Unfortunately, my PC can't handle with such big dataset, I tried different methods (including StratifiedKFold from sklearn) to reduce the size and save the same data's distribution, but it also didn't work because of memory issues. I will use 5000 rows from shuffled dataset

In [13]:
```
from sklearn.utils import shuffle
df_taxi = shuffle(df_taxi,random_state=0)
```

In [17]:
```
df_taxi=df_taxi.iloc[0:5000].reset_index().drop(columns=["index"])
```

In [18]:
```
#split taxi dataset to 2 datasets: 1 with start point, the other one with destination point
df_taxi_start=gpd.GeoDataFrame(df_taxi.merge(taxi_zones_3857, left_on="PULocationID",right_on="LocationID"),geo
df_taxi_end=gpd.GeoDataFrame(df_taxi.merge(taxi_zones_3857, left_on="DOLocationID",right_on="LocationID"),geome
```

In [14]:
```
df_taxi_start_4326=df_taxi_start.to_crs("EPSG:4326")
df_taxi_end_4326=df_taxi_end.to_crs("EPSG:4326")
```

In [19]:
```
df_taxi_start["x"]=df_taxi_start["geometry"].x
df_taxi_start["y"]=df_taxi_start["geometry"].y
df_taxi_end["x"]=df_taxi_end["geometry"].x
df_taxi_end["y"]=df_taxi_end["geometry"].y
```

In [20]:
```
points_start=df_taxi_start[["x","y"]].values.tolist()
```

In [21]:
```
points_end=df_taxi_end[["x","y"]].values.tolist()
```

In [22]:
```
from pointpats import PointPattern, PoissonPointProcess, as_window, G, F, J, K, L, Genv, Fenv, Jenv, Kenv, Lenv
```

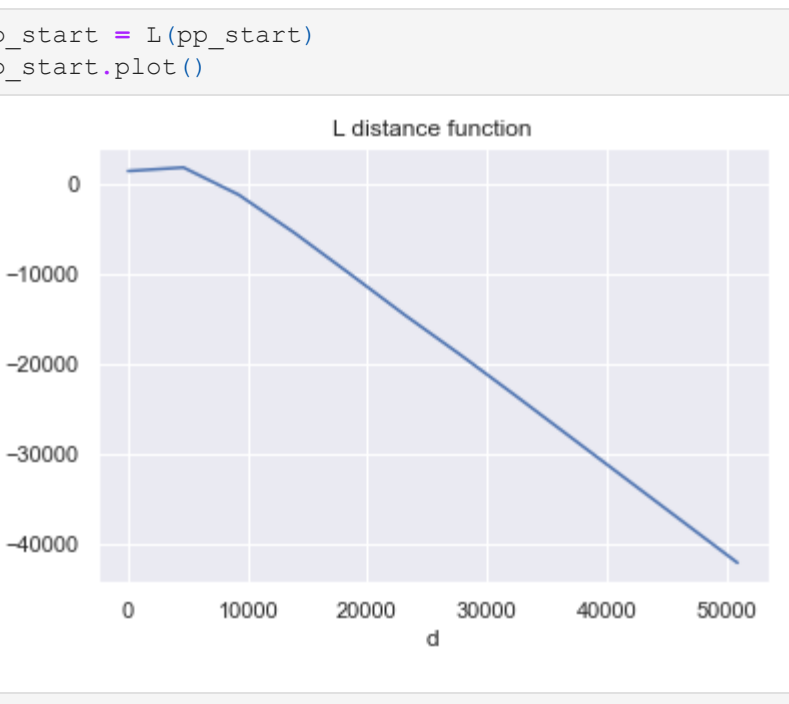
In [23]:
```
import pointpats
pointpats.__version__
```

Out[23]:
```
'2.1.0'
```

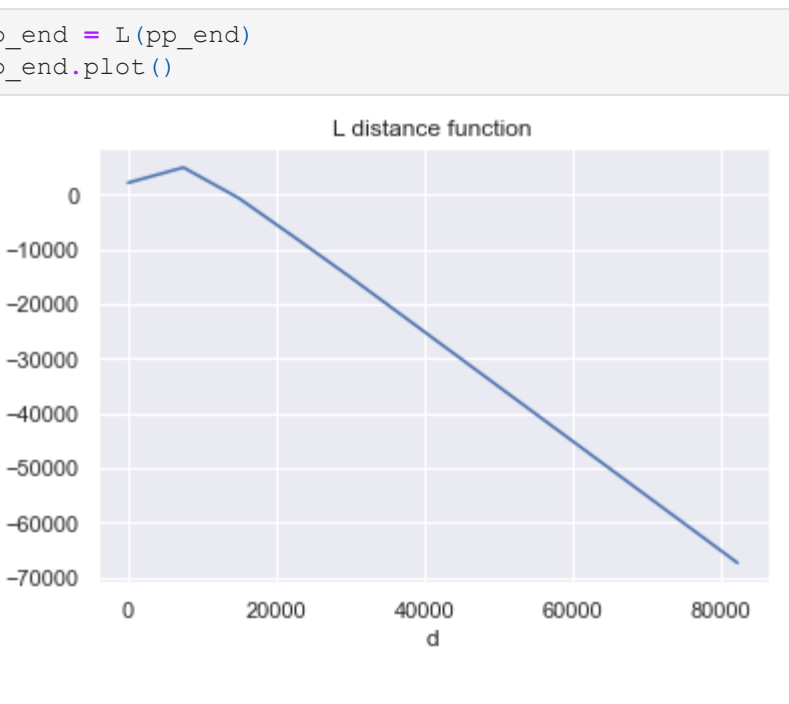
In [25]:
```
pp_start = PointPattern(points_start)
pp_end = PointPattern(points_end)
```

In [1]:
```
#I used L function to identify whether the data is clustered or not
#values above 0 indicate clustering, while below 0 show that data is dispersed https://mgimond.github.io/Spatia
#According to plots, start and end points are a little bit clustered until 5000m, but afterwards they become mo
#dispersed
#manual - https://nbviewer.org/github/pysal/pointpats/blob/master/notebooks/distance_statistics.ipynb
```

In [28]:
```
lp_start = L(pp_start)
lp_start.plot()
```



In [29]:
```
lp_end = L(pp_end)
lp_end.plot()
```



Hotspots for nightlife activities in NYC

In [153…
```
taxi_zones=pd.read_csv("taxi_zones.csv")
taxi_zones['geometry'] = taxi_zones.the_geom.apply(wkt.loads)
taxi_zones=taxi_zones.drop(columns=["the_geom"])
taxi_zones_gdf=gpd.GeoDataFrame(taxi_zones,geometry="geometry",crs="EPSG:4326")
taxi_zones_3857=taxi_zones_gdf.to_crs("EPSG:3857")
```

In [155…
```
df_taxi=pd.read_csv("yellow_tripdata_2020-01.csv")
df_taxi=df_taxi.dropna()
df_taxi["tpep_pickup_datetime"]=pd.to_datetime(df_taxi["tpep_pickup_datetime"])
df_taxi["tpep_dropoff_datetime"]=pd.to_datetime(df_taxi["tpep_dropoff_datetime"])
df_taxi_start=gpd.GeoDataFrame(df_taxi.merge(taxi_zones_3857, left_on="PULocationID",right_on="LocationID"),geo
df_taxi_end=gpd.GeoDataFrame(df_taxi.merge(taxi_zones_3857, left_on="DOLocationID",right_on="LocationID"),geome
```

In [159…
```
df_taxi_start["hour"]=df_taxi_start["tpep_pickup_datetime"].dt.hour
df_taxi_end["hour"]=df_taxi_start["tpep_dropoff_datetime"].dt.hour
```

In [160…
```
#let's assume that night time is from 0 until 5 a.m.
df_taxi_start["is_night"]=((df_taxi_start["hour"]>0) & (df_taxi_start["hour"]<5)).astype("int")
df_taxi_end["is_night"]=((df_taxi_end["hour"]>0) & (df_taxi_end["hour"]<5)).astype("int")
```

In [161…
```
df_taxi_start=df_taxi_start.groupby("PULocationID").sum().reset_index()
```

In [162…
```
df_taxi_end=df_taxi_end.groupby("DOLocationID").sum().reset_index()
```

In [164…
```
df_taxi_start=df_taxi_start.merge(taxi_zones_3857[["geometry","LocationID"]],left_on="PULocationID",right_on="L
df_taxi_end=df_taxi_end.merge(taxi_zones_3857[["geometry","LocationID"]],left_on="DOLocationID",right_on="Locat
```

In [165…
```
df_taxi_start=gpd.GeoDataFrame(df_taxi_start,geometry="geometry",crs="EPSG:4326")
df_taxi_end=gpd.GeoDataFrame(df_taxi_end,geometry="geometry",crs="EPSG:4326")
```

In [167…
```
df_taxi_start=df_taxi_start[df_taxi_start["is_night"]>0]
df_taxi_end=df_taxi_end[df_taxi_end["is_night"]>0]
```

In [168…
```
#spatial weights

import libpysal as lps

wq_start =  lps.weights.Queen.from_dataframe(df_taxi_start)
wq_end =  lps.weights.Queen.from_dataframe(df_taxi_end)
```

In [169…
```
#Moran's I test for global autocorrelation
```

In [170…
```
wq_start.transform = 'r'
wq_end.transform = 'r'
```

```
('WARNING: ', 0, ' is an island (no neighbors)')
('WARNING: ', 40, ' is an island (no neighbors)')
('WARNING: ', 181, ' is an island (no neighbors)')
('WARNING: ', 0, ' is an island (no neighbors)')
('WARNING: ', 30, ' is an island (no neighbors)')
('WARNING: ', 38, ' is an island (no neighbors)')
('WARNING: ', 69, ' is an island (no neighbors)')
```

In [171…
```
df_taxi_start=df_taxi_start.reset_index().drop(columns=["index"]).drop(index=[0,40,181]).reset_index().drop(col
df_taxi_end=df_taxi_end.reset_index().drop(columns=["index"]).drop(index=[0,30,38,69]).reset_index().drop(colum
wq_start =  lps.weights.Queen.from_dataframe(df_taxi_start)
wq_end =  lps.weights.Queen.from_dataframe(df_taxi_end)
wq_start.transform = 'r'
wq_end.transform = 'r'
```

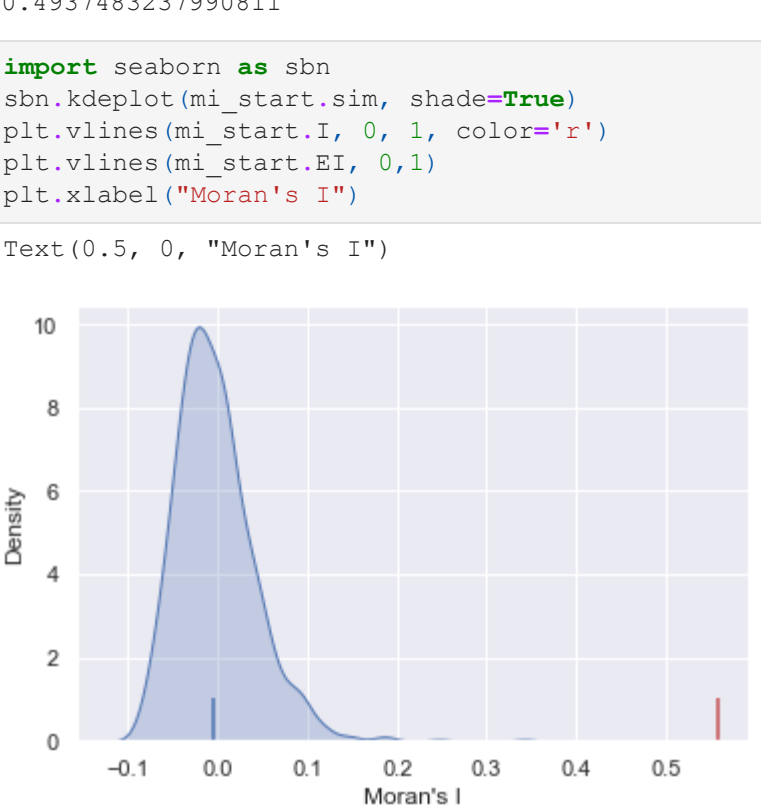
In [172…
```
import esda
mi_start = esda.moran.Moran(df_taxi_start["is_night"], wq_start)
mi_end = esda.moran.Moran(df_taxi_end["is_night"], wq_end)
```

In [173…
```
print(mi_start.I)
print(mi_end.I)
```

```
0.5572768987741263
0.4937483237990811
```

In [174…
```
import seaborn as sbn
sbn.kdeplot(mi_start.sim, shade=True)
plt.vlines(mi_start.I, 0, 1, color='r')
plt.vlines(mi_start.EI, 0,1)
plt.xlabel("Moran's I")
```

Out[174]:
```
Text(0.5, 0, "Moran's I")
```

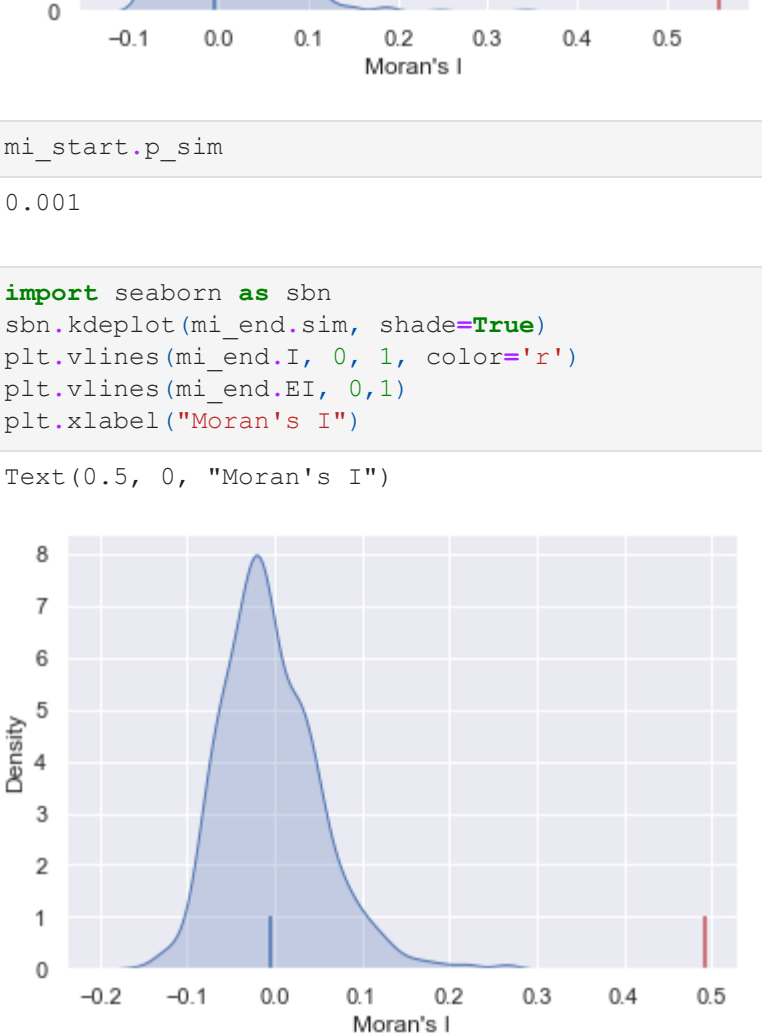


In [175…
```
mi_start.p_sim
```

Out[175]:
```
0.001
```

In [176…
```
import seaborn as sbn
sbn.kdeplot(mi_end.sim, shade=True)
plt.vlines(mi_end.I, 0, 1, color='r')
plt.vlines(mi_end.EI, 0,1)
plt.xlabel("Moran's I")
```

Out[176]:
```
Text(0.5, 0, "Moran's I")
```



In [177…
```
mi_end.p_sim
```

Out[177]:
```
0.001
```

In [178…
```
#statistical tests and plots above demonstrate that there is spatial autocorrelation
```

In [179…
```
#Let's compute a local Moran statistic
wq_start.transform = 'r'
wq_end.transform = 'r'
```

In [180…
```
li_start = esda.moran.Moran_Local(df_taxi_start["is_night"], wq_start, transformation = "r")
li_end = esda.moran.Moran_Local(df_taxi_end["is_night"], wq_end, transformation = "r")
```

In [181…
```
np.unique(li_start.q,return_counts=True)
```

Out[181]:
```
(array([1, 2, 3, 4]), array([ 39,  11, 181,   3], dtype=int64))
```

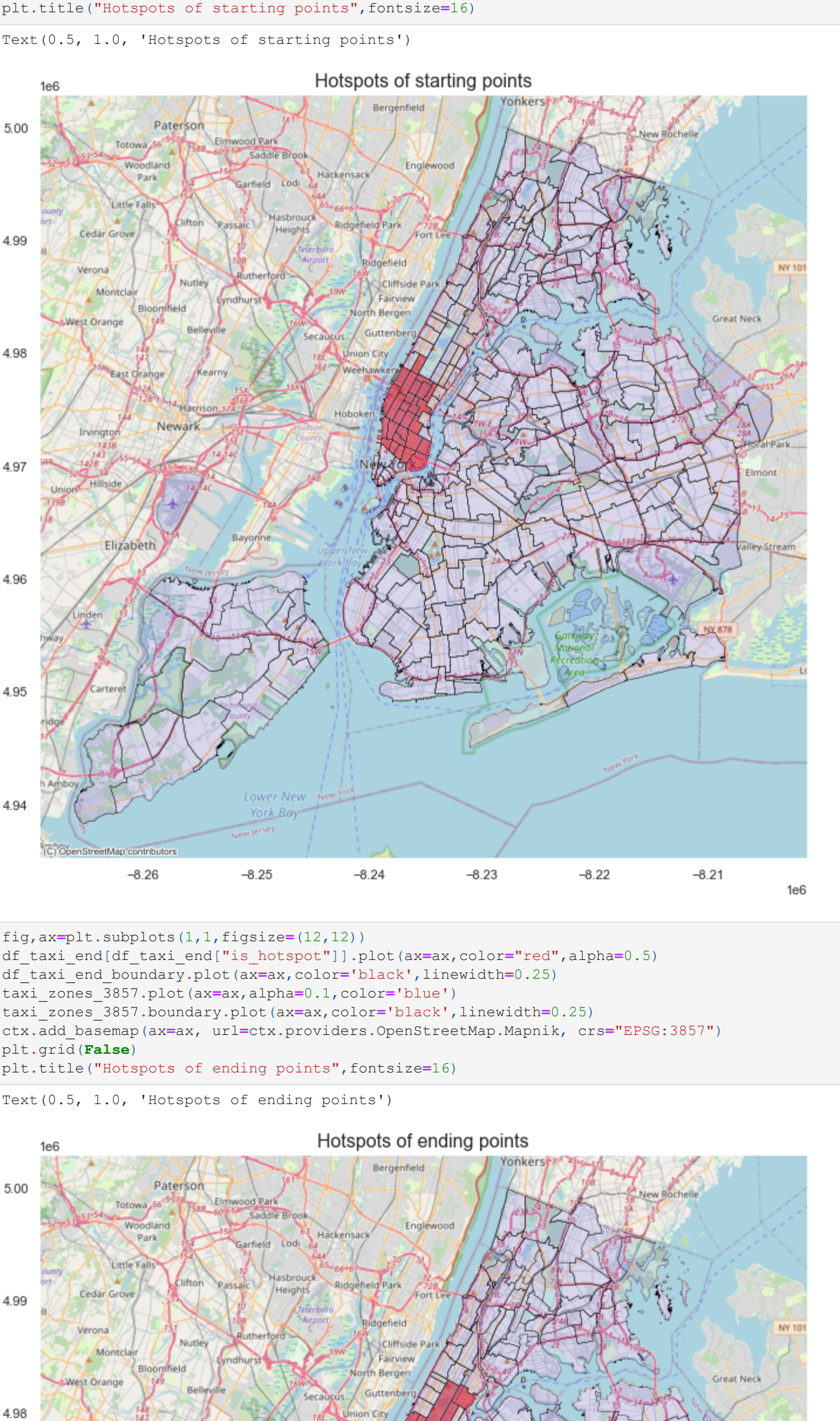
In [182…
```
sig_start = li_start.p_sim < 0.05
hotspot_start = sig_start * li_start.q==1
sig_end = li_end.p_sim < 0.05
hotspot_end = sig_end * li_end.q==1
```

In [183…
```
df_taxi_start["is_hotspot"]=hotspot_start
df_taxi_end["is_hotspot"]=hotspot_end
```

In [186…
```
df_taxi_start_boundary=df_taxi_start.boundary.to_crs("EPSG:3857")
df_taxi_end_boundary=df_taxi_end.boundary.to_crs("EPSG:3857")
```

In [202…
```
fig,ax=plt.subplots(1,1,figsize=(12,12))
df_taxi_start[df_taxi_start["is_hotspot"]].plot(ax=ax,color="red",alpha=0.5)
df_taxi_start_boundary.plot(ax=ax,color='black',linewidth=0.25)
taxi_zones_3857.plot(ax=ax,alpha=0.1,color='blue')
taxi_zones_3857.boundary.plot(ax=ax,color='black',linewidth=0.25)
ctx.add_basemap(ax=ax, url=ctx.providers.OpenStreetMap.Mapnik, crs="EPSG:3857")
plt.grid(False)
plt.title("Hotspots of starting points",fontsize=16)
```

Out[202]:
```
Text(0.5, 1.0, 'Hotspots of starting points')
```



In [198…
```
fig,ax=plt.subplots(1,1,figsize=(12,12))
df_taxi_end[df_taxi_end["is_hotspot"]].plot(ax=ax,color="red",alpha=0.5)
df_taxi_end_boundary.plot(ax=ax,color='black',linewidth=0.25)
taxi_zones_3857.plot(ax=ax,alpha=0.1,color='blue')
taxi_zones_3857.boundary.plot(ax=ax,color='black',linewidth=0.25)
ctx.add_basemap(ax=ax, url=ctx.providers.OpenStreetMap.Mapnik, crs="EPSG:3857")
plt.grid(False)
plt.title("Hotspots of ending points",fontsize=16)
```

Out[198]:
```
Text(0.5, 1.0, 'Hotspots of ending points')
```

In [ ]: