet Extraction (StreamListener)

Step 1: Creating a StreamListener

This simple stream listener prints status text. The **on_data** method of Tweepy's **StreamListener** conveniently passes data from statuses to the **on_status** method. Create class **MyStreamListener** inheriting from **StreamListener** and overriding **on_status.**:

```
import tweepy
#override tweepy.StreamListener to add logic to on_status
class MyStreamListener(tweepy.StreamListener):

    def on_status(self, status):
        print(status.text)
```

Tweepy: Tweet Extraction (StreamListener)

Step 2: Creating a Stream

We need an api to stream. See Authentication Tutorial to learn how to get an api object. Once we have an api and a status listener we can create our stream object.:

```
myStreamListener = MyStreamListener()
myStream = tweepy.Stream(auth = api.auth, listener=myStreamListener())
```

Tweepy: Tweet Extraction (StreamListener)

Step 3: Starting a Stream

A number of twitter streams are available through Tweepy. Most cases will use filter, the user_stream, or the sitestream. For more information on the capabilities and limitations of the different streams see Twitter Streaming API Documentation.

In this example we will use **filter** to stream all tweets containing the word *python*. The **track** parameter is an array of search terms to stream.

```
myStream.filter(track=['python'])
```

Tweepy: Tweet Extraction (StreamListener)

```
customer_listener = CustomListener(api)
listener = tweepy.Stream(auth, customer_listener)
# listener.filter(track=['Bitcoin'])
listener.filter(locations=GEOBOX_WORLD)
```

```
GEOBOX_WORLD = [-180,-90,180,90]

class CustomListener(tweepy.StreamListener):
    MAX_COUNT = 1000

    def __init__(self, api):
        self.api = api
        self.num_tweets = 0
        self.tweet_analyzer = TweetAnalyser()

        self.results = []

    def on_status(self, tweet):
        """
        Called when raw data is received from connection.
        Override this method if you wish to manually handle
        the stream data. Return False to stop stream and close connection.
        """

        data = {} # initialize empty dictionary

        # consider tweets only if they have geo locations
        if tweet.coordinates is not None:
            # and 'bitcoin' in tweet.text

            data['text'] = tweet.text # extract tweet's content
            data['coordinates'] = tweet.coordinates['coordinates'] # add coordinate to date
            data['score'] = self.tweet_analyzer.get_sentiment_score(tweet) # calculate sentiment score

            self.num_tweets += 1
            if self.num_tweets > self.MAX_COUNT:
                return False

            print data['text']
            print data['coordinates']
            self.results.append(data)
```

Tweepy: Tweet Composition

```
created_at: 2017-12-09 03:45:24
geo: None
id: 939340175846625280
lang: en
place: None
text: @Garurusama Take out payday loans and use them to buy bitcoin
```

```
{
 'contributors': None,
 'truncated': False,
 'text': 'My Top Followers in 2010: @tkang1 @serin23 @uhrunland @aliassculpto
 'in_reply_to_status_id': None,
 'id': 21041793667694593,
 '_api': <tweepy.api.api object="" at="" 0x6bebc50="">,
 'author': <tweepy.models.user object="" at="" 0x6c16610="">,
 'retweeted': False,
 'coordinates': None,
 'source': 'My Top Followers in 2010',
 'in_reply_to_screen_name': None,
 'id_str': '21041793667694593',
 'retweet_count': 0,
 'in_reply_to_user_id': None,
 'favorited': False,
 'retweeted_status': <tweepy.models.status object="" at="" 0xb2b5190="">,
 'source_url': 'http://mytopfollowersin2010.com',
 'user': <tweepy.models.user object="" at="" 0x6c16610="">,
 'geo': None,
 'in_reply_to_user_id_str': None,
 'created_at': datetime.datetime(2011, 1, 1, 3, 15, 29),
 'in_reply_to_status_id_str': None,
 'place': None
}
```

Sentiment Analysis: TextBlob

Sentiment Analysis using textblob

> *text = "Wow, Bitcoin prices really soared up this morning !"*

```
analysis = TextBlob(text)
> print analysis.sentiment
Sentiment(polarity=0.6, subjectivity=1.0)

> print analysis.sentiment.polarity
0.6
```

```
: print TextBlob('This has been a wonderful day !').sentiment.polarity
  print TextBlob('My phone broke yesterday.').sentiment.polarity
  print TextBlob('Pizza from that place are awfully bad').sentiment.polarity
```

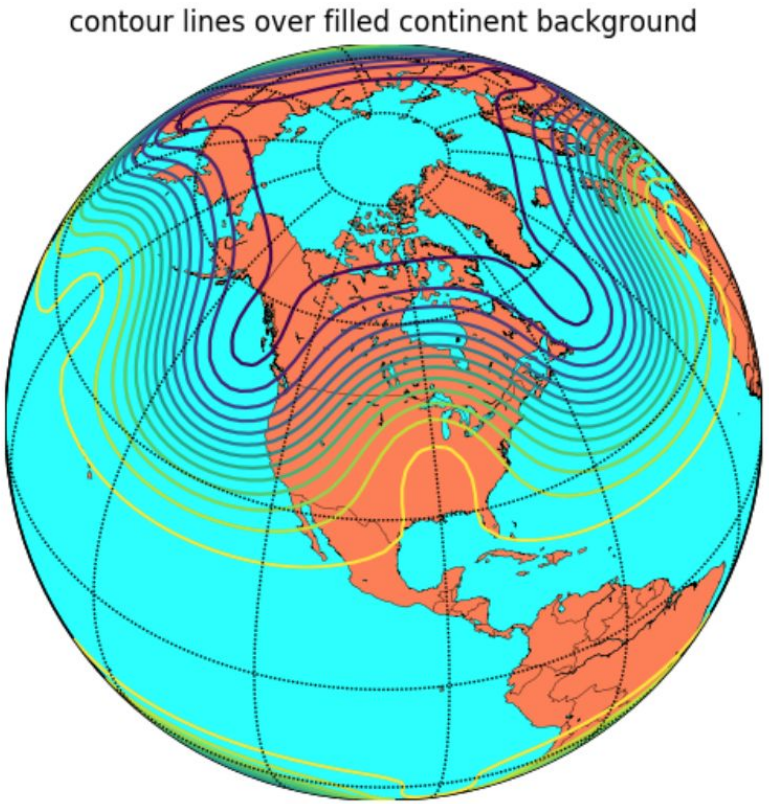
```
1.0
0.0
-0.7
```

Plotting in Python

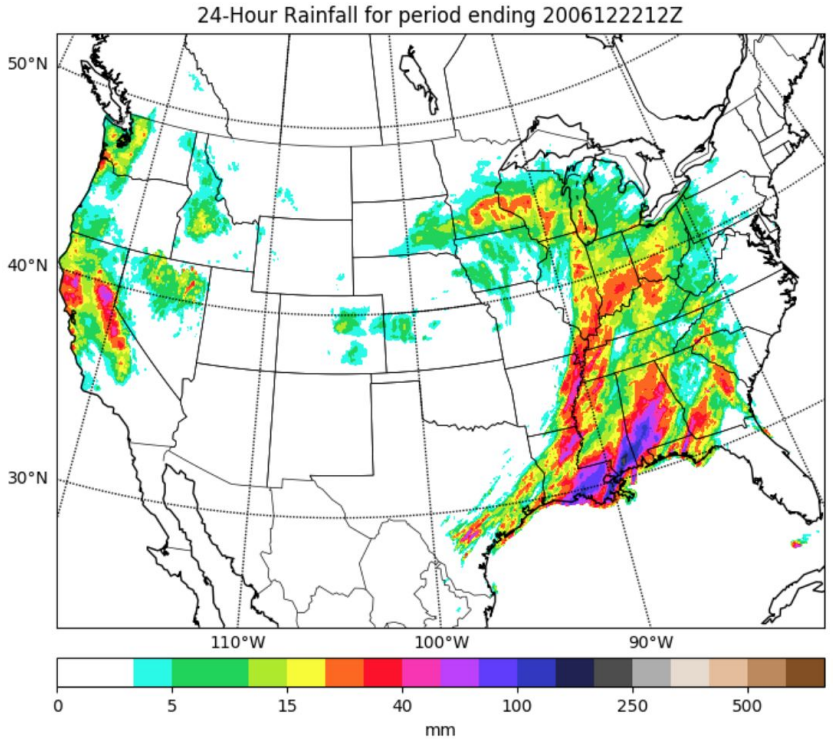
- Tools
  - Matplotlib
    - BaseMap
  - Plotly
  - D3 / Vega Translators
    - Folium

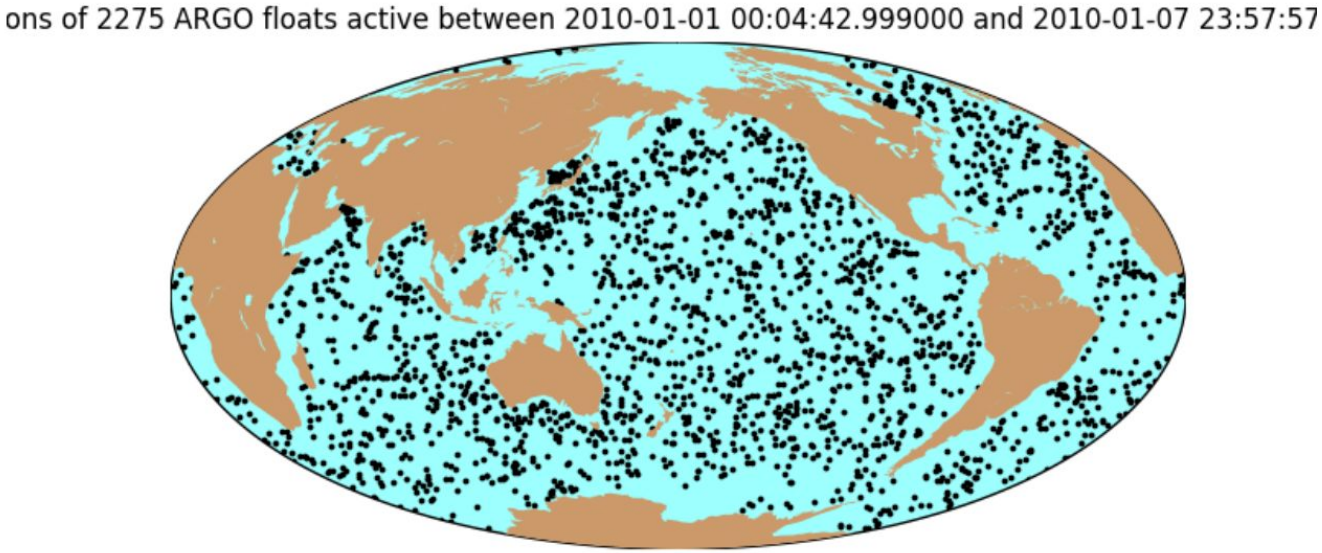
Plotting in Python: Matplotlib

- Benefits
  - Easy to install
  - Works directly with python data structures
  - Lots of different styles available
  - Variety of plots available
    - Statistical Plots
    - Colours and contours
  - Subplots
- Disadvantages
  - Render quality poor
  - Lot of code need to written
  - *Bad support for maps*



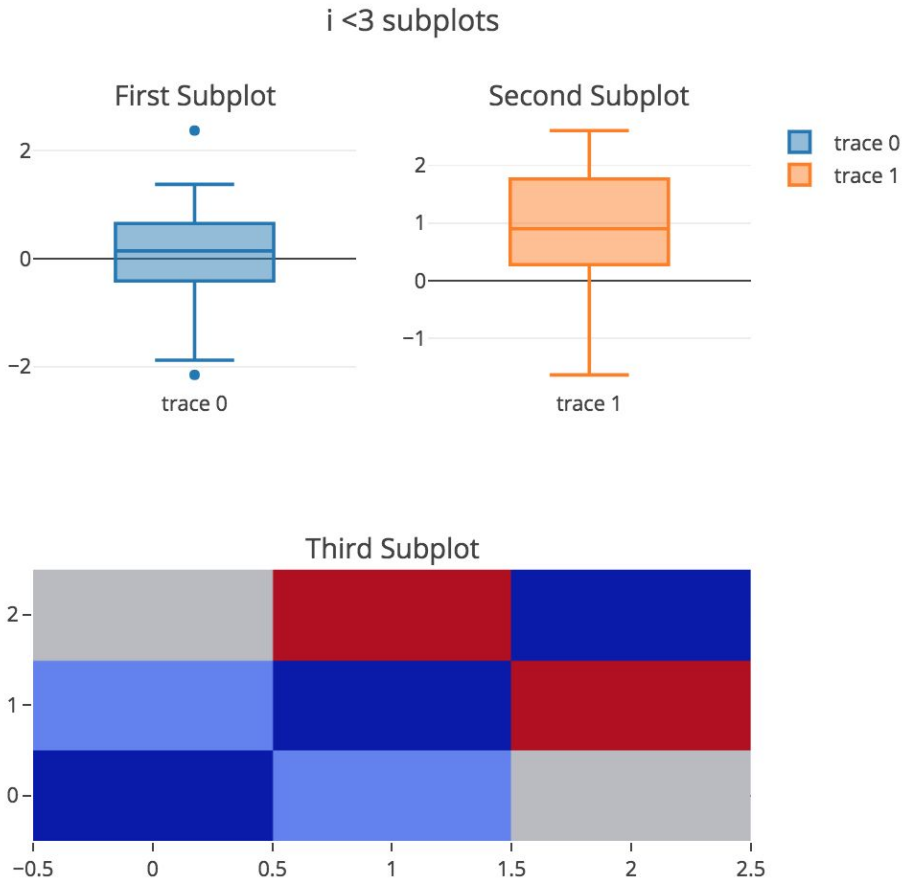
Plotting in Python: Matplotlib





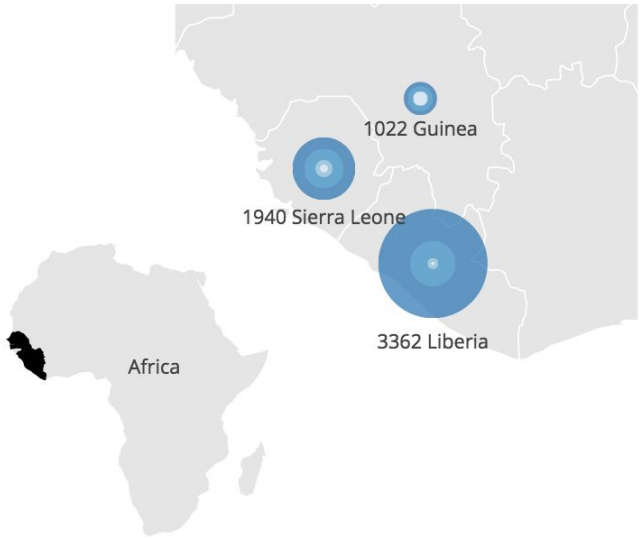
Plotting in Python: Plotly

- Benefits
  - Works great with Pandas
  - Ideal for sharing over the web
  - Interactive
    - Zoom in and zoom out
    - Tooltips available on hovering
  - Good rendering quality
- Disadvantages
  - Not free for premium features
  - Like matplotlib, has a big boilerplate
  - Steep learning curve for beginners

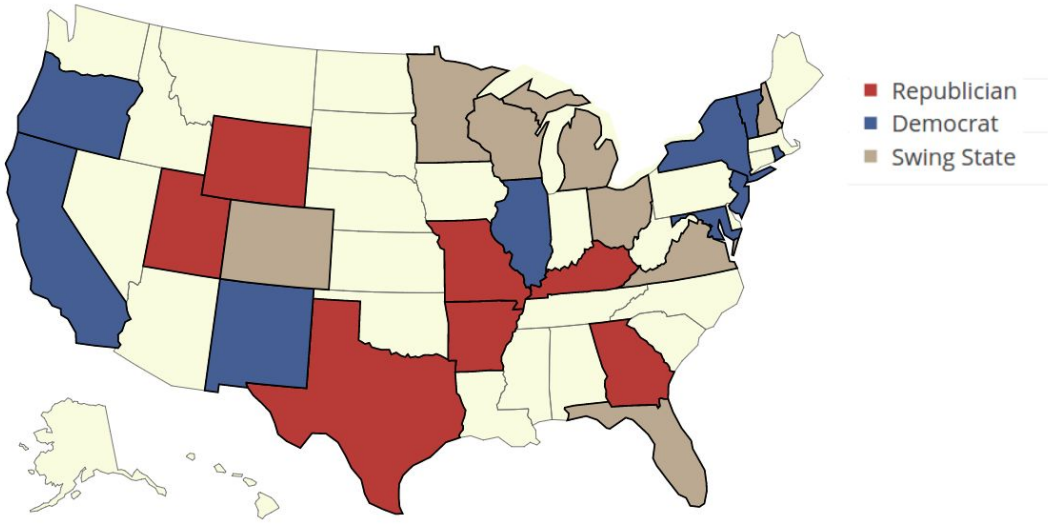


Plotting in Python: Plotly

Ebola cases reported by month in West Africa 2014
Source: HDX

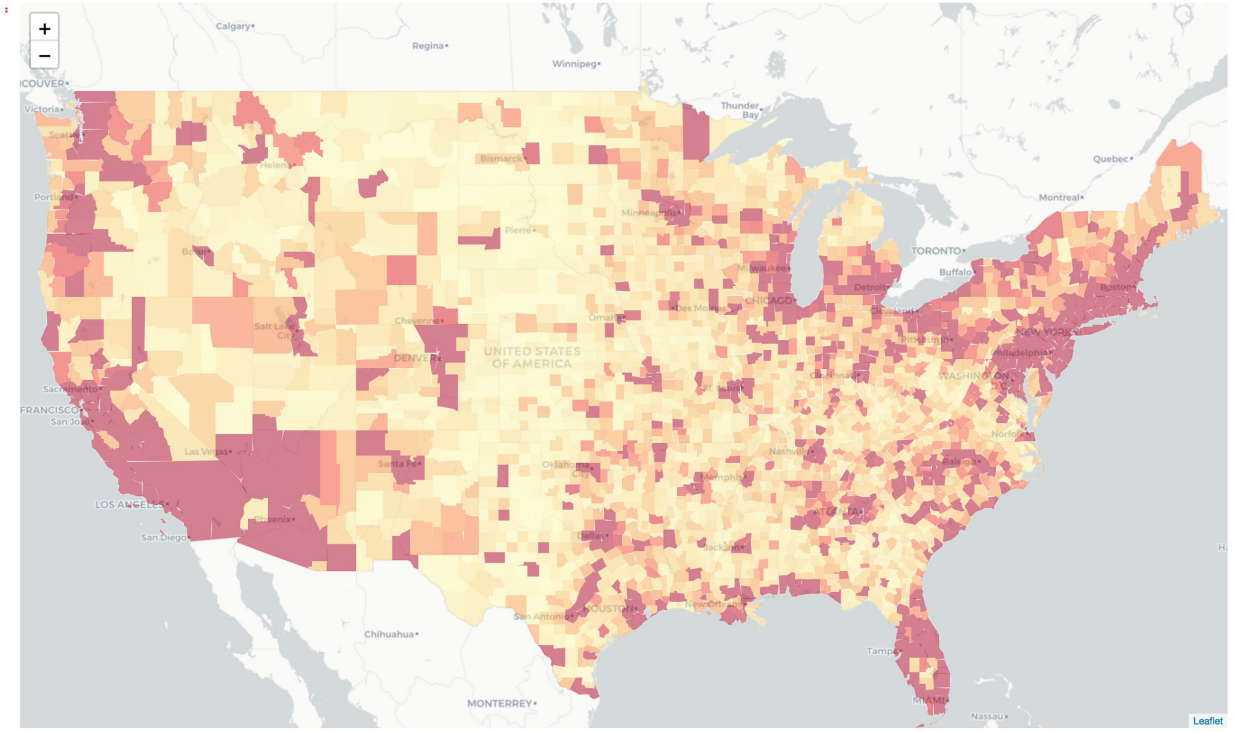


PACE Approved legislation

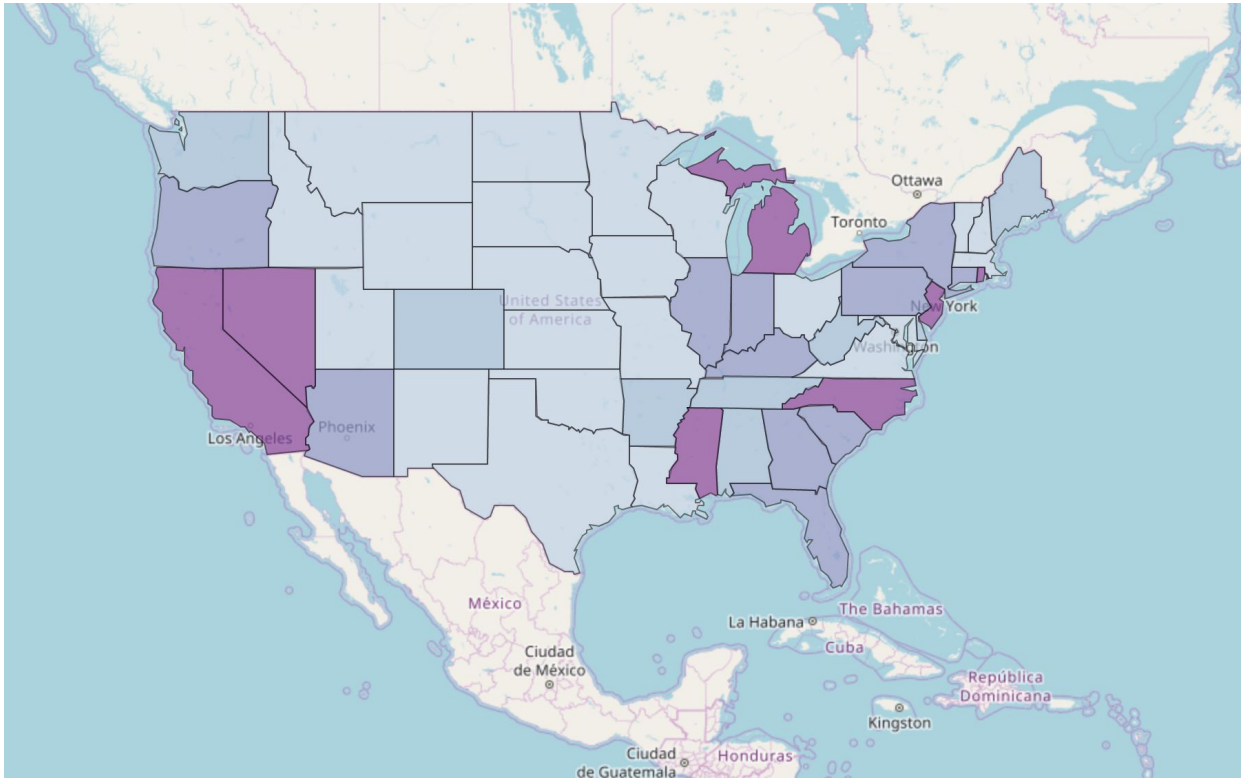


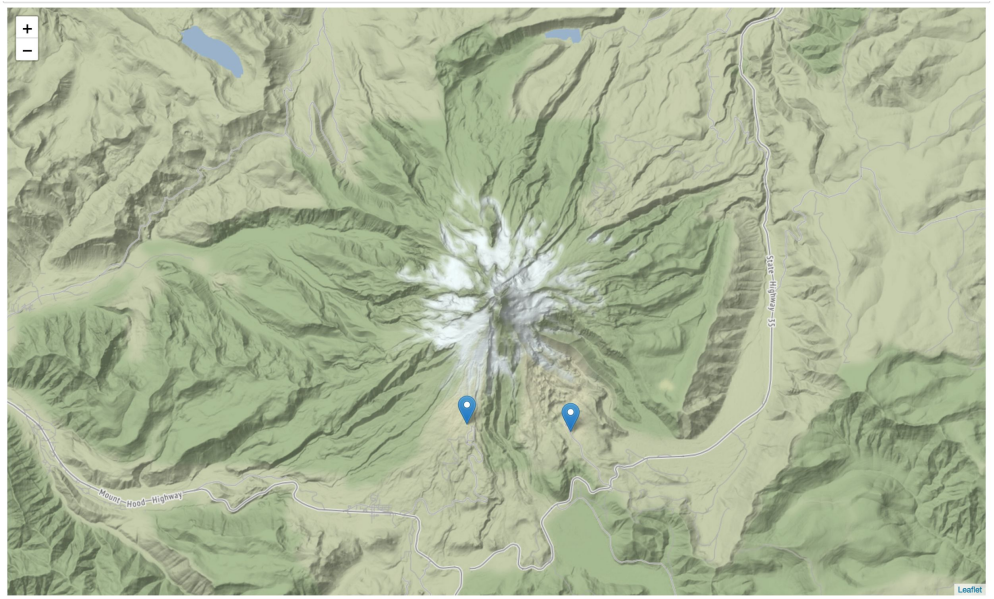
Plotting in Python: Folium

- Benefits
  - Based on D3/Vega Python Translators
  - Wonderful rendering
  - Interactive Plots built-in
  - One of the best supports for maps
  - Simple learning curve for beginners
- Disadvantages
  - Limited to Maps



Plotting in Python: Folium *choropleth*





Plotting in Python: Folium Initialize

```
import folium

m = folium.Map(location=[38, -102], zoom_start=4)
m
```

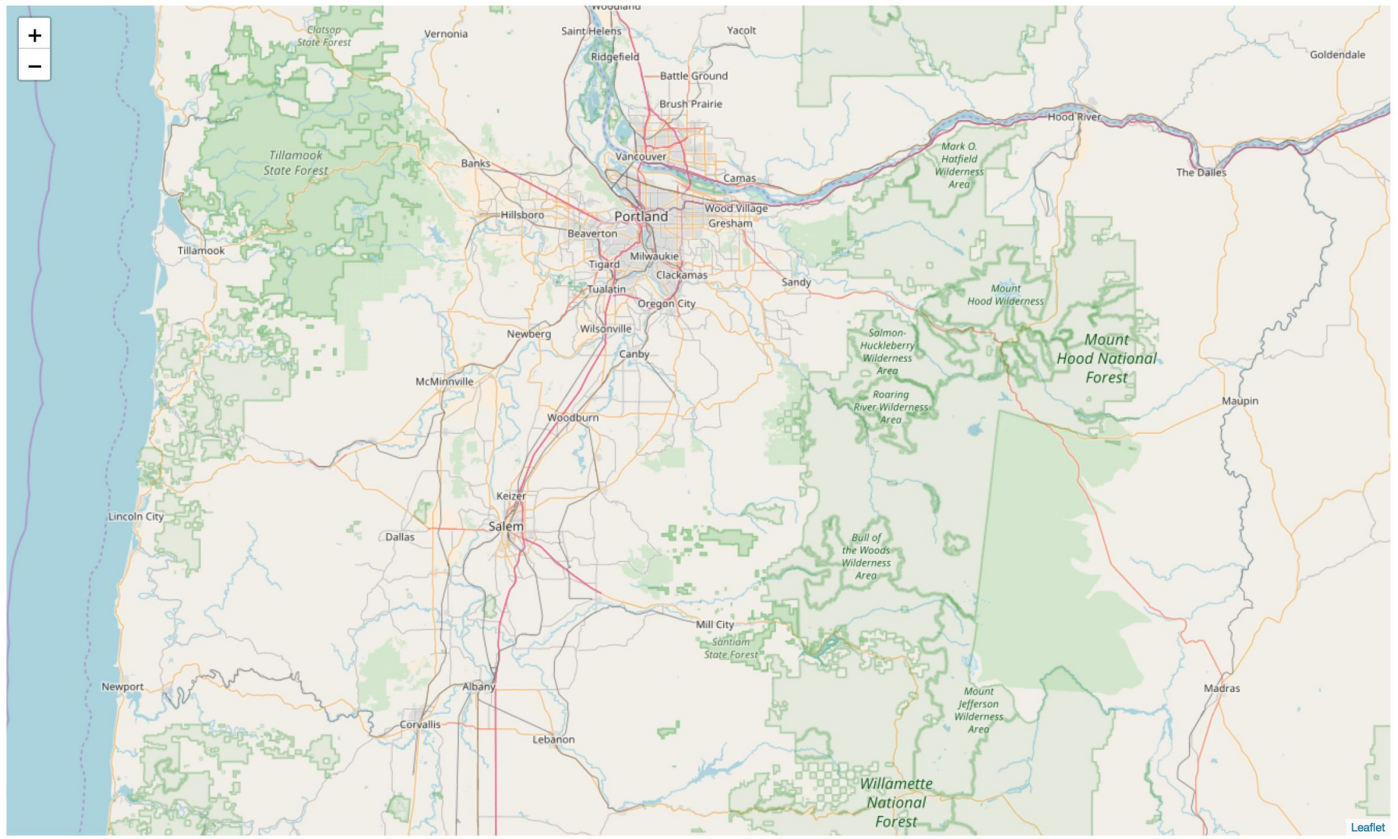
In [1]:
```
import folium

m = folium.Map(location=[45.5236, -122.6750])
```

To display it in a Jupyter notebook, simply ask for the object representation:

In [2]:
```
m
```

Out[2]:



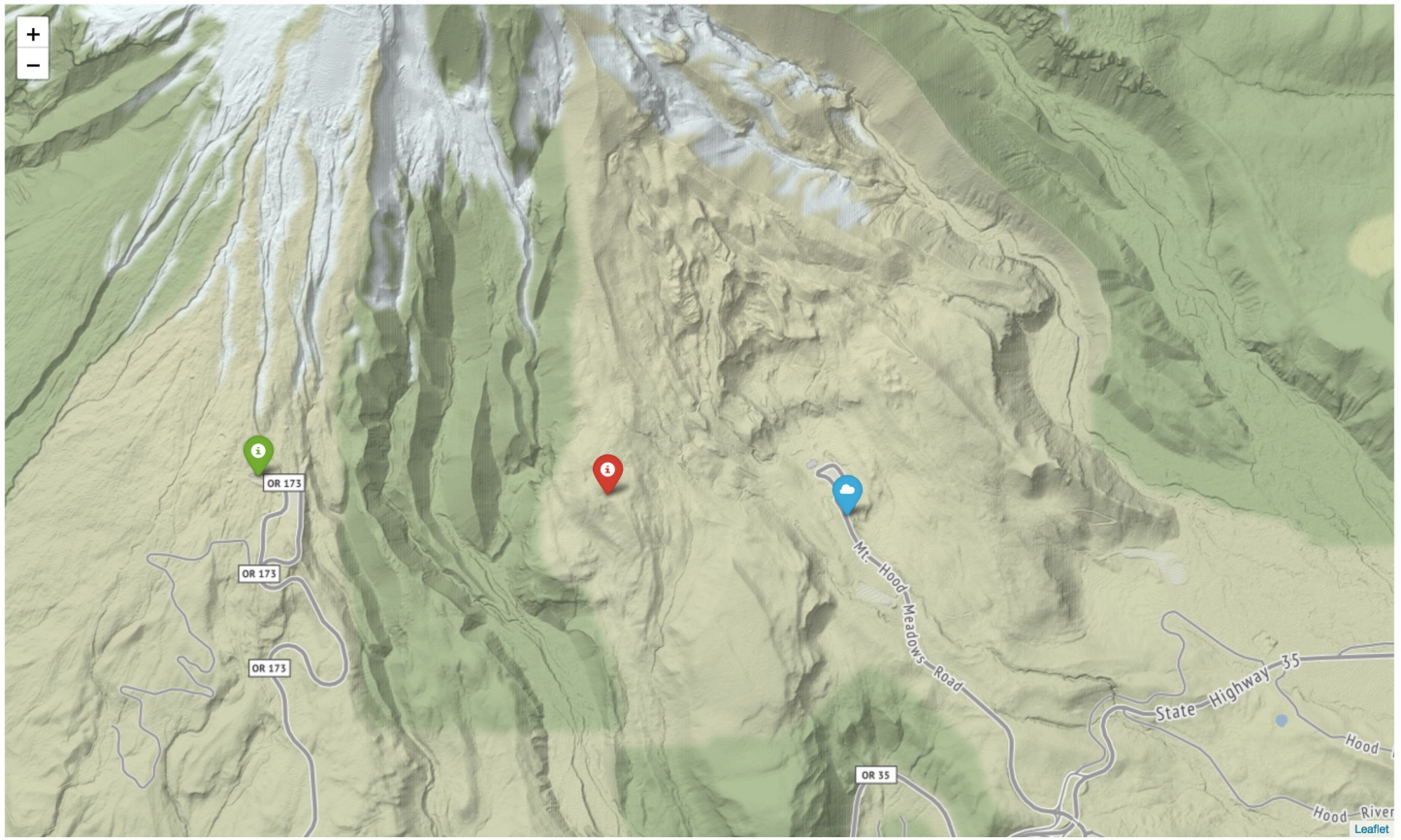
Plotting in Python: Folium Add Markers

```
m = folium.Map(location=[38, -102],
               tiles='Mapbox Bright',
               zoom_start=4)

folium.Marker([45.3288, -121.6625],
              popup='Hello there from Oregon !',
              icon=folium.Icon(color='green')).add_to(m)
folium.Marker([47.3288, -115.6625],
              popup='Its too cold out here !!',
              icon=folium.Icon(color='blue')).add_to(m)
folium.Marker([40.3288, -90.6625],
              popup='Chicago, here I come :)',
              icon=folium.Icon(color='red')).add_to(m)
m
```


```
folium.Marker(
    location=[45.3300, -121.6823],
    popup='Some Other Location',
    icon=folium.Icon(color='red', icon='info-sign')
).add_to(m)


m
```



Plotting in Python: Folium Deciding Markers

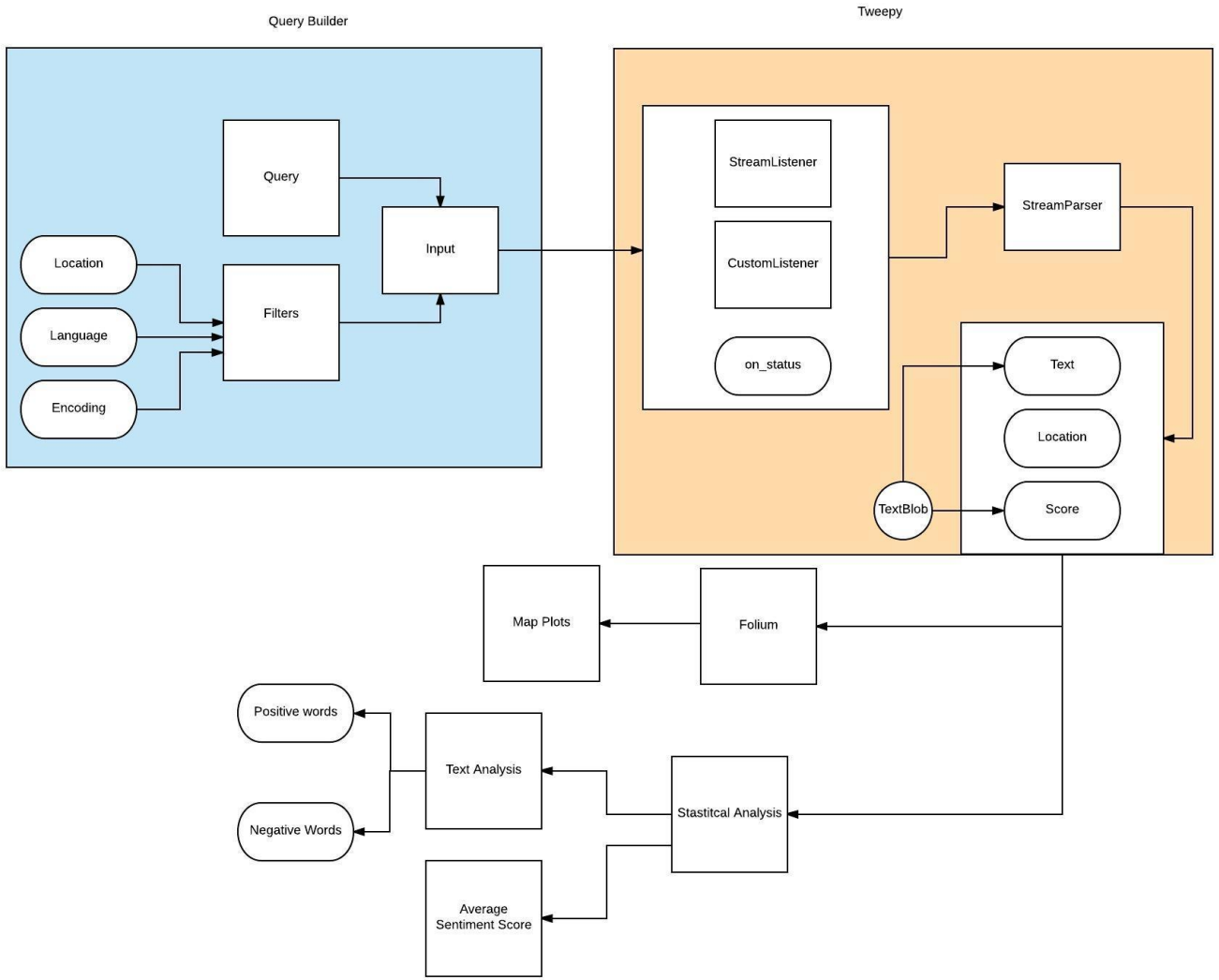
```
for data in sample_list:

    score = data['score']
    if score < 0:
        color = 'red'
    elif score > 0:
        color = 'green'
    else:
        color = 'blue'

    folium.Marker(data['location'],
                  popup=data['text'],
                  icon=folium.Icon(color=color)).add_to(m)
```

Plotting in Python: Folium Final Render

Review (Code Review)



References

References

Tweepy

- `github`: https://github.com/tweepy/tweepy
- `searchAPI`: http://docs.tweepy.org/en/v3.5.0/api.html#API.search
- `me`: http://docs.tweepy.org/en/v3.5.0/api.html#API.me
- `streaming code`: https://github.com/tweepy/tweepy/blob/master/tweepy/streaming.py
- `StreamListener`: https://github.com/tweepy/tweepy/blob/master/tweepy/streaming.py#L31
- `filter`: https://github.com/tweepy/tweepy/blob/master/tweepy/streaming.py#L31
- `on_status`: https://github.com/tweepy/tweepy/blob/master/tweepy/streaming.py#L45

TextBlob

- http://textblob.readthedocs.io/en/dev/

Folium

- Github: https://github.com/python-visualization/folium
- Documentation: http://python-visualization.github.io/folium/

Pickle

- Documentation: https://docs.python.org/2/library/pickle.html

Cleaning Tweets

- reference: https://dev.to/rodolfoferro/sentiment-analysis-on-trumpss-tweets-using-python-

Thank You !