Chapter 7 - Measure contextual - spatially lagged characters

November 10, 2020

[ ]:
```
import geopandas as gpd
import momepy as mm
from tqdm import tqdm
from momepy import limit_range
import numpy as np
import pandas as pd
from inequality.theil import Theil
import libpysal
import scipy as sp
import mapclassify
```

Load data from the previous notebook:

[ ]:
```
gdf = pd.read_parquet('files/primary.parquet')

spatial_weights = libpysal.io.open('files/AMSqueen3.gal', 'r').read()
spatial_weights.neighbors = {int(k): [int(i) for i in v] for k, v in␣
 ↪spatial_weights.neighbors.items()}  # we need values as integers again
```

Measure contextual characters:

[ ]:
```
means = {}
ranges = {}
theils = {}

for ch in characters:
    means[ch] = []
    ranges[ch] = []
    theils[ch] = []
```

[ ]:
```
unique_id = 'uID'
```

[ ]:
```
gdf = gdf.fillna(0)  # normally does not happen, but to be sure
chars = gdf.columns
```

[ ]:
```
gdf['lcdMes'] = gdf.apply(
            lambda row: row.lcdMes if row.lcdMes >= 0 else 0,
            axis=1,
```

```
        )  # normally does not happen, but to be sure
```

[ ]:
```
def theil(y):
    y = np.array(y)
    n = len(y)
    plus = y + np.finfo('float').tiny * (y == 0)  # can't have 0 values
    yt = plus.sum(axis=0)
    s = plus / (yt * 1.0)
    lns = np.log(n * s)
    slns = s * lns
    t = sum(slns)
    return t


for index, row in tqdm(gdf.iterrows(), total=gdf.shape[0]):
    neighbours = spatial_weights.neighbors[index].copy()
    neighbours.append(index)

    for ch in characters:
        values_list = gdf.loc[neighbours][ch]
        idec = limit_range(values_list.tolist(), rng=(10, 90))
        iquar = limit_range(values_list.tolist(), rng=(25, 75))

        means[ch].append(np.mean(iquar))
        ranges[ch].append(sp.stats.iqr(values_list, rng=(25, 75)))
        theils[ch].append(theil(idec))
```

[ ]:
```
for ch in characters:
    gdf[ch + '_meanIQ3'] = means[ch]
    gdf[ch + '_rangeIQ3'] = ranges[ch]
    gdf[ch + '_theilID3'] = theils[ch]
```

[ ]:
```
skewness = pd.DataFrame(index=chars)
for c in chars:
    skewness.loc[c, 'skewness'] =sp.stats.skew(gdf[c])
headtail = list(skewness.loc[skewness.skewness >= 1].index)
to_invert = skewness.loc[skewness.skewness <= -1].index

for inv in to_invert:
    gdf[inv + '_r'] = gdf[inv].max() - gdf[inv]
inverted = [x for x in gdf.columns if '_r' in x]
headtail = headtail + inverted
natural = [x for x in chars if x not in headtail]
```

[ ]:
```
def _simpson_di(data):

    def p(n, N):
```

```
        if n == 0:
            return 0
        return float(n) / N

    N = sum(data.values())

    return sum(p(n, N) ** 2 for n in data.values() if n != 0)
```

[ ]:
```
import mapclassify.classifiers as classifiers
schemes = {}
for classifier in classifiers.CLASSIFIERS:
    schemes[classifier.lower()] = getattr(classifiers, classifier)
```

[ ]:
```
results = {}
for c in headtail + natural:
    results[c] = []
bins = {}
for c in headtail:
    bins[c] = schemes['headtailbreaks'](gdf[c]).bins
for c in natural:
    bins[c] = mapclassify.gadf(gdf[c], method='NaturalBreaks')[1].bins
```

[ ]:
```
for index, row in tqdm(gdf.iterrows(), total=gdf.shape[0]):
    neighbours = spatial_weights.neighbors[index].copy()
    neighbours.append(index)

    subset = gdf.loc[neighbours]
    for c in headtail + natural:
        values = subset[c]
        sample_bins = classifiers.UserDefined(values, list(bins[c]))
        counts = dict(zip(bins[c], sample_bins.counts))
        results[c].append(_simpson_di(counts))
```

[ ]:
```
for c in headtail + natural:
    gdf[c + '_simpson'] = results[c]
```

[ ]:
```
gdf.rename(columns={'sscERI_r_simpson': 'sscERI_simpson',
 'ssbERI_r_simpson': 'ssbERI_simpson',}, inplace=True)
```

[ ]:
```
pat = [x for x in gdf.columns if '_' in x]
```

[ ]:
```
gdf[pat].to_parquet('files/contextual.parquet')
```