Urban Form Exploration

In [ ]:
```
#Import all packages
```

In [1]:
```
import pandas as pd
import numpy as np
import datetime as dt
import seaborn as sns
import matplotlib
import os
import matplotlib.pyplot as plt
import matplotlib.dates as md
import pandas as pd
import matplotlib.pyplot as plt #if using matplotlib
import plotly.express as px #if using plotly
import geopandas as gpd
import warnings

import momepy

import warnings

import geopandas
import libpysal
import momepy
import osmnx
import pandas
```

In [2]:
```
# set the filepath and load in a shapefile
fp = r'MSOA.geojson'
map_df = gpd.read_file(fp)
map_df.head()
```

Out[2]:

	OBJECTID	MSOA11CD	MSOA11NM	BNG_E	BNG_N	LONG_	LAT	Shape_Leng	Shape__Area	Shape__Length	geometry
0	1	E02000001	City of London 001	532384	181355	-0.093490	51.5156	9651.221144	3.151475e+06	9651.221144	POLYGON ((-0.09650 51.52295, -0.09644 51.52282...
1	2	E02000002	Barking and Dagenham 001	548267	189685	0.138756	51.5865	8306.888230	2.161561e+06	8306.888230	POLYGON ((0.14810 51.59656, 0.14809 51.59640, ...
2	3	E02000003	Barking and Dagenham 002	548259	188520	0.138149	51.5760	9359.512342	2.141515e+06	9359.512342	POLYGON ((0.14847 51.58083, 0.14841 51.58075, ...
3	4	E02000004	Barking and Dagenham 003	551004	186412	0.176828	51.5564	8475.840309	2.492946e+06	8475.840309	POLYGON ((0.18511 51.56480, 0.18479 51.56455, ...
4	5	E02000005	Barking and Dagenham 004	548733	186824	0.144267	51.5607	7321.657104	1.187954e+06	7321.657104	POLYGON ((0.15187 51.56778, 0.15193 51.56775, ...

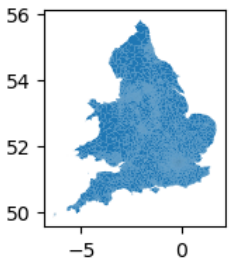
In [3]:
```
eng_df = map_df[map_df['MSOA11CD'].str[0]=='E']
```

In [4]:
```
len(eng_df)
```

Out[4]:
```
6791
```

In [5]:
```
map_df.plot(figsize=(2, 2),linewidth=0.01)
```

Out[5]:
```
<AxesSubplot: >
```



In [6]:
```
eng_df.crs
```

Out[6]:
```
<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
```

In [7]:
```
eng_df.to_crs('EPSG:27700')
```

Out[7]:

	OBJECTID	MSOA11CD	MSOA11NM	BNG_E	BNG_N	LONG_	LAT	Shape_Leng	Shape__Area	Shape__Length	geometry
0	1	E02000001	City of London 001	532384	181355	-0.093490	51.5156	9651.221144	3.151475e+06	9651.221144	POLYGON ((532153.715 182165.158, 532158.262 18...
1	2	E02000002	Barking and Dagenham 001	548267	189685	0.138756	51.5865	8306.888230	2.161561e+06	8306.888230	POLYGON ((548881.317 190819.983, 548881.138 19...
2	3	E02000003	Barking and Dagenham 002	548259	188520	0.138149	51.5760	9359.512342	2.141515e+06	9359.512342	POLYGON ((548958.570 189072.181, 548954.531 18...
3	4	E02000004	Barking and Dagenham 003	551004	186412	0.176828	51.5564	8475.840309	2.492946e+06	8475.840309	POLYGON ((551550.071 187364.710, 551528.648 18...
4	5	E02000005	Barking and Dagenham 004	548733	186824	0.144267	51.5607	7321.657104	1.187954e+06	7321.657104	POLYGON ((549237.064 187627.944, 549241.332 18...
...	...	...	...	...	...	...	...	...	...	...	...
6819	6820	E02006930	Greenwich 037	540126	178333	0.016812	51.4866	8103.297524	8.674410e+05	8103.297524	MULTIPOLYGON (((540659.012 178728.004, 540601....
6820	6821	E02006931	Greenwich 038	538710	177146	-0.004030	51.4763	9351.043860	2.049169e+06	9351.043860	POLYGON ((538710.355 177960.764, 538716.506 17...
6821	6822	E02006932	Liverpool 060	334899	390314	-2.980710	53.4056	12791.918754	1.477077e+06	12791.918754	POLYGON ((335723.013 391178.004, 335737.225 39...
6822	6823	E02006933	Liverpool 061	334647	389248	-2.984270	53.3960	10815.142852	2.824636e+06	10815.142852	POLYGON ((335092.922 390247.027, 335096.012 39...
6823	6824	E02006934	Liverpool 062	334077	390779	-2.993170	53.4097	8241.211993	2.315879e+06	8241.211993	POLYGON ((334674.989 391540.461, 334679.738 39...

6791 rows × 11 columns

In [8]:
```
eng_df['geometry']
```

Out[8]:
```
0       POLYGON ((-0.09650 51.52295, -0.09644 51.52282...
1       POLYGON ((0.14810 51.59656, 0.14809 51.59640, ...
2       POLYGON ((0.14847 51.58083, 0.14841 51.58075, ...
3       POLYGON ((0.18511 51.56480, 0.18479 51.56455, ...
4       POLYGON ((0.15187 51.56778, 0.15193 51.56775, ...
                              ...
6819    MULTIPOLYGON (((0.02464 51.49001, 0.02381 51.4...
6820    POLYGON ((-0.00371 51.48360, -0.00363 51.48350...
6821    POLYGON ((-2.96849 53.41349, -2.96827 53.41305...
6822    POLYGON ((-2.97778 53.40505, -2.97773 53.40505...
6823    POLYGON ((-2.98433 53.41662, -2.98426 53.41651...
Name: geometry, Length: 6791, dtype: geometry
```

In [9]:
```
eng_df[eng_df['MSOA11CD']=='E02000001']
```

Out[9]:

	OBJECTID	MSOA11CD	MSOA11NM	BNG_E	BNG_N	LONG_	LAT	Shape_Leng	Shape__Area	Shape__Length	geometry
0	1	E02000001	City of London 001	532384	181355	-0.09349	51.5156	9651.221144	3.151475e+06	9651.221144	POLYGON ((-0.09650 51.52295, -0.09644 51.52282...

In [10]:
```
df = pd.DataFrame()
x=0 #index 0 for MSOA, which is city of London E02000001 to explore in Urban Form
```

In [11]:
```
#sample to generate some of the urban form features, this is for 1 feature but then used in a loop later

msoa = eng_df.iloc[x].geometry #retrieve data for index 0 E0200001
msoa_name = eng_df.iloc[x].MSOA11CD
buildings = osmnx.geometries.geometries_from_polygon(msoa, tags={'building':True}) #retrieve data from polygon geometry
buildings = buildings[buildings.geom_type == "Polygon"].reset_index(drop=True)
buildings = buildings[["geometry"]].to_crs('EPSG:27700') #set geometry to British Coordinate System
buildings["uID"] = range(len(buildings)) #Lenght of buildings in
buildings['MSOA11CD'] = msoa_name #tag as MSOA

osm_graph = osmnx.graph_from_polygon(msoa, network_type='drive') #retrieve drive, street data
osm_graph = osmnx.projection.project_graph(osm_graph, to_crs='EPSG:27700') #change to british coordinate
streets = osmnx.graph_to_gdfs(osm_graph,nodes=False,edges=True,
                              node_geometry=False,fill_edge_geometry=True)
streets = momepy.remove_false_nodes(streets)
streets = streets[["geometry"]]
streets["nID"] = range(len(streets)) #street data range

limit = momepy.buffered_limit(buildings, 100)
tessellation = momepy.Tessellation(buildings, "uID", limit, verbose=False, segment=1) #generate tessellation
tessellation = tessellation.tessellation
buildings = buildings.sjoin_nearest(streets, max_distance=1000, how="left") #using building street data
buildings = buildings.drop_duplicates("uID").drop(columns="index_right")
tessellation = tessellation.merge(buildings[['uID', 'nID']], on='uID', how='left')
buildings["area"] = buildings.area
tessellation["area"] = tessellation.area
streets["length"] = streets.length

buildings['eri'] = momepy.EquivalentRectangularIndex(buildings).series #generate urban form features
buildings['elongation'] = momepy.Elongation(buildings).series
tessellation['convexity'] = momepy.Convexity(tessellation).series
streets["linearity"] = momepy.Linearity(streets).series
```

```
buildings["shared_walls"] = momepy.SharedWallsRatio(buildings).series
queen_1 = libpysal.weights.contiguity.Queen.from_dataframe(tessellation, ids="uID", silence_warnings=True)
tessellation["neighbors"] = momepy.Neighbors(tessellation, queen_1, "uID", weighted=True, verbose=False).series
tessellation["covered_area"] = momepy.CoveredArea(tessellation, queen_1, "uID", verbose=False).series

with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    buildings["neighbor_distance"] = momepy.NeighborDistance(buildings, queen_1, "uID", verbose=False).series
    queen_3 = momepy.sw_high(k=3, weights=queen_1)
    buildings_q1 = libpysal.weights.contiguity.Queen.from_dataframe(buildings, silence_warnings=True)
    buildings['interbuilding_distance'] = momepy.MeanInterbuildingDistance(buildings, queen_1, 'uID', queen_3, verbose=False).series
    buildings['adjacency'] = momepy.BuildingAdjacency(buildings, queen_3, 'uID', buildings_q1, verbose=False).series
    profile = momepy.StreetProfile(streets, buildings)
    streets["width"] = profile.w
    streets["width_deviation"] = profile.wd
    streets["openness"] = profile.o
    tessellation['car'] = momepy.AreaRatio(tessellation, buildings, 'area', 'area', 'uID').series
    graph = momepy.gdf_to_nx(streets)
    graph = momepy.node_degree(graph)
    graph = momepy.closeness_centrality(graph, radius=400, distance="mm_len")
    graph = momepy.meshedness(graph, radius=400, distance="mm_len")
    nodes, streets = momepy.nx_to_gdf(graph)
    buildings["nodeID"] = momepy.get_node_id(buildings, nodes, streets, "nodeID", "nID")
    merged = tessellation.merge(buildings.drop(columns=['nID', 'geometry']), on='uID')
    merged = merged.merge(streets.drop(columns='geometry'), on='nID', how='left')
    merged = merged.merge(nodes.drop(columns='geometry'), on='nodeID', how='left')

    temporary_df=pd.DataFrame(merged.median()).transpose()
    temporary_df['MSOA11CD'] = msoa_name
    temporary_df = temporary_df[['MSOA11CD','area_x', 'convexity', 'neighbors', 'covered_area', 'car', 'area_y',
                                 'eri', 'elongation', 'shared_walls', 'neighbor_distance',
                                 'interbuilding_distance', 'adjacency', 'length', 'linearity', 'width',
                                 'width_deviation', 'openness', 'degree', 'closeness', 'meshedness']]
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\geopandas\array.py:1406: UserWarning: CRS not set for some of the concatenation inputs. Setting output's CRS as OSGB36 / British National Grid (the single non-null crs provided).
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\momepy\elements.py:278: FutureWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To retain the old behavior, use either `df[df.columns[i]] = newvals` or, if columns are non-unique, `df.isetitem(i, newvals)`
  objects.loc[mask, objects.geometry.name] = objects[mask].buffer(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\geopandas\tools\clip.py:67: FutureWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To retain the old behavior, use either `df[df.columns[i]] = newvals` or, if columns are non-unique, `df.isetitem(i, newvals)`
  clipped.loc[
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\momepy\elements.py:384: UserWarning: Tessellation does not fully match buildings. 1 element(s) collapsed during generation - unique_id: {1236}
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\momepy\elements.py:395: UserWarning: Tessellation contains MultiPolygon elements. Initial objects should be edited. unique_id of affected elements: [1799, 405, 655, 1719, 446, 654, 663, 445, 317, 638, 633, 634, 640, 631, 796, 799, 815, 551, 671, 1786, 723, 552, 174, 1796, 1459, 1787, 1797, 1782, 842, 1450, 621, 1785, 1403, 291, 188, 1829, 1830, 181, 1461, 189, 947, 1466, 179, 1744, 629, 844, 1922, 1924, 851, 857, 860, 343, 1165, 1736, 644, 1628, 274, 1278, 1938, 591, 1937, 1276, 1270, 134, 1267, 1269, 142, 1629, 1811, 270, 1886, 1810, 1123, 1122, 1129, 1130, 1124, 1112, 1232, 1930, 229, 1912, 1929, 1923, 1458, 1317, 1045, 87, 297, 1365, 1368, 1360, 95, 1500, 570, 573, 571, 1058, 98, 1869, 1792]
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\momepy\distribution.py:893: FutureWarning: iteritems is deprecated and will be removed in a future version. Use .items instead.
  gdf.set_index(unique_id).geometry.iteritems(),
  0%|          | 0/657 [00:00<?, ?it/s]
  0%|          | 0/657 [00:00<?, ?it/s]
  0%|          | 0/1947 [00:00<?, ?it/s]
```

In [12]:
```
buildings.head(1)
```

Out[12]:

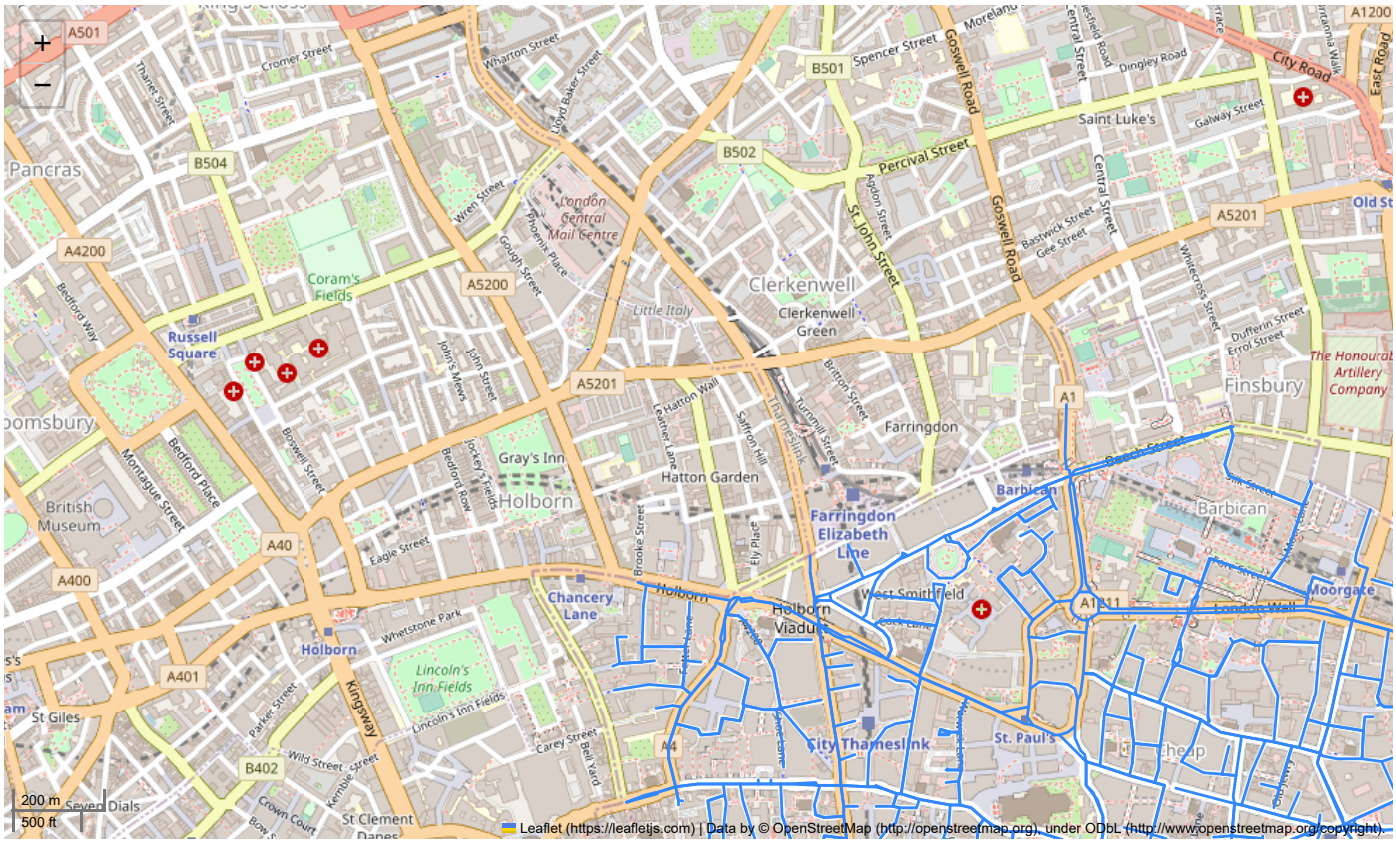
	geometry	uID	MSOA11CD	nID	area	eri	elongation	shared_walls	neighbor_distance	interbuilding_distance	adjacency	nodeID
0	POLYGON ((531541.552 181119.077, 531541.397 18...	0	E02000001	230	681.10988	1.001729	0.5879	0.0	10.15555	9.839758	0.393939	182

In [13]:
```
buildings.explore() #explore all buildings in central london (city of london)
```

Out[13]:

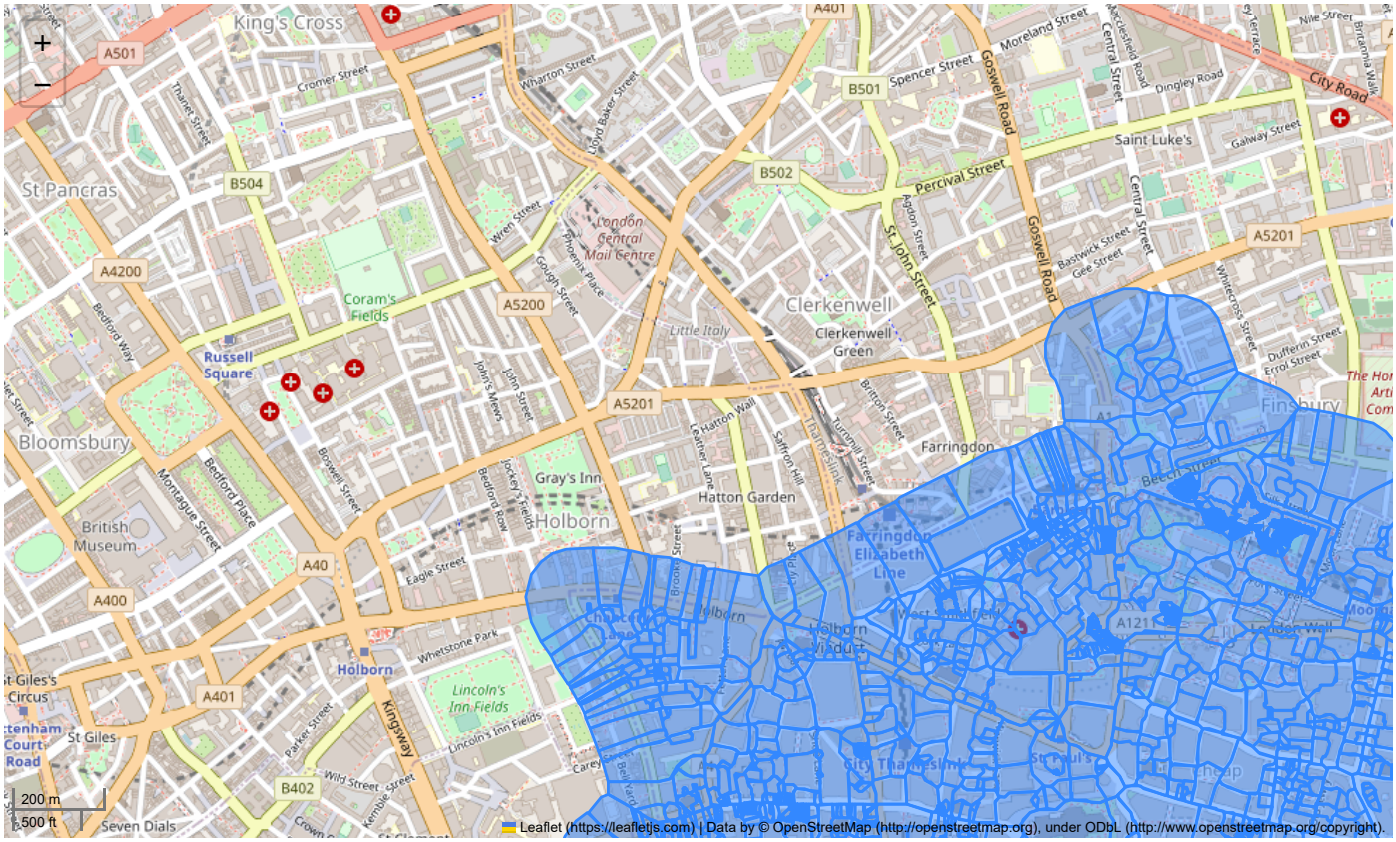
In [14]:
```
streets.explore()
```

Out[14]:



In [15]:
```
tessellation.explore()
```

Out[15]:



In [16]:

```
blocks = momepy.Blocks(tessellation, streets, buildings, 'bID', 'uID')
blocks.blocks.head()
blocks.blocks.explore()
```

Out[16]:

In [17]:

```
f, ax = plt.subplots(figsize=(10, 10))
buildings.plot('area', ax=ax, legend=True, scheme='quantiles', cmap='Blues',
               legend_kwds={'loc': 'lower left'})
ax.set_axis_off()
plt.show()
```

In [18]:
```
fig, ax = plt.subplots(1, 2, figsize=(24, 12))

buildings.plot("eri", ax=ax[0], scheme="natural_breaks", legend=True)
buildings.plot("elongation", ax=ax[1], scheme="natural_breaks", legend=True)

ax[0].set_axis_off()
ax[1].set_axis_off()
```

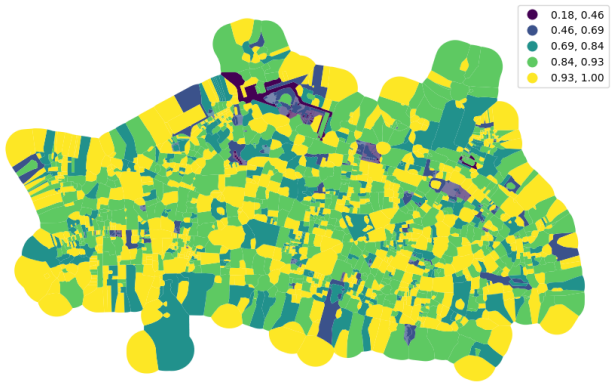
```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=8.
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=8.
  warnings.warn(
```

In [19]:
```
fig, ax = plt.subplots(1, 2, figsize=(24, 12))

tessellation.plot("convexity", ax=ax[0], scheme="natural_breaks", legend=True)
streets.plot("linearity", ax=ax[1], scheme="natural_breaks", legend=True)

ax[0].set_axis_off()
ax[1].set_axis_off()
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=8.
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=5.
  warnings.warn(
```

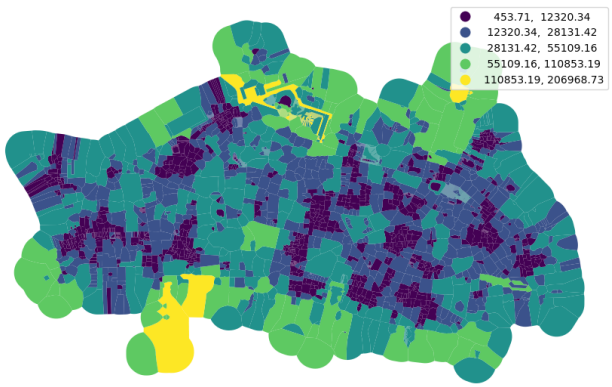


In [20]:
```
fig, ax = plt.subplots(1, 2, figsize=(24, 12))

buildings.plot("neighbor_distance", ax=ax[0], scheme="natural_breaks", legend=True)
tessellation.plot("covered_area", ax=ax[1], scheme="natural_breaks", legend=True)

ax[0].set_axis_off()
ax[1].set_axis_off()
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=8.
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=8.
  warnings.warn(
```



In [21]:
```
fig, ax = plt.subplots(1, 2, figsize=(24, 12))

buildings.plot("interbuilding_distance", ax=ax[0], scheme="natural_breaks", legend=True)
buildings.plot("adjacency", ax=ax[1], scheme="natural_breaks", legend=True)

ax[0].set_axis_off()
ax[1].set_axis_off()
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=8.
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=8.
  warnings.warn(
```

In [22]:
```
fig, ax = plt.subplots(1, 3, figsize=(24, 12))

streets.plot("width", ax=ax[0], scheme="natural_breaks", legend=True) #street width
streets.plot("width_deviation", ax=ax[1], scheme="natural_breaks", legend=True) #street width deviation
```

```
streets.plot("openness", ax=ax[2], scheme="natural_breaks", legend=True) #street openness

ax[0].set_axis_off()
ax[1].set_axis_off()
ax[2].set_axis_off()
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_TH
READS=5.
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_TH
READS=5.
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_TH
READS=5.
  warnings.warn(
```

In [33]:
```
#Street Alignment

streets_graph = osmnx.graph_from_polygon(msoa, network_type='drive')
streets_graph= osmnx.projection.project_graph(streets_graph, to_crs='EPSG:27700')
edges = osmnx.graph_to_gdfs(streets_graph, nodes=False, edges=True,
                                   node_geometry=False, fill_edge_geometry=True)
edges['networkID'] = momepy.unique_id(edges)
buildings['networkID'] = momepy.get_network_id(buildings, edges,
                                               'networkID').values
```

```
Snapping:   0%|          | 0/1947 [00:00<?, ?it/s]
```

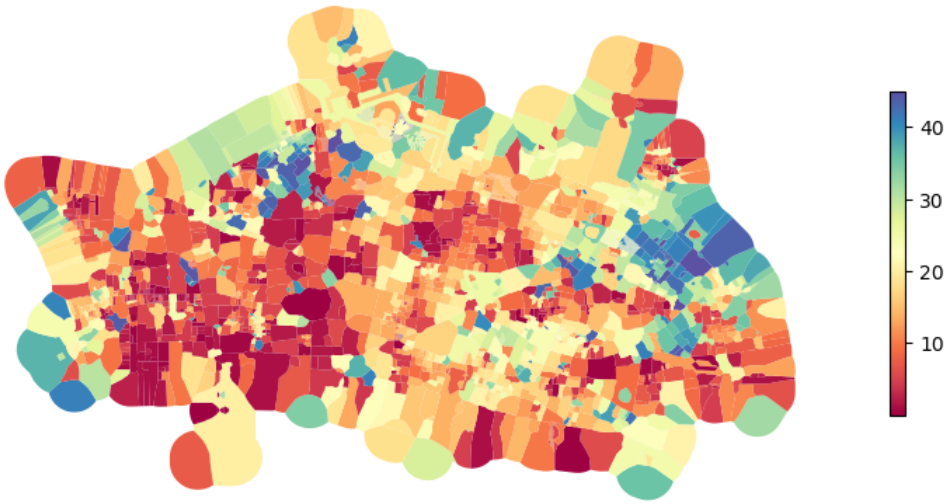
```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\momepy\elements.py:760: UserWarning: Some objects were not attached to the net
work. Set larger min_size. 36 affected elements
  warnings.warn(
```

In [23]:
```
buildings['orientation'] = momepy.Orientation(buildings).series
tessellation['orientation'] = momepy.Orientation(tessellation).series
```

```
  0%|          | 0/1947 [00:00<?, ?it/s]
  0%|          | 0/1946 [00:00<?, ?it/s]
```

In [24]:
```
##cell alignment
buildings.plot(column='orientation',legend=True,legend_kwds={"shrink":.3}, cmap='Spectral',
               figsize=(10, 10)).set_axis_off() #building solar orientation
```

In [25]:
```
##cell alignment
tessellation.plot(column='orientation',legend=True,legend_kwds={"shrink":.3}, cmap='Spectral',
               figsize=(10, 10)).set_axis_off() #tessellation orientation
```



In [31]:
```
blg_cell_align = momepy.CellAlignment(buildings, tessellation,
                                      'orientation', 'orientation',
                                      'uID', 'uID')
buildings['cell_align'] = blg_cell_align.series #generating cell alignment metric
```

In [32]:
```
buildings.plot(column='cell_align',legend=True,legend_kwds={"shrink":.3}, cmap='Spectral',
               figsize=(10, 10)).set_axis_off() #calculating cell alignment building orientation - tessellation orientation
```

In [34]:
```
buildings_net = buildings.dropna(subset=["networkID"])
buildings_net.plot(figsize=(10, 10)).set_axis_off()
```

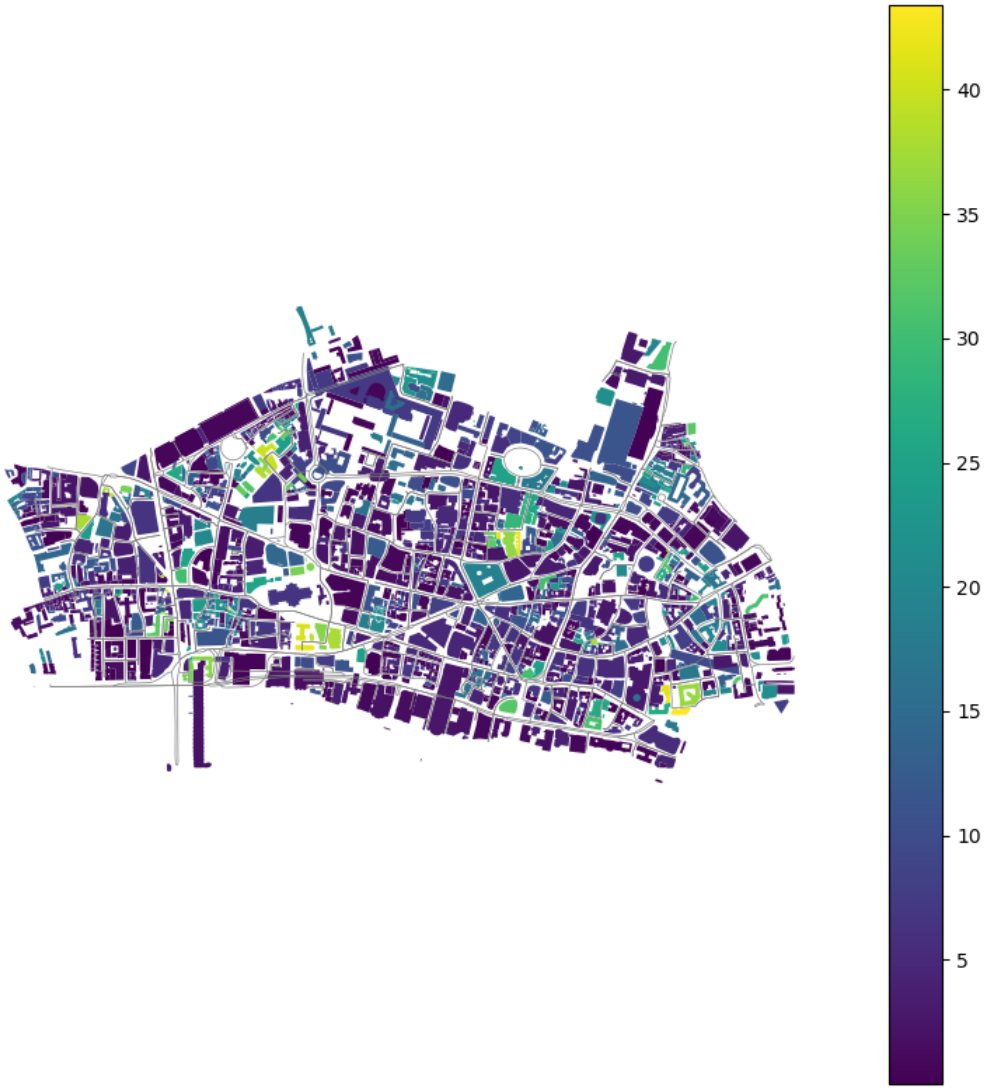
In [35]:
```
str_align = momepy.StreetAlignment(buildings_net, edges,
```

```
                                  'orientation', 'networkID',
                                  'networkID')
buildings_net['str_align'] = str_align.series
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\geopandas\geodataframe.py:1443: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  super().__setitem__(key, value)
```

In [36]:
```
ax = edges.plot(color='grey', linewidth=0.5, figsize=(10, 10))
buildings_net.plot(ax=ax, column='str_align', legend=True)
ax.set_axis_off()
```



In [37]:
```
print(eng_df.iloc[x].geometry.centroid)
```

```
POINT (-0.0923889468524288 51.514381323272744)
```

In [58]:
```
import osmnx as ox
point = (51.514381323272744, -0.0923889468524288)
dist = 2000
gdf = ox.geometries.geometries_from_point(point, dist=dist, tags={'building':True})
buildings = ox.projection.project_gdf(gdf)
buildings = buildings[buildings.geom_type.isin(['Polygon', 'MultiPolygon'])]

buildings['uID'] = momepy.unique_id(buildings)
limit = momepy.buffered_limit(buildings)
tessellation = momepy.Tessellation(buildings, unique_id='uID', limit=limit).tessellation

def clean_heights(x):
    try:
        return float(x)
    except ValueError:
        return 0

buildings['height'] = buildings['height'].fillna(0).apply(clean_heights)
buildings = buildings.reset_index().explode()
buildings.reset_index(inplace=True, drop=True)
streets_graph = ox.graph_from_point(point, dist, network_type='drive')
streets_graph = ox.projection.project_graph(streets_graph)
edges = ox.graph_to_gdfs(streets_graph, nodes=False, edges=True,
                                  node_geometry=False, fill_edge_geometry=True)
```

```
Inward offset...
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\momepy\elements.py:278: FutureWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To retain the old behavior, use either `df[df.columns[i]] = newvals` or, if columns are non-unique, `df.isetitem(i, newvals)`
  objects.loc[mask, objects.geometry.name] = objects[mask].buffer(
Generating input point array...
Generating Voronoi diagram...
Generating GeoDataFrame...
Dissolving Voronoi polygons...
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\geopandas\tools\clip.py:67: FutureWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To retain the old behavior, use either `df[df.columns[i]] = newvals` or, if columns are non-unique, `df.isetitem(i, newvals)`
  clipped.loc[
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\momepy\elements.py:384: UserWarning: Tessellation does not fully match buildings. 6 element(s) collapsed during generation - unique_id: {8455, 10291, 10293, 10296, 13627, 13629}
  warnings.warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\momepy\elements.py:395: UserWarning: Tessellation contains MultiPolygon elements. Initial objects should be edited. unique_id of affected elements: [1936, 385, 1327, 14406, 498, 14522, 10512, 5458, 5461, 12138, 12600, 12599, 8454, 5619, 8340, 8338, 8788, 8783, 7994, 12345, 962, 10788, 1357, 5516, 8553, 13898, 13233, 13234, 4134, 6470, 6473, 6489, 11186, 4138, 11187, 5489, 8551, 14510, 14509, 8545, 1011, 8542, 8544, 1027, 14426, 3522, 10683, 11809, 7872, 7871, 7879, 7873, 7878, 7861, 8354, 14471, 8725, 508, 4285, 506, 519, 8913, 9177, 9180, 10242, 516, 5378, 5386, 7755, 7768, 5380, 13620, 12362, 14463, 11842, 11845, 11843, 843, 11829, 5849, 14537, 5850, 5829, 5842, 751, 11821, 11823, 2756, 11813, 5448, 5423, 660, 11677, 8203, 14469, 14114, 2237, 14113, 14439, 7777, 5168, 12437, 10280, 157, 12136, 1795, 1852, 12368, 14530, 1681, 5073, 13973, 13972, 13631, 10233, 14515, 13593, 13560, 13559, 9428, 8752, 9427, 13589, 13588, 13590, 13623, 13979, 13624, 14549, 13981, 13980, 10251, 13982, 10279, 13983, 2075, 2074, 13592, 13586, 14542, 14498, 8885, 8886, 8910, 7775, 14474, 14475, 8309, 5941, 5947, 5938, 5951, 5945, 5940, 5962, 5074, 11638, 11612, 4449, 7940, 5209, 5961, 5970, 5208, 1990, 13417, 13418, 10015, 1991, 6979, 10030, 9868, 10089, 13521, 7649, 12377, 1875, 10013, 5699, 11653, 1903, 1905, 12378, 12279, 6534, 6516, 6518, 14452, 6525, 14454, 14453, 10012, 6531, 9390, 9389, 9392, 8486, 12333, 11871, 8950, 8949, 9019, 9021, 9016, 9017, 9014, 9015, 6015, 12335, 4193, 5328, 6079, 5329, 12334, 9599, 12336, 5691, 9981, 8995, 8996, 8999, 11342, 10159, 11340, 10158, 11337, 11338, 14534, 9119, 9363, 23, 9353, 9358, 9137, 9149, 9154, 9153, 12492, 9237, 6273, 6259, 7481, 7545, 7546, 7476, 7474, 7557, 6250, 6226, 14526, 11924, 7569, 7127, 7139, 7652, 7164, 10643, 12505, 12382, 4269, 9301]
  warnings.warn(
C:\Users\neetm\AppData\Local\Temp\ipykernel_17432\697467702.py:19: FutureWarning: Currently, index_parts defaults to True, but in the future, it will default to False to be consistent with Pandas. Use `index_parts=True` to keep the current behavior and True/False to silence the warning.
  buildings = buildings.reset_index().explode()
```

In [59]:
```
ax = buildings.plot(figsize=(10, 10), color='lightgrey')
edges.plot(ax=ax)
ax.set_axis_off()
```

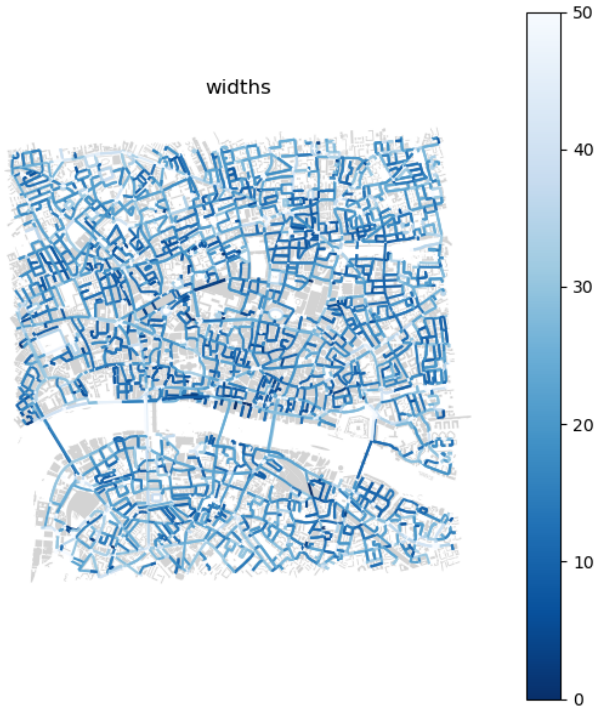
In [60]:
```
profile = momepy.StreetProfile(edges, buildings, heights='height')
```

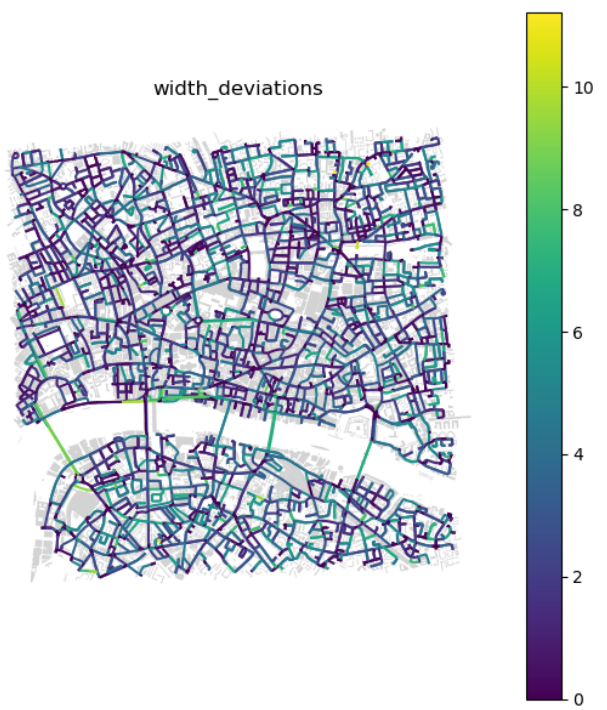
In [61]:
```
edges['widths'] = profile.w
```

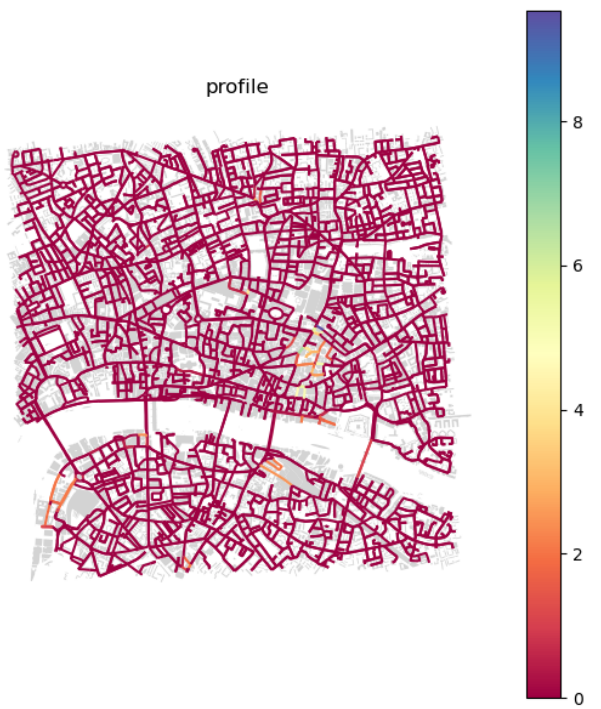
```
edges['width_deviations'] = profile.wd
edges['openness'] = profile.o
edges['heights'] = profile.h
edges['heights_deviations'] = profile.hd
edges['profile'] = profile.p
```

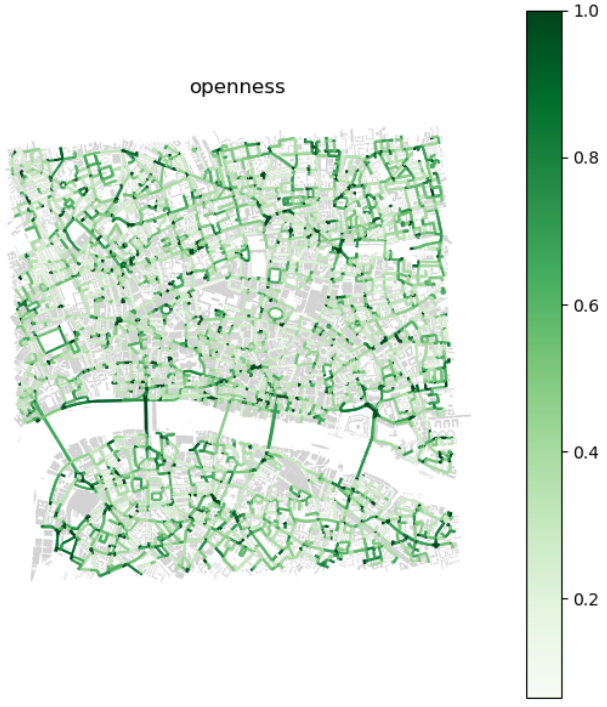
In [62]:

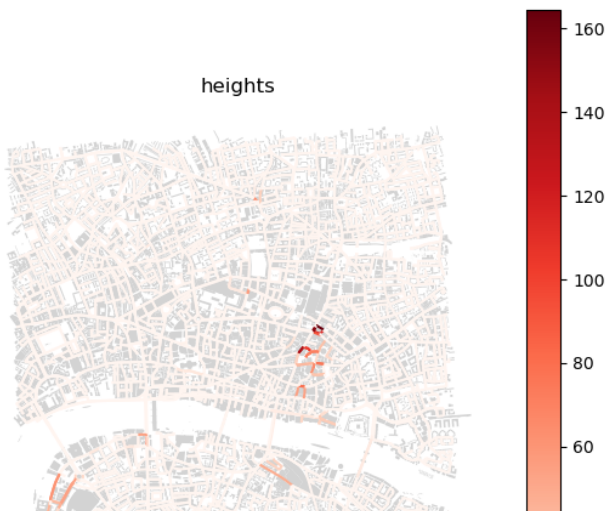
```
f, axes = plt.subplots(figsize=(15, 25), ncols=2, nrows=3)
edges.plot(ax=axes[0][0], column='widths', legend=True, cmap='Blues_r')
buildings.plot(ax=axes[0][0], color='lightgrey')
edges.plot(ax=axes[0][1], column='width_deviations', legend=True)
buildings.plot(ax=axes[0][1], color='lightgrey')
axes[0][0].set_axis_off()
axes[0][0].set_title('widths')
axes[0][1].set_axis_off()
axes[0][1].set_title('width_deviations')
edges.plot(ax=axes[1][0], column='profile', legend=True, cmap='Spectral')
buildings.plot(ax=axes[1][0], color='lightgrey')
edges.plot(ax=axes[1][1], column='openness', legend=True, cmap='Greens')
buildings.plot(ax=axes[1][1], color='lightgrey')
axes[1][0].set_axis_off()
axes[1][0].set_title('profile')
axes[1][1].set_axis_off()
axes[1][1].set_title('openness')
edges.plot(ax=axes[2][0], column='heights', legend=True, cmap='Reds')
buildings.plot(ax=axes[2][0], color='lightgrey')
edges.plot(ax=axes[2][1], column='heights_deviations', legend=True)
buildings.plot(ax=axes[2][1], color='lightgrey')
axes[2][0].set_axis_off()
axes[2][0].set_title('heights')
axes[2][1].set_axis_off()
axes[2][1].set_title('heights_deviations')
plt.show()
```

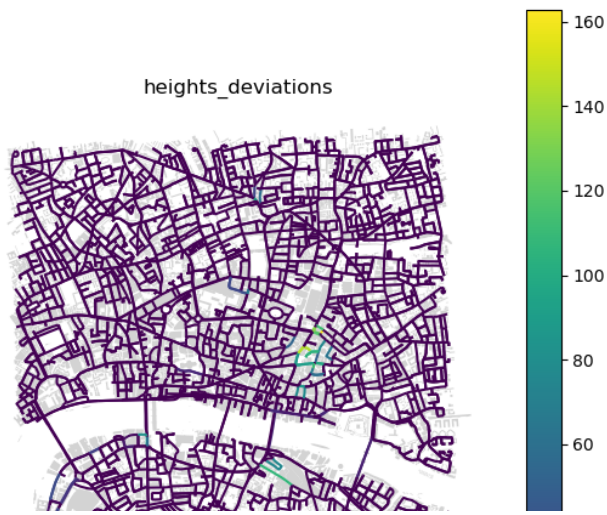

widths
50
40
30
20
10
0


width_deviations
10
8
6
4
2
0


profile
8
6
4
2
0


openness
1.0
0.8
0.6
0.4
0.2

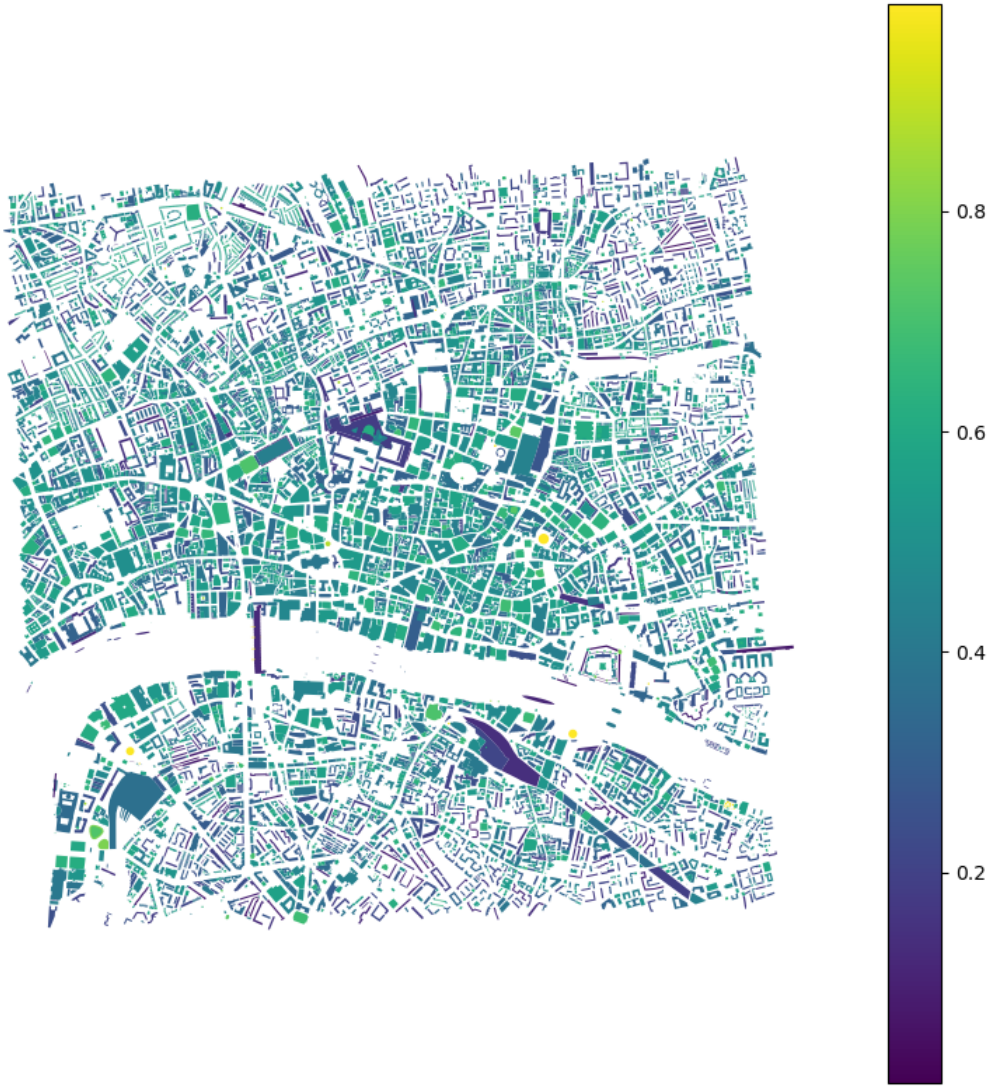

heights
160
140
120
100
80
60


heights_deviations
160
140
120
100
80
60

In [63]:
```
f, ax = plt.subplots(figsize=(10, 10))
tessellation.plot(ax=ax)
buildings.plot(ax=ax, color='white', alpha=.5)
ax.set_axis_off()
plt.show()
```

In [70]:
```
blg_cc = momepy.CircularCompactness(buildings)
buildings['circular_com'] = blg_cc.series #Circular compactness measures the ratio of object area to the area of its smallest circumsribed ci
```

In [71]:
```
f, ax = plt.subplots(figsize=(10, 10))
buildings.plot(ax=ax, column='circular_com', legend=True, cmap='viridis')
ax.set_axis_off()
plt.show()
```

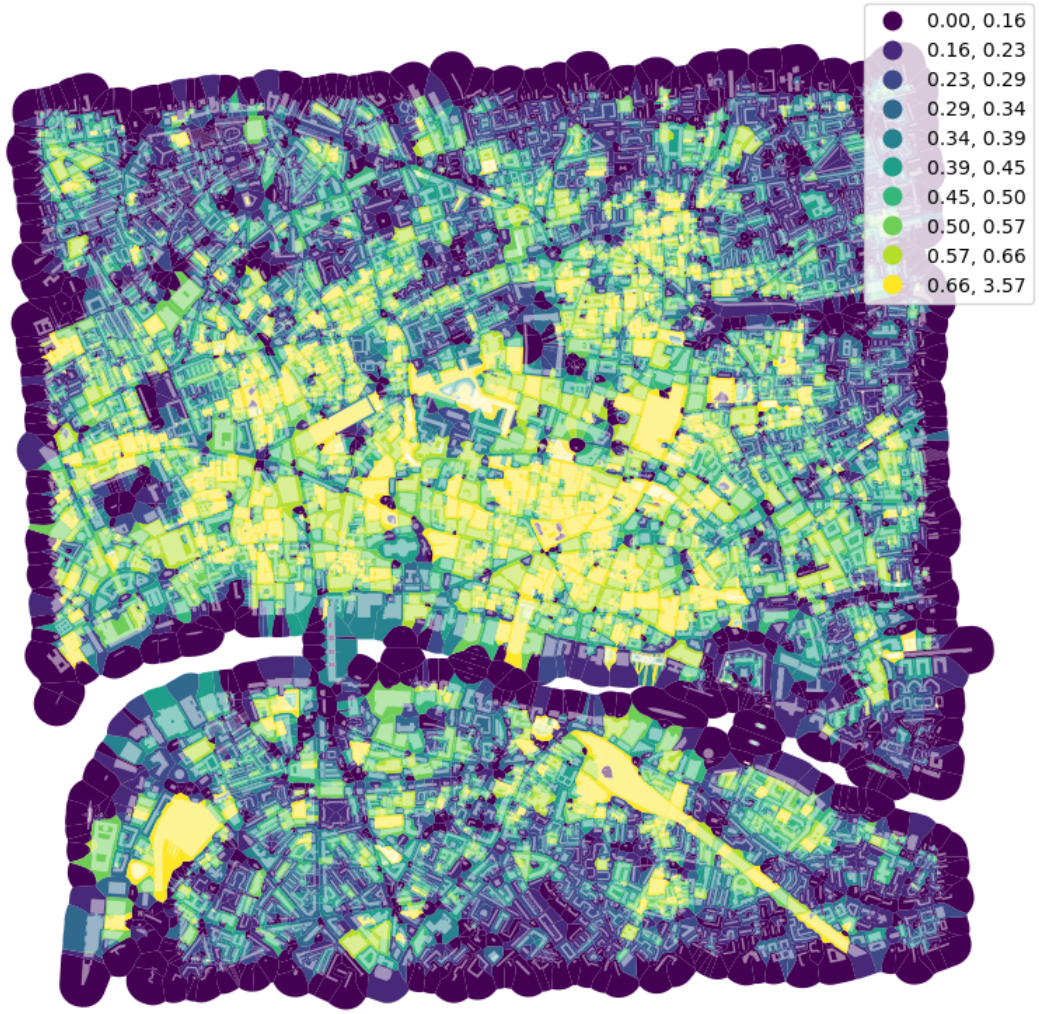


measuring density

In [64]:
```
tessellation['area'] = momepy.Area(tessellation).series #tessellation area
buildings['area'] = momepy.Area(buildings).series
tess_car = momepy.AreaRatio(tessellation, buildings, 'area', 'area', 'uID') #tessellation area ratio
tessellation['CAR'] = tess_car.series #coverage ratio
```

In [65]:
```
tess_car = momepy.AreaRatio(tessellation, buildings,
                            momepy.Area(tessellation).series,
                            momepy.Area(buildings).series, 'uID')
tessellation['CAR'] = tess_car.series
```

In [66]:
```
f, ax = plt.subplots(figsize=(10, 10))
tessellation.plot(ax=ax, column='CAR', legend=True, scheme='quantiles', k=10, cmap='viridis')
buildings.plot(ax=ax, color='white', alpha=0.5)
ax.set_axis_off()
plt.show() #coverage area ratio
```



In [ ]:
```
def clean_heights(x):
    try:
        return float(x)
    except ValueError:
        return 0

buildings['height'] = buildings['height'].fillna(0).apply(clean_heights)

buildings['floor_area'] = momepy.FloorArea(buildings, 'height').series
```

In [68]:
```
tessellation['FAR'] = momepy.AreaRatio(tessellation, buildings,
                                       'area', 'floor_area', 'uID').series #floor area ratio
```

In [69]:
```
f, ax = plt.subplots(figsize=(10, 10))
tessellation.plot(ax=ax, column='FAR', legend=True, scheme='quantiles', k=10, cmap='viridis')
buildings.plot(ax=ax, color='white', alpha=0.5)
ax.set_axis_off()
plt.show()
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\mapclassify\classifiers.py:238: UserWarning: Warning: Not enough unique values in array to form k classes
  Warn(
C:\Users\neetm\anaconda3\envs\geopandas_env\Lib\site-packages\mapclassify\classifiers.py:241: UserWarning: Warning: setting k to 2
  Warn("Warning: setting k to %d" % k_q, UserWarning)
```

import data for Urban Form via Parquet Files

In [ ]:
```
import pandas as pd
import matplotlib as plt
import os
import osmnx
import dask
import dask.dataframe as dd
import glob
from dask.distributed import Client
import fastparquet
import geopandas as gpd
import dask_geopandas
```

In [ ]:
```
path = r'C:\Users\neetm\Desktop\Dissertation\Data\MSOAs'
urban1 = dask_geopandas.read_parquet(path + f'/msoas_urbanform_med.pq')
```

In [6]:
```
urban1[urban1['msoa11cd']=='E02000001']
```

Out[6]: **Dask-GeoPandas GeoDataFrame Structure:**

	msoa11cd	sdbAre	sdbHei	sdbPer	sdbVol	ssbFoF	ssbVFR	sdbCoA	ssbCCo	ssbCor	ssbSqu	ssbERI	ssbElo	ssbCCM	ssbCCD	stbOri	sdcLAL	sdc
npartitions=1																		
	object	float64	float64	float64	float64	float64	float64	float64	float64	float64	float64	float64	float64	float64	float64	float64	float64	flo
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

Dask Name: getitem, 4 tasks

In [8]:
```
path = r'C:\Users\neetm\Desktop\Dissertation\Data\MSOAs'
urban2 = dask_geopandas.read_parquet(path + f'/msoas_urbanform_med.pq')
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\dask\dataframe\io\parquet\arrow.py:1903: FutureWarning: 'ParquetDataset.partitions' attribute is deprecated as of pyarrow 5.0.0 and will be removed in a future version. Specify 'use_legacy_dataset=False' while constructing the ParquetDataset, and then use the '.partitioning' attribute instead.
  if dataset.partitions is not None:
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\dask\dataframe\io\parquet\arrow.py:1931: FutureWarning: 'ParquetDataset.metadata' attribute is deprecated as of pyarrow 5.0.0 and will be removed in a future version.
  if dataset.metadata:
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\dask\dataframe\io\parquet\arrow.py:1950: FutureWarning: 'ParquetDataset.schema' attribute is deprecated as of pyarrow 5.0.0 and will be removed in a future version. Specify 'use_legacy_dataset=False' while constructing the ParquetDataset, and then use the '.schema' attribute instead (which will return an Arrow schema instead of a Parquet schema).
  if dataset.schema is not None:
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\dask\dataframe\io\parquet\arrow.py:1978: FutureWarning: 'ParquetDataset.pieces' attribute is deprecated as of pyarrow 5.0.0 and will be removed in a future version. Specify 'use_legacy_dataset=False' while constructing the ParquetDataset, and then use the '.fragments' attribute instead.
  if len(dataset.pieces) > 1:
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\dask\dataframe\io\parquet\arrow.py:1990: FutureWarning: 'ParquetDataset.pieces' attribute is deprecated as of pyarrow 5.0.0 and will be removed in a future version. Specify 'use_legacy_dataset=False' while constructing the ParquetDataset, and then use the '.fragments' attribute instead.
  for piece, fn in zip(dataset.pieces, fns):
```

In [9]:
```
urban2.head()
```

Out[9]:

	msoa11cd	sdbAre	sdbHei	sdbPer	sdbVol	ssbFoF	ssbVFR	sdbCoA	ssbCCo	ssbCor	...	misCel	mdsAre	lisCel	ldsAre	ob
0	E02000001	136.281525	23.1	58.283511	2974.246398	0.700039	2.390425	0.0	0.449578	6.0	...	29.000000	49620.352109	111.122772	227273.375613	
1	E02000002	50.414375	7.5	29.718539	391.234663	0.980618	1.726172	0.0	0.567912	4.0	...	67.195382	39354.746259	223.000000	175858.425524	
2	E02000003	56.730325	8.5	33.221816	478.474295	0.943607	1.719994	0.0	0.527822	6.0	...	88.000000	33209.520648	298.271847	149024.773835	
3	E02000004	52.297068	7.8	30.696216	402.000240	0.982960	1.692187	0.0	0.540299	4.0	...	118.520702	43289.679552	376.186231	183850.215778	
4	E02000005	47.146250	8.3	29.397671	391.262617	0.893687	1.610037	0.0	0.557735	6.0	...	131.000000	44284.845682	586.000000	221713.532750	

5 rows × 76 columns

In [10]:
```
urban2.columns.to_list()
```

Out[10]:
```
['msoa11cd',
 'sdbAre',
 'sdbHei',
 'sdbPer',
 'sdbVol',
 'ssbFoF',
 'ssbVFR',
 'sdbCoA',
 'ssbCCo',
 'ssbCor',
 'ssbSqu',
 'ssbERI',
 'ssbElo',
 'ssbCCM',
 'ssbCCD',
 'stbOri',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sscERI',
 'stcOri',
 'sicCAR',
 'stbCeA',
 'mtbAli',
 'mtbNDi',
 'mtcWNe',
 'mdcAre',
 'ltcWRE',
 'ltbIBD',
 'ltcRea',
 'ltcAre',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSHD',
 'sdsSPR',
 'sdsLen',
 'sssLin',
 'ldsMSL',
 'mtdDeg',
 'lcdMes',
 'linP3W',
 'linP4W',
 'linPDE',
 'lcnClo',
 'ldsCDL',
 'xcnSCl',
 'mtdMDi',
 'lddNDe',
 'linWID',
 'ldeAre',
 'ldePer',
 'lseCCo',
 'lseERI',
 'lseCWA',
 'lteOri',
 'lteWNB',
 'lieWCe',
 'stbSAl',
 'ldbPWL',
 'mtbSWR',
 'sddAre',
 'sdsAre',
 'sisBpM',
 'misCel',
 'mdsAre',
 'lisCel',
 'ldsAre',
 'objectid',
 'msoa11nm',
 'msoa11nmw',
 'st_areasha',
 'st_lengths',
 'geometry']
```

In [14]:
```
urban_mean = gpd.GeoDataFrame(urban1)
```

In [16]:
```
urban_mean.columns = ['msoa11cd',
 'sdbAre',
 'sdbHei',
 'sdbPer',
 'sdbVol',
 'ssbFoF',
 'ssbVFR',
 'sdbCoA',
 'ssbCCo',
 'ssbCor',
 'ssbSqu',
```

```
 'ssbERI',
 'ssbElo',
 'ssbCCM',
 'ssbCCD',
 'stbOri',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sscERI',
 'stcOri',
 'sicCAR',
 'stbCeA',
 'mtbAli',
 'mtbNDi',
 'mtcWNe',
 'mdcAre',
 'ltcWRE',
 'ltbIBD',
 'ltcRea',
 'ltcAre',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSHD',
 'sdsSPR',
 'sdsLen',
 'sssLin',
 'ldsMSL',
 'mtdDeg',
 'lcdMes',
 'linP3W',
 'linP4W',
 'linPDE',
 'lcnClo',
 'ldsCDL',
 'xcnSCl',
 'mtdMDi',
 'lddNDe',
 'linWID',
 'ldeAre',
 'ldePer',
 'lseCCo',
 'lseERI',
 'lseCWA',
 'lteOri',
 'lteWNB',
 'lieWCe',
 'stbSAl',
 'ldbPWL',
 'mtbSWR',
 'sddAre',
 'sdsAre',
 'sisBpM',
 'misCel',
 'mdsAre',
 'lisCel',
 'ldsAre',
 'geometry']
```

In [18]:
```
urban_mean.to_csv('urbanmean.csv')
```

In [11]:
```
urban2.columns.to_list()
```

Out[11]:
```
['msoa11cd',
 'sdbAre',
 'sdbHei',
 'sdbPer',
 'sdbVol',
 'ssbFoF',
 'ssbVFR',
 'sdbCoA',
 'ssbCCo',
 'ssbCor',
 'ssbSqu',
 'ssbERI',
 'ssbElo',
 'ssbCCM',
 'ssbCCD',
 'stbOri',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sscERI',
 'stcOri',
 'sicCAR',
 'stbCeA',
 'mtbAli',
 'mtbNDi',
 'mtcWNe',
 'mdcAre',
 'ltcWRE',
 'ltbIBD',
 'ltcRea',
 'ltcAre',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSHD',
 'sdsSPR',
 'sdsLen',
 'sssLin',
 'ldsMSL',
 'mtdDeg',
 'lcdMes',
 'linP3W',
 'linP4W',
 'linPDE',
 'lcnClo',
 'ldsCDL',
 'xcnSCl',
 'mtdMDi',
 'lddNDe',
 'linWID',
 'ldeAre',
 'ldePer',
 'lseCCo',
 'lseERI',
 'lseCWA',
 'lteOri',
 'lteWNB',
 'lieWCe',
 'stbSAl',
 'ldbPWL',
 'mtbSWR',
 'sddAre',
 'sdsAre',
 'sisBpM',
 'misCel',
 'mdsAre',
 'lisCel',
 'ldsAre',
 'objectid',
 'msoa11nm',
 'msoa11nmw',
 'st_areasha',
 'st_lengths',
 'geometry']
```

In [12]:
```
urban_median = gpd.GeoDataFrame(urban2)
```

In [13]:
```
urban_median.head()
```

```
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\core\dtypes\inference.py:387: ShapelyDeprecationWarning: Iteration over
multi-part geometries is deprecated and will be removed in Shapely 2.0. Use the `geoms` property to access the constituent parts of a multi-
part geometry.
  iter(obj)  # Can iterate over it.
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\core\dtypes\inference.py:388: ShapelyDeprecationWarning: __len__ for mu
lti-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` property instead to get the  number of
parts of a multi-part geometry.
  len(obj)  # Has a length associated with it.
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\io\formats\printing.py:117: ShapelyDeprecationWarning: Iteration over m
ulti-part geometries is deprecated and will be removed in Shapely 2.0. Use the `geoms` property to access the constituent parts of a multi-p
art geometry.
  s = iter(seq)
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\io\formats\printing.py:121: ShapelyDeprecationWarning: __len__ for mult
i-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` property instead to get the  number of p
arts of a multi-part geometry.
  for i in range(min(nitems, len(seq)))
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\io\formats\printing.py:125: ShapelyDeprecationWarning: __len__ for mult
i-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` property instead to get the  number of p
arts of a multi-part geometry.
  if nitems < len(seq):
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\core\dtypes\inference.py:387: ShapelyDeprecationWarning: Iteration over
multi-part geometries is deprecated and will be removed in Shapely 2.0. Use the `geoms` property to access the constituent parts of a multi-
part geometry.
  iter(obj)  # Can iterate over it.
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\core\dtypes\inference.py:388: ShapelyDeprecationWarning: __len__ for mu
lti-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` property instead to get the  number of
parts of a multi-part geometry.
  len(obj)  # Has a length associated with it.
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\io\formats\printing.py:117: ShapelyDeprecationWarning: Iteration over m
ulti-part geometries is deprecated and will be removed in Shapely 2.0. Use the `geoms` property to access the constituent parts of a multi-p
art geometry.
  s = iter(seq)
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\io\formats\printing.py:121: ShapelyDeprecationWarning: __len__ for mult
i-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` property instead to get the  number of p
arts of a multi-part geometry.
  for i in range(min(nitems, len(seq)))
C:\Users\neetm\anaconda3\envs\geopandas_env\lib\site-packages\pandas\io\formats\printing.py:125: ShapelyDeprecationWarning: __len__ for mult
i-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` property instead to get the  number of p
arts of a multi-part geometry.
  if nitems < len(seq):
```

Out[13]:

	0	1	2	3	4	5	6	7	8	9	...	66	67	68	69	70	7
0	E02000001	136.281525	23.1	58.283511	2974.246398	0.700039	2.390425	0.0	0.449578	6.0	...	29.0	49620.352109	111.122772	227273.375613	1	City o Londo 00
1	E02000002	50.414375	7.5	29.718539	391.234663	0.980618	1.726172	0.0	0.567912	4.0	...	67.195382	39354.746259	223.0	175858.425524	2	Barkin an Dagenhar 00
2	E02000003	56.730325	8.5	33.221816	478.474295	0.943607	1.719994	0.0	0.527822	6.0	...	88.0	33209.520648	298.271847	149024.773835	3	Barkin an Dagenhar 00
3	E02000004	52.297068	7.8	30.696216	402.00024	0.98296	1.692187	0.0	0.540299	4.0	...	118.520702	43289.679552	376.186231	183850.215778	4	Barkin an Dagenhar 00
4	E02000005	47.14625	8.3	29.397671	391.262617	0.893687	1.610037	0.0	0.557735	6.0	...	131.0	44284.845682	586.0	221713.53275	5	Barkin an Dagenhar 00

5 rows × 76 columns

In [16]:
```
urban_median.columns = ['msoa11cd',
 'sdbAre',
 'sdbHei',
 'sdbPer',
 'sdbVol',
 'ssbFoF',
 'ssbVFR',
 'sdbCoA',
 'ssbCCo',
 'ssbCor',
 'ssbSqu',
 'ssbERI',
 'ssbElo',
 'ssbCCM',
 'ssbCCD',
 'stbOri',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sscERI',
 'stcOri',
```

```
 'sicCAR',
 'stbCeA',
 'mtbAli',
 'mtbNDi',
 'mtcWNe',
 'mdcAre',
 'ltcWRE',
 'ltbIBD',
 'ltcRea',
 'ltcAre',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSHD',
 'sdsSPR',
 'sdsLen',
 'sssLin',
 'ldsMSL',
 'mtdDeg',
 'lcdMes',
 'linP3W',
 'linP4W',
 'linPDE',
 'lcnClo',
 'ldsCDL',
 'xcnSCl',
 'mtdMDi',
 'lddNDe',
 'linWID',
 'ldeAre',
 'ldePer',
 'lseCCo',
 'lseERI',
 'lseCWA',
 'lteOri',
 'lteWNB',
 'lieWCe',
 'stbSAl',
 'ldbPWL',
 'mtbSWR',
 'sddAre',
 'sdsAre',
 'sisBpM',
 'misCel',
 'mdsAre',
 'lisCel',
 'ldsAre',
 'objectid',
 'msoa11nm',
 'msoa11nmw',
 'st_areasha',
 'st_lengths',
 'geometry']
```

In [17]:
```
urban_median.to_csv('urbanmedian.csv') #export to csv to be used in next python code including with all machine learning
```