Brief Python

Python Course for Programmers

Learn Python Language for Data Science

CHAPTER 9: DATA VISUALIZATION WITH PYLAB

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Data Sets
- Using API (Geopy)
- Data Analysis
- Web Crawler
- Google Architecture

Data Set

LECTURE 1


Agriculture
Climate
Consumer
Ecosystems
Education
Energy
Finance
Health
Local Government
Manufacturing
Maritime
Ocean
Public Safety
Science & Research

Data.gov

UC Irvine Machine Learning Repository

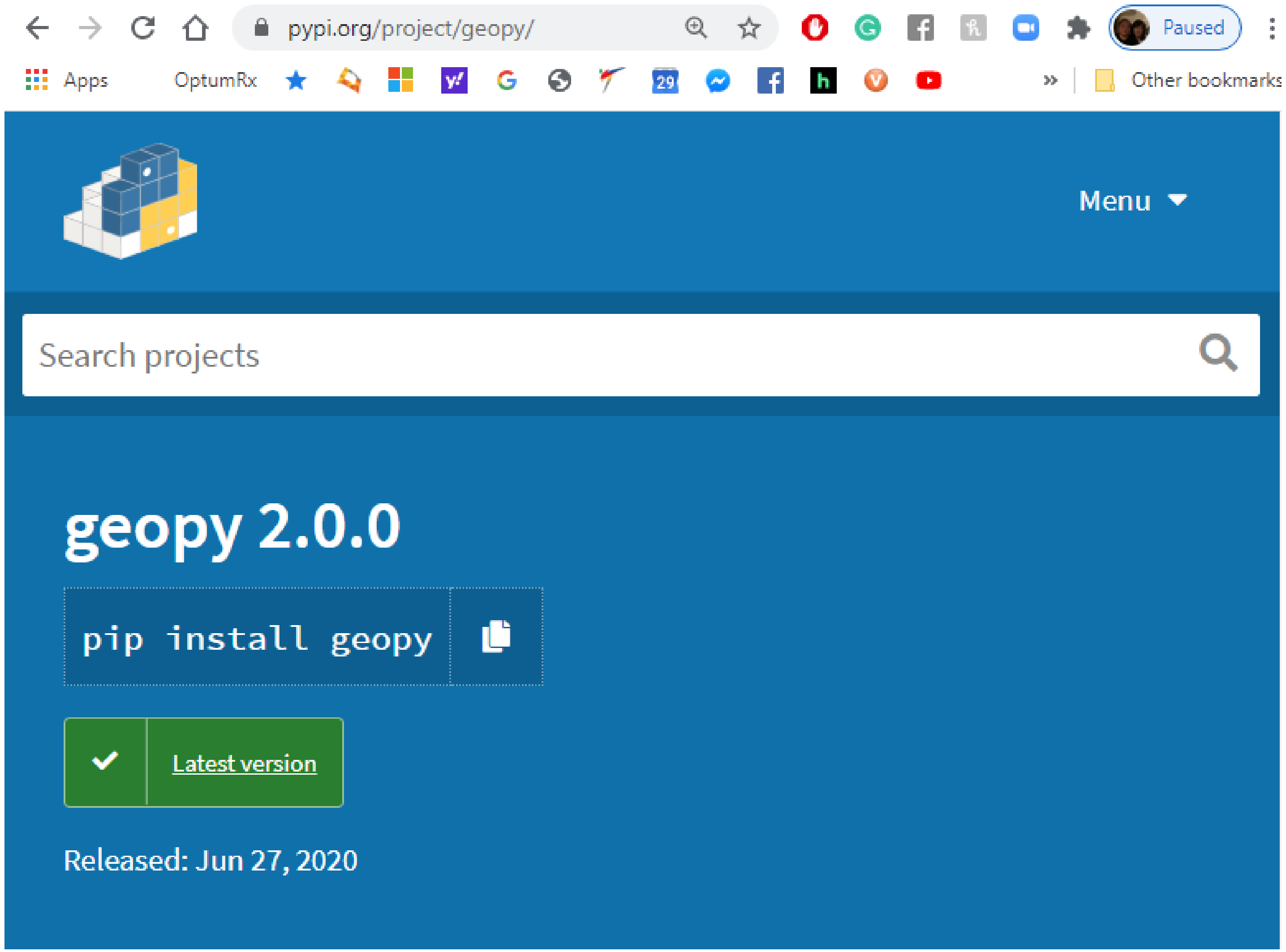
UCI

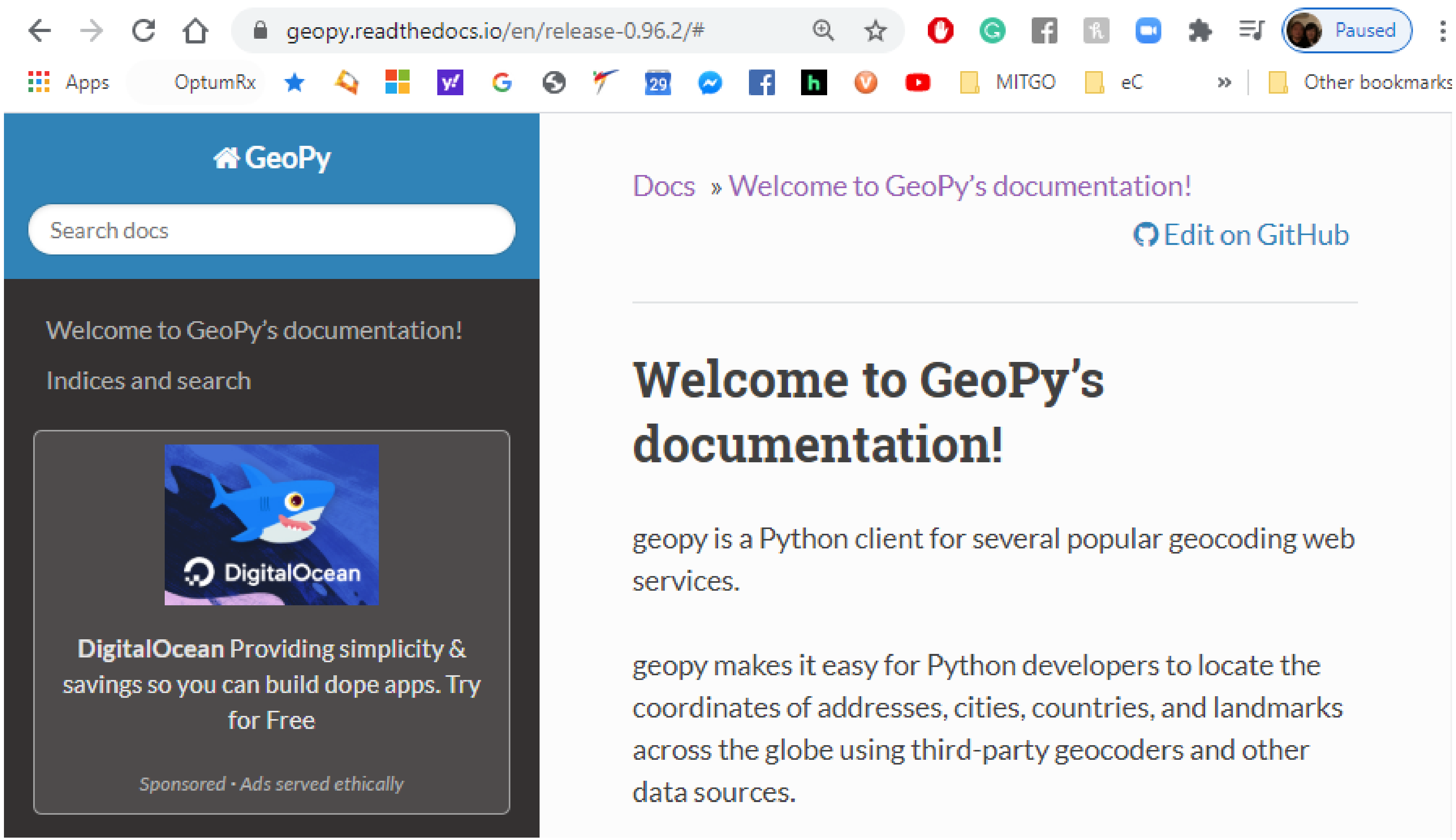
Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Buckingham Palace

LECTURE 2


pypi.org/project/geopy/
Paused
Apps
OptumRx
Other bookmarks
Menu
Search projects
geopy 2.0.0
pip install geopy
Latest version
Released: Jun 27, 2020


geopy.readthedocs.io/en/release-0.96.2/#
Paused
Apps
OptumRx
MITGO
eC
Other bookmarks
GeoPy
Search docs
Welcome to GeoPy's documentation!
Indices and search
DigitalOcean
DigitalOcean Providing simplicity & savings so you can build dope apps. Try for Free
Sponsored · Ads served ethically
Docs » Welcome to GeoPy's documentation!
Edit on GitHub
Welcome to GeoPy's documentation!
geopy is a Python client for several popular geocoding web services.
geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources.

Numpy cos/sin

Home Data

Home <-0.4198, -0.6724, 0.6096>

```
from geopy import distance
# Data Model
# 38781 Tyson Lane, Fremont, CA 94536
tyson_home = (37.562001,-121.9768045)         # (lat, long) of my home
tlat  = np.deg2rad(tyson_home[0])
tlong = np.deg2rad(tyson_home[1])

xt = np.cos(tlat) * np.cos(tlong)
yt = np.cos(tlat) * np.sin(tlong)
zt = np.sin(tlat)

print("Home  <%7.4f, %7.4f, %7.4f>" % (xt, yt, zt))
```

Numpy cos/sin

Buckingham Data

Palace< 0.6225, -0.0016, 0.7826>

```
# Westminster, London SW1A 1AA, UK
buckingham_palace = (51.5013673,-0.1440787) # (lat, long) of my home
blat  = np.deg2rad(buckingham_palace[0])
blong = np.deg2rad(buckingham_palace[1])

xb = np.cos(blat) * np.cos(blong)
yb = np.cos(blat) * np.sin(blong)
zb = np.sin(blat)
print("Palace<%7.4f, %7.4f, %7.4f>" % (xb, yb, zb))
```

Python Code (Linear Algebra Calculation for Distance)

```
v = np.array([xt, yt, zt])
w = np.array([xb, yb, zb])
zero = np.array([0, 0, 0])
print("v=", v)
print("w=", w)
vw = np.inner(v, w)
print("v . w=", vw)
v_abs = np.linalg.norm(v-zero)
w_abs = np.linalg.norm(w-zero)
print("|v|=", v_abs)     # both are 1
print("|w|=", w_abs)
print("(vw)/(v_abs*w_abs)=%8.4f" % ((vw)/(v_abs*w_abs)))
print()
theta = np.arccos((vw)/(v_abs*w_abs))
print("theta(deg)=%8.4f\u00B0" % np.rad2deg(theta))
print("theta(rad)=%8.4f (rad)" % theta)
print()
print("Distance(Home, Buckingham Palace)=%10.4f Km" % (6371*theta))
print("Distance(Home, Buckingham Palace)=%10.4f miles" % (6371*theta*0.621371))
```

```
v= [-0.4197917  -0.67241274  0.60961958]
w= [ 0.62249399 -0.00156536  0.78262301]
v . w= 0.2168370628092262
|v|= 1.0
|w|= 1.0
(vw)/(v_abs*w_abs)=  0.2168
theta(deg)= 77.4767°
theta(rad)=  1.3522 (rad)
Distance(Home, Buckingham Palace)= 8615.0131 Km
Distance(Home, Buckingham Palace)= 5353.1193 miles
```

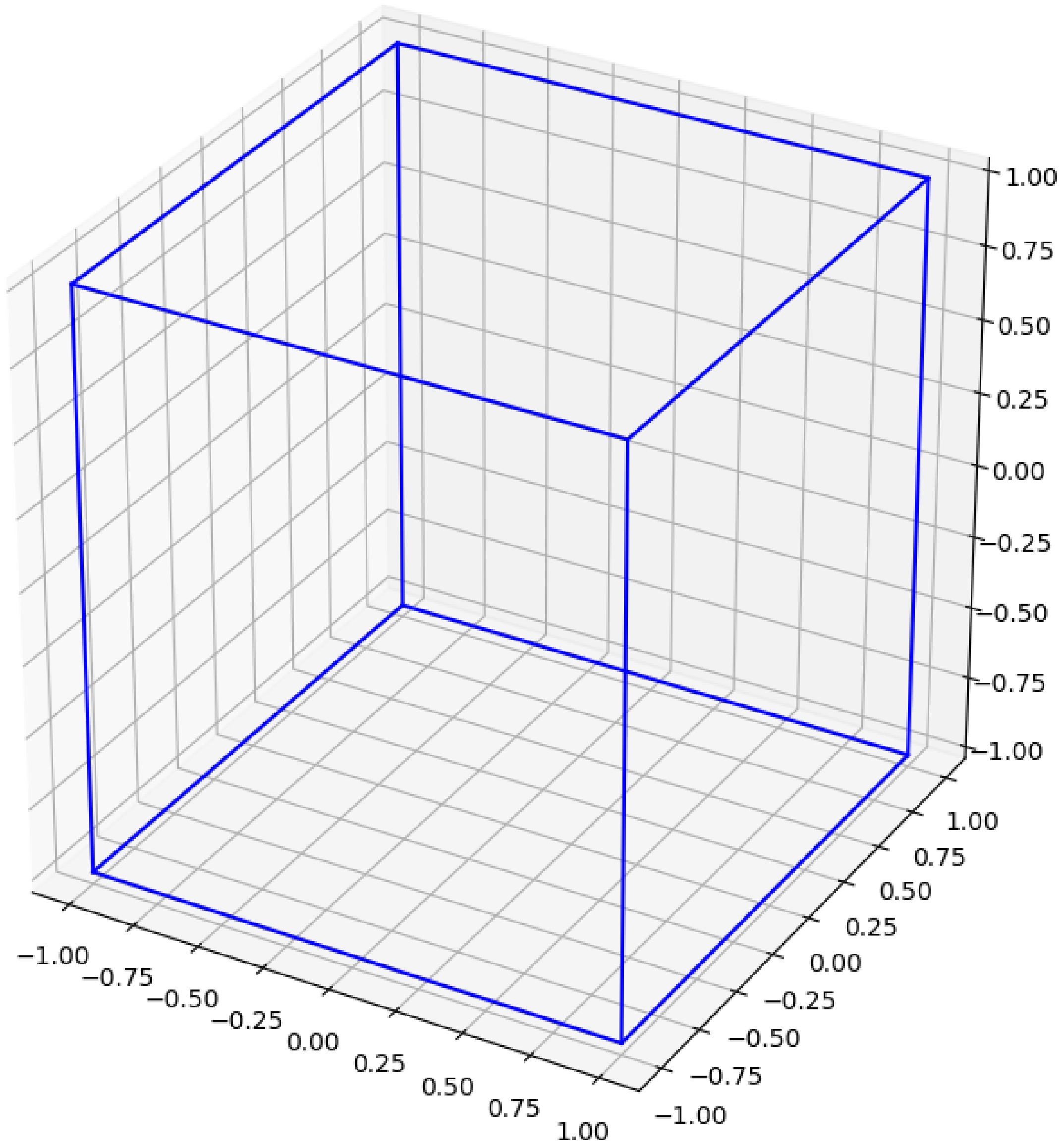
Distance Calculation by Geopy

```
# distance calculation by geopy
dis_miles = distance.distance(tyson_home, buckingham_palace).miles
dis_km    = distance.distance(tyson_home, buckingham_palace).km
print("Distance(Home, Buckingham Palace)=%10.4f Km (geopy)" % dis_km)
print("Distance(Home, Buckingham Palace)=%10.4f miles (geopy)" % dis_miles)
```

```
Distance(Home, Buckingham Palace)= 8637.2216 Km (geopy)
Distance(Home, Buckingham Palace)= 5366.9207 miles (geopy)
```

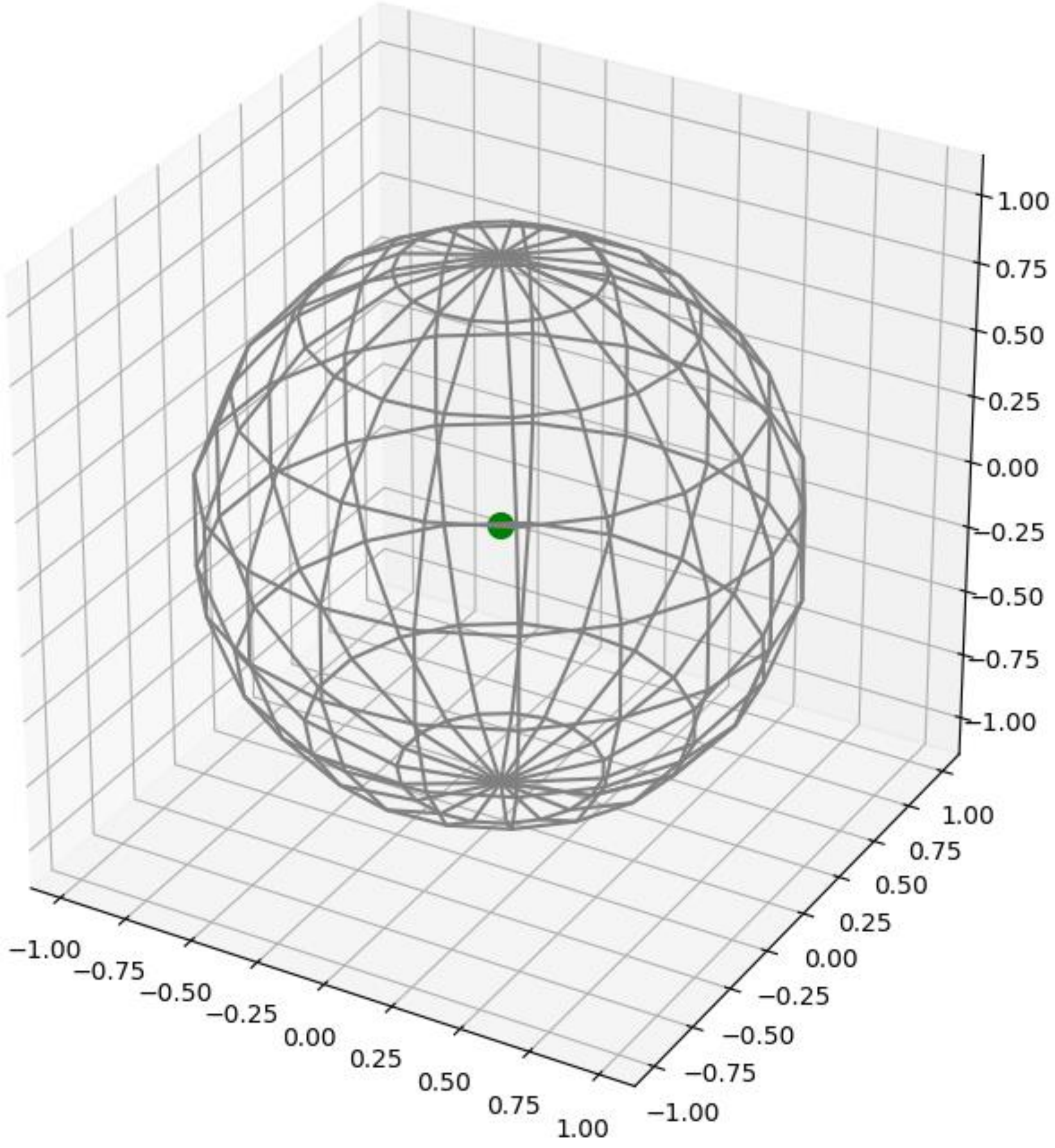
DrawCube.py

```python
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from itertools import product, combinations

fig = plt.figure("Earth", figsize=(12, 9))
ax = fig.gca(projection='3d')
ax.set_aspect("equal")

# draw cube
r = [-1, 1]
for s, e in combinations(np.array(list(product(r, r, r))), 2):
    if np.sum(np.abs(s-e)) == r[1]-r[0]:
        ax.plot3D(*zip(s, e), color="b")
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

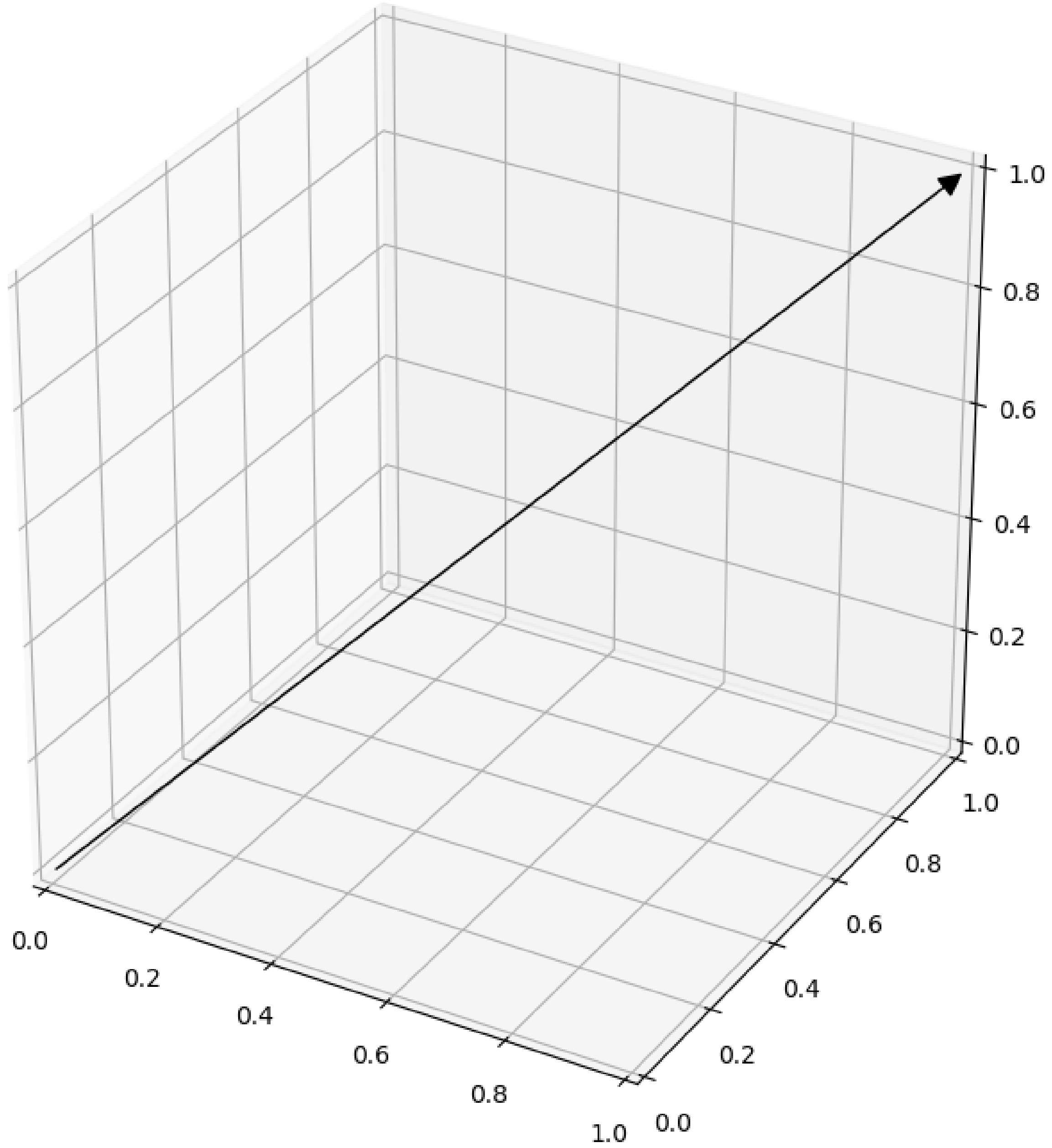
DrawBall.py

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from itertools import product, combinations

fig = plt.figure("Earth", figsize=(12, 9))
ax = fig.gca(projection='3d')
ax.set_aspect("equal")

# draw sphere
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
ax.plot_wireframe(x, y, z, color="gray")
# draw a point
ax.scatter([0], [0], [0], color="g", s=100)
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

DrawVector.py

```python
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from itertools import product, combinations

fig = plt.figure("Earth", figsize=(12, 9))
ax = fig.gca(projection='3d')
ax.set_aspect("equal")

# draw a vector
from matplotlib.patches import FancyArrowPatch
from mpl_toolkits.mplot3d import proj3d

class Arrow3D(FancyArrowPatch):
    def __init__(self, xs, ys, zs, *args, **kwargs):
        FancyArrowPatch.__init__(self, (0, 0), (0, 0), *args, **kwargs)
        self._verts3d = xs, ys, zs
    def draw(self, renderer):
        xs3d, ys3d, zs3d = self._verts3d
        xs, ys, zs = proj3d.proj_transform(xs3d, ys3d, zs3d, renderer.M)
        self.set_positions((xs[0], ys[0]), (xs[1], ys[1]))
        FancyArrowPatch.draw(self, renderer)

a = Arrow3D([0, 1], [0, 1], [0, 1], mutation_scale=20,
            lw=1, arrowstyle="-|>", color="k")
ax.add_artist(a)
plt.show()
```

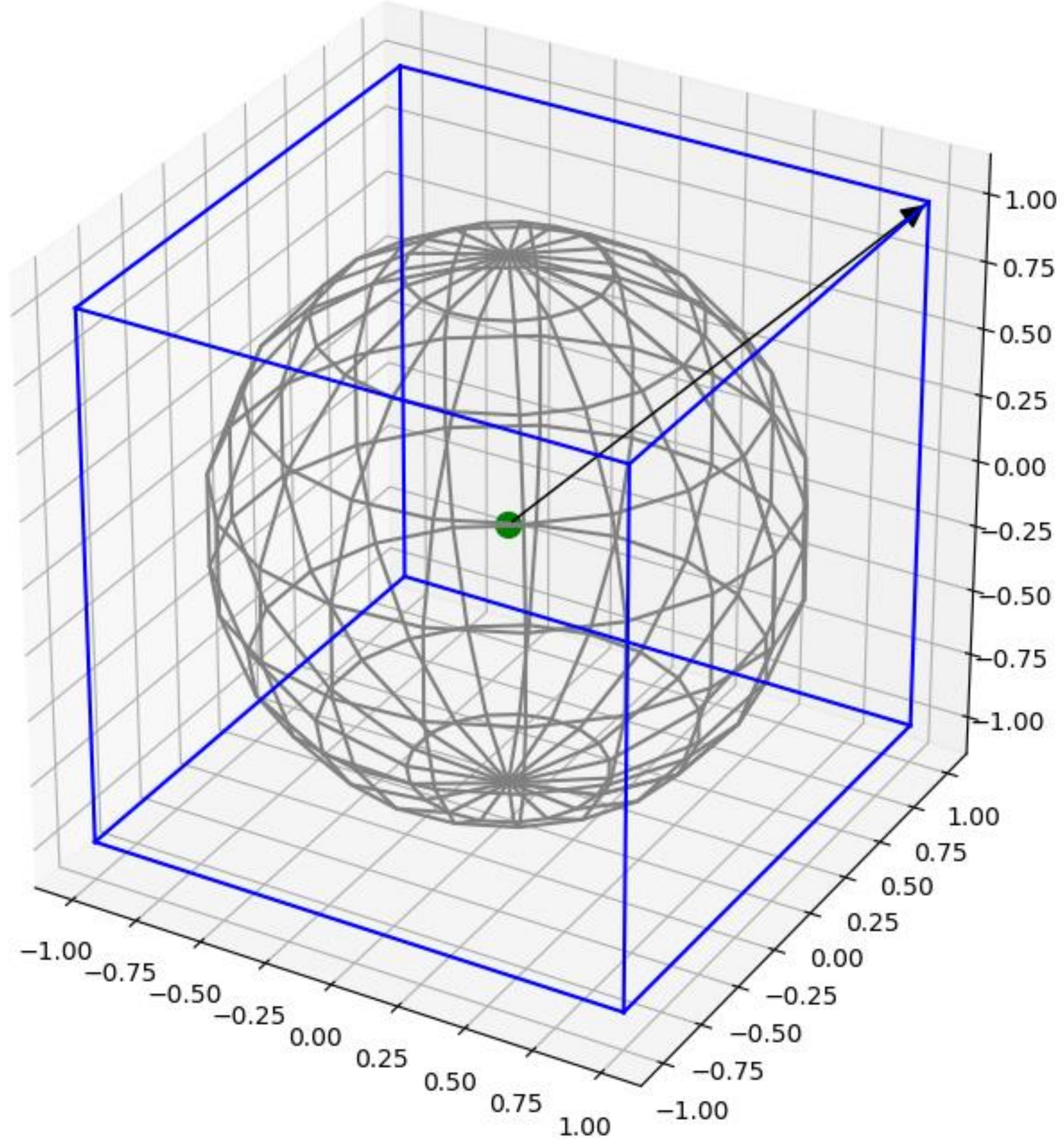

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

ThreeDBall.py

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from itertools import product, combinations

fig = plt.figure("Earth", figsize=(12, 9))
ax = fig.gca(projection='3d')
ax.set_aspect("equal")

# draw cube
r = [-1, 1]
for s, e in combinations(np.array(list(product(r, r, r))), 2):
    if np.sum(np.abs(s-e)) == r[1]-r[0]:
        ax.plot3D(*zip(s, e), color="b")
# draw sphere
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
ax.plot_wireframe(x, y, z, color="gray")
# draw a point
ax.scatter([0], [0], [0], color="g", s=100)
```

ThreeDBall.py

```
# draw a vector
from matplotlib.patches import FancyArrowPatch
from mpl_toolkits.mplot3d import proj3d

class Arrow3D(FancyArrowPatch):
    def __init__(self, xs, ys, zs, *args, **kwargs):
        FancyArrowPatch.__init__(self, (0, 0), (0, 0), *args, **kwargs)
        self._verts3d = xs, ys, zs
    def draw(self, renderer):
        xs3d, ys3d, zs3d = self._verts3d
        xs, ys, zs = proj3d.proj_transform(xs3d, ys3d, zs3d, renderer.M)
        self.set_positions((xs[0], ys[0]), (xs[1], ys[1]))
        FancyArrowPatch.draw(self, renderer)

a = Arrow3D([0, 1], [0, 1], [0, 1], mutation_scale=20,
            lw=1, arrowstyle="-|>", color="k")
ax.add_artist(a)
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75 −1.00

Draw Points

Demo Program: BuckinghamPalace.py

```
# draw a point
ax.scatter([0], [0], [0], color="g", s=100)
ax.scatter([1], [0], [0], color="k", s=20)
ax.scatter([-1], [0], [0], color="k", s=20)
ax.scatter([0], [1], [0], color="k", s=20)
ax.scatter([0], [-1], [0], color="k", s=20)
ax.scatter([0], [0], [1], color="k", s=20)
ax.scatter([0], [0], [-1], color="k", s=20)
ax.scatter([xt], [yt], [zt], color="r", s=100)
ax.scatter([xb], [yb], [zb], color="b", s=100)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
```

Draw Vectors

Demo Program: BuckinghamPalace.py

```
a = Arrow3D([0, 1], [0, 0], [0, 0], mutation_scale=20,
            lw=1, arrowstyle="-", color="k")
ax.add_artist(a)
c = Arrow3D([0, -1], [0, 0], [0, 0], mutation_scale=20,
            lw=1, arrowstyle="-", color="k")
ax.add_artist(c)
d = Arrow3D([0, 0], [0, 1], [0, 0], mutation_scale=20,
            lw=1, arrowstyle="-", color="k")
ax.add_artist(d)
e = Arrow3D([0, 0], [0, -1], [0, 0], mutation_scale=20,
            lw=1, arrowstyle="-", color="k")
ax.add_artist(e)
f = Arrow3D([0, 0], [0, 0], [0, -1], mutation_scale=20,
            lw=1, arrowstyle="-", color="k")
ax.add_artist(f)
```

Draw Vectors

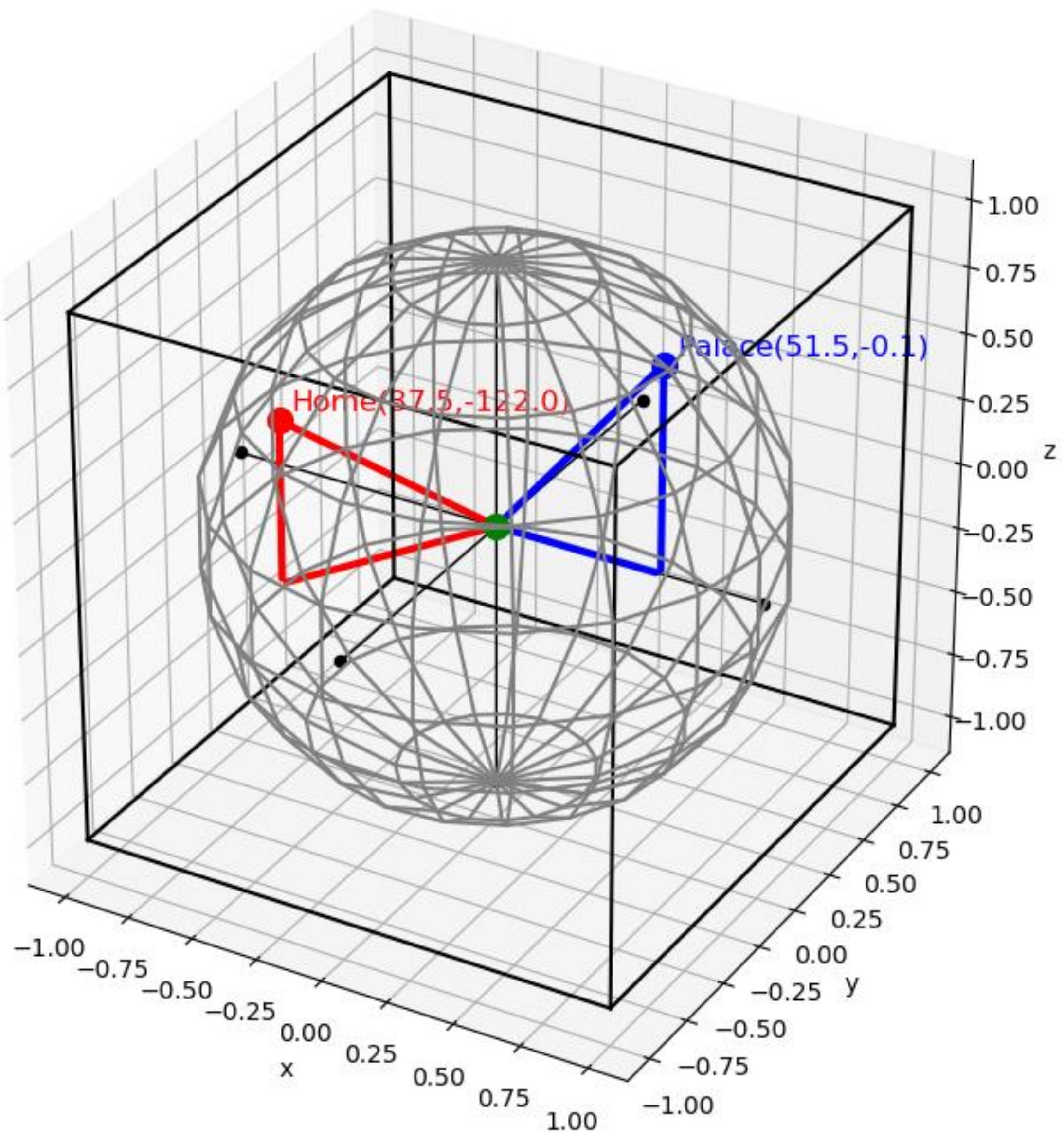
Demo Program: BuckinghamPalace.py

