1 数据读取

- PopulationData 文件夹中包含 251 个csv文件，分别存储了世界上 251 个国家和地区 1950~2021 年 72 年间的人口数据，同时包括了不同年龄段的人口数据
- 请你读取 251 个文件并将数据汇总到一个数据结构之中（推荐使用pandas库中的DataFrame结构）
- 请输出 1950 年和 2021 年的世界总人口

In [1]:
```
import os
import pandas as pd
datapath = './PopulationData'
dirs = os.listdir(datapath)           #dirs 得到 PopulationData 文件夹中所有文件名
# 使用列表存储
# data = []
# for file in dirs:
#     position = datapath + '/' + file   # 逐个读取文件
#     data.append(pd.read_csv(position))
# 使用数据表存储
# data = pd.DataFrame()
# for file in dirs:
#     position = datapath + '/' + file   # 逐个读取文件
#     data = pd.concat([data,pd.read_csv(position)])
# data.reset_index(inplace=True)  # 重新编制索引
# 使用字典存储
data = {}
for file in dirs:
    position = datapath + '/' + file   # 逐个读取文件
    temp = pd.read_csv(position,index_col=0)
    data[temp.loc[temp.index[0],"Country name"]] = temp
```

- 可以使用多种数据结构来存储数据

In [21]:
```
data["China"]
```

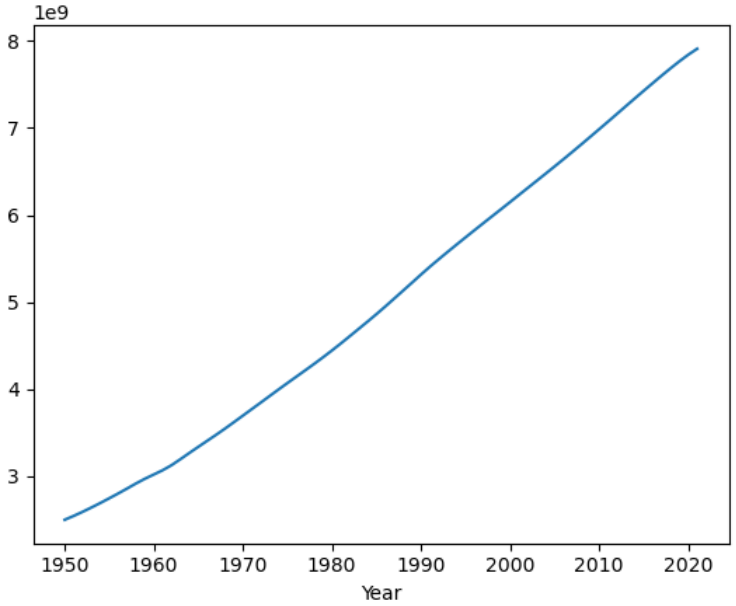
Out[21]:

Year	Country name	Population	Population of children under the age of 1	Population of children under the age of 5	Population of children under the age of 15	Population under the age of 25	Population aged 15 to 64 years	Population older than 15 years	Population older than 18 years	Population at age 1	...	Population aged 15 to 19 years	Population aged 20 to 29 years	Pop age 3
1950	China	543979200	19160980.0	77977320	189265000	287266180	327285950	354713730	322352130	16626031.0	...	52148756	87191640	70
1951	China	553614000	20258918.0	83163090	195505600	295216830	331027200	358107840	325480480	18034404.0	...	53035384	88490280	71
1952	China	564954500	21740164.0	88931460	203306750	304626080	334776500	361647170	329119900	19109946.0	...	53736870	89933530	72
1953	China	577378700	22683808.0	94657064	212066960	314522660	338512540	365311040	332975230	20542490.0	...	53831230	91808290	73
1954	China	589936000	22700958.0	99442360	221026320	324229860	342125920	368908930	337003840	21457060.0	...	53410508	93621200	74
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2017	China	1410276000	18226306.0	90595096	259556770	429759230	996345340	1150694500	1102487800	17703608.0	...	80517490	208634610	208
2018	China	1417069400	16853878.0	88891210	260606880	427163650	993782300	1156434800	1108832100	18200252.0	...	79819810	199880270	214
2019	China	1421864100	14975504.0	85553390	259653860	423412200	991272060	1162178800	1114996000	16830034.0	...	79582520	190966400	220
2020	China	1424929800	13362303.0	80999340	256845310	418674200	988568060	1168048500	1120545000	14956799.0	...	79383550	181492180	226
2021	China	1425893500	11501936.0	74789700	251927890	412677800	986464100	1173924600	1125760900	13346962.0	...	79508270	174028860	229

72 rows × 23 columns

In [22]:
```
data["World"].Population.plot()
```

Out[22]: <AxesSubplot:xlabel='Year'>



In [23]:
```
# data.keys()
p_1950_2021 = pd.DataFrame(columns=["p1950","p2021"])
for line in data.keys():
    p_1950_2021.loc[line,"p1950"] = data[line].loc[1950,"Population"]
    p_1950_2021.loc[line,"p2021"] = data[line].loc[2021,"Population"]
p_1950_2021.loc["Sum"] = [0,0]
for line in p_1950_2021.index[:-1]:
    p_1950_2021.loc["Sum","p1950"] += int(p_1950_2021.loc[line,"p1950"])
    p_1950_2021.loc["Sum","p2021"] += int(p_1950_2021.loc[line,"p2021"])
p_1950_2021
```

Out[23]:

	p1950	p2021
Afghanistan	7480464	40099460
Africa (UN)	227549260	1393676400
Albania	1252587	2854710
Algeria	9019866	44177964
American Samoa	19057	45056
...	...	...
World	2499322000	7909295000
Yemen	4712813	32981644
Zambia	2318453	19473132
Zimbabwe	2791336	15993525
Sum	15440875359	51918617925

252 rows × 2 columns

2 异常值处理

- 识别数据中的异常值并对异常值进行修复（提示：某国某年的总人口等于各个年龄段的人口之和）

> 异常数据：

> Afghanistan 2005 Population:24411196 -> 0

> Germany 2013 Population:81680590 -> None

> Somalia 2009 整体缺失

In [24]:

```
# 首先，了解数据的属性
for line in data["Germany"].columns:
    print(type(data["China"].loc[2021,line]),line,data["China"].loc[2021,line])
print(data["China"].index[0],type(data["China"].index[0]))
```

```
<class 'str'> Country name China
<class 'numpy.int64'> Population 1425893500
<class 'numpy.float64'> Population of children under the age of 1 11501936.0
<class 'numpy.int64'> Population of children under the age of 5 74789700
<class 'numpy.int64'> Population of children under the age of 15 251927890
<class 'numpy.int64'> Population under the age of 25 412677800
<class 'numpy.int64'> Population aged 15 to 64 years 986464100
<class 'numpy.int64'> Population older than 15 years 1173924600
<class 'numpy.int64'> Population older than 18 years 1125760900
<class 'numpy.float64'> Population at age 1 13346962.0
<class 'numpy.float64'> Population aged 1 to 4 years 63287764.0
<class 'numpy.int64'> Population aged 5 to 9 years 90508240
<class 'numpy.int64'> Population aged 10 to 14 years 86629944
<class 'numpy.int64'> Population aged 15 to 19 years 79508270
<class 'numpy.int64'> Population aged 20 to 29 years 174028860
<class 'numpy.int64'> Population aged 30 to 39 years 229977170
<class 'numpy.int64'> Population aged 40 to 49 years 204141890
<class 'numpy.int64'> Population aged 50 to 59 years 228613400
<class 'numpy.int64'> Population aged 60 to 69 years 144414740
<class 'numpy.int64'> Population aged 70 to 79 years 80093240
<class 'numpy.int64'> Population aged 80 to 89 years 29252108
<class 'numpy.int64'> Population aged 90 to 99 years 3894913
<class 'numpy.float64'> Population older than 100 years 40972.0
1950 <class 'numpy.int64'>
```

In [6]:

```
import numpy
# 格式错误（类型错误）
def formatError(country):
    for row in country.index:
        for col in country.columns[1:]:
            if type(country.loc[row,col])!=numpy.int64:
                try:
                    country.loc[row,col] = int(country.loc[row,col])
                except ValueError:
                    print(row,col,country.loc[row,col],type(country.loc[row,col]))
                    country.loc[row,col] = 0
# 缺失值
def defultError(country):
    # 1950~2021
    judge = [True]
    for line in range(1950,2022):
        if line not in country.index:
            judge[0] = False
            judge.append(line)
            print(line,country.iloc[0,0])
    return judge
# 修复缺失值
def repairDefult(country,judge):
    if not judge[0]:
        temp = [country.iloc[0,0]]
        index = judge[1]
        for line in country.columns[1:]:
            temp.append(int((country.loc[index-1,line] + country.loc[index+1,line])/2))
        country.loc[index] = temp
# 数值异常
def valueError(country):
    for row in country.index:
        for col in country.columns[1:-2]:
            if country.loc[row,col]<1:
                print(row," ",col," ",country.loc[row,col]," ",country.iloc[0,0])
                repairValue(country,row,col)
def repairValue(country,row,col):
    country.loc[row,col] = country.loc[row,country.columns[4]] + country.loc[row,country.columns[7]]
```

In [8]:

```
for line in data.keys():
    formatError(data[line])
    judge = defultError(data[line])
    repairDefult(data[line],judge)
    valueError(data[line])
```

In [5]:
```
data["Monaco"]
```

Out[5]:

Year	Country name	Population	Population of children under the age of 1	Population of children under the age of 5	Population of children under the age of 15	Population under the age of 25	Population aged 15 to 64 years	Population older than 15 years	Population older than 18 years	Population at age 1	...	Population aged 15 to 19 years	Population aged 20 to 29 years	Popul aged 39 y
1950	Monaco	19755	387	1006	2767	4850	13899	16988	16443	192.0	...	934	2499	
1951	Monaco	19926	397	1243	3021	5027	13765	16905	16373	415.0	...	912	2467	
1952	Monaco	19692	367	1672	3414	5199	13152	16278	15816	498.0	...	806	2258	
1953	Monaco	20043	296	1593	3395	5205	13391	16648	16171	306.0	...	823	2251	
1954	Monaco	20296	230	1331	3229	5076	13657	17067	16558	236.0	...	860	2251	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2017	Monaco	37069	339	1650	4513	7664	19508	32171	31217	343.0	...	1615	3017	
2018	Monaco	37050	334	1666	4558	7717	19386	32092	31165	337.0	...	1589	3060	
2019	Monaco	37059	334	1673	4613	7761	19243	32039	31144	332.0	...	1543	3107	
2020	Monaco	36943	327	1665	4660	7783	19037	31881	31015	333.0	...	1498	3147	
2021	Monaco	36713	322	1646	4700	7786	18805	31625	30779	326.0	...	1457	3173	

72 rows × 23 columns

3 数据挖掘

- 计算每个国家 72 年间的人口平均增长率，并列出增长率最高的10个国家，计算公式为：
  人口增长率 = （年末人口数 − 年初人口数）/年平均人口 × 1000‰

- 计算平均增长率与平均人口数之间的相关系数，计算公式为：

$$\rho = \frac{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2}}$$

- 用极差变换法对数据进行标准化

$$X_i^* = \frac{X_i - Min(X)}{Max(X) - Min(X)}$$

In [23]:
```
# 默认为年末人口
def growthRate(country):
    rate = pd.DataFrame(columns=["rate"])
    for line in country.index[1:]:
        rate.loc[line] =( country.Population[line] - country.Population[line-1]) / country.Population[line-1]
    rate.loc["agv"] = [0]
    for line in rate.index[:-1]:
        rate.loc["agv"] += rate.rate[line]
    rate.loc["agv"] = rate.loc["agv"] / (len(rate)-1)
#     rate.rate = rate.rate.apply(lambda x:format(x,".5%"))
    return rate.iloc[-1,-1]
def avgPopulation(country):
    avg = 0
    for line in country.index:
        avg += country.Population[line]
    avg = avg / len(country)
    return avg
```

Out[23]: 1035531644.5833334

In [24]:
```
rel = pd.DataFrame(columns=["rate","population"])
for line in data.keys():
    rel.loc[line,"rate"] = growthRate(data[line])
    rel.loc[line,"population"] = avgPopulation(data[line])
rel
```

Out[24]:

	rate	population
Afghanistan	0.024527	16640670.138889
Africa (UN)	0.025856	643444770.833333
Albania	0.011797	2632759.152778
Algeria	0.022654	23456439.236111
American Samoa	0.012373	39193.625
...	...	...
Western Sahara	0.055564	199341.736111
World	0.016364	5008492981.944445
Yemen	0.027808	14089068.277778
Zambia	0.030431	8013770.708333
Zimbabwe	0.024955	8680139.041667

251 rows × 2 columns

- 计算平均增长率与平均人口数之间的相关系数，计算公式为：

$$\rho = \frac{\sum_{i=1}^{n}\left(X_i-\bar{X}\right)\left(Y_i-\bar{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i-\bar{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i-\bar{Y}\right)^2}}$$

- 用极差变换法对数据进行标准化

$$X_i^* = \frac{X_i - Min(X)}{Max(X) - Min(X)}$$

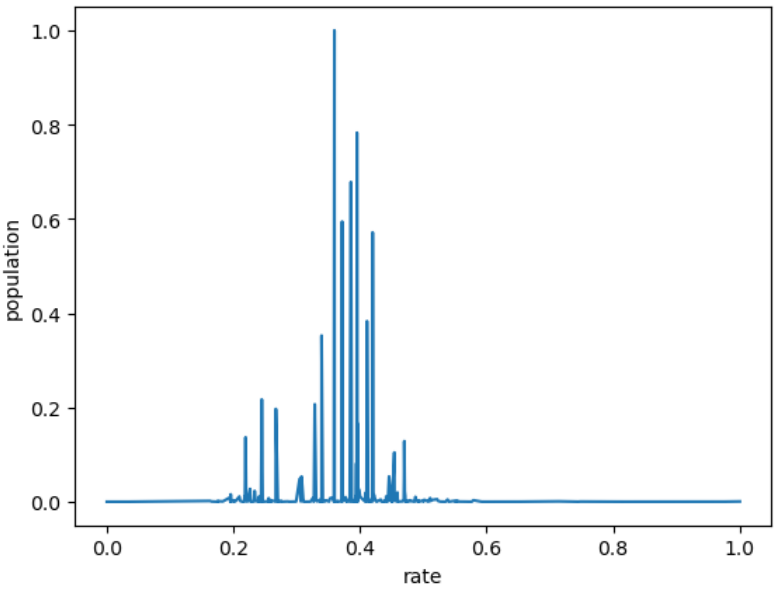
In [32]:
```
# 数据标准化
def Max_min(col):
    maxV = max(col)
    minV = min(col)
    for line in col.index:
        col[line]= (col[line]-minV) / (maxV-minV)
Max_min(rel.population)
Max_min(rel.rate)
```

In [37]:
```
def pearson(rel):
    up = 0
    downX = 0
    downY = 0
    meanX = sum(rel.rate) / len(rel)
    meanY = sum(rel.population) / len(rel)
    for line in rel.index:
        up += (rel.rate[line]-meanX)*(rel.population[line]-meanY)
        downX += (rel.rate[line]-meanX)**2
        downY += (rel.population[line]-meanY)**2
    rho = up / (downX**0.5 * downY**0.5)
    return rho
pearson(rel)
```

Out[37]: -0.010442013647858222

In [34]:
```
import seaborn as sns
sns.lineplot(x="rate",y="population",data=rel)
```

Out[34]: <AxesSubplot:xlabel='rate', ylabel='population'>



4 数据预测

- 根据已有数据，对 China 未来三年（2022~2024）的总人口数进行预测（提示：使用移动平均法）

$$F(t)=\frac{F(t-1)+\ldots+F(t-N)}{N}$$

In [62]:
```
china = pd.DataFrame(data["China"]["Population"])
china
```

Out[62]:

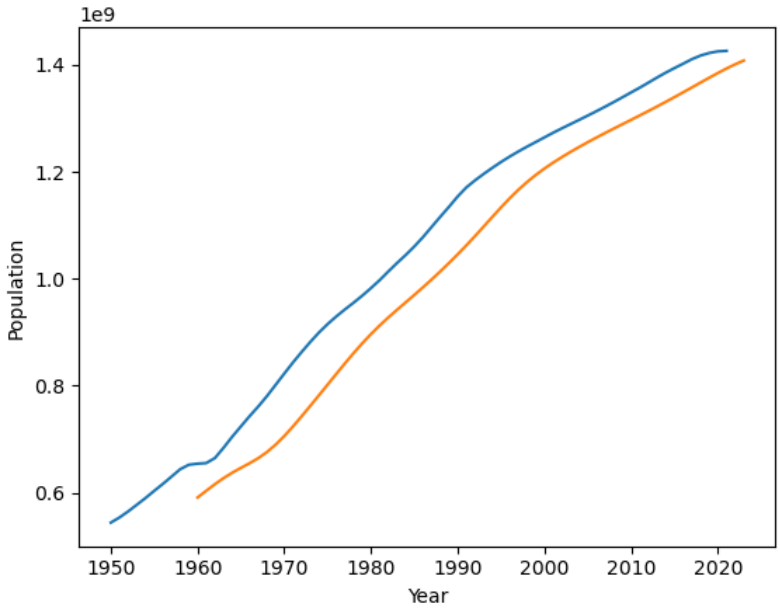
Year	Population
1950	543979200
1951	553614000
1952	564954500
1953	577378700
1954	589936000
...	...
2017	1410276000
2018	1417069400
2019	1421864100
2020	1424929800
2021	1425893500

72 rows × 1 columns

In [64]:
```
def move(china,n):
    temp = pd.DataFrame(columns=["Population"])
    for line in china.index[n-1:]:
        temp.loc[line+1] = 0
        for line2 in range(n):
            temp.loc[line+1] += china.Population[line-line2]
        temp.loc[line+1] = temp.loc[line+1] / n
    return temp
```

In [70]:
```
sns.lineplot(data=china,y="Population",x=china.index)
new_china = move(china,5)
new_china = move(new_china,5)
sns.lineplot(data=new_china,y="Population",x=new_china.index)
```

Out[70]: <AxesSubplot:xlabel='Year', ylabel='Population'>



5 数据可视化

In [ ]:
```
- 绘制 China 2021 年各个年龄段人口的频数分布直方图
```

In [76]:
```
data["China"].loc[2021]
```

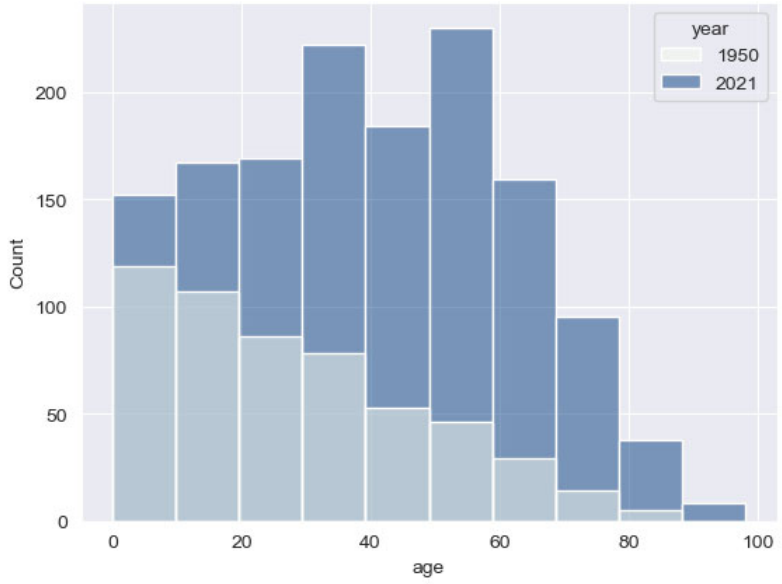
Out[76]:
```
Country name                                  China
Population                                1425893500
Population of children under the age of 1     11501936.0
Population of children under the age of 5       74789700
Population of children under the age of 15     251927890
Population under the age of 25                 412677800
Population aged 15 to 64 years                 986464100
Population older than 15 years                1173924600
Population older than 18 years                1125760900
Population at age 1                            13346962.0
Population aged 1 to 4 years                   63287764.0
Population aged 5 to 9 years                    90508240
Population aged 10 to 14 years                  86629944
Population aged 15 to 19 years                  79508270
Population aged 20 to 29 years                 174028860
Population aged 30 to 39 years                 229977170
Population aged 40 to 49 years                 204141890
Population aged 50 to 59 years                 228613400
Population aged 60 to 69 years                 144414740
Population aged 70 to 79 years                  80093240
Population aged 80 to 89 years                  29252108
Population aged 90 to 99 years                   3894913
Population older than 100 years                  40972.0
Name: 2021, dtype: object
```

In [118]:
```
from random import randint
def pinc(country,year):
    pin = {}
    n = year-1950
    pin[10] = int(sum(country.iloc[n,9:12]))
    pin[20] = int(sum(country.iloc[n,12:14]))
    for line in range(8):
        pin[30+line*10] = int(country.iloc[n,14+line])
    pin[100] += int(country.iloc[n,-1])
    temp = pd.DataFrame(columns=["age","year"])
    n = 0
    for line in pin.keys():
        for num in range(int(pin[line]/1000000)):
            temp.loc[n]=[line-randint(0,10),year]
            n += 1
    return temp
```

In [135]:
```
d = pinc(data["China"],2021)
d = pd.concat([d,pinc(data["China"],1950)])
d.reset_index(inplace=True)  # 重新编制索引
```

In [140]:
```
sns.set_style("darkgrid")
sns.histplot(data=d,x="age",
             bins=10,
             hue="year",
             palette='GnBu',  #'Blues' , "winter" ,'GnBu'
#              multiple="fill",  #"stack","fill","layer"
#             kde=True,
#             fill=False,
#             element = 'step',  #bars  step  poly
#              cumulative=True,
            )
```

Out[140]: <AxesSubplot:xlabel='age', ylabel='Count'>



In [ ]: