This routine generates multiple social indicators at the municipal level for household surveys from 2011 to 2018.

Household surveys are created annually by the General Directorate of Statistics and Census. The survey allows generating indicators at the departmental level, however, there are 50 municipalities that are self-represented. In other words, the sample in these municipalities is large enough to be representative of themselves.

El Salvador has 262 municipalities, however in these 50 municipalities lives more than 65% of the population.

In [1]:
```
import pandas as pd
import numpy as np
import os
```

GINI and the MEDIAN INCOME

I am interested in calculating the GINI and the median income in each municipality.

The survey calculates a frequency weighting for each respondent.

I have created a function that takes into account the frequency of each respondent. In addition, the GINI is calculated at the household income level and at the individual level.

In [2]:
```
######## defining the GINI indicator########

def gini(EHPM, n_deciles=10, hogar=False):
    """Calculate the GINI of household surveys in El Salvador"""

    #main input
    x2 = EHPM[['ytot_ch','factor_ch','idp_ci','factor_ci','pc_ytot_ch']]

    #ytot_ch: Total household income
    #factor_ch: Weighted frequency of households
    #idp_ci: Type of survey (1=households; 0=individual)
    #factor_ci: Weighted frequency of individuals
    #pc_ytot_ch: Average income of each household member

    #defining the vars depending on the level required (Hogar: True = at the household level)
    if hogar==True:
        x2 = x2[x2['idp_ci']==1]
        x2['factor']= x2['factor_ch']
        x2['ytot'] = x2['ytot_ch']
    else:
        x2['factor'] = x2['factor_ci']
        x2['ytot'] = x2['pc_ytot_ch']

    #######sorting the data and creating the accumulated frequencies and then generating the deciles#####

    #sorting the data
    x2 = x2[x2['ytot'].notna()]
    x2 = x2.sort_values(by='ytot',ascending=True)
    x2['ingre'] = x2['ytot']*x2['factor']
    x2['x'] = x2['factor'].cumsum()/x2['factor'].sum()

    #creating a var for each decile
    deciles = np.linspace(0, 1, n_deciles + 1)
    x2['deciles']=((pd.cut(x2['x'], deciles,labels=False))+1)

    #creating a table with the accumulated income in each decile and the expected income
    x3 = pd.pivot_table(x2,values='ingre',index=['deciles'],aggfunc=np.sum).reset_index()
    x3['ingreso acumulado'] = x3['ingre'].cumsum()/x3['ingre'].sum()
    x3['ingreso igualitario'] = x3['deciles']*(1/n_deciles)

    #calculating income difference between the accumulated per qunitil and the expected
    x3['dif']=x3['ingreso igualitario']-x3['ingreso acumulado']

    #calculating the difference relative to the total expected value (GINI)
    return x3.dif.sum()/x3['ingreso igualitario'].sum()
```

In [3]:
```
##### calculating the median of any indicator in the database, taking intoaccount the weighted frequency

def median(dataframe):
    """Extract the average of "values" given the weighted frequency "weights" (dataframe['values', 'weights'])"""

    #sorting the data
    dataframe.sort_values(by=dataframe.columns[0], ascending=True, inplace=True)

    #definind the vars
    values = dataframe.iloc[:,0]
    weights = dataframe.iloc[:,1]

    #deleting data containing na values
    weights = weights[values.notna()]
    values = values.dropna()
    order = weights.cumsum()

    #extrayendo el valor de la mediana
    median = values[order<=weights.sum()/2].max()

    return median
```

In [5]:
```
#Folder address
os.chdir(r'C:\Users\eleno\Documents\EHPM\Data')
```

Creating a routine for every survey

The next code create multiples social indicators for each municipio

In [6]:
```python
#defining the years to use in each survey
year = ['2010','2011','2012','2013','2014','2015','2016','2017','2018']
#year =['2017','2018']

#creating an empty dataframe to store the new data
data = pd.DataFrame()

for year in year:
    #reading the files in dta format
    x1 = pd.read_stata('SLV_'+year+'a_BID.dta', encoding='latin-1', convert_categoricals=False, convert_missing=False)

    #### Preparing the data

    #using data at the individual level
    x1 = x1[x1['miembros_ci']==1]

    #calculating total income
    x1['ytot_ci'] = x1[['ylm_ci','ylnm_ci','ynlm_ci','ynlnm_ci']].sum(axis=1)

    #defining na values
    x1.loc[x1[['ylm_ci','ylnm_ci','ynlm_ci','ynlnm_ci']].isna().product(axis=1)==1,'ytot_ci']=np.nan

    #calculating income per capita at each home
    x1['ytot_ch'] = x1.groupby('idh_ch')['ytot_ci'].transform('sum')
    x1.loc[x1.ytot_ch<=0,'ytot_ch']=np.nan
    x1['pc_ytot_ch'] = x1['ytot_ch']/x1['nmiembros_ch']

    ###Extracting the municipality code from the survey

    #the variables created to define the municipalities is different depending on the survey
    #so I have built several cases to define the variable

    if (year=='2015')|(year=='2016')|(year=='2017')|(year=='2018'):
        municipios = x1[x1['autorrepresentado']==1]['codigomunic'].drop_duplicates().values

        if (municipios>100).sum()!=50:
            x1['codigomunic']=x1['r005']
            municipios = x1[x1['municauto']<50]['r005'].drop_duplicates().values

    if (year=='2011')|(year=='2012')|(year=='2013'):
        x1['product']=100
        x1['codigomunic']= x1['munic']+x1['product']*x1['region_c']
        municipios = x1[x1['munic_auto']<50]['codigomunic'].drop_duplicates().values

    if year=='2014':
        x1['codigomunic']=x1['municipio']
        municipios = x1[x1['munic_auto']<50]['municipio'].drop_duplicates().values

    if year == '2010':
        x1['product']=100
        x1['codigomunic']= x1['munic_auto']+x1['product']*x1['region_c']
        municipios = x1[x1['munic_auto']<50]['codigomunic'].drop_duplicates().values

    #calculating multiple indicators for each municipality

    for mun in municipios:

        x = x1[x1['codigomunic']==mun]

        #defining the year
        anio = float(year)

        #Average and median total income
        y = x[['ytot_ci','factor_ci']][x['ytot_ci'].notna()] #droping na values
        ytot_av = np.average(y['ytot_ci'],weights=y['factor_ci'])
        ytot_med = median(y[['ytot_ci','factor_ci']])

        #Average and median household income per capita
        y = x[['ytot_ci','factor_ci','pc_ytot_ch']][(x['ytot_ci'].notna()) & (x['pc_ytot_ch'].notna())] #droping na values
        pc_ytot_av = np.average(y['pc_ytot_ch'],weights=y['factor_ci'])
        pc_ytot_med = median(y[['pc_ytot_ch','factor_ci']])

        #Population
        Population = x['factor_ci'].sum()

        #Number of poor people in the municipality
        pob_extr = x[x['pobreza']==1]['factor_ci'].sum()
        pob_rela = x[x['pobreza']==2]['factor_ci'].sum()
        no_pobre = x[x['pobreza']==3]['factor_ci'].sum()

        #Percentage of household receiving remittances
        remesas_percentage = x[x['remesas_ch']>0]['factor_ch'].sum()/x['factor_ch'].sum()

        #Percentage of people working in differents sectors
        agriculture_wf = x[x['rama_ci']==1]['factor_ch'].sum()/x[x['rama_ci']>0]['factor_ch'].sum()
        industrial_wf = x[x['rama_ci']==3]['factor_ch'].sum()/x[x['rama_ci']>0]['factor_ch'].sum()
        services_wf = x[x['rama_ci']==6]['factor_ch'].sum()/x[x['rama_ci']>0]['factor_ch'].sum()

        #Percentage of economically active people
        pea = x[x['pea_ci']==1]['factor_ch'].sum()/x['factor_ch'].sum()

        #Number of unemployed people in the municipality
        desempleo = x[x['desemp_ci']==1]['factor_ci'].sum()
        empleo =  x[x['desemp_ci']==0]['factor_ci'].sum()

        #Number of people employed in the informal sector in the municipality
        informal = x[x['formal_ci']==0]['factor_ci'].sum()
        formal = x[x['formal_ci']==1]['factor_ci'].sum()

        #Years studied on average by population
        y = x[['aedu_ci','factor_ci']][x['aedu_ci'].notna()]

        escolar = np.average(y['aedu_ci'],weights=y['factor_ci'])

        #Gini
        gini_ci = gini(x,n_deciles=10,hogar=False) #calculating individual inequality
        gini_ch = gini(x,n_deciles=10,hogar=True) #calculating household inequality
```

```
        #storing the values created in the dataframe
        df = pd.DataFrame({'COD_MUN':mun,
                          'Year': anio,
                          'Average Income': ytot_av,
                          'Median Income': ytot_med,
                          'Average Household Income per capita': pc_ytot_av,
                          'Median Household Income per capita': pc_ytot_med,
                          'Extreme poverty': pob_extr,
                          'Relative poverty': pob_rela,
                          'Not poor': no_pobre,
                          'Unemployment': desempleo,
                          'Working population':empleo,
                          'Informal jobs': informal,
                          'Formal jobs':formal,
                          'Years of schooling': escolar,
                          'Gini_ci': gini_ci,
                          'Gini_ch': gini_ch,
                          'Remittances %': remesas_percentage,
                          'Agriculture workforce %': agriculture_wf,
                          'Industrial workforce %': industrial_wf,
                          'Services workforce %': services_wf,
                          'Economically active population %': pea,
                          'Population':Population
                         },index=[0])

         #appending the data for each municipality
        data = data.append(df, ignore_index = True)

data = data.reset_index(drop=True)
```

```
C:\Users\ELENOC\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  import sys
C:\Users\ELENOC\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:21: 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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
C:\Users\ELENOC\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:22: 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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

In [7]:

```
#Calculating percentage of poverty, informality and unemployment

#Percentage of poverty
data['Poverty ratio'] = data[['Extreme poverty','Relative poverty']].sum(axis=1)/\
                          data[['Extreme poverty','Relative poverty','Not poor']].sum(axis=1)

#Percentage of unemployment
data['Unemployment %'] = data['Unemployment']/\
                          data[['Unemployment','Working population']].sum(axis=1)

#Percentage of informality
data['Informal jobs ratio'] = data['Informal jobs']/\
                          data[['Informal jobs','Formal jobs']].sum(axis=1)
```

In [8]:

```
#calculating job growth

#creating a table to facilitate iteration
x2 = pd.pivot_table(data, values='Working population',index=['COD_MUN'],columns=['Year'],aggfunc=np.sum).reset_index()

anios = np.arange(2011,2019)
anios2 = np.arange(2010,2018)

y2 = pd.DataFrame()

for x,y in zip(anios,anios2):
    y3 = pd.DataFrame()
    y3['Jobs growth'] = (x2[x]-x2[y])/x2[y]
    y3['Year']= x
    y3['COD_MUN'] = x2['COD_MUN']
    y2 = y2.append(y3)

#Merging the new variables
data = pd.merge(data,y2,on=['Year','COD_MUN'],how='inner')
```

Printing output data

In [9]:

```
data.head()
```

Out[9]:

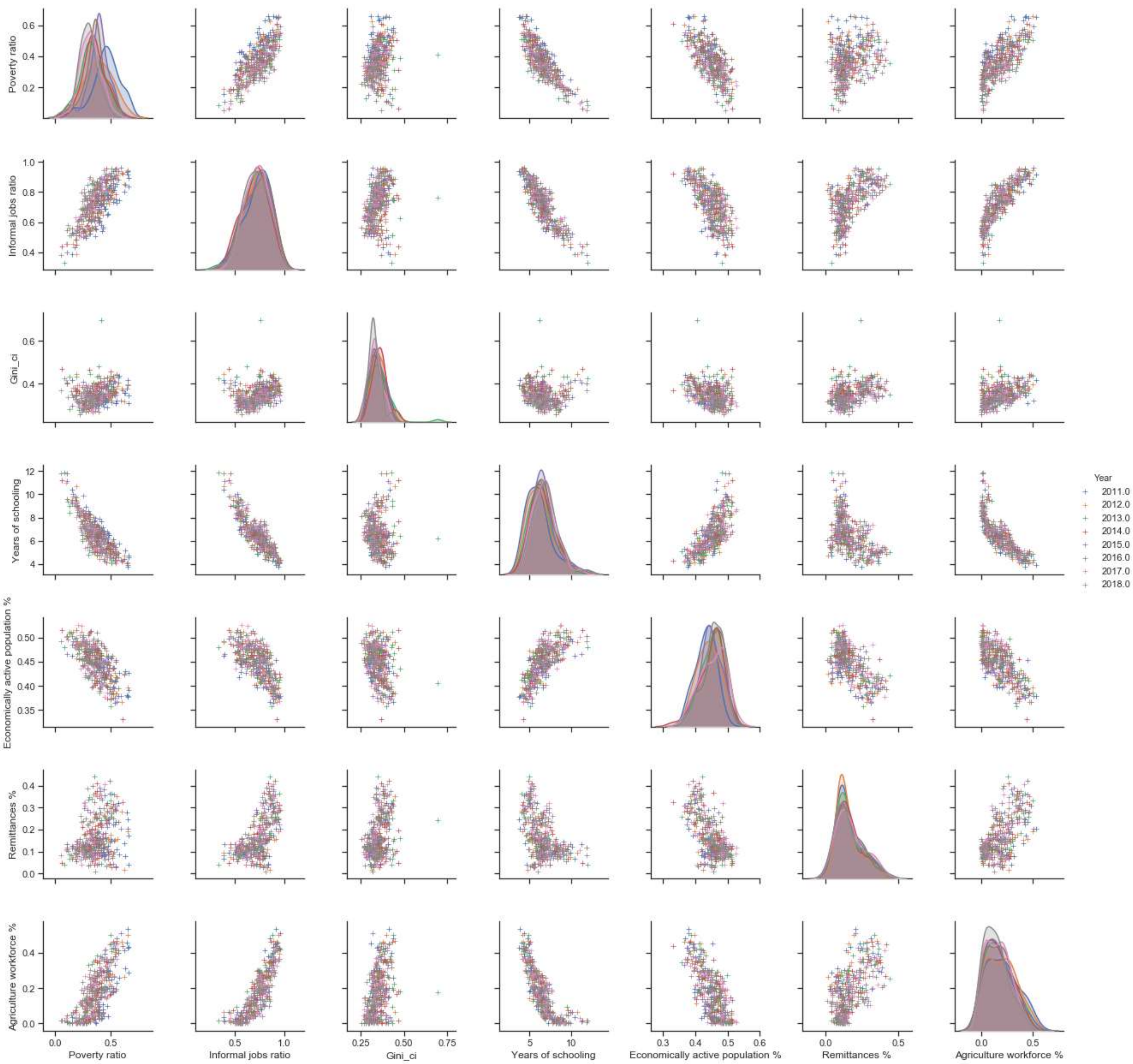
	COD_MUN	Year	Average Income	Median Income	Average Household Income per capita	Median Household Income per capita	Extreme poverty	Relative poverty	Not poor	Unemployment	...	Remittances %	Agriculture workforce %	Industrial workforce %	Services workforce %	Economically active population %	Population	Poverty ratio	Unemplo
0	311	2011.0	148.904140	70.0	124.572034	100.933334	2911	9731	16791	642	...	0.133422	0.081689	0.183189	0.375217	0.452281	29433	0.429518	0.0
1	202	2011.0	90.550311	0.0	79.882443	66.083333	8604	16538	14796	295	...	0.138114	0.410612	0.169129	0.163220	0.422655	39938	0.629526	0.0
2	315	2011.0	138.954372	50.0	123.001383	83.333333	13052	27969	37494	1269	...	0.090238	0.200042	0.115442	0.346083	0.435738	78515	0.522461	0.0
3	407	2011.0	162.465473	50.0	141.739855	104.616666	3305	7564	20789	813	...	0.168172	0.217109	0.145772	0.235245	0.443679	31658	0.343326	0.0
4	501	2011.0	443.120287	196.0	390.837702	284.000000	692	4938	30677	576	...	0.067232	0.033941	0.102316	0.268264	0.462996	36307	0.155067	0.0

5 rows × 26 columns

Analyzing the data

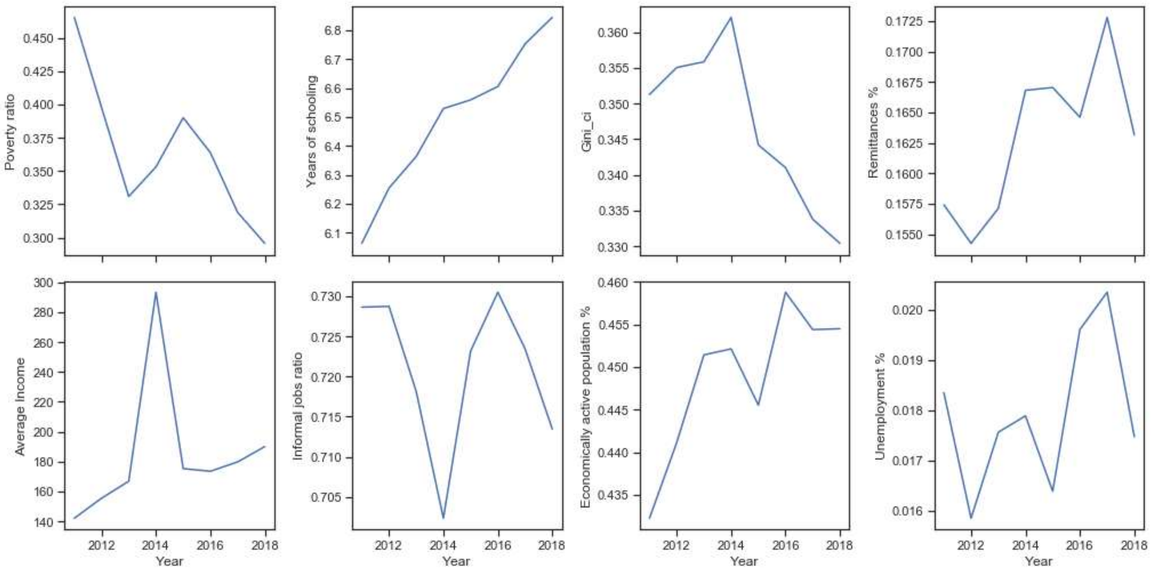
1. Multiple scatter plots to look for correlations

In [11]:
```
import seaborn as sns; sns.set(style="ticks", color_codes=True)

g = sns.pairplot(data,
                vars = ['Poverty ratio','Informal jobs ratio','Gini_ci','Years of schooling',
                        'Economically active population %','Remittances %','Agriculture workforce %'],
                markers="+",
                hue = 'Year')
```



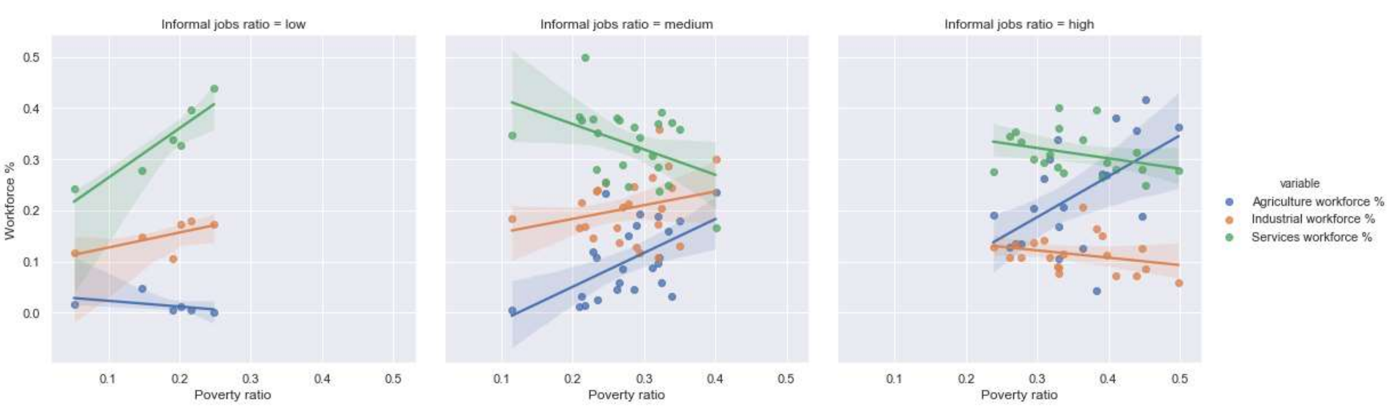
2. How some indicators have changed during time

```
In [12]: import seaborn as sns
         import matplotlib.pyplot as plt
         from scipy import stats

         #variables to plot
         variables = ['Poverty ratio','Years of schooling', 'Gini_ci','Remittances %',
                      'Average Income','Informal jobs ratio','Economically active population %',
                      'Unemployment %', 'Year']


         # Set up the matplotlib figure
         f, axes = plt.subplots(2, 4, figsize=(14, 7), sharex=True, constrained_layout=True)


         #plot each variable during time
         variables.remove('Year')
         cols = [0,1,2,3,0,1,2,3]
         rows = [0,0,0,0,1,1,1,1]

         for var,x, y in zip(variables,cols,rows):

             #droping outliers (Z score below 3)
             df1 = data[[var,'Year']][(np.abs(stats.zscore(data[[var,'Year']])) < 3).all(axis=1)]

             sns.lineplot(x='Year', y = var, ax = axes[y,x], data = df1, ci=None)
```



3. Observing correlation between poverty and economic classification of work in the municipalities in 2018

```
In [13]: #data ordered
         df = pd.melt(data, id_vars=['Year','Poverty ratio','Informal jobs ratio'],
                 value_vars=['Agriculture workforce %','Industrial workforce %', 'Services workforce %'])

         df = df[df['Year']==2018]

         df['Informal jobs ratio'] = pd.cut(df['Informal jobs ratio'],bins=3,labels=['low','medium','high'])

         sns.set()

         # Plot sepal with as a function of sepal_length across days
         g = sns.lmplot(x='Poverty ratio', y="value", col='Informal jobs ratio',
                        truncate=True, height=5, data=df, hue='variable')

         # Use more informative axis labels than are provided by default
         g.set_axis_labels("Poverty ratio", "Workforce %")

Out[13]: <seaborn.axisgrid.FacetGrid at 0x222850ea3c8>
```

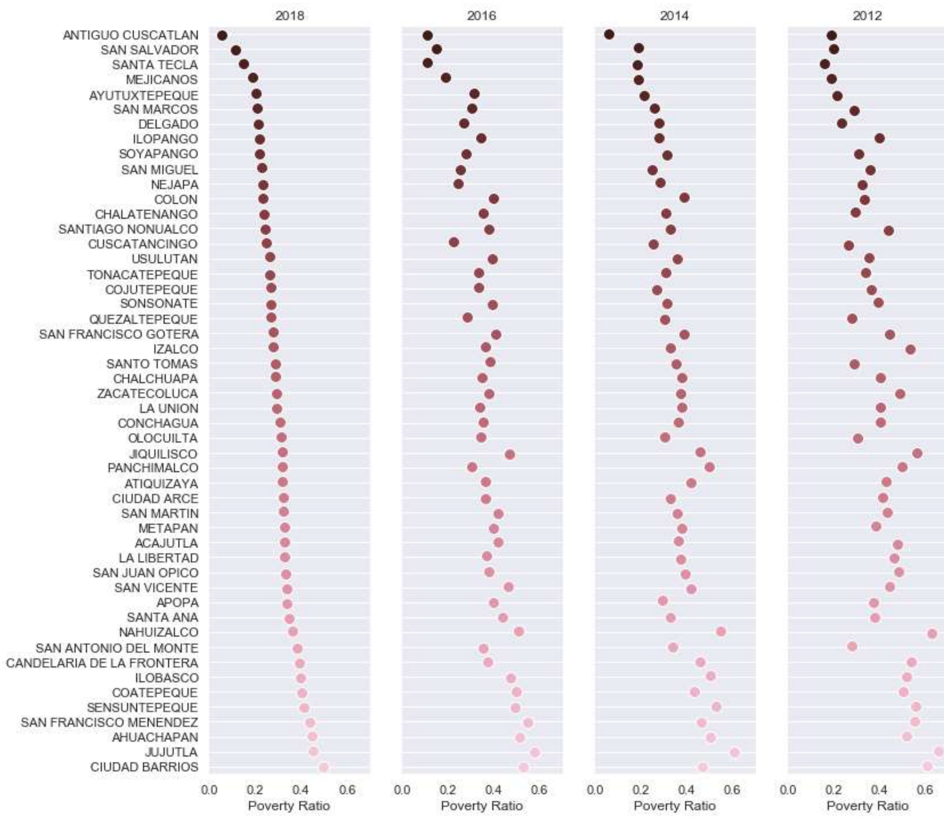


Reading csv file that contains more information of the municipalities as name and department

This information is not found in household surveys.

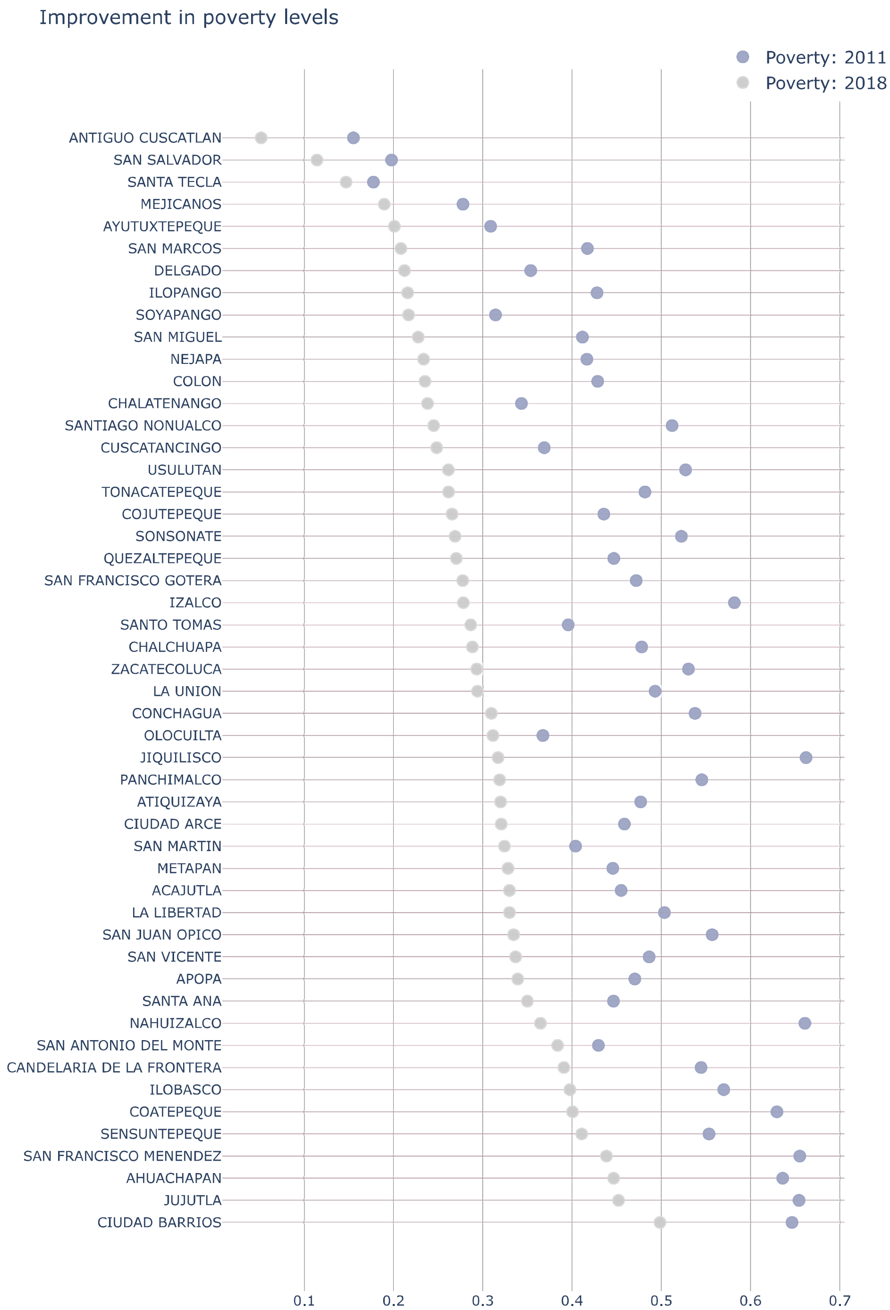
In [20]:
```python
inf_mun = pd.read_csv("info_mun.csv", encoding = 'latin-1')

#merging the new data
data2 = pd.merge(data, inf_mun[['NOM_MUN','COD_MUN4','NOM_DPTO','REG_GEO_DE']],
                 left_on='COD_MUN',right_on='COD_MUN4',how='left')

data2.rename(columns={'REG_GEO_DE':"Region", 'NOM_DPTO': "Departamento", 'NOM_MUN': "Municipio"}, inplace=True)
```

In [22]:
```python
#preparing data for the new graph
df = pd.pivot_table(data2, values = 'Poverty ratio', index = 'Municipio', columns = 'Year', aggfunc=np.sum).reset_index()

# Make the PairGrid
g = sns.PairGrid(df.sort_values(2018, ascending=True),
                 x_vars=[2018,2016,2014,2012], y_vars=["Municipio"],
                 height=10, aspect=.25)

# Draw a dot plot using the stripplot function
g.map(sns.stripplot, size=10, orient="h",
      palette="ch:s=1,r=-.1,h=1_r", linewidth=1, edgecolor="w")

# Use the same x axis limits on all columns and add better labels
g.set(xlim=(0, 0.70), xlabel="Poverty Ratio", ylabel="")

# Use semantically meaningful titles for the columns
titles = [2018,2016,2014,2012]

for ax, title in zip(g.axes.flat, titles):

    # Set a different title for each axes
    ax.set(title=title)

    # Make the grid horizontal instead of vertical
    ax.xaxis.grid(False)
    ax.yaxis.grid(True)

sns.despine(left=True, bottom=True)
```



In [43]:
```
import plotly.express as px
from plotly.offline import init_notebook_mode

init_notebook_mode(connected=True)

df = data2
fig = px.scatter(df, x='Years of schooling', y='Poverty ratio', color='Region', size='Population',
                 facet_col='Year', facet_col_wrap=4)

fig.show()
```

In [44]:
```python
import plotly.graph_objects as go

#preparing data for the new graph
df = pd.pivot_table(data2, values = 'Poverty ratio', index = 'Municipio', columns = 'Year', aggfunc=np.sum).reset_index()
df.sort_values(2018, ascending=False,inplace=True)

fig = go.Figure()

fig.add_trace(go.Scatter(
    x=df[2011],
    y=df['Municipio'],
    name='Poverty: 2011',
    marker=dict(
        color='rgba(156, 165, 196, 0.95)',
        line_color='rgba(156, 165, 196, 1.0)',
    )
))
fig.add_trace(go.Scatter(
    x=df[2018], y=df['Municipio'],
    name='Poverty: 2018',
    marker=dict(
        color='rgba(204, 204, 204, 0.95)',
        line_color='rgba(217, 217, 217, 1.0)'
    )
))

fig.update_traces(mode='markers', marker=dict(line_width=1, symbol='circle', size=10))

fig.update_layout(
    title="Improvement in poverty levels",
    margin=dict(l=140, r=40, b=50, t=80),
    legend=dict(
        font_size=15,
        yanchor='middle',
        x=0.8, y=1
    ),
    width=800,
    height=1200,
    plot_bgcolor='white',
    hovermode='closest'
)
fig.update_yaxes(gridwidth=2, gridcolor='LightPink')

fig.show()
```


Improvement in poverty levels
Poverty: 2011
Poverty: 2018
ANTIGUO CUSCATLAN
SAN SALVADOR
SANTA TECLA
MEJICANOS
AYUTUXTEPEQUE
SAN MARCOS
DELGADO
ILOPANGO
SOYAPANGO
SAN MIGUEL
NEJAPA
COLON
CHALATENANGO
SANTIAGO NONUALCO
CUSCATANCINGO
USULUTAN
TONACATEPEQUE
COJUTEPEQUE
SONSONATE
QUEZALTEPEQUE
SAN FRANCISCO GOTERA
IZALCO
SANTO TOMAS
CHALCHUAPA
ZACATECOLUCA
LA UNION
CONCHAGUA
OLOCUILTA
JIQUILISCO
PANCHIMALCO
ATIQUIZAYA
CIUDAD ARCE
SAN MARTIN
METAPAN
ACAJUTLA
LA LIBERTAD
SAN JUAN OPICO
SAN VICENTE
APOPA
SANTA ANA
NAHUIZALCO
SAN ANTONIO DEL MONTE
CANDELARIA DE LA FRONTERA
ILOBASCO
COATEPEQUE
SENSUNTEPEQUE
SAN FRANCISCO MENENDEZ
AHUACHAPAN
JUJUTLA
CIUDAD BARRIOS
0.1
0.2
0.3
0.4
0.5
0.6
0.7