```
g = Arrow3D([0, 0], [0, 0], [0, 1], mutation_scale=20,
            lw=1, arrowstyle="-", color="k")
ax.add_artist(g)
ax.text(xt+0.03,yt+0.03,zt+0.03,  '%s' % "Home(37.5,-122.0)", size=12,
zorder=1,  color='r')
ht = Arrow3D([0, xt], [0, yt], [0, zt], mutation_scale=20,
            lw=3, arrowstyle="-", color="r")
ax.add_artist(ht)
it = Arrow3D([0, xt], [0, yt], [0, 0], mutation_scale=20,
            lw=3, arrowstyle="-", color="r")
ax.add_artist(it)
jt = Arrow3D([xt, xt], [yt, yt], [zt, 0], mutation_scale=20,
            lw=3, arrowstyle="-", color="r")
ax.add_artist(jt)
ax.text(xb+0.03,yb+0.03,zb+0.03,  '%s' % "Palace(51.5,-0.1)", size=12,
zorder=1,  color='b')
```

Draw Vectors

Demo Program: BuckinghamPalace.py

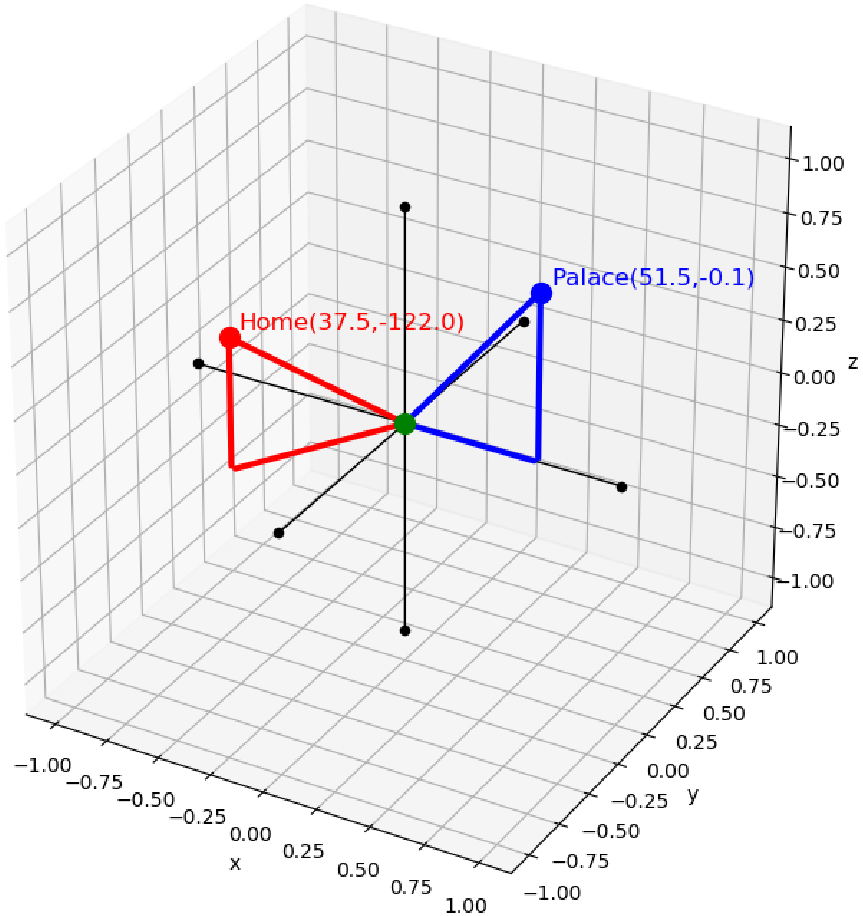
```
hb = Arrow3D([0, xb], [0, yb], [0, zb], mutation_scale=20,
            lw=3, arrowstyle="-", color="b")
ax.add_artist(hb)
ib = Arrow3D([0, xb], [0, yb], [0, 0], mutation_scale=20,
            lw=3, arrowstyle="-", color="b")
ax.add_artist(ib)
jb = Arrow3D([xb, xb], [yb, yb], [zb, 0], mutation_scale=20,
            lw=3, arrowstyle="-", color="b")
ax.add_artist(jb)
```



Home(37.5,-122.0)
Palace(51.5,-0.1)
x
y
z

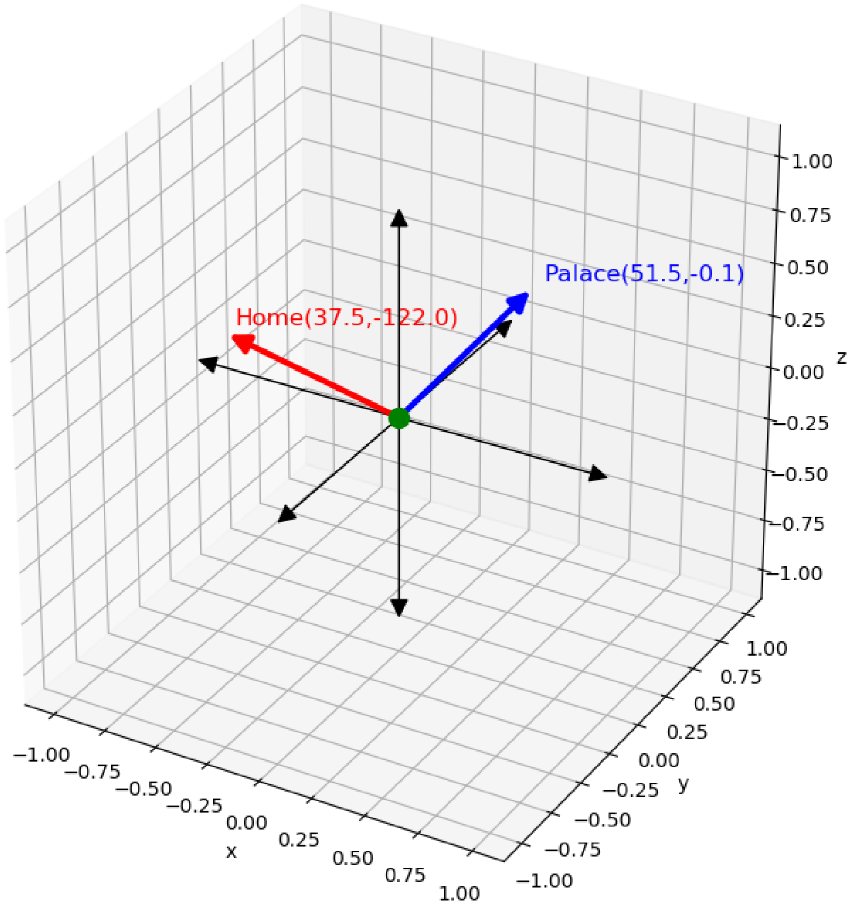
Buckingham Palace

Demo Program: BuckinghamPalaceNoCubeNoBall.py



Buckingham Palace

Demo Program: BuckinghamPalaceVector.py



Storm Tracker

LECTURE 3

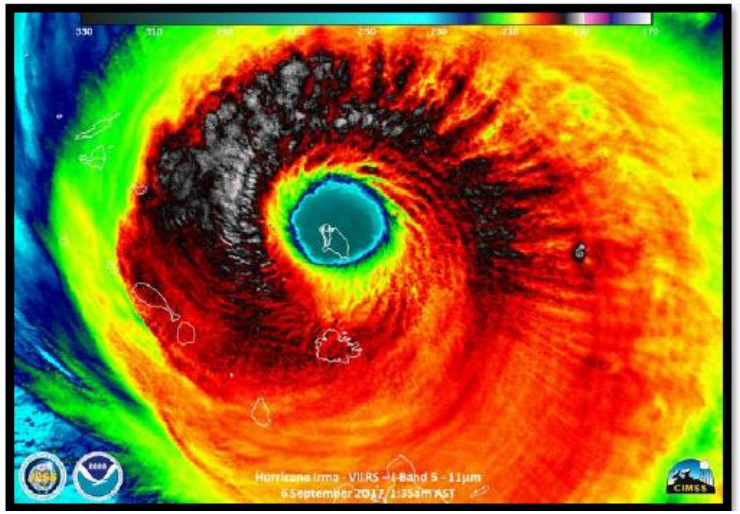
NATIONAL HURRICANE CENTER TROPICAL CYCLONE REPORT

HURRICANE IRMA

(AL112017)

30 August–12 September 2017

John P. Cangialosi, Andrew S. Latto, and Robbie Berg
National Hurricane Center
30 June 2018[1]



https://coast.noaa.gov/hurricanes/#map=4/32/-80

HISTORICAL
HURRICANE TRACKS

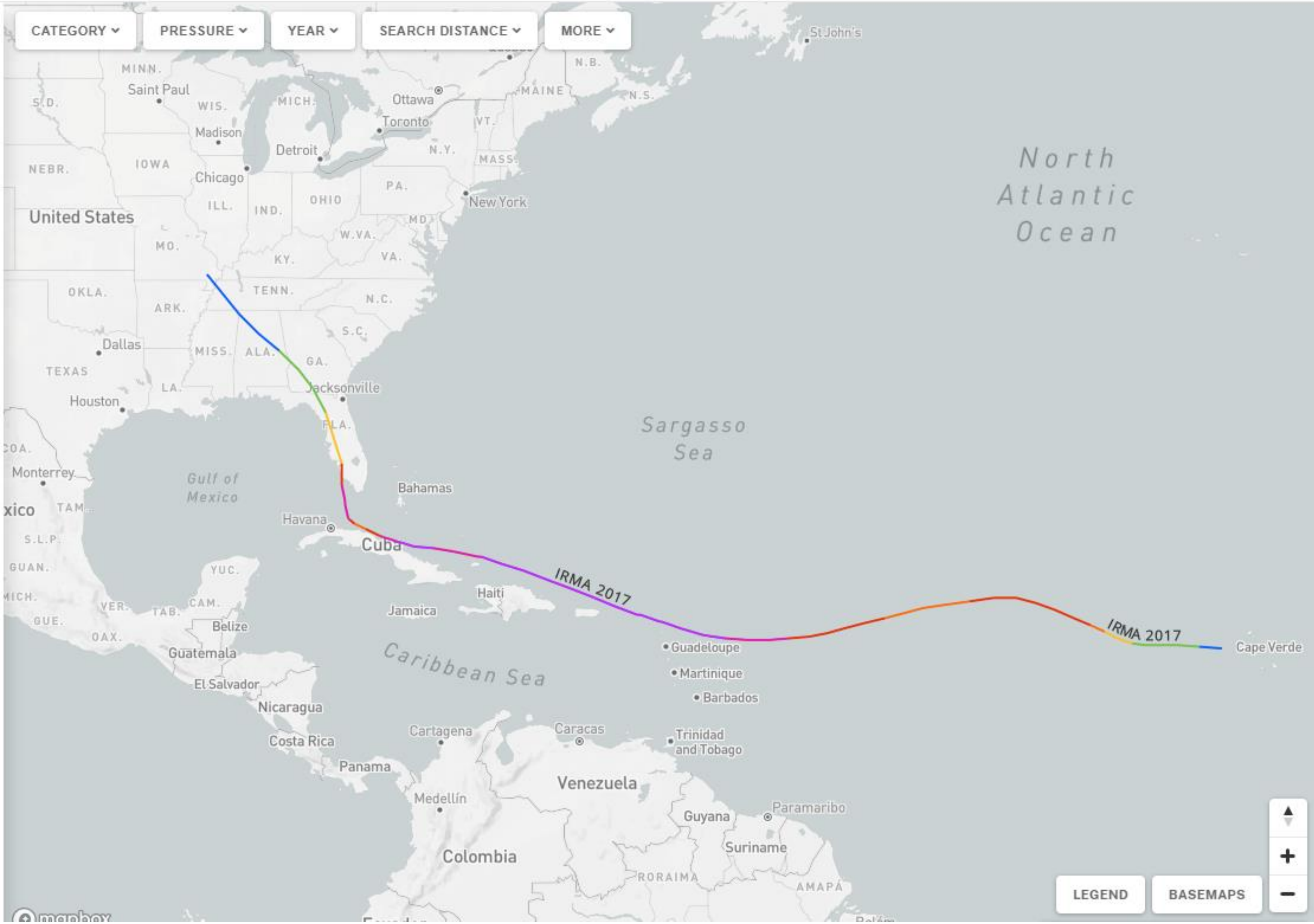
IRMA 2017

MATCHING STORMS
1

SORTED BY
Year (Newest)

HURRICANE
IRMA 2017
Aug 30, 2017 to Sep 13, 2017

CATEGORY | PRESSURE | YEAR | SEARCH DISTANCE | MORE



LEGEND | BASEMAPS

Data Set
File: Irma.txt

- Date/Time(UTC),
- Latitude(°N),
- Longitude(°W),
- Pressure(mb),
- WindSpeed (kt),
- Stage

```python
from pylab import *
from statistics import mean
f = open("Irma.txt", 'r')
tokens = f.readline().split(',')
for i in range(len(tokens)):
    tokens[i] = tokens[i].strip().lstrip()
    #print(tokens[i])
# assign the title strings
title_dete_time = tokens[0]
title_latitude  = tokens[1]
title_longitude = tokens[2]
title_pressure  = tokens[3]
title_windspeed = tokens[4]
title_stage     = tokens[5]
print("%-14s %-13s %-14s %-12s %-14s %-8s" % (
    title_dete_time,
    title_latitude,
    title_longitude,
    title_pressure,
    title_windspeed,
    title_stage
))
```

30	/	0000	16.1	26.9	1008	30	tropical depression
30	/	0600	16.2	28.3	1007	35	tropical storm
30	/	1200	16.3	29.7	1006	45	"
30	/	1800	16.3	30.8	1004	50	"
31	/	0000	16.3	31.7	999	55	"
31	/	0600	16.4	32.5	994	65	hurricane
31	/	1200	16.7	33.4	983	80	"
31	/	1800	17.1	34.2	970	95	"
01	/	0000	17.5	35.1	967	100	"
01	/	0600	17.9	36.1	967	100	"
01	/	1200	18.4	37.3	967	100	"
01	/	1800	18.8	38.5	967	100	"
02	/	0000	19.1	39.7	967	100	"
02	/	0600	19.1	41.1	967	100	"
02	/	1200	18.9	42.6	973	95	"
02	/	1800	18.7	44.1	973	95	"
03	/	0000	18.5	45.5	973	95	"
03	/	0600	18.2	46.7	973	95	"
03	/	1200	17.9	47.9	969	100	"
03	/	1800	17.6	49.2	965	100	"
04	/	0000	17.3	50.4	959	100	"
04	/	0600	17.0	51.5	952	105	"
04	/	1200	16.8	52.6	945	110	"
04	/	1800	16.7	53.9	944	115	"
05	/	0000	16.6	55.1	943	125	"
05	/	0600	16.6	56.4	933	135	"
05	/	1200	16.7	57.8	929	150	"
05	/	1800	16.9	59.2	926	155	"
06	/	0000	17.3	60.6	915	155	"
06	/	0600	17.7	61.9	914	155	"
06	/	1200	18.1	63.3	915	155	"
06	/	1800	18.6	64.7	916	150	"
07	/	0000	19.2	66.2	916	150	"
07	/	0600	19.7	67.6	920	145	"
07	/	1200	20.2	69.0	921	145	"
07	/	1800	20.7	70.4	922	145	"
08	/	0000	21.1	71.8	919	140	"
08	/	0600	21.5	73.2	925	135	"
08	/	1200	21.8	74.7	927	135	"
08	/	1800	22.0	76.0	925	140	"
09	/	0000	22.1	77.2	924	145	"
09	/	0600	22.4	78.3	930	130	"
09	/	1200	22.7	79.3	941	110	"
09	/	1800	23.1	80.2	938	95	"
10	/	0000	23.4	80.9	932	100	"
10	/	0600	23.7	81.3	930	115	"
10	/	1200	24.5	81.5	931	115	"
10	/	1800	25.6	81.7	936	100	"
11	/	0000	26.8	81.7	942	80	"
11	/	0600	28.2	82.2	961	65	"
11	/	1200	29.6	82.7	970	50	tropical storm
11	/	1800	30.9	83.5	980	45	"
12	/	0000	31.9	84.4	986	35	"
12	/	0600	32.9	85.6	997	25	low
12	/	1200	33.8	86.9	1000	20	"
12	/	1800	34.8	88.1	1003	15	"
13	/	0000	35.6	88.9	1004	15	"
13	/	0600	36.2	89.5	1004	15	"
13	/	1200	36.8	90.1	1005	15	"

```python
irma_path = []
lines = f.readlines()
for line in lines:
    tokens = line.split(' ')
    d = dict()
    for i in range(len(tokens)):
        tokens[i] = tokens[i].strip().lstrip()
        #print(tokens[i], end=" ")
        d['date'] = int(tokens[0])
        d['hour'] = tokens[2]
        d['location'] = (float(tokens[3]), float(tokens[4]))
        d['pressure'] = int(tokens[5])
        d['wind'] = int(tokens[6])
        d['type'] = tokens[7]
        for j in range(8, len(tokens)):
            d['type'] += " "+tokens[j]

    print("    %-14s %-13.2f %-14.2f %-10d %-14d %-s" % (
        str(""+str(d['date'])+"/"+d['hour']),
        d['location'][0],
        d['location'][1],
        d['pressure'],
        d['wind'],
        d['type']
    ))
    irma_path.append(d)
```

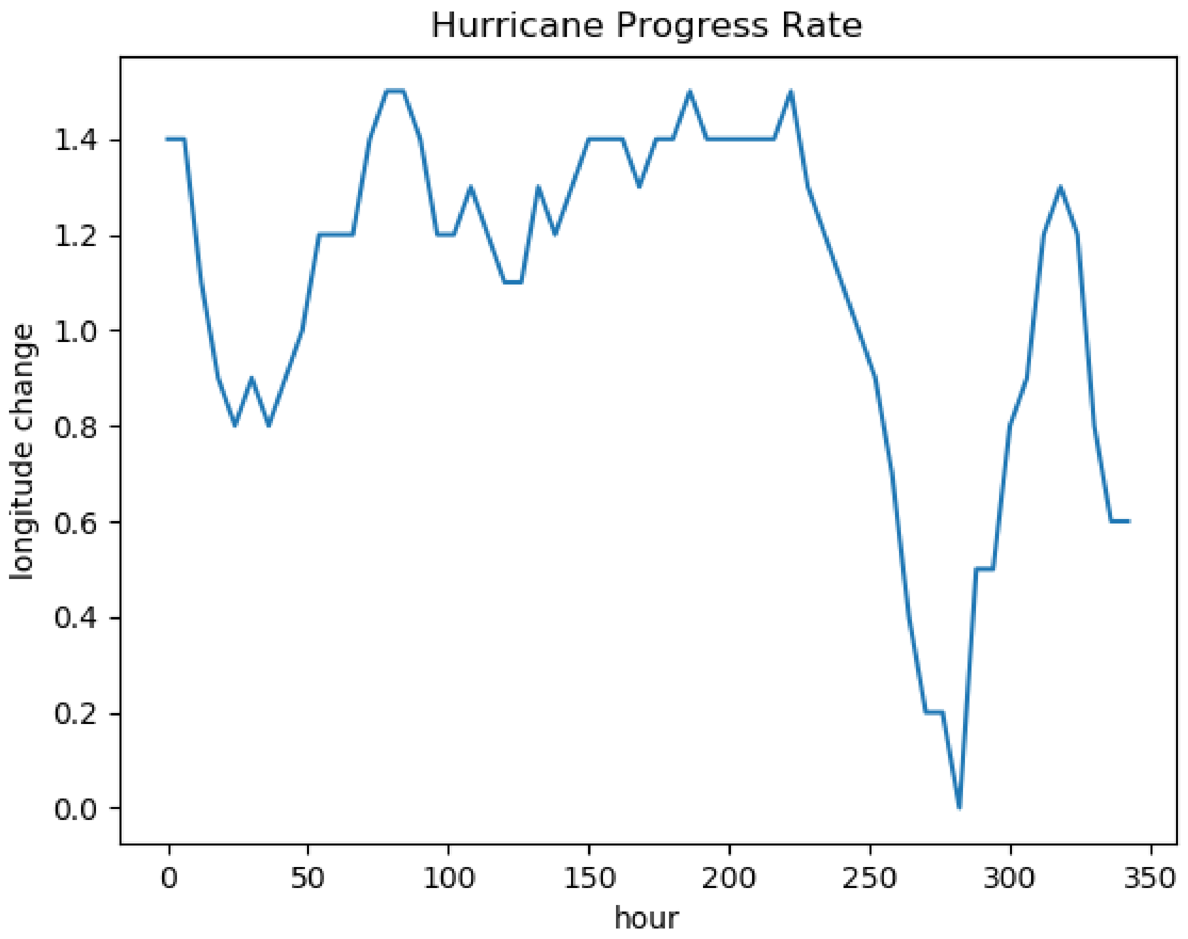
eC Learning Channel

```python
def category(wind):
    if (wind>=157): return 5
    if (wind>=130): return 4
    if (wind>=111): return 3
    if (wind>=96): return 2
    if (wind>=74): return 1
    return 0
```

10 WEATHER SAFFIR-SIMPSON HURRICANE WIND SCALE

CATEGORY	WINDS (MPH)	DAMAGE
1	74-95	SOME
2	96-110	EXTENSIVE
3	111-129	DEVASTATING
4	130-156	CATASTROPHIC
5	157+	CATASTROPHIC

```
g = open("Irma_cat.txt", "w+")
print()
print()
print("Date:Time              Lat.   Long. Cat.", file=g)
# filtering out non-hurricane points.
for d in irma_path:
    if (category(d['wind']*1.15078)==0):
        del d
    elif (int(d['hour'][:2])%6!=0): del d         # take out extra points
    else:
        d['cat'] = category(d['wind']*1.15078)
        if (d['date']>15): print('2017/08', end='', file=g)
        else: print('2017/09', end='', file=g)
        print("/%02d" % d['date'], end=":", file=g)
        print("[%2s:%2s]" % (d['hour'][:2], d['hour'][2:]), end='  ', file=g)
        print("%5.2f" % d['location'][0], end='  ', file=g)
        print("%5.2f" % d['location'][1], end='  ', file=g)
        print(" %d " % d['cat'], file=g)

lat_list = [d['location'][0] for d in irma_path]
lon_list = [d['location'][1] for d in irma_path]

lon = [i*6 for i in range(len(lon_list)-1)]
lon_diff = [(lon_list[i+1]-lon_list[i]) for i in range(len(lon_list)-1)]
avg = mean(lon_diff)
```




```
figure()
title('Hurricane Progress Rate')
xlabel('hour')
ylabel('longitude change')
plot(lon, lon_diff)
show()
```

```
print("Total Number of Data Points: ", len(lat_list))
# regression (Exponential Regression)
t = linspace(0, 6*(len(lon_list)-1), len(lon_list))
x = 26.9 + avg * (t//6)
print(x)
print(avg)
a = 16.1
b = 1.013
y = a * b ** (x-26.9)

a1 = 0.000165
b1 = 0.0000013
c1 = 16.1
y1 = a1*t**2 + b1*t + c1

a2 = 0.00000142
b2 = -0.00050
c2 = 0.060
d2 = 16.1
y2 = a2*t**3 + b2*t**2 + c2 * t + d2
```

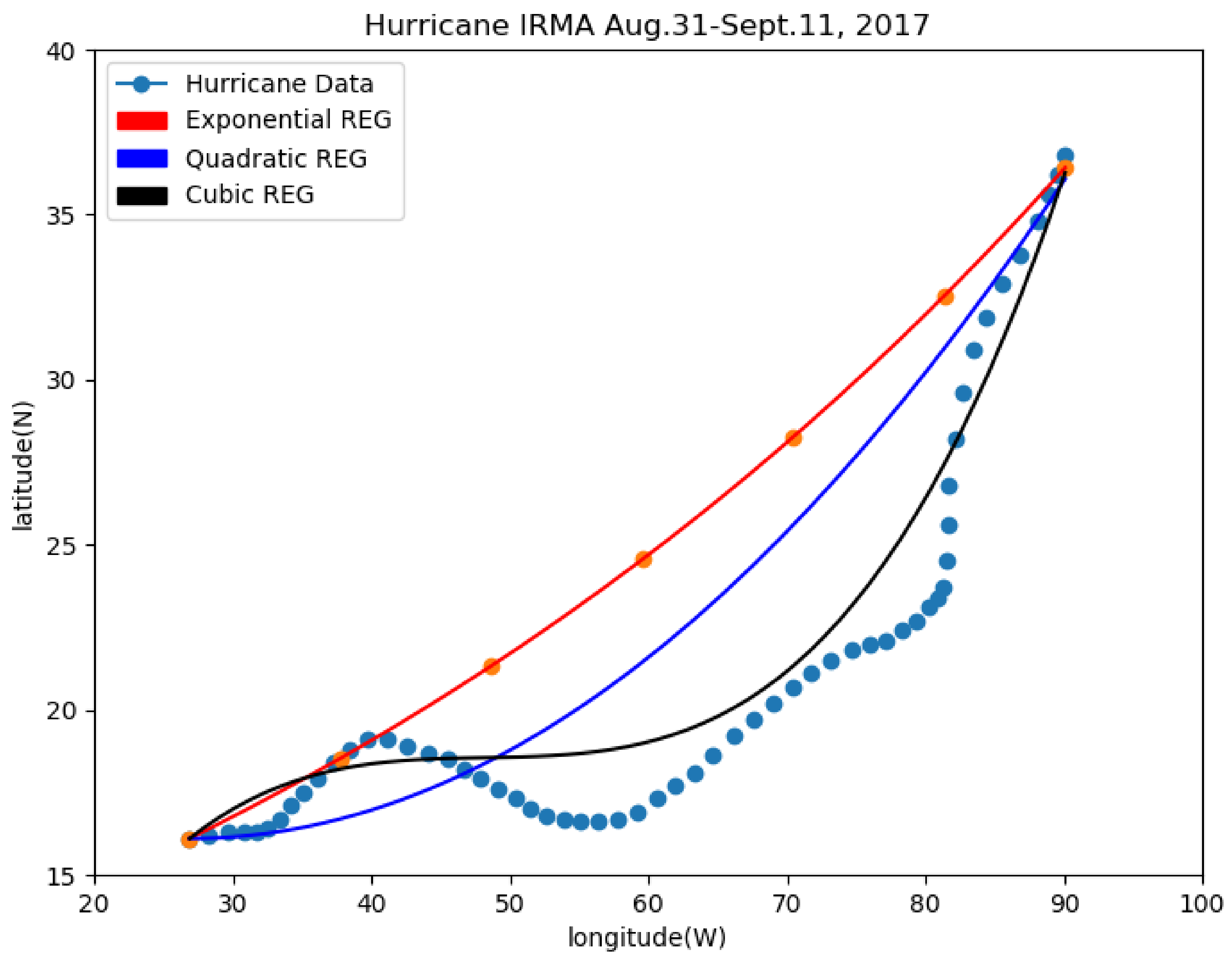
```
Total Number of Data Points:  59
[26.9        27.98965517 29.07931034 30.16896552 31.25862069 32.34827586
 33.43793103 34.52758621 35.61724138 36.70689655 37.79655172 38.8862069
 39.97586207 41.06551724 42.15517241 43.24482759 44.33448276 45.42413793
 46.5137931  47.60344828 48.69310345 49.78275862 50.87241379 51.96206897
 53.05172414 54.14137931 55.23103448 56.32068966 57.41034483 58.5
 59.58965517 60.67931034 61.76896552 62.85862069 63.94827586 65.03793103
 66.12758621 67.21724138 68.30689655 69.39655172 70.4862069  71.57586207
 72.66551724 73.75517241 74.84482759 75.93448276 77.02413793 78.1137931
 79.20344828 80.29310345 81.38275862 82.47241379 83.56206897 84.65172414
 85.74137931 86.83103448 87.92068966 89.01034483 90.1       ]
1.089655172413793
```

```
print("Date:Time             Lat.   Long. ")
z = 0
for d in irma_path:
    if (d['date'] > 15): print('2017/08', end='')
    else: print('2017/09', end='')
    print("/%02d" % d['date'], end=":")
    print("[%2s:%2s]" % (d['hour'][:2], d['hour'][2:]), end='  ')
    print("%5.2f" % y2[z], end='  ')
    print("%5.2f" % x[z])
    z += 1
```

Date:Time	Lat.	Long.
2017/08/30:[00:00]	16.10	26.90
2017/08/30:[06:00]	16.44	27.99
2017/08/30:[12:00]	16.75	29.08
2017/08/30:[18:00]	17.03	30.17
2017/08/31:[00:00]	17.27	31.26
2017/08/31:[06:00]	17.49	32.35
2017/08/31:[12:00]	17.68	33.44
2017/08/31:[18:00]	17.84	34.53
2017/09/01:[00:00]	17.99	35.62
2017/09/01:[06:00]	18.11	36.71
2017/09/01:[12:00]	18.21	37.80
2017/09/01:[18:00]	18.29	38.89
2017/09/02:[00:00]	18.36	39.98
2017/09/02:[06:00]	18.41	41.07
2017/09/02:[12:00]	18.45	42.16
2017/09/02:[18:00]	18.49	43.24
2017/09/03:[00:00]	18.51	44.33
2017/09/03:[06:00]	18.52	45.42
2017/09/03:[12:00]	18.54	46.51
2017/09/03:[18:00]	18.55	47.60
2017/09/04:[00:00]	18.55	48.69
2017/09/04:[06:00]	18.56	49.78
2017/09/04:[12:00]	18.57	50.87
2017/09/04:[18:00]	18.59	51.96
2017/09/05:[00:00]	18.61	53.05
2017/09/05:[06:00]	18.64	54.14
2017/09/05:[12:00]	18.68	55.23
2017/09/05:[18:00]	18.74	56.32
2017/09/06:[00:00]	18.80	57.41
2017/09/06:[06:00]	18.88	58.50

Date:Time	Lat.	Long.
2017/09/06:[12:00]	18.98	59.59
2017/09/06:[18:00]	19.10	60.68
2017/09/07:[00:00]	19.24	61.77
2017/09/07:[06:00]	19.40	62.86
2017/09/07:[12:00]	19.59	63.95
2017/09/07:[18:00]	19.80	65.04
2017/09/08:[00:00]	20.04	66.13
2017/09/08:[06:00]	20.31	67.22
2017/09/08:[12:00]	20.62	68.31
2017/09/08:[18:00]	20.96	69.40
2017/09/09:[00:00]	21.33	70.49
2017/09/09:[06:00]	21.74	71.58
2017/09/09:[12:00]	22.19	72.67
2017/09/09:[18:00]	22.68	73.76
2017/09/10:[00:00]	23.22	74.84
2017/09/10:[06:00]	23.80	75.93
2017/09/10:[12:00]	24.43	77.02
2017/09/10:[18:00]	25.10	78.11
2017/09/11:[00:00]	25.83	79.20
2017/09/11:[06:00]	26.61	80.29
2017/09/11:[12:00]	27.44	81.38
2017/09/11:[18:00]	28.33	82.47
2017/09/12:[00:00]	29.28	83.56
2017/09/12:[06:00]	30.28	84.65
2017/09/12:[12:00]	31.35	85.74
2017/09/12:[18:00]	32.48	86.83
2017/09/13:[00:00]	33.68	87.92
2017/09/13:[06:00]	34.94	89.01
2017/09/13:[12:00]	36.27	90.10

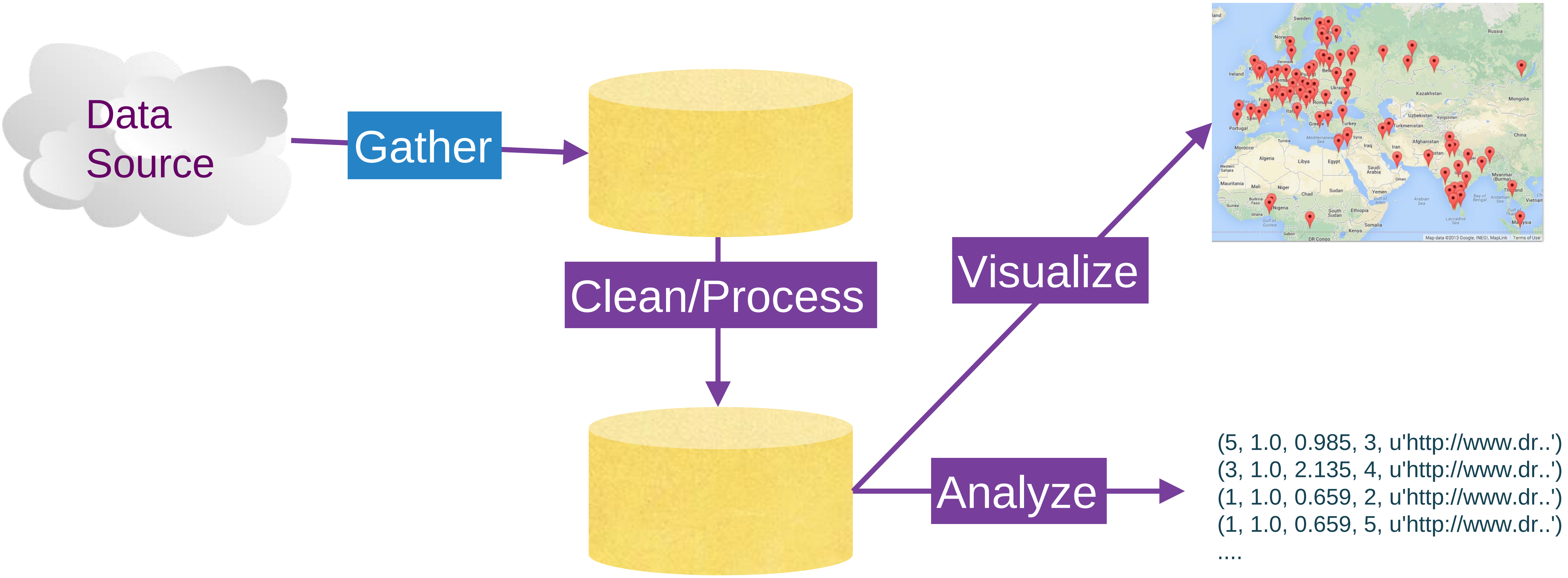
```python
import matplotlib.patches as mpatches
figure('Hurricane IRMA Aug.31-Sept.11, 2017', figsize=(8, 6))
xscale('linear')
yscale('linear')
title('Hurricane IRMA Aug.31-Sept.11, 2017')
xlim(20, 100)
ylim(15, 40)
xlabel('longitude(W)')
ylabel('latitude(N)')
scatter(lon_list, lat_list)
plot(x, y, 'r')

scatter([x[0], x[10], x[20], x[30], x[40], x[50], x[58]],
        [y[0], y[10], y[20], y[30], y[40], y[50], y[58]])
plot(x, y1, 'b')
plot(x, y2, 'k')

data_patch = Line2D([], [], marker='o', label='Hurricane Data')
red_patch = mpatches.Patch(color='r', label='Exponential REG')
blue_patch = mpatches.Patch(color='b', label='Quadratic REG')
black_patch = mpatches.Patch(color='k', label='Cubic REG')
legend(handles=[data_patch, red_patch, blue_patch, black_patch], loc=2)
show()
```


Hurricane IRMA Aug.31-Sept.11, 2017
Hurricane Data
Exponential REG
Quadratic REG
Cubic REG
latitude(N)
longitude(W)
40
35
30
25
20
15
20
30
40
50
60
70
80
90
100

World Universities

LECTURE 4

Multi-Step Data Analysis



Many Data Mining Technologies

https://hadoop.apache.org/

http://spark.apache.org/

https://aws.amazon.com/redshift/

http://community.pentaho.com/

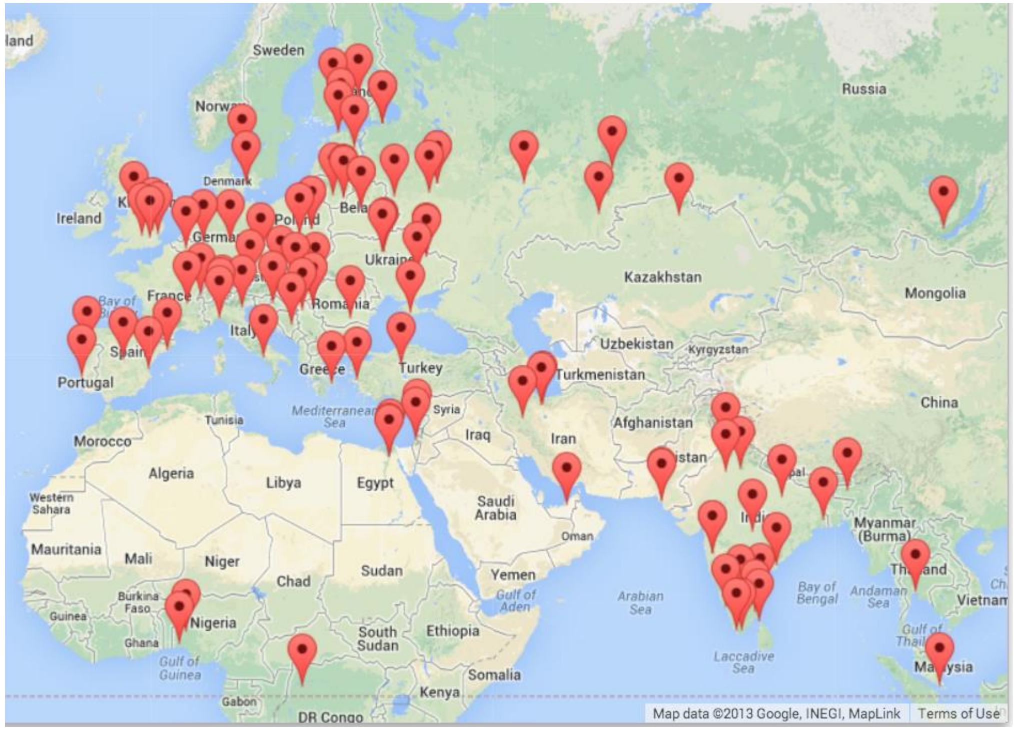
....

"Personal Data Mining"

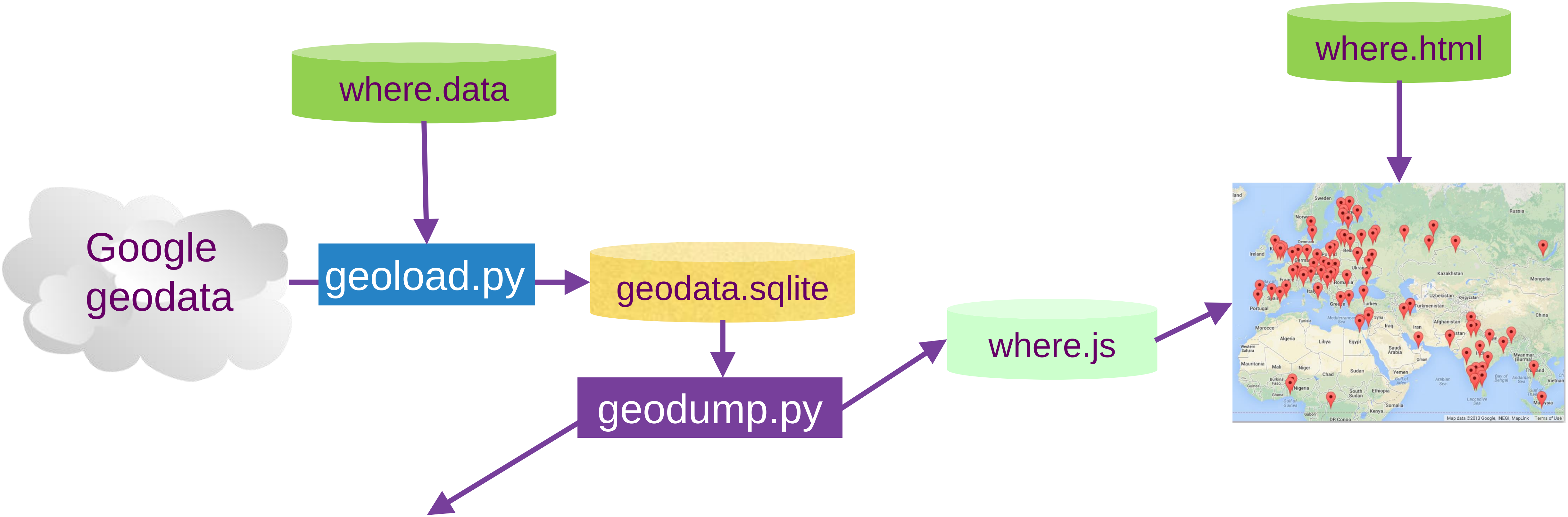
- Our goal is to make you better programmers – not to make you data mining experts

GeoData

- Makes a Google Map from user entered data
- Uses the Google Geodata API
- Caches data in a database to avoid rate limiting and allow restarting
- Visualized in a browser using the Google Maps API



http://www.py4e.com/code3/geodata.zip



Northeastern University, ... Boston, MA 02115, USA 42.3396998 -71.08975
Bradley University, 1501 ... Peoria, IL 61625, USA 40.6963857 -89.6160811
...
Technion, Viazman 87, Kesalsaba, 32000, Israel 32.7775 35.0216667
Monash University Clayton ... VIC 3800, Australia -37.9152113 145.134682
Kokshetau, Kazakhstan 53.2833333 69.3833333
...
12 records written to where.js
Open where.html to view the data in a browser

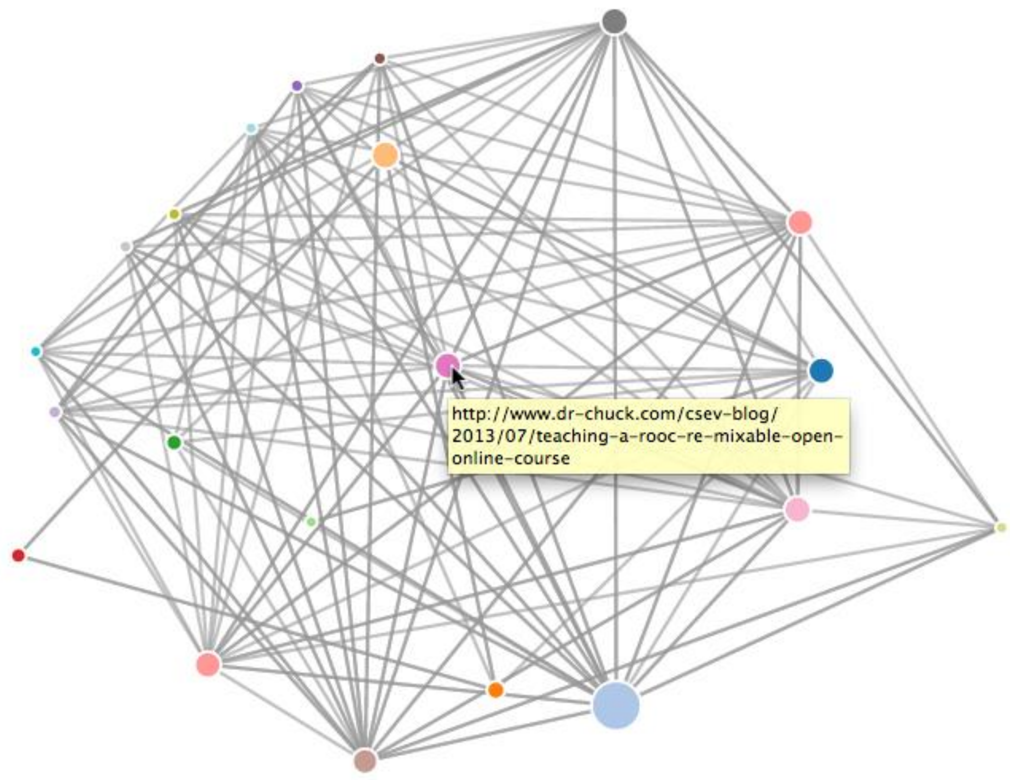
http://www.py4e.com/code3/geodata.zip

Web Crawler

LECTURE 5

Page Rank

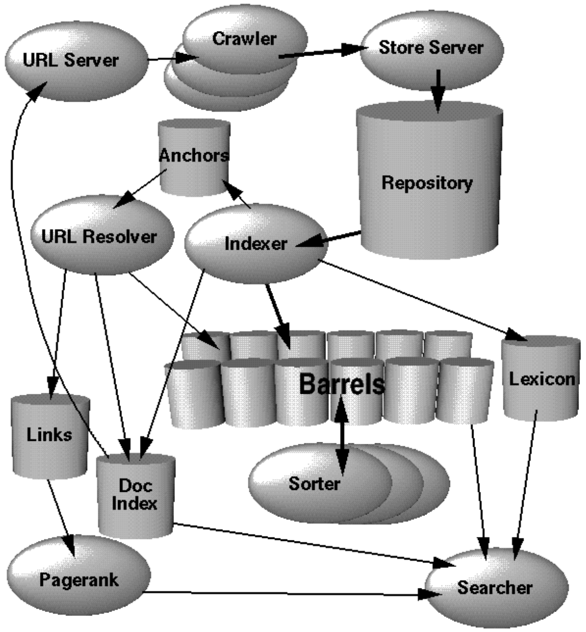
- Write a simple web page crawler
- Compute a simple version of Google's Page Rank algorithm
- Visualize the resulting network



http://www.py4e.com/code3/pagerank.zip

Search Engine Architecture

- Web Crawling
- Index Building
- Searching



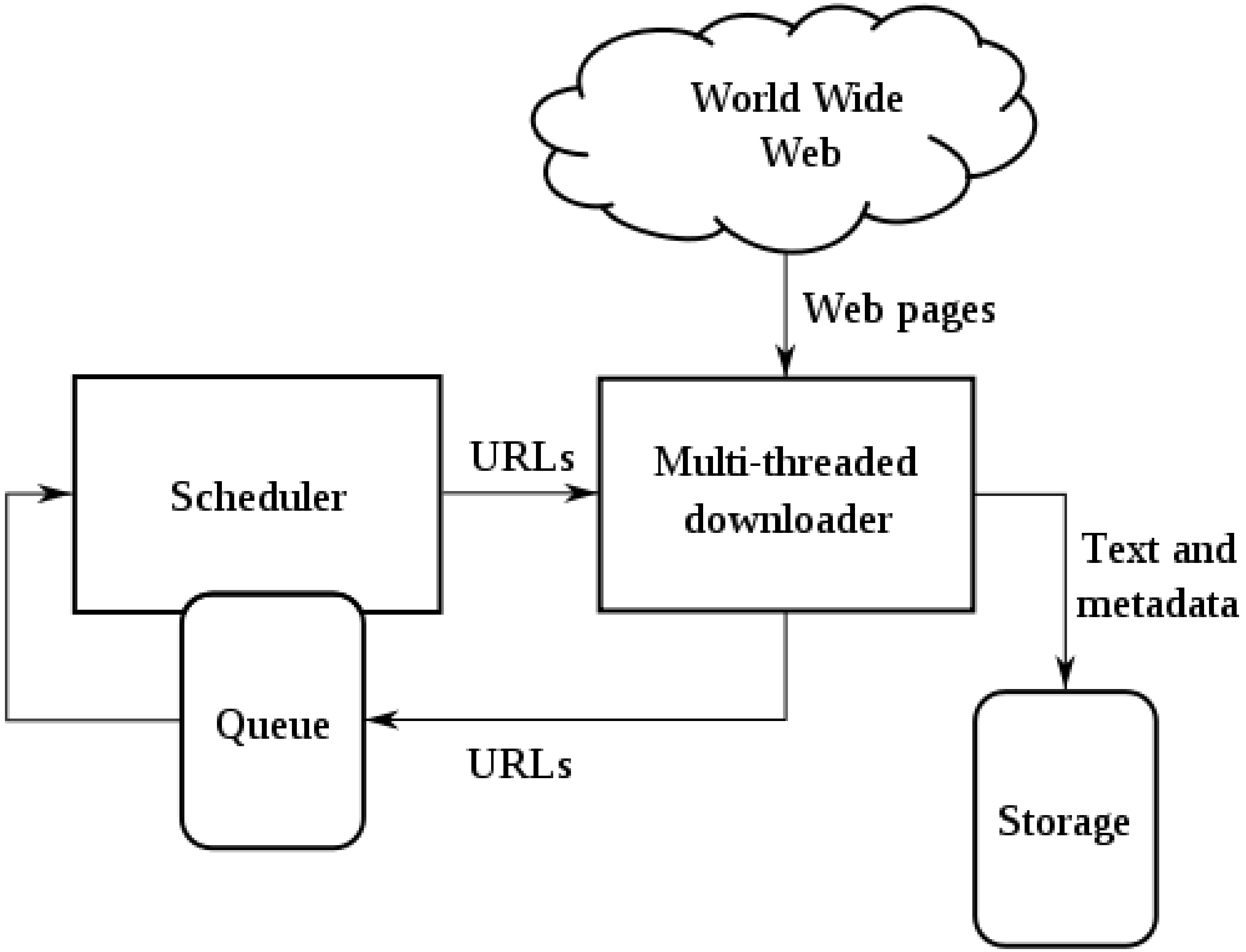
http://infolab.stanford.edu/~backrub/google.html

Web Crawler

- A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches.

Web Crawler

- Retrieve a page
- Look through the page for links
- Add the links to a list of “to be retrieved” sites
- Repeat…



Web Crawling Policy

- a selection policy that states which pages to download,
- a re-visit policy that states when to check for changes to the pages,
- a politeness policy that states how to avoid overloading Web sites, and
- a parallelization policy that states how to coordinate distributed Web crawlers

robots.txt

- A way for a web site to communicate with web crawlers
- An informal and voluntary standard
- Sometimes folks make a “Spider Trap” to catch “bad” spiders

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /images/
Disallow: /tmp/
Disallow: /private/
```

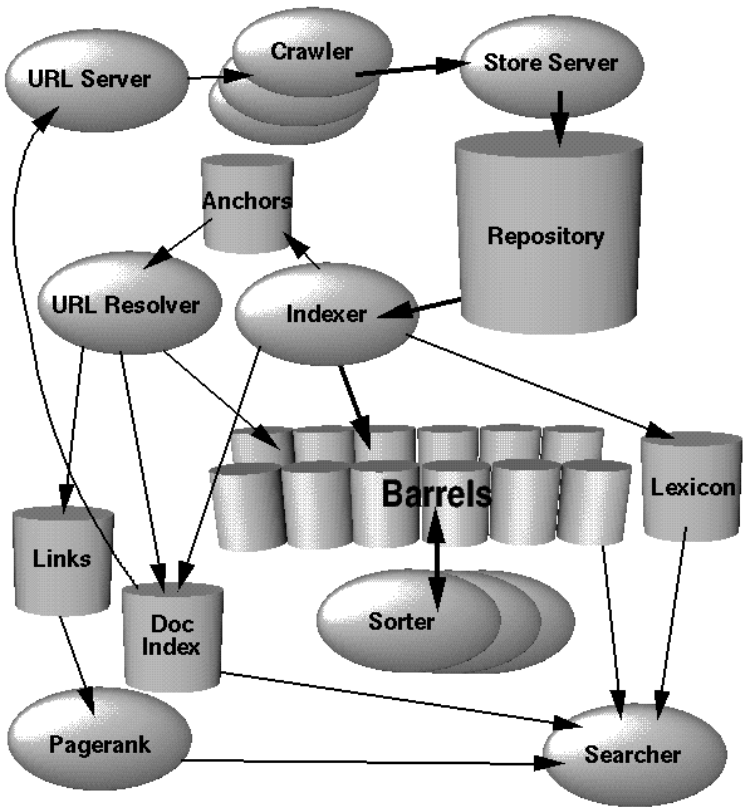
http://en.wikipedia.org/wiki/Robots_Exclusion_Standard
http://en.wikipedia.org/wiki/Spider_trap

Google Architecture

LECTURE 6

Google Architecture

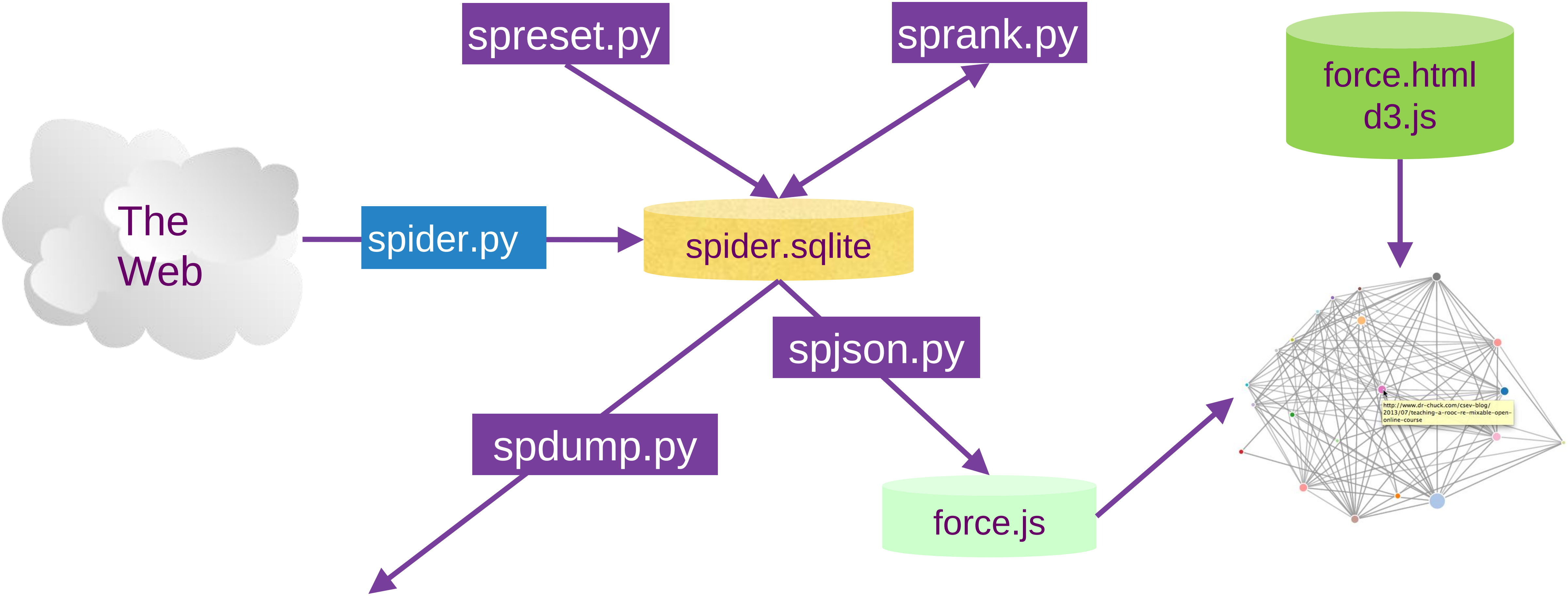
- Web Crawling
- Index Building
- Searching



http://infolab.stanford.edu/~backrub/google.html

Search Indexing

- Search engine indexing collects, parses, and stores data to facilitate fast and accurate information retrieval. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power.



(5, None, 1.0, 3, u'http://www.dr-chuck.com/csev-blog')
(3, None, 1.0, 4, u'http://www.dr-chuck.com/dr-chuck/resume/speaking.htm')
(1, None, 1.0, 2, u'http://www.dr-chuck.com/csev-blog/')
(1, None, 1.0, 5, u'http://www.dr-chuck.com/dr-chuck/resume/index.htm')
4 rows.

http://www.py4e.com/code3/pagerank.zip

Mailing Lists - Gmane

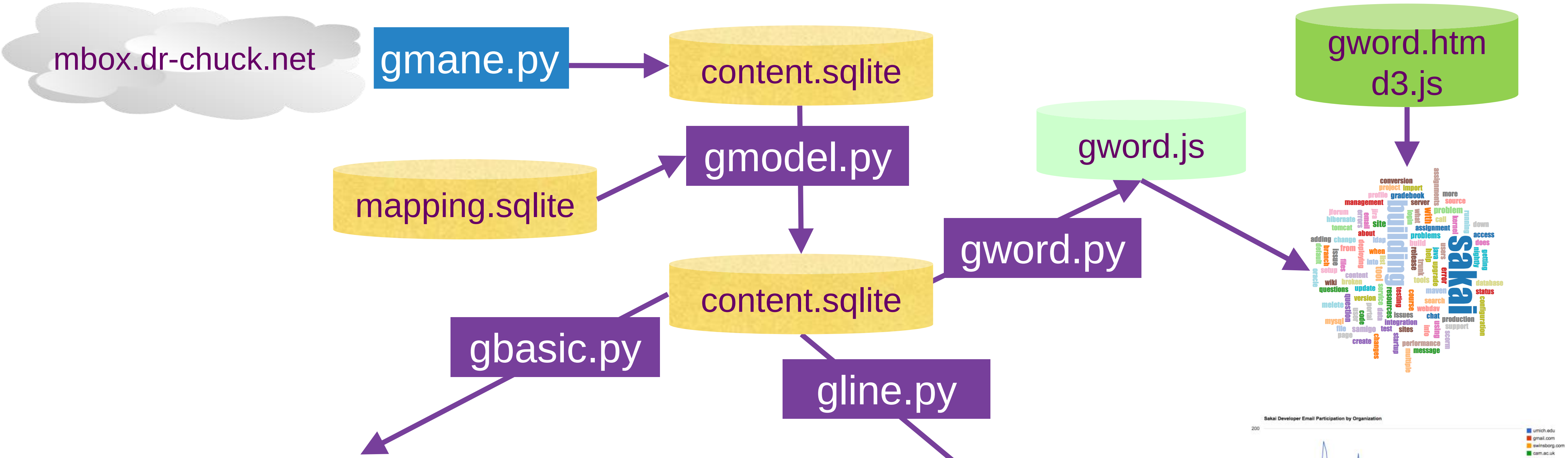
- Crawl the archive of a mailing list
- Do some analysis / cleanup
- Visualize the data as word cloud and lines

http://www.py4e.com/code3/gmane.zip

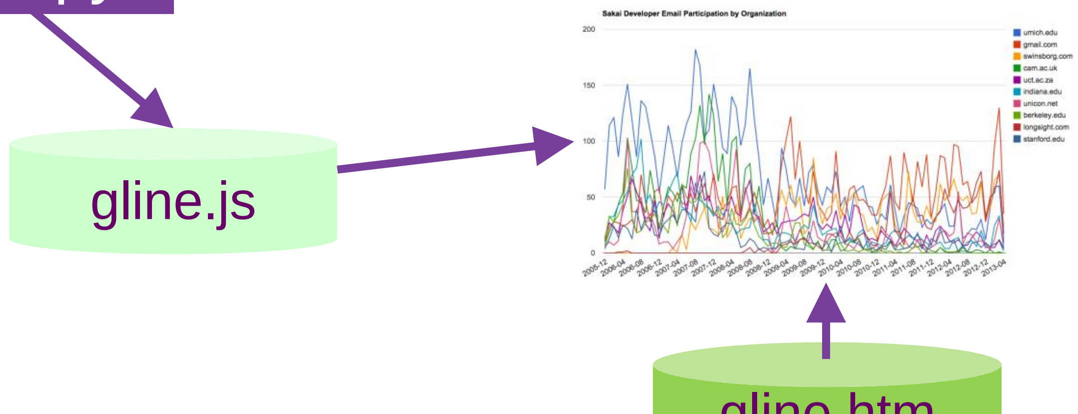
Warning: This Dataset is > 1GB

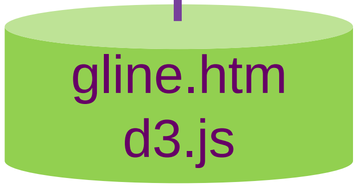
- Do not just point this application at gmane.org and let it run
- There is no rate limit – these are cool folks

```
Use this for your testing:
http://mbox.dr-chuck.net/sakai.devel/4/5
```



How many to dump? 5
Loaded messages= 51330 subjects= 25033 senders= 1584
Top 5 Email list participants
steve.swinsburg@gmail.com 2657
azeckoski@unicon.net 1742
ieb@tfd.co.uk 1591
csev@umich.edu 1304
david.horwitz@uct.ac.za 1184
...





http://www.py4e.com/code3/gmane.zip