How suicide rate has evolved since 1985 worldwide in regards of some socio-economic and demographic indicators ? ¶

The World Health Organisation (WHO) is a specialised agency for the United Nations that is concerned with international public health (Wikipedia). For a long period of time the WHO has collected data on the suicide worlwide since 1985.
According to the WHO close to 800 000 people die due to suicide every year, which is one person every 40 seconds. Suicide is a global phenomenon and occurs throughout the lifespan. Effective and evidence-based interventions can be implemented at population, sub-population and individual levels to prevent suicide and suicide attempts.

In [2]:
```
# Loading the libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
%matplotlib inline
```

In [3]:
```
# Plot settings
plt.style.use("ggplot")
plt.rcParams["figure.figsize"] = [12, 6]
plt.rcParams["figure.dpi"] = 150
```

In [4]:
```
# Reading the data into memory
suicide = pd.read_csv("master.csv")
```

Let's view the shape of the dataset.

In [5]:
```
suicide.shape
```

Out[5]:
```
(27820, 12)
```

The dataset contains 27820 observation and 14 columns.

In [6]:
```
suicide.columns
```

Out[6]:
```
Index(['country', 'year', 'sex', 'age', 'suicides_no', 'population',
       'suicides/100k pop', 'country-year', 'HDI for year',
       ' gdp_for_year ($) ', 'gdp_per_capita ($)', 'generation'],
      dtype='object')
```

In [7]:
```
suicide.head()
```

Out[7]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gd
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	2,1
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	2,1
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	2,1
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	2,1
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	2,1

For the purpose of consistency on the column names, let's redefine the columns `suicides/100k pop` , `country-year`, `gdp_for_year ($)` , `gdp_per_capita ($)`

In [8]:
```
replacement_cols = {"suicides/100k pop" : "suicide_100k", "country-year": "country_year",
                    "HDI for year": "hdi_for_year", " gdp_for_year ($) " : "current_gdp",
                    "gdp_per_capita ($)" : "gdp_capita"}
```

In [9]:
```
suicide = suicide.rename(columns = replacement_cols)
suicide["current_gdp"] = suicide["current_gdp"].str.replace(",", "").astype("float64")
```

In [10]:
```
suicide.head()
```

Out[10]:

	country	year	sex	age	suicides_no	population	suicide_100k	country_year	hdi_for_
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	

In [11]:
```
suicide.isna().sum()
```

Out[11]:
```
country            0
year               0
sex                0
age                0
suicides_no        0
population         0
suicide_100k       0
country_year       0
hdi_for_year    19456
current_gdp        0
gdp_capita         0
generation         0
dtype: int64
```

Fortunately the dataset doesn't have missing values except the `hdi_for_year` column.

Data dictionary

This short section aims to present what some of the columns stand for.

- suicide_100k : This column represents the number of death by suicide out of total of 100.000 deaths. You can think of it as percentage but with a base 100.000 instead of 100.
- hdi_for_year : It stands for the the Human Development Index for the year. It's between 0 and 1. Higher values represent higher life quality (education, health, and income)
- current_gdp : It stands for the the Gross Domestic Product (the amount of wealth produced in the country in one year).

Note on the structure of the data

As we can see on the first five rows of the dataset, there seem to be redundant entries but there are not. There seems to have redundant entries because of the levels of the categorical columns (sex, age, generation). Each row represents different entries with one or more changes in the categorical columns.

Exploration

Here we want to understand the dataset, the number of countries, the number of levels in different group categories, visualisations etc...

In [12]:
```
# Number of unique countries
len(pd.unique(suicide["country"]))
```

Out[12]:
```
101
```

There is a total of 101 unique countries in the dataset. But does this mean that every countries have the same number of occurences ? Let's check for the number of occurences

In [13]:
```
Counter(suicide["country"]).most_common()[:10]
```

Out[13]:
```
[('Austria', 382),
 ('Iceland', 382),
 ('Mauritius', 382),
 ('Netherlands', 382),
 ('Argentina', 372),
 ('Belgium', 372),
 ('Brazil', 372),
 ('Chile', 372),
 ('Colombia', 372),
 ('Ecuador', 372)]
```

In [14]:
```
Counter(suicide["country"]).most_common()[-10:]
```

Out[14]:
```
[('Nicaragua', 72),
 ('United Arab Emirates', 72),
 ('Oman', 36),
 ('Saint Kitts and Nevis', 36),
 ('San Marino', 36),
 ('Bosnia and Herzegovina', 24),
 ('Cabo Verde', 12),
 ('Dominica', 12),
 ('Macau', 12),
 ('Mongolia', 10)]
```

The dataset does not contain the same number of occurence for each country. This is mainly because the data is not available for every year for each country.
Let's explore the number of data points available for each years.

In [15]:
```
len(pd.unique(suicide["year"]))
```

Out[15]:
```
32
```

The dataset covers a period of 32 years. What is the range of the dataset ?

