Untitled

October 27, 2021

[1]:
```
from pprint import pprint
from datetime import datetime
import time
import os
import random
import uuid
import json
import numpy as np
import pandas as pd
import geojson
import geopandas as gpd
from shapely.geometry import Point
from faker import Faker
fake = Faker()

import requests
access_key = "fxeZJJV_ybTUld6lxUBF4jcu_T7ol67H801lr1pIR-4"
secret_key = "_4ErP8rCoHwvgOVbmxIQjh4HSj2z53PDC53BqH-tWTI"
```

[2]:
```
df = gpd.read_file("nsw_poi.json")
```

[3]:
```
original_crs = df['geometry'].crs
original_crs
```

[3]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

[4]:
```
df['geometry'] = df['geometry'].to_crs(epsg='7856')
df['geometry'].crs
```

[4]:
```
<Projected CRS: EPSG:7856>
Name: GDA2020 / MGA zone 56
Axis Info [cartesian]:
- E[east]: Easting (metre)
- N[north]: Northing (metre)
Area of Use:
- name: Australia - onshore and offshore between 150°E and 156°E.
- bounds: (150.0, -58.96, 156.0, -13.87)
Coordinate Operation:
- name: Map Grid of Australia zone 56
- method: Transverse Mercator
Datum: Geocentric Datum of Australia 2020
- Ellipsoid: GRS 1980
- Prime Meridian: Greenwich
```

[5]:
```
df['geometry'] = df['geometry'].centroid
```

[6]:
```
# sydney town hall
wgs84_pt = Point(151.206323, -33.873235)


selected_point = gpd.GeoDataFrame(pd.DataFrame({'geometry':[wgs84_pt]}),␣
↪crs=original_crs)
selected_point.to_crs(epsg='7856', inplace=True)
```

[7]:
```
selected_point['geometry']
```

[7]:
```
0    POINT (334102.302 6250453.133)
Name: geometry, dtype: geometry
```

[8]:
```
df['distance'] = df.geometry.distance(selected_point['geometry'][0])
df
```

[8]:

	osm_id	fclass	name	geometry	distance
0	4000006	pitch	None		
1	4085520	park	Ollie Webb Reserve		
2	4292638	golf_course	Cammeray Golf Course		
3	4292684	park	Green Park		
4	4294907	park	Irene Auston Reserve		
…	…	…	…		
57261	954669787	swimming_pool	None		
57262	954669788	swimming_pool	None		
57263	954669789	swimming_pool	None		
57264	954669790	swimming_pool	None		
57265	954669791	swimming_pool	None		

```
0      POINT (335848.309 6248440.619)   2664.348021
1      POINT (314648.596 6255798.270)  20174.666741
2      POINT (334893.317 6255779.923)   5385.201747
3      POINT (335400.408 6249772.912)   1465.530040
4      POINT (363426.155 6334423.934)  88943.711679
…                                 …           …
57261  POINT (339425.436 6243120.959)   9060.713327
57262  POINT (339434.891 6243131.825)   9057.486389
57263  POINT (339449.962 6243127.006)   9070.258580
57264  POINT (339439.176 6243183.163)   9018.574291
57265  POINT (339501.195 6243048.209)   9164.111554

[57266 rows x 5 columns]
```

[9]:
```
df[['distance']].describe()
```

[9]:
```
           distance
count  5.726600e+04
mean   1.492268e+05
std    1.823581e+05
min    1.452481e+01
25%    1.874553e+04
50%    5.702816e+04
75%    2.430072e+05
max    1.006332e+06
```

[10]:
```
df = df[df['distance'] < 10000].copy()
df
```

[10]:
```
          osm_id         fclass                  name  \
0        4000006          pitch                  None
2        4292638    golf_course  Cammeray Golf Course
3        4292684           park            Green Park
16       4331551           park           Milson Park
17       4331586           park                  None
…            …              …                     …
57261  954669787  swimming_pool                  None
57262  954669788  swimming_pool                  None
57263  954669789  swimming_pool                  None
57264  954669790  swimming_pool                  None
57265  954669791  swimming_pool                  None

                            geometry     distance
0      POINT (335848.309 6248440.619)  2664.348021
2      POINT (334893.317 6255779.923)  5385.201747
3      POINT (335400.408 6249772.912)  1465.530040
16     POINT (334815.532 6253676.343)  3301.178303
```

```
17     POINT (335617.691 6253568.811)  3464.657405
…                                  …          …
57261  POINT (339425.436 6243120.959)  9060.713327
57262  POINT (339434.891 6243131.825)  9057.486389
57263  POINT (339449.962 6243127.006)  9070.258580
57264  POINT (339439.176 6243183.163)  9018.574291
57265  POINT (339501.195 6243048.209)  9164.111554

[5853 rows x 5 columns]
```

