world-population-prediction

June 18, 2024

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Population Growth Evolution and Prediction !

The Project is divided into 4 parts-

1.Data prepartion + Web scrapping.

2.Global population growth analysis in the world's regions.

3.Specific case China vs India.

4.Prediction of the world population for the next year.

DATA PREPARATION

[4]:
```python
PercGrow=pd.read_csv("C:\\Users\\shind\\Dropbox\\PC\\Downloads\\Population␣
↪growth.csv")
```

[5]:
```python
PercGrow.shape
```

[5]:
```
(264, 65)
```

[6]:
```python
PercGrow.head()
```

[6]:
```
  Country Name Country Code                Indicator Name Indicator Code  \
0        Aruba          ABW  Population growth (annual %)    SP.POP.GROW
1  Afghanistan          AFG  Population growth (annual %)    SP.POP.GROW
2       Angola          AGO  Population growth (annual %)    SP.POP.GROW
3      Albania          ALB  Population growth (annual %)    SP.POP.GROW
4      Andorra          AND  Population growth (annual %)    SP.POP.GROW

   1960      1961      1962      1963      1964      1965  …      2011  \
0   NaN  2.238144  1.409622  0.832453  0.592649  0.573468  …  0.370125
1   NaN  1.898476  1.965751  2.029893  2.090248  2.147567  …  3.143126
2   NaN  1.393363  1.383629  1.256555  0.973962  0.617544  …  3.634159
3   NaN  3.120855  3.056731  2.953749  2.880686  2.754021  … -0.269017
```

```
4   NaN  6.941532  6.692697  6.559522  6.241511  5.998800  … -0.834745

        2012      2013      2014      2015      2016      2017      2018  \
0  0.502430  0.582349  0.594397  0.544892  0.507618  0.469944  0.453576
1  3.407587  3.494589  3.355582  3.077084  2.778317  2.548347  2.384761
2  3.597774  3.551950  3.497493  3.438851  3.378273  3.322081  3.276134
3 -0.165151 -0.183211 -0.207047 -0.291206 -0.159880 -0.091972 -0.246732
4 -1.588730 -2.025792 -1.951470 -1.529058 -0.919470 -0.383674  0.006493

        2019  2020
0  0.442122   NaN
1  2.311817   NaN
2  3.242983   NaN
3 -0.426007   NaN
4  0.176454   NaN

[5 rows x 65 columns]
```

The first dataset contains the country code and name and one column for each year (from 1960 to 2020)

[8]:
```
IncomeGroup=pd.read_csv("C:
↪\\Users\\shind\\Dropbox\\PC\\Downloads\\IncomeGroupdata.csv")
IncomeGroup.head()
```

[8]:
```
  Country Code                     Region          IncomeGroup SpecialNotes  \
0          ABW  Latin America & Caribbean          High income          NaN
1          AFG                 South Asia           Low income          NaN
2          AGO         Sub-Saharan Africa  Lower middle income          NaN
3          ALB      Europe & Central Asia  Upper middle income          NaN
4          AND      Europe & Central Asia          High income          NaN

     TableName
0        Aruba
1  Afghanistan
2       Angola
3      Albania
4      Andorra
```

The second set of data tells us about the income group of each country as well as the region around the world.

Thanks to the country code, we will be able to join these data to the main dataset if we need it.

[9]:
```
print(f"{PercGrow[PercGrow.columns[2:4]].value_counts()} \n")


print(PercGrow['1960'].isnull().sum() )
```

```
print(PercGrow['2020'].isnull().sum() )


#column 1960 & 2020 contains null values

PercGrow=PercGrow.drop(columns=['1960','2020'],axis=1)
PercGrow = PercGrow.drop(PercGrow.columns[2:4],axis=1)
PercGrow.head()
```

```
Indicator Name                Indicator Code
Population growth (annual %)  SP.POP.GROW       264
dtype: int64

264
264
```

[9]:

```
  Country Name Country Code      1961      1962      1963      1964      1965  \
0        Aruba          ABW  2.238144  1.409622  0.832453  0.592649  0.573468
1  Afghanistan          AFG  1.898476  1.965751  2.029893  2.090248  2.147567
2       Angola          AGO  1.393363  1.383629  1.256555  0.973962  0.617544
3      Albania          ALB  3.120855  3.056731  2.953749  2.880686  2.754021
4      Andorra          AND  6.941532  6.692697  6.559522  6.241511  5.998800

       1966      1967      1968  …      2010      2011      2012      2013  \
0  0.616991  0.587373  0.568530  …  0.210709  0.370125  0.502430  0.582349
1  2.171009  2.188108  2.254572  …  2.746576  3.143126  3.407587  3.494589
2  0.184283 -0.120653 -0.044882  …  3.671462  3.634159  3.597774  3.551950
3  2.634564  2.630190  2.842511  … -0.496462 -0.269017 -0.165151 -0.183211
4  5.750878  5.500706  5.309820  … -0.016577 -0.834745 -1.588730 -2.025792

       2014      2015      2016      2017      2018      2019
0  0.594397  0.544892  0.507618  0.469944  0.453576  0.442122
1  3.355582  3.077084  2.778317  2.548347  2.384761  2.311817
2  3.497493  3.438851  3.378273  3.322081  3.276134  3.242983
3 -0.207047 -0.291206 -0.159880 -0.091972 -0.246732 -0.426007
4 -1.951470 -1.529058 -0.919470 -0.383674  0.006493  0.176454

[5 rows x 61 columns]
```

[10]:

```
PercGrow.iloc[:,4:].isnull().sum()
```

[10]:

```
1963    4
1964    4
1965    4
1966    4
1967    4
1968    4
```

1969	4
1970	4
1971	4
1972	4
1973	4
1974	4
1975	4
1976	4
1977	4
1978	4
1979	4
1980	4
1981	4
1982	4
1983	4
1984	4
1985	4
1986	4
1987	4
1988	4
1989	4
1990	3
1991	3
1992	3
1993	3
1994	3
1995	3
1996	2
1997	2
1998	2
1999	1
2000	1
2001	1
2002	1
2003	1
2004	1
2005	1
2006	1
2007	1
2008	1
2009	1
2010	1
2011	1
2012	2
2013	2
2014	2
2015	2

```
2016    2
2017    2
2018    2
2019    2
dtype: int64
```

There are still NaN values, but by removing the countries with missing values from the year 1961, we end up with a cleaner dataset

[11]:
```
PercGrow = PercGrow.drop(PercGrow[PercGrow["1961"].isnull() == True].index,␣
  ↪axis=0) #Null values dropped
PercGrow[PercGrow.columns[5:-1]].isnull().sum()
PercGrow = PercGrow.fillna(0)
```

From now, the cases seem to be isolated, so one can claim that one NaN means a 0

[12]:
```
PercGrow.head()
```

[12]:
```
  Country Name Country Code      1961      1962      1963      1964      1965  \
0        Aruba          ABW  2.238144  1.409622  0.832453  0.592649  0.573468
1  Afghanistan          AFG  1.898476  1.965751  2.029893  2.090248  2.147567
2       Angola          AGO  1.393363  1.383629  1.256555  0.973962  0.617544
3      Albania          ALB  3.120855  3.056731  2.953749  2.880686  2.754021
4      Andorra          AND  6.941532  6.692697  6.559522  6.241511  5.998800

       1966      1967      1968  …      2010      2011      2012      2013  \
0  0.616991  0.587373  0.568530  …  0.210709  0.370125  0.502430  0.582349
1  2.171009  2.188108  2.254572  …  2.746576  3.143126  3.407587  3.494589
2  0.184283 -0.120653 -0.044882  …  3.671462  3.634159  3.597774  3.551950
3  2.634564  2.630190  2.842511  … -0.496462 -0.269017 -0.165151 -0.183211
4  5.750878  5.500706  5.309820  … -0.016577 -0.834745 -1.588730 -2.025792

       2014      2015      2016      2017      2018      2019
0  0.594397  0.544892  0.507618  0.469944  0.453576  0.442122
1  3.355582  3.077084  2.778317  2.548347  2.384761  2.311817
2  3.497493  3.438851  3.378273  3.322081  3.276134  3.242983
3 -0.207047 -0.291206 -0.159880 -0.091972 -0.246732 -0.426007
4 -1.951470 -1.529058 -0.919470 -0.383674  0.006493  0.176454

[5 rows x 61 columns]
```

ADD Population size with WEB SCRAPPING-

The % growth tells us how much a population has increased in comparison to the previous year. However, our dataset does not provide any information on the size of a population.

We need population of 1961 ,brought it from website

[13]:
```python
#Libraries used for webscrapping-


from bs4 import BeautifulSoup
import requests
import re
```

[14]:
```python
#website url
url="https://www.bluemarblecitizen.com/world-population/1961"

#website availability
try:
    page=requests.get(url)

except:
    Print('Problem with requests')

#getting text data from website and then extracting table text,results contains␣
 ↪tables

soup=BeautifulSoup(page.content,"html.parser")
results = soup.find_all(class_="popTable")


def cleanhtml(raw_html):
    return re.sub(re.compile('<.*?>'),'/',raw_html).split('//')[1:-2]

dataset=[]

for table in results:
    job_section=table.find_all("tr")
    for job in job_section:
        scapped=[item.replace("/","") for item in cleanhtml(str(job))[1:3]]
        if scapped not in dataset:
            dataset.append(scapped)



d={str(PercGrow.columns[0]):[item[0] for item in dataset[1:]]␣
 ↪,'Population_1961':[int(item[1].replace(",","")) for item in dataset[1:] ]}

population_size_1961=pd.DataFrame(d)
population_size_1961.head()
```

[14]:

	Country Name	Population_1961
0	China	645409760
1	India	454425502

```
2  United States        182992000
3         Russia        121324346
4      Indonesia        102381963
```

[15]:
```
TestNaN=pd.merge(PercGrow,population_size_1961,on=PercGrow.
 ↪columns[0],how='left')
TestNaN[TestNaN['Population_1961'].isnull()==True]

#Just to check NaN values we would have to deal with
```

[15]:
```
                       Country Name Country Code      1961      1962  \
5                        Arab World          ARB  2.740584  2.755287
21                     Bahamas, The          BHS  4.974875  5.055858
29                Brunei Darussalam          BRN  4.616278  4.478505
34   Central Europe and the Baltics          CEB  0.909144  0.842174
36                  Channel Islands          CHI  0.890739  0.953779
..                              …            …         …         …
247  St. Vincent and the Grenadines          VCT  1.461294  1.291386
248                   Venezuela, RB          VEN  3.587842  3.533313
250           Virgin Islands (U.S.)          VIR  5.390526  2.020271
253                           World          WLD  1.353920  1.724198
256                     Yemen, Rep.          YEM  1.450869  1.484101

          1963      1964      1965      1966      1967       1968  …  \
5     2.773671  2.797625  2.823683  2.858219  2.887821   2.892612  …
21    5.031049  4.881298  4.639797  4.415588  4.173153   3.873674  …
29    4.436284  4.465812  4.570272  4.663632  4.710458   4.732895  …
34    0.892939  0.933268  0.765652  0.739041  0.932884   0.866368  …
36    1.015844  1.040793  1.068341  1.065653  1.067186   1.060128  …
..         …         …         …         …         …          …  …
247   1.147921  1.078512  1.051891  1.031736  1.000695   1.003188  …
248   3.482800  3.436074  3.391287  3.351226  3.308588   3.253021  …
250  12.851885  2.481517  6.407886  6.021886  6.087924  12.612111  …
253   2.083131  2.052951  2.054892  2.108305  2.046206   2.032273  …
256   1.506678  1.515090  1.515578  1.533225  1.563083   1.578205  …

         2011      2012      2013      2014      2015      2016      2017  \
5    2.329922  2.281329  2.224341  2.160102  2.093418  2.019087  1.949024
21   1.297394  1.108202  0.980915  0.939285  0.959409  0.990519  1.008312
29   1.288981  1.337513  1.352257  1.313717  1.246081  1.172401  1.106999
34  -0.236933 -0.229155 -0.213202 -0.209757 -0.230243 -0.255291 -0.248193
36   0.788201  0.655001  0.604135  0.649270  0.780009  0.925654  1.038187
..        …         …         …         …         …         …         …
247  0.056333  0.109803  0.172305  0.219787  0.263292  0.284529  0.335635
248  1.564456  1.627756  1.424032  0.874378  0.122058 -0.786448 -1.538843
250 -0.060928 -0.093310 -0.135963 -0.148198 -0.161415 -0.185856 -0.225349
253  1.170040  1.183831  1.183727  1.179816  1.168120  1.162578  1.142040
```

```
256  2.779988  2.757339  2.716520  2.654141  2.578072  2.498247  2.424025

         2018      2019  Population_1961
5    1.915913  1.924693              NaN
21   1.010953  0.991336              NaN
29   1.051994  1.002737              NaN
34  -0.196252 -0.154527              NaN
36   1.081493  1.026973              NaN
..        …         …              …
247  0.348124  0.343299              NaN
248 -1.785865 -1.235041              NaN
250 -0.271652 -0.323958              NaN
253  1.103609  1.074675              NaN
256  2.357023  2.300580              NaN

[75 rows x 62 columns]
```

So if we only use the countries that have data for their population size, we already lose 75 of them.

We can also see that we have a line : "World"!

So we will remove it from the data set.

[ ]:
```

```

[13]:
```python
World = PercGrow[PercGrow["Country Name"] == "World"].copy()
World["Population_1961"] = 3091843507
World
```

[13]:
```
    Country Name Country Code     1961      1962      1963      1964  \
257        World          WLD  1.35392  1.724198  2.083131  2.052951

         1965      1966      1967      1968  …     2011      2012      2013  \
257  2.054892  2.108305  2.046206  2.032273  …  1.17004  1.183831  1.183727

         2014     2015      2016     2017      2018      2019  Population_1961
257  1.179816  1.16812  1.162578  1.14204  1.103609  1.074675       3091843507

[1 rows x 62 columns]
```

In order to know the evolution of the population of each selected countries,

we multiply the population of the previous year with the growth percentage of the current year to calculate the population growth

[14]:
```python
PopulationSize = pd.merge(PercGrow,population_size_1961,on=str(PercGrow.
 ↪columns[0]),how='inner')

for col in PopulationSize.columns[3:-1]:
```

```
    PopulationSize[f"Population_{col}"] =␣
  ↪round(PopulationSize[f"Population_{int(col)-1}"] +␣
  ↪PopulationSize[f"Population_{int(col)-1}"] * (PopulationSize[col]/100))

PopulationSize.head(5)

#Pnext=Pbefore+Pbefore*Growth/100
```

[14]:

```
  Country Name Country Code      1961      1962      1963      1964      1965  \
0        Aruba          ABW  2.238144  1.409622  0.832453  0.592649  0.573468
1  Afghanistan          AFG  1.898476  1.965751  2.029893  2.090248  2.147567
2       Angola          AGO  1.393363  1.383629  1.256555  0.973962  0.617544
3      Albania          ALB  3.120855  3.056731  2.953749  2.880686  2.754021
4      Andorra          AND  6.941532  6.692697  6.559522  6.241511  5.998800

       1966      1967      1968  …  Population_2010  Population_2011  \
0  0.616991  0.587373  0.568530  …         103502.0         103885.0
1  2.171009  2.188108  2.254572  …       31060834.0       32037115.0
2  0.184283 -0.120653 -0.044882  …       19816151.0       20536301.0
3  2.634564  2.630190  2.842511  …        2917711.0        2909862.0
4  5.750878  5.500706  5.309820  …          50676.0          50253.0

   Population_2012  Population_2013  Population_2014  Population_2015  \
0         104407.0         105015.0         105639.0         106215.0
1       33128808.0       34286524.0       35437036.0       36527463.0
2       21275151.0       22030834.0       22801361.0       23585466.0
3        2905056.0        2899734.0        2893730.0        2885303.0
4          49455.0          48453.0          47507.0          46781.0

   Population_2016  Population_2017  Population_2018  Population_2019
0         106754.0         107256.0         107742.0         108218.0
1       37542312.0       38499020.0       39417130.0       40328382.0
2       24382247.0       25192245.0       26017577.0       26861322.0
3        2880690.0        2878041.0        2870940.0        2858710.0
4          46351.0          46173.0          46176.0          46257.0

[5 rows x 120 columns]
```

Finally, we just add the regions to our dataset and we are ready to start the analysis.

[15]:

```
df=pd.merge(PopulationSize,IncomeGroup[IncomeGroup.columns[0:3]],on='Country␣
  ↪Code',how='inner')
df.head()
```

[15]:

```
  Country Name Country Code      1961      1962      1963      1964      1965  \
0        Aruba          ABW  2.238144  1.409622  0.832453  0.592649  0.573468
1  Afghanistan          AFG  1.898476  1.965751  2.029893  2.090248  2.147567
```

```
2       Angola          AGO  1.393363  1.383629  1.256555  0.973962  0.617544
3      Albania          ALB  3.120855  3.056731  2.953749  2.880686  2.754021
4      Andorra          AND  6.941532  6.692697  6.559522  6.241511  5.998800

       1966      1967      1968  …  Population_2012  Population_2013  \
0  0.616991  0.587373  0.568530  …         104407.0         105015.0
1  2.171009  2.188108  2.254572  …       33128808.0       34286524.0
2  0.184283 -0.120653 -0.044882  …       21275151.0       22030834.0
3  2.634564  2.630190  2.842511  …        2905056.0        2899734.0
4  5.750878  5.500706  5.309820  …          49455.0          48453.0

   Population_2014  Population_2015  Population_2016  Population_2017  \
0         105639.0         106215.0         106754.0         107256.0
1       35437036.0       36527463.0       37542312.0       38499020.0
2       22801361.0       23585466.0       24382247.0       25192245.0
3        2893730.0        2885303.0        2880690.0        2878041.0
4          47507.0          46781.0          46351.0          46173.0

   Population_2018  Population_2019                     Region  \
0         107742.0         108218.0  Latin America & Caribbean
1       39417130.0       40328382.0                 South Asia
2       26017577.0       26861322.0         Sub-Saharan Africa
3        2870940.0        2858710.0      Europe & Central Asia
4          46176.0          46257.0      Europe & Central Asia

           IncomeGroup
0          High income
1           Low income
2  Lower middle income
3  Upper middle income
4          High income

[5 rows x 122 columns]
```

GROWTH EVOLUTION AROUND THE WORLD

Now that our dataset is ready, we can start by observing the evolution of the last 60 years at the level of the regions of the world but also in its totality.

We will compare both the Population in quantity and the evolution in percentage.

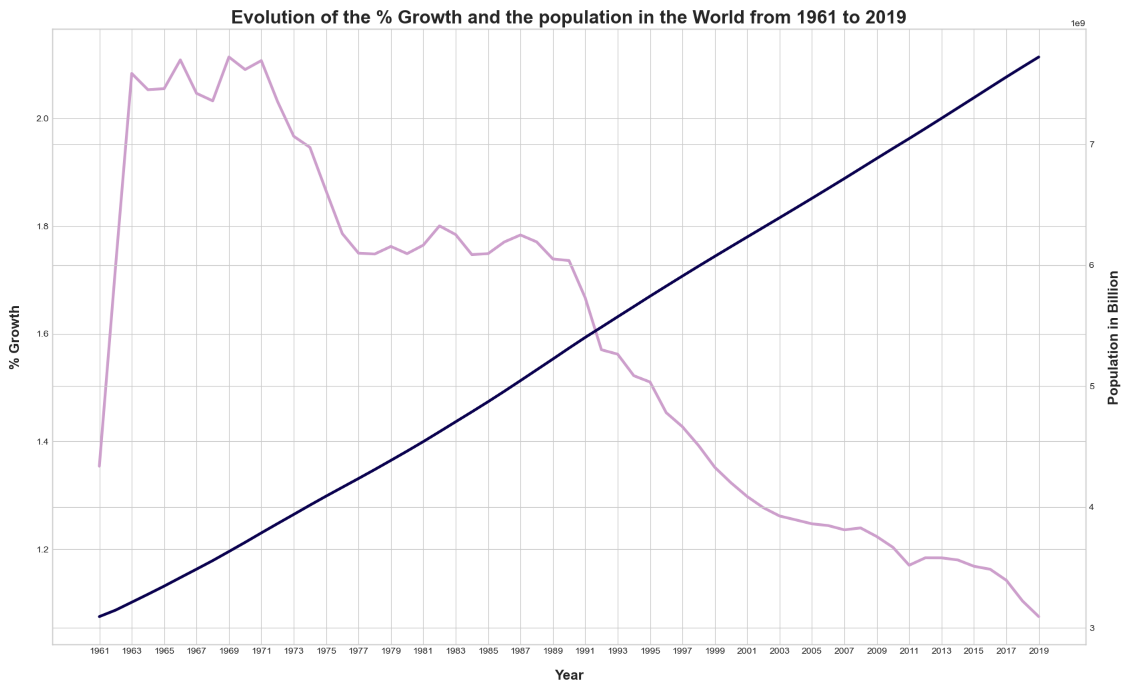
[16]:
```
import matplotlib.pyplot as plt
import seaborn as sns
columms = World.columns[3:-1].copy()

for col in columms:
    World[f"Population_{col}"] = round(World[f"Population_{int(col)-1}"] +
    ↪World[f"Population_{int(col)-1}"] * (World[col]/100))
```

```
World[World.columns[2:61]].values[0]
```

[17]:
```
#Set a figure
plt.style.use('seaborn-whitegrid')
fig, ax_G = plt.subplots(figsize=[20,12])
ax_P = ax_G.twinx()
ax_G.set_xlabel("Year",fontsize=15,fontweight=550,labelpad=15)
ax_G.set_ylabel("% Growth",fontsize=15,fontweight=550,labelpad=15)
ax_P.set_ylabel("Population in Billion",fontsize=15,fontweight=550,labelpad=15)
ax_G.set_title("Evolution of the % Growth and the population in the World from␣
↪1961 to 2019",fontsize=20,fontweight=600)
ax_P.set_xticks([i*2 for i in range(0,30)])
ax_P.set_xticklabels(labels  = [1961 + i*2 for i in range(0,30)])

ax_G.plot([int(item) for item in World.columns[2:61].tolist()],World[World.
↪columns[2:61]].values[0],linewidth=3,label="% Growth",color = "#CD9FCC")
ax_P.plot([int(item[-4:]) for item in World.columns[61:].tolist()],World[World.
↪columns[61:]].values[0],linewidth=3,label="Population",color = "#0A014F")

ax_G.set_xticks([1961 + i*2 for i in range(0,30)])
ax_G.set_xticklabels([1961 + i*2 for i in range(0,30)])
```

[17]:
```
[Text(1961, 0, '1961'),
 Text(1963, 0, '1963'),
 Text(1965, 0, '1965'),
 Text(1967, 0, '1967'),
 Text(1969, 0, '1969'),
 Text(1971, 0, '1971'),
 Text(1973, 0, '1973'),
 Text(1975, 0, '1975'),
 Text(1977, 0, '1977'),
 Text(1979, 0, '1979'),
 Text(1981, 0, '1981'),
 Text(1983, 0, '1983'),
 Text(1985, 0, '1985'),
 Text(1987, 0, '1987'),
 Text(1989, 0, '1989'),
 Text(1991, 0, '1991'),
 Text(1993, 0, '1993'),
 Text(1995, 0, '1995'),
 Text(1997, 0, '1997'),
 Text(1999, 0, '1999'),
 Text(2001, 0, '2001'),
 Text(2003, 0, '2003'),
 Text(2005, 0, '2005'),
 Text(2007, 0, '2007'),
```

```
 Text(2009, 0, '2009'),
 Text(2011, 0, '2011'),
 Text(2013, 0, '2013'),
 Text(2015, 0, '2015'),
 Text(2017, 0, '2017'),
 Text(2019, 0, '2019')]
```



Despite a declining percentage of growth, the number of people on Earth is increasing. as of 2019 data world population is : 7.674 Billion

Let's see now according to the regions I start by grouping the countries into regions using the "groupby" function and calculating their average.

[18]:
```
Evolution_by_region = df.groupby(["Region"]).mean()
Percentage_years_columns = Evolution_by_region.columns[0:59] #% growth columns
Population_years_columns = Evolution_by_region.columns[59:] #population size␣
 ↪columns
Evolution_by_region
```

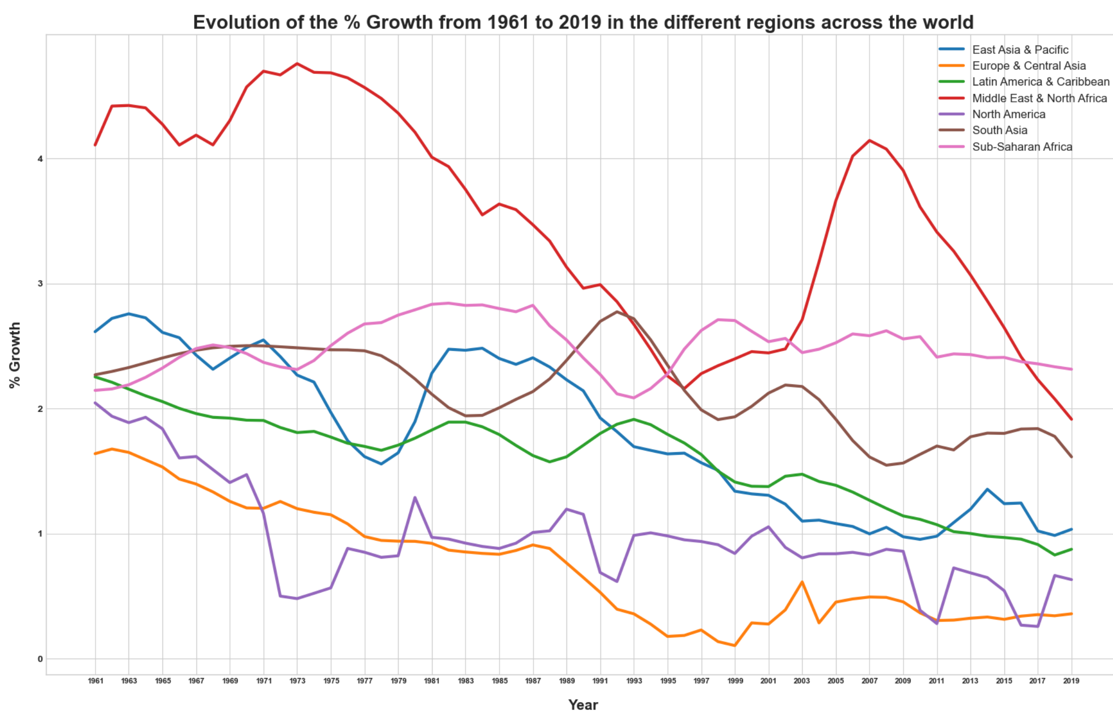
[18]:
```
                                1961      1962      1963      1964      1965  \
Region
East Asia & Pacific         2.615088  2.721297  2.758067  2.725930  2.609322
Europe & Central Asia       1.639127  1.676346  1.649338  1.590681  1.532719
Latin America & Caribbean   2.253311  2.210517  2.155360  2.102156  2.055885
Middle East & North Africa  4.108606  4.419977  4.424852  4.405541  4.274520
North America               2.045443  1.938489  1.887509  1.930161  1.838109
```

```
South Asia                  2.270356  2.296972  2.328190  2.365515  2.404615
Sub-Saharan Africa          2.145534  2.156893  2.190256  2.249910  2.324253

                                1966      1967      1968      1969      1970  \
Region
East Asia & Pacific         2.566872  2.427968  2.314403  2.403668  2.488723
Europe & Central Asia       1.436181  1.396342  1.333351  1.257902  1.205030
Latin America & Caribbean   2.001875  1.959895  1.930298  1.923506  1.907114
Middle East & North Africa  4.107937  4.187468  4.109380  4.303763  4.572730
North America               1.604705  1.616341  1.511649  1.407940  1.471562
South Asia                  2.439058  2.466724  2.486182  2.497952  2.503123
Sub-Saharan Africa          2.409822  2.480043  2.508875  2.488287  2.439713

                            …  Population_2010  Population_2011  \
Region                      …
East Asia & Pacific         …     7.018177e+07     7.065903e+07
Europe & Central Asia       …     1.404618e+07     1.408644e+07
Latin America & Caribbean   …     1.617019e+07     1.634663e+07
Middle East & North Africa  …     1.111190e+07     1.140679e+07
North America               …     1.139587e+08     1.148086e+08
South Asia                  …     2.068002e+08     2.096744e+08
Sub-Saharan Africa          …     1.797090e+07     1.845100e+07

                            Population_2012  Population_2013  Population_2014  \
Region
East Asia & Pacific            7.114908e+07     7.164415e+07     7.214366e+07
Europe & Central Asia          1.415094e+07     1.422666e+07     1.430346e+07
Latin America & Caribbean      1.652094e+07     1.669514e+07     1.687101e+07
Middle East & North Africa     1.171439e+07     1.202609e+07     1.233096e+07
North America                  1.156852e+08     1.165231e+08     1.174037e+08
South Asia                     2.124860e+08     2.152770e+08     2.180504e+08
Sub-Saharan Africa             1.894285e+07     1.944726e+07     1.996337e+07

                            Population_2015  Population_2016  Population_2017  \
Region
East Asia & Pacific            7.264220e+07     7.315343e+07     7.366609e+07
Europe & Central Asia          1.437959e+07     1.445421e+07     1.452034e+07
Latin America & Caribbean      1.705025e+07     1.723391e+07     1.742025e+07
Middle East & North Africa     1.262077e+07     1.289154e+07     1.314466e+07
North America                  1.182632e+08     1.191606e+08     1.199810e+08
South Asia                     2.208133e+08     2.235737e+08     2.263248e+08
Sub-Saharan Africa             2.049049e+07     2.102833e+07     2.157673e+07

                            Population_2018  Population_2019
Region
East Asia & Pacific            7.412363e+07     7.453389e+07
Europe & Central Asia          1.458288e+07     1.464443e+07
```

```
Latin America & Caribbean       1.760395e+07     1.778461e+07
Middle East & North Africa      1.338262e+07     1.361078e+07
North America                   1.207159e+08     1.214071e+08
South Asia                      2.290622e+08     2.317798e+08
Sub-Saharan Africa              2.213505e+07     2.270282e+07

[7 rows x 118 columns]
```

[19]:
```python
#Set a figure
fig = plt.figure(figsize=[20,12])
plt.yticks(fontsize=10,fontweight=550)
plt.xticks([i*2 for i in range(0,30)],fontsize=8,fontweight=550)
plt.xlabel("Year",fontsize=15,fontweight=550,labelpad=15)
plt.ylabel("% Growth",fontsize=15,fontweight=550,labelpad=15)
plt.title("Evolution of the % Growth from 1961 to 2019 in the different regions␣
 ↪across the world",fontsize=20,fontweight=600)

for index in Evolution_by_region.index:
    plt.plot(Evolution_by_region[Percentage_years_columns].loc[index],label =␣
 ↪index,linewidth=3)

plt.legend(fontsize=12)
```

[19]: <matplotlib.legend.Legend at 0x1ffdce79f70>



[20]:
```python
print("We can see that overall, the % of growth has decreased over the last 60␣
↪years !\n")
print(f"Decrease for each region from 1961 to 2019 :␣
↪\n\n\n\n{Evolution_by_region['1961'] - Evolution_by_region['2019']}")
Evolution_by_region[["1961","2019"]]
```

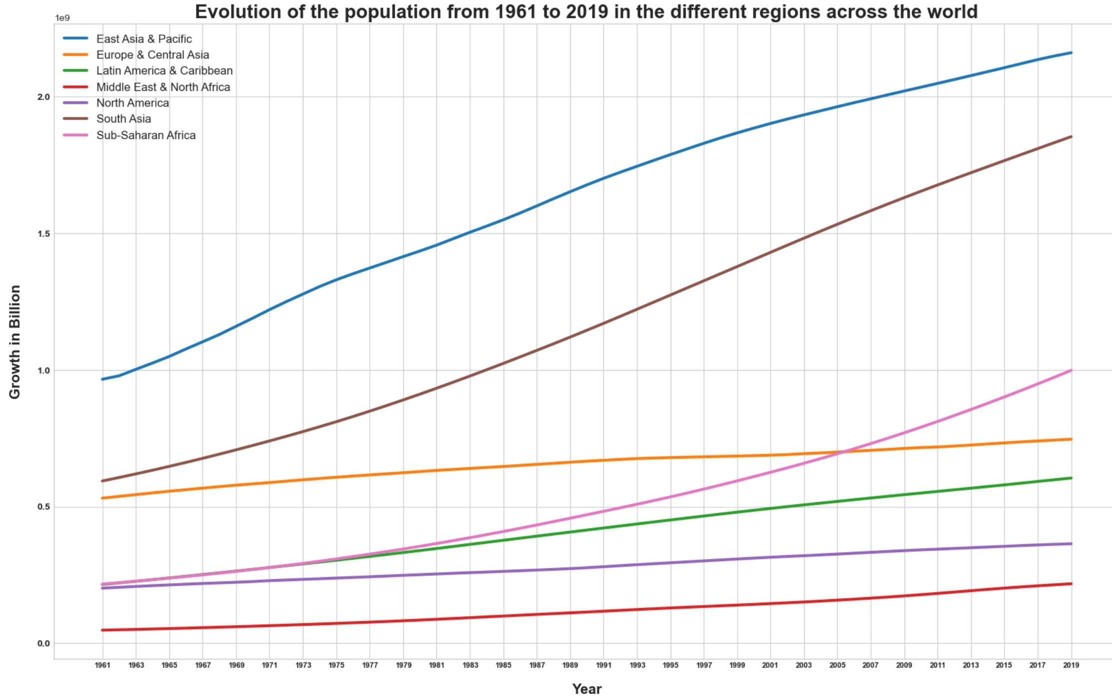
```
We can see that overall, the % of growth has decreased over the last 60 years !

Decrease for each region from 1961 to 2019 :



Region
East Asia & Pacific            1.580760
Europe & Central Asia          1.280836
Latin America & Caribbean      1.378774
Middle East & North Africa     2.194281
North America                  1.413818
South Asia                     0.656689
Sub-Saharan Africa            -0.169169
dtype: float64
```

[20]:

Region	1961	2019
East Asia & Pacific	2.615088	1.034327
Europe & Central Asia	1.639127	0.358291
Latin America & Caribbean	2.253311	0.874537
Middle East & North Africa	4.108606	1.914325
North America	2.045443	0.631625
South Asia	2.270356	1.613668
Sub-Saharan Africa	2.145534	2.314702

[21]:
```python
#Set a figure
fig = plt.figure(figsize=[20,12])
plt.yticks(fontsize=10,fontweight=550)
plt.xticks([i*2 for i in range(0,30)],labels=[1961+i*2 for i in␣
↪range(0,30)],fontsize=8,fontweight=550)
plt.xlabel("Year",fontsize=15,fontweight=550,labelpad=15)
plt.ylabel("Growth in Billion",fontsize=15,fontweight=550,labelpad=15)
plt.title("Evolution of the population from 1961 to 2019 in the different␣
↪regions across the world",fontsize=20,fontweight=600)

for index in Evolution_by_region.index:
    plt.plot(df.groupby(['Region']).sum()[Population_years_columns].
↪loc[index],label = index,linewidth=3)

plt.legend(fontsize=12)
```

[21]: <matplotlib.legend.Legend at 0x1ffdd58bb20>



[22]:
```python
print("On the contrary, despite a decrease of % of growth, the world population has increased for each region ! \n")
print(f"Population Evolution from 1961 to 2019 : \n\n\n{df.groupby(['Region']).sum()['Population_2019'] - df.groupby(['Region']).sum()['Population_1961']}")
df.groupby(["Region"]).sum()[["Population_1961","Population_2019"]]
```

```
On the contrary, despite a decrease of % of growth, the world population has
increased for each region !

Population Evolution from 1961 to 2019 :


Region
East Asia & Pacific           1.195135e+09
Europe & Central Asia         2.155857e+08
Latin America & Caribbean     3.896233e+08
Middle East & North Africa    1.698075e+08
North America                 1.625491e+08
South Asia                    1.260283e+09
Sub-Saharan Africa            7.822757e+08
dtype: float64
```

[22]:
```
                              Population_1961  Population_2019
Region
East Asia & Pacific                 966347771     2.161483e+09
Europe & Central Asia               531280112     7.468658e+08
Latin America & Caribbean           215053386     6.046767e+08
Middle East & North Africa           47965036     2.177725e+08
North America                       201672125     3.642212e+08
South Asia                          593955746     1.854239e+09
Sub-Saharan Africa                  216648225     9.989239e+08
```

[23]:
```
result = sum(df.groupby(["Region"]).sum()["Population_2019"].iloc[[0,5]])
print(f"If we add 'East Asia & Pacific' and 'South Asia', we can observe that a␣
↪lot of people lives in these region :\n{result}")
print("Soit 4 billions peoples, more than half of the total earth population")
```

```
If we add 'East Asia & Pacific' and 'South Asia', we can observe that a lot of
people lives in these region :
4015721464.0
Soit 4 billions peoples, more than half of the total earth population
```

[24]:
```
EastAsia = df[df["Region"] == 'East Asia & Pacific'].copy()
SouthAsia = df[df["Region"] == 'South Asia'].copy()
print(SouthAsia[SouthAsia["Population_2019"] == SouthAsia["Population_2019"].
↪max()]["Country Name"])
print(EastAsia[EastAsia["Population_2019"] == EastAsia["Population_2019"].
↪max()]["Country Name"])
```

```
79    India
Name: Country Name, dtype: object
33    China
Name: Country Name, dtype: object
```

[ ]:
```

```

COMPARISON OF WORLD'S TWO POPULOUS COUNRTRIES

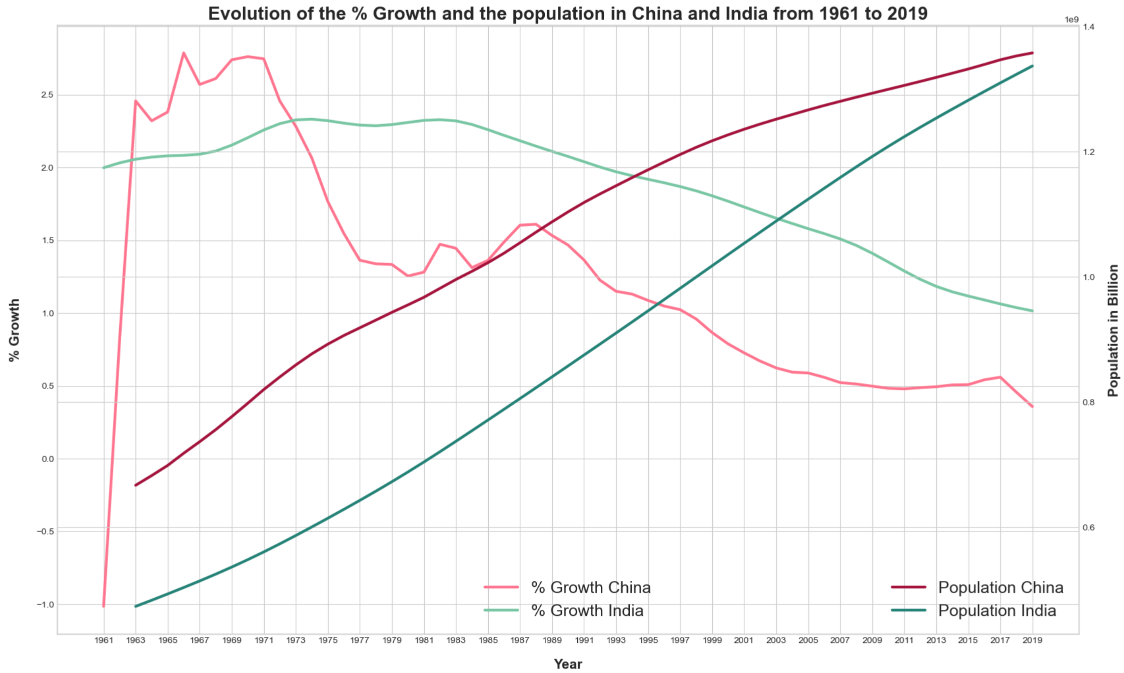
INDIA VS CHINA

For each of these regions, one country has more than one billion inhabitants at present. China and India, often referred to as the massive population places.

[25]:
```
#Create datasets
China = df[df["Country Name"] == "China"]
India = df[df["Country Name"] == "India"]
```

[26]:
```
#Set a figure
fig, ax_G = plt.subplots(figsize=[20,12])
ax_P = ax_G.twinx()
fig = plt.figure()
ax_G.set_xlabel("Year",fontsize=15,fontweight=550,labelpad=15)
ax_G.set_ylabel("% Growth",fontsize=15,fontweight=550,labelpad=15)
ax_P.set_ylabel("Population in Billion",fontsize=15,fontweight=550,labelpad=15)
ax_G.set_title("Evolution of the % Growth and the population in China and India␣
↪from 1961 to 2019",fontsize=20,fontweight=600)
ax_P.set_xticks([i*2 for i in range(0,30)])
ax_P.set_xticklabels(labels  = [1961 + i*2 for i in range(0,30)])

ax_G.plot([int(item) for item in China.columns[2:61].tolist()],China[China.
↪columns[2:61]].values[0],linewidth=3,label="% Growth China",color =␣
↪"#ff758f")
ax_P.plot([int(item[-4:]) for item in China.columns[63:-2].
↪tolist()],China[China.columns[63:-2]].
↪values[0],linewidth=3,label="Population China",color = "#a4133c")

ax_G.plot([int(item) for item in India.columns[2:61].tolist()],India[India.
↪columns[2:61]].values[0],linewidth=3,label="% Growth India",color =␣
↪"#78c6a3")
ax_P.plot([int(item[-4:]) for item in India.columns[63:-2].
↪tolist()],India[India.columns[63:-2]].
↪values[0],linewidth=3,label="Population India",color = "#248277")



ax_G.set_xticks([1961 + i*2 for i in range(0,30)])
ax_G.set_xticklabels([1961 + i*2 for i in range(0,30)])
ax_G.legend(fontsize=18,loc=8)
ax_P.legend(fontsize=18,loc=4)
```

[26]:
```
<matplotlib.legend.Legend at 0x1ffdd62fee0>
```



<Figure size 640x480 with 0 Axes>

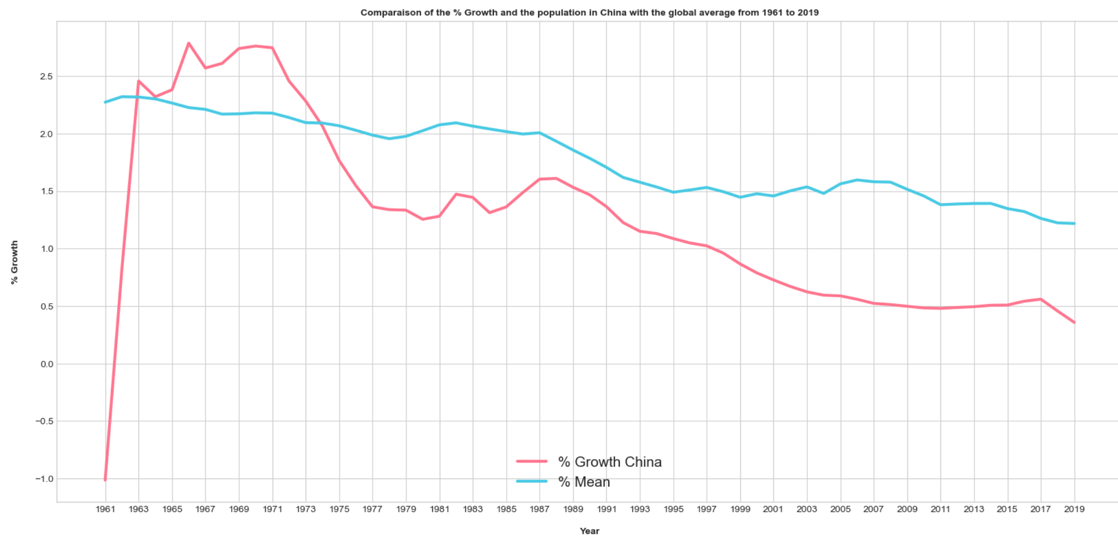
COMPARISON WITH WORLD'S AVERAGE

[27]:
```
Growth_AVG = df[df.columns[4:63]].mean()
Pop_AVG = df[df.columns[63:]].mean()
```

[28]:
```
plt.figure(figsize=[20,20])
ax_G = plt.subplot(211)

fig = plt.figure()
ax_G.set_xlabel("Year",fontsize=10,fontweight=550,labelpad=15)
ax_G.set_ylabel("% Growth",fontsize=10,fontweight=550,labelpad=15)
ax_G.set_title("Comparaison of the % Growth and the population in China with␣
↪the global average from 1961 to 2019",fontsize=10,fontweight=600)
ax_G.set_xticks([i*2 for i in range(0,30)])
ax_G.set_xticklabels(labels  = [1961 + i*2 for i in range(0,30)])

ax_G.plot([int(item) for item in China.columns[2:61].tolist()],China[China.
↪columns[2:61]].values[0],linewidth=3,label="% Growth China",color =␣
↪"#ff758f")

ax_G.plot([int(item) for item in China.columns[2:61].tolist()],df[df.columns[2:
↪61]].mean(),linewidth=3,label="% Mean",color = "#48cae4")
```

```
ax_G.set_xticks([1961 + i*2 for i in range(0,30)])
ax_G.set_xticklabels([1961 + i*2 for i in range(0,30)])
ax_G.legend(fontsize=15,loc=8)
```

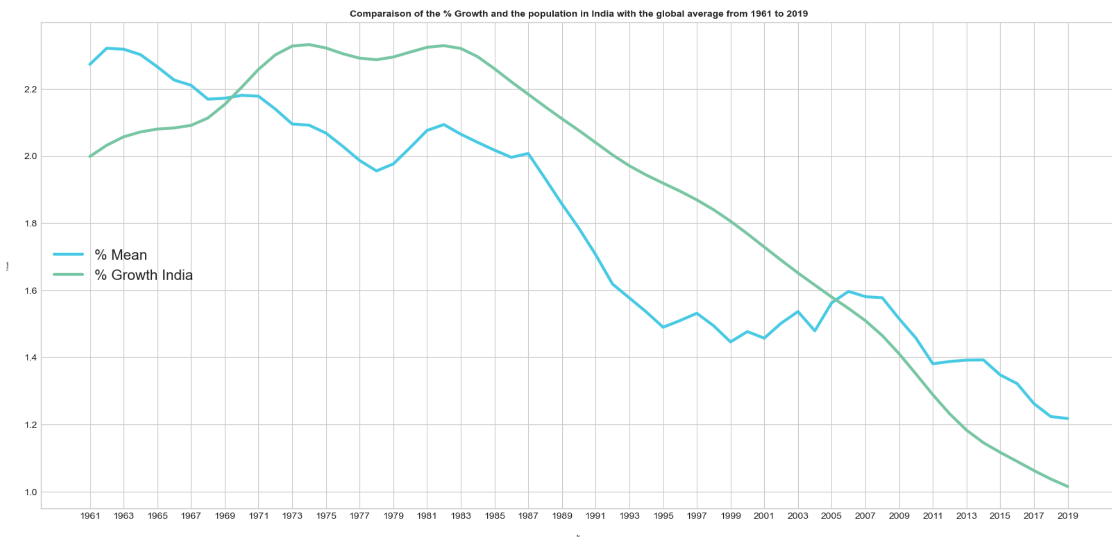
[28]: <matplotlib.legend.Legend at 0x1ffde5bbee0>



<Figure size 640x480 with 0 Axes>

[29]:
```
#Set a figure
plt.figure(figsize=[20,20])
ax_G2 = plt.subplot(212)

ax_G2.set_xlabel("Year",fontsize=2,fontweight=550,labelpad=15)
ax_G2.set_ylabel("% Growth",fontsize=2,fontweight=550,labelpad=15)
ax_G2.set_title("Comparaison of the % Growth and the population in India with␣
↪the global average from 1961 to 2019",fontsize=10,fontweight=600)
ax_G2.set_xticks([i*2 for i in range(0,30)])
ax_G2.set_xticklabels(labels  = [1961 + i*2 for i in range(0,30)])

ax_G2.plot([int(item) for item in China.columns[2:61].tolist()],df[df.columns[2:
↪61]].mean(),linewidth=3,label="% Mean",color = "#48cae4")
ax_G2.plot([int(item) for item in India.columns[2:61].tolist()],India[India.
↪columns[2:61]].values[0],linewidth=3,label="% Growth India",color =␣
↪"#78c6a3")


ax_G2.set_xticks([1961 + i*2 for i in range(0,30)])
ax_G2.set_xticklabels([1961 + i*2 for i in range(0,30)])
ax_G2.legend(fontsize=15,loc=6)
```

[29]: <matplotlib.legend.Legend at 0x1ffdeb65910>



The graph above shows that China's % growth fell below the world average in 1975. In contrast, India's growth rate rose above the world average in 1969 and remained above it for more than 35 years (until 2005).

This highlights the significant population growth that India has undergone in recent years, and which risks becoming the most populous country in the world. An important factor is the one-child policy, or Planned Parenthood policy, the public birth control policy implemented by the People's Republic of China from 1979 to 2015.

This reinforces the fall in the % growth over this period.

[ ]:

```
```

WORLD'S GROWTH & POPULATION PREDICTION

First Model - Autoregression (AR)

Second Model- Holt's Linear Smoothing

[30]:
```
PercentGrowth = pd.melt(World[World.columns[2:61]]).rename(columns={"variable" :
 ↪ "Year","value" : "PercGrowth"})
PercentGrowth["Year"] = PercentGrowth["Year"].map(lambda x : pd.
 ↪to_datetime(f"{x}-12-31"))
PercentGrowth.head(5)
```

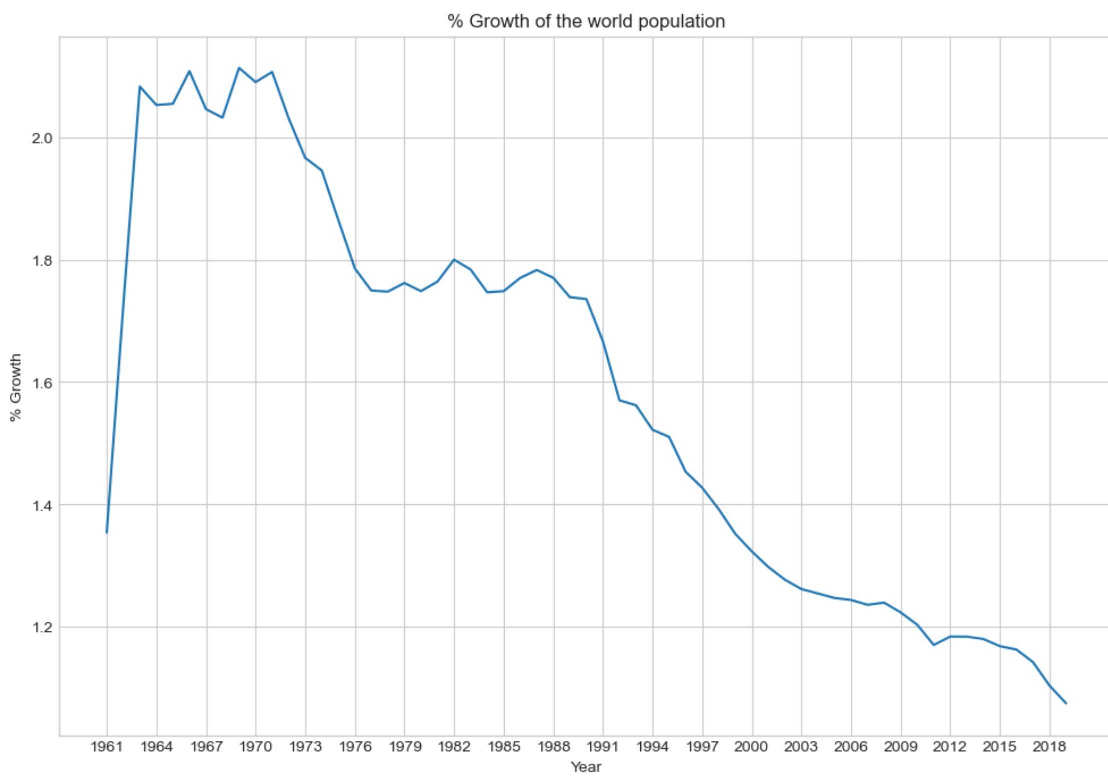
[30]:

	Year	PercGrowth
0	1961-12-31	1.353920
1	1962-12-31	1.724198
2	1963-12-31	2.083131
3	1964-12-31	2.052951

```
4 1965-12-31    2.054892
```

[31]:
```
plt.figure(figsize=[12,8])
plt.title("% Growth of the world population")
plt.xlabel("Year")
plt.ylabel("% Growth")
plt.xticks([i*3 for i in range(0,20)],labels = [1961 + i*3 for i in range(20)])
PercentGrowth['PercGrowth'].plot()
```

[31]:
```
<AxesSubplot:title={'center':'% Growth of the world population'}, xlabel='Year',
ylabel='% Growth'>
```



AUTOREGRESSION MODEL

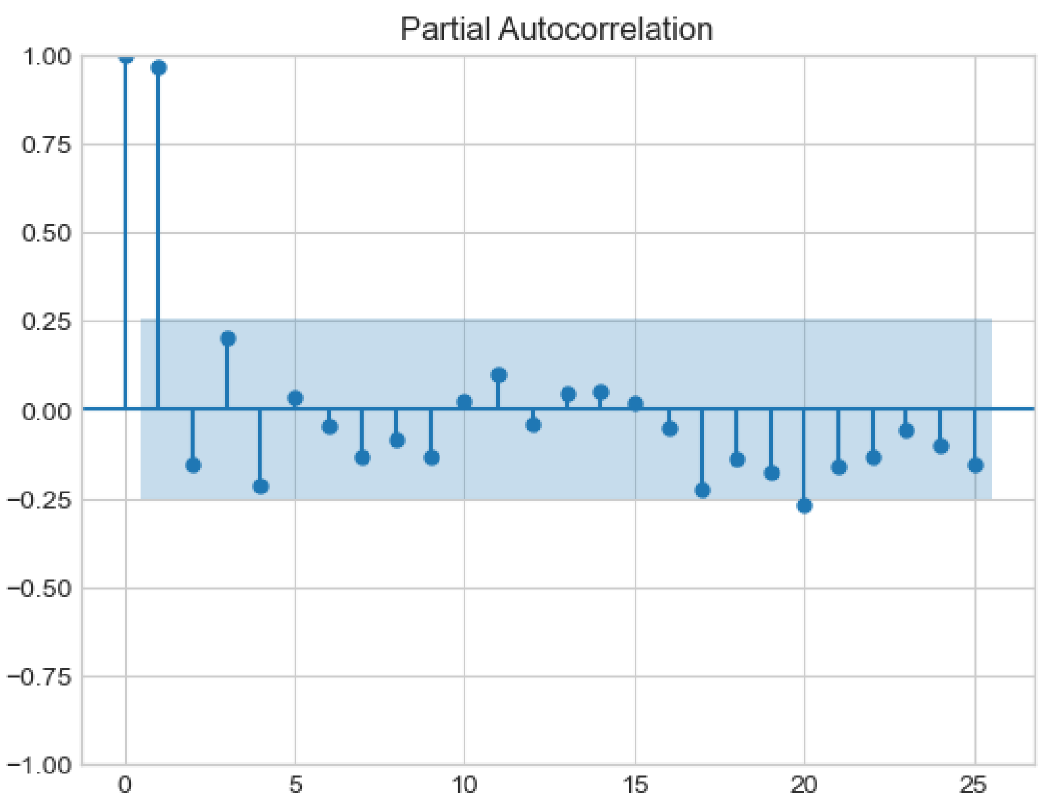
This model uses past values to predict the future values,this going to help us in this kind of problem.

[32]:
```
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.ar_model import AutoReg

# Run the test
PG_stationarityTest = adfuller(PercentGrowth['PercGrowth'], autolag='AIC')
```

```python
# Check the value of p-value
print("P-value: ", PG_stationarityTest[1],", then no stationarity.")

plt.figure(figsize=[12,8])
pacf = plot_pacf(PercentGrowth['PercGrowth'], lags=25)

# Create training and test data
train_data = PercentGrowth['PercGrowth'][:
 ↪round(len(PercentGrowth['PercGrowth'])*0.8)]
test_data =␣
 ↪PercentGrowth['PercGrowth'][round(len(PercentGrowth['PercGrowth'])*0.8):]
```

```
P-value:  0.9437518182894581 , then no stationarity.
```

```
<Figure size 1200x800 with 0 Axes>
```



[33]:
```python
# Instantiate and fit the AR model with training data
ar_model = AutoReg(train_data, lags=6).fit()
print(ar_model.summary())
plt.show()
```

```
                            AutoReg Model Results
```

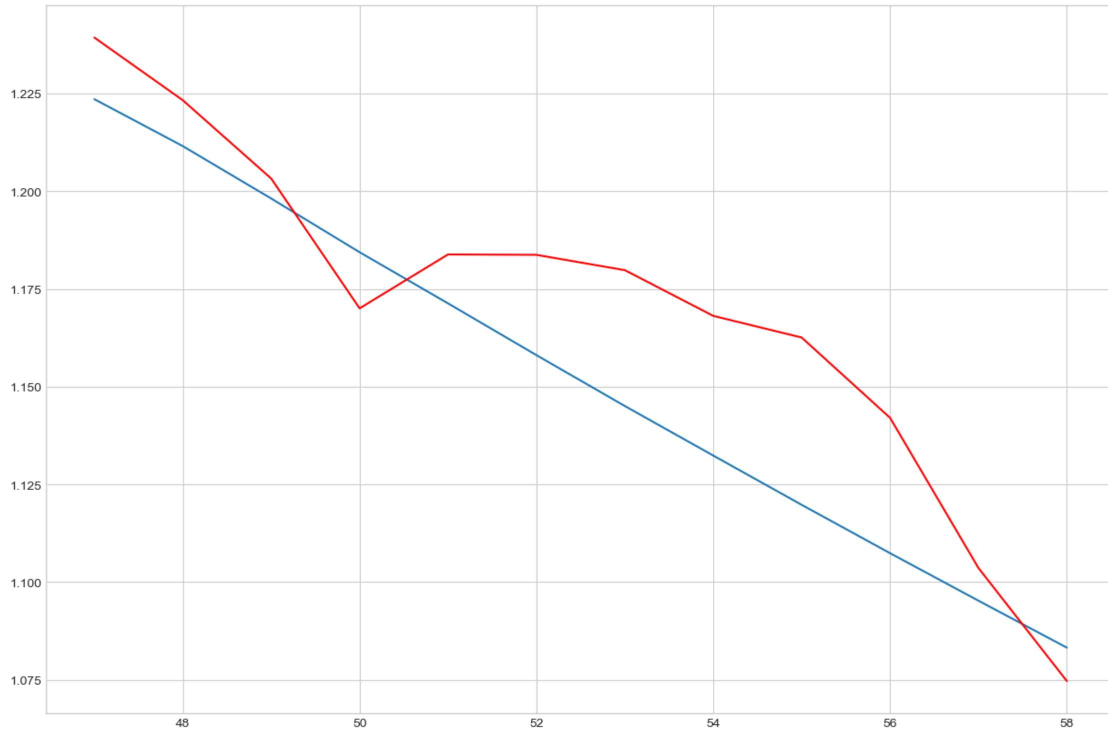
```
==============================================================================
Dep. Variable:              PercGrowth   No. Observations:                   47
Model:                     AutoReg(6)   Log Likelihood                  83.798
Method:               Conditional MLE   S.D. of innovations              0.031
Date:                Wed, 01 Mar 2023   AIC                           -151.596
Time:                        23:04:38   BIC                           -137.887
Sample:                             6   HQIC                          -146.604
                                   47
===============================================================================
=
                    coef    std err          z      P>|z|      [0.025
0.975]
-------------------------------------------------------------------------------
-
const             0.0029      0.037      0.079      0.937      -0.070
0.076
PercGrowth.L1     1.2875      0.155      8.303      0.000       0.984
1.591
PercGrowth.L2    -0.3271      0.246     -1.328      0.184      -0.810
0.155
PercGrowth.L3     0.2506      0.229      1.094      0.274      -0.198
0.700
PercGrowth.L4    -0.3831      0.178     -2.153      0.031      -0.732
-0.034
PercGrowth.L5     0.1959      0.156      1.258      0.208      -0.109
0.501
PercGrowth.L6    -0.0335      0.076     -0.441      0.660      -0.183
0.116
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.8583          -1.1679j            1.4494           -0.3509
AR.2           -0.8583          +1.1679j            1.4494            0.3509
AR.3            1.0154          -0.0000j            1.0154           -0.0000
AR.4            1.9891          -1.2210j            2.3340           -0.0876
AR.5            1.9891          +1.2210j            2.3340            0.0876
AR.6            2.5670          -0.0000j            2.5670           -0.0000
-----------------------------------------------------------------------------
```

[34]:

```
# Make the predictions

plt.figure(figsize=[15,10])
pred = ar_model.predict(start=len(train_data), end=(len(PercentGrowth)-1),
 ↪dynamic=False)

# Plot the prediction vs test data
```

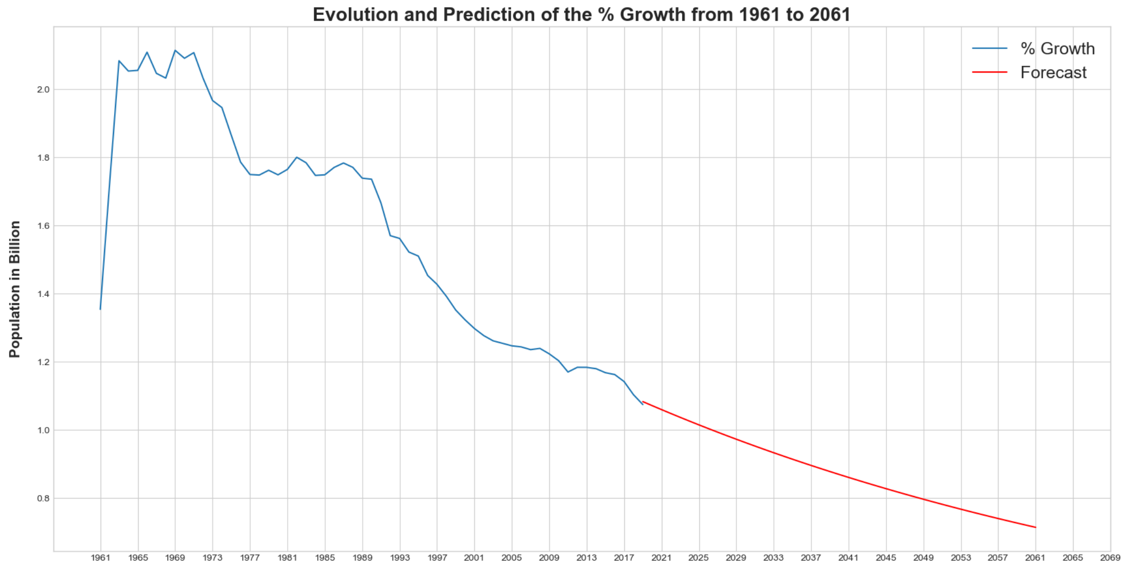
```
plt.plot(pred)
plt.plot(test_data, color='red')
```

[34]: [<matplotlib.lines.Line2D at 0x1ffe0856eb0>]



[35]:
```
# Make the predictions
plt.figure(figsize=[20,10])
plt.plot(PercentGrowth['PercGrowth'],label="% Growth")
pred = ar_model.predict(start=len(PercentGrowth)-1,␣
 ↪end=(len(PercentGrowth)+41), dynamic=False)
plt.plot(pred,color='red',label="Forecast")
plt.xticks([i*4 for i in range(28)],labels = [1961 + i*4 for i in range(28)])
plt.ylabel("Population in Billion",fontsize=15,fontweight=550,labelpad=15)
plt.title("Evolution and Prediction of the % Growth from 1961 to␣
 ↪2061",fontsize=20,fontweight=600)

plt.legend(fontsize=18)
```

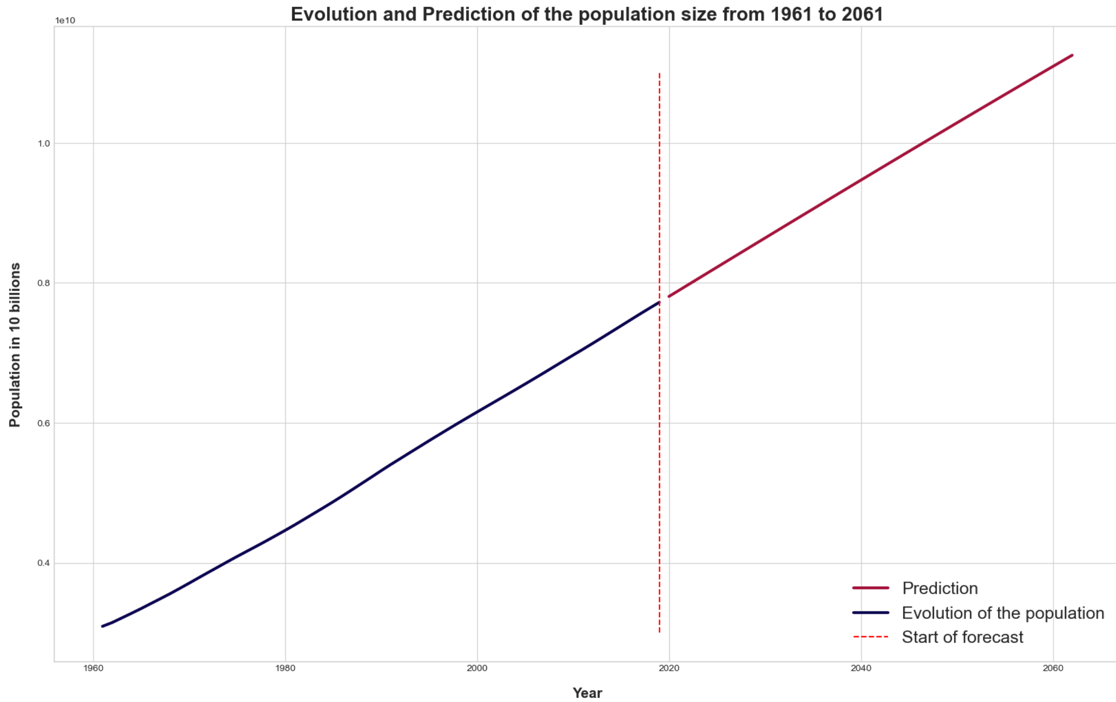
[35]: <matplotlib.legend.Legend at 0x1ffd883b280>



[36]:
```
PopIni = World["Population_2019"].values[0]
Pred_pop_size = []

for percGrowth in pred:
    PopIni = PopIni + PopIni * (percGrowth/100)
    Pred_pop_size.append(PopIni)
```

[37]:
```
#Set a figure
fig = plt.figure(figsize=[20,12])
plt.xlabel("Year",fontsize=15,fontweight=550,labelpad=15)
plt.ylabel("Population in 10 billions",fontsize=15,fontweight=550,labelpad=15)
plt.title("Evolution and Prediction of the population size from 1961 to␣
 ↪2061",fontsize=20,fontweight=600)
#ax_P.set_xticks([i*2 for i in range(0,30)])
#ax_P.set_xticklabels(labels  = [1961 + i*2 for i in range(0,30)])

plt.plot([2020 + i for i in␣
 ↪range(43)],Pred_pop_size,linewidth=3,label="Prediction",color = "#a4133c")
plt.plot([int(item[-4:]) for item in World.columns[61:].tolist()],World[World.
 ↪columns[61:]].values[0],linewidth=3,label="Evolution of the␣
 ↪population",color = "#0A014F")
plt.vlines(2019, 3000000000, 11000000000, linestyle='--', color='r',␣
 ↪label='Start of forecast');
plt.legend(fontsize=18,loc=4)
```

[37]: <matplotlib.legend.Legend at 0x1ffe1eba3d0>



A beautiful line is then predicted with more than 10 billion people on Earth by 2050.

EXPONENTIAL SMOOTHING

In order to compare our result with an another method, we are going to use Exponential smoothing for prediction

For our Exponential smoothing, we will use Holt's linear smoothing because our data have a downward trend and no seasonality.

[38]:
```
PercentGrowth = PercentGrowth.set_index("Year")
PercentGrowth.head(5)
```

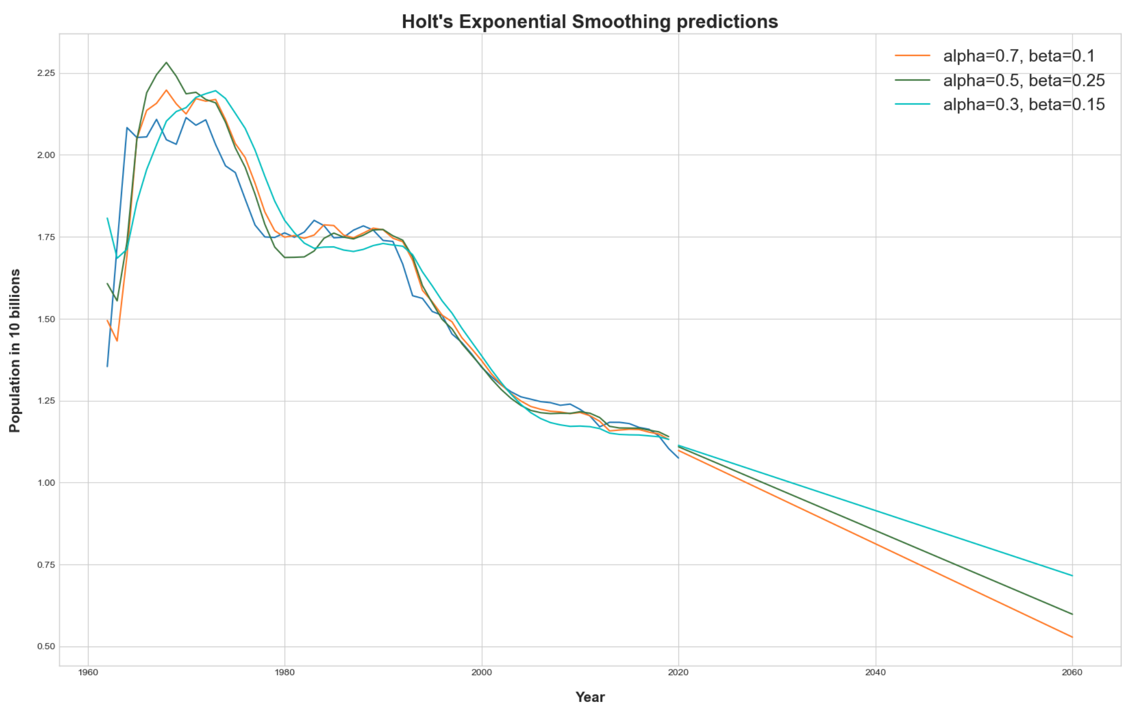
[38]:
```
            PercGrowth
Year
1961-12-31    1.353920
1962-12-31    1.724198
1963-12-31    2.083131
1964-12-31    2.052951
1965-12-31    2.054892
```

[39]:
```
from statsmodels.tsa.holtwinters import Holt
df = PercentGrowth.copy()
train = df.iloc[:58, :]
test = df.iloc[58:, :]
train.index = pd.to_datetime(train.index,format="%Y")
```

```
test.index = pd.to_datetime(test.index,format="%Y")
pred = test.copy()
```

[40]:
```
model = Holt(np.asarray(train['PercGrowth']))
#model._index = pd.to_datetime(df.index)
fit1 = model.fit(smoothing_level=.7, smoothing_trend=.1)
pred1 = fit1.forecast(len(test)+40)
fit2 = model.fit(smoothing_level=.5, smoothing_trend=.25)
pred2 = fit2.forecast(len(test)+40)
fit3 = model.fit(smoothing_level=.3, smoothing_trend=.15)
pred3 = fit3.forecast(len(test)+40)


fig, ax = plt.subplots(figsize=(20, 12))
ax.plot(df.index, df.values)
for p, f, c in zip((pred1, pred2, pred3),(fit1, fit2,␣
↪fit3),('#ff7823','#3c763d','c')):
    ax.plot(train.index, f.fittedvalues, color=c)
    ax.plot([pd.to_datetime(f'{year}-12-31T00:00:00.000000000') for year in␣
↪range(2019,2060)], p, label="alpha="+str(f.params['smoothing_level'])[:4]+",␣
↪beta="+str(f.params['smoothing_trend'])[:4], color=c)
plt.xlabel("Year",fontsize=15,fontweight=550,labelpad=15)
plt.ylabel("Population in 10 billions",fontsize=15,fontweight=550,labelpad=15)
plt.title("Holt's Exponential Smoothing predictions",fontsize=20,fontweight=600)
plt.legend(fontsize=18)
```

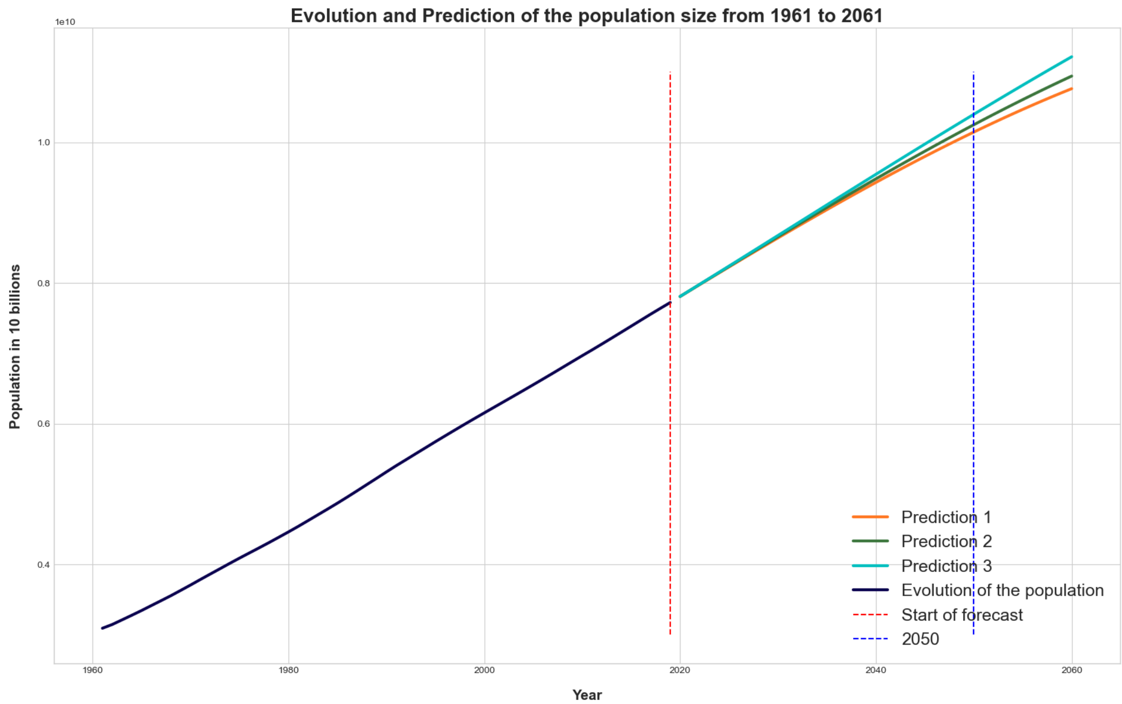
[40]: <matplotlib.legend.Legend at 0x1ffe243a4f0>



[41]:
```
PopIni = World["Population_2019"].values[0]
#orange one
Pred_pop_size1 = []
for percGrowth in pred1:
    PopIni = PopIni + PopIni * (percGrowth/100)
    Pred_pop_size1.append(PopIni)

#green one
PopIni = World["Population_2019"].values[0]
Pred_pop_size2 = []
for percGrowth in pred2:
    PopIni = PopIni + PopIni * (percGrowth/100)
    Pred_pop_size2.append(PopIni)

#blue one
PopIni = World["Population_2019"].values[0]
Pred_pop_size3 = []
for percGrowth in pred3:
    PopIni = PopIni + PopIni * (percGrowth/100)
    Pred_pop_size3.append(PopIni)
```

[42]:
```
#Set a figure
fig = plt.figure(figsize=[20,12])
plt.xlabel("Year",fontsize=15,fontweight=550,labelpad=15)
plt.ylabel("Population in 10 billions",fontsize=15,fontweight=550,labelpad=15)
plt.title("Evolution and Prediction of the population size from 1961 to␣
 ↪2061",fontsize=20,fontweight=600)

plt.plot([2020 + i for i in␣
 ↪range(41)],Pred_pop_size1,linewidth=3,label="Prediction 1",color = "#ff7823")
plt.plot([2020 + i for i in␣
 ↪range(41)],Pred_pop_size2,linewidth=3,label="Prediction 2",color = "#3c763d")
plt.plot([2020 + i for i in␣
 ↪range(41)],Pred_pop_size3,linewidth=3,label="Prediction 3 ",color = "c")
plt.plot([int(item[-4:]) for item in World.columns[61:].tolist()],World[World.
 ↪columns[61:]].values[0],linewidth=3,label="Evolution of the␣
 ↪population",color = "#0A014F")
plt.vlines(2019, 3000000000, 11000000000, linestyle='--', color='r',␣
 ↪label='Start of forecast');

plt.vlines(2050, 3000000000, 11000000000, linestyle='--', color='b',␣
 ↪label='2050');
plt.legend(fontsize=18,loc=4)
```

[42]: <matplotlib.legend.Legend at 0x1ffe24ef130>



With this method, we find also a result between 10 and 10.5 billions people by 2050 !!