Facility List Coder - Physical Activity, version 24/07/2018

The Facility List coder (FLC) is an open source tool that allow efficiently combine GIS analysis with standard data techniques. Besides the data management tools, FLC code retrieves GIS information on facilities location using two open source datasets: Google Maps and Open Street Maps.

FLC is built upon two main requirements.

- First, researchers will need to provide a specific data set for the specific location of the reference point (e.g. school, university, among others). We call them as location of interest (LI).
- Second in order to classify the results obtained from the spatial query on the traditional GIS engines, researchers will need to define a set of key words or metadata that allow the algorithm classify the facilities. Based on the literature, we have developed a multi-language key words list based on the European context that allows to classify each facility within a pre-defined category. These categories could be modified in order to fulfill specific needs of researchers related to geographical location, languages or research questions. This pre-defined key word offers an important innovation for the research on food community environment in the European context. The empirical studies for Europe often use categories created for United States, which might miss-estimate the particularities of European food traditions. Nonetheless, this list will be easily extended depending on the needs of the researchers. We called this dimension as key words by categories (KW).

Code Preliminaries

FLC code retrieves GIS information on facilities location using two open source datasets: Google Maps and Open Street Maps. All the code was developed on Python 3.6.5.

Defining functions

Now, we need to create functions that will be use during the process

In [27]:

```
### Function to save results
##
def save_result(res,prints):
    for place in res.places:
        place.get_details()
        if prints==1:
            print(place.name.encode('utf-8'),place.types,place.place_id)
        else:
            if place.place_id not in places_flc:

raw={'place_name':place.name,'place_lat':float(place.geo_location['lat']),'place_lng':float(place.geo_location['lng']),'place_address':place.formatted_address,'google_id':place.place_id,'place_types':place.types,'place_web':place.url}
                places_flc[place.place_id]=raw


## Read
## This functions was made to test the results
def read_result(res):
    for place in res.places:
        place.get_details()
        print(place.name.encode('utf-8'),place.types,place.place_id)


### Save Results
## Save Results
def sav_final(place,cat,final_ds):
    ## Save Final Data Set
    raw=places_flc_cleaned[place]
    raw['categ']=cat
    final_ds.append(raw)
    ## Delete the dictionary
```

Loading packages

Here we need to load the needed packages

In [28]:

```
# coding=utf-8
###########################
###### Libraries and Initial SetUp
###########################

### My Data key
YOUR_API_KEY = 'AIzaSyDfJ-C6nJJaEIiwsWuh3JMmD6rcXijwKqA'

## --- Google maps
import googlemaps
gmaps = googlemaps.Client(key=YOUR_API_KEY)
from datetime import datetime

# ----- Google Place
from googleplaces import GooglePlaces, types, lang
import google_streetview.api
google_places = GooglePlaces(YOUR_API_KEY)

# ----- Download the data
import urllib.request # Download the files
import os, sys ## create a new
import pandas as pd
from xlsxwriter.utility import xl_rowcol_to_cell

# ------ from openpyxl import load_workbook
from openpyxl import load_workbook
import fiona

# ------ Import Regex
import re

# ----- Load information
import json
```

Input 1: Key Words

In this section we load the key words that will be used for the classification of the results.

Key words for the Catalan context

The first part of the key words are a detailed description of the economic activities in a specific region. This document brings a details description of the different types of establishments present in the region.

All the keywords were translated in English, Spanish and Catalan. Everything is saved at: **Palabras_clave_v0_120917.xlsx**

In [29]:

```
data_keywords="/home/datascience/Documents/FLC_Andreu/01_DataBase_Georeferencion/Palabras_clave_v0_
917.xlsx"
wb = load_workbook(filename = data_keywords)
palabras = wb['Palabras_Claves_Google2']

### Create the dictionary
categ_est={}
keywords=[]
for row in range(2,32):
    cat=str(palabras.cell(row=row, column=1).value.encode('utf-8')).lower().strip()
    try:
        categ_est[cat]=[]
    except:
        pass;

#### Now the key words
for row in range(2,32):
```

```
for row in range(2,32):
    cat=str(palabras.cell(row=row, column=1).value.encode('utf-8')).lower().strip()
    for col in [2,3,4]:
        if palabras.cell(row=row, column=col).value is not None:
            word=str(palabras.cell(row=row, column=col).value.encode('utf-8')).lower().strip()
            if word not in categ_est[cat]:
                categ_est[cat].append(str(word))
```

Type Excluded

In order to optimize the query over the google maps we restrict the search to particular set of places types (here the list of accepted types in Google).

In order to exclude/include a given category you only have to change: **google_types.xlsx**

In [30]:

```
data_types="/home/datascience/Documents/FLC_Andreu/01_DataBase_Georeferencion/google_types.xlsx"
wb = load_workbook(filename = data_types)
palabras = wb['Sheet1']
types_accepted=[]
types_excluded=[]
for row in range(2,98):
    selected=int(palabras.cell(row=row, column=2).value)
    if selected==0:
        types_accepted.append(str(palabras.cell(row=row, column=1).value))
    if selected==1:
        types_excluded.append(str(palabras.cell(row=row, column=1).value))

print (types_accepted)
```

```
['amusement_park', 'bowling_alley', 'campground', 'gym', 'park']
```

Input 2: Defining Location of interest (LI)

In this section we load the location of the point of interest. For the validation procedure, we include the Grids.

Note that you need a shapefile with the centroids. It can be generalized even more using any type of argument.

In [31]:

```
grids="/home/datascience/Documents/FLC_Andreu/01_DataBase_Georeferencion/Colegios_shp/Colegios.shp'

grids_data={}
with fiona.open(grids,'r') as shp:
    for feat in shp:
        grid_id = feat['properties']['1']
        lng,lat=feat['geometry']['coordinates']
        grids_data[grid_id]={'lat':lat,'lng':lng}
```

Facility List Coder in action: Building the dataset

In this section we run the spatial query in Google maps based on the point of interest. Technically, the FLC will get all the location within an specific folder, then we will classify then using the key words.

The first step is defining the data set.

In [2]:

```
# Data Information
places_flc={}
```

Spatial Query Strategy 1: Using only Google types

This strategy will get all the places within a buffer that belongs to a specific type (look above to check the list of types)

Spatial Query Strategy 2: Using only Google types

In [54]:

```
i=1

### To Print results
prints=0

#for grid in [755]:
for grid in grids_data:
    print("%d of 301 (grid=%d)" %(i,grid))
    i+=1
    ### First Step Search
    for typ in types_accepted:
   #for typ in ['restaurant']:
        ### First 20 results
        res1=google_places.nearby_search(lat_lng=grids_data[grid],radius=100,types=[typ])
        save_result(res1,prints)
        # For more than 20 results
        try:
            res2 = google_places.nearby_search(pagetoken=res1.next_page_token)
            save_result(res2,prints)
            try:
                res3 = google_places.nearby_search(pagetoken=res2.next_page_token)
                save_result(res3,prints)
                try:
                    res4 = google_places.nearby_search(pagetoken=res3.next_page_token)
                    save_result(res4,prints)
                except:
                    pass
            except:
                pass
        except:
            pass
```

```
1 of 301 (grid=2)
2 of 301 (grid=4)
3 of 301 (grid=5)
4 of 301 (grid=6)
5 of 301 (grid=7)
6 of 301 (grid=8)
7 of 301 (grid=9)
8 of 301 (grid=10)
9 of 301 (grid=11)
10 of 301 (grid=12)
11 of 301 (grid=13)
12 of 301 (grid=14)
13 of 301 (grid=15)
14 of 301 (grid=16)
15 of 301 (grid=17)
16 of 301 (grid=18)
17 of 301 (grid=19)
18 of 301 (grid=20)
19 of 301 (grid=21)
```

This data is complementary to the last one, but it will search by keywords instead of type. This part will search at any information gathered by google.

In [55]:

```
#types=types_accepted+['supermarket']

### To Print results
prints=0

i=1
for grid in grids_data:
    print("%d of 301 (grid=%d)" %(i,grid))
    i+=1
#for grid in [1020]:
    ### First Step Search
    for typ in types_accepted:
        #print typ
        ### First 20 results
        res1=google_places.nearby_search(lat_lng=grids_data[grid],radius=100,keyword=typ)
        # For more than 20 results
        save_result(res1,prints)
        try:
            res2 = google_places.nearby_search(pagetoken=res1.next_page_token)
            save_result(res2,prints)
            try:
                res3 = google_places.nearby_search(pagetoken=res2.next_page_token)
                save_result(res3,prints)
                try:
                    res4 = google_places.nearby_search(pagetoken=res3.next_page_token)
                    save_result(res4,prints)
                except:
                    pass
            except:
                pass
        except:
            pass
```

```
1 of 301 (grid=2)
2 of 301 (grid=4)
3 of 301 (grid=5)
4 of 301 (grid=6)
5 of 301 (grid=7)
6 of 301 (grid=8)
7 of 301 (grid=9)
8 of 301 (grid=10)
9 of 301 (grid=11)
10 of 301 (grid=12)
11 of 301 (grid=13)
12 of 301 (grid=14)
13 of 301 (grid=15)
14 of 301 (grid=16)
15 of 301 (grid=17)
16 of 301 (grid=18)
17 of 301 (grid=19)
18 of 301 (grid=20)
19 of 301 (grid=21)
```

In [56]:

```
print(len(places_flc))
```

```
50
```

Facility List Coder in action: Classifying the places

Now, once all the places are gathered using the two strategies, we now need to clean them.

Clean Data Set

We will delete those establishments that will be excluded.

In [3]:

```
### First data
places_flc_cleaned={}

# Excluded Types
excluded_types=['beauty_salon','finance','pharmacy','electrician','church','parking']
# Excluded Names (part of)
excluded_names=['parking']

#### First Step ---> Those in the excluded categories
for i in places_flc:
    ### Check for those with a excluded
    ### Check for those with a excluded
    name=places_flc[i]['place_name'].lower().encode('utf-8').strip().split()
    if any(x in places_flc[i]['place_types'] for x in types_excluded):
        pass
    if any(x in name for x in excluded_names):
        pass
    else:
        places_flc_cleaned[i]=places_flc[i]
```

Save RawData

Finally, we save the entire data set.

In [5]:

```
import json
### Save
json = json.dumps(places_flc_cleaned)
f = open("/home/datascience/Documents/FLC_Andreu/Results/flc_rawresults_grids_13022018.json","w")
f.write(json)
f.close()
```