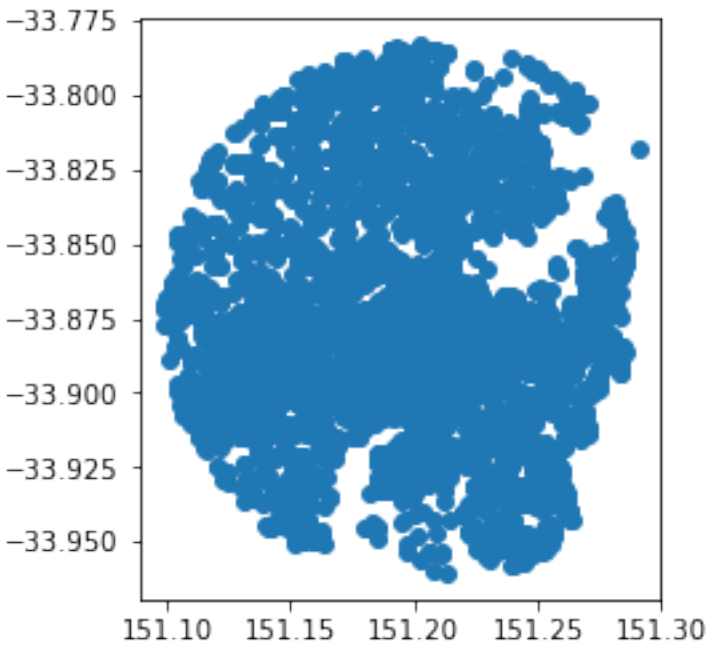
[11]:
```
df['geometry'] = df['geometry'].to_crs(original_crs)
df['geometry'].crs
```

[11]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

[12]:
```
df.drop(columns=['distance'], inplace=True)
```

[13]:
```
df.plot()
```

[13]:
```
<AxesSubplot:>
```



[14]:
```
df = df[df['name'].isnull() == False]
```

[15]:
```
df
```

[15]:

	osm_id	fclass	name	geometry
2	4292638	golf_course	Cammeray Golf Course	POINT (151.21588 -33.82533)
3	4292684	park	Green Park	POINT (151.22023 -33.87956)
16	4331551	park	Milson Park	POINT (151.21464 -33.84428)
18	4331682	golf_course	The Australian Golf Club	POINT (151.21490 -33.91721)
19	4331768	park	Booralee Park	POINT (151.20085 -33.94098)
…	…	…	…	…
56560	943440280	hospital	Castlecrag Private Hosptial	POINT (151.21701 -33.80176)
56561	2790872	park	Lovetts Reserve	
56717	945947718	cafe	Five Dock Falcons Baseball Club	
56718	945947723	cafe	Livvi's Cafe	
56816	948785248	pub	The Red Lion	

```
56561  POINT (151.14793 -33.81886)
56717  POINT (151.13701 -33.87159)
56718  POINT (151.13672 -33.87217)
56816  POINT (151.16880 -33.86490)

[2423 rows x 4 columns]
```

[16]:
```
data = []

for idx, row in df.iterrows():
    elem = {}
    elem['lat'] = row['geometry'].y
    elem['lon'] = row['geometry'].x
    elem['name'] = row['name']
    data += elem,

pois = data
```

[17]:
```
with open("../app/src/main/res/raw/pois.json", "w") as f:
    json.dump(pois, f)
```

