pca-using_dicts

May 29, 2021

[2]:
```
import numpy as np
import pandas as pd
import sklearn as sk
```

[3]:
```
features = ['gdp', 'fdi', 'lcs', 'prd', 'gsp', 'ind', 'trd', 'acf', 'cns',␣
 ↪'trs', 'emp']
reg_codes =['TBS', 'ADJ', 'GUR', 'IME', 'KAH', 'MTS', 'RLQ', 'SZS', 'SJV',␣
 ↪'QQR', 'SHQ']
years = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019]
```

0.1 Prepare Data

[4]:
```
# result
# two dicts named absolute and relative;
#       structure - {year: data, ...}
#       keys: year - vector dim 1X10
#       values: data - ndarray dim 11X11, reg_codes X features
# ndarray population
#       dim 11X10
```

[5]:
```
# import raw data in dataframes, transforming to ndarray
absolute = {}
for year in years:
    filepath = "dt/" + str(year) + ".csv"
    absolute[year] = pd.read_csv(filepath, header=0).iloc[:, 1:].values
```

[6]:
```
# import population data in dataframe, transforming to ndarray
population = pd.read_csv('dt/population.csv', header=0).iloc[:, 1:].values
```

[7]:
```
# calculate relative indicators.
relative = {}
for  idx, year in enumerate(years):
    relative[year] = absolute[year] / population[:, idx].reshape(11,1)
```

0.2 Standartize Data

[8]:
```
from sklearn.preprocessing import StandardScaler
```

[9]:
```
sc = StandardScaler()
for year in years:
    absolute[year] = sc.fit_transform(absolute[year])
    relative[year] = sc.fit_transform(relative[year])
```

0.3 PCA

[10]:
```
from sklearn.decomposition import PCA
x = {}
y = {}
# explained ratio

pca = PCA(n_components=1)

pca.fit(absolute[2010])
for year in years:
    x[year] = pca.transform(absolute[year])

pca.fit(relative[2010])
for year in years:
    y[year] = pca.transform(relative[year])
```

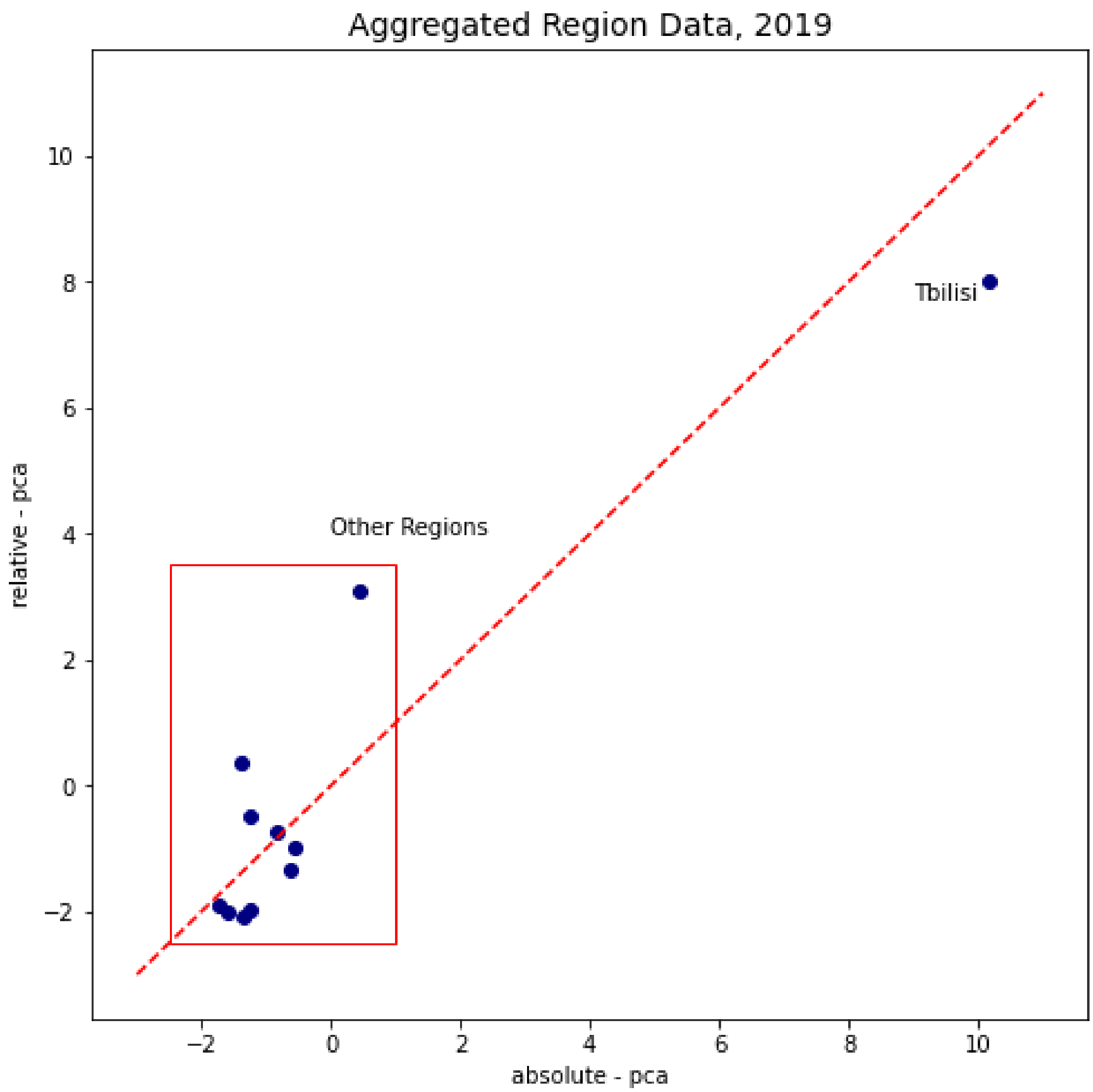
0.4 Visualize

[11]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
```

[12]:
```
fig, ax = plt.subplots(figsize=(8,8))
plt.scatter(x[2019], y[2019], color='navy')

plt.annotate('Tbilisi', (9, 7.7))
plt.annotate('Other Regions', (0, 4))
rect = patches.Rectangle((-2.5,-2.5),3.
 ↪5,6,linewidth=1,edgecolor='r',facecolor='none')
ax.add_patch(rect)
ax.plot(np.linspace(-3, 11, 100), np.linspace(-3, 11, 100), color='red',␣
 ↪linestyle='--')
ax.set_title('Aggregated Region Data, 2019', fontsize = 14)
plt.xlabel('absolute - pca')
plt.ylabel('relative - pca')
```

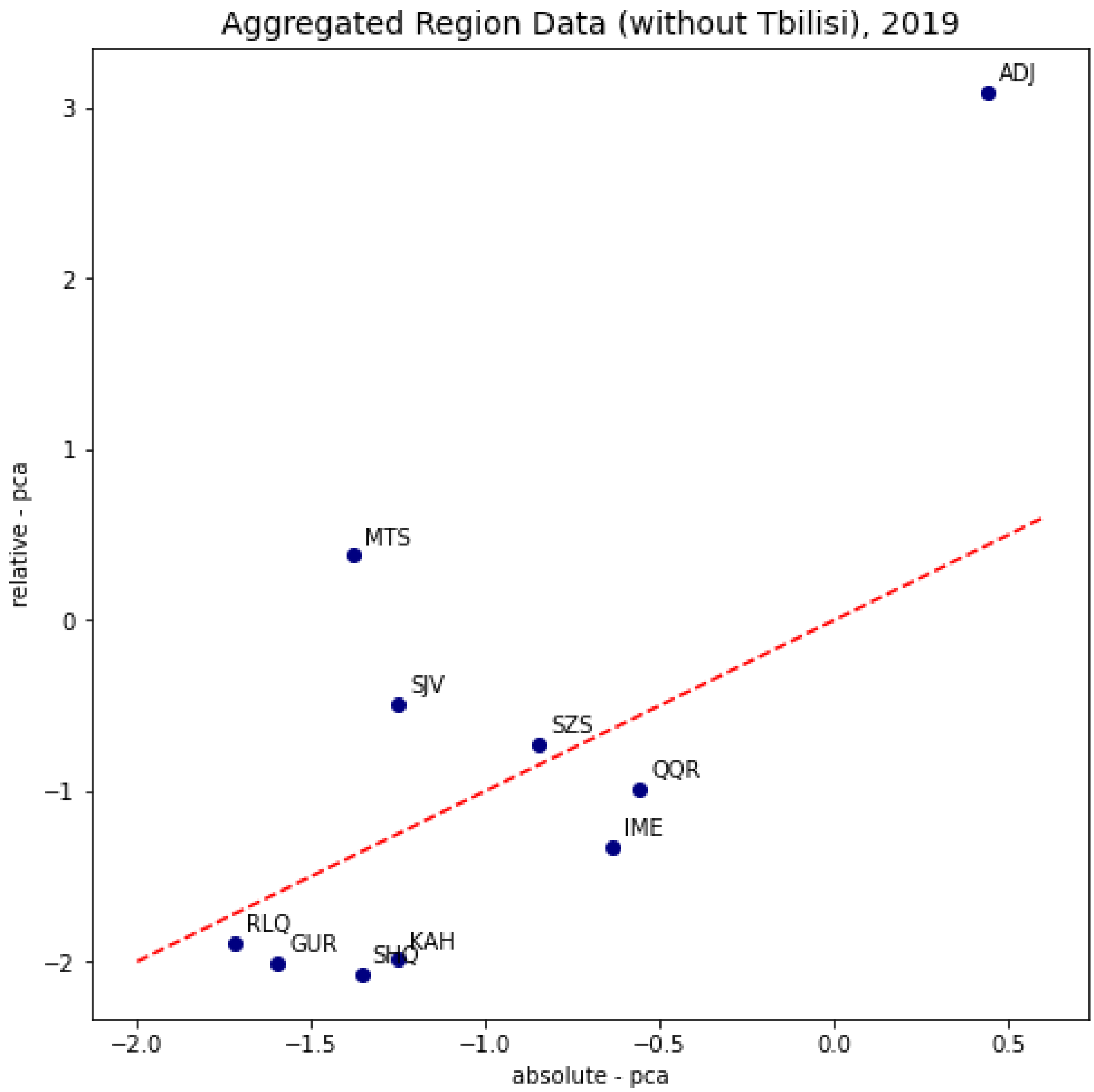
[12]:
```
Text(0, 0.5, 'relative - pca')
```



[13]:
```
fig, ax = plt.subplots(figsize=(8,8))

for i, txt in zip(range(1,11), reg_codes[1:]):
    plt.annotate(txt, (x[2019][i], y[2019][i]), xytext=(5,5),␣
 ↪textcoords='offset points')
    plt.scatter(x[2019][i], y[2019][i], color='navy', label=txt)
ax.plot(np.linspace(-2, 0.6, 100), np.linspace(-2, 0.6, 100), color='red',␣
 ↪linestyle='--')
ax.set_title('Aggregated Region Data (without Tbilisi), 2019', fontsize = 14)
plt.xlabel('absolute - pca')
plt.ylabel('relative - pca')
```

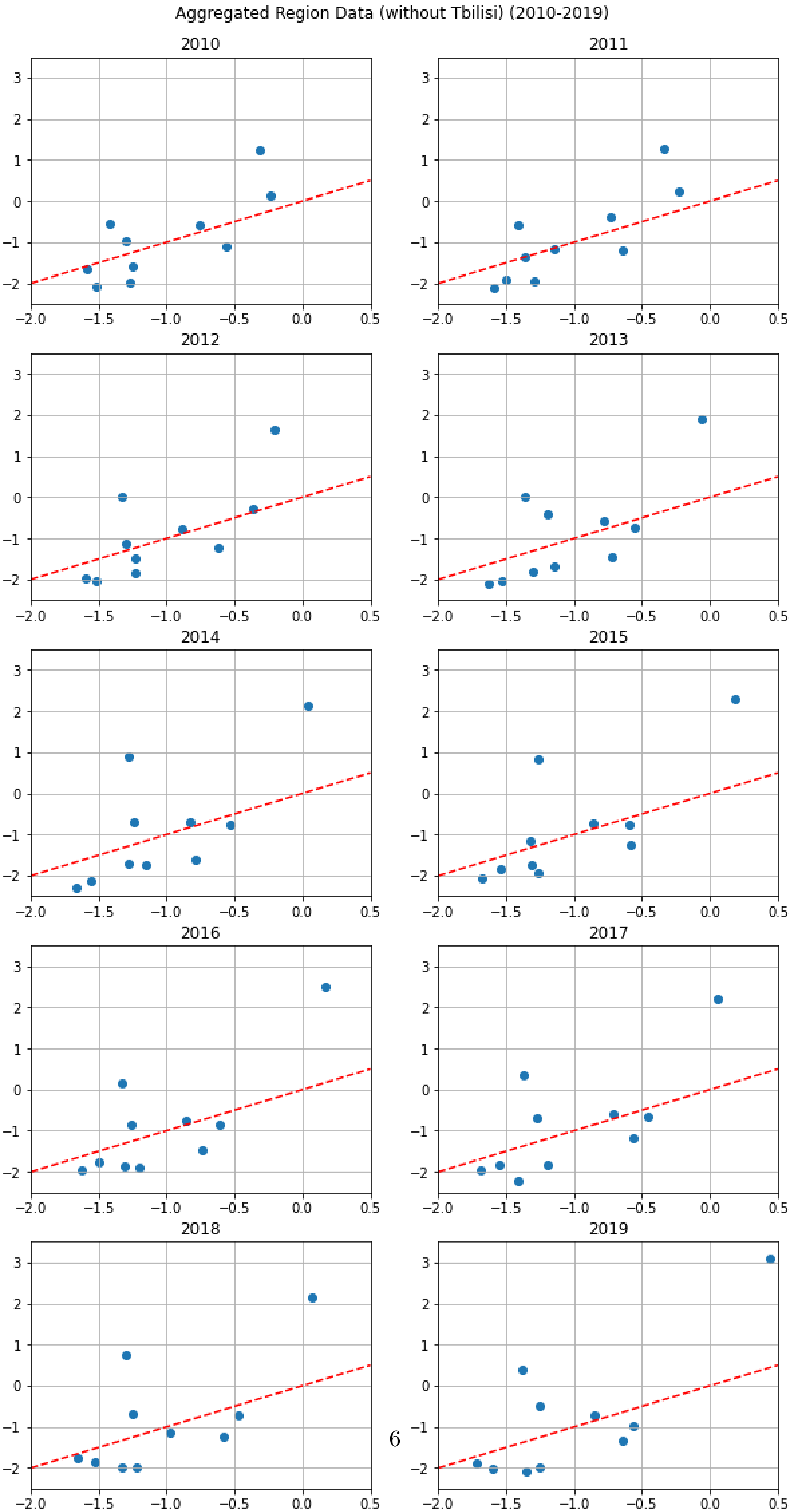
[13]: Text(0, 0.5, 'relative - pca')



[14]:
```
fig, ax = plt.subplots(figsize=(10,18),nrows = 5, ncols =2)
fig.suptitle('Aggregated Region Data (without Tbilisi) (2010-2019)')
fig.subplots_adjust(top=0.95)

for i, year in enumerate([2010,  2012, 2014, 2016, 2018]):
    ax[i,0].scatter(x[year][1:], y[year][1:])
    ax[i,0].set_title(year)
    ax[i,0].set_xlim([-2,0.5])
    ax[i,0].set_ylim([-2.5,3.5])
    ax[i,0].grid()
    ax[i,1].scatter(x[year+1][1:], y[year+1][1:])
```

```
    ax[i,1].set_title(year+1)
    ax[i,1].set_xlim([-2,0.5])
    ax[i,1].set_ylim([-2.5,3.5])
    ax[i,1].grid()
    ax[i,0].plot(np.linspace(-2, 0.5, 100), np.linspace(-2, 0.5, 100),␣
↪color='red', linestyle='--')
    ax[i,1].plot(np.linspace(-2, 0.5, 100), np.linspace(-2, 0.5, 100),␣
↪color='red', linestyle='--')
```

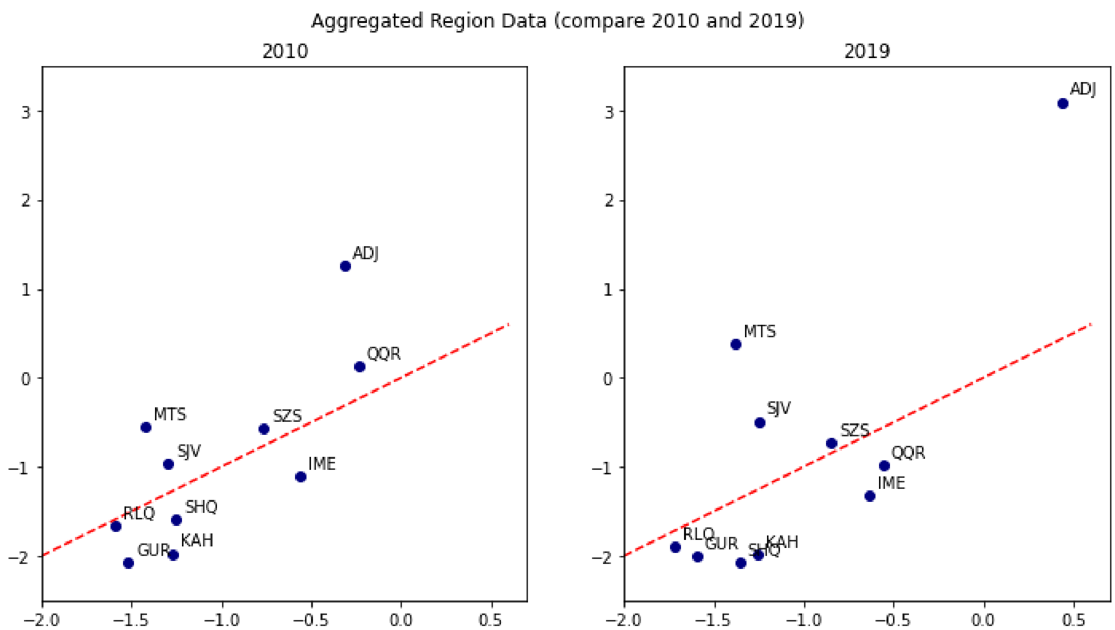

Aggregated Region Data (without Tbilisi) (2010-2019)
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
−2.0
−1.5
−1.0
−0.5
0.0
0.5
−2
−1
0
1
2
3

[15]:
```
fig, ax = plt.subplots(figsize=(12,6),nrows = 1, ncols =2)
fig.suptitle('Aggregated Region Data (compare 2010 and 2019)')
fig.subplots_adjust(top=0.9)

for i, txt in zip(range(1,11), reg_codes[1:]):
    ax[0].annotate(txt, (x[2010][i], y[2010][i]), xytext=(5,5),␣
 ↪textcoords='offset points')
    ax[0].scatter(x[2010][i], y[2010][i], color='navy', label=txt)
ax[0].plot(np.linspace(-2, 0.6, 100), np.linspace(-2, 0.6, 100), color='red',␣
 ↪linestyle='--')
ax[0].set_title('2010', fontsize = 12)
ax[0].set_xlim([-2,0.7])
ax[0].set_ylim([-2.5,3.5])

for i, txt in zip(range(1,11), reg_codes[1:]):
    ax[1].annotate(txt, (x[2019][i], y[2019][i]), xytext=(5,5),␣
 ↪textcoords='offset points')
    ax[1].scatter(x[2019][i], y[2019][i], color='navy', label=txt)
ax[1].plot(np.linspace(-2, 0.6, 100), np.linspace(-2, 0.6, 100), color='red',␣
 ↪linestyle='--')
ax[1].set_title('2019', fontsize = 12)
ax[1].set_xlim([-2,0.7])
ax[1].set_ylim([-2.5,3.5])
```

[15]: (-2.5, 3.5)



0.5 Calculations

[16]:
```
import scipy as sp
```

[17]:
```
# calculate centroids.  centroid -  for all regions; centroid_other - for␣
↪regions without TBS
centroid = {}
centroid_other ={}
for year in years:
    centroid[year] = [x[year].mean(), y[year].mean()]
    centroid_other[year] = [x[year][1:].mean(), y[year][1:].mean()]
```

[18]:
```
centroid
```

[18]:
```
{2010: [-4.0371746350005693e-17, -1.0092936587501423e-16],
 2011: [-1.4130111222501992e-16, 4.0371746350005693e-17],
 2012: [2.0185873175002847e-17, 1.6148698540002277e-16],
 2013: [-8.074349270001139e-17, 4.0371746350005693e-17],
 2014: [-1.2111523905001707e-16, 1.6148698540002277e-16],
 2015: [1.6148698540002277e-16, 3.431598439750484e-16],
 2016: [-8.074349270001139e-17, -2.0185873175002846e-16],
 2017: [-1.8167285857502563e-16, 0.0],
 2018: [0.0, -8.074349270001139e-17],
 2019: [-1.4130111222501992e-16, 3.6334571715005125e-16]}
```

[19]:
```
centroid_other
```

[19]:
```
{2010: [-1.023465299770003, -0.9126987473171153],
 2011: [-1.025366499471027, -0.9196702794815351],
 2012: [-1.0259369697893532, -0.9099822728330084],
 2013: [-1.0289054411984488, -0.8887836637820667],
 2014: [-1.026980478752908, -0.858113718522713],
 2015: [-1.021605242297669, -0.8379267395609545],
 2016: [-1.0271670792001142, -0.8733888023574418],
 2017: [-1.0159793650998146, -0.8404805962100536],
 2018: [-1.024550925256365, -0.8522401828207219],
 2019: [-1.0166541808841676, -0.8014340611022404]}
```

[20]:
```
# calculate pairwise distances.  dist
dist = {}
for year in years:
    dist_raw = sp.spatial.distance.pdist(np.concatenate((x[year], y[year]),␣
↪axis=1), 'euclidean')
    dist[year] = np.round(sp.spatial.distance.squareform(dist_raw),2)
```

```
pd.DataFrame(data=dist[2019], index=reg_codes, columns=reg_codes)
```

[20]:

```
       TBS    ADJ    GUR    IME    KAH    MTS    RLQ    SZS    SJV    QQR  \
TBS   0.00  10.90  15.45  14.28  15.18  13.84  15.48  14.06  14.24  14.00
ADJ  10.90   0.00   5.49   4.55   5.35   3.27   5.43   4.03   3.96   4.20
GUR  15.45   5.49   0.00   1.17   0.34   2.39   0.17   1.48   1.55   1.45
IME  14.28   4.55   1.17   0.00   0.90   1.86   1.22   0.63   1.03   0.35
KAH  15.18   5.35   0.34   0.90   0.00   2.37   0.48   1.32   1.50   1.22
MTS  13.84   3.27   2.39   1.86   2.37   0.00   2.30   1.23   0.88   1.59
RLQ  15.48   5.43   0.17   1.22   0.48   2.30   0.00   1.45   1.48   1.48
SZS  14.06   4.03   1.48   0.63   1.32   1.23   1.45   0.00   0.47   0.39
SJV  14.24   3.96   1.55   1.03   1.50   0.88   1.48   0.47   0.00   0.85
QQR  14.00   4.20   1.45   0.35   1.22   1.59   1.48   0.39   0.85   0.00
SHQ  15.32   5.47   0.25   1.04   0.13   2.45   0.41   1.44   1.59   1.35

       SHQ
TBS  15.32
ADJ   5.47
GUR   0.25
IME   1.04
KAH   0.13
MTS   2.45
RLQ   0.41
SZS   1.44
SJV   1.59
QQR   1.35
SHQ   0.00
```

[21]:

```
# calculate distance to centroid.  dist_c - matrix: reg_codes X years
dist_c = []
for idx in range(11):
    reg_row = []
    for year in years:
        dc = sp.spatial.distance.euclidean(np.concatenate((x[year][idx],
 ↪y[year][idx])), centroid[year])
        reg_row.append(dc)
    dist_c.append(reg_row)
dist_c = np.round(np.asarray(dist_c), 2)
pd.DataFrame(data=dist_c, index=reg_codes, columns=years)
```

[21]:

```
      2010   2011   2012   2013   2014   2015   2016   2017   2018   2019
TBS  13.71  13.77  13.71  13.60  13.38  13.21  13.48  13.19  13.33  12.95
ADJ   1.29   1.32   1.64   1.91   2.13   2.31   2.52   2.20   2.14   3.12
GUR   2.58   2.45   2.55   2.53   2.63   2.39   2.31   2.40   2.41   2.56
IME   1.25   1.35   1.36   1.63   1.79   1.38   1.64   1.31   1.38   1.47
KAH   2.35   2.35   2.21   2.03   2.08   2.31   2.24   2.18   2.34   2.35
MTS   1.53   1.52   1.33   1.36   1.57   1.51   1.34   1.41   1.51   1.43
```

RLQ	2.30	2.64	2.54	2.67	2.83	2.67	2.54	2.58	2.42	2.56
SZS	0.95	0.82	1.18	0.96	1.07	1.13	1.14	0.92	1.50	1.12
SJV	1.62	1.93	1.71	1.27	1.42	1.75	1.52	1.45	1.43	1.34
QQR	0.27	0.32	0.47	0.92	0.93	0.98	1.06	0.80	0.85	1.13
SHQ	2.02	1.65	1.93	2.23	2.13	2.18	2.28	2.63	2.40	2.48

[22]:
```
# calculate distance to centroid for cluster (without TBS).  dist_c_other -␣
↪matrix: reg_codes X years
dist_c_other = []
for idx in range(1,11):
    reg_row = []
    for year in years:
        dc = sp.spatial.distance.euclidean(np.concatenate((x[year][idx],␣
↪y[year][idx])), centroid_other[year])
        reg_row.append(dc)
    dist_c_other.append(reg_row)
dist_c_other = np.round(np.asarray(dist_c_other), 2)
pd.DataFrame(data=dist_c_other, index=reg_codes[1:], columns=years)
```

[22]:

	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
ADJ	2.28	2.30	2.67	2.96	3.17	3.36	3.59	3.22	3.19	4.16
GUR	1.27	1.12	1.24	1.24	1.37	1.12	1.00	1.12	1.13	1.33
IME	0.50	0.47	0.51	0.66	0.78	0.60	0.66	0.57	0.60	0.65
KAH	1.09	1.08	0.95	0.79	0.89	1.13	1.03	1.00	1.16	1.21
MTS	0.53	0.51	0.98	0.95	1.78	1.67	1.07	1.23	1.63	1.24
RLQ	0.94	1.32	1.21	1.37	1.57	1.40	1.23	1.29	1.11	1.30
SZS	0.43	0.62	0.19	0.41	0.26	0.19	0.21	0.40	0.29	0.18
SJV	0.28	0.56	0.34	0.51	0.27	0.43	0.24	0.29	0.28	0.39
QQR	1.31	1.39	0.91	0.49	0.50	0.44	0.42	0.59	0.57	0.49
SHQ	0.71	0.29	0.62	0.96	0.88	0.95	1.03	1.43	1.19	1.32

[23]:
```
# calculate coefficient
coeff = []
for idx in range(11):
    coeff_row = []
    for year in range(10):
        coeff_row.append(abs(round((dist_c[idx, year]/dist_c[:, year].mean()),␣
↪2)))
    coeff.append(coeff_row)
coeff = np.round(np.asarray(coeff), 2)
pd.DataFrame(data=coeff, index=reg_codes, columns=years)
```

[23]:

	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
TBS	5.05	5.03	4.92	4.81	4.61	4.57	4.62	4.67	4.62	4.38
ADJ	0.48	0.48	0.59	0.68	0.73	0.80	0.86	0.78	0.74	1.06
GUR	0.95	0.89	0.92	0.89	0.91	0.83	0.79	0.85	0.84	0.87
IME	0.46	0.49	0.49	0.58	0.62	0.48	0.56	0.46	0.48	0.50

KAH	0.87	0.86	0.79	0.72	0.72	0.80	0.77	0.77	0.81	0.80
MTS	0.56	0.56	0.48	0.48	0.54	0.52	0.46	0.50	0.52	0.48
RLQ	0.85	0.96	0.91	0.94	0.97	0.92	0.87	0.91	0.84	0.87
SZS	0.35	0.30	0.42	0.34	0.37	0.39	0.39	0.33	0.52	0.38
SJV	0.60	0.70	0.61	0.45	0.49	0.60	0.52	0.51	0.50	0.45
QQR	0.10	0.12	0.17	0.33	0.32	0.34	0.36	0.28	0.29	0.38
SHQ	0.74	0.60	0.69	0.79	0.73	0.75	0.78	0.93	0.83	0.84

[37]:

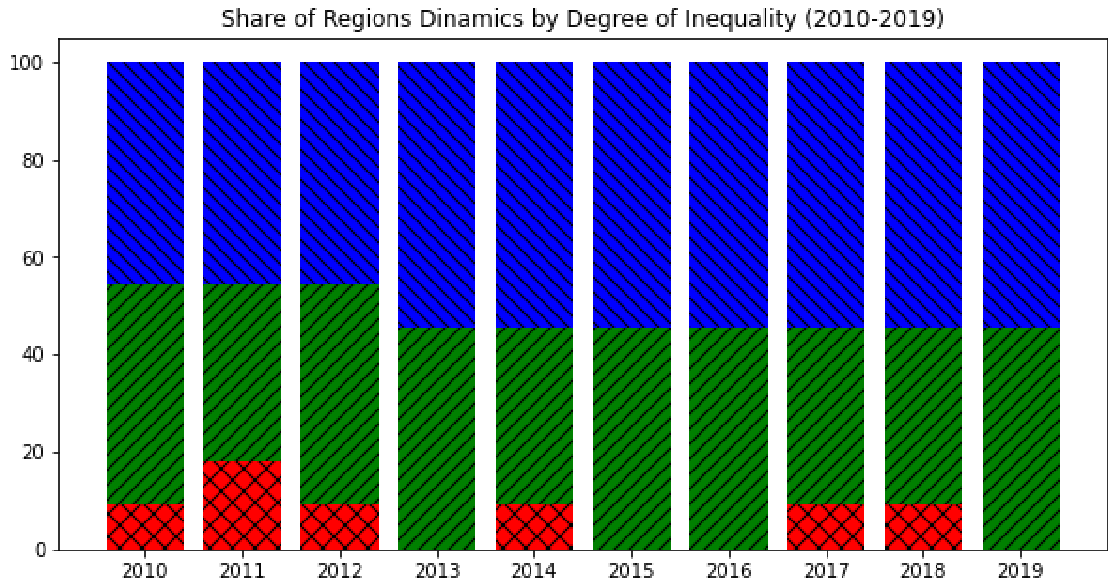
```python
# changings in dinamics

rate_67 = np.round((coeff>0.67).sum(axis=0)*100/11, 2)
rate_33 = np.round((coeff<0.33).sum(axis=0)*100/11, 2)
rate_33_67  = 100 - rate_67 - rate_33

fig, ax = plt.subplots(figsize=(10,5))
plt.title('Share of Regions Dinamics by Degree of Inequality (2010-2019)')
plt.bar(years,rate_33,color='r',hatch='xx')
plt.bar(years, rate_33_67, color='g', hatch='///', bottom=rate_33)
plt.bar(years, rate_67, color='b', hatch='\\\\\\', bottom=(rate_33_67+rate_33))
plt.xticks(years)
```

[37]:

```
([<matplotlib.axis.XTick at 0x157a812aa00>,
  <matplotlib.axis.XTick at 0x157a812a9d0>,
  <matplotlib.axis.XTick at 0x157a8128580>,
  <matplotlib.axis.XTick at 0x157a81a2760>,
  <matplotlib.axis.XTick at 0x157a81a2c70>,
  <matplotlib.axis.XTick at 0x157a81a91c0>,
  <matplotlib.axis.XTick at 0x157a81a96d0>,
  <matplotlib.axis.XTick at 0x157a81a9be0>,
  <matplotlib.axis.XTick at 0x157a81a27f0>,
  <matplotlib.axis.XTick at 0x157a81a9760>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



[51]:
```
# each regions share in common inequality
sum_sq_coeff = (coeff[:, -1]*coeff[:, -1]).sum()
each_rate = []
for idx in range(len(reg_codes)):
    each_rate.append(np.round(coeff[:, -1][idx]*coeff[:, -1][idx]*100/
 ↪sum_sq_coeff, 1))
pd.DataFrame(data=each_rate, index=reg_codes, columns=['rate'])
```

[51]:

	rate
TBS	79.5
ADJ	4.7
GUR	3.1
IME	1.0
KAH	2.7
MTS	1.0
RLQ	3.1
SZS	0.6
SJV	0.8
QQR	0.6
SHQ	2.9