# 04_Spatial_autocorrelation

March 29, 2020

## 1 Spatial autocorrelation

Computational notebook 04 for **Morphological tessellation as a way of partitioning space: Improving consistency in urban morphology at the plot scale**.

Fleischmann, M., Feliciotti, A., Romice, O. and Porta, S. (2020) *'Morphological tessellation as a way of partitioning space: Improving consistency in urban morphology at the plot scale'*, Computers, Environment and Urban Systems, 80, p. 101441. doi: 10.1016/j.compenvurbsys.2019.101441.

Contact: martin@martinfleischmann.net

Date: 29/03/2020

Note: notebook has been cleaned and released retroactively. It is likely that different versions of packages were initially used, but we made sure that the results remained unaltered.

**Data**

The source of the data used wihtin the research is the Amtliche Vermessung dataset accessible from the Zurich municipal GIS open data portal (https://maps.zh.ch). From it can be extracted the cadastral layer (`Liegenschaften_Liegenschaft_Area`) and the layer of buildings (all features named `Gebäude`). All data are licensed under CC-BY 4.0.

Source data: Vektor-Übersichtsplan des Kantons Zürich, 13.03.2018, Amt für Raumentwicklung Geoinformation / GIS-Produkte, Kanton Zürich, https://opendata.swiss/de/dataset/vektor-ubersichtsplan1

Data structure:

```
data/
    single_uids.csv - IDs of buildings being alone on a single plot (QGIS generated)

    cadastre/
        blg_cadvals.shp - Cadastral values spatially joined to buildings to allow 1:1 comparis

    tessellation/
        {k}_tessellation.shp - tessellation layers
```

[3]:
```
import matplotlib.pyplot as plt
import geopandas as gpd
import esda
import libpysal
```

```
from splot.esda import plot_local_autocorrelation
import multiprocessing
import pandas as pd
```

[4]:
```
esda.__version__, gpd.__version__, libpysal.__version__, pd.__version__
```

[4]:
```
('2.2.1', '0.7.0', '4.2.2', '1.0.3')
```

[ ]:
```
blg = gpd.read_file('data/cadastre/blg_cadvals.shp')

characters = ['area', 'lal', 'circom', 'shapeix', 'rectan', 'fractal',
              'orient', 'freq', 'car', 'gini_area', 'gini_car', 'Reach']

singleuids = pd.read_csv('data/single_uids.csv')
singles = singleuids['2'].to_list()

def worker(k):
    print(k)
    file = gpd.read_file('data/tessellation/{k}_tessellation.shp'.format(k=k))
    file_s = file.loc[file['uID'].isin(singles)].copy()
    file_m = file.loc[~file['uID'].isin(singles)].copy()
    for ch in characters:
        print(ch)
        try:
            local_moran = esda.Moran_Local(file[[ch]], weights)
            file['m_{}'.format(ch)] = local_moran.q
            file['p_{}'.format(ch)] = local_moran.p_sim
            single_moran = esda.Moran_Local(file_s[[ch]], weights_s)
            file.loc[file['uID'].isin(singles), 'ms_{}'.format(ch)] =␣
 ↪single_moran.q
            file.loc[file['uID'].isin(singles), 'ps_{}'.format(ch)] =␣
 ↪single_moran.p_sim
            multi_moran = esda.Moran_Local(file_m[[ch]], weights_m)
            file.loc[~file['uID'].isin(singles), 'mm_{}'.format(ch)] =␣
 ↪multi_moran.q
            file.loc[~file['uID'].isin(singles), 'pm_{}'.format(ch)] =␣
 ↪multi_moran.p_sim
        # plot_local_autocorrelation(local_moran, file, ch)
        # plt.savefig('files/moran/{k}_{ch}.png'.format(k=k, ch=ch))
        # plt.gcf().clear()
        except Exception:
            print('missing, skipped')
    file.to_file('data/tessellation/{k}_tessellation.shp'.format(k=k))
    print('saved')

weights = libpysal.weights.DistanceBand.from_dataframe(blg, 200)
single = blg.loc[blg['uID_left'].isin(singles)].copy()
```

```
weights_s = libpysal.weights.DistanceBand.from_dataframe(single, 200)
multi = blg.loc[~blg['uID_left'].isin(singles)].copy()
weights_m = libpysal.weights.DistanceBand.from_dataframe(multi, 200)

for ch in characters:
    print(ch)
    local_moran = esda.Moran_Local(blg[[ch]], weights)
    blg['m_{}'.format(ch)] = local_moran.q
    blg['p_{}'.format(ch)] = local_moran.p_sim
    single_moran = esda.Moran_Local(single[[ch]], weights_s)
    blg.loc[blg['uID_left'].isin(singles), 'ms_{}'.format(ch)] = single_moran.q
    blg.loc[blg['uID_left'].isin(singles), 'ps_{}'.format(ch)] = single_moran.
 ↪p_sim
    multi_moran = esda.Moran_Local(multi[[ch]], weights_m)
    blg.loc[~blg['uID_left'].isin(singles), 'mm_{}'.format(ch)] = multi_moran.q
    blg.loc[~blg['uID_left'].isin(singles), 'pm_{}'.format(ch)] = multi_moran.
 ↪p_sim
blg.to_file('data/cadastre/blg_cadvals.shp')
```

[ ]: