Spatial Auto Correlation

March 8, 2021

0.1 Spatial Auto Correlation: Group Assignment 4

0.1.1 Load the Libraries

First step is load libraries needed for data wrangling, spatial statistics, basemaps, and graphics.

[1]:
```
import pandas as pd
import geopandas as gpd
import contextily as ctx
import esda
from esda.moran import Moran, Moran_Local
import splot
from splot.esda import moran_scatterplot, plot_moran,␣
↪lisa_cluster,plot_moran_simulation
import libpysal as lps
import matplotlib.pyplot as plt
import plotly.express as px
```

```
/opt/conda/lib/python3.8/site-packages/geopandas/_compat.py:106: UserWarning:
The Shapely GEOS version (3.8.1-CAPI-1.13.3) is incompatible with the GEOS
version PyGEOS was compiled with (3.9.0-CAPI-1.16.2). Conversions between both
will be slow.
  warnings.warn(
```

Next, I'll upload my data and clean it a bit.

0.1.2 Upload and Clean Data

I am uploading the Water Bill Debt data I've been working with all quarter as well as shapefiles of ZIP codes across the state of California.

[2]:
```
acs = pd.read_csv('Data/Updated Bill Data 2_22.csv')

zips = gpd.read_file('Data/ca_california_zip_codes_geo.min.json')
```

Now that the data is uploaded, I need to clean and merge the two datasets. I'll start with the Water Bill data. First, I only want to keep the columns of interest.

[3]:
```
refined_columns = ['Zip Codes',
 'Sum of Less than $100',
```

```
 'Sum of $100-$200',
 'Sum of $200-$300',
 'Sum of $300-$400',
 'Sum of $400-$500',
 'Sum of $500-$600',
 'Sum of $600-$700',
 'Sum of $700-$800',
 'Sum of $800-$900',
 'Sum of $900-$1000',
 'Sum of More than $1000',
 'Sum of Total number of delinquent residential accounts',
 'pop',
 'mhhi',
 'pct_nhw',
 'pct_black',
 'pct_hisp',
 'pct_asian',
 'pct_povt',
 'pct_overcrowded',
 'pct_no_veh_hh',
 'pct_broadband',
 'pct_no_broadband',
 'pct_uninsured_19_64',
 'pct_noncitizen',
 'pct_immigrants',
 'pct_lep_hh',
 'pct_no_hins',
 '% Renter Pop',
  '% Owner Pop']
```

[4]:

```
acs = acs[refined_columns]
```

Now that I have only the columns I want, I will create a for loop to standardize the data into percents.

[5]:

```
acs['Percent Delinquent'] = acs['Sum of Total number of delinquent residential␣
 ↪accounts']/acs['pop']*100


pct_debt_buckets = ['Percent Less than $100', 'Percent $100-$200', 'Percent␣
 ↪$200-$300' ,
                 'Percent $300-$400', 'Percent $400-$500', 'Percent $500-$600',␣
 ↪'Percent $600-$700',
                 'Percent $700-$800', 'Percent $800-$900', 'Percent $900-$1000',␣
 ↪'Percent More than $1000']
```

```
debt_buckets =  ['Sum of Less than $100', 'Sum of $100-$200', 'Sum of␣
 ↪$200-$300' ,
                'Sum of $300-$400', 'Sum of $400-$500', 'Sum of $500-$600',␣
 ↪'Sum of $600-$700',
                'Sum of $700-$800', 'Sum of $800-$900', 'Sum of $900-$1000',␣
 ↪'Sum of More than $1000']

sum_total = 'Sum of Total number of delinquent residential accounts'

demographics = ['pct_nhw','pct_black',␣
 ↪'pct_hisp','pct_asian','pct_povt','pct_overcrowded','pct_no_veh_hh',
                'pct_broadband','pct_no_broadband',␣
 ↪'pct_uninsured_19_64','pct_noncitizen','pct_immigrants',
                'pct_lep_hh','pct_no_hins','% Renter Pop','% Owner Pop']

for pct, debt in zip(pct_debt_buckets, debt_buckets):
    acs[pct] = acs[debt] / acs[sum_total]*100

for dem in demographics:
    acs[dem] = acs[dem]*100
```

Now that the debt buckets and demographics have been standardized, I'll drop the columns I no longer need.

[6]:
```
columns_drop = [
 'Sum of $100-$200',
 'Sum of $200-$300',
 'Sum of $300-$400',
 'Sum of $400-$500',
 'Sum of $600-$700',
 'Sum of $700-$800',
 'Sum of $800-$900',
 'Sum of $900-$1000',]

acs = acs.drop(columns_drop, axis = 1)
```

I also know from experience working with this data that there are a few lines of "NaN" values that might hinder further analysis. I will remove those as well.

[7]:
```
acs = acs.dropna()
```

I also know from experience with this data there are a few outliers in the Percent Delinquent column, so I will remove those too.

[8]:
```
acs.drop(acs[acs['Percent Delinquent'] > 100].index, inplace = True)
```

Now I will move on to the Zip code shapefiles. I need to clean this data and merge it with the debt data. Fist, I need to get a sense of which columns I need to keep.

[9]:
```
zips.head()
```

[9]:
```
  STATEFP10 ZCTA5CE10  GEOID10 CLASSFP10 MTFCC10 FUNCSTAT10   ALAND10  \
0        06     94601  0694601        B5   G6350          S   8410939
1        06     94501  0694501        B5   G6350          S  20539466
2        06     94560  0694560        B5   G6350          S  35757865
3        06     94587  0694587        B5   G6350          S  51075108
4        06     94580  0694580        B5   G6350          S   8929836

   AWATER10   INTPTLAT10    INTPTLON10 PARTFLG10  \
0    310703  +37.7755447  -122.2187049         N
1   9005303  +37.7737968  -122.2781230         N
2     60530  +37.5041413  -122.0323587         N
3         0  +37.6031556  -122.0186382         N
4     17052  +37.6757312  -122.1330170         N

                                            geometry
0  POLYGON ((-122.22717 37.79197, -122.22693 37.7…
1  POLYGON ((-122.29181 37.76301, -122.30661 37.7…
2  POLYGON ((-122.05499 37.54959, -122.05441 37.5…
3  POLYGON ((-122.06515 37.60485, -122.06499 37.6…
4  POLYGON ((-122.12999 37.68445, -122.12995 37.6…
```

I only need the Zip Code column (ZCTA5CE10) and the geometry column. So, I will keep those and save them as the old dataframe.

[10]:
```
zips_keep = ['ZCTA5CE10', 'geometry']
```

[11]:
```
zips = zips[zips_keep]
```

I need to rename the Zip Code columns so it matches the name on the ACS file so I can merge the two datasets together.

[12]:
```
zips.columns = ['Zip Codes', 'geometry']
```

Now that both datasets are clean and they have one column that matches, I can merge them together.

[13]:
```
merged = acs.merge(zips,
                   on='Zip Codes')
```

I also need to convert my dataset back to a geodataframe.

[14]:
```
merged = gpd.GeoDataFrame(merged, geometry='geometry')
```

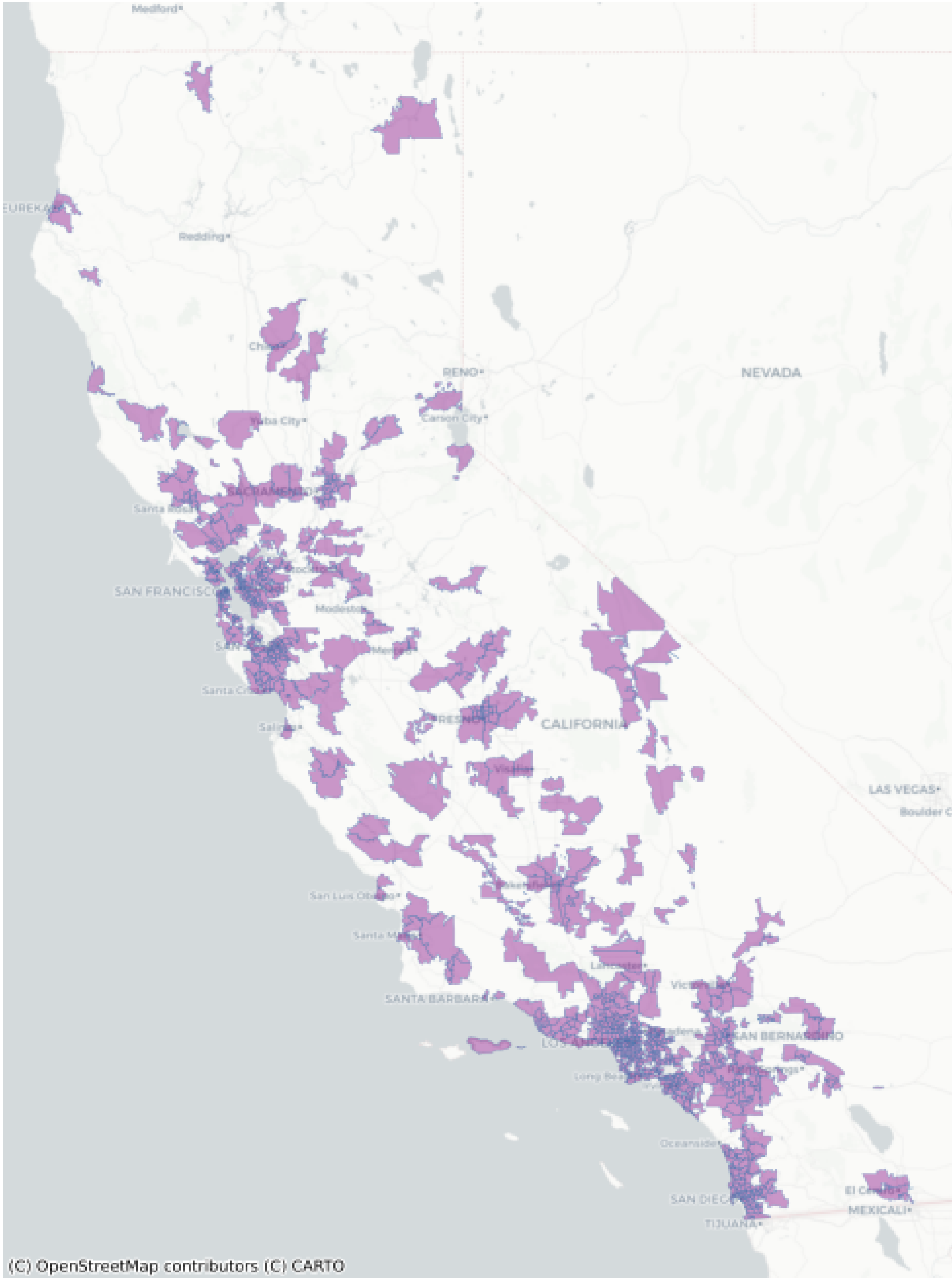
0.1.3 Create Maps

First I get a web mercator projection

[15]:
```
merged = merged.to_crs(epsg=3857)
```

Next, I create a basemap of the Zip Codes.

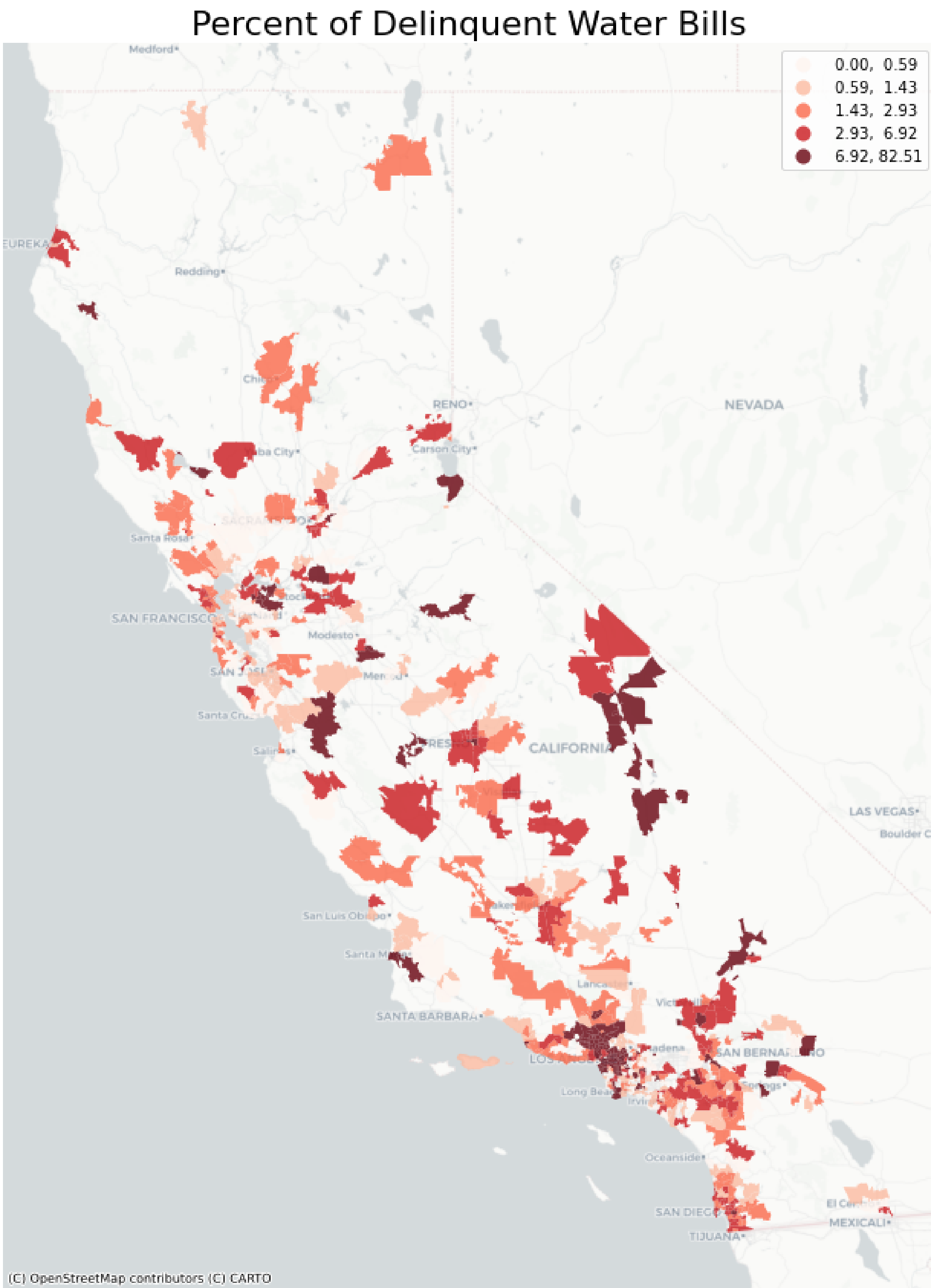
[16]:
```
fig, ax = plt.subplots(figsize=(12,12))

merged.plot(ax=ax,
         color='purple',
         edgecolor='steelblue',
         lw=0.5,
         alpha=0.4)


ax.axis('off')


ctx.add_basemap(ax,source=ctx.providers.CartoDB.Positron)
```



Next, I want to map percent of water bill debt in quantiles, by zip code.

[88]:

```
fig,ax = plt.subplots(figsize=(15,15))

merged.plot(ax=ax,
        column='Percent Delinquent',
        legend=True,
        alpha=0.8,
        cmap='Reds',
        scheme='quantiles')

ax.axis('off')
ax.set_title('Percent of Delinquent Water Bills',fontsize=22)
ctx.add_basemap(ax,source=ctx.providers.CartoDB.Positron)
```



Next, I want to calculate the spatial lag based on the percentage of delinquent accounts.

[21]:
```
wq =  lps.weights.KNN.from_dataframe(merged,k=8)

wq.transform = 'r'

merged['percent_delinquent_lag'] = lps.weights.lag_spatial(wq, merged['Percent␣
↪Delinquent'])
```

Then I want to create a colum that shows the difference between the percent delinquent and the spatial lag.

[24]:
```
merged['delinquent_lag_diff'] = merged['Percent Delinquent'] -␣
↪merged['percent_delinquent_lag']
```

Now that I have created the new column, I want to sort values and see where the largest differece in spatial lag lies.

[43]:
```
pd.set_option('display.max_columns', None)

merged.sort_values(by='delinquent_lag_diff')
```

[43]:
```
     Zip Codes  Sum of Less than $100  Sum of $500-$600  \
72       90255                  427.0              12.0
435      93010                   42.0               1.0
434      93004                   43.0              10.0
199      91505                   13.0               2.0
446      93110                    5.0               3.0
..         …                      …                 …
484      93455                 7660.0              88.0
749      95422                 1817.0              64.0
335      92397                 1217.0               7.0
615      94569                   41.0               0.0
440      93035                  176.0              26.0

     Sum of More than $1000  \
72                     14.0
435                    12.0
434                     6.0
199                    21.0
446                     7.0
..                      …
484                    52.0
749                    41.0
335                     6.0
615                     0.0
440                    92.0

     Sum of Total number of delinquent residential accounts      pop  \
72                                                   982.0   75560.0
```

435	201.0	45092.0
434	275.0	30502.0
199	74.0	30680.0
446	34.0	16660.0
..	…	…
484	13942.0	45116.0
749	5584.0	15550.0
335	1775.0	4571.0
615	81.0	205.0
440	24261.0	29404.0

	mhhi	pct_nhw	pct_black	pct_hisp	pct_asian	pct_povt \
72	42581.0	1.467708	1.012440	96.913711	0.600847	24.302688
435	92045.0	55.723853	2.408409	28.965227	8.941719	7.587329
434	82240.0	54.698053	2.724412	36.115665	4.658711	8.479609
199	84651.0	54.374185	2.956323	25.296610	12.441330	9.584832
446	81429.0	63.103241	1.950780	26.878751	5.138055	10.213287
..	…	…	…	…	…	…
484	84404.0	55.346219	1.256760	34.841298	4.603688	6.925473
749	29069.0	60.990354	5.003215	28.688103	0.186495	33.888817
335	61496.0	72.019252	0.218771	22.445854	0.700066	17.763158
615	153750.0	100.000000	0.000000	0.000000	0.000000	0.000000
440	86630.0	46.908584	2.775133	42.460210	4.961910	7.334813

	pct_overcrowded	pct_no_veh_hh	pct_broadband	pct_no_broadband \
72	15.140541	12.400000	71.345946	28.654054
435	1.025446	5.158881	89.821496	10.178504
434	0.875092	5.646647	86.256448	13.743552
199	0.997273	5.259057	85.734320	14.265680
446	0.251651	4.608367	86.520919	13.479081
..	…	…	…	…
484	1.154477	2.512299	88.265005	11.734995
749	0.146843	12.204275	67.564040	32.435960
335	1.390498	0.000000	84.820394	15.179606
615	0.000000	0.000000	100.000000	0.000000
440	1.021839	0.921659	88.639551	11.360449

	pct_uninsured_19_64	pct_noncitizen	pct_immigrants	pct_lep_hh \
72	26.321071	31.283748	47.301482	27.800000
435	6.876302	6.327065	15.262131	2.487657
434	6.851771	4.976723	13.389286	4.384672
199	8.055883	8.497392	24.240548	4.885080
446	5.099778	6.056423	14.597839	3.759044
..	…	…	…	…
484	8.001897	5.443745	12.345953	2.787799
749	16.135755	6.765273	9.646302	3.768967
335	8.533917	3.565959	6.125574	0.695249

```
615           16.585366        0.000000        0.000000    0.000000
440            7.928006        6.182832       15.875391    3.145662

     pct_no_hins  % Renter Pop  % Owner Pop  Percent Delinquent  \
72     18.937687     61.248015    38.751985            1.299629
435     5.079661     37.776102    62.223898            0.445755
434     5.925536     33.069963    66.930037            0.901580
199     6.346719     44.217731    55.782269            0.241199
446     4.156196     31.368547    68.631453            0.204082
..           …            …            …                 …
484     6.622679     30.040340    69.959660           30.902562
749    10.801551     50.372990    49.627010           35.909968
335     5.338000     22.117699    77.882301           38.831765
615    16.585366     87.804878    12.195122           39.512195
440     6.902673     41.419535    58.580465           82.509182

     Percent Less than $100  Percent $100-$200  Percent $200-$300  \
72                43.482688          31.975560          10.285132
435               20.895522          37.313433          17.910448
434               15.636364          33.454545          25.454545
199               17.567568           9.459459           9.459459
446               14.705882          14.705882          11.764706
..                      …                  …                  …
484               54.941902          32.592168           7.064984
749               32.539398          24.391117           6.518625
335               68.563380          18.985915           6.084507
615               50.617284          41.975309           3.703704
440                0.725444           1.916656           0.478134

     Percent $300-$400  Percent $400-$500  Percent $500-$600  \
72            5.193483           4.073320           1.221996
435           8.457711           3.482587           0.497512
434          11.272727           4.363636           3.636364
199          10.810811           6.756757           2.702703
446          14.705882           5.882353           8.823529
..                 …                  …                  …
484           2.474537           1.219337           0.631186
749           3.492120           1.952006           1.146132
335           2.873239           1.408451           0.394366
615           2.469136           0.000000           0.000000
440           0.259676           0.189605           0.107168

     Percent $600-$700  Percent $700-$800  Percent $800-$900  \
72            1.018330           0.610998           0.203666
435           2.487562           0.497512           1.990050
434           0.727273           2.181818           0.727273
199           2.702703           4.054054           5.405405
```

```
446           2.941176           5.882353           0.000000
..                 …                  …                  …
484           0.322766           0.179314           0.100416
749           0.841691           0.447708           0.268625
335           0.507042           0.450704           0.281690
615           1.234568           0.000000           0.000000
440           0.094802           0.041218           0.041218

     Percent $900-$1000  Percent More than $1000  \
72             0.509165                 1.425662
435            0.497512                 5.970149
434            0.363636                 2.181818
199            2.702703                28.378378
446            0.000000                20.588235
..                  …                       …
484            0.100416                 0.372974
749            0.214900                 0.734241
335            0.112676                 0.338028
615            0.000000                 0.000000
440            0.078315                 0.379209

                                              geometry  \
72   POLYGON ((-13161961.629 4026170.260, -13161964…
435  POLYGON ((-13252229.491 4062635.596, -13252225…
434  POLYGON ((-13267617.518 4063417.954, -13267592…
199  MULTIPOLYGON (((-13174515.573 4056583.401, -13…
446  POLYGON ((-13334876.197 4088290.630, -13334875…
..                                                 …
484  POLYGON ((-13406884.101 4152561.331, -13406863…
749  POLYGON ((-13654812.755 4720710.325, -13654811…
335  POLYGON ((-13091768.122 4080344.489, -13091729…
615  POLYGON ((-13600421.718 4584508.980, -13600452…
440  POLYGON ((-13273391.660 4051973.347, -13273590…

     percent_delinquent_lag  delinquent_lag_diff
72                13.255126           -11.955497
435               12.146177           -11.700421
434               11.943624           -11.042044
199               11.153436           -10.912237
446               11.055213           -10.851131
..                      …                    …
484                1.526552            29.376011
749                3.600366            32.309602
335                2.657683            36.174082
615                2.702154            36.810042
440                1.742674            80.766509
```

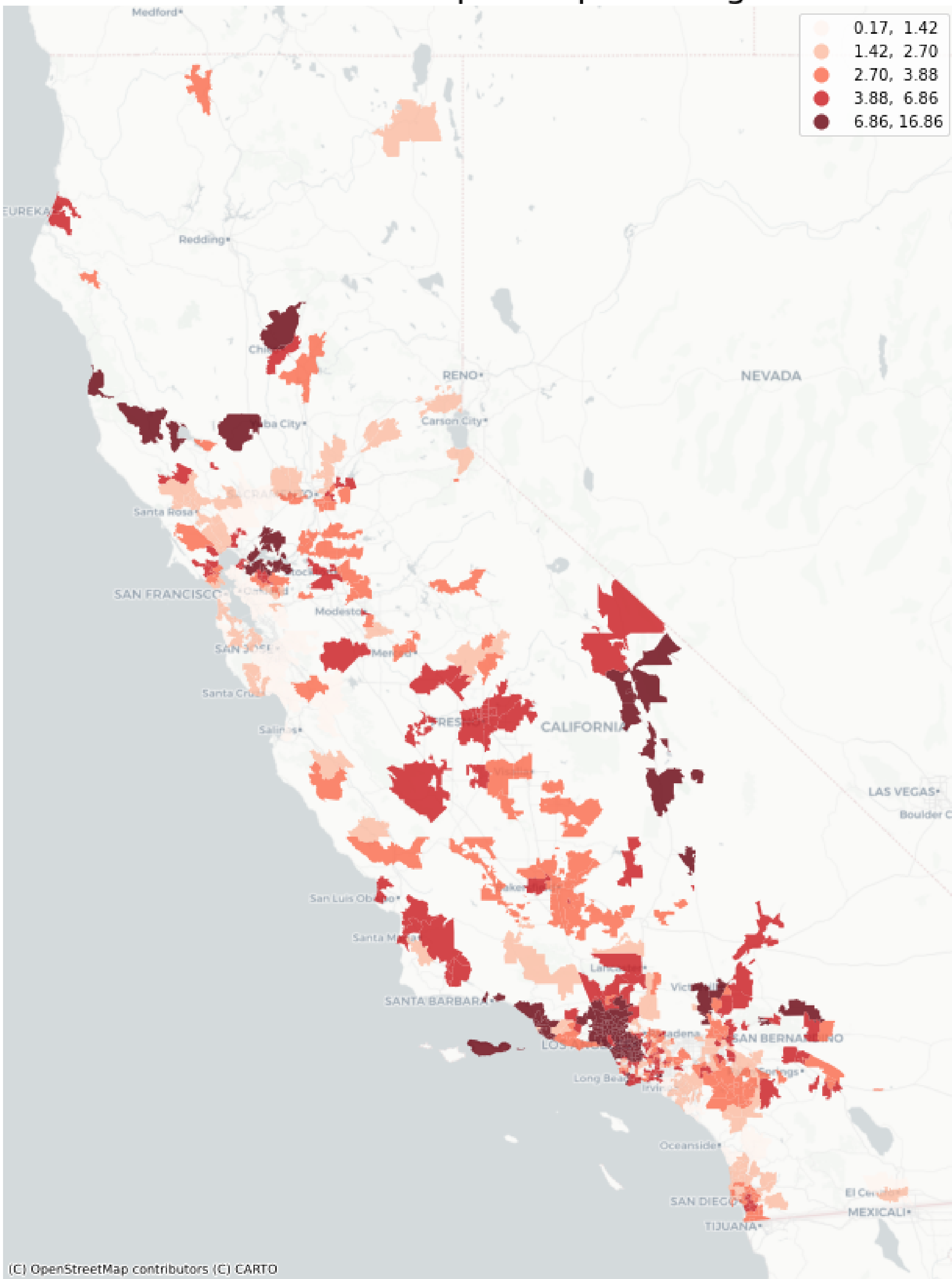
```
[814 rows x 38 columns]
```

One zip code in partcular has a massive spatial lag difference at 80.76, 93035, which is located in Oxnard, CA. This zip code also has the highest percent delinquent population of any other zip code in the state.

My next step is to map the spatial lag.

[86]:
```
fig, ax = plt.subplots(figsize=(15, 15))

merged.plot(ax=ax,
         figsize=(15,15),
         column='percent_delinquent_lag',
         legend=True,
         alpha=0.8,
         cmap='Reds',
         scheme='quantiles')

ax.axis('off')
ax.set_title('Percent Delinquent Spatial Lag',fontsize=22)

ctx.add_basemap(ax,source=ctx.providers.CartoDB.Positron)
```

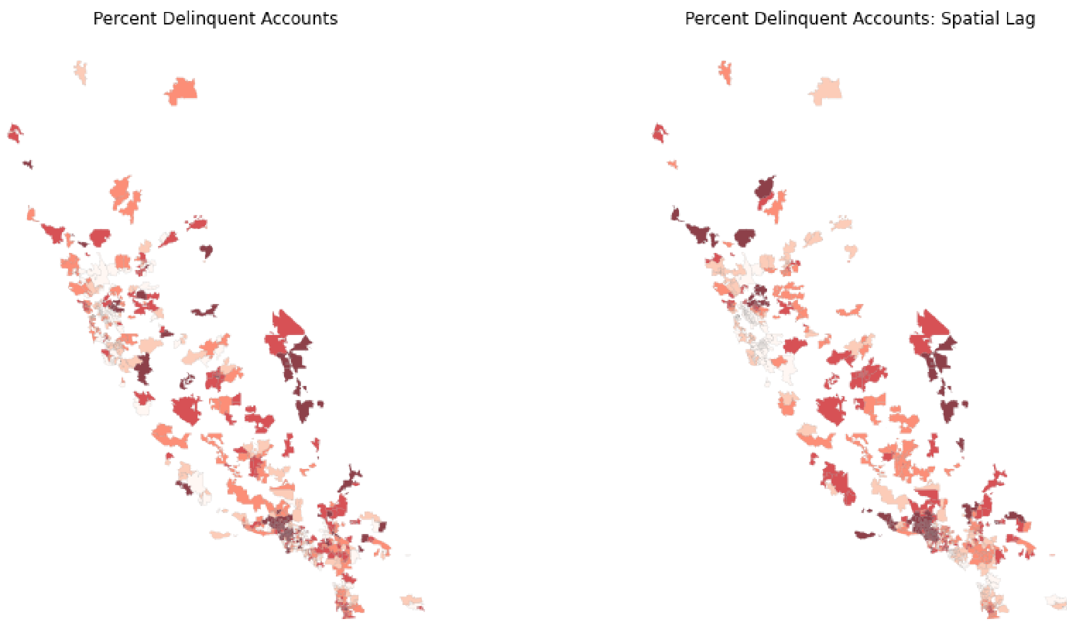
Percent Delinquent Spatial Lag



The spatial lag map seems to have similar trends as the percent delinquent map, but I want to compare them side by side.

[87]:

```
fig, ax = plt.subplots(1, 2, figsize=(15, 8))

merged.plot(ax=ax[0],
         column='Percent Delinquent',
         cmap='Reds',
         scheme='quantiles',
         k=5,
         edgecolor='gray',
         linewidth=0.1,
         alpha=0.75,
           )


ax[0].axis("off")
ax[0].set_title("Percent Delinquent Accounts")

merged.plot(ax=ax[1],
         column='percent_delinquent_lag',
         cmap='Reds',
         scheme='quantiles',
         k=5,
         edgecolor='gray',
         linewidth=0.1,
         alpha=0.75
           )

ax[1].axis("off")
ax[1].set_title("Percent Delinquent Accounts: Spatial Lag")

plt.show()
```



It seems that the spatial lag map shows more zip codes on the higher end of the spatial lag scale in the northern part of CA (north of SF and Sacramento) and a larger cluster of red and orange around LA.

It is a little easier to see the trends in delinquent accounts in and around LA, north of SF/Sacramento, near San Barnardino, and sort of near the Death Valley area.

0.1.4 Moran's Plot

The next step is the quantify the degree of the correlations. To start, I will calculate the Moran's I value.

[56]:
```
merged_web = merged.to_crs('EPSG:4326')
```

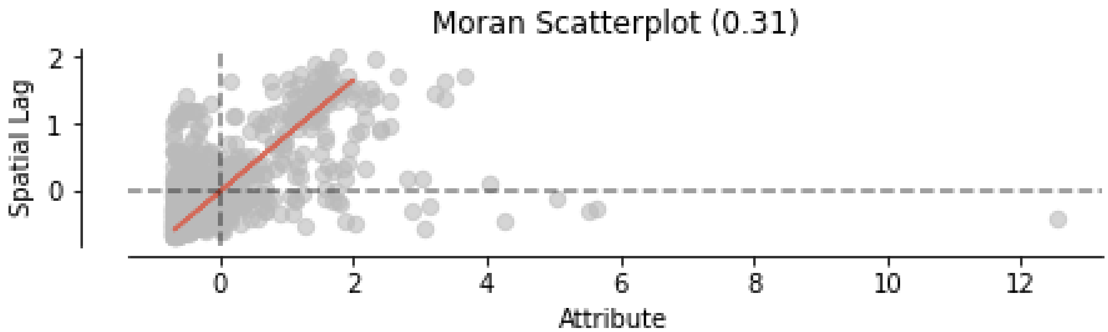
[61]:
```
y = merged['Percent Delinquent']
moran = Moran(y, wq)
moran.I
```

[61]: 0.30961259338574426

The positive Moran's I value indicates a positive autocorrelation, meaning that "high values are close to high values, and/or low values are close to low values."

I also will create a scatterplot with this value.

[58]:
```
fig, ax = moran_scatterplot(moran, aspect_equal=True)
plt.show()
```



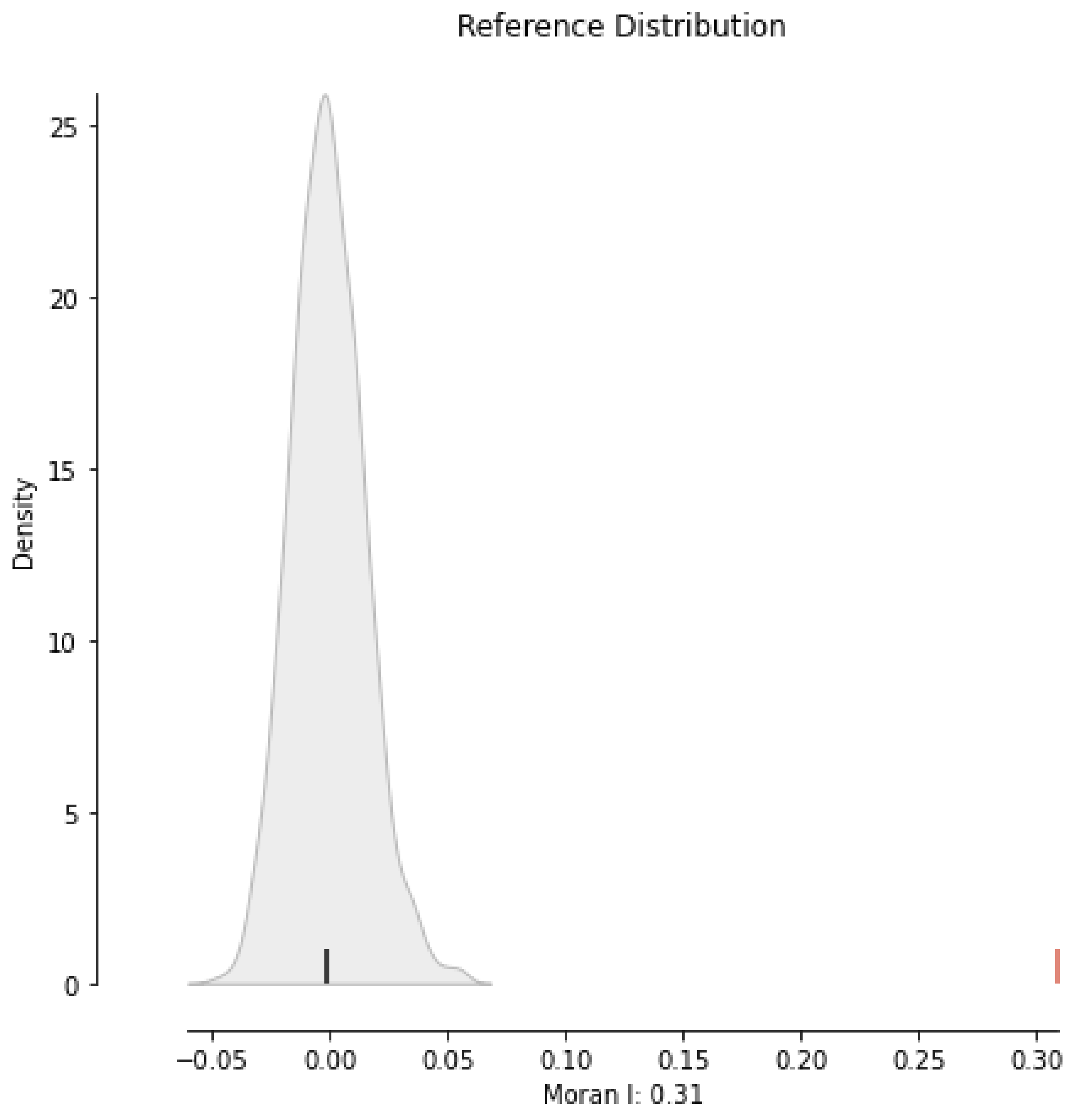
Another way to think about the Moran's I value is that it is the slope of the Percent Delinquent and Percent Delinquent Spatial Lag columns in the data.

Next, we need to determine how likely it is that this spatial configuration would occur on a map entirely randomly. To determine this likelihood, we must plot the distribution of 999 random simulations.

[59]:
```
plot_moran_simulation(moran,aspect_equal=False)
```

```
/opt/conda/lib/python3.8/site-packages/splot/_viz_esda_mpl.py:47:
MatplotlibDeprecationWarning:
The set_smart_bounds function was deprecated in Matplotlib 3.2 and will be
removed two minor releases later.
  ax.spines['left'].set_smart_bounds(True)
/opt/conda/lib/python3.8/site-packages/splot/_viz_esda_mpl.py:48:
MatplotlibDeprecationWarning:
The set_smart_bounds function was deprecated in Matplotlib 3.2 and will be
removed two minor releases later.
  ax.spines['bottom'].set_smart_bounds(True)
```

[59]:
```
(<Figure size 504x504 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f1ed6bdad30>)
```



We can see from the plot where our Moran's value has been plotted near 0.30 on the y axis that it is so far from the other simulations, there is no way it occured randomly.

Next, I will calculate a p value.

[62]:
```
moran.p_sim
```

[62]:
```
0.001
```

The p value is low, indicating significance. Here we can reject the hypothesis that the map is random. Now we know there is a positive spatial autocorrelation between percentage of delinquent water bill accounts by zip code and their location in California.

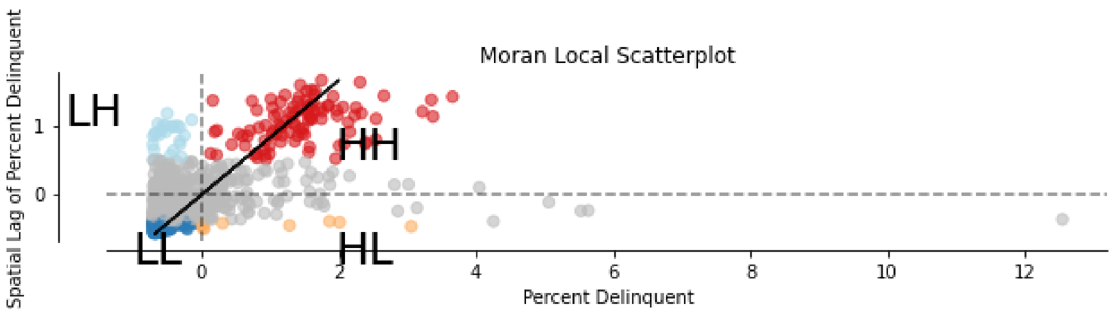
0.1.5 Moral Local Scatterplot

First, we calculate local indicators of spatial association

[63]:
```
lisa = esda.moran.Moran_Local(y, wq)
```

Next, we plot those values

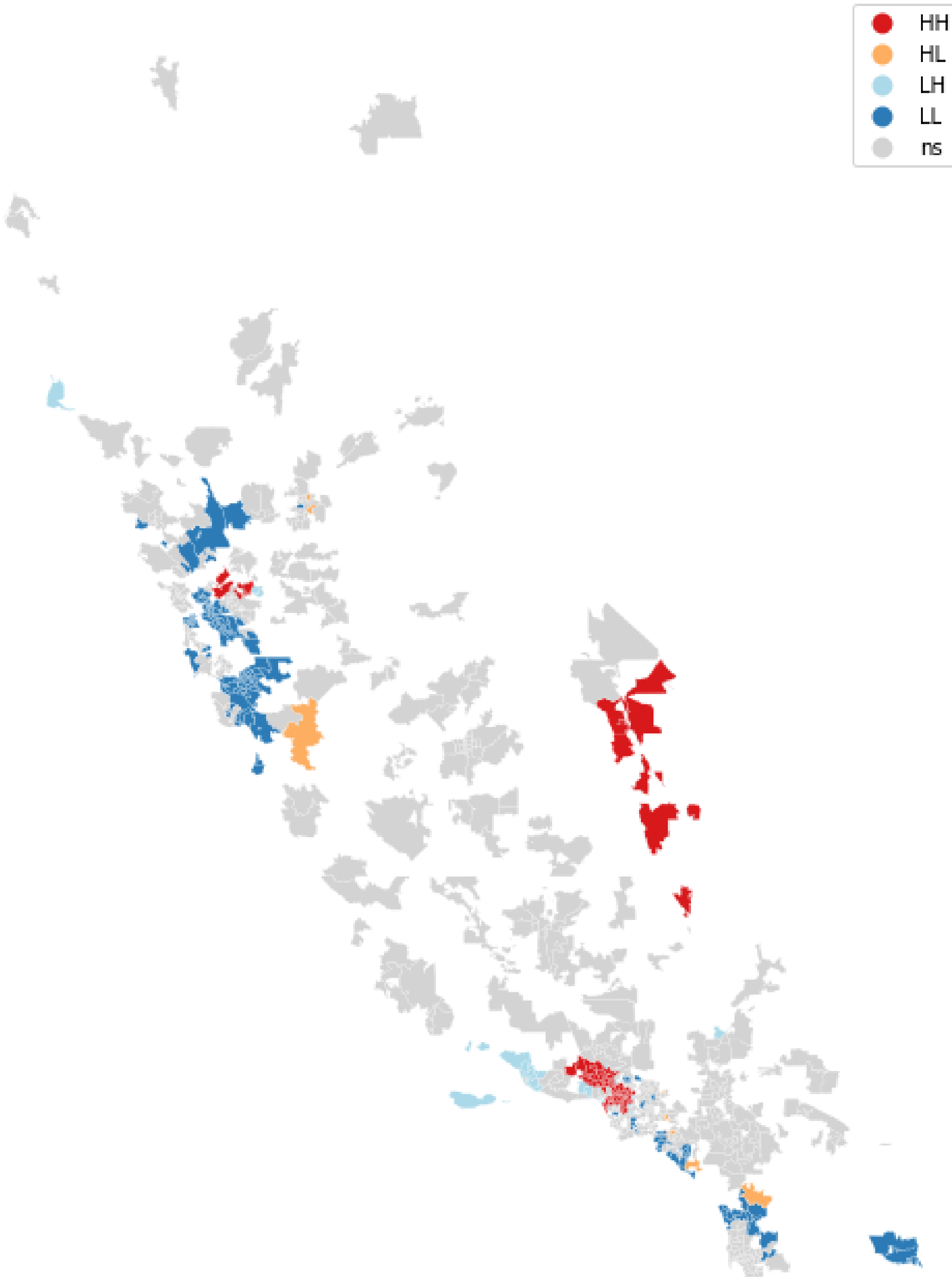
[64]:
```
fig,ax = plt.subplots(figsize=(10,15))

moran_scatterplot(lisa, ax=ax, p=0.05)
ax.set_xlabel("Percent Delinquent")
ax.set_ylabel('Spatial Lag of Percent Delinquent')


plt.text(1.95, 0.5, "HH", fontsize=20)
plt.text(1.95, -1, "HL", fontsize=20)
plt.text(-2, 1, "LH", fontsize=20)
plt.text(-1, -1, "LL", fontsize=20)
plt.show()
```



The scatterplot shows statistically significant, autocorrelated zip codes.

We can also map these results.

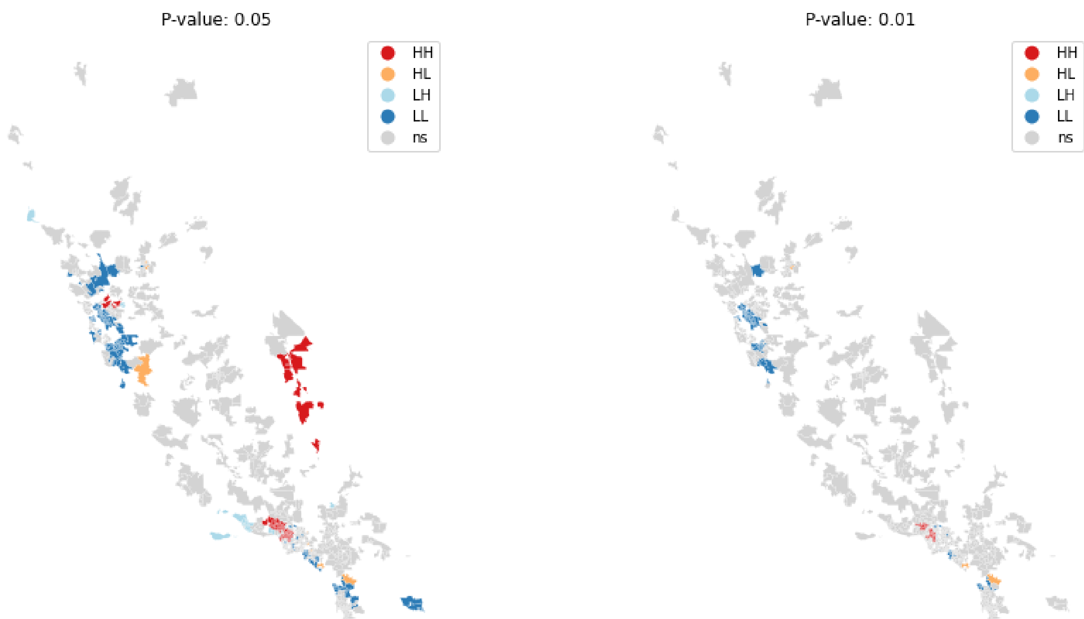
[66]:
```
fig, ax = plt.subplots(figsize=(14,12))
lisa_cluster(lisa, merged, p=0.05, ax=ax)
plt.show()
```



Above we can see the high delinquency trends near LA and Dealth Valley and the low delinquency trends in the SF/Sacramento Area.

We can also plot to compare statistical sigificance between p values.

[67]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 8))


lisa_cluster(lisa, merged, p=0.05, ax=ax[0])

ax[0].axis("off")
ax[0].set_title("P-value: 0.05")


lisa_cluster(lisa, merged, p=0.01, ax=ax[1])
ax[1].axis("off")
ax[1].set_title("P-value: 0.01")

plt.show()
```



It seems there are fewer localities that have a p value of 0.01, but the cluster of low delinquency is still present in northern CA and there are a few zip codes in LA that still have high delinquency. Next, I want to add additional demographic factors to my analysis to determine how they impact the significance of high vs low delinquency in these key areas of the map.

[ ]:
```
```