In [16]:
```
min(suicide["year"])
```

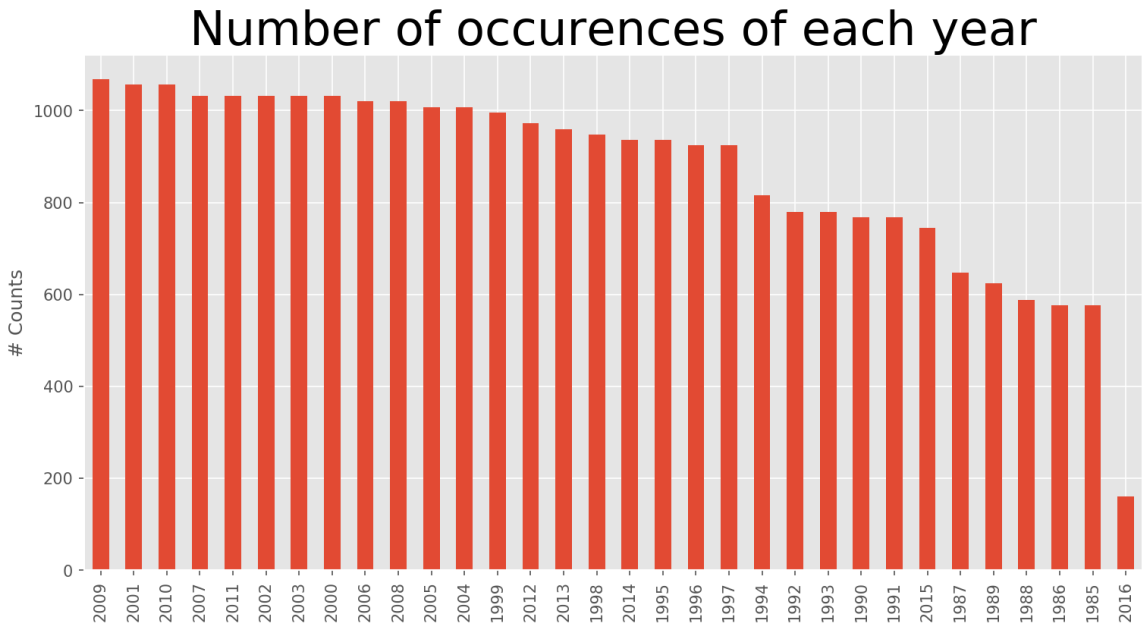
Out[16]:
```
1985
```

In [17]:
```
max(suicide["year"])
```

Out[17]:
```
2016
```

The data covers the period of 1985 to 2016

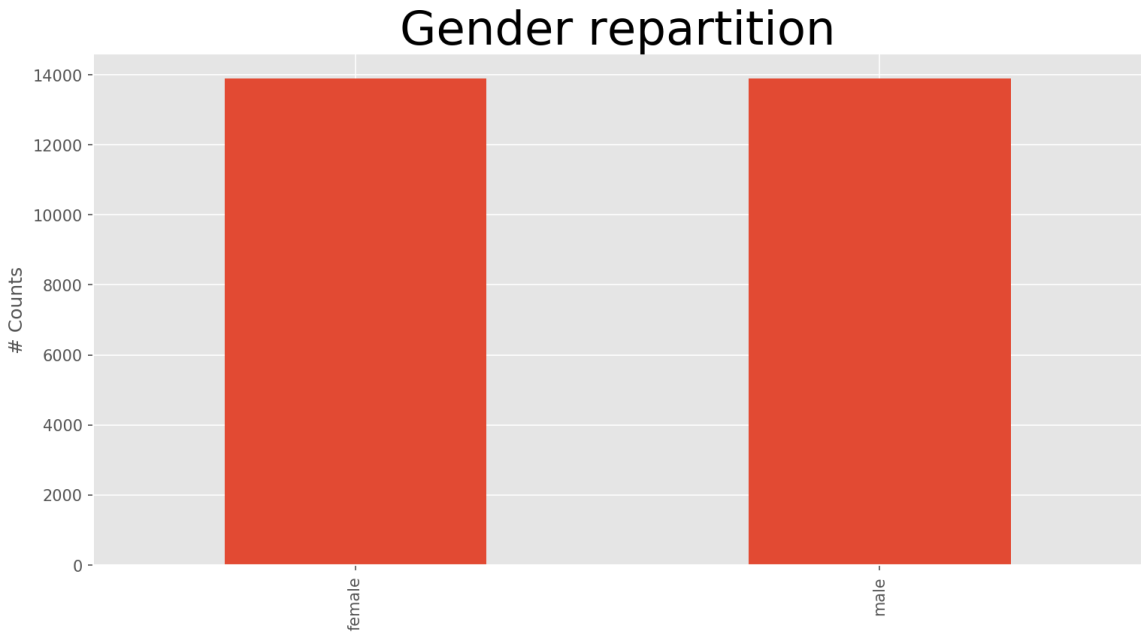
In [18]:
```
suicide["year"].value_counts().plot.bar()
plt.title("Number of occurences of each year", size = 30)
plt.ylabel("# Counts");
```



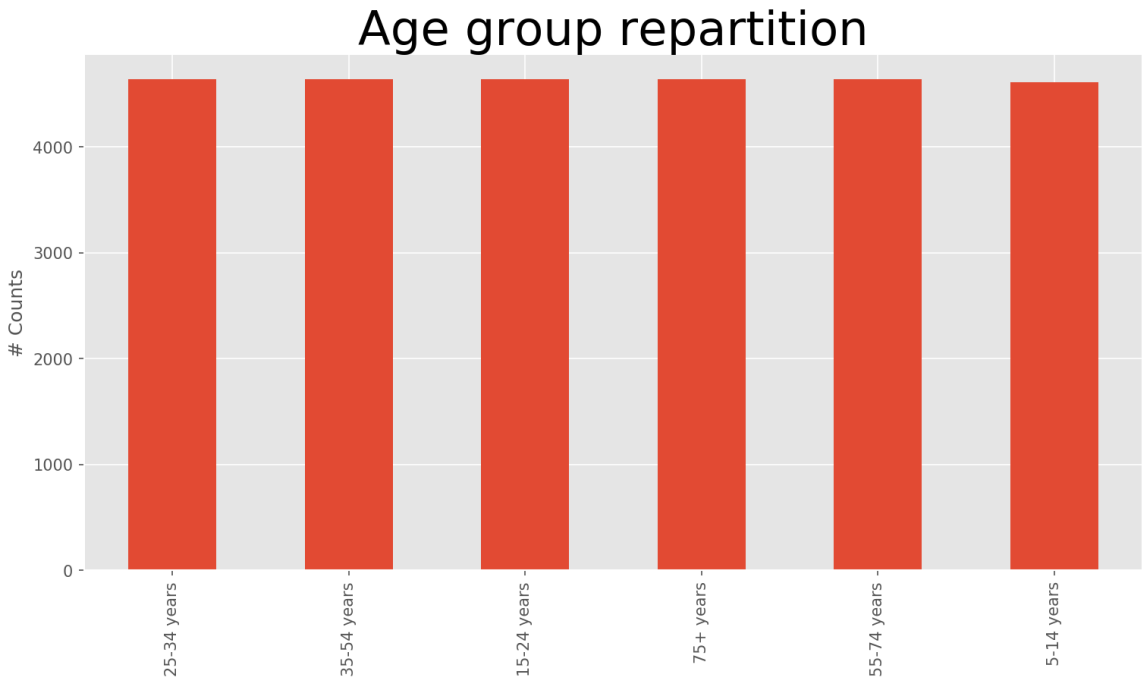
We don't have data for all the countries everytime. Some years have very few observation, that's the case of 2016.

Categorical variables

In [19]:
```
suicide["sex"].value_counts().plot.bar()
plt.title("Gender repartition", size = 30)
plt.ylabel("# Counts");
```

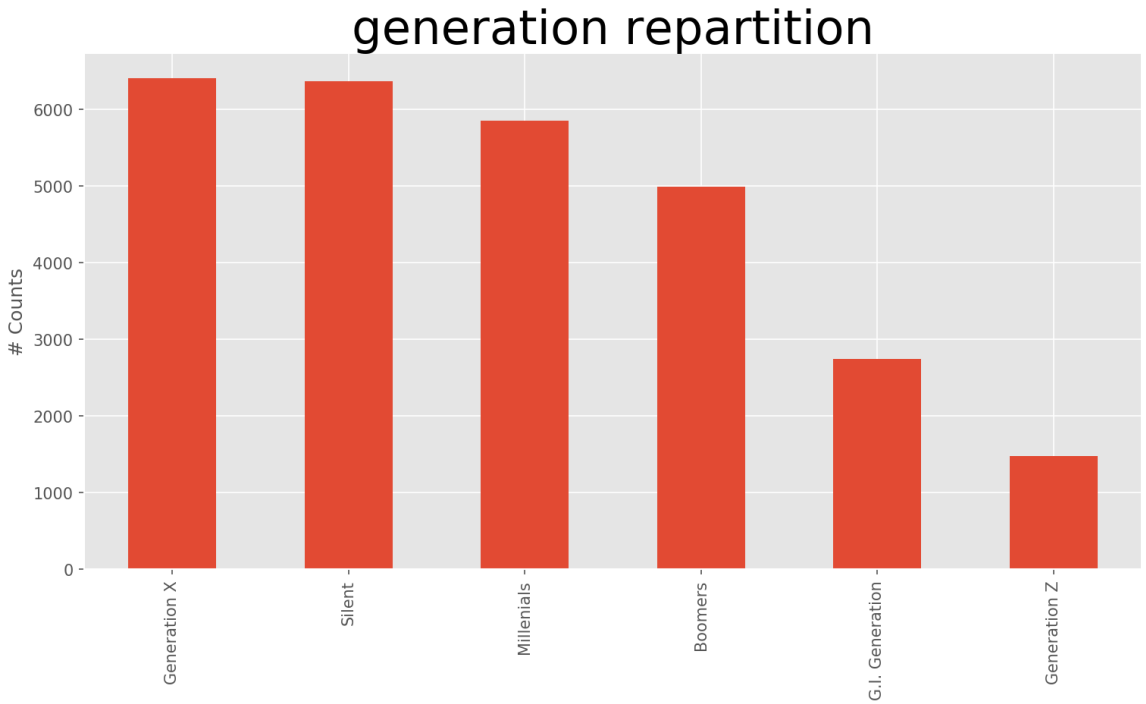


In [20]:
```
suicide["age"].value_counts().plot.bar()
plt.title("Age group repartition", size = 30)
plt.ylabel("# Counts");
```



The two previous columns are well balanced

In [21]:
```
suicide["generation"].value_counts().plot.bar()
plt.title("generation repartition", size = 30)
plt.ylabel("# Counts");
```

