Python primer

Mining the social web with python

Goals for todays talk

- Highlight the elegance of the python programming language
- Quick demonstration on how python can be used to “easily” mine social media data
- Introduce some interesting python libraries
  - “Someone has already done that!”
  - huge selection of libraries
- A tweet is much more than 140 characters!

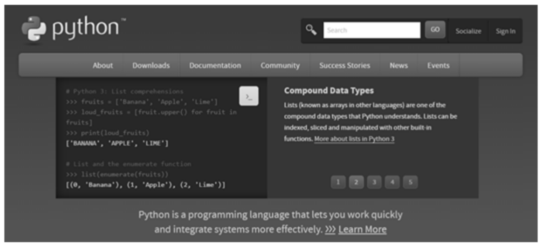
Tools and libraries

- If you take home only one thing the Research Bazar, make it Linux!

Setting up a Lubuntu virtual machine with virtual box [and get guest additions working]

https://dwheelerau.com/2014/01/25/setting-up-a-lubuntu-virtual-machine-with-virtual-box/

Tools and libraries



```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

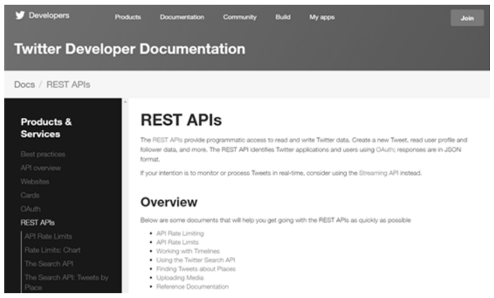
Compound Data Types

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. More about lists in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

https://www.python.org/

Tools and libraries



Twitter Developer Documentation

REST APIs

The REST APIs provide programmatic access to read and write Twitter data. Create a new Tweet, read user profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are in JSON format.

If your intention is to monitor or process Tweets in real-time, consider using the Streaming API instead.

Overview

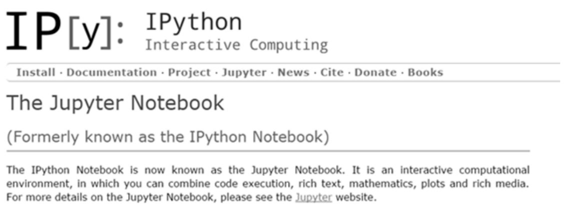
Below are some documents that will help you get going with the REST APIs as quickly as possible

- API Rate Limiting
- API Rate Limits
- Working with Timelines
- Using the Twitter Search API
- Finding Tweets about Places
- Uploading Media
- Reference Documentation

```
~/RESEARCH/presentations/resbazar_2017
dwheeler@dwheeler-HP$ (git::master) pip install python-twitter
```

https://github.com/bear/python-twitter https://dev.twitter.com/rest/public

Tools and libraries


IP[y]:
IPython
Interactive Computing
Install · Documentation · Project · Jupyter · News · Cite · Donate · Books

The Jupyter Notebook

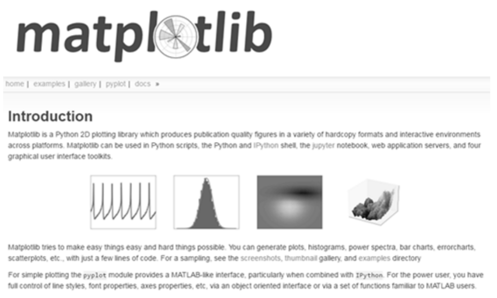
(Formerly known as the IPython Notebook)

The IPython Notebook is now known as the Jupyter Notebook. It is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media. For more details on the Jupyter Notebook, please see the Jupyter website.

```
~/RESEARCH/presentations/resbazar_2017
dwheeler@dwheeler-HP$ (git::master) pip install jupyter
```

https://ipython.org/notebook.html

Tools and libraries



Introduction

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

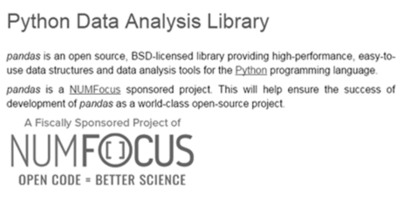
Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For a sampling, see the screenshots, thumbnail gallery, and examples directory

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with `IPython`. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

```
~/RESEARCH/presentations/resbazar_2017
dwheeler@dwheeler-HP$ (git::master) pip install matplotlib
```

http://matplotlib.org/

Tools and libraries



Python Data Analysis Library

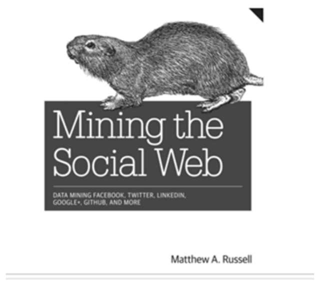
pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

pandas is a NUMFocus sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project.

```
~/RESEARCH/presentations/resbazar_2017
dwheeler@dwheeler-HP$ (git::master) pip install pandas
```

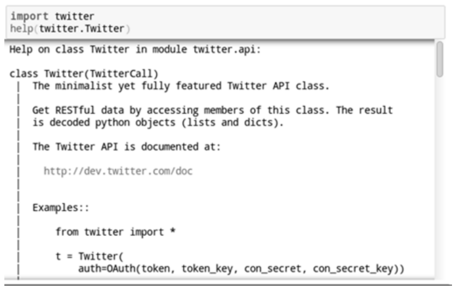
http://pandas.pydata.org/

Tools and libraries


Mining the Social Web
DATA MINING FACEBOOK, TWITTER, LINKEDIN, GOOGLE+, GITHUB, AND MORE
Matthew A. Russell

http://shop.oreilly.com/product/0636920030195.do

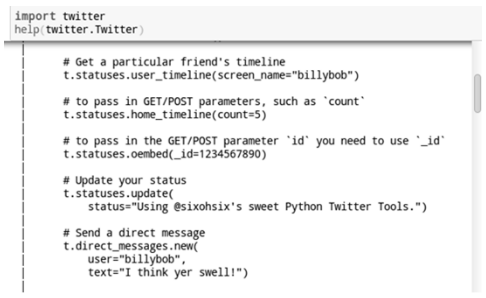
The python Twitter library



```
import twitter
help(twitter.Twitter)
```

```
Help on class Twitter in module twitter.api:

class Twitter(TwitterCall)
 |  The minimalist yet fully featured Twitter API class.
 |
 |  Get RESTful data by accessing members of this class. The result
 |  is decoded python objects (lists and dicts).
 |
 |  The Twitter API is documented at:
 |
 |    http://dev.twitter.com/doc
 |
 |
 |  Examples::
 |
 |      from twitter import *
 |
 |      t = Twitter(
 |          auth=OAuth(token, token_key, con_secret, con_secret_key))
```

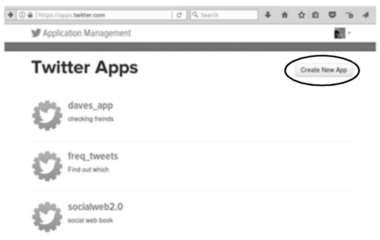
The python Twitter library



```
import twitter
help(twitter.Twitter)
```

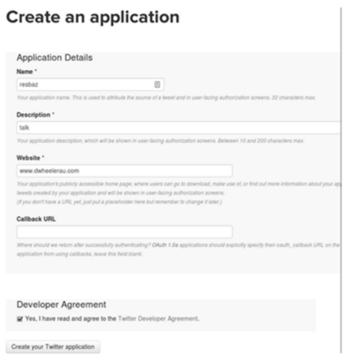
```
 |
 |      # Get a particular friend's timeline
 |      t.statuses.user_timeline(screen_name="billybob")
 |
 |      # to pass in GET/POST parameters, such as `count`
 |      t.statuses.home_timeline(count=5)
 |
 |      # to pass in the GET/POST parameter `id` you need to use `_id`
 |      t.statuses.oembed(_id=1234567890)
 |
 |      # Update your status
 |      t.statuses.update(
 |          status="Using @sixohsix's sweet Python Twitter Tools.")
 |
 |      # Send a direct message
 |      t.direct_messages.new(
 |          user="billybob",
 |          text="I think yer swell!")
 |
```

Getting access to the twitter API



https://apps.twitter.com/

Getting access to the twitter API



Create an application

Application Details

Name *

resbaz

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

talk

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

www.dwheelerau.com

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your ap
tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

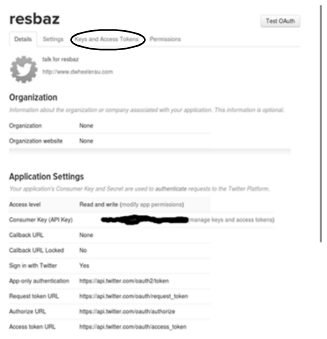
Callback URL

*Where should we return after successfully authenticating? **OAuth 1.0a** applications should explicitly specify their oauth_callback URL on the*
application from using callbacks, leave this field blank.

Developer Agreement

☑ Yes, I have read and agree to the Twitter Developer Agreement.

Getting access to the twitter API



resbaz

talk for resbaz

http://www.dwheelerau.com

Organization

Information about the organization or company associated with your application. This information is optional.

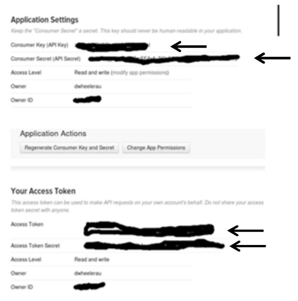
Organization	None
Organization website	None

Application Settings

*Your application's Consumer Key and Secret are used to **authenticate** requests to the Twitter Platform.*

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	[redacted] (manage keys and access tokens)
Callback URL	None
Callback URL Locked	No
Sign in with Twitter	Yes
App-only authentication	https://api.twitter.com/oauth2/token
Request token URL	https://api.twitter.com/oauth/request_token
Authorize URL	https://api.twitter.com/oauth/authorize
Access token URL	https://api.twitter.com/oauth/access_token

Getting access to the twitter API



Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[redacted] ←
Consumer Secret (API Secret)	[redacted] ←
Access Level	Read and write (modify app permissions)
Owner	dwheelerau
Owner ID	[redacted]

Application Actions

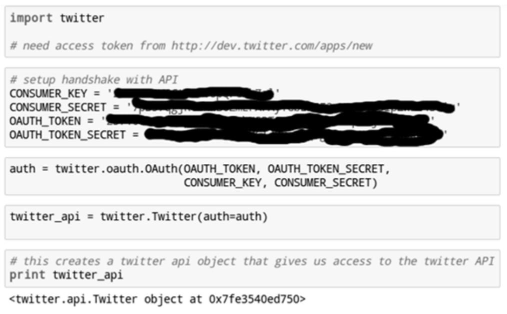
Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[redacted] ←
Access Token Secret	[redacted] ←
Access Level	Read and write
Owner	dwheelerau
Owner ID	[redacted]

You need the consumer Key, Consumer Secret, Access token and Access Token secret

Python twitter



```
import twitter

# need access token from http://dev.twitter.com/apps/new
```

```
# setup handshake with API
CONSUMER_KEY = [redacted]
CONSUMER_SECRET = [redacted]
OAUTH_TOKEN = [redacted]
OAUTH_TOKEN_SECRET = [redacted]
```

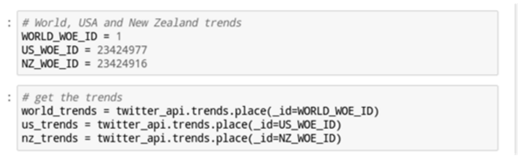
```
auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
                           CONSUMER_KEY, CONSUMER_SECRET)
```

```
twitter_api = twitter.Twitter(auth=auth)
```

```
# this creates a twitter api object that gives us access to the twitter API
print twitter_api
<twitter.api.Twitter object at 0x7fe3540ed750>
```

Twitter trends

- Need GEO codes that we can get from http://developer.yahoo.com/geo/geoplanet

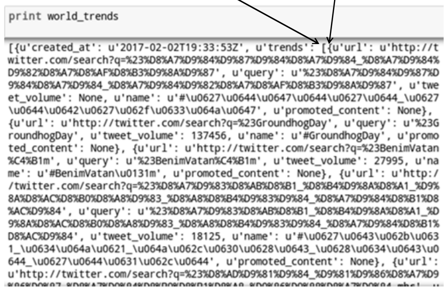


```
# World, USA and New Zealand trends
WORLD_WOE_ID = 1
US_WOE_ID = 23424977
NZ_WOE_ID = 23424916
```

```
# get the trends
world_trends = twitter_api.trends.place(_id=WORLD_WOE_ID)
us_trends = twitter_api.trends.place(_id=US_WOE_ID)
nz_trends = twitter_api.trends.place(_id=NZ_WOE_ID)
```

Twitter trends

Returns data as nested python lists and python dictionaries



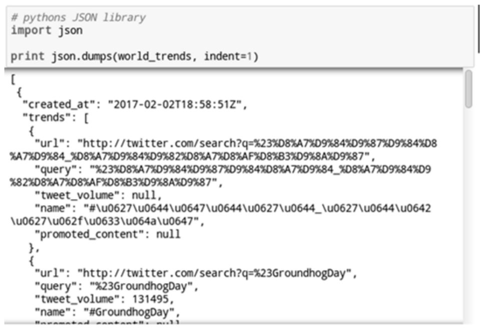
```
print world_trends
```

```
[{u'created_at': u'2017-02-02T19:33:53Z', u'trends': [{u'url': u'http://t
witter.com/search?q=%23%D8%A7%D9%84%D9%87%D9%84%D8%A7%D9%84_%D8%A7%D9%84%
D9%82%D8%A7%D8%AF%D8%B3%D9%8A%D9%87', u'query': u'%23%D8%A7%D9%84%D9%87%D
9%84%D8%A7%D9%84_%D8%A7%D9%84%D9%82%D8%A7%D8%AF%D8%B3%D9%8A%D9%87', u'twe
et_volume': None, u'name': u'#\u0627\u0644\u0647\u0644\u0627\u0644_\u0627
\u0644\u0642\u0627\u062f\u0633\u064a\u0647', u'promoted_content': None},
{u'url': u'http://twitter.com/search?q=%23GroundhogDay', u'query': u'%23G
roundhogDay', u'tweet_volume': 137456, u'name': u'#GroundhogDay', u'promo
ted_content': None}, {u'url': u'http://twitter.com/search?q=%23BenimVatan
%C4%B1m', u'query': u'%23BenimVatan%C4%B1m', u'tweet_volume': 27995, u'na
me': u'#BenimVatan\u0131m', u'promoted_content': None}, {u'url': u'http:/
/twitter.com/search?q=%23%D8%A7%D9%83%D8%AB%D8%B1_%D8%B4%D9%8A%D8%A1_%D9%
8A%D8%AC%D8%B0%D8%A8%D9%83_%D8%A8%D8%B4%D9%83%D9%84_%D8%A7%D9%84%D8%B1%D8
%AC%D9%84', u'query': u'%23%D8%A7%D9%83%D8%AB%D8%B1_%D8%B4%D9%8A%D8%A1_%D
9%8A%D8%AC%D8%B0%D8%A8%D9%83_%D8%A8%D8%B4%D9%83%D9%84_%D8%A7%D9%84%D8%B1%
D8%AC%D9%84', u'tweet_volume': 18125, u'name': u'#\u0627\u0643\u062b\u063
1_\u0634\u064a\u0621_\u064a\u062c\u0630\u0628\u0643_\u0628\u0634\u0643\u0
644_\u0627\u0644\u0631\u062c\u0644', u'promoted_content': None}, {u'url':
u'http://twitter.com/search?q=%23%D8%AD%D9%81%D9%84_%D9%81%D9%86%D8%A7%D9
```

The twitter API

- Limited to ~150 requests in minute
- The python twitter library is just a wrapper for web requests using the REST philosophy
- Returns data in nested lists and dictionaries that are compatible with the JSON format

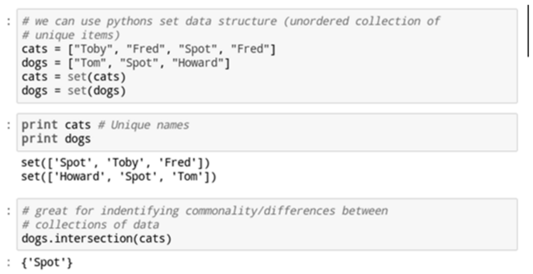
Twitter trends



```
# pythons JSON library
import json

print json.dumps(world_trends, indent=1)
```

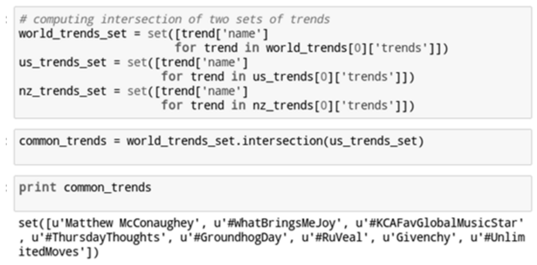
```
[
 {
  "created_at": "2017-02-02T18:58:51Z",
  "trends": [
   {
    "url": "http://twitter.com/search?q=%23%D8%A7%D9%84%D9%87%D9%84%D8
%A7%D9%84_%D8%A7%D9%84%D9%82%D8%A7%D8%AF%D8%B3%D9%8A%D9%87",
    "query": "%23%D8%A7%D9%84%D9%87%D9%84%D8%A7%D9%84_%D8%A7%D9%84%D9
%82%D8%A7%D8%AF%D8%B3%D9%8A%D9%87",
    "tweet_volume": null,
    "name": "#\u0627\u0644\u0647\u0644\u0627\u0644_\u0627\u0644\u0642
\u0627\u062f\u0633\u064a\u0647",
    "promoted_content": null
   },
   {
    "url": "http://twitter.com/search?q=%23GroundhogDay",
    "query": "%23GroundhogDay",
    "tweet_volume": 131495,
    "name": "#GroundhogDay",
```

Geographical twitter trends



```
: # we can use pythons set data structure (unordered collection of
  # unique items)
  cats = ["Toby", "Fred", "Spot", "Fred"]
  dogs = ["Tom", "Spot", "Howard"]
  cats = set(cats)
  dogs = set(dogs)

: print cats # Unique names
  print dogs

  set(['Spot', 'Toby', 'Fred'])
  set(['Howard', 'Spot', 'Tom'])

: # great for indentifying commonality/differences between
  # collections of data
  dogs.intersection(cats)

: {'Spot'}
```

Geographical twitter trends



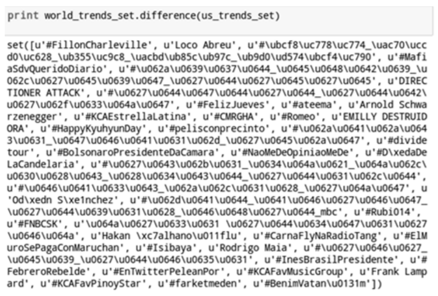
```
# computing intersection of two sets of trends
world_trends_set = set([trend['name']
                        for trend in world_trends[0]['trends']])
us_trends_set = set([trend['name']
                     for trend in us_trends[0]['trends']])
nz_trends_set = set([trend['name']
                     for trend in nz_trends[0]['trends']])
```

```
common_trends = world_trends_set.intersection(us_trends_set)
```

```
print common_trends
```

```
set([u'Matthew McConaughey', u'#WhatBringsMeJoy', u'#KCAFavGlobalMusicStar'
, u'#ThursdayThoughts', u'#GroundhogDay', u'#RuVeal', u'Givenchy', u'#Unlim
itedMoves'])
```

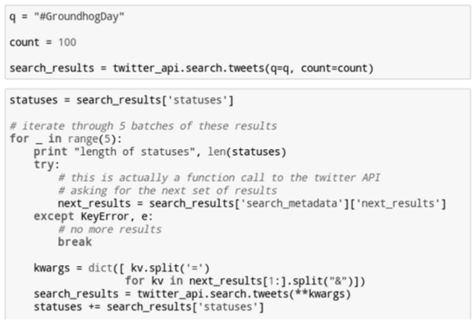
Geographical twitter trends



```
print world_trends_set.difference(us_trends_set)
```

```
set([u'#FillonCharleville', u'Loco Abreu', u'#\ubcf8\uc778\uc774_\uac70\ucc
d0\uc628_\ub355\uc9c8_\uacbd\ub85c\ub97c_\ub9d0\ud574\ubcf4\uc790', u'#Mafi
aSdvQueridoDiario', u'#\u062a\u0639\u0637\u0644_\u0645\u0648\u0642\u0639_\u
062c\u0627\u0645\u0639\u0647_\u0627\u0644\u0627\u0645\u0627\u0645', u'DIREC
TIONER ATTACK', u'#\u0627\u0644\u0647\u0644\u0627\u0644_\u0627\u0644\u0642\
u0627\u062f\u0633\u064a\u0647', u'#FelizJueves', u'#ateema', u'Arnold Schwa
rzenegger', u'#KCAEstrellaLatina', u'#CMRGHA', u'#Romeo', u'EMILLY DESTRUID
ORA', u'#HappyKyuhyunDay', u'#pelisconprecinto', u'#\u062a\u0641\u062a\u064
3\u0631_\u0647\u0646\u0641\u0631\u062d_\u0627\u0645\u062a\u0647', u'#divide
tour', u'#BolsonaroPresidenteDaCamara', u'#NaoMeDeOpiniaoMeDe', u'#D\xedaDe
LaCandelaria', u'#\u0627\u0643\u062b\u0631_\u0634\u064a\u0621_\u064a\u062c\
u0630\u0628\u0643_\u0628\u0634\u0643\u0644_\u0627\u0644\u0631\u062c\u0644',
u'#\u0646\u0641\u0633\u0643_\u062a\u062c\u0631\u0628_\u0627\u064a\u0647', u
'Od\xedn S\xe1nchez', u'#\u062d\u0641\u0644_\u0641\u0646\u0627\u0646\u0647_
\u0627\u0644\u0639\u0631\u0628_\u0646\u0648\u0627\u0644_mbc', u'#Rubi014',
u'#FNBCSK', u'\u064a\u0627\u0633\u0631 \u0627\u0644\u0634\u0647\u0631\u0627
\u0646\u064a', u'Hakan \xc7alhano\u011flu', u'#CarnaFlyNaRadioTang', u'#ElM
uroSePagaConMaruchan', u'#Isibaya', u'Rodrigo Maia', u'#\u0627\u0646\u0627_
\u0645\u0639_\u0627\u0644\u0646\u0635\u0631', u'#InesBrasilPresidente', u'#
FebreroRebelde', u'#EnTwitterPeleanPor', u'#KCAFavMusicGroup', u'Frank Lamp
ard', u'#KCAFavPinoyStar', u'#farketmeden', u'#BenimVatan\u0131m'])
```

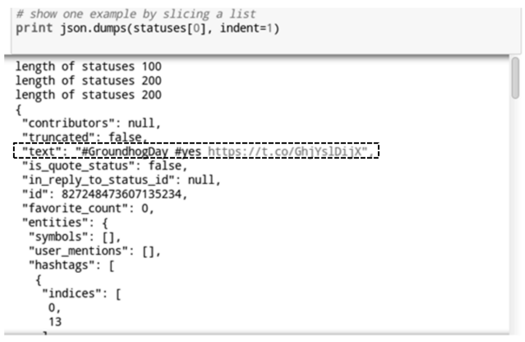
Twitter trends



```
q = "#GroundhogDay"

count = 100

search_results = twitter_api.search.tweets(q=q, count=count)
```

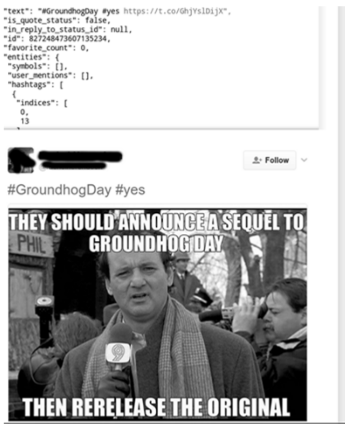
```
statuses = search_results['statuses']

# iterate through 5 batches of these results
for _ in range(5):
    print "length of statuses", len(statuses)
    try:
        # this is actually a function call to the twitter API
        # asking for the next set of results
        next_results = search_results['search_metadata']['next_results']
    except KeyError, e:
        # no more results
        break

    kwargs = dict([ kv.split('=')
                   for kv in next_results[1:].split("&")])
    search_results = twitter_api.search.tweets(**kwargs)
    statuses += search_results['statuses']
```

Twitter trends



```
# show one example by slicing a list
print json.dumps(statuses[0], indent=1)
```

```
length of statuses 100
length of statuses 200
length of statuses 200
{
 "contributors": null,
 "truncated": false,
 "text": "#GroundhogDay #yes https://t.co/GhjYslDijX",
 "is_quote_status": false,
 "in_reply_to_status_id": null,
 "id": 827248473607135234,
 "favorite_count": 0,
 "entities": {
  "symbols": [],
  "user_mentions": [],
  "hashtags": [
   {
    "indices": [
     0,
     13
```



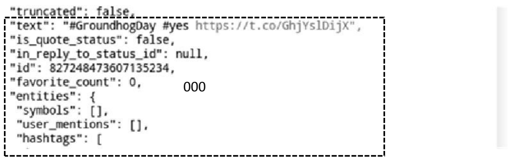
```
"text": "#GroundhogDay #yes https://t.co/GhjYslDijX",
"is_quote_status": false,
"in_reply_to_status_id": null,
"id": 827248473607135234,
"favorite_count": 0,
"entities": {
 "symbols": [],
 "user_mentions": [],
 "hashtags": [
  {
   "indices": [
    0,
    13
```

[redacted]

#GroundhogDay #yes

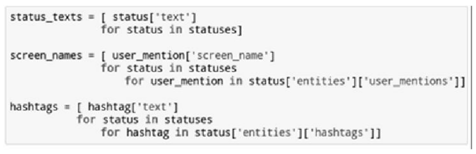
THEY SHOULD ANNOUNCE A SEQUEL TO GROUNDHOG DAY
PHIL
THEN RERELEASE THE ORIGINAL

Twitter trends



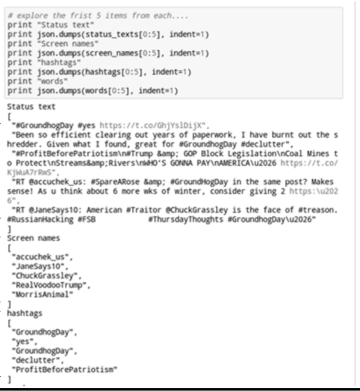
```
"truncated": false,
"text": "#GroundhogDay #yes https://t.co/GhjYslDijX",
"is_quote_status": false,
"in_reply_to_status_id": null,
"id": 827248473607135234,
"favorite_count": 0,
"entities": {
 "symbols": [],
 "user_mentions": [],
 "hashtags": [
```

000



```
status_texts = [ status['text']
                 for status in statuses]

screen_names = [ user_mention['screen_name']
                 for status in statuses
                     for user_mention in status['entities']['user_mentions']]

hashtags = [ hashtag['text']
             for status in statuses
                 for hashtag in status['entities']['hashtags']]
```

Twitter trends



```
# explore the frist 5 items from each....
print "Status text"
print json.dumps(status_texts[0:5], indent=1)
print "Screen names"
print json.dumps(screen_names[0:5], indent=1)
print "hashtags"
print json.dumps(hashtags[0:5], indent=1)
print "words"
print json.dumps(words[0:5], indent=1)
```

Search for “#GroundhogDay”

Status text

```
Status text
[
 "#GroundhogDay #yes https://t.co/GhjYslDijX",
 "Been so efficient clearing out years of paperwork, I have burnt out the s
hredder. Given what I found, great for #GroundhogDay #declutter",
 "#ProfitBeforePatriotism\n#Trump &amp; GOP Block Legislation\nCoal Mines t
o Protect\nStreams&amp;Rivers\nWHO'S GONNA PAY\nAMERICA\u2026 https://t.co/
KjWuA7rRwS",
 "RT @accuchek_us: #SpareARose &amp; #GroundHogDay in the same post? Makes
sense! As u think about 6 more wks of winter, consider giving 2 https:\u202
6",
 "RT @JaneSays10: American #Traitor @ChuckGrassley is the face of #treason.
#RussianHacking #FSB           #ThursdayThoughts #GroundhogDay\u2026"
]
```

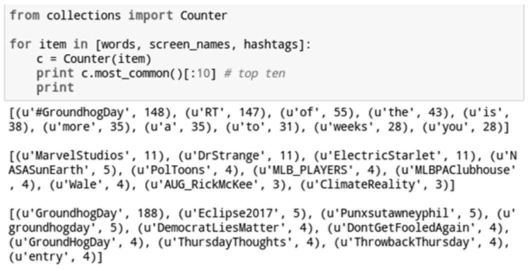
Screen names

```
Screen names
[
 "accuchek_us",
 "JaneSays10",
 "ChuckGrassley",
 "RealVoodooTrump",
 "MorrisAnimal"
]
```

Hashtags

```
hashtags
[
 "GroundhogDay",
 "yes",
 "GroundhogDay",
 "declutter",
 "ProfitBeforePatriotism"
]
```

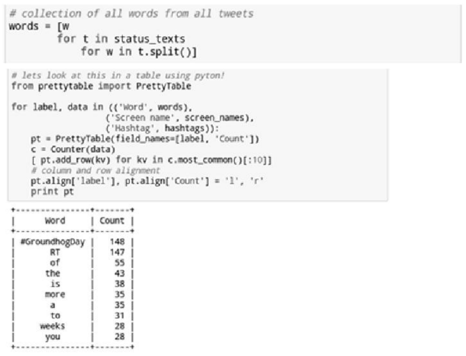
Twitter trends



```
from collections import Counter

for item in [words, screen_names, hashtags]:
    c = Counter(item)
    print c.most_common()[:10] # top ten
    print
```

```
[(u'#GroundhogDay', 148), (u'RT', 147), (u'of', 55), (u'the', 43), (u'is',
38), (u'more', 35), (u'a', 35), (u'to', 31), (u'weeks', 28), (u'you', 28)]

[(u'MarvelStudios', 11), (u'DrStrange', 11), (u'ElectricStarlet', 11), (u'N
ASASunEarth', 5), (u'PolToons', 4), (u'MLB_PLAYERS', 4), (u'MLBPAClubhouse'
, 4), (u'Wale', 4), (u'AUG_RickMcKee', 3), (u'ClimateReality', 3)]

[(u'GroundhogDay', 188), (u'Eclipse2017', 5), (u'Punxsutawneyphil', 5), (u'
groundhogday', 5), (u'DemocratLiesMatter', 4), (u'DontGetFooledAgain', 4),
(u'GroundHogDay', 4), (u'ThursdayThoughts', 4), (u'ThrowbackThursday', 4),
(u'entry', 4)]
```

Twitter trends



```
# collection of all words from all tweets
words = [w
        for t in status_texts
            for w in t.split()]
```

```
# lets look at this in a table using pyton!
from prettytable import PrettyTable

for label, data in (('Word', words),
                    ('Screen name', screen_names),
                    ('Hashtag', hashtags)):
    pt = PrettyTable(field_names=[label, 'Count'])
    c = Counter(data)
    [ pt.add_row(kv) for kv in c.most_common()[:10]]
    # column and row alignment
    pt.align['label'], pt.align['Count'] = 'l', 'r'
    print pt
```

```
+---------------+-------+
|      Word     | Count |
+---------------+-------+
| #GroundhogDay |   148 |
|       RT      |   147 |
|       of      |    55 |
|      the      |    43 |
|       is      |    38 |
|      more     |    35 |
|       a       |    35 |
|       to      |    31 |
|     weeks     |    28 |
|      you      |    28 |
+---------------+-------+
```

The tweets of @realDonaldTrump

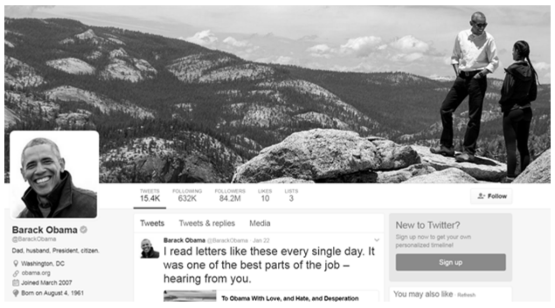
TWEETS 34.4K FOLLOWING 41 FOLLOWERS 24M LIKES 45

Donald J. Trump
@realDonaldTrump
45th President of the United States of America
Washington, DC
Joined March 2009

Tweets Tweets & replies Media

Donald J. Trump @realDonaldTrump · 6h
The failing @nytimes writes total fiction concerning me. They have gotten it wrong for two years, and now are making up stories & sources!
27K 12K 56K

…and @BarackObama



TWEETS 15.4K FOLLOWING 632K FOLLOWERS 84.2M LIKES 10 LISTS 3 Follow

Barack Obama
@BarackObama
Dad, husband, President, citizen.
Washington, DC
obama.org
Joined March 2007
Born on August 4, 1961

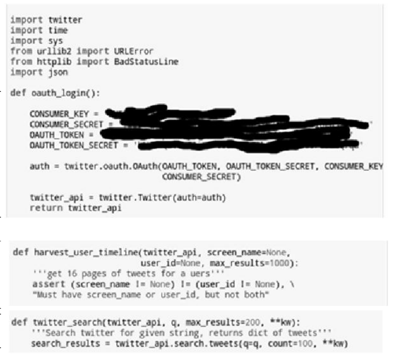
Tweets Tweets & replies Media

Barack Obama @BarackObama · Jan 22
I read letters like these every single day. It was one of the best parts of the job – hearing from you.
To Obama With Love, and Hate, and Desperation

New to Twitter?
Sign up now to get your own personalized timeline!
Sign up

You may also like · Refresh

Writing some helper functions



```python
import twitter
import time
import sys
from urllib2 import URLError
from httplib import BadStatusLine
import json

def oauth_login():

    CONSUMER_KEY = [redacted]
    CONSUMER_SECRET = [redacted]
    OAUTH_TOKEN = [redacted]
    OAUTH_TOKEN_SECRET = '[redacted]'

    auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET, CONSUMER_KEY
                               CONSUMER_SECRET)

    twitter_api = twitter.Twitter(auth=auth)
    return twitter_api
```

Create twitter object

```python
def harvest_user_timeline(twitter_api, screen_name=None,
                          user_id=None, max_results=1000):
    '''get 16 pages of tweets for a uers'''
    assert (screen_name != None) != (user_id != None), \
    "Must have screen_name or user_id, but not both"
```

Collect user tweets

```python
def twitter_search(twitter_api, q, max_results=200, **kw):
    '''Search twitter for given string, returns dict of tweets'''
    search_results = twitter_api.search.tweets(q=q, count=100, **kw)
```

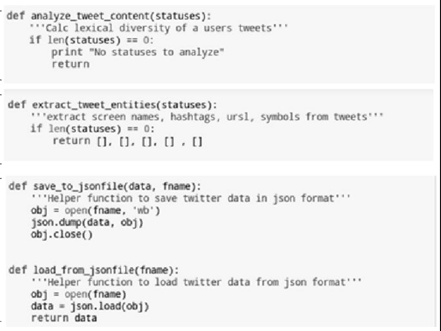
Search for tweets

Writing some helper functions

Lexical diversity

Extract tweet info

Save/load info
In JSON format

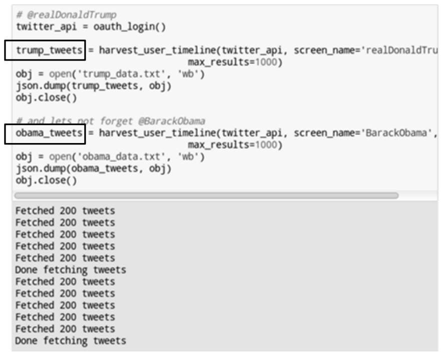


```
def analyze_tweet_content(statuses):
    '''Calc lexical diversity of a users tweets'''
    if len(statuses) == 0:
        print "No statuses to analyze"
        return
```

```
def extract_tweet_entities(statuses):
    '''extract screen names, hashtags, ursl, symbols from tweets'''
    if len(statuses) == 0:
        return [], [], [], [] , []
```

```
def save_to_jsonfile(data, fname):
    '''Helper function to save twitter data in json format'''
    obj = open(fname, 'wb')
    json.dump(data, obj)
    obj.close()


def load_from_jsonfile(fname):
    '''Helper function to load twitter data from json format'''
    obj = open(fname)
    data = json.load(obj)
    return data
```

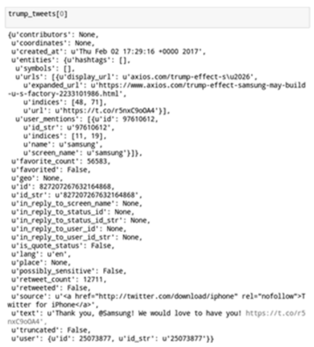
Presidential tweets



```
# @realDonaldTrump
twitter_api = oauth_login()

trump_tweets = harvest_user_timeline(twitter_api, screen_name='realDonaldTru
                                     max_results=1000)
obj = open('trump_data.txt', 'wb')
json.dump(trump_tweets, obj)
obj.close()

# and lets not forget @BarackObama
obama_tweets = harvest_user_timeline(twitter_api, screen_name='BarackObama',
                                     max_results=1000)
obj = open('obama_data.txt', 'wb')
json.dump(obama_tweets, obj)
obj.close()
```