0.1 Get random pics for posts in feed

[18]:
```
images = []
path = '/photos/random'

for i in range(30):
    result = requests.get(f"http://api.unsplash.com{path}/?
 ↪client_id={access_key}", params={'count':30, 'width':300})
    if result.json():
        images.extend(result.json())

pprint(images[0])
```

```
{'alt_description': 'white concrete building with rows of glass windows',
 'blur_hash': 'LLJb5Gt7Rjj[_NayWBj[ofofoLj[',
 'categories': [],
 'color': '#c0c0c0',
 'created_at': '2019-08-07T00:45:55-04:00',
 'current_user_collections': [],
 'description': None,
 'downloads': 3100,
 'exif': {'aperture': '2.4',
          'exposure_time': '1/1900',
          'focal_length': '4.3',
          'iso': 50,
          'make': 'samsung',
          'model': 'SM-G970F',
```

```
          'name': 'samsung, SM-G970F'},
 'height': 3024,
 'id': 'yhuU_U28Vhk',
 'liked_by_user': False,
 'likes': 51,
 'links': {'download': 'https://unsplash.com/photos/yhuU_U28Vhk/download?ixid=Mn
wyNjA3NjR8MHwxfHJhbmRvbXx8fHx8fHx8fDE2MzQ4NzU0NTE',
           'download_location': 'https://api.unsplash.com/photos/yhuU_U28Vhk/dow
nload?ixid=MnwyNjA3NjR8MHwxfHJhbmRvbXx8fHx8fHx8fDE2MzQ4NzU0NTE',
           'html': 'https://unsplash.com/photos/yhuU_U28Vhk',
           'self': 'https://api.unsplash.com/photos/yhuU_U28Vhk'},
 'location': {'city': None,
              'country': None,
              'name': None,
              'position': {'latitude': None, 'longitude': None},
              'title': None},
 'promoted_at': '2021-10-06T22:24:01-04:00',
 'sponsorship': None,
 'topic_submissions': {'architecture': {'status': 'rejected'}},
 'updated_at': '2021-10-21T23:59:19-04:00',
 'urls': {'full': 'https://images.unsplash.com/photo-1565153149760-49d028a7f657?
crop=entropy&cs=srgb&fm=jpg&ixid=MnwyNjA3NjR8MHwxfHJhbmRvbXx8fHx8fHx8fDE2MzQ4NzU
0NTE&ixlib=rb-1.2.1&q=85',
          'raw': 'https://images.unsplash.com/photo-1565153149760-49d028a7f657?i
xid=MnwyNjA3NjR8MHwxfHJhbmRvbXx8fHx8fHx8fDE2MzQ4NzU0NTE&ixlib=rb-1.2.1',
          'regular': 'https://images.unsplash.com/photo-1565153149760-49d028a7f6
57?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyNjA3NjR8MHwxfHJhbmRvbXx8fHx8
fHx8fDE2MzQ4NzU0NTE&ixlib=rb-1.2.1&q=80&w=1080',
          'small': 'https://images.unsplash.com/photo-1565153149760-49d028a7f657
?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyNjA3NjR8MHwxfHJhbmRvbXx8fHx8fH
x8fDE2MzQ4NzU0NTE&ixlib=rb-1.2.1&q=80&w=400',
          'thumb': 'https://images.unsplash.com/photo-1565153149760-49d028a7f657
?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyNjA3NjR8MHwxfHJhbmRvbXx8fHx8fH
x8fDE2MzQ4NzU0NTE&ixlib=rb-1.2.1&q=80&w=200'},
 'user': {'accepted_tos': True,
          'bio': 'Looking to explore more street photography. \r\n'
                 ' Please feel free to donate if you are using my photos for '
                 'your websites or other social media.',
          'first_name': 'OpticalNomad',
          'for_hire': False,
          'id': 'kt7ChDgBdYg',
          'instagram_username': 'Cqyao',
          'last_name': None,
          'links': {'followers':
'https://api.unsplash.com/users/opticalnomad/followers',
                    'following':
'https://api.unsplash.com/users/opticalnomad/following',
                    'html': 'https://unsplash.com/@opticalnomad',
```

```
                    'likes':
'https://api.unsplash.com/users/opticalnomad/likes',
                    'photos':
'https://api.unsplash.com/users/opticalnomad/photos',
                    'portfolio':
'https://api.unsplash.com/users/opticalnomad/portfolio',
                    'self': 'https://api.unsplash.com/users/opticalnomad'},
          'location': 'Singapore',
          'name': 'OpticalNomad',
          'portfolio_url': None,
          'profile_image': {'large': 'https://images.unsplash.com/profile-162055
7959546-5c8bb2aad73bimage?ixlib=rb-1.2.1&q=80&fm=jpg&crop=faces&cs=tinysrgb&fit=
crop&h=128&w=128',
                            'medium': 'https://images.unsplash.com/profile-16205
57959546-5c8bb2aad73bimage?ixlib=rb-1.2.1&q=80&fm=jpg&crop=faces&cs=tinysrgb&fit
=crop&h=64&w=64',
                            'small': 'https://images.unsplash.com/profile-162055
7959546-5c8bb2aad73bimage?ixlib=rb-1.2.1&q=80&fm=jpg&crop=faces&cs=tinysrgb&fit=
crop&h=32&w=32'},
          'social': {'instagram_username': 'Cqyao',
                     'paypal_email': None,
                     'portfolio_url': None,
                     'twitter_username': 'timotaitun'},
          'total_collections': 2,
          'total_likes': 58,
          'total_photos': 51,
          'twitter_username': 'timotaitun',
          'updated_at': '2021-10-21T08:05:55-04:00',
          'username': 'opticalnomad'},
 'views': 308455,
 'width': 4032}
```

[19]:
```
with open("../app/src/main/res/raw/images.json", "w") as f:
    json.dump(images, f)
```

0.2 Get portrait photos

[20]:
```
profile_pics = []
path = '/photos/random'

for i in range(30):
    result = requests.get(f"http://api.unsplash.com{path}/?
 ↪client_id={access_key}", params={'count':30, 'width':300, 'collections':␣
 ↪'772333'})
    if result.json():
        profile_pics.extend(result.json())
```

```
pprint(profile_pics[0])
```

```
---------------------------------------------------------------------------
JSONDecodeError                           Traceback (most recent call last)
/var/folders/gk/r228z2xd5y30x4nv6b1lsl6h0000gn/T/ipykernel_17261/399953287.py i␣
↪<module>
      4 for i in range(30):
      5     result = requests.get(f"http://api.unsplash.com{path}/?
↪client_id={access_key}", params={'count':30, 'width':300, 'collections':␣
↪'772333'})
----> 6     if result.json():
      7         profile_pics.extend(result.json())
      8

~/opt/anaconda3/envs/geo/lib/python3.8/site-packages/requests/models.py in␣
↪json(self, **kwargs)
    886             if encoding is not None:
    887                 try:
--> 888                     return complexjson.loads(
    889                         self.content.decode(encoding), **kwargs
    890                     )

~/opt/anaconda3/envs/geo/lib/python3.8/json/__init__.py in loads(s, cls,␣
↪object_hook, parse_float, parse_int, parse_constant, object_pairs_hook, **kw)
    355             parse_int is None and parse_float is None and
    356             parse_constant is None and object_pairs_hook is None and no␣
↪kw):
--> 357         return _default_decoder.decode(s)
    358     if cls is None:
    359         cls = JSONDecoder

~/opt/anaconda3/envs/geo/lib/python3.8/json/decoder.py in decode(self, s, _w)
    335
    336         """
--> 337         obj, end = self.raw_decode(s, idx=_w(s, 0).end())
    338         end = _w(s, end).end()
    339         if end != len(s):

~/opt/anaconda3/envs/geo/lib/python3.8/json/decoder.py in raw_decode(self, s,␣
↪idx)
    353             obj, end = self.scan_once(s, idx)
    354         except StopIteration as err:
--> 355             raise JSONDecodeError("Expecting value", s, err.value) from␣
↪None
    356         return obj, end
```

```
JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

[27]:
```
with open("../app/src/main/res/raw/profile_pics.json", "w") as f:
    json.dump(profile_pics, f)
```

[ ]:
```
with open("../app/src/main/res/raw/images.json") as f:
    images = json.load(f)
```

[22]:
```
with open("../app/src/main/res/raw/profile_pics.json") as f:
    profile_pics2 = json.load(f)
```

1 Generate data

[28]:
```
hashtags = """#runnerscommunity #marathontraining #l #cycling #nature #correr␣
↪#km #sports #strava #love #corrida #crossfit #runnerslife #nikerunning␣
↪#cardio #trailrun #runningman #trailrunner #exercise #instagood #lifestyle␣
↪#health #healthylifestyle #athlete #adidas #fitfam #o #halfmarathon␣
↪#instarunning #bieganie #running #run #runner #fitness #runningmotivation␣
↪#runnersofinstagram #instarunners #trailrunning #runners #training #workout␣
↪#sport #motivation #runhappy #k #marathon #garmin #instarun #fit␣
↪#instarunner #gym #triathlon #trail #fitnessmotivation #nike #laufen␣
↪#runningcommunity #runnersworld #runninggirl #bhfyp #runhappy #running #run␣
↪#runner #runnersofinstagram #instarunners #runningmotivation #runners␣
↪#instarun #instarunner #fitness #trailrunning #runnersworld␣
↪#runnerscommunity #runningcommunity #marathon #training #marathontraining␣
↪#garmin #runnerslife #happyrunner #runtoinspire #laufen #motivation␣
↪#runninggirl #k #workout #runrunrun #runnergirl #bhfyp"""
```

[29]:
```
hashtags = set(hashtags.split())
```

[30]:
```
def get_person(n=1):
    data = []
    for i in range(n):
        person = {}
        person['id'] = str(uuid.uuid4())
        person['name'] = fake.name()
        person['email'] = fake.email()
        person['userName'] = fake.user_name()

        person['isPublic'] = int(np.random.choice([1, 0], p=[0.9, 0.1]))
        person['profilePicUrl'] = random.choice(profile_pics)['urls']['thumb']

        data += person,
    return data
```

[31]:
```
users = get_person(1000)
```

[32]:
```
data = []
following = []
followers = []
blocked = []

for p in users:
    if len(following) > 0:
        p['following'] = following
    if len(following) >= 10:
        following.pop(0)
    following.append(p['id'])

    if len(followers) > 0:
        p['followers'] = followers
    if len(followers) >= 10:
        followers.pop(0)
    followers.append(p['id'])

    if len(data) > 10:
        p['blockedBy'] = [u['id'] for u in np.random.choice(data, size=np.
 ↪random.randint(0, 10))]

    if 'following' in p and 'blockedBy' in p:
        p['following'] = [uid for uid in p['following'] if uid not in␣
 ↪p['blockedBy']]
    data.append(p)

users = data
```

[33]:
```
pprint(users[-10])
```

```
{'blockedBy': ['402b90b6-c915-45e7-90e2-32fd8705a1a0',
               'fc95cc1a-456b-4c28-b77a-e67e5d50b31b',
               'e9c6dd3b-9fe9-4cb7-8ea1-38676a544aba',
               'f279519a-ff49-4247-9484-62a10bb75199',
               'e35d1c2f-bc49-4e47-8cf5-f1bfa9e0b9a7',
               'd2e8e847-0fa7-4e27-9fe2-ff5b821fca31',
               '19a5c3ca-2383-4f42-8541-0dff863a1782'],
 'email': 'amanda06@hotmail.com',
 'followers': ['ffe8fd3e-2218-4600-811c-890ffcd969e5',
               '692fda71-7db8-435b-9f03-e4b837c8b222',
               '59961b23-854a-4e9d-8816-b8e89a688f0f',
               'b4710a91-6274-4a36-89fa-145e60abf38b',
               'f4efb027-7d67-4d21-ae07-8886d8bc10ff',
               '17ea1052-5750-433b-a5fc-5d821e003e99',
               '6a552196-144f-405c-aaba-ed24eb0a3689',
               '7c9e0264-34eb-4dc2-9327-50ec006f3f02',
               '2f51211f-184c-43cf-ac98-bf9c3e5e23be',
```

```
                '2d8b0798-174f-4f3f-a42c-55c58cbd70ff'],
 'following': ['cc6fb94d-f76a-4092-8e0d-b9c4e6f63def',
               '4005d03b-cd75-4f09-80a0-842d95d366ae',
               'de075fd4-4209-44d7-8d25-141bf6e4a5e6',
               'ed4b64d9-a3eb-4f64-9956-de16383c1bfa',
               '88d834a7-9490-4b9a-9a4e-afa5bf7475b0',
               'a2c5c472-d7dd-4c6a-9b96-223d7420cbfa',
               '4075de0a-b09e-4484-b68b-366f87ffb661',
               '45852110-074d-4bc0-9bd3-838492636794',
               'e88d56ba-9c40-477f-abc5-936d29cc4f0d',
               'ffe8fd3e-2218-4600-811c-890ffcd969e5'],
 'id': 'ffe8fd3e-2218-4600-811c-890ffcd969e5',
 'isPublic': 1,
 'name': 'Gina Potter',
 'profilePicUrl': 'https://images.unsplash.com/photo-1531764117131-cbd26e7885f0?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyNjA3NjR8MHwxfHJhbmRvbXx8fHx8fHx
8fDE2MzQ4NzU0NzU&ixlib=rb-1.2.1&q=80&w=200',
 'userName': 'robert39'}
```

[34]:
```
with open("../app/src/main/res/raw/users.json", "w") as f:
    json.dump(users, f)
```

[35]:
```
def get_points(n=1):
    p = df.sample(n)
    data = []
    ts = None
    for longitude, latitude in zip(p.geometry.x, p.geometry.y):
        point = {}
        point['latitude'] = latitude
        point['longitude'] = longitude

        if not ts:
            point['ts'] = time.time() - random.uniform(3600 * 24, (3600 * 24) *␣
 ↪5)
            ts = point['ts']
        else:
            ts += random.uniform(55, 65)
            point['ts'] = ts

        data += point,
    return data
```

[36]:
```
def get_route(n=1):
    routes = []
    for i in range(n):
        route = {}
        route['points'] = get_points(random.randint(10, 20))
```

```
        for i in range(len(route['points'])):
            route['points'][i]['ts'] = route['points'][i]['ts']
        routes += route,
    return routes
```

[37]:
```
import geopy
from geopy import distance

# assuming data point recorded every 10 seconds, 60 data points represents a 10␣
 ↪miute run

def add_distance(lon, lat, bearing, dist):
    """
    lng: starting longitude
    lat: starting latitude
    bearing: azimuth, the direction of movement
    dist: distance in meters
    """
    pt = geopy.Point(lat, lon)
    d = distance.distance(meters=dist)
    next_point = d.destination(point=pt, bearing=bearing)
    return next_point.latitude, next_point.longitude

def get_route_athletic(n_points: int, stride: int, lon: float, lat: float):
    data = []
    next_point = {}
    bearing = random.randint(0, 359)

    if lon and lat:
        initial_position_poi = df.sample(1)
        next_point['longitude'] = initial_position_poi.geometry.x.values[0]
        next_point['latitude'] = initial_position_poi.geometry.y.values[0]
    else:
        next_point['longitude'] = lon
        next_point['latitude'] = lat

    for i in range(n_points):
        lat, lon = add_distance(
            next_point['longitude'],
            next_point['latitude'],
            bearing + random.uniform(-20, 20),
            stride + random.uniform(-10, 10)
        )
        next_point = {
            "longitude": lon,
            "latitude": lat
```

```
        }
        data += next_point,
    return data
```

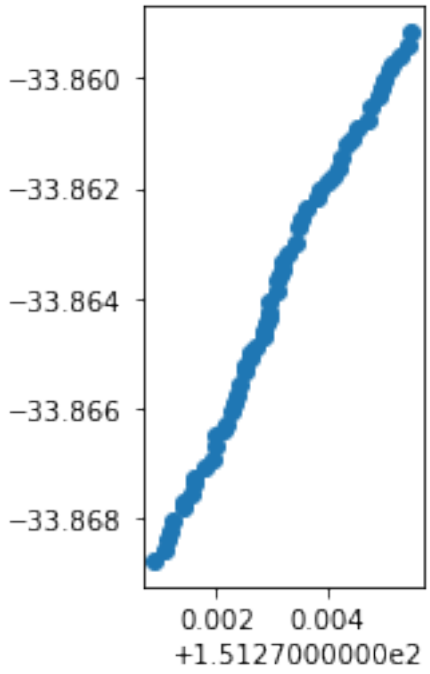
[38]:
```
def get_post(n=1):
    data = []
    for i in range(n):
        user = random.choice(users)

        post = {}
        post['id'] = str(uuid.uuid4())

        post['uid'] = user['id']
        post['userName'] = user['userName']
        post['isPublic'] = user['isPublic']
        post['profilePicUrl'] = user['profilePicUrl']

        img = random.choice(images)
        post['imageUrl'] = img['urls']['small']
        post['postDescription'] = img['description']

        post['hashtags'] = list(np.random.choice(list(hashtags), size=np.random.
 ↪randint(0, 10)))

        poi = random.choice(pois)
        post['latitude'] = poi['lat']
        post['longitude'] = poi['lon']
        post['locationName'] = poi['name']
        post['route'] = get_route_athletic(n_points=60, stride=20,␣
 ↪lon=post['longitude'], lat=post['latitude'])

        likedby = np.random.choice(users, np.random.randint(0, 20), False)
        post['likeCount'] = len(likedby)
        post['likedBy'] = []
        for u in likedby:
            post['likedBy'] += u['id'],

        data += post,
    return data
```

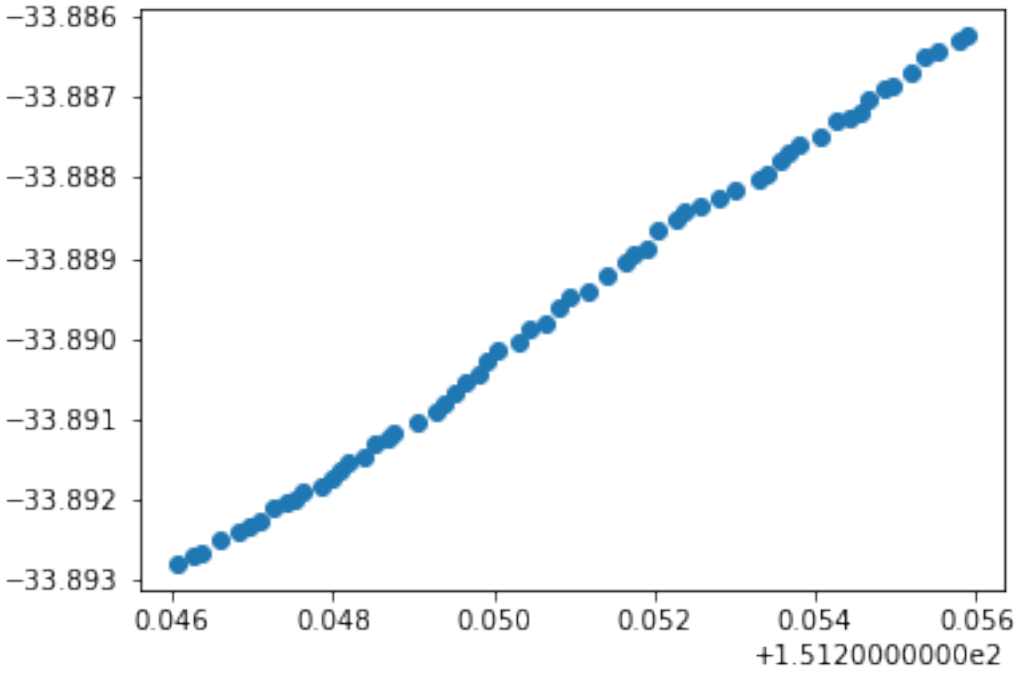
[39]:
```
posts = get_post(1000)
posts_demo = get_post(500)
```

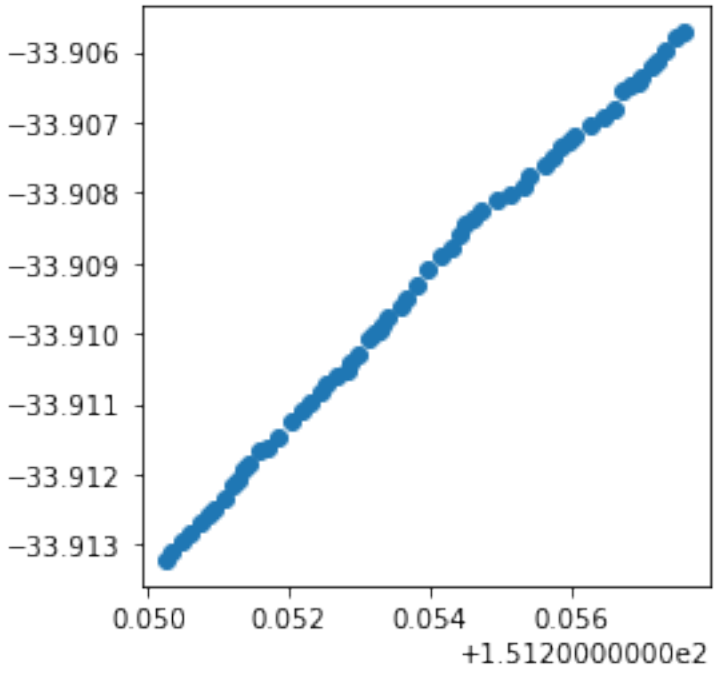
[40]:
```
def plot_line(post):
    data = pd.DataFrame(post['route'])
    data['geometry'] = [Point(x, y) for x, y in zip(data.longitude, data.
 ↪latitude)]
```

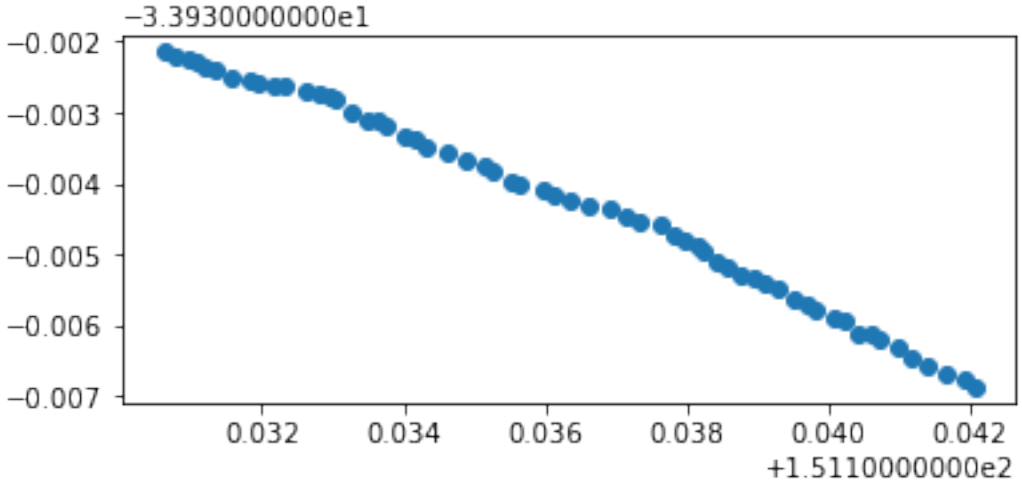
```
    data = gpd.GeoDataFrame(data, geometry='geometry')
    data.plot()
```

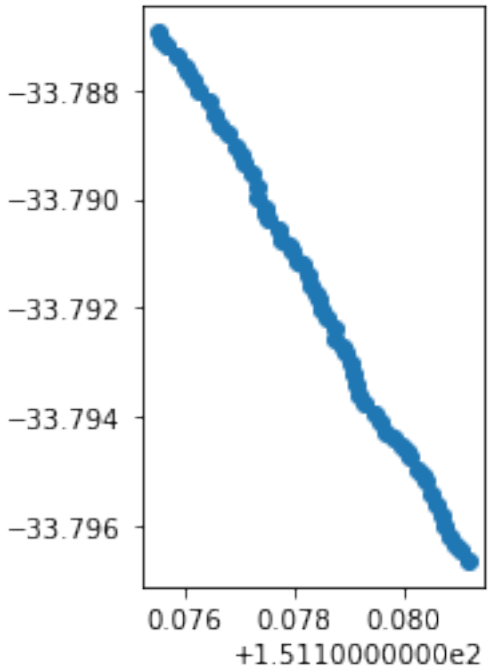
[41]:
```
for i in range(10):
    plot_line(random.choice(posts))
```

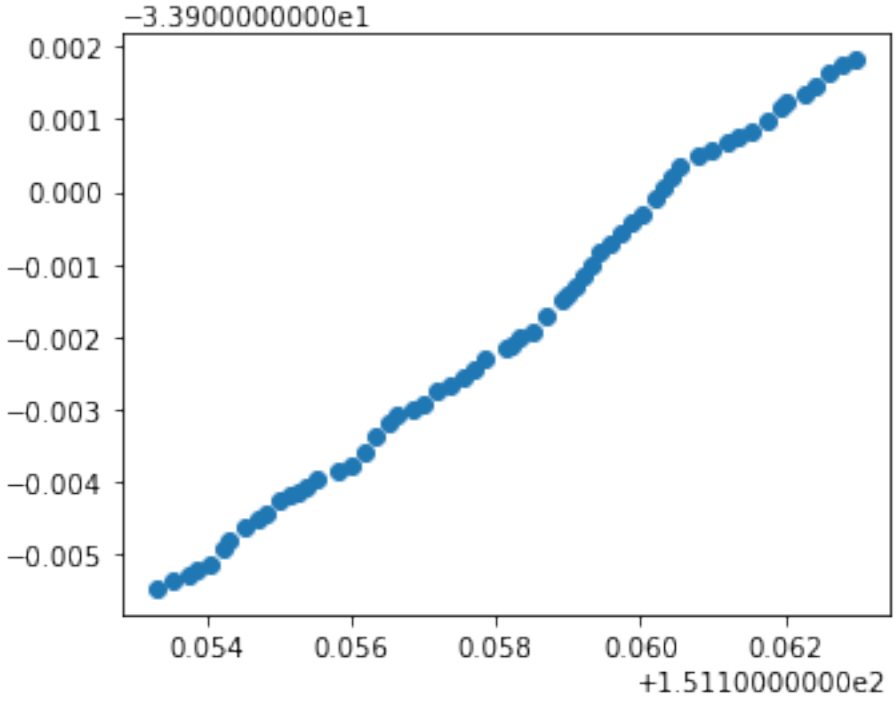


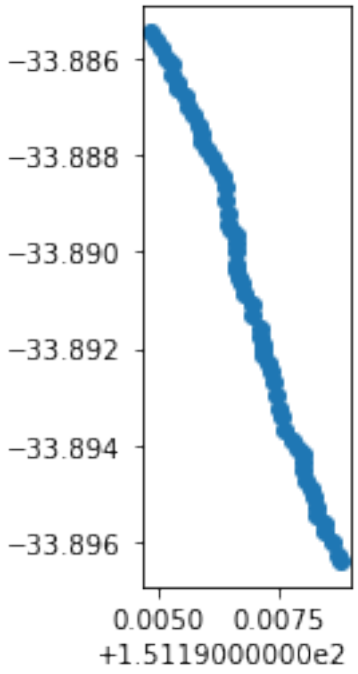


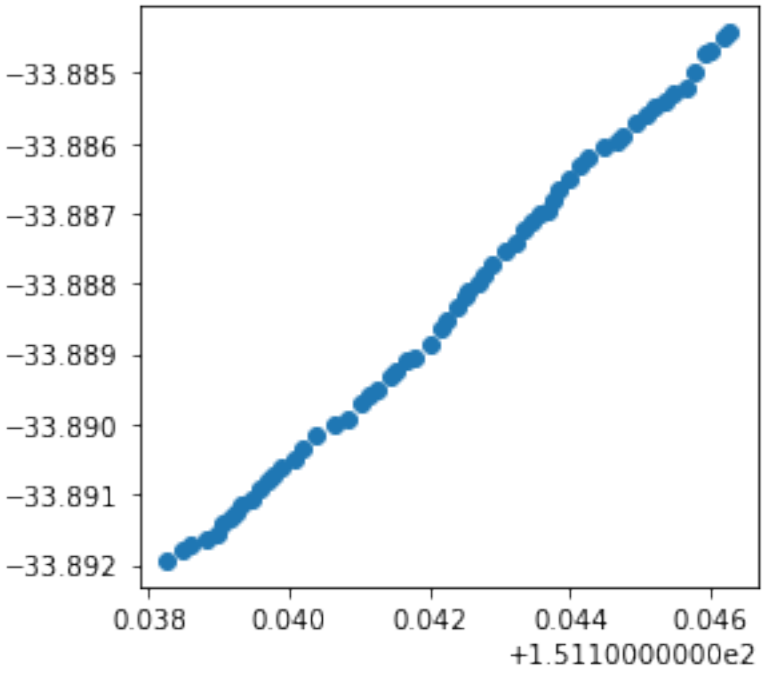

−33.906
−33.907
−33.908
−33.909
−33.910
−33.911
−33.912
−33.913
0.050
0.052
0.054
0.056
+1.5120000000e2

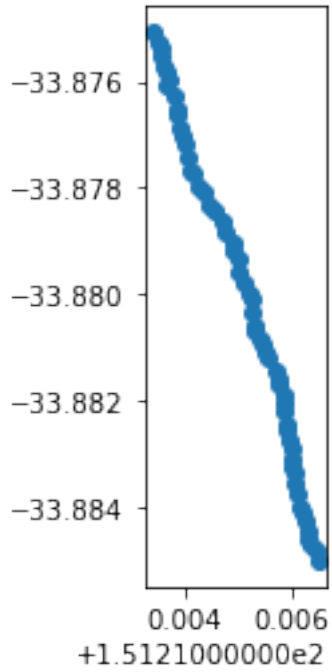

−3.3930000000e1
−0.002
−0.003
−0.004
−0.005
−0.006
−0.007
0.032
0.034
0.036
0.038
0.040
0.042
+1.5110000000e2

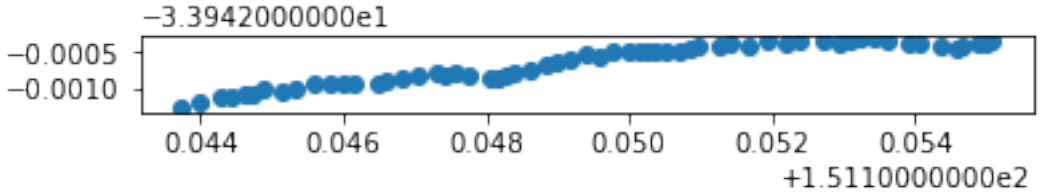

−33.788
−33.790
−33.792
−33.794
−33.796
0.076
0.078
0.080
+1.5110000000e2


−3.3900000000e1
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
−0.005
0.054
0.056
0.058
0.060
0.062
+1.5110000000e2


−33.886
−33.888
−33.890
−33.892
−33.894
−33.896
0.0050
0.0075
+1.5119000000e2


−33.885
−33.886
−33.887
−33.888
−33.889
−33.890
−33.891
−33.892
0.038
0.040
0.042
0.044
0.046
+1.5110000000e2





[42]:
```
with open("../app/src/main/res/raw/posts.json", "w") as f:
    json.dump(posts, f)

with open("../app/src/main/res/raw/posts_demo.json", "w") as f:
    json.dump(posts_demo, f)
```

2 Load data

[54]:
```
# import firebase_admin
# from firebase_admin import credentials
# from firebase_admin import firestore

# cred = credentials.
 ↪Certificate("softwareconstruction42-firebase-adminsdk-5s99m-f2a401fa7e.json")
# firebase_admin.initialize_app(cred, {
```

```
#   'projectId': 'softwareconstruction42',
# })
```

[55]:
```
# db = firestore.client()
```

[56]:
```
# users = []
# docs = db.collection(u'users').stream()

# for doc in docs:
#     print(f'{doc.id} => {doc.to_dict()}')
#     users += doc.to_dict(),
#     break
```

[57]:
```
# user = users[0]
```

[58]:
```
# user['followers']
```

[59]:
```
# ref = user['followers'][4]
```

[60]:
```
# user2 = ref.get().to_dict()
```

[61]:
```
# ref.id
```

[62]:
```
# ref.id.strip("\"")
```

[63]:
```
# user2 = db.collection(u'users').document(ref.id.strip("\""))
```

[64]:
```
# u2 = user2.get()
```

[65]:
```
# u2.exists
```

[66]:
```
# u2.to_dict()
```

[ ]:
```

```