In [3]:
```
#import stats packages
import pandas as pd
import statsmodels.api as sm
from numpy import random
import numpy as np

#import plot packages
import plotly.express as plt
import geopandas as gpd
import plotly.graph_objects as go
import matplotlib.pyplot as mp
from shapely.geometry import Point
```

In [4]:
```
#opening files
cities = pd.read_csv('pset3_cities.csv')
distances = pd.read_csv('pset3_distances.csv')
stations = pd.read_csv('pset3_stations.csv')
lines = pd.read_csv('pset3_lines.csv')
```

In [5]:
```
#creating deltalines

stations = stations.merge(lines, how = 'left')
stations['year_opening'] = stations['year_opening'].replace([2017, 201
stations = stations.dropna()


stationsdum = pd.get_dummies(stations['cityid'])

cities['numstat'] = cities['cityid']
cities['numlinks'] = cities['cityid']
cities['avgspd'] = cities['cityid']
a = []

for i in cities['cityid']:

    if i in stationsdum.columns:
        cities['numstat'] = cities['numstat'].replace(i, stationsdum[i

    else:
        cities['numstat'] = cities['numstat'].replace(i, 0)
```

In [6]:
```
#creating log distance to nearest station

cities['logdist'] = cities['cityid']
for i in list(set(list(cities['cityid']))):

    trunc = distances[distances['cityid1']==i]
    if cities[cities['cityid']==i]['numstat'].mean() > 0:
        cities['logdist'] = cities['logdist'].replace(i, trunc[trunc['
    else:
        citiestr = cities[cities['numstat'] > 0]
        df = pd.DataFrame(columns = ['cityid1', 'cityid2', 'dist'])
        for a in list(set(list(citiestr['cityid']))):
            df = df.append(trunc[trunc['cityid2'] == a])
        cities['logdist'] = cities['logdist'].replace(i, df['dist'].mi
```

In [10]:
```
#re-initiating stations
stations = pd.read_csv('pset3_stations.csv')
```

In [11]:
```
#converting to logs
cities['logdist'] = np.log(cities['logdist'])
```

In [12]:
```
#A no FE no constant

citiesempt = cities.dropna(subset = ['empgrowth'])

treat = ['logdist']

vals = sm.OLS(citiesempt['empgrowth'], citiesempt[treat])
out = vals.fit(cov_type = 'HC0')
print(out.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:              empgrowth   R-squared (uncentered):
0.460
Model:                            OLS   Adj. R-squared (uncentered):
0.458
Method:                 Least Squares   F-statistic:
211.8
Date:                Sat, 18 Nov 2023   Prob (F-statistic):
6.12e-36
Time:                        20:12:06   Log-Likelihood:
-31.535
No. Observations:                 275   AIC:
65.07
Df Residuals:                     274   BIC:
68.69
Df Model:                           1
Covariance Type:                  HC0
==============================================================================
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
logdist        0.0580      0.004     14.554      0.000       0.050
0.066
==============================================================================
Omnibus:                       13.243   Durbin-Watson:
1.332
Prob(Omnibus):                  0.001   Jarque-Bera (JB):
25.376
Skew:                          -0.222   Prob(JB):
3.09e-06
Kurtosis:                       4.421   Cond. No.
1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model doe
s not contain a constant.
[2] Standard Errors are heteroscedasticity robust (HC0)
```

In [13]:
```
#A no FE with constant

citiesempt = cities.dropna(subset = ['empgrowth'])

treat = ['logdist']

vals = sm.OLS(citiesempt['empgrowth'], sm.add_constant(citiesempt[trea
out = vals.fit(cov_type = 'HC0')
print(out.summary())

FEs = out.predict(sm.add_constant(citiesempt[treat]))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              empgrowth   R-squared:                       0.119
Model:                            OLS   Adj. R-squared:                  0.115
Method:                 Least Squares   F-statistic:                     23.45
Date:                Sat, 18 Nov 2023   Prob (F-statistic):           2.15e-06
Time:                        20:12:16   Log-Likelihood:                 2.1976
No. Observations:                 275   AIC:                           -0.3952
Df Residuals:                     273   BIC:                             6.838
Df Model:                           1
Covariance Type:                  HC0
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.8526      0.119      7.158      0.000       0.619       1.086
logdist       -0.1373      0.028     -4.842      0.000      -0.193      -0.082
==============================================================================
Omnibus:                        6.982   Durbin-Watson:                   1.606
Prob(Omnibus):                  0.030   Jarque-Bera (JB):                9.296
Skew:                           0.181   Prob(JB):                      0.00958
Kurtosis:                       3.825   Cond. No.                         30.6
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC0)
```

In [14]:
```
#A with FE
regions = pd.get_dummies(cities['province_en'])
cities = pd.concat([cities, regions], axis = 1)

citiesempt = cities.dropna(subset = ['empgrowth'])

treat = ['logdist']
dummies = list(set(list(citiesempt['province_en'])))

vals = sm.OLS(citiesempt['empgrowth'], citiesempt[treat+dummies])
out = vals.fit(cov_type = 'HC0')
print(out.summary())

FEs = out.predict(citiesempt[treat+dummies])
```

```
                            OLS Regression Results
==============================================================================
=========
Dep. Variable:              empgrowth   R-squared:
0.460
Model:                            OLS   Adj. R-squared:
0.393
Method:                 Least Squares   F-statistic:
nan
Date:                Sat, 18 Nov 2023   Prob (F-statistic):
nan
Time:                        20:12:17   Log-Likelihood:
69.538
No. Observations:                 275   AIC:
-77.08
Df Residuals:                     244   BIC:
35.04
Df Model:                          30
Covariance Type:                  HC0
==============================================================================
=============
                     coef    std err          z      P>|z|      [0.02
5      0.975]
------------------------------------------------------------------------------
-------------
logdist           -0.0812      0.023     -3.472      0.001     -0.12
7      -0.035
tibet              0.4938      0.083      5.942      0.000      0.33
1       0.657
tibet              0.4938      0.083      5.942      0.000      0.33
1       0.657
jilin              0.2667      0.062      4.312      0.000      0.14
5       0.388
jilin              0.2667      0.062      4.312      0.000      0.14
5       0.388
inner mongolia     0.2763      0.070      3.946      0.000      0.13
9       0.414
inner mongolia     0.2763      0.070      3.946      0.000      0.13
9       0.414
henan              0.3574      0.049      7.252      0.000      0.26
1       0.454
```

```
henan                  0.3574      0.049      7.252      0.000       0.26
1       0.454
sichuan                0.2920      0.066      4.404      0.000       0.16
2       0.422
sichuan                0.2920      0.066      4.404      0.000       0.16
2       0.422
hunan                  0.2532      0.052      4.845      0.000       0.15
1       0.356
hunan                  0.2532      0.052      4.845      0.000       0.15
1       0.356
heilongjiang           0.0754      0.064      1.173      0.241      -0.05
1       0.201
heilongjiang           0.0754      0.064      1.173      0.241      -0.05
1       0.201
shaanxi                0.3388      0.056      6.086      0.000       0.23
0       0.448
shaanxi                0.3388      0.056      6.086      0.000       0.23
0       0.448
chongqing              0.4771      0.058      8.191      0.000       0.36
3       0.591
chongqing              0.4771      0.058      8.191      0.000       0.36
3       0.591
jiangsu                0.4371      0.057      7.601      0.000       0.32
4       0.550
jiangsu                0.4371      0.057      7.601      0.000       0.32
4       0.550
jiangxi                0.3842      0.053      7.294      0.000       0.28
1       0.487
jiangxi                0.3842      0.053      7.294      0.000       0.28
1       0.487
anhui                  0.3302      0.053      6.234      0.000       0.22
6       0.434
anhui                  0.3302      0.053      6.234      0.000       0.22
6       0.434
hubei                  0.4566      0.069      6.643      0.000       0.32
2       0.591
hubei                  0.4566      0.069      6.643      0.000       0.32
2       0.591
ningxia                0.2745      0.067      4.127      0.000       0.14
4       0.405
ningxia                0.2745      0.067      4.127      0.000       0.14
4       0.405
shandong               0.3070      0.050      6.193      0.000       0.21
0       0.404
shandong               0.3070      0.050      6.193      0.000       0.21
0       0.404
qinghai                0.3345      0.044      7.546      0.000       0.24
8       0.421
qinghai                0.3345      0.044      7.546      0.000       0.24
8       0.421
guangxi                0.2974      0.053      5.625      0.000       0.19
4       0.401
guangxi                0.2974      0.053      5.625      0.000       0.19
4       0.401
zhejiang               0.3350      0.059      5.636      0.000       0.21
9       0.452
```

```
zhejiang           0.3350      0.059      5.636      0.000       0.21
9       0.452
gansu              0.3205      0.056      5.739      0.000       0.21
1       0.430
gansu              0.3205      0.056      5.739      0.000       0.21
1       0.430
hebei              0.2624      0.055      4.763      0.000       0.15
4       0.370
hebei              0.2624      0.055      4.763      0.000       0.15
4       0.370
guangdong          0.3392      0.053      6.429      0.000       0.23
6       0.443
guangdong          0.3392      0.053      6.429      0.000       0.23
6       0.443
fujian             0.3094      0.054      5.724      0.000       0.20
3       0.415
fujian             0.3094      0.054      5.724      0.000       0.20
3       0.415
tianjin            0.3423      0.047      7.301      0.000       0.25
0       0.434
tianjin            0.3423      0.047      7.301      0.000       0.25
0       0.434
shanghai           0.5034      0.043     11.722      0.000       0.41
9       0.588
shanghai           0.5034      0.043     11.722      0.000       0.41
9       0.588
xinjiang           0.3474      0.070      4.939      0.000       0.20
9       0.485
xinjiang           0.3474      0.070      4.939      0.000       0.20
9       0.485
liaoning           0.2012      0.054      3.715      0.000       0.09
5       0.307
liaoning           0.2012      0.054      3.715      0.000       0.09
5       0.307
yunnan             0.3329      0.065      5.091      0.000       0.20
5       0.461
yunnan             0.3329      0.065      5.091      0.000       0.20
5       0.461
beijing            0.3872      0.049      7.924      0.000       0.29
1       0.483
beijing            0.3872      0.049      7.924      0.000       0.29
1       0.483
shanxi             0.2353      0.053      4.475      0.000       0.13
2       0.338
shanxi             0.2353      0.053      4.475      0.000       0.13
2       0.338
guizhou            0.3187      0.056      5.676      0.000       0.20
9       0.429
guizhou            0.3187      0.056      5.676      0.000       0.20
9       0.429
======================================================================
=========
Omnibus:                       15.713   Durbin-Watson:
2.209
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
24.932
```

```
Skew:                           0.366   Prob(JB):
3.86e-06
Kurtosis:                       4.281   Cond. No.
1.03e+17
======================================================================
=========

Notes:
[1] Standard Errors are heteroscedasticity robust (HC0)
[2] The smallest eigenvalue is 4.86e-31. This might indicate that the
re are
strong multicollinearity problems or that the design matrix is singul
ar.
```

In [30]:
```
lines['draws1'] =  0


for h in range(999):
    adds = []
    for i in range(19):
        short = lines[lines['nlinks'] == i]
        xtrashort = len(short[short['year_opening']<=2016])
        new = list(set(list(short.index)))
        new2 = list(random.permutation(new)[:xtrashort])
        adds= adds + new2
    sl = adds

    lines_temp = lines.iloc[sl]
    stations_temp = stations.merge(lines_temp)
    stationsdum = pd.get_dummies(stations_temp['cityid'])



    cities['numstattemp'] = cities['cityid']
    for i in cities['cityid']:
        if i in stationsdum.columns:
            cities['numstattemp'] = cities['numstattemp'].replace(i, s

        else:
            cities['numstattemp'] = cities['numstattemp'].replace(i, 0

    citiestr = cities[cities['numstattemp'] > 0]
    cities['logdist'] = cities['cityid']
    for i in list(set(list(cities['cityid']))):

        trunc = distances[distances['cityid1']==i]
        if cities[cities['cityid']==i]['numstattemp'].mean() > 0:
            cities['logdist'] = cities['logdist'].replace(i, trunc[tru
        else:
            df = pd.DataFrame(columns = ['cityid1', 'cityid2', 'dist']
            for x in list(set(list(citiestr['cityid']))):
                df = df.append(trunc[trunc['cityid2'] == x])
            cities['logdist'] = cities['logdist'].replace(i, df['dist'

    if h == 0:
        cities['logdistfinal1'] = cities['logdist']
        cities['logdistl'] = cities['logdist']
    else:
        cities['logdistfinal1'] = cities['logdistfinal1'] +cities['log


    print(h)
cities['logdistfinal1'] = cities['logdistfinal1']/(h+1)
cities.head(30)

#i dont wanna talk about how ugly this code is. i'm sorry. i'm just so
```

```
0
1
2
```

```
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

In [31]:
```
np.log(cities['logdistfinal1']).describe()
```

Out[31]:
```
count    340.000000
mean       4.487112
std        0.790581
min        2.832481
25%        3.996449
50%        4.295818
75%        4.769178
max        7.338823
Name: logdistfinal1, dtype: float64
```

In [32]:
```
cities.to_csv('interpset3.csv')
```

In [33]:
```
stations = pd.read_csv('pset3_stations.csv')

stations = stations.merge(lines, how = 'left')
stations['year_opening'] = stations['year_opening'].replace([2017, 201
stations = stations.dropna()


stationsdum = pd.get_dummies(stations['cityid'])

cities['numstat'] = cities['cityid']
cities['numlinks'] = cities['cityid']
cities['avgspd'] = cities['cityid']
a = []

for i in cities['cityid']:

    if i in stationsdum.columns:
        cities['numstat'] = cities['numstat'].replace(i, stationsdum[i

    else:
        cities['numstat'] = cities['numstat'].replace(i, 0)


cities['logdist'] = cities['cityid']
for i in list(set(list(cities['cityid']))):

    trunc = distances[distances['cityid1']==i]
    if cities[cities['cityid']==i]['numstat'].mean() > 0:
        cities['logdist'] = cities['logdist'].replace(i, trunc[trunc['
    else:
        citiestr = cities[cities['numstat'] > 0]
        df = pd.DataFrame(columns = ['cityid1', 'cityid2', 'dist'])
        for a in list(set(list(citiestr['cityid']))):
            df = df.append(trunc[trunc['cityid2'] == a])
        cities['logdist'] = cities['logdist'].replace(i, df['dist'].mi
```

In [54]:
```
cities.to_csv('final2_pset3.csv')
```

In [55]:
```
cities = pd.read_csv('final2_pset3.csv')
```

In [56]:
```
cities['logdistf'].describe()
```

Out[56]:
```
count    340.000000
mean       4.487112
std        0.790581
min        2.832481
25%        3.996449
50%        4.295818
75%        4.769178
max        7.338823
Name: logdistf, dtype: float64
```

In [64]:
```
cities['loglogdist'] = np.log(cities['logdist'])
```

In [58]:
```
cities['logdistf'] = np.log(cities['logdistfinal1'])
```

In [65]:
```
#reading in csv so i dont have to rerun this crazy long simulation

#no FE with constant

citiesempt = cities.dropna(subset = ['empgrowth'])

treat = ['loglogdist', 'logdistf']

vals = sm.OLS(citiesempt['empgrowth'], sm.add_constant(citiesempt[trea
out = vals.fit(cov_type = 'HC0')
print(out.summary())
```

```
                            OLS Regression Results
==============================================================================
=========
Dep. Variable:              empgrowth   R-squared:
0.123
Model:                            OLS   Adj. R-squared:
0.117
Method:                 Least Squares   F-statistic:
11.64
Date:                Sun, 19 Nov 2023   Prob (F-statistic):
1.42e-05
Time:                        21:20:37   Log-Likelihood:
2.9031
No. Observations:                 275   AIC:
0.1938
Df Residuals:                     272   BIC:
11.04
Df Model:                           2
Covariance Type:                  HC0
==============================================================================
=========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const          0.9278      0.141      6.585      0.000       0.652
1.204
loglogdist    -0.0936      0.042     -2.224      0.026      -0.176
-0.011
logdistf      -0.0607      0.048     -1.276      0.202      -0.154
0.033
==============================================================================
=========
Omnibus:                        7.107   Durbin-Watson:
1.617
Prob(Omnibus):                  0.029   Jarque-Bera (JB):
9.807
Skew:                           0.168   Prob(JB):
0.00742
Kurtosis:                       3.862   Cond. No.
51.5
==============================================================================
=========
```

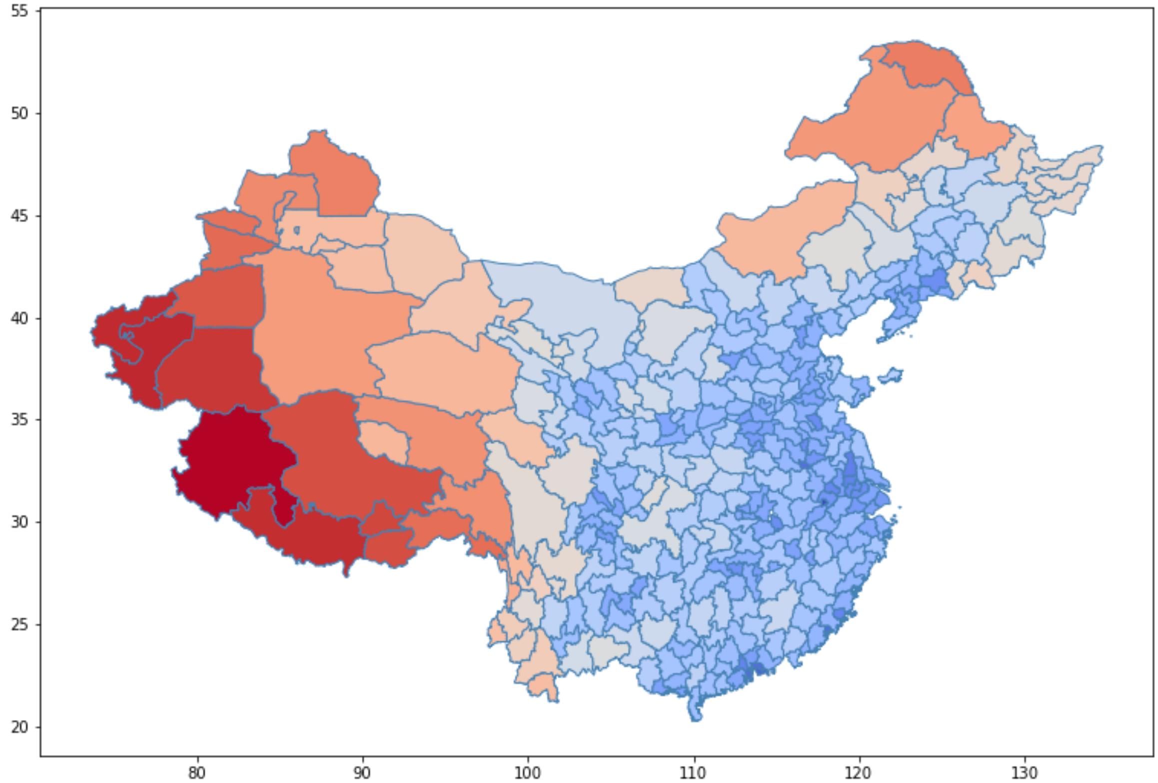
```
Notes:
[1] Standard Errors are heteroscedasticity robust (HC0)
```

In [66]:
```
#no FE with constant

citiesempt = cities.dropna(subset = ['empgrowth'])

treat = citiesempt['loglogdist'] - citiesempt['logdistf']

vals = sm.OLS(citiesempt['empgrowth'], sm.add_constant(treat))
out = vals.fit(cov_type = 'HC0')
print(out.summary())
```

```
                            OLS Regression Results
==============================================================================
=========
Dep. Variable:              empgrowth   R-squared:
0.016
Model:                            OLS   Adj. R-squared:
0.013
Method:                 Least Squares   F-statistic:
4.863
Date:                Sun, 19 Nov 2023   Prob (F-statistic):
0.0283
Time:                        21:20:42   Log-Likelihood:
-12.877
No. Observations:                 275   AIC:
29.75
Df Residuals:                     273   BIC:
36.99
Df Model:                           1
Covariance Type:                  HC0
==============================================================================
=========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const          0.2624      0.016     16.605      0.000       0.231
0.293
0             -0.0976      0.044     -2.205      0.027      -0.184
-0.011
==============================================================================
=========
Omnibus:                        8.219   Durbin-Watson:
1.433
Prob(Omnibus):                  0.016   Jarque-Bera (JB):
14.592
Skew:                          -0.019   Prob(JB):
0.000678
Kurtosis:                       4.128   Cond. No.
2.99
==============================================================================
=========

Notes:
[1] Standard Errors are heteroscedasticity robust (HC0)
```

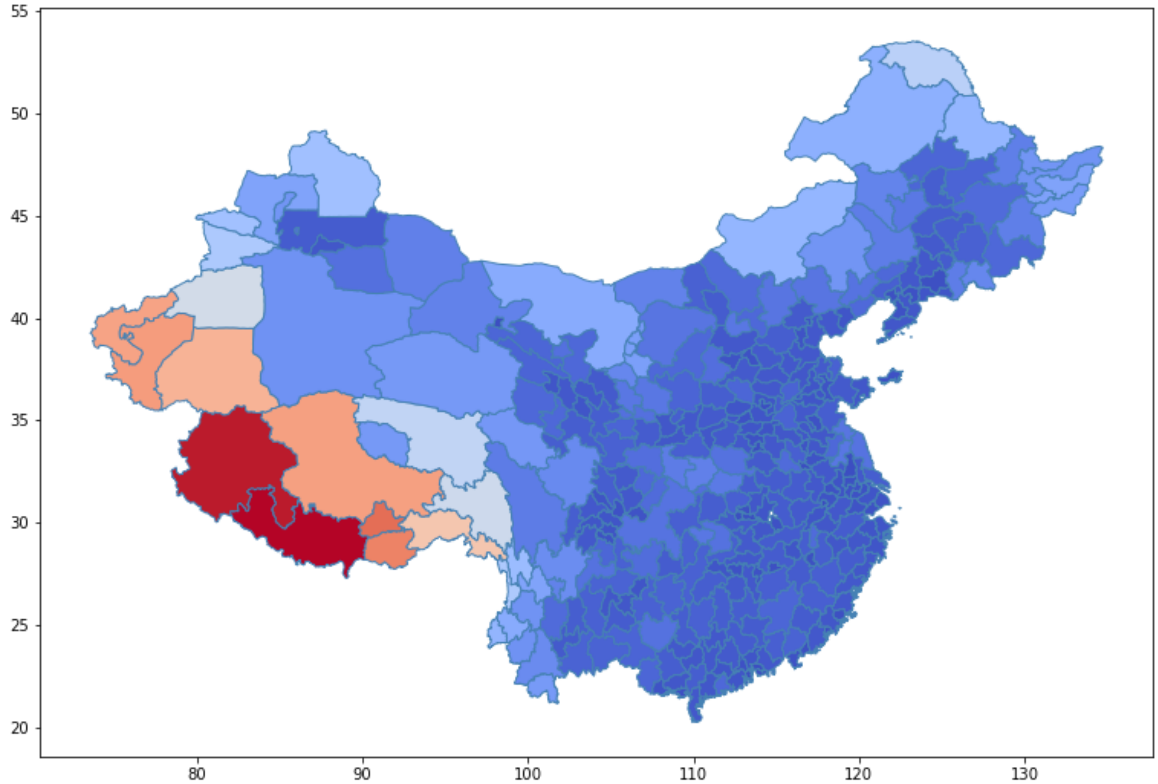
In [51]:
```
#maps!