```
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Done fetching tweets
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Done fetching tweets
```

Presidential tweets



```
trump_tweets[0]
```

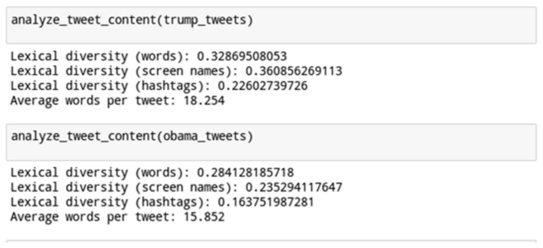
```
{u'contributors': None,
 u'coordinates': None,
 u'created_at': u'Thu Feb 02 17:29:16 +0000 2017',
 u'entities': {u'hashtags': [],
  u'symbols': [],
  u'urls': [{u'display_url': u'axios.com/trump-effect-s\u2026',
    u'expanded_url': u'https://www.axios.com/trump-effect-samsung-may-build
-u-s-factory-2233101986.html',
    u'indices': [48, 71],
    u'url': u'https://t.co/r5nxC9oOA4'}],
  u'user_mentions': [{u'id': 97610612,
    u'id_str': u'97610612',
    u'indices': [11, 19],
    u'name': u'samsung',
    u'screen_name': u'samsung'}]},
 u'favorite_count': 56583,
 u'favorited': False,
 u'geo': None,
 u'id': 827207267632164868,
 u'id_str': u'827207267632164868',
 u'in_reply_to_screen_name': None,
 u'in_reply_to_status_id': None,
 u'in_reply_to_status_id_str': None,
 u'in_reply_to_user_id': None,
 u'in_reply_to_user_id_str': None,
 u'is_quote_status': False,
 u'lang': u'en',
 u'place': None,
 u'possibly_sensitive': False,
 u'retweet_count': 12711,
 u'retweeted': False,
 u'source': u'<a href="http://twitter.com/download/iphone" rel="nofollow">T
witter for iPhone</a>',
 u'text': u'Thank you, @Samsung! We would love to have you! https://t.co/r5
nxC9oOA4',
 u'truncated': False,
 u'user': {u'id': 25073877, u'id_str': u'25073877'}}
```

Lexical diversity

- Number of unique “words” in text divided by total number of words
- Or the “unique information” gained from each tweet
- The function “analyze_tweet_content” calculates this by:
  - Count the number of words
  - Use “set()” to count the number of unique words
- A lexical diversity of 0.25 would equate to around ¼ words are unique within aggregated tweets (about 3 words in an average 14 word tweet)

Lexical diversity

- Trump is a winner (alternative facts?)



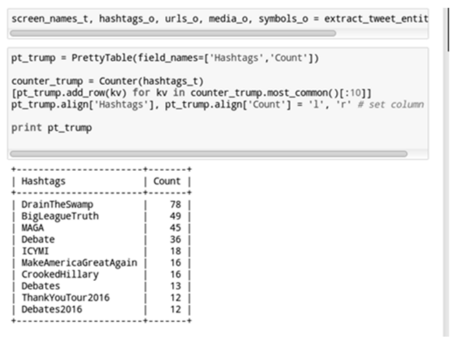
```
analyze_tweet_content(trump_tweets)
```

```
Lexical diversity (words): 0.32869508053
Lexical diversity (screen names): 0.360856269113
Lexical diversity (hashtags): 0.22602739726
Average words per tweet: 18.254
```

```
analyze_tweet_content(obama_tweets)
```

```
Lexical diversity (words): 0.284128185718
Lexical diversity (screen names): 0.235294117647
Lexical diversity (hashtags): 0.163751987281
Average words per tweet: 15.852
```

Presidential tweets

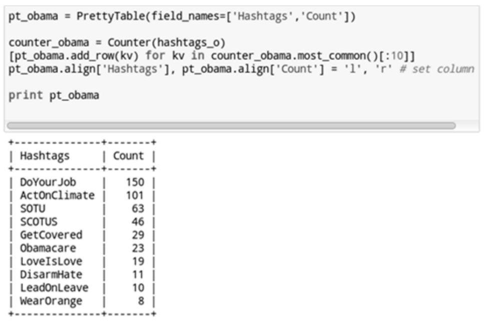


```
screen_names_t, hashtags_o, urls_o, media_o, symbols_o = extract_tweet_entit
```

```
pt_trump = PrettyTable(field_names=['Hashtags','Count'])

counter_trump = Counter(hashtags_t)
[pt_trump.add_row(kv) for kv in counter_trump.most_common()[:10]]
pt_trump.align['Hashtags'], pt_trump.align['Count'] = 'l', 'r' # set column

print pt_trump
```

```
+-----------------------+-------+
| Hashtags              | Count |
+-----------------------+-------+
| DrainTheSwamp         |    78 |
| BigLeagueTruth        |    49 |
| MAGA                  |    45 |
| Debate                |    36 |
| ICYMI                 |    18 |
| MakeAmericaGreatAgain |    16 |
| CrookedHillary        |    16 |
| Debates               |    13 |
| ThankYouTour2016      |    12 |
| Debates2016           |    12 |
+-----------------------+-------+
```

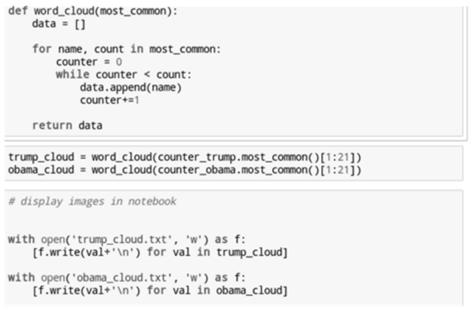
Presidential tweets



```
pt_obama = PrettyTable(field_names=['Hashtags','Count'])

counter_obama = Counter(hashtags_o)
[pt_obama.add_row(kv) for kv in counter_obama.most_common()[:10]]
pt_obama.align['Hashtags'], pt_obama.align['Count'] = 'l', 'r' # set column

print pt_obama
```

```
+--------------+-------+
| Hashtags     | Count |
+--------------+-------+
| DoYourJob    |   150 |
| ActOnClimate |   101 |
| SOTU         |    63 |
| SCOTUS       |    46 |
| GetCovered   |    29 |
| Obamacare    |    23 |
| LoveIsLove   |    19 |
| DisarmHate   |    11 |
| LeadOnLeave  |    10 |
| WearOrange   |     8 |
+--------------+-------+
```

Presidential tweets

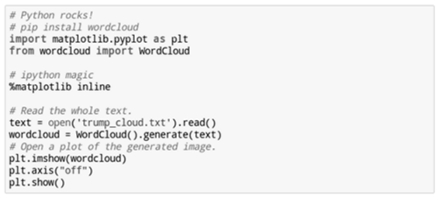


```
def word_cloud(most_common):
    data = []

    for name, count in most_common:
        counter = 0
        while counter < count:
            data.append(name)
            counter+=1

    return data
```

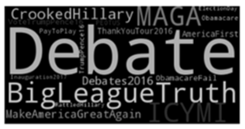
```
trump_cloud = word_cloud(counter_trump.most_common()[1:21])
obama_cloud = word_cloud(counter_obama.most_common()[1:21])
```

```
# display images in notebook


with open('trump_cloud.txt', 'w') as f:
    [f.write(val+'\n') for val in trump_cloud]

with open('obama_cloud.txt', 'w') as f:
    [f.write(val+'\n') for val in obama_cloud]
```

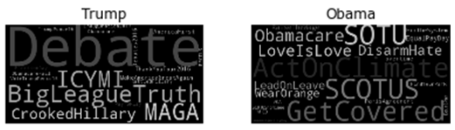
Presidential tweets



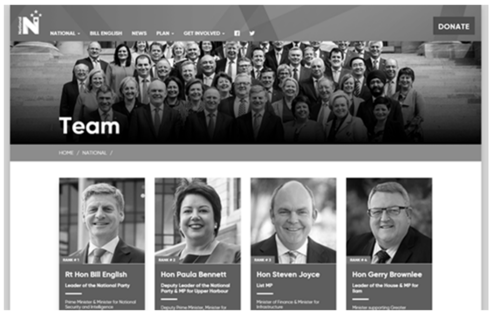
```
# Python rocks!
# pip install wordcloud
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# ipython magic
%matplotlib inline

# Read the whole text.
text = open('trump_cloud.txt').read()
wordcloud = WordCloud().generate(text)
# Open a plot of the generated image.
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```



Presidential tweets

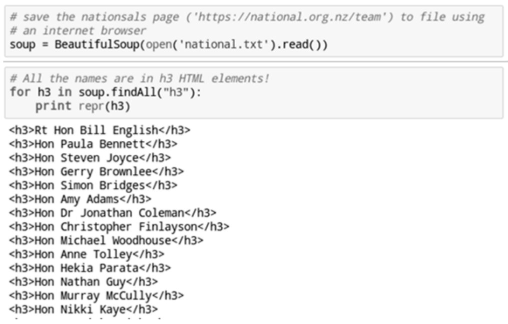


What about NZ?



https://www.national.org.nz/team

An aside: Beautifulsoup

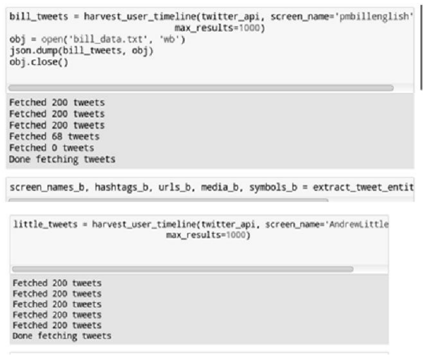


```
# save the nationsals page ('https://national.org.nz/team') to file using
# an internet browser
soup = BeautifulSoup(open('national.txt').read())
```

```
# All the names are in h3 HTML elements!
for h3 in soup.findAll("h3"):
    print repr(h3)
```

```
<h3>Rt Hon Bill English</h3>
<h3>Hon Paula Bennett</h3>
<h3>Hon Steven Joyce</h3>
<h3>Hon Gerry Brownlee</h3>
<h3>Hon Simon Bridges</h3>
<h3>Hon Amy Adams</h3>
<h3>Hon Dr Jonathan Coleman</h3>
<h3>Hon Christopher Finlayson</h3>
<h3>Hon Michael Woodhouse</h3>
<h3>Hon Anne Tolley</h3>
<h3>Hon Hekia Parata</h3>
<h3>Hon Nathan Guy</h3>
<h3>Hon Murray McCully</h3>
<h3>Hon Nikki Kaye</h3>
```

What about NZ?



```
bill_tweets = harvest_user_timeline(twitter_api, screen_name='pmbillenglish'
                                    max_results=1000)
obj = open('bill_data.txt', 'wb')
json.dump(bill_tweets, obj)
obj.close()
```

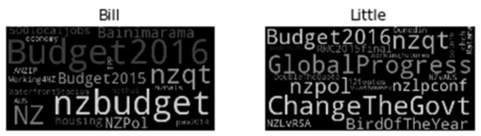
```
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Fetched 68 tweets
Fetched 0 tweets
Done fetching tweets
```

```
screen_names_b, hashtags_b, urls_b, media_b, symbols_b = extract_tweet_entit
```

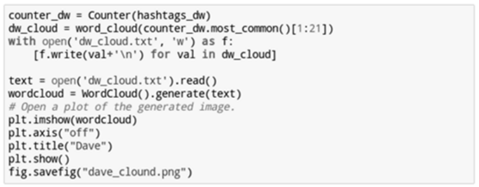
```
little_tweets = harvest_user_timeline(twitter_api, screen_name='AndrewLittle
                                      max_results=1000)
```

```
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Fetched 200 tweets
Done fetching tweets
```

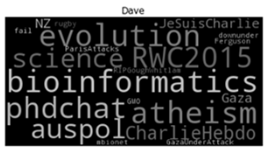
What about NZ?



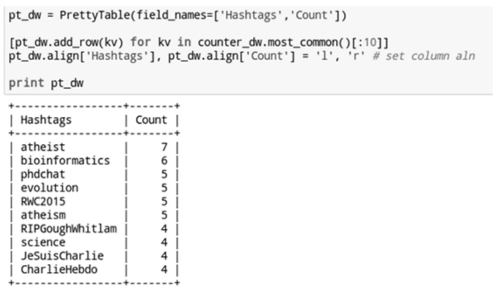
Leave the pollies alone @dwheelerau



```
counter_dw = Counter(hashtags_dw)
dw_cloud = word_cloud(counter_dw.most_common()[1:21])
with open('dw_cloud.txt', 'w') as f:
    [f.write(val+'\n') for val in dw_cloud]

text = open('dw_cloud.txt').read()
wordcloud = WordCloud().generate(text)
# Open a plot of the generated image.
plt.imshow(wordcloud)
plt.axis("off")
plt.title("Dave")
plt.show()
fig.savefig("dave_clound.png")
```



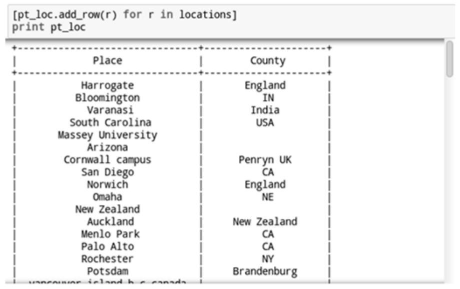
Leave the pollies alone @dwheelerau



```
pt_dw = PrettyTable(field_names=['Hashtags','Count'])

[pt_dw.add_row(kv) for kv in counter_dw.most_common()[:10]]
pt_dw.align['Hashtags'], pt_dw.align['Count'] = 'l', 'r' # set column aln

print pt_dw
```

```
+----------------+-------+
| Hashtags       | Count |
+----------------+-------+
| atheist        |     7 |
| bioinformatics |     6 |
| phdchat        |     5 |
| evolution      |     5 |
| RWC2015        |     5 |
| atheism        |     5 |
| RIPGoughWhitlam|     4 |
| science        |     4 |
| JeSuisCharlie  |     4 |
| CharlieHebdo   |     4 |
+----------------+-------+
```

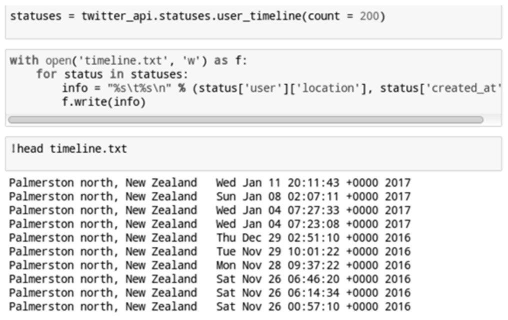
Where do my followers live?



```
[pt_loc.add_row(r) for r in locations]
print pt_loc
```

```
+------------------------------+--------------------+
|            Place             |       County       |
+------------------------------+--------------------+
|          Harrogate           |      England       |
|         Bloomington          |         IN         |
|           Varanasi           |       India        |
|        South Carolina        |        USA         |
|       Massey University      |                    |
|           Arizona            |                    |
|        Cornwall campus       |     Penryn UK      |
|          San Diego           |         CA         |
|           Norwich            |      England       |
|            Omaha             |         NE         |
|         New Zealand          |                    |
|           Auckland           |    New Zealand     |
|          Menlo Park          |         CA         |
|          Palo Alto           |         CA         |
|          Rochester           |         NY         |
|           Potsdam            |    Brandenburg     |
|  vancouver island b.c canada |
```

How often do I tweet?



```
statuses = twitter_api.statuses.user_timeline(count = 200)
```

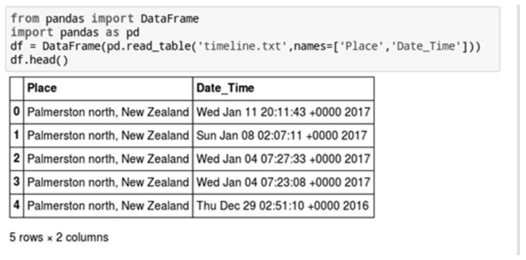
```
with open('timeline.txt', 'w') as f:
    for status in statuses:
        info = "%s\t%s\n" % (status['user']['location'], status['created_at'
        f.write(info)
```

```
!head timeline.txt
```

```
Palmerston north, New Zealand   Wed Jan 11 20:11:43 +0000 2017
Palmerston north, New Zealand   Sun Jan 08 02:07:11 +0000 2017
Palmerston north, New Zealand   Wed Jan 04 07:27:33 +0000 2017
Palmerston north, New Zealand   Wed Jan 04 07:23:08 +0000 2017
Palmerston north, New Zealand   Thu Dec 29 02:51:10 +0000 2016
Palmerston north, New Zealand   Tue Nov 29 10:01:22 +0000 2016
Palmerston north, New Zealand   Mon Nov 28 09:37:22 +0000 2016
Palmerston north, New Zealand   Sat Nov 26 06:46:20 +0000 2016
Palmerston north, New Zealand   Sat Nov 26 06:14:34 +0000 2016
Palmerston north, New Zealand   Sat Nov 26 00:57:10 +0000 2016
```

How often do I tweet?

- Processing tabular data with pandas

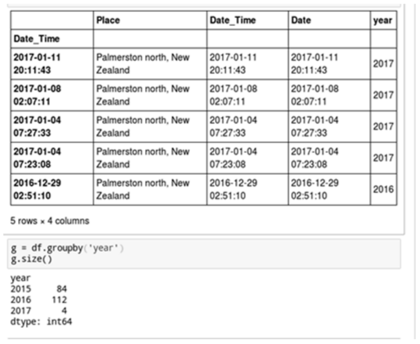


```
from pandas import DataFrame
import pandas as pd
df = DataFrame(pd.read_table('timeline.txt',names=['Place','Date_Time']))
df.head()
```

	Place	Date_Time
0	Palmerston north, New Zealand	Wed Jan 11 20:11:43 +0000 2017
1	Palmerston north, New Zealand	Sun Jan 08 02:07:11 +0000 2017
2	Palmerston north, New Zealand	Wed Jan 04 07:27:33 +0000 2017
3	Palmerston north, New Zealand	Wed Jan 04 07:23:08 +0000 2017
4	Palmerston north, New Zealand	Thu Dec 29 02:51:10 +0000 2016

5 rows × 2 columns

How often do I tweet?



	Place	Date_Time	Date	year
Date_Time				
2017-01-11 20:11:43	Palmerston north, New Zealand	2017-01-11 20:11:43	2017-01-11 20:11:43	2017
2017-01-08 02:07:11	Palmerston north, New Zealand	2017-01-08 02:07:11	2017-01-08 02:07:11	2017
2017-01-04 07:27:33	Palmerston north, New Zealand	2017-01-04 07:27:33	2017-01-04 07:27:33	2017
2017-01-04 07:23:08	Palmerston north, New Zealand	2017-01-04 07:23:08	2017-01-04 07:23:08	2017
2016-12-29 02:51:10	Palmerston north, New Zealand	2016-12-29 02:51:10	2016-12-29 02:51:10	2016

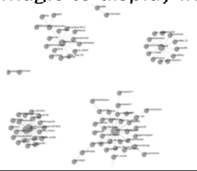
5 rows × 4 columns

```
g = df.groupby('year')
g.size()
```

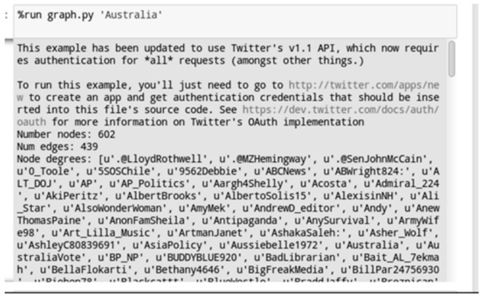
```
year
2015      84
2016     112
2017       4
dtype: int64
```

This is only the start

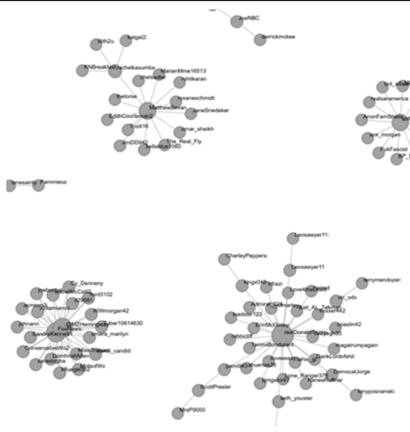
- Finding patterns in tweets and re-tweets
  - Use “Australia” as a search term (back to Trump)
- Nodes represent usernames and edges represent a re-tweet relationship
- Use HTML5 magic to display interactively



This is only the start



```
: %run graph.py 'Australia'

This example has been updated to use Twitter's v1.1 API, which now requir
es authentication for *all* requests (amongst other things.)

To run this example, you'll just need to go to http://twitter.com/apps/ne
w to create an app and get authentication credentials that should be inse
rted into this file's source code. See https://dev.twitter.com/docs/auth/
oauth for more information on Twitter's OAuth implementation
Number nodes: 602
Num edges: 439
Node degrees: [u'.@LloydRothwell', u'.@MZHemingway', u'.@SenJohnMcCain',
u'0_Toole', u'5SOSChile', u'9562Debbie', u'ABCNews', u'ABWright824:', u'A
LT_DOJ', u'AP', u'AP_Politics', u'Aargh4Shelly', u'Acosta', u'Admiral_224
', u'AkiPeritz', u'AlbertBrooks', u'AlbertoSolis15', u'AlexisinNH', u'Ali
_Star', u'AlsoWonderWoman', u'AmyMek', u'AndrewD_editor', u'Andy', u'Anew
ThomasPaine', u'AnonFamSheila', u'Antipaganda', u'AnySurvival', u'ArmyWif
e98', u'Art_Lilla_Music', u'ArtmanJanet', u'AshakaSaleh:', u'Asher_Wolf',
u'AshleyC80839691', u'AsiaPolicy', u'Aussiebelle1972', u'Australia', u'Au
straliaVote', u'BP_NP', u'BUDDYBLUE920', u'BadLibrarian', u'Bait_AL_7ekma
h', u'BellaFlokarti', u'Bethany4646', u'BigFreakMedia', u'BillPar24756930
```



Thank you for your time

- Thanks to the open source community that make this all possible!
- Clone this talk @github (https://github.com/dwheelerau/ResBazPub.git)
- Follow me on twitter (@dwheelerau)
- Bioinformatics and data science blog (www.dwheelerau.com)
- Rm D5.31 IFS, Massey University