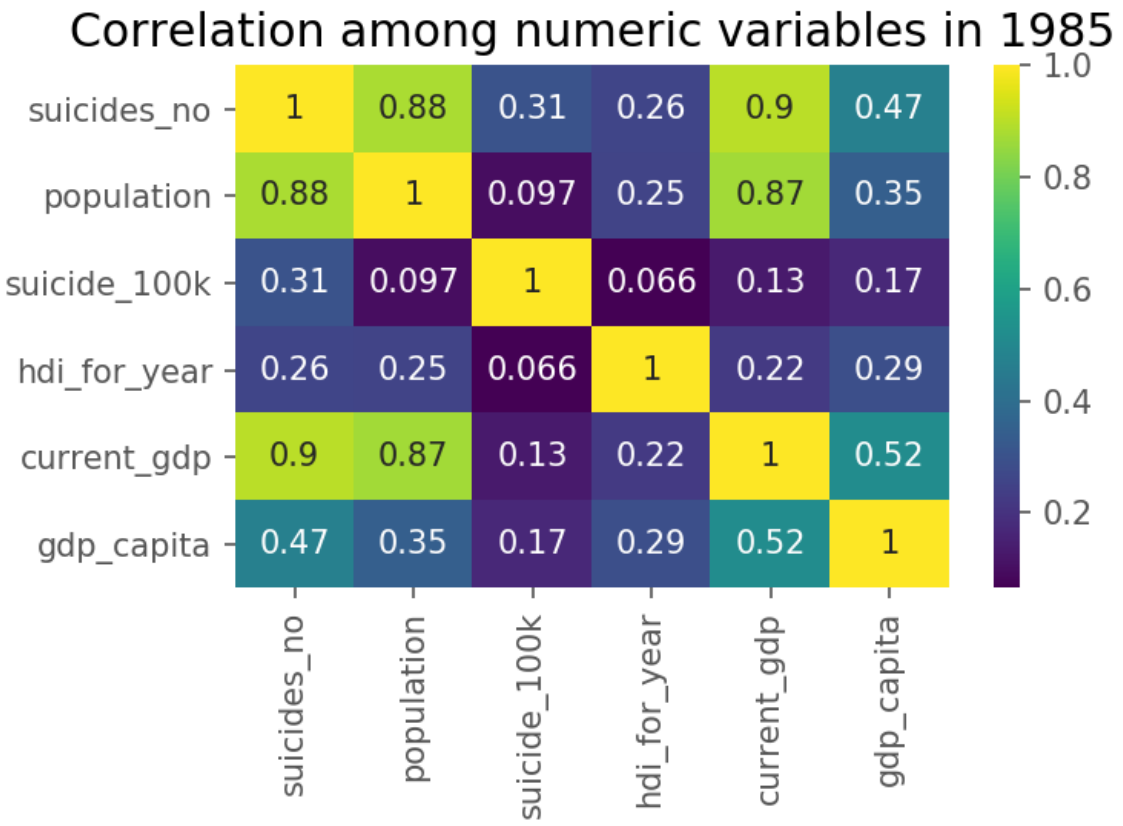


Correlation among numeric variables

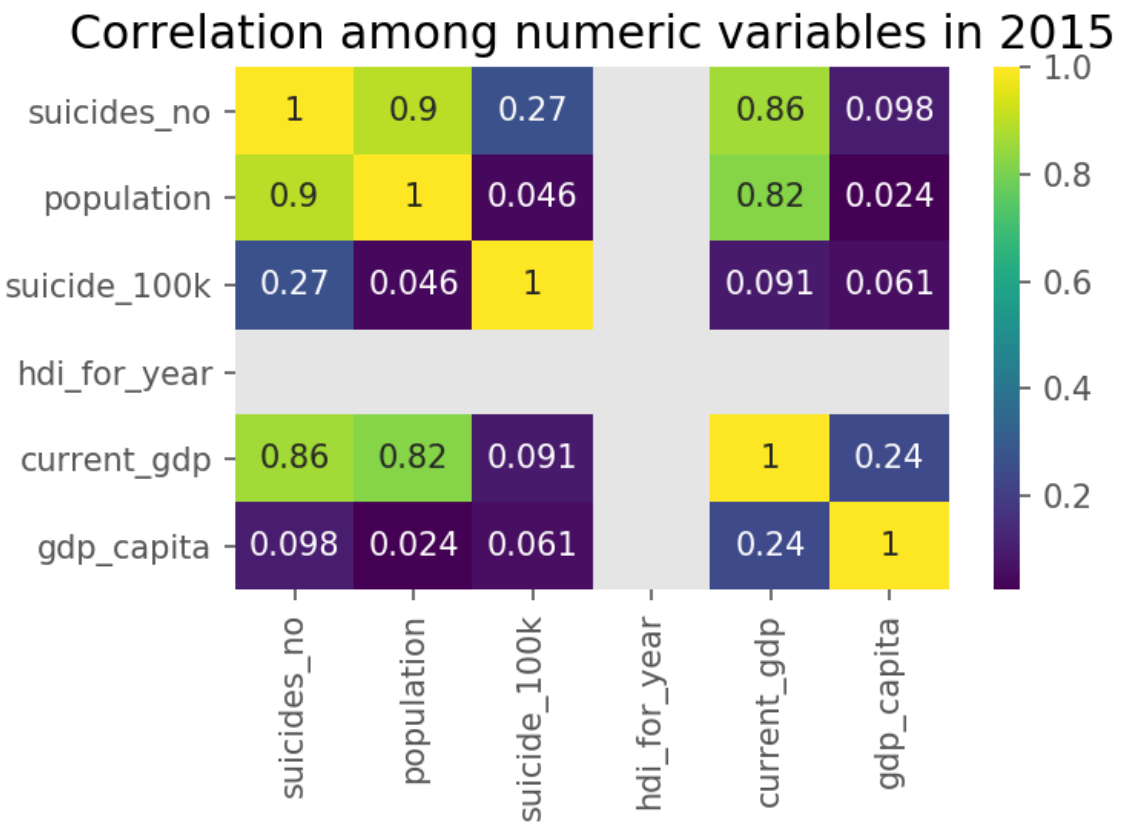
We need to group the data in order to aggregate the values that are spread into multiple rows.

In [22]:
```
def correlation_plot(data, year):
    df = suicide[suicide["year"] == year].drop(columns = "year").copy()
    sns.heatmap(
    df.groupby("country").agg("sum").corr(), cmap = "viridis", annot = True)
    plt.title(f"Correlation among numeric variables in {year}")
    plt.show();
```

In [23]:
```
plt.figure(figsize= (5, 3))
correlation_plot(suicide, 1985)
```



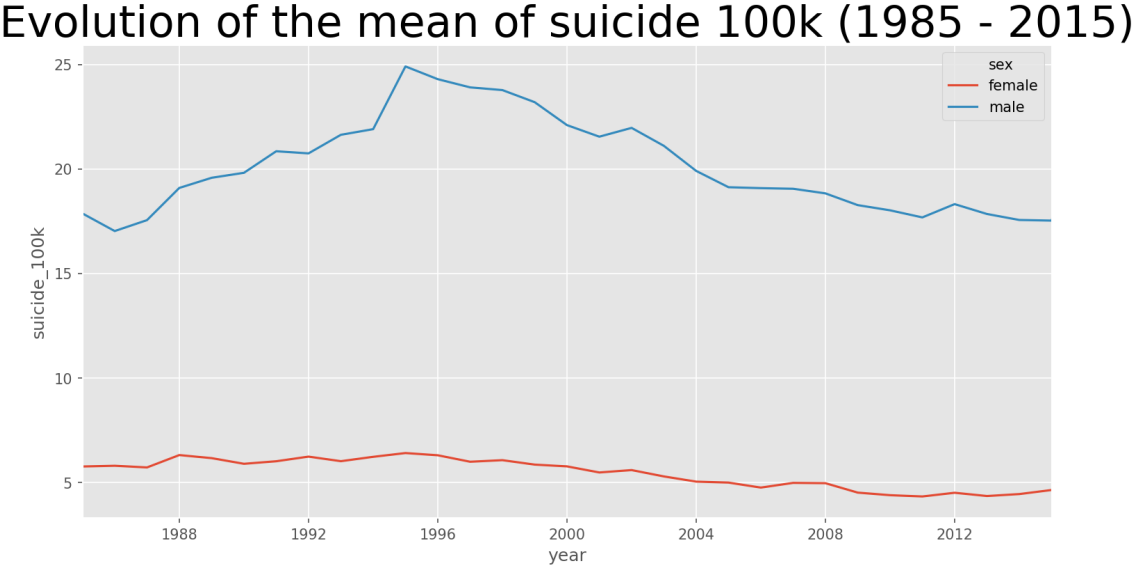
In [24]:
```
plt.figure(figsize= (5, 3))
correlation_plot(suicide, 2015)
```



Evolution of the variables

In [25]:
```
suicide["year"] = pd.to_datetime(suicide["year"], format = "%Y")
```

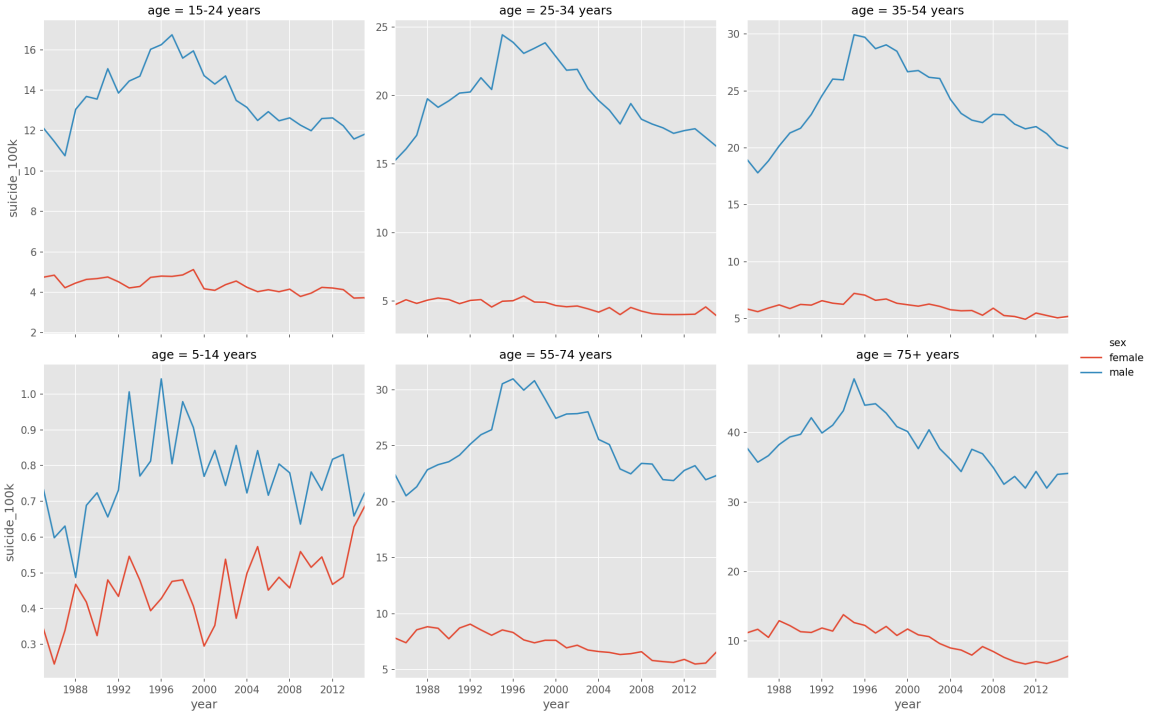
In [26]:
```
df = suicide.groupby(["year", "sex"]).agg("mean").reset_index()
sns.lineplot(x = "year", y = "suicide_100k", hue = "sex", data = df)
plt.xlim("1985", "2015")
plt.title("Evolution of the mean of suicide 100k (1985 - 2015)", size = 30);
```



In [27]:
```
df = suicide.groupby(["year", "sex", "age"]).agg("mean").reset_index()

sns.relplot(x = "year", y = "suicide_100k",
            hue = "sex", col = "age", col_wrap = 3, data = df,
            facet_kws=dict(sharey=False), kind = "line")

plt.xlim("1985", "2015");
```



There is a significant gap between the suicide rate of man and woman.

Top / Bottom countries in variables

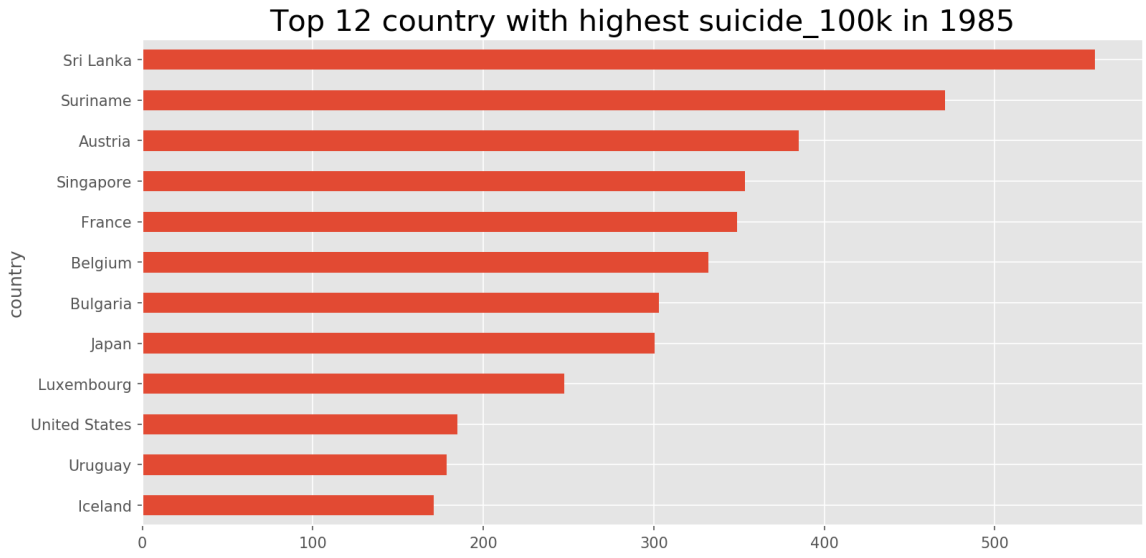
Let's define a function that plots horizontal barplots to represent the countries with higher or smallest value for variables.

In [28]:
```
def barplot(year:int, nb:int, column = "suicide_100k"):
    """
    This function plots the top / bottom n countries of the specified variable.
    """
    df = suicide.groupby(["year", "country"]).agg("sum").reset_index()
    suicide_rate_year = df[df["year"] == f"{year}-01-01"].set_index("country")[column]
    if nb > 0:
        suicide_rate_year.nlargest(nb).sort_values().plot(kind = "barh")
        plt.title(f"Top {nb} country with highest {column} in {year}", size = 20)
        plt.show()
    elif nb < 0:
        suicide_rate_year.nsmallest(abs(nb)).sort_values().plot(kind = "barh")
        plt.title(f" Top {abs(nb)} countries with smallest {column} in {year}", size = 20)
        plt.show()
```

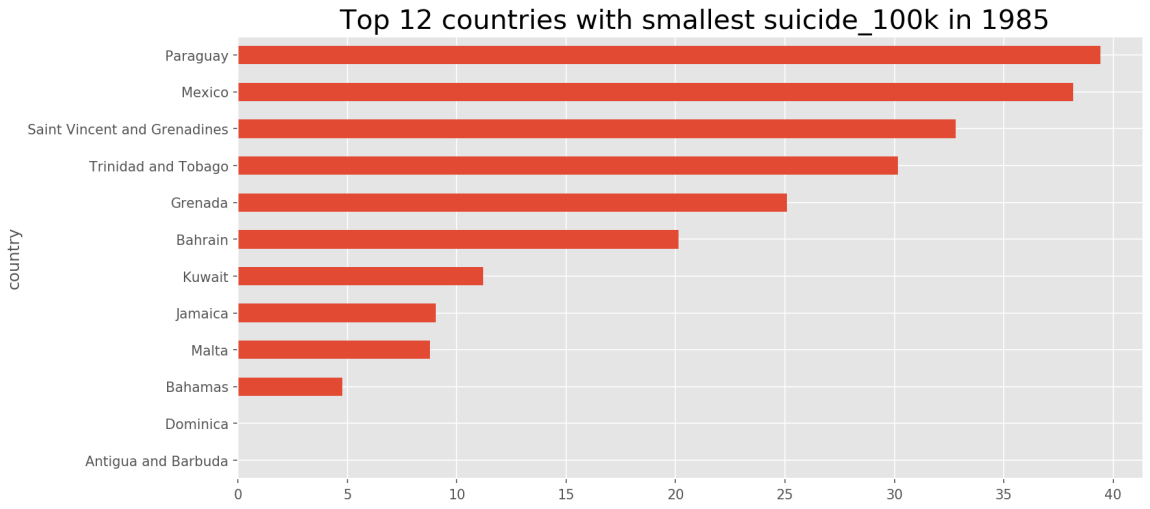
Let's now use the function

Countries with highest suicide rate over years

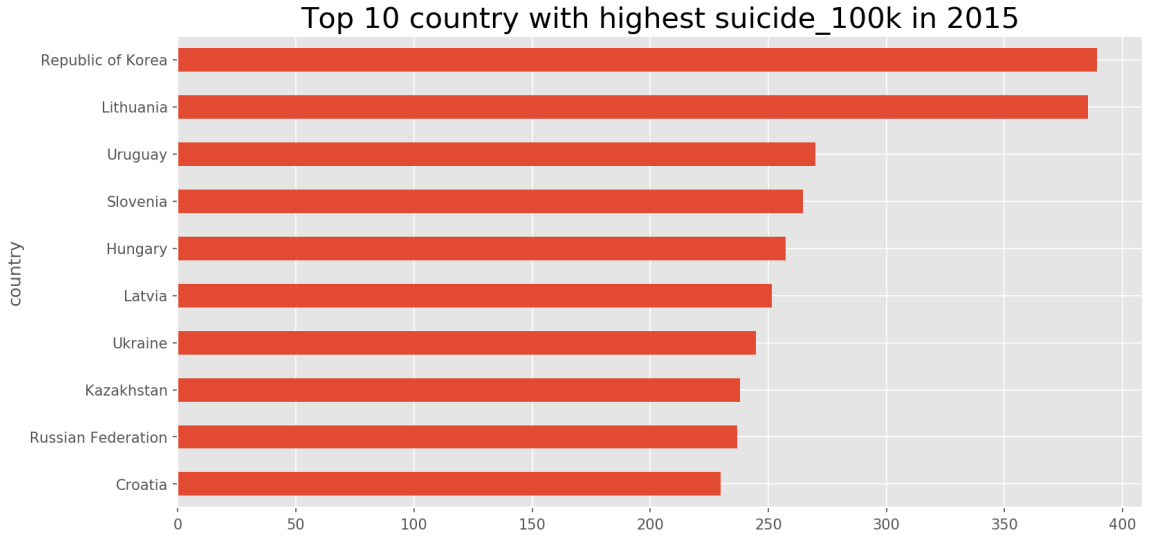
In [29]:
```
barplot(year= 1985, nb = 12, column= "suicide_100k")
```



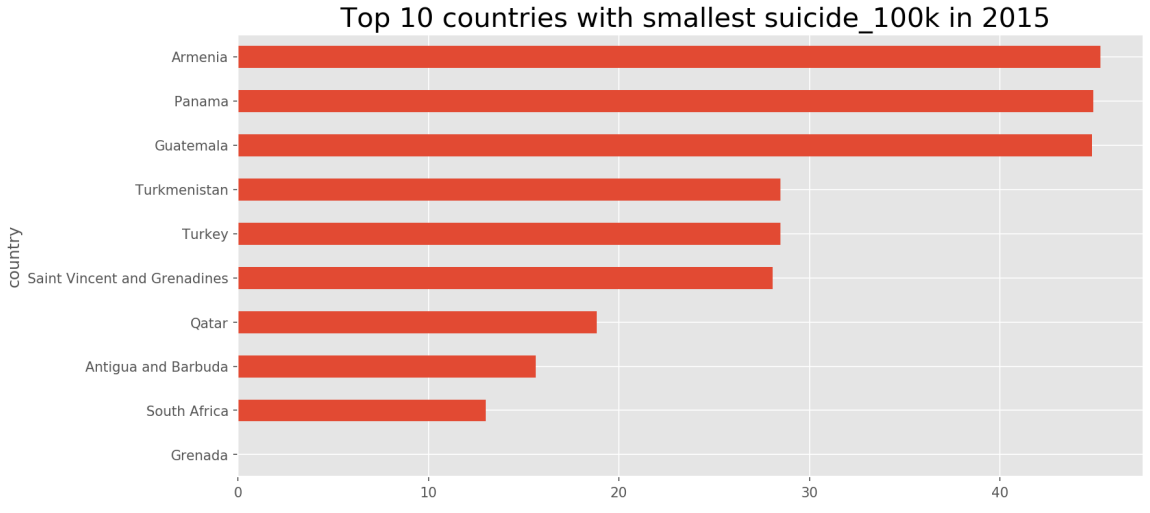
In [30]:
```
barplot(year= 1985, nb = -12, column= "suicide_100k")
```



In [31]:
```
barplot(2015, 10, "suicide_100k")
```



In [32]:
```
barplot(2015, -10, "suicide_100k")
```



Link between the evolution of suicide and the evolution of other variables

We want to see whether there has been links amongs the numeric variables

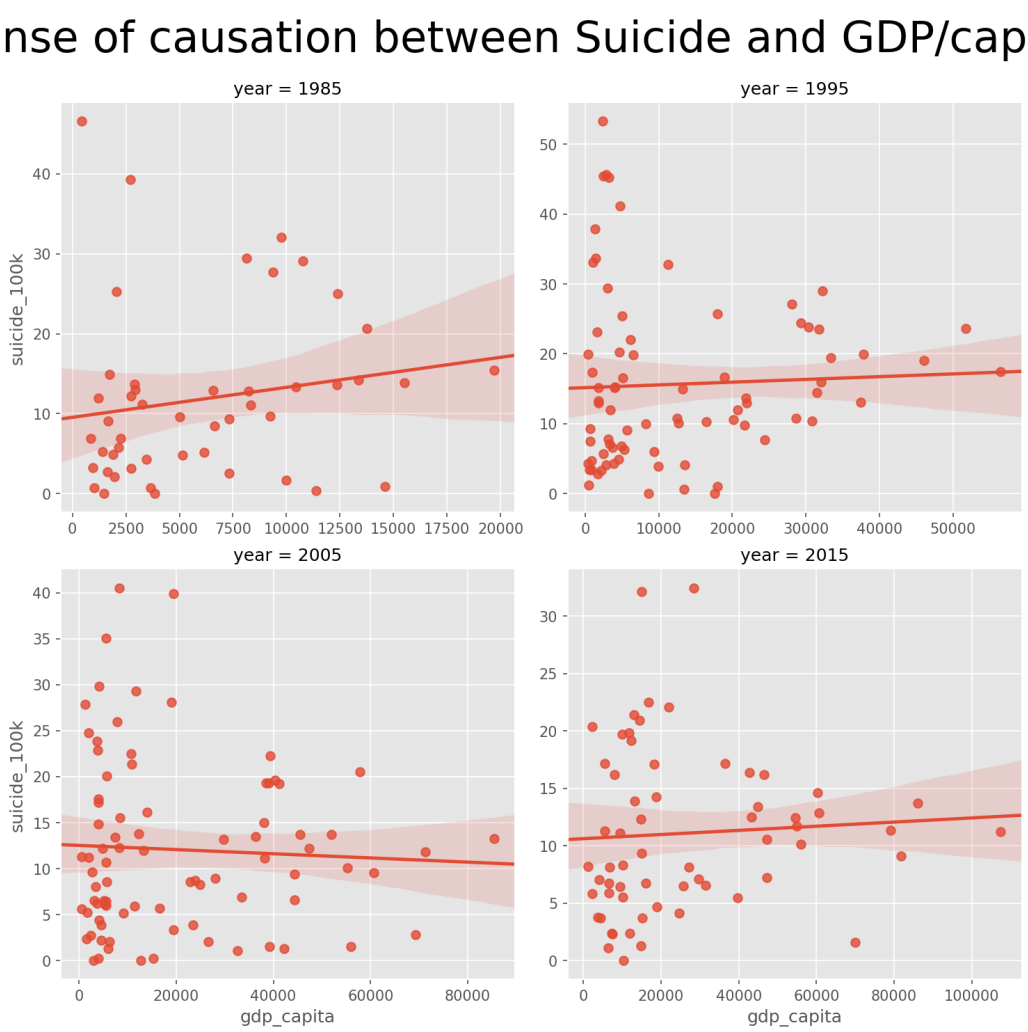
In [68]:
```
def plot_scatter_links(x:str, y:str, years:list,
                       agg_fun = "sum", n_col = None):

    df = suicide[suicide["year"].isin(years)]

    df = df.groupby(["year", "country"]).agg(agg_fun).reset_index()
    df["year"] = df["year"].dt.year.astype("object")
    sns.lmplot(x, y, scatter = True, fit_reg= True,
               data = df, col = "year", col_wrap = n_col, sharex = False, sh
arey = False)
```

Suicide vs GDP per capita

In [70]:
```
plot_scatter_links(x = "gdp_capita", y = "suicide_100k",
                   years = ["1985","1995", "2005", "2015"], n_col= 2, agg_fu
n= "mean")
plt.subplots_adjust(top=.9)
plt.suptitle("Sense of causation between Suicide and GDP/capita", size = 30
);
```



Cluster Analysis

In [37]:
```
suicide.head()
```

Out[37]:

	country	year	sex	age	suicides_no	population	suicide_100k	country_year	hdi_for
0	Albania	1987-01-01	male	15-24 years	21	312900	6.71	Albania1987	
1	Albania	1987-01-01	male	35-54 years	16	308000	5.19	Albania1987	
2	Albania	1987-01-01	female	15-24 years	14	289700	4.83	Albania1987	
3	Albania	1987-01-01	male	75+ years	1	21800	4.59	Albania1987	
4	Albania	1987-01-01	male	25-34 years	9	274300	3.28	Albania1987	

Before we aggregate the data, it's important we track every single occurence so we can have back the original values. For that purpose I'm adding a new column called `occurence` which gives tha value 1 to every row so that whenever we group and sum the numeric data we can come back to the original data by dividing it.

In [38]:
```
suicide["occurence"] = 1
suicide["part_generation"] = 0
```

In [39]:
```
suicide_wide = suicide.groupby(["year", "country"]).agg("sum").reset_index()

suicide_wide["year"] = suicide_wide["year"].dt.year
```

In [40]:
```
suicide_wide.head(10)
```

Out[40]:

	year	country	suicides_no	population	suicide_100k	hdi_for_year	current_gdp	gdp_ca
0	1985	Antigua and Barbuda	0	62574	0.00	0.000	2.891087e+09	46
1	1985	Argentina	1988	27090800	134.47	8.328	1.061000e+12	39
2	1985	Australia	1861	14562500	163.41	0.000	2.162292e+12	148
3	1985	Austria	2091	7110115	384.81	9.168	8.326413e+11	117
4	1985	Bahamas	1	203700	4.76	0.000	2.784840e+10	136
5	1985	Bahrain	11	365900	20.12	8.724	4.382234e+10	119
6	1985	Barbados	7	231000	61.72	8.400	1.701091e+10	73
7	1985	Belgium	2281	9269600	332.04	9.288	1.040760e+12	112
8	1985	Brazil	4228	117478900	59.18	6.912	2.675313e+12	22
9	1985	Bulgaria	1456	8392500	303.12	8.232	2.058651e+11	24

In [41]:
```
suicide_wide.shape
```

Out[41]:
```
(2321, 10)
```

Make the dataset wider

Here I want to have a column for each of the generation. For that purpose I'll use the pivot function.

In [42]:
```
interm = suicide.pivot(columns = "generation", values = "population")
suicide_new = pd.concat([suicide, interm], axis = 1).fillna(0)
suicide_new = suicide_new.groupby(["year", "country"]).agg("sum").reset_inde
x()
```

In [43]:
```
suicide_new.shape
```

Out[43]:
```
(2321, 16)
```

In [44]:
```
suicide_new["current_gdp"] = suicide_new["current_gdp"] / suicide_new["occur
ence"]
suicide_new["gdp_capita"] = suicide_new["gdp_capita"] / suicide_new["occuren
ce"]
suicide_new["hdi_for_year"] = suicide_new["hdi_for_year"] / suicide_new["occ
urence"]
del suicide_new["occurence"]
del suicide_new["population"]
```

In [45]:
```
suicide_new.head()
```

Out[45]:

	year	country	suicides_no	suicide_100k	hdi_for_year	current_gdp	gdp_capita	part_g
0	1985-01-01	Antigua and Barbuda	0	0.00	0.000	2.409239e+08	3850.0	
1	1985-01-01	Argentina	1988	134.47	0.694	8.841667e+10	3264.0	
2	1985-01-01	Australia	1861	163.41	0.000	1.801910e+11	12374.0	
3	1985-01-01	Austria	2091	384.81	0.764	6.938677e+10	9759.0	
4	1985-01-01	Bahamas	1	4.76	0.000	2.320700e+09	11393.0	

Now we have a column for each of the generation.

Scaling the dataset

It's now important to scale the dataset so each column will have the same weight. This is important because of the distance metric the KNN algorithm uses.

In [46]:
```
from sklearn.preprocessing import StandardScaler
```

In [47]:
```
X = StandardScaler()
```

In [48]:
```
X = X.fit_transform(suicide_new.drop(["year", "country"], axis = 1))
```

In [49]:
```
selected_columns = ['suicides_no', 'suicide_100k', 'hdi_for_year',
       'current_gdp', 'gdp_capita', 'part_generation', 'Boomers',
       'G.I. Generation', 'Generation X', 'Generation Z', 'Millenials',
       'Silent']

suicide_new[selected_columns] = X
```

In [50]:
```
suicide_new.head()
```

Out[50]:

	year	country	suicides_no	suicide_100k	hdi_for_year	current_gdp	gdp_capita	part_ge
0	1985-01-01	Antigua and Barbuda	-0.410885	-1.313957	-0.648461	-0.306332	-0.689515	
1	1985-01-01	Argentina	-0.129948	-0.163769	1.281230	-0.245639	-0.720533	
2	1985-01-01	Australia	-0.147895	0.083769	-0.648461	-0.182468	-0.238327	
3	1985-01-01	Austria	-0.115392	1.977512	1.475867	-0.258737	-0.376743	
4	1985-01-01	Bahamas	-0.410744	-1.273243	-0.648461	-0.304901	-0.290253	

Kmeans clustering

for year 1985

In [73]:
```
from sklearn.cluster import KMeans
data = suicide_new[suicide_new["year"] == "1985"].drop(["year", "suicides_no"],
                                                       axis = 1).set_index("country")
```

In [74]:
```
model = KMeans(init= "k-means++", n_clusters= 4, n_init = 20)
model = model.fit(data)
```

In [75]:
```
pd.value_counts(model.labels_, sort = False)
```

Out[75]:
```
0     7
1    33
2     7
3     1
dtype: int64
```

In [76]:
```
k = 4
clusters = []
for j in range(0,k):
    clusters.append([])
for i in range(0,data.shape[0]):
    clusters[model.labels_[i]].append(data.index[i])
#
# Print out clusters
#
for j in range(0,k):
    print(30*"-", "\n")
    print( j, clusters[j])
```

```
------------------------------

0 ['Austria', 'Belgium', 'Bulgaria', 'Luxembourg', 'Singapore', 'Sri Lank
a', 'Suriname']
------------------------------

1 ['Antigua and Barbuda', 'Argentina', 'Australia', 'Bahamas', 'Bahrain',
'Barbados', 'Canada', 'Chile', 'Colombia', 'Costa Rica', 'Dominica', 'Ecuad
or', 'Greece', 'Grenada', 'Iceland', 'Ireland', 'Israel', 'Jamaica', 'Kuwai
t', 'Malta', 'Mauritius', 'Netherlands', 'New Zealand', 'Panama', 'Paragua
y', 'Portugal', 'Puerto Rico', 'Republic of Korea', 'Saint Vincent and Gren
adines', 'Seychelles', 'Thailand', 'Trinidad and Tobago', 'Uruguay']
------------------------------

2 ['Brazil', 'France', 'Italy', 'Japan', 'Mexico', 'Spain', 'United Kingdo
m']
------------------------------

3 ['United States']
```

For year 2015

In [77]:
```
from sklearn.cluster import KMeans
data = suicide_new[suicide_new["year"] == "2015"].drop(["year", "suicides_n
o"],
                                                       axis = 1).set_index(
"country")
model = KMeans(init= "k-means++", n_clusters= 4, n_init = 20)
model = model.fit(data)
```

In [78]:
```
k = 4
clusters = []
for j in range(0,k):
    clusters.append([])
for i in range(0,data.shape[0]):
    clusters[model.labels_[i]].append(data.index[i])
#
# Print out clusters
#
for j in range(0,k):
    print(30*"-", "\n")
    print( j, clusters[j])
```

```
------------------------------

0 ['Australia', 'Austria', 'Belgium', 'Denmark', 'Finland', 'Germany', 'Ice
land', 'Israel', 'Luxembourg', 'Netherlands', 'Norway', 'Qatar', 'Singapor
e', 'Sweden', 'Switzerland', 'United Kingdom']
------------------------------

1 ['Brazil', 'Japan', 'Mexico', 'Russian Federation', 'Turkey']
------------------------------

2 ['United States']
------------------------------

3 ['Antigua and Barbuda', 'Argentina', 'Armenia', 'Belize', 'Chile', 'Colom
bia', 'Croatia', 'Cuba', 'Cyprus', 'Czech Republic', 'Ecuador', 'Estonia',
'Georgia', 'Greece', 'Grenada', 'Guatemala', 'Hungary', 'Italy', 'Kazakhsta
n', 'Kyrgyzstan', 'Latvia', 'Lithuania', 'Malta', 'Mauritius', 'Nicaragua',
'Panama', 'Poland', 'Puerto Rico', 'Republic of Korea', 'Romania', 'Saint V
incent and Grenadines', 'Serbia', 'Seychelles', 'Slovenia', 'South Africa',
'Spain', 'Thailand', 'Turkmenistan', 'Ukraine', 'Uruguay']
```

Hierarchical clustering algorithm

According to the KMeans Algorithms this is how similar are the countries in 1985. Let's build something much more deeper

In [79]:
```
from scipy.cluster import hierarchy as sch
```

In [80]:
```
z = sch.linkage(data, method = "ward")
```

In [81]:
```
info = sch.dendrogram(z, orientation = "top", labels= data.index)
plt.title("Countries similarities in 2015", size = 30);
```