shapefile = gpd.read_file("chn_admbnda_adm2_ocha.shp")

crs = {'init': shapefile.crs}
geometry = [Point(xy) for xy in zip(cities['longitude'], cities['latit
gdf1 = gpd.GeoDataFrame(cities, crs=crs, geometry=geometry)

shapefile2 = gpd.sjoin(gdf1, shapefile, how="right", op='intersects')

base = shapefile2.plot(figsize=(15,9), column = 'logdistf', edgecolor=
```

```
/opt/anaconda3/lib/python3.8/site-packages/pyproj/crs/crs.py:141: Fut
ureWarning: '+init=<authority>:<code>' syntax is deprecated. '<author
ity>:<code>' is the preferred initialization method. When making the
change, be mindful of axis order changes: https://pyproj4.github.io/p
yproj/stable/gotchas.html#axis-order-changes-in-proj-6 (https://pypro
j4.github.io/pyproj/stable/gotchas.html#axis-order-changes-in-proj-6)
  in_crs_string = _prepare_from_proj_string(in_crs_string)
/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactivesh
ell.py:3357: FutureWarning: The `op` parameter is deprecated and will
be removed in a future release. Please use the `predicate` parameter
instead.
  if (await self.run_code(code, result,  async_=asy)):
<ipython-input-51-aba84b46b721>:9: UserWarning: CRS mismatch between
the CRS of left geometries and the CRS of right geometries.
Use `to_crs()` to reproject one of the input geometries to match the
CRS of the other.

Left CRS: +init=epsg:4326 +type=crs
Right CRS: EPSG:4326

  shapefile2 = gpd.sjoin(gdf1, shapefile, how="right", op='intersects
')
```


55
50
45
40
35
30
25
20
80
90
100
110
120
130

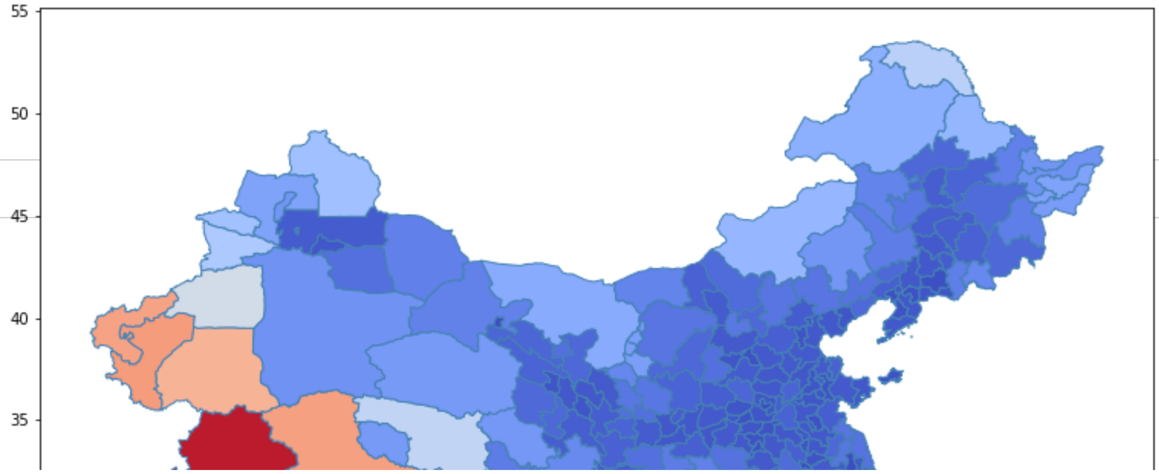
In [52]:
```
#maps!

shapefile = gpd.read_file("chn_admbnda_adm2_ocha.shp")

cities['demeaned'] = cities['logdist'] - cities['logdistf']

crs = {'init': shapefile.crs}
geometry = [Point(xy) for xy in zip(cities['longitude'], cities['latit
gdf1 = gpd.GeoDataFrame(cities, crs=crs, geometry=geometry)

shapefile2 = gpd.sjoin(gdf1, shapefile, how="right", op='intersects')

base = shapefile2.plot(figsize=(15,9), column = 'demeaned', edgecolor=
```

```
/opt/anaconda3/lib/python3.8/site-packages/pyproj/crs/crs.py:141: FutureWarning: '+init=<authority>:<code>' syntax is deprecated. '<authority>:<code>' is the preferred initialization method. When making the change, be mindful of axis order changes: https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-changes-in-proj-6 (https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-changes-in-proj-6)
  in_crs_string = _prepare_from_proj_string(in_crs_string)
/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3357: FutureWarning: The `op` parameter is deprecated and will be removed in a future release. Please use the `predicate` parameter instead.
  if (await self.run_code(code, result,  async_=asy)):
<ipython-input-52-03edc0c08ced>:11: UserWarning: CRS mismatch between the CRS of left geometries and the CRS of right geometries.
Use `to_crs()` to reproject one of the input geometries to match the CRS of the other.

Left CRS: +init=epsg:4326 +type=crs
Right CRS: EPSG:4326

  shapefile2 = gpd.sjoin(gdf1, shapefile, how="right", op='intersects')
```


55
50
45
40
35
30
25
20
80
90
100
110
120
130

In [53]:
```
#maps!

shapefile = gpd.read_file("chn_admbnda_adm2_ocha.shp")


crs = {'init': shapefile.crs}
geometry = [Point(xy) for xy in zip(cities['longitude'], cities['latit
gdf1 = gpd.GeoDataFrame(cities, crs=crs, geometry=geometry)

shapefile2 = gpd.sjoin(gdf1, shapefile, how="right", op='intersects')

base = shapefile2.plot(figsize=(15,9), column = 'logdist', edgecolor='
```

```
/opt/anaconda3/lib/python3.8/site-packages/pyproj/crs/crs.py:141: Fut
ureWarning: '+init=<authority>:<code>' syntax is deprecated. '<author
ity>:<code>' is the preferred initialization method. When making the
change, be mindful of axis order changes: https://pyproj4.github.io/p
yproj/stable/gotchas.html#axis-order-changes-in-proj-6 (https://pypro
j4.github.io/pyproj/stable/gotchas.html#axis-order-changes-in-proj-6)
  in_crs_string = _prepare_from_proj_string(in_crs_string)
/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactivesh
ell.py:3357: FutureWarning: The `op` parameter is deprecated and will
be removed in a future release. Please use the `predicate` parameter
instead.
  if (await self.run_code(code, result,  async_=asy)):
<ipython-input-53-5232853c41c3>:10: UserWarning: CRS mismatch between
the CRS of left geometries and the CRS of right geometries.
Use `to_crs()` to reproject one of the input geometries to match the
CRS of the other.

Left CRS: +init=epsg:4326 +type=crs
Right CRS: EPSG:4326

  shapefile2 = gpd.sjoin(gdf1, shapefile, how="right", op='intersects
')
```



In [ ]: