In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
#funtion to read the name of the dataset and return two datasets
def read_data(data):
    """Takes the name of the dataset as an Input.
       Returns two Dataframes"""

    #reading the data
    df = pd.read_csv(r'C:\Users\ADMIN\Downloads\%s'%data)

    #Transpose the data and name the new column as 'Years'
    transpose_df = df.set_index('Country Name').T.rename_axis('Years',axis = 0).rename_

    return df, transpose_df
```

In [3]:
```
df, t_df = read_data('climate_change.csv') #getting the two datasets
```

In [4]:
```
#checking all the available indicators
df['Series Name'].unique()
```

Out[4]:
```
array(['Agricultural land (sq. km)', 'Agricultural land (% of land area)',
       'Arable land (% of land area)',
       'Rural land area where elevation is below 5 meters (sq. km)',
       'Rural land area where elevation is below 5 meters (% of total land area)',
       'Urban land area where elevation is below 5 meters (sq. km)',
       'Urban land area where elevation is below 5 meters (% of total land area)',
       'Land area where elevation is below 5 meters (% of total land area)',
       'Forest area (sq. km)', 'Forest area (% of land area)',
       'Agricultural irrigated land (% of total agricultural land)',
       'Average precipitation in depth (mm per year)',
       'Cereal yield (kg per hectare)', 'Population growth (annual %)',
       'Population, total', 'Urban population growth (annual %)',
       'Urban population', 'Urban population (% of total population)',
       'Foreign direct investment, net inflows (% of GDP)',
       'Ease of doing business rank (1=most business-friendly regulations)',
       'CPIA public sector management and institutions cluster average (1=low to 6=hig
h)',
       'School enrollment, primary and secondary (gross), gender parity index (GPI)',
       'Primary completion rate, total (% of relevant age group)',
       'Mortality rate, under-5 (per 1,000 live births)',
       'Community health workers (per 1,000 people)',
       'Prevalence of underweight, weight for age (% of children under 5)',
       'Poverty headcount ratio at $1.90 a day (2011 PPP) (% of population)',
       'Access to electricity (% of population)',
       'Electricity production from coal sources (% of total)',
       'Electricity production from hydroelectric sources (% of total)',
       'Electricity production from natural gas sources (% of total)',
       'Electricity production from nuclear sources (% of total)',
       'Electricity production from oil sources (% of total)',
       'Renewable electricity output (% of total electricity output)',
       'Electricity production from renewable sources, excluding hydroelectric (kWh)',
       'Electricity production from renewable sources, excluding hydroelectric (% of tot
```

```
al)',
       'Renewable energy consumption (% of total final energy consumption)',
       'Energy use (kg of oil equivalent) per $1,000 GDP (constant 2017 PPP)',
       'Electric power consumption (kWh per capita)',
       'Energy use (kg of oil equivalent per capita)',
       'CO2 intensity (kg per kg of oil equivalent energy use)',
       'CO2 emissions from gaseous fuel consumption (kt)',
       'CO2 emissions from gaseous fuel consumption (% of total)',
       'CO2 emissions (kg per 2015 US$ of GDP)', 'CO2 emissions (kt)',
       'CO2 emissions from liquid fuel consumption (kt)',
       'CO2 emissions from liquid fuel consumption (% of total)',
       'CO2 emissions (metric tons per capita)',
       'CO2 emissions (kg per PPP $ of GDP)',
       'CO2 emissions (kg per 2017 PPP $ of GDP)',
       'CO2 emissions from solid fuel consumption (kt)',
       'CO2 emissions from solid fuel consumption (% of total)',
       'Other greenhouse gas emissions, HFC, PFC and SF6 (thousand metric tons of CO2 eq
uivalent)',
       'Other greenhouse gas emissions (% change from 1990)',
       'Total greenhouse gas emissions (kt of CO2 equivalent)',
       'Total greenhouse gas emissions (% change from 1990)',
       'HFC gas emissions (thousand metric tons of CO2 equivalent)',
       'Methane emissions (kt of CO2 equivalent)',
       'Methane emissions (% change from 1990)',
       'Nitrous oxide emissions (thousand metric tons of CO2 equivalent)',
       'Nitrous oxide emissions (% change from 1990)',
       'PFC gas emissions (thousand metric tons of CO2 equivalent)',
       'SF6 gas emissions (thousand metric tons of CO2 equivalent)',
       'Disaster risk reduction progress score (1-5 scale; 5=best)',
       'GHG net emissions/removals by LUCF (Mt of CO2 equivalent)',
       'Droughts, floods, extreme temperatures (% of population, average 1990-2009)',
       'Rural population living in areas where elevation is below 5 meters (% of total p
opulation)',
       'Urban population living in areas where elevation is below 5 meters (% of total p
opulation)',
       'Population living in areas where elevation is below 5 meters (% of total populat
ion)',
       'Population in urban agglomerations of more than 1 million (% of total populatio
n)',
       'Annual freshwater withdrawals, total (billion cubic meters)',
       'Annual freshwater withdrawals, total (% of internal resources)',
       'Terrestrial protected areas (% of total land area)',
       'Marine protected areas (% of territorial waters)',
       'Terrestrial and marine protected areas (% of total territorial area)',
       'Agriculture, forestry, and fishing, value added (% of GDP)', nan,
       'Data from database: World Development Indicators',
       'Last Updated: 04/27/2022'], dtype=object)
```

In [5]:

```
#Normal DataFrame
df.head()
```

Out[5]:

	Series Name	Series Code	Country Name	Country Code	1990 [YR1990]	2000 [YR2000]	2012 [YR2012]	2013 [YR2013]	201 [YR201
0	Agricultural land (sq. km)	AG.LND.AGRI.K2	Afghanistan	AFG	380400	377530	379100	379100	3791(

	Series Name	Series Code	Country Name	Country Code	1990 [YR1990]	2000 [YR2000]	2012 [YR2012]	2013 [YR2013]	201 [YR201
1	Agricultural land (sq. km)	AG.LND.AGRI.K2	Albania	ALB	11210	11440	12013	11873	11742
2	Agricultural land (sq. km)	AG.LND.AGRI.K2	Algeria	DZA	386760	400210	413981.9	414316.35	41431
3	Agricultural land (sq. km)	AG.LND.AGRI.K2	American Samoa	ASM	30	48	49	49	4
4	Agricultural land (sq. km)	AG.LND.AGRI.K2	Andorra	AND	230	230	187.6	188.1	18

In [6]:
```
#Tranposed DataFrame
t_df.head()
```

Out[6]:

	Country Name	years	Afghanistan	Albania	Algeria	American Samoa	Andorra
0	Series Name	Agricultural land (sq. km)	Agricultural land (sq. km)	Agricultural land (sq. km)	Agricultural land (sq. km)	Agricultural land (sq. km)	
1	Series Code	AG.LND.AGRI.K2	AG.LND.AGRI.K2	AG.LND.AGRI.K2	AG.LND.AGRI.K2	AG.LND.AGRI.K2	AG
2	Country Code	AFG	ALB	DZA	ASM	AND	
3	1990 [YR1990]	380400	11210	386760	30	230	
4	2000 [YR2000]	377530	11440	400210	48	230	

5 rows × 20222 columns

In [7]:
```
#creating 3 dataframes with 3 indicators by subsetting the original dataframe
df_popgrowth = df[df['Series Name']=='Population growth (annual %)']
df_enercons = df[df['Series Name'] == 'Renewable energy consumption (% of total final e
df_etrc_acc = df[df['Series Name'] == 'Access to electricity (% of population)']
```

In [8]:
```
#merging the 3 dataframes
df_merge = pd.concat([df_popgrowth,df_enercons,df_etrc_acc])
```

In [9]:
```
df_merge.head()
```

Out[9]:

	Series Name	Series Code	Country Name	Country Code	1990 [YR1990]	2000 [YR2000]	2012
3458	Population growth (annual %)	SP.POP.GROW	Afghanistan	AFG	4.47695437953718	2.97505722281038	3.407600
3459	Population growth (annual %)	SP.POP.GROW	Albania	ALB	1.79908559333186	-0.637356833943492	-0.1651510
3460	Population growth (annual %)	SP.POP.GROW	Algeria	DZA	2.56646025164596	1.35841747006607	1.951463
3461	Population growth (annual %)	SP.POP.GROW	American Samoa	ASM	3.1253549165475	1.33550063100072	-0.1543653
3462	Population growth (annual %)	SP.POP.GROW	Andorra	AND	3.84490142183965	1.57527263657523	-1.589923

In [10]:
```
#resetting the index
df_merge.reset_index(inplace=True,drop=True)

#replace '..' values with NaN values in data
df_merge.replace('..',np.nan,inplace = True)
```

In [11]:
```
#dropping the rows with more than 10 missing values
df_merge.dropna(thresh=10,inplace = True)
```

In [12]:
```
#removing the past 4 years since there is very little data
df_merge = df_merge.iloc[:,:-3]
```

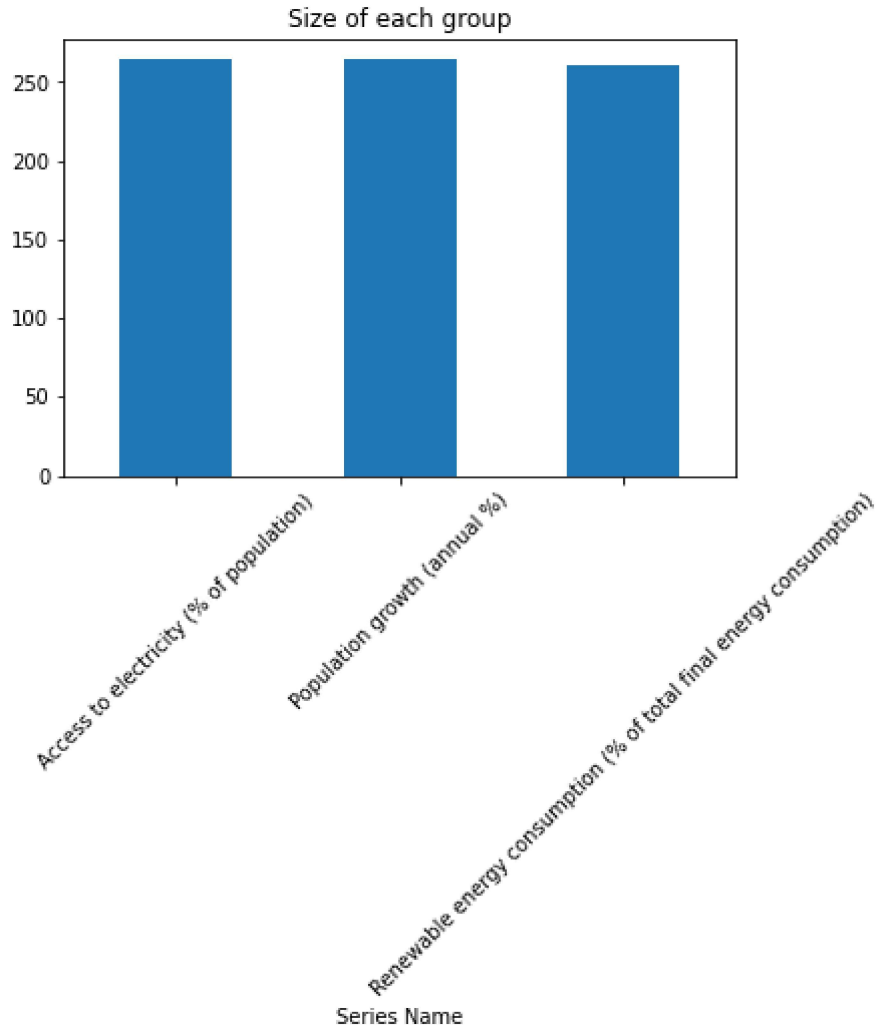
In [13]:
```
#replacing missing values with 0
df_merge.fillna('0',inplace=True) #replacing NaN with 0's
```

In [14]:
```
#dropping redundant columns
df_merge.drop(['Series Code','Country Code'],axis=1,inplace = True)
```

In [15]:
```
years = df_merge.columns[2:]
#removing wrong values final time
df_merge.replace('..',0,inplace = True)
```

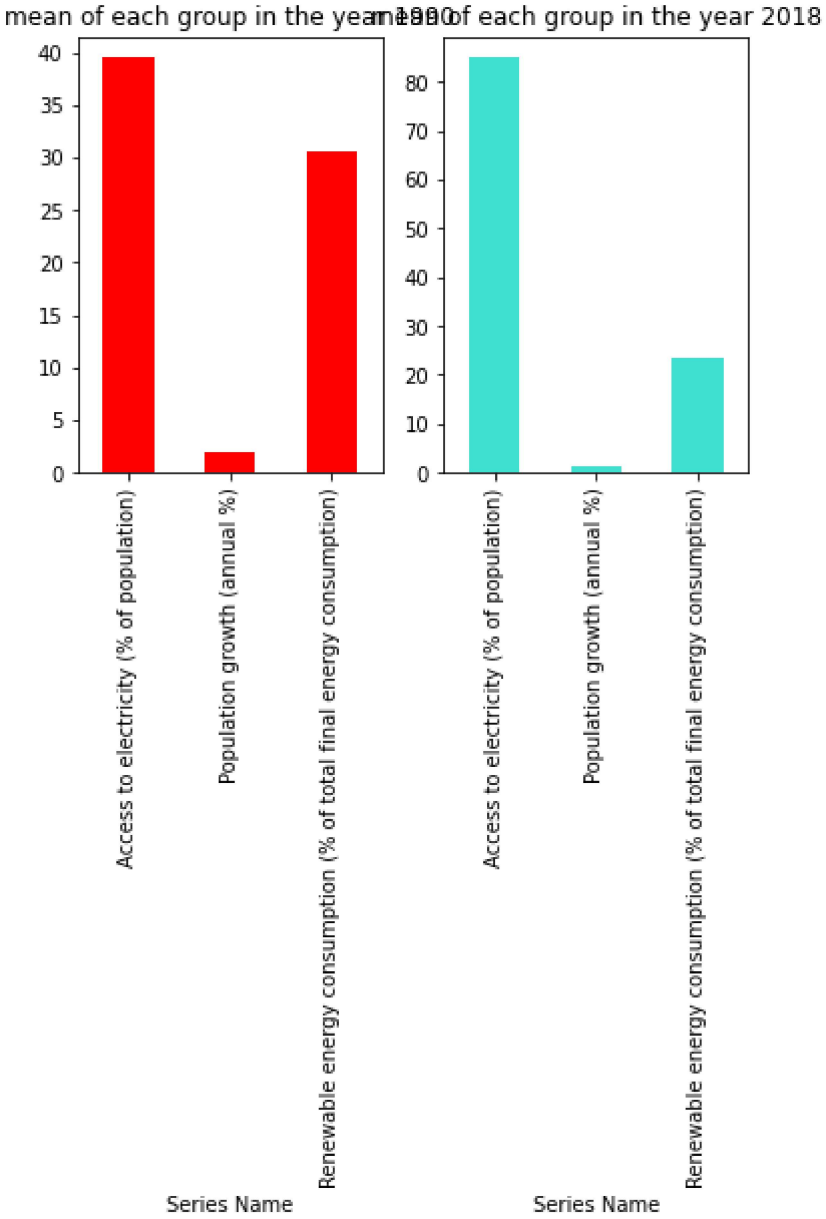
In [16]:
```
#ocnverting the data to numeric and rounding off to 2 decimals
df_merge[years] = np.abs(df_merge[years].astype('float').round(decimals=2))
```

In [17]:
```
#checking of the size of the 3 datasets
df_merge.groupby(['Series Name'])['1990 [YR1990]'].size().plot(kind='bar')
plt.title('Size of each group') #title
plt.xticks(rotation = 45)
plt.show()
```



We have equal number of observations

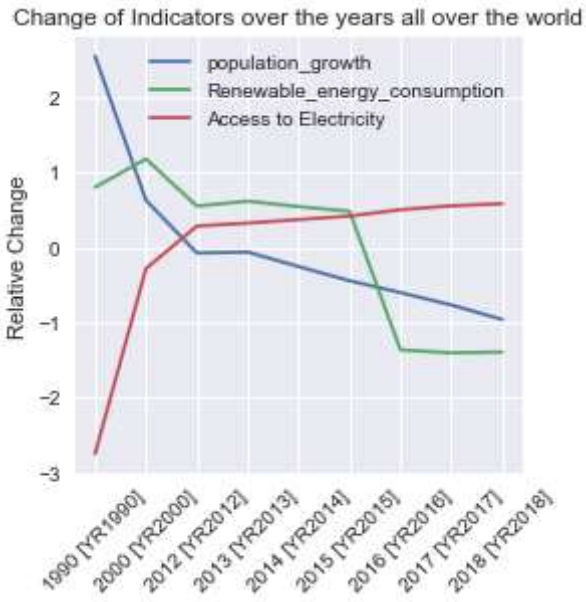
In [18]:
```
#plotting the means of the 3 groups of the data over the years

figure, axes = plt.subplots(1, 2) #calling axes

df_merge.groupby(['Series Name'])['1990 [YR1990]'].mean().plot(kind='bar',ax=axes[0],co
axes[0].title.set_text('mean of each group in the year 1990') #plot title


df_merge.groupby(['Series Name'])['2018 [YR2018]'].mean().plot(kind ='bar',ax=axes[1],c
plt.title('mean of each group in the year 2018') #plot title


plt.savefig('change.jpeg') #saving the plot for future use

plt.show()
```



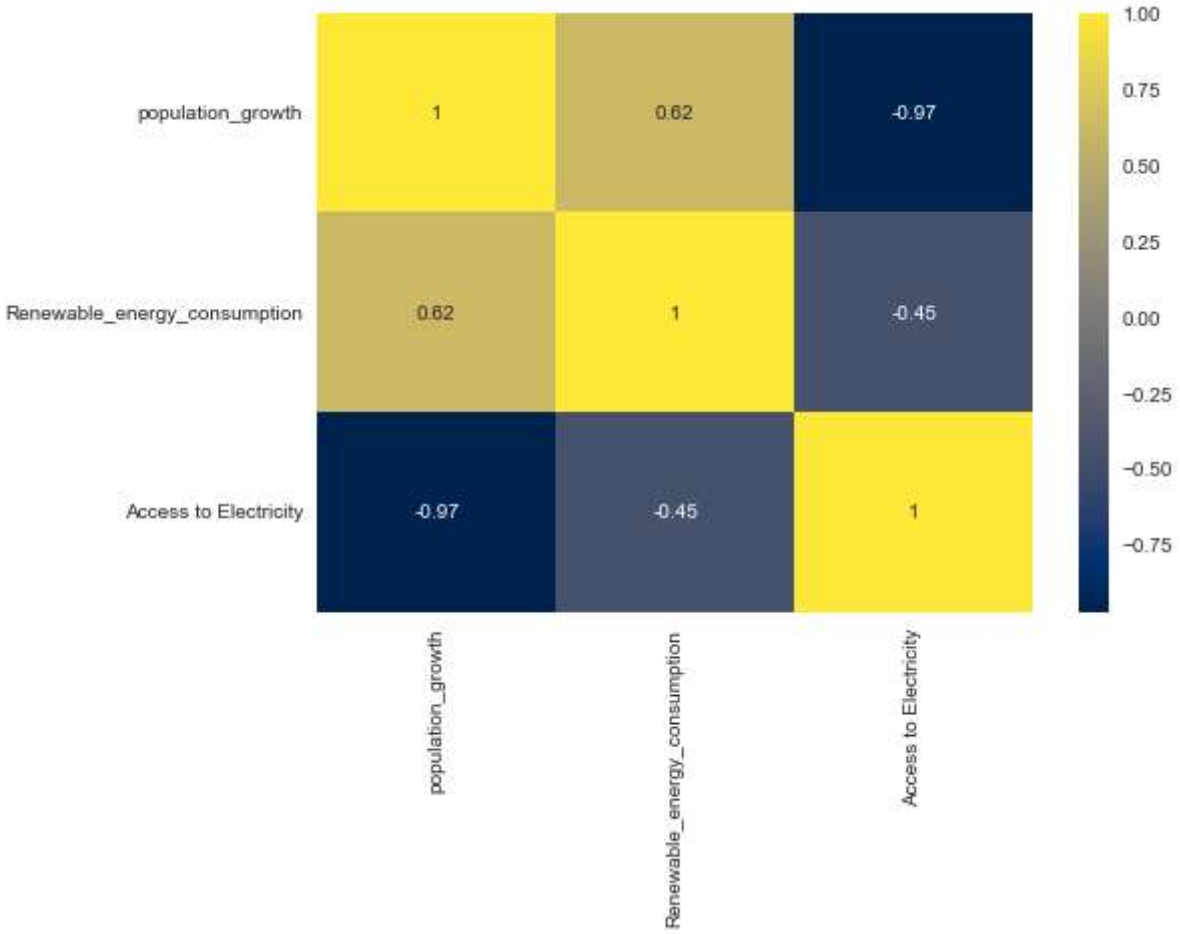
In [19]:
```
#diving the data back to 3 datasets to perform summary statistics
df_merge_pgrow = df_merge[df_merge['Series Name']=='Population growth (annual %)']
df_merge_ecsmp = df_merge[df_merge['Series Name']=='Renewable energy consumption (% of
df_merge_lcrxs = df_merge[df_merge['Series Name'] == 'Access to electricity (% of popul
```

In [20]:
```
#saving the summary statistics in a dataframe
stats = pd.DataFrame()
stats['p_grow'] = df_merge_pgrow[years].mean(axis = 0).to_frame()
stats['e_cons'] = df_merge_ecsmp[years].mean(axis = 0).to_frame()
stats['acc_el'] = df_merge_lcrxs[years].mean(axis= 0).to_frame()


#scaling the data since all the groups are not in the same scale
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

#scaling the data using standard scaler and saving it with full column names
stats_sc = pd.DataFrame(scaler.fit_transform(stats),columns=['population_growth','Renew
```

In [21]:
```
stats_sc.index = years #setting the index

#plotting the general trend of the 3 groups over the years
plt.style.use('seaborn')

plt.figure(figsize=(4,4)) #size of the image
for i in stats_sc.columns:
    plt.plot(stats_sc.index,stats_sc[i],label = i) #plotting the 3 lines
plt.legend(loc = 'best') #setting the location of the legends
plt.xticks(stats_sc.index,rotation = 45) #changing the x axis values
plt.ylabel('Relative Change') #setting y axis label
plt.title('Change of Indicators over the years all over the world') #setting the title

plt.savefig('trend.jpeg') #saving the image
plt.show()
```



In [22]:
```
corr_mat = stats_sc.corr() #calculating the correlation of the data
import seaborn as sns
sns.heatmap(corr_mat, annot=True,cmap ='cividis')
plt.savefig('correlation_heatmap.jpeg')
plt.show()
```

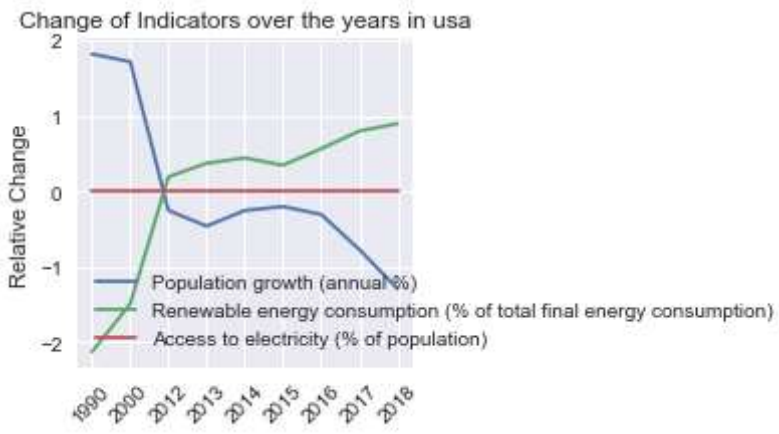


In [23]:
```
usa = df_merge[df_merge['Country Name'] == 'United States']
usa = usa.drop(['Country Name'],axis = 1)
usa = usa.set_index('Series Name').T
```

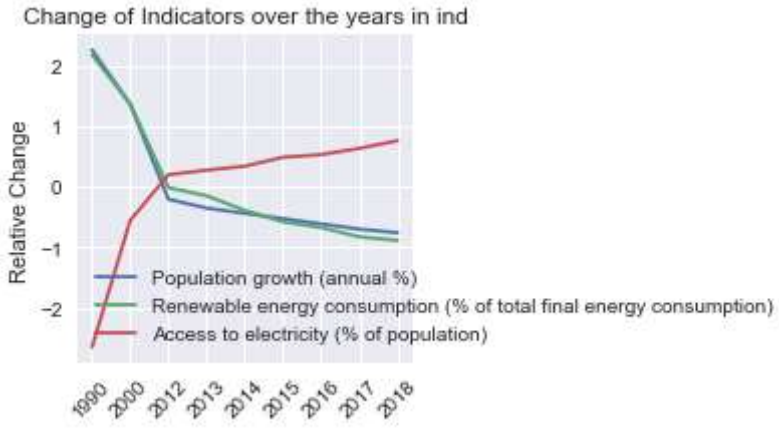
In [24]:
```
ind = df_merge[df_merge['Country Name'] == 'India']
ind = ind.drop(['Country Name'], axis = 1)
ind = ind.set_index('Series Name').T
```

In [25]:
```
def plot_country(data, name):
    plt.figure(figsize=(3,3))
    for i in data.columns:

        columns = data.columns
        years = ['1990','2000','2012','2013','2014','2015','2016','2017','2018']
        scaler = StandardScaler()
        #scaling the data using standard scaler and saving it with full column names
        country_sc = pd.DataFrame(scaler.fit_transform(data),columns=columns)
        country_sc.index = years
        plt.plot(country_sc.index,country_sc[i],label = i) #plotting the 3 lines
        plt.legend(loc = 'best') #setting the location of the legends
        plt.xticks(country_sc.index,rotation = 45) #changing the x axis values
        plt.ylabel('Relative Change') #setting y axis label
    plt.title('Change of Indicators over the years in %s'%name) #setting the title
    plt.show()
```

In [26]:
```
plot_country(usa,'usa')
```



In [27]:
```
plot_country(ind,'ind')
```



In [ ]